リポジトリ設定マニュアル

株式会社ジャスミンソフト

2020年4月30日

目次

モデル

モデル

- モデルの操作 51
- ビルド対象の指定 54
- モデルとテーブルの関係 54
- モデルIDの命名規則 54
- モデルの変更と削除 55
- システムモデルとユーザモデル 55
- 定義できるモデル数の上限 55
- リポジトリフォルダとの関係 55

モデル項目

- モデル項目の操作 55
- 項目名の命名規則 59
- 項目IDの命名規則 59
- 項目の変更と削除 59
- 定義できるモデル項目数の上限 60

型

- Wagbyで利用できる型 60
- データベースの型との関係 61
- 浮動小数について 61
- 型の変更に関する注意点 61

主キーの指定

- 主キーの指定方法 62
- 主キーアイコン 63
- 単一キー 63
- 複合キー 64
- 順序 64
- 仕様・制約 65

検索条件の指定

- 定義方法 65
- 範囲検索 66
- インデックス 66
- 表示順を制御する 66
- 仕様・制約 66

一覧表示項目の指定

- 定義方法 66
- 表示順を制御する 67
- ラベルの幅を設定する 67

ソート

- 例 67
- 定義方法 68
- 仕様・制約 69

データベースの詳細

- 概要 70
- テーブルを作成する 70
- 物理テーブル名 71
- キャッシュを有効にする 71
- 識別子の区切り文字 72
- ロック方式 72
- 既存テーブルを再利用する場合の注意点 72

ロック方式

- 楽観ロックと悲観ロック 73
- ロック方式を選択する 73

どのロック方式を選択するべきか. 74
内部での実装方法. 74
(悲観)ロックテーブル. 75
ロックキーのカスタマイズ. 75
サブデータベースを指定する. . . . . .
メインデータベースとサブデータベース. 75
定義方法. 76
サブデータベースの設定を無効にする. 77
スクリプトを使ってサブデータベースで順序を利用する. 78
データベースのビューを利用する. . . . . .
ビューの利用. 78
Wagbyのグループ権限をビューに適用する. 78
テーブル操作に関する注意点まとめ. . . . . .
この説明が必要な場合. 79
ロック. 79
キャッシュ. 79
モデル項目. . . . . .
文字列. . . . . .
文字列型の種類. 80
入力フィルタ. 80
データベースとの関係. 81
数値. . . . . .
数値型の種類. 81
入力項目の種類. 81
入力フィルタ. 81
入力ルール. 82
出力フォーマット. 83
小数点扱い時の注意. 86
初期値 0 を表示しない. 87
検索条件としての利用. 87
入力欄を文字列として扱う. 87
データベースとの関係. 88
日付. . . . . .
日付型. 88
新元号への対応. 89
仕様・制約. 89
[応用] カレンダ利用時の入力チェックルールの変更 (8.3.0). 89
入力フィルタ. 90
カレンダの利用. 90
リストボックスの利用. 92
オートフォーマットの利用. 96
日付フォーマットの指定. 99
オプション表示. 100
システム時刻(1970年1月1日)を表示しない. 101
検索条件としての利用. 101
和暦の元年表記. 102
時刻. . . . . .
時刻型. 102
入力フィルタ. 102
リストボックスの利用. 102
リストボックスの表示ルール. 104
テキストフィールドの利用. 105
日付時刻. . . . . .
日付時刻型. 106
入力フィルタ. 107

ユーザーインタフェース (8.1.0) 107
テキストフィールドの利用 108
仕様・制約 109
文字列 - 郵便番号
郵便番号から住所を求める 109
郵便番号の候補を出力する 111
ハイフンの扱い 113
入力フィルタの手動設定 113
郵便番号辞書 113
仕様・制約 114
文字列 - 住所正規化コンバータとの連携
住所正規化コンバータとは 114
例 114
定義方法 115
住所正規化コンバータサーバの設定 116
文字列 - メールアドレス
例と定義方法 116
文字列 - URL
例と定義方法 117
リンク先画像を表示する 118
リンク先画像の幅と高さを指定する 120
画像表示時にURLを表示しない 121
リンクを無効にする 122
応用 GoogleMapsとの連携 123
文字列 - ファイル
例と定義方法 128
アップロードできるファイルに制約を設ける 130
アップロードの保存先フォルダを変更する 132
ダウンロード時のログ 132
テキストファイルの文字エンコーディング指定 132
ファイルダウンロードのセキュリティ 133
検索 133
繰り返しコンテナ内のファイル項目 133
仕様 135
保存ファイル名の扱い 136
ファイルの削除ポリシー 136
画像の幅、高さを指定する 136
ファイル名を非表示にする 138
ファイルへのリンクを無効にする 139
画像ファイルをダイアログで表示する (8.1.2) 140
文字列 - テキストエリア
テキストエリア 142
行数と文字数を指定する 143
テキストエリア(リッチ) 143
テキストエリアをラベルで表示する 148
入力可能文字数を表示する (8.1.0) 149
一覧更新画面の場合 151
データベースとの関係 152
ブラウザとの関係 152
内部で利用する改行コード 152
文字列 - 追記型リストボックス
例と定義方法 153
検索条件で利用する場合 154
仕様・制約 154
固定値
固定値とは 155

例 155
定義方法（項目テンプレートを使う） 156
データベースの詳細
概要 160
値をデータベースに保存する 160
インデックスを利用する 161
テーブル定義の型 161
参照先項目のテーブル定義の型と同じ設定を適用する 161
物理カラム名 162
一意制約 162
ロック名 163
トランザクション制御 163
データベースの詳細 - SQL式(SQL)
概要 163
[例1] モデル参照項目をデータベースに保存しないが、対応する参照連動項目を検索したい 164
[例2] 計算式（自モデル） 165
[例3] 計算式（子モデル） 166
[例4] 保存前の値を保持する項目を用意する 167
[例5] NULL 値のソートルールを変える 168
デバッグ方法 168
仕様・制約 168
項目ルールを定義する（データ辞書としての利用）
データ辞書 169
項目ルールを定義する 169
項目ルールを使う 174
項目ルールの適用状況を確認する 174
項目ルールの内容を変更する 175
名前と型の扱い 175
仕様・制約 177
関連する機能 177
入力フィルタの仕様
入力フィルタの選択 177
検索時の入力フィルタ (8.4.0) 179
詳細な仕様 179
標準のフィルタルールを変更する 180
他モデルの参照
選択肢モデル
選択肢モデル 181
テーブルを作成しない選択肢モデル 181
入力フィルタ 182
選択肢モデルの作成 182
選択肢モデルの参照方法 185
運用中に選択肢を追加する 189
選択肢の表示順を変更する 190
選択肢を無効にする 191
無効な選択肢を示すマークを変更する 194
無効な項目も検索対象とする 194
参照先モデルのデータが削除されていた場合 196
データベースとの関係 196
リストボックス
例と定義方法 196
参照先モデルのデータが削除されていた場合 198
データベースとの関係 198
複合キーのモデルを参照する 199
関連するページ 200
初期値 200

選択肢の表示順を変更する. 201
選択肢を無効にする. 204
無効な項目も検索対象とする. 205
未選択項目を用意する. 205
全件を検索する. 207
検索画面での切り替え. 208
複数の項目を連結して表示する. 209
必須を指定する. 211
ラジオボタン. . . . . .
例と定義方法. 211
データベースとの関係. 213
複合キーのモデルを参照する. 213
初期値. 213
選択肢の表示順を変更する. 215
選択肢を無効にする. 215
無効な項目も検索対象とする. 215
折り返し数の設定. 215
選択肢のクリアボタン. 217
検索画面での切り替え. 217
選択されなかった項目名を表示する. 218
参照先モデルのデータが削除されていた場合. 219
チェックボックス. . . . . .
例と定義方法. 219
最大N個までという入力ルール. 221
検索画面での切り替え. 222
参照先モデルのデータが削除されていた場合. 222
データベースとの関係. 222
複合キーのモデルを参照する. 222
仕様・制約. 222
選択肢の全チェックと全クリア. 222
初期値. 223
選択肢の表示順を変更する. 224
選択肢を無効にする. 225
無効な項目も検索対象とする. 225
折り返し数の設定. 225
「チェック画像」の表示切り替え. 226
一覧表示画面への適用. 227
選択されなかった項目名を表示する. 228
検索画面. . . . . .
例と定義方法. 230
参照先モデルのデータが削除されていた場合. 232
複合キーのモデルを参照する. 232
仕様・制約. 233
入力フィールド長. 233
選択情報の全クリア. 233
検索ボタンの名称変更. 235
検索ウィンドウの幅と高さ. 235
表示される項目のルール. 235
主キー（コード）を直接入力する. 237
メインウィンドウの入力値を検索条件として再利用する. 240
サジェスト. 243
リセットボタンを非表示にする. 243
参照連動. . . . . .
例と定義方法. 244
仕様. 247
参照連動項目を自モデルに保存する. 249
参照連動項目を自モデルに保存する(上書き). 251

参照連動項目を参照先モデルに保存する. 256
参照連動項目の入力フィルタ. 259
参照連動項目を表示のみの検索条件とする. 259
参照連動された値を追記する. 260
チェックボックス（サブウィンドウ）. . . . . .
例. 261
定義方法. 263
仕様・制約. 265
チェックボックス（有無フラグ）. . . . . .
背景. 265
選択肢モデルを参照する. 265
ストアモデルを参照する. 269
変更履歴と併用する場合の注意点 (8.3.3). 272
参照元のトレース. . . . . .
例と定義方法. 272
権限管理との関係. 274
アカウントとの関係. 274
参照連動との関係. 274
仕様・制約. 274
参照先のトレース. . . . . .
例と定義方法. 274
仕様・制約. 276
参照モデル同士の関係性で選択肢を絞り込む. . . . . .
リストボックスで絞り込み、リストボックスで選択する. 276
ラジオボタンで絞り込み、リストボックスで選択する. 281
チェックボックスで絞り込み、リストボックスで選択する. 282
リストボックスで絞り込み、チェックボックスで選択する. 283
ラジオボタンで絞り込み、チェックボックスで選択する. 283
チェックボックスで絞り込み、チェックボックスで選択する. 284
仕様・制約. 284
パフォーマンス. 285
トラブルシューティング. 285
他項目の値で選択肢を絞り込む. . . . . .
例と定義方法. 285
無効となった選択肢の選択状態を維持しない. 290
複数の条件で絞込む. 291
繰り返しコンテナ内の項目で絞り込む. 293
固定値に関数を利用する (8.3.0). 297
IS NULL/IS NOT NULLで絞り込む (8.3.2). 299
仕様・制約. 299
トラブルシューティング. 300
関連するページ. 300
スクリプトを使った選択肢の絞り込み(スクリプト). . . . . .
例. 300
定義方法. 302
サジェスト. . . . . .
例. 304
定義方法. 305
仕様・制約. 307
[例] 固定メッセージの選択・追加による入力方式の実現. . . . . .
例. 308
定義方法. 310
仕様・制約. 314
[例] 選択肢の絞り込みを用いて、操作時点での消費税を定める. . . . . .
例. 315

定義方法 315
関係性のないリストボックスを計算によって連動させる
例 320
定義方法 321
繰り返し
繰り返し項目
例 325
定義方法 327
ルール 327
データベースとの関係 327
繰り返し項目 － 初期表示数
例 327
定義方法 328
繰り返し項目 － 繰り返し項目の上限数
例 328
定義方法 329
繰り返し項目 － 区切り文字
例 329
定義方法 331
応用 － 重複を除く 331
繰り返し項目 － 数量の表示
例 333
定義方法 334
繰り返しコンテナ
例 334
定義方法 336
列幅の設定 337
繰り返しコンテナID 337
データベースとの関係 337
繰り返しコンテナ － 初期表示数
例 338
定義方法 338
初期表示数を0にする 338
初期表示数を式で設定する 339
仕様・制約 340
繰り返しコンテナ － 繰り返し項目の上限数
例 340
定義方法 341
仕様・制約 341
繰り返しコンテナ － 追加時に値をコピーする
例 341
定義方法 342
繰り返しコンテナ － ソート
定義方法 343
実行時にソートルールを決める 344
ソートルールを固定にする 344
追加時の動作の詳細 345
繰り返しコンテナ － 並びを入れ替える
例 345
定義方法 346
式による制御 347
繰り返しコンテナ － 操作ボタンの表示制御
ボタン名を変更する 347
ボタンを非表示にする 348

式による制御 348
繰り返しコンテナ － コンテナIDを非表示にする
例 348
定義方法 349
複数段一行明細利用時の注意点 349
仕様・制約 349
繰り返しコンテナ － 更新のタイミングで別の繰り返しコンテナの同期増減を行う
例 350
定義方法 351
繰り返しコンテナのコピー入力 352
繰り返しコンテナ － 折りたたみ
例 354
定義方法 355
繰り返しコンテナの参照連動
例 355
定義方法 357
参照連動を自モデル保存する 359
条件によって参照連動するデータを絞り込む 361
トラブルシューティング 363
繰り返しコンテナでデータの変更履歴を管理する
変更履歴を有効にする 363
モデルの詳細画面に変更履歴を表示する(1) － 先頭に追加される 364
モデルの詳細画面に変更履歴を表示する(2) － 末尾に追加される 369
テンプレートモデルとして用意する 370
仕様・制約 373
繰り返しコンテナでデータの既読状態を管理する(スクリプト)
例 373
定義方法 377
URLを直接入力して閲覧する場合 381
仕様・制約 382
例 既読状態をチェックボックスで管理する 382
繰り返しコンテナ － 独立したモデルとして扱う
例 383
定義方法 385
スクリプトで利用する 388
トラブルシューティング 388
親子関係
親子関係を定義する
例 388
定義方法 390
子モデル作成時の親の主キーの扱い 391
グリッド表示 393
外部キーアイコン 394
子モデルの表示順序を制御する 395
繰り返しコンテナと外部キーの違い 395
参照連動との組み合わせ 395
トラブルシューティング 397
一つの親に複数の子を紐づける
例 397
定義方法 398
子モデル表示レイアウトの変更（タブ表示と一覧表示） 400
子モデル表示の並び 401
親モデルが存在しないと子モデルの作成が行えないようにする
例 403
定義方法 405

仕様・制約 406
親モデルが存在しない子モデルの作成を許可する
例 406
定義方法 409
親の任意の項目で選択できるようにする 410
参照連動との組み合わせ 411
仕様・制約 412
一つの子に複数の親を紐づける
モデルの説明 412
例 413
定義方法 418
親キー以外の項目値を用いて子モデル一覧表示を行う(スクリプト)
例 421
定義方法 424
仕様・制約 426
連携に関する設定
子モデル表示のレイアウト 426
一覧のタイトルをリンクにする 426
子モデル一覧名を変更する 429
一覧データの表示件数を制限する 429
説明文を表示する 430
ボタンの表示を制御する 431
親モデルの画面に子モデルのデーター覧を表示しない 433
削除レベルの指定 433
再計算の指定（親） 435
再計算の指定（子） 436
子モデルの同時更新
例と定義方法 437
初期表示数 439
子モデル一覧表示から直接、子モデルの更新画面へ遷移する 439
1つの親に複数の子を紐付ける 440
コピー登録 441
スクリプトを使って子モデル毎に異なる初期値を設定する 442
仕様・制約 443
入力チェック
必須・警告
必須入力 443
警告を出すが、入力を許可する 444
初期値について 445
データベーススキーマへの影響 445
主キーとの関係 445
仕様・制約 446
関連するページ 446
禁止文字列
例 446
定義方法 446
複数の禁止文字列を定義する 447
警告の設定 447
トラブルシューティング 447
仕様・制約 448
文字数チェック
例 448
定義方法 － 数値型項目の場合 448
定義方法 － それ以外 448
文字列項目とデータベースの関係 449

仕様・制約 449
同一項目値チェック
例 450
定義方法 450
警告の設定 451
仕様・制約 451
範囲チェック
数値の範囲 451
数値の範囲を定義する 451
日付型の範囲 452
日付の範囲を定義する 452
警告の設定 453
仕様・制約 453
正規表現チェック
例 453
定義方法 453
警告の設定 454
仕様・制約 454
利用できる構文 454
例 455
他項目比較チェック
例 456
定義方法 456
警告の設定 457
仕様・制約 459
ユーザ定義チェック
例 459
定義方法 459
警告の設定 460
例 460
入力チェックスクリプトの利用 461
国際化対応 461
仕様・制約 461
許容文字種
例 462
定義方法 463
自動設定される型 463
設定の詳細 463
警告の設定 465
必要最小文字数の指定 466
フィルタとの関係 466
任意の文字を指定する (8.1.0) 467
仕様・制約 468
入力制御
最大入力文字数
例 469
定義方法 469
仕様・制約 469
入力フィールド長
方法1 入力フィールド長を指定する 469
方法2 スタイルを指定する 470
テキストエリア 472
隠し項目（登録・更新）
定義方法 472
仕様 472

読み込み専用項目
定義方法 473
仕様 473
日付・数値の初期値は表示しない
例 473
定義方法 475
繰り返し項目への自動設定 475
Enterキーの取り扱い
定義方法 475
利用例 476
制約 476
IME制御
定義方法 476
対応するWebブラウザ 477
仕様・制約 477
タブインデックス
定義方法 477
繰り返しコンテナの場合 479
仕様・制約 480
別項目の値をコピーする
例 480
定義方法 481
入力済みの場合はコピーしない 482
検索条件 - 範囲検索 482
仕様・制約 483
別項目をグループとして扱い、値をまとめてコピーする
例 483
定義方法 484
クリアボタンを用意する 486
ボタンの位置を指定する 487
オリジナルボタンの処理を記述する 492
仕様・制約 492
項目のオリジナルボタン押下で JavaScript を動作させる(JavaScript)
例 493
他項目の値による入力制御
入力可制御 494
入力不可制御 497
繰り返しコンテナ (8.3.0) 497
仕様・制約 498
入力欄のスタイル指定
例 499
定義方法 500
例 リストボックスの幅を指定する 501
例 ラジオボタン・チェックボックスの高さを指定する 501
式を適用する 504
入力欄の前後に説明文を付与する
例 504
定義方法 505
テキストエリアでの利用 506
仕様・制約 506
入力欄の表示位置の制御
横位置の設定 507
縦位置の設定 507
カーソル位置の設定 508

手書き入力
実行例 509
定義方法 511
JasperReportに署名画像を埋め込む 512
仕様・制約 513
電話番号入力時、自動的にハイフン区切りを行う
実行例 513
定義方法 513
仕様・制約 514
振り仮名の自動入力
実行例 514
定義方法 515
仕様・制約 515
入力フォーカス
実行例 515
フォーカスのカスタマイズ 516
バーコード読み取り
実行例 516
定義方法 518
仕様・制約 519
関連するページ 519
[事例] 百万円単位での入力欄を用意する
例 519
定義方法 520
計算式
基本的な書き方
定義方法 523
三項演算子 523
三項演算子を徐算で用いる 523
初期値との関係 524
繰り返し項目との関係 524
範囲検索項目との関係 524
親子モデルとの関係 524
モデル参照項目の扱い 524
スクリプトを利用する 524
四則演算式 525
関数 525
モデル項目の指定 525
条件判定式 525
式の入力ダイアログ 526
値をデータベースに保存する 531
精度 532
空文字とnull 533
未入力項目の扱い 533
モデル参照
モデル参照項目の指定方法 533
参照連動項目 534
関数と組み合わせたときの型情報 535
主キーが複合キーのモデルを参照する項目 535
ジョブにおける式記述の制約 535
計算式の例
空白・改行の除去 536
条件によって算出値を変える 536
計画日と実績日の差異 536
文字列の連結 537

条件付きSUM関数 537
数値の按分 537
入力前の値との差 538
数値項目にnullをセットする 538
加入日からの経過月数 538
生年月日から年齢 538
所要時間 (1) 539
所要時間 (2) 539
所要時間 (3) 539
所要時間 (4) 539
日付から曜日 540
日付から年、月、日を切り出す 540
ゼロ詰め文字の作成 540
計算式かつ編集可能な項目を用意する
例 541
定義方法 541
JavaScriptを利用する 544
関数
数値関数
COSH 544
DEGREES 545
DIV 545
EVEN 545
EXP 546
FACT 546
FIXED 546
FLOOR 546
HYPOT 547
IEEEremainder 547
ABS 547
INT 547
LN 547
LOG 548
LOG1P 548
LOG10 548
MOD 548
MROUND 549
MUL 549
ODD 549
PI 550
ACOS 550
POWER 550
QUOTIENT 550
RADIANS 551
RAND 551
RANDBETWEEN 551
RINT 551
ROUND 551
ROUNDDOWN 552
ROUNDUP 552
SIN 552
ADD 553
SINH 553
SIGN 553
SIGNUM 553
SQRT 554
SUB 554

TAN. 554
TANH. 555
TRUNC. 555
ATAN. 555
ATAN2. 555
ASIN. 555
CBRT. 556
CEILING. 556
COS. 556
文字列関数. . . . . .
A1FORMAT. 557
ISBLANK. 557
ISINCLUDE. 557
JIS. 558
JOIN. 558
JPKANANORM. 558
LEFT. 558
LEN. 558
LENB (8.2.0). 559
LOWER. 559
LPAD. 559
ACCOUNTKANANORM. 559
LTRIM. 560
MAILADDRESS. 560
MATCHRE. 560
MATCHRE1. 561
MID. 561
NUMTOJTEXT. 562
PADDING. 562
PATTERN. 562
PROPER. 563
RANDSTR. 563
ANORMPART. 563
REMOVEHTMLTAG. 564
REPLACE. 564
REPT. 565
RIGHT. 565
RPAD. 565
RTRIM. 565
SLICE. 565
SPLIT. 566
SUBSTITUTE. 566
STR2BYTEA (8.3.1). 566
ASC. 567
TEXT. 567
TOBYTE. 568
TOBYTE_ARRAY. 568
TOSHORT. 568
TOSHORT_ARRAY. 568
TOINT. 568
TOINT_ARRAY. 568
TOLONG. 569
TOLONG_ARRAY. 569
TOFLOAT. 569
CONCATENATE. 569
TOFLOAT_ARRAY. 569
TODOUBLE. 569
TODOUBLE_ARRAY. 569

TOSTR. 570
TOSTR_ARRAY. 570
TRIM. 570
UPPER. 570
VALUE. 570
VALUE_ARRAY. 571
COUNT. 571
EXACT. 571
FIND. 572
FINDTAIL. 572
FINDRE. 572
日付時刻関数. . . . . .
EDATE. 573
EOMONTH. 573
FISCALYEAR. 573
HOUR. 573
MINUTE. 573
MONTH. 574
MOVEDAY. 574
MOVEHOUR. 574
MOVEMINUTE. 574
MOVEMONTH. 574
BETWEENTIME. 574
MOVESECOND. 575
MOVEYEAR. 575
NETWORKDAYS. 575
NOW. 575
SECOND. 576
TIME. 576
TIMEVALUE. 576
TODATE. 576
TODAY. 576
TOTIME. 577
BOMONTH. 577
WEEKDAY. 577
WEEKNUM. 577
WORKDAY. 578
YEAR. 578
CLEARHMS. 578
DATE. 578
DATEDIF. 578
DATESTRING. 579
DATETIMEVALUE. 579
DATEVALUE. 579
DAY. 579
集合処理. . . . . .
DJOIN. 579
DDJOIN. 580
RANK. 580
LARGE. 581
SMALL. 581
GCD. 581
LCM. 582
ARRAY. 582
ARRAY_OBJ (8.3.0). 582
SORT. 583
COUNT. 583
REVERSE. 584

CONTAINS. 584
ISEMPTY. 584
ADD. 584
GET. 584
INDEXOF. 585
LASTINDEXOF. 585
ISEQUALS. 585
ISSAMELENGTH. 585
REMOVE. 585
SUM. 585
REMOVEELEMENT. 586
REMOVEELEMENTALL. 586
UBOUND. 586
AGCOUNT. 587
AGSUM. 587
AGAVG. 587
AGMAX. 587
AGMIN. 587
モデル参照項目の扱い. 588
外部キー子モデルの指定方法. 588
AVERAGE. 589
制約. 589
MAX. 589
MIN. 590
FIRST. 590
LAST. 591
SLICE. 591
JOIN. 592
統計処理. . . . .
MEDIAN. 592
MODE. 593
STDEV. 593
STDEVP. 593
VAR. 594
VARP. 594
条件分岐. . . . .
AND. 594
IF. 595
N. 595
NOT. 595
OR. 596
TRUE. 596
FALSE. 596
選択. . . . .
CHOOSE. 596
CID. 597
権限. . . . .
HASMODELPERMISSION. 597
HASPRINCIPAL. 597
アカウント. . . . .
USERNAME. 598
USERINFO. 598
JGROUPID. 599
画面制御. . . . .
BROWSERTYPE. 599
EVENT. 600
RESULTSIZE. 600

SCREENID. 600
SCREENTYPE. 600
WIZARDPAGE. 601
リソース・国際化. . . . . .
LOCALE. 601
MESSAGE. 601
ERRORMESSAGE. 601
モデル参照. . . . . .
ISEMPTY. 602
SELECTED. 602
CHECKED. 603
EXACT. 603
CONTAINS. 603
ISNULL. 603
制約. 604
財務. . . . . .
FV. 604
NPV. 604
PMT. 604
PV. 605
ワークフロー. . . . . .
WF_NEXT_GROUPID_MAIL. 605
WF_NEXT_MAIL. 606
WF_GROUPID. 606
WF_IS_ADMITUSER. 606
WF_APPLICANT. 606
WF_APPLICANT_MAIL. 607
WF_LASTPROCESSDATE. 607
WF_LASTPROCESS_USERID. 607
WF_LASTPROCESS_USERNAME. 607
FLOWSTATUS. 608
WF_EVENT. 608
WF_LASTEVENT. 608
WF_CURRENT_USERID. 609
WF_NEXT_USERID. 609
WF_NEXT_USERID_MAIL. 609
WF_USERID. 610
WF_CURRENT_GROUPID. 610
WF_NEXT_GROUPID. 610
外部連携. . . . . .
ISREST. 610
HTTPSEND. 611
JSONPATH. 611
REQUEST. 612
SESSION. 612
ATTRIBUTE. 613
XML2JSON (8.3.1). 613
その他. . . . . .
BARCODE (8.3.0). 614
CLIENTHOSTNAME (8.3.0). 614
CHANGELOG. 614
EXEC. 614
SERVERHOSTNAME (8.3.0). 615
Me. 615
初期値. . . . . .
基本的な書き方. . . . . .

定義方法 615
初期値が設定されるタイミング 617
式にモデル参照項目を含める場合 617
別画面または別システムとの連携 618
スクリプトを利用する 618
文字列型で指定できるオプション
関数利用時の注意点 618
URLの初期値 618
応用例：ランダム文字列をセットする 618
応用例：ログオンIDをセットする 618
日付・時刻・日付時刻型で指定できる特殊表記 now
定義方法と例 618
関数との組み合わせ 620
型とnowの関係 620
チェックボックス
すべてを選択する 620
「所属グループ」の利用 621
関数を適用する 622
REQUEST関数利用時の注意点 622
計算式によって初期値を求める
ラジオボタンへの適用例（三項演算子との組み合わせ） 623
チェックボックスへの適用例 623
主キーが複合キーのモデルを参照する項目 623
版の管理
例 624
定義方法 625
主キーオプション
用意されたパターン
0 詰めの順序 626
接頭語+区切文字+順序 628
日付+区切文字+順序 629
任意の書式を設定する（ユーザ定義パターン）
ステータスコード + 年（西暦下二桁）+ 月（ゼロ詰め）+ 連番（ゼロ詰め四桁） 630
固定コード + 年度（西暦四桁）+ 連番（ゼロ詰め六桁） 632
複合キーによる枝番
例1 主キーの手動入力 633
例2 コピー登録時に枝番を付与する 636
枝番をアルファベット表記とする 638
複合キーによる制約を回避したい場合 641
単一キーで絞込機能を応用した枝番
この説明を読む前に 641
仕様と例 641
定義方法 643
枝番をゼロ詰め3桁で表現する 647
枝番をアルファベット表記かつハイフン区切りなしとする 647
検索制御
文字列のAND/OR検索
例 649
定義方法 650
仕様・制約 650
標準を OR 検索にする 650
文字列の一致ルールを指定する（ワイルドカードの利用）
定義方法 651
ワイルドカード 651

検索条件を必須入力にする
例 651
定義方法 652
仕様・制約 653
無効判定項目を無視する
例 653
定義方法 655
範囲検索を行う
例 656
定義方法 657
日付型項目の開始および終了の初期値の設定 657
月日の自動補完 658
日付型項目の開始のみを設定する 660
テキストフィールドを用いた年、年度、年月日の検索 661
仕様・制約 664
未入力値の検索
例 664
定義方法 665
検索値の初期値を未入力値とする 666
仕様・制約 667
自己参照による検索項目を指定する
例 667
定義方法 667
仕様・制約 668
隠し項目・読み込み専用項目にする
隠し項目にする 668
定義方法 669
読み込み専用項目にする 671
定義方法 672
参照連動項目を検索条件に指定する
例 673
定義方法 675
仕様・制約 678
暗黙条件
例 678
定義方法 680
暗黙条件の仕組み 683
関数を利用する 683
OR検索 684
仕様・制約 684
トラブルシューティング 684
全文検索
全文検索とは 684
例と定義方法 684
添付ファイルの内容を検索する 687
文字抽出プログラムの指定 688
仕様・制約 688
インデックスファイルについて 690
サジェスト機能 690
検索画面のみに表示され、検索条件には使われない項目を用意する
例 690
定義方法 691
濁音や拗音の違いを考慮したカナ検索の実現
例 695
定義方法 696

入力フィルタを使わずに全角半角文字が混在した文字列を検索する
例 699
定義方法 701
通常の検索画面とサブウインドウ検索画面で検索条件の表示・非表示を個別に指定する
検索条件の隠し項目設定 704
絞り込み設定との併用 707
検索条件テンプレート
例 710
定義方法 711
スクリプトを利用して、テンプレートを選択するとすぐに検索を行うようにする 719
外部から検索条件をパラメータとして指定する
基本形 719
パラメータを指定する 719
範囲指定 720
モデル参照 720
複数のパラメータを指定する 721
検索条件に名前をつけて再利用する
設定方法 721
ユーザ単位での利用 721
検索条件を共通利用する 725
削除 726
仕様・制約 727
クライテリアを直接操作して、検索条件をカスタマイズする(スクリプト)
クライテリア 727
例 OR検索 727
クライテリアの詳細 729
利用できる検索条件 730
仕様・制約 731
レイアウト
基本構成
標準レイアウト 731
項目を横に並べる 732
項目を縦に並べる 739
横並びと縦並びを組み合わせる 740
画面サイズを変えてみる 744
隠し項目の扱い 745
操作に関するTips 745
入れ子レイアウトと隠しレイアウト
縦並びの中に横並び 745
横並びの中に縦並び 749
隠しレイアウトの混在 753
隠しレイアウトへの入れ子レイアウト 758
レイアウトラベルを独立して用意する
縦並び 763
横並び 765
組み合わせ 766
ラベル幅を合わせる 767
ウィザードとタブレイアウト
ウィザードとタブレイアウトの違い 769
タブレイアウトを設定する 770
デフォルトタブの指定 775
項目の値によってデフォルトタブを制御する 776
タブの入れ子 780
タブと他レイアウトとの組み合わせ 783

複数のタブコンテナを用意する. 785
タブの表示位置. 787
タブ表示制御. . . . . .
ラジオボタンによる制御. 789
チェックボックスによる制御. 791
値による制御. 794
仕様・制約. 797
タイトルペイン. . . . . .
実行例. 797
設定方法. 798
仕様・制約. 802
検索条件の並び順. . . . . .
定義方法. 803
例. 804
表示順の設定に関する詳細. 805
範囲検索条件の入力欄のレイアウト. . . . . .
定義方法. 805
例. 805
検索条件部の段組レイアウト. . . . . .
例. 806
定義方法. 807
仕様・制約. 809
検索条件欄を折り畳む. . . . . .
例. 809
定義方法. 810
表示設定の詳細. 811
一覧表示の並び順. . . . . .
定義方法. 812
実行例. 814
表示順の設定に関する詳細. 815
一覧表示 － 複数段一行明細(多段)レイアウト. . . . . .
基本的な設定. 815
セル結合（列）. 818
セル結合（行）. 820
一覧更新への適用. 822
外部キー子モデル － 複数段一行明細(多段)レイアウト. . . . . .
例. 823
定義方法. 825
繰り返しコンテナ － 列幅の調整. . . . . .
例. 825
定義方法. 826
設計上の注意点. 827
繰り返しコンテナ － スクロールバーの表示（と幅、高さの設定）. . . . . .
横スクロールバー. 827
高さと幅の設定 (8.3.3). 828
繰り返しコンテナ － タブ. . . . . .
例. 829
定義方法. 830
デフォルトタブを指定する. 831
繰り返しコンテナ － 複数段一行明細(多段)レイアウト. . . . . .
複数段一行明細レイアウトの設定. 832
列の並びを合わせる. 834
セル結合（行）. 836
セル結合（列）. 838

トラブルシューティング... 841
繰り返しコンテナ - タブと複数段一行明細(多段)レイアウト...
例... 842
定義方法... 842
繰り返しコンテナ - ヘッダ部のみの複数段一行明細(多段)レイアウト...
例1... 845
定義方法1... 846
例2... 847
定義方法2... 848
例3... 850
定義方法3... 850
仕様・制約... 853
繰り返しコンテナ - 縦並び...
縦並びレイアウト... 853
縦並びと横並びの組み合わせ... 855
仕様・制約... 856
繰り返しコンテナ - 複数の繰り返しコンテナの横並び...
横並びレイアウト... 856
定義方法... 857
繰り返しコンテナ - 複雑なタブ設定...
タブの中に通常項目と繰り返しコンテナを含める... 858
タブの中に通常項目と繰り返しコンテナを含め、通常項目を横並びとする... 860
一つのタブに繰り返しコンテナを丸ごと格納する... 862
繰り返しコンテナ横並びをタブに格納する... 863
繰り返しコンテナ複数段一行明細レイアウトをタブに格納する... 864
外部キー子モデルおよび繰り返しコンテナを親モデルのタブにまとめる...
外部キー子モデル... 866
繰り返しコンテナ... 869
外部キー子モデルと繰り返しコンテナ... 871
仕様・制約... 873
繰り返しコンテナ、外部キー子モデル一覧を任意の位置に配置する...
繰り返しコンテナの配置... 873
外部キー子モデル一覧の配置... 875
説明文・脚注を挿入する...
例... 877
定義方法... 878
説明文・脚注を枠で囲う... 881
CSS レイアウトを利用する (CSS)...
CSS レイアウトとは... 882
定義方法... 882
カスタマイズ... 883
仕様・制約... 883
ログオン画面の背景画像を変更する...
背景画像を保存する... 883
スタイルの設定 (8.3.0)... 883
CSSのカスタマイズ (CSS)...
変更するファイル... 884
例 ページタイトルの背景色を変更する... 885
例 フォーカスセット時に背景色と入力欄を囲む枠を表示する... 885
例 一覧表示の項目間を黒色にする... 886
例 モダンカレンダビューの文字サイズを固定する... 887
任意のCSSを追加する... 887
JavaScriptのカスタマイズ (JavaScript)...
変更するファイル... 887
任意のJavaScriptを追加する... 888

例 ボタン押下時にダイアログを表示する 888
例 特定のボタンを非表示にする 889
例 任意のボタンにTipsを用意する 889
入力フォーカスの制御 891
メニュー表示エリアの高さを変更する 892
出力制御（スタイル）
隠し項目（詳細・一覧）
詳細画面 892
一覧画面 893
ラベルの表示内容
登録・更新・詳細画面 893
一覧表示画面 894
検索画面 894
色のルール 895
ラベルと値の表示方法
画面例 895
定義方法 896
すべてを縦とした場合 897
ラベルを表示しない
画面例 898
定義方法 899
仕様・制約 899
ラベルの幅
標準レイアウトでラベル幅を変更する 900
縦並びレイアウトでラベル幅を変更する 901
横並びレイアウトでラベル幅を変更する 902
比率ではなくピクセル値を指定する 903
一覧表示画面への設定 903
検索画面への設定 904
入力ルール (8.3.2) 904
ラベルの高さ
画面例 904
定義方法 905
値の高さとの関係 905
表示位置の制御（横）
詳細画面 907
一覧画面 908
表示位置の制御（縦）
例 910
定義方法 911
表示の折り返し
詳細画面 912
一覧画面 914
折り返されない場合の対応方法 917
CSSクラス名・スタイルの指定
CSSクラス名 917
ラベルのスタイル 917
値のスタイル 918
例 ラジオボタン・チェックボックスの高さを指定する 919
一覧表示への設定 921
条件によってスタイルを変更する 922
仕様・制約 923
改行
改行指定が必要な例 923

定義方法 924
設定後の画面 924
スタイルシート 925
ルビ
詳細画面 925
詳細画面 － 隠し項目指定との併用 926
一覧画面 927
ヘルプメッセージ
ヘルプボタン 928
項目ラベルへのメッセージ 930
入力フィールドへのメッセージ 931
マウスオーバー時のメッセージ（ツールチップ） 933
プレースホルダのメッセージ 934
改行の扱い (8.3.0) 935
バーゲージ
実行例 936
定義方法 938
一覧表示の幅を調整する 939
[カスタマイズ] JavaScriptを使って色を変更する 940
バーコード出力
実行例 940
定義方法 941
カスタマーバーコードを出力する 942
帳票での利用 947
関連するページ 948
ラベルの表示内容を式で制御する(スクリプト)
実行例 948
定義方法 949
画面機能
検索・一覧表示（検索）
検索と一覧を同時に表示する 952
検索条件の保持ルール 953
ページタイトル 953
説明文 953
検索画面遷移直後の検索処理 955
リセットボタン押下時の検索処理 955
検索結果が1件のとき詳細画面へ 956
検索結果が1件のとき更新画面へ 956
検索結果が0件のとき登録画面へ 957
検索結果数の上限指定 958
検索ボタンの表示位置 959
検索・一覧表示（一覧表示）
同時表示数 960
ページネーション 960
テーブルの幅 962
横スクロールバー表示 962
スクロールタイプ 963
アクションボタンの表示位置 963
ボタン（機能） 964
ダウンロード 967
グラフの出力 971
仕様・制約 971
行クリックで詳細画面へ 971
検索・一覧表示（一覧表示）グリッドを利用する
定義方法 972

実行 973
同時表示数とページネーション 976
縦スクロールバーを表示する 977
スクロール時にデータを読み込む(onDemand Grid) 978
一括処理 979
設定を共通で利用する 979
グラフの出力 983
仕様・制約 983
一括処理
一括処理とは 984
一括削除 985
一括更新 986
選択方式 987
グリッドでの設定 990
一覧表示の集計
例 992
定義方法 992
数値の表示フォーマット 993
対応する関数 994
仕様・制約 994
詳細表示
ページタイトル 995
説明文 996
アクションボタンの表示位置 996
用意されているボタン（機能） 998
前後のデータにアクセスする 999
新規登録
ページタイトル 1000
説明文 1000
確認画面を用意する 1001
アクションボタンの表示位置 1004
用意されているボタン（機能） 1005
キャンセル操作時のダイアログ 1007
仕様 1008
コピー登録
コピー登録画面を用意する 1008
キャンセル操作時のダイアログ 1010
コピー対象項目を指定する 1010
複数のコピーボタンを用意する 1011
コピー時に式を適用する 1012
ページタイトル 1013
説明文 1013
アクションボタンの表示位置 1013
用意されているボタン（機能） 1013
外部キー子モデルを含む場合 1013
コピー登録時に枝番を付与する 1013
更新
悲観ロック 1013
ページタイトル 1014
説明文 1015
同一ユーザのロックを自動的に解除する（悲観ロック利用時） 1016
確認画面を用意する 1016
更新対象データが存在しないとき登録画面に遷移する 1019
アクションボタンの表示位置 1019
用意されているボタン（機能） 1021
前後のデータにアクセスする 1023
キャンセル操作時のダイアログ 1023

削除
詳細表示画面 1024
一覧表示画面（一括削除） 1024
削除後に遷移する画面 1026
一覧更新
基本動作 1026
テキストエリアの行数・文字数 1027
一覧更新画面の同時表示数 1029
一覧更新画面のページネーション 1029
一覧更新ボタンの表示制御 1029
用意されているボタン（機能） 1030
データベースコミット前に呼び出されるスクリプト 1030
ページタイトル 1031
説明文 1032
更新処理を許可する 1033
削除のチェックボックスを表示する 1034
ロック制御 1035
一覧表示部分のスクロールタイプ（列固定） 1035
一覧表示部分のスクロールタイプ（ヘッダ行固定） 1037
仕様と制約 1039
一覧更新　グリッドを利用する
定義方法 1040
基本機能 1040
ページタイトル 1042
説明文 1042
更新処理を許可する 1042
ロック制御 1043
テキストエリア 1043
一覧更新画面の同時表示数 1044
一覧更新画面のページネーション 1044
仕様と制約 1044
用意されているボタン（機能） 1044
ウィザード
基本的な使い方 1044
タブレイアウトとの違い 1047
画面指定ボタン 1047
画面移動ボタン 1048
全画面共通の項目 1049
確認画面を用意する 1051
ウィザード表示を一つにまとめる 1052
条件によってボタンを制御する 1054
仕様と制約 1058
カレンダビュー（クラシック）
例 1058
定義方法 1062
複数のモデルを同時に表示する 1065
六曜を表示する 1067
カレンダー移動方式 1067
日付を検索条件に加える 1069
ポータル画面に表示する 1070
休日の扱い 1070
仕様・制約 1073
クラシックカレンダビューからモダンカレンダビューへの移行 1074
カレンダビュー（モダン）
例 1074
休日の扱い 1076
モダンカレンダビューの文字サイズを固定する 1076

仕様・制約 1076
クラシックカレンダビューからモダンカレンダビューへの移行 1077
定義方法 1077
カレンダ表示時のモデルの色 1079
複数のモデルを同時に表示する 1080
カレンダの時刻表示範囲 1082
ドラッグ&ドロップによる変更 1083
終日イベントとする 1083
ポータル画面に表示する 1085
カレンダビューから新規登録画面へ遷移する 1085
独自の検索初期値を設定する 1087
マップビュー（Google Maps）
ご利用の前に 1088
APIキーの設定 1088
例 1089
定義方法 1089
マップビュー定義のルール 1091
一覧表示画面 1092
半径、MBR、重ね画像 1093
マーカーの色と画像を指定する 1095
スタイルを指定する 1101
マップビューにポリゴンを描画する
例 1103
定義方法 1105
ダウンロード 1108
メインモデルとサブモデル
概要 1108
サブモデルの作成方法 1109
例 1111
具体的な活用シーン 1113
権限設定との組み合わせ 1113
メインモデルを変更する 1120
サブモデル一覧画面からメインモデル詳細画面へ遷移する 1120
仕様・制約 1122
よくある質問と回答 1123
ボタン名と表示条件
ボタン名の変更例 1124
ボタン名の定義方法 1124
ボタン表示条件 1125
外部キー関係での式の扱い 1127
Ajaxによる画面描画
Ajaxの概要 1128
Ajaxが動作するパターン 1128
Ajaxの多重度 1129
画面遷移 (1) 詳細画面から登録画面へ遷移する
詳細画面から登録画面への遷移 1129
定義方法 1130
モデル参照項目の扱い 1132
仕様・制約 1134
画面遷移 (2) 検索画面へ遷移する
詳細画面から検索画面への遷移 1135
定義方法 1136
モデル参照項目の扱い 1138
検索画面から検索画面への遷移 1141
仕様・制約 1143
画面遷移 (3) 一覧表示画面から別のモデルへ遷移する

一覧表示画面から詳細画面への遷移. 1143
一覧表示画面から更新画面への遷移. 1146
一覧表示画面から一覧更新画面への遷移. 1147
削除. 1148
印刷. 1150
パラメータに検索条件の値を用いる (8.2.0). 1152
仕様・制約. 1152
画面遷移 (4) 詳細画面から更新画面へ遷移する（更新データがない場合、登録画面へ自動的に遷移する）. . . . . .
例. 1152
定義方法. 1154
仕様・制約. 1158
画面遷移 (5) メインモデルの更新画面からサブモデルの更新画面へ遷移する(スクリプト). . . . . .
例. 1158
定義方法. 1159
仕様・制約. 1161
画面遷移 (6) スクリプトを使った画面遷移(スクリプト). . . . . .
スクリプトの作成方法. 1161
呼び出されるタイミング. 1162
スクリプトの記述ルール. 1162
[例] 登録画面から、再び登録画面へ遷移する. 1162
[例] 更新処理から、メニューへ遷移する. 1163
[例] ウィザードの任意の画面へ遷移する. 1168
[例] 独自ボタン押下で、マルチセッションを使って任意の画面を開く. 1168
フォワードとリダイレクト. 1169
仕様・制約. 1169
画面遷移 (7) 独自ボタンから外部コマンドを呼び出す(スクリプト). . . . . .
例. 1170
定義方法. 1171
スクリプトを用意する. 1171
ボタン押下後に再び自画面に戻る. 1173
画面遷移 (8) 登録・更新画面に独自ボタンを用意する(スクリプト). . . . . .
仕様・制約. 1174
例 登録画面から別モデルの検索画面へ遷移する. 1174
例 ボタン押下時に繰り返しコンテナを追加する. 1174
リアルタイム入力チェックを無効にする (8.3.1). 1176
画面遷移 (9) 画面遷移パラメータまとめ. . . . . .
アクション名. 1177
イベント名. 1177
モデル参照項目のルール. 1180
遷移先モデルが複合キーの場合. 1181
遷移先モデルのモデル参照項目が複合キーの場合. 1181
遷移元モデルのモデル参照項目が複合キーの場合. 1182
チェックボックスの場合. 1182
画面遷移 (10) JavaScript(フロントエンド)との連携(スクリプト)(JavaScript). . .
画面遷移時に確認ダイアログを表示する. 1183
論理削除. . . . . .
一般利用者と業務管理者を用意する. 1185
例. 1186
定義方法. 1189
判定項目を独自に設定する. 1192
仕様・制約. 1193
登録データの一時保存. . . . . .
例. 1193

定義方法 1196
確認画面を利用する場合 1200
ワークフローとの関係 1204
設計上の注意点 1205
一覧表示画面から詳細画面（サブウィンドウ）を開く
例 1205
定義方法 1206
ウィンドウの高さ、幅、位置、名称をボタン毎に指定する 1208
仕様・制約 1208
一覧表示からの選択（独自ボタン利用）（スクリプト）
概要 1209
設定方法(画面に表示されているデータを一括処理の対象とする) 1209
動作例 1210
全データを一括処理の対象とする方式の場合 1212
仕様・制約 1214
スクリプト
ヘルパとコントローラ （スクリプト）
ヘルパのスクリプト 1214
コントローラのスクリプト 1218
共通スクリプト 1218
トランザクション 1218
キャメル記法 1219
呼び出されるスクリプトファイル名を確認する 1219
モデルの基本 (1) （スクリプト）
モデルとオブジェクト 1220
ファイル型 1220
生成されたモデルのJavaコードを直接確認する 1221
モデルの種類 1221
ストアモデル 1223
モデルの基本操作 1225
DAO 1225
繰り返し 1226
モデル参照（リストボックス、ラジオボタン、検索ウィンドウ） 1228
モデル参照（チェックボックス） 1228
外部キー 1230
参照連動 1230
モデルの基本 (2) （スクリプト）
プレゼンテーションモデル 1231
コンディションモデル 1235
リストモデル 1235
生成されるスクリプトファイルの詳細 （スクリプト）
ジョブパラメータ (8.2.0) 1235
基本テンプレート 1235
ルール 1236
ドキュメンテーションコメント (8.3.0) 1237
Java 標準クラスを利用する 1237
利用できる変数 1237
共通して利用できる関数 (8.3.0) 1239
独自に拡張したスクリプトファイルを読み込む (8.1.1) 1240
関数 preprocess を用意する 1240
複数のスクリプト間でデータを受け渡す 1241
Wagby関数を利用する （スクリプト）
利用方法 1241
利用できない関数 1241
Javaクラスによって代替する関数 1242

スクリプトを用いた主キーの拡張（スクリプト）
例 1243
定義方法 1244
順序を定義する 1246
スクリプトを実装する 1247
サブデータベースで順序を使う（スクリプト）
準備 1248
例 PostgreSQLからserial値を取得する 1248
例 MySQLのAUTO_INCREMENTを利用する 1249
スクリプトでの初期化処理（スクリプト）
定義方法 1250
モデルを新規に作成する場合 1250
[例] 親子モデル同時更新機能で親モデル登録時、子モデル毎に異なる初期値を設定する 1251
入力チェック・整合性チェック（スクリプト）
定義方法 1251
[例] EntityServiceとCriteriaを用いた存在チェック 1252
[例] 一覧更新画面の入力チェック (1) 画面に表示されているデータのみ 1252
[例] 一覧更新画面の入力チェック (2) 画面に表示されていないデータも含める 1258
プレゼンテーションモデル 1259
エラーメッセージ 1259
警告 1260
入力欄の背景色の設定 1260
エラーメッセージにHTMLタグを含める 1261
[例] 整合性チェック 1261
[例] 子モデルの存在チェック 1262
[例] 親モデルと子モデルの値を比較する 1262
[例] SQLを用いた件数チェック 1262
入力フィルタ（スクリプト）
定義方法 1263
[例] 入力値を補完する 1264
[例] 文字列拡張検索の標準を OR 検索とする 1264
一括更新（スクリプト）
定義方法 1265
[例] 選択されたモデルのフラグをセットする 1265
[例] チェックボックスの値をトグルスイッチとして扱う 1265
[例] 複数のボタンを扱う 1265
更新後に再び計算処理を行う 1266
一括更新後の画面遷移 1266
[例] 更新された数を画面遷移後に表示させる 1267
仕様・制約 1268
一覧帳票出力後の更新（スクリプト）
定義方法 1268
[例] 出力後に「帳票出力日」という項目に値をセットする 1268
[例] 出力した数をカウントアップする 1269
[例] 複数の帳票ボタンによって処理を変える 1269
仕様 1270
モデルをまたがる計算（トランザクション）（スクリプト）
在庫管理のイメージ 1270
例 1271
定義方法 1273
文法 1276
繰り返しコンテナとの連携 1276
外部キー子モデルとの連携 1278
対象データがロックされている場合の動作 1278
Service/DaoクラスやSQLを利用する 1279
トランザクション境界 1279

仕様・制約 1279
Service/Daoクラス、ヘルパ、SQLを利用する（スクリプト）
Serviceを利用する 1279
トランザクションスクリプトでSQLを併用する 1281
Service内部で動作するスクリプトを用意する 1282
SQLを利用する 1283
ストアドプロシージャを利用する 1284
SQLやストアドプロシージャでデータを更新する 1287
プログラムによるトランザクション管理 1289
もっと詳しく 1290
Daoを利用する 1290
ヘルパを利用する 1291
ロックの取得と解除 1291
クライテリアを利用する 1292
複合キーの場合 1294
例外発生によるロールバック 1294
トランザクション境界 1295
トランザクションスクリプトでもServiceを利用するケース 1296
トランザクションスクリプトで参照連動の解決を抑制する 1299
環境タブで設定できる共通スクリプト（スクリプト）
共通関数 1300
ファイル項目ダウンロード 1301
ファイル項目アップロード 1301
応用 ウィルスチェック 1302
メールを送信する（スクリプト）
MailMessageマネージャを利用する 1303
ログ出力（スクリプト）
ログ出力 1303
サンプルコード（スクリプト）
繰り返しコンテナ項目の日付を使って、新規データを生成する 1304
繰り返しコンテナで、ある値が空のコンテナ行を削除する 1305
参照連動処理を呼び出す 1305
外部コマンドを実行する 1306
スクリプトの再読み込み処理を制御する
起動時のパラメータオプション 1306
仕様・制約
暗黙オブジェクトrequestやsessionへ値を保存する場合の注意点 1306
ポータル
ポータル機能を使う
ポータル機能を有効にする 1307
利用するポートレットを選択する 1308
表示エリアと表示順 1310
ドラッグ&ドロップでポートレットの配置を変える 1311
ポートレットの位置を固定にする 1312
システム管理者以外のアカウントでログオンする 1312
メニュータブとの関係 1313
ポータルの表示エリアの高さを変更する 1313
ポータルのタブ名を変更する 1313
トラブルシューティング 1313
お知らせポートレット
お知らせの登録 1314
お知らせポートレットの表示 1315
高さの調整 1316
検索パラメータ 1317
見栄えの変更 1318

カレンダポートレット（クラシック）
カレンダポートレットの設定 1319
カレンダポートレットの表示 1320
カスタマイズ設定 1322
仕様・制約 1323
カレンダポートレット（モダン）
カレンダポートレットの設定 1323
カレンダポートレットの表示 1324
カスタマイズ設定 1324
仕様・制約 1325
ログオンユーザー覧ポートレット
ログオンユーザー覧ポートレットの設定 1325
ログオンユーザー覧ポートレットの表示 1325
カスタマイズ設定 1326
グラフポートレット
REST API を有効にする 1326
グラフポートレットの設定 1327
グラフポートレットの表示 1328
複数のグラフポートレットを表示する 1328
利用上のヒント 1330
ワークフローポートレット
ワークフローポートレットの設定 1330
ワークフローポートレットの表示 1332
決裁処理を行う 1332
外部リンクポートレット (JavaScript)
例 1334
定義方法 1334
ブログパーツを利用する (JavaScript)
ブログパーツ 1336
アナログ時計 1336
任意のモデルの値を表示する (JavaScript)
REST API と JSON 1338
「日報モデル」を用いたサンプル実装 1338
見栄えの修正 1342
任意のモデルの値を表示する (2) (JavaScript)
準備 1343
プレゼンテーションモデル 1343
dojo/requestとdojo/dom-constructを用いたサンプル 1345
JSONオブジェクトへの変換 1347
JSONオブジェクトから値を取得する 1347
ゲージ部品を利用する(JavaScript)
準備 1348
GlossyCircularGaugeを使う 1348
GlossySemiCircularGaugeを使う 1349
GlossyHorizontalGaugeを使う 1351
百分率として利用する場合 1352
CSV・Excel入出力
基本的な使い方
定義方法 1352
ダウンロード機能を試す 1354
アップロード更新機能を試す 1356
CSVファイルの仕様 1358
ダウンロード
項目の型に応じたルール 1359

iPadでダウンロードする場合 1360
多言語対応 1360
トラブルシューティング 1361
参考:Excelが提供する「データ取り込み」機能の使い方 1361
仕様・制約 1363
ダウンロード画面の設定 1364
ダウンロード機能の設定 1364
ダウンロードファイル名の設計 1366
項目の設定 1369
繰り返し項目・繰り返しコンテナ項目の扱いに関するルール 1369
ダウンロードファイルの取り扱いに関するルール 1370
権限管理機能との組み合わせ 1371
一覧表示画面で直接、ダウンロードする 1372
サーバにファイルを保存する|8.0.2 1372
実行時にダウンロード項目を指定する
例 1373
定義方法 1374
設定を共通で利用する 1376
仕様・制約 1376
アップロード更新
処理の詳細 1379
ログオンセッションの数え方 1381
トラブルシューティング 1382
アップロード更新画面の設定 1383
アップロード更新機能の設定 1384
項目の設定 1387
処理件数の扱い (8.3.0) 1388
エラー時の動作 1388
更新結果レポートの確認 1388
独自に用意したファイルをアップロードする 1389
複数のファイルをzip圧縮して送信する 1390
Excel利用時に数値フォーマットを指定する 1390
アップロード更新（繰り返し項目と繰り返しコンテナ）
繰り返し項目（モデル全体） 1391
繰り返し項目（独立したファイル） 1391
繰り返しコンテナ（モデル全体） 1392
繰り返しコンテナ（独立したファイル） 1393
制約 1394
アップロード更新（ファイル型項目）
用途 1395
アップロード方法 1395
繰り返しコンテナへの対応 1396
添付ファイルをサブフォルダで管理する 1396
ファイルが正常にアップロードされない場合 1397
ダウンロードジョブ
定義方法 1397
ジョブの登録 1398
指定できるジョブパラメータ 1399
ジョブの実行を停止する 1400
実行結果をメールで送信する 1400
多重実行の抑制 1400
すぐに実行する 1400
仕様・制約 1401
アップロード更新ジョブ
定義方法 1401
ジョブの登録 1401
指定できるジョブパラメータ 1402

ジョブの実行を停止する... 1406
実行結果をメールで送信する... 1406
多重実行の抑制... 1406
すぐに実行する... 1406
仕様・制約... 1406
ダウンロード時にスクリプトを呼び出す(スクリプト)...
ヘルパ... 1407
コントローラ... 1408
アップロード更新時にスクリプトを呼び出す(スクリプト)...
ヘルパ... 1409
コントローラ... 1411
固定長ファイルのダウンロードとアップロード更新(スクリプト)...
ダウンロード... 1413
アップロード更新... 1414
帳票...
HTML帳票...
一覧表示... 1416
詳細表示... 1417
Excel帳票 - 詳細表示...
例... 1418
ライブラリの選択... 1420
仕様・制約... 1420
トラブルシューティング... 1420
定義方法... 1421
ボタン名を変更する... 1425
出力方法を変更する... 1426
出力ファイルを変更する... 1427
繰り返し項目の固定長指定... 1428
繰り返し項目の可変長指定... 1429
外部キー連携モデルの扱い... 1430
ワークフロー（フローイベント）の扱い... 1431
登録・更新画面に帳票ボタンを用意する... 1432
Excel帳票 - 一覧表示...
例... 1433
出力データが存在しなかった場合のエラーメッセージ... 1434
ライブラリの選択... 1436
仕様・制約... 1437
トラブルシューティング... 1437
定義方法... 1437
ボタン名を変更する... 1441
検索処理後にボタンを表示させる... 1442
出力ファイルを変更する... 1443
出力方法を変更する... 1444
一覧表示画面には表示させないが、一覧帳票には出力させる... 1444
一覧表示画面には表示させるが、一覧帳票には出力させない... 1444
一覧表示テンプレートのルール... 1445
同一値によるグルーピング... 1446
プレースホルダ...
プレースホルダの書式... 1447
プレースホルダのルール... 1447
表記一覧（詳細表示）... 1448
表記一覧（一覧表示）... 1451
表記一覧（共通）... 1452
制約... 1452
複雑なレイアウトの作成...

1シート目にプレースホルダを集約させる... 1453
複数データの帳票を一度に出力する(一括帳票出力)...
例... 1455
定義方法... 1457
Excel出力の詳細... 1458
PDF出力の詳細... 1460
詳細帳票のファイル名にモデル項目の値を指定する... 1460
仕様・制約... 1461
一覧表示を非同期で出力する...
例... 1461
定義方法... 1463
一覧帳票出力ジョブの確認... 1465
仕様・制約... 1465
一覧帳票出力用の専用画面を用意する...
例... 1465
定義方法... 1466
出力する帳票の初期値... 1467
選択肢が1件の場合に出力する帳票を非表示とする... 1467
独自ボタンを用意する (8.2.0)... 1468
仕様・制約... 1469
出力内容をカスタマイズする(スクリプト)...
定義方法... 1469
仕様・制約... 1470
画像の扱い... 1471
JasperReportsを利用したPDF出力 (1) 詳細表示...
JasperReportsとは... 1471
準備するもの... 1472
Jaspersoft Studioを使ったレイアウト設計... 1473
XPath指定の詳細... 1479
ビルドと実行... 1479
JDBCによるデータベース接続... 1481
利用するライブラリのバージョン... 1482
よくある質問と回答... 1482
トラブルシューティング... 1483
JasperReportsを利用したPDF出力 (2) 一覧表示...
はじめに... 1483
準備するもの... 1483
Jaspersoft Studioを使ったレイアウト設計... 1484
XPath指定の詳細... 1486
JDBCによるデータベース接続... 1486
利用するライブラリのバージョン... 1487
トラブルシューティング... 1487
JasperReportsを利用したPDF出力 (3) ファイルの結合...
動作... 1487
定義方法... 1487
メール送受信...
メール送信（メールテンプレートを固定する）...
定義方法... 1488
成功時、失敗時の処理... 1489
送信条件... 1490
リトライ回数と間隔を指定する... 1490
ログにCcとBccアドレスを含める... 1490
トラブルシューティング... 1491
仕様・制約... 1491
メールテンプレート... 1491

プレースホルダ... 1495
例 (1) 処理完了時の自動送信... 1496
例 (2) メール送信ボタンによる手動送信... 1497
宛先人の詳細... 1498
ファイルを添付する... 1498
メールヘッダの設定... 1498
メールサーバとプロトコルの詳細... 1499
ボタン名と表示条件... 1499
メール送信（実行時にメールテンプレートを指定する）...
定義方法... 1499
例... 1502
メール送信（ジョブによる時間指定）...
ジョブの設定... 1503
メールテンプレートの設定... 1505
ジョブスケジュールの設定... 1505
ジョブの実行を停止する... 1507
実行結果をメールで送信する... 1507
多重実行の抑制... 1507
メール受信（ジョブによる時間指定）...
設定... 1507
仕様・制約... 1510
トラブルシューティング... 1510
ジョブスケジュールの設定... 1512
例... 1513
メール本文の自動解析... 1514
メール本文をリッチテキストエリアに格納する... 1515
ジョブに指定できるパラメータ... 1515
ジョブの実行を停止する... 1515
実行結果をメールで送信する... 1516
多重実行の抑制... 1516
juserモデルのメール受信... 1516
メールサーバのサンプル...
Gmail... 1516
Office365... 1517
トラブルシューティング... 1517
メールヘッダを使ってメールのスレッド表示を実現する...
例... 1517
定義方法... 1519
注意点... 1521
集計・グラフ...
数値の集計...
例... 1522
定義方法... 1522
数値フォーマットの扱い... 1524
集計結果が不定の場合... 1524
選択肢の集計...
例... 1524
定義方法... 1525
文字列の集計...
例... 1527
定義方法... 1528
メジャー毎に合計・個数・平均・最大・最小・積・中央・最頻・標準偏差・分散を指定する...
例... 1529
定義方法... 1530

１つのモデルに複数の集計ビューを用意する
例 1532
定義方法 1534
グラフも同時に用意する 1535
1つの画面に複数の集計ビューを表示する 1536
集計結果をキャッシュする
本機能の概要と効果 1538
定義方法 1538
キャッシュのキー 1538
グラフ
例 1538
定義方法 1539
検索条件（コンディション）の自動設定 1543
比率を表示する 1544
仕様・制約 1545
グラフ － 実行時の指定
気象データサンプル 1545
ディメンジョンとメジャーの切り替え 1548
グラフの種類（全体） 1548
グラフの設定（個別） 1548
詳細設定 1550
グラフを非表示にする 1552
設定を保存する 1552
ディメンジョンの並び替え 1555
ディメンジョンを分類する
例 1556
定義方法 1557
見栄えの調整
共通スタイル 1560
メジャー項目毎のラベル、スタイル設定 1561
一覧表示画面にグラフを表示する
折れ線グラフ 1562
棒グラフ 1565
[応用] JavaScript を使ってグラフをカスタマイズする 1567
[応用] 一覧表示グリッドでグラフを表示する 1567
[応用] 散布図 1567
仕様・制約 1567
一覧表示画面にグラフを表示する (2) (JavaScript)
準備 1567
表示位置の変更、ラベルの作成 1568
取得するデータサイズを変更する 1569
一覧表示グリッドでグラフを表示する 1569
一つのグラフにまとめる 1572
散布図 1575
ダウンロード 1577
詳細画面にグラフを表示する (JavaScript)
実行例 1577
Designerでの設定 1577
JavaScriptでの追加設定 1588
仕様・制約 1590
認証・認可
権限管理の概要
ロール 1590
パーミッションとプリンシパル 1591
標準のパーミッション 1592

標準のプリンシパル 1593
カテゴリ (8.4.0) 1595
ログオンアカウントにプリンシパルを割り当てる 1596
モデル項目単位で権限を設定する 1598
権限管理の対象範囲 1598
システム管理者の扱い 1598
技術情報 1599
モデル単位の制御
新しいプリンシパルの追加 1599
プリンシパルにパーミッションを設定する 1600
テストアカウントの作成 1602
実行例 1604
メニューの扱い 1606
応用 1607
項目単位の閲覧権限（閲覧または不可視）
定義方法 1607
テストアカウントの作成 1611
例 1612
本設定により影響を受ける機能 1614
仕様・制約 1614
項目単位の更新権限（更新可または閲覧のみ）
定義方法 1614
テストアカウントの作成 1617
例 1619
本設定により影響を受ける機能 1620
本設定の影響を受けない機能 1620
仕様・制約 1620
項目単位の閲覧更新権限の同時設定
定義方法 1620
例 1621
条件によって権限の適用可否を変更する
概要 1623
定義方法 1624
例 1625
登録画面では入力可能だが、更新画面では入力不可としたい 1626
更新権限が無効になったときの動作 (8.3.0) 1627
仕様・制約 1628
グループ権限
概要 1629
グループモデル - jgroup 1629
グループ権限のパターン 1629
グループ権限の動作の仕組み 1630
グループ管理者 1632
代理登録 1633
グループ権限設定を解除する 1634
データをグループの持ち物とする 1634
グループ階層を利用する 1635
仕様・制約 1639
応用例 複数項目の組み合わせパターンに応じたデータの閲覧制御
サンプルの仕様 1639
例 1640
定義方法 1643
応用例 データベースのビューにグループ権限を適用する
Wagbyが利用するカラムの追加 1646
アカウント

juserモデルの基本
用意されている項目 1646
項目を追加する 1648
アバター画像 1651
仕様・制約 1653
パスワード
定義方法 1653
パスワードハッシュ化 1653
アカウントロック 1654
パスワード有効日数 1655
パスワード有効日数警告 1655
パスワード再利用制限回数 1656
パスワードの一括更新 1656
利用できる文字を制限する 1656
パスワードを忘れたとき 1656
運用中のアプリケーションの挙動を直接、変更する 1657
CSVダウンロードとアップロード更新
ダウンロード 1657
アップロード更新 1658
パスワード項目の扱い 1659
よくある質問と回答 1660
パスワードリマインダ
利用イメージ 1660
セキュリティ上の問題と対策 1663
定義方法 1663
メールアドレスを利用者に設定させる場合 1665
システム内部でログオンアカウントに紐づくメールアドレスを保持している場合 1665
パスワードリマインダ機能の利用制御 1667
オートスケール環境での動作 (8.3.1) 1668
よくある質問と回答 1668
仕様・制約 1668
IPアドレスによるログオン制限
定義方法と運用 1669
テスト 1670
juserのサブモデルの作成
制約 1671
作成方法 1671
グループの使い方
グループの作成 1677
グループの使い方 1679
グループ階層 1679
仕様・制約 1684
Windows Active Directory/LDAP, SSO による認証
概要 1684
認証方式の設定 1684
アカウントとパスワードの設定 1685
仕様・制約 1686
技術情報 設定ファイルの詳細 1686
旧バージョンからの移行 1686
シングルサインオン 1686
トラブルシューティング 1687
サポートポリシー 1687
OpenID Conneect による認証
概要 1688
準備するもの 1689
Googleを利用する 1690

ユーザ識別子 1696
Azure Active Directory を利用する 1696
その他のOPを利用する 1698
サポートポリシー 1698
デフォルトログオン
概要 1699
定義方法 1699
例 1700
例2 パスワードなしでの利用に限定する 1701
運用中にアカウント名、パスワードを変更する 1701
レイアウト 1702
Active Directory と併用する場合 1702
デフォルトログオンを回避するパラメータ (8.4.2) 1702
仕様・制約 1702
マルチセッション
概要 1702
定義方法 1703
応用 スクリプトによる画面遷移 1704
仕様・制約 1704
ワークフロー
モデルにワークフロー定義を行う
年休申請モデルの定義 1704
グループの準備 1707
アカウントの準備 1708
フローパターンの設定を行う
標準で提供されるフローパターン 1709
独自のフローパターンを設計する 1710
合議ノードを拡張する 1712
フロー定義ファイル 1713
フローの参加者を設定する
定義方法 1713
設定上の注意 1716
中規模以上の規模での運用（スクリプトを利用する） 1717
モデルにワークフロー定義を設定する
定義方法 1717
合議の承認数の確認 (8.3.0) 1719
注意点 1720
ワークフローを開始する
実行 1720
トラブルシューティング 1725
申請したフローを取り消す 1725
各処理の動作 1726
処理ボタンの表示ルール 1726
ワークフローの状態とイベント 1727
「却下」「取り消し」ボタンを非表示にする (8.3.0) 1728
ボタン名の変更 (8.3.0) 1728
アイコンの変更 (8.3.1) 1729
申請状況の並び順を指定する (8.3.1) 1729
PCで申請したワークフローを、スマートフォンで承認・決裁する (8.1.2) 1729
処理者に電子印影を用いる
実行 1730
定義方法 1731
メールを送信する [1] 設定
概要 1732
アカウントとグループの設定 1733

ワークフローの設定 1734
ワークフロー関数の適用 1738
メールテンプレートの設定 1739
メール送信タイプとノードの組み合わせ 1741
メール文中にワークフロー内部の情報を利用する 1742
メールを送信する [2] 実行
順次・順次・順次フロー 1742
順次・合議・順次フロー 1746
メールを送信する [3] 自動通知
処理日付時刻項目を用意する 1750
メール送信ジョブの定義 1752
メールテンプレートを用意する 1753
メール送信ジョブの起動 1753
例 1754
代理承認
代理承認のイメージ 1754
準備 1755
申請 1755
代理者の設定 1757
代理者の解除 1759
仕様・制約 1760
申請したワークフローの状態を確認する
準備 1760
実行 1760
カスタマイズ事例 - モデル名以外の検索項目を非表示にする 1761
カスタマイズ事例 - 一覧表示項目の変更 1763
カスタマイズ事例 - 開始フロー状態専用のプリンシパルを用意する 1766
仕様・制約 1770
承認すべきワークフローを確認する
準備 1770
実行 1770
保留フローポートレットを利用する 1771
カスタマイズ事例 - モデル名以外の検索項目を非表示にする 1771
カスタマイズ事例 - 一覧表示項目の変更 1774
カスタマイズ事例 - 保留フロー状態専用のプリンシパルを用意する 1777
カスタマイズ事例 - 却下ボタンを非表示にする 1780
仕様・制約 1780
モデルのワークフロー状態で検索する
例 1781
定義方法 1781
サブモデルを用いて特定のフロー状態のみ検索させる 1783
制約 1787
条件によってフローを制御する(スクリプト)
物品購入伺いワークフロー 1787
サンプル (1) 後続の承認者をスキップさせる 1788
実行 1791
サンプル (2) 最初の承認者を選択する 1795
仕様・制約 1796
条件によって申請フローを絞込む(スクリプト)
実行 1796
定義方法 1798
条件の書き方 1803
任意のノードに決裁ボタンを用意する(スクリプト)
例(1) 1803
例(2) 1805
定義方法 1806

条件の書き方 1807
グループ階層とプリンシパルを利用したワークフローの制御（スクリプト）
概要 1807
準備 1807
フロー参加者にスクリプトを設定する 1808
実行 1809
スクリプト内で利用できる暗黙変数 1811
仕様・制約 1818
その他の情報 1819
フローイベントスクリプトを用意する(スクリプト)
定義方法 1819
例外を発生させてワークフローを進めないようにする (8.3.0) 1819
仕様・制約 1821
スクリプトからワークフローを操作する(スクリプト)
申請処理を行うスクリプト 1821
[運用] ワークフローを手動で取り消す
フローイベントとは 1822
操作例 1822
スクリプトからの利用 1824
[応用] ワークフロー決裁後に、フローイベントをリセットして再申請させる（スクリプト）
実行例 1824
定義方法 1826
[応用] ワークフロー差し戻し後の再申請処理で、一度承認した人をスキップして承認待ちの方へ進める（スクリプト）
実行例 1832
定義方法 1834
メニュー
ボタン設定
追加 1839
仕様 1841
トラブルシューティング 1841
移動 1842
背景色 1842
文字色 1843
文字の変更 1844
アイコンの変更 (8.1.0) 1846
ツールチップ 1848
画像を指定する 1848
プロパティ 1851
削除 1851
階層化
準備 1854
階層メニューの追加 1854
階層メニュー間でのメニューの移動 1857
表示方法の切り替え 1858
仕様・制約 1858
タブの管理
ラベルの変更 1858
タブの移動 1859
タブの追加 1860
タブ間のメニューの移動 1862
仕様・制約 1863
外部サイトへのリンク

例 1863
定義方法 1863
Wagbyのユーザアカウントをリンクに埋め込む 1868
権限によってリンクの表示を制御する 1868
外部サイトではなくWagby内部の任意の画面を開く 1869
同じモデルのメニューボタンを異なるタブに重複して配置する
定義方法 1870
例 1872
スマートフォン
スマートフォンの利用
例 1873
定義方法 1899
テスト方法 1900
仕様・制約 1900
iPad利用時のデフォルトをPC画面とする 1901
スマートフォン利用時のデフォルトをPC画面とする 1901
変更履歴 1902
メッセージの変更・国際化
リソースファイルの編集
リソースファイル 1902
リソースファイルの内容 1903
ファイル置換方式（非推奨） 1903
差分ファイル作成方式（推奨） 1904
例 モデルごとにコントローラ処理の画面表示メッセージを変更する 1904
Dojotoolkitに含まれるリソース 1904
リアルタイムエラーチェックのリソースファイル 1907
トラブルシューティング 1907
国際化
リソースファイル 1907
英語リソースファイルの編集 1908
差分ファイルの作成 1908
休日ファイルの変更 1909
ユーザ定義入力チェックメッセージ 1910
言語パック（英語） 1910
日・英以外の国用のリソースファイルの作成 1910
仕様・制約 1912
言語の選択
ログオン画面による指定 1912
プレファレンスによる指定 1913
選択できる言語を追加する 1914
選択肢モデルの特別表記
表記方法 1915
例 1915
環境
プロジェクト
プロジェクト情報 1917
フルビルド 1918
データベース
データベースの設定 1918
メインデータベースとサブデータベース 1918
データベース接続 1918
データベース環境 1919
対応する設定ファイル 1921
注意事項 1921

コネクションプーリング 1921
関連するページ 1921
サーバ
サーバの設定 1921
Tomcat 1922
ビルド 1923
キャッシュ 1923
HTTPセッション 1924
認証 1924
クラスタリング 1925
関連するページ 1925
メール
メールの設定 1925
メール設定(共通) 1925
メール設定(送信) 1925
アプリケーション
アプリケーションの設定 1926
初期パラメータ 1926
アカウントパスワード 1928
全文検索 1928
メッセージキュー 1928
パスワードリマインダ 1928
デフォルトログオン 1928
カスタマイズ
カスタマイズの設定 1928
詳細 1928
プレファレンス 1930
キーカスタマイズ 1930
プレファレンス
概要 1930
CSSテーマ 1931
表示文字の大きさの指定 1934
メニューアイコン 1935
ログオフアイコン 1937
サブメニュー 1937
パンくず 1939
ログオンユーザ 1939
パスワードリマインダ 1939
カレンダの週の開始曜日 1939
メッセージ、ボタンアイコン
メッセージの設定 1939
ボタンアイコンの設定 (8.3.0) 1940
ファンクションキーを含む、キーイベント動作を変更する (JavaScript)
設定方法 1940
Alt,Ctrl,Shiftキーの判定 1942
特定のモデルの場合のみ有効とする 1942
保存ボタンを割り当てる 1943
メインメニューを開く 1943
メインメニューのサービスタブを開く 1944
ツリー形式のサブメニューを開く 1944
ポータル画面を開く 1945
モデル1の新規登録画面を開く 1945
モデル1の検索画面を開く 1945
モデル1のダウンロード画面を開く 1946
マルチセッションで別画面を開く 1946
すべてのキーイベントを一つのファイルにまとめる 1946

仕様・制約 1948
更新履歴 1948
ログオフ 1948
対応するデータベースを追加する(SQL)
設定ファイルdatabase.txt 1948
同梱されている設定 1953
同梱されている設定を上書きする 1961
ジャスミンソフトが提供するクラスファイルについて 1961
クラスタリング環境で運用する
クラスタリング環境とは 1962
設定方法 1963
動作の確認 1964
クラスタリング環境での運用 1965
ロックの扱い 1966
ジョブスケジュールの制御 1966
仕様・制約 1966
オートスケール環境で運用する
オートスケール環境とは 1966
設定方法 1967
動作の確認 1969
REST API サーバとして利用する 1969
ロックの扱い 1969
ジョブスケジュールの制御 1969
システムログ閲覧 1970
仕様・制約 1970
よくある質問と回答 1971
Designerの機能・その他のツール
ビルド処理の詳細
ビルド対象モデルを指定する 1971
コマンドラインからビルドを行う 1972
リモートのDesignerのリポジトリを転送する (8.3.4) (8.4.0) 1977
ビルドタブを非表示にする (8.4.0) 1979
フィルタ機能
操作例 1979
仕様・制約 1980
モデル項目の一括設定
基本的な操作 1981
全項目の一括設定 1984
選択形式の場合 1985
リポジトリキーを直接指定する 1987
関連:項目ルール 1987
プロジェクト・モデル・モデル項目の説明文を用意する
プロジェクトの説明 1987
モデルの説明 1988
モデル項目の説明 1989
リポジトリのバックアップとリストア
リストア（リポジトリの復元） 1990
手動バックアップ 1991
自動バックアップ 1991
リポジトリをファイルとして取得する 1992
リポジトリファイルを読み込む 1992
操作例 他者が作成したリポジトリを読み込む 1992
トラブルシューティング 1995
技術情報 1995
リポジトリの再読み込み

リポジトリの再読み込み方法 1995
再読み込みが必要なケース 1996
関連図の表示
関連図 1996
表示ルール 1998
モデル、モデル項目のテンプレートを用意する
項目テンプレートの設定 1998
項目テンプレートを使う 1999
モデルのテンプレートを自作する 2000
関連する機能 2002
設計書の出力
設計書出力機能とは 2002
変更履歴 2004
使い方 2005
自動生成される設計書の内容 2006
ER図,画面遷移図,スクリプト一覧表 (8.3.0) 2007
テンプレートファイルを修正する 2009
スクリプトを使ったカスタマイズ 2010
ファイル名に付与される連番の桁数を変更する (8.4.0) 2012
サンプルのダウンロード 2012
トラブルシューティング 2012
仕様・制約 2013
本番運用サーバ転送用のzipファイルの作成
createWagbyappZipコマンド 2013
使い方 2013
仕様 2016
既存テーブル定義からリポジトリを生成する
概要 2017
使い方 2017
指定できるパラメータ 2019
Microsoft Excel の場合 2020
Microsoft Access の場合 2020
仕様 2021
手動で修正すること 2021
謝辞 2021
変更履歴 2021
kintoneの設計情報からリポジトリを生成する
概要 2021
モデルID.csvファイル 2022
「型」について 2022
選択肢モデルの扱い 2023
サブテーブルの扱い 2023
使い方 2024
手動で修正すること 2025
ファイル名の扱いを「オリジナルファイル名を残す」方式に変更する
概要 2025
準備 2025
操作方法 2025
仕様・制約 2026
Designerにログオンできるアカウントの追加・変更・削除
アカウントの扱いについて 2026
新しいアカウントの登録 2027
アカウントの変更 2028
アカウントの削除 2028
Unlimited Enterprise ではない版でのアカウントとパスワードの変更方法 2029
パスワードのハッシュ値を取得する

PasswordEncoderによるカスタマイズを行う場合 2029
PasswordEncoderによるカスタマイズを行わない場合 2029
ハッシュ値について 2033
関連するページ 2033
統計情報
実行方法 2034
関連するページ 2035
影響分析
実行方法 2035
仕様・制約 2036
リポジトリ構造とチーム開発
モデル項目の詳細
項目の並び順 2036
メニュー
.menuと.menuorderフォルダ 2037
チーム開発時の注意点 2037
Appendix
DDL
DDLの作成ルール 2037
DDLの場所 2037
制約に関する命名 2038
インデックス 2038
文字エンコード 2038
SQL文における識別子の区切り文字
区切り文字一覧 2038
SQL の例 2038
区切り文字を設定しない方法とその影響 2039
予約語と利用できる文字
Javaの予約語 2039
その他の予約語 2040
Windows OS で利用できない名前 2041
利用できる文字の制限 2041
クラス名の重複を回避する仕組み 2041
システムモデルとユーザモデル
システムモデル一覧 2041
システムモデルの制約 2044
その他の情報 2045
複合キーの内部的な表現
テーブル定義 2045
Webフォームのパラメータ 2046
ロック処理の詳細
ロックの実装方法 2047
(悲観)ロックテーブル 2047
ロックキーのカスタマイズ 2049
キャッシュ処理の詳細
キャッシュについて 2053
よくある質問と回答 2054
関連するページ 2054
項目の型とデータベースの関係
文字列 2054
数値 2055
テキストエリア 2055
選択肢モデル/リストボックス/ラジオボタン 2056

チェックボックス 2056
繰り返し項目 2057
繰り返しコンテナ 2057
郵便番号辞書を手動で用意する
手順 2058
住所正規化コンバータ(R6)との連携方法
Tomcat の設定 2061
動作テスト 2067
ファイル型の詳細
内部の構造 2069
オリジナルファイル名をそのまま使う 2070
ファイルの実体の扱い 2071
保存先フォルダの変更 2071
MIMEタイプ (8.3.1) 2072
一意制約の詳細
データベースによる挙動の違い 2072
生成されるDDL 2073
一意制約名の自動調整 2073
仕様・制約 2073
入力フィルタの詳細
標準のフィルタルールを変更する 2073
仕様 2074
全文検索のインデックス
格納場所 2074
変更されるタイミング 2074
ファイルサイズ 2074
生成ルール 2075
最適化処理 2075
手動によるインデックスファイルの再作成 2075
組織図の表示
表示できるグループ数の上限 (8.1.2) 2076
選択肢の無効化マーク
選択肢の無効化マークを変更する 2076
プリンシパルの詳細
プリンシパルを追加したときに行うこと 2077
プリンシパルの削除または名前の変更 2078
プリンシパルモデルの詳細 2078
外部システムから直接、Wagby の任意の画面を開く
URLを指定する 2079
スマートフォンからの操作 2080
JSPおよびエラー画面のカスタマイズ
JSPのカスタマイズ 2080
エラー画面(JSP)のカスタマイズ 2080
ビルド時のメモリサイズの指定
設定方法 2081
ログコード一覧
ログコード 2081
変更履歴 2090
パフォーマンスチューニング
パフォーマンスチューニング
大量の選択肢を持つリストボックス・ラジオボタンをつくらない 2090
リストボックスの絞込設定の方式を見直す 2091
被参照モデルのサブモデルを作成する 2092

ビューを使う場合でもキャッシュを有効にする..... 2093
前後のデータにアクセスするボタンの動作..... 2093
SQL式の見直し..... 2093
集合処理関数の見直し..... 2094
スクリプトのリロード方式の変更 (8.2.2)..... 2094
アプリケーションの起動時間を短縮する (8.3.2)..... 2094

モデル

モデル

モデルの操作

WagbyDesignerを使ったモデルの登録、削除方法を説明します。

モデルを作成する

WagbyDesignerのメニューから「モデル」を選択します。「ユーザ」と「システム」の二つのタブがありますが、新規開発するモデルは「ユーザ」タブに格納します。

図に示すように、ギアアイコンから「新規モデル」を選択します。図のように新規モデルの定義画面が開きます。

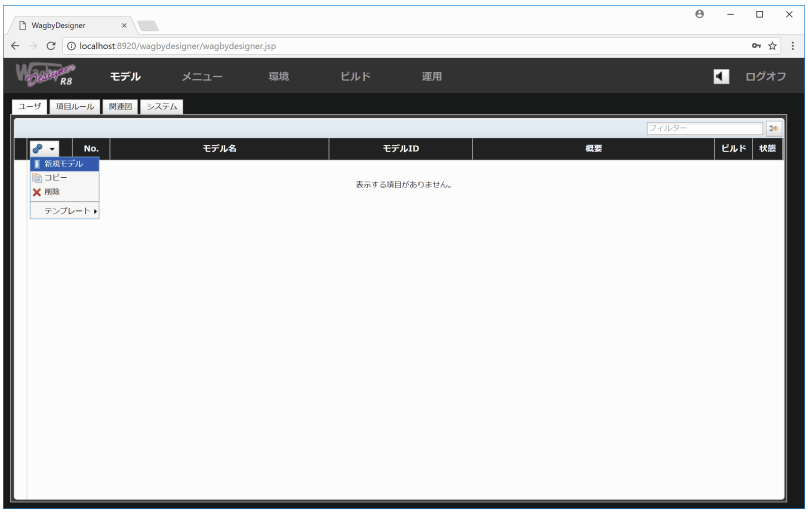


図 新規モデルを定義する

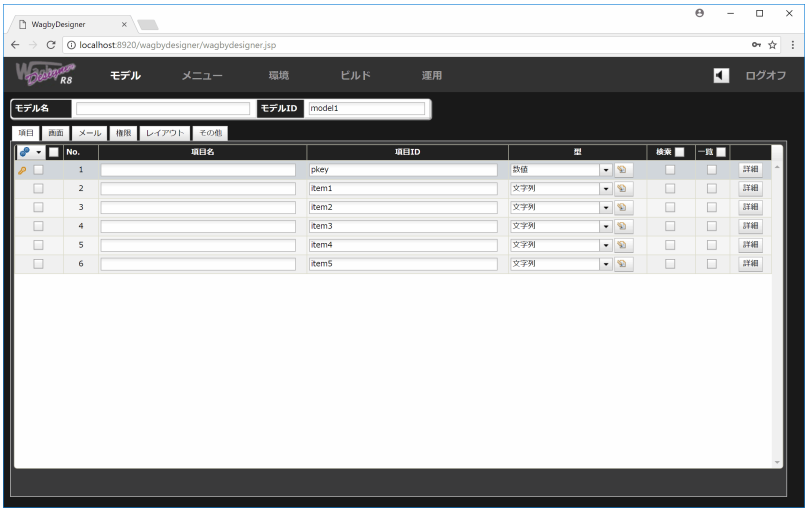


図 新規モデルの定義画面

モデル名の変更

「モデル名」は、画面（およびメニュー）に表示される名称です。 「モデルID」は、データベースのテーブル名および生成されるソースコード内の識別子として用いられます。 モデルIDに日本語（ひらがな、カタカナ、漢数字）を使うことはできません。英数字を使ってください。

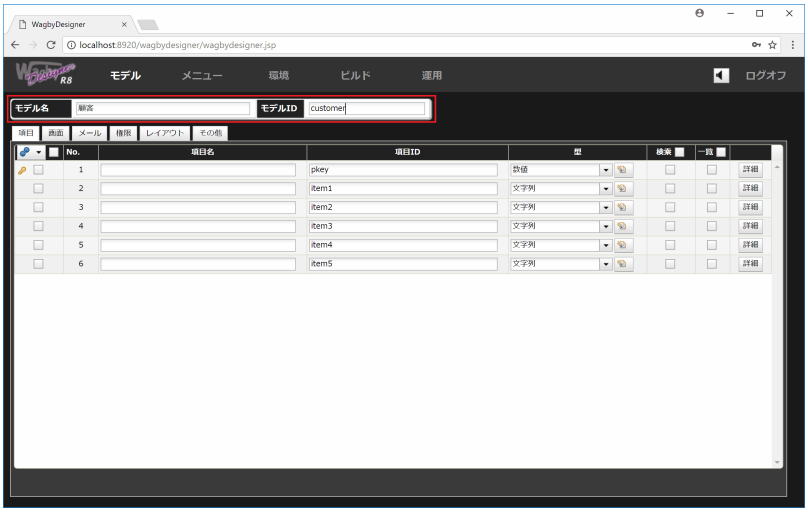


図 モデル名とモデルIDの設定

モデル名やモデルIDの変更は即座にリポジトリに反映されます。ビルドを行うタイミングで、変更された名称を使ってソースコードが再生成されます。

モデルの削除

「モデル」メニューからモデル一覧を表示します。削除したいモデルをチェックします。

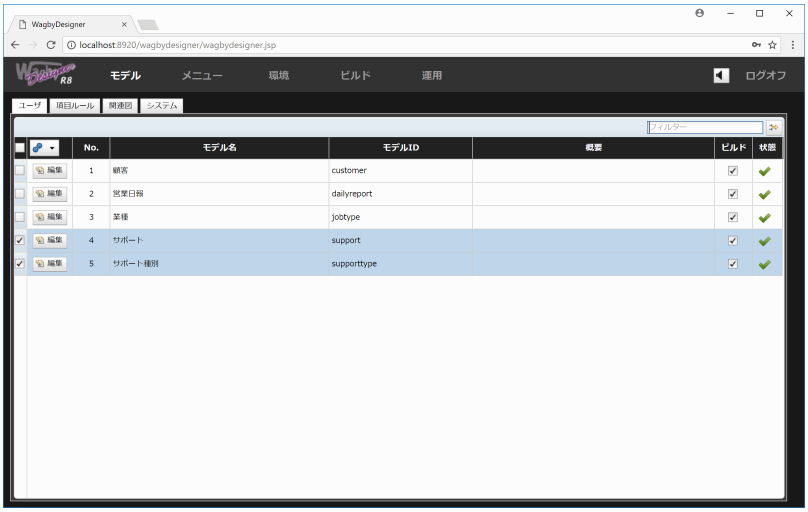


図 削除したいモデルをチェックする

ギアアイコンから「削除」を選びます。

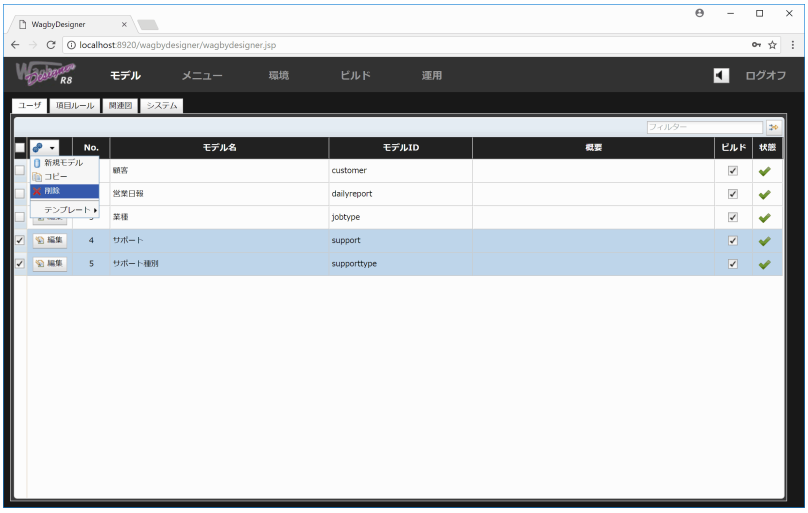


図 削除処理を選択する

確認ダイアログが表示されますので「OK」ボタンを押下します。

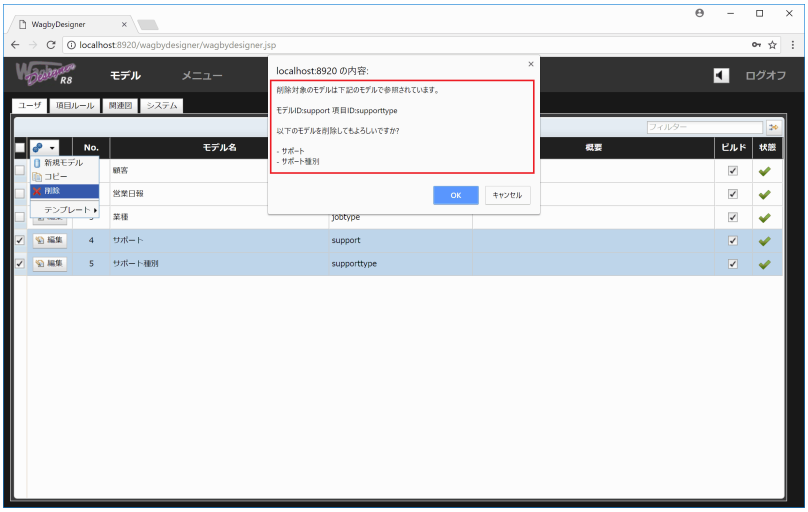


図 削除の確認ダイアログ

モデルが削除されました。

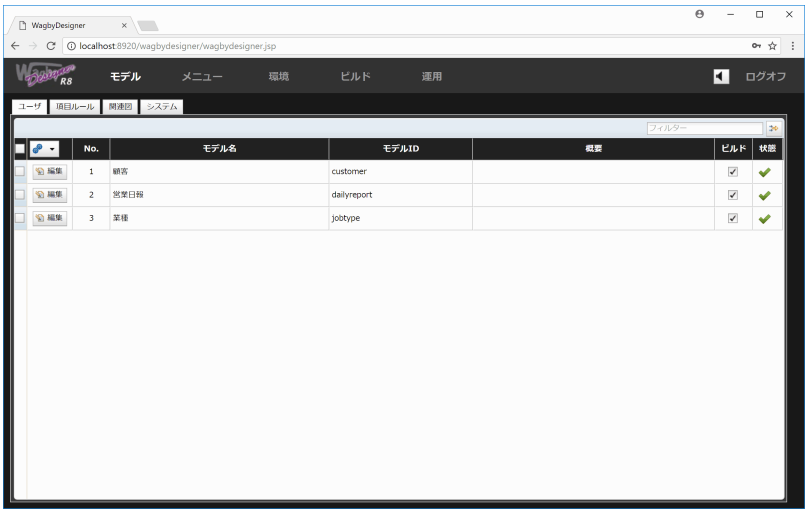


図 モデルが削除された

リポジトリのリストア機能を使うことで、モデルを削除する前のリポジトリに戻すことができます。

ビルド対象の指定

ビルド対象モデルを個別に指定することができます。「ビルド」列のチェックボックスを指定してください。標準はすべてのモデルがビルド対象となっています。

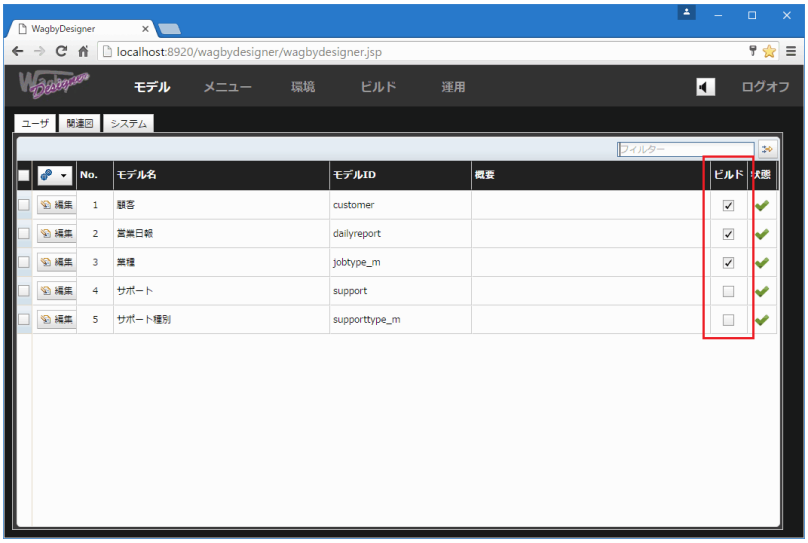


図 ビルド対象モデルを指定する

開発者はモデルの依存関係を考慮して指定してください。ビルド対象となっていないモデルを参照している場合、ビルドエラーになります。

この指定は設計書出力ツールにも反映されます。チェックされたモデルだけが設計書として出力されます。

モデルとテーブルの関係

モデルに対応した（リレーショナルデータベース用の）テーブルが生成されます。

より正確には、モデル定義に対してデータベース用のDDL（create table, drop table）が自動生成されます。このDDLを使って、データベース上にテーブルを作成・削除することができます。DDLの詳細...

モデルIDの命名規則

- 1 文字のモデルIDを用いることはできません。モデルIDは 2 文字以上として下さい。
- 先頭は英文字または「_(アンダースコア)」を指定して下さい。
- 二文字目以降は英文字または「_(アンダースコア)」、半角数字を指定して下さい。
- 「_(アンダースコア)」と半角数字のみで構成することはできません。
- XMLの名前として使用できない文字は指定できません。よって、半角カタカナや全角数字、全角英字、記号は使用できません。（XMLの名前として使用できる文字についてはAppendix 使用できる文字をお読み下さい。）
- 予約語を指定することはできません。（予約語についてはAppendix 予約語をお読み下さい。）
- すべての定義されているモデルIDはアルファベット大文字小文字を区別せず、アンダースコアを無視した状態で重複しないように指定して下さい。(1)
- モデルIDの末尾に「P」「C」「CP」「L」「LP」「T」「TP」「ULP」を付加したモデルIDを別に作成することはできません。(2)

(1) 例えば「ab」「Ab」「aB」「AB」「a_b」「A_b」「a_B」「A_B」を指定した場合、これらはすべて重複しているとみなされ、エラーになります。

(2) 例えば「ab」というモデルIDを用意した場合、別モデルIDとして「abp」「abc」「abcp」「abl」「ablp」「abt」「abtp」「abulp」を定義することはできません。

- 「システム」タブで用意されているモデルIDを指定することはできません。
- 接頭語が ″jfc″ ではじまるモデルIDはシステムが将来、利用するために予約されています。指定することはできません。
- Wagby が内部で使用している以下のモデルIDを指定することはできません。
  - hour_m
  - minute_m
  - movestep_m
  - jpermission_m
  - jprincipal
  - service
  - master
  - 末尾が「_p」「_c」「_cp」「_l」「_lp」「_ulp」で終わるモデル名

モデルの変更と削除

モデルIDの変更

このモデルを参照している他のモデルがあった場合、変更の前に警告ダイアログが表示されます。OKを押下すると、参照しているすべてのモデルに対して、変更後のモデルIDに置換されます。（計算式内の項目も同様です。）

モデルの削除

このモデルを参照している他のモデルがあった場合、削除の前に警告ダイアログが表示されます。このモデル（内の項目）を計算式内で参照している場合も警告対象に含まれます。

OKを押下するとモデルは削除されますが、影響があると示された他のモデルについては、開発者が手動で適切に修正してください。

システムモデルとユーザモデル

「システム」タブに含まれているモデルは、Wagbyの管理機能で用いるものです。これを「システムモデル」と呼びます。これに対して、開発者が作成したものを「ユーザモデル」と呼びます。[システムモデルの詳細...]

定義できるモデル数の上限

理論上の制限はありません。 WagbyDesignerが利用するメモリを増やすことで、多くのモデルを含むアプリケーションをビルドすることができます。 またWagbyでビルドしたアプリケーションが利用するメモリを増やすことで、大規模なアプリケーションを運用することができます。

リポジトリフォルダとの関係

- repository/trunk フォルダ直下に、モデルID名のフォルダが生成されます。
- 上記フォルダ内に「モデルID.txt」が生成されます。

モデル項目

モデル項目の操作

モデルには複数の「モデル項目」を定義することができます。

項目の詳細定義

該当するモデル項目の「詳細」ボタンを押すと、モデル項目詳細ダイアログが表示されます。

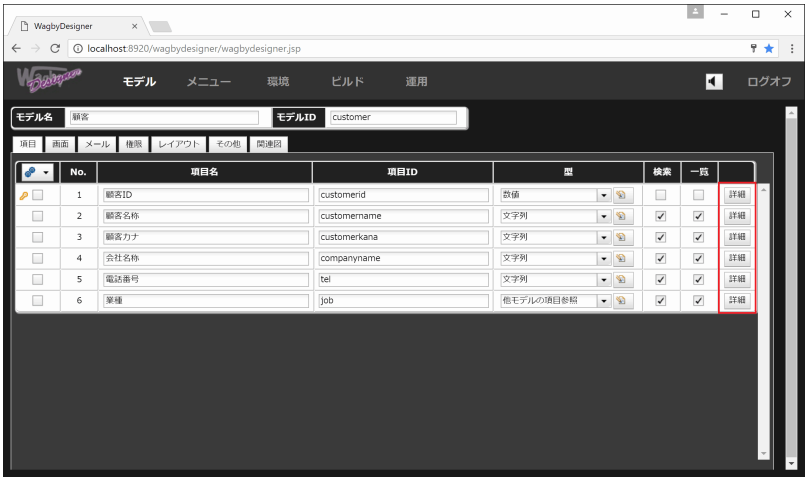


図 詳細ボタン

図 モデル項目詳細を設定するダイアログ

どのような設定が行えるかは、このマニュアル全体に渡って説明されています。これらをお読みいただくことで Wagby でできることがわかるようになっています。

図で、対象項目の行をダブルクリックすることでも図のダイアログを開くことができます。

項目の挿入

挿入先の項目を選択（チェック）し、ギアアイコン内のメニュー「新規」を選びます。

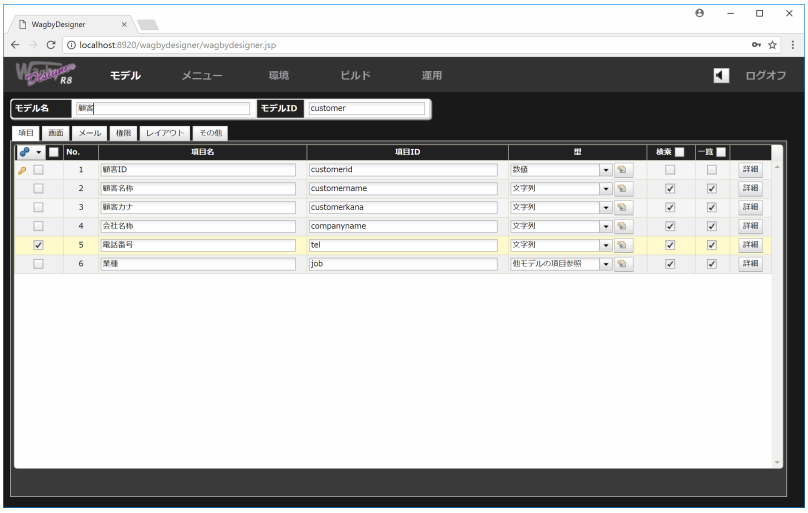


図 挿入先の項目を選択

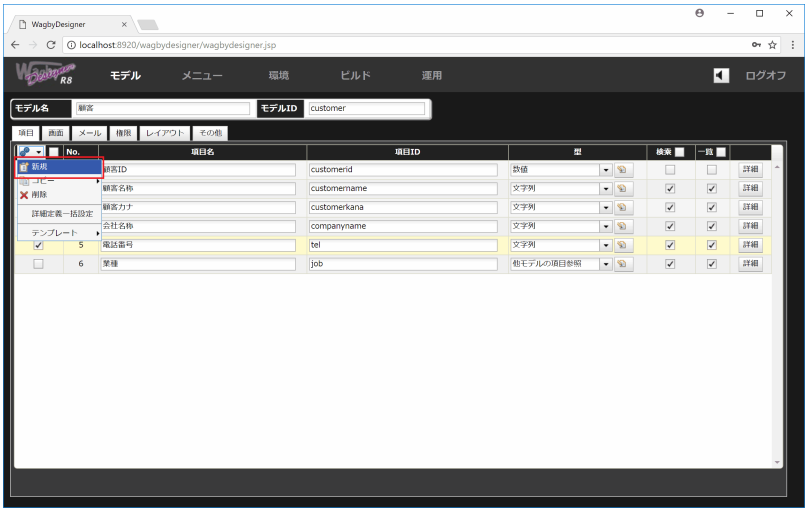


図 新規ボタン

空白行が挿入されます。

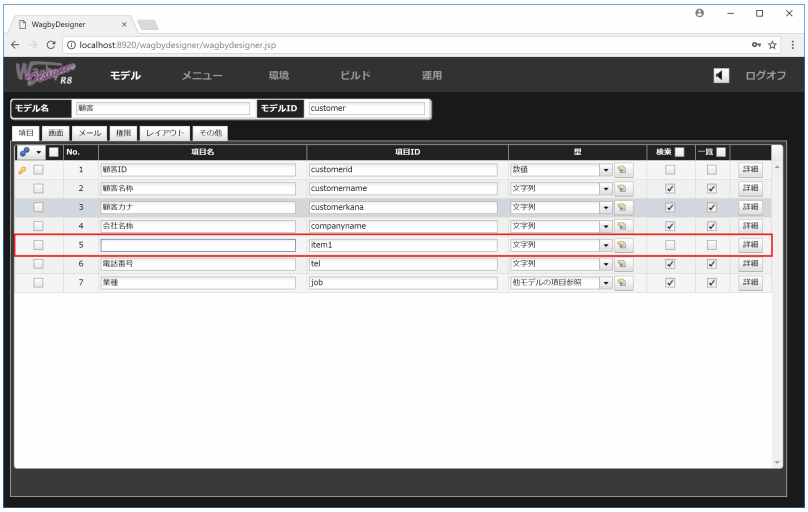


図 空白行の挿入

項目の移動

移動したい項目をマウスでドラッグし、挿入したい位置でドロップします。

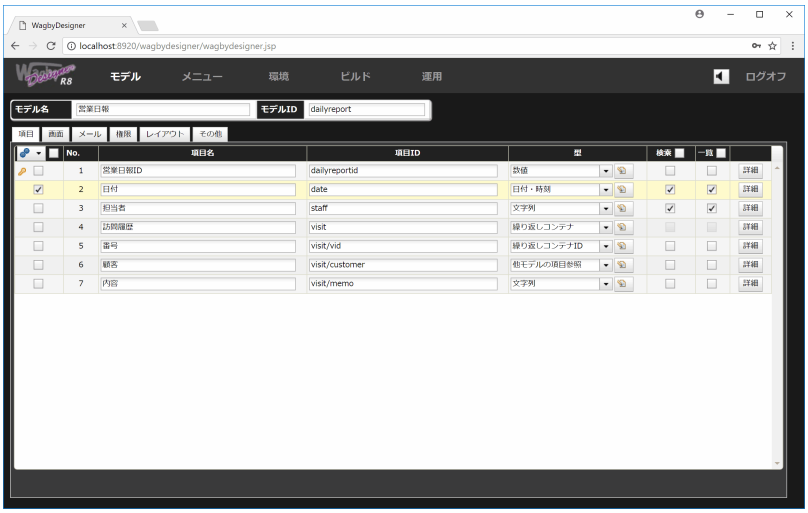


図 項目の移動（ドラッグ）

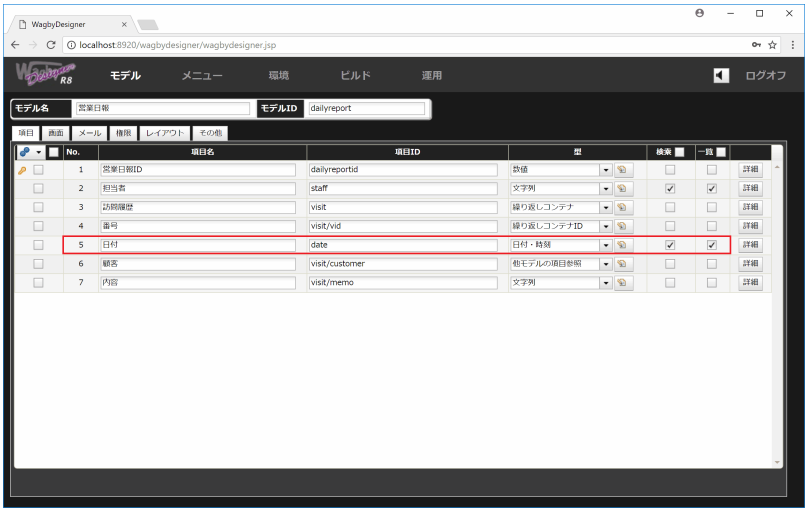


図 項目の移動（ドロップ）

項目の移動がうまく操作できない場合、リポジトリの再読み込みをお試しください。

項目の削除

削除対象行にチェックを入れ、ギアアイコンから「削除」を選択します。

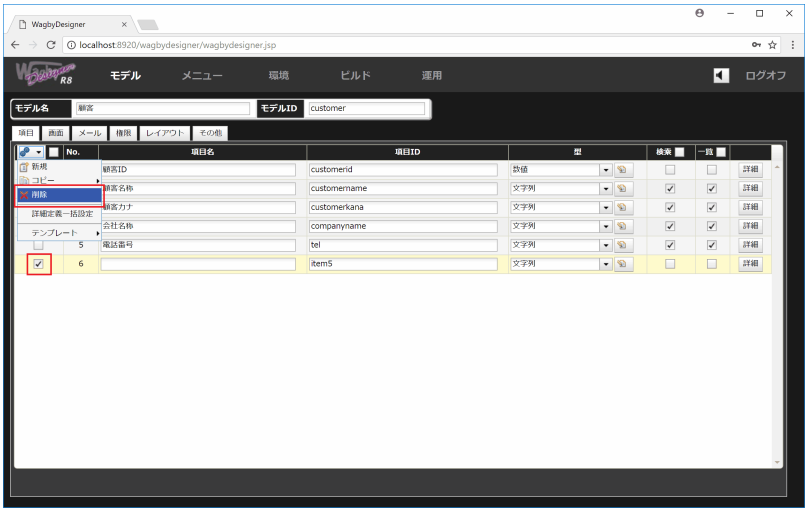


図 項目の削除

項目名の命名規則

「*(アスタリスク)」「>」「'」「¥」「<」「>」「&」を使うことはできません。

項目IDの命名規則

- 1 文字の項目IDを用いることはできません。項目IDは 2 文字以上として下さい。
- 先頭は英文字または「_(アンダースコア)」を指定して下さい。
- 二文字目以降は英文字または「_(アンダースコア)」、半角数字、「/(スラッシュ)」を指定してください。「/」は繰り返し項目のコンテナの区切り文字として用いられます。
- 「_(アンダースコア)」と半角数字のみで構成することはできません。
- XMLの名前として使用できない文字は指定できません。よって、半角カタカナや全角数字、全角英字、記号は使用できません。（XMLの名前として使用できる文字についてはAppendix 使用できる文字をお読み下さい。）
- 予約語を指定することはできません。（予約語についてはAppendix 予約語をお読み下さい。）
- 同一モデル内に定義されている項目IDはアルファベット大文字小文字を区別せず、アンダースコアを無視した状態で重複しないように指定して下さい。(3)
- 繰り返し項目のコンテナ（後述）の区切り文字「/」で区切られた子項目名も上記の条件で重複しないように指定して下さい。

項目の変更と削除

項目IDの変更

この項目を参照している他のモデルがあった場合、変更の前に警告ダイアログが表示されます。OKを押下すると、参照しているすべてのモデルに対して、変更後の項目IDに置換されます。（計算式内の項目も同様です。）

項目の削除

この項目を参照している他のモデルがあった場合、削除の前に警告ダイアログが表示されます。この項目を計算式内で参照している場合も警告対象に含まれます。

OKを押下すると項目は削除されますが、影響があると示された他のモデルについては、開発者が手動で適切に修正してください。

(3) 例えば「ab」「Ab」「aB」「AB」「a_b」「A_b」「a_B」「A_B」「aaa/ab」を指定した場合、これらはすべて重複することとなります。（最後の例は、区切り文字「/」の右側部分「ab」が重複しています。）

定義できるモデル項目数の上限

1つのモデルに含まれるモデル項目数に理論上の制限はありません。

ただし実際には、おおよその目安は300項目前後となります。この理由は「Javaは1メソッドのコンパイル結果のクラスファイルが64KBまで」という制限によるものです。　具体的には、画面表示を行うJSP部で、この制限にかかることがあります。（JSPは、1つのファイルが1つのメソッドに変換される仕様です。）

型

Wagbyで利用できる型

型	属性	説明	Javaの型
文字列	理論上の長さの制限なし。利用するデータベースに従う。		java.lang.String
	郵便番号	住所項目を指定することができる。	（同上）
	メールアドレス	画面表示時にメールアドレスのリンク(mailto:) を出力する。	（同上）
	URL	画面表示時に URL のリンク (http://...) を出力する。指定先が画像の場合は、URL文字列だけでなくリンク先の画像も表示する。日本語のリンク表記にも対応。	（同上）
	ファイル名	画像やドキュメントなどのファイルを扱うことができる。	（同上）
数値	整数	-2,147,483,648 ～ 2,147,483,647	int
	1バイト整数	-128 ～ 128	byte
	2バイト整数	-32,768 ～ 32,767	short
	4バイト整数	「整数」に同じ	
	8バイト整数	内部で扱える範囲は-9,223,372,036,854,7～9,223,372,036,854,775,入力欄は-9,000,000,000,000,0～9,000,000,000,000,000	long
	4バイト浮動小数	小数点以下およそ6桁の精度［非推奨:後述］	float
	8バイト浮動小数	小数点以下およそ15桁の精度	double
日付・時刻	日付	年月日を扱う。年は1753年～9999年の範囲で指定できる。	java.sql.Date
	時刻	時刻を扱う。	java.sql.Time
	日付時刻	年月日と時刻を扱う。年は1753年～9999年の範囲で指定できる。	java.sql.Timestamp

他モデルの項目参照	リストボックス	参照項目をリストボックス形式で表示する。検索時は「未選択」項目を用意できる。	参照先の主キー項目の型に準拠する。
	ラジオボタン	参照項目をラジオボタン形式で表示する。クリアボタンを用意できる。	（同上）
	チェックボックス	参照項目をチェックボックス形式で表示する。全選択とクリアボタンを用意できる。	（同上）
	検索画面	参照項目を別画面検索で指定できる。参照先項目が文字列型の場合はサジェスト入力にも対応している。	（同上）
	参照連動	上記4パターンのいずれかで参照先モデルを確定すると、それに連動する。	参照先で指定した項目の型に準拠する。
繰り返しコンテナ	明細を格納する。		コンテナ名のクラスが用意される。
繰り返しコンテナID	繰り返しコンテナ内の先頭に必要となる。コンテナの番号を管理する。		int

データベースの型との関係

Wagby が提供している項目型は、実際には利用するデータベース製品が提供する適切な型へ自動的に変換されます。例えば文字列型の場合、データベースに Oracle 9i/10g/11g を使うと varchar2(255) となります。この詳細は変換ルールは、各データベース毎の説明をお読み下さい。

浮動小数について

浮動小数の選択

特別な理由がない限り「8バイト浮動小数」を選択してください。コンピュータによる小数の演算には誤差が含まれるため、4バイト浮動小数より8バイト浮動小数の方が、より誤差の影響を最小限に抑えることができます。

小数点を含む巨大な数を扱う場合

8バイト浮動小数の仮数部は10進数15桁までとなります。仮に小数点に3桁使うと整数部は12桁(≒1兆円)までが表現可能な範囲となります。

型の変更に関する注意点

項目の型を変更した場合、以下の点にご注意ください。

その項目の設定はリセットされる

例えば、すでに読み込み専用設定を行っていた項目について型を変更すると、その（読み込み専用）設定は解除されます。原則として、型の変更は生成されるコードへの影響が大きいため、いくつかの設定は初期状態に戻ります。(*1)

自動計算式などへの影響

自動計算式を設定していた場合、型の変更によって式そのものを修正する必要が発生する可能性があります。

運用中のアプリケーションの変更

運用中のアプリケーションで項目の型を変更すると、過去にエクスポートしたデータをインポートできなくなる可能性があります。この場合は手動でデータそのものを変更するなどの対応が必要となります。

一般に、システム設計における「項目の型の変更」は全体の完成が近づくほど、大きな手戻りを引き起こすことが知られています。型の変更は慎重に行ってください。

(4)

主キーの指定

主キーの指定方法

主キーとして用いる項目の右側にある「詳細」ボタンを押下します。

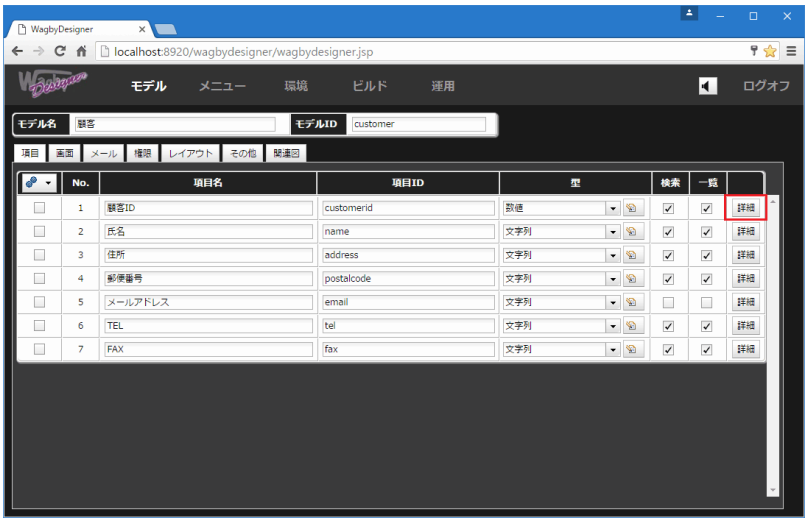


図 詳細定義へ

「主キーとして利用する」にチェックが入っている場合、この項目は主キーになります。

(4) 1. 「暗黙条件」「初期値」「日付・数値フォーマット」「入力フィルタ」「主キーの自動採番」「外部キー」「絞り込み」「入力可・不可制御」「必須指定」「禁止文字設定」「文字数チェック設定」「同一項目値チェック設定」「範囲チェック設定」「正規表現チェック設定」「他項目比較チェック設定」「汎用チェック式設定」「繰り返し」「自動計算」「ユニーク制約」「ロック名」「トランザクション指定」「SQL式」「モデル参照」「住所と郵便番号の連携」

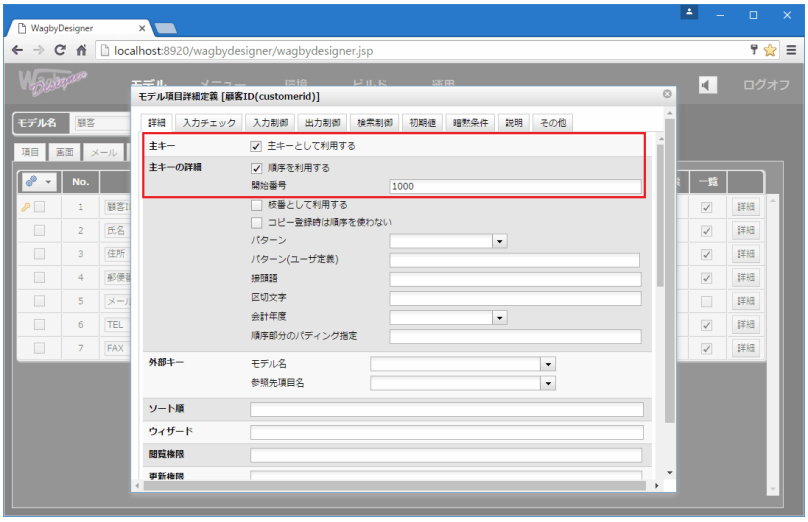


図 主キーの設定

このウィンドウの下部に用意されている「OK」ボタンを押下して（ウィンドウを）閉じます。

主キーアイコン

主キー指定された項目は、モデル項目一覧で鍵のアイコンが表示されます。

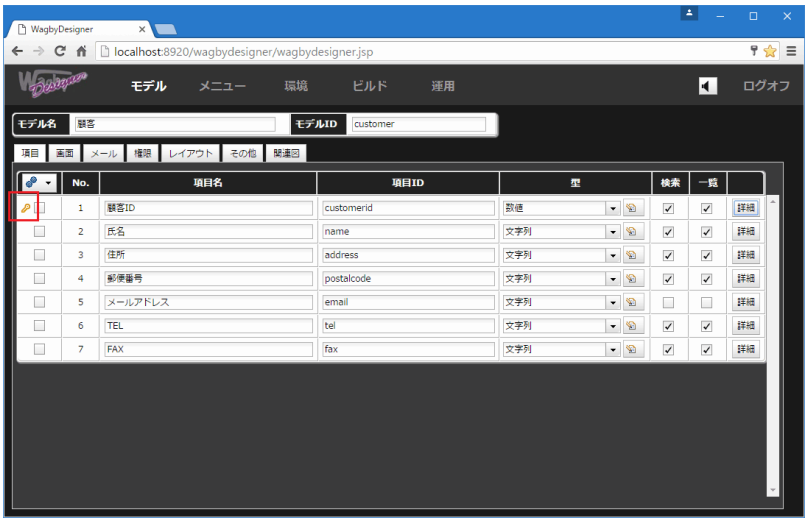


図 鍵アイコンが表示される

単一キー

1つの項目だけでデータが一意に識別できるものを「単一キー」といいます。

人工キー

整数型の項目を主キーに指定した場合、データの新規登録時に既存データの主キーと値が重複しないようにシステム側で連番を振ることができます。これを「人工キー（または代理キー）」と呼びます。

自然キー

データの特性を活かした主キーを「自然キー」と呼びます。社員番号や品番といった業務特性上、重複しない値を主キーとして用いる場合です。

複合キー

複数の項目を組み合わせてデータが一意に識別できるものを「複合キー」といいます。 複数の自然キーを組み合わせる、または自然キーと人工キーを組み合わせるといった方法があります。

複合キーの内部的な表現...

順序

「主キーの詳細」欄で「順序を利用する」にチェックが入っている場合、開始番号からユニークな番号が自動的に割り当てられるようになります。

この「順序を利用する」を無効にすることもできます。この場合、主キーの値は手入力となります。

対応している型

R8.4.0以降

整数（4バイト整数）、8バイト整数、文字列型。

R8.0.0～R8.3.x

整数（4バイト整数）のみ。

業務ルールを加味した順序の設計

接頭語の付与、ゼロ詰め、任意の書式といった主キーの表現を拡張することができます。詳細...

順序の実装方法

Oracle, PostgreSQL, SQL Server 2012以降, DB2

利用するデータベースが「順序」(Sequence) を提供している場合、Wagby はこれを利用します。

上記以外のデータベース

内蔵データベース (HSQLDB) や MySQL など、順序をもたないデータベースを利用する場合は、Wagbyが内部で "seq" というテーブルを用意します。このテーブルで順序値を管理しています。

順序の初期化 !-- N20190409_145452 -->

Wagbyが提供するテーブル初期化コマンド (init_db.bat) で、順序を作成するためのSQL (create sequence) を発行します。順序機能を提供しないデータベースを利用している場合、同じくテーブル初期化コマンドで、順序テーブル（seqテーブル）を作成します。ただし内蔵データベース (HSQLDB) 利用時は、ビルド処理でテーブルおよび順序の作成が行われます。

すでに運用中のアプリケーションがあり、そこで順序を初期化したい場合は次の方法のいずれかを行なってください。

- 現在のテーブルおよび順序を削除するコマンド drop_db.bat を実行し、そののち init_db.bat で初期化する。
- データベースに対して手動で順序(sequence)をdropする。もしくは順序テーブル(seqテーブル)をdropする。そののちテーブル初期化コマンド init_db.bat を実行する。あるいは手動で順序もしくは順序テーブルを作成する。

順序（自動採番）における欠番・飛び番

主キーに人工キー（代理キー）を適用した場合でも、欠番が生じる場合があります。具体的には次のような場合です。

- 何らかの理由で登録処理がデータベース側でエラーとなった。
- そのデータを削除した。

データベースではいくつかの順序をあらかじめメモリに保持しておくことで採番のパフォーマンスを上げるものがありますが、メモリに保持していた番号を消去することがあります。結果として飛び番号が生じます。

データベースの順序を利用しても欠番や飛び番が発生することがあります。詳細は、利用するデータベースの仕様を確認してください。

複合キーの制約

主キーに複合キーを使用したとき、順序に利用できるのはいずれか一つの項目のみとなります。複数の主キー項目で順序を使う設定はできません。

仕様・制約

一度保存された主キーの値を変更することはできません。

これを踏まえて、次の方針で設計することを推奨します。

できるだけ人工キー（代理キー）を使う

主キーに意味をもたせず、整数型かつ順序を用いるようにします。意味をもたせないことで、値の変更しなければならないという状況を発生させません。

一意制約の設定と組み合わせる

業務要件で多いのが、複数の項目の組み合わせが一つしか存在しないことを保証するため、これらを複合キーとすることです。しかしキーの一部にしてしまうと、体系が変わったときなどに（変更できないため）対応が難しくなります。

そこでキー自体は人工キーとし、組み合わせが一つしかないことを保証したい（複数の）項目に一意制約を設定します。これで要件を満たすことができます。

検索条件の指定

定義方法

モデル定義で、項目毎に用意された「検索」欄にチェックを入れると、検索条件として利用できます。

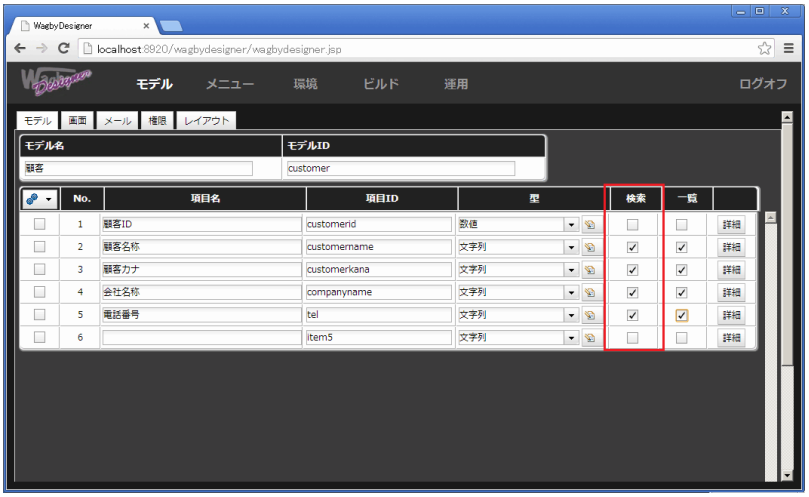


図 検索条件の指定

範囲検索

数値型および日付・時刻型の項目を検索条件に指定すると、自動的に範囲検索が有効になります。「AからBまで」という範囲を指定した検索が行えます。（この範囲検索を解除することもできます。）

詳細は"モデル項目 ＞ 数値 ＞ 検索条件としての利用"をお読みください。

インデックス

検索条件と指定された項目は、データベースのインデックスが作成されます。これによって検索のパフォーマンスは常に最良となるように、データベースは自動的に最適化されます。

ただし多くの項目を検索条件と指定した場合は、更新処理のパフォーマンスが低下します。（更新の都度、インデックスを変更するためです。）検索条件項目として用意するが、あまり検索に用いられることはないという場合は、インデックス作成処理を行わないように設定することで更新処理とのバランスに配慮するようにしてください。DDLの詳細...

表示順を制御する

検索画面の表示順は、図でチェックを入れた項目の並び順となります。

この並び順を変更することができます。詳細は「検索条件の並び順」をお読み下さい。

仕様・制約

- 繰り返しコンテナ内の項目が「他モデルの項目参照 - 参照連動」かつ、参照先項目が「他モデルの項目参照 - チェックボックス」の場合、検索条件に指定するためにはこの項目をデータベースに保存する必要があります。
- 郵便番号項目を検索条件とした場合、対応する住所項目も合わせて検索条件に指定してください。

一覧表示項目の指定

定義方法

モデル定義で、項目毎に用意された「一覧」の欄にチェックを入れると、一覧表示項目として利用できます。

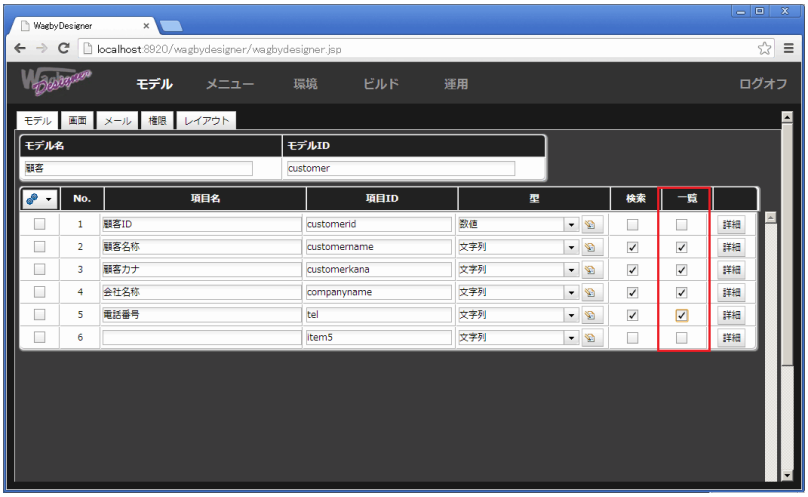


図 一覧表示項目の指定

一つも指定しなかった場合は、一覧表示画面に詳細表示ボタンの列だけが表示されます。

表示順を制御する

一覧表示画面の表示順は、図でチェックを入れた項目の並び順となります。

この並び順を変更することができます。詳細は"一覧表示の並び順"をお読み下さい。

ラベルの幅を設定する

一覧表示の項目数が多いと、ブラウザが自動的に列幅を縮めて表示します。　これを回避するには、"ラベル幅の設定"を行ってください。

ソート

例

図は、顧客モデルの一覧表示画面において「氏名」「氏名かな」「電子メール」の3つの項目をソートキーと指定した例です。

画面表示時は、第一ソートキーで並びます。（この例ではアルファベット降順としています。）

図　ソートキーを指定した一覧表示画面

ソートキーと指定した項目は、項目名部分をクリックできます。クリックするとその項目でソートが行われます。同じ項目をクリックすると、ソート順を反転させることができます。

図は電子メール項目をアルファベット昇順ソートに切り替えた例です。

図 電子メール項目の昇順ソートに切り替えた

ソートキーは記憶されます

ソートキーは画面ごと、かつ、利用者ごとに記憶されます。利用を終了し、再びログオンすると、最後に指定したソートキーで並び替えられた一覧画面が表示されます。

ソートキーを初期化する

検索画面で「検索条件のクリア」を行うと、再検索と同時に初期ソート順に戻ります。

定義方法

サンプルで用いた顧客モデルの定義を図に示します。ここでは「氏名」「氏名かな」「電子メール」の3つの項目にソートキーを設定します。

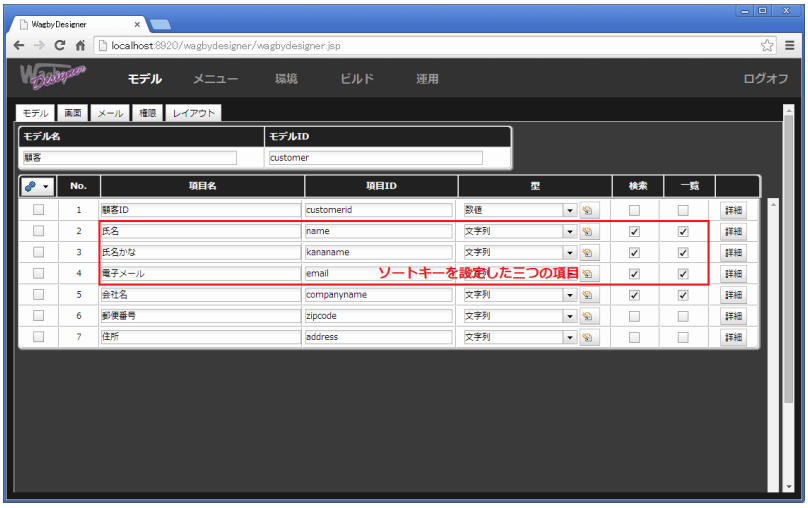


図 顧客モデルの定義

モデル項目詳細定義を開きます。各項目の「ソート順」という欄に数字を指定します。数字の若い順に第一ソートキー、第二ソートキー... と割り当てられます。

数値の前に「マイナス」を指定すると、初期ソートルールが逆順ソートとなります。

図 「電子メール」項目を第一ソートキーかつ逆順とする

図 「氏名かな」項目を第二ソートキーとする

図 「氏名」項目を第三ソートキーとする

複数のソートキーを指定した場合の動作

上の図,5,6のように複数のソートキーを指定した場合、一覧表示画面の初期表示時は第一ソートキーのみが働くわけではなく、第二以降のソートキーもソート順として反映されます。（後述する SQL の ORDER BY 句に、第一ソートキーから順に列挙されます。）

初期設定についての補足

上の図,5,6に示した初期設定の昇順/降順にかかわらず、一覧表示画面の列のクリック操作は一度目のクリックで昇順、二度目は降順となります。

仕様・制約

ソートはデータベースの SQL 処理（ORDER BY 句）によって実現しています。 そのため、データベースによってルールが変わります。

データベースに保存しない項目はソートキーに指定できません

ソートはデータベースの ORDER BY 句によって実現しています。そのため、データベースに保存しない項目はソートキーとして指定することはできません。

一般的なルール

- 数値、日付型は昇順。
- アルファベットについては、A,B,C... 順。
- 日本語文字については、そのデータベースを作成した文字コード順。（文字コードの確認および定義方法についてはデータベースのドキュメントをご確認ください。）

日本語文字でのソートは、利用者が期待した並びにならないことがあります。これは文化的な並びではなく、文字コード順となるためです。

主キーの扱い

主キーは必ずソートキーに含まれます。

(5)

(5) ページネーション機能を実現するために、order by 句に主キーを指定する必要があります。

主キーがソート項目として指定されていなければ、最も最後のソート条件になります。(order by 句の末尾に付与されます。) 主キーがソート項目として指定されていれば、その設定に従います。

NULL 値のソート

値が "NULL" のソートは、データベースによって扱いが異なります。

- SQL Server, MySQL では、最も小さい値として扱われます。
- Oracle, DB2, PostgreSQL では、最も大きい値として扱われます。(Oralce 9i 以降では、NULL の扱いを指定することもできます。)
- 内臓データベース (HSQLDB) では、ソート対象となりません。NULLを含む行が常に先頭になります。

SQL式を使って、NULL ソートのルールを変えることもできます。詳細...

モデル参照項目のソート

モデル参照項目をソート対象とすることは非推奨です。

その理由は、モデル参照項目は "ID値" でソートするためです。多くの場合、表示されている内容と、ソートの並びが一致しないという印象を受けることになります。

モデル参照（参照連動）項目のソート

参照連動項目が自モデル保存となっている必要があります。

外部キー関係における子モデル部の表示

- 本設定は、親モデルの詳細表示画面の下部に用意される子モデル一覧の表示順にも反映されます。
- 項目名をクリックしてソートする機能は無効になります。（指定したソート順で固定されます。）

データベースの詳細

概要

「画面>その他>データベースの詳細」を指定することができます。

図 データベースの詳細

テーブルを作成する

標準では有効となっています。この場合、モデルに対応するテーブルが作成されます。 この設定を無効にすると、テーブルが用意されません。正確には、利用データベースに対して CREATE TABLE 文を発行しません。

次の場合に利用できます。

- 既存システムのテーブルを再利用する。
- テーブルの代わりにビューを用意する。

(6)

(6) テーブルを作成しないと設定した場合でも、テーブル定義(DDL)は用意されます。DDLの詳細...

物理テーブル名

テーブルを作成し、かつ物理テーブル名を指定した場合、ここで指定されたテーブル名で CREATE TABLE 文を実行します。

テーブルを作成しないとし、かつ物理テーブル名を指定した場合、既存システムのテーブルを操作するようになります。

異なるモデルで同じテーブルを共有する

異なるモデル A,B をそれぞれ定義したとき、モデルBで「テーブルを作成しない」とし、かつ「物理テーブル名にモデルA」を指定することで、モデルAとBは同じテーブルを共有することになります。[7]

スキーマは指定できない

物理テーブル名にスキーマを含めることはできません。

代案として、データベースの「シノニム」が使う方法があります。例えば Oracle では、シノニムを使って別スキーマのテーブルを別名のテーブルとして扱える仕組みがあります。詳細はご利用のデータベースのマニュアルをご確認ください。

日本語テーブル名を指定する

物理テーブル名に日本語テーブル名を指定することができます。[Appendix > DDL > 文字エンコード...]

キャッシュを有効にする

Wagbyはデータベースから読み込んだ値をメモリにキャッシュします。これによりデータベース操作回数を減らし、パフォーマンスを向上させています。標準ではキャッシュは有効となっています。

> 本設定を無効にすると、常にデータベースを操作して最新のデータを取得するようになります。パフォーマンスに影響があるため、理由がなければキャッシュは有効とすることを推奨します。

モデルのキャッシュ情報を定期的に消去する

ジョブを使ってモデルのキャッシュ情報を定期的に消去することができます。

1リクエスト間のキャッシュ

キャッシュを無効にした場合でも、1リクエスト間（WebブラウザとWebサーバ間の通信のやりとり）で発生したデータベースとのやりとりはメモリにキャッシュされます。レスポンスを返した段階で、このキャッシュは自動的に消去されます。キャッシュ有効期間は非常に短いですが、若干のパフォーマンス改善につながります。

この「1リクエスト間のキャッシュ」を無効にすることもできます。この場合、常にデータベースから値を読み込むようになります。

定義方法

「画面 > その他 > データベースの詳細」で「キャッシュを有効にする」のチェックをはずしたとき、さらに「Javaソースコードの設定」にある「データベースのキャッシュを完全に無効にする」をチェックします。（標準ではチェックされていないため、1リクエスト間のキャッシュは有効となっています。）

---

[7] 同様の仕組みに「サブモデル」があります。サブモデルを設定すると、Wagbyがメインモデルとサブモデルに関係がある、と解釈できるため、ロックやキャッシュの処理は適切に行われます。一方、サブモデル設定ではなく、単に物理テーブルを共有した場合、そのような対応は行われません。両方のモデルでの整合性の維持は、開発者が注意して行うようにしてください。[詳細...]

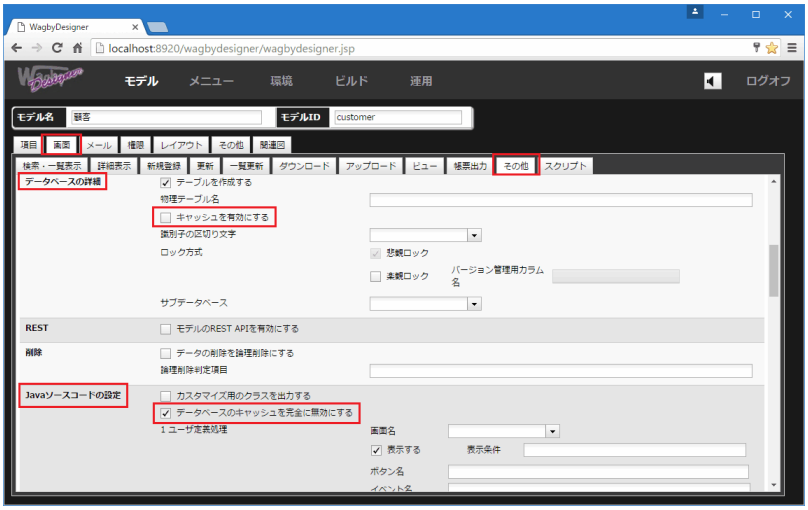


図 データベースのキャッシュを完全に無効にする

ワークフローを有効にしたモデルでは、"データベースのキャッシュを完全に無効にする" をチェックすることはできません。

識別子の区切り文字

Wagby は標準で、SQL文における識別子の区切り文字を設定するようになっています。 詳細は"環境 > データベース > 識別子の区切り文字"をお読みください。

ロック方式

"ロック方式"をお読みください。

既存テーブルを再利用する場合の注意点

既存テーブルを再利用する場合の注意点をまとめました。

- 「テーブルを作成する」を無効とし、既存テーブル（またはビュー）名を「物理テーブル名」に記載します。
- モデル内の項目は、既存テーブル（またはビュー）の項目と同じとなるようにします。物理カラム名や型も指定することができます。
- パフォーマンスのため、Wagbyのキャッシュはできるだけ有効としてください。既存テーブルが更新された場合、Wagbyのキャッシュを定期的に再作成する方法があります。またはマテリアライズドビューを使う方法も有効です。
- 既存テーブルが読み込み専用であれば、ロックについて考慮する必要はありません。既存テーブルをWagbyからも更新し、かつ既存システムからも更新される可能性がある場合は"テーブル操作に関する注意点まとめ > ロック"をお読みください。

トラブルシューティング

必須項目の扱い

Wagbyで定義したモデルで「必須」とした項目で、実際には既存テーブル（またはビュー）側に NULL が存在していないかどうかを確認してください。Wagby は必須項目では NULL は存在しないという前提であるため、NULL が見つかると正しくデータを取得できず、エラーになります。

日付型項目の扱い

既存テーブルで日付を文字列で定義している場合があります。例: yyyyMMdd の 8文字。この場合、Wagby のモデル定義でも文字列とするしかありませんが、日付型の範囲検索やカレンダー補助入力機能が利用できなくなります。回避策として、既存テーブルのビューを定義し、データベースの関数を使って文

字列から日付型へ置き換えることを検討してください。ビューで日付項目となっていれば、Wagbyの日付型を利用できるようになります。

ロック方式

楽観ロックと悲観ロック

Wagbyではモデル単位でロック方式を指定することができます。標準は悲観ロックです。　両者の違いは次のとおりです。

	楽観ロック	悲観ロック（標準）
特徴	同じデータの更新画面を、同時に複数の利用者が開くことができます。最初の更新者のみが書き込みに成功します。それ以外の更新はすべて失敗します。	同じデータの更新画面を開くことができる利用者はただ一人に限定されます。他の利用者は、そのデータを閲覧することはできますが、更新画面を開くことはできません。
書き込みの競合	いったん更新画面をキャンセルし、再び更新画面に入り直す必要があります。修正途中のデータは無効になります。	動作の仕組み上、書き込みの競合は発生しません。最初の利用者が更新を終えるまで、他の利用者は更新処理を待つことになります。
競合時の対応	競合が発生して更新が失敗した場合でも、画面上には入力データが残っているため、利用者の方で復旧処理を工夫することはできます。例えば、入力済みの値をテキストエディタなどに一時コピーし、再び更新処理を行うといったことが考えられます。	更新画面を開いた利用者が、長時間、画面を更新状態にしておくことに一定の時間制限を用意するようにします。これをセッションタイムアウトといいます。Wagbyの標準は20分です。"環境 > アプリケーション > 初期パラメータ"で変更することができます。
ロックの実現方法	モデル内に「バージョン管理用カラム」という項目を用意します。型は8バイト整数です。更新画面を開いたとき、同項目の値を読み込みます。更新処理時に再び値を読み込み、最初に読み込んだ値と変わっていなければ更新を成功させます。	Wagby内部の「ロックマネージャ」を利用します。更新画面を開いた際に、ロックを取得します。更新終了時にロックを解放します。
ロック解放のタイミング	-	・ データを更新する(トランザクション終了時にロック解放) ・ メニュー画面を表示する ・ ログオフする(セッションタイムアウトも含む)

更新画面での悲観ロックの挙動を"画面機能 > 更新"で説明しています。

ロック方式を選択する

「画面>その他>データベースの詳細」から、ロック方式を選択することができます。

図 ロック方式の選択

楽観ロックと悲観ロックは、少なくともいずれか一方を選択する必要があります。標準では悲観ロックです。（悲観ロックにチェックが付いた状態で、かつ入力不可となっています。）

楽観ロックのチェックボックスを有効にすると、悲観ロックのチェックボックスとバージョン管理用カラム名が入力可となります。ここで入力した項目名で、型が「8バイト整数」の項目が自動的に追加されます。

(8)

バージョン管理用カラム名は必須チェックを行ないます。未入力状態でフォーカスがセットされると「この値は必須です」というエラーメッセージが表示されます。

なお「悲観ロック」と「楽観ロック」は両方、有効にすることができます。この場合、Wagbyの利用者視点からは、悲観ロックで動作しますが、更新処理で（楽観ロックの）バージョンカラムも更新されます。そのため、Wagbyと外部のシステム（または自作のバッチ処理プログラム）が、データベースに用意されたバージョン管理用カラムを共用したいという場合に使うことができます。

サブモデル利用時の注意点

ロック方式はメインモデルとサブモデルで一貫性を保つようにしてください。

例えばサブモデルのみ楽観ロックを有効にした場合、メインモデルに存在しないバージョン管理カラムを利用しようとしてエラーとなります。

どのロック方式を選択するべきか

通常は悲観ロックとする

Wagbyの標準は「悲観ロック」です。競合が発生しないため、もっとも問題の少ない方法です。

楽観ロックを選択した方がよい場合

外部システムとの連携を考慮する場合、楽観ロックを用いた方がよい場合があります。

悲観ロックは、Wagbyが内部で用意したロックマネージャを使います。そのため、今どのデータがロックされているかを外部システムが知るためには特別な工夫が必要です。

楽観ロックは、Wagbyと外部のシステムが、データベースに用意されたバージョン管理用カラムを共用できます。いずれのシステムから更新された場合でも、確実に競合を検出することができます。（例えばバッチ処理プログラムを独自に開発した場合、楽観ロックのバージョン管理カラムを参照することで競合の検出が可能です。）

内部での実装方法

Wagbyの楽観ロックと悲観ロックの実装方法をこちらにまとめています。

(8) ここで入力した項目名は「モデル項目」欄には表示されませんが、内部で管理されます。そのため「モデル項目」の欄に、この項目を定義する必要はありません。

(悲観)ロックテーブル

Wagbyの「悲観ロック」は、内部でロックマネージャをもっています。このロックマネージャはメモリ内にロックを保持します。詳細...

ロックキーのカスタマイズ

Wagbyの「悲観ロック」のロックキーをカスタマイズすることができます。トランザクション処理のためのカスタマイズコードを記述するときに対象データのロック管理を細かく制御したい場合に適用できます。詳細...

サブデータベースを指定する

メインデータベースとサブデータベース

Wagbyではモデル単位でどのデータベースを利用するかを指定することができます。一つ目を「メインデータベース」と呼び、二つ目以降を「サブデータベース」と呼びます。

サブデータベースの用途は、既存システムのマスタデータを直接、Wagbyアプリケーションが利用することです。

両者の違いは次のとおりです。

	メインデータベース（標準）	サブデータベース
設定できる数	1つだけです。	オプションという位置付けです。3つまでのサブデータベースを指定できます。
機能	読み込みと書き込みができます。書き込みはトランザクション制御にも対応しています。	読み込みと書き込みができます。(*1) メインデータベースとサブデータベースの両方の更新を一つのトランザクションとすることはできません。それぞれの書き込み処理は独立して行われます。(*2)

(9) (10)

サブデータベース利用時の制約

サブデータベースを使うモデルには次の制約があります。

- テーブルの作成は行えない。（既存テーブルの再利用を想定している）
- 全文検索機能は利用できない。
- ワークフローは利用できない。
- メインモデル、サブモデルは利用できない。（メインモデルとサブモデルで別々のデータベースを指定することはない。）
- 登録、更新、削除などの更新系ボタンは標準で無効になる。（登録や更新処理を行う場合は、手動でこれを再設定すること。）
- システムモデル(juserなど)をサブデータベースに指定することはできない。
- 内蔵データベース(HSQLDB)は利用できない。
- 参照先モデルがサブデータベースに存在する場合、"参照連動項目を参照先モデルに保存する"機能の利用は行わないようにすること。これはトランザクション境界が分かれているため。
- 参照先モデルがサブデータベースに存在する場合、参照連動項目(DB保存なし)を検索項目として指定することはできない。データベースをまたがる検索になるため。
- サブデータベースのテーブルにスキーマを含めることはできない。

(9) 1. R8.1.0よりサブデータベースの書き込みが行えるようになりました。それ以前の Wagby は読み込みのみとなります。

(10) 2. トランザクション境界はデータベースごととなります。そのため、例えばメインデータベースとサブデータベースの両方を更新したいが、サブデータベースの更新が失敗したらメインデータベースの更新も失敗させ、このトランザクション全体をロールバックするという指定を行うことはできません。どちらか一方が成功すれば更新は完了となります。

- サブデータベースへ登録処理を行う場合、順序を利用することはできない。そのため順序の利用は無効にする必要がある。
- 親子モデルのいずれも（同じ）サブデータベースを利用しているとき、その親子モデルの同時更新はトランザクション対象にならない。つまり親と子はそれぞれ独立にコミットされる。 R8.4.2で改善されました。

キャッシュの扱い

サブデータベースを利用しているモデルのキャッシュは標準では有効となっていますが、利用状況に応じてキャッシュを無効にすることを検討して下さい。

キャッシュを有効としても良いケース

サブデータベースで参照しているテーブルのデータが別システムから変更されない、または、データの変更頻度が少ない場合。 データの変更頻度に応じてジョブを使ってキャッシュをクリアすることで、(リアルタイムではありませんが)最新の情報を利用できます。

- ［参考］モデルのキャッシュを消去する

キャッシュを無効とした方が良いケース

リアルタイムでの最新データの表示が必要なモデルではキャッシュは無効として下さい。

定義方法

データベースの設定

「環境>データベース」から、メインデータベースとサブデータベースの指定を行います。

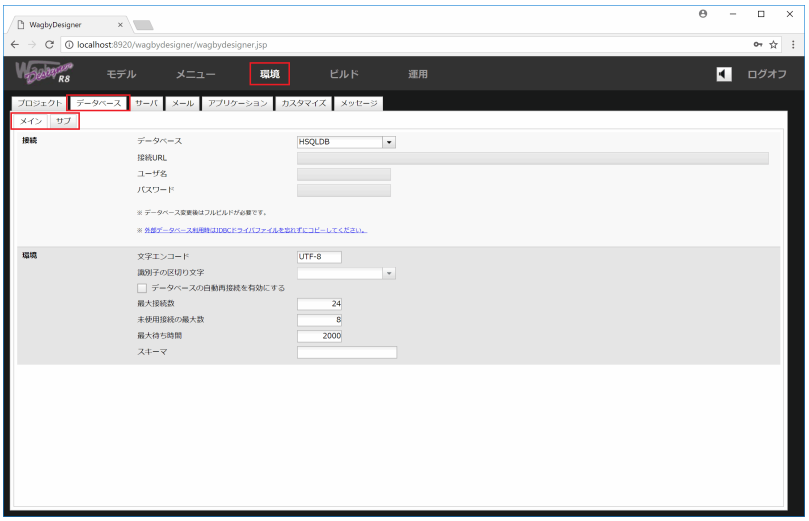


図 メインデータベースの設定（標準）

サブデータベースは最大5つまで指定できます。

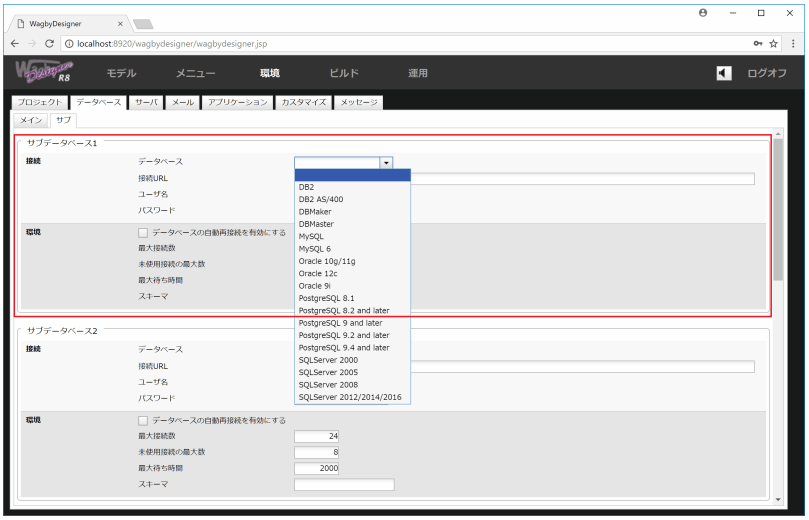


図 サブデータベースの設定。

モデルの設定

モデル毎に、どのデータベースを利用するかを指定できます。設定したサブデータベースのみ選択肢に用意されます。

空白時（標準）はメインデータベースを利用します。

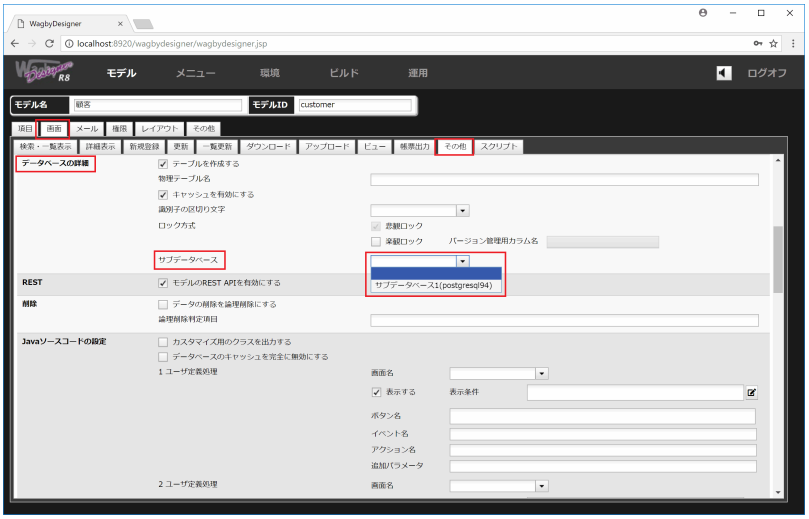


図 データベースを指定する

application.properties のカスタマイズ（8.4.0/8.4.1）

R8.4.0/R8.4.1のWagbyで必要になります。R8.3.xまで、ならびにR8.4.2以降はこの設定は不要です。

開発者は customize/resources/myapplication.properties を用意し、次の行を含めるようにしてください。

```
spring.main.allow-bean-definition-overriding=true
```

サブデータベースの設定を無効にする

サブデータベースの設定を無効にするには、環境タブのサブデータベース設定を消去することに加え、サブデータベースを利用しているモデル側の設定も解除してください。

スクリプトを使ってサブデータベースで順序を利用する

サブデータベースにデータを登録する場合、Wagbyが標準で提供する「順序」の設定は無効にする必要があります。(Wagbyが提供する「順序」はメインデータベースのみを対象にしています。)

しかしスクリプトを使ってサブデータベースの登録時に順序を使うことはできます。

データベースのビューを利用する

ビューの利用

Wagbyで定義したモデルの物理テーブルを作成せず、データベースに用意したビューを参照させるようにすることができます。

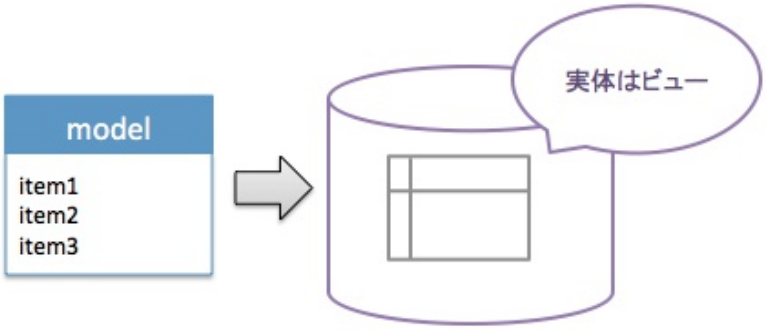


図 ビューを参照させる

定義方法は次のとおりです。

1. テーブルを作成しない設定とします。
2. 物理テーブル名にビュー名を指定します。(モデル名とビュー名が同じ場合は省略可能です。)

主キーの扱い

ビューを利用する場合でも、主キーが必要です。(Wagbyのモデルは主キーの指定が必須です。)

既存データベースのビューに主キーがない、という場合は、追加してください。

必須項目の扱い

Wagbyで定義したモデルで「必須」とした項目で、実際にはビュー側に NULL が存在していないかどうかを確認してください。Wagby は必須項目では NULL は存在しないという前提であるため、NULL が見つかると正しくデータを取得できず、エラーになります。

キャッシュの扱い

更新を伴うビューを用いる場合、キャッシュの扱いを考慮する必要があります。 詳細は"テーブル操作に関する注意点まとめ > キャッシュ"をお読みください。

Wagbyのグループ権限をビューに適用する

(利用するデータベースの)ビュー側にカラムを追加することで、Wagbyが提供するグループ権限を利用することができます。詳細...

テーブル操作に関する注意点まとめ

この説明が必要な場合

次のような利用を行う場合に、本文書の内容を参考にしてください。

- 既存のテーブルを再利用する。（別のシステムからもこのテーブルを利用している。）
- 運用中に(Wagbyを経由せずに)直接、テーブルのデータの変更(登録/更新/削除)を行う。
- 別々のモデルで、同じ物理テーブルを共有する。（メインモデルとサブモデルの関係であれば、ここに記載した注意点はすべて考慮されています。メインモデルとサブモデル関係を使わない場合、本文書の注意点に配慮してください。）

ロック

Wagbyでデータを更新する場合、常に更新対象データは「ロック」されます。これによって同じデータの更新によるデータ破壊を防ぎます。

ロック方式は「悲観ロック」と「楽観ロック」の二種類を選択できます。標準は「悲観ロック」です。詳細は"ロック方式"をお読みください。

ロックはWagby内部で制御されています。そのため Wagby の外から直接、テーブルを操作された場合、Wagbyのロック機構は働きません。

ロックを考慮しないでよい場合

対象テーブルが読み込み専用であれば、ロックについて考慮する必要はありません。対象テーブルをWagbyからも更新し、かつ別システムからも更新される可能性がある場合、更新の競合を避けるため、ロックを考慮する必要があります。

楽観ロックが適用できる場合

対象テーブルが楽観ロックに対応しており、かつ「バージョン管理用カラム」をもっている場合は、Wagbyの楽観ロックを利用できます。具体的には、テーブル内に用意された「バージョン管理用カラム」を Wagby および他のシステムで共通に利用するということです。

悲観ロック情報を共有する場合

楽観ロックが使えない場合、Wagbyの悲観ロックテーブルを既存システムと共用することを検討してください。具体的には悲観ロックの情報をデータベースのテーブルとして表し、これを他システムから参照するようにします。

ロックキー

別々のモデルで同じ物理テーブルを共有する場合、ロックキーを同一にするとよいでしょう。 詳細は"ロックキーのカスタマイズ"をお読みください。

キャッシュ

Wagby はデータベースへ発行する SQL を可能な限り削減するようになっています。そのため、一度読み込んだデータをメモリ内に保持し、再利用します。これを「キャッシュ」と呼びます。

そのため Wagby の外から直接、テーブルを操作された場合、Wagbyの保持するキャッシュと実際のデータの整合性がとれなくなります。

そこでモデルのキャッシュを無効にする方法が提供されています。 しかしキャッシュを無効にするとパフォーマンスに影響があります。この問題に対する、いくつかの代案を示します。

定期的にWagbyのキャッシュ情報を消去する

更新頻度が少ない場合、キャッシュは使うが、ジョブを使ってキャッシュ情報を定期的に消去する方法があります。

外部からWagbyのキャッシュ情報を消去する

REST API を使って、Wagbyが保持する（モデルの）キャッシュ情報をクリアすることができます。外部から直接データベースを更新した場合にこの API を呼び出すという方法があります。

マテリアライズドビューを使う

Wagbyのキャッシュは無効としつつ、データベースが提供する「マテリアライズドビュー」を使う方法もあります。[Wikipedia...]

マテリアライズドビューはデータベース側でキャッシュされており、かつ、元のテーブルが変更されるたびにキャッシュも更新されます。そのため最新のデータを取得できます。

マテリアライズドビューが使えるデータベースであれば、こちらの利用も検討してください。

モデル項目

文字列

文字列型の種類

文字列型はさらにいくつかの属性を指定することができます。"モデル > 型 >Wagbyで利用できる型"をお読み下さい。

入力フィルタ

入力された値は「フィルタ」によって補正されます。 詳細は"入力フィルタの仕様"をお読みください。

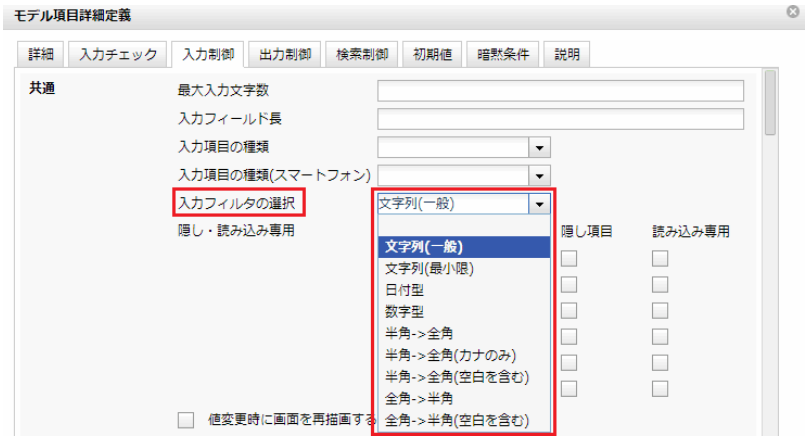


図 入力フィルタ

入力フィルタと許容文字種

許容文字種を指定した場合は、自動的に「文字列（最小）」フィルタが適用されます。

例えば許容文字種に全角数字を指定した場合、入力フィルタが「文字列(一般)」のままですとフィルタが「全角数字」を「半角数字」に変換してしまうため入力ができなくなってしまいます。 フィルタを「文字列(最小限)」とすることで、この問題を回避します。

入力フィルタを手動で設定する

業務仕様によってはフィルタを変更した方がよい場合があります。

例 郵便番号、住所項目

"文字列>郵便番号"をお読み下さい。

例 半角カナ文字のみを入力させたい

フィルタを文字列(最小限)にします。その上で、許容文字種を次のように指定します。

- 許容文字種の設定：カタカナ-半角文字
- 許容文字種の設定：スペース

データベースとの関係

Wagbyの内部では、文字列型のサイズの制限はありません。ただしデータベースに保存されるときは、データベースのカラム内におさまる必要があります。詳細...

数値

数値型の種類

数値型はさらにいくつかの属性を指定することができます。 "Wagbyで利用できる型"をお読み下さい。

入力項目の種類

後述する「入力ルール」や「出力フォーマット」などの設定は、入力項目の種類が「数字」の場合に有効です。通常、入力項目の種類は「数字」となっています。この設定を空白にすると、入力項目は文字列として扱われます。[詳細...]

図 入力項目の種類

入力フィルタ

入力された値は「フィルタ」によって補正されます。数値型項目は標準で「数字型」フィルタが選択されています。 詳細は"入力フィルタの仕様"をお読みください。

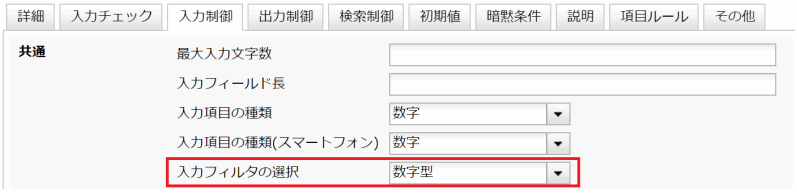


図 入力フィルタ

入力ルール

カーソル位置

数値型項目では、登録・更新画面のカーソル位置は「右」になります。（変更することはできません。）いずれの型であっても、このルールが適用されます。[この制約を解除する...]

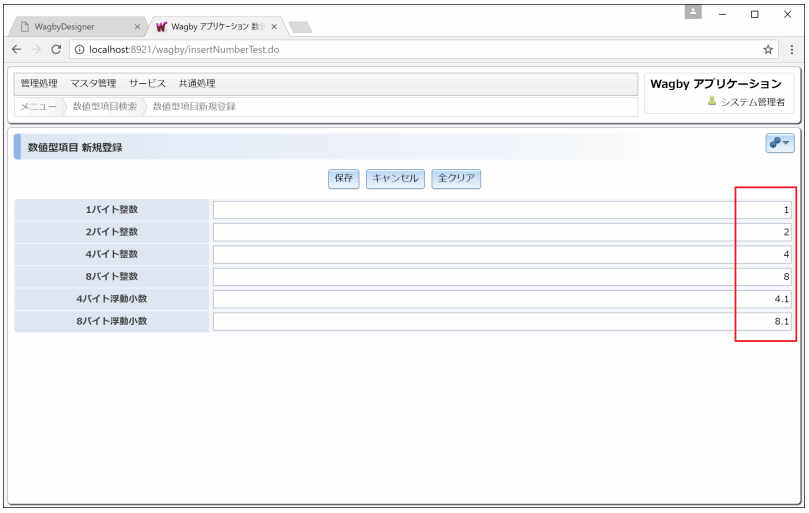


図 カーソル位置

数字以外の文字のチェック

数値型項目の場合、数字以外の入力を行うとリアルタイムに入力チェックが適用されます。 全角文字も入力することはできません。

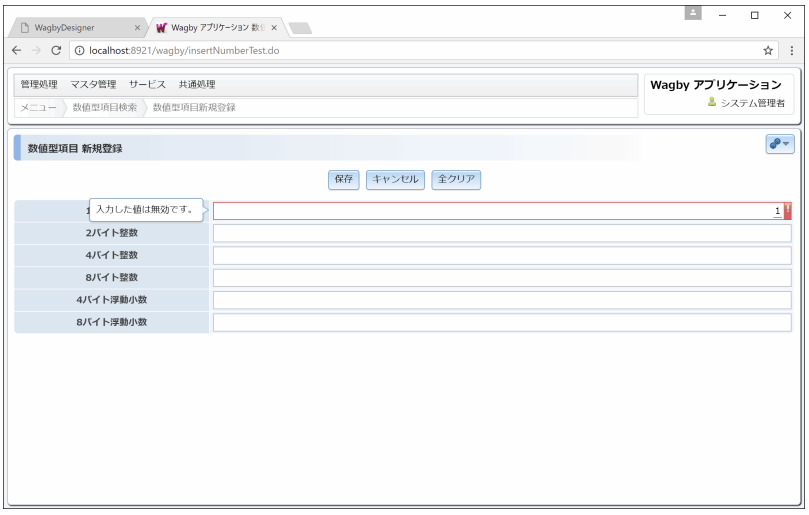


図 全角の「１」を入力するとエラー

入力できる数値の範囲

入力できる数値の範囲は -9, 000, 000, 000, 000, 000 から 9, 000, 000, 000, 000, 000 までとなっています。いずれの型であっても、このルールが適用されます。[この制約を解除する...]

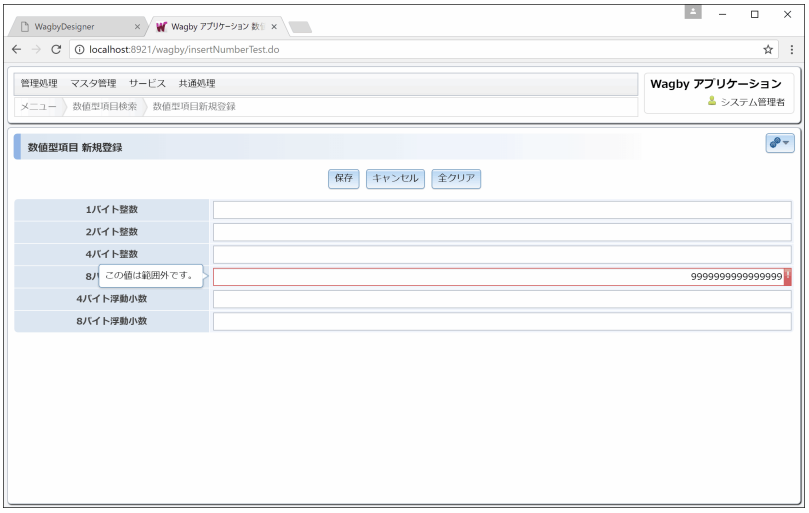


図 入力できる範囲を超えた場合のエラーメッセージ

入力できる小数部

入力できる小数部は6桁となっています。いずれの型であっても、このルールが適用されます。[この制約を解除する...]

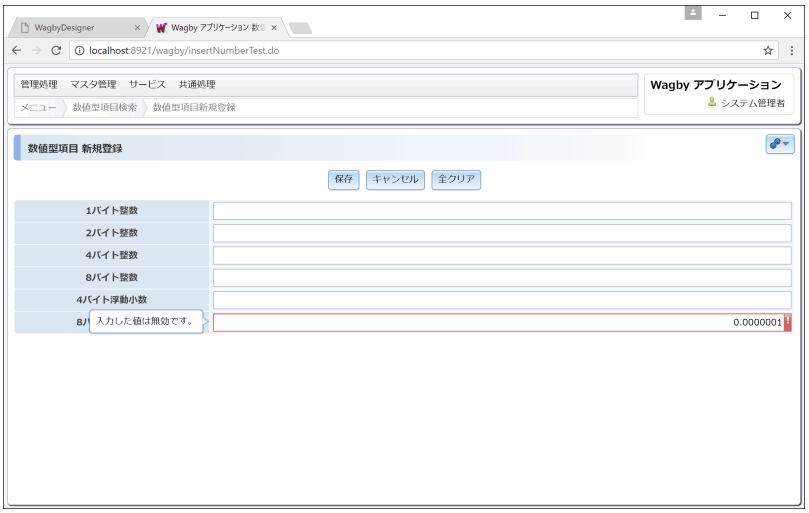


図 入力できる小数部を超えた場合のエラーメッセージ

> このルールは入力画面にのみ適用されます。演算によって求められた値やアップロード更新、REST API利用時は、指定した型の範囲内で保持することができます。

> 業務上、有効な数値範囲を定め、範囲チェックを指定するとよいでしょう。

IME制御 (8.2.0)

数値入力欄では、自動的に IME がオフになります。

(11)

出力フォーマット

(11) IME制御が有効なブラウザでのみ、機能します。2019年1月時点では IE11 と Firefox で動作することを確認しています。

例

数値を3桁区切りで出力するように設定した例を図に示します。

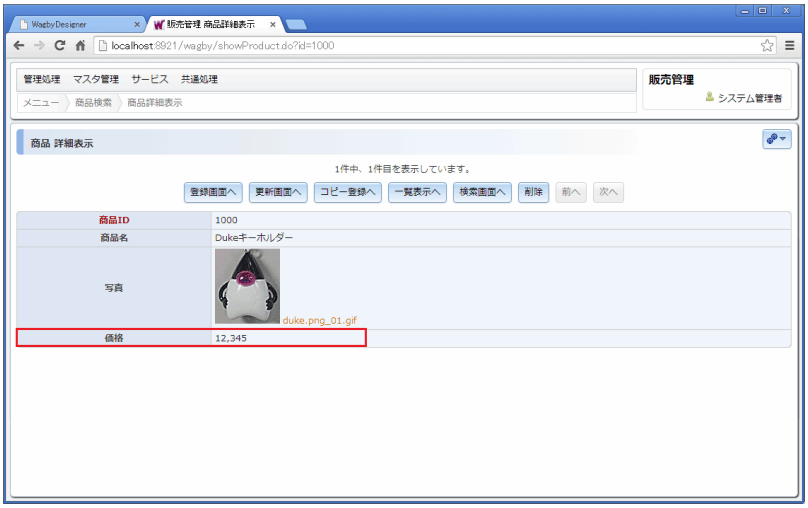


図 価格が3桁区切りで表示されている

このフォーマット指定は入力時にも適用されます。カーソルのフォーカスが移動したタイミングで、フォーマットされます。（入力中ではありません。）

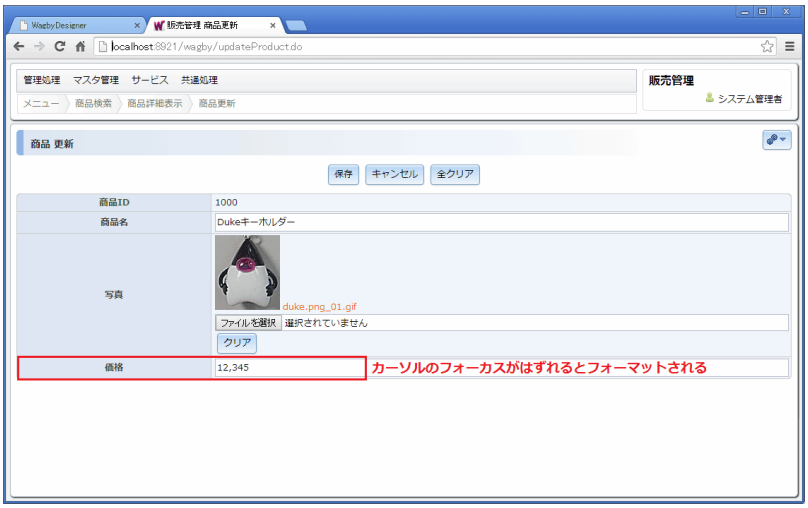


図 入力が終わったタイミングでフォーマットが適用される

定義方法

モデル項目詳細定義ダイアログの「出力制御>詳細画面>フォーマット」を指定します。 （一覧画面での設定欄はありません。一覧画面には、ここで指定したフォーマットが適用されます。）

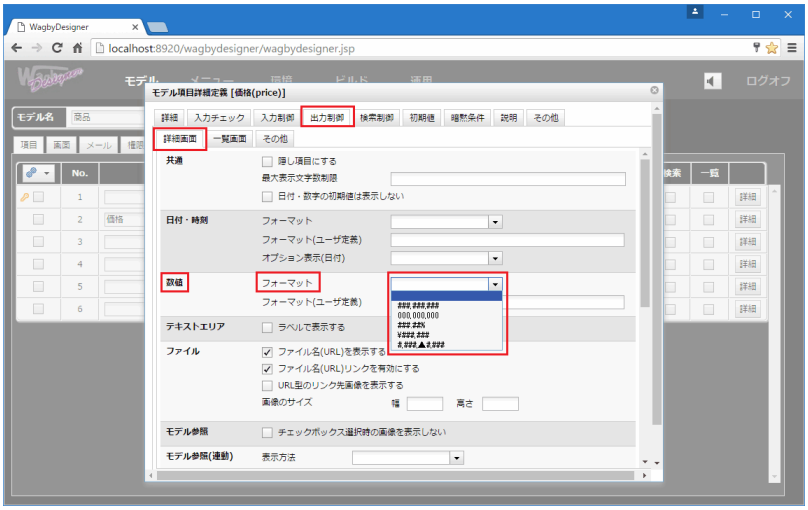


図 出力フォーマットの指定

用意されているフォーマット

次のようなフォーマット表記が選択肢で用意されています。

フォーマット指定	格納された数値	出力された表記
###,###.###	1234.56	1,234.56
000,000.000	1234.56	001,234.560
###.##%	0.5	50%
¥###,### (*1)	10000	¥10,000
#,###;▲#,###	12800	12,800
	-12800	▲12,800

フォーマット未指定時の動作

4バイト浮動小数点/8バイト浮動小数点の項目では、フォーマット未指定時は ##0.0#### になります。 このため、"5" と入力すると "5.0" と表示されます。

フォーマットを独自に定義する

ユーザが独自に数値フォーマットを用意することもできます。 次のパターン文字を使うことができます。[12]

記号	意味
0	数字
#	数字。ゼロだと表示されない。
.	数字の桁区切り（または通過の桁区切り）
-	マイナス記号
,	グループ区切り
;	正と負のサブパターンを区切る
%	100 倍してパーセントを表す

例：整数部は3桁区切りとし、小数点第五位までの数値を出力する

```
#,##0.00000
```

例：整数部はそのままに、小数点第一位までの数値を出力する

値 "1" の場合、表示時は "1.0" になります。

```
0.0
```

[12] 詳細は Java クラスライブラリが提供する java.text.DecimalFormat クラスのドキュメントを参照してください。

例：整数値の場合は整数のみ、小数点まで入力されれば小数点まで出力する

```
###.##
```

例えば ″1500″ と（整数のみ）入力すると ″1500″ と表示されます。″141.14″ と（小数点を含んだ値を）入力すると ″141.14″ と表示されます。

整数部を3桁カンマ区切りとしたい場合は次のようにします。

```
###,###.##
```

例えば ″1500″ と（整数のみ）入力すると ″1,500″ と表示されます。

例：パーセントの扱い

フォーマットに「%」文字を指定すると、計算結果が自動的に100倍されます。計算式を使わずに(100倍を)実現できます。

```
###.0%
```

1を入力すると ″100.0%″ と表示されます。

0.1を入力すると10%と、0.8を入力すると80%としたい場合、次のようにします。

```
###.##%
```

小数点以下の位を2桁以上表示させたい場合は、「0」を増やして指定してください。例えば、小数点第3位まで表示させる場合は次のように指定します。

```
###.000%
```

入力された小数点の値だけを表示したい場合は、次のようにします。

```
###.0##%
```

上のフォーマットでは、0.152 を入力すると 15.2% と表示されます。

非推奨フォーマット

###% という表記は小数点の利用は非推奨です。この書式は小数点第1位で四捨五入するため、0.8は100%と表示され、0.1は0%と表示されてしまいます。

円マークの扱い

円マークつきのフォーマット（例 ¥###,###）を指定すると、新規登録画面でも円マークが表示されます。

参照連動項目の場合

参照連動項目型の場合、出力フォーマットは、参照側モデルの設定が反映されます。

制約

- 一覧表示グリッド形式、一覧更新グリッド形式では出力フォーマット指定は無視されます。

トラブルシューティング：出力フォーマットが反映されない

入力項目の種類が「数字」となっているかどうかを確認してください。[詳細...]

小数点扱い時の注意

小数の場合、書式指定した桁数よりも実際の値が多い場合は「丸め」処理（「四捨五入」「切り上げ」「切り捨て」）が自動的に行われます。この判定は、端数が出た場合、その上下の近い方の数字に合わせます。

画面からの入力

Wagbyが内部で利用しているJavaScriptライブラリ(Dojotoolkit)により、次のような動作になります。
例） 指定桁数：小数点2桁の場合

入力文字	保存される値
0.123	0.12
0.127	0.13
0.125	0.13

CSVアップロード更新やREST APIなどからの入力

JavaScriptを経由しない入力の場合は、Javaの仕様により、次のような動作になります。 例） 指定桁数：小数点2桁の場合

入力文字	保存される値
0.123	0.12
0.127	0.13
0.125	0.12

(13)

初期値 0 を表示しない

"入力制御>日付・数値の初期値は表示しない"をお読み下さい。

検索条件としての利用

"検索制御>範囲検索を行う"をお読み下さい。

入力欄を文字列として扱う

数値の場合、入力欄は自動的に次のルールが適用されます。

- 入力カーソルの位置は右。
- 入力できる範囲は -9,000,000,000,000,000 から 9,000,000,000,000,000 まで。
- 入力できる小数部は6桁。

業務用件で、この制約を解除する必要がある場合、次の設定を行います。

対象項目の「入力制御」タブを開きます。数値の場合、「入力項目の種類」「入力項目の種類（スマートフォン）」のいずれも「数字」になっています。

(13) Java は小数の扱いについての、日本工業規格の「JIS丸め」(JIS Z 8401)に準拠しています。 この仕様では、"0.125" は偶数側に丸められ "0.12" となります。 ("0.125"の"5"は、上下の数字"0.12","0.13" 両方に同じ距離となるので、偶数である "0.12" に丸められます。) "0.135" は偶数側に丸められ、"0.14" となります。

図 入力項目の種類

この欄を空白にします。入力欄は文字列型として扱われるため、入力できる数字の範囲チェック処理が行われなくなります。

図 入力項目の種類を解除する

15桁以上の大きな数字を扱う場合、型も「8バイト整数」と変更してください。ただし扱える数値は-9,223,372,036,854,775,808 ～ 9,223,372,036,854,775,807の範囲となります。同様に6桁を超える小数点を扱う場合、型も「8バイト浮動小数」と変更してください。扱える精度はおおむね小数点以下およそ15桁の精度です。[詳細...]

入力項目の種類を解除すると入力ルールや出力フォーマットも無効になります。

データベースとの関係

数値型はデータベースによって型が変わります。詳細...

日付

日付型

日付・時刻型で「日付」「時刻」「日付と時刻」の3種類から「日付」を選択すると、年月日を扱う項目が用意されます。 年は1753年～9999年の範囲で指定できます。

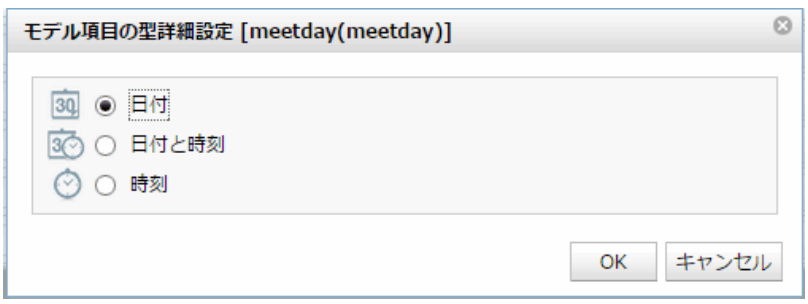


図 日付型

Javaとの関係

日付型項目は、内部では java.sql.Date クラスとして管理されます。

新元号への対応

設定ファイルを修正し、新元号を追加することができます。

仕様・制約

- 日付型項目を主キーとしたとき、日付の書式は "yyyy-MM-dd" のみが許容されます。他の書式は指定できません。
- 日付型項目で「年」を含めないフォーマット指定を行うと、年部分は "1970" で補完されます。同様に、月や日を省略すると "01" で補完されます。
- 和暦フォーマットを指定し、かつ、日付オートフォーマットと併用するときは、入力欄をカレンダ入力形式としてください。（入力欄がテキストフィールドの場合は、保存後の詳細画面で和暦フォーマットが適用されます。）
- カレンダ形式の入力、リストボックス形式の入力と、日付オートフォーマットを同時に設定することはできません。
- オプション表示で曜日を出力する場合、年月日すべての情報が必要です。いずれが欠けても曜日は正しく出力されません。

[応用] カレンダ利用時の入力チェックルールの変更 (8.3.0)

カレンダを利用した入力では、この項目に設定した日付フォーマットの正規表現を入力チェックに利用します。

(14)

正規表現ルールを追加する

この正規表現ルールを追加することができます。アプリケーション共通で用いられる mycommon.js に次のスクリプトを設定します。

```
function getMyDateRegExp() {
  return 追加したい正規表現式;
}
```

例えば数字2桁ハイフン区切りの入力も許容させる場合は次のようにします。

```
function getMyDateRegExp() {
  return "¥¥d{1,2}-¥¥d{1,2}-¥¥d{1,2}";
}
```

正規表現ルール自体をカスタマイズする

正規表現での入力チェック自体を変更することもできます。Wagby に同梱されているカレンダ入力機能を提供する DateTextBox.js を直接、修正します。手順を示します。

1. wagbyapp/webapps/$(APPNAME)/dojo-release-x.y.z/wagby/DateTextBox.js をテキストエディタで開きます。dojo-release-x.y.z フォルダの x,y,z はインストールされている Wagby が同梱している Dojotoolkit ライブラリのバージョン番号に読み替えてください。

2. この JavaScript ファイルに次の内容を加えます。

```
validator: function(value, constraints) {
  // 成功すれば true を返す処理を記述する。
}
```

例えば日付のリアルタイム入力チェックを行わないようにするためには、次のようにします。

```
validator: function(value, constraints) {
  return true;
```

(14) 日付フォーマットが "yyyy-MM-dd" の場合、正規表現式¥d{1,4}-¥d{1,2}-¥d{1,2}が自動的に適用されます。

```
}
```

3. ブラウザをリロードして動作を確認します。

4. カスタマイズコードができましたら、同じ内容を次のファイルにも記述します。 wagbydesigner/bin/webpage/dojo-release-x.y.z/wagby/DateTextBox.js これによって次回のビルド以降もこの修正が反映されるようになります。

ご注意ください ： この修正は標準の動作を直接、変更しています。Wagby のバージョンアップを行なった場合、同梱されている新しい DateTextBox.js ファイルに対して、同じような修正コードを再び記述するようにしてください。

入力フィルタ

入力された値は「フィルタ」によって補正されます。 詳細は"入力フィルタの仕様"をお読みください。

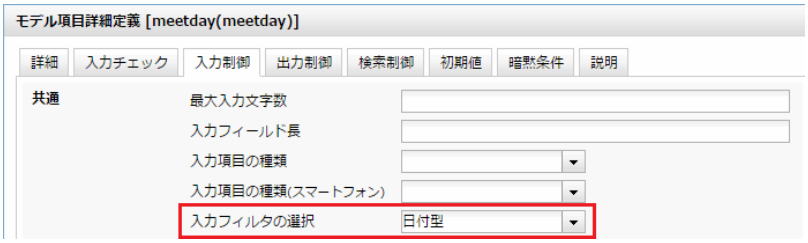


図 入力フィルタ

カレンダの利用

日付型項目は図のような入力欄が用意されます。

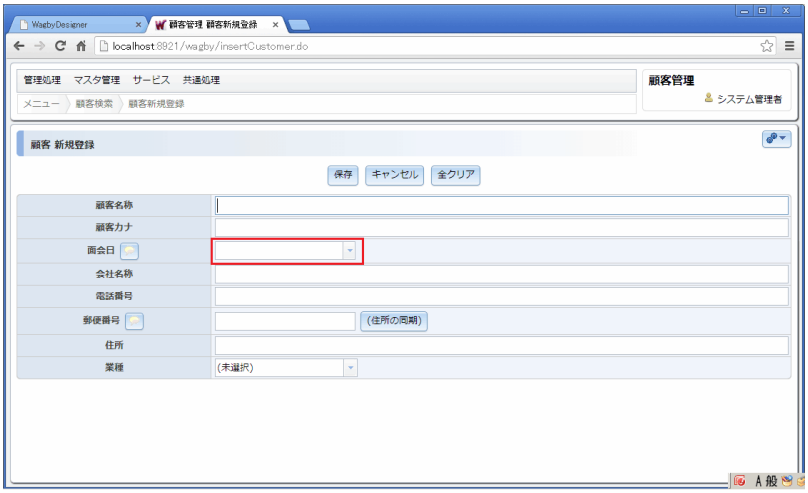


図 日付型項目の入力欄

数字とハイフン（「-」）以外の入力は図のようなエラーになります。正常に入力した場合は図のようになります。

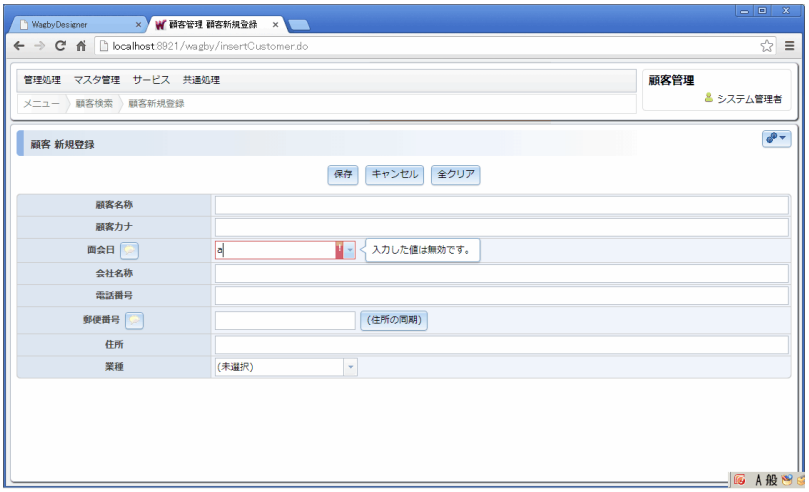


図 入力エラー

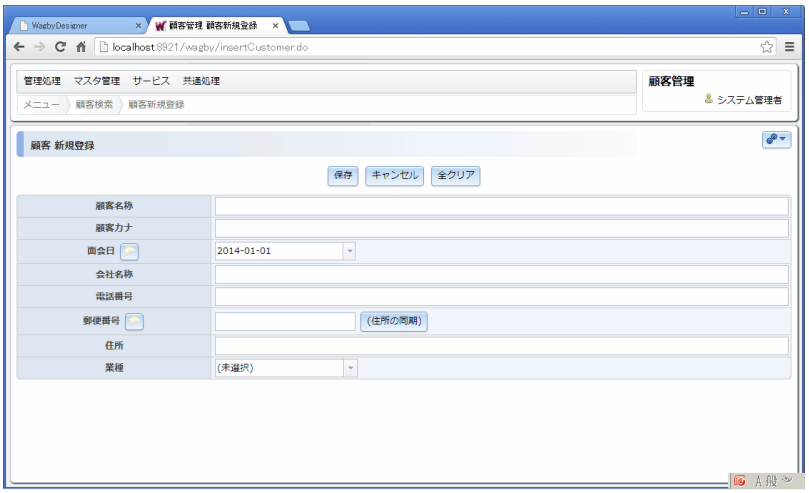


図 入力が正常に行えた

入力欄をクリックすると図のようにカレンダが表示されます。[15]

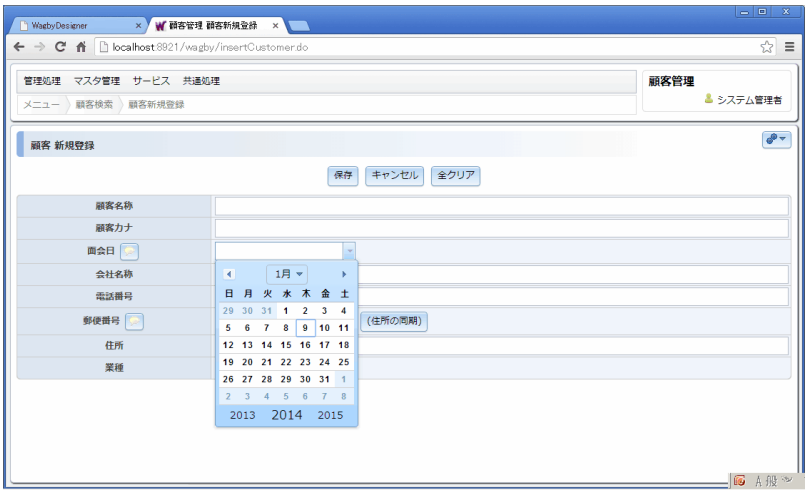


図 カレンダによる入力

(15) Date Picker と呼びます。

定義方法

モデル項目詳細定義ダイアログの「入力制御」タブから、日付入力方式を「カレンダの利用」とします。（日付型の標準は、「カレンダの利用」となっています。）

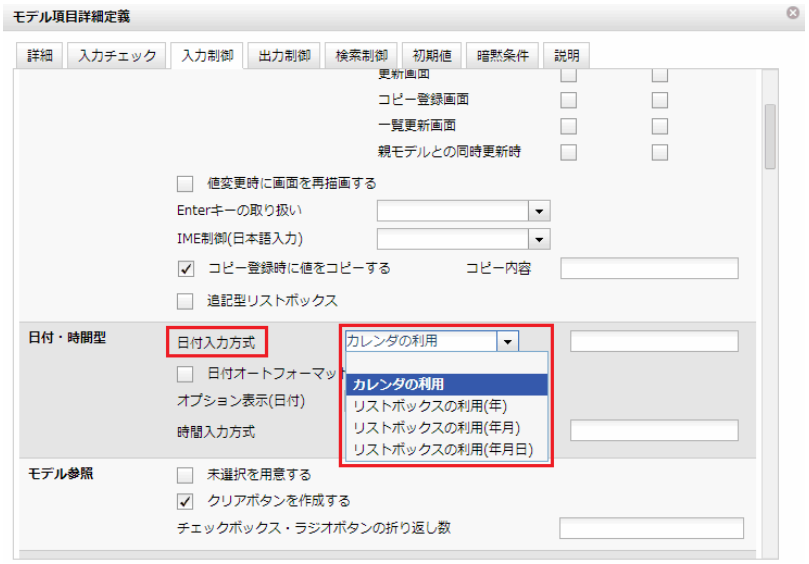


図 カレンダの利用を行う設定

入力フィールドの幅

標準では、プルダウンで表示されるカレンダの幅がそのまま入力フィールドの幅となります。この幅を調整することができます。 詳細は"入力制御 > 入力フィールド長 > 日付型（カレンダ利用）の場合"をお読みください。

入力チェックルールの変更

カレンダを利用した入力では、この項目に設定した日付フォーマットの正規表現を入力チェックに利用します。このルールの追加や変更を行うことができます。

リストボックスの利用

カレンダではなくリストボックス形式で日付を入力することもできます。 図は、年のみを入力するリストボックスです。（月日は自動的に1月1日として扱います。）

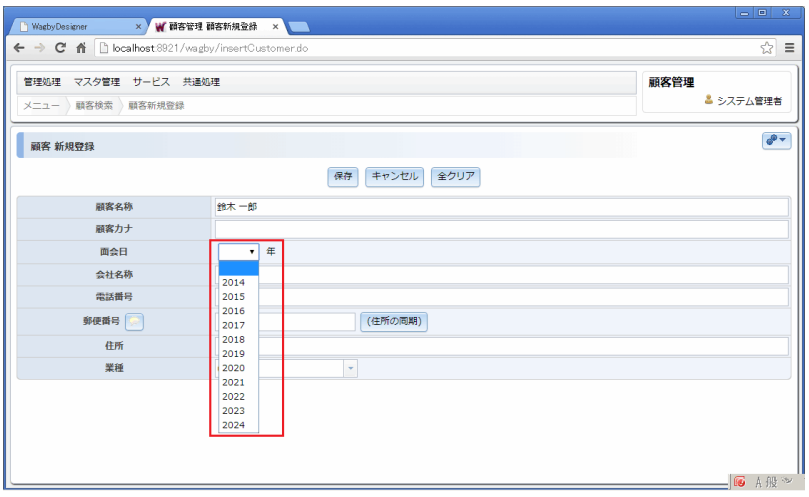


図 リストボックスで年を指定する

図は年と月をリストボックス形式で入力する例です。（日は自動的に1日として扱います。）

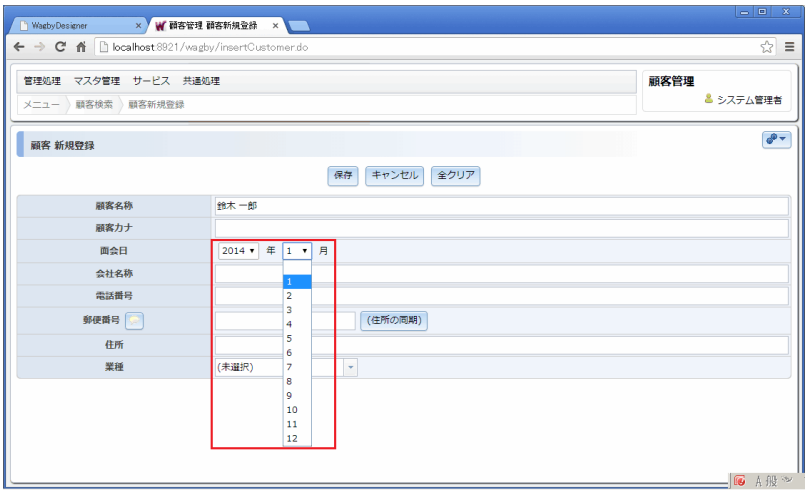


図 リストボックスで年月を指定する

図は年月日をリストボックス形式で入力する例です。

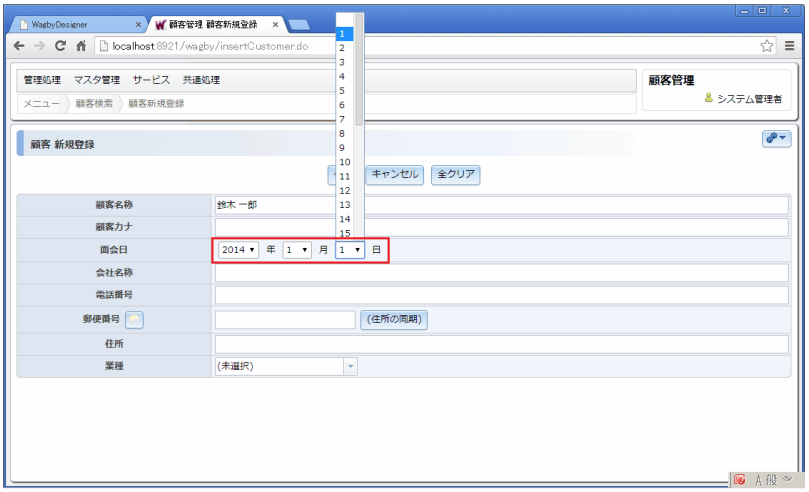


図 リストボックスで年月日を指定する

開始年・終了年を指定する

リストボックス利用時は、（選択肢として表示される）開始年と終了年を指定できます。 図は、開始年を1950年とし、終了年を現在の年とした例です。

図 開始年と終了年を指定する

図のように年の開始が1950年からとなります。

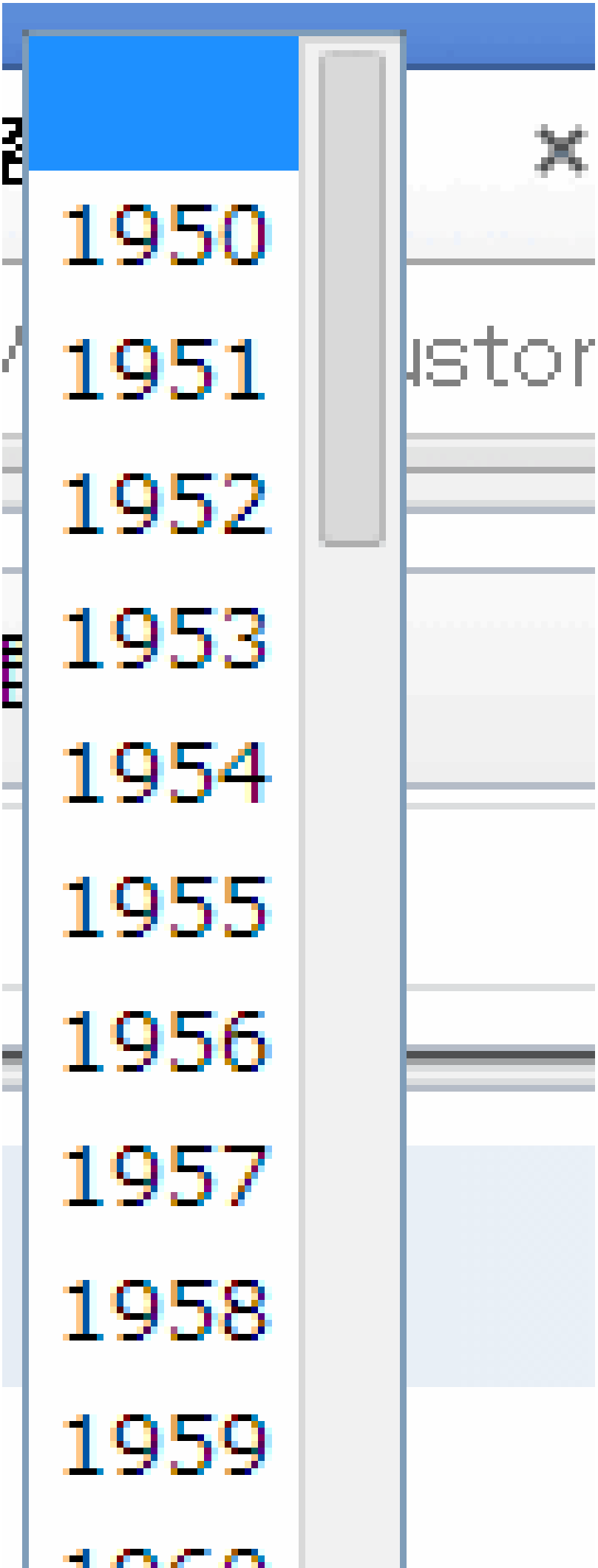

1950
1951
1952
1953
1954
1955
1956
1957
1958
1959

図 1950年から選択できる

指定書式は次のとおりコンマで区切ります。

```
開始年,終了年
```

開始年と終了年に「now」という表記を用いることもできます。この now は、現在の年という意味です。例を示します。

1950年から、2020年までを選択できるようにする。

```
1950,2020
```

1950年から、今年までを選択できるようにする。

```
1950,now
```

10年前から、2020年までを選択できるようにする。

```
now-10,2020
```

50年前から、10年後までを選択できるようにする。

```
now-50,now+10
```

検索制御

検索画面の項目でも、同じ設定を行うことができます。

図 検索制御タブの設定欄

オートフォーマットの利用

日付オートフォーマット機能とは、さまざまな入力パターンからyyyy-MM-dd形式に自動変換する機能です。

例（カレンダの場合）

"5 20" と入力し、カーソルを離すと自動的に変換されます。

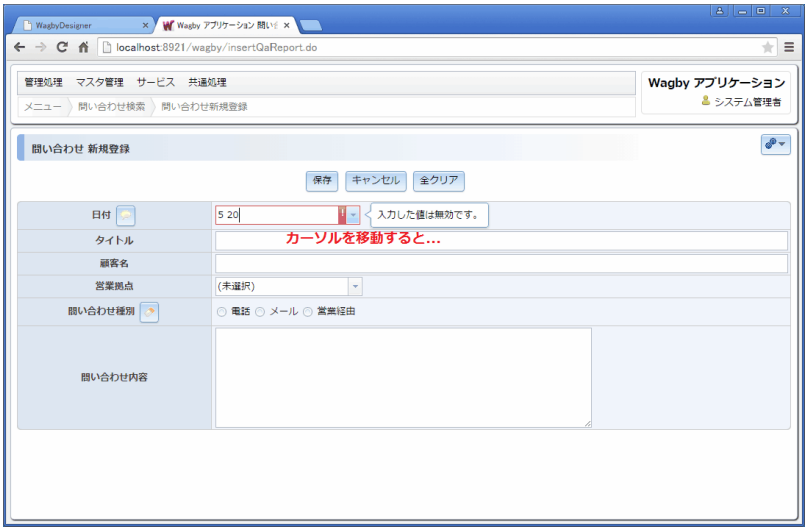


図 日付オートフォーマット入力時

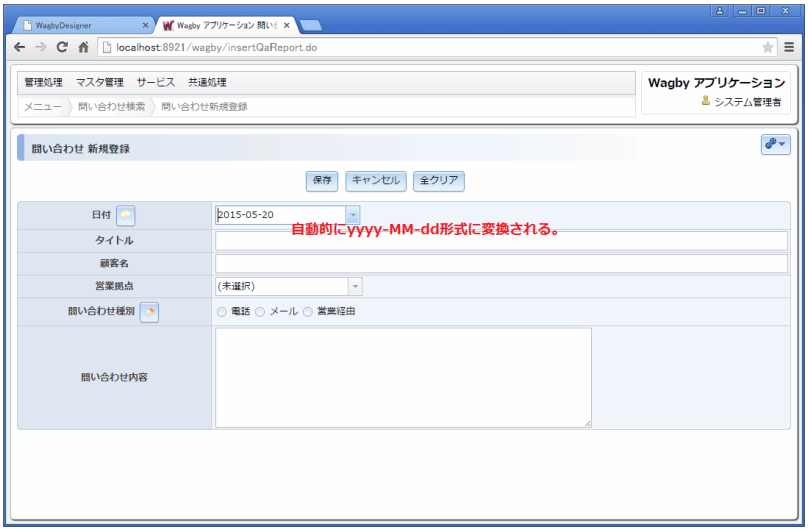


図 日付オートフォーマットカーソル移動直後

定義方法（カレンダの場合）

定義箇所はありません。カレンダ入力の場合、日付オートフォーマットは自動的に有効になります。（無効にすることはできません。）

例（テキストフィールドの場合）

″11 7″と入力し、カーソルを離すと自動的に変換されます。

図 日付オートフォーマット入力時

図 日付オートフォーマットカーソル移動直後

定義方法（テキストフィールドの場合）

モデル項目詳細ダイアログで日付入力方式の選択肢を空(直接入力)にし、日付をテキストボックスによる直接入力とします。 その上で「日付オートフォーマット機能を有効にする」のチェックを設定します。

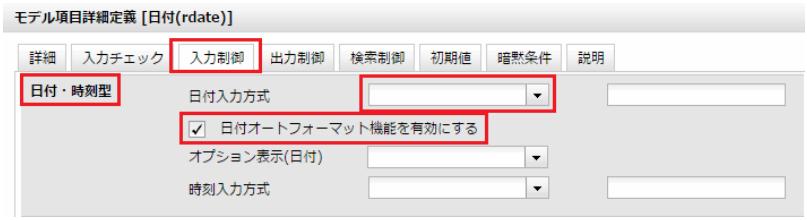


図 日付オートフォーマットの設定

変換のパターン

変換パターン例を以下に示します。

入力値	変換後
11 7	2014-11-07
H26 11 7	2014-11-07
2014/11/7 または 2014/11/07	2014-11-07
2014 11 07	2014-11-07
20141107	2014-11-07
s 47 5 15	1972-05-15

(16)

日付フォーマットの指定

日付・時刻のフォーマットは「出力制御」タブで指定します。

図 日付・時刻フォーマットの指定

出力制御という枠組みになっていますが、入力時も適用されます。

図 yyyy-MM-dd

図 yyyy年MM月dd日

MM月指定とM月指定

フォーマットは「MM月」「dd日」と「M月」「d日」が用意されています。MMやdd書式では、1月は"01月"と、必ず2桁で表記されます。

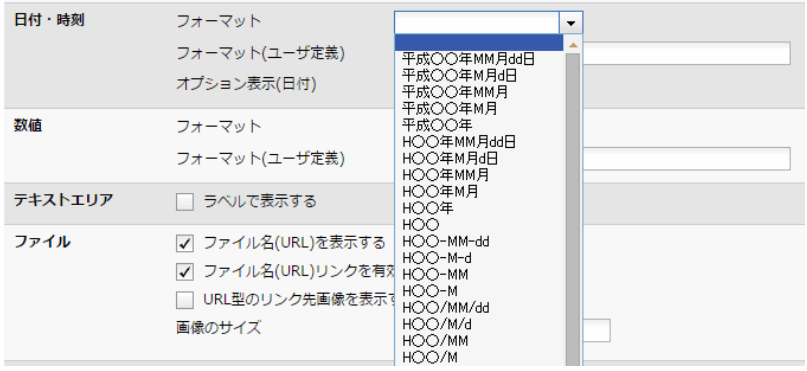


図 日付・時刻フォーマットの選択肢

ユーザ定義

Wagbyが標準で用意したフォーマットではなく、開発者が直接、フォーマットを記述することもできます。

型	フォーマット例	出力例	説明

(16) m(またはM)、t(またはT)、s(またはS)、h(またはH)はそれぞれ明治、大正、昭和、平成を意味する特別表記です。

日付	wa{kanji02}年MM月dd日	平成02年01月01日	年月日のゼロ詰め(*)
	wa{kanji01}年MM月dd日	平02年01月01日	
	wa{alphabet0}年MM月dd日	H02年01月01日	
日付と時刻	wa{kanji2}年MM月dd日 HH:mm:ss	平成30年09月01日 0:00:00	和暦と時刻
日付と時刻	wa{alphabet}年M月d日 HH:mm:ss	H30年9月1日 0:00:00	和暦アルファベット表記

(17)

オプション表示

入力制御

入力欄の横に、別の日付フォーマットで表示させることができます。　　図は「（平成〇年〇月〇日〇曜日）」というフォーマットを指定した例です。

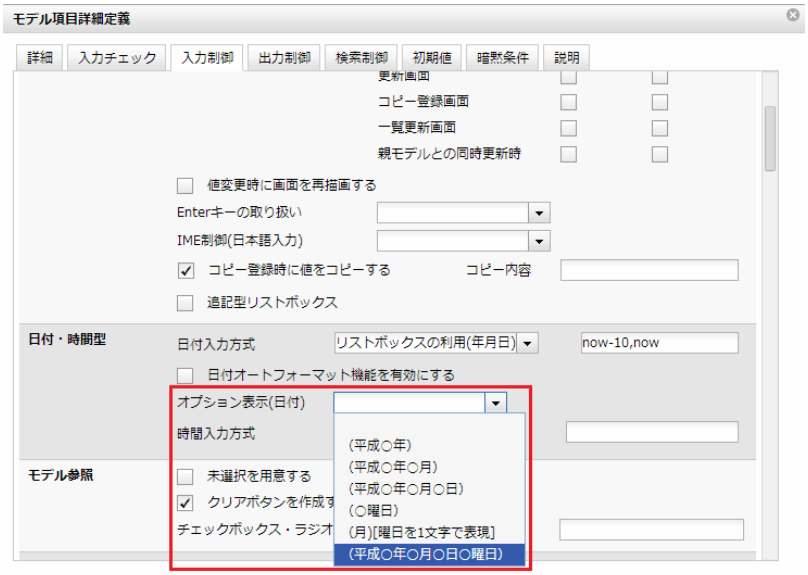


図 オプション表示の設定

図のように、入力欄の横に日付が表示されます。このオプション表示は登録、更新画面で有効です。

(17) 和暦が1桁の場合の年度に0を埋めるフォーマット wa{kanji02},wa{kanji01},wa{alphabet0} は R8.1.2 以降の Wagby で利用できます。

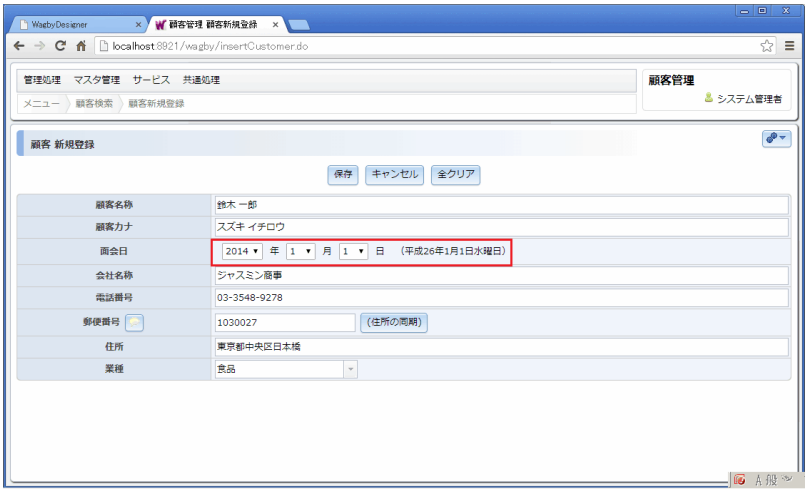


図 オプション表示の例

用意されているフォーマットは次のとおりです。

- 平成〇年
- 平成〇年〇月
- 平成〇年〇月〇日
- 〇曜日
- 月[曜日を1文字で表現]
- 平成〇年〇月〇日〇曜日

「日付、日付と時間、時間フォーマット（ユーザ定義）」を指定している場合、本設定を利用することはできません。

出力制御

出力制御にもオプションを指定することができます。

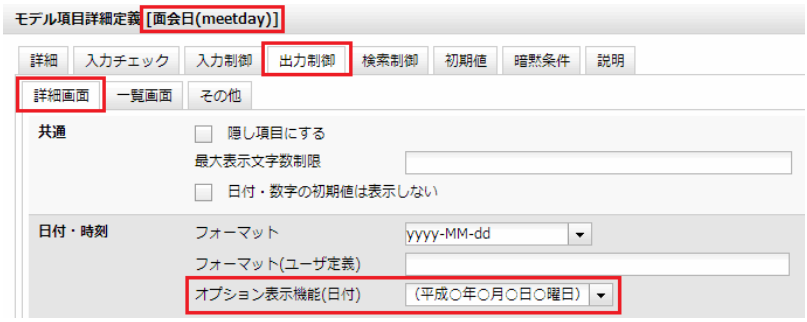


図 オプション表示の設定（出力制御）

図 オプション表示の例（出力時）

システム時刻(1970年1月1日)を表示しない

"入力制御>日付・数値の初期値は表示しない"をお読み下さい。

検索条件としての利用

"検索制御>範囲検索を行う"をお読み下さい。

和暦の元年表記

「入力制御 ＞ 日付・時刻 ＞ オプション表示(日付)」および「出力制御 ＞ フォーマット」で和暦フォーマットを指定した場合、平成1年を平成元年と表示します。

入力画面では「オプション表示(日付)」の対応を行っていますが、ドロップダウンカレンダの入力フィールドの日付は"平成1年"と表示されます。これは仕様となります。
(18)

時刻

時刻型

日付・時刻型で「日付」「時刻」「日付と時刻」の3種類から「時刻」を選択すると、時刻を扱う項目が用意されます。

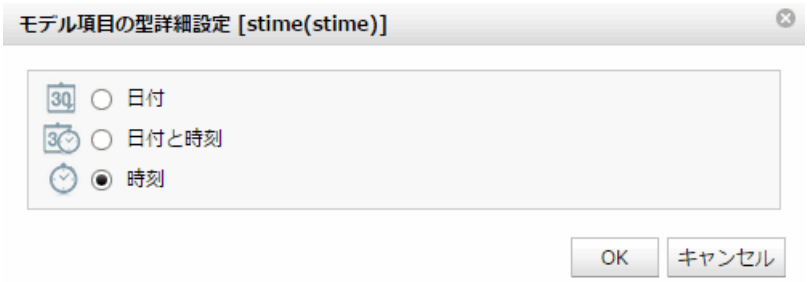


図 時刻型

Javaとの関係

時刻型項目は、内部では java.sql.Time クラスとして管理されます。

入力フィルタ

入力された値は「フィルタ」によって補正されます。 詳細は"入力フィルタの仕様"をお読みください。

時刻型項目で選択できるフィルタは日付型と同じです。 詳細は"日付型>入力フィルタ"をお読みください。

リストボックスの利用

時刻をリストボックスで入力することができます。 図,図,図はそれぞれ時、時分、時分秒の入力イメージです。

(18) この動作をカスタマイズすることができます。 詳細は"Wagby Developer Network ＞ 和暦の扱い"をお読みください。

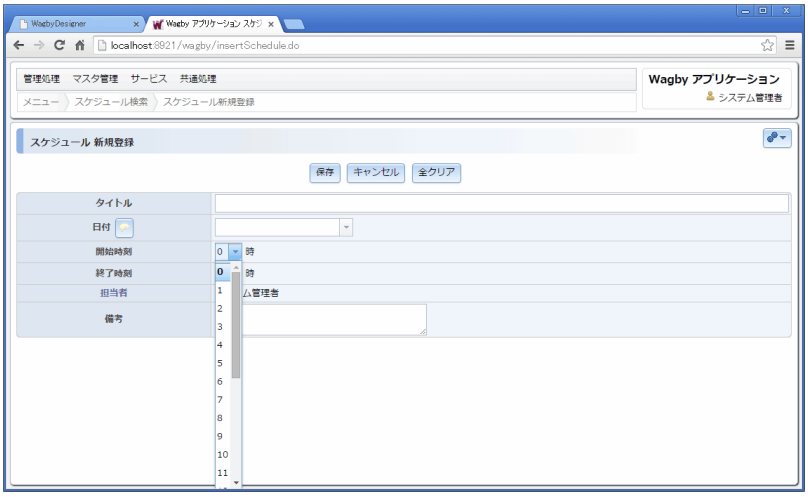


図 「時」入力（分と秒はゼロになる）

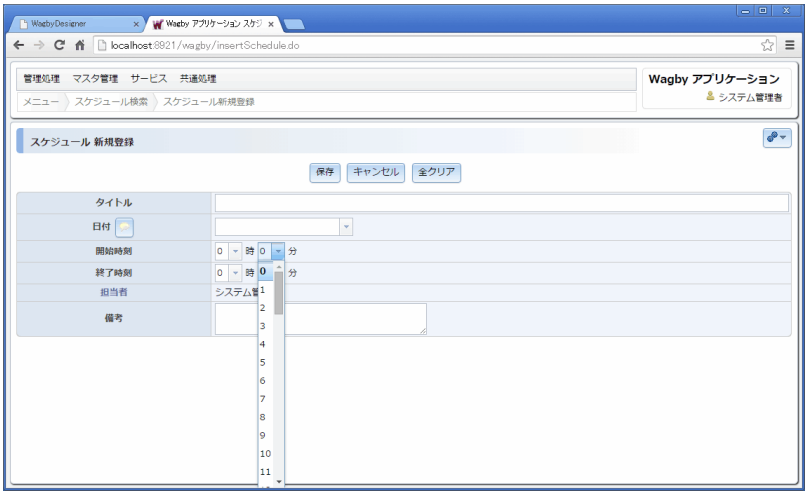


図 「時分」入力（秒はゼロになる）

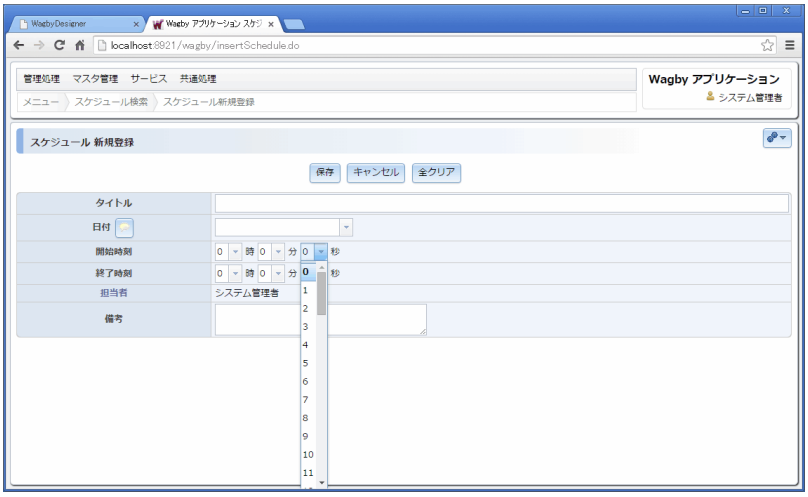


図 「時分秒」入力

定義方法

モデル項目詳細定義ダイアログの「入力制御」タブから、時刻入力方式を「リストボックスの利用」とします。

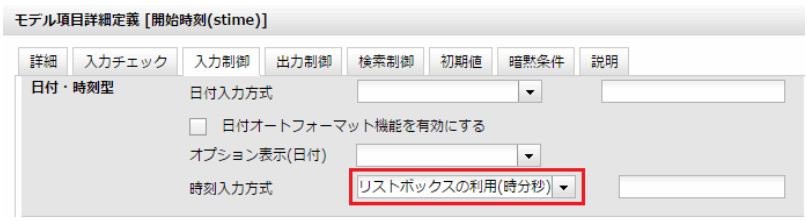


図 時刻入力方式の設定

リストボックスの表示ルール

リストボックスに用意される選択肢にルールを設定することができます。　　図は、時間に「7時から開始で、3時間おき」というルールを設定した例です。

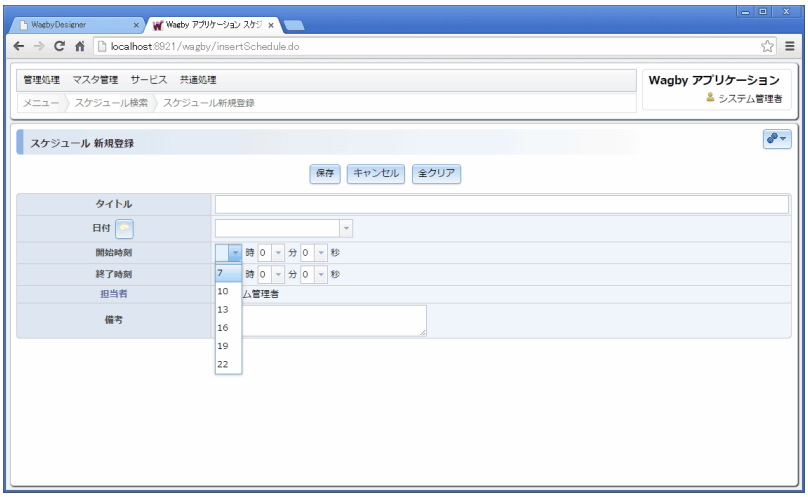


図 時間の選択肢へのルール設定

同じく分を「5分おき」とした例を図に示します。

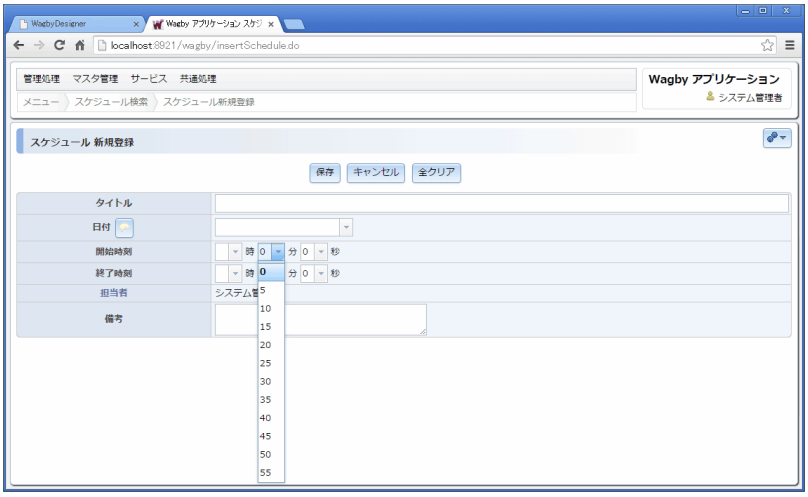


図 分の選択肢へのルール設定

同じく秒を「10秒おき」とした例を図に示します。

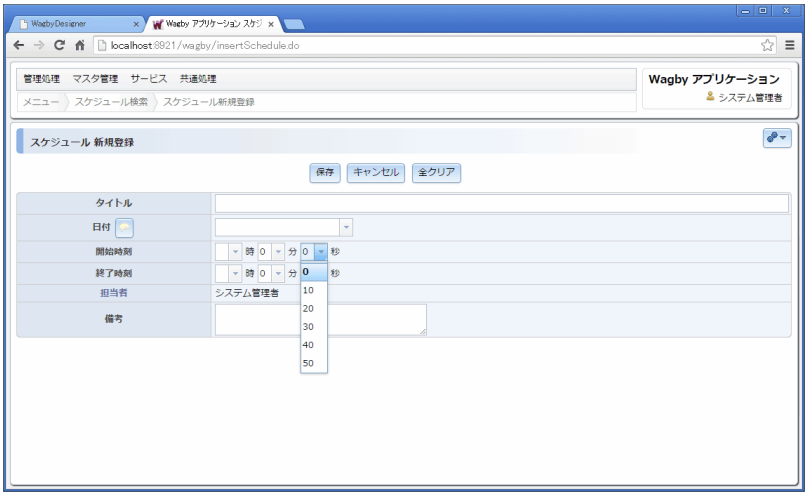


図 秒の選択肢へのルール設定

定義方法

時刻入力方式の右覧にルールを設定します。

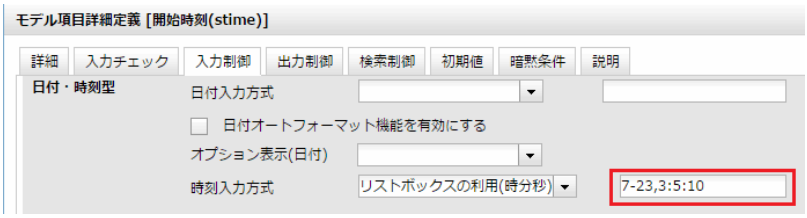


図 時刻入力方式のルール設定

ルールのサンプルを示します。

7時から23時まで、1時間おき。

```
7-23
```

7時から23時まで、3時間おき。

```
7-23,3
```

7時から24時まで、分は5分おき

```
7-24:5
```

7時から24時まで、3時間おき。分は15分おき。秒は30秒おき

```
7-24,3:15:30
```

テキストフィールドの利用

時刻入力方式を空欄とした場合、テキストフィールドで直接、時刻を入力します。 テキストフィールドで入力する場合の入力方式は、「出力制御 > 日付・時刻 > フォーマット」に従います。例を示します。

図 時刻入力方式のフォーマット設定

HH:mm:ss

- 時間は 0 から 23 まで、分と秒は 0 から 59 の範囲で入力できます。
- 時間、分、秒をそれぞれ空白またはコロン（":"）文字で区切って入力します。（例 10 10 10なら10時10分10秒となります）

HH:mm

- 時間は 0 から 23 まで、分は 0 から 59 の範囲で入力できます。
- 時間、分をそれぞれ空白またはコロン（":"）文字で区切って入力します。（例 10 10 なら10時10分となります）
- 秒を入力することはできません。入力しても無視されます。

HHmmss

- 時間は 0 から 23 まで、分と秒は 0 から 59 の範囲で入力できます。
- 時間、分、秒を連続して入力します。（例 101010 なら10時10分10秒となります。123なら12時3分0秒となります。）

このルールにそぐわない値を入力した場合は、時刻として認識されません。保存すると空白になります。

日付時刻

日付時刻型

日付・時刻型で「日付」「時刻」「日付と時刻」の3種類から「日付時刻」を選択すると、日付時刻を扱う項目が用意されます。年月日と時刻を同時に扱うもので、年は1753年～9999年の範囲で指定できます。

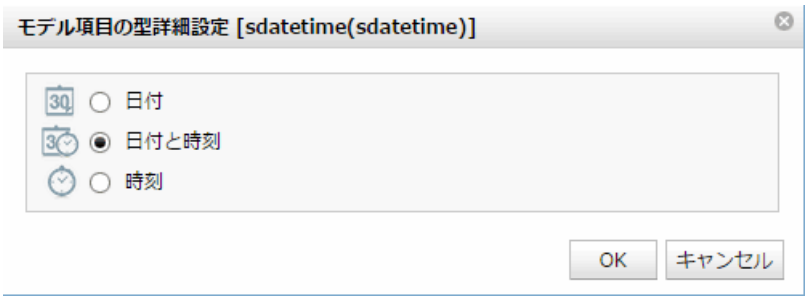


図 日付時刻型

Javaとの関係

時刻型項目は、内部では java.sql.Timestamp クラスとして管理されます。

入力フィルタ

入力された値は「フィルタ」によって補正されます。詳細は"入力フィルタの仕様"をお読みください。

日付時刻型項目で選択できるフィルタは日付型と同じです。詳細は"日付型 > 入力フィルタ"をお読みください。

ユーザーインタフェース (8.1.0)

「入力制御 > 日付・時刻型 > 日付入力方式」に「カレンダの利用」を指定すると、図のような入力欄が用意されます。

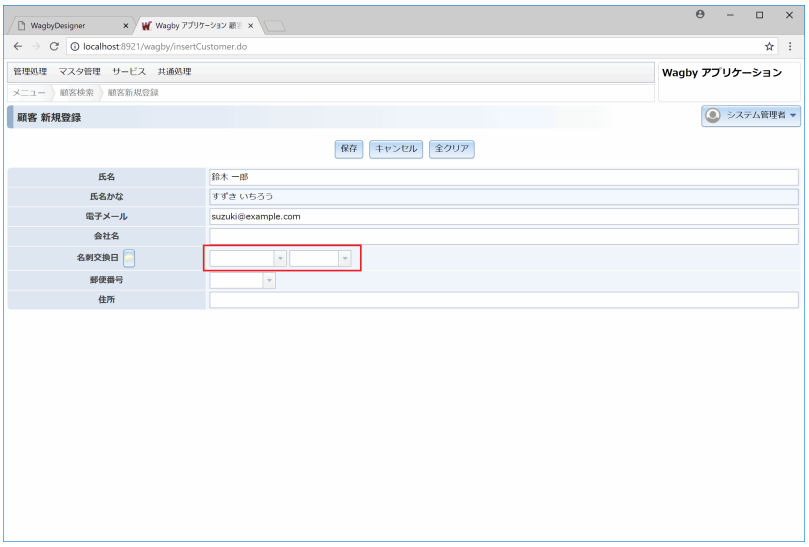


図 日付時刻型項目の入力欄

日付部分はカレンダで入力できます。

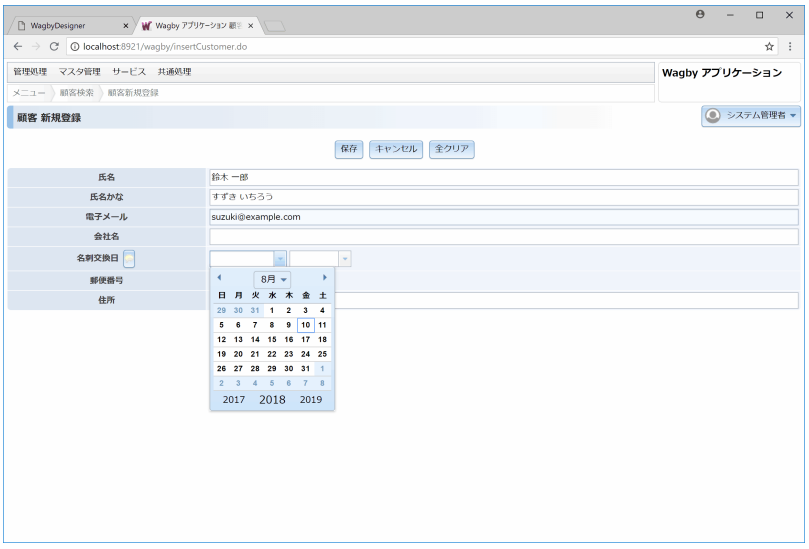


図 日付の入力

時刻部分はリストボックスで入力できます。

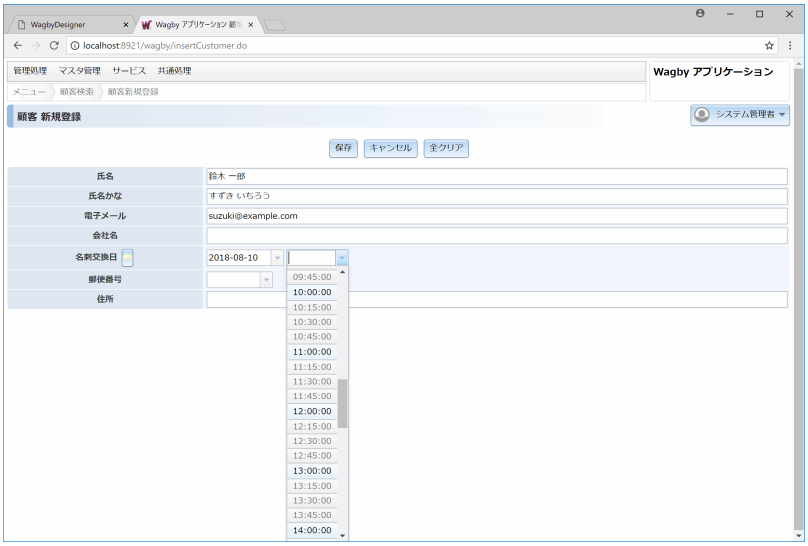


図 時刻の入力

入力は日付と時刻それぞれ別れていますが、データベースへは「日付時刻型」として（一つのデータとして）登録されます。

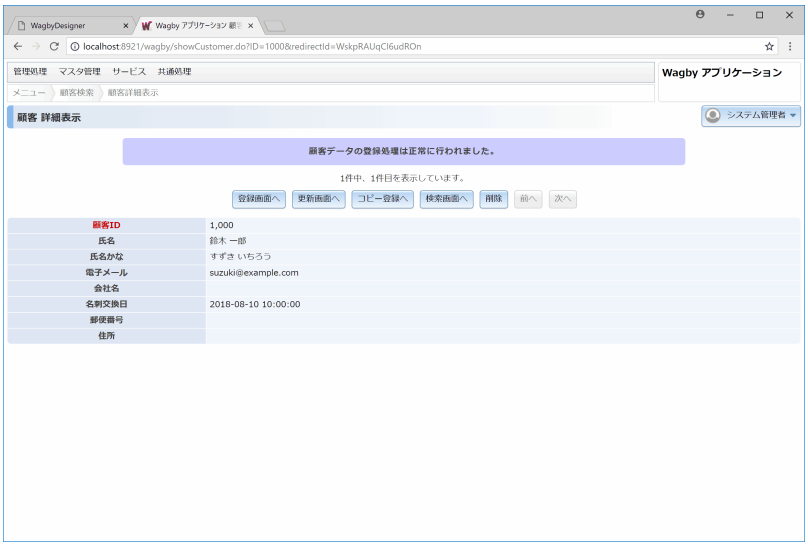


図 データを登録する

テキストフィールドの利用

日付入力方式を空欄とした場合、テキストフィールドで直接、日付時刻を入力します。 テキストフィールドで入力する場合の入力方式は、「出力制御 〉 日付・時刻 〉 フォーマット」に従います。例を示します。

図 日付時刻入力方式のフォーマット設定

yyyy-MM-dd HH:mm:ss

- 年を西暦4桁で入力します。
- 月を 1 から 12 の範囲で入力します。
- 日を 1 から 31 の範囲で入力します。
- 時間は 0 から 23 まで、分と秒は 0 から 59 の範囲で入力できます。
- 年、月、日それぞれ空白で区切って入力します。時間、分、秒をそれぞれ空白またはコロン（":"）文字で区切って入力します。（例 2014 1 2 3 4 5 なら2014年1月2日3時4分5秒となります）

このルールにそぐわない値を入力した場合は、日付時刻として認識されません。保存すると空白になります。

仕様・制約

- R8.1.0 より日付部分と時刻部分の入力それぞれを GUI で操作できるようになりました。(R8.0.5 までは時刻部分はマウスで入力欄にカーソルをあわせ、手動で入力する方式です。)
- 日付部分と時刻部分をテキストフィールド入力とすることはできません。ただし日付、時刻部分それぞれ、キーボードによる直接入力を行うことができます。
- モデル項目詳細定義 > 出力制御 > 詳細画面 > 日付・時刻フォーマットに "yyyy-MM-dd" など日付部分のみを指定すると、時刻部分の入力欄が表示されなくなります。内部では "0:0:0" としてデータベースに登録されます。

文字列 - 郵便番号

郵便番号から住所を求める

例

郵便番号を入力します。フォーカスを移動すると、住所情報が補完されます。

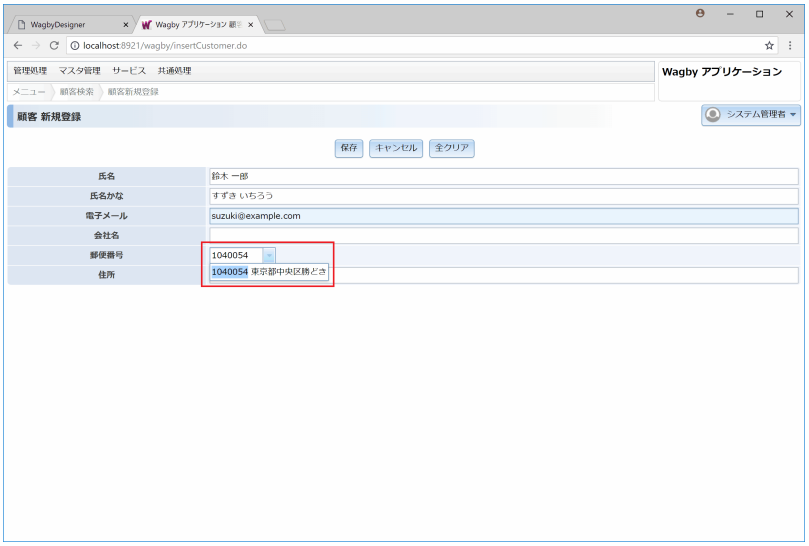


図 郵便番号を入力する

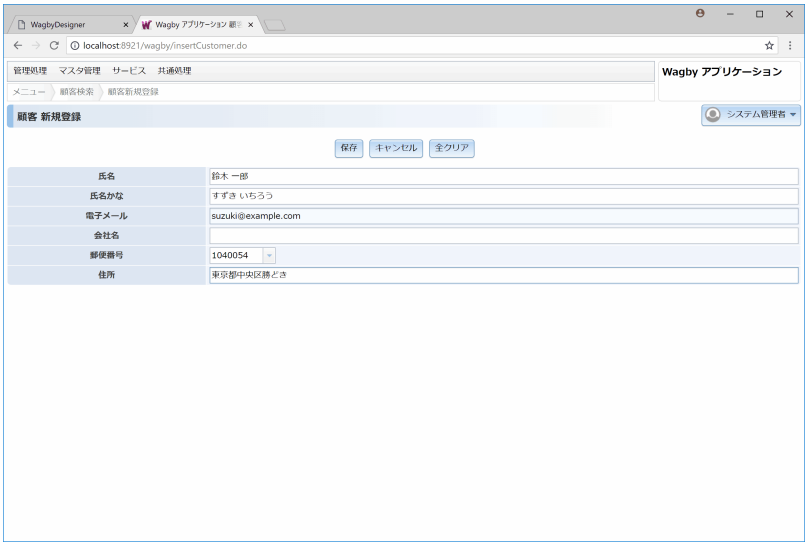


図 住所が設定される

定義方法

文字列型項目の型詳細設定で「郵便番号」を選択します。このときOKボタンは押すことができません。

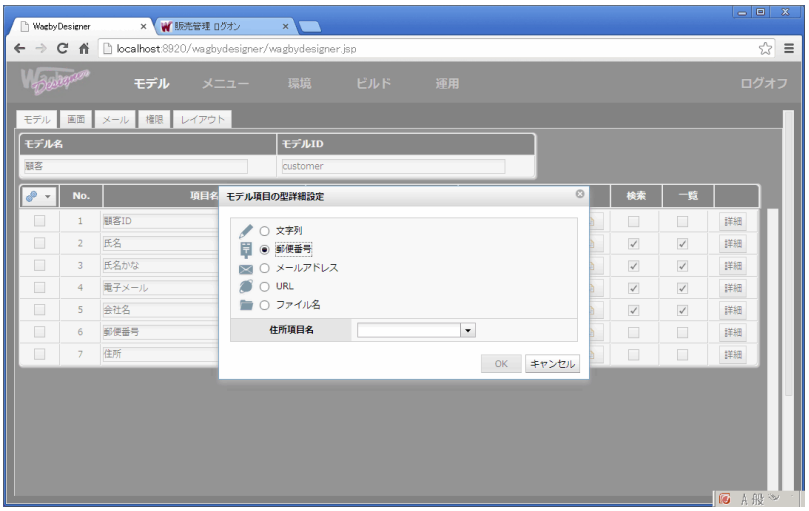


図 郵便番号を選択する

郵便番号に対応する項目を指定します。指定後、OKボタンを押下します。

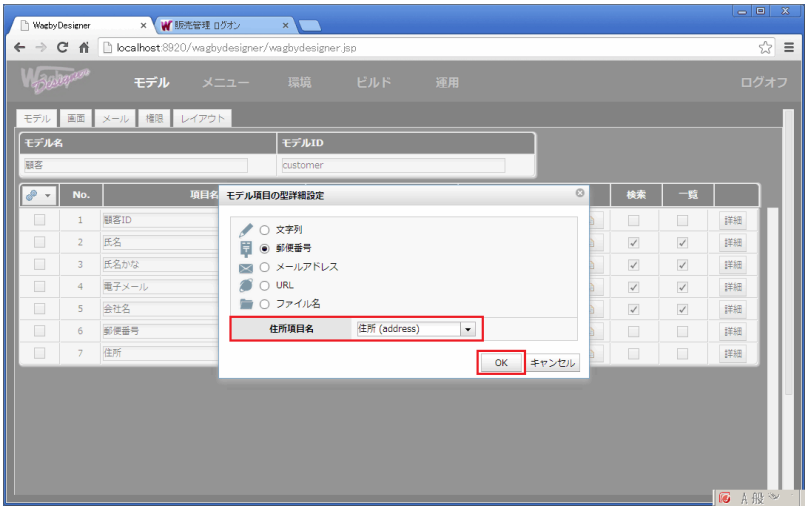


図 郵便番号に対応した住所項目を指定する

住所に該当する項目がない場合、郵便番号型を使うことはできません。

検索に関するルール

「郵便番号」型の項目に検索条件を指定した場合、対応する住所項目も合わせて検索条件に指定する必要があります。

郵便番号型項目に対する正規表現チェック

標準では自動的に正規表現チェックが有効になります。「3桁数値 + ハイフン（なしも可）+ 4桁数値」となります。

郵便番号の候補を出力する

例

郵便番号を3桁以上入力すると、郵便番号と住所の候補についての選択肢が表示されます。 図では、先頭3桁を入力したところです。

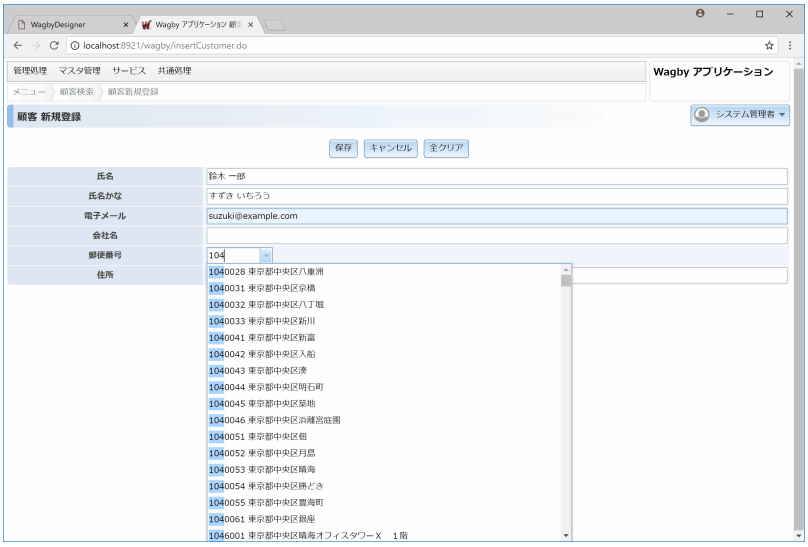


図 郵便番号を3桁入力した

続けて郵便番号を入力すると、候補が絞り込まれます。

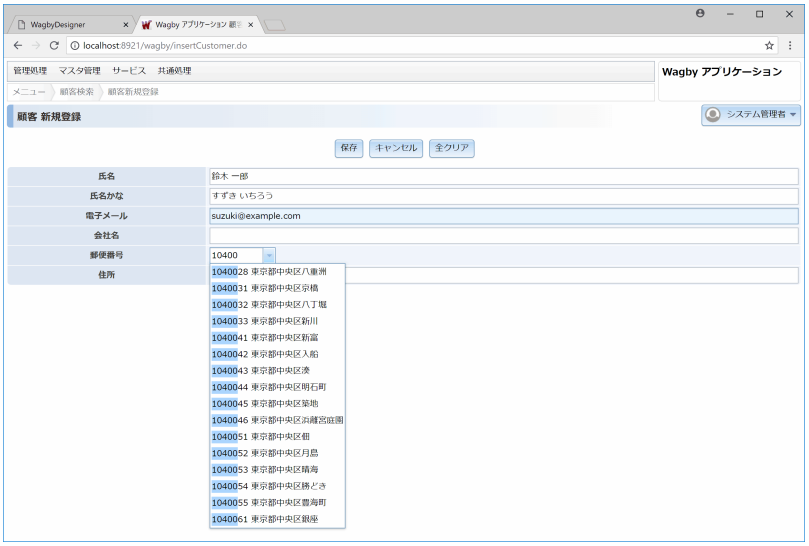


図 候補が絞り込まれる

7桁目を入力することで、郵便番号と住所が確定します。

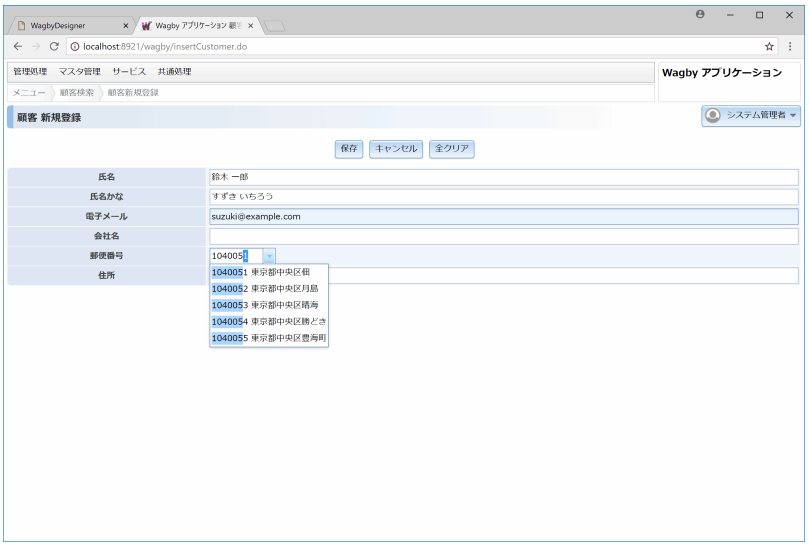


図 郵便番号を7桁まで入力する

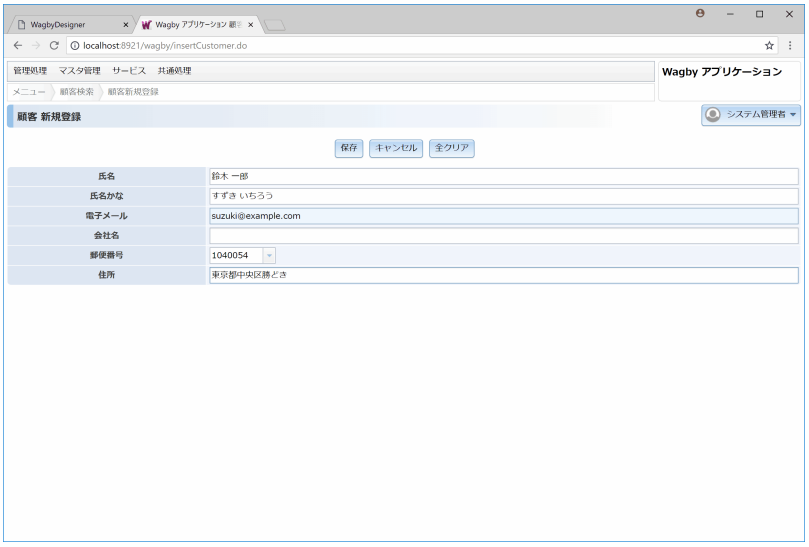


図 住所が確定する

ハイフンの扱い

郵便番号を3桁入力後、ハイフンを入力することもできます。その場合、候補もハイフン付きで用意されます。

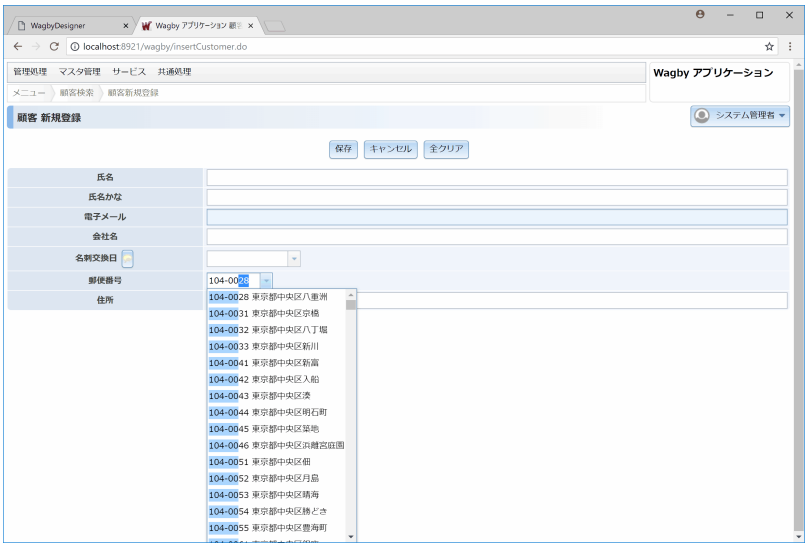


図 ハイフン付きの場合

入力フィルタの手動設定

郵便番号は文字列型の一種であるため、標準では「文字列（一般）」フィルタが選択されます。 このフィルタは数値を半角文字に変換しますが、"ー"（全角ハイフン）は全角のままとなります。[詳細...]

そこで郵便番号の場合、フィルタを「文字列（一般）」から「全角→半角」を選択しなおすことを推奨します。 この変更によってハイフンを半角に統一することができるようになります。

郵便番号辞書

Wagby には株式会社アイビスが提供する「zipcloud」と、日本郵政公社が提供する郵便番号データ（事業所別郵便番号）をベースに生成された郵便番号辞書ファイルが同梱されています。

日本郵政公社が提供する郵便番号には、イレギュラーなデータが含まれています。具体的には、プログラムで解釈不可能な、特別表記です。（例："以下に記載のない住所は xxx とする" など）

株式会社アイビスの「zipcloud」では、このようなイレギュラーデータを可能な限り排除した形で加工されたものを提供しています。（しかし、すべての郵便番号の変換を保証するものではない、ということを予めご了承ください。）

利用している郵便番号辞書

Wagby は出荷時点でもっとも最新の郵便番号辞書を同梱しています。

ファイルの場所

郵便番号辞書の本体は WEB-INF/jmaster フォルダにある zipcode.dat というバイナリファイルです。

最新の郵便番号情報に更新する

最新の情報を使って zipcode.dat を作成することができます。詳細...

仕様・制約

- 郵便番号辞書データはWagbyアプリケーション起動時にメモリに読み込まれます。起動後に郵便番号辞書データを差し替えても、実行中のアプリケーションには差し替えデータは反映されません。反映させるためには、Wagbyアプリケーションを再起動させる必要があります。
- 郵便番号項目を検索条件とした場合、対応する住所項目も合わせて検索条件に指定してください。

文字列 - 住所正規化コンバータとの連携

住所正規化コンバータとは

住所正規化コンバータは、住所の"ゆれ"を補正する製品です。

Wagbyは、住所正規化コンバータAPIを利用する機能を提供しています。画面からの住所入力時にゆれを補正することで、高品質で検索性の高い住所情報を維持することができます。

さらに ANORMPART 関数と組み合わせることで、住所の構成要素（都道府県、市区町村、町域名、小字・丁目...）を取得することができます。

例

例（1）住所から郵便番号を求める

住所を入力後に「住所の同期」ボタンをクリックすると、郵便番号が補完されます。

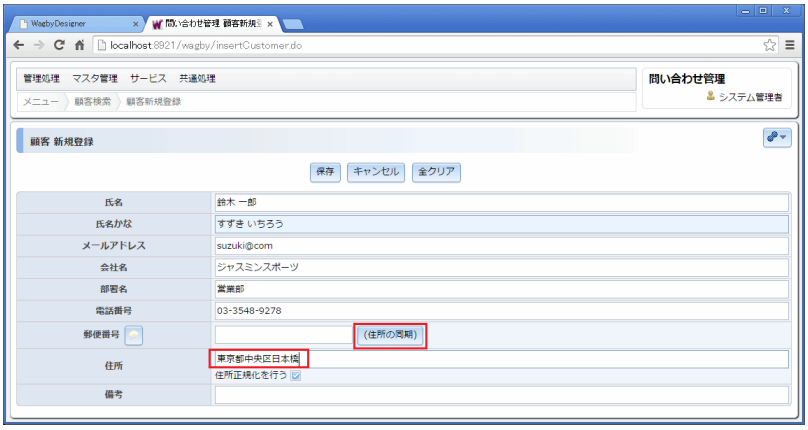


図 住所から郵便番号を補完する(1)

図 住所から郵便番号を補完する(2)

例（2）住所の正規化

住所文字列の表記ゆれを正規化（漢数字表記や異体字など、さまざまな表記の住所を、統一表記に自動的に変換）します。

正規化を行うには、住所入力欄にある「住所正規化を行う」というチェックボックスを有効にします。標準では有効になっています。

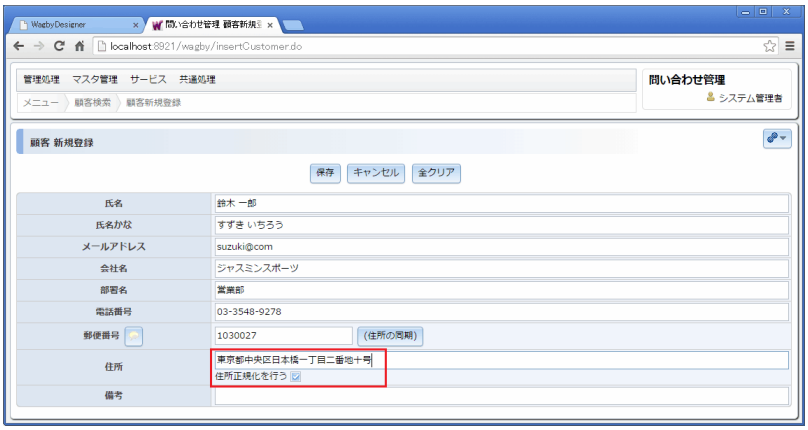


図 住所の正規化を行う(1)

図 住所の正規化を行う(2)

定義方法

「環境 > アプリケーション > 住所正規化コンバータ」を設定します。

「バージョン」は "R5(JSON)" または "R6(JSON)" とします。住所正規化コンバータR5/R6に対応しています。

URLには、住所正規化コンバータのサービスを提供するWebサーバを記載します。ここでは同一マシン(localhost)を使うものとして、次のように設定しています。

```
http://localhost:8080/anorm/v1/anorm
```

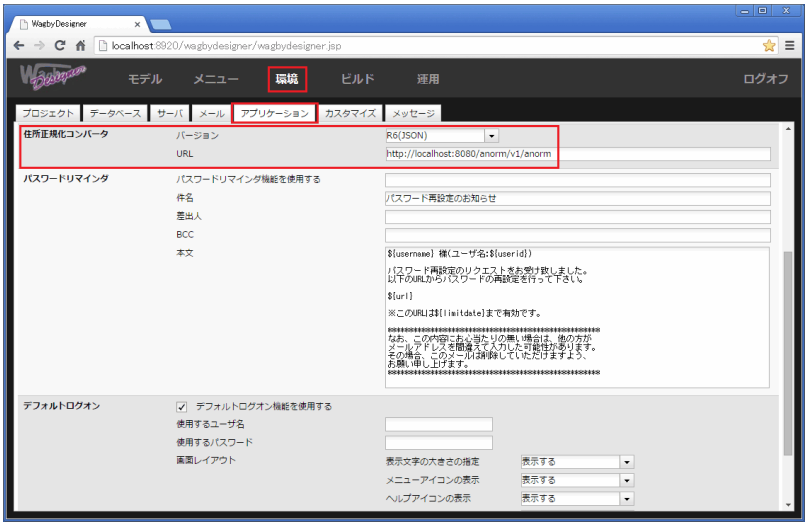


図 住所正規化コンバータを使う設定

次節より、住所正規化コンバータサーバの定義方法を説明します。

住所正規化コンバータサーバの設定

住所正規化コンバータサーバモジュールのインストール方法をまとめています。詳細...

文字列 - メールアドレス

例と定義方法

文字列型項目の属性にメールアドレスを指定すると、詳細表示画面および一覧表示画面がリンク表示となります。 クリックするとメールソフトが起動します。

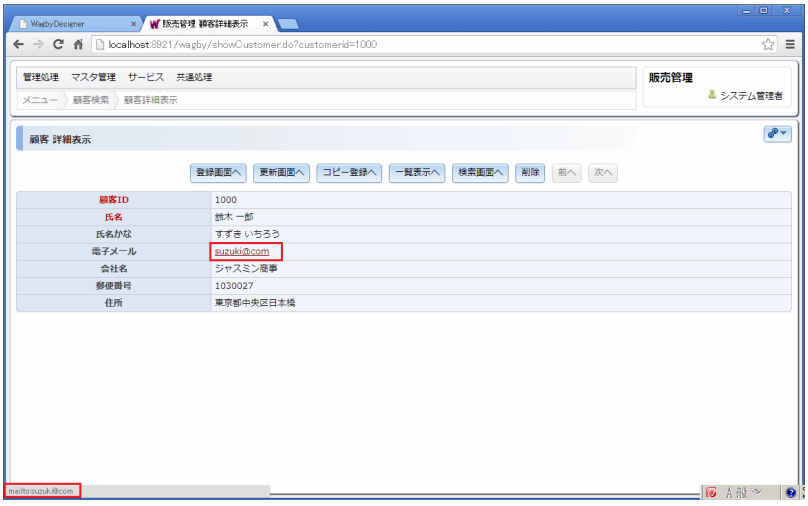


図 メールアドレスを指定した項目

リンクは「mailto:値」となります。

定義方法

文字列型項目の型詳細設定で「メールアドレス」を選択します。

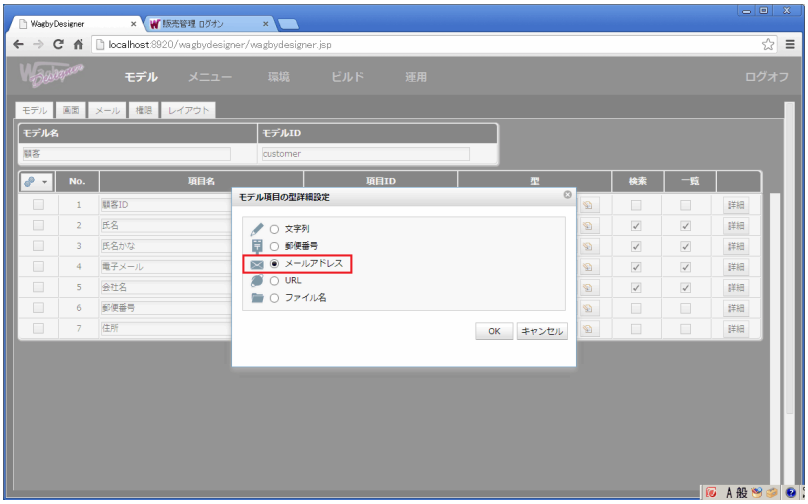


図 メールアドレスを選択する

文字列 - URL

例と定義方法

文字列型項目の属性にURLを指定すると、登録・更新画面で「http://」という文字列が自動的に設定されます。

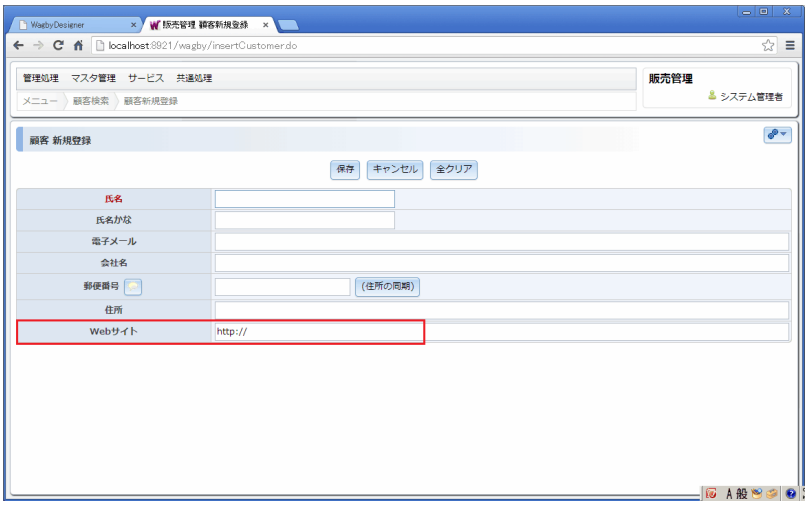


図 URLを指定した項目の登録画面

詳細表示画面および一覧表示画面がリンク表示となります。　クリックすると、そのURLが示すサイトが開きます。

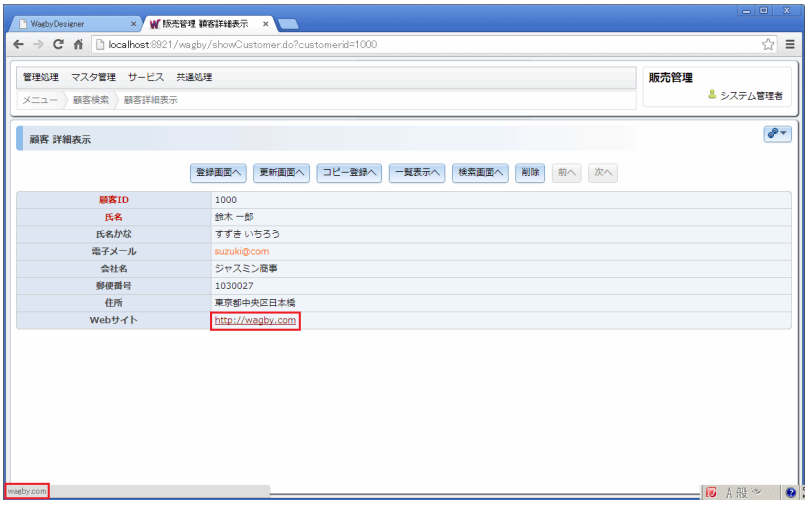


図 リンクが有効になった詳細表示画面

定義方法

文字列型項目の型詳細設定で「URL」を選択します。

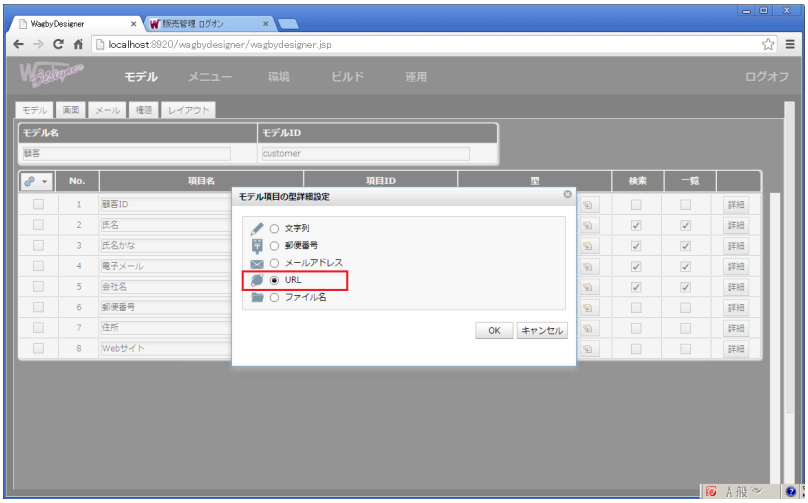


図 URLを指定する

(19) (20)

図 初期値を空にする

リンク先画像を表示する

例

(19) 新規登録画面を開いたときの初期値 "http://" 文字列を変更することもできます。モデル項目詳細定義ダイアログから "初期値>登録画面表示時" の設定に任意の URL を記述してください。

(20) 新規登録画面を開いたときの初期値を空とすることもできます。モデル項目詳細定義ダイアログから "初期値>登録画面表示時" に "自動計算式" を選択し、値の部分を "" としてください。

項目が URL 型でリンク先が画像ファイルの場合、画像を表示することができます。

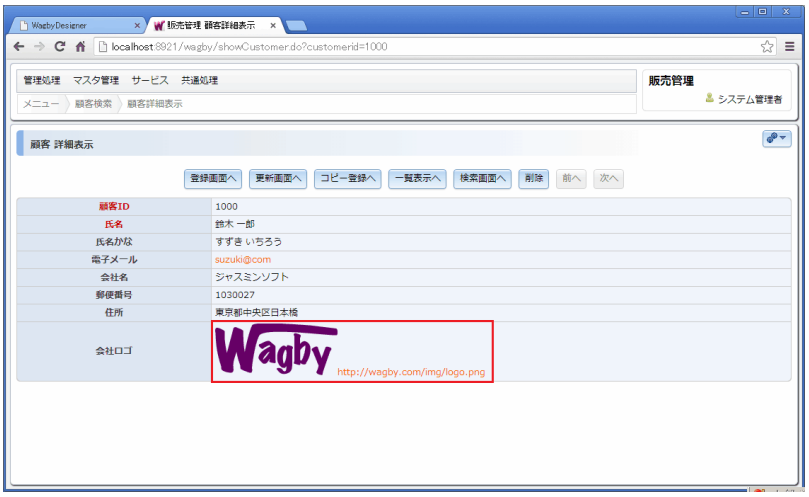


図 詳細画面での画像表示

図 一覧画面での画像表示

定義方法

出力制御タブの詳細画面、一覧画面を選択します。「URL型のリンク先画像を表示する」のチェックを有効にします。（標準ではこのチェックは無効になっています。）

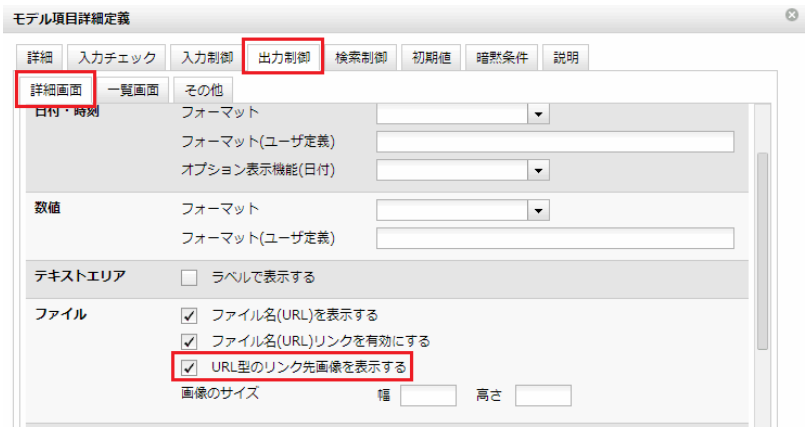


図 詳細画面での定義

図 一覧画面での定義

リンク先画像の幅と高さを指定する

例

リンク先画像の表示幅と高さを指定することができます。図, 図はそれぞれ画像幅を64ピクセルとした例です。

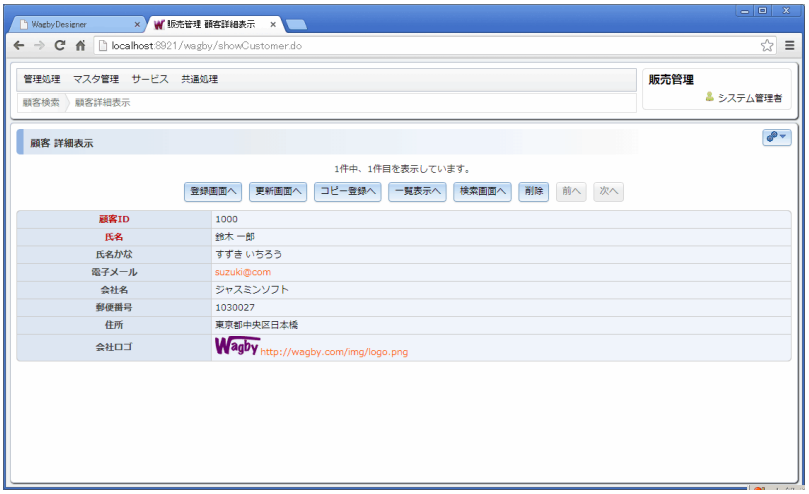


図 詳細画面での画像サイズを幅64ピクセルとした

図 一覧画面での画像サイズを幅64ピクセルとした

定義方法

出力制御タブの詳細画面、一覧画面を選択します。「URL型のリンク先画像を表示する」のチェックを有効とし、さらに「画像のサイズ」を指定します。単位はピクセルです。幅と高さの両方を指定できますが、一方を指定した場合、もう一方は比率を維持して自動調整されます。

図 詳細画面の画像サイズ幅を指定する

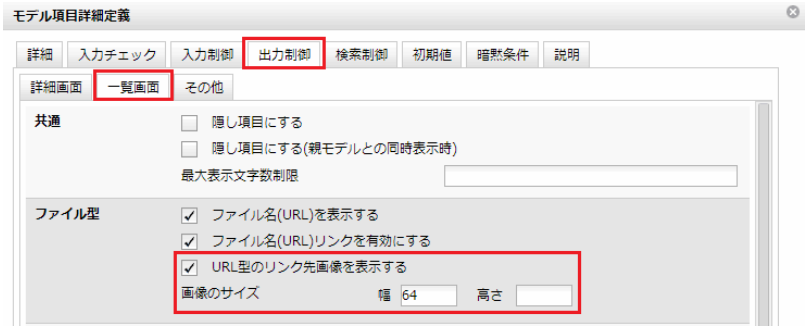


図 一覧画面で画像サイズ幅を指定する

画像表示時にURLを表示しない

例

URLを非表示にすることができます。

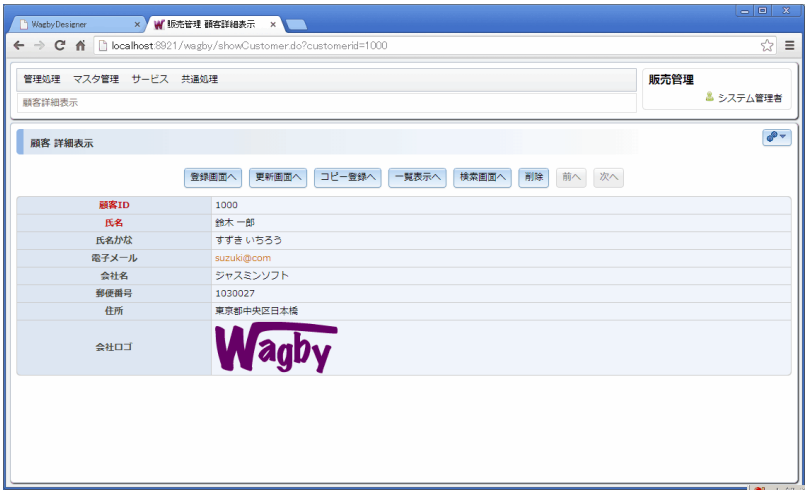


図 詳細画面でのURLを非表示とした

図 一覧画面でのURLを非表示とした

定義方法

モデル項目詳細定義の出力制御タブを選択します。詳細画面と一覧画面のタブにある「ファイル名(URL)を表示する」のチェックをはずします。（標準ではチェックされています。）

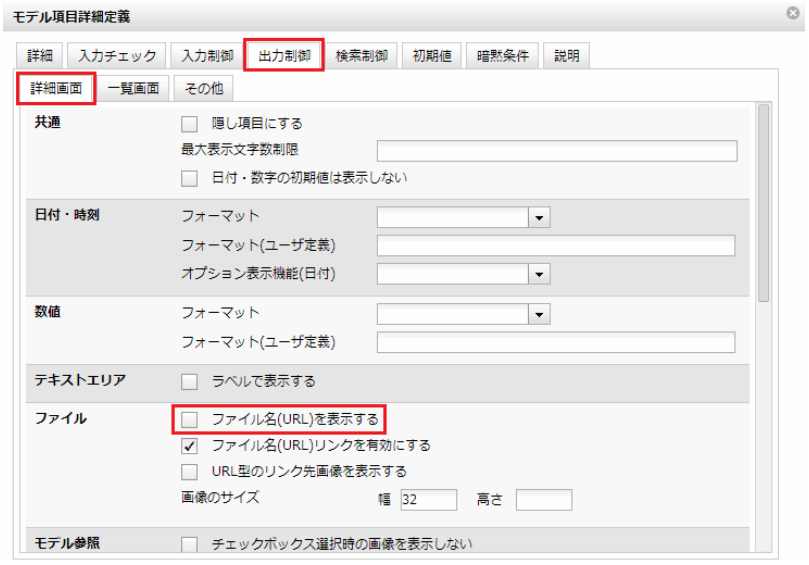


図 ファイル名(URL)を表示する、のチェックをはずす

リンクを無効にする

URLまたは画像をクリックすると、そのページを開くという機能を無効にすることができます。　図では、アイコンをクリックしてもそのページへ遷移しません。

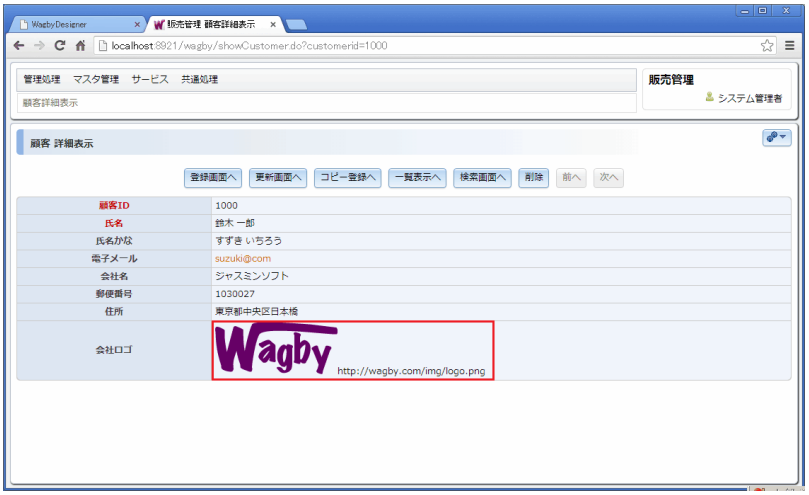


図 リンクが無効になっている

定義方法

モデル項目詳細定義の出力制御タブを選択します。詳細画面と一覧画面のタブにある「ファイル名(URL)リンクを有効にする」のチェックをはずします。（標準ではチェックされています。）

図 詳細画面の定義

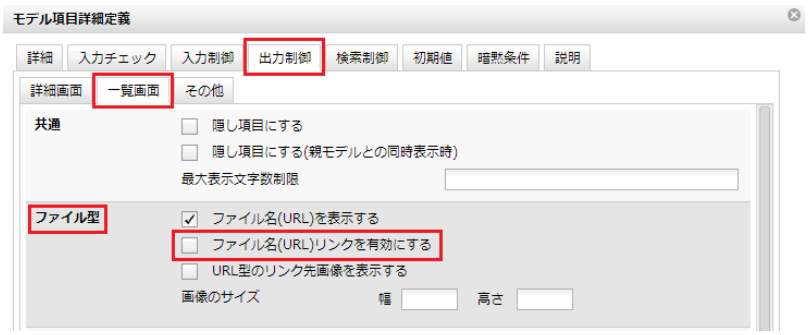


図 一覧画面の定義

応用 GoogleMapsとの連携

実行例

住所情報の横「地図で探す」という文字（実際にはGoogle Mapsへのリンク）を用意してみます。 はじめに郵便番号から住所を確定させます。

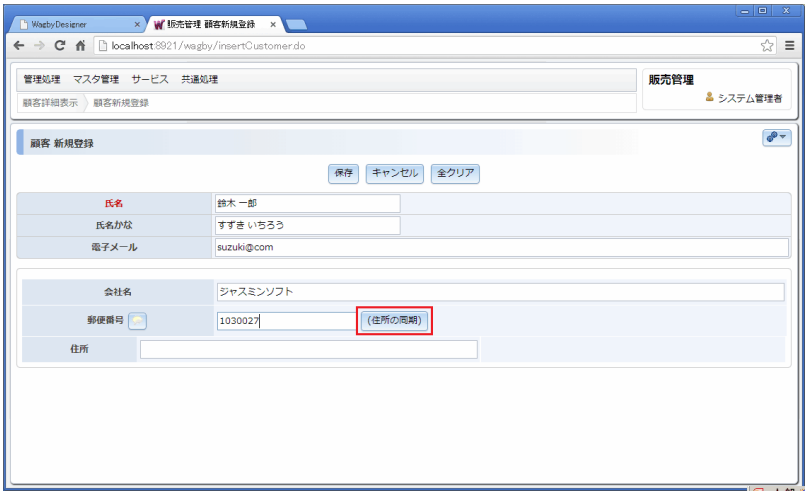


図 郵便番号から住所を求める

住所入力欄の横に「地図で探す」という文字が表示されます。これは実際にはリンクになっています。

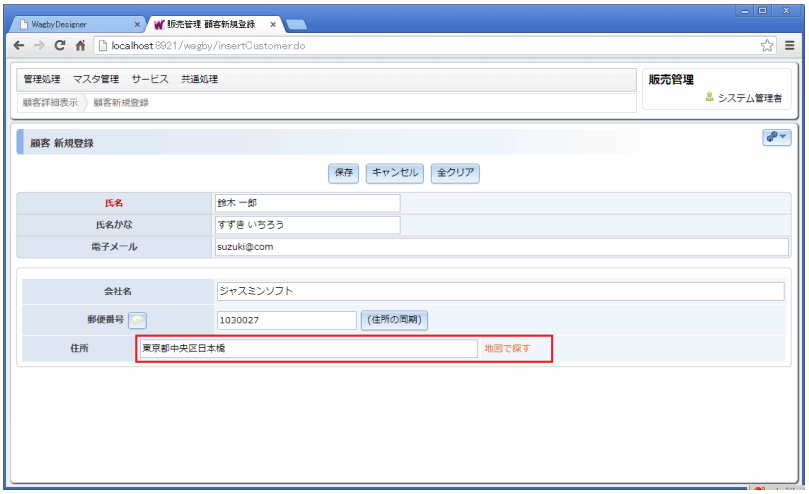


図 住所文字列の入力

「地図で探す」の文字上にマウスカーソルをあてると、次のリンクになっていることがわかります。

```
http://maps.google.co.jp/maps?q=東京都中央区日本橋
```

マウスでクリックすると、Google Maps が開きます。

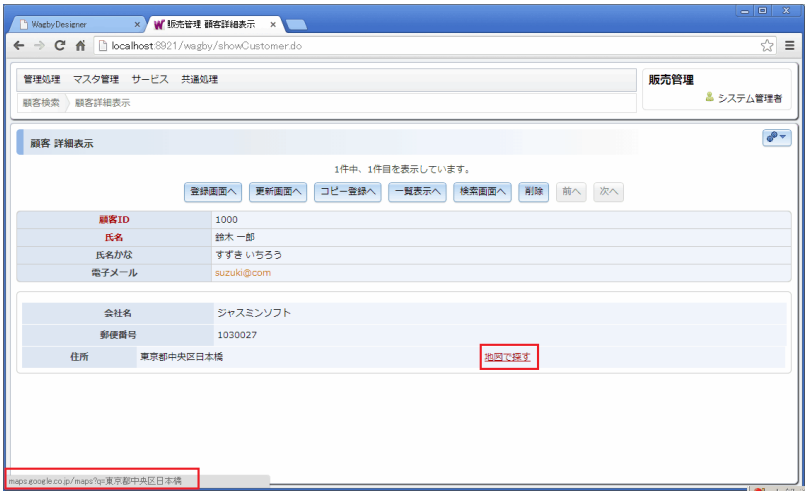


図 GoogleMapsへのリンク

設定方法

住所とURLに関する3つの項目を用意します。 「地図をみる」項目は文字列型で、URLとします。

項目名(日本語)	項目名(英語)	項目の型	隠し（入力）	隠し（出力）
住所	address	文字列		
地図をみる	map	文字列 - URL		
地図をみる表示文字列	mapdisp	文字列	○	○

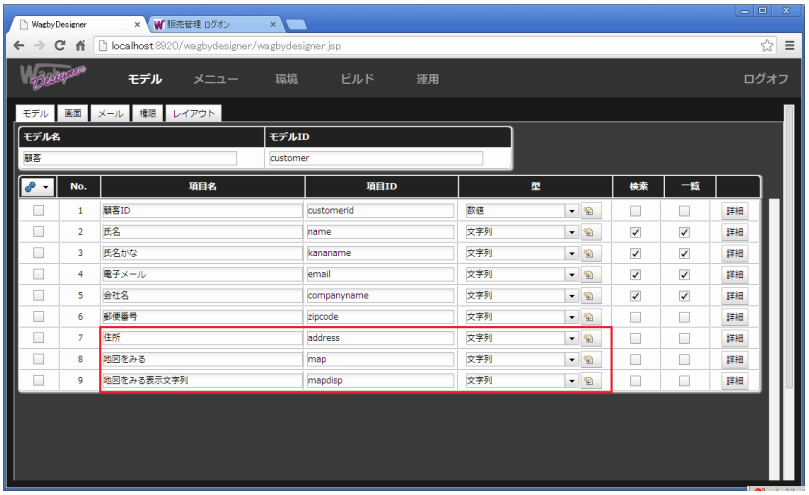


図 項目の定義

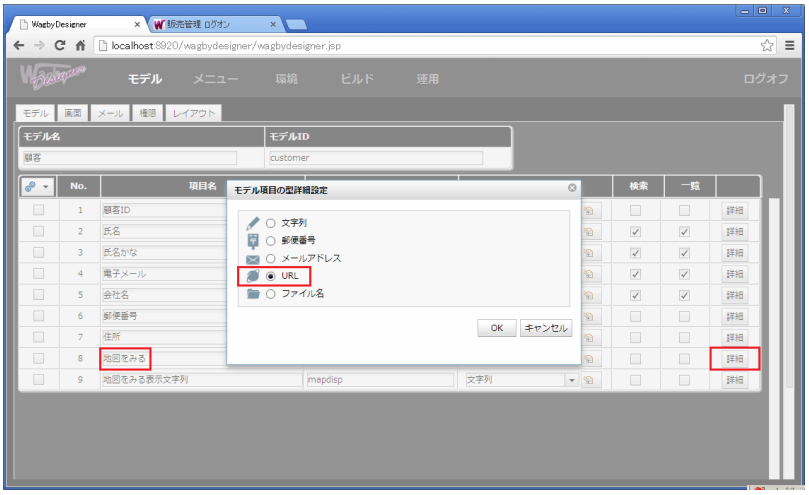


図 「地図をみる」項目はURLとする

「地図をみる」項目の詳細

この項目は計算式で求めます。住所項目が空でなければGoogle Mapsへのリンクを用意します。パラメータ(q)への住所文字列を加えます。住所項目が未入力（空）であれば空白とします。

```
IF(
  NOT(ISBLANK(${address})),
  CONCATENATE("http://maps.google.co.jp/maps?q=",${address}),
  ""
)
```

データベースに保存する必要はありませんので「データベースの設定>値をデータベースに保存する」のチェックを解除します。

「URLの詳細」設定欄で、リンク表示に利用する項目名として "mapdisp" を指定します。（この項目は隠し設定をしていることがポイントです。）

図 「地図をみる」項目の詳細設定(1)

図 「地図をみる」項目の詳細設定(2)

別画面で開くときのtarget属性

生成される HTML の anchor 要素の target 属性に反映される値を指定します。空白時は "_blank" という値がセットされるため、クリックするとブラウザに新しいタブが用意されます。上の例では、その新しいタブに Google Map が表示されます。

[Tips] ここに "false" と記述すると、target 属性自体が出力されません。そのためクリックすると（その画面が）リンク先に書き換わります。

「住所」項目とで並べて表示するため、ラベルを表示しない設定にします。

図 ラベルを表示しない

「地図をみる表示文字列」の詳細

この項目の値は、上記「地図をみる」のアンカー部として利用されます。

```
IF(NOT(ISBLANK(${address})), "地図で探す", "")
```

データベースに保存する必要はありませんので「データベースの設定>値をデータベースに保存する」のチェックを解除します。

図 「地図をみる表示文字列」の詳細設定

この項目は入力時ならびに出力時の双方で隠し項目とします。

図 隠し項目の設定（入力制御）

図 隠し項目の設定（出力制御）

レイアウト

最後にレイアウトを定義します。会社関連の項目を隠しパネル *P1 に、さらに住所と地図へのリンク文字列は隠しパネル *P2 にまとめます。

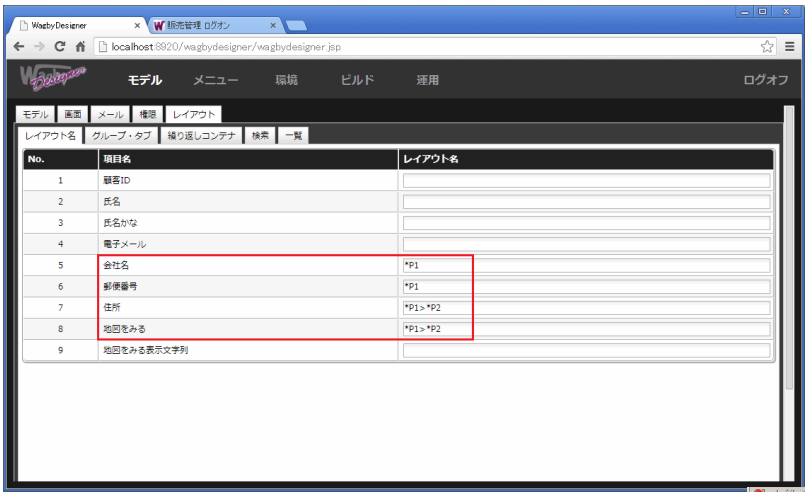


図 レイアウト名を定義する

*P1>*P2の部分はグループの表示方法を指定せず、横並びとします。「地図をみる」項目はラベル部分も非表示なので、連続して並ぶようになります。

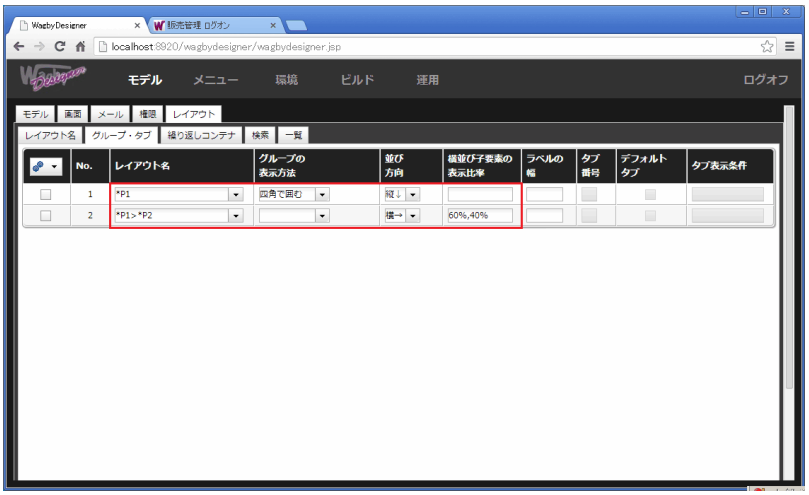


図 並び方向を指定する

文字列 - ファイル

例と定義方法

例

ファイル項目を指定すると図のような入力欄が用意されます。

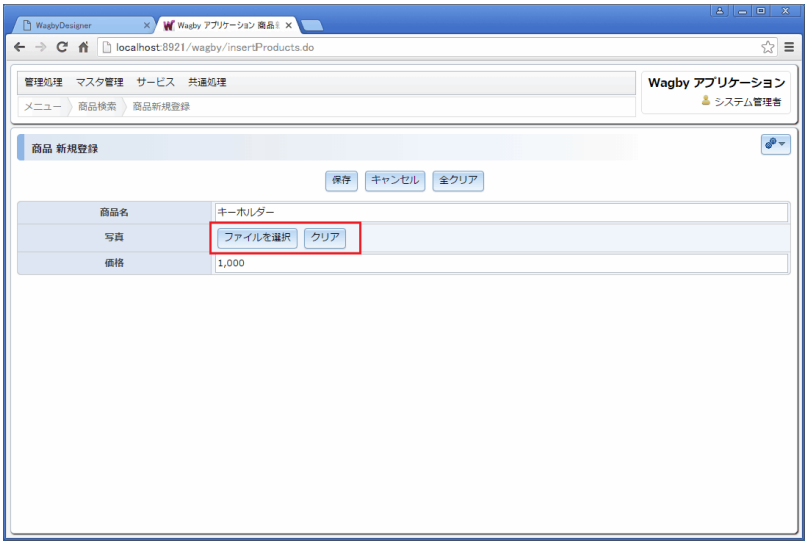


図 ファイル項目の入力欄

「ファイルを選択」ボタンへファイルをドラッグ&ドロップして登録することができます。

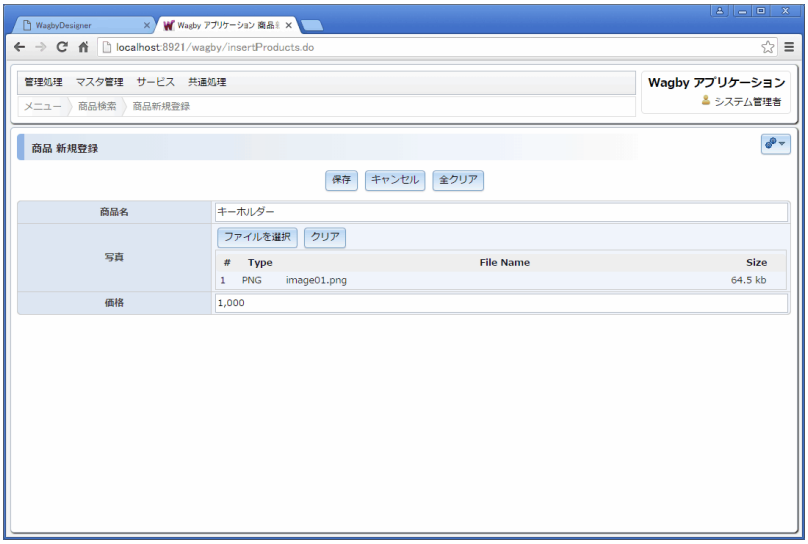


図 画像をドラッグ&ドロップした

詳細表示画面では図のようにイメージが表示されます。

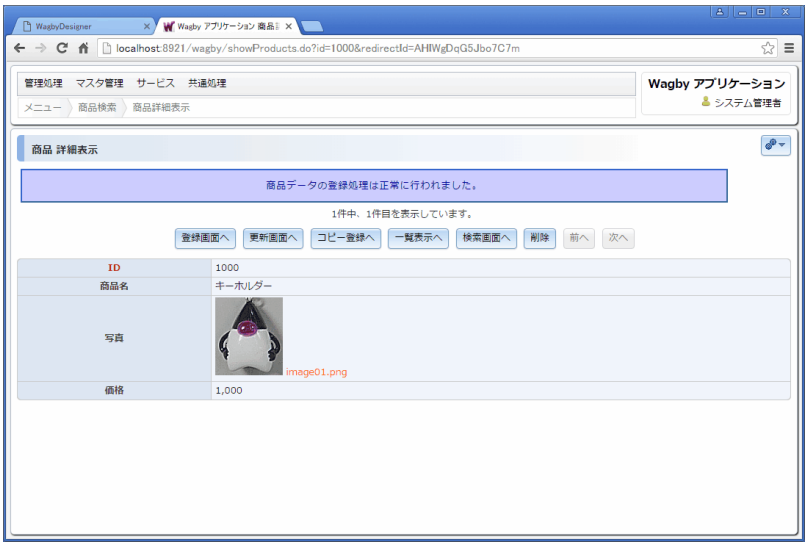


図 画像を登録した

WordやExcelといった非画像ファイルを登録することもできます。詳細表示画面では図のようにアイコンが表示されます。

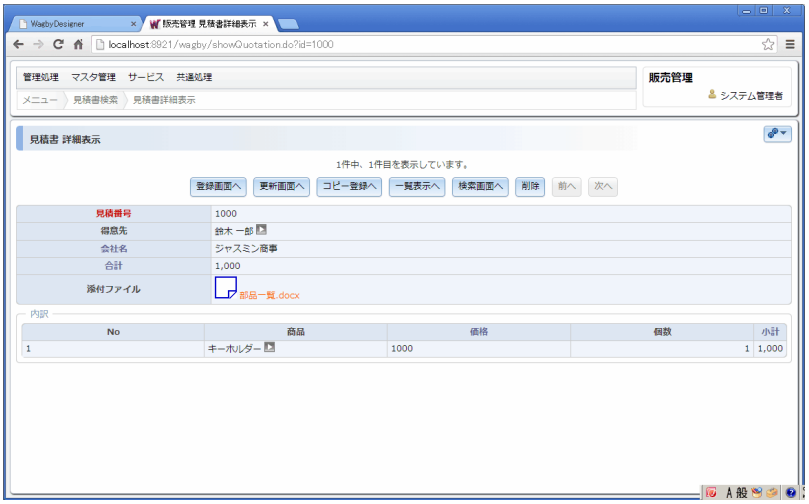


図 Word文書を登録した

アイコンをクリックすると、そのファイルをダウンロードすることができます。

定義方法

文字列型項目の型詳細設定で「ファイル型」を選択します。

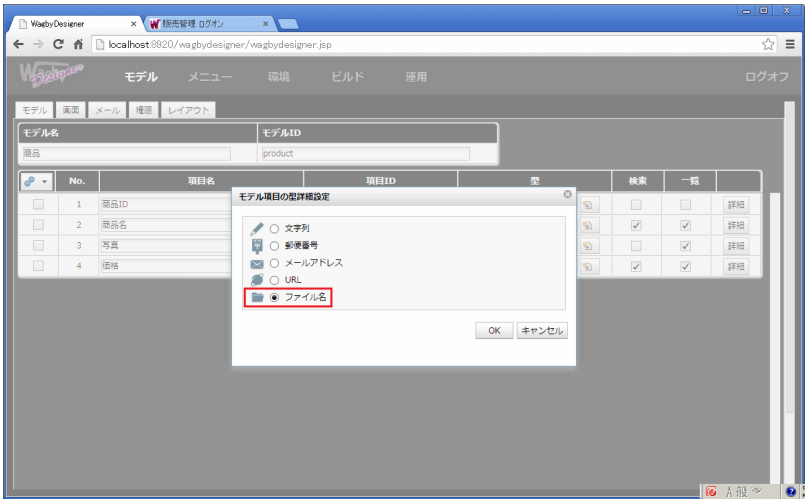


図 ファイル型の指定

アップロードできるファイルに制約を設ける

ファイルサイズを指定する

標準ではアップロードするファイルサイズの制限はありません。設定により、ファイルの最小サイズおよび最大サイズを指定することができます。単位は「バイト」です。

(21)

(21) 巨大なファイルをアップロードすると時間がかかります。最大サイズを設定することで、無制限のアップロードを防ぐことができます。

図 ファイルサイズを1メガバイトから100メガバイトの範囲とする

指定サイズに満たない、または指定サイズを超えた場合は図のようなエラーメッセージが表示されます。

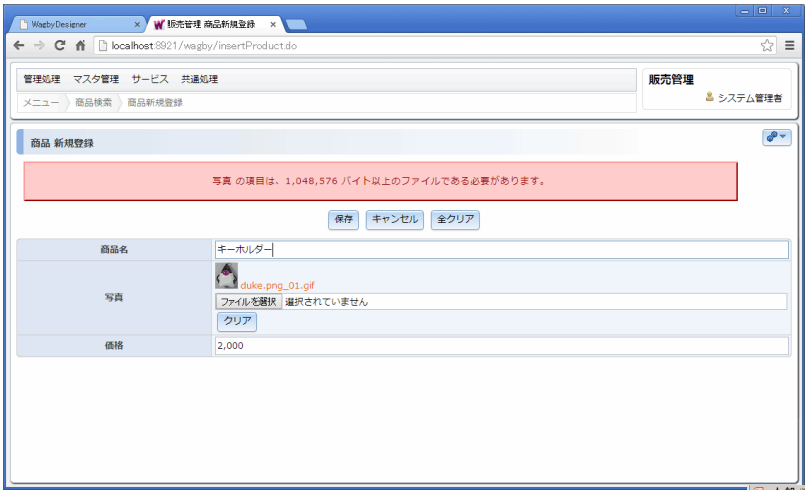


図 サイズエラー

サイズを指定する場合、"K" や "M" での指定も行えます。（大文字小文字は区別しません。） 1Kは1024バイト、1Mは1024Kバイトです。

サイズ指定の数値は、整数型のみ使用できます。小数点は使用できません。

ファイル拡張子を指定する

アップロードするファイルの拡張子を指定することができます。複数個の場合はコンマ「,」で区切ります。

図 ファイル拡張子がpngとjpgのみを許容する

許容されるファイル拡張子でない場合、図のようなエラーメッセージが表示されます。

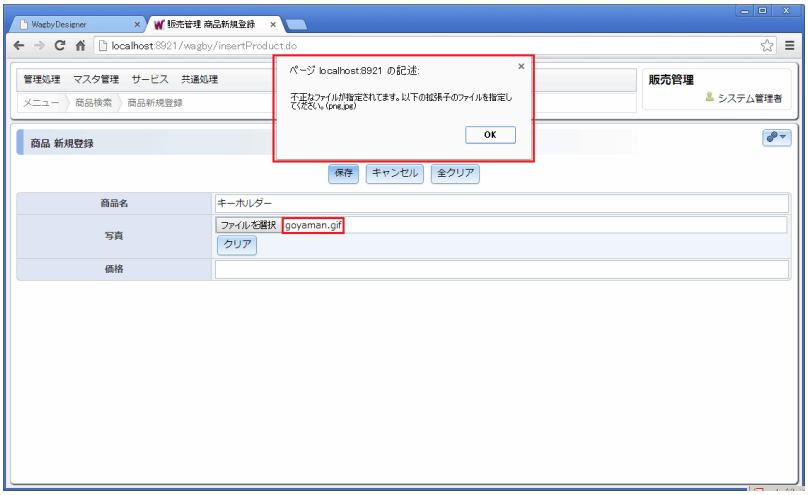


図 拡張子エラー

上記例では PNG と JPG も許容されます。大文字小文字は区別しません。

アップロードの保存先フォルダを変更する

物理ファイルの保存先をモデル毎に指定することができます。

ダウンロード時のログ

ファイル型項目で登録されたファイルをクリックしてダウンロードしたとき、ログが出力されます。

```
2016-xx-xx 00:01:02 [INFO wagbyDownload2 _jspService] (admin) Download test.xls
```

- 画像ファイルの場合、ログは出力されません。（ファイル拡張子が jpg/jpeg/png/gif/bmp の場合、画像ファイルと解釈します。）
- ファイルをクリックすると、ダウンロードを行うかどうかを確認するためのダイアログが表示されることがあります。その時点でログは出力されています。そのため（ダウンロードを）キャンセルしても、ログ上ではダウンロードした、と記録されます。
- ダウンロードではなく、指定したアプリケーションで開いた場合でも、ログ上ではダウンロードした、と記録されます。

テキストファイルの文字エンコーディング指定

MIME タイプのタイプ名が text の場合（テキストファイルやHTMLファイルなど）、文字エンコーディングは "utf-8" が標準となります。

「環境 > アプリケーション > 初期パラメータ > ダウンロードファイルの文字エンコーディング」を指定して、標準の文字エンコーディングを変更することもできます。これはプロジェクト全体で統一されます。

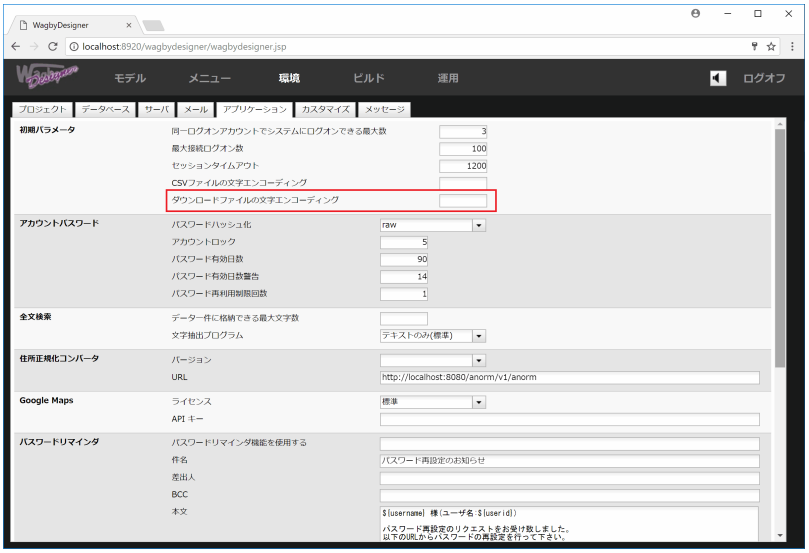


図 ダウンロードファイルの文字エンコーディング

ファイルダウンロードのセキュリティ

ファイル型項目では、Webフォームからアップロードされた物理ファイルは OS のフォルダ内に管理されます。Webブラウザ（Wagby アプリケーション）でリンクをクリックするとファイルをダウンロードすることができます。

これはセキュリティ上の懸念はないでしょうか。ファイルの格納場所（フォルダのパス）を第三者に知られてしまうと不正なダウンロードにつながる可能性を心配されるかも知れません。結論からいうと、このようなことが生じないよう、すでに対策済みです。

- ファイルダウンロードのURLに含まれるパスは一時的なものです。具体的には、ログオンしているユーザのログオン中のみ有効となっています。このため、このURLをコピーして別ユーザが再利用することはできないようになっています。（仮にそのような操作をしても無視されます。）
- いったんログアウトして再ログオンすると、パスの表記が変わります。
- このパスは実ファイルのパスを指していません。実ファイルのありかを知ることはできないようになっています。

検索

ファイル名の検索

「ファイル名」型の項目を検索条件とした場合、ファイル名が検索対象となります。文字列型として扱われるため部分一致、前方一致といった検索方法を指定することもできます。

ファイルの内容を検索する

Word,Excel,PowerPoint,PDFといったファイルの内容を検索することもできます。　詳細は"全文検索"をお読みください。

繰り返しコンテナ内のファイル項目

繰り返しコンテナ内のファイル項目は、複数ファイルを一括でドラッグ&ドロップして登録することができます。ボタン名が「複数ファイルを選択」となっています。

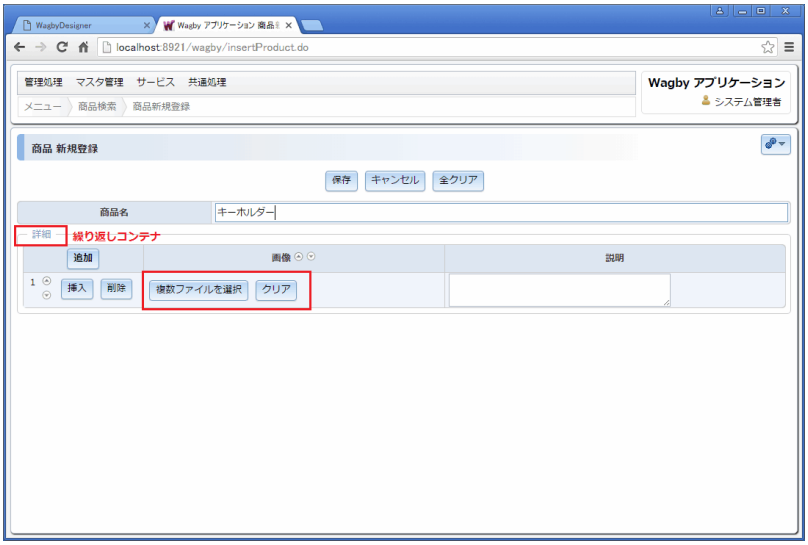


図 複数ファイルを選択するボタンが用意される

2つのファイルをドラッグ&ドロップした状態です。

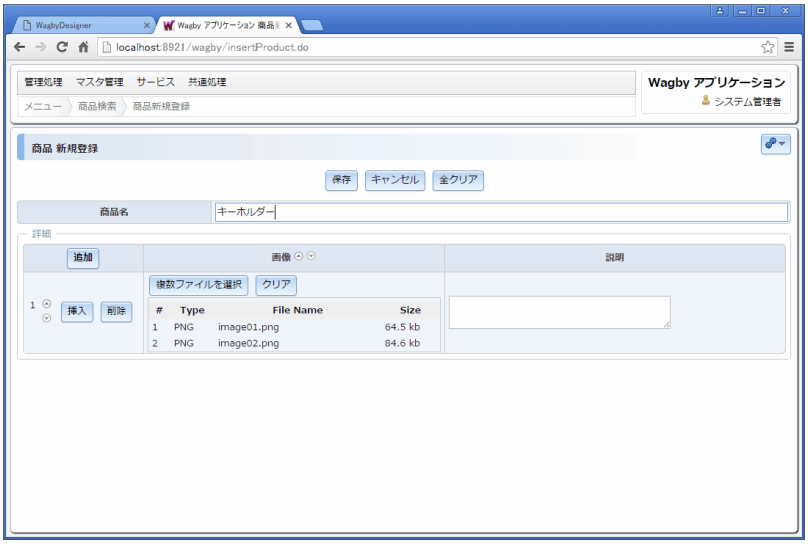


図 2つのファイルをドラッグ&ドロップした

保存ボタンを押下すると、繰り返しコンテナ部分に「2レコード」が登録されます。つまり、1ファイル毎に1レコードとなって登録されます。

図 1ファイル毎に1レコードとなって登録された

仕様

Wagbyのファイル型は、次のような動作になります。

基本

- Word,Excel,PowerPoint,PDF,CADデータ,画像など、どのようなファイルも登録できます。
- 登録できるファイルサイズの上限はありません。
- 画像ファイル（拡張子 gif, jpg, png など）を登録すると、画面上で閲覧できるようになります。表示サイズを指定することもできます。
- その他のファイル（拡張子 doc, xls, zip など）を登録すると、画面上ではアイコンとして表示されます。アイコンをクリックすることでデータをダウンロードすることができます。
- 新規登録・更新画面の処理は、Web ブラウザが提供するユーザーインタフェースに従います。一般的には、ダイアログボックスを使ってファイルを選択し、送信（アップロード）する方式となっています。
- 画面上に用意された「クリア」ボタンをクリック（クリック後、入力欄が空の状態で保存）することでアップロードしたファイル名を消去することができます。
- 日本語のファイル名にも対応しています。
- ファイル名にスラッシュ "/"を利用することはできません。
- ファイル型項目をダウンロードして保存する場合、IE/Edge は保存ダイアログに元のファイル名が反映されません。代案として、詳細表示画面に用意されたファイル名リンクを右クリックして「対象をファイルに保存」を選択する操作を行なってください。

ドラッグ&ドロップの動作

ブラウザによって挙動が異なります。

ブラウザ	複数ファイルアップロード	ドラッグ&ドロップ操作	バックグラウンドアップロード
Google ChromeFirefoxIE 11	○	○	○
Edge	○	-	○

ウィルスチェックソフトへの除外フォルダ指定 !--N20190225_155832-->

ファイルサイズが大きい場合、ダウンロード時にウィルススキャンチェックに時間がかかることがあります。圧縮されたファイルであれば展開後にチェックするため、より時間がかかります。

これが問題になる場合、アンチウィルスソフトに、ダウンロードファイルを格納するフォルダを除外する指定を行うという回避策があります。 例えば Windows ディフェンダーをご利用の場合は、以下のページを参考にしてください。

Windows　　セキュリティに除外を追加する　　https://support.microsoft.com/ja-jp/help/4028485/windows-10-add-an-exclusion-to-windows-security

(22)

制約

・　ファイル型項目のダウンロード時、ファイル名に日本語を含んだファイルをダウンロードするとFirefox　ではファイル名を正しく扱うことができません。（この制約は　R8.3.1　で改善する計画です。）

保存ファイル名の扱い

アップロードしたファイルの実体は　upload_dir　フォルダに格納されます。ファイル名は内部で機械的に変更され、拡張子は ".tmp"となります。

「画面 ＞ その他 ＞ ファイル名型項目の制御」内の「サーバに保存される実ファイル名」を指定することで、このルールを次の3種類から選択することができます。

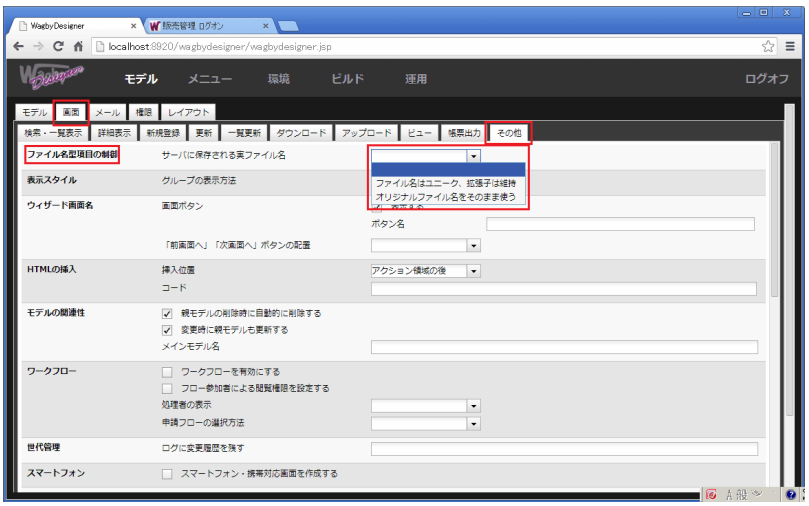


図　ファイル保存の設定

（空白時）

ファイル名は機械的に付与されます。拡張子は　.tmp　となります。（標準）

ファイル名はユニーク、拡張子は維持

ファイル名は機械的に付与されますが、拡張子はアップロードしたファイルを踏襲します。

オリジナルファイル名をそのまま使う

ファイル名、拡張子ともアップロードしたファイルと同じものを使います。

ファイルの削除ポリシー

ファイルの削除処理を行なった場合、データベースからファイル名の情報は削除されますが、ファイルの実体は残っています。ファイル名にオリジナルファイル名をそのまま使う設定を行っていた場合、同名のファイルをアップロードすることで上書きされます。

ファイルの実体を削除する方法を別途、用意しています。

画像の幅、高さを指定する

例

(22)　ご注意ください。この対応はセキュリティレベルを低下させます。社内のネットワークポリシーを規定している担当者と相談することを推奨します。

詳細表示画面と一覧表示画面でそれぞれ、画像の幅と高さを指定することができます。

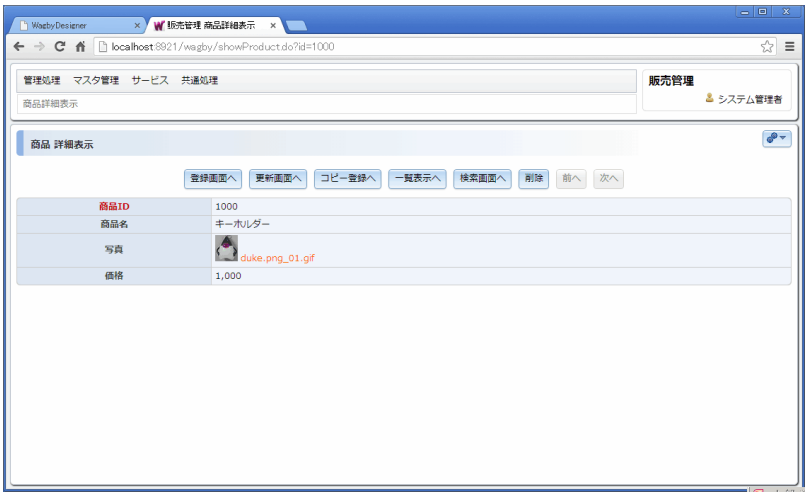


図 詳細表示画面の画像サイズを幅32ピクセルとした

図 一覧表示画面の画像サイズを幅32ピクセルとした

定義方法

モデル項目詳細定義の出力制御タブを選択します。詳細画面と一覧画面のタブにある「画像のサイズ」を指定します。単位はピクセルです。幅と高さの両方を指定できますが、一方を指定した場合、もう一方は比率を維持して自動調整されます。

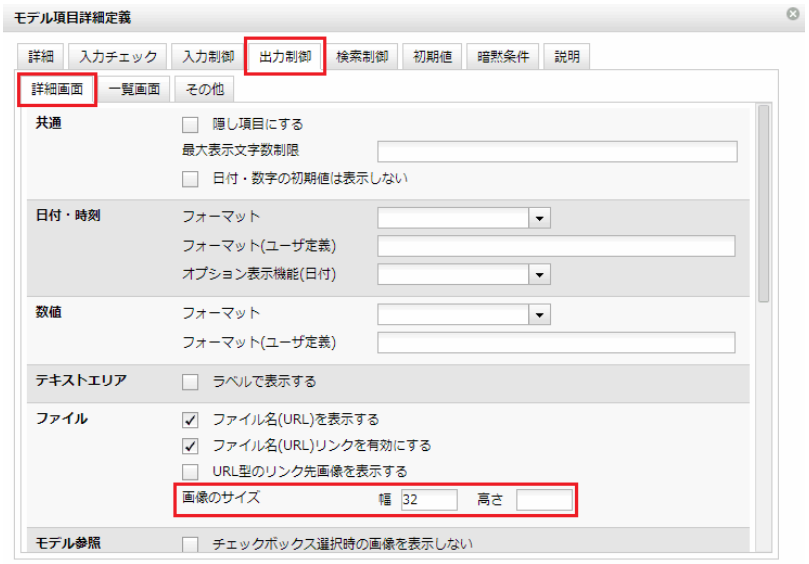


図 詳細画面の画像サイズ幅を指定する

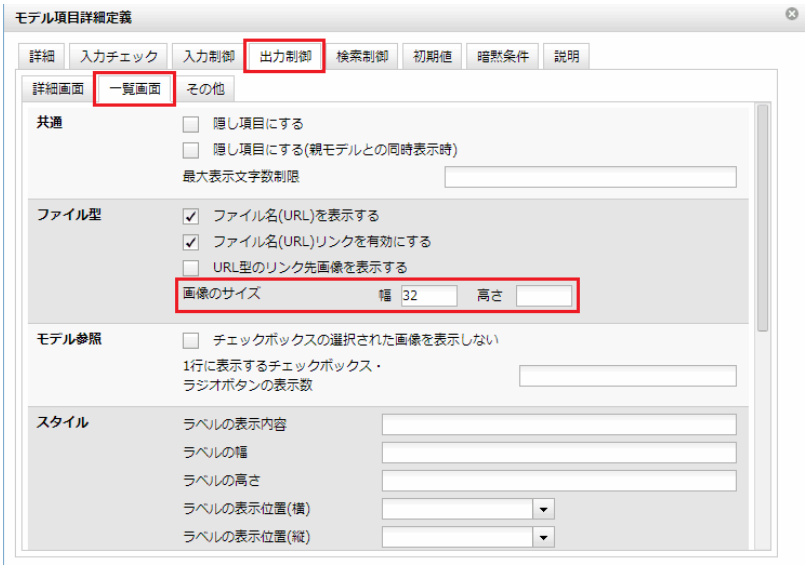


図 一覧画面の画像サイズ幅を指定する

ファイル名を非表示にする

例

ファイル名を非表示にすることができます。

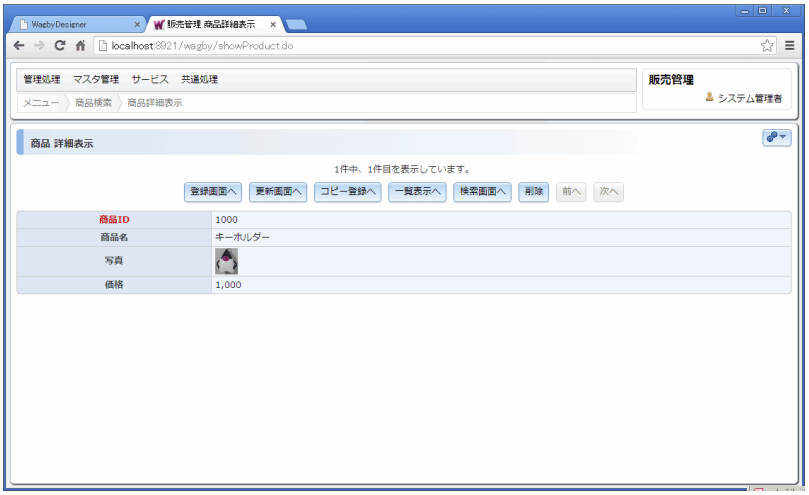


図 ファイル名を非表示にする

定義方法

モデル項目詳細定義の出力制御タブを選択します。詳細画面と一覧画面のタブにある「ファイル名(URL)を表示する」のチェックをはずします。（標準ではチェックされています。）

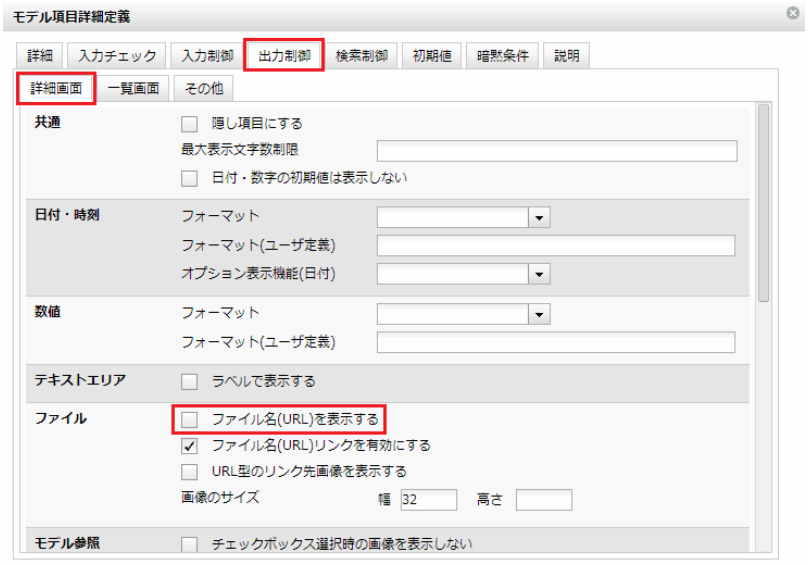


図 ファイル名を表示する、のチェックをはずす

ファイルへのリンクを無効にする

ファイルのアイコンをクリックすると、そのファイルをダウンロードできるという機能を無効にすることができます。 図は、アイコンをクリックしてもファイルがダウンロードされません。

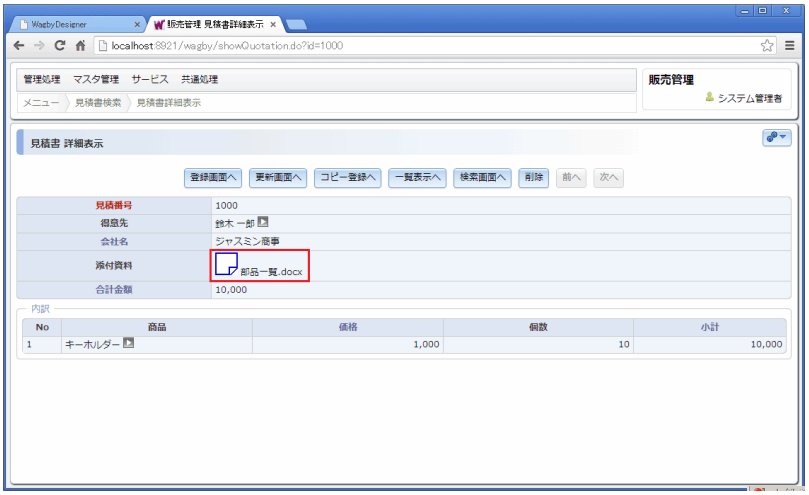


図 ファイルアイコンをクリックしてもダウンロードされない

定義方法

モデル項目詳細定義の出力制御タブを選択します。詳細画面と一覧画面のタブにある「ファイル名(URL)リンクを有効にする」のチェックをはずします。（標準ではチェックされています。）

図 詳細画面の定義

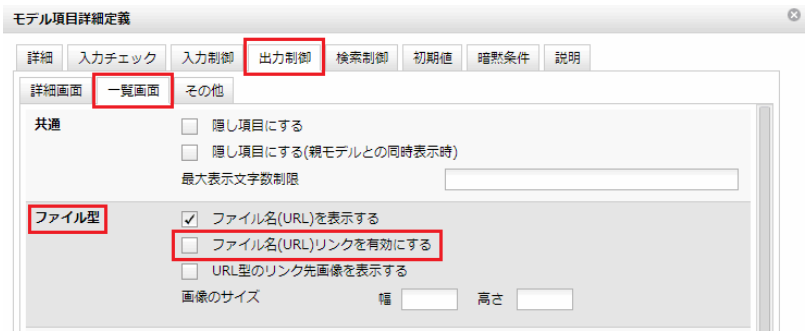


図 一覧画面の定義

画像ファイルをダイアログで表示する（8.1.2）

出力制御 > 詳細画面 > ファイルの設定で、画像をダイアログで表示する、を指定することができます。

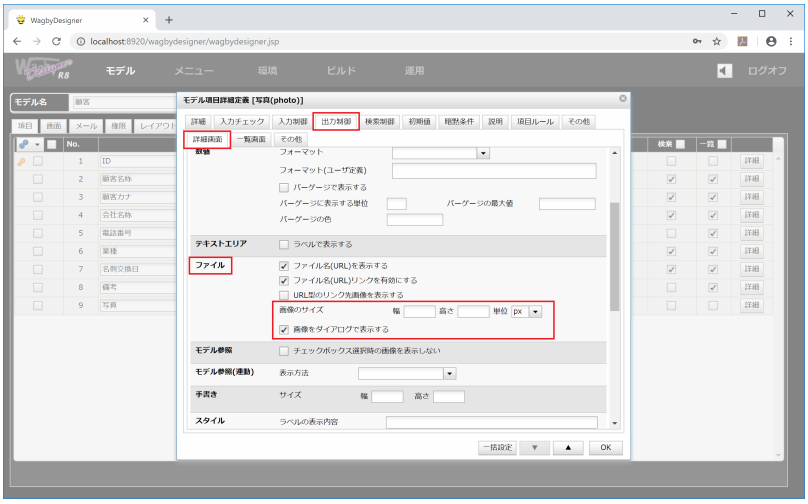


図 画像をダイアログで表示する設定を有効にする

詳細画面で画像をクリックすると、図のようにダイアログに表示されます。

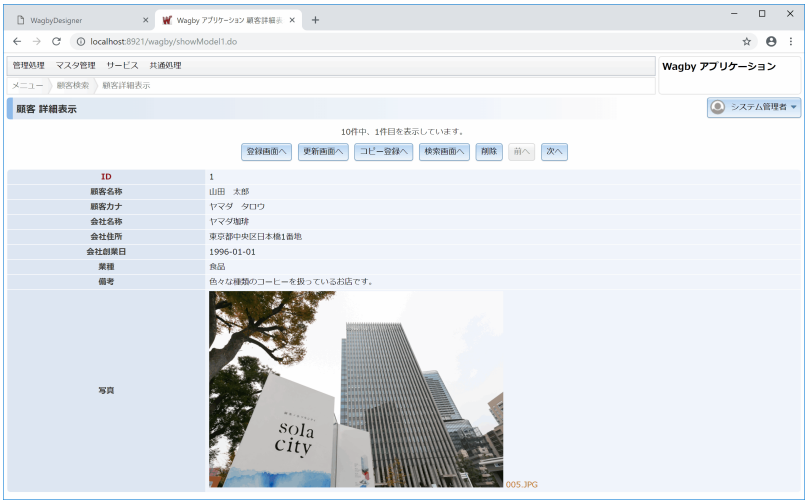


図 画像をダイアログで表示する(1)

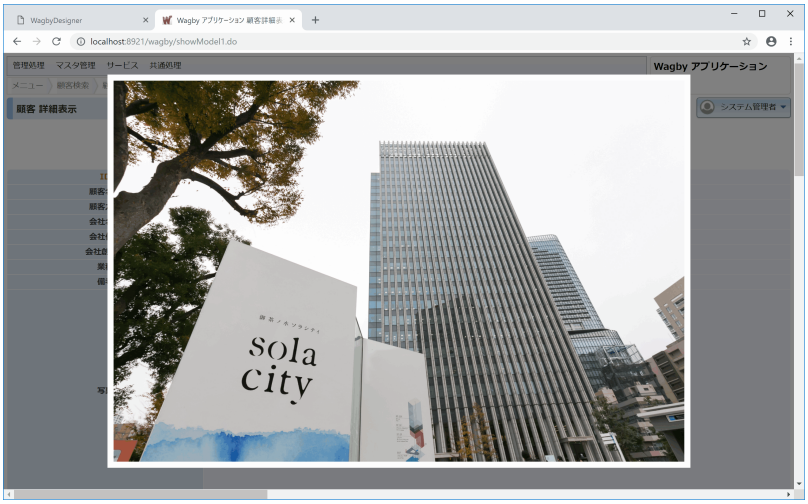


図 画像をダイアログで表示する(2)

この設定を行わない場合、画像をクリックすると別タブが開き、そのタブに画像が表示されます。

更新、コピー登録画面の対応（8.2.0）

詳細画面だけでなく更新、コピー登録画面でも画像がダイアログで表示されます。

表示比率の設定（8.2.0）

画像サイズの設定欄に用意された「単位」を指定できます。パーセントを単位とすると、表示比率を設定することができます。

文字列 - テキストエリア

テキストエリア

例

図は「備考」項目をテキストエリアとして指定した例です。標準では3行の入力欄が用意されます。

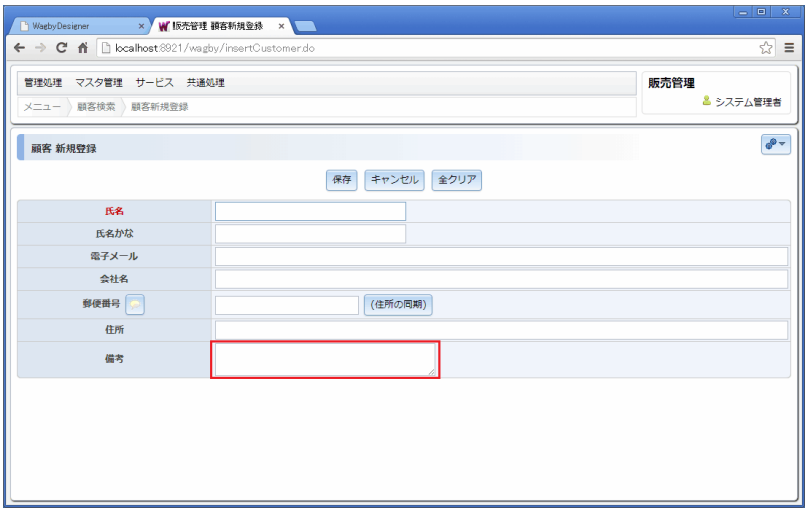


図 テキストエリア項目の入力欄

定義方法

「備考」項目は文字列型とします。　モデル項目詳細定義ダイアログを開きます。「入力制御>入力項目の種類」で〝テキストエリア〟を指定します。

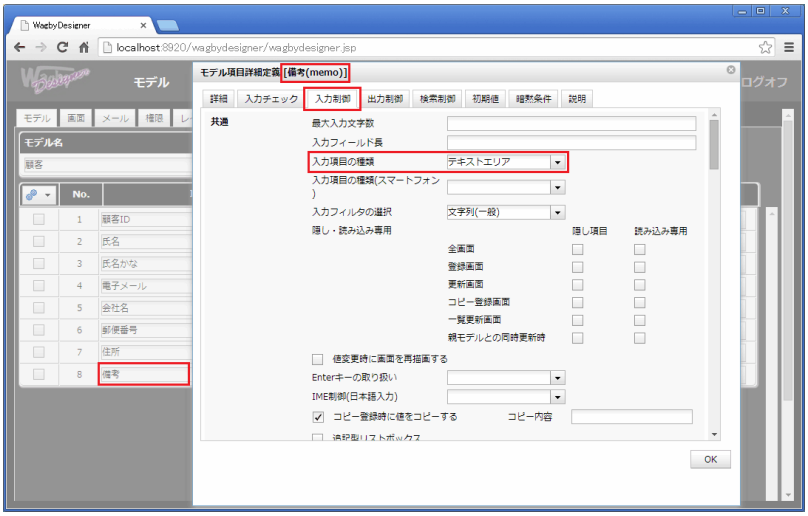


図 テキストエリアの指定

行数と文字数を指定する

例

図はテキストエリアの行数を10行，(1行あたりの文字数を)70文字と指定した例です。

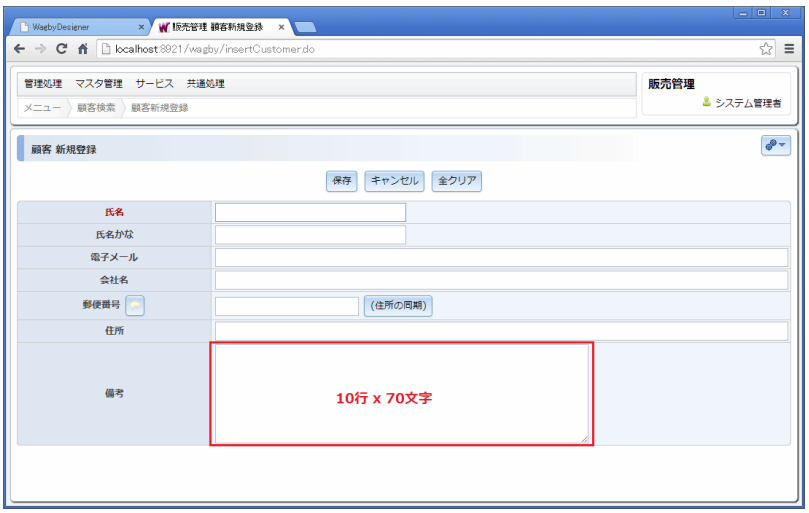


図 行数を指定した例

定義方法

入力制御タブ内にある「テキストエリア>通常」の「行数」と「1行の文字数」をそれぞれ指定します。

図 行数と1行の文字数を指定する

入力欄の幅を直接、指定する

1行に入力できる文字数ではなく、入力欄の幅を直接、指定することもできます。 例えば 70 文字分の入力欄の幅を表示したい場合は、"スタイル ＞ 全画面" に "width:70em", "入力欄のみ" を指定してください。"em" は文字サイズを意味する単位です。

一括設定

項目ルールでもスタイルの設定を行なうことができます。例えば次のようにします。

カテゴリ	リポジトリキー	設定値
入力制御	presentation/displayitem/@style_inputall:全画面	width:70em
入力制御	presentation/displayitem/@style_inputall_2:	入力欄のみ

このルールをテキストエリア項目に適用することができます。項目ルールを使うと、のちの仕様変更が発生した場合でも "70em" を別の値に変更することで、一斉に変更できるメリットがあります。

テキストエリア(リッチ)

テキストエリア(リッチ)を指定すると、文字の修飾やレイアウトを指定することができます。

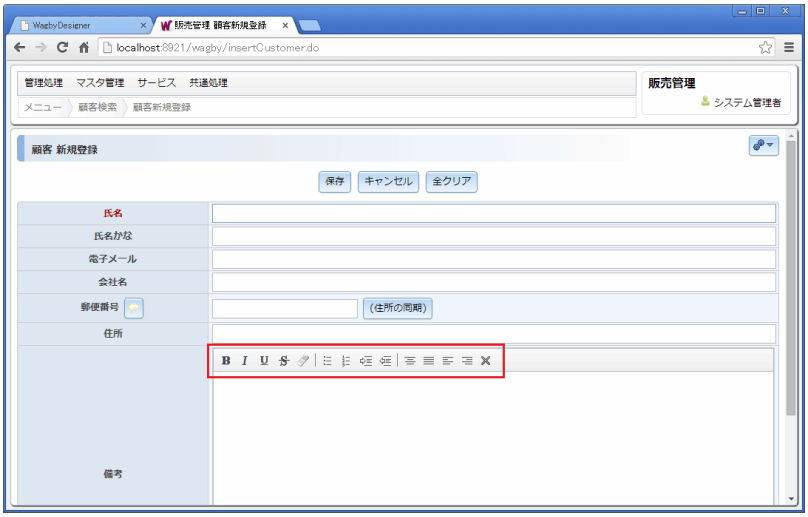


図 テキストエリア(リッチ)(1)

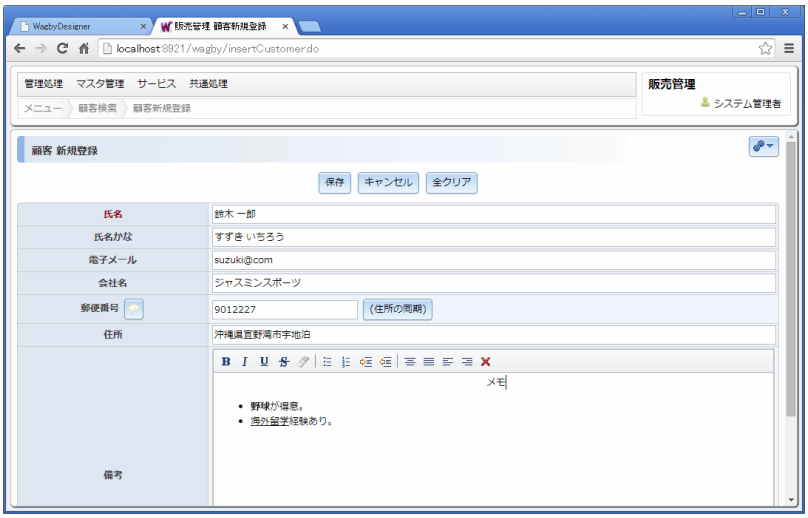


図 テキストエリア(リッチ)(2)

保存後の詳細表示画面でも、修飾された状態となります。

図 テキストエリア(リッチ)(2)

修飾はHTMLタグによって行われます。

定義方法

モデル項目詳細定義ダイアログを開きます。「入力制御>入力項目の種類」で ″テキストエリア(リッチ)″ を指定します。

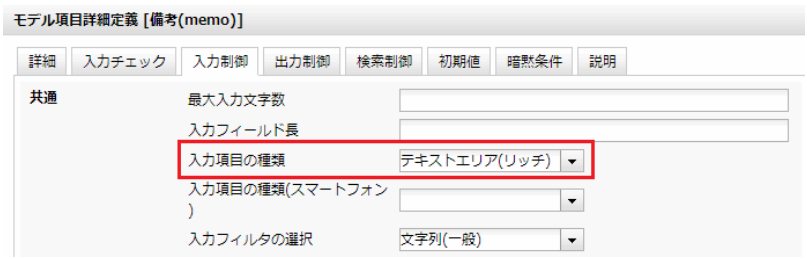


図 テキストエリア(リッチ)の指定

高さと幅を指定する

リッチテキストエリアの「高さ」と「幅」を設定できます。

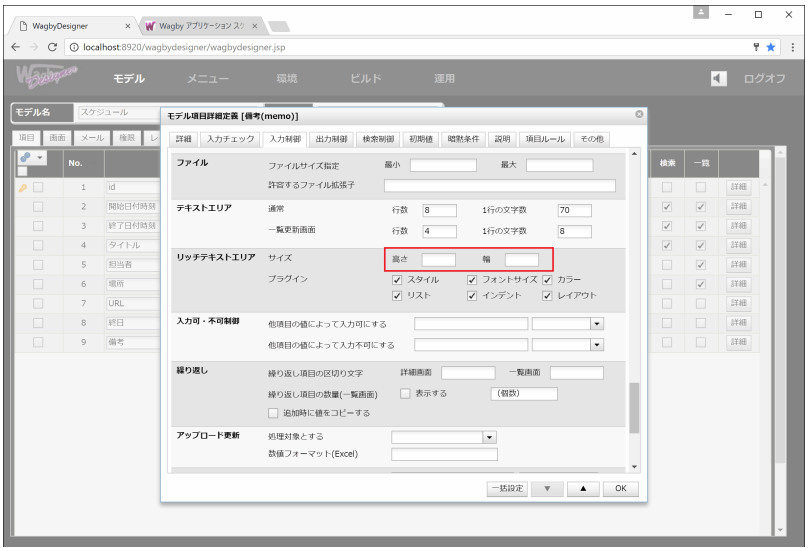


図 テキストエリア(リッチ)の高さと幅の設定

単位を指定する

幅の設定では、単位に px、em、% を設定できます。高さの設定は、親要素に対する相対値が設定できないため、px、em のみ設定できます。

数値のみを設定した場合の単位は「px」となります。

画像を埋め込む

画像を埋め込むことができます。対象となる画像は URL で指定できる必要があります。

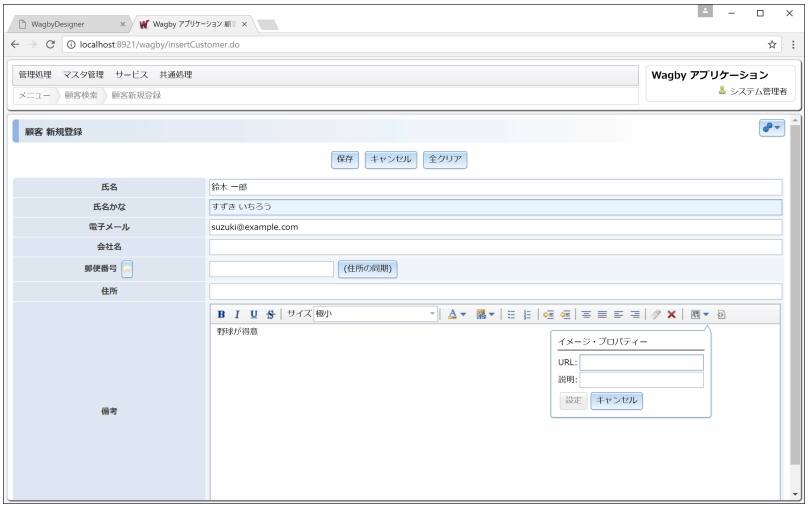


図 画像をURLで指定する

画像ファイルに Wagby の img フォルダ下のファイルを指定することもできます。URL に "./img/画像ファイル名" と指定してください。

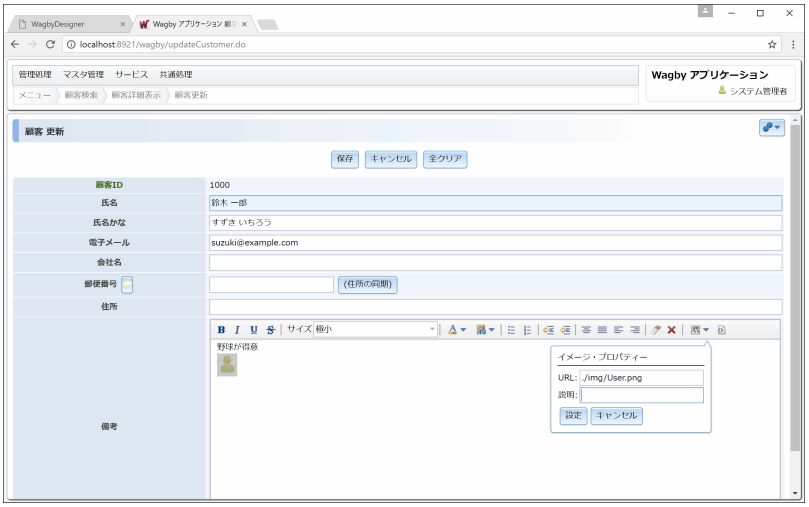


図 URLにimgフォルダ内の画像を指定する

- "./img" を指定しない場合、"http://" が先頭に付与されます。
- 説明欄に入力した文言は、画像にマウスをのせることで表示されます。
- 画像ファイルを更新する場合は、アイコンをダブルクリックしてください。ダイアログが開きます。

HTMLソースを確認する

入力画面で、HTMLソースを直接、確認することができます。

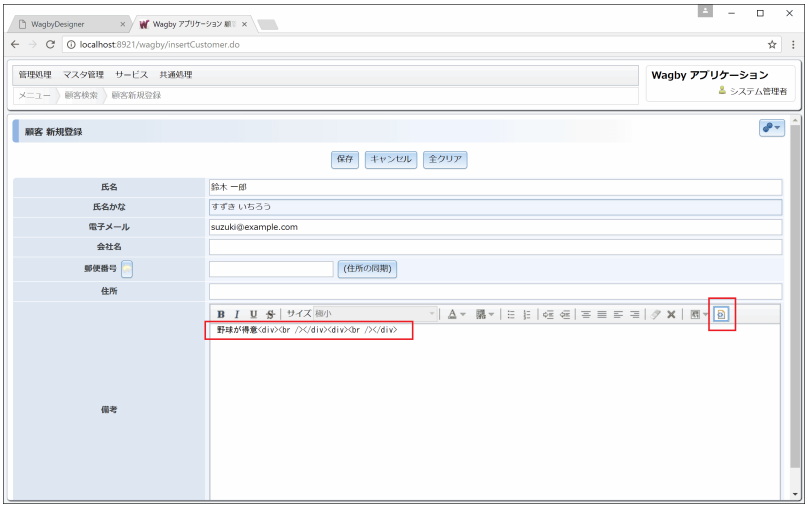


図 HTMLソースを確認する

HTMLの知識のある方は、タグを直接、編集することもできます。（セキュリティを考慮し、編集可能なタグはこのリッチテキストエリアが表現できるものに限定されています。任意のタグを記述できるようにはなっていません。）

プラグインを選択する

6種類のプラグインのうち、利用するものを選択することができます。標準では「スタイル」「フォントサイズ」「カラー」「リスト」「インデント」「レイアウト」が選択されています。

- スタイル
- フォントサイズ
- カラー(背景色と文字色の設定)
- リスト(中黒、数字の設定)
- インデント
- レイアウト(右寄せ、左寄せの設定)
- 画像
- HTMLソース表示

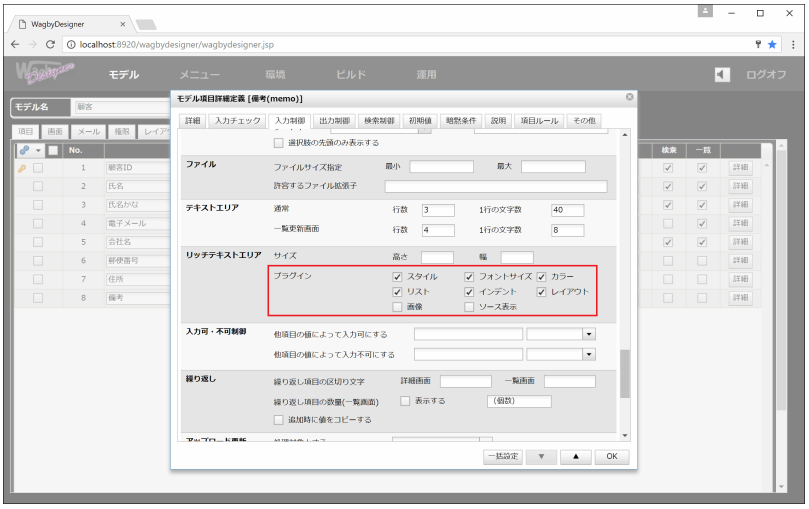


図 テキストエリア(リッチ)のプラグインの選択

(23)

使用上の注意点

(23) 消しゴムアイコンと1文字消去アイコン（バツ印のアイコン）は常に表示されます。

テキストエリア(リッチ)を用いる場合は、次の点にご注意ください。

- テキストエリア(リッチ)は詳細画面および登録／更新画面で有効です。検索画面における条件入力時および一覧更新画面には適用されません。
- テキストエリア(リッチ)の場合、「行数」と「1行の文字数」は反映されません。入力欄の大きさは固定です。
- これまでのテキストエリアを、テキストエリア（リッチ）に変更することは推奨していません。テキストエリア内の改行は、テキストエリア（リッチ）ではすべて無視されます。
- 同じく、テキストエリア（リッチ）をテキストエリアに変更することも推奨していません。テキストエリア（リッチ）では HTML のタグを用いて文字修飾を行います。テキストエリアに変更すると、これまでのデータは HTML タグ付きの文字列となって表示されるようになります。
- テキストエリア（リッチ）の項目をメール送信に適用する場合、REMOVEHTMLTAG関数を適用してください。送信するメールに HTML タグが付与されるためです。
- テキストエリア(リッチ)では、文字列中に ″<″ と ″>″ を含めることができません。（この文字は除かれます。）

テキストエリアをラベルで表示する

画面例

詳細表示画面では、テキストエリア項目は「読み込み専用」として表示されます。

図 テキストエリア項目の詳細表示画面での例

テキストエリアをラベルで表示するを設定すると、テキストエリアの枠を消去することができます。

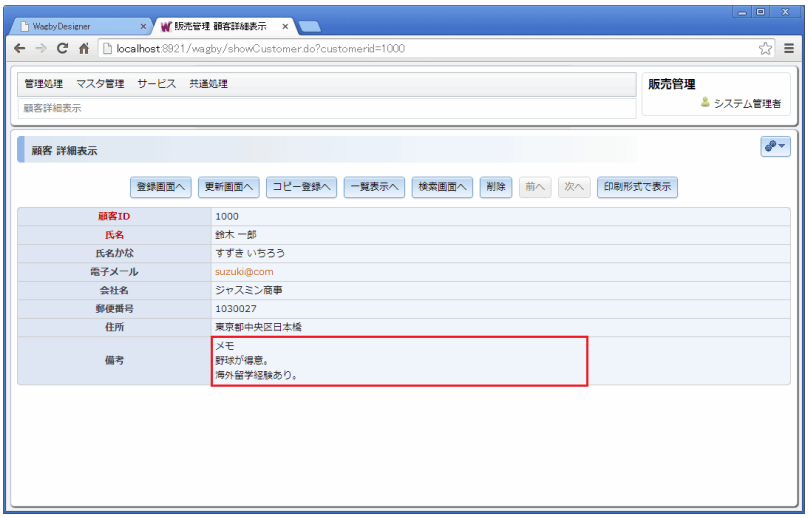


図 テキストエリアをラベルで表示する

定義方法

モデル項目詳細定義ダイアログの「出力制御」タブを開きます。「テキストエリア>ラベルで表示する」を有効にします。

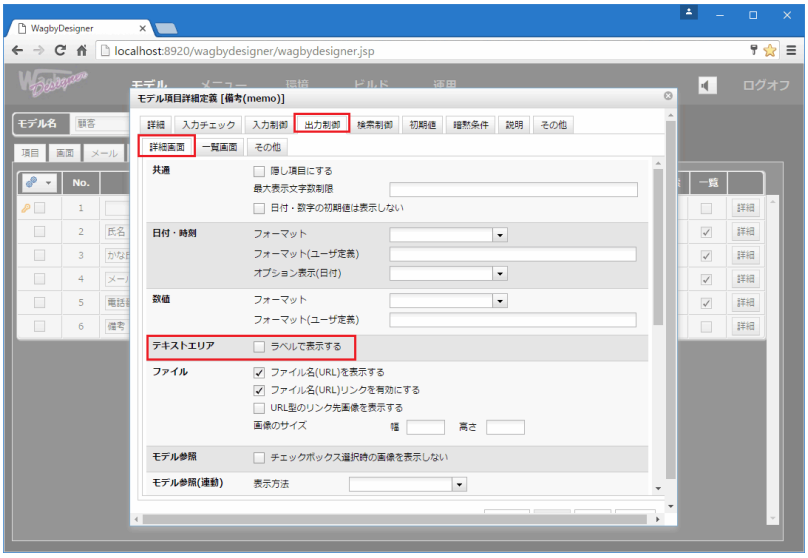


図 詳細画面の設定

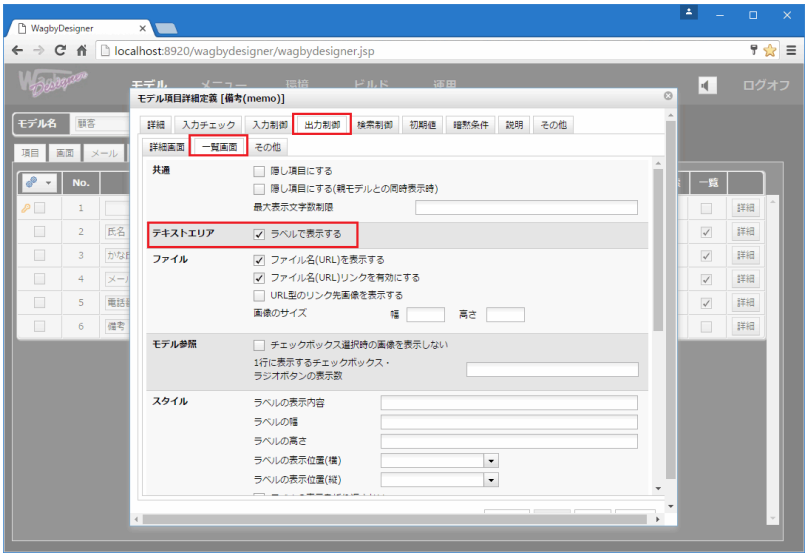


図 一覧画面の設定

この設定は「リッチテキストエリア」では利用できません。リッチテキストエリアで保存したデータにはHTMLタグが含まれてるため、ラベル表示を行うことはできません。

入力可能文字数を表示する (8.1.0)

テキストエリアの入力可能な残り文字数をリアルタイムに表示することができます。図にあるように、入力欄の右隅にゲージが表示されます。

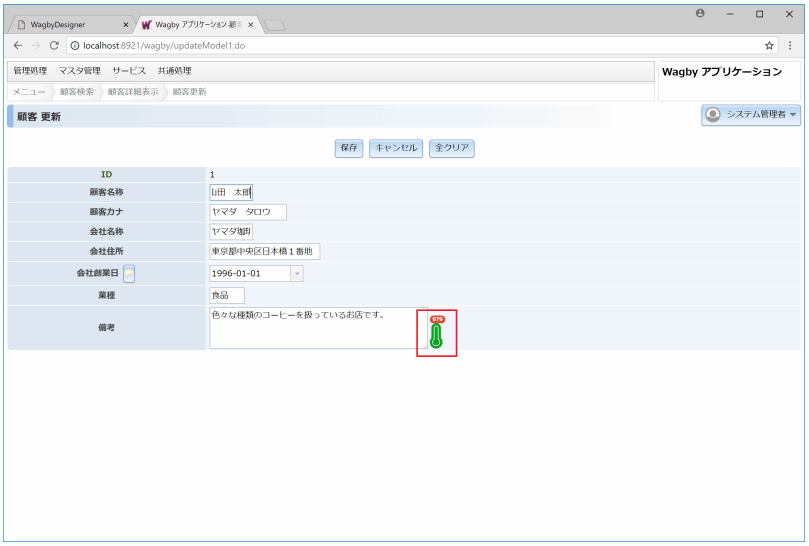


図 入力可能文字数を表現するゲージ

ゲージは残り入力可能文字数によって色が赤に変わります。

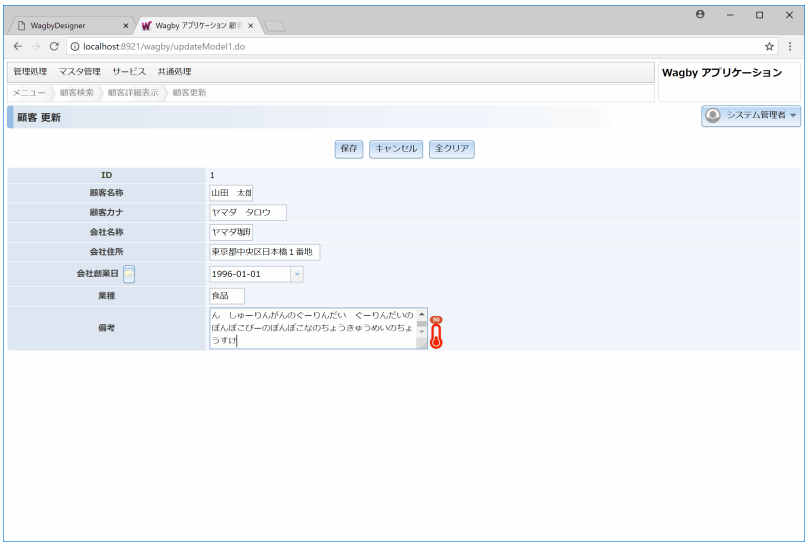


図 ゲージの色が変化する

残り入力可能文字数がゼロになると、これ以上の文字の入力は行えません。

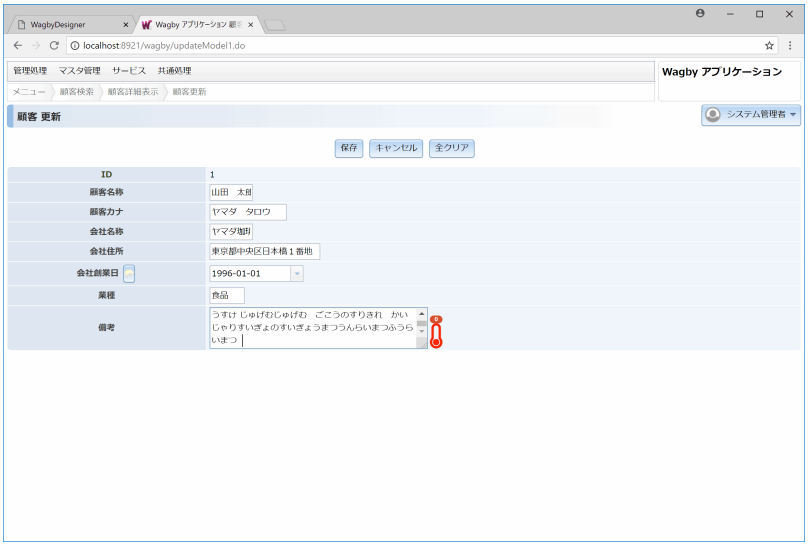


図 入力可能文字がゼロとなったとき

定義方法

モデル項目詳細定義ダイアログの「入力制御」タブを開きます。「最大入力文字数」を指定します。また、入力項目の種類は「テキストエリア」とします。

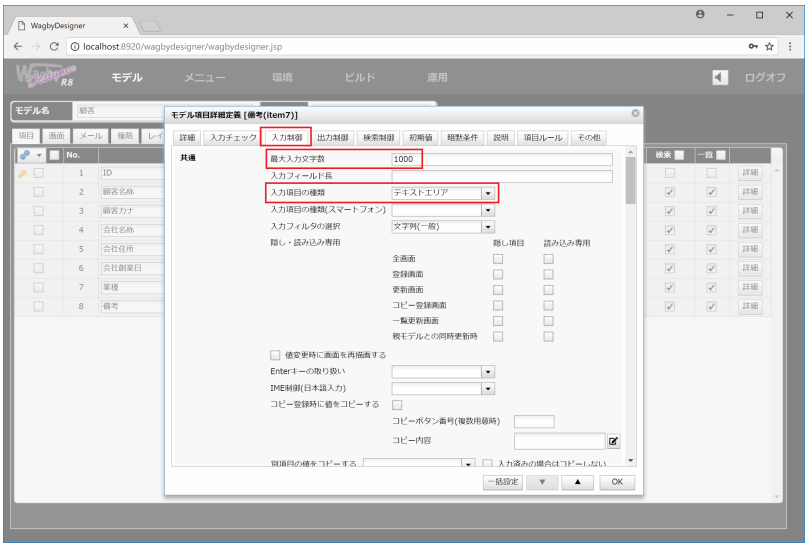


図 最大入力文字数と、入力項目の種類の設定

テキストエリア ＞ 入力可能文字数を表示する設定を有効にします。

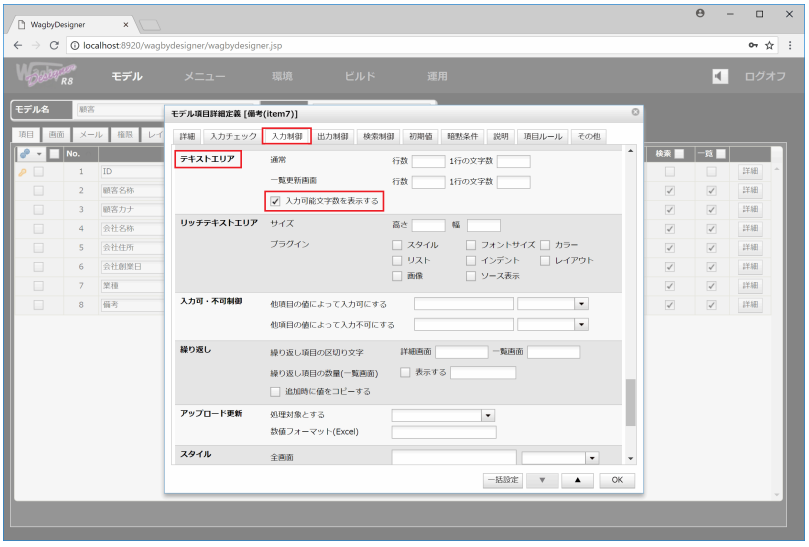


図 入力可能文字数を表示する

仕様・制約

- ″入力制御 ＞ 共通 ＞ 最大入力文字数″ を指定しない場合、入力可能文字数を表示するアイコンが正しく表示されません。
- 固定メッセージの選択・追加による入力方式と併用することはできません。（入力可能文字数は変更されません。）

一覧更新画面の場合

一覧更新画面でのテキストエリア行数は別途、指定できます。　詳細は″画面>一覧更新>テキストエリアの行数・文字数″をお読みください。

データベースとの関係

テキストエリアに格納できる文字数は利用するデータベースによって変わります。詳細...

ブラウザとの関係

利用するWebブラウザによって、テキストエリアの動作が変わります。

リサイズ

テキストエリアのリサイズ機能は、Webブラウザによっては利用できません。図は Windows 7 で Google Chrome を使ったとき、図 は Internet Explorer 11 を使ったときの例です。

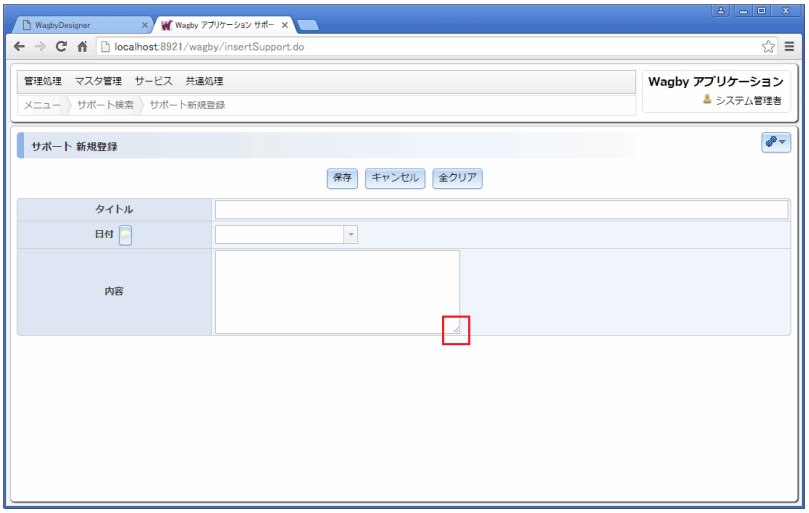


図 GoogleChromeではリサイズが行える

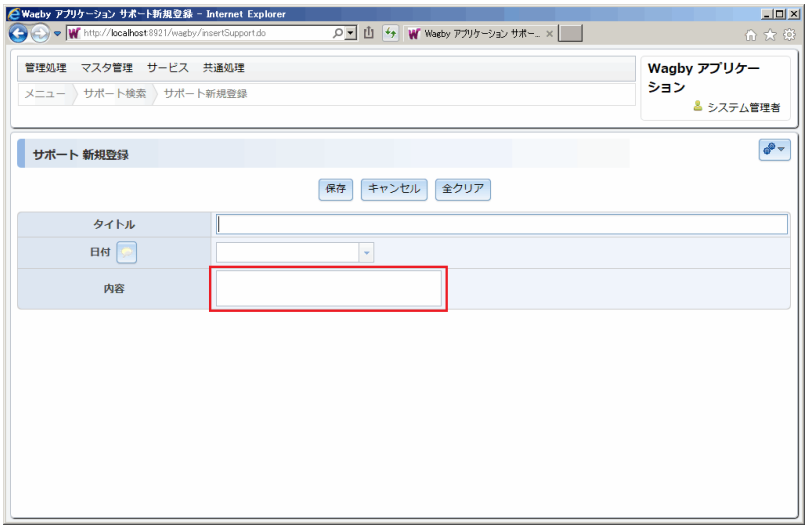


図 InternetExplorer11ではリサイズが行えない

内部で利用する改行コード

リッチではない方のテキストエリアは、改行文字を "LF" (¥n) で保持します。

(24)

(24) クライアントPC がどのような OS であっても、内部では"フィルタ"によって改行文字は LF に統一されます。

文字列 - 追記型リストボックス

例と定義方法

顧客モデルの「業種」項目を追記型リストボックスとした例を示します。 図は初期状態です。リストボックスには値が入っていません。ここで〝卸売業〟と直接、入力して保存します。

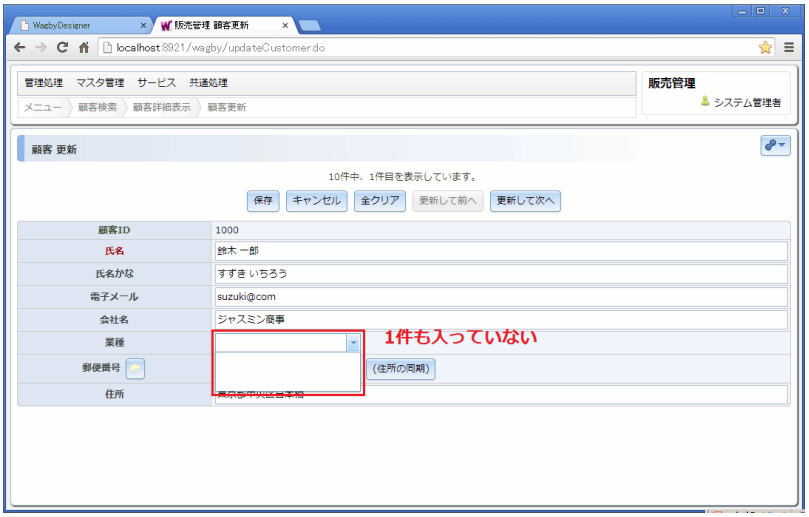


図 追記型リストボックスの利用(1)

更新画面で、先に入力した値が選択できるようになっています。ここでは選択せず、再び〝健康〟と入力して保存します。

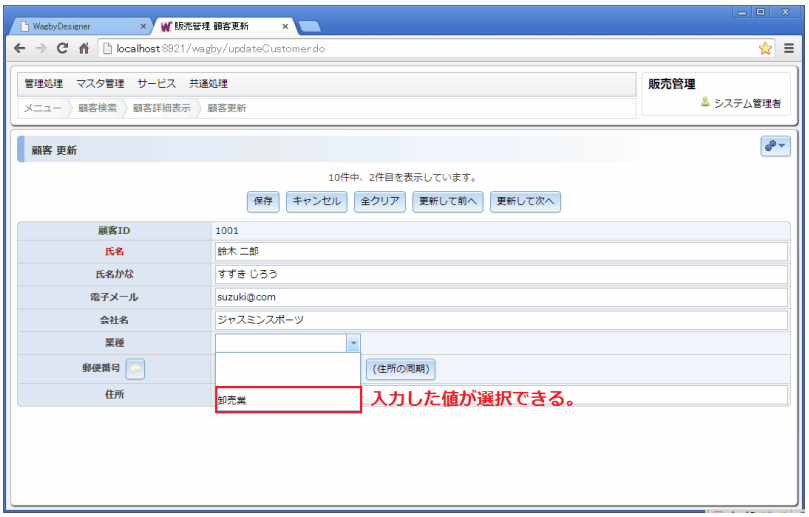


図 追記型リストボックスの利用(2)

このように直接、入力することもできますが、過去のデータを選択することもできます。

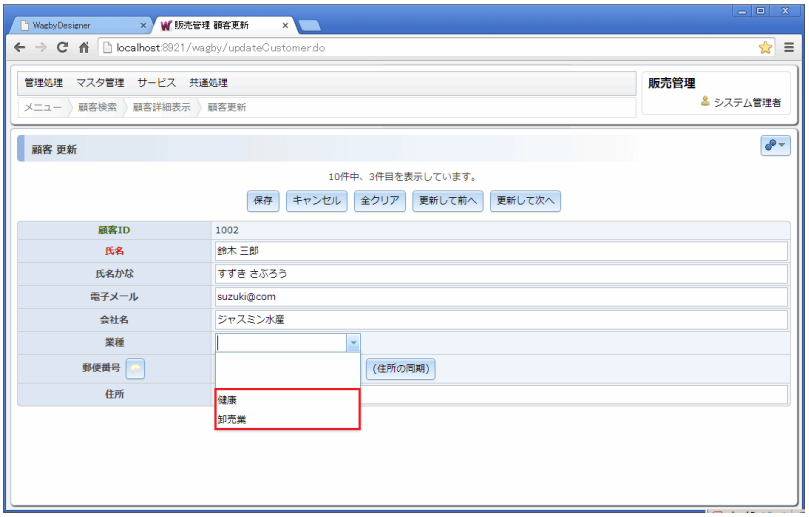


図 追記型リストボックスの利用(3)

定義方法

「業種」項目は文字列型とします。　モデル項目詳細定義ダイアログを開きます。「入力制御>追記型リストボックス」を有効にします。

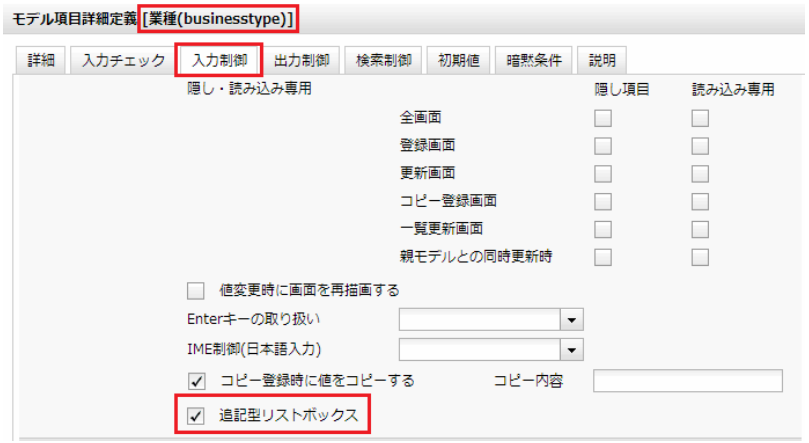


図 追記型リストボックスの設定

検索条件で利用する場合

この設定を行った項目を検索条件として用いた場合、検索条件入力画面でも追記型リストボックスとなります。

仕様・制約

この機能は、すべてのデータをリストボックスに表示します。そのため、格納されるデータ量が多い場合は、この指定は行わない方が良いでしょう。入力パターン数が限定されているようなケースでは有効です。

以下に該当する場合、本機能は指定できません。

- 型が「文字列」以外の項目
- 繰り返し項目
- 繰り返しコンテナ内の項目
- 郵便番号と住所の連動を設定した項目
- 一覧更新画面
- 子モデル同時更新画面

固定値

固定値とは

ここでは「登録ユーザ」「登録時刻」「更新ユーザ」「更新時刻」という4つの項目を追加する方法を説明します。 それぞれの意図は次のとおりです。

項目	型	説明
登録ユーザ	文字列	このデータの新規登録を行ったログオンアカウントを記録します。登録時に一度だけ値が設定され、以降、変更されることはありません。
登録時刻	日付と時刻	このデータの新規登録を行った日付時刻を記録します。登録時に一度だけ値が設定され、以降、変更されることはありません。
更新ユーザ	文字列	このデータの更新を行ったログオンアカウントを記録します。更新の都度、値が上書きされます。最後に更新されたログオンアカウントが記録されている状態です。
更新時刻	日付と時刻	このデータの更新を行った日付時刻を記録します。更新の都度、値が上書きされます。最後に更新された日付と時刻が記録されている状態です。

例

図は、新規登録画面を開いた画面です。各項目の初期値が設定されており、かつ変更不可（読み取り専用）となっています。この段階では、データベースに値は保存されていません。

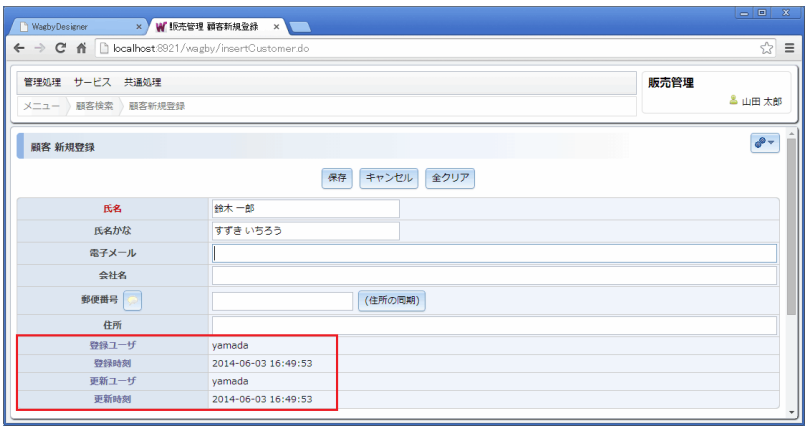


図 新規登録画面を開く

図は、登録が終わった直後の画面です。登録時刻と更新時刻は変更されています。この段階でデータベースに値が保存されます。つまり図の状態は表示のみで、図の状態（登録を行った）が確定となります。

図 登録の直後

図は、別の利用者が更新を行った画面です。登録時刻と登録ユーザは変更されていません。更新時刻と更新ユーザが上書きされています。

図 更新の直後

定義方法（項目テンプレートを使う）

「登録ユーザ」「登録時刻」「更新ユーザ」「更新時刻」の4つはテンプレートとして用意されています。 ギアアイコンからテンプレートを選択することで追加することができます。

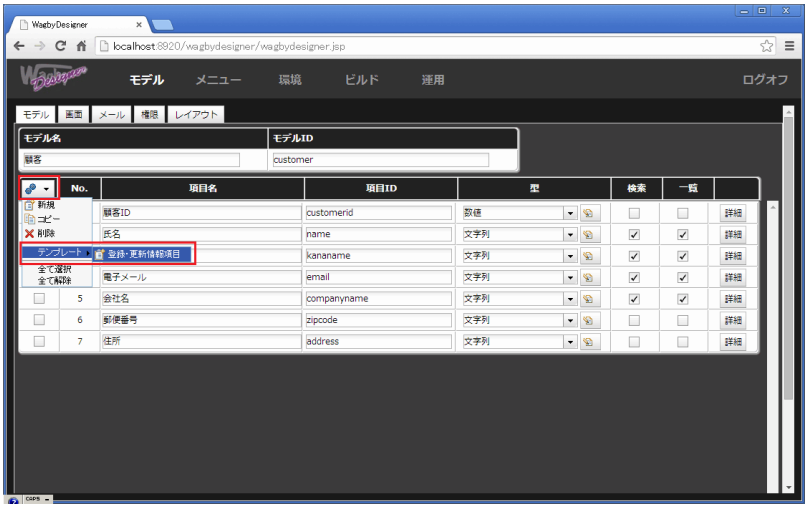


図 テンプレートを選択する

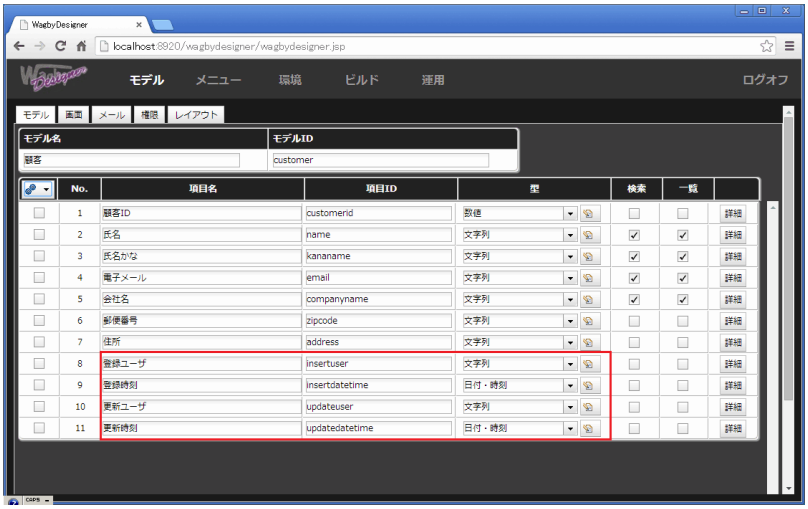


図 「登録ユーザ」「登録時刻」「更新ユーザ」「更新時刻」を追加する

テンプレートは開発者が定義することもできます。 詳細は"Designerの機能・その他のツール > 項目テンプレートを用意する"をお読みください。

項目の詳細

追加された項目の名前は変更可能です。標準では以下のような設定になっています。

「登録時刻」と「更新時刻」の項目の型は「日付と時間」となっています。

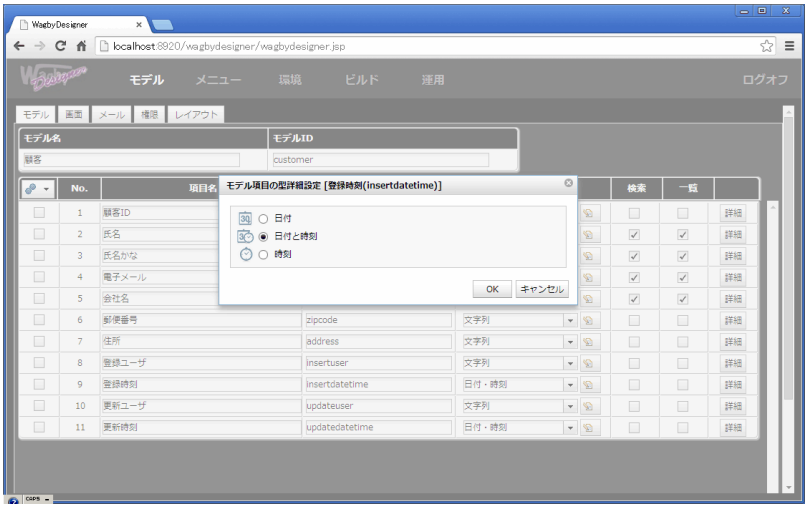


図 型の詳細-日付と時間

モデル項目詳細 - 共通

追加した4つの項目すべてで、「読み込み専用」が設定されています

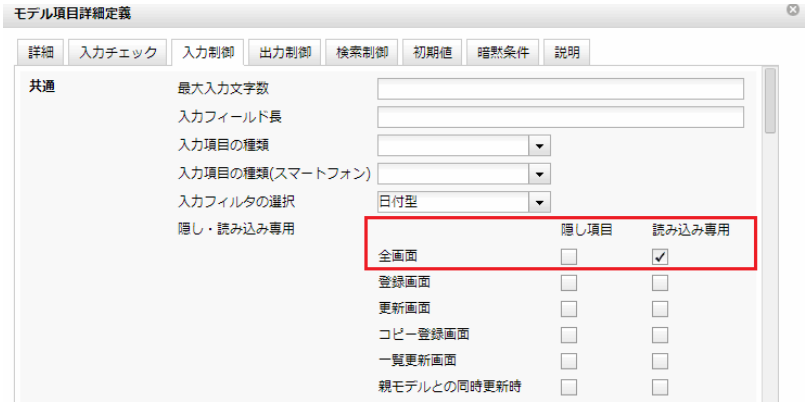


図 読み込み専用が設定されている

登録時刻の初期値

「登録時刻」項目の初期値として、「登録画面表示時」と「登録時」に ″now″ が設定されています。これは現在の日付時刻を返す特別な表記です。[25]

```
now
```

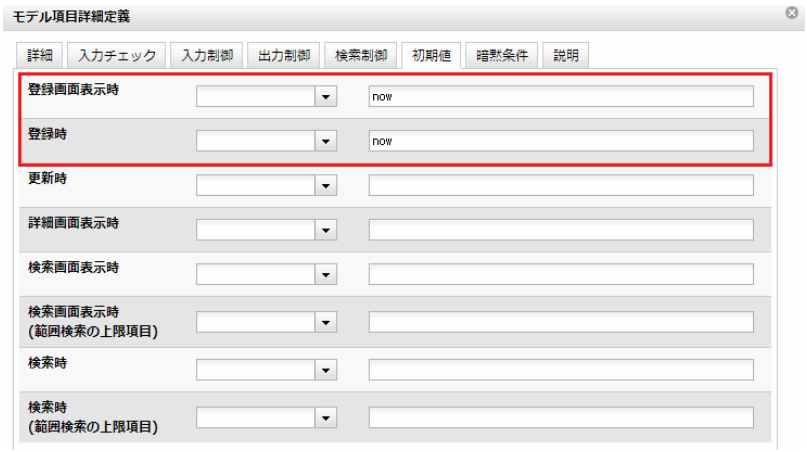


図 nowの設定

登録ユーザの初期値

「登録ユーザ」項目の「初期値」として、「登録画面表示時」と「登録時」に ″ログオンID″ が設定されています。これは現在のログオンアカウントを返します。

[25] now は初期値の設定でのみ指定できます。

図 ログオンIDの設定

更新時刻の初期値

「更新時刻」項目の「初期値」として、「登録画面表示時」「登録時」「更新時」に ″now″ が設定されています。これは現在の日付時刻を返す特別な表記です。[(26)]

```
now
```

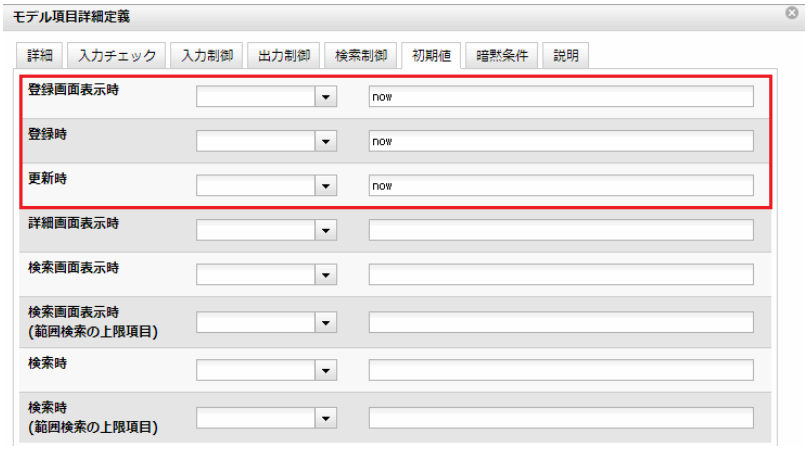


図 nowの設定

更新ユーザの初期値

「更新ユーザ」項目の「初期値」として、「登録画面表示時」「登録時」「更新時」に ″ログオンID″ が設定されています。これは現在のログオンアカウントを返します。

(26) now は初期値の設定でのみ指定できます。

図 ログオンIDの設定

now 指定と同様なものに、NOW関数があります。しかし固定値として動作させる場合はNOW関数は使わないようにしてください。nowと指定した場合、再計算処理・アップロード更新・メール送信ジョブ機能で更新日付を上書きするかどうかを個別に設定できるようになります。NOW関数を使った場合、一律に更新されます。

「ログオンID」指定と同様なものに、USERNAME関数があります。しかし固定値として動作させる場合はUSERNAME関数は使わないようにしてください。「ログオンID」と指定した場合、再計算処理・アップロード更新・メール送信ジョブ機能で更新者を上書きするかどうかを個別に設定できるようになります。USERNAME関数を使った場合、一律に更新されます。

データベースの詳細

概要

「モデル項目詳細定義>詳細>データベースの詳細」を指定することができます。

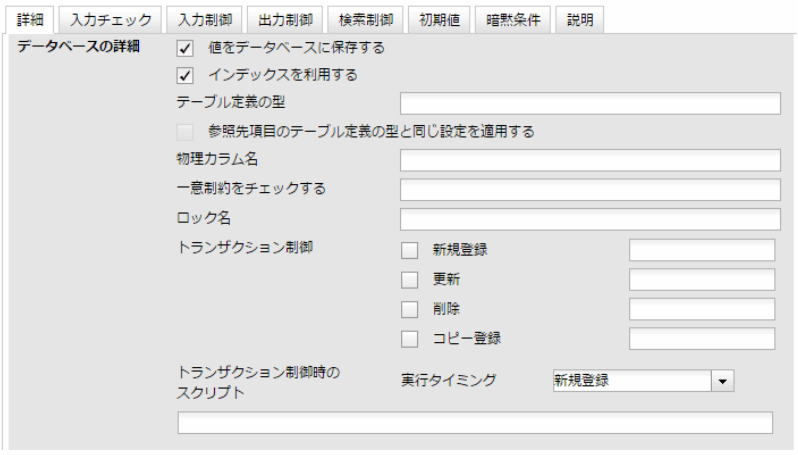


図 モデル項目詳細定義

値をデータベースに保存する

標準では有効となっています。この場合、モデルに対応するテーブルにカラム（列）が作成されます。 この設定を無効にすると、テーブルにカラムが用意されません。

計算によって導出される項目でデータベースに保存する必要がない、という場合に利用できます。

ご注意ください：データベースに保存しない項目は「検索項目として利用できない」「ソート項目として利用できない」という制約があります。また、一覧表示や集計ビューに適用すると（常に再計算されるため）パフォーマンス劣化の一因となります。

インデックスを利用する

Wagby は次の条件に合致した項目は自動的に（テーブルに対する）インデックスを作成します。[27]

- データベースに保存されている。
- 検索項目として指定されている。
- 数字型または日付型である。
- 文字列型の場合、完全一致検索が指定されている。また、テキストエリアではない。
- モデル参照項目のキー格納部分である。ただしチェックボックスを除く。
- このモデルに対する検索行為が生じること。（″画面 > 検索・一覧表示 > 画面を作成する″ が有効になっている、または他のモデルから「モデル参照」が行われ、サブウインドウで検索を行うこと）

本設定を無効とすることで、インデックスの作成を抑制します。

テーブル定義の型

リポジトリに指定する「整数型」や「文字列型」は、各データベースが用意するそれぞれの型へ自動変換されます。

画面から入力した値をデータベースに格納できるような適切な長さを指定することが必要な場合もあります。[28]

この型を詳細に指定することができます。図では、文字列型の項目について、テーブル定義の型を varchar(1024) へ変更した例です。

図 型を指定する

既存システムのテーブル定義を Wagby で再利用するような場合、テーブル定義側の型情報をこの欄へ転記してください。

文字列型の場合の特殊表記

文字列型項目の場合、図の欄に数字を記述することで「文字列長」として解釈されます。[29]

参照先項目のテーブル定義の型と同じ設定を適用する

モデル参照項目で、参照先モデルの主キーがテーブル定義の型を指定している場合、「参照先項目のテーブル定義の型と同じ設定を適用する」をチェックしてください。

---

(27) データベースに対して CREATE INDEX... という命令を発行します。発行するタイミングは、外部データベースに対する初期データ読み込み（init_db.bat ファイル実行）時です。このときテーブル作成と同時にインデックスも作成します。

(28) 例えば MySQL を用いた場合、文字列型は varchar(255) として扱われます。初期サイズとして小さい場合は、変更が必要です。

(29) Wagbyが自動生成する create ddl 文に、指定した数字が反映されます。この数字がバイトなのか文字なのか、については、ご利用のデータベースの仕様をご確認ください。

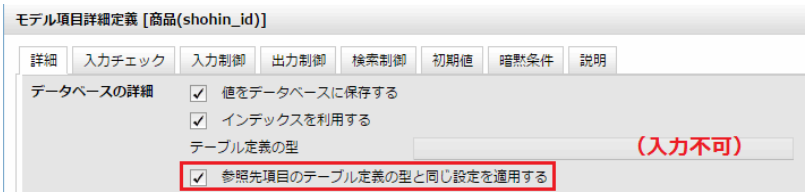


図 参照先項目のテーブル定義の型と同じ設定を適用する

本設定は、この項目がモデル参照の場合にのみ、チェックすることができます。チェックした場合、「テーブル定義の型」は入力不可となります。（本設定が優先されます。）

物理カラム名

通常はモデル項目の「項目ID」（英語で表記）がテーブルのカラム名として用いられます。「物理カラム名」を指定することで、この動作を変更することができます。

図 物理カラム名を指定する

既存システムのテーブル定義を Wagby で再利用するような場合、テーブル定義側のカラム名をこの欄へ転記してください。

複合キーモデル参照の場合

複合キーモデルを参照する他モデル参照項目でも物理カラム名を任意に指定することができます。

物理カラム名項目の入力欄に、コンマ区切りで（複数の）物理カラム名を記入します。

日本語カラム名を指定する

物理カラム名に日本語を利用することができます。[文字コードの設定について...]

一意制約

一意制約を設定すると、同じデータを重複して登録できないようになります。図にあるように、制約名を指定します。

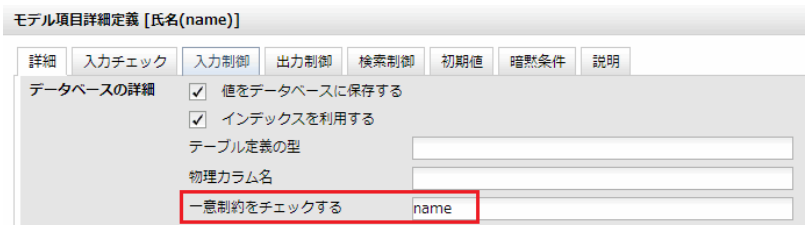


図 一意制約を指定する

複数項目の組み合わせ

複数の項目を組み合わせた一意制約を設定することもできます。図, 図は「氏名」と「氏名かな」項目の二つで一意制約としています。このように同じ制約名をそれぞれの項目に設定します。

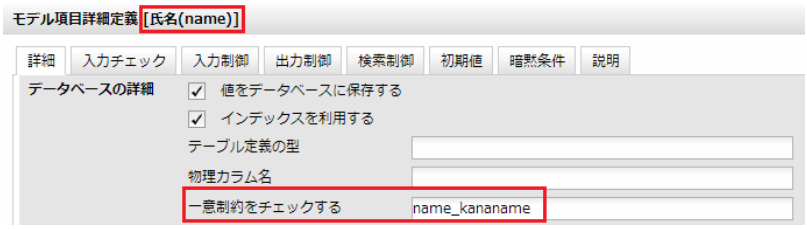


図 氏名項目に指定した一意制約

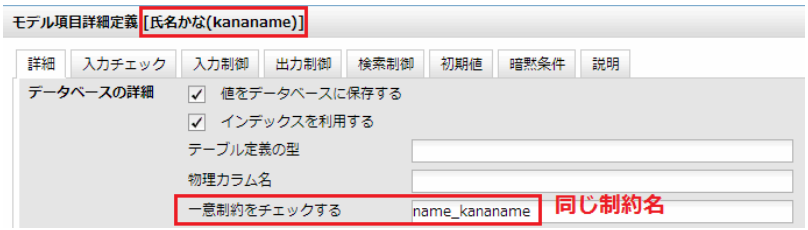


図 氏名かな項目に指定した一意制約

複数の一意制約名を指定する（8.3.3）

一意制約に″a1,b1″というようにコンマ区切りで記述することができます。

仕様・制約

- 一意制約に関する仕様・制約をまとめています。
- カンマ区切りで複数の一意性制約名を指定したとき、個々の制約名の長さチェックではなく、全体での長さチェックとなっていました。R8.3.7/R8.4.2で修正されました。制約名の長さでエラーとなる場合、名前を短くするか、R8.3.7/R8.4.2以降のWagbyをご利用ください。

ロック名

Wagbyが内部で管理するロックキーのルールを変更することができます。通常、この設定を行うことはありません。

詳細は″ロックキーのカスタマイズ″をお読みください。

トランザクション制御

″モデルをまたがる計算（トランザクション）″をお読み下さい。

データベースの詳細 - SQL式(SQL)

概要

Wagbyが発行するSQLの書式は次のようになります。

```
SELECT 項目名 FROM モデルID WHERE 項目名=...
```

ここで紹介する「SQL式」とは、上記の「項目名」部分を、特定の式に置換するものです。

想定している利用方法は、この「項目名」がデータベース非保存のとき（テーブルには存在しない）、項目名部分に特定の式を適用してSQLを実行する、というものです。

定義方法

「モデル項目詳細定義>詳細>データベースの詳細>SQL式」に、SQL文（の一部）を記述することができます。

図 SQL式の設定欄

データベースに保存しない項目であっても、ここに指定されたSQLを用いて項目を取得するようになります。

本設定はご利用になるデータベースに依存します。本ページの記載は特に断りのない限り、MySQL 用の SQL となっています。

開発に関する Tips

開発者が記述した式は、Wagby に同梱されている Hibernate という O/R マッパーソフトの設定ファイル（hbmファイル）における formula 要素に反映されます。

具体的には、wagbyapp¥webapps¥[プロジェクト名]¥WEB-INF¥classes¥[モデル名].hbm.xml という設定ファイルの formula 要素に出力されます。例を示します。

```
<property name="item3_">
   <formula>`num1`+1</formula>
</property>
```

開発者はこの設定ファイルを修正して、Wagbyアプリケーションを再起動することにより、すぐに動作を確認することができます。式を試す場合は、この方法を推奨します。

動作する SQL 式ができましたら、Designer内の設定に反映させます。（これを忘れてビルドしないようにしてください。この設定ファイルは、ビルド時に再作成されます。）

[例1] モデル参照項目をデータベースに保存しないが、対応する参照連動項目を検索したい

staffモデルとstaff2モデルがあるとします。別モデルですが、同じIDであれば、同じスタッフを指すとします。（例えば ID="1000"なら、同一人物 "山田" を指します。）

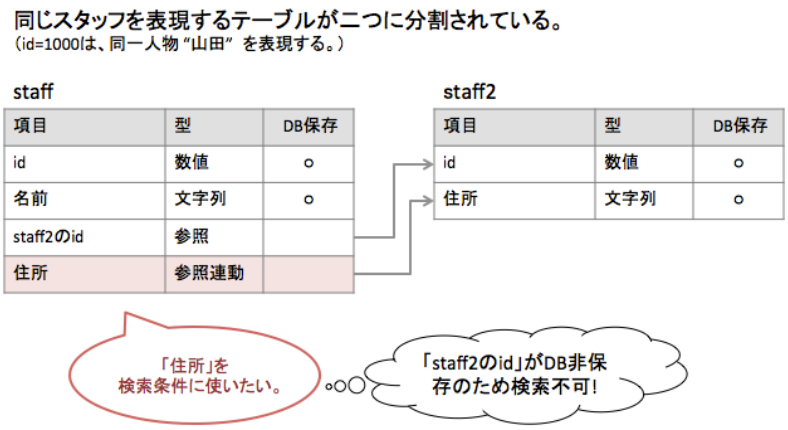


図 staffモデルとstaff2モデル

ここでstaffモデルの「住所」項目（参照連動項目）を検索で用いたいという要件があります。しかし、参照連動項目の元となる「staff2のID」はデータベース非保存としたため、検索することができません。

SQL式の設定

「staff2のID」項目に、自動計算式として id 項目を参照させます。これで表示は自IDとなります。（今回はstaffモデルのid項目と、staff2モデルのid項目は同じ値が入るという前提です。）

さらに、同項目のSQL式に、次の設定を行います。

```
`ID`
```

検索時に、この式を解釈します。データベース非保存でも、この値を使うようになりますので、今回のケースでは検索が成功します。

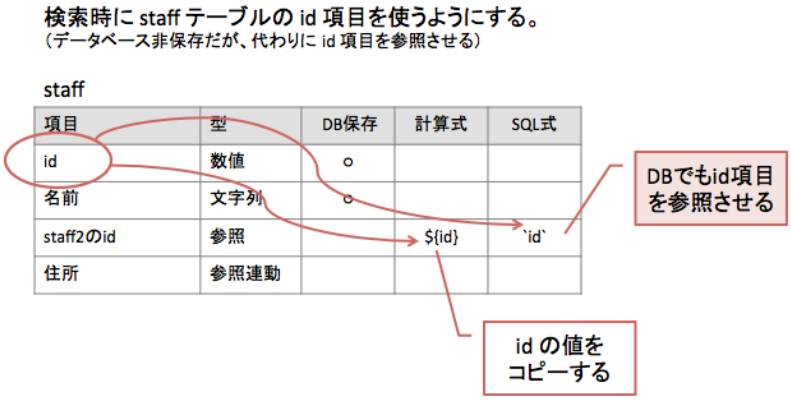


図 SQL式設定のイメージ

(30)

[例2] 計算式（自モデル）

Wagby の自動計算式を使わず、SQL の式で計算を行う例を紹介します。

モデル counttest は、入力値 num1 の値に 1 を加えた項目 calc1 と calc2 があります。いずれもデータベースには保存しない、としています。

```
`num1`+1
```

(30) SQL式では、クォート(`)で囲むことで自テーブルに対応するエイリアスを付与します。この例では staff.ID として扱われます。(staff2.ID ではありません)

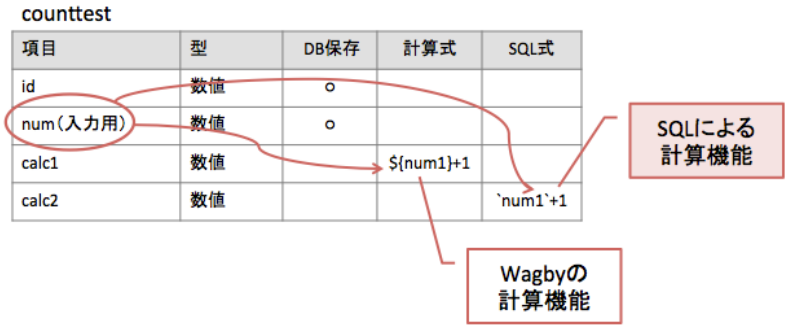


図 SQL式による計算

calc1 は（データベースに保存しないため）検索項目として用いることはできません。しかし calc2 は検索項目として用いることができます。計算によって導出される項目で、検索条件としても使いたいが、データベースには保存したくないという場合に SQL 式を用いることができます。

> calc1 項目は、登録・更新画面で入力値 num1 が変更されると即座に値が +1 され再描画されます。 しかし calc2 項目は、保存しないと値が反映されません。また、calc2 項目は（Wagbyの）自動計算式が設定されていない状態だと入力可能となってしまうため、別途「読み込み専用」の設定を行うようにしてください。または、calc2 項目に計算式もあわせて指定してもかまいません。この場合は num1 の変更で即座に再描画されます。

[例3] 計算式（子モデル）

親モデル parent と、子モデル child を想定します。parent 側には、子モデルの数（COUNT）を保持する項目を用意しますが、データベースに保存しない、とします。

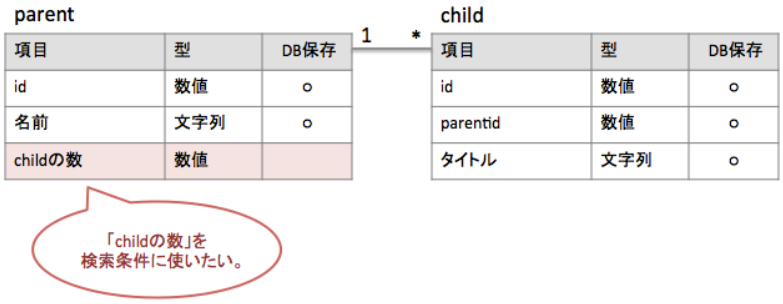


図 parentとchildの関係

そこでparentの項目「childの数」に、次のSQL式を用います。

```
(SELECT COUNT(*) FROM child WHERE child.parentid = `id`)
```
(31)

SQL文中の`id`はparentテーブルの列を用いたいので、クォート(`)で囲みます。これによって parent テーブルに対応するエイリアスを付与するようになります。

childやchild.parentidはそれぞれchildテーブルおよびchildテーブルのparentid列を指すため、クォートで囲まずに指定します。

> parentモデルはキャッシュを無効にする必要があるかも知れません。子モデルを追加した場合に、親モデルの詳細表示はキャッシュを使わず、常に計算した値を使うようにします。（古いデータを表示させないようにします。）

(31) この式は内蔵データベース(HSQLDB)では動作しません。MySQLで動作することを確認しています。列名を括る文字 ` は、MySQLで使える区切り文字です。

LIKE 演算子を使う

上記の SQL で、child.parentid = `id` の部分を部分一致検索にする場合は LIKE 演算子を使います。（id は文字列型と仮定します。）

LIKE 演算子の書き方は、利用するデータベースによって変わります。 例えば SQL Server の場合は、次のようになります。

```
(SELECT COUNT(*) FROM child WHER child.parentid LIKE '%'+[parentid]+'%')
```

(32)

[例4] 保存前の値を保持する項目を用意する

登録・更新処理では、保存ボタンを押下する前の情報がデータベースに格納されています。保存ボタンを押下することにより、データベースの値が更新されます。

そこで、データベースの値を保持する項目を用意することで、保存前の値を使う方法を紹介します。

例

入力値と、入力前の値の差を計算する例を示します。

図 計算例(1)

図 計算例(2)

定義方法

次の項目を用意しました。

項目	型	SQL式	計算式
num1	数値		
num2	数値	`num1`	
num3	数値		${num1}-${num2}

num2項目はDB保存なしとし、SQL式を設定しています。また、読み込み専用としています。

(32) SQL Server では、列名を括る文字は［列名］と記述します。

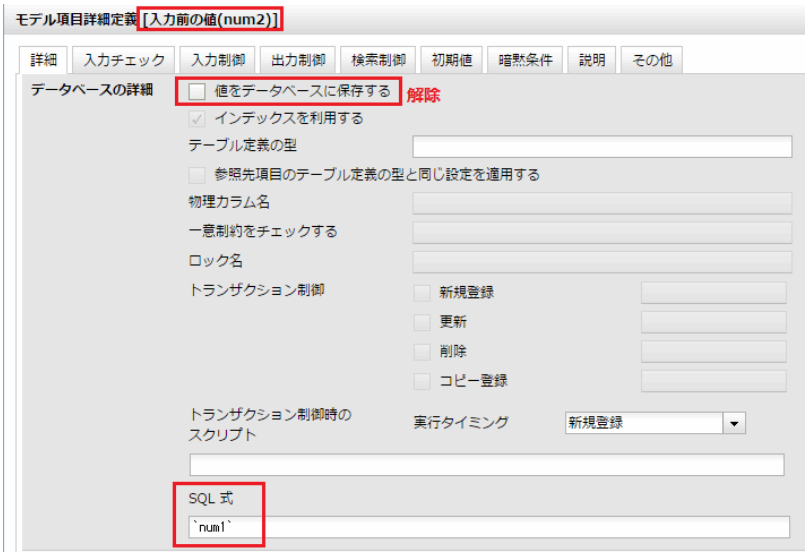


図 SQL式の設定

num3項目はDB保存とし、計算式を設定しています。

図 計算式の設定

[例5] NULL 値のソートルールを変える

SQL Server の場合

SQL Server ではソート対象項目に NULL 値が含まれていた場合、NULL を最も小さい値として扱うため、一覧表示の先頭に表示されます。これを変更し、NULL を含む行を（一覧表示の）最後に表示させる方法を紹介します。

SQL式に下記の値を指定した項目を用意し、この項目をソート順の1番目とします。2番目に対象項目を指定します。

```
(CASE WHEN [対象項目の列名] IS NULL THEN 1 ELSE 0 END)
```

(33)

デバッグ方法

SQL は利用するデータベースに依存します。発行される SQL を確認しながら記述してください。 SQL の確認方法は"Wagby Developer Network > Javaを用いたカスタマイズ > 発行されるSQLを確認する"をお読みください。

仕様・制約

SQLの戻り値は1つの値であること

SQL式によって複数の結果セットが戻るような場合には対応できません。

(33) この設定は「一覧画面で項目をクリックすることで項目順にソート」にも対応しています。

データベースをまたぐSQLは記述できない

サブデータベースを利用時、メインデータベースとサブデータベースをまたぐ SQL 式を記述することはできません。（サブデータベースはメインデータベースと別のデータベースとなりますので、一つのSQL文の中で扱うことができないためです。）このような場合はスクリプトを利用するとよいでしょう。

SQL式の項目を一覧表示のソート項目にしない

一覧表示のソート（並び替え）は SQL の ORDER BY 句で実現しています。ここに SQL 式を適用するとインデックスが使われないためパフォーマンス低下の一因になります。

複雑なSQLは避ける

複雑なSQL式では、Hibernate の解釈によって接頭語 "this_." が含まれることがあります。開発者がこの制御を行うことは難しいため、このような場合は別途、データベースのビューを用意し、SQL式は単純にする（ビューの項目を示させるのみとする）アプローチを推奨します。

項目ルールを定義する（データ辞書としての利用）

データ辞書

「データ辞書」とは、開発するアプリケーションが管理するデータの説明文をまとめたものを指します。一般的にはデータの型や有効桁数といった情報を一元的に管理する目的で整備します。入力ルールや説明用の補助情報といったものを含めて管理できれば、なお良いとされています。Wagbyの設計情報（リポジトリ）は、データ辞書そのものです。

「データ辞書」は、データ同士の関連情報を持っています。そのため、例えばモデル項目を削除しようとしたとき、この項目を参照している影響範囲を調べるといったことにも活用できます。

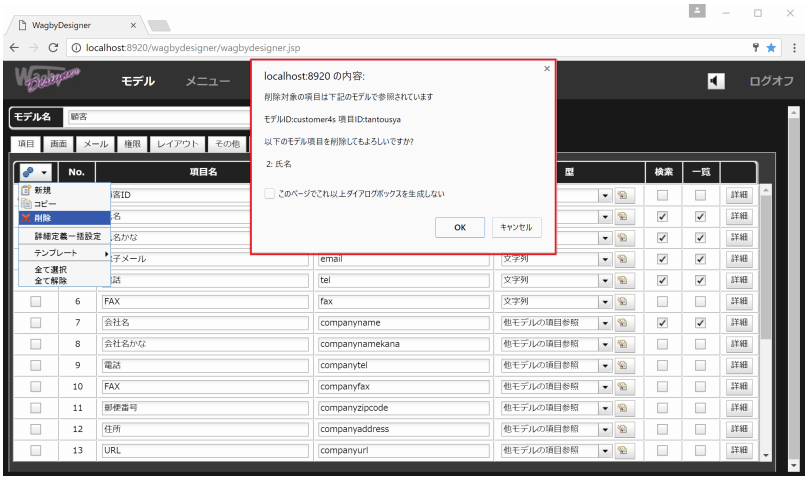


図 モデル項目削除時の確認ダイアログ（関連する項目を確認できる）

項目ルールを定義する

項目ルールとは

モデル項目に関するいくつかの設定を、複数の項目に同じように設定するというケースを考えてみます。例えば、次のようなルールです。

・　日付型項目で、和暦情報を常に表示する。

Wagbyでは、共通で利用できる設計情報を部品化することができます。これを「項目ルール」と呼びます。 ここでは、定義方法と使い方を説明します。

定義方法

Designerの「項目ルール」タブを選択します。

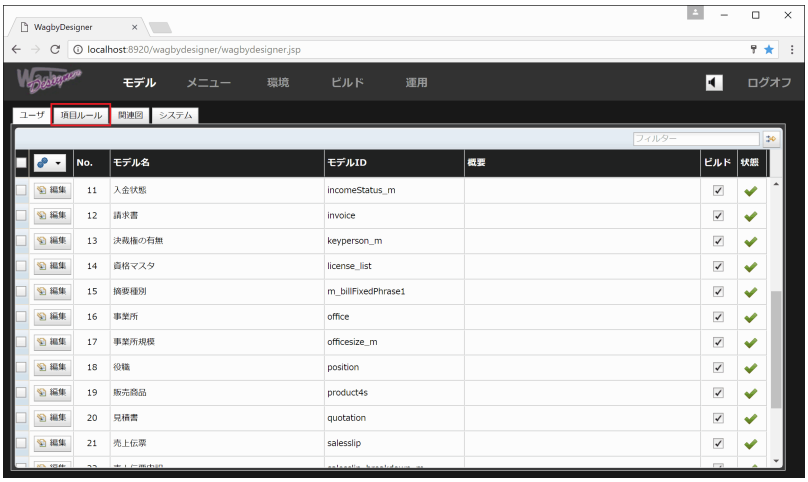


図 「項目ルール」タブ

初期状態では、項目ルールはありません。左上のギアアイコンから「新規」を選択します。

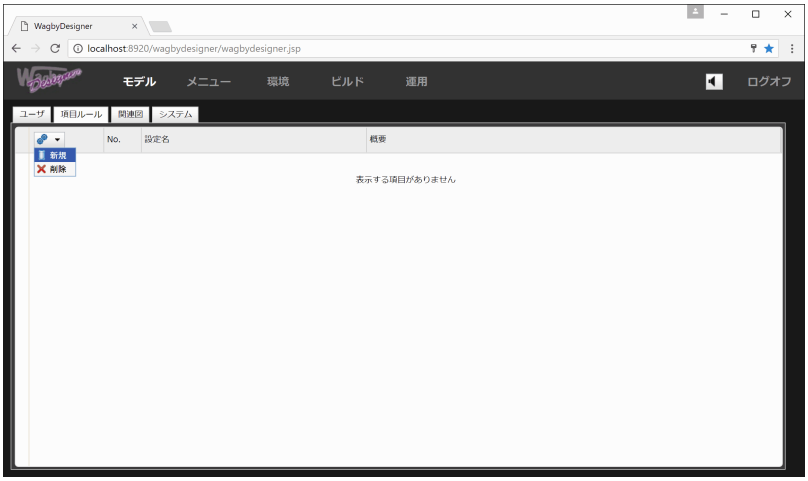


図 新しい項目ルールを作成する

項目ルール設定ダイアログが開きます。

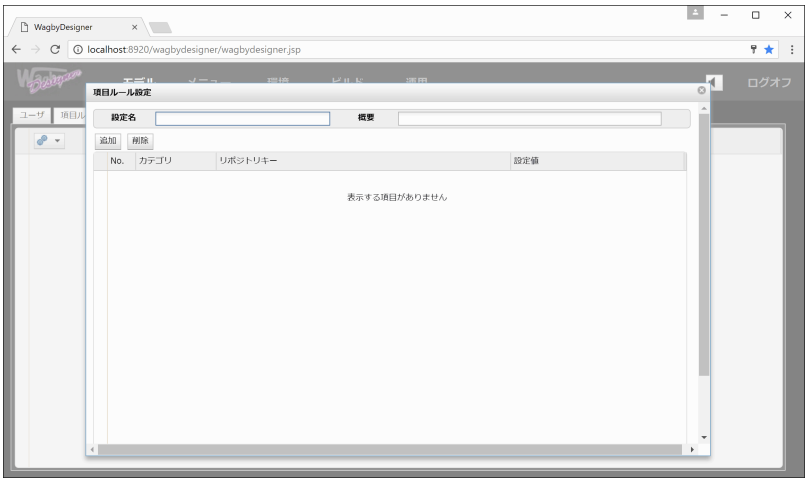


図 項目ルール設定ダイアログ

「設定名」と「概要」を用意します。設定名は必須項目です。

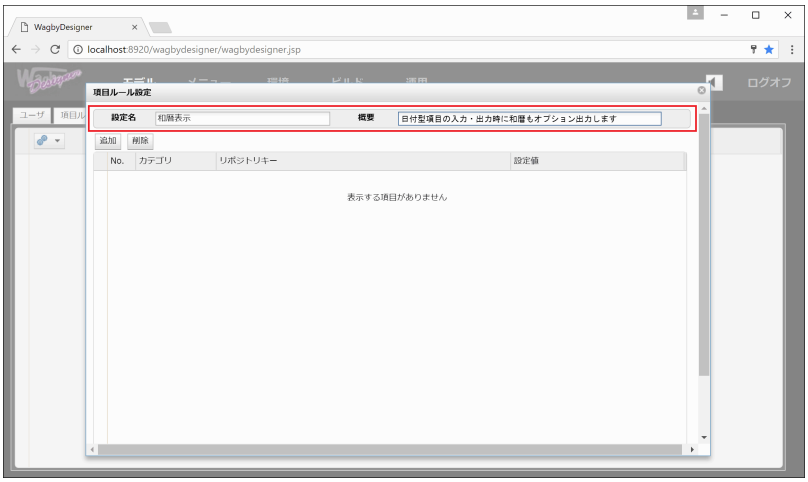


図 「設定名」と「概要」を用意する

リポジトリを用意します。「追加」ボタンを押下します。

図 リポジトリを用意する

モデル項目詳細設定ダイアログのタブ名が「カテゴリ」として選択できます。はじめに、このカテゴリを指定します。

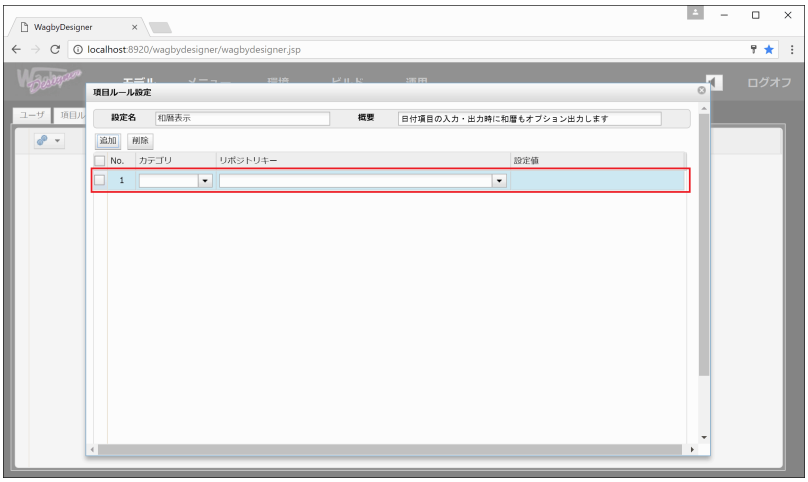


図 カテゴリを選択する

選択したカテゴリ内に含まれる、リポジトリー覧が表示されます。設定したいリポジトリを選択してください。

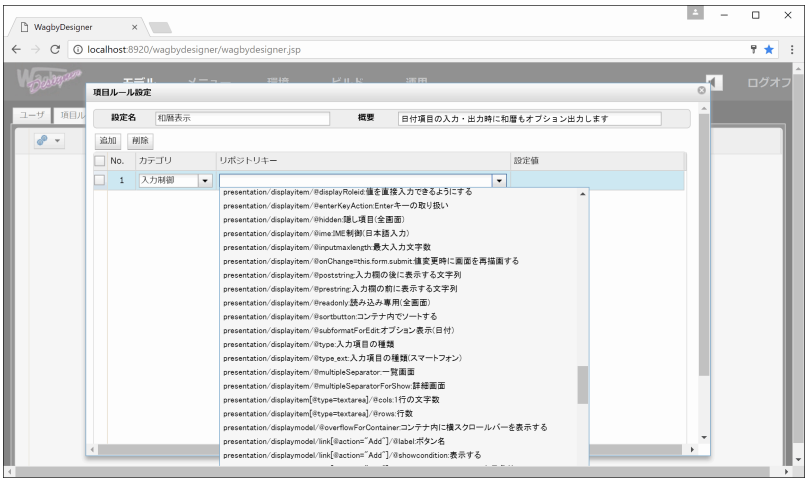


図 リポジトリー覧

すべてのリポジトリは「リポジトリ構造とチーム開発 〉 リポジトリー覧をご覧ください。

続いて「設定値」を指定します。リポジトリによって、選択式か、あるいは手動入力となります。

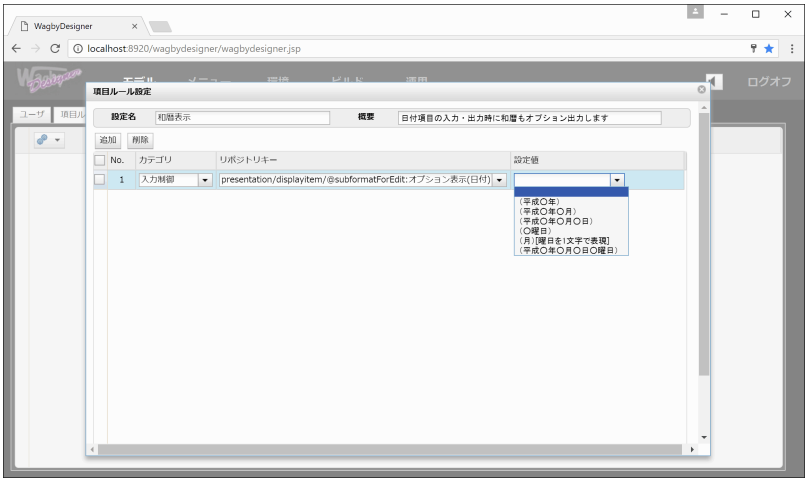


図 設定値を指定する

一つの項目ルールに、複数のリポジトリ設定を含めることができます。この例では入力フォーマットと出力フォーマットの療法を設定しています。

図 複数のリポジトリ設定を行う

「保存」ボタンを押下し、この設定を保存します。

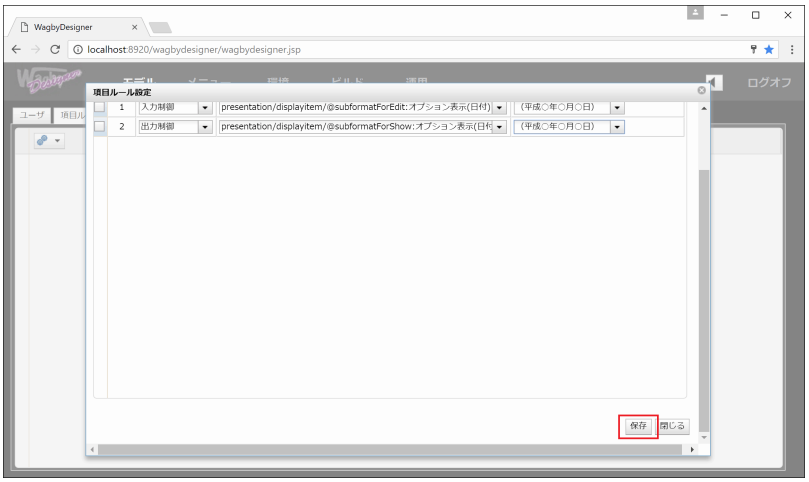


図 保存ボタン

項目ルールを一つ、作成しました。同じ手順で、複数の項目ルールを用意することができます。

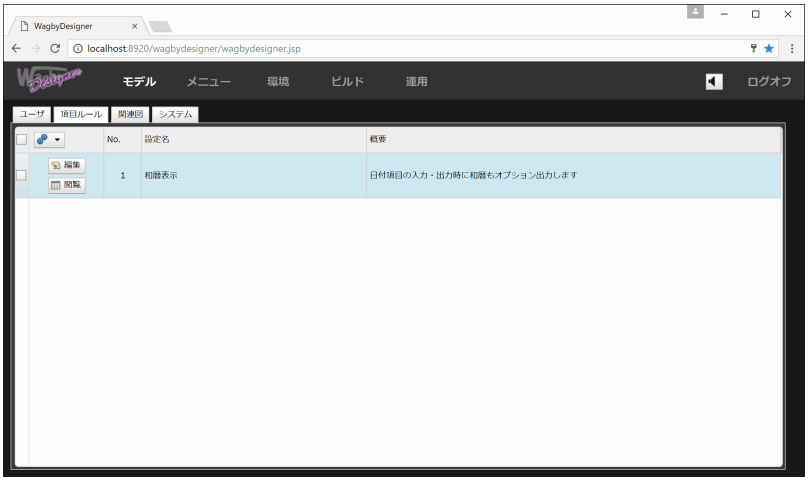


図 作成した項目ルール

優先順位

複数の項目ルールを用意したとき、先頭から順に適用の優先度が高くなっています。同じリポジトリキーに関する設定を含んでいたとき、より上位の項目ルールに含まれているリポジトリ設定が有効になります。

図で、ドラッグ&ドロップによって項目ルールを並び替えることができます。

項目ルールを使う

モデル項目詳細設定ダイアログの「項目ルール」タブを開きます。用意した項目ルールを適用することができます。

図 項目ルールタブ

項目ルール名にチェックすることで、適用できます。

図 適用したい項目ルールを選択する

一つのモデル項目に、複数の項目ルールを適用することもできます。

設定後、ビルド処理を行ってください。項目ルールが反映されます。

項目ルールの適用状況を確認する

図で、項目ルール設定タブの「閲覧」ボタンを押下すると、この項目ルールが適用されているモデルおよびモデル項目一覧を表示することができます。これによって、どの項目が適用済みかを把握することができます。

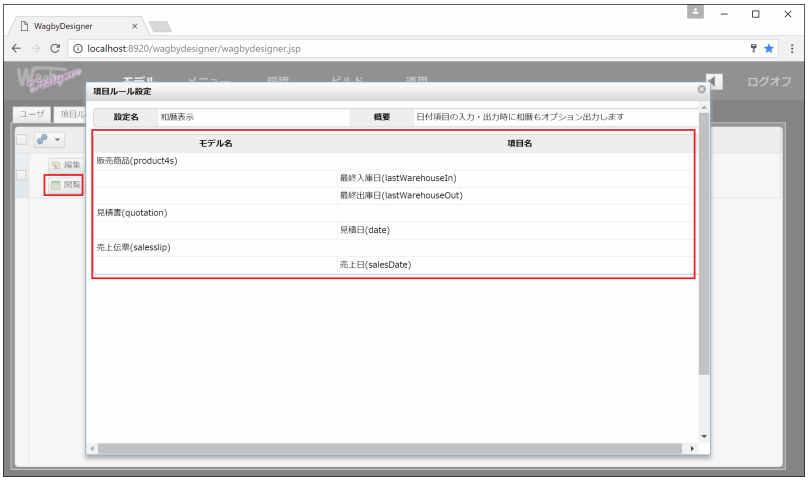


図 項目ルール適用状況を確認する

項目ルールの内容を変更する

図で、項目ルール設定タブの「編集」ボタンを押下すると、リポジトリ情報の追加・変更・削除を行うことができます。

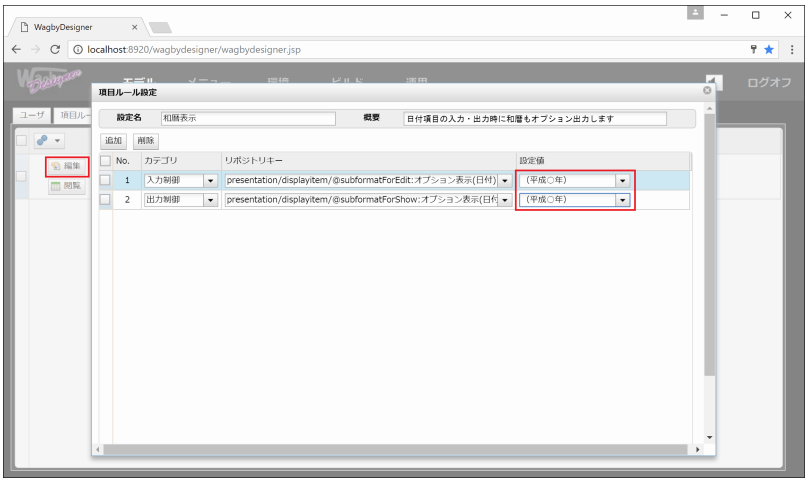


図 リポジトリ情報の編集

編集後、ビルド処理を行ってください。適用されているすべての項目が一斉に変更されます。

名前と型の扱い

モデル項目の一覧で設定する「項目名」「型」「検索（条件に利用する）」「一覧（表示に利用する）」を設定することができます。

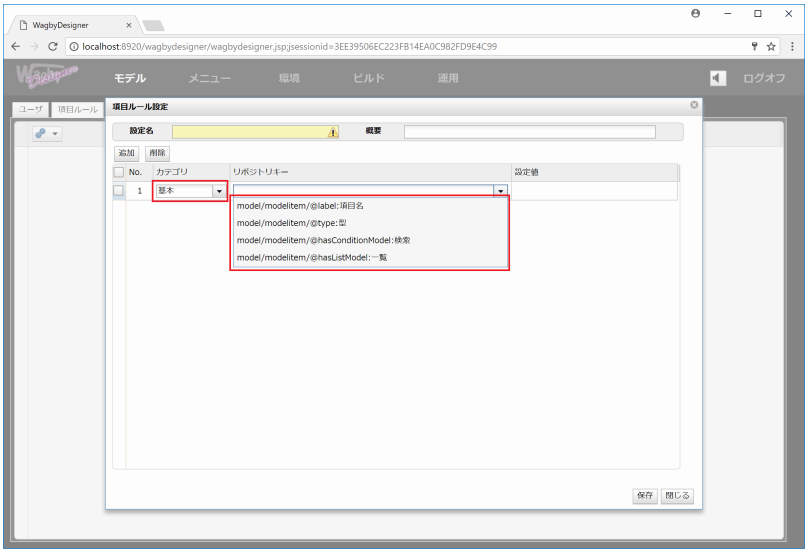


図 基本カテゴリの設定欄

型の設定

型の設定では、次の値を入力することができます。

区分	設定できる型
文字列	"文字列", "郵便番号", "メールアドレス", "URL", "ファイル名"
数値	"整数", "1バイト整数", "2バイト整数", "4バイト整数", "8バイト整数", "4バイト浮動小数点", "8バイト浮動小数点"
日付・時刻	"日付", "日付と時刻", "時刻"
他モデルの項目参照	"モデル参照(リストボックス)", "モデル参照(ラジオボタン)", "モデル参照(チェックボックス)", "モデル参照(検索画面)", "参照連動"
繰り返しコンテナ	"(繰り返し項目のコンテナ)"
繰り返しコンテナID	"(繰り返し項目コンテナ用のID)"

(34)

型の設定に関する注意点

Designerで型を変更した場合、いくつかの設定は初期状態に戻ります。しかしこの項目ルール設定では、型を変更してもその他の設定は初期状態に戻さず、維持されます。そのため、型の変更に伴い影響のありそうな関連設定は、開発者自身で注意深く見直しを行なってください。

具体的には、型の変更を行なったとき、この項目ルールを適用しているすべての項目について、次の設定をチェックすることを推奨します。

- 検索画面の範囲検索。（数字型や日付型から文字列型に変更した場合でも、範囲検索の設定は残っている。）
- 入力チェック。許容文字種を含む。
- 暗黙条件。
- 初期値。
- 日付、数値フォーマット。
- 入力フィルタ。
- 主キーの順序設定。
- 絞り込みの設定。
- 入力制御。（入力可・不可の式、必須、禁止文字、正規表現など）
- 計算式。

(34) 実際に設定する場合は、ダブルクォーテーションは除いてください。

- データベースのユニーク制約。
- ロック名。
- トランザクション設定。
- SQL式。

このように型の変更に伴う影響は大きいため、項目ルールにおける型の指定は「将来、変更する可能性」を考えるのではなく、「この型を使っている項目一覧を洗い出す」目的で利用することがよいでしょう。

仕様・制約

- 項目ルールに関する設定情報は、repository/trunk フォルダ内の .userDefinedItem フォルダに格納されます。
- モデル項目には model/modelitem/userDefinedItem というリポジトリキーで、どの項目ルールを適用するかという情報を含みます。複数適用時はカンマ区切りとなります。
- 項目ルールは、モデル項目詳細設定ダイアログで指定可能なリポジトリキーが対象となります。画面、レイアウト、権限、メニュー、環境設定などに関するものを含めることはできません。

関連する機能

- 「Designerの機能・その他のツール ＞ 項目テンプレートを用意する」

入力フィルタの仕様

入力フィルタの選択

Wagbyでは、入力された値は「フィルタ」によって補正されます。　フィルタにより「ゆれ」を自動的に補正することができます。この機能は自動的に設定されます。

(35)

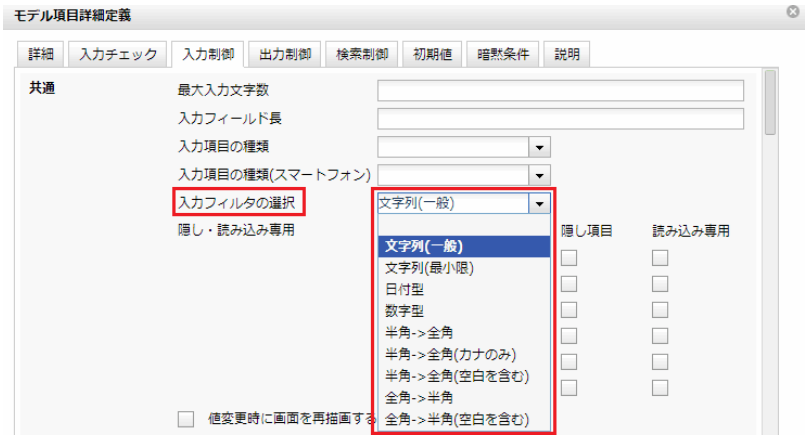


図 文字列型項目の入力フィルタの設定

文字列（一般）

入力値を「文字列」と判定するフィルタです。日本語が含まれる値が入る項目は「文字列（一般）」フィルタを選択することを推奨します。次のような処理が行われます。

- 前後のスペース(半角/全角空白及びタブ)を削除
- 改行コードを "LF" に統一します。
- 全角アルファベットを半角にします。

(35) Web フォームから入力された文字列は、最初にこのフィルタ処理が適用されます。フィルタ適用後に計算処理が行われ、最後にデータベースへの更新が行われます。

・　数字を半角にします。
・　記号を半角にします。
・　半角カタカナを全角カタカナにします。
・　「-」(マイナス)、「ー」(長音)、「‐」(ダッシュ)、「-」(ハイフン)　　等の全角マイナス文字を「ー」(長音)に統一します。

すなわち、本指定は後述する「文字列（最小限）」「半角→全角」「全角→半角」を含んだものとなっています。

文字列（最小限）

改行コードを "LF" に統一します。

(36)

日付型

入力値を日付と判断して、日付型に変換するフィルタです。全角で入力された数値はすべて半角に統一します。

数字型

入力値を数値と判断して、数値型に変換するフィルタです。全角で入力された数値はすべて半角に統一します。

半角→全角

入力値に半角文字が含まれていた場合、この半角文字に対応する全角文字に変換します。

半角→全角（カナのみ）

入力値に半角カタカナが含まれていた場合、この半角カタカナを全角カタカナに変換します。

半角→全角（空白を含む）

入力値に半角文字が含まれていた場合、この半角文字に対応する全角文字に変換します。半角空白も全角空白に変換します。

全角→半角

入力値に全角文字が含まれていた場合、この全角文字に対応する半角文字に変換します。対応する半角文字がなければ変換はされません。

全角→半角（空白を含む）

入力値に全角文字が含まれていた場合、この全角文字に対応する半角文字に変換します。全角空白も半角空白に変換します。対応する半角文字がなければ変換はされません。

フィルタを「なし」とすることはできません。この欄を空白とした場合、フィルタなしではなく、型に応じた標準のフィルタが自動選択されます。例えば文字列型の場合は「文字列(一般)」が適用されます。全角英数字や半角カナ文字をそのまま保存したい場合は「文字列（最小限）」を選択してください。

フィルタはWebフォーム、アップロード更新、REST APIいずれも共通で利用されます。

(36) 改行コードを "LF" へ統一することで、ユーザが利用するクライアント OS にかかわらず、データベースに格納された文字列は、常に改行文字は LF であることを保証します。

検索時の入力フィルタ（8.4.0）

入力時（登録・更新画面）と検索画面でフィルタを個別に指定することができます。モデル項目詳細設定ダイアログの「検索制御」タブで「入力フィルタ」を指定できます。

(37)

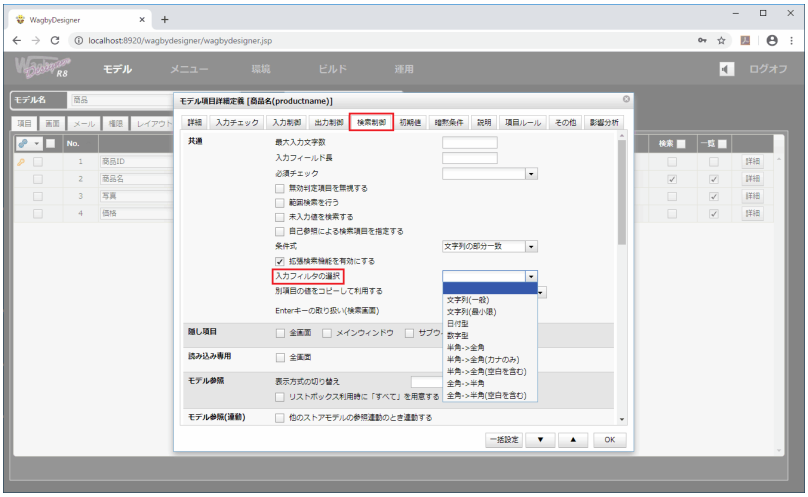


図 検索制御の入力フィルタ指定

詳細な仕様

各フィルタの詳細な仕様は次のとおりです。（数値はフィルタ処理の順番です。）

	文	文（小）	数	日	半→全	半→全（カ）	半→全（空）	全→半	全→半（空）
XML文字	1	1	1	1	1	1	1	1	1
空白除去	2		2	2	2	2	2	2	2
改行除去	3	2	3	3	3	3	3	3	3
英全→半	4			4				4	4
英半→全					4		4		
数全→半	5		4	5				5	5
数半→全					5		5		
記全→半	6		5	6				6	6
記半→全					6		6		

(37) この設定が空白の場合は、入力制御のフィルタ指定値を使います。

仮 全→ 半								7	7
仮 半→ 全	7				7	4	7		
マ 全→ 半			6	7				8	8
マ 半→ 全					8		8		
正 半マ								9	9
正 全マ	8		7	8	9		9		
空 全→ 半									10
空 半→ 全							10		

- 「XML文字」とは、"&#xxxx;" 形式でWebブラウザから送られてきたUnicode文字をJavaが扱える文字に変換するフィルタです。IE など一部のブラウザがこのような送信を行うことがあります。
- 「空白除去」は、入力文字列の前後にある半角、全角、タブ文字を取り除きます。
- 「改行除去」は、改行文字を "LF" に統一します。
- 「英 全→半」は、全角のアルファベット文字を半角に置き換えます。
- 「英 半→全」は、半角のアルファベット文字を全角に置き換えます。
- 「数 全→半」は、全角の数字文字を半角に置き換えます。
- 「数 半→全」は、半角の数字文字を全角に置き換えます。
- 「記 全→半」は、全角の記号文字を半角に置き換えます。
- 「記 半→全」は、半角の記号文字を全角に置き換えます。
- 「仮 全→半」は、全角の仮名文字を半角に置き換えます。
- 「仮 半→全」は、半角の仮名文字を全角に置き換えます。
- 「マ 全→半」は、全角のマイナス（長音）文字を半角に置き換えます。対象となる全角文字は次のとおりで
  す。'¥u2212','¥u2015','¥u30fc','¥uff0d','¥u2500','¥u2014','¥u2010','¥u2011','¥u2012','¥u2013'
- 「マ 半→全」は、半角のマイナス文字を全角('¥u30fc')に置き換えます。
- 「正 半マ」は、半角マイナス文字を '-' に統一します。
- 「正 全マ」は、さまざまな全角のマイナス（長音）文字を'¥u30fc'に統一します。
- 「空 全→半」は、全角スペース文字を半角スペース文字に置き換えます。
- 「空 半→全」は、半角スペース文字を全角スペース文字に置き換えます。

参照連動項目の入力フィルタ

参照連動項目にもフィルタが適用されます。 詳細は"他モデルの参照 > 参照連動 > 参照連動項目の入力フィルタ"をお読みください。

標準のフィルタルールを変更する

標準のフィルタルールをカスタマイズすることができます。[詳細...]

他モデルの参照

選択肢モデル

選択肢モデル

選択肢モデルは、モデル項目として「主キー」「内容」「表示優先度」「有効期限」という項目が予め定義済みとなっています。 開発者はこのモデル定義を雛形（テンプレート）として、選択肢の内容を格納するために用いることができます。

項目名(日本語)	項目名(英語)	項目の型	役割
主キー	id	数値文字列日付・時刻	主キー
内容	content	文字列	画面に表示される文言
表示優先度	priority	数値	リストボックス等の並び順
有効期限	valid	文字列	何か値がセットされていた場合には無効扱いとなる

通常、これらの選択肢はリストボックス、ラジオボタン、チェックボックスのいずれかの方式で利用します。

テーブルを作成しない選択肢モデル

選択肢モデルの「初期データで固定とする」を有効にすると、運用中に選択肢を変更することができなくなります。固定値として扱うため、内部ではテーブルを作成せず、メモリ上で管理するようになります。

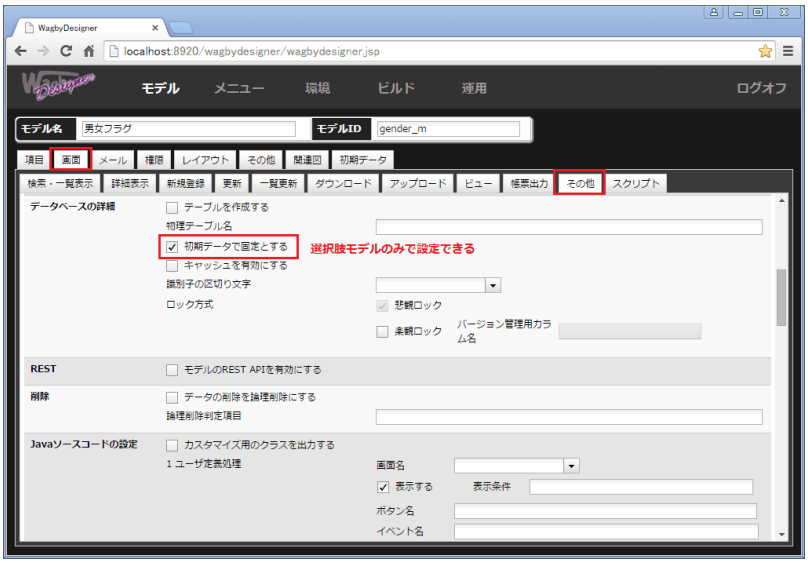


図 初期データで固定とする

このモデルには、次のルールが適用されます。

- テーブルを作成しない。
- キャッシュを無効とする。（常にメモリ上の値を使うため）
- 詳細画面、登録画面、更新画面、ダウンロード画面、アップロード更新画面、一覧更新画面は作成されない。（運用中の値の追加・変更・削除は行えない。）
- 一覧表示画面では、検索条件部は表示されない。常に一覧表示となる。
- 一覧表示画面の同時表示数の初期値は100となる。最大100個のデータが一画面に表示される。
- 他モデルの参照 > 他項目の値で選択肢を絞り込む 設定に用いることはできない。

・　「モデル参照（検索画面）」として用いる場合、メインウィンドウの入力値を検索条件として再利用する指定を行うことはできません。（検索条件は無視されます。）

(38)

本設定は「選択肢モデル」のみで利用できます。（通常のモデルでは利用できません。）

設計上の注意点

この設定を行うと、内部では一切のデータベース操作を無効として扱います。システム稼働中のデータの追加、更新、削除も行えません。検索も検索条件を指定した絞り込みは行えず、常に全件のデータを取得します。

そのため、次の設定と組み合わせることはできません。

・　外部キー
・　メインモデルまたはサブモデル
・　全文検索
・　ワークフロー
・　楽観ロック
・　カレンダビュー、集計ビューをはじめとする各種ビュー
・　暗黙条件
・　グループ権限
・　この選択肢モデルのIDをキーに、自身のモデルの項目（内容、表示優先度など）を参照連動するような使い方

これらの制約から、本設定は選択肢としての初期データとして扱うのみとしてご利用ください。

運用上の注意点

・　選択肢モデルに項目を追加することはできません。同様に、用意されている項目を削除することもできません。
・　「テーブルを作成しない選択肢モデル」をサブデータベースで用意することはできません。

仕様

・　アプリケーション起動時に WEB-INF¥export¥init¥<モデルID> フォルダ内に格納されている xml ファイルを読み込み、メモリに格納します。

入力フィルタ

選択肢モデルの内容部の入力フィルタは ″文字列(一般)″ となっています。そのためDesignerの内容部に全角文字を入力した場合でも、インポートされる初期データは半角文字に変換されます。

入力フィルタを変更する

モデル項目一覧の左上のギアアイコンを押下します。表示されるドロップダウンメニューから「詳細定義一括設定」を選択すると「モデル項目詳細定義一括設定」ダイアログが表示されます。このダイアログから設定を変更する方法を説明します。

1.　「モデル項目詳細定義一括設定」ダイアログのカテゴリで「入力制御」を選択します。
2.　リポジトリキーに「@filter」または「入力フィルタの選択」を入力します。
3.　選択肢に「model/modelitem/@filter:入力フィルタの選択」が表示されるので、選択します。
4.　変更対象の「内容」行の左側のチェックボックスにチェックを付けます。
5.　設定値のリストボックスから「文字列(最小限)」を選択します。
6.　「保存」ボタンを押下します。
7.　更新確認ダイアログで「はい」を押下すると変更した設定値が保存されます。

選択肢モデルの作成

「業種」を格納するための選択肢モデルを用意する例を示します。

(38) Designer上の設定は無視されます。強制的に上のルールが適用されます。

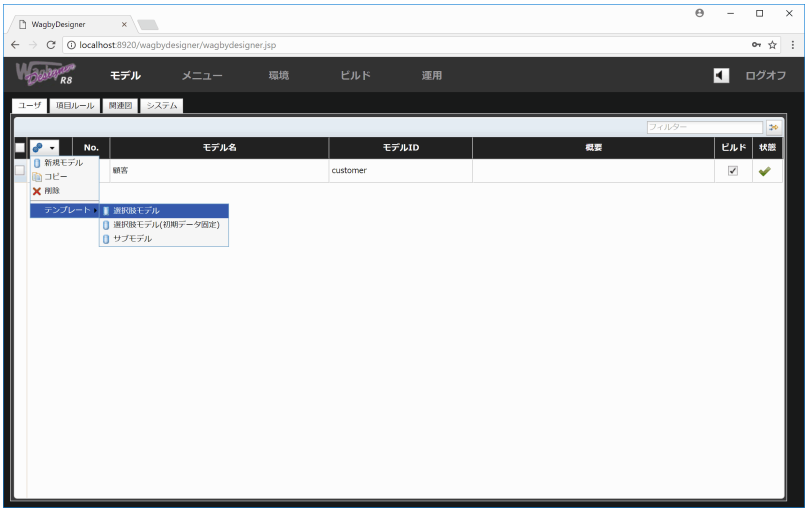


図 選択肢モデルを作成する

「選択肢モデル」「選択肢モデル(初期データ固定)」のいずれかを用います。「選択肢モデル(初期データ固定)」の場合、運用中に選択肢を変更することはできません。男女フラグなど、変更する予定のない場合にご利用ください。[詳細...]

選択肢モデルは標準で4つの項目「主キー」「内容」「表示優先度」「有効期限」が予め定義済みとなっています。 (これらの項目の名前およびIDは変更できません。型は主キーのみ変更できます。)

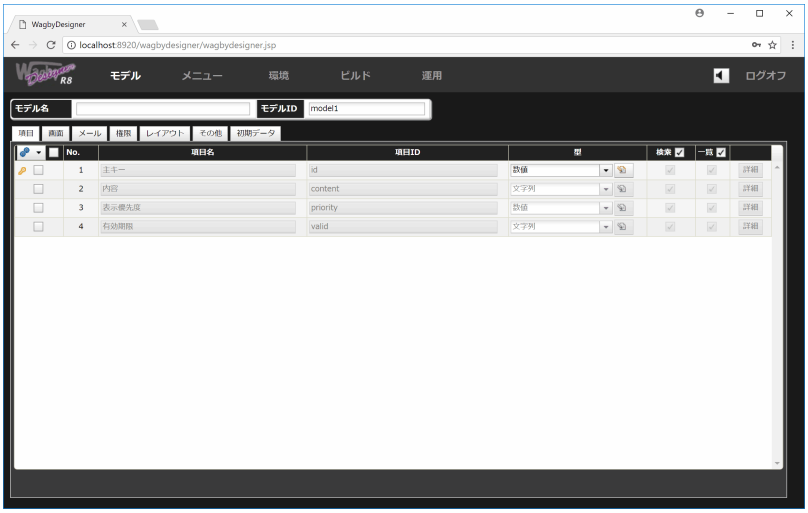


図 選択肢モデルの標準形

ここではモデル名を"業種"とし、モデルIDを"jobtype_m"と設定します。 ("_m"は、このモデルが選択肢であるという意味を含めましたが、必ずしもこのような名前にすることはありません。)

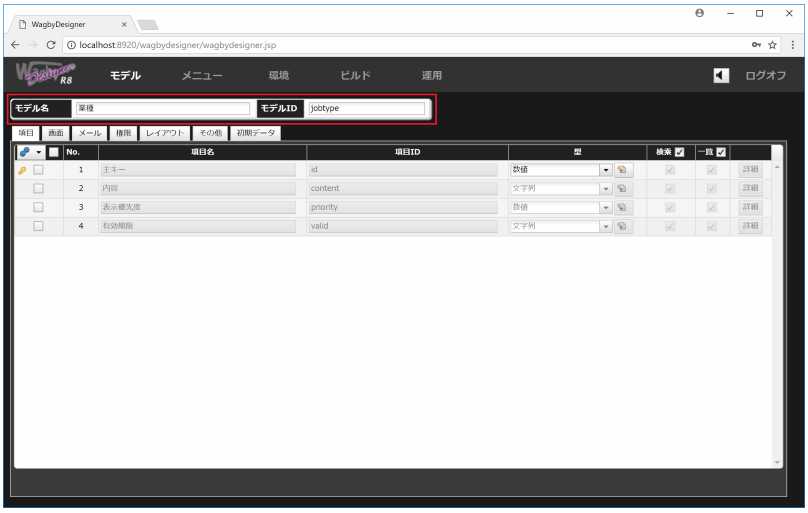


図 モデル名とモデルIDの設定

主キーのみ型を変更できます。「文字列」「数値」「日付・時刻」から選択できます。標準は数値です。

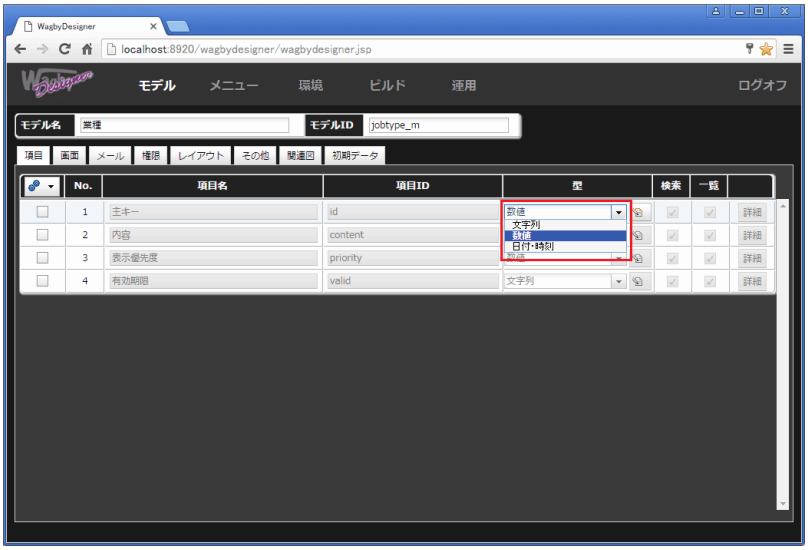


図 型の指定

選択肢モデルは特別に、初期値を設定することができます。「初期データ」タブを押下すると図の入力画面が開きます。

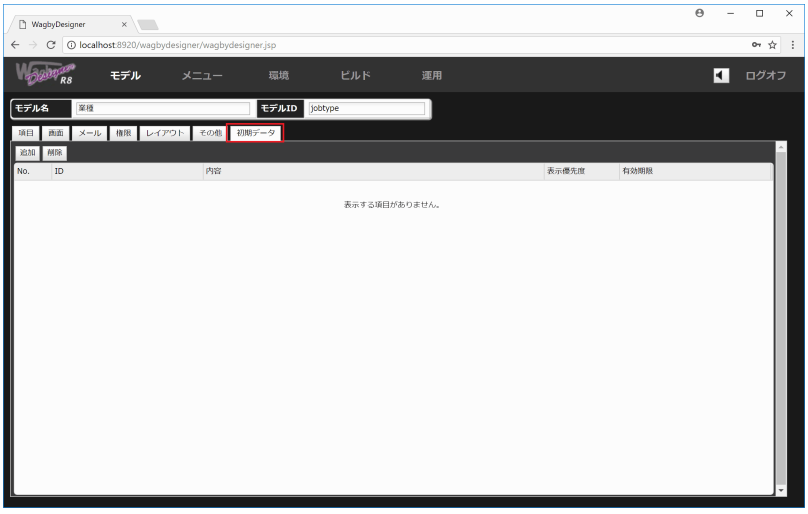


図 選択肢の初期値を設定する

「追加」ボタンを押下することで、複数個の初期値を設定できます。左欄の「No.」は固定（連番）ですが、これは表示のためだけに用いられるもので、データベースには格納されません。「ID」が主キーとなります。

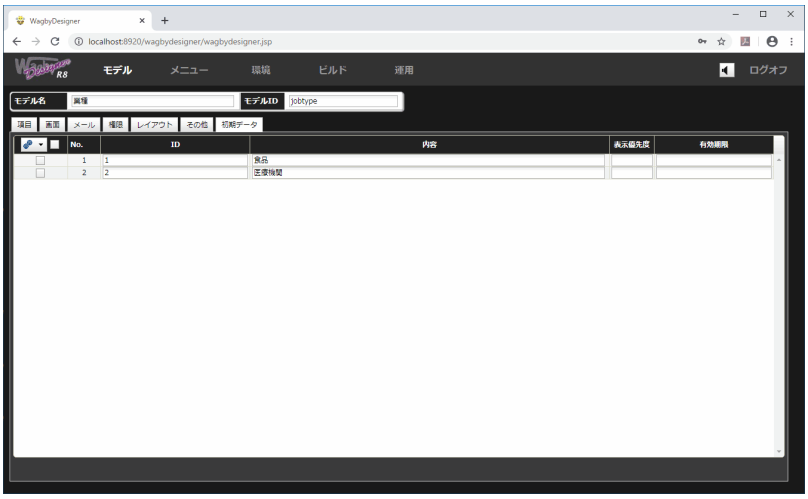


図 「内容」項目の初期値を設定する

運用を開始したあとでも、選択肢の値を追加・変更することができます。詳細は「${content_3_name}」にて説明します。

選択肢モデルの参照方法

顧客モデルに、先ほど追加した業種を参照する例を説明します。

項目名を"業種"とします。項目IDを"job"とします。型は"他モデルの項目参照"を指定します。

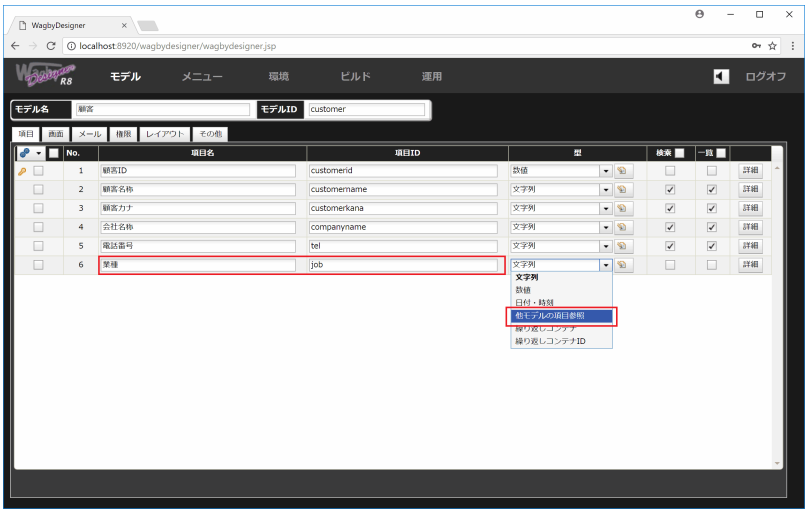


図 業種モデル項目の情報を設定

型に「他モデルの項目参照」を選択すると、図「モデル項目の型詳細設定」ダイアログが表示されます。

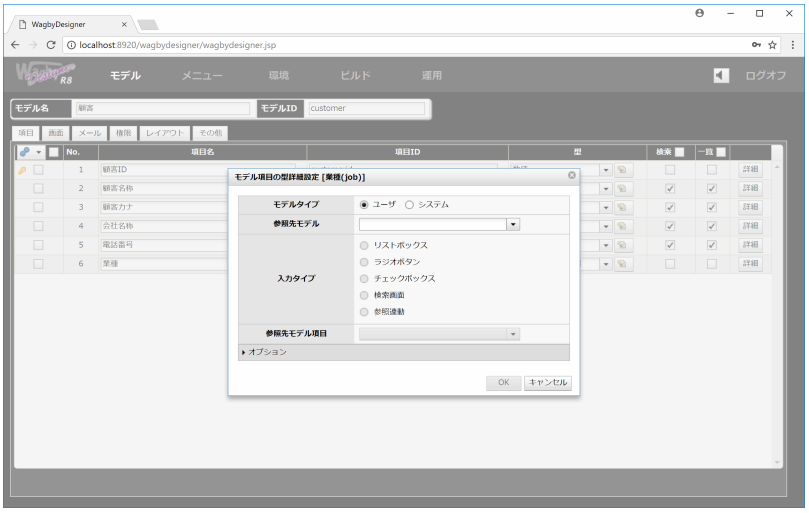


図 モデル項目の型詳細設定

参照先モデルとして、図で用意した「業種」モデルを選択します。

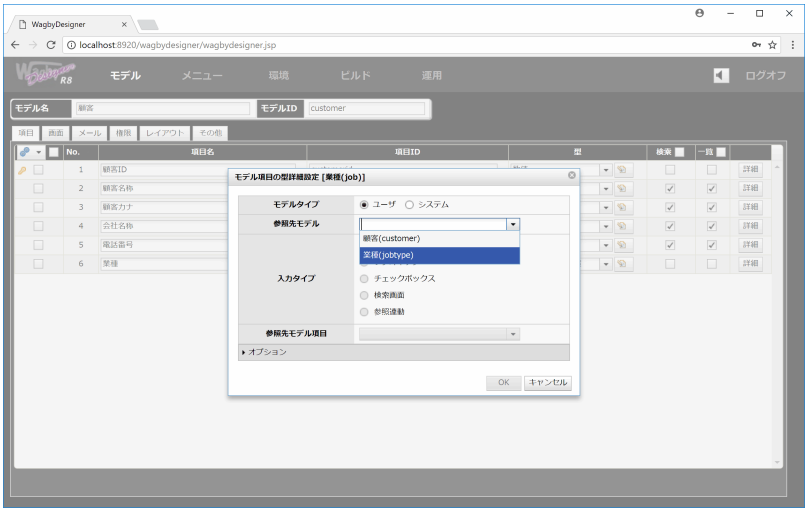


図 参照先モデルの指定

入力タイプを「リストボックス」と指定します。また参照先モデル項目は、「内容」を指定します。

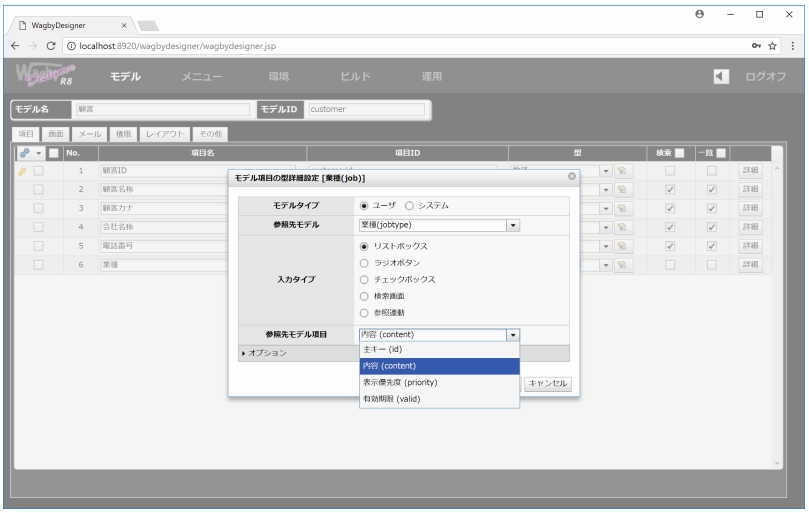


図 入力タイプと参照先モデル項目の指定

さらにオプションを設定します。表示優先度項目には選択肢モデル側に用意された「表示優先度」を、無効判定項目には「有効期限」を、それぞれ紐付けます。

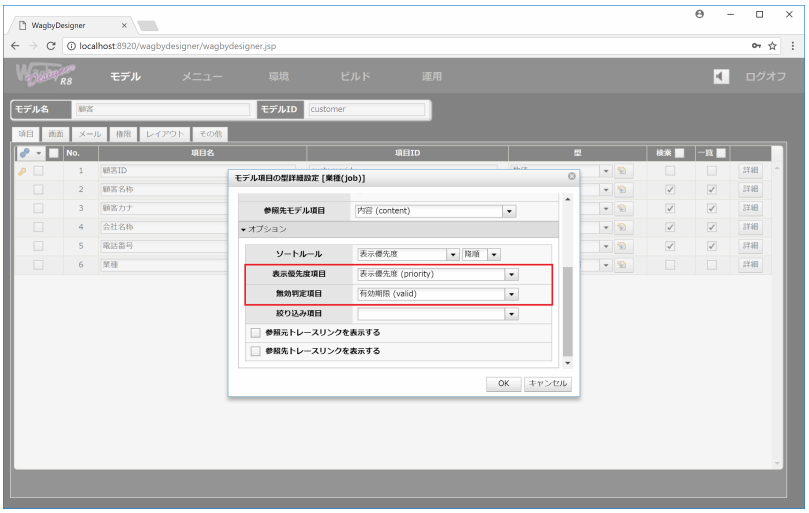


図 表示優先度項目と無効判定項目の指定

標準では、選択肢モデルを参照する項目では「表示優先度」と「有効期限」が自動的にセットされます。

リストボックスを用いた参照

ビルドしたアプリケーションを確認します。「業種」項目がリストボックスで表示されています。また初期データが読み込まれていることがわかります。

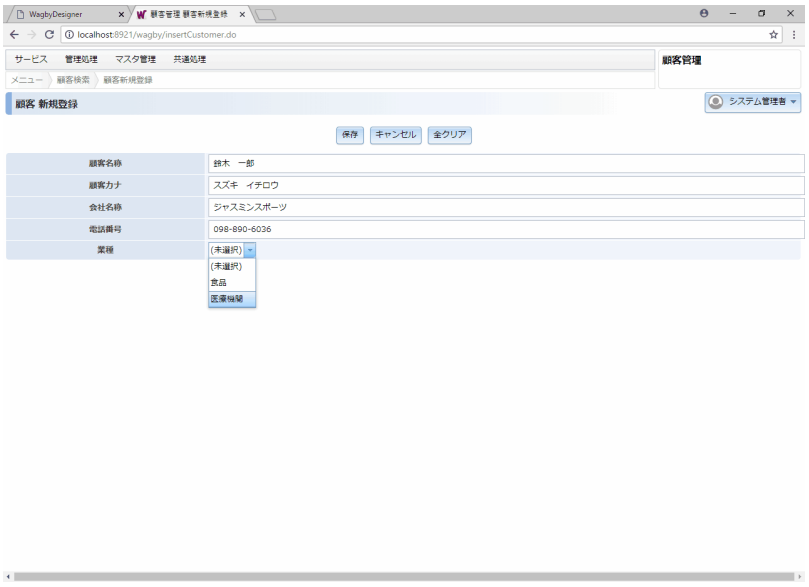


図 リストボックスによる表示

選択肢の中に　”(未選択)”　が自動的に加わっています。これはいずれも選択しない状態を意図しています。この ”(未選択)” を除去することもできます。[詳細...]

ラジオボタンを用いた参照

ラジオボタンを使った例を図に示します。

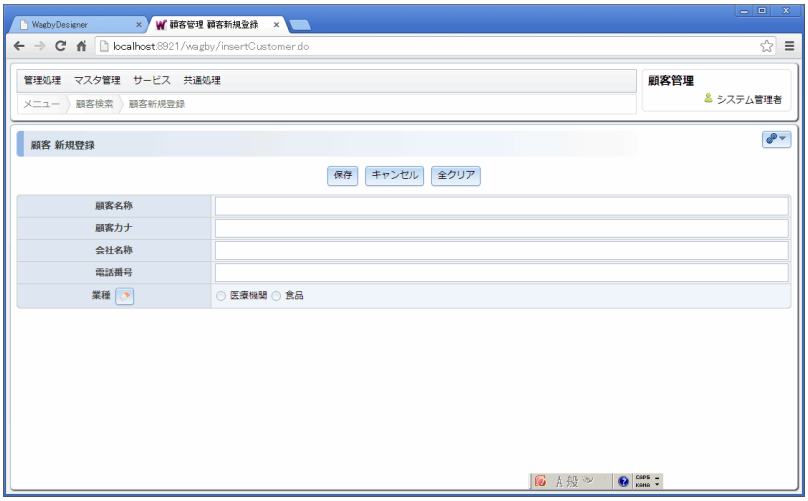


図 ラジオボタンによる表示

ラジオボタン利用時は、一度何かを選択すると、全解除（一つも選択しない状態）することができません。 そのため、図に示す消しゴムアイコンが用意されています。このアイコンを押下すると全ての選択を解除することができます。

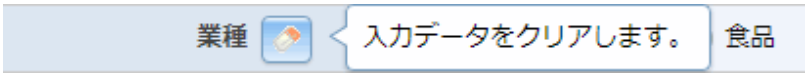


図 消しゴム機能

チェックボックスを用いた参照

チェックボックスを使った例を図に示します。

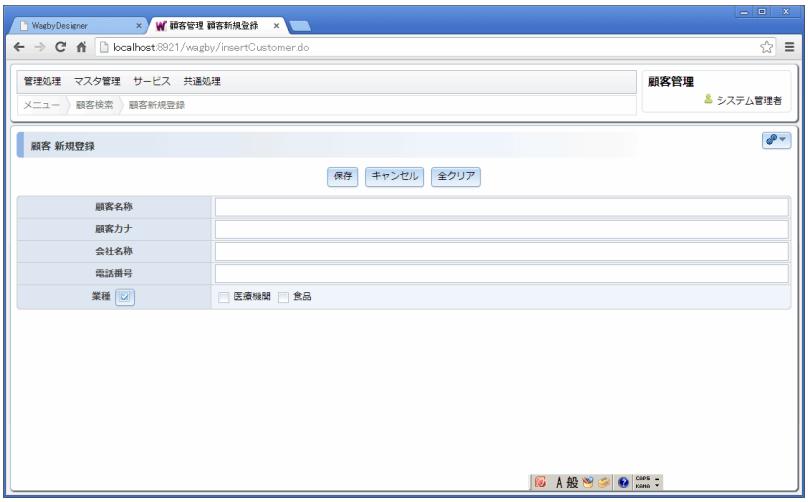


図 チェックボックスによる表示

すべての選択肢を一括して選択状態にすることができます。図に示す全チェックアイコンを押下します。

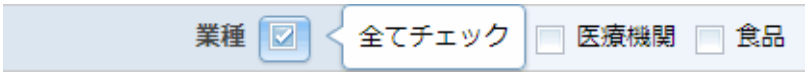


図 全チェックアイコン

チェック後は、アイコンの形が消しゴムに変わります。

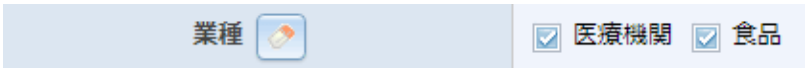


図 アイコンの形が消しゴムに変わった

消しゴム機能を使うことで、すべての選択を一度に解除することができます。

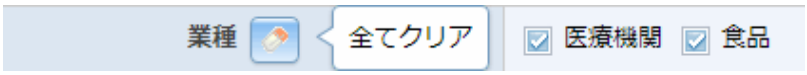


図 アイコンの上にマウスをのせると説明がポップアップで表示される

運用中に選択肢を追加する

選択肢の値は、運用中に追加することができます。 業種モデルの検索画面を選択します。一覧更新ボタンを押下します。

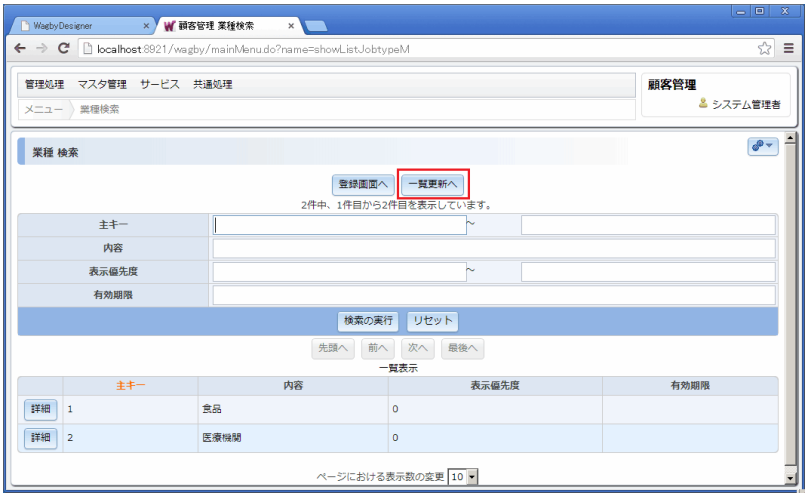


図 業種モデルの検索画面

一覧更新画面では、データの編集や新規登録を行うことができます。

図 業種モデルの一覧更新画面

任意の行の「新規」ボタンを押下すると、最下行に入力欄が用意されます。ここで新しい選択肢を入力し、画面上部の「保存」ボタンを押下します。

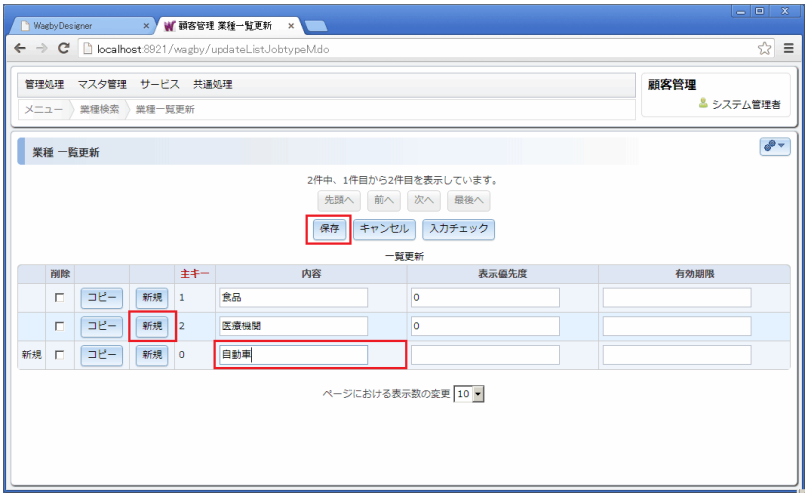


図 業種データを一件追加する

再び顧客モデルの新規登録画面を開きます。業種の選択肢が追加されていることがわかります。

図 業種の選択肢が追加された

選択肢の表示順を変更する

運用中に、選択肢の表示順を変えることができます。

再び業種の一覧更新画面を開きます。「表示優先度」の欄に、数字を設定します。この数字の大きい順に並び替えが行われます。

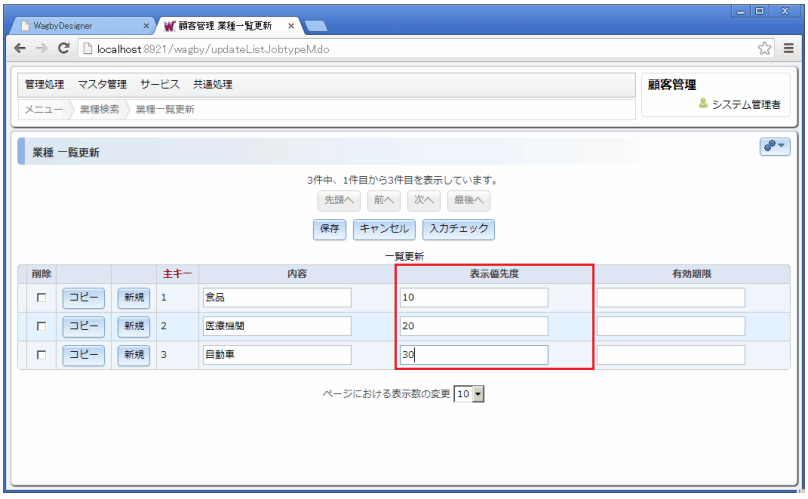


図 表示優先度を設定する

顧客データの新規登録画面を開くと、指定した並び順で選択肢が表示されていることを確認できます。

図 表示優先度によって選択肢が並び替えられる

表示順を変更する

標準は「降順（数字の大きい順）」ですが、これを「昇順（数字の小さい順）」ルールに変えることもできます。 （昇順は ″1, 2, 3, . . .″ であり、降順は ″10, 9, 8, . . .″ という意味です。）

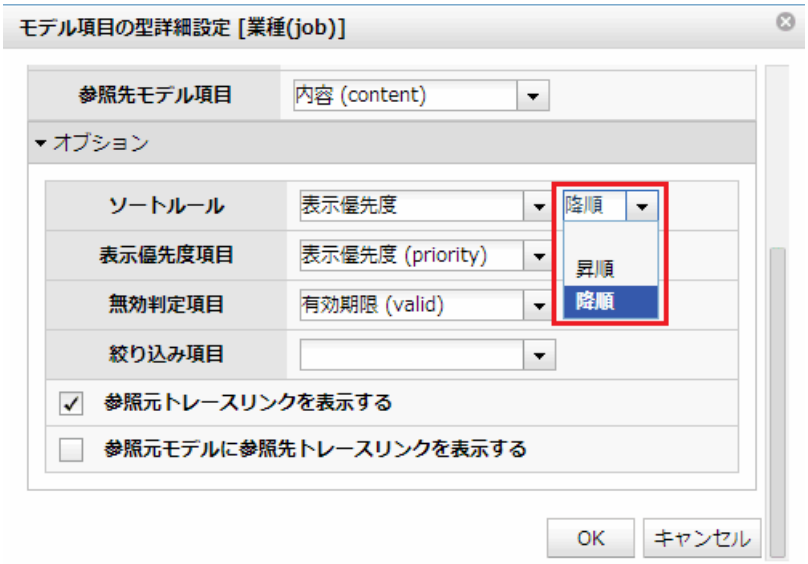


図 表示順の設定

ソートルールの詳細

表示優先度

「表示優先度項目」に指定した項目を指定します。型は整数型となっている必要があります。

ID

選択肢モデルの主キー値によってソートされます。「表示優先度項目」は用いられません。

内容

選択肢モデルの内容によってソートされます。文字列型の場合は辞書順になります。「表示優先度項目」は用いられません。

参照先モデルの固定ソートキー

（選択肢モデルでは設定できません。）

選択肢を無効にする

Wagby では選択肢のデータを削除することができないようになっています。選択肢はさまざまなモデルから参照されるという性質をもっているため、削除してしまうと参照先データが存在しないという不整合が発生するためです。その代わり、選択肢を無効にすることができます。ここでは、その方法を説明します。

運用中に、選択肢を無効にしてみます。 再び業種の一覧更新画面を開きます。「有効期限」の欄に、任意の文字を設定します。ここに文字が含まれていた場合、データは無効になります。

図 データの有効期限を設定する

有効期限を設定された選択肢を参照している顧客データを確認してみます。図 のように、選択肢データの横に「(-)」というマークが表示されています。これは「現在は無効となっている（古い）選択肢データを参照している」ことを意味します。

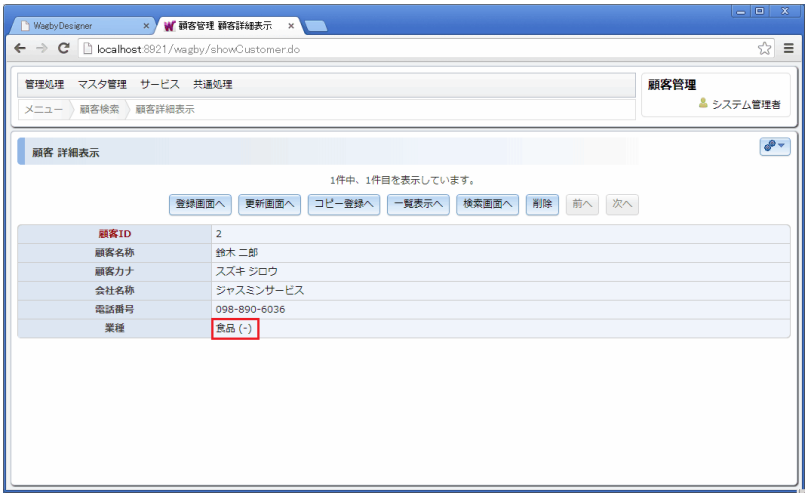


図 無効な選択肢を参照しているデータの表示

この顧客データを更新してみます。業種の欄には、現在の無効データを含めた、選択可能なリストが表示されます。

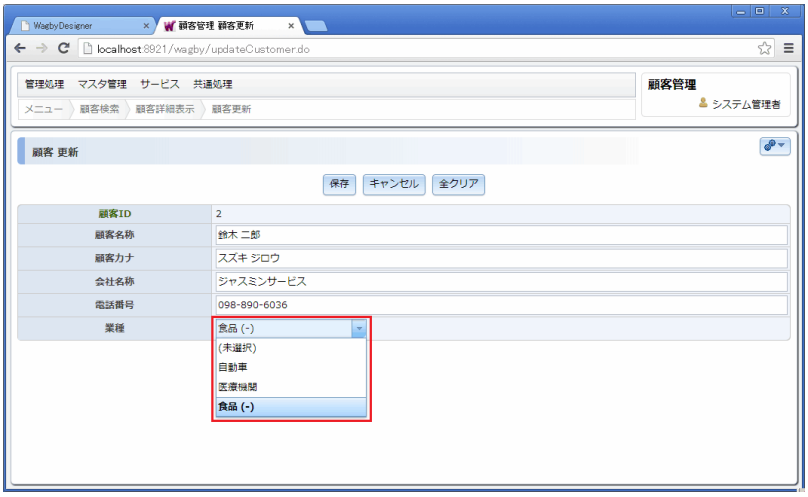


図 現在利用可能な選択肢の一覧

新しい業種を選択します。

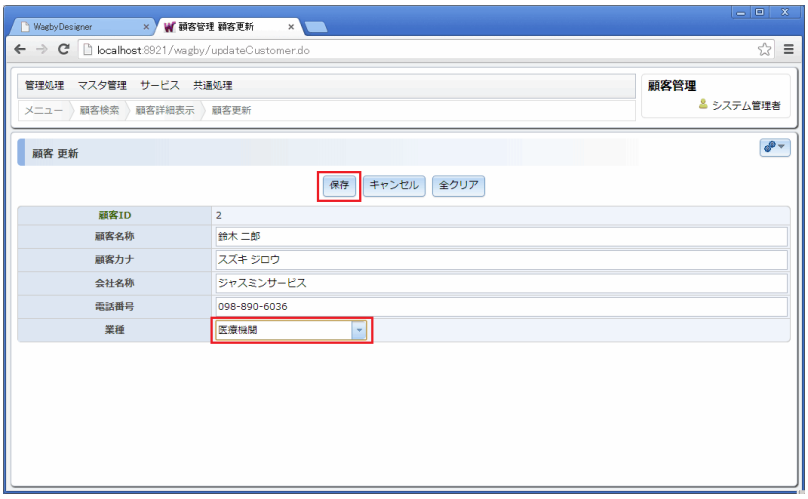


図 新しい業種を選択する

再び、同データの更新を行ってみます。すると、無効となった業種のデータは、もはや選択肢欄には表示されないことがわかります。

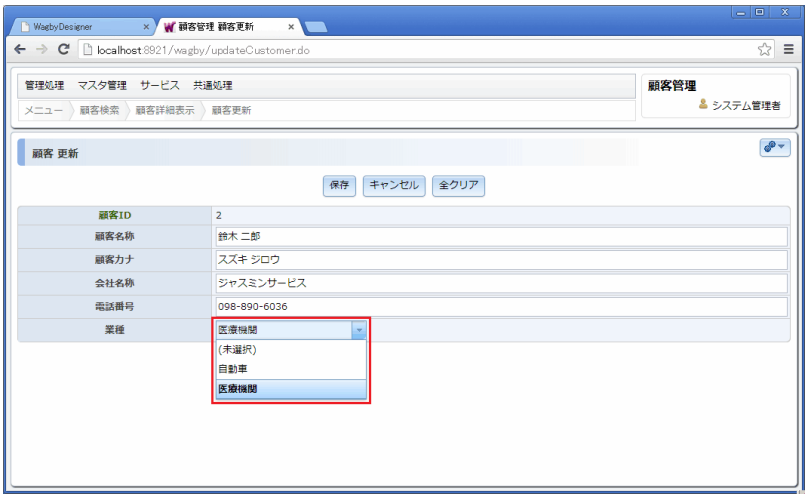


図 無効なデータは選択できない

Wagby ではこのように、運用しながら緩やかに無効なデータを（有効なデータに）置き換えることができます。

この動作はリストボックス・ラジオボタン・チェックボックスのいずれにも対応しています。「検索画面」方式では利用できません。

有効期限に日付範囲をする

有効期限には任意の文字列を指定できますが、日付を表す書式 (yyyy/MM/dd) の場合は特別の意味を持ちます。

有効期限の設定値	判定
2016/1/1-2016/12/31	2016/1/1 から 2016/12/31 まで有効。例：本日が 2016/7/1 の場合は有効。本日が 2017/1/1 の場合は無効となる。

-2016/12/31	2016/12/31 まで有効。例：本日が 2016/7/1 の場合は有効。本日が 2017/1/1 の場合は無効となる。
2016/1/1-	2016/1/1 から有効。例：本日が 2016/7/1 の場合は有効。本日が 2015/12/31 の場合は無効となる。
2016/12/31	2016/12/31 のみ有効。例：本日が 2016/12/31 の場合は有効。それ以外の日であれば無効となる。
"無効"という文字列	無効となる。
"有効"または"有効なデータ"という文字列	無効となる。日付以外であれば、文字列がセットされていた時点で無効と判定される。
(空文字)	有効となる。

無効な選択肢を示すマークを変更する

"Appendix > 選択肢の無効化マークを変更する"をお読みください。

無効な項目も検索対象とする

例

無効と設定したデータを検索の対象とする設定を用意しています。図では、業種モデルの "食品" データを無効としています。

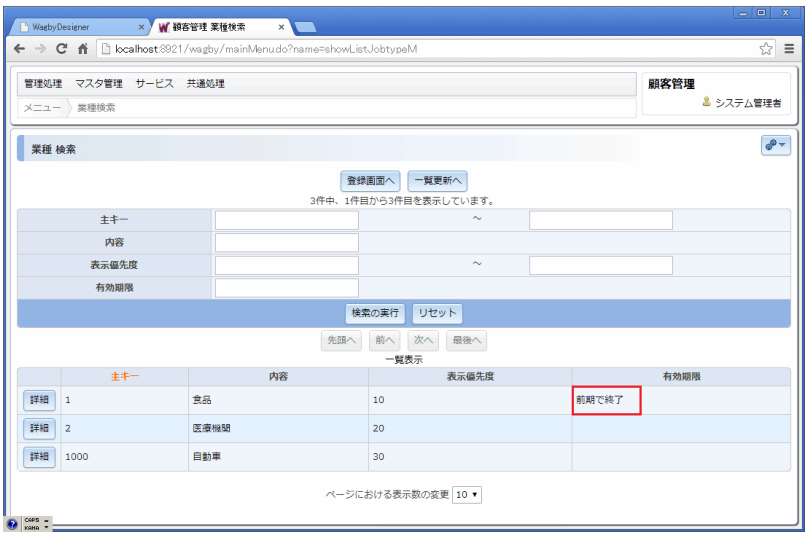


図 業種データ

この業種モデルを参照している顧客モデルの検索画面で、無効となったデータ "食品" で検索できるようになっています。

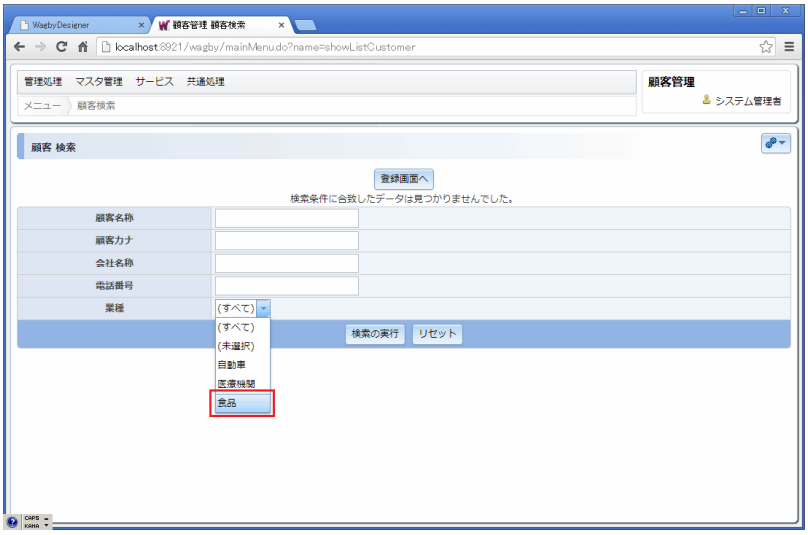


図 顧客検索画面では、無効データも検索可能となっている

新規登録・更新画面では、無効データは表示されません。

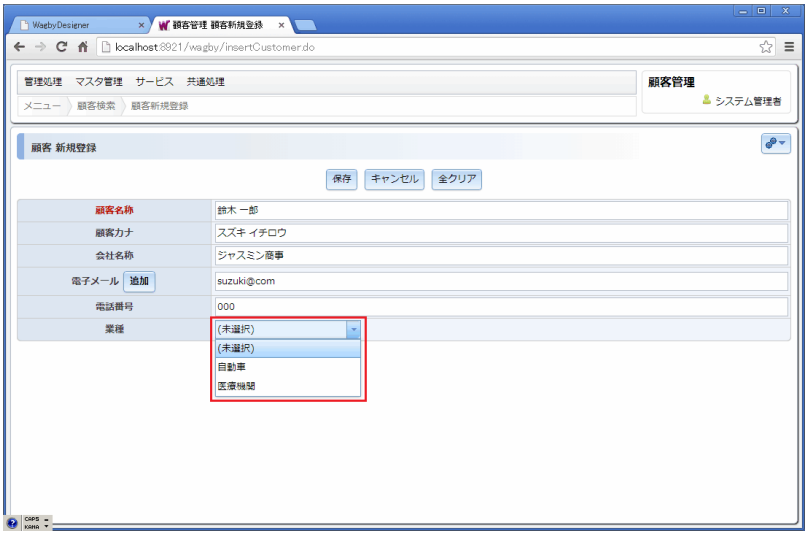


図 顧客新規登録画面では無効データは選択できない

定義方法

顧客モデルの業種項目についての詳細定義ダイアログを開きます。検索制御タブ内の「無効判定項目を無視する」を有効にします。

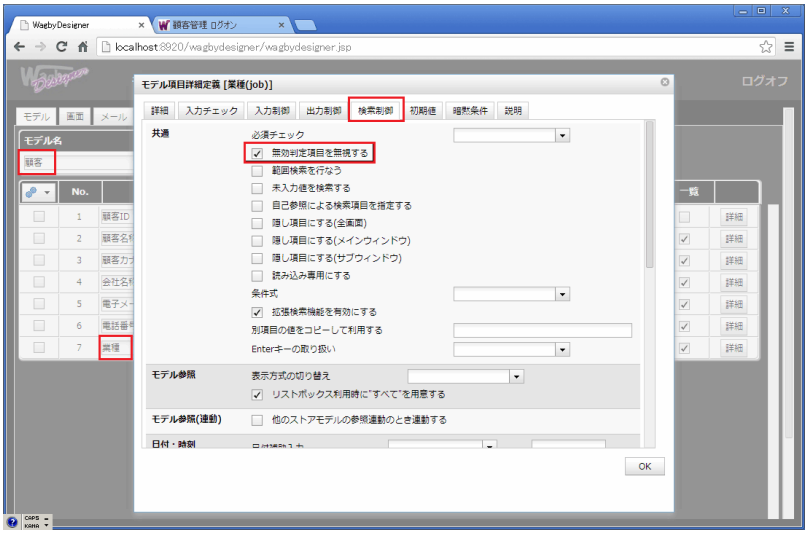


図 無効判定項目を無視する

参照先モデルのデータが削除されていた場合

参照先のデータが削除されていた場合、詳細表示画面には〝存在しないデータ〟と表示されます。

文言を変更する

この〝存在しないデータ〟という文言を変更することができます。

1. カスタマイズ用のリソースファイル myjfcapp_ja.properties.UTF8 ファイルを新規に作成します。このファイルを customize/resources フォルダに保存します。
2. 次のエントリを用意します。

```
__jfc_common.data.noexist=XXX
```

XXXには任意の代替文字を指定します。なお、空文字とすると空白で表示されます。

(39)

データベースとの関係

選択肢モデルを参照する側（参照元モデル。上記例では「顧客」モデル）の項目では、選択肢モデルのID値を保持します。内容部ではありません。

(40)

チェックボックスを用いた場合は複数の選択値(ID)を保持するため、専用のテーブルが作成されます。詳細...

リストボックス

例と定義方法

例

予め商品モデルを図のように用意します。

---

(39) myjfcapp_ja.properties.UTF8 によるリソースの置換についての詳細は"メッセージの変更・国際化 > リソースファイルの編集 > 差分ファイル作成方式"をお読みください。

(40) データベース上のテーブルを確認すると、ID値が格納されていることがわかります。

図 事前に用意した商品モデル

この商品モデルの値をリストボックスで選択する画面例を図に示します。 参照先モデル「商品」の名前をリストボックスから選択することができます。

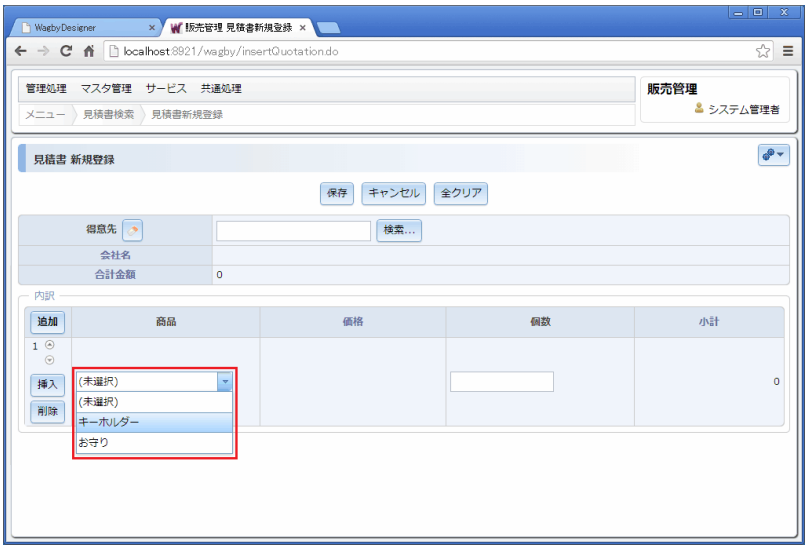


図 リストボックスの例

定義方法

図の見積書モデルにおいて、商品項目の型を"他モデルの項目参照"とします。

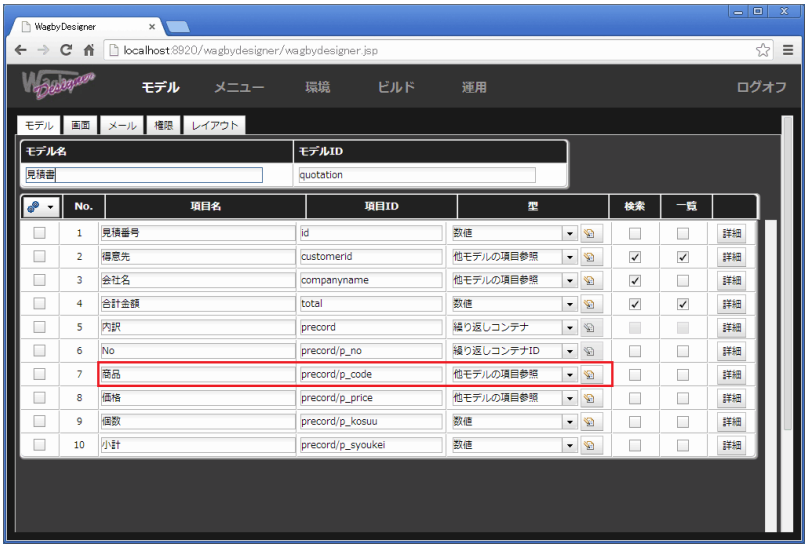


図 見積モデルの定義

型の詳細設定で、参照先モデルを「商品」とし、リストボックスの選択肢となる参照先モデルの項目を「商品名」とします。 入力タイプに「リストボックス」を選びます。

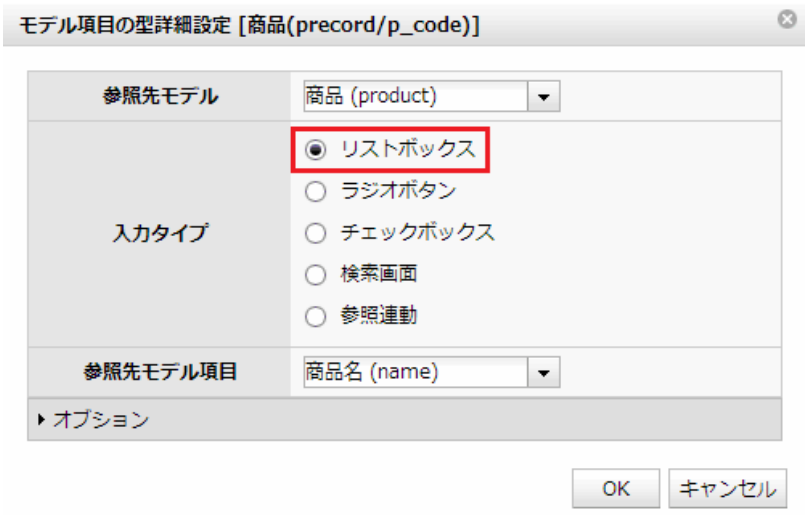


図 リストボックスを利用する設定

参照先モデルのデータが削除されていた場合

参照先のデータが削除されていた場合、詳細表示画面には〝存在しないデータ〟と表示されます。

文言を変更する

この〝存在しないデータ〟という文言を変更することができます。

データベースとの関係

参照元モデル（上記例では「見積書」モデル）の参照キー項目では、参照先モデルのID値を保持します。内容部ではありません。[41]

[41] データベース上のテーブルを確認すると、ID値が格納されていることがわかります。

未選択の扱い

「(未選択)」はデータベース内では NULL で保存されます。

複合キーのモデルを参照する

主キーが複合キー形式となっている「商品」モデルを用意します。

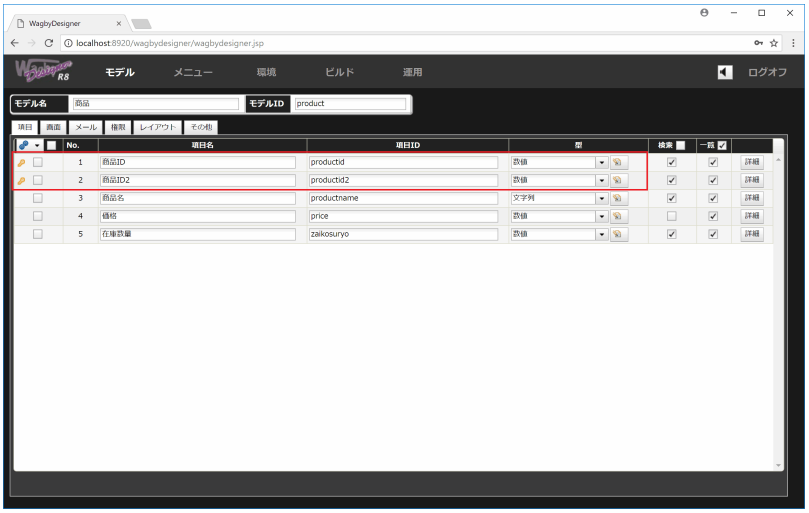


図 商品モデル（主キーが複合キー形式）

「売上伝票」モデルの「明細」部（繰り返しコンテナ）で、商品モデルを参照します。

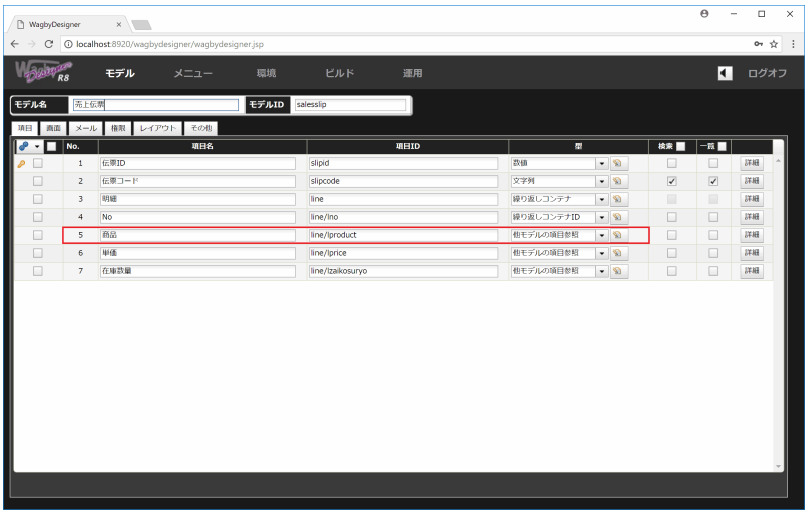


図 売上伝票モデル

参照先が複合キーであっても、設定方法に違いはありません。

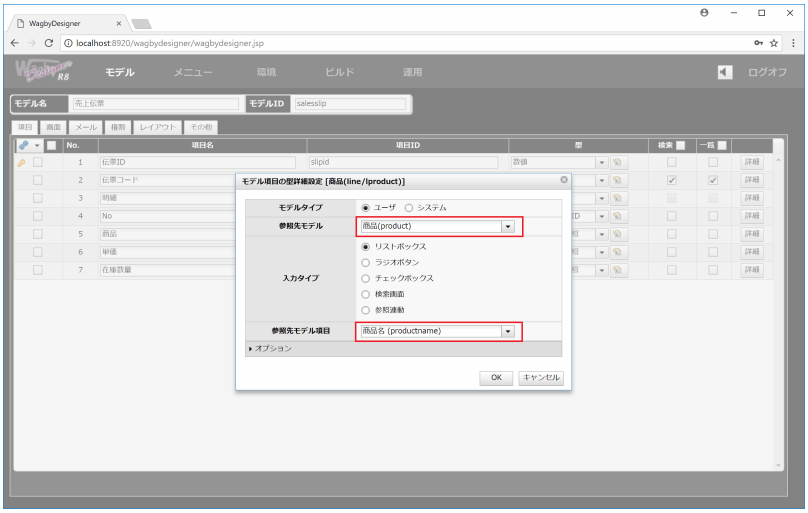


図 売上伝票モデルで、商品モデルを参照する

なお、Wagbyの設計情報の「商品」項目は、物理テーブル定義では二つの列が用意されています。[詳細...]

関連するページ

- 計算式 > モデル参照 > リストボックス項目の扱い
- 入力制御 > 入力フィールド長 > リストボックスの場合

初期値

例

リストボックスの初期値を指定することができます。見積書モデルの新規登録画面を開いたとき、リストボックスに「キーホルダー」が選択された状態にする例を説明します。

対象データの主キー（ここでは商品ID）の値を調べます。

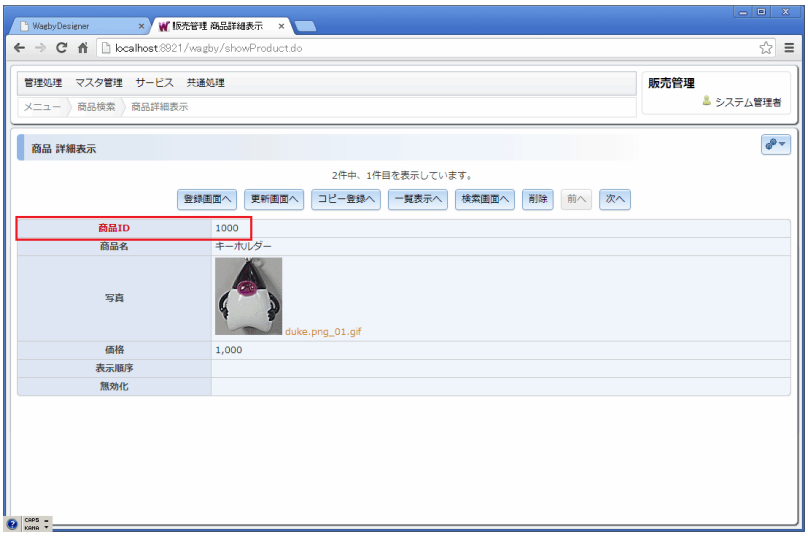


図 初期値として用いるデータの主キーを調べる

見積書モデルの新規登録画面を開くと、図のデータが初期値として選択されていることがわかります。

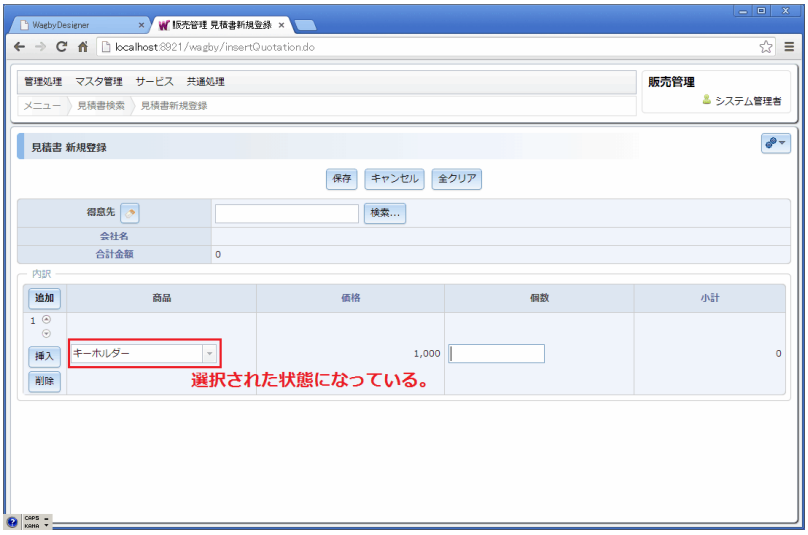


図 初期値を設定したときの画面例

定義方法

対象データの主キーを指定します。

図 新規登録画面の初期値の設定

主キーの値はシステム運用中に変更されることはありません。

選択肢の表示順を変更する

選択肢モデルにおける、選択肢の表示順変更と同じルールを、任意のモデルに設定することができます。

例

ここでは商品モデルに「表示順序」という項目を追加しました。

図 表示順序項目を追加した

並びを「降順」とした場合、表示順序の数値が大きい順に選択肢が並びます。

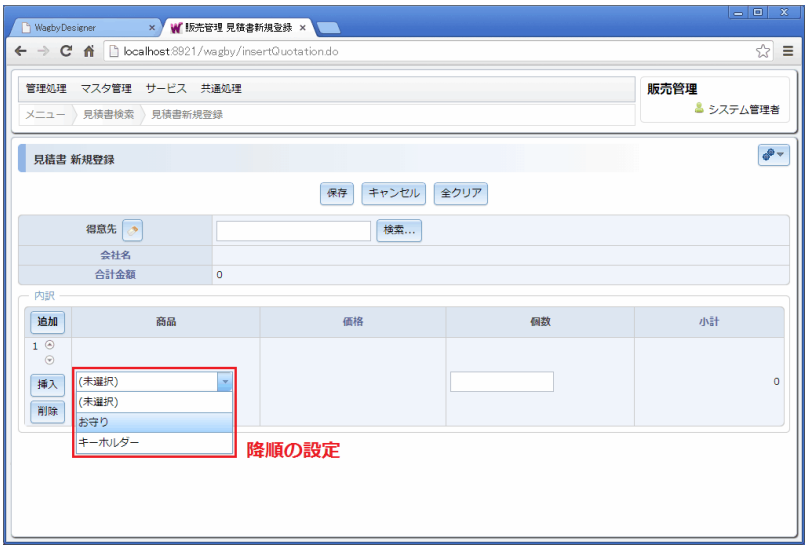


図 選択肢の並びが変わった

定義方法

「表示順序」項目を定義します。型を整数型とします。

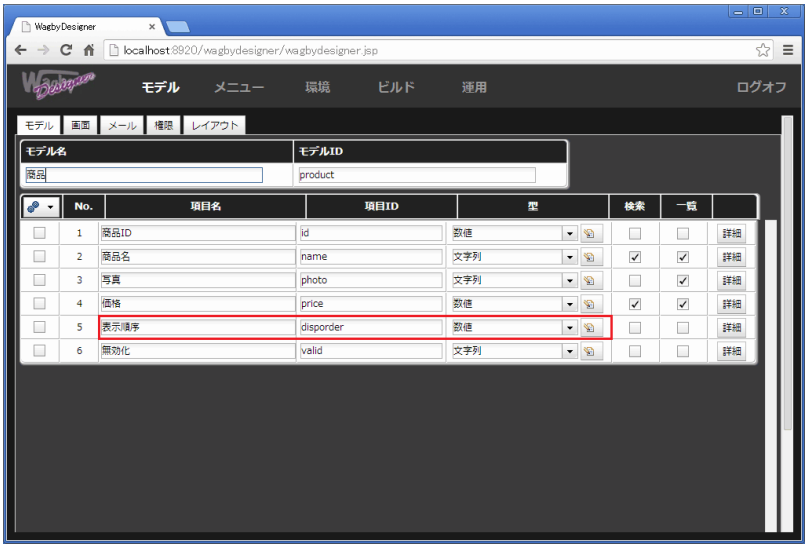


図 表示順序項目の定義

見積書モデルでは、オプション設定で「ソートルール」を表示優先度とし、表示優先度項目に図の「表示順序」を指定します。（ここでは、型が整数型の項目を任意に指定できます。）

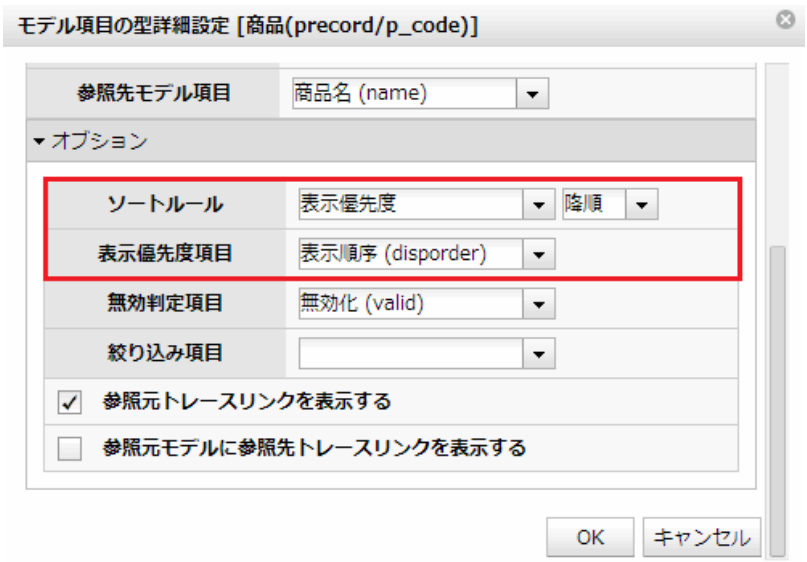


図 表示優先度項目を指定する

表示順序の値が未設定（空白）の場合は 0 と解釈されます。

ソートルールの詳細

表示優先度

「表示優先度項目」に指定した項目を指定します。型は整数型となっている必要があります。表示順の標準は「昇順（数字の小さい順）」ですが、これを「降順（数字の大きい順）」ルールに変えることもできます。 （昇順は ″1,2,3,...″ であり、降順は ″10,9,8,...″ という意味です。）

ID

参照先モデルの主キー値によってソートされます。「表示優先度項目」は用いられません。

内容

参照先モデルの内容によってソートされます。文字列型の場合は辞書順になります。「表示優先度項目」は用いられません。

参照先モデルの固定ソートキー

参照先モデルの一覧表示設定で指定したソートキーの並び順になります。「表示優先度項目」は用いられません。

選択肢を無効にする

選択肢モデルにおける、選択肢の無効化と同じルールを、任意のモデルに設定することができます。

参照先のデータを削除すると、データ間の不整合が生じます。これを避けるため、データを（削除するのではなく残しつつ）無効化することができます。無効化によってシステム上は不可視になりますが、データベースには残っています。無効化を解除して、復活させることもできます。

例

商品モデルに「無効化」という項目を用意しました。ここに任意の文字列が設定されていると、本データは無効となります。 図において、1件のデータに〝生産終了〟という文字列をセットしています。

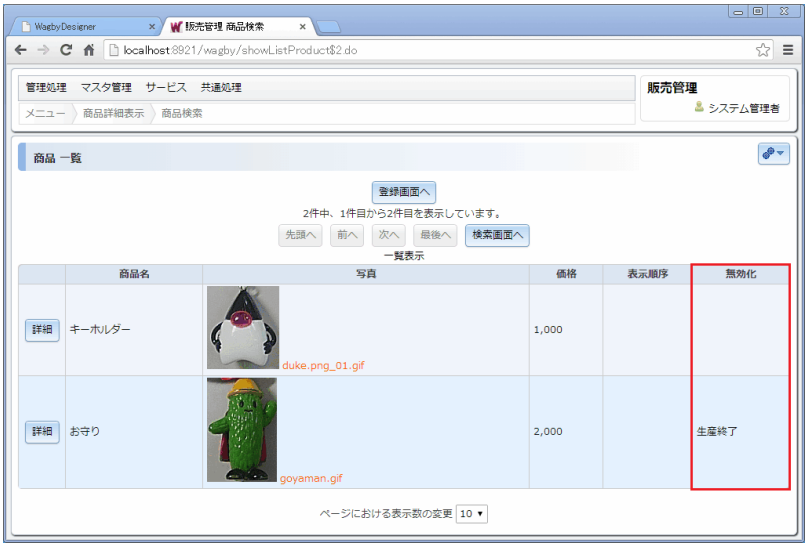


図 1件のデータを無効化する

見積モデルの選択肢には、無効化されたデータは表示されません。

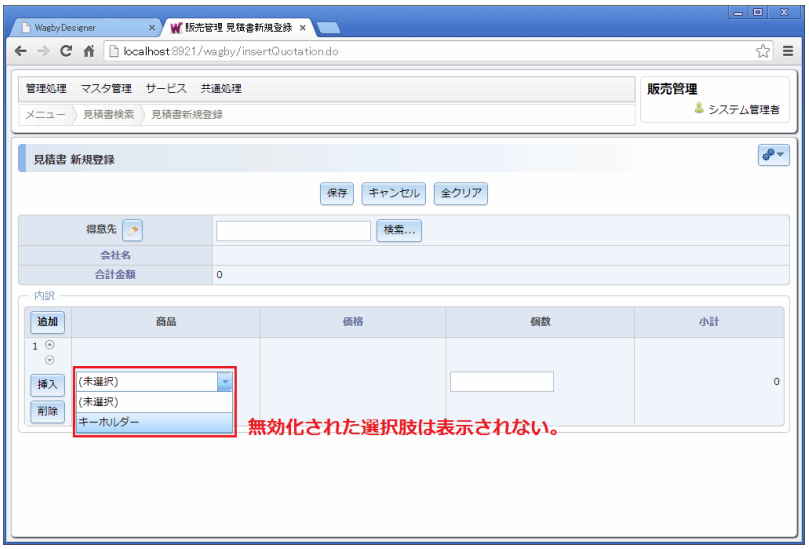


図 無効化されたデータは表示されない

無効化されたデータの扱いの詳細は〝選択肢モデルにおける、選択肢の無効化〟をお読み下さい。

定義方法

「無効化」項目を定義します。項目名、項目IDは任意です。型を文字列型とします。

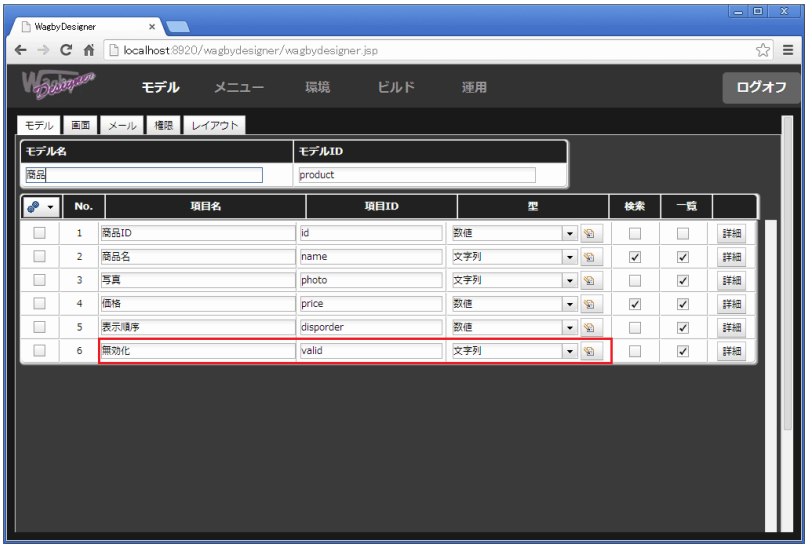


図 無効化項目の定義

見積書モデルでは、オプション設定で「無効判定項目」に、図で用意した項目を指定します。型は文字列型です。

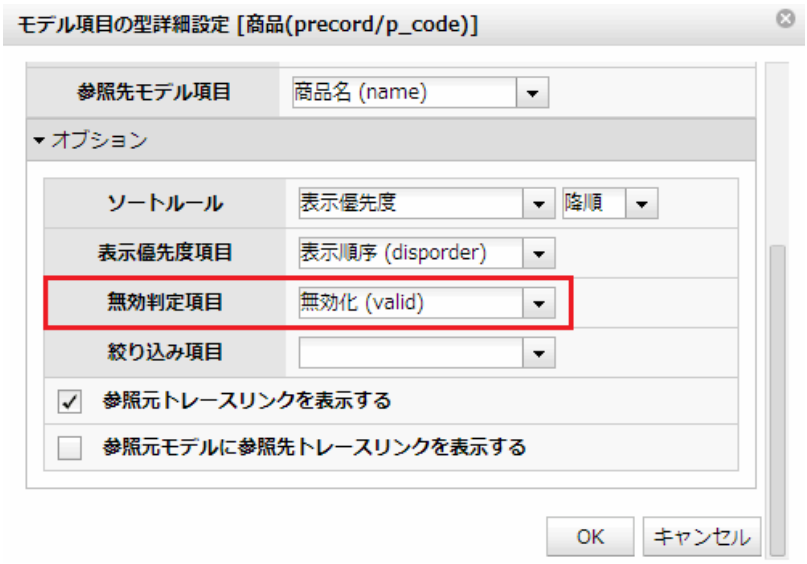


図 無効化項目を指定する

無効を意味するマークを変更する

無効を意味するマーク「(-)」を変更することができます。

無効な項目も検索対象とする

"選択肢モデルにおける、無効な項目も検索対象とする"と同じルールを、任意のモデルに設定することができます。

未選択項目を用意する

例

　リストボックスでは標準で ″何も選択していない″ ことを意味する選択肢「(未選択)」が用意されます。これは常に先頭に配置されます。（この順序を変えることはできません。）

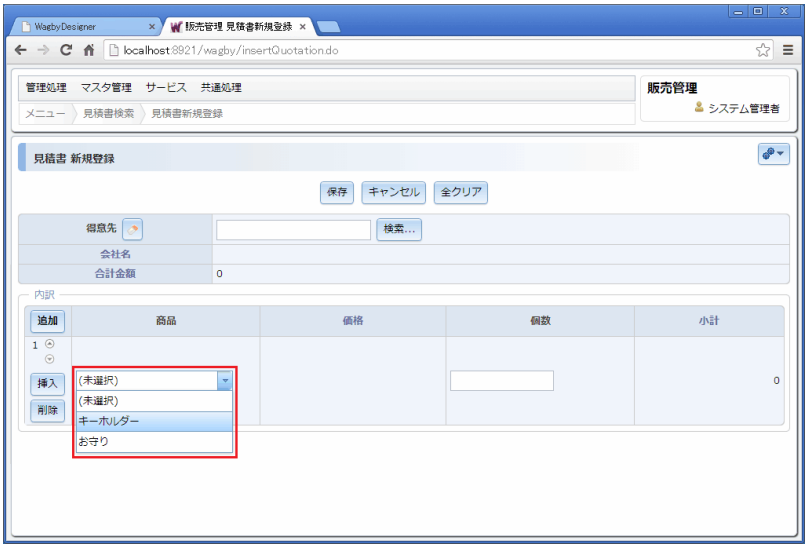


図 未選択項目が用意されている

検索画面への対応

　この項目を検索条件として用いたとき、自動的に未選択項目も用意されます。「(未選択)」となっているデータを検索することができます。

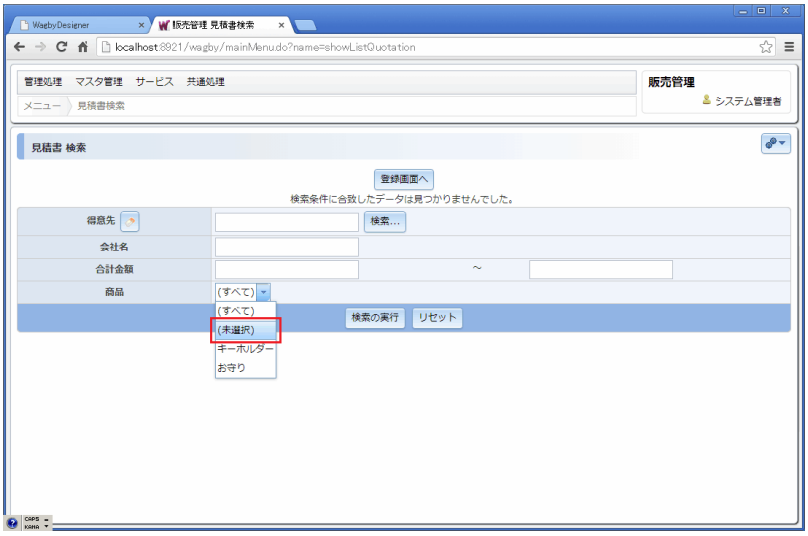


図 検索条件にも「(未選択)」が用意される

ラジオボタンまたはチェックボックスを使った場合は、何も選択しないという状態があるため、本機能は不要です。

定義方法

　参照元モデル（この例では見積書モデル）のモデル項目詳細定義にて入力制御タブを開きます。「モデル参照>未選択を用意する」を有効にします。本設定は標準で有効となっているため、不要な場合にこれを解除してください。

図 未選択項目を用意する設定

未選択項目を用意しない場合、図のようになります。この場合、選択しないという状態を用意させない、という運用になります。

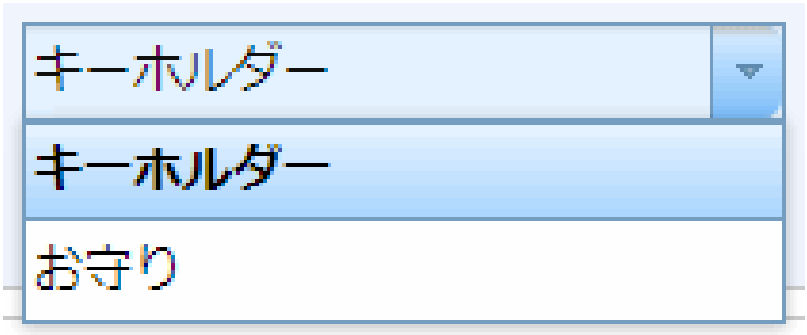


図 未選択項目を用意しないリストボックス

式からこの項目を参照する場合

この項目を計算式で用いる場合、新規登録画面を開いたときの初期値は null となります。

未選択項目を作成しない場合の初期値は、選択肢の先頭の値を（#id や #content を指定して）取得できます。

選択肢の絞り込み指定を行った場合、未選択項目を作成しない場合でも初期値は null となります。絞り込み操作を行うことで、はじめて選択値がセットされるためです。

上のいずれかのルールによって初期値が null となった場合、この項目に紐づく参照連動項目も解決されません。（連動元項目が null であるため。）

全件を検索する

例

リストボックスを検索条件に指定した場合、「すべて」という選択肢を指定することで全データを対象に検索を行うことができます。

（これは検索時のみ標準で用意されます。データの新規登録・更新時には表示されません。）

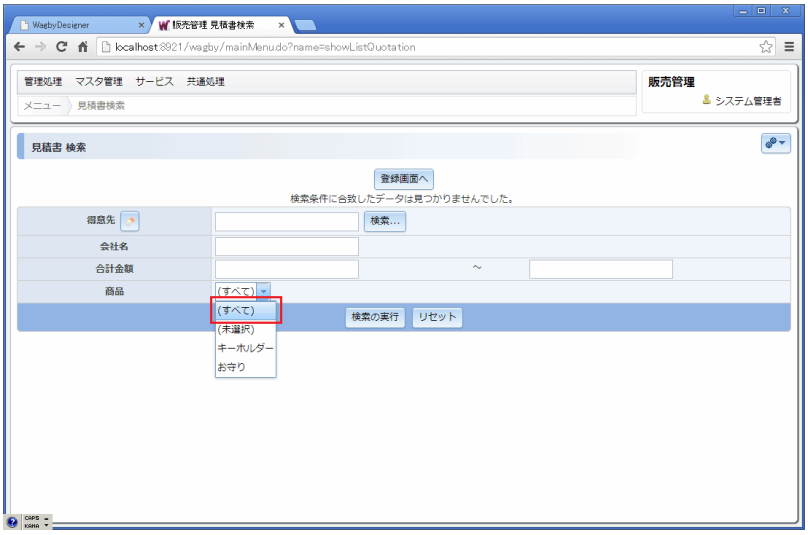


図 検索条件に選択肢「すべて」が用意される

定義方法

モデル項目詳細定義の検索制御タブを開きます。モデル参照欄に用意された「リストボックス利用時に"すべて"を用意する」を有効にします。本設定は標準で有効となっているため、不要な場合にこれを解除してください。

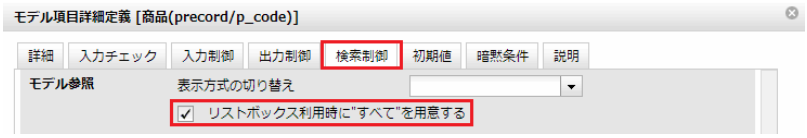


図 リストボックス利用時に「すべて」を用意する設定

検索画面での切り替え

例

登録・更新画面と検索画面の入力タイプを変更することができます。ここでは検索画面にラジオボタンを用いた例を示します。

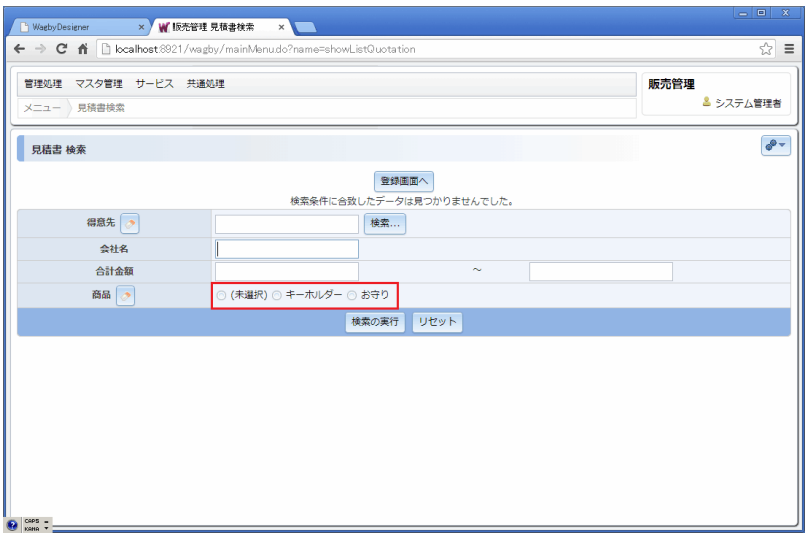


図 検索画面でラジオボタンを用いる

定義方法

モデル項目詳細定義から、検索制御タブを開きます。モデル参照内の「表示方式の切り替え」を指定します。（標準では空白となっています。）

図 表示方式の切り替え設定

複数の項目を連結して表示する

例

得意先をリストボックスで選択する例を図に示します。選択肢は〝氏名（氏名かな)〟となっています。

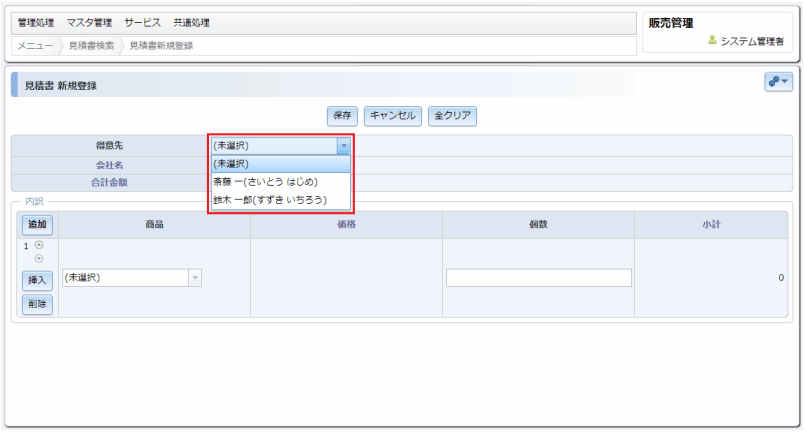


図 選択肢が「氏名(氏名かな)」となっている。

定義方法

顧客モデル側に「〝氏名（氏名かな)〟」を格納する項目を追加します。

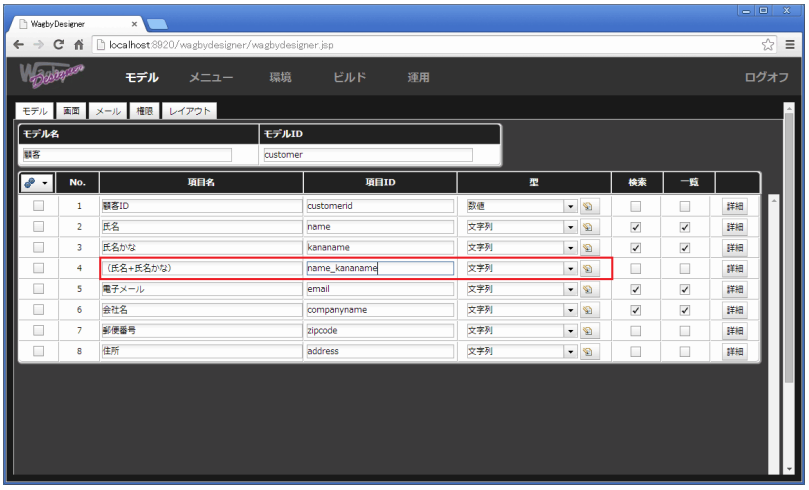


図 選択肢用の項目を追加定義する

この項目は自動計算によって求めます。データベースには保存しません。 この例では、式を次のようにしています。

```
${name}+"("+${kananame}+")"
```

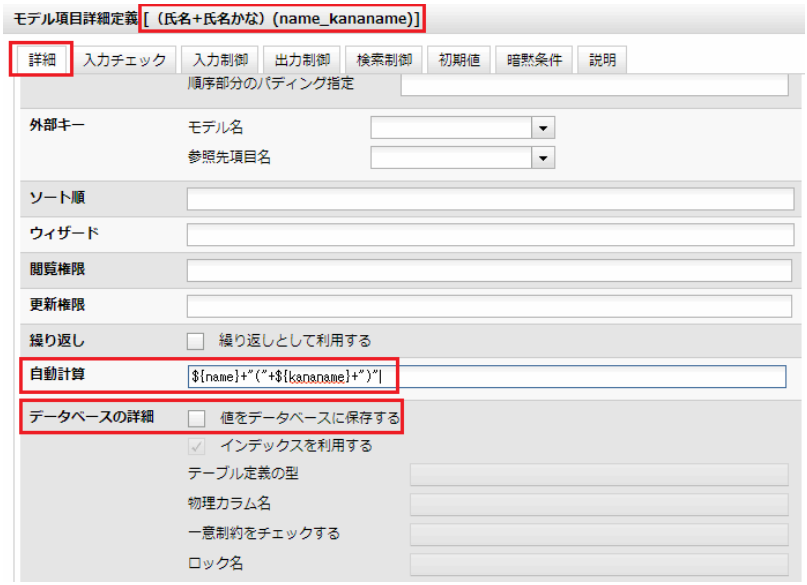


図 式の設定

この項目は入力時を隠し項目とします。 また出力時も隠し項目とします。

図 入力時の隠し項目設定

図 出力時の隠し項目設定

参照元モデル（このモデルを参照する側）では、参照先モデル項目として上記項目を指定します。

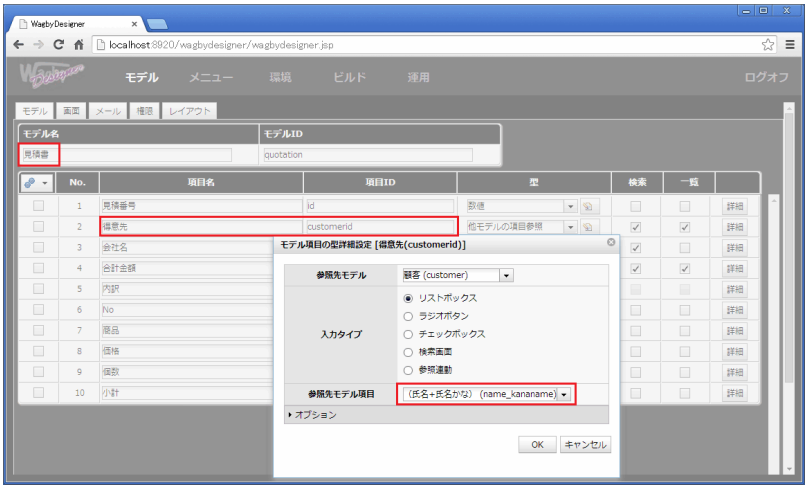


図 参照先モデル項目の設定

必須を指定する

リストボックスの項目に必須指定を行うと、リアルタイムで入力チェックが行われます。

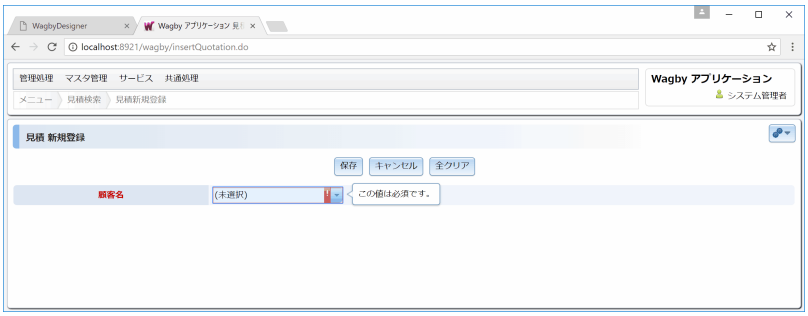


図 必須チェック

ラジオボタン

例と定義方法

例

予め地区モデルを図のように用意します。

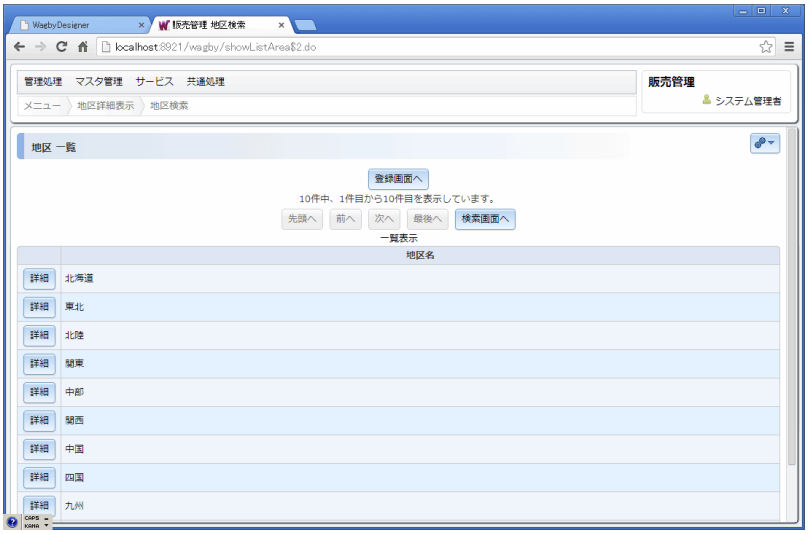


図 事前に用意した地区モデル

この地区モデルの値をラジオボタンで選択する画面例を図に示します。 参照先モデル「地区」の名前をラジオボタンから選択することができます。

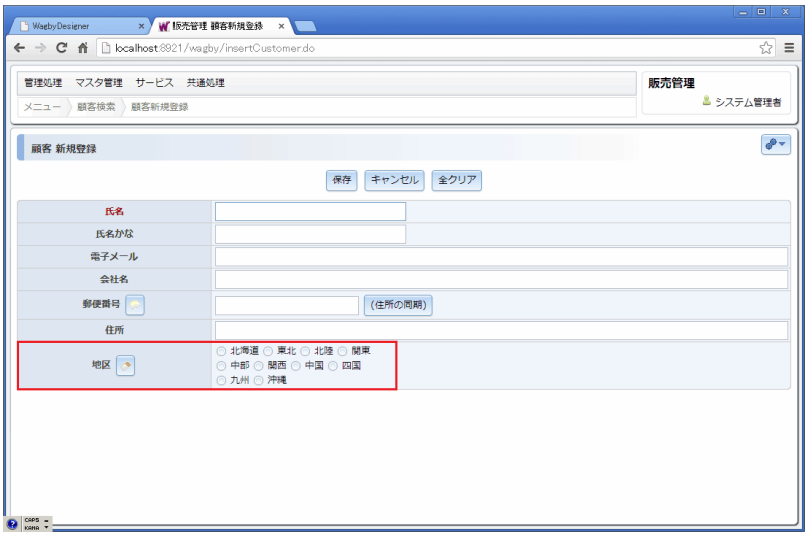


図 ラジオボタンの例

定義方法

図の顧客モデルにおいて、地区項目の型を"他モデルの項目参照"とします。

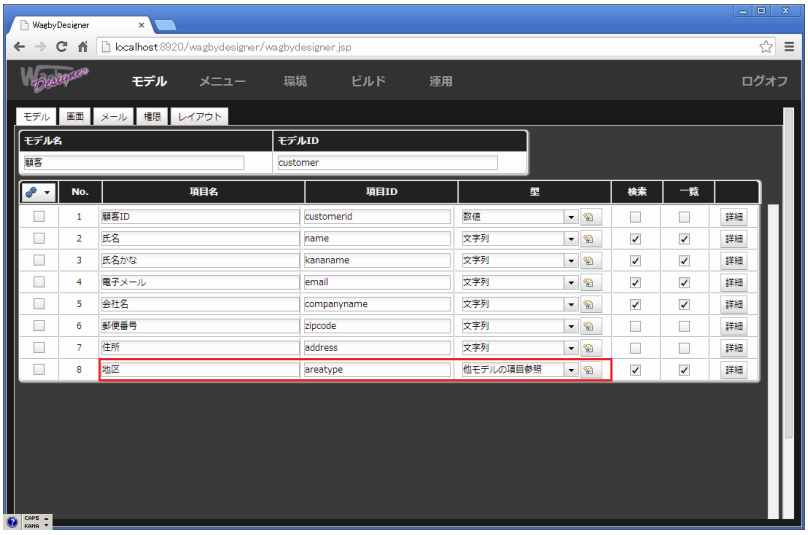


図 顧客モデルの定義

型の詳細設定で、参照先モデルを「地区」とし、ラジオボタンの選択肢となる参照先モデルの項目を「地区名」とします。 入力タイプに「ラジオボタン」を選びます。

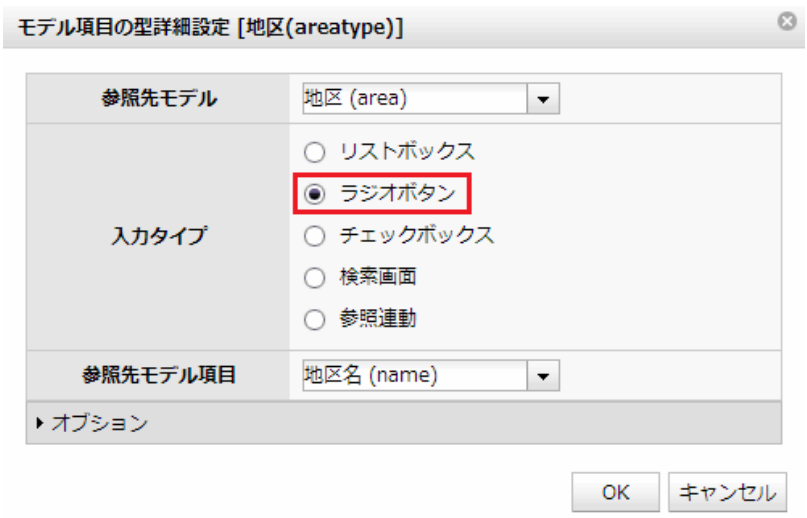


図 ラジオボタンを利用する設定

データベースとの関係

参照元モデル（上記例では「見積書」モデル）の参照キー項目では、参照先モデルのID値を保持します。内容部ではありません。一つも選択していない状態は NULL となります。[42]

複合キーのモデルを参照する

「リストボックス ＞ 複合キーのモデルを参照する」をお読みください。

初期値

[42] データベース上のテーブルを確認すると、ID値が格納されていることがわかります。

例

ラジオボタンの初期値を指定することができます。顧客モデルの新規登録画面を開いたとき、ラジオボタンに「北海道」が選択された状態にする例を説明します。

対象データの主キー（ここでは地区ID）の値を調べます。

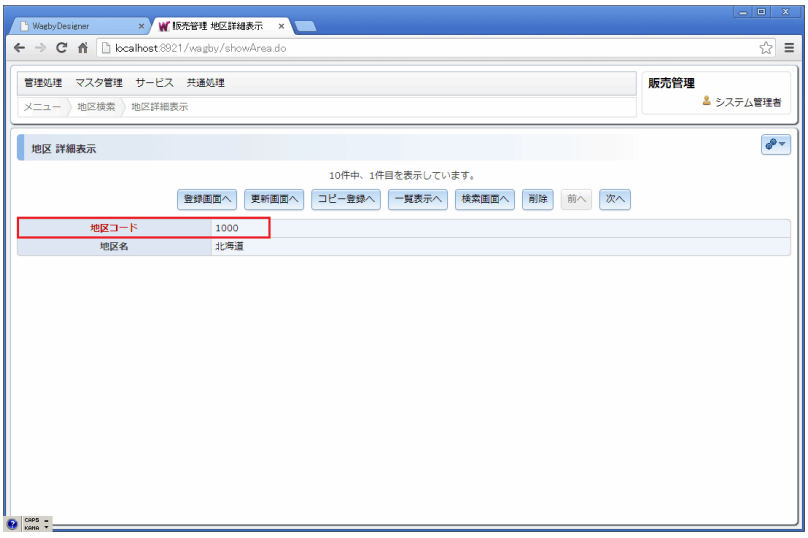


図 初期値として用いるデータの主キーを調べる

顧客モデルの新規登録画面を開くと、図のデータが初期値として選択されていることがわかります。

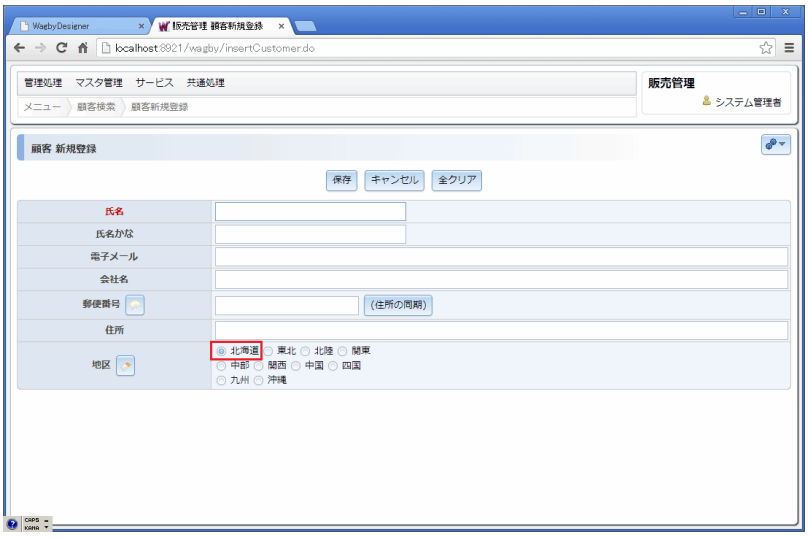


図 初期値を設定したときの画面例

定義方法

対象データの主キーを指定します。

図 新規登録画面の初期値の設定

主キーの値はシステム運用中に変更されることはありません。

選択肢の表示順を変更する

選択肢モデルにおける、選択肢の表示順変更と同じルールを、任意のモデルに設定することができます。
詳細は″リストボックスの説明″をお読みください。

選択肢を無効にする

選択肢モデルにおける、選択肢の無効化と同じルールを、任意のモデルに設定することができます。
詳細は″リストボックスの説明″をお読みください。

無効な項目も検索対象とする

″選択肢モデルにおける、無効な項目も検索対象とする″と同じルールを、任意のモデルに設定することができます。

折り返し数の設定

例

ラジオボタンは標準で1行あたり4つの選択肢が表示されるようになっています。

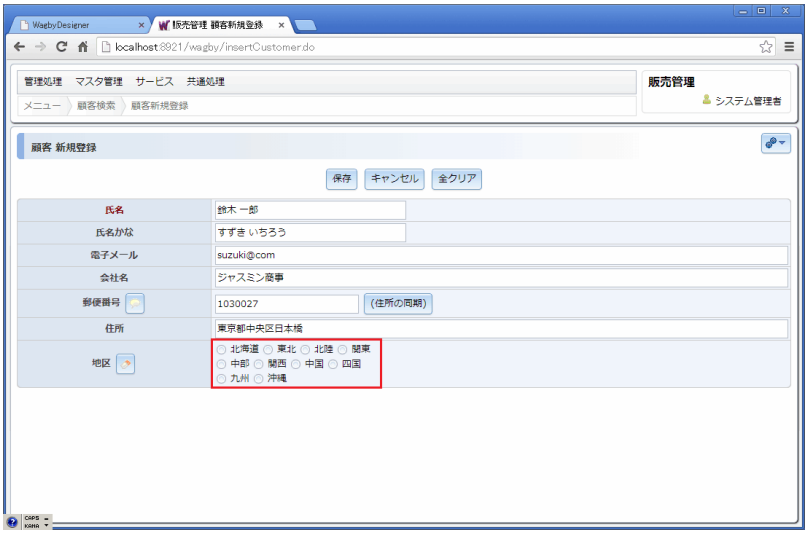


図 4つ表示で折り返し

これを8つ毎に折り返しと指定すると図のように表示されます。

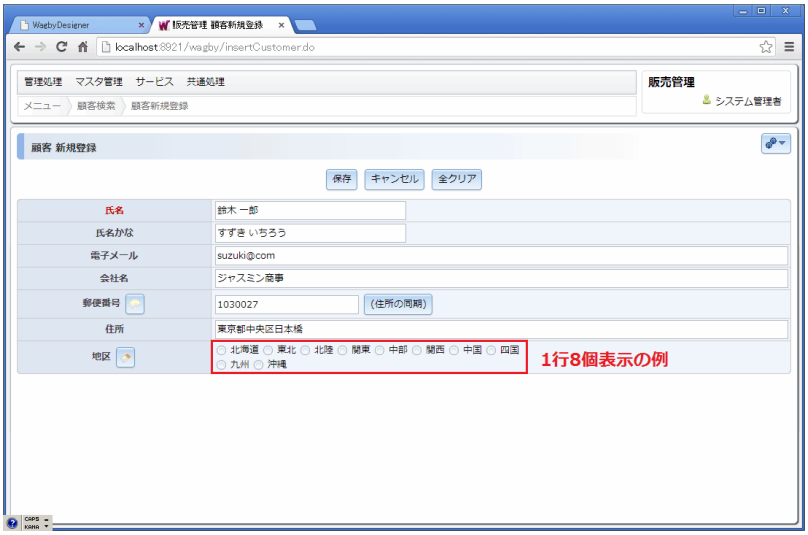


図 8つ表示で折り返し

定義方法

モデル項目詳細定義ダイアログの入力制御タブから、モデル参照内の「チェックボックス・ラジオボタンの折り返し数」を指定します。空白時は〝4〟と解釈されます。

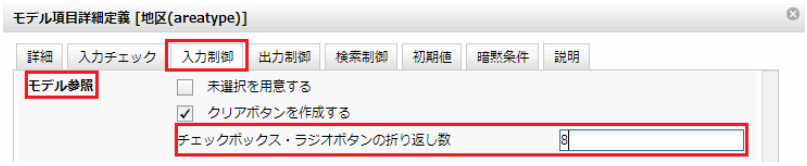


図 折り返し数の設定

一覧画面の設定

〝出力制御 ＞ 一覧画面 ＞ モデル参照 ＞ 1行に表示するチェックボックス・ラジオボタンの表示数〟 を設定してください。

選択肢のクリアボタン

例

ラジオボタン利用時は、現在の選択を解除するためのクリアボタン（消しゴムのアイコン）が用意されます。

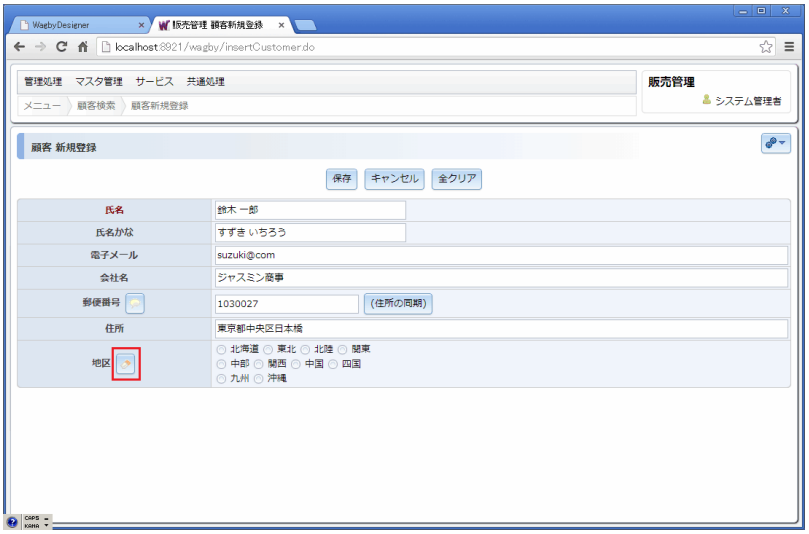


図 クリアボタン

クリアボタンを非表示とすることもできます。この場合、一度何かを選択すると、必ず一つは選択されるという状態になります。（全解除することはできません。）

図 クリアボタンを用意しない場合

定義方法

モデル項目詳細定義ダイアログの入力制御タブから、モデル参照内の「クリアボタンを作成する」を有効にします。（標準では有効になっています。）

本設定は登録・更新・検索画面に反映されます。

図 クリアボタンを作成する

本設定が無効となる場合

ラジオボタンを必須項目とすると、本クリアボタンは自動的に非表示となります。

検索画面での切り替え

登録・更新画面と検索画面の入力タイプを変更することができます。 標準では、ラジオボタンの項目は検索画面ではリストボックスとして扱われます。

切り替え方法の詳細は"リストボックスの説明"をお読みください。

選択されなかった項目名を表示する

例

詳細画面では、選択した項目のみが表示されます。これが標準の動作になります。

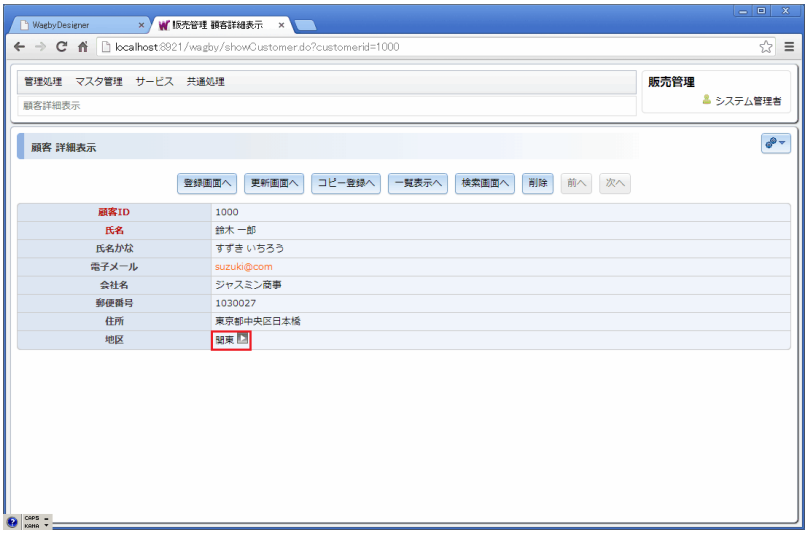


図 詳細画面での表示

選択されなかった項目名を表示すると設定した場合は次のようになります。

図 選択されなかった項目名を表示する設定を有効にした場合

定義方法

モデル項目の型詳細設定ダイアログのオプションに用意されている「選択されなかった項目名を表示する」設定を有効にします。

図 モデル項目の型詳細設定ダイアログ

この設定は一覧画面にも反映されます。

参照先モデルのデータが削除されていた場合

参照先のデータが削除されていた場合、詳細表示画面には〝存在しないデータ〟と表示されます。

文言を変更する

この〝存在しないデータ〟という文言を変更することができます。

チェックボックス

例と定義方法

例

予め地区モデルを図のように用意します。

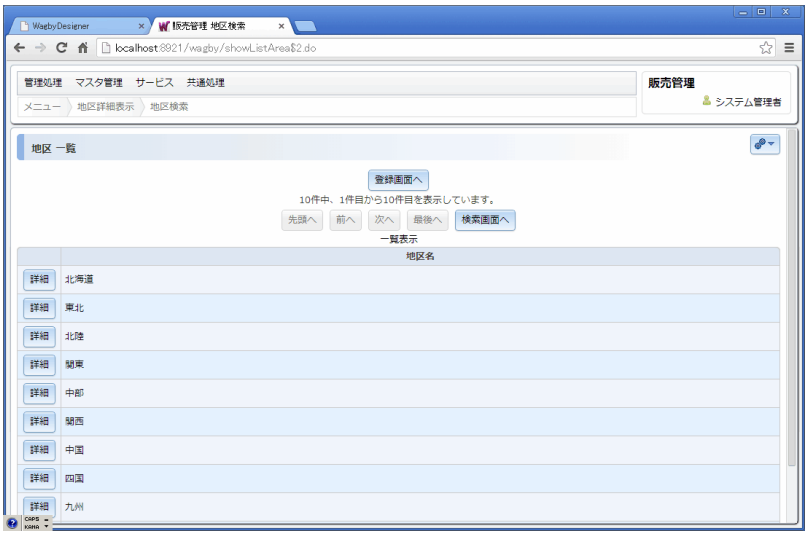


図 事前に用意した地区モデル

この地区モデルの値をチェックボックスで選択する画面例を図に示します。 参照先モデル「地区」の名前をチエックボックスから選択することができます。

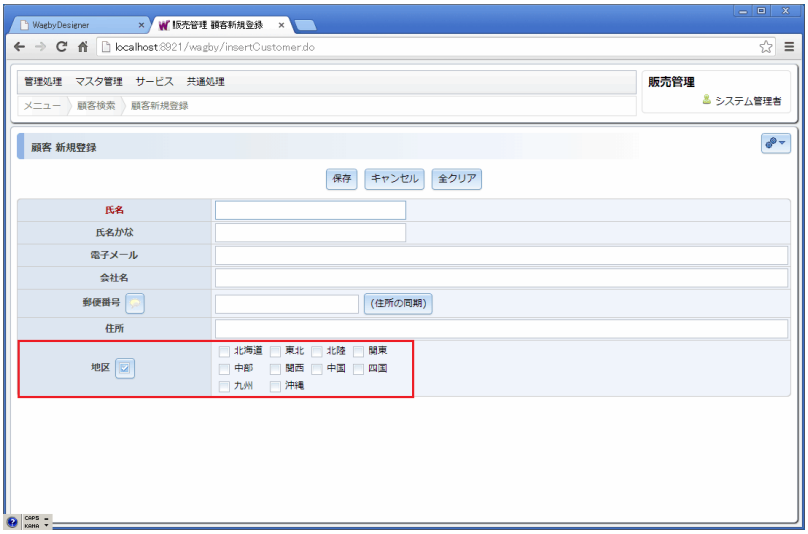


図 チェックボックスの例

定義方法

図の顧客モデルにおいて、地区項目の型を"他モデルの項目参照"とします。

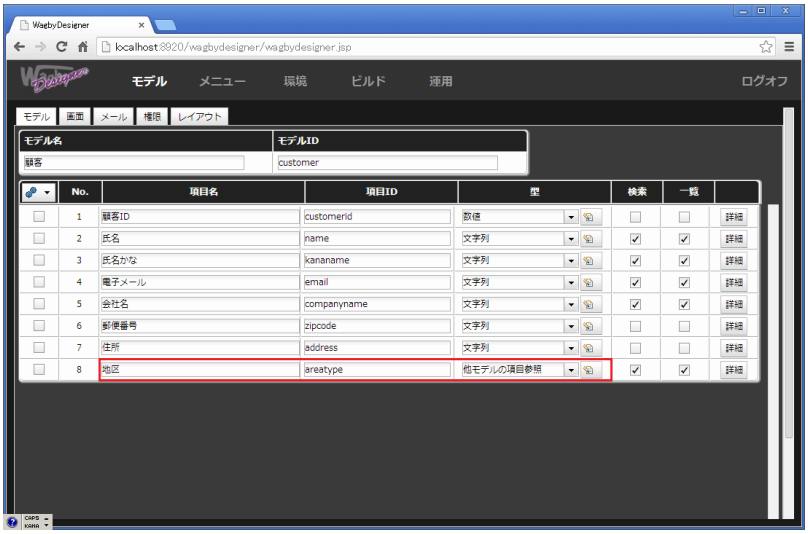


図 顧客モデルの定義

型の詳細設定で、参照先モデルを「地区」とし、チェックボックスの選択肢となる参照先モデルの項目を「地区名」とします。 入力タイプに「チェックボックス」を選びます。

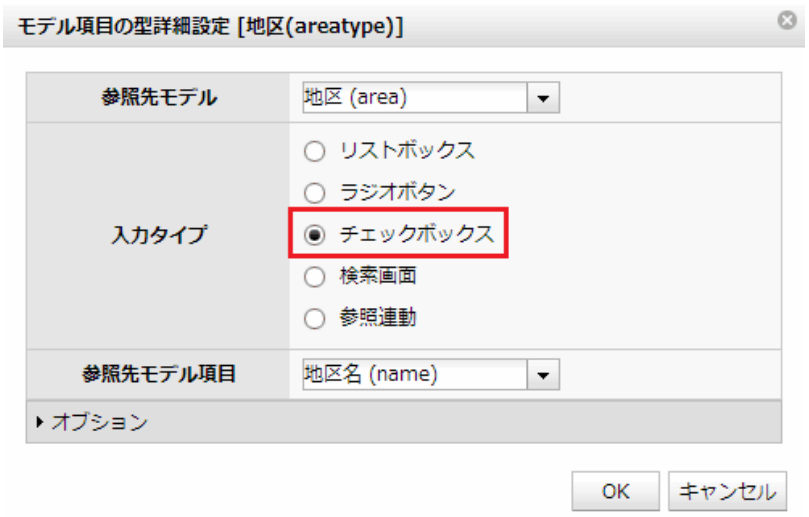


図 チェックボックスを利用する設定

最大N個までという入力ルール

例

　複数の選択肢の中から、最大で 3 つまで選択できる（4つ以上選択すると入力エラーにする）例を紹介します。

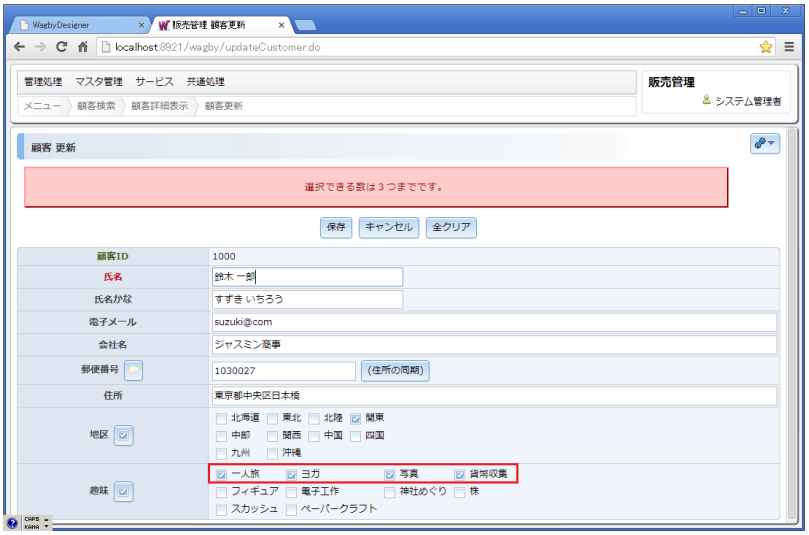


図 4つ選択するとエラーとする

定義方法

　モデル項目詳細定義ダイアログの入力チェックタブ内にあるユーザ定義チェック欄に、許容できる（エラーにならない）条件を記述します。エラー時のメッセージも設定できます。

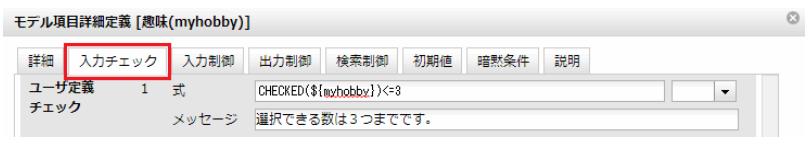


図 ユーザ定義チェックの設定

```
CHECKED(${myhobby})<=3
```

詳細は"ユーザ定義チェック"をお読みください。

検索画面での切り替え

登録・更新画面と検索画面の入力タイプを変更することができます。

切り替え方法の詳細は"リストボックスの説明"をお読みください。

参照先モデルのデータが削除されていた場合

参照先のデータが削除されていた場合、詳細表示画面には ″存在しないデータ″ と表示されます。

文言を変更する

この ″存在しないデータ″ という文言を変更することができます。

データベースとの関係

チェックボックスで選択された値を管理するため、別のテーブルが用意されます。詳細...

複合キーのモデルを参照する

「リストボックス 〉 複合キーのモデルを参照する」をお読みください。

仕様・制約

参照先のモデルの主キーが複合キーの場合、本機能（チェックボックスとして参照する）は利用できません。

選択肢の全チェックと全クリア

すべての選択肢を一括して選択状態にすることができます。図に示す全チェックアイコンを押下します。

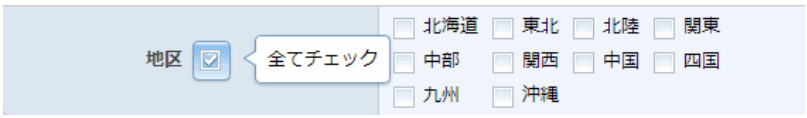


図 全チェックアイコン

チェック後は、アイコンの形が消しゴムに変わります。

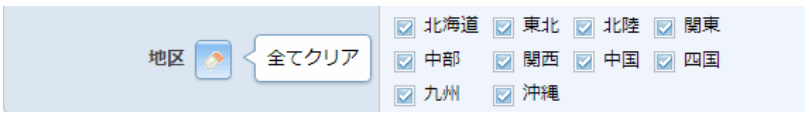


図 アイコンの形が消しゴムに変わった

消しゴム機能を使うことで、すべての選択を一度に解除することができます。アイコンが全チェックに戻ります。

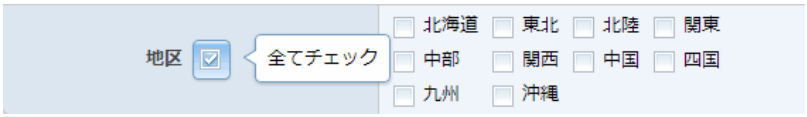


図 消しゴムアイコンを押下するとアイコンの形が全クリアに変わる

チェックボックスの全選択・クリアボタンの状態切替はチェックボックス押下時にも行なわれます。チェックボックスを全てチェックすると、自動的にクリアボタンに変わります。全てチェックされた状態から一つでもチェックを外すと全選択ボタンへ変わります。

定義方法

モデル項目詳細定義ダイアログの入力制御タブから、モデル参照内の「クリアボタンを作成する」を有効にします。（標準では有効になっています。）

本設定は登録・更新・検索画面に反映されます。

図 クリアボタンを作成する

初期値

例

チェックボックスの初期値を指定することができます。顧客モデルの新規登録画面を開いたとき、「北海道」「東北」「北陸」「関東」の4つを選択状態にする例を説明します。

対象データの主キー（ここでは地区ID）の値を調べます。

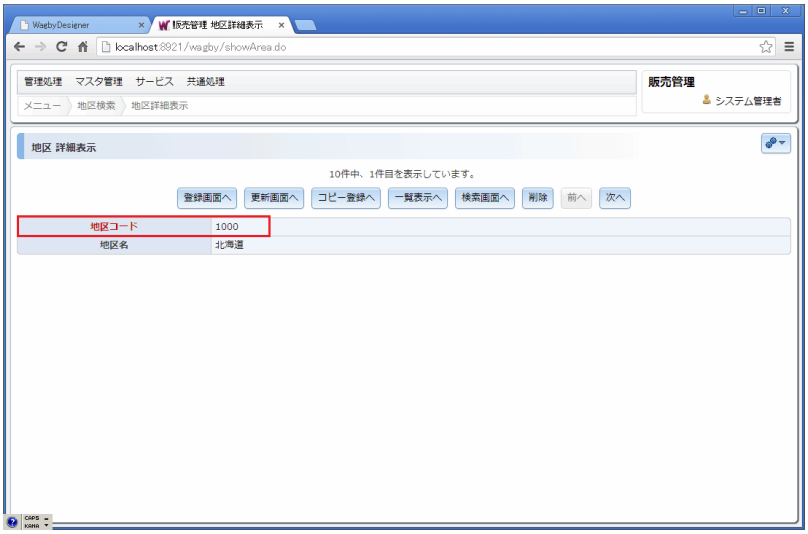


図 初期値として用いるデータの主キーを調べる

顧客モデルの新規登録画面を開くと、図のデータが初期値として選択されていることがわかります。

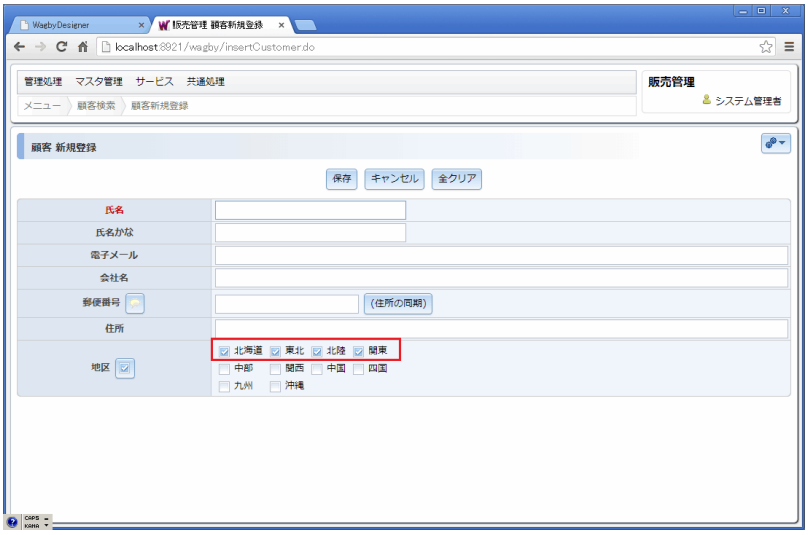


図 初期値を設定したときの画面例

定義方法

対象データの主キーを指定します。対象がチェックボックスの場合、コンマ区切りで複数の値を指定することができます。

図 新規登録画面の初期値の設定

主キーの値はシステム運用中に変更されることはありません。

参照先のモデルが単一主キーでかつ整数型の場合のみ、この方法を用いることができます。

チェックボックスに関する特殊表記・応用例

「初期値 〉 チェックボックス」をお読みください。全選択のための ALL 表記や、関数との組み合わせで使えるテクニックなどがあります。

選択肢の表示順を変更する

選択肢モデルにおける、選択肢の表示順変更と同じルールを、任意のモデルに設定することができます。

詳細は"リストボックスの説明"をお読みください。

選択肢を無効にする

選択肢モデルにおける、選択肢の無効化と同じルールを、任意のモデルに設定することができます。

詳細は"リストボックスの説明"をお読みください。

無効な項目も検索対象とする

"選択肢モデルにおける、無効な項目も検索対象とする"と同じルールを、任意のモデルに設定することができます。

折り返し数の設定

例

チェックボックスは標準で1行あたり4つの選択肢が表示されるようになっています。

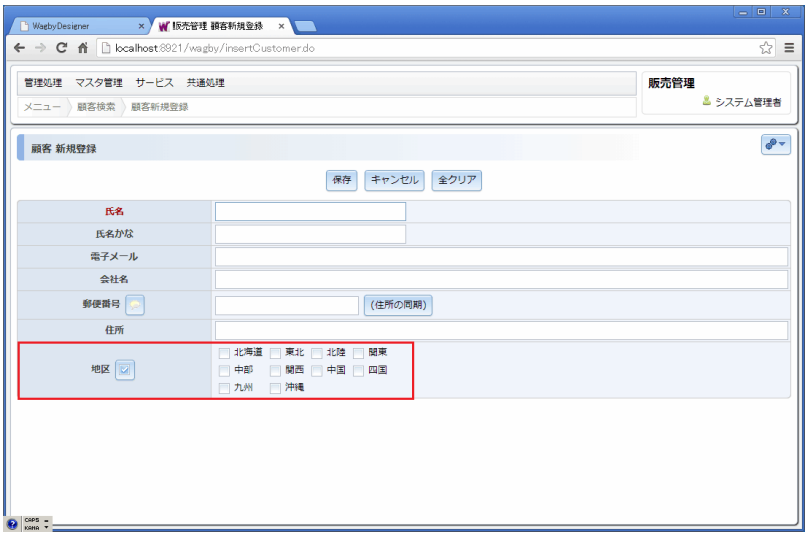


図 4つ表示で折り返し

これを8つ毎に折り返しと指定すると図のように表示されます。

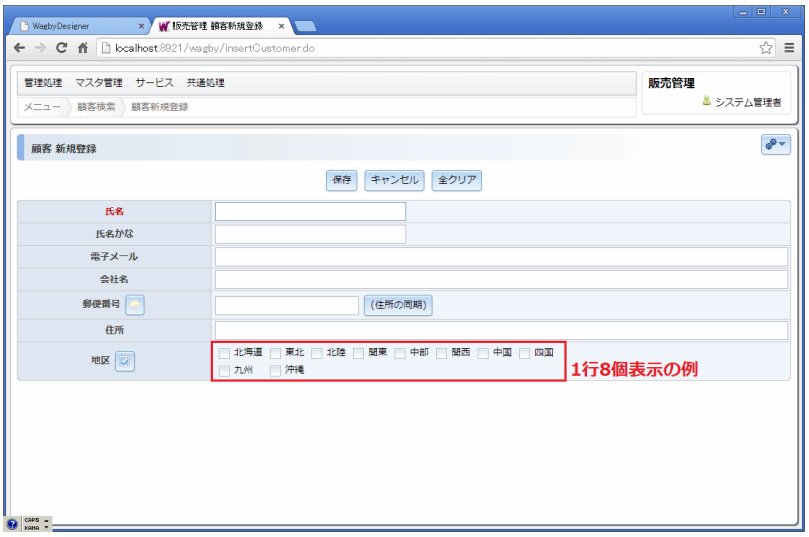


図 8つ表示で折り返し

定義方法

モデル項目詳細定義ダイアログの入力制御タブから、モデル参照内の「チェックボックス・ラジオボタンの折り返し数」を指定します。標準は〝4〟となっています。

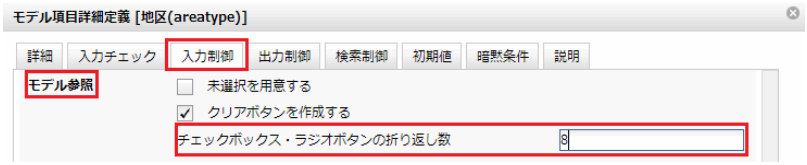


図 折り返し数の設定

検索画面での切り替え時

「検索制御 ＞ モデル参照 ＞ 表示方式の切り替え」を指定した場合にも、この設定は反映されます。

例えば登録や更新画面ではリストボックスだが検索画面ではチェックボックスにするとした場合、検索画面の項目の折り返し数は本設定が使われます。

一覧画面の設定

〝出力制御 ＞ 一覧画面 ＞ モデル参照 ＞ 1行に表示するチェックボックス・ラジオボタンの表示数〟を設定してください。

「チェック画像」の表示切り替え

例

詳細表示画面では、選択したデータの横にはチェック画像が表示されます。

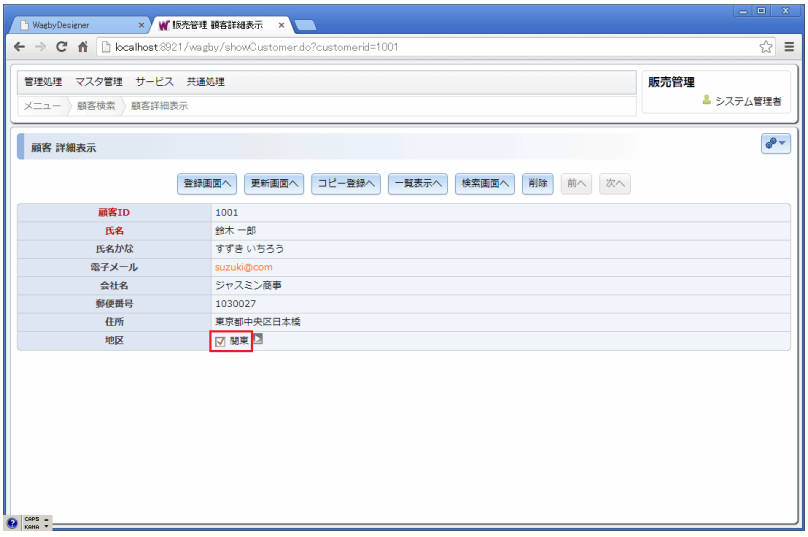


図 チェック画像が表示される（標準）

このチェック画像を非表示にすることができます。

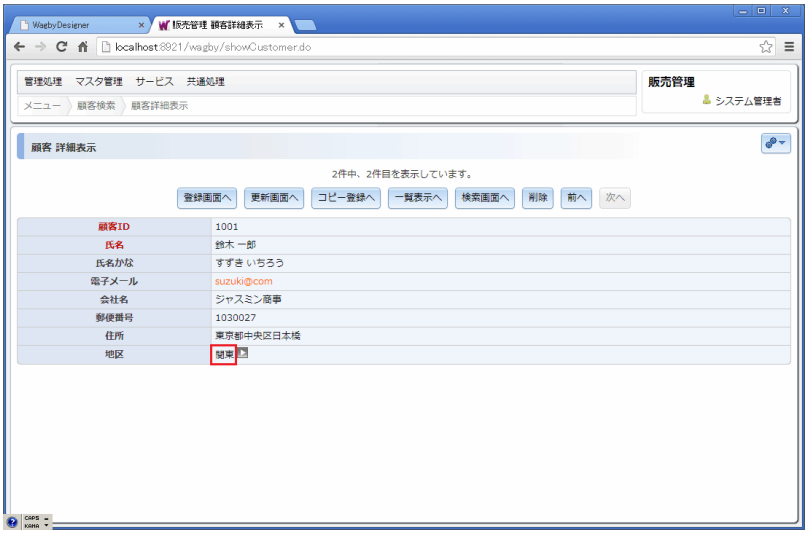


図 チェック画像を非表示とした例

定義方法

モデル項目詳細定義の出力制御タブを開きます。詳細画面内に用意されたモデル参照の「チェックボックス選択時の画像を表示しない」を有効にします。（標準では、無効となっています。）

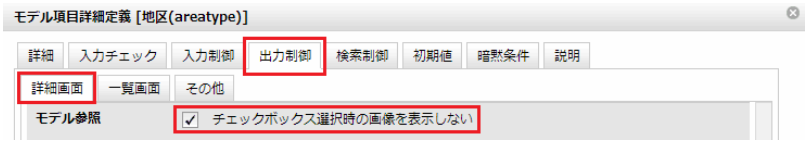


図 チェックボックス選択時の画像を表示しない設定

仕様・制約

- この設定は詳細画面のみで有効です。一覧画面用の設定も別途、用意されています。
- 参照連動項目の参照先がチェックボックス型の場合、参照先のチェックボックスの「チェックボックス選択時の画像を表示しない」設定が詳細画面/一覧画面に反映されます。登録・更新画面には反映されません。

一覧表示画面への適用

例

「折り返し数」と「チェック画像の表示切り替え」を一覧表示画面に設定することができます。 図に、折り返し数を〝1〟とし、チェック画像を非表示とした一覧表示画面の例を示します。

図 一覧画面の例

定義方法

モデル項目詳細定義ダイアログの出力制御タブ内にある「一覧画面」タブを選択します。モデル参照欄の「チェックボックスの選択された画像を表示しない」「1行に表示するチェックボックス・ラジオボタンの表示数」をそれぞれ設定します。

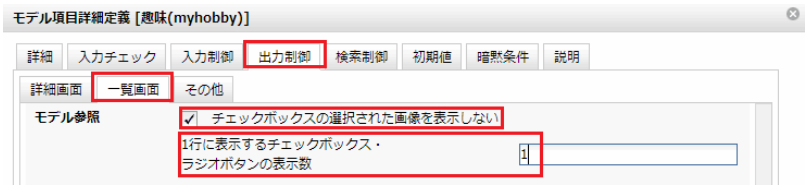


図 一覧画面の設定

仕様・制約

本設定はグリッド形式には対応していません。

選択されなかった項目名を表示する

例

詳細画面では、選択した項目のみが表示されます。これが標準の動作になります。

図 詳細画面での表示

選択されなかった項目名を表示すると設定した場合は次のようになります。

図 選択されなかった項目名を表示する設定を有効にした場合

定義方法

モデル項目の型詳細設定ダイアログのオプションに用意されている「選択されなかった項目名を表示する」設定を有効にします。

図 モデル項目の型詳細設定ダイアログ

この設定は一覧画面にも反映されます。

検索画面

例と定義方法

例

図に用意した見積書モデルでは、得意先項目を顧客モデルから参照します。入力欄の横に用意された「検索」ボタンを押下すると、図の検索画面が開きます。

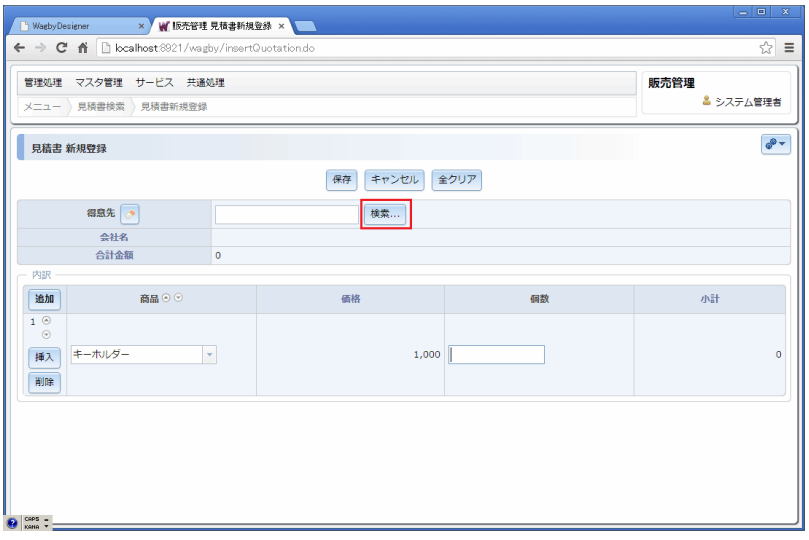


図 見積書の新規登録画面で「検索」ボタンを押下する

図の検索画面において、検索条件で絞込んだあとに「一覧表示へ」ボタンを押します。

図 顧客検索画面

一覧表示画面の左側に並んでいる値（ここでは氏名）を選択します。具体的にはマウスでクリックします。

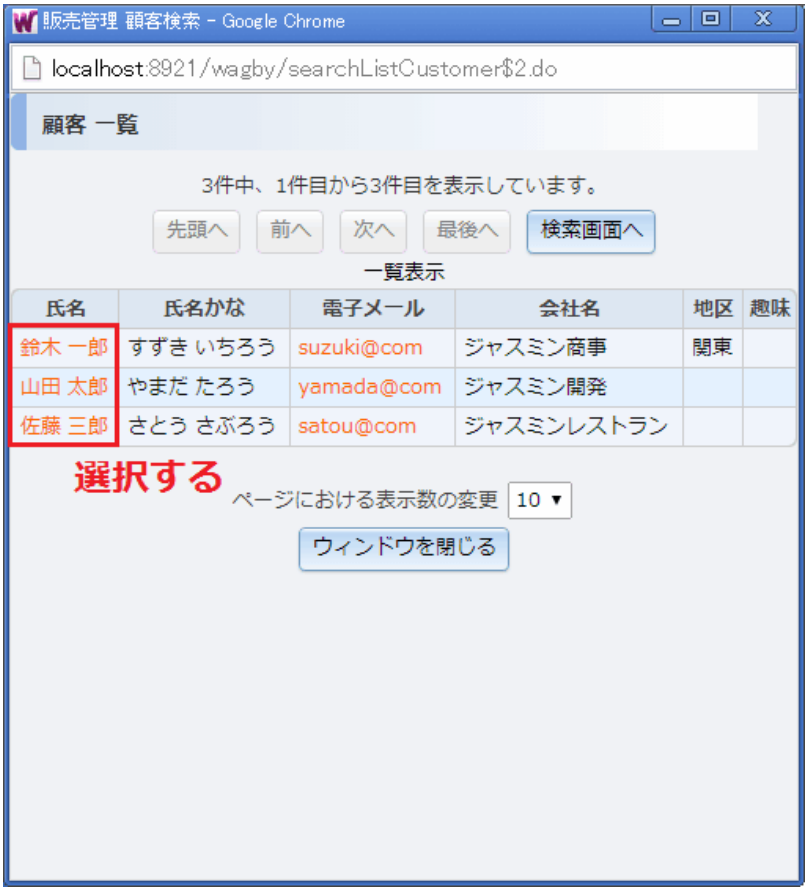


図 一覧表示画面

顧客が選択されます。

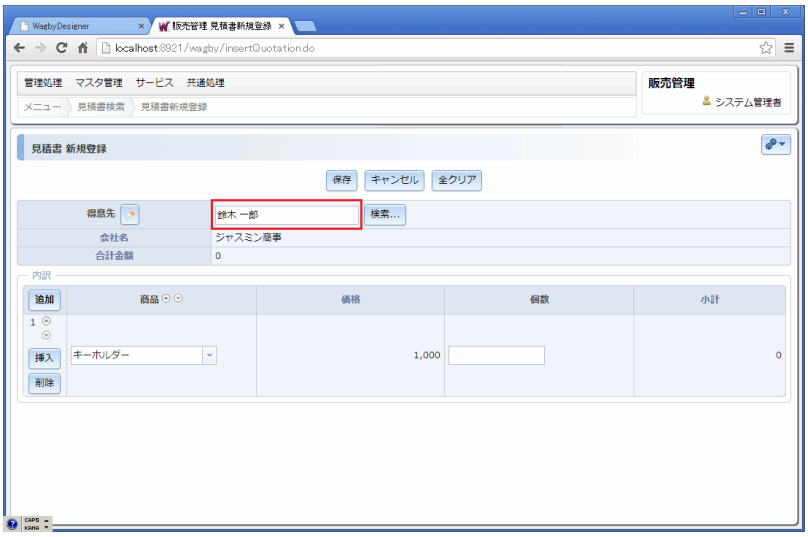


図 顧客が選択された

定義方法

モデル項目の型詳細設定ダイアログを開きます。入力タイプに「検索画面」を選択します。

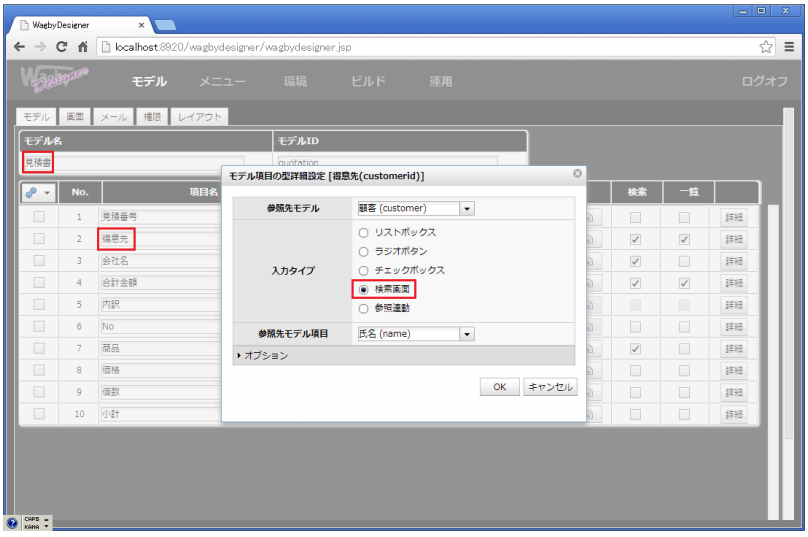


図 検索画面を利用する設定

参照先モデルのデータが削除されていた場合

参照先のデータが削除されていた場合、詳細表示画面には〝存在しないデータ〟と表示されます。

文言を変更する

この〝存在しないデータ〟という文言を変更することができます。

複合キーのモデルを参照する

「リストボックス ＞ 複合キーのモデルを参照する」をお読みください。

仕様・制約

- サブウィンドウ画面は、レイアウト指定に対応していません。常にシンプルな横並びのレイアウトになります。(*1)
- ″主キー（コード）を直接入力する″設定の仕様・制約はこちらをお読みください。
- ″メインウィンドウの入力値を検索条件として再利用する″設定の仕様・制約はこちらをお読みください。

(43)

入力フィールド長

入力フィールド長は次のルールで定められます。

- 入力フィールド長を変更することができます。詳細は″入力制御>入力フィールド長″をお読みください。
- 入力フィールド長を指定しない場合、Webブラウザが自動的に調整します。
- 繰り返しコンテナ内に用意した項目は、標準でコンテナの列の幅をすべて使うため、スタイル指定width:100%が暗黙で設定されます。この項目を検索条件に用いたときも同様のスタイル指定が適用されます。この動作を変更する場合、入力フィールド長を指定してください。

選択情報の全クリア

例

選択した内容を解除する場合、項目名の横に用意されたクリアボタン（消しゴムの形をしたアイコン）を押下します。（マウスカーソルをのせると説明文が表示されます。）

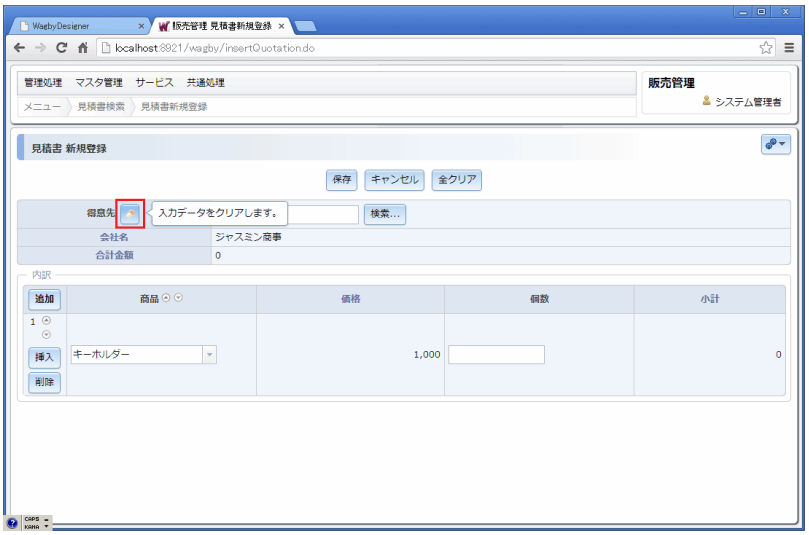


図 選択状態を解除するクリアボタン

選択状態が解除されます。

(43) 1. サブウィンドウは一覧表示画面と全く同じ項目を用いるのではなく、他モデル参照(検索画面) で参照している項目が一覧に自動的に追加されることがあります。開発者が用意したレイアウトに対して Wagby が自動的に追加した項目を含めた状態で、更にレイアウトを自動調整することは難しいため、サブウィンドウではレイアウトの設定は反映していません。

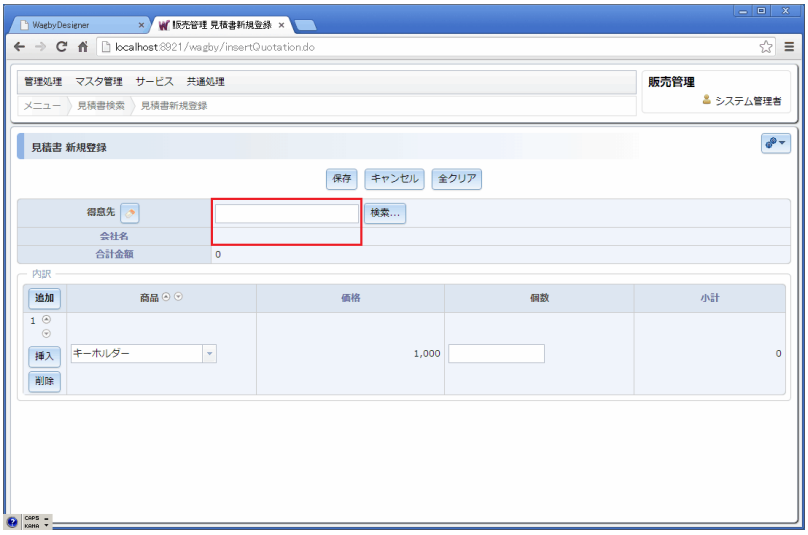


図 未選択の状態に戻った

クリアボタンを非表示にする

モデル項目詳細定義ダイアログの入力制御タブ内に用意された、モデル参照の「クリアボタンを作成する」を無効にします。

図 クリアボタンを作成する

クリアボタンが用意されない例を図に示します。この場合、一度何かを選択すると、未選択という状態に戻すことはできません。

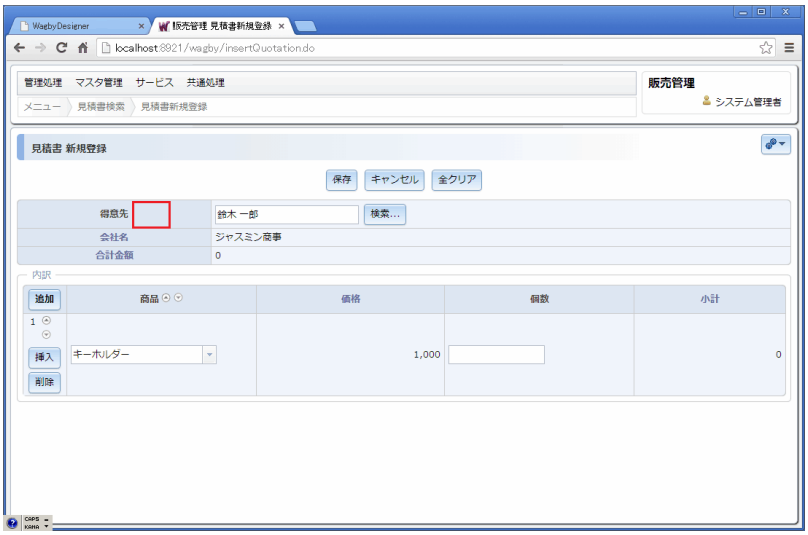


図 クリアボタンは非表示となる

本設定が無効となる場合

検索ボタンを非表示に設定すると、本クリアボタンも自動的に非表示となります。

検索ボタンの名称変更

例

標準の「検索...」ボタンを「検」へと変更した例を図に示します。

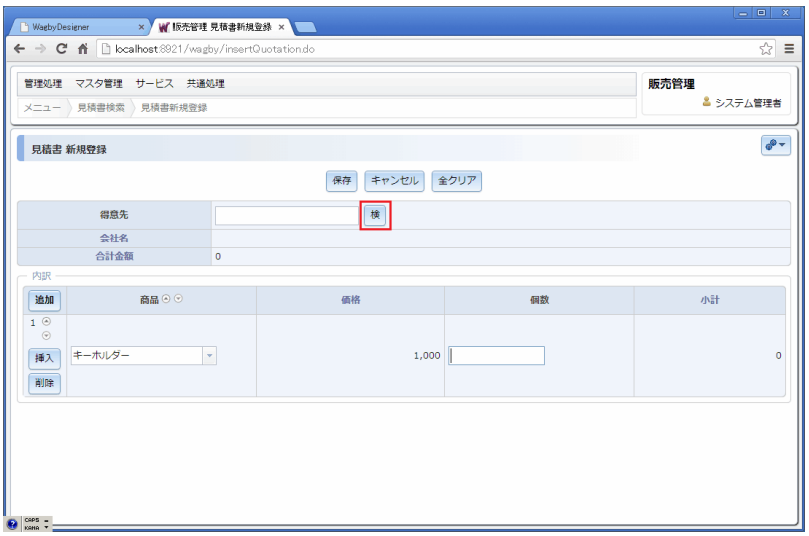


図 ボタン名を「検」へ変更した

定義方法

モデル項目詳細定義ダイアログの入力制御タブを開きます。「モデル参照（検索）」内の「ボタンを表示する」を有効とし、ボタン名を入力します。

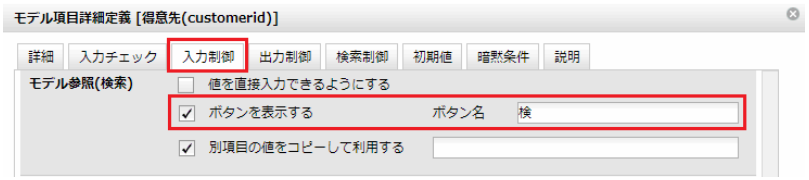


図 ボタン名を設定する

この設定は登録・更新画面に加え、検索画面にも反映されます。

検索ウィンドウの幅と高さ

検索画面(サブウィンドウ)の標準の高さと幅を指定できます。初期値は500ピクセルとなっています。

(44)

表示される項目のルール

サブウィンドウに表示される項目は、そのモデルで「一覧表示」を有効とした項目となっています。しかし、有効としなかった項目であっても（サブウィンドウに）表示される場合があります。

理由

サブウィンドウの一覧表示では、サブウィンドウを呼び出した親ウィンドウへ値を渡す必要のある項目は、必ず（一覧表示項目として）表示される必要があります。

(44) この設定はアプリケーション全体で共通です。

例えば、アカウント(juser)モデルは標準で「アカウント」「所属グループ」「名前」が一覧表示の対象項目となっています。 ここで「アカウント」項目を一覧表示から解除したとします。このとき、アカウント一覧表示画面からは「アカウント」項目は非表示となりますが、検索サブウィンドウには「アカウント」項目は表示されたままとなっています。

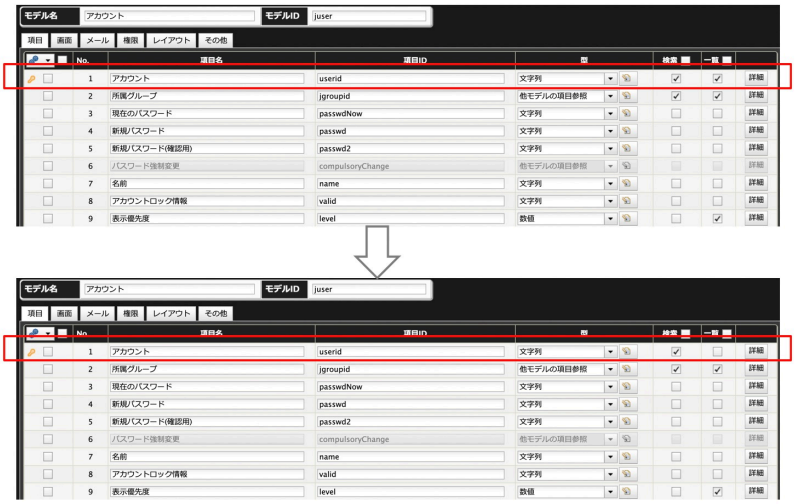


図 アカウントモデルのアカウント項目を一覧表示項目の対象外に変更する

このケースでは、このアカウントモデルをサブウィンドウ検索で呼び出すモデルがあり、これが「アカウント」項目を表示する、という設定があるためでした。

図 別モデルからアカウントモデルを参照する際、参照先モデル項目として「アカウント」項目を指定する

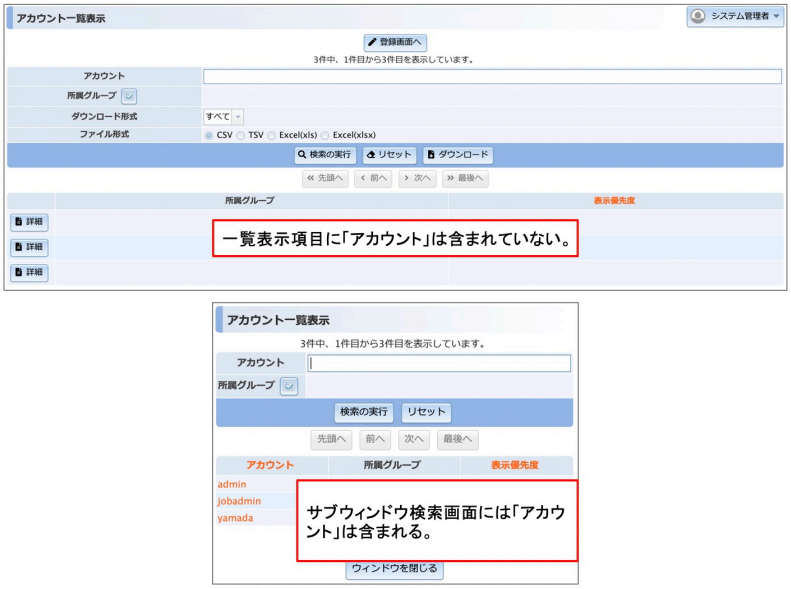


図 「アカウント」項目はアカウントモデルの一覧表示には含まれないが、サブウィンドウ検索には含まれる

では、一覧表示項目ではない「名前」項目を参照先モデル項目として指定した場合はどうでしょうか。この場合、サブウィンドウ検索に「名前」項目が自動的に追加されます。

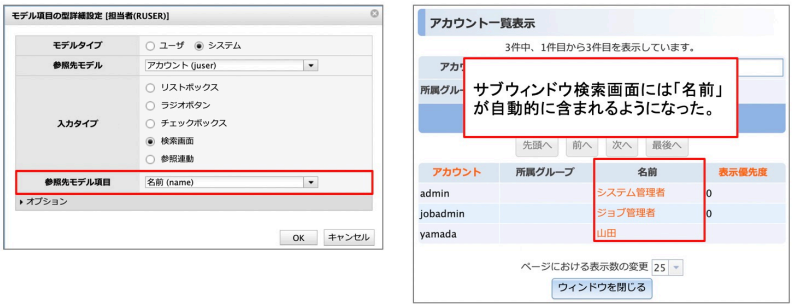


図 「名前」項目が自動的に追加された

モデルAに対して検索画面を使った参照を行う場合、モデルB,C,Dがそれぞれ異なる参照先モデル項目 item1,item2,item3 を指定することができます。このとき、このサブウィンドウはシステム全体で共通利用されるため、結果として item1,item2,item3 すべてがサブウィンドウ画面の一覧表示項目に含まれることになります。

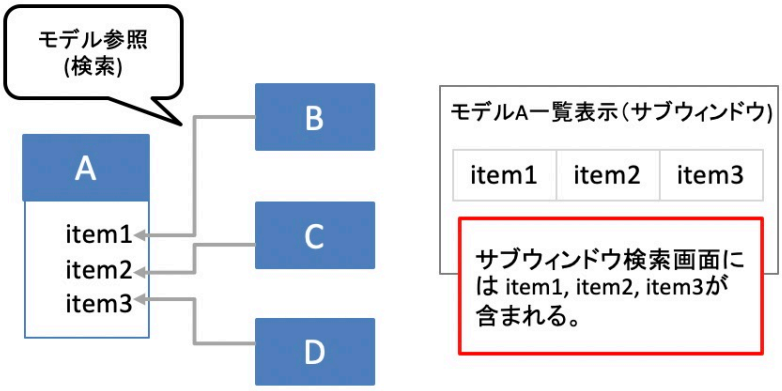


図 指定した項目はすべて一覧表示項目に含まれる

主キー（コード）を直接入力する

この機能と同等以上の使い勝手を「サジェスト」で提供しています。サジェスト機能では主キーの値だけでなく、他の項目の値を入力することもできます。以下の説明は旧版との互換性のために残していますが、できるだけサジェストをご利用いただくことを推奨します。

例

参照先モデルの主キーを直接入力し、値を参照することができます。業務的に、主キーが業務コードとして用いられており、現場の操作者がコード体系を覚えているという場合に有効です。

図では、商品コードを直接、入力しています。「検索...」ボタンは用意されていません。

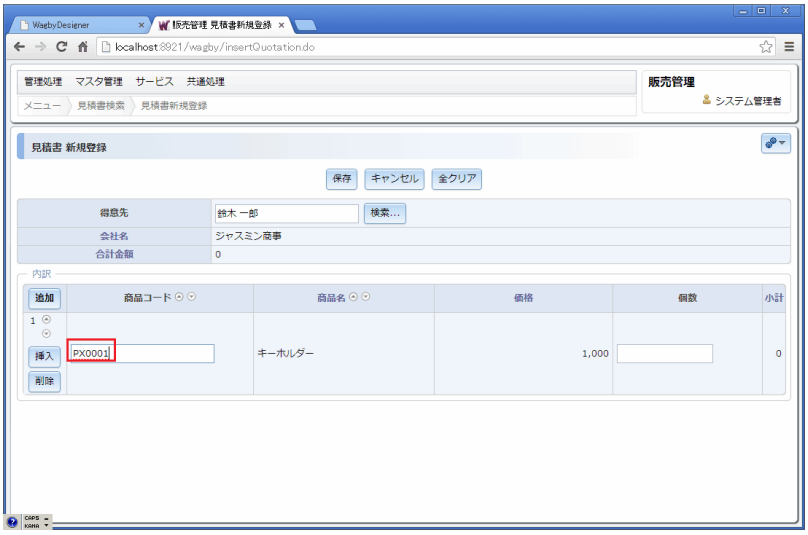


図 コードを直接入力する

商品コードに対応商品名、価格が連動して表示されています。

図 商品コードによる商品名、価格の参照連動

複数の商品情報を入力した例です。

図 商品情報を明細として管理している

なお、存在しない商品コードを入力すると、保存ボタン押下時にエラーとなります。

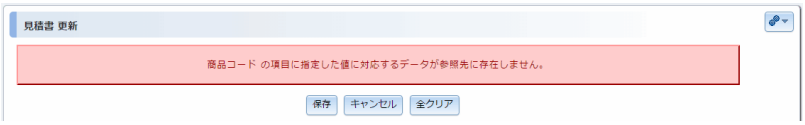


図 存在しないコードはエラーとなる

定義方法

入力伝票となる見積書モデルで「商品コード」項目を用意します。

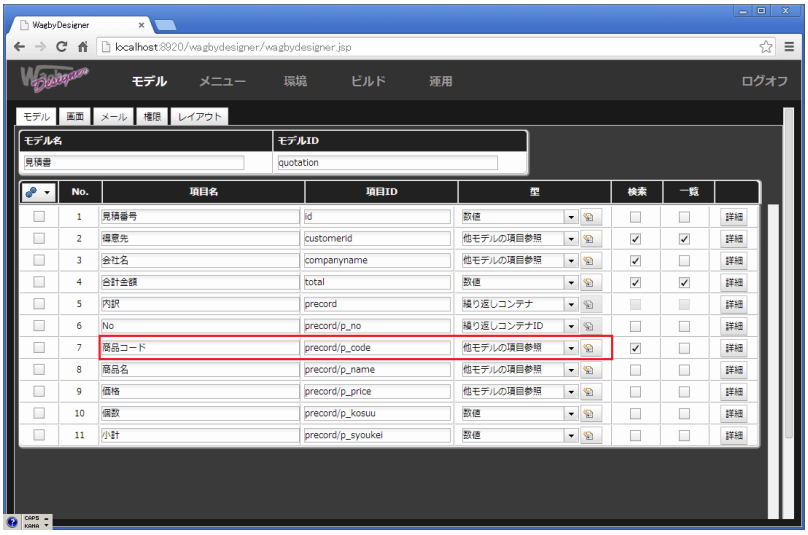


図 商品コード項目を用意する

入力タイプを「検索画面」とします。参照先モデル項目は主キー項目を指定します。ここで主キーは業務コードとしての役割を担っているものとします。

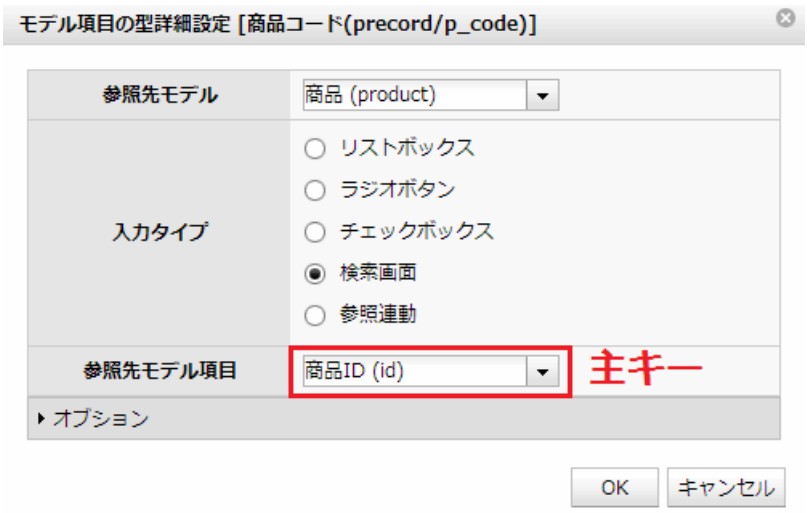


図 参照先モデル項目を主キー項目とする

モデル項目詳細定義ダイアログを開きます。入力制御タブ内の「モデル参照（検索）」欄で「値を直接入力できるようにする」を有効にします。あわせて「ボタンを表示する」を無効にします。（これを無効にし

なかった場合は、直接入力かつ「検索...」機能の両方を利用できるユーザーインターフェースになります。）

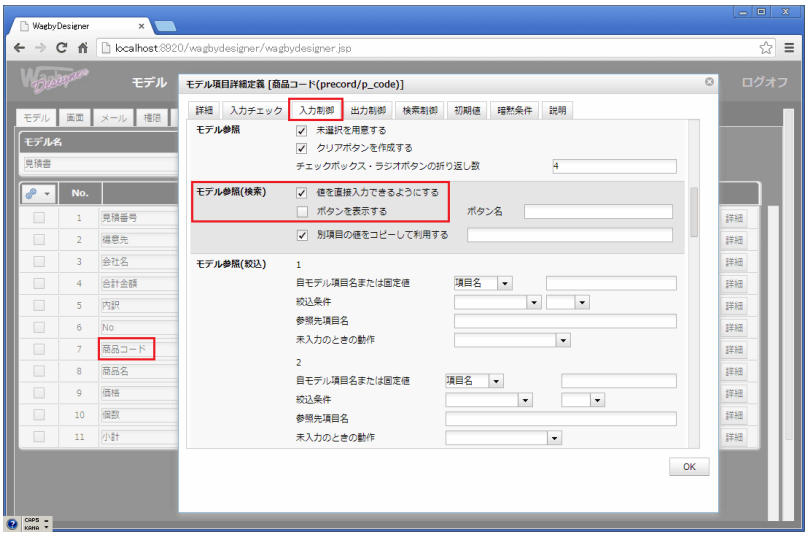


図 値を直接入力できるようにする

仕様・制約

- 本設定は、参照先モデルの主キー項目を直接入力することになります。主キー以外の項目には対応していません。
- 本設定と「自動計算」を同時に定義することはできません。
- 本設定は参照連動にはご利用いただけません。
- 参照先モデルの主キーが複合キーの場合、本設定を利用することはできません。（複数のコード入力欄は用意されません。）
- 参照先モデルの項目がサジェストを利用している場合、本機能と併用することはできません。サジェストをご利用ください。
- 本設定は絞り込みとの併用はできません。絞り込みは下記「${content_7_name}」の説明によって実現されます。

メインウィンドウの入力値を検索条件として再利用する

例

登録、更新画面(メインウィンドウ)の入力値を、検索画面(サブウィンドウ)の検索値として受け渡すことができます。

例を示します。「営業報告書」モデルの項目「顧客名」は「顧客」モデルを参照しています。 ここで「営業報告書」モデルの項目「職種」の値を（「顧客」モデルの）検索条件として再利用させます。

職種の欄に予め値を設定し、「検索」ボタンを押下します。

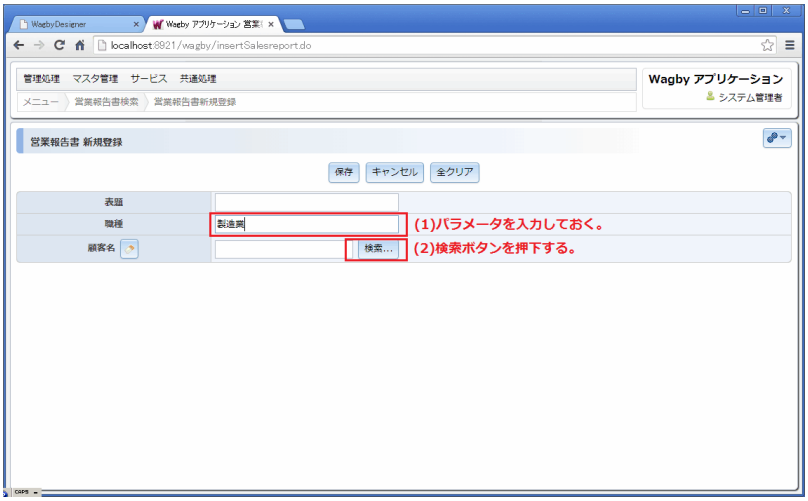


図 検索ボタンを押下する

メインウィンドウ（営業報告書）の値が、サブウィンドウ（顧客モデルの検索画面）に渡されていることがわかります。

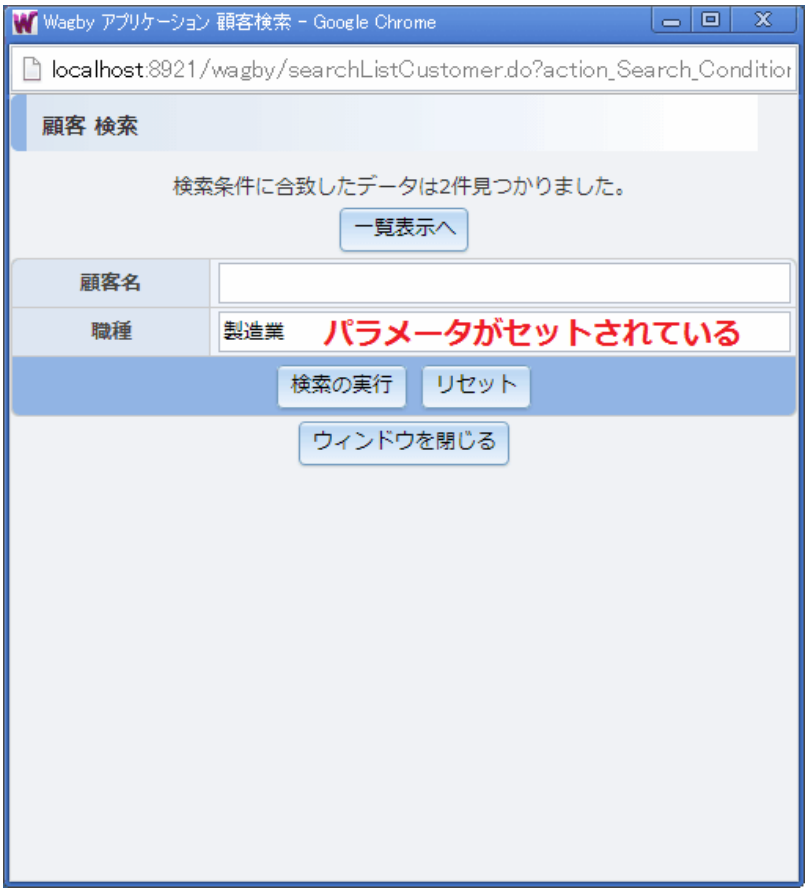


図 検索条件の受け渡し

定義方法

顧客モデルの定義を示します。「業種」項目(jobtype)は検索条件として用います。

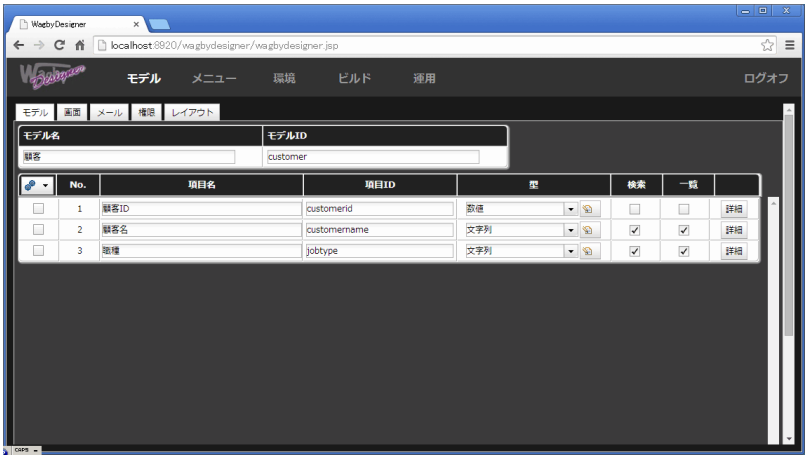


図 顧客モデルの定義

営業報告書モデルの定義を示します。検索条件として引き渡す「業種」項目(customertype)も、同様に検索条件として指定します。

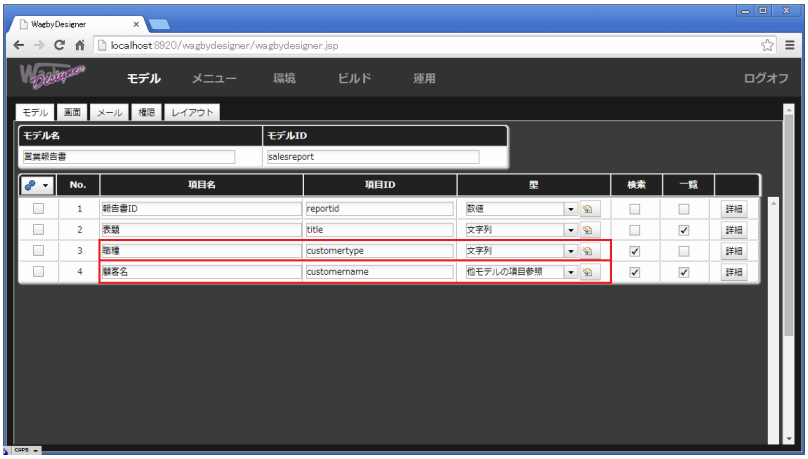


図 営業報告書モデルの定義

営業報告書モデルの「顧客名」は、顧客モデルを参照します。入力タイプは検索画面とします。

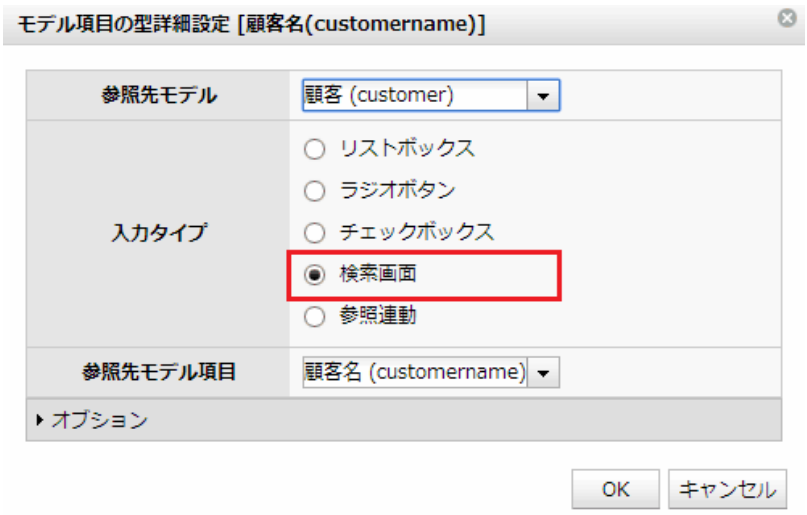


図 関連の設定

絞り込みの設定

ここで、営業報告書モデル（自モデル）内の項目「業種　(customertype)」を、参照先モデル「顧客」の「業種（jobtype)」と関連づけます。これが絞り込み設定になります。

図 双方のモデル内の項目を関連づける（絞り込み）

固定値を渡す

図では（受け渡す値を保持する）項目名を指定しましたが、固定値を渡すこともできます。

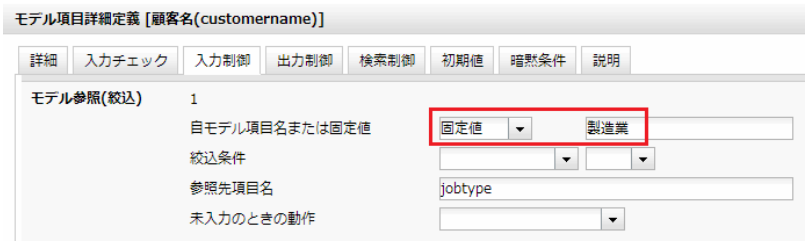


図 固定値を渡す

仕様・制約

- この絞り込みを設定する項目が「検索条件」の場合、検索画面でも本絞込設定は有効になります。そのため、図で指定する「項目名」も検索条件となっている必要があります。（検索画面に用意されない項目を、サブウィンドウに渡すことはできません。）
- 本機能は一覧更新画面および子モデル同時更新画面では動作しません。
- メインウィンドウの読み込み専用項目（計算によって求められた項目も含みます）をサブウィンドウの検索条件に渡すことはできません。渡すことができるのは、入力欄として用意された項目です。隠し項目は利用できます。

この例のような文字列型での絞り込み以外にも、モデル参照項目を使った絞り込み設定も行えます。詳細は"他項目の値で選択肢を絞り込む"をお読みください。

応用例

- 検索条件の隠し項目設定
- 絞り込み設定との併用

サジェスト

"サジェスト"をお読み下さい。

リセットボタンを非表示にする

検索用サブウィンドウ画面のリセットボタンを非表示とすることができます。

「画面 ＞ 検索・一覧表示 ＞ ボタン表示」の「リセットボタン(検索用サブウィンドウ画面)」を解除します。標準では有効（リセットボタンを表示する）となっています。

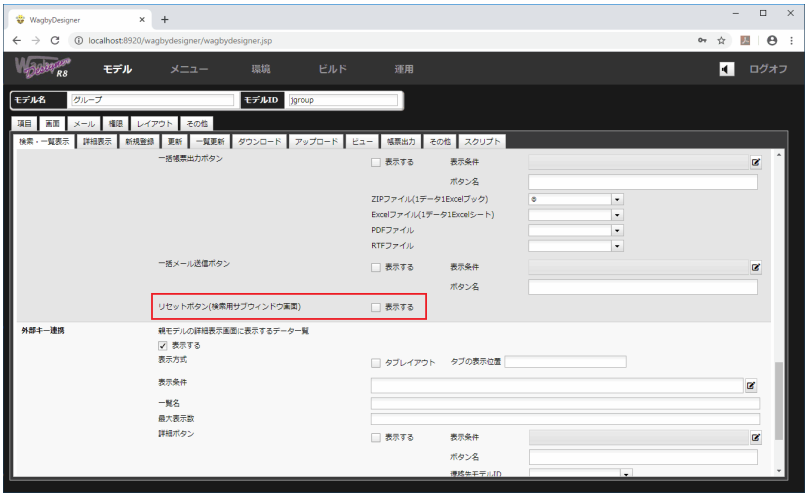


図 リセットボタン(検索用サブウィンドウ画面)の表示設定

例は次のとおりです。リセットボタンが表示されていません。

図 リセットボタンが表示されていない

(45)

参照連動

例と定義方法

例

(45) モデルAの検索用サブウィンドウ画面をモデルB,C,Dで利用する場合、すべてのケースでモデルAの検索用サブウィンドウ画面のリセットボタンは非表示となります。このため、例えばモデルBでは、モデルAの検索用サブウィンドウ画面を使う場合にリセットボタンを表示させたい、というような個別の指定を行うことはできません。

「顧客モデル」と「サポートモデル」を例に説明します。サポートモデルで「顧客名」を参照すると、それに関連する最新の「顧客会社名」が同時に表示されます。この方式によりデータの重複入力の手間がなくなり、かつ、常に最新のデータを参照することができます。

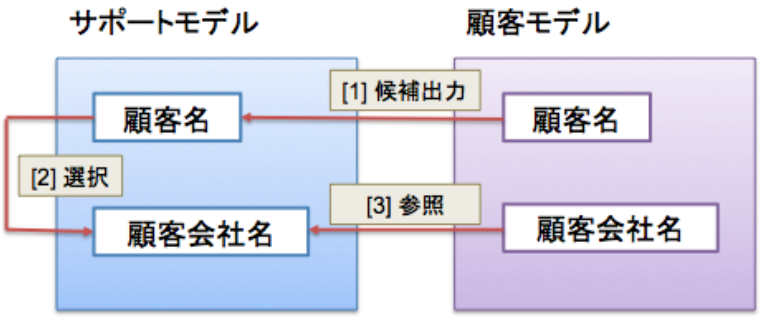


図 関連項目を自動的に参照させる

図に参照連動を設定したサポートモデルの新規登録画面を紹介します。 「検索...」ボタンから、顧客名を確定させます。（この詳細は「参照先モデルの参照>検索画面」をお読みください。）

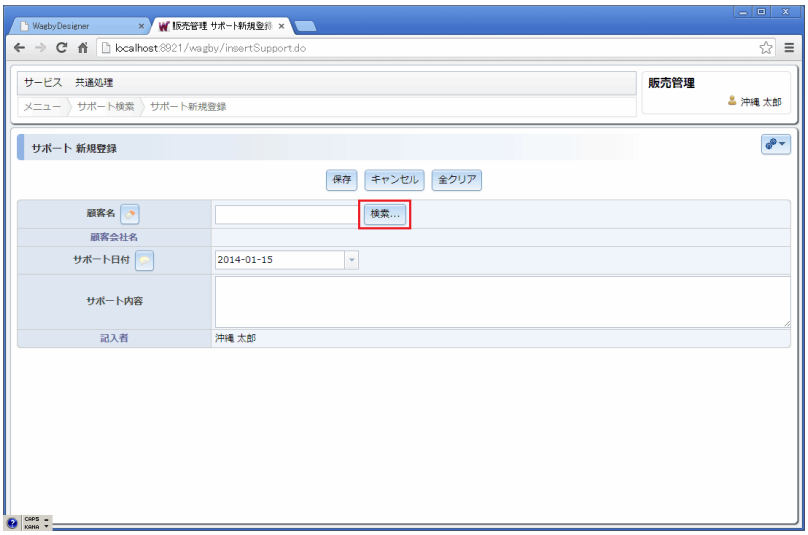


図 サポート新規登録画面

顧客名を確定させると会社名が自動設定されます。会社名項目は参照連動型として定義しています。

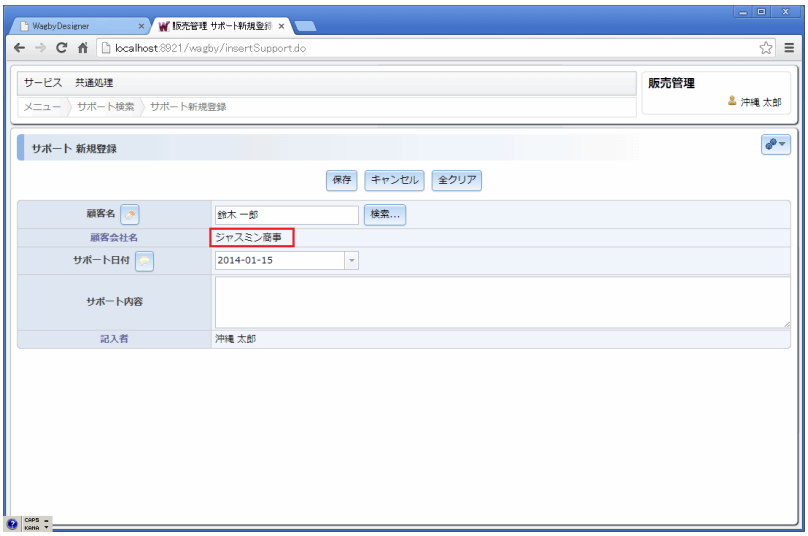


図 顧客を確定させると会社名が自動設定される（参照連動）

定義方法

サポートモデルを図のように定義しています。

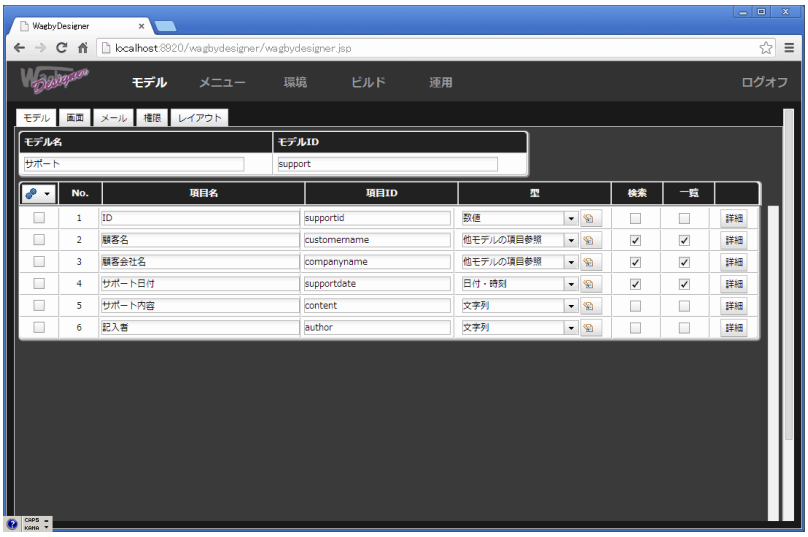


図 サポートモデルの定義

「顧客名」項目は、顧客モデルを参照しています。

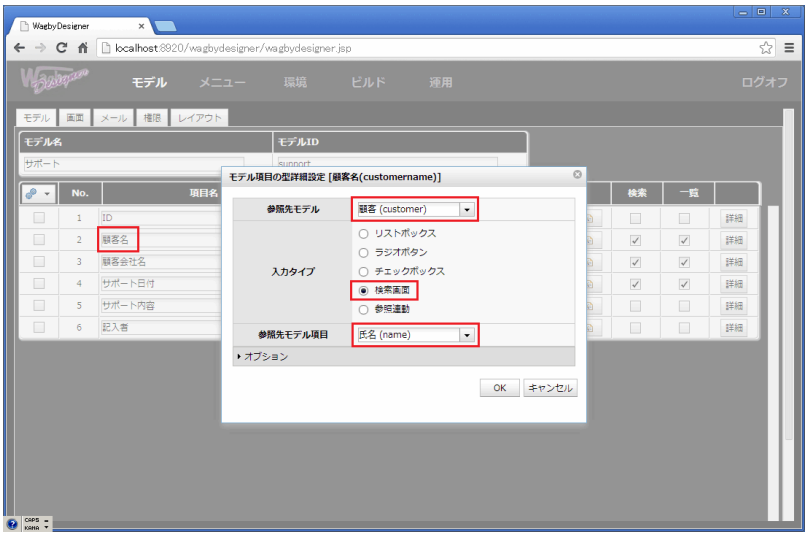


図 顧客名項目の定義

「顧客会社名」項目も、同じ顧客モデルを参照しています。連動する項目名「顧客名」が決まると、参照先モデル項目名「会社名」が定まる、としています。

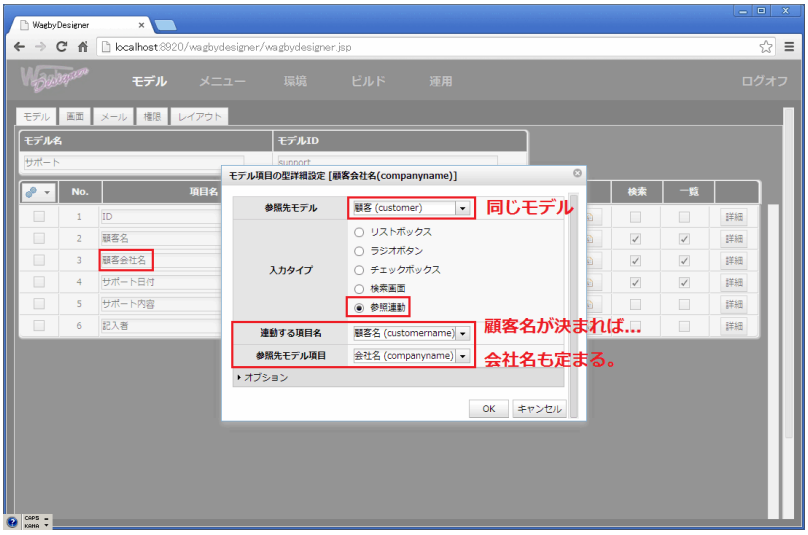


図 顧客会社名を参照連動として定義する

この例のように、モデルには「データベースに保持される情報」だけでなく「画面に表示される情報」も含めることができます。

仕様

データベースとの関係

参照連動項目は標準ではデータベースに保存されません。（テーブルにもカラムが生成されません。）

参照連動を設定すると、データベース保存設定は自動的に解除されます。表示のタイミングで、データベースから最新の値を取得します。

（保存できるように設定することもできます。詳細は"参照連動項目を自モデルに保存する"をお読みください。"参照連動項目を参照先モデルに保存する"も関連する内容です。）

図 参照連動を設定すると、データベース保存設定は自動的に解除される

自動的に読み込み専用になる

参照連動項目は自動的に「読み込み専用」となります。

図 参照連動項目を選択すると、読み込み専用設定が自動的に行われる

検索時の動作

参照連動が数珠つなぎになっている場合でも、検索することができます。

参照連動 > 参照連動 > 参照連動 > 文字列型項目

参照連動 > 参照連動 > モデル参照(検索画面)

などのパターンで検索できます。

参照先項目がテキストエリアの場合

参照先項目が文字列型（テキストエリア）の場合、これを参照連動した場合は次のようになります。

- 登録・更新・詳細・一覧画面では、テキストエリア（読み取り専用）で表示されます。
- 参照連動元にて「出力制御>テキストエリア>ラベルで表示する」を有効にすると、詳細画面ではテキストエリアにしない、という設定を行うことができます。

設計上の留意点

参照元項目(上記例ではサポートモデルの顧客名）が"DB保存なし"の場合、参照連動は機能しません。

参照連動型項目に、次の設定を行うことはできません。

- 参照元項目(上記例ではサポートモデルの顧客会社名)が繰り返し項目である。
- 参照元項目が繰り返しコンテナで、参照先項目(上記例では顧客モデルの顧客会社名)が繰り返し項目である。
- 参照先項目(上記例では顧客モデルの顧客会社名)が繰り返しコンテナ内にある。
- データ指定項目(上記例ではサポートモデルの顧客名)に下記以外を指定すること。
  - 項目の型が「参照先モデルの項目参照(リストボックス)」
  - 項目の型が「参照先モデルの項目参照(ラジオボタン)」
  - 項目の型が「参照先モデルの項目参照(検索画面)」
  - 外部キー項目

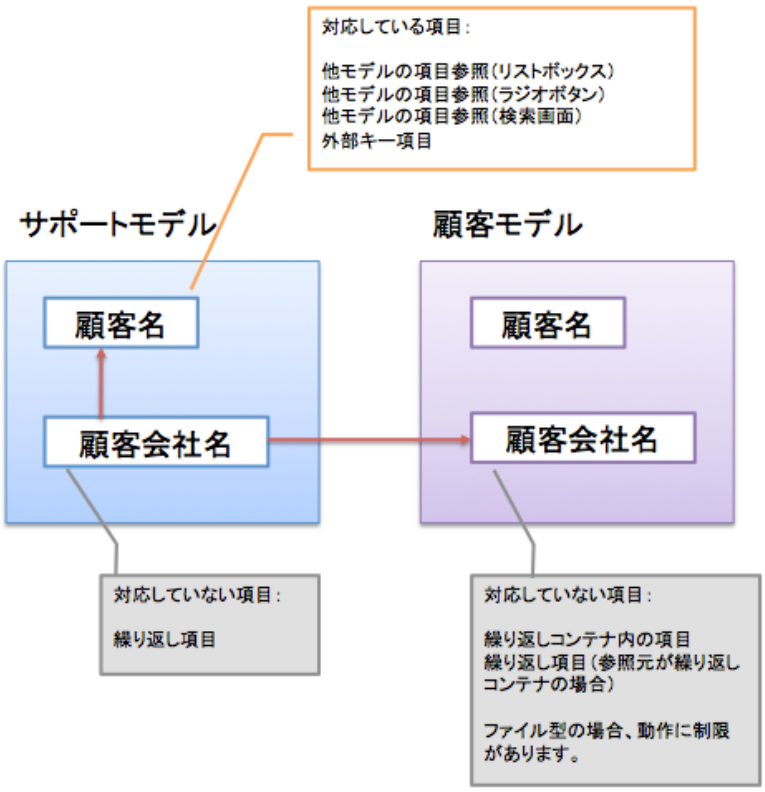


図 参照連動として指定できる項目と、指定できない項目

参照連動項目を自モデルに保存する

参照連動項目をデータベースに保存する設定を行うこともできます。

例

図は、参照連動項目「顧客会社名」をサポートモデルで保存可能とした例です。参照連動された値を修正して保存することができます。

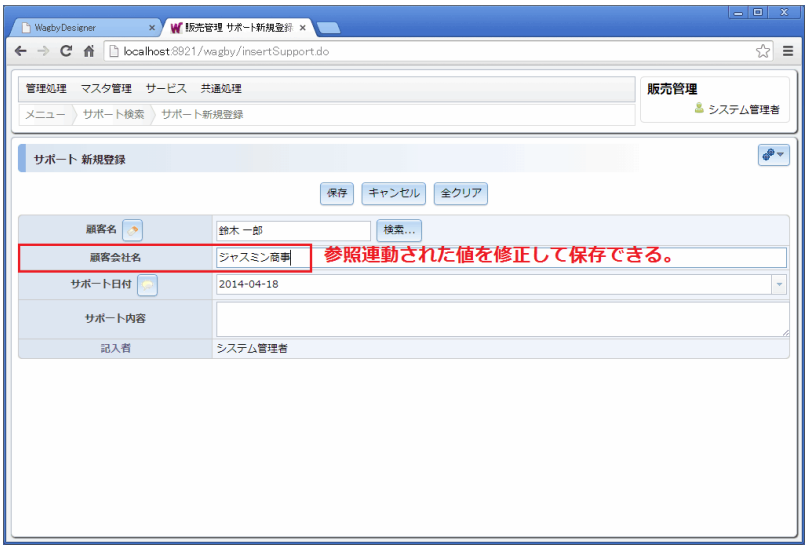


図 参照連動項目を修正して保存する

定義方法

サポートモデルの「顧客会社名」項目の詳細定義ダイアログを開きます。「詳細>データベースの詳細」で、「値をデータベースに保存する」を有効とします。さらに、参照モデルの格納先を「自モデル」とします。

図 参照連動項目の値をデータベースに保存する設定

また、入力できるようにするため、読み込み専用の設定を解除します。

図 読み込み専用設定を解除する

データベースとの関係

保存後の値は、顧客モデルの顧客会社名と切り離されて管理されます。その後、顧客モデルの変更が発生した場合でも、サポートモデル側に反映されることはありません。

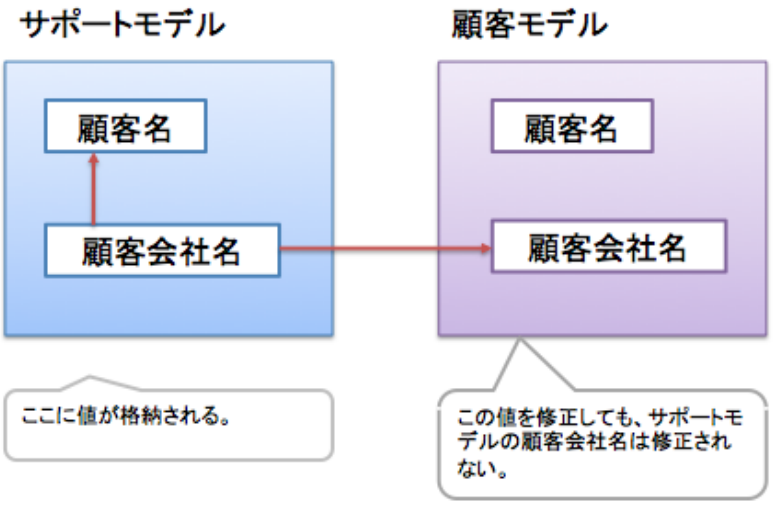


図 自モデル格納のイメージ

この対策として、顧客モデルが修正されたとき、この顧客モデルを参照しているサポートモデルを非同期で更新するというアプローチがあります。この対応は非同期処理のためのメッセージサーバを用意し、かつスクリプトで更新処理を行うというものです。[詳細...]

参照連動項目を自モデルに保存する(上書き)

上で説明した参照連動自モデル保存は次のような動作になっています。

- 参照連動項目が空白であれば、参照連動を機能させる。
- 参照連動項目に値があれば、参照連動させない。

ここで、参照連動元が計算式で求められる場合、次のように変更することができます。

- 参照連動項目に値があっても、参照連動元を変更したときは参照連動させる。

動作例

参照先のモデル「取引先」を用意しておきます。

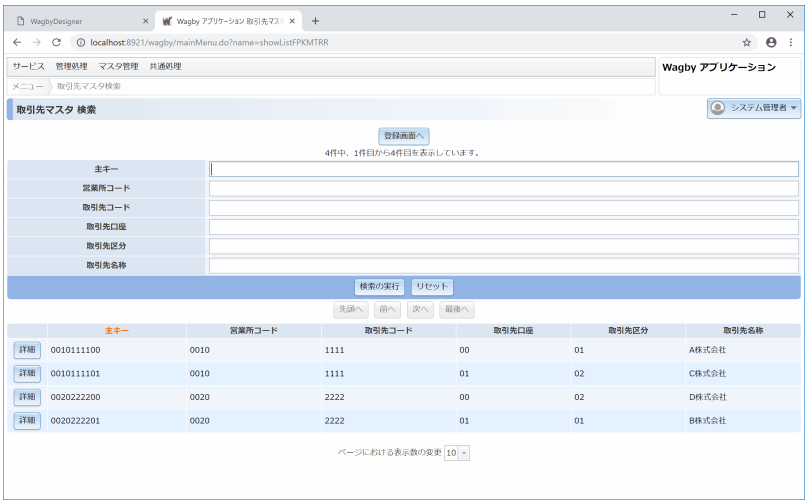


図 取引先マスタ

上の取引先を参照する「受注入力」の新規登録画面を開きます。ここで「取引先」を選択するのはサブウィンドウ（検索画面）ですが、この値のコピーを「取引先キー連動元（リストボックス）」という項目で保持するようにしています。実際の参照連動元となるのは、この項目です。

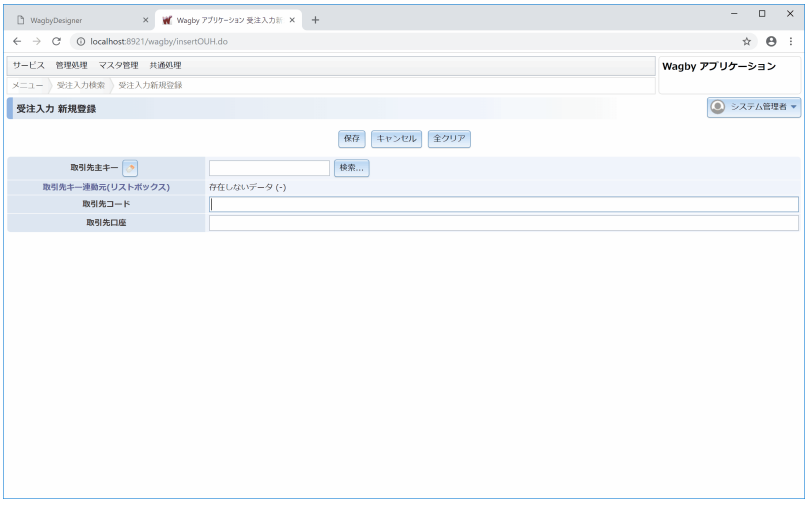


図 受注入力新規登録

(46)

「取引先主キー」を確定すると、その値が「取引先キー連動元（リストボックス）」に転記されます。さらにこの値を元に「取引先コード」「取引先口座」が参照連動されます。これらは自モデル保存項目ですので、参照連動後に値を書き換えて保存することができます。

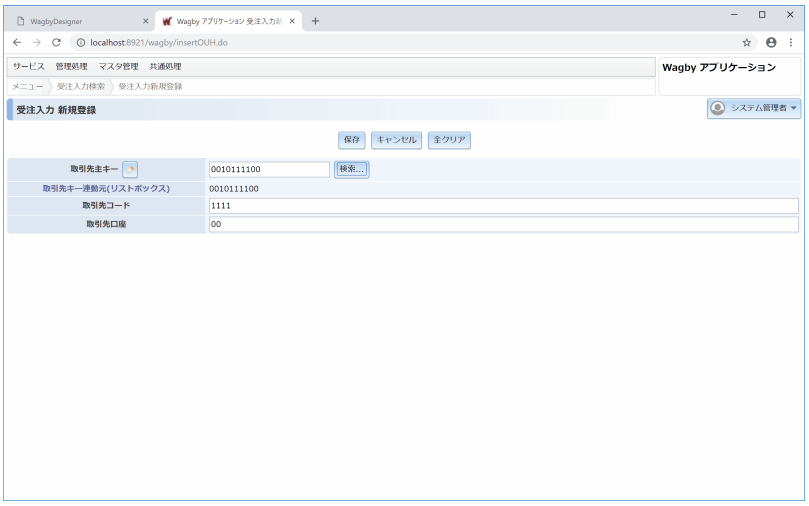


図 取引先主キーを確定する

ここで「取引先主キー」を変更してみます。標準では「取引先コード」「取引先口座」が空白でなければ、参照連動されません。これは参照連動後に値を書き換えた場合、書き換えた値を優先するという動作になっているためです。

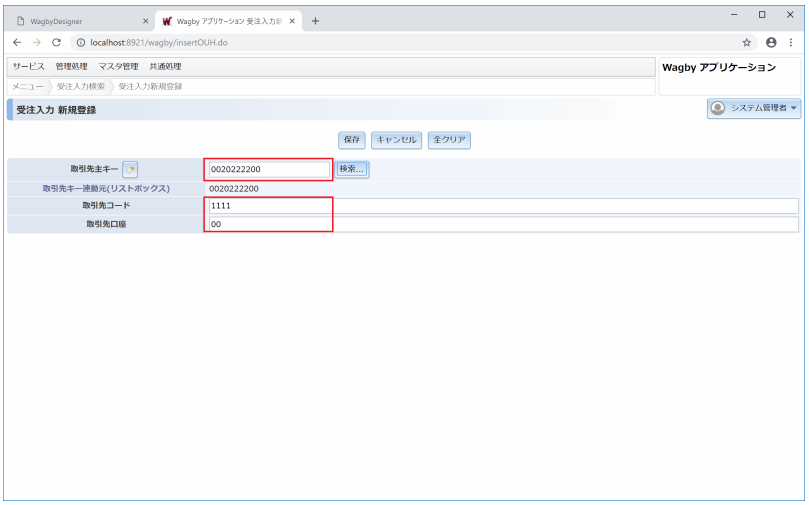


図 参照連動で上書きされない

この動作を変更し、「取引先キー連動元（リストボックス）」が変わった場合は常に参照連動させるという設定を行うことができます。

(46) 本番運用では、「取引先キー連動元（リストボックス）」は隠し項目とするとよいでしょう。

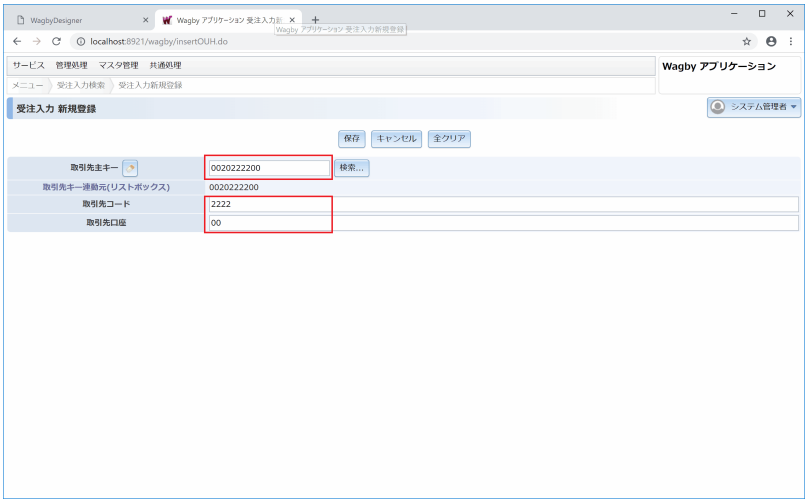


図 常に参照連動で上書きした場合

定義方法

今回の例で用いた「取引先マスタ」と「受注入力」モデルです。

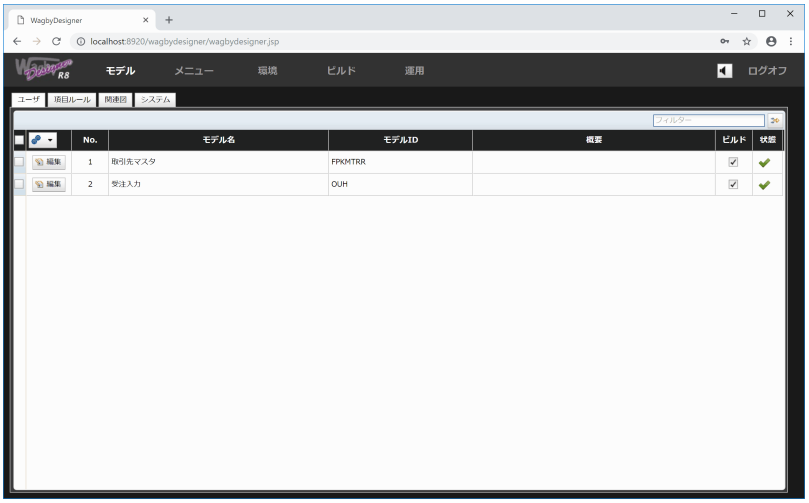


図 「取引先マスタ」と「受注入力」モデル

受注入力モデルです。「取引先キー連動元（リストボックス）」を参照連動元としています。

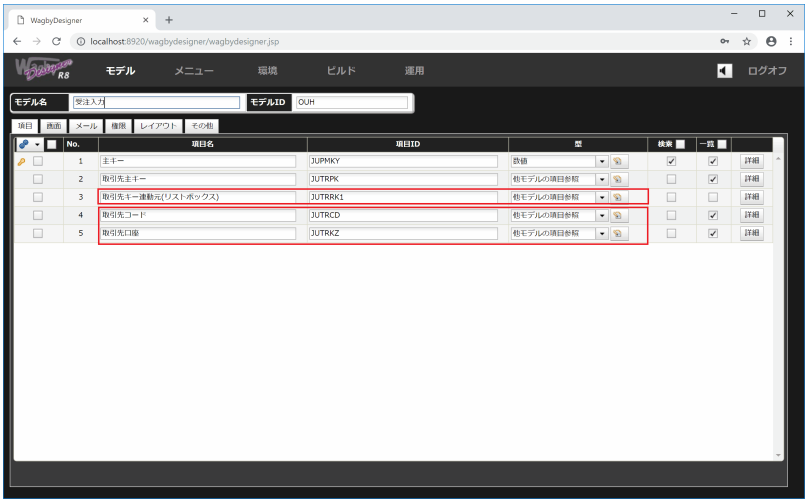


図 「取引先キー連動元（リストボックス）」が参照連動元となっている

「取引先キー」はモデル参照（検索画面）としています。

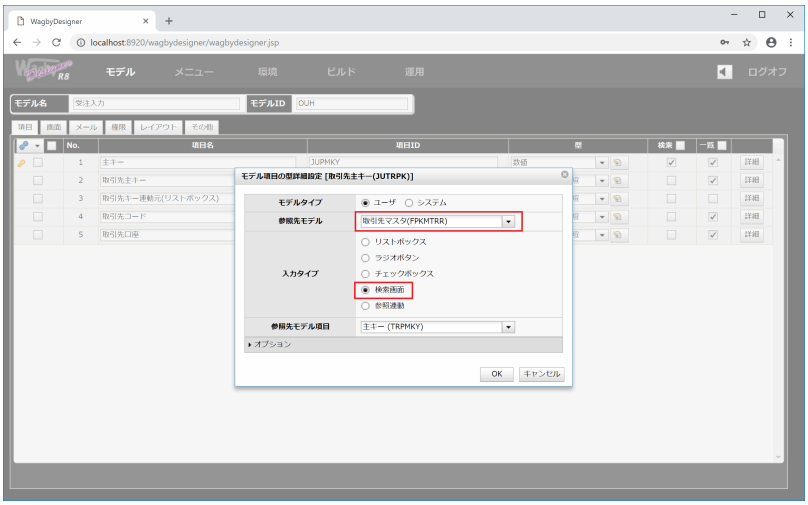


図 「取引先キー」の設定

「取引先キー連動元（リストボックス）」は、モデル参照（リストボックス）としています。

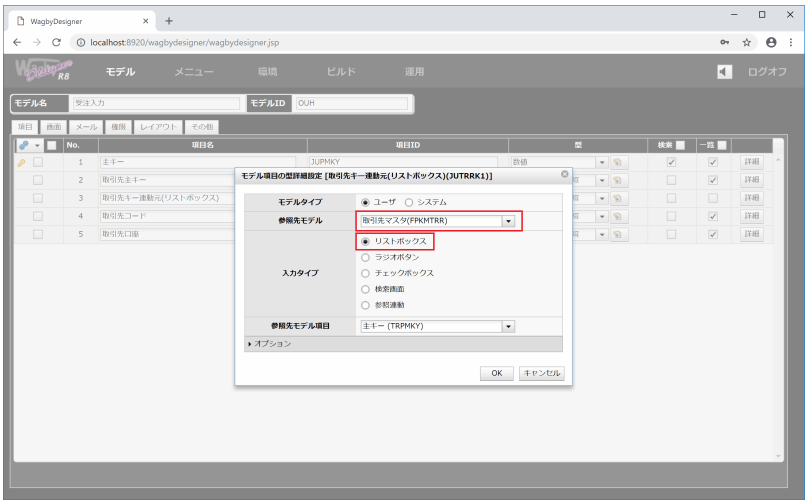


図 「取引先キー連動元（リストボックス）」の設定

この「取引先キー連動元（リストボックス）」は、自動計算によって求められます。今回の設定のポイントは、参照連動元は入力ではなく計算によって求めるようにするというものです。

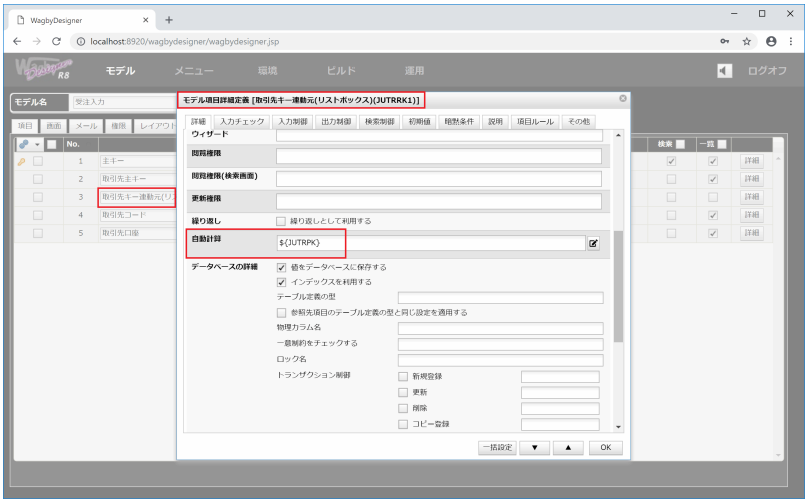


図 「取引先キー」の値を転記する

「取引先コード」「取引先口座」は参照連動です。

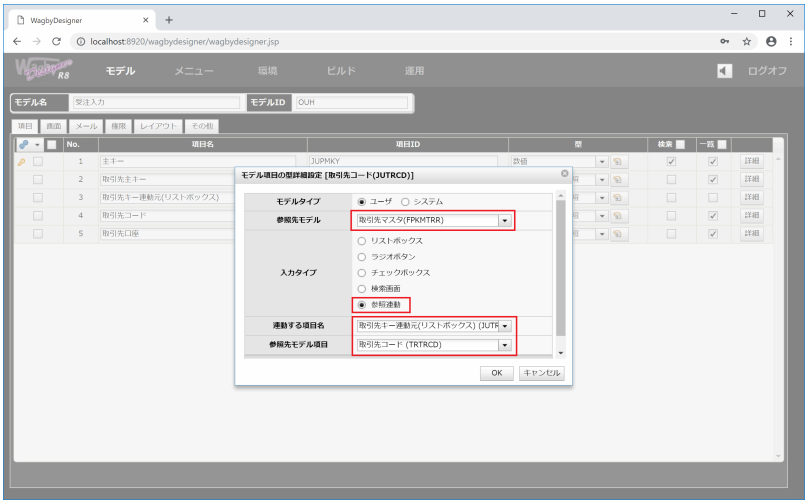


図 「取引先コード」の設定

自モデル保存ですので、値をデータベースに保存します。

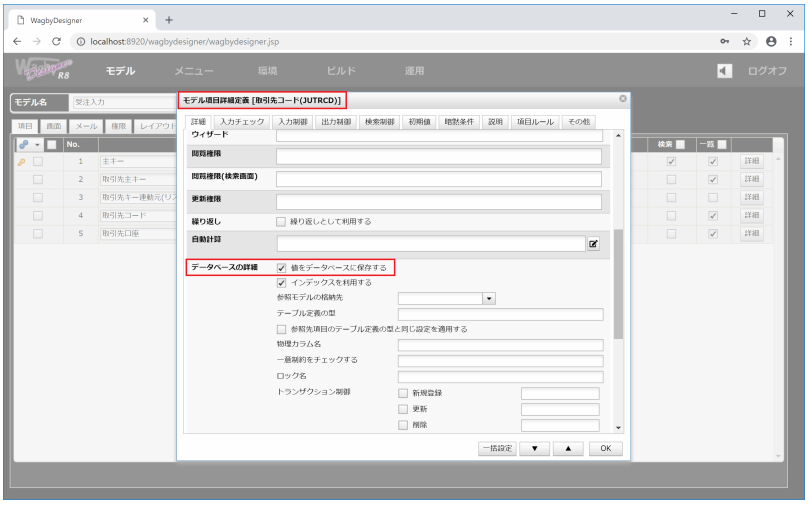


図 値をデータベースに保存する

また、読み込み専用を解除し、入力可能とします。

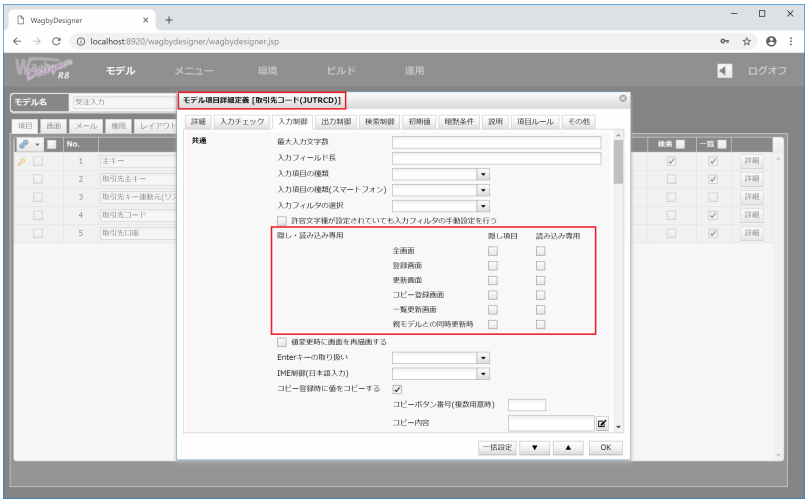


図 読み込み専用を解除する

参照連動元が変わったときに上書きするための設定を行います。モデル参照の設定のオプションから、「連動タイミング（参照元が自動計算）」を ″連動する項目の変更時″ とします。標準は空白であり、空白のときは値が空のときのみ参照連動するようになっています。

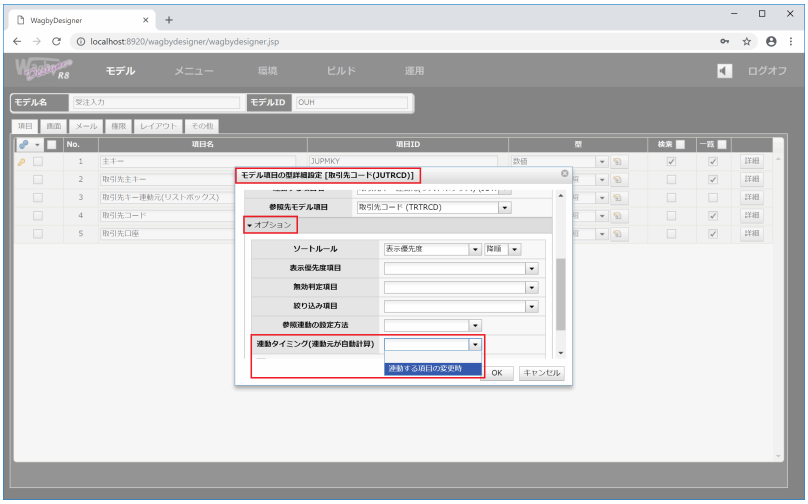


図 「連動タイミング（参照元が自動計算）」の設定

参照連動項目を参照先モデルに保存する

動作例

図は、参照連動項目「顧客会社名」をサポートモデルで保存可能とした例です。修正した値は、参照先である「顧客」モデルに反映されます。

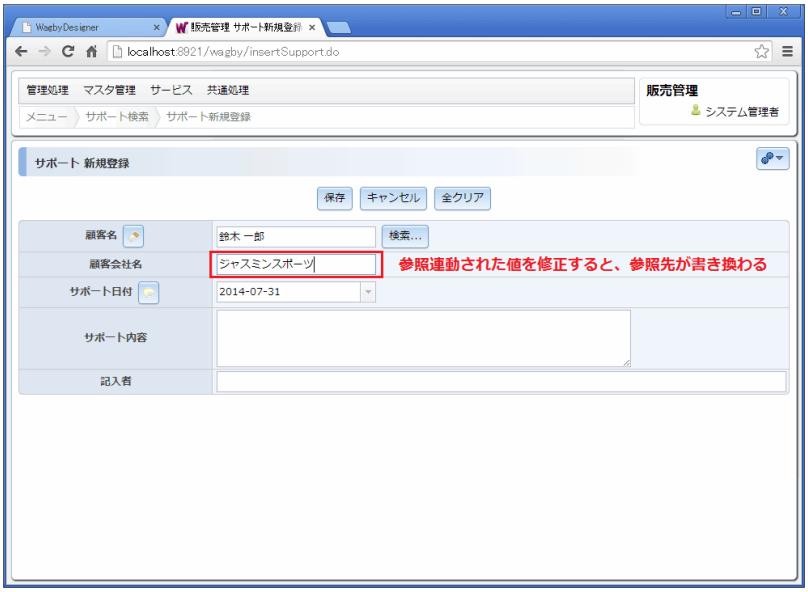


図 参照連動項目を修正して保存する

（参照先である）顧客モデルが修正されています。

図 参照先モデルの項目が修正されている

定義方法

サポートモデルの「顧客会社名」項目の詳細定義ダイアログを開きます。「詳細>データベースの詳細」で、「値をデータベースに保存する」を有効とします。さらに、参照モデルの格納先を「参照先のモデル」とします。

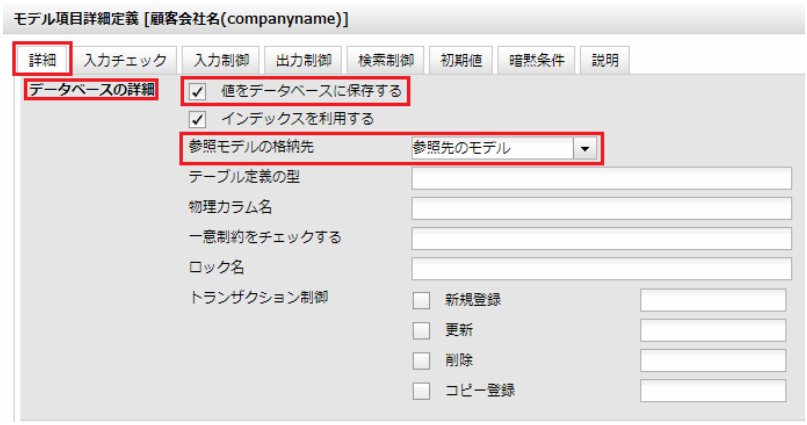


図 参照連動項目の値をデータベースに保存する設定（参照先モデル）

また、入力できるようにするため、読み込み専用の設定を解除します。

図 読み込み専用設定を解除する

データベースとの関係

この設定は「トランザクション」として扱われます。複数のモデルを更新し、データベースにコミットされます。

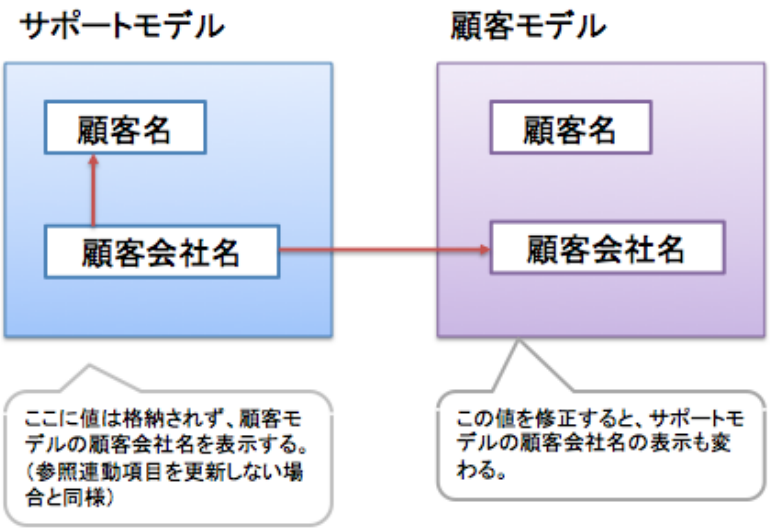


図 参照先モデル格納のイメージ

自動計算式の制約 !-- BR 10836 -->

モデルAがモデルBを参照する関係において、モデルAの保存時に参照先モデルBのitem1項目を書き換えるとします。このとき、モデルBのitem2項目はitem1項目の値を使って計算される項目であったとします。

- モデルBのitem2項目が "値をデータベースに保存する" となっていた場合、モデルAの保存時に参照先モデルBのitem1項目は書き変わりますが、item2項目の再計算は行われません。
- そのため、このようなケースでは、モデルBのitem2項目はデータベースに保存しないようにするとよいでしょう。
- データベース保存が必要な場合は、モデルAの保存時にスクリプトを用いてモデルBの再計算処理を必ず呼び出すように工夫してください。

参照連動項目の入力フィルタ

参照連動自モデル（または他モデル）保存項目の入力フィルタは次のルールとなります。

フィルタ指定を空白とする	フィルタを指定する
参照先項目と同じフィルタが使われます。	指定したフィルタが使われます。

参照連動項目を表示のみの検索条件とする

動作例

検索条件に参照連動の値を表示させることができます。検索項目としては機能しませんので、確認用という位置づけになります。

図 検索条件に参照連動の値を表示する

定義方法

サポートモデル定義の全体を図に示します。「顧客名」と「会社名」は顧客モデルを参照しており、かつ「会社名」は参照連動項目で、検索条件を有効にしています。

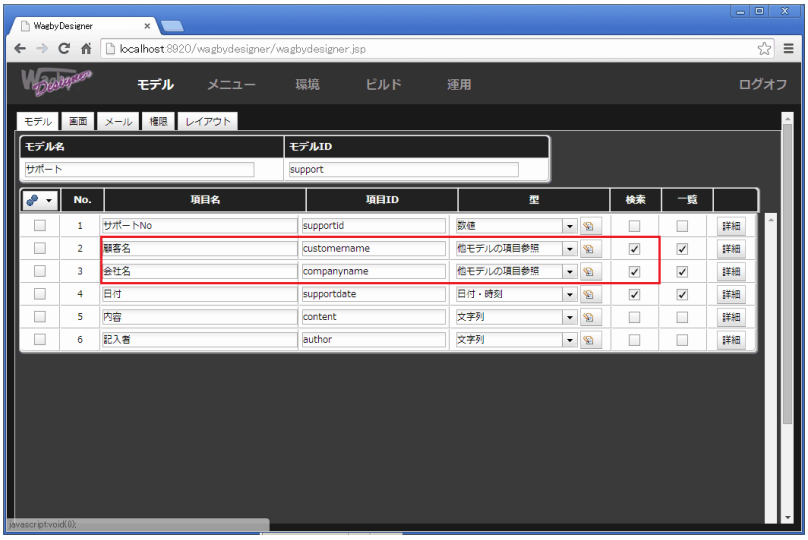


図 サポートモデルの定義

会社名項目について、「検索制御>読み込み専用にする」を有効にします。

図 読み込み専用の設定

制約

本機能は、次の検索画面には適用されません。

- カレンダビュー
- CSV・Excel ダウンロード

参照連動された値を追記する

"固定メッセージの選択・追加による入力方式の実現"をお読みください。

チェックボックス（サブウィンドウ）

例

顧客モデルでの例を示します。図の「趣味」項目はチェックボックス型ですが、「検索..」ボタンが用意されています。

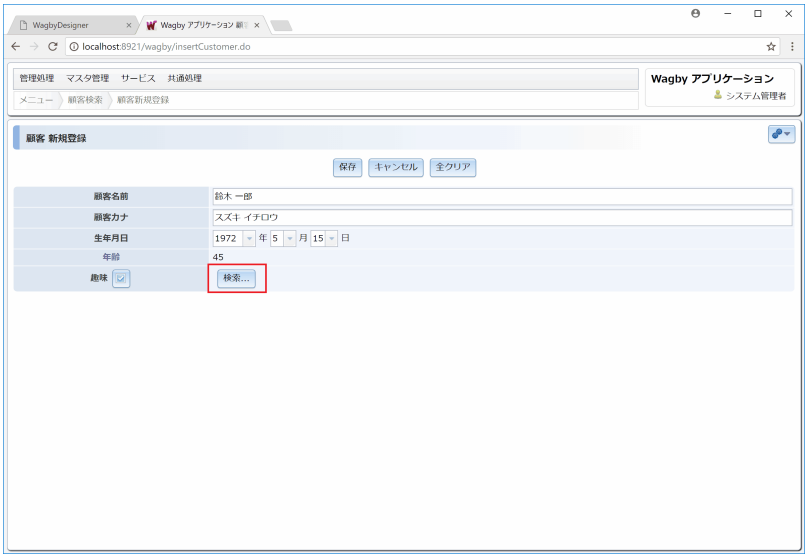


図 チェックボックス型項目の検索ボタン

ボタンを押下するとサブウィンドウが開きます。複数の項目を選択することができます。

図 サブウィンドウ(1)

項目を選択後、「選択値をセット」ボタンを押下します。

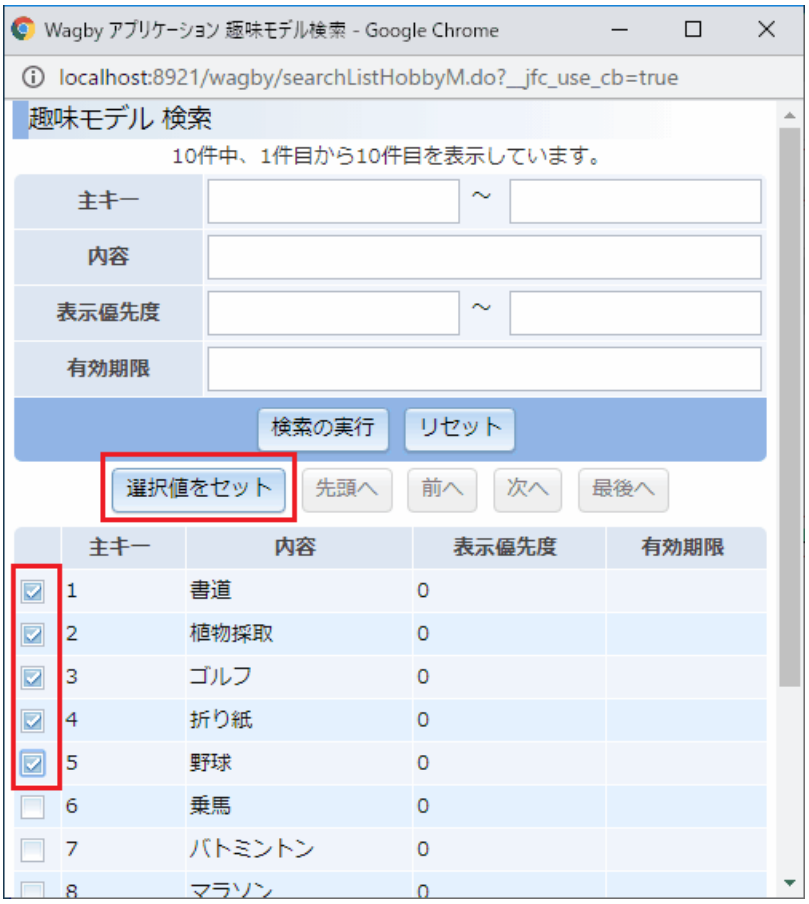


図 サブウィンドウ(2)

選択された値がチェックされた状態になります。

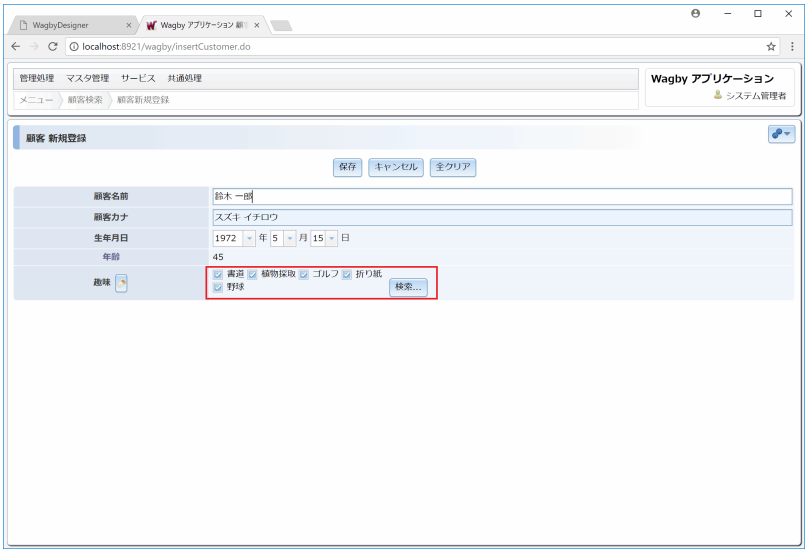


図 チェックボックスが表示される

保存後は、チェックボックス型と同じ扱いになります。

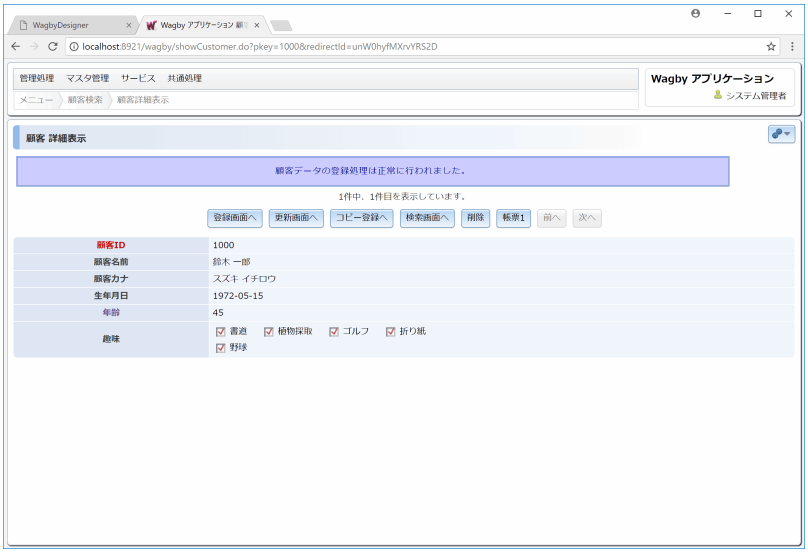


図 詳細表示画面

定義方法

上の例で用意した選択肢モデルは図のとおりです。

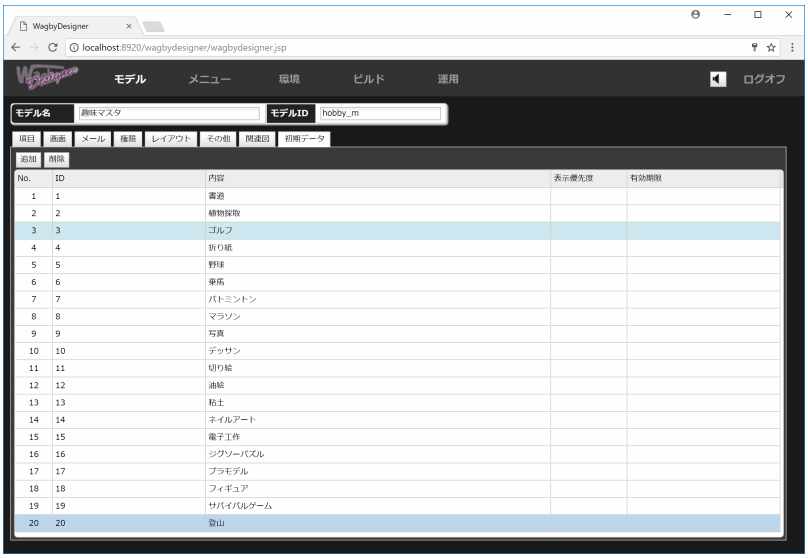


図 サンプルの選択肢モデル

顧客モデル内の「趣味」項目は、「他モデルの項目参照 - チェックボックス」として用意します。

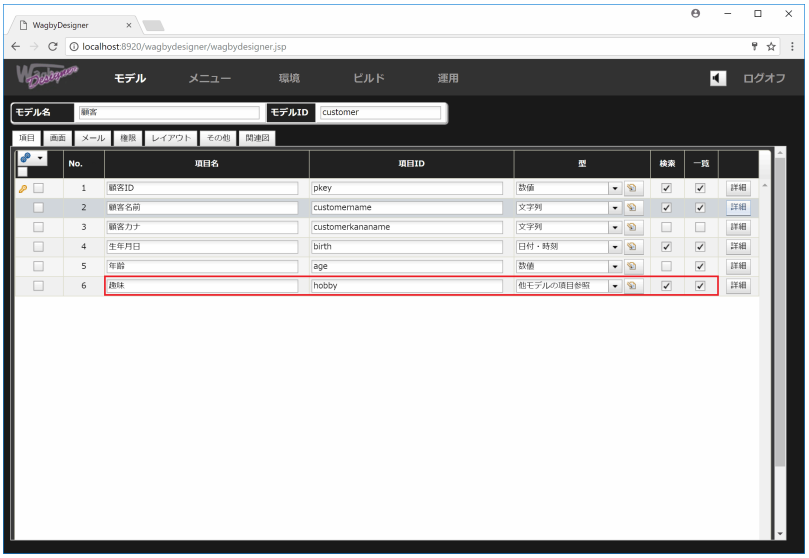


図 趣味項目の定義(1)

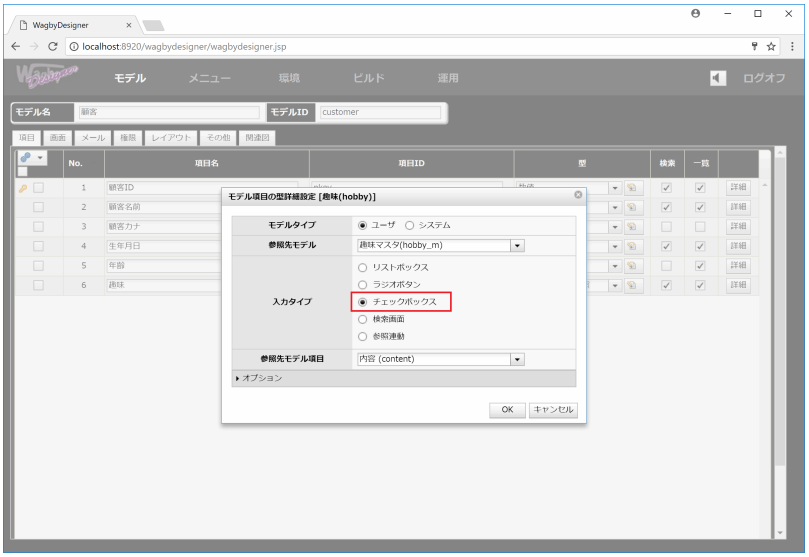


図 趣味項目の定義(2)

チェックボックスの場合、「オプション > チェックボックスの選択方法」を指定できます。ここで"検索画面"を指定します。

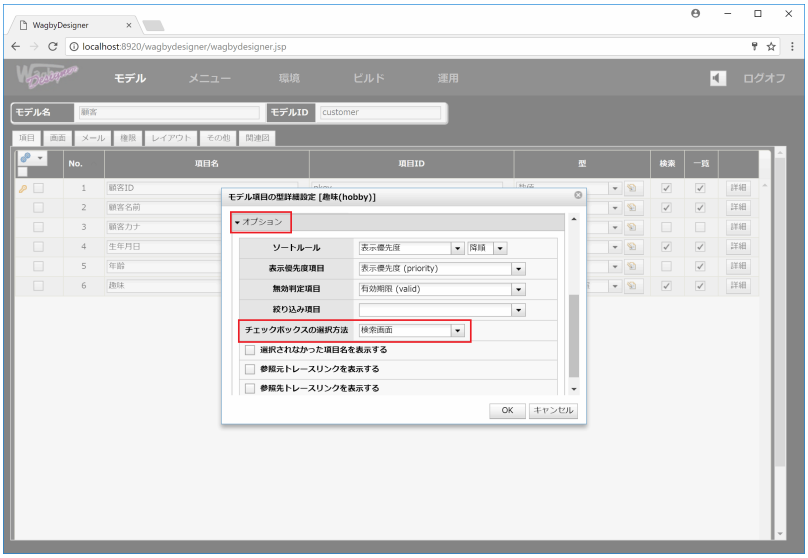


図 チェックボックスの選択方法の指定する

仕様・制約

- この設定は、検索画面にも自動的に反映されます。
- サブウィンドウは「検索画面と一覧表示画面を同時に表示する」設定を有効にしてください。そうでない場合、すでにいくつかを選択状態にしたあとサブウィンドウを開いたとき、現在の選択肢はサブウィンドウに反映されません。
- もともとがリストボックスやラジオボタンで「検索画面での切り替え」の設定でチェックボックスを選択した場合は、本設定は機能しません。
- サブウィンドウでは、項目名をクリックしたソート機能を利用することができません。

チェックボックス（有無フラグ）

背景

Wagbyが提供するチェックボックスは、別テーブルで値を管理します。これは複数のチェックボックスが選択されることを想定しているためです。

しかしユーザーインタフェースはチェックボックスだが、テーブルには一つの値を保持したい（別テーブルに分割する必要がない）というケースがあります。具体的には「有無フラグ」のような二値の状態を保持するものです。

そこで Wagby では、内部構造はラジオボタン（一つの値だけを保持する）が、ユーザーインタフェースはチェックボックスという形式を用意することができます。ここではその方法を説明します。

選択肢モデルを参照する

選択肢モデルを準備する

次に示す選択肢モデルを事前に用意したとします。モデルID/モデル名は任意です。主キーの型は数字とします。

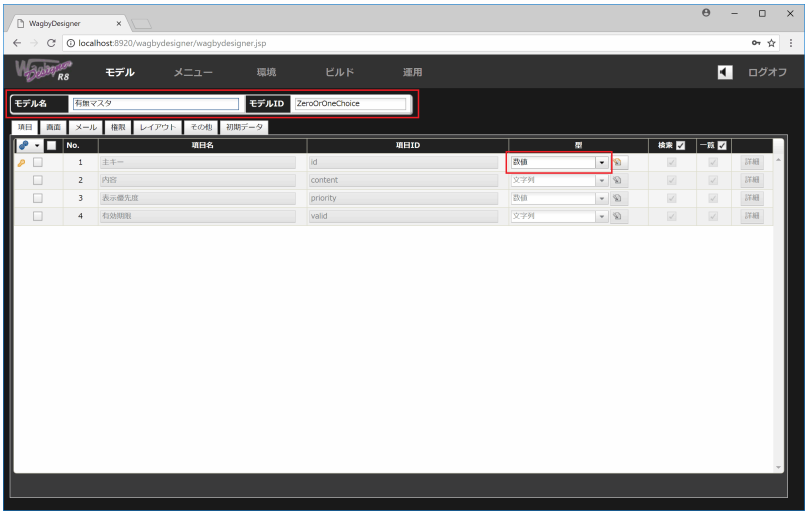


図 有無状態を管理する選択肢モデル

ここでは ″0″ を ″無″ とし、″1″ を ″有″ とします。

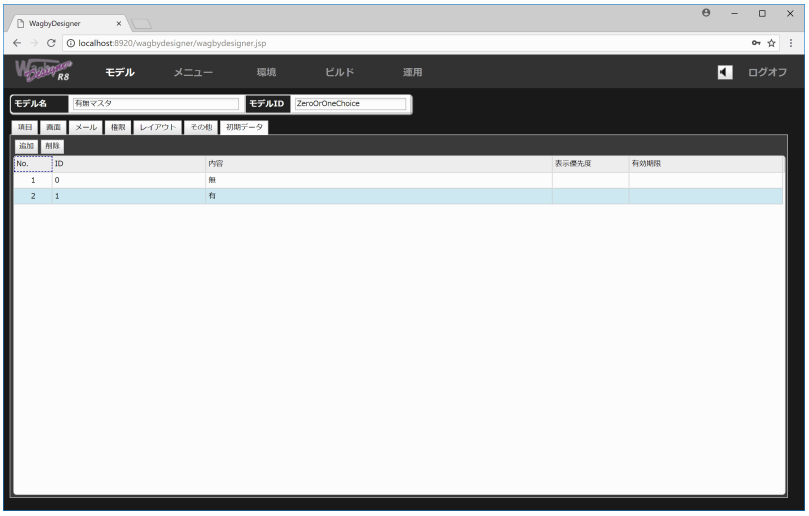


図 有無状態に対応したID

顧客モデルで利用する

顧客モデルに「挨拶状」という項目を用意します。この値が選択された顧客へは挨拶状を送付するという運用を想定します。

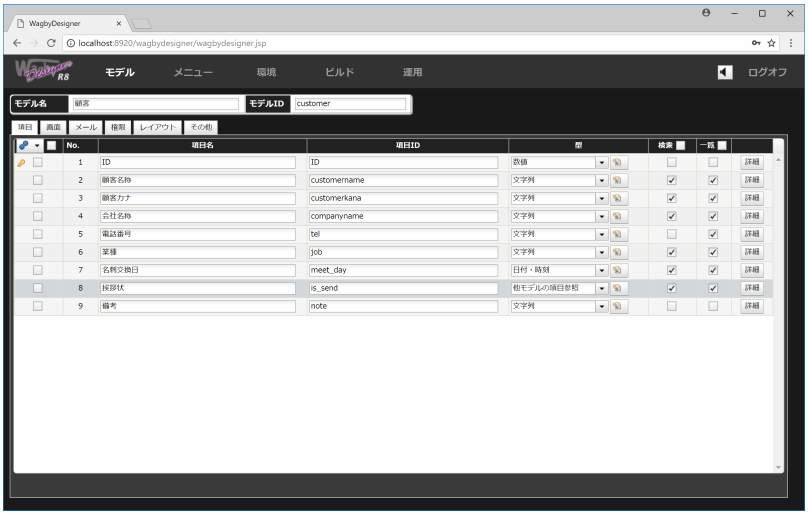


図 顧客モデル

この項目はラジオボタンとします。この設定により、生成されるテーブルには数値を管理するカラムが1つ用意されます。

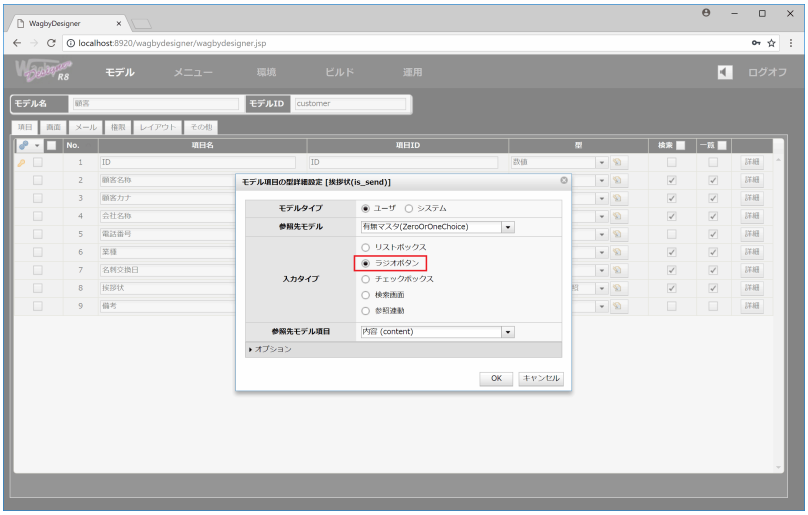


図 挨拶状項目の型の設定

入力制御 > モデル参照の設定で「ラジオボタンをチェックボックスで表示する」を有効にします。あわせて「チェックボックスの値」で、選択状態の値を指定します。この例では〝有〟は〝1〟ですので、1 を指定します。

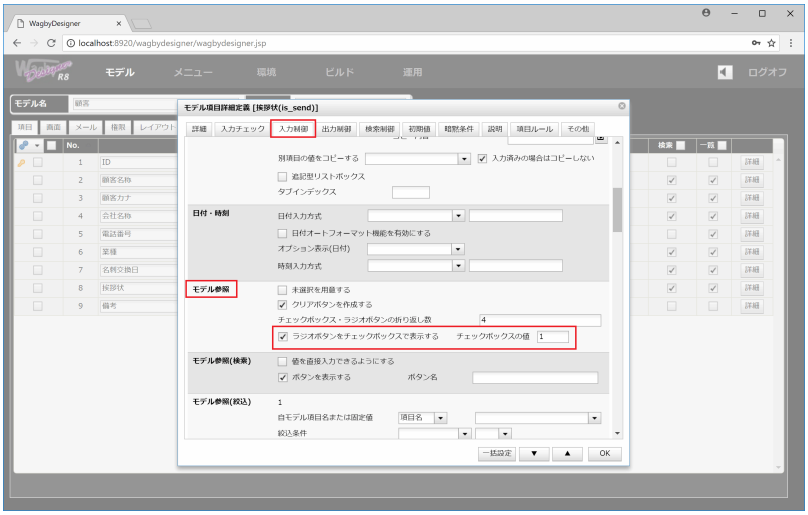


図 ラジオボタンをチェックボックスで表示する

(47)

初期値の設定

ここまでの設定で、画面上で項目を選択（チェック）するとサーバに値が送信されるようになりますが、Webアプリケーションでは送信される値は常に文字列となっています。Wagbyでは通常、型の変換は適切に行われるため開発者が意識する必要はありません。しかし今回のケースは「内部的にはラジオボタンだが、見た目はチェックボックス」という仕組みであるため、明示的な型変換処理を設定する必要があります。

具体的には、登録時と更新時の初期値に次の式を設定します。

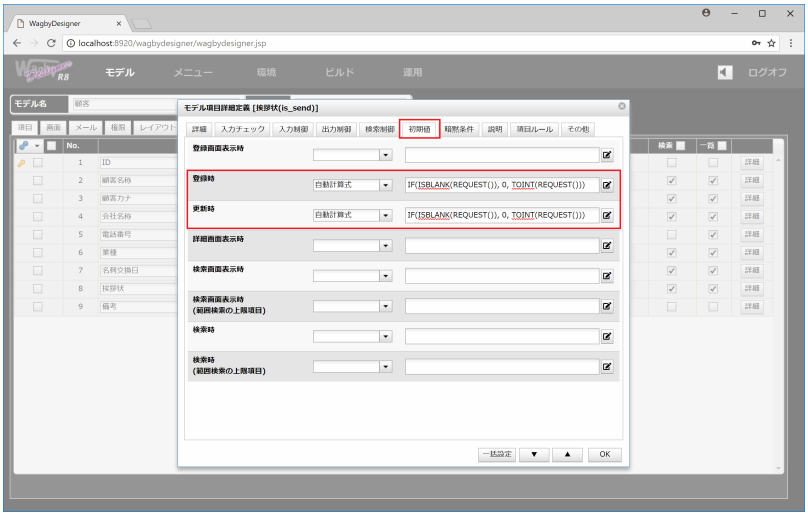


図 初期値の設定

```
IF(ISBLANK(REQUEST()), 0, TOINT(REQUEST()))
```

REQUEST関数は入力パラメータを意味します。戻り値は文字列型です。これに TOINT 関数を通すことで数値型に変換します。ISBLANK関数は値が空かどうかを判定します。

チェックボックスを選択しなかった場合、サーバに値は送信されません。つまり「値が送信されないか、"1" という文字列が送信されるか」になります。この式では、値が送信されなかった場合は 0 を、値が送信された場合は文字列の "1" を数値に変換することで、適切な値に変換しています。

実行例

(47) この設定を行うと、クリアボタンは用意されません。図では「クリアボタンを作成する」が有効になっていますが、この設定は無視されます。

挨拶状項目は一つのチェックボックスが用意されています。

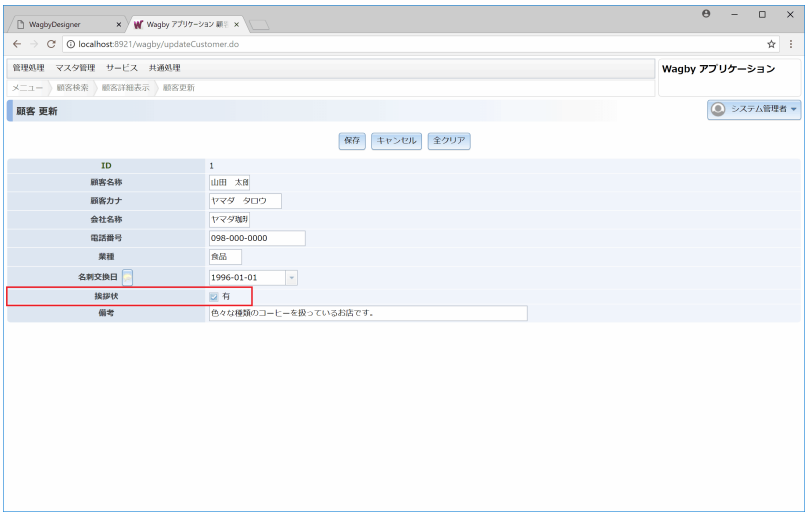


図 挨拶状項目を含んだ顧客更新画面

ストアモデルを参照する

有無モデルを準備する

Wagbyが提供する選択肢モデルという形式ではなく、独自に（有無状態を管理する）モデルを用意する例を説明します。モデルID/モデル名は任意です。ここでは主キーの型は文字列型とします。

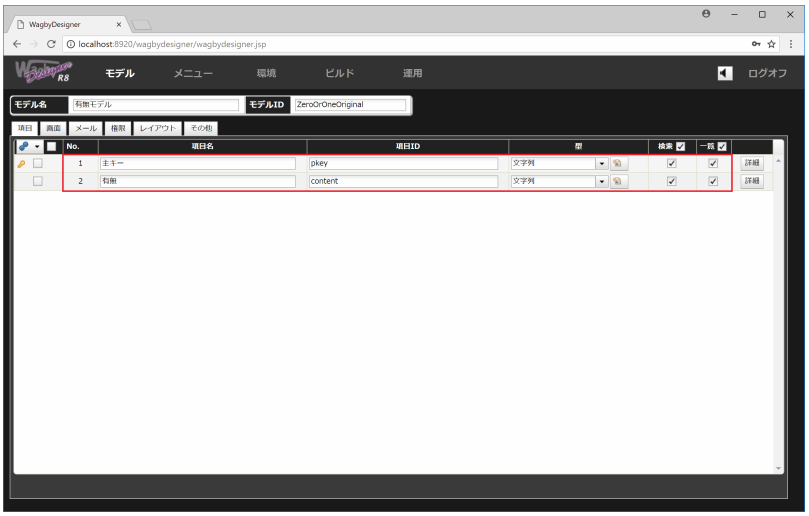


図 独自に用意した有無状態を管理するモデル

顧客モデルの挨拶状項目の参照先モデルを、上で用意したモデルとします。

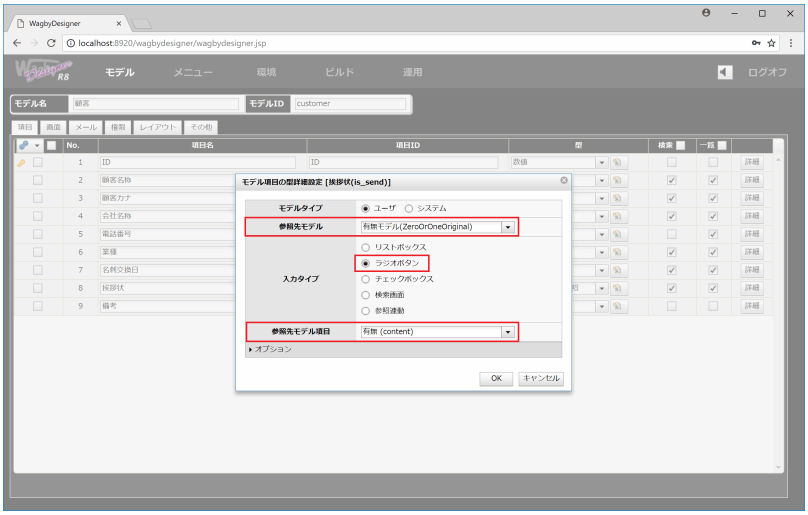


図 挨拶状項目の参照先モデルの設定

入力制御 〉 モデル参照の設定で「ラジオボタンをチェックボックスで表示する」を有効にします。あわせて「チェックボックスの値」で、選択状態の値を指定します。今回も ″有″ を ″1″ として扱うようにします。

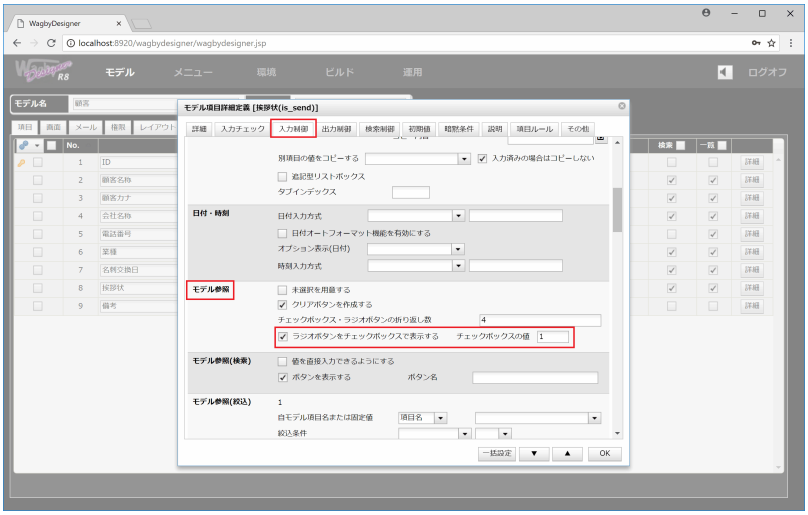


図 ラジオボタンをチェックボックスで表示する

初期値の式を設定します。参照先モデルの主キーが文字列型なので、文字列型の ″0″ と ″1″ が格納されるようにします。

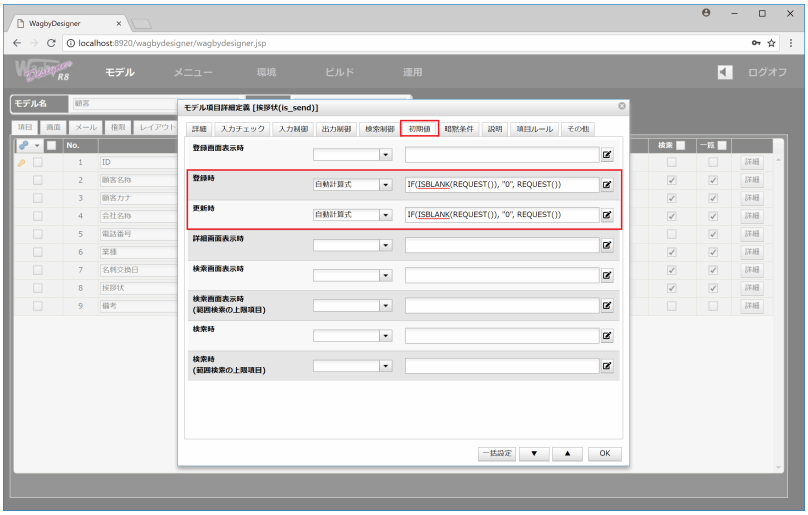


図 初期値の設定

```
IF(ISBLANK(REQUEST()), "0", REQUEST())
```

実行例

有無モデルの値を事前に用意しておきます。

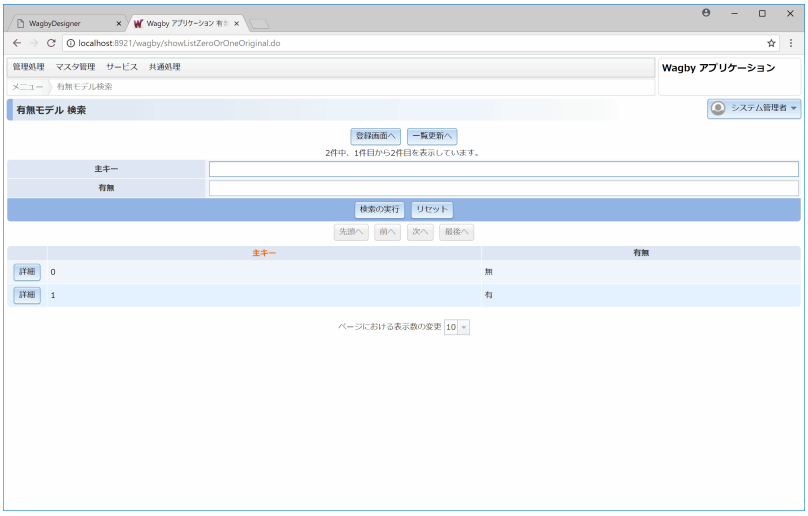


図 有無モデルの値を設定しておく

挨拶状項目はチェックボックスとして操作できます。

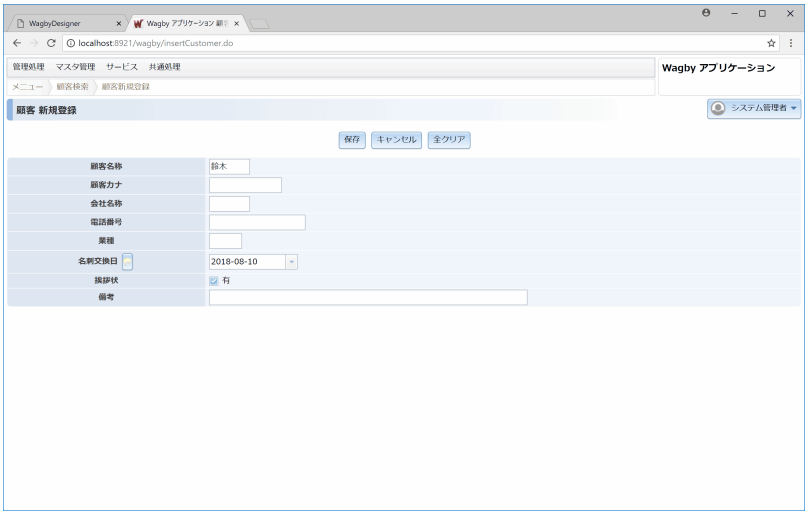


図 挨拶状項目を含んだ顧客登録画面

変更履歴と併用する場合の注意点 (8.3.3)

繰り返しコンテナでデータの変更履歴を管理する機能と併用する場合、注意が必要です。

上の例ではチェック「なし」の場合、入力値が空白になります。しかし既存データには ″無″ という文字が設定されているため、″無が削除されました″ という変更履歴が残ってしまいます。

対策

customize/resources/changelog_ja.properties.UTF8 を用意します。次の書式で対象モデルおよび項目を指定します。

```
<モデルID>.<項目>=replaceblank:無
```

例えば model1 の item1 項目であれば model1.item1=replaceblank:無 とします。″無″ の文字は図の「内容」部分に設定した文字列としてください。

(48)

参照元のトレース

例と定義方法

例

モデル参照を行っている項目については、図 のように「参照元トレースリンク」を表示することができます。これは参照先モデルの詳細画面へジャンプするためのリンクです。

(48) ビルド後のアプリケーション wagbyapp の WEB-INF/classes フォルダに changelog_ja.properties が生成されます。

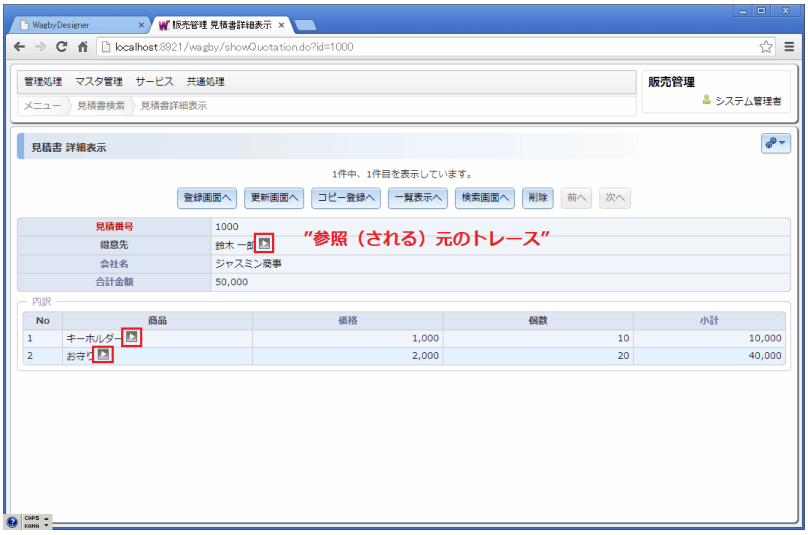


図 参照元トレースリンク

図は、見積書モデルの「商品」から参照している商品モデルへの移動を行った例です。なお、「パンくず」を使って、元の画面に戻ることができます。

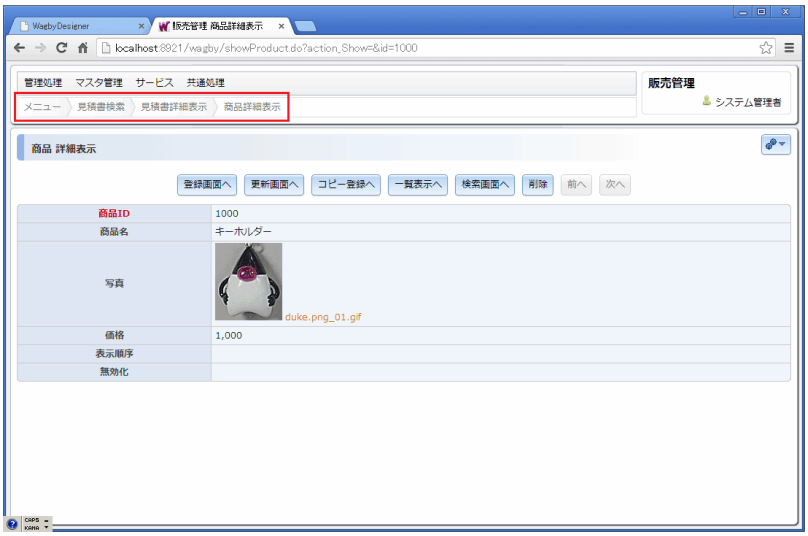


図 画面移動後のパンくず情報

「パンくず」は、画面の移動履歴を残した情報を意味します。

定義方法

「他モデルの項目参照」を設定した項目定義で、モデル項目の型詳細設定ダイアログを開きます。オプションの「参照元トレースリンクを表示する」を有効にします。（この設定は標準で有効になっています。）

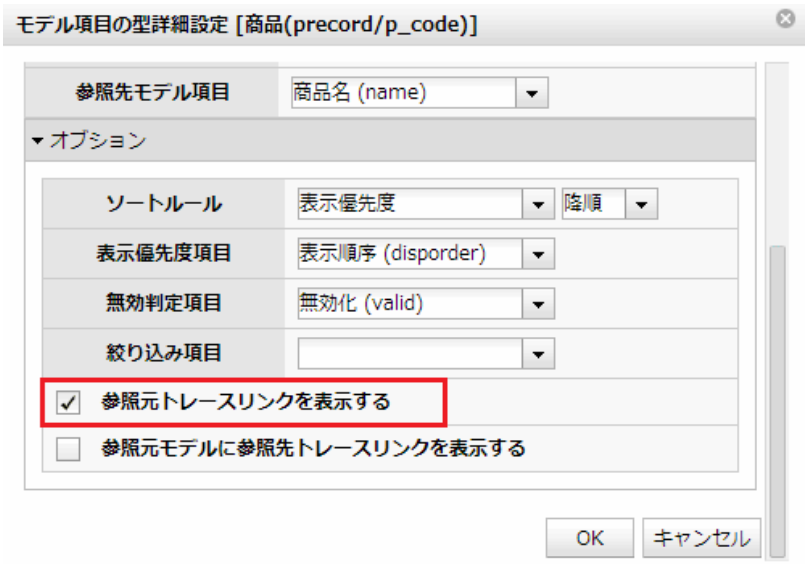


図 参照元トレースリンク表示の設定

この設定を解除すると、参照元トレースリンクは非表示になります。

権限管理との関係

参照元トレースリンクは権限管理情報を考慮します。権限がないモデルへのリンクは自動的に非表示となります。

アカウントとの関係

Wagby が標準で備えているアカウント情報(juserモデル)への参照元トレースリンクを用意することはできません。これはセキュリティ的な観点から制限しています。

参照連動との関係

参照連動項目へのリンクにも対応しています。

具体的には「項目 x1 がモデル Y の項目 y1を参照」し、また「y1は参照連動で、モデル Z を参照」している関係において、項目 x1 にリンクが用意されます。

仕様・制約

- 登録・更新の確認画面では、参照元トレースリンクは無効になります。（確認画面で参照元トレースリンクを辿られると、入力内容が破棄されてしまうため。）

参照先のトレース

例と定義方法

例

「商品」モデルを参照する「見積書」モデルを用意します。本機能を有効にしたとき、商品モデルの詳細画面に、自分を参照している見積書モデルの表示欄（参照先トレースリンク）が用意されます。

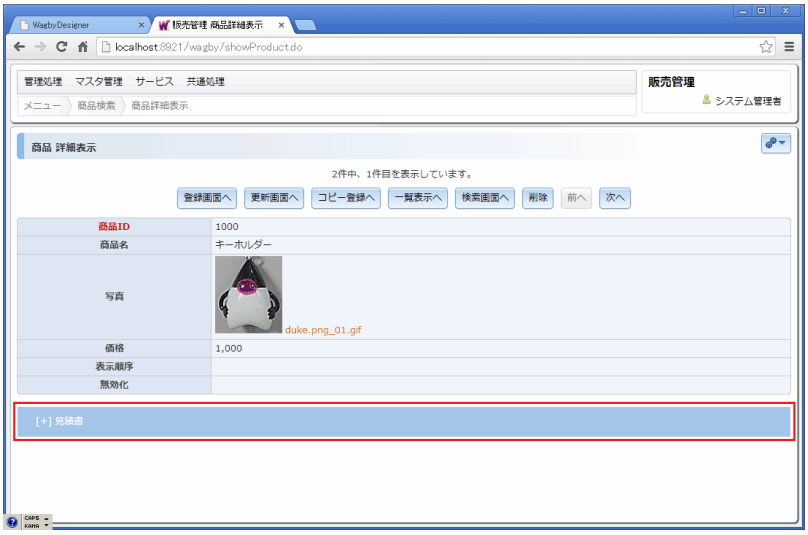


図 参照先トレースリンク

図の〝[+]〟部をクリックすると、一覧表示形式で展開されます。

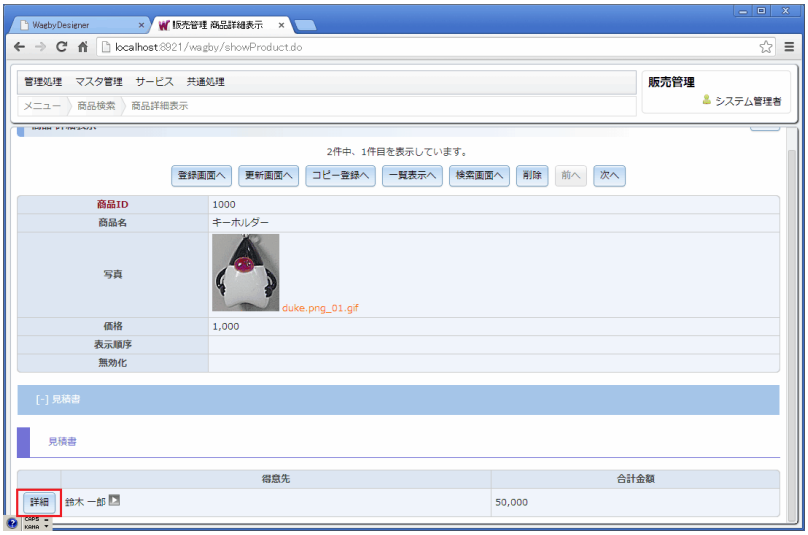


図 参照先トレースリンクの内容

図の「詳細」ボタンを押下すると、この商品モデルを参照している見積書へ遷移することができます。

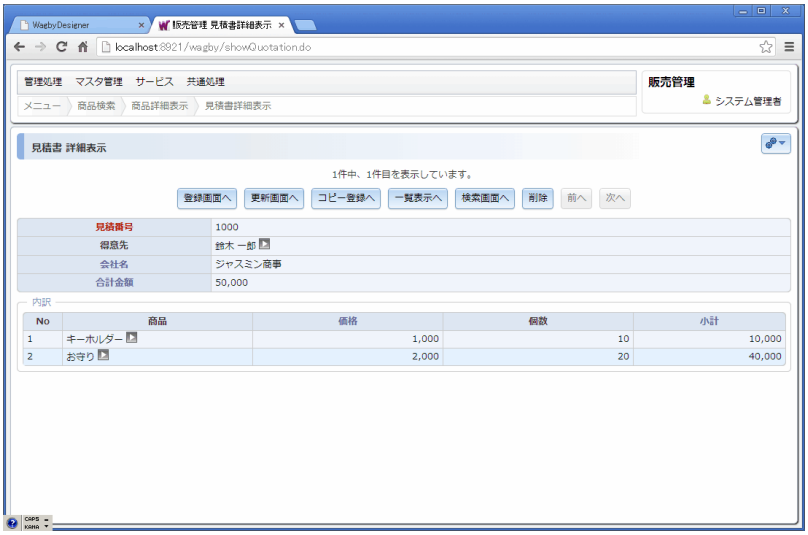


図 このデータを参照している見積書

定義方法

「他モデルの項目参照」を設定した項目定義で、モデル項目の型詳細設定ダイアログを開きます。オプションの「参照先トレースリンクを表示する」を有効にします。（この設定は標準で無効になっています。）

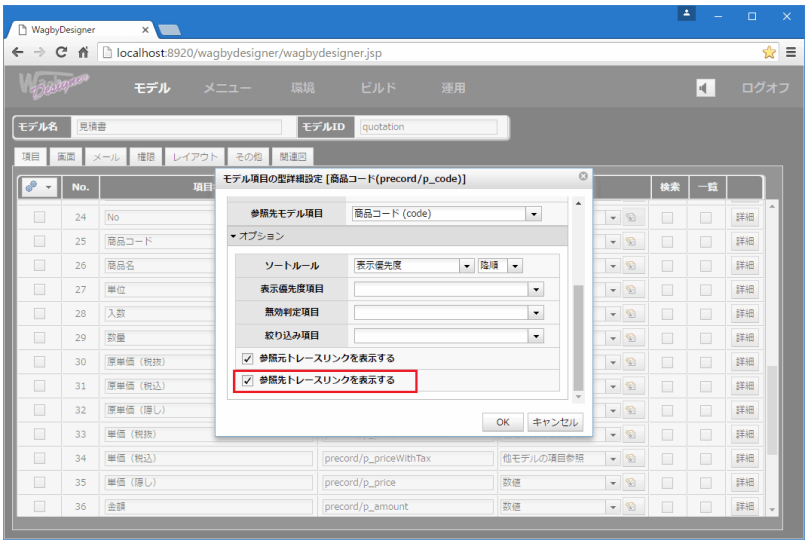


図 参照先トレースリンク表示の設定

仕様・制約

- 参照元モデルに関する一覧表示の内容は、そのモデルの一覧表示設定に準拠します。ただし、参照キー項目は（冗長となるため）表示されません。
- 参照元モデルが存在しない（0 件）のときは、リンクをクリックしても表示されません。
- 外部キー子モデル一覧表示と併用する場合、表示順序は（1）外部キー子モデル（2）参照元モデル となります。
- 複数の項目が同じモデルを参照していた場合、それぞれに「参照先トレースリンクを表示する」を設定することはできません。あるモデルに対する参照先トレースリンクの設定は、常に一つとしてください。(*1)

(49)

参照モデル同士の関係性で選択肢を絞り込む

リストボックスで絞り込み、リストボックスで選択する

例

例として、社員モデルの社員名を参照するケースを扱います。ここで参照先の社員数が多いとリストボックスから探すことが難しくなってしまいます。　そこで最初に「部署」を選択し、その部署に所属する「社員」を選択するといった絞り込み機能を実現します。

「営業担当部署」によって「営業担当者」の選択肢が変わります。図は営業担当部署が未選択の状態です。

(49) 1 上の例で「見積書」モデルに「商品1」「商品2」という項目があり、それぞれ同じ「商品」モデルを参照しているとき、参照先トレースリンクの設定は「商品1」または「商品2」のいずれかとしてください。

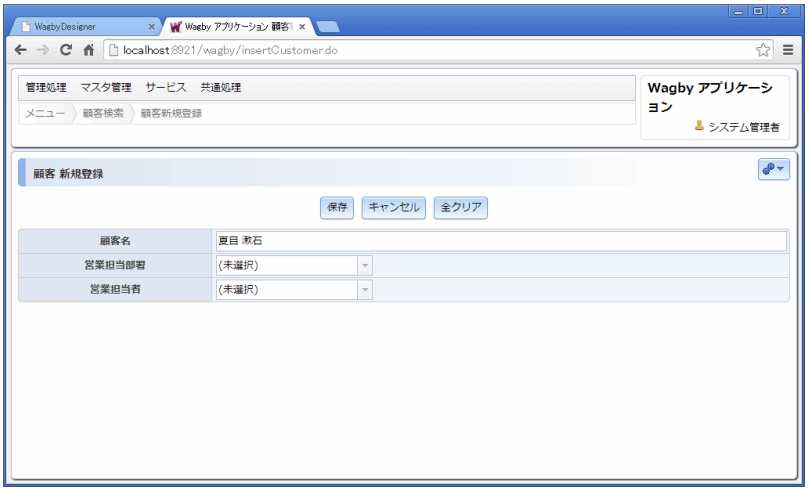


図 営業担当部署を決定しないと営業担当者の選択肢も表示されない

担当部署を"営業一部"とすると、同部に所属する担当者が選択肢として表示されます。

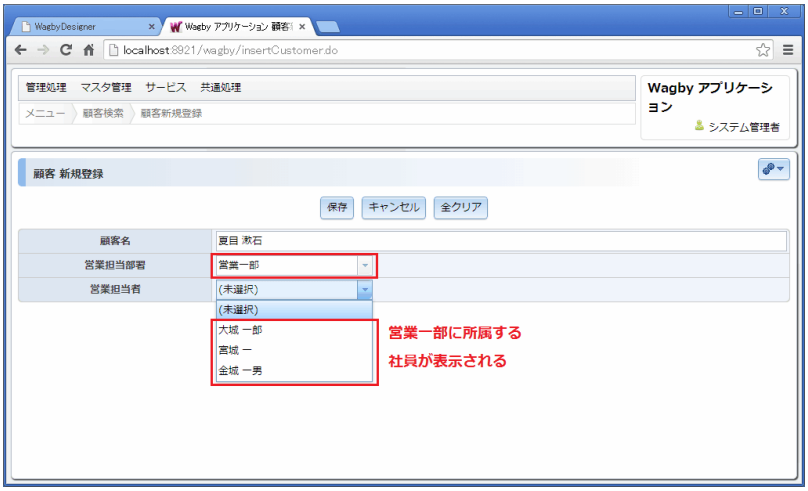


図 営業一部を選択する

同じく担当部署を"営業二部"とすると、同部に所属する担当者が選択肢として表示されます。

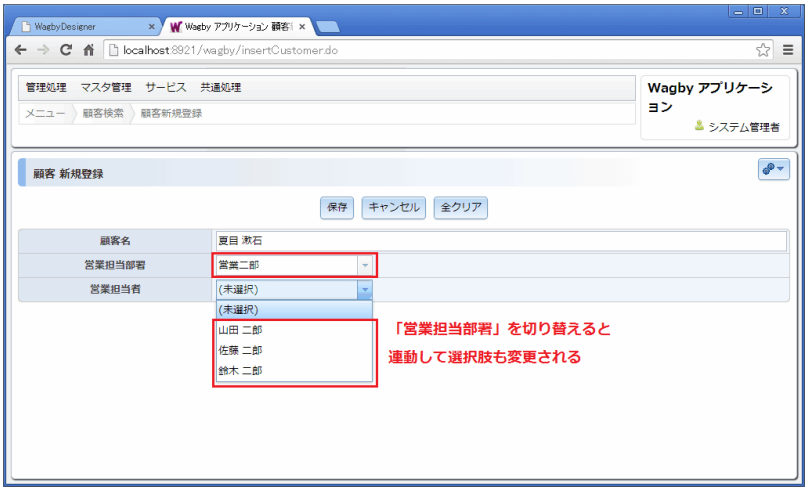


図 営業二部を選択する

検索条件に用いる

両項目を検索条件に用いることもできます。

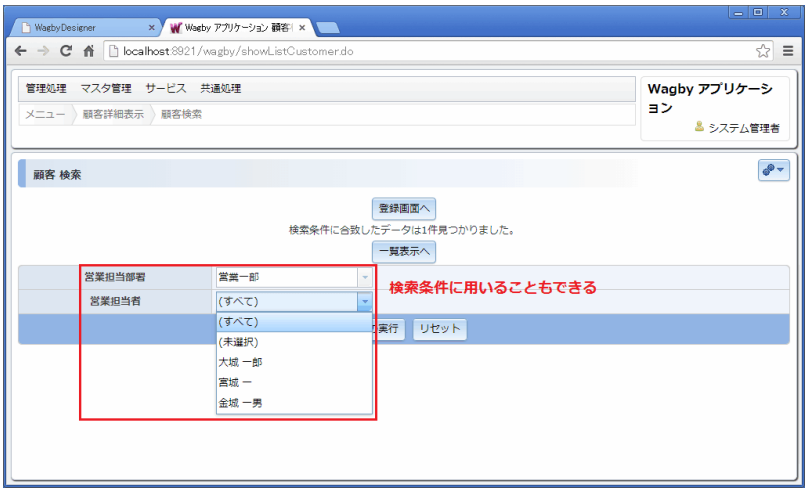


図 営業担当部署と営業担当者項目を検索条件に用いる

定義方法

3つのモデル「部署」「社員」「顧客」を用意します。

部署モデル

「部署」モデルは主キー「部署ID」と、「部署名」を用意します。

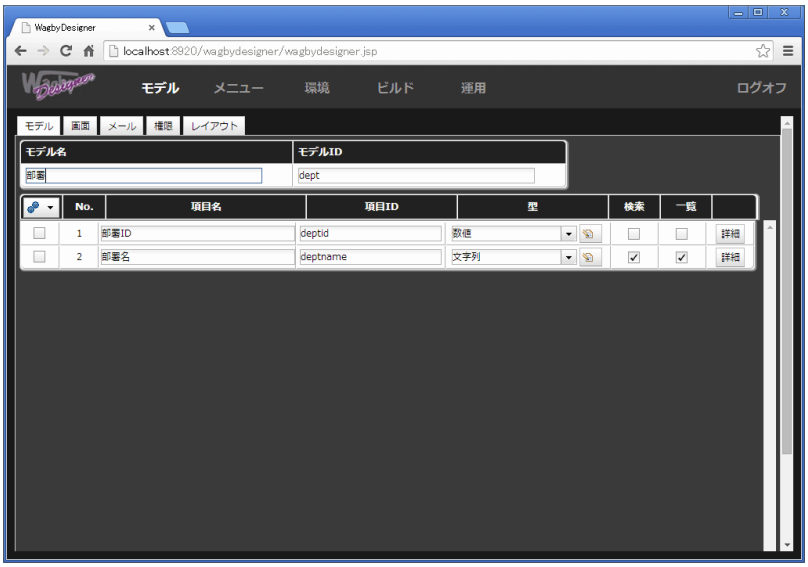


図 部署モデル

社員モデル

「社員」モデルは「部署」モデルを参照します。

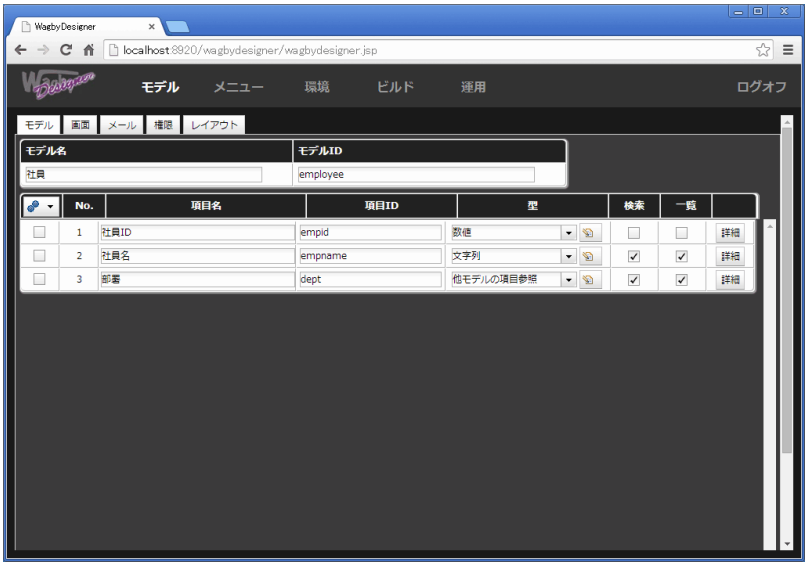


図 社員モデルの定義

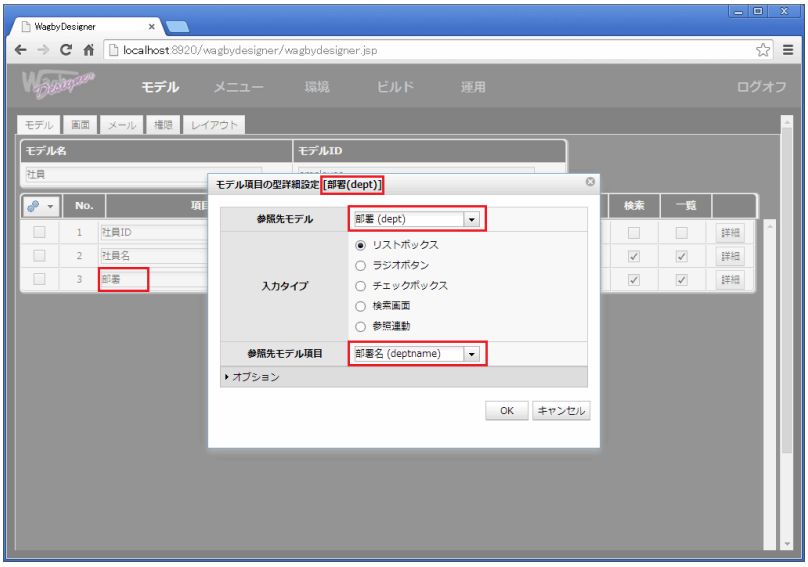


図 社員モデルにおける部署項目の関連設定

顧客モデル

「顧客」モデルは「営業担当部署」として部署モデルを、「営業担当者」として社員モデルをそれぞれ参照します。

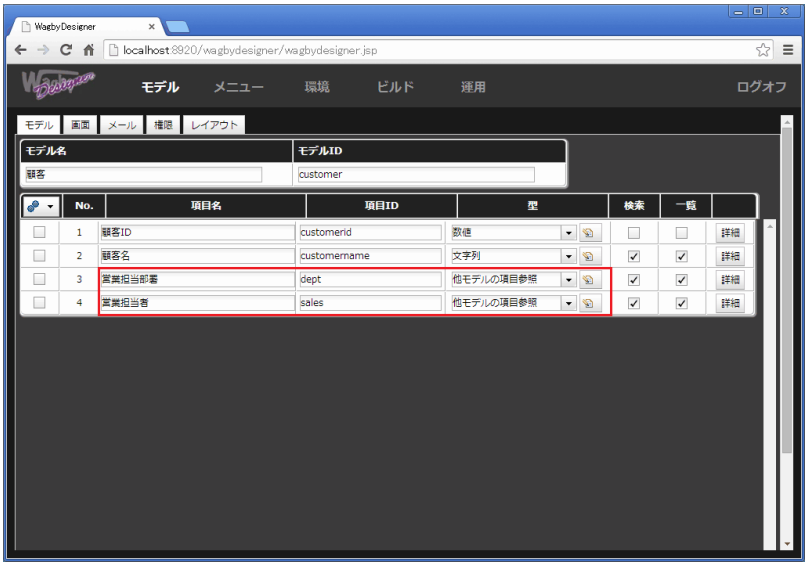


図 顧客モデルの定義

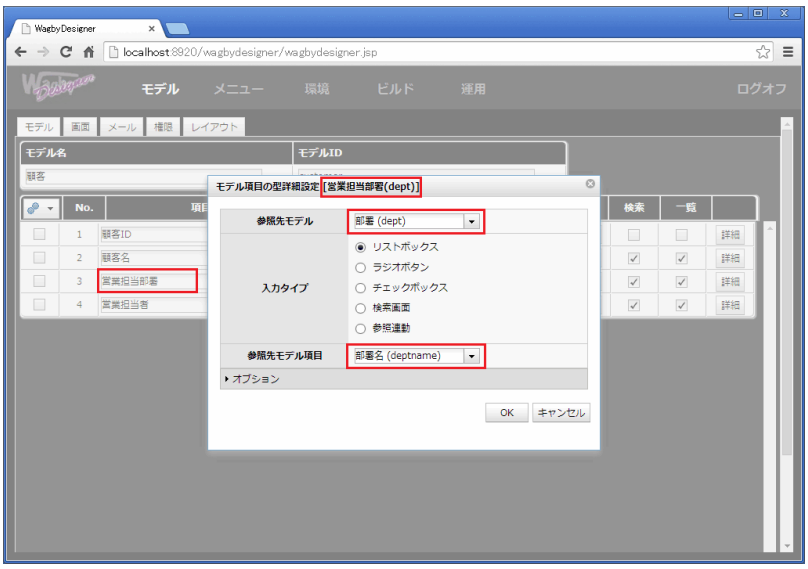


図 顧客モデルにおける営業担当部署の関連設定

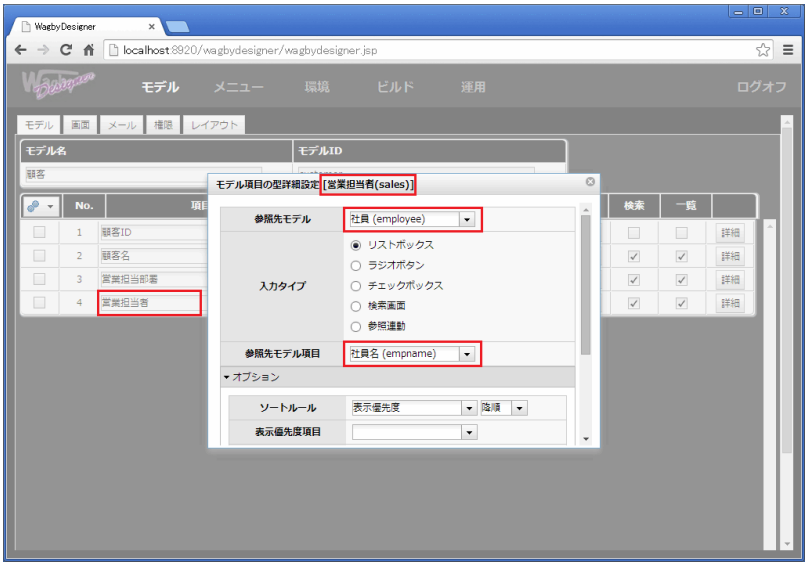


図 顧客モデルにおける営業担当者の関連設定(1)

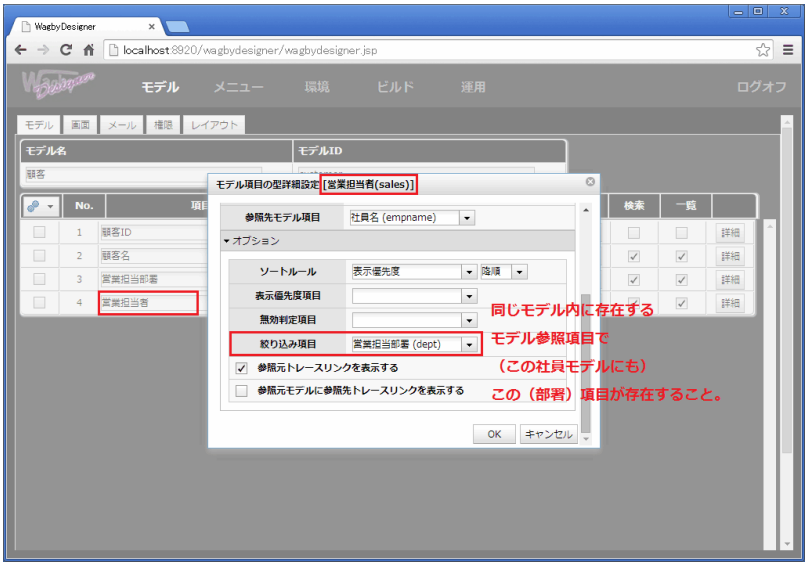


図 顧客モデルにおける営業担当者の関連設定(2)

設定のポイントは図における「オプション>絞り込み項目」です。

- このモデルに用意されたモデル参照項目であること。（数字、文字列、日付型は利用できません。モデル参照です。）
- 参照先モデル（ここでは社員モデル）にも含まれる項目であること。

この設定を図示すると次のようになります。

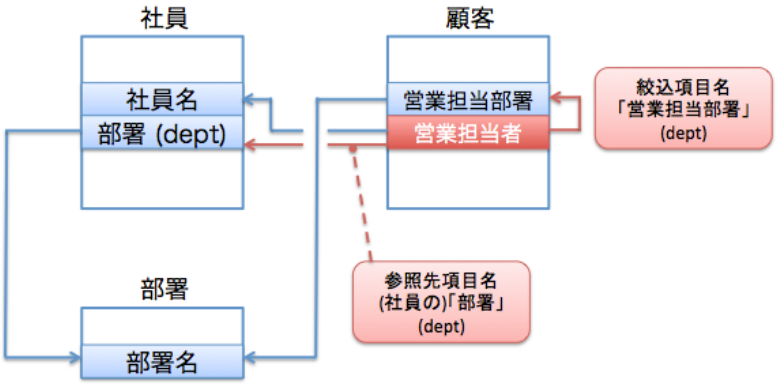


図 設定イメージ

ラジオボタンで絞り込み、リストボックスで選択する

営業担当部署項目をラジオボタンとし、営業担当者項目をリストボックスとした例を図に示します。

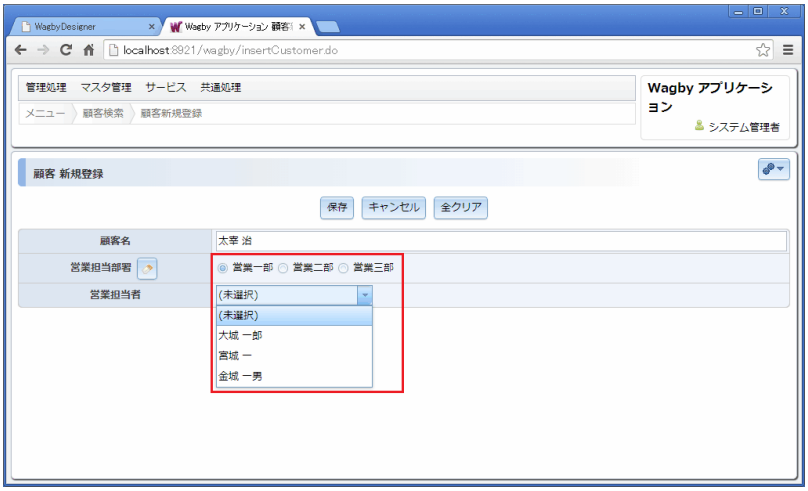


図 ラジオボタン-リストボックスの組み合わせ

チェックボックスで絞り込み、リストボックスで選択する

営業担当部署項目をチェックボックスとし、営業担当者項目をリストボックスとした例を図に示します。

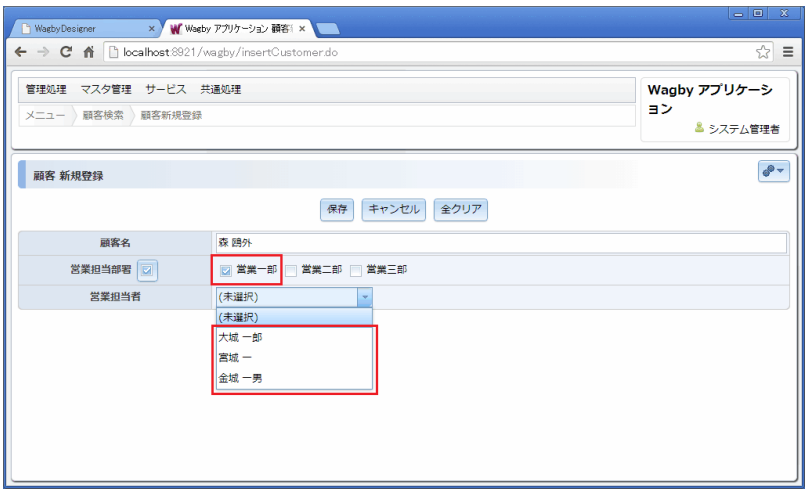


図 チェックボックス-リストボックスの組み合わせ

チェックボックスを複数選択すると、絞り込み項目も増えます。

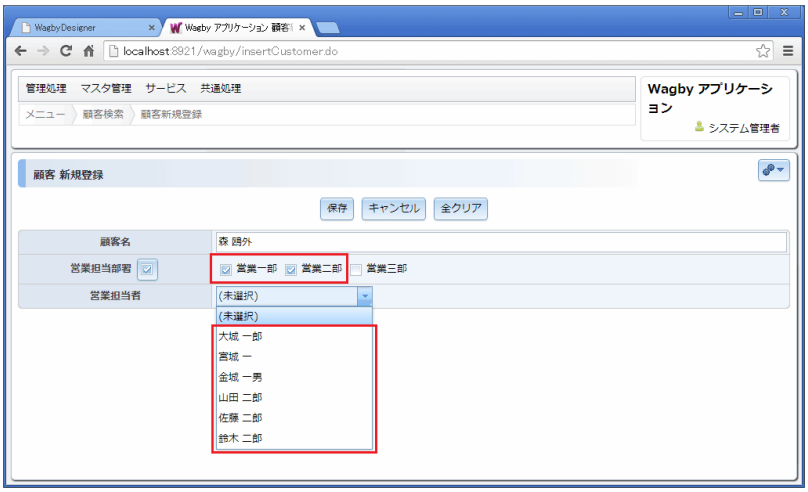


図 チェックボックス-リストボックスの組み合わせ(2)

リストボックスで絞り込み、チェックボックスで選択する

営業担当部署項目をリストボックスとし、営業担当者項目をチェックボックスとした例を図に示します。リストボックスが未選択の場合、営業担当者は表示されません。

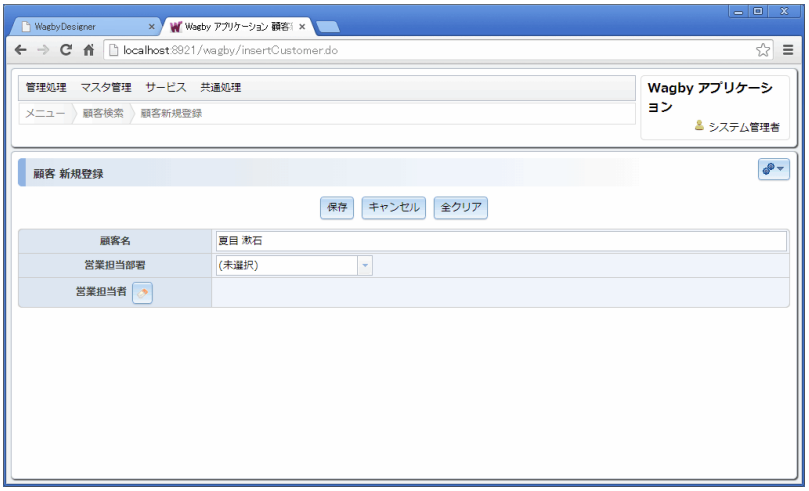


図 リストボックス-チェックボックスの組み合わせ(1)

営業担当部署を選択すると、チェックボックスの選択肢が表示されます。

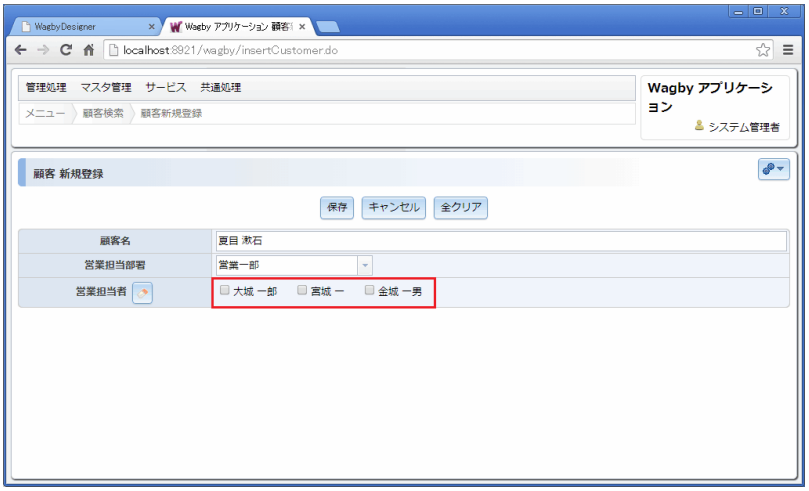


図 リストボックス-チェックボックスの組み合わせ(2)

ラジオボタンで絞り込み、チェックボックスで選択する

営業担当部署項目をラジオボタンとし、営業担当者項目をチェックボックスとした例を図に示します。

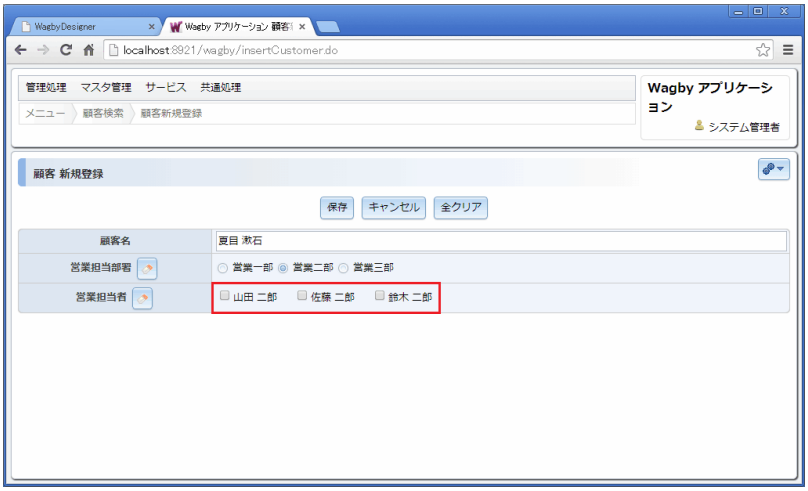


図 ラジオボタン-チェックボックスの組み合わせ

チェックボックスで絞り込み、チェックボックスで選択する

営業担当部署項目と営業担当者項目のいずれもチェックボックスとした例を図に示します。

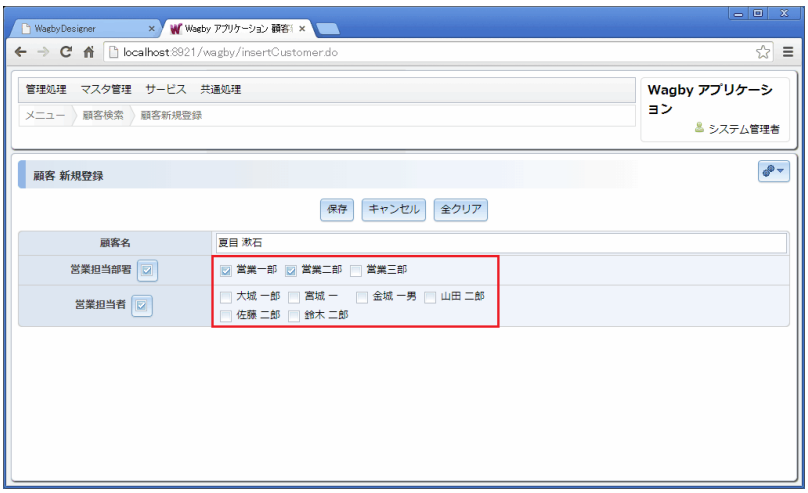


図 チェックボックス-チェックボックスの組み合わせ

仕様・制約

動作の仕組み

この機能は、絞り込み元データをいったんすべてメモリに取り込み、Javaプログラムで条件判定を行います。[パフォーマンスについて...]

指定できないパターン

次の場合は、絞り込み項目に指定することはできません。

- モデル参照ではない項目（数字型、文字列型、日付型など）
- モデル参照（検索画面）
- 指定した項目の参照先モデルの主キーが複合キーの場合

この場合は、もう一つの絞込設定である「他項目の値で選択肢を絞り込む」をご利用ください。

絞り込み元の値が変わった場合の動作

絞り込み元の値を変えた場合（上記例では「営業担当部署」の値を変更した場合）、「営業担当」として選択されていた値はクリアされます。

サジェスト項目

サジェスト項目に絞り込みは設定できません。（設定しても機能しません。）

二つ以上の項目間での絞り込み

本機能は二つの項目間の関係性に着目した絞り込み動作を行います。 これを応用することで、二つ以上の項目間での絞り込みを定義することができます。

例えば「組織」「所属」「氏名」という項目について、「組織」と「所属」および「所属」と「氏名」に絞り込み関係を定義することができます。 このとき、組織に″営業部″を選択すると、所属に″第一営業部″″第二営業部″が絞込まれるといったことができます。 さらに″第一営業部″を選択すると、氏名に″田中″″佐藤″が絞込まれるように設定することができます。

このとき、組織が″本社″だが、所属が空白で、氏名が″斉藤″というデータを用意することはできないということにご注意ください。 この場合、組織を″営業部″としつつ、所属を未決定とした場合でも、氏名欄には ″斉藤″ が選択肢として表示されてしまいます。なぜなら、氏名は「組織と所属のペア」で絞込まれるのではなく、あくまでも「所属」で絞込まれるためです。所属が未決定の場合、組織が何であっても氏名欄には″斉藤″が選択肢に加わってしまいます。

これを避けるため、ダミーの所属をご用意ください。例えば上の例ですと、組織 ″本社″ に対する所属として ″(本社直轄)″ などをご用意いただくことで解決できます。

パフォーマンス

この機能は、絞り込み元のデータをいったんメモリにすべて読み込み、Javaプログラムで1件ずつ条件に合致するかどうかを判定します。そのため、絞り込み元データが大量の場合は不向きです。絞り込み元（上記の例では「営業担当部署」）は画面に表示できる程度の数（数個から十数個程度）で用いるようにしてください。

(50)

トラブルシューティング

一覧表示画面で絞込が動作しない

一覧表示画面に絞込設定したモデル参照項目を含める場合、絞り込み項目も一覧表示に含める必要があります。

一覧更新画面で絞込が動作しない !-- N20160216_160819 -->

1. 一覧更新画面に絞込設定したモデル参照項目を含める場合、絞り込み項目も一覧表示に含める必要があります。上の説明（「一覧表示画面で絞込が動作しない」）と同様です。
2. 絞り込み項目に「入力制御 > 共通 > 値変更時に画面を再描画する」を設定してください。画面再描画によって絞り込みが機能します。

他項目の値で選択肢を絞り込む

例と定義方法

設定イメージ

「当社担当部署」を指定すると「当社担当者」の選択肢が変わる例を示します。

(50) 例えば、絞り込み元はリストボックスだが数百件以上あり、これを画面上は隠し項目にするという設計では問題となります。この場合は「他項目の値で選択肢を絞り込む」方式をご利用ください。これは SQLで絞り込みを行うため、大量データを扱う場合は本方式よりも高速に動作します。

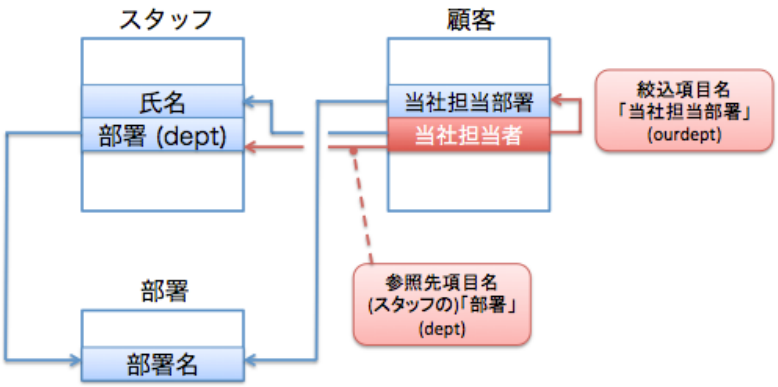


図 設定イメージ

例

図は担当部署を選択していない状態です。このとき、担当者はいずれも選択できません。

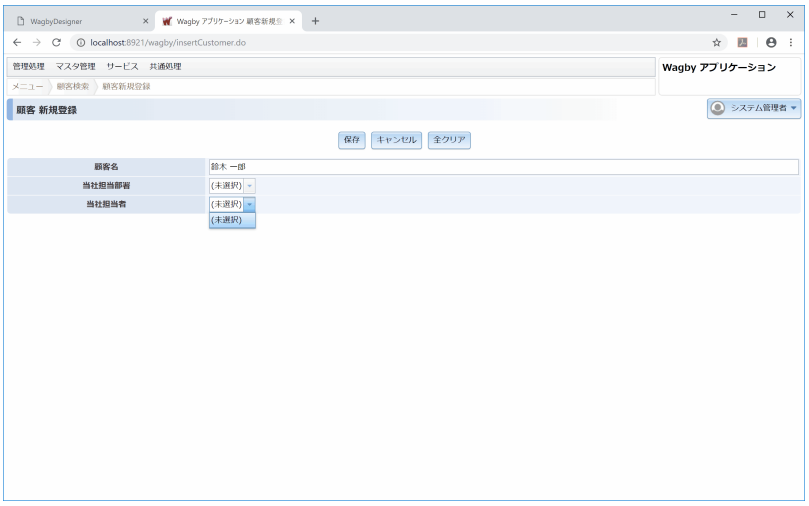


図 担当部署を決定しないと担当者の選択肢も表示されない

担当部署を総務部とすると、総務部に所属する担当者が選択肢として表示されます。

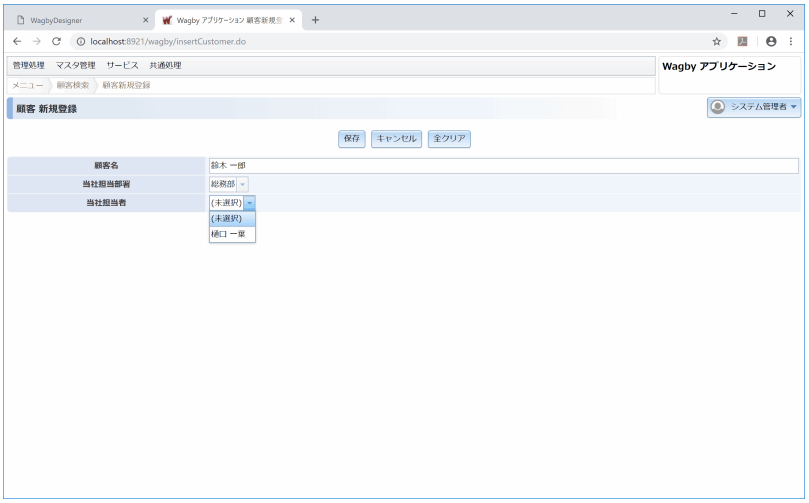


図 担当部署に総務部を選択する

同じく担当部署を営業部とすると、営業部に所属する担当者が選択肢として表示されます。

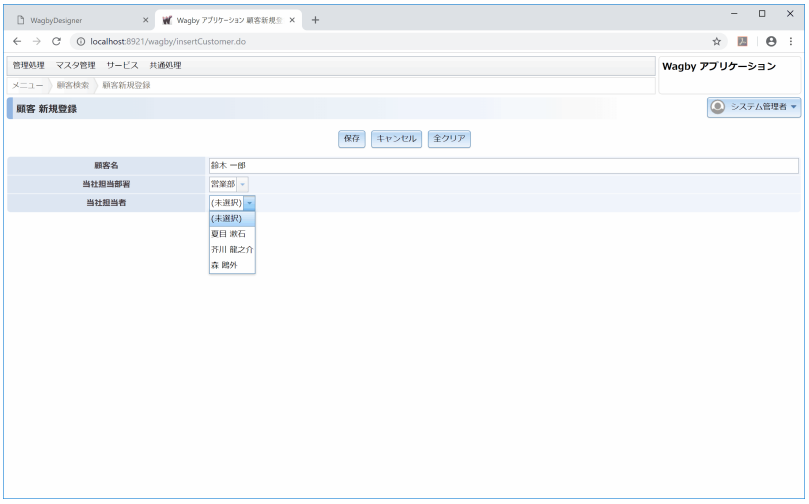


図 担当部署に営業部を選択する

定義方法

モデルは「スタッフ」「職種」「部署」「顧客」を用意しています。なお「職種」は選択肢モデルなので説明を割愛します。

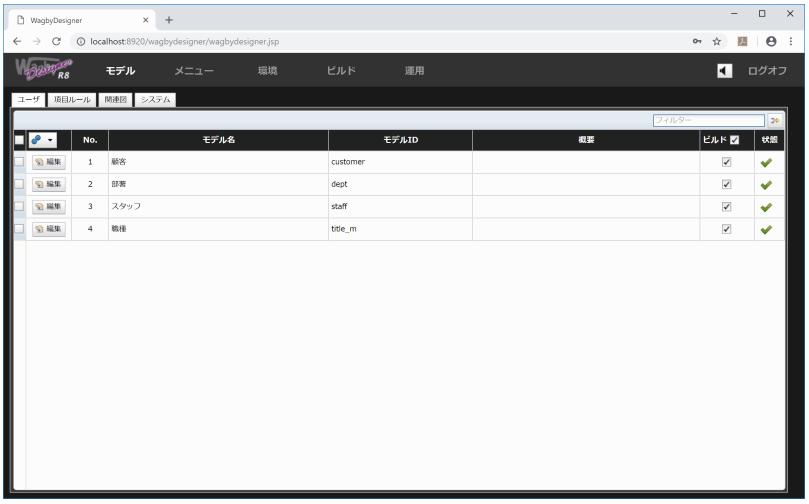


図 用意したモデル

「部署」モデルは主キー「部署ID」と、「部署名」を用意します。

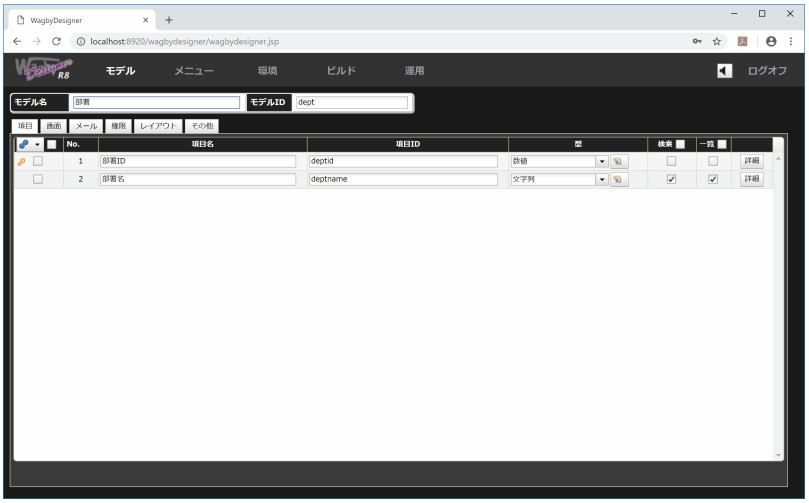


図 部署モデルの定義

「スタッフ」モデルは「部署」モデルを参照します。

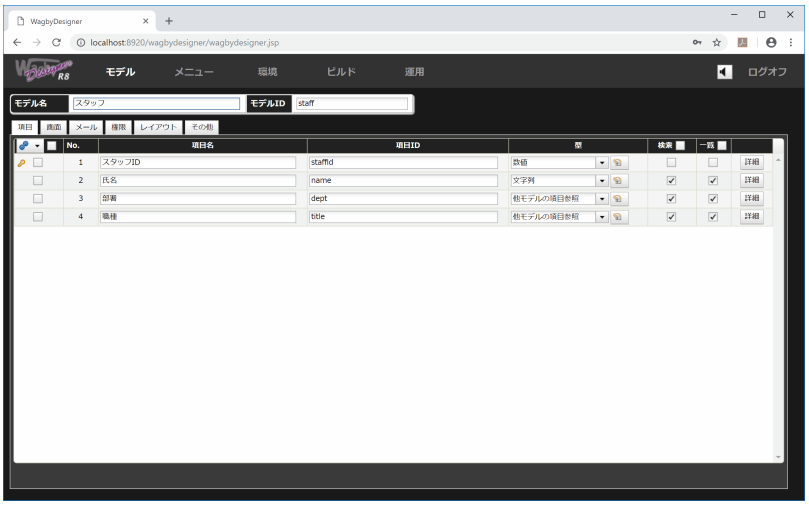


図 スタッフモデルの定義

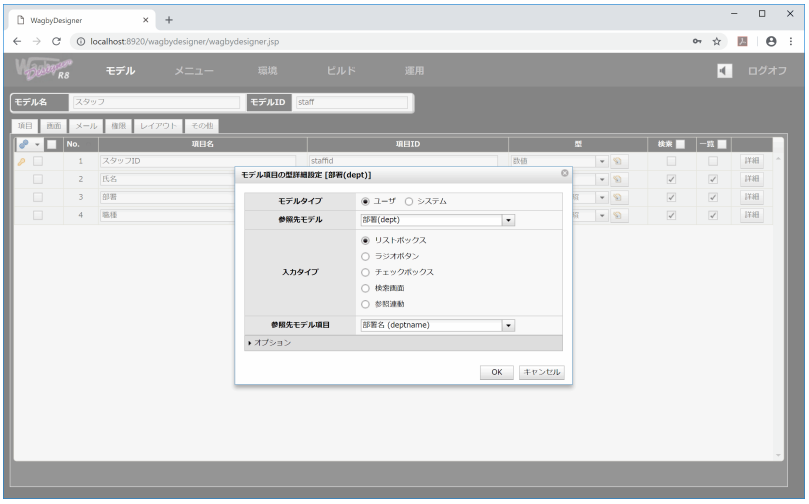


図 スタッフモデルの部署項目の定義

「顧客」モデルは「当社担当部署」として部署モデルを、「当社担当者」としてスタッフモデルをそれぞれ参照します。

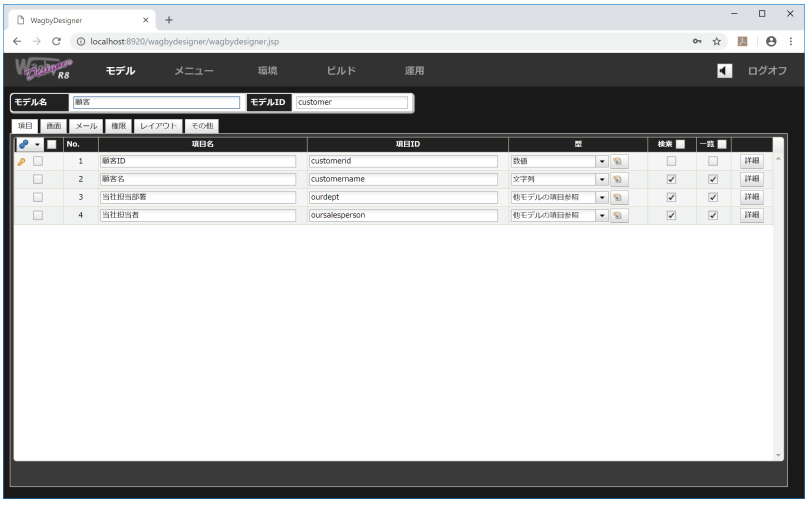


図 顧客モデルの定義

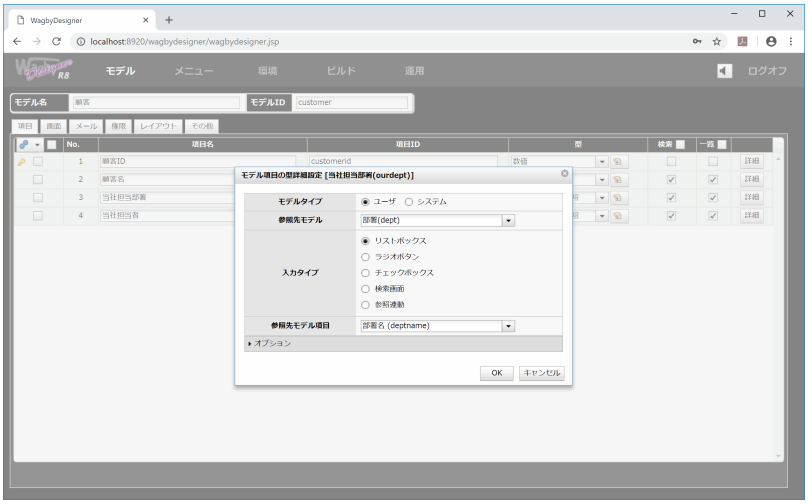


図 顧客モデルの部署項目の定義

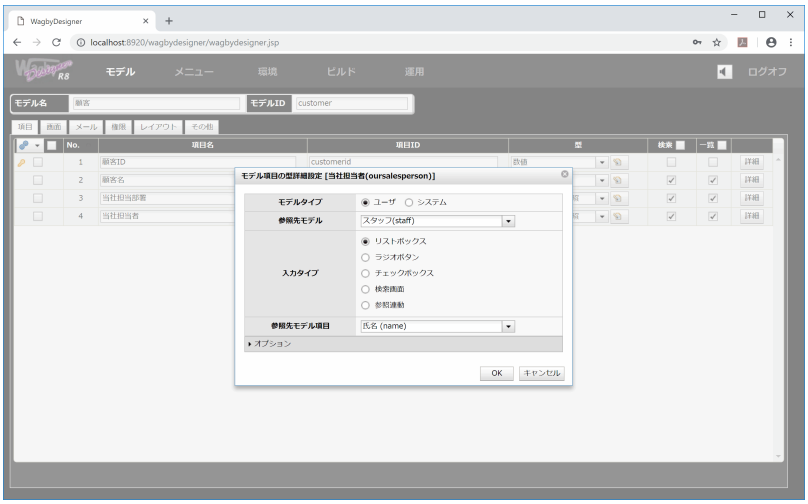


図 顧客モデルのスタッフ項目の定義

ここで絞り込みの設定を行います。「当社担当者」項目の詳細ダイアログから「入力制御>モデル参照(絞込)」を設定します。

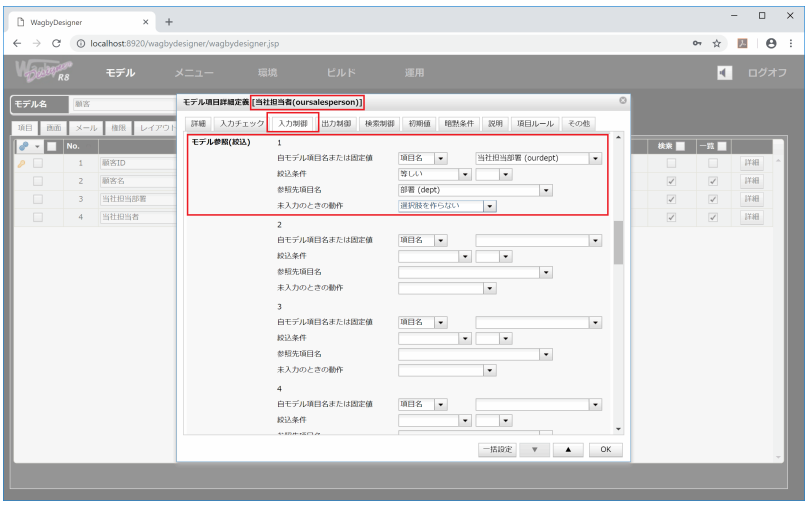


図 モデル参照(絞込)の設定

項目名

自モデル（ここでは顧客モデル）内の、どの項目が絞込の元かを指定します。

絞込条件

一致していたものを選択するため″等しい″を指定します。

参照先項目名

「当社担当者」項目はスタッフモデルを参照しています。スタッフモデル側の部署項目である″dept″を指定します。

未入力のときの動作

図にあるように、絞込元が未選択のときは選択肢を用意しないため、″選択肢を作らない″を指定します。

無効となった選択肢の選択状態を維持しない

標準では、絞り込み元の値を変えた場合（上記例では「当社担当部署」の値を変更した場合）「当社担当者」として選択されていた値は維持されます。（絞り込み元の値をクリアした場合も同様に、絞り込み先の選択肢は維持されます。）

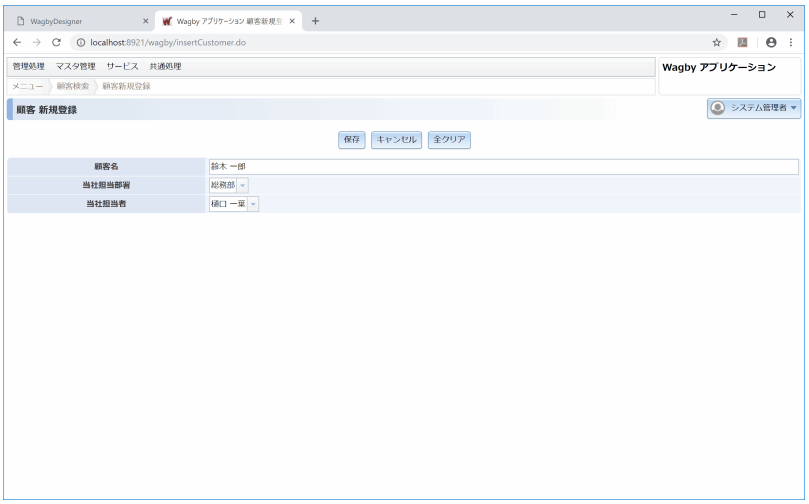


図 営業部担当者が選択済みの状態で、部署を変更する

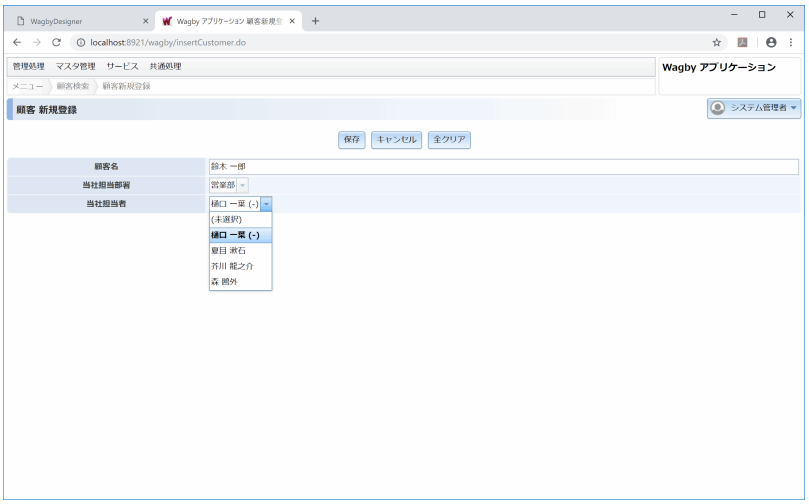


図 選択されていた担当者は維持されるが、無効化されたマークが付与される

この動作を変更することができます。「無効となった選択肢の選択状態を維持しない」を有効にしたとき、絞り込み元の値を変えると選択肢はクリアされます。（この機能はリストボックス-リストボックスの組み合わせでのみ動作します。）

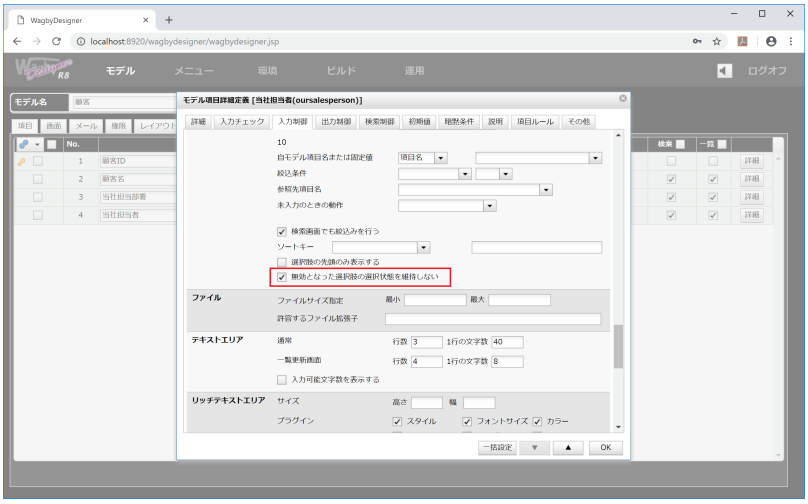


図 無効となった選択肢の選択状態を維持しない設定

この設定を行うと、絞り込み元の値を変えた場合、選択されていた候補は消去されます。

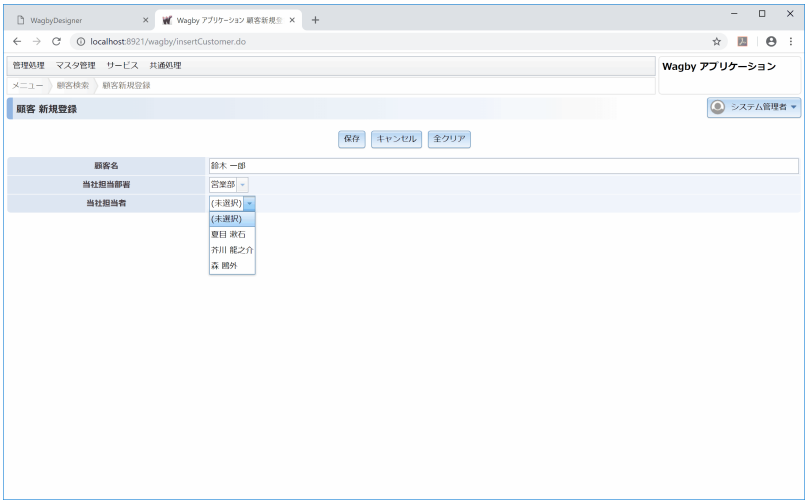


図 選択されていた担当者は維持されない

複数の条件で絞込む

複数の絞り込み条件を指定することができます。条件はすべて　AND　（「かつ」）で絞り込みが行われます。

設定イメージ

図に示す絞り込みの例を説明します。絞り込み条件を二つ用意します。

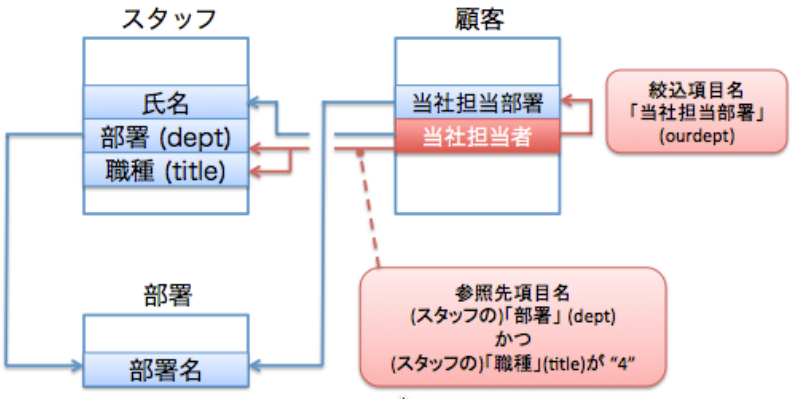


図 設定イメージ

例

選択肢モデル「職種」が次のように用意されているとします。"一般" という職種のIDは "4" です。

図 職種の情報

絞込条件に「職種が一般のみ」という条件を追加しました。図に比較すると選択肢が減っています。

図 営業部かつ一般職種のみが選択肢として用意される

定義方法

顧客モデルの当社担当者項目の絞込条件を追加します。 職種(title)のID値が"4"という固定条件を加えています。

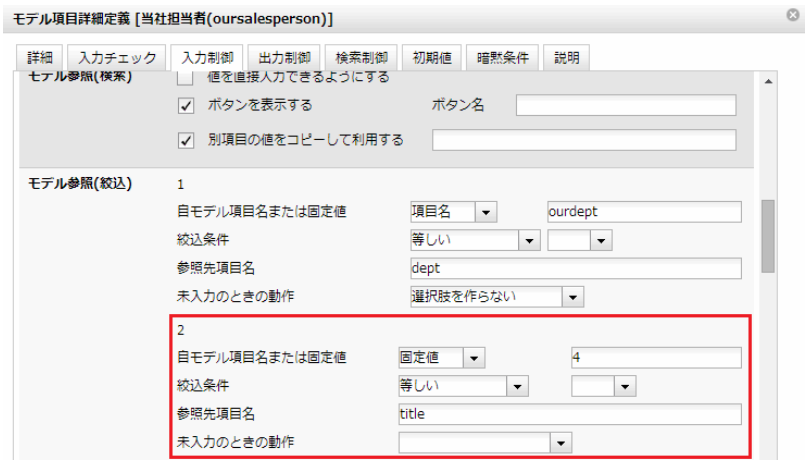


図 職種の条件を加える

繰り返しコンテナ内の項目で絞り込む

設定イメージ

絞り込みむための条件に、繰り返しコンテナ内の項目を指定することができます。

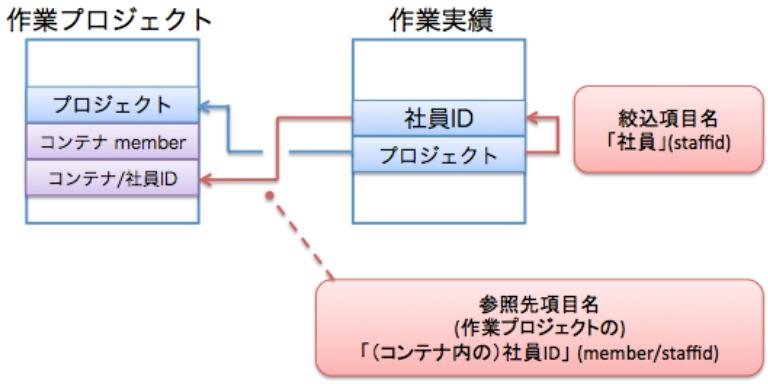


図 設定イメージ

例

4人の社員を登録しておきます。

図 社員データ

″プロジェクト1″ と ″プロジェクト2″ にそれぞれ社員を二名ずつ割り当てます。

図 プロジェクト1にユーザー1とユーザー2を割り当てる

図 プロジェクト2にユーザー3とユーザー4を割り当てる

作業実績を登録します。選択した社員に連動して、プロジェクトの選択肢が絞り込まれます。

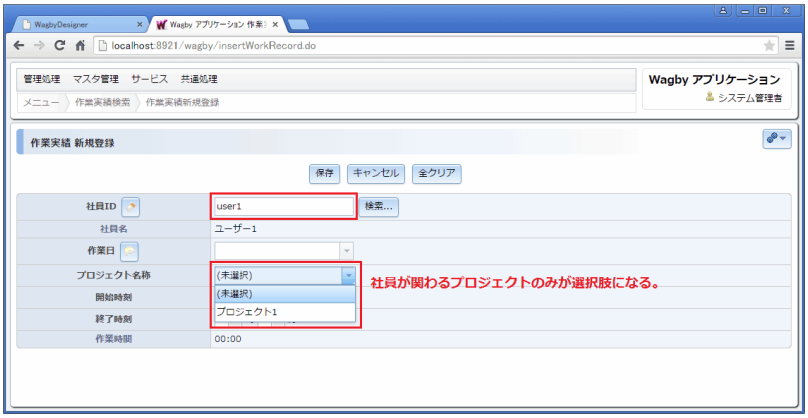


図 ユーザー1を選択した場合は、プロジェクト1が選択できるようになる

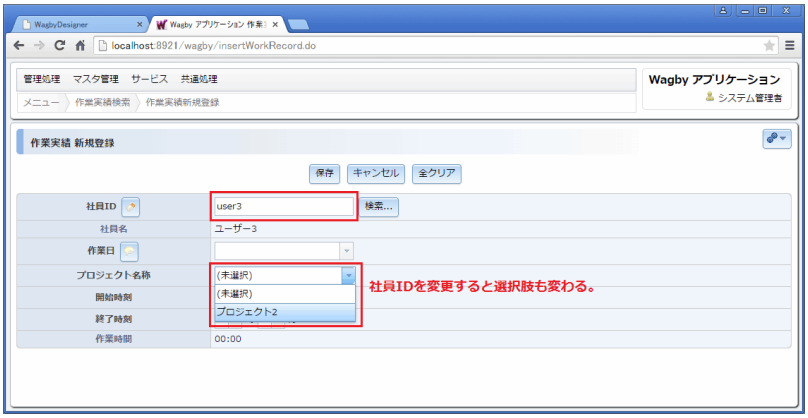


図 ユーザー3を選択した場合は、プロジェクト2が選択できるようになる

定義方法

モデルは「社員」「作業プロジェクト」「作業実績」を用意しています。

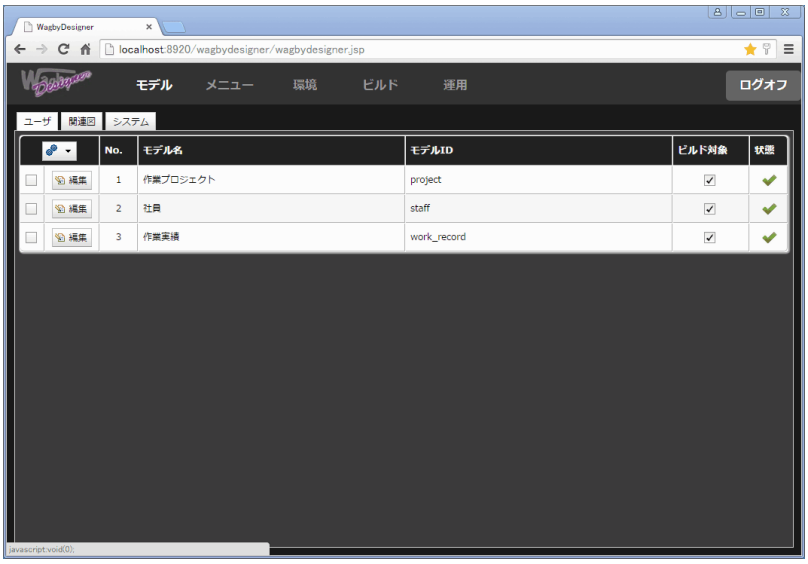


図 モデル一覧

「社員」モデルは ID と名称から構成されています。

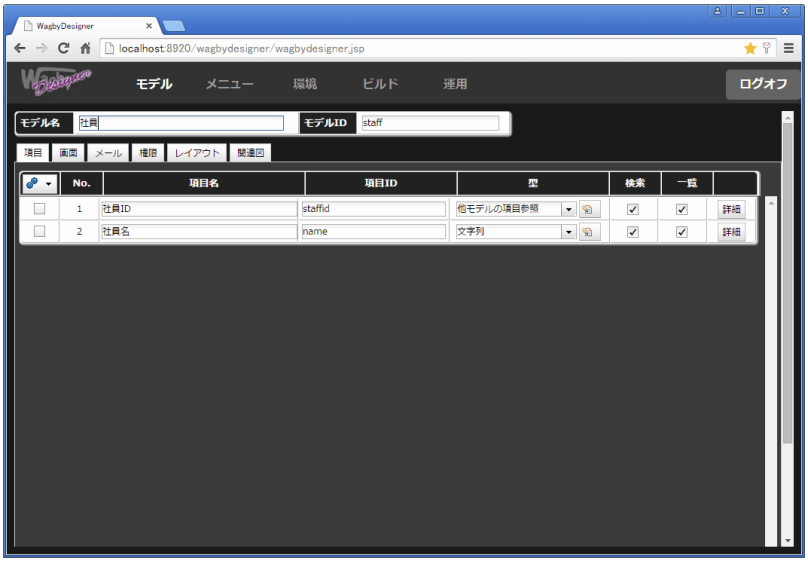


図 社員モデル

「作業プロジェクト」モデルには、1つのプロジェクトに複数名の社員が連なるようになっています。繰り返しコンテナで実現します。

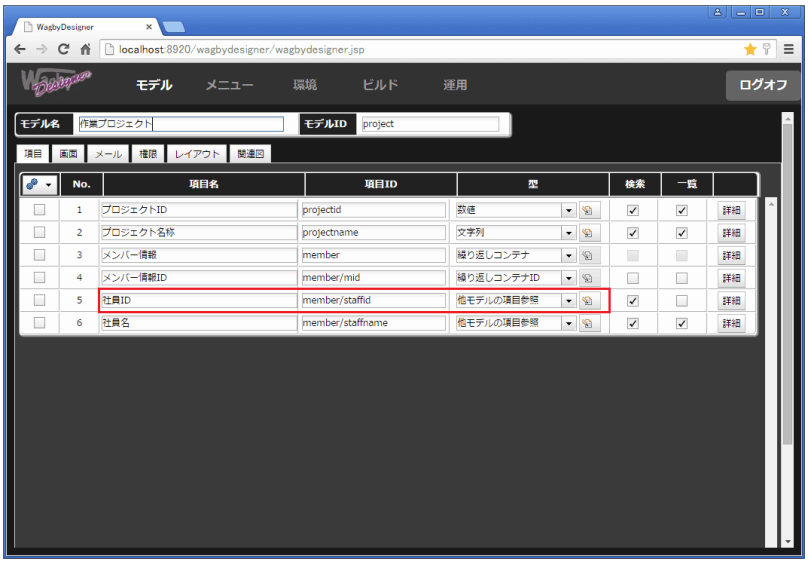


図 作業プロジェクトモデル

「作業実績」モデルのプロジェクト項目に絞り込みを設定します。

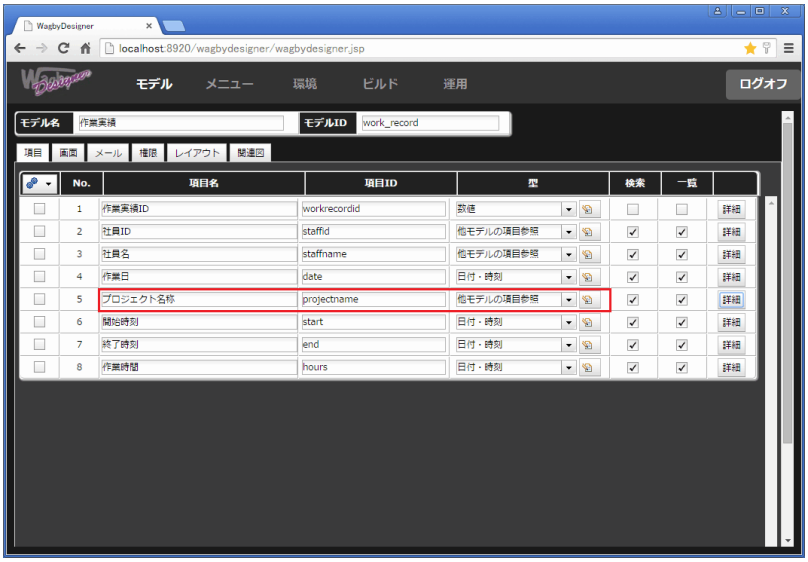


図 作業実績モデル

（同じモデルに含まれる）「社員ID」項目が、（参照先モデル「作業プロジェクト」の項目）「社員ID(member/staffid)」に含まれているもの、という絞り込みルールを設定します。

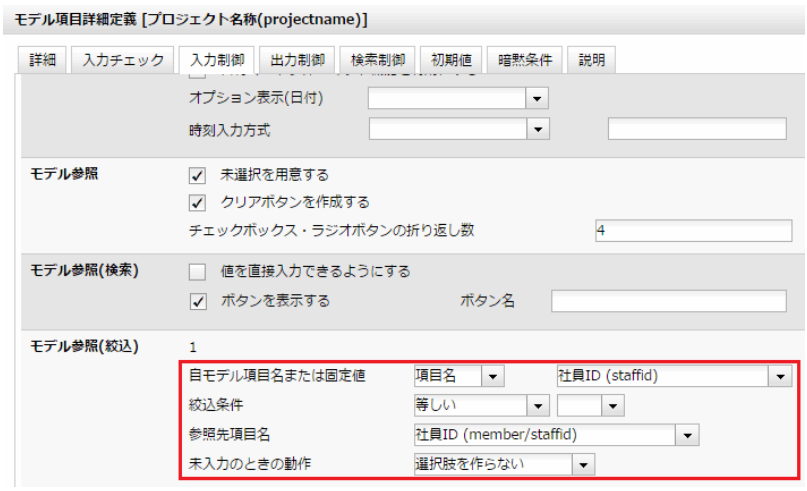


図 絞り込みの設定

固定値に関数を利用する（8.3.0）

絞り込み条件の固定値に関数を指定することができます。図, 33はログオン者が所属するグループによって「報告部署」項目が絞り込まれた例です。

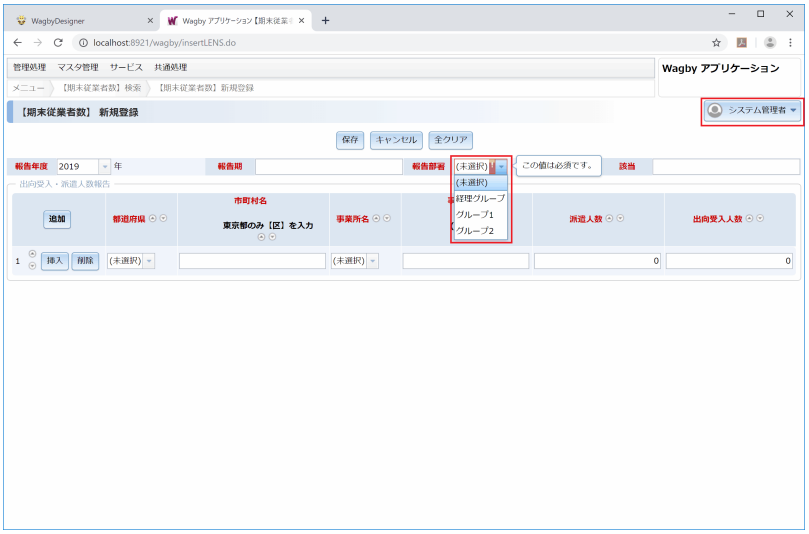


図 システム管理者は複数のグループに所属している

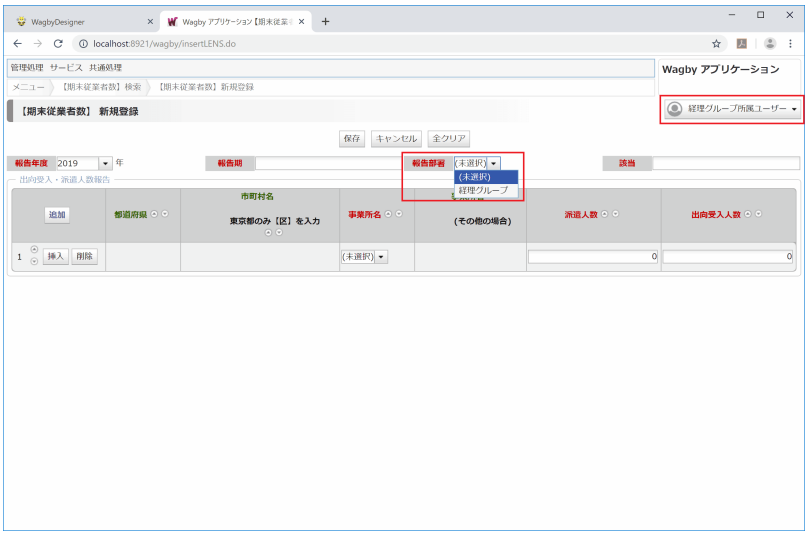


図 経理グループ所属ユーザでログオンすると、経理グループで絞り込まれる

定義方法

「報告部署」項目はグループモデル(jgroup)をリストボックス形式で参照する、としています。

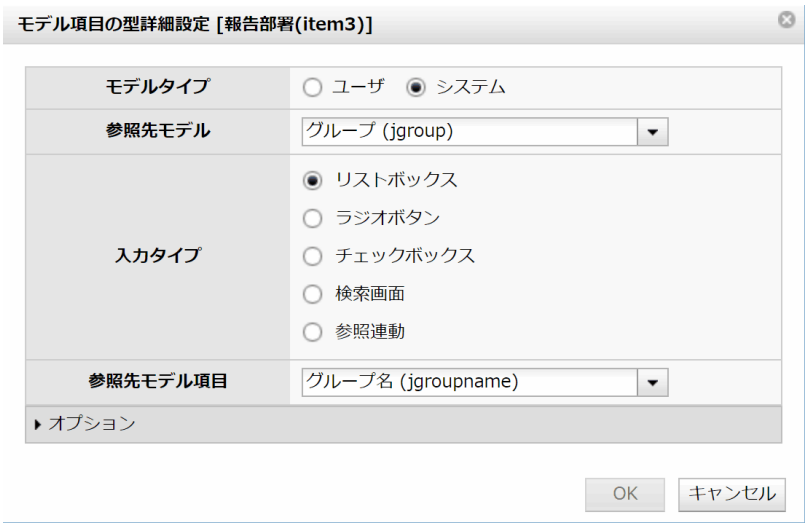


図 報告部署項目をモデル参照項目として用意する

絞り込みの設定を行います。固定値ですが、値部分に関数を用いています。

図 絞り込みの設定

- 関数 JGROUPID は、ログオンアカウントが所属するグループIDの配列を返します。
- 関数 ARRAY_OBJ は、関数 JGROUPID が返す基本型の数値型配列をオブジェクトの配列に変換します。

利用できる関数

"検索制御 > 暗黙条件 > 関数を利用する" と同じです。こちらをお読み下さい。

IS NULL/IS NOT NULLで絞り込む (8.3.2)

絞り込みの設定で IS NULL/IS NOT NULL 検索を行う場合、次のようにします。

検索方法	固定値	絞り込み条件
IS NULL	null	等しい
IS NOT NULL	null	等しくない

仕様・制約

動作の仕組み

本機能は、指定した絞り込み条件を SQL の where 句に含めます。すなわち絞り込み処理はデータベースで行います。

(51)

固定値と特殊表記now

日付型項目の絞り込みでは、固定値として特殊表記 nowを利用することができます。本日の日付による絞り込みが行えます。

(51) もう一つの絞り込み方法である「参照モデル同士の関係性で選択肢を絞り込む」は絞り込み元データをいったんすべてメモリに取り込み、Javaプログラムで条件判定を行います。

参照連動項目 !-- N20190110_133627 -->

参照連動項目に絞り込みは設定できません。（設定しても機能しません。）

サジェスト項目

サジェスト項目に絞り込みは設定できません。（設定しても機能しません。）

繰り返しコンテナ利用時の制約

絞り込み先項目がコンテナ外項目として定義され、かつ、絞り込み元項目が繰り返しコンテナ内項目として定義されている場合は、絞り込みとして利用されるのはコンテナの1行目のデータのみとなります。

2行目以降のデータは絞り込みとしては利用されず、無視されます。

ウィザード利用時の制約

関連する項目（絞り込みの関係をもった項目）は同じ画面に含まれている必要があります。別々のウィザード画面に配置した場合は適切な絞り込みが行われません。

トラブルシューティング

絞り込みが適切に行われない場合、次の点をご確認ください。

- 複数の絞り込み条件を指定している場合、その中の一つでも値が ″null″ となっていた場合、絞り込みが行えません。絞り込み条件に用いる値が適切かどうかを確認してください。
- 絞り込み条件に用いる項目が「参照連動」や「計算式」によって求められる場合、当該絞り込み処理の前にこれらの値が確定している必要があります。未確定の場合は ″null″ となるため、絞り込みが行えません。設計情報の項目の記載順序を工夫し、参照連動の連動元項目を、絞込を定義している項目よりも前に定義しておくことで問題を回避できる場合があります。
- 一覧表示画面および一覧更新画面に絞込設定したモデル参照項目を含める場合、絞り込み項目も一覧表示に含めるようにしてください。
- 対象項目の型が数値型や日付型の場合、絞込条件を「文字列の部分一致」にすることはできません。

関連するページ

- 検索画面>メインウィンドウの入力値を検索条件として再利用する
- チュートリアル>アドオンギャラリーを利用する>消費税アドオンの使い方

スクリプトを使った選択肢の絞り込み(スクリプト)

例

次のモデル参照項目を用意するものとします。

(a) アカウント(juser)を参照し、リストボックスとして扱う項目。 (b) グループ(jgroup)を参照し、リストボックスとして扱う項目。

ここで (b) の選択肢に、(a) が所属するグループのみを表示する設定を行うものとします。

新規登録画面の初期状態は次の通りです。「グループID (隠し項目)」は本来は隠しとするものですが、内容確認のために表示させています。二つのリストボックスはいずれも ″未選択″ となっています。

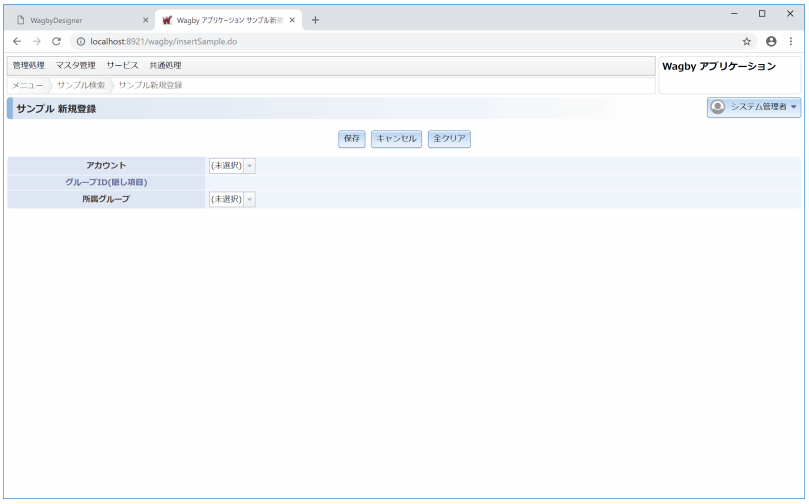


図 サンプルアプリケーション新規登録画面(1)

アカウントを選択します。

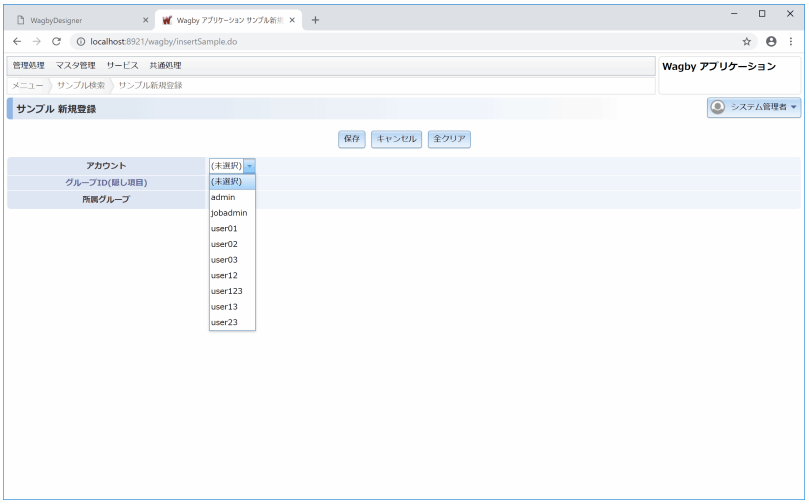


図 サンプルアプリケーション新規登録画面(2)

アカウントに紐づくグループは参照連動項目「グループID（隠し項目）」に反映されます。さらに、これが「所属グループ」項目のリストボックスの選択肢になっています。

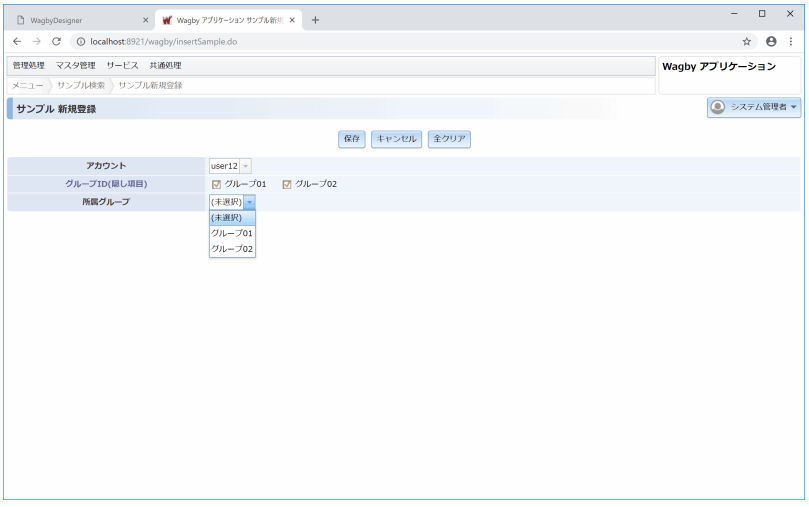


図 サンプルアプリケーション新規登録画面(3)

アカウントを選択しなおしたとき、グループの選択肢も変更されます。

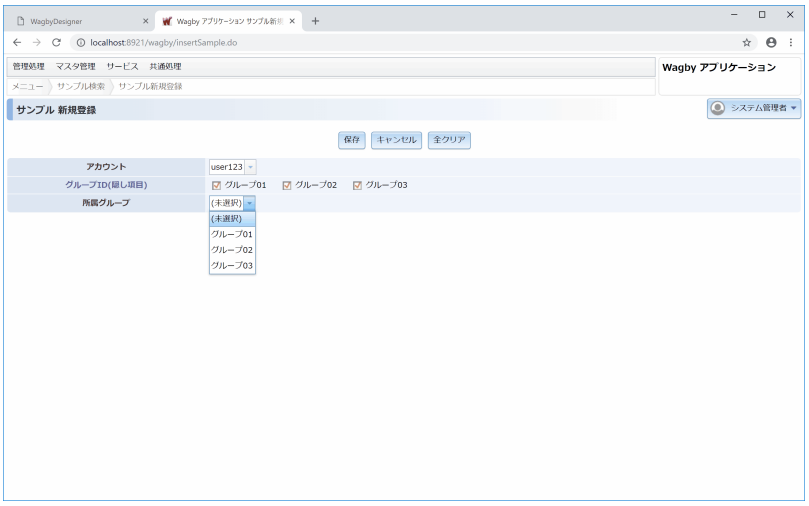


図 サンプルアプリケーション新規登録画面(4)

定義方法

サンプルモデルの定義を示します。

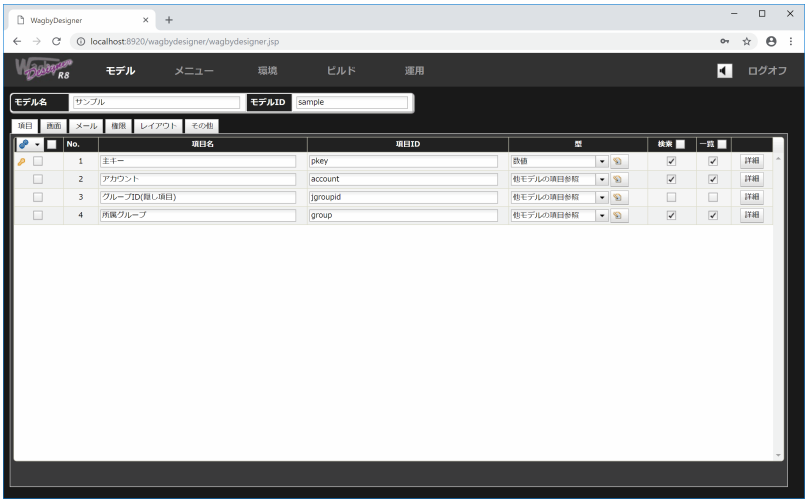


図 サンプルモデルの定義(1)

アカウントは juser モデルをリストボックスとして参照しています。また、この項目は″値が変更されたときに画面を再描画する″も有効にします。

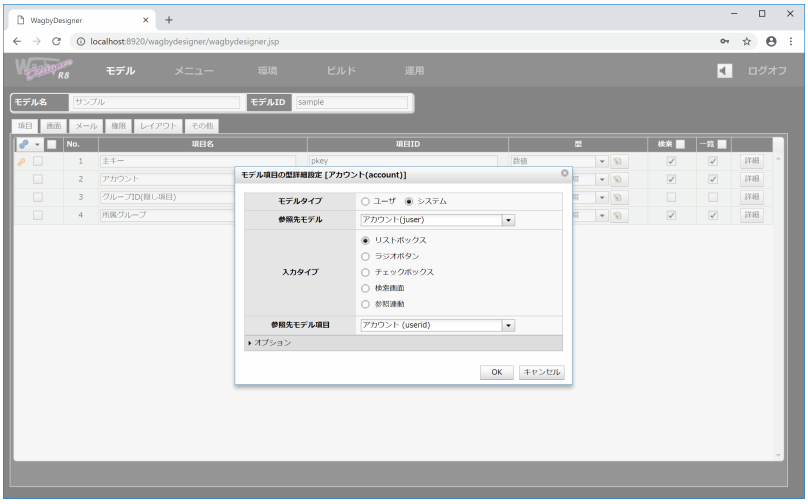


図 サンプルモデルの定義(2)

アカウントに紐づく所属グループを参照連動する項目です。この項目はチェックボックスとなります。

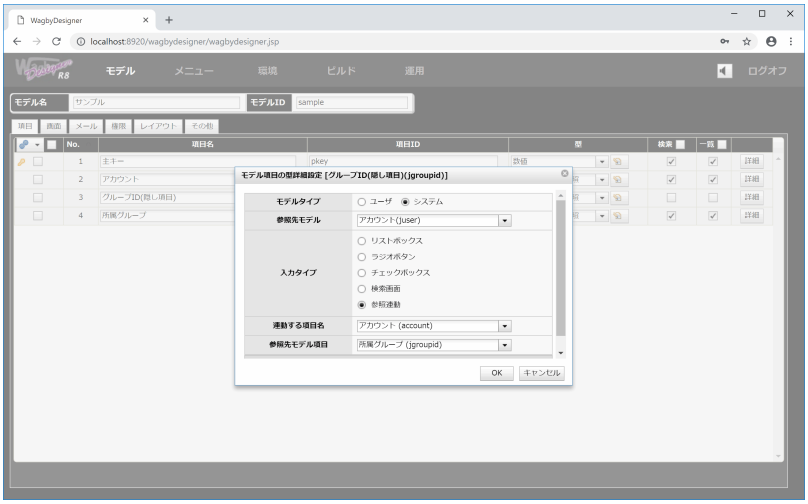


図 サンプルモデルの定義(3)

参照連動項目とは別に、所属グループをリストボックスとして用意します。

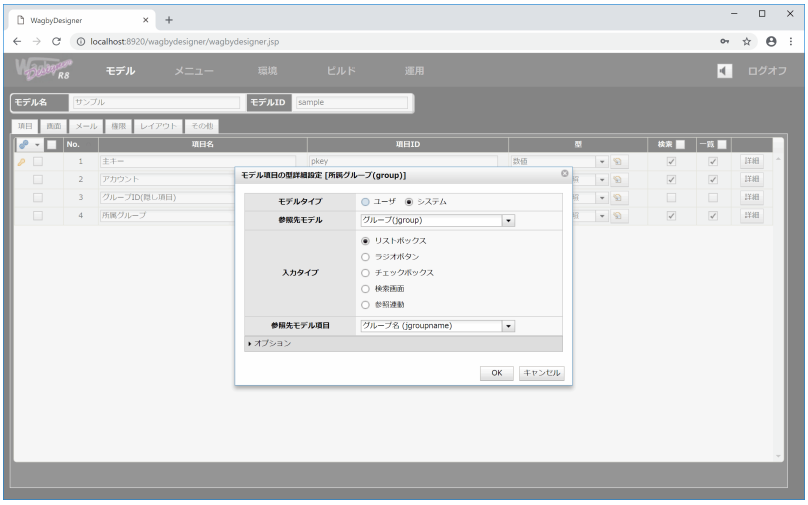


図 サンプルモデルの定義(4)

参照連動項目(jgroupid)を、リストボックス項目(group)に転記するためのスクリプトを用意します。このスクリプトは Wagby 内部で、ストアモデルからプレゼンテーションモデルを作成するタイミングで呼び出されます。

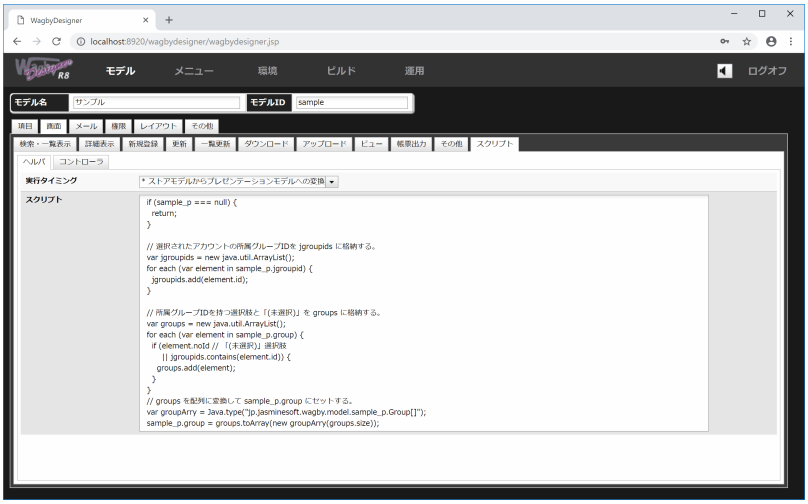


図 サンプルモデルの定義(5)

```
if (sample_p === null) {
  return;
}

// 選択されたアカウントの所属グループIDを jgroupids に格納する。
var jgroupids = new java.util.ArrayList();
for each (var element in sample_p.jgroupid) {
  jgroupids.add(element.id);
}

// 所属グループIDを持つ選択肢と「(未選択)」を groups に格納する。
var groups = new java.util.ArrayList();
for each (var element in sample_p.group) {
  if (element.noId //  「(未選択)」選択肢
      || jgroupids.contains(element.id)) {
    groups.add(element);
  }
}
// groups を配列に変換して sample_p.group にセットする。
var groupArry = Java.type("jp.jasminesoft.wagby.model.sample_p.Group[]");
sample_p.group = groups.toArray(new groupArry(groups.size));
```

サジェスト

例

「見積書」モデルにおいて、「得意先」の項目にサジェストを適用した例を図に示します。得意先コードとして "0" を入力すると、すぐに候補が表示されます。

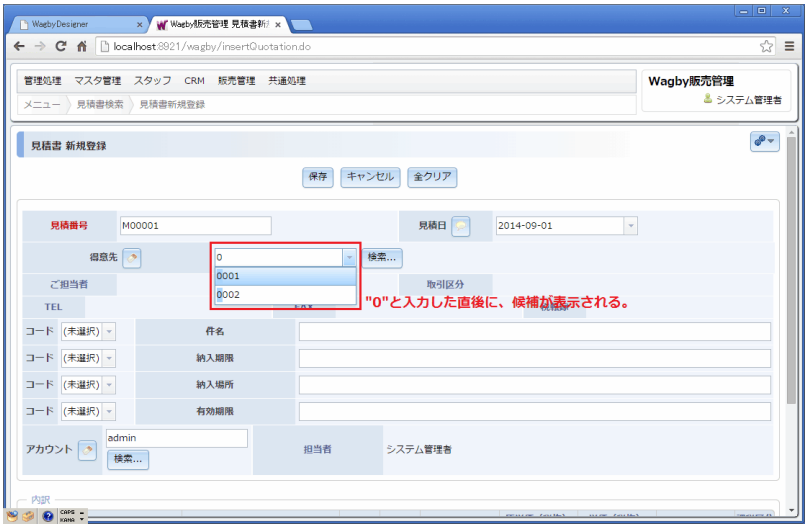


図 候補が表示された

候補から選択します。ここで参照連動を設定しておくと、候補から確定したタイミングで、参照連動の解決が行われます。

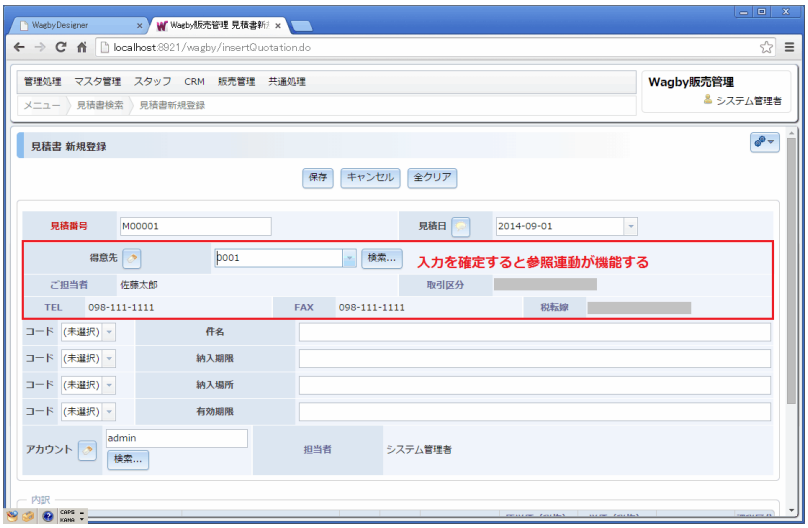


図 参照連動機能

定義方法

サジェストされる側のモデル

「得意先モデル」の「得意先コード」項目を検索条件として使うように設定します。また、この項目を「モデル参照（検索）」とします。

さらにモデル項目の詳細定義で、全文検索の欄を設定します。

図 全文検索の設定

図の設定は、見積書モデルに行うものではありません。参照先のモデル（ここでは「得意先モデル」）の「得意先コード」項目への設定となります。

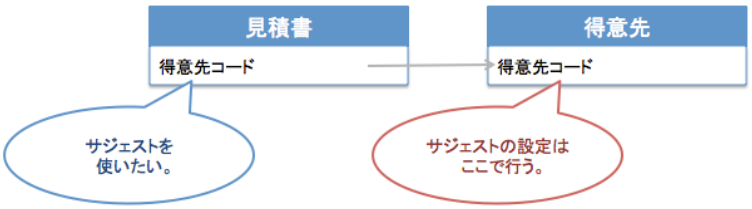


図 設定するモデル

サジェストのみの設定

「全文検索を行う」設定で、″全文検索″ を指定すると全文検索とサジェストの両方を利用できます。 ここで ″サジェストのみ″ を指定することもできます。この場合は全文検索機能は利用できません。

全文検索とサジェストの両方を適用すると、検索用のインデックスファイルのサイズが肥大化します。サジェストでしか用いない場合は ″サジェストのみ″ を選択するようにしてください。

文字列の一致方法の選択

通常は ″前方一致″ を利用してください。″部分一致″ を指定した場合は、候補が表示されるタイミングが2文字以上入力した後となります。

候補リストの同時表示数

サジェストの候補表示数を指定します。未設定時の標準は 20 件です。この値は 1 以上を指定してください。

サジェストを用いる側

サジェスト入力機能を用いる側のモデル（上の例では「見積書」モデル）側の項目に設定します。

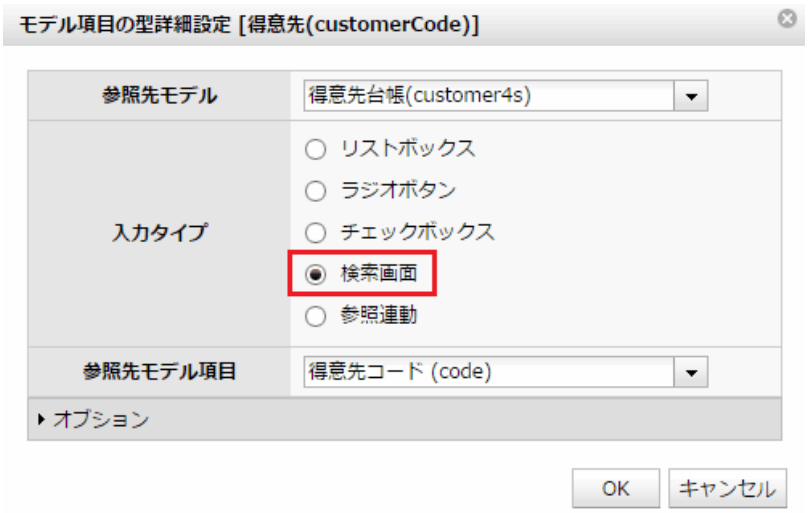


図 参照の設定

サジェスト入力欄の長さの指定

入力長を指定することもできます。

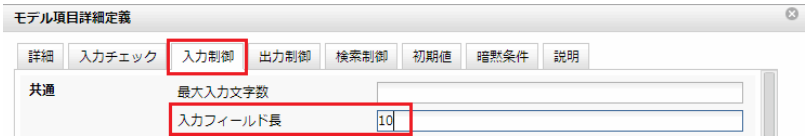


図 入力フィールド長の指定

仕様・制約

設定できる項目

- サジェストは参照先が文字列型の場合のみ使用できます。それ以外の型の場合は、通常のサブウィンドウ検索として機能します。
- （参照元が）繰り返しコンテナ内の項目の場合にも使用できます。

検索機能

文字列の一致方法を指定することができます。

前方一致検索

- ″前方一致″ を指定した場合、例えば ″北九州″ という文字列に対して ″北″ はヒットしますが、″九州″ はヒットしません。
- 前方一致検索の最大文字列長は 64 文字です。

部分一致

- ″部分一致″ を指定した場合、例えば ″北九州″ という文字列に対して ″九州″ もヒットします。
- ″部分一致″ を指定した場合、候補が表示されるタイミングが2文字以上入力した後となります。

表示順序（ソート）

- 標準では主キー項目の昇順でソートします。
- サジェストが参照するモデルの、全文検索対象項目に対してソート順の設定を行っていた場合、この数値の「符号」がサジェストの並び順にも反映されます。（ソート順の数値が負数の場合は逆順となります。）

表示順序をカスタマイズすることはできません。サジェストで表示される候補の標準値は20件ですが、実際には大量のデータをソートした上で最上位の20件のみを表示しています。このソートルールは昇順か降順かのいずれか、となります。

制約

- サジェストを使う項目は検索条件として使うように設定する必要があります。
- サジェストを使う項目は「データベースに保存する」が有効になっている必要があります。
- サジェストによって表示される候補は、有効期限の設定が考慮されません。（有効期限の設定があっても、無視して表示されます。）
- 「主キー（コード）を直接入力する」機能との併用は行えません。どちらを使うか迷ったときは、サジェストの利用を推奨します。
- 絞り込み設定との併用はできません。
- サジェスト入力欄への「項目の値が変更されたら画面を再描画する」設定は無視されます。

その他

全文検索・サジェストのために利用する検索用インデックスファイルはデータベースとは別に作成されます。詳細. . .

［例］固定メッセージの選択・追加による入力方式の実現

例

最初に、図に示す例文を管理する「例文」モデルを用意します。

図 例文モデル

図の例文モデルを利用する「報告書」モデルを新規作成します。 例文コードは「サジェスト」を設定しています。コードを入力すると候補が表示されます。

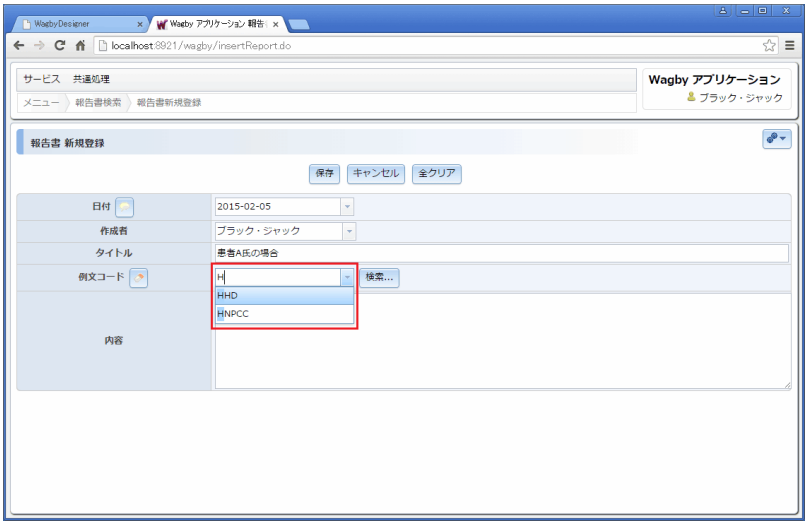


図 サジェストによる例文コードの指定

「内容」項目は例文コードに紐づく文字列を参照連動（かつ自モデル保存）すると設定しているため、参照連動された値がセットされます。

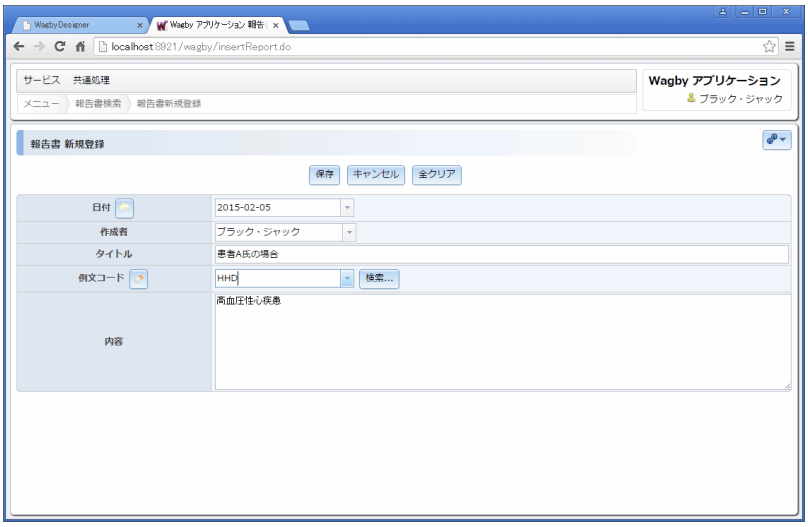


図 参照連動による例文の設定

「内容」項目に文字列を追加記入し、再び例文コードを指定します。

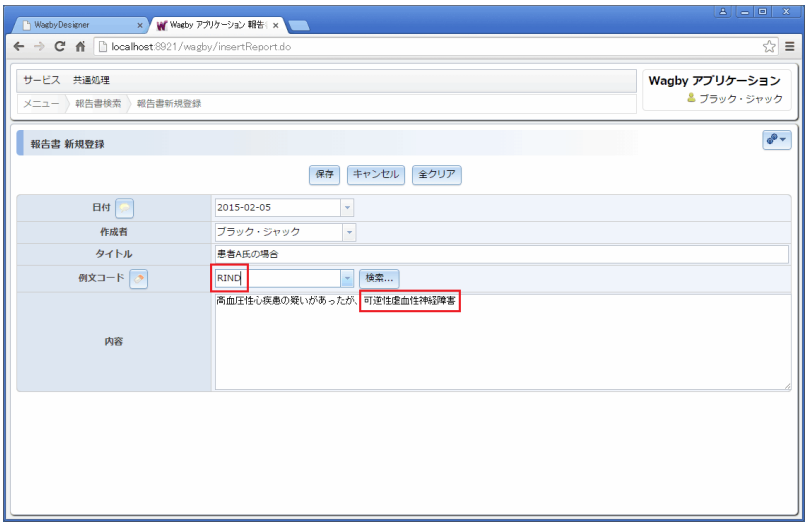


図 再び例文コードを指定する

これまでの入力文字列を保持したまま、その後ろに参照連動された値が追記されました。 この参照連動を何度でも行うことができます。 最終的に、数回の参照連動による入力支援で、入力を終了します。

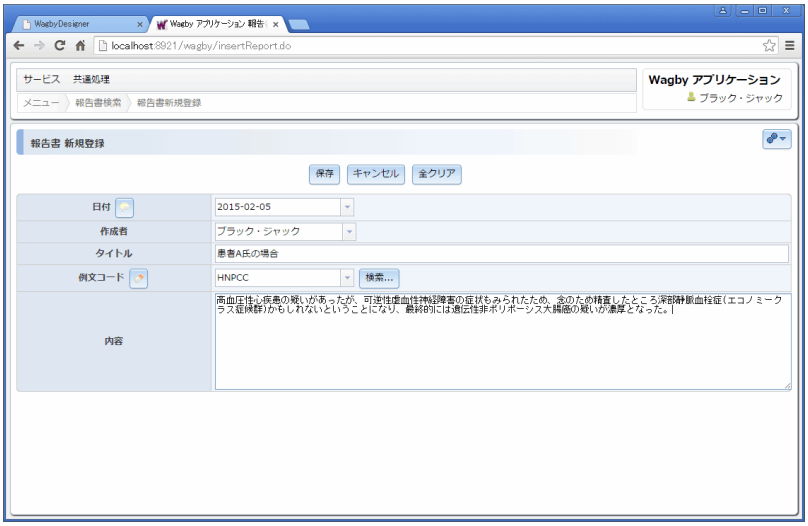


図 参照連動された文字列が追記される

詳細表示画面では「例文コード」を非表示（隠し項目）としています。

図 詳細表示画面の例

定義方法

今回のアプリケーションは二つのモデル「例文コード」と「報告書」を用意しています。

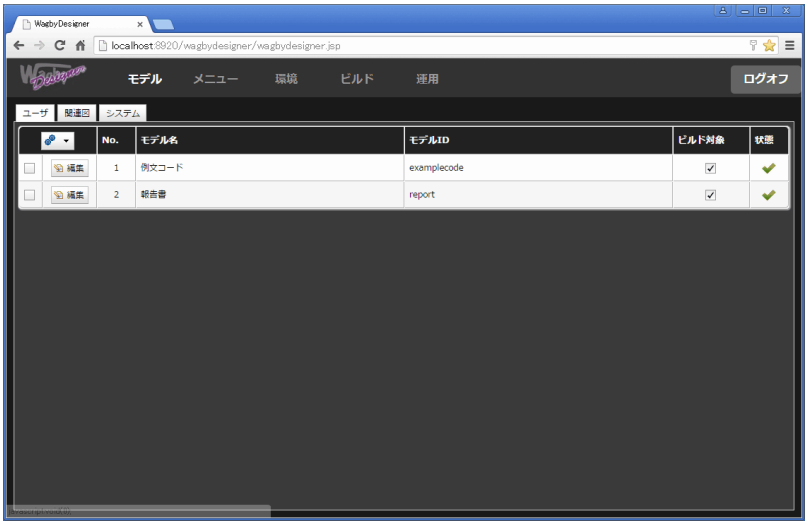


図 モデル定義

例文コードモデル

「例文コード」モデルは、例文コードを文字列型の主キーとしています。

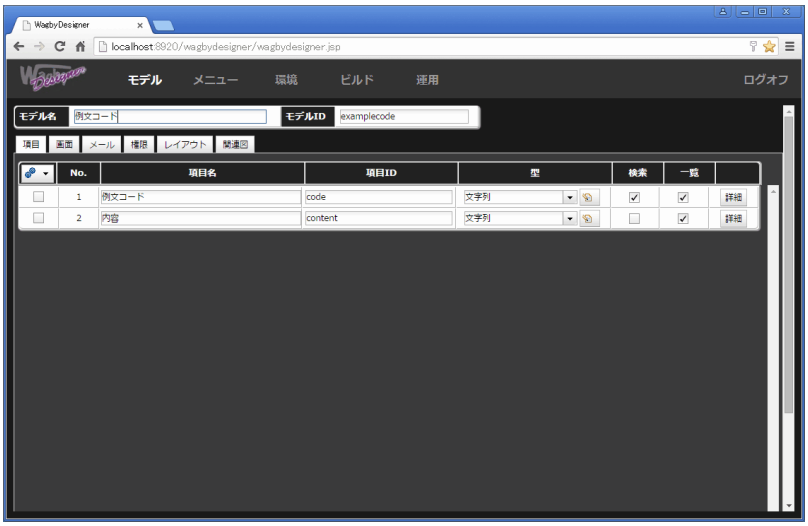


図 例文コードモデルの定義

例文コード項目は「サジェスト」を有効にしています。

図 サジェストを有効にする

報告書モデル

ポイントは「例文コード」項目と「内容」項目です。

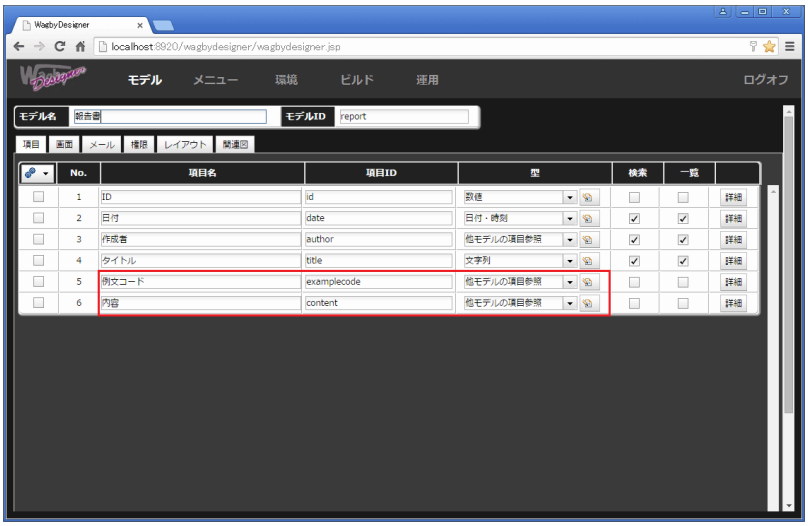


図 報告書モデル

例文コード項目

例文コードモデルを参照（検索）として定義します。

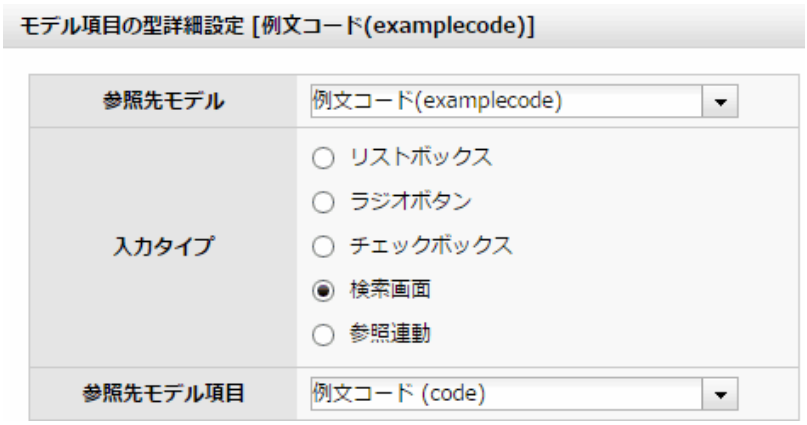


図 モデル参照の設定

詳細画面では非表示とします。

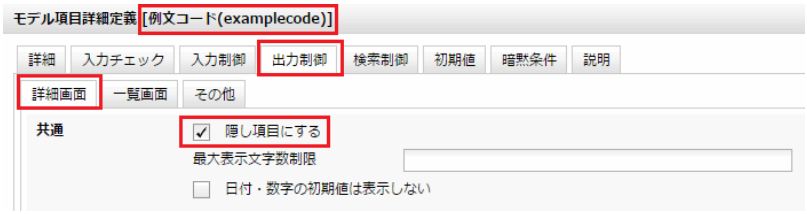


図 隠し設定

内容項目

内容項目は参照連動とします。

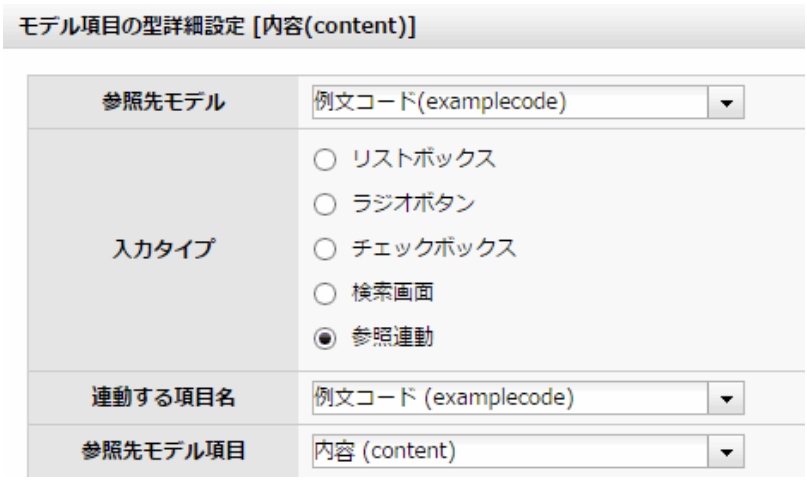


図 参照連動の設定

ポイントは、「参照連動の定義方法」に「追記」を指定していることです。（この選択肢が空白、または「上書き」の場合は、常に参照連動された値で上書きします。）

図 参照連動を追記とする

また、この「内容」項目は自モデル保存とします。

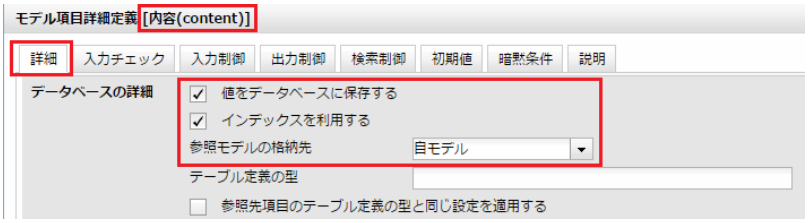


図 自モデル保存の設定

最後に、この「内容」項目はテキストエリアとし、読み込み専用指定を解除します。

図 テキストエリア＆読み込み専用を解除

仕様・制約

- 本機能は参照連動された値が格納される項目が文字列型であることが前提です。（数値型、日付型、ストアモデル参照項目やマスタモデル参照項目などは対象となりません。）
- アップロード更新については通常の編集画面と違い、アップロード時に参照先が変更されている場合、参照連動項目に追加してデータベースに保存されます。追加された値を編集する場合は、再度ダウンロードして、編集し、アップロードします。また、追加後の値に関する入力チェックは行われません。
- 一覧更新では、参照先を変更した際に画面をリロードし、追記された値が表示されます。追加後の値に関する入力チェックは行われません。

・　一覧更新画面における更新処理では、最初の変更は追記されます。しかし元の値に戻した場合に参照連動は追記されず、元の値に戻ります。
・　テキストエリア － 入力可能文字数を表示する機能との併用を行うことはできません。

［例］選択肢の絞り込みを用いて、操作時点での消費税を定める

例

「消費税率」モデルを用意します。図に示す初期データを登録しておきます。

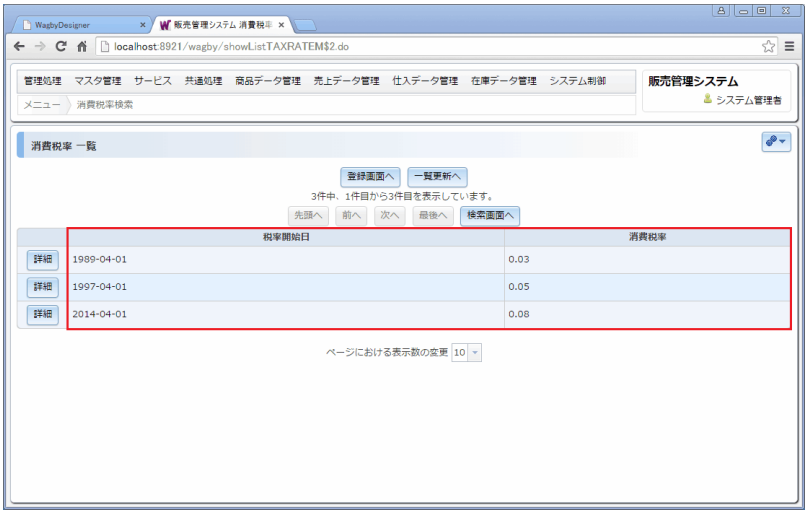


図　消費税率モデルの設定値

受注モデルで利用した例です。販売額に対して、操作時点での消費税率が適用されています。

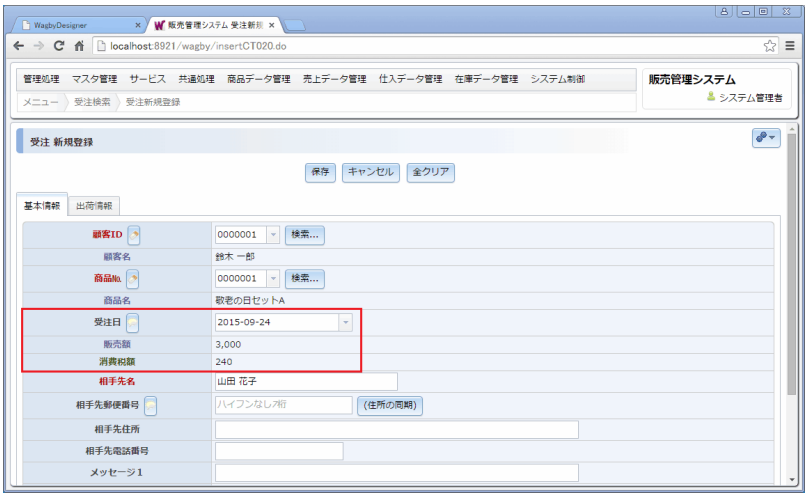


図　消費税額の計算例

定義方法

消費税率モデル

「税率開始日」を日付型とします。「消費税率」を8バイト浮動小数点型とします。

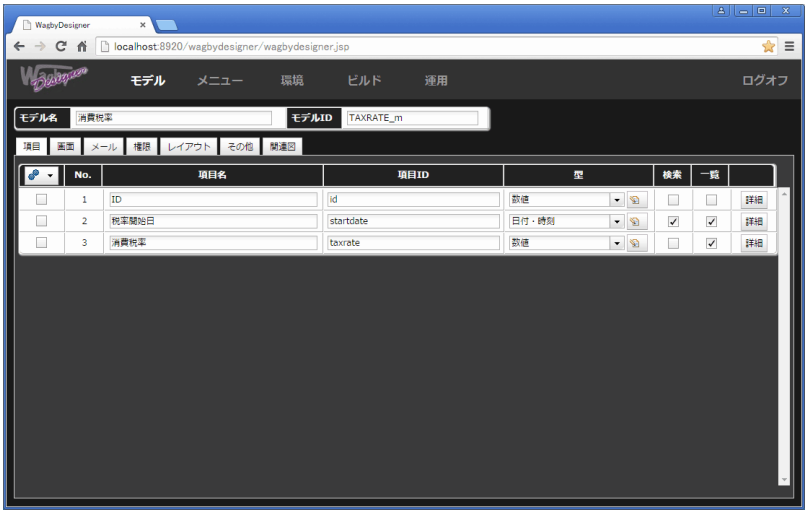


図 消費税率モデル

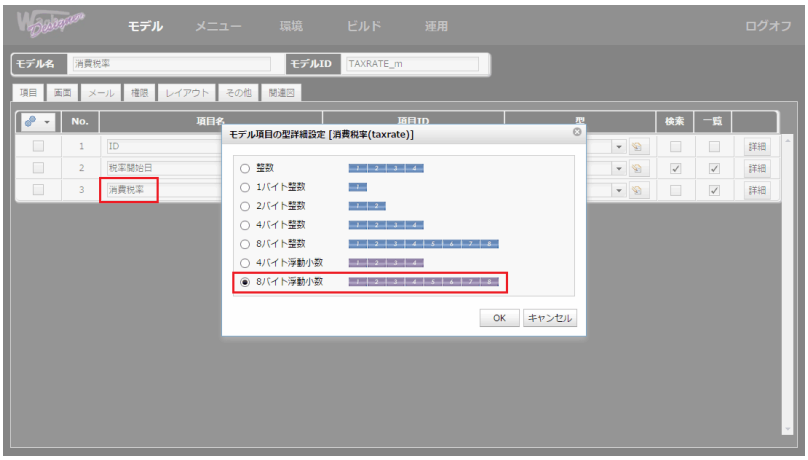


図 消費税率項目の型

消費税計算を行うモデル

消費税の計算を行うモデルでは「日付」「金額」「消費税」「消費税額」に相当する項目を用意します。

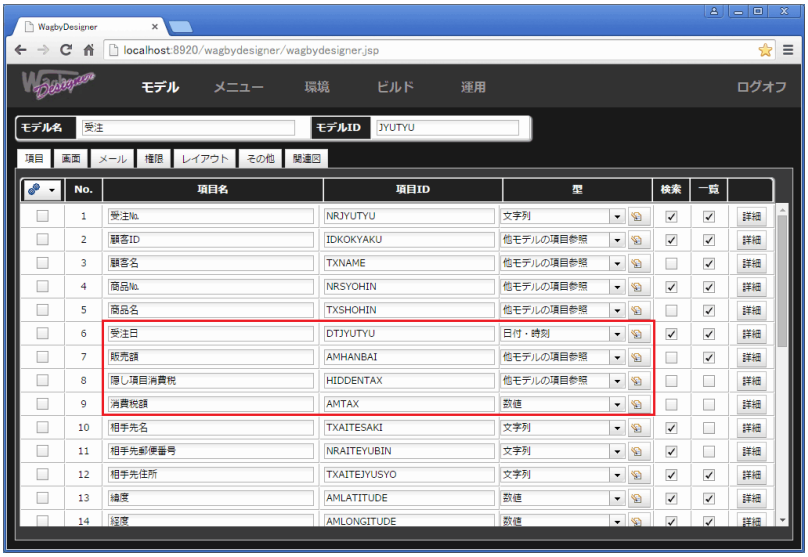


図 受注日、販売額、消費税、消費税額の項目

項目	説明	型
日付	税率を求める時点の日付項目	日付型。対象日となる。
金額	税率を求める対象となる項目	整数型。参照連動を用いて、商品マスタにある値を転記する、といった使い方もできる。
消費税	算出日時点での消費税	消費税モデルを参照する。後述する設定により、対象日付の消費税に絞り込む。
消費税額	金額と消費税を掛け合わせて求める。	誤差がないような式とする。（後述）

隠し項目消費税

この項目は消費税モデルの参照となります。リストボックスとしてください。

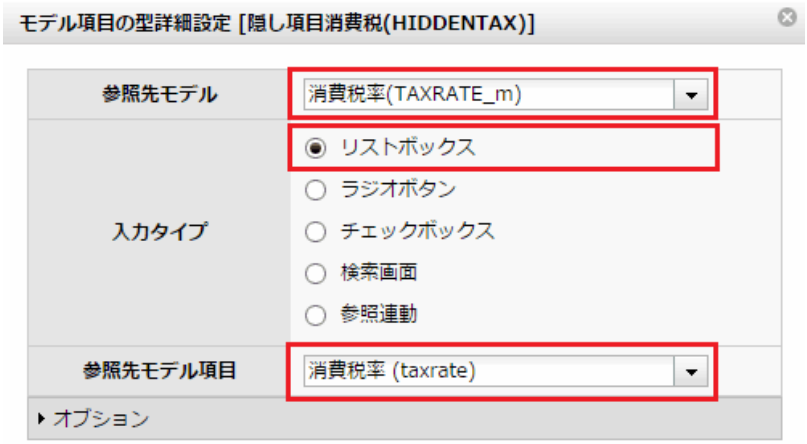


図 消費税モデルをリストボックスで参照する。

この値はデータベースに保存しない、とします。

図 データベースに保存しない

計算時の消費税を記録として残しておく、という場合は、データベースに保存してください。

「他項目の値で選択肢を絞り込む」設定を行います。次のようにします。

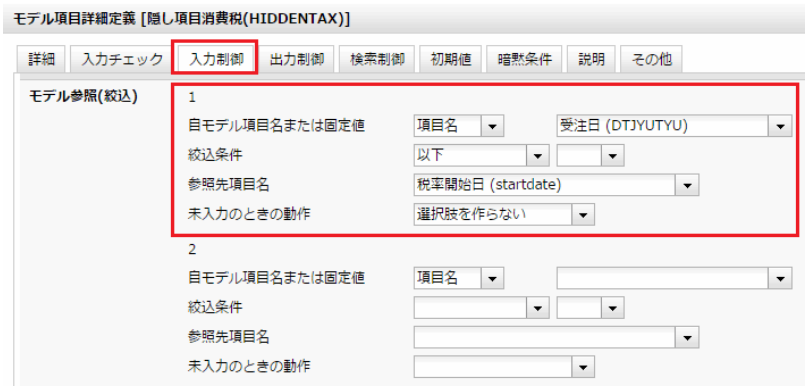


図 他項目の値で選択肢を絞り込む設定

項目名

対象日が格納された日付項目を指定します。

絞込条件

対象日より前で絞り込むため、″以下″を指定します。

参照先項目名

消費税モデルの「税率開始日」項目を指定します。

未入力のときの動作

″選択肢を作らない″とします。

さらに、この選択肢は「税率開始日」の降順であらかじめソートさせておき、かつ、（複数ある選択肢の）先頭1つだけを用いる、とします。

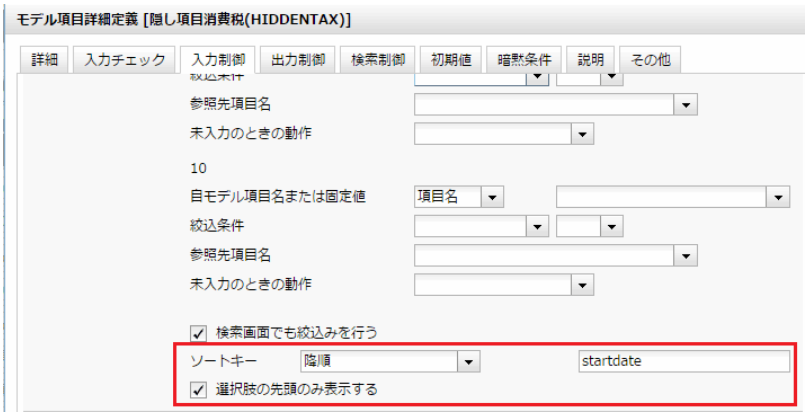


図 選択肢の先頭のみ表示する

この設定により、現時点における消費税率が確定されます。この設定のイメージは次のとおりです。

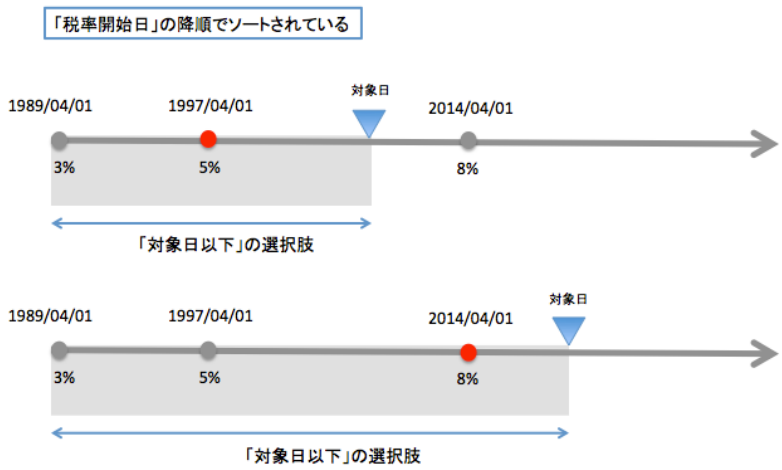


図 絞り込みのイメージ

さらに、この項目は入力時に隠しとします。さらに出力時にも隠しとします。これによって画面には表示されません。

図 入力時に隠し項目とする

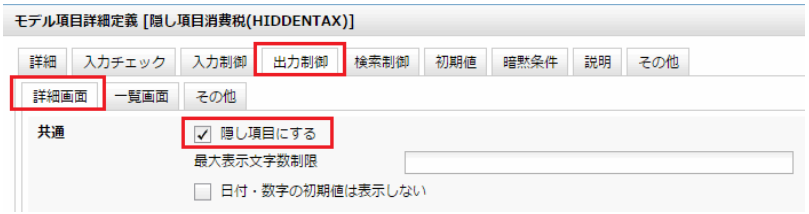
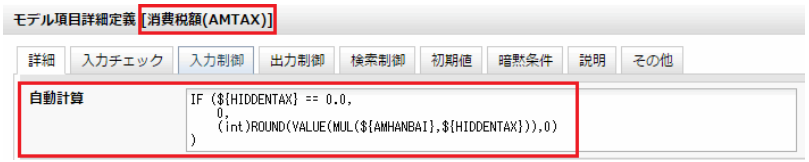


図 出力時に隠し項目とする

開発時は、隠し指定を解除するとよいでしょう。現在の消費税率を画面上で確認できます。

消費税額

消費税額を求める式を示します。




```
IF (${HIDDENTAX} == 0.0,
    0,
    (int)ROUND(VALUE(MUL(${AMHANBAI},${HIDDENTAX})),0)
)
```

- 消費税項目はモデル参照です。${HIDDENTAX} と記述すると、内容部（消費税率）を取得します。
- 販売額と消費税率の掛け算は一般に小数点部分で誤差が生じます。この解決のために、通常の乗算ではなく、MUL関数を用います。詳細は"計算式 > 演算精度"をお読みください。
- MUL関数の戻り値は文字列型です。VALUE関数を適用して、これを8バイト浮動小数点に変換します。
- ROUND 関数に桁数に 0 を指定し、数値を最も近い整数として四捨五入するようにしています。
- 最終的な計算結果は整数型になるため、最後に（int）を先頭に付与して明示的に型変換を行っています。
- 消費税マスタに存在しない日付（最初の消費税が設定されるより前の日付）を受注日とした場合、「隠し項目消費税(HIDDENTAX)」項目は空白(null)となりますが、${HIDDENTAX} と記述すると 0.0 が返されます。これは Wagby が展開した後の式で、選択した結果が null の場合は（8バイト浮動小数点型の初期値である）0.0 を返すようになっているためです。この性質を用いて、この場合は消費税を 0 としています。

関係性のないリストボックスを計算によって連動させる

例

サンプルの顧客モデルに「血液型」項目と「血液型説明」項目を用意しています。血液型が"(未選択)"の場合、説明欄は空白です。

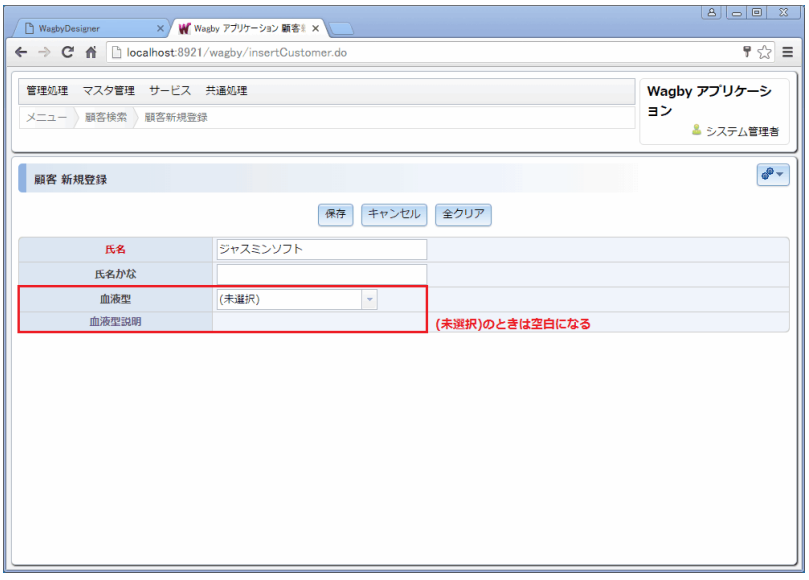


図 サンプルの登録画面

「血液型」を選択すると、これに対応した説明文が表示されます。

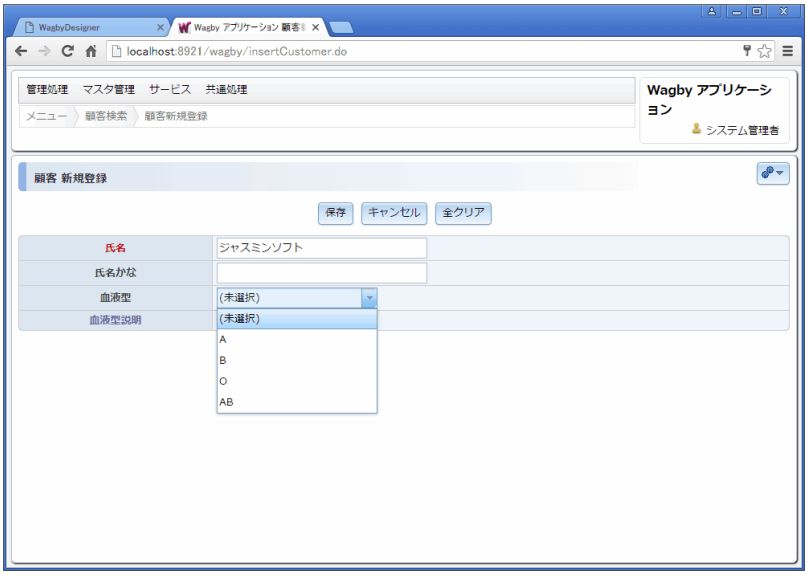


図 血液型を選択する

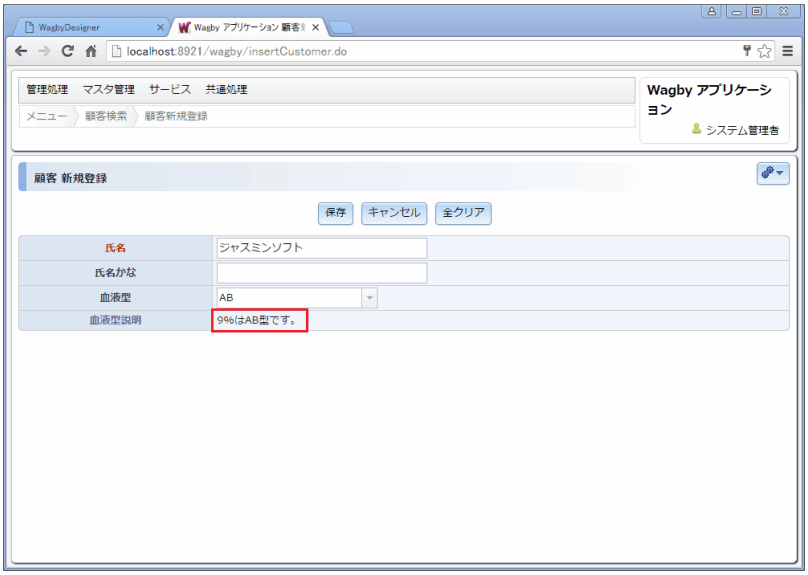


図 説明文が表示される

定義方法

この例の特徴は、「血液型」モデルと「血液型説明」モデルにはデータベースの観点から関連性をもたせていないことです。計算式によって表示するIDを特定させます。

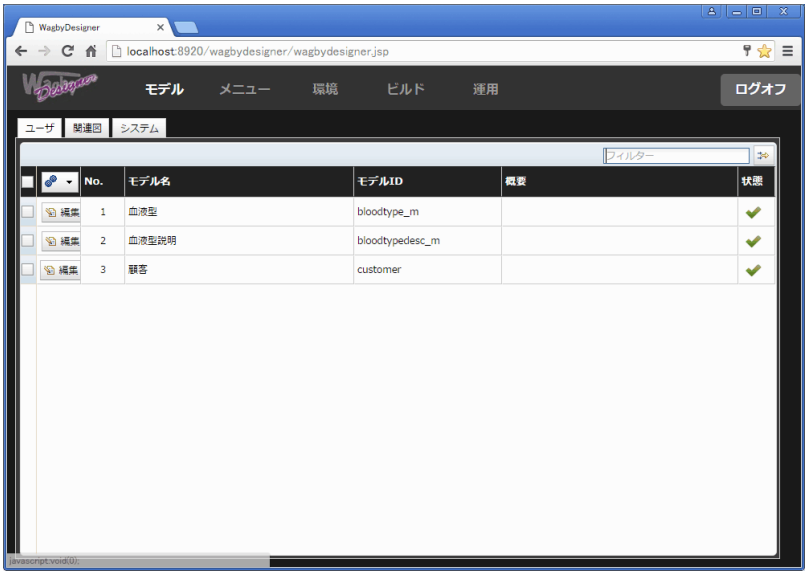


図 今回のサンプルで用いたモデル

血液型モデル

「選択肢モデル（初期データ固定）」としています。

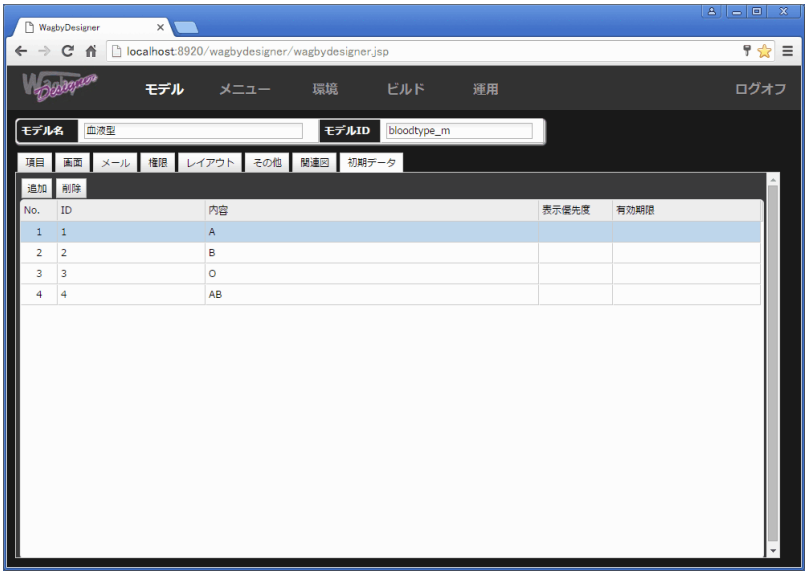


図 血液型モデルの初期値

血液型説明モデル

「選択肢モデル（初期データ固定）」としています。

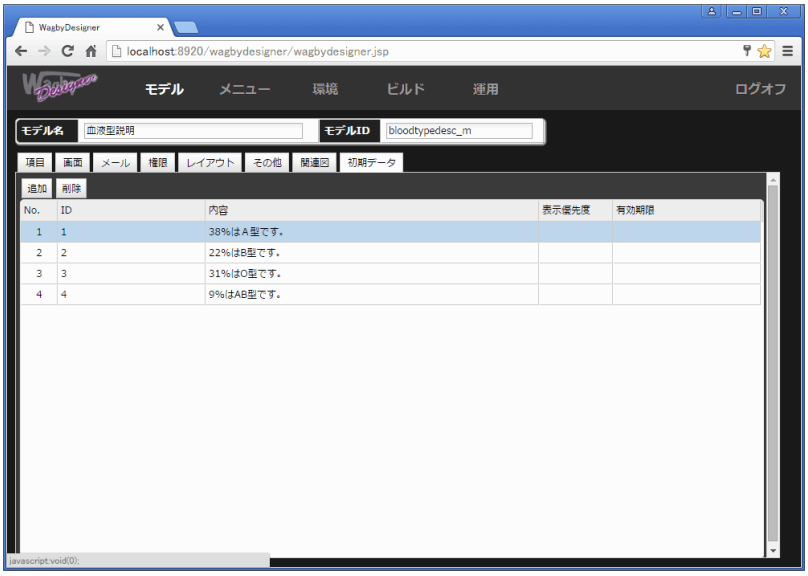


図 血液型説明モデルの初期値

顧客モデル

「血液型」項目ならびに「血液型説明」項目を用意します。

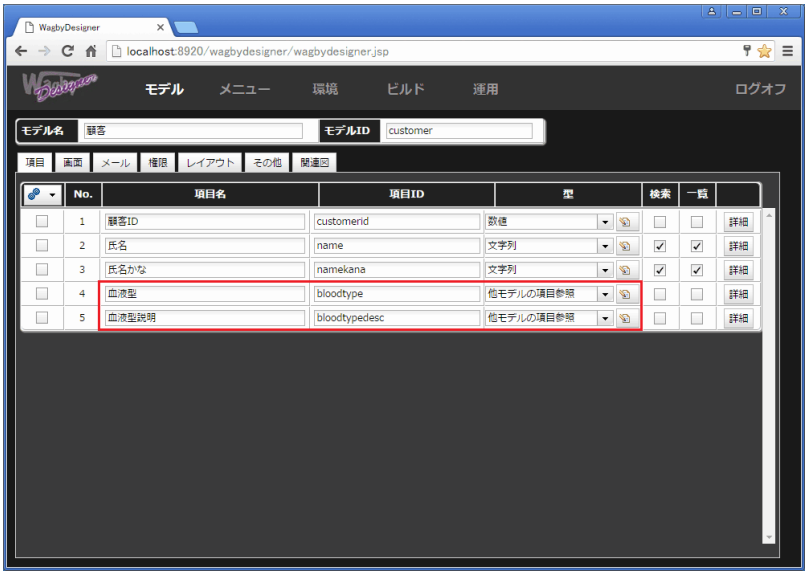


図 顧客モデル

血液型項目

血液型モデルをリストボックス形式で参照します。

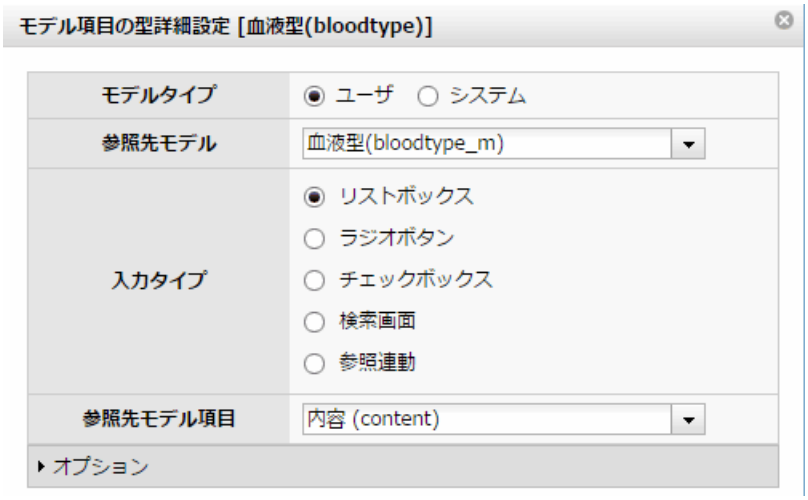


図 血液型項目（モデル参照）

このリストボックスには"(未選択)"を用意します。

図 未選択を用意する

血液型説明項目

血液型説明モデルをリストボックス形式で参照します。参照連動ではありません。また、血液型モデルと（データベースの項目による）絞り込み設定もありません。

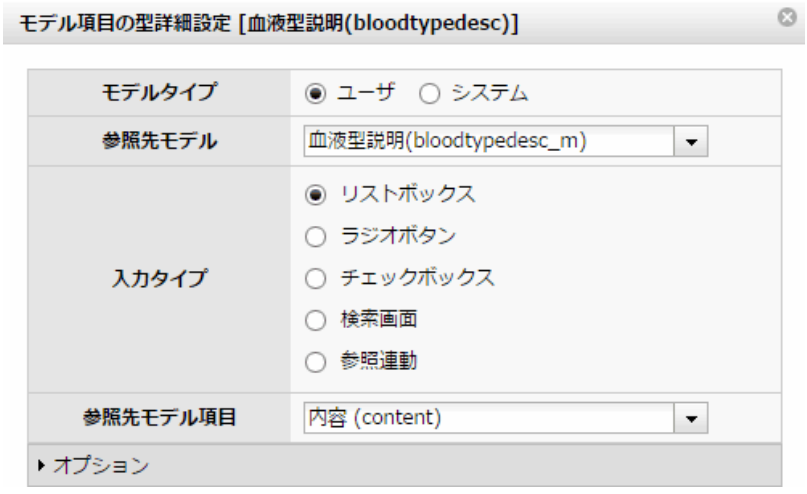


図 血液型説明項目（モデル参照）

このリストボックスには"(未選択)"を用意しません。

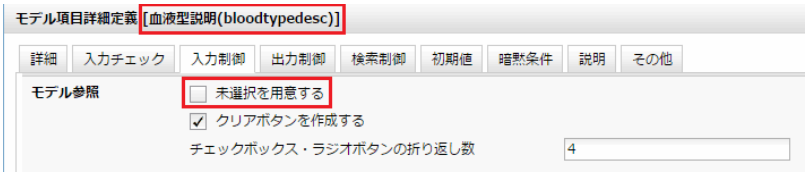


図 未選択を用意しない

式の設定

血液型説明項目に式を設定します。血液型の選択肢(ID部)の値によって、自身の表示内容を制御します。血液型が未選択状態の場合、自身をnullとします。（この項目は未選択を用意しないため、図のような空白表記になります。）

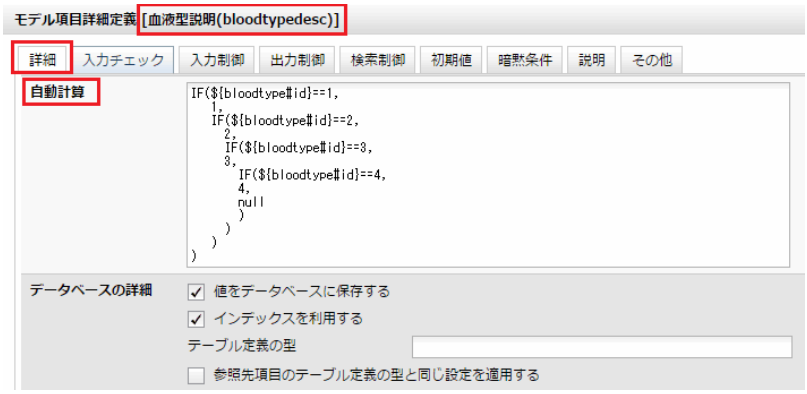


図 式の設定

```
IF(${bloodtype#id}==1,
    1,
    IF(${bloodtype#id}==2,
      2,
      IF(${bloodtype#id}==3,
      3,
        IF(${bloodtype#id}==4,
        4,
        null
        )
      )
    )
)
```

式を設定するため、この項目は自動的に「読み込み専用」となります。リストボックスとしては動作しません。

繰り返し

繰り返し項目

例

文字列型の「電話番号」項目を繰り返しと設定した例を図に示します。項目名の横に「追加」ボタンが用意されます。

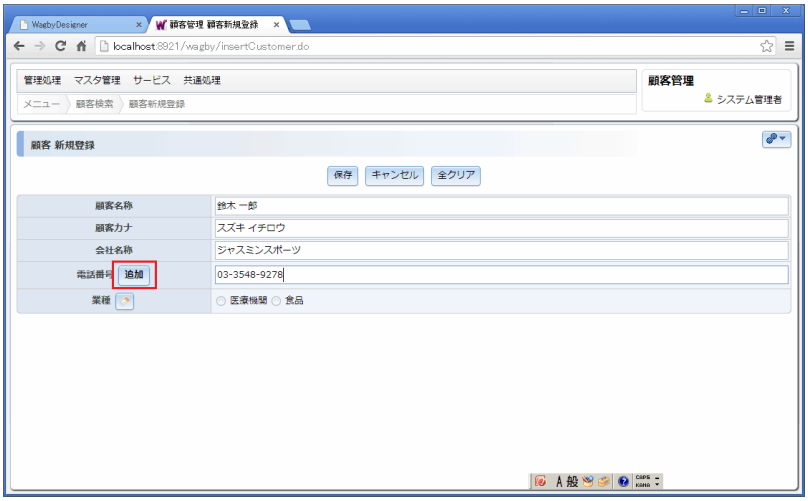


図 繰り返し設定を行うと「追加」ボタンが用意される

追加ボタンを押下すると入力欄が一つ増えます。

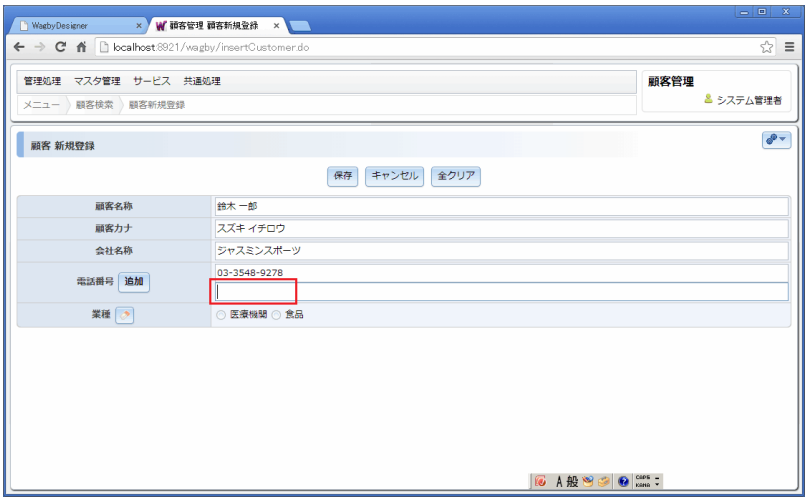


図 入力欄が追加される

詳細画面ではカンマ区切りで列挙されます。

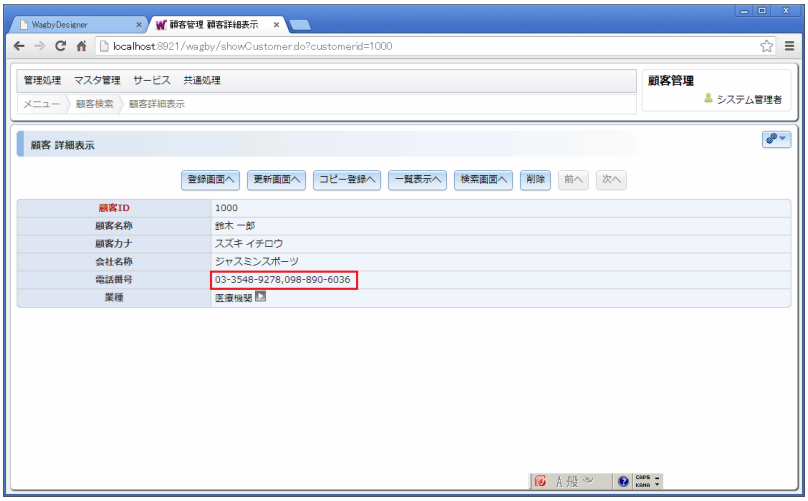


図 繰り返し項目の詳細表示画面

1つのモデルに、複数の「繰り返し」項目を含めることができます。（上限はありません。）

定義方法

モデル項目詳細定義の「詳細」タブ内に用意された「繰り返しとして利用する」をチェックします。

図 繰り返しとして利用する

設定できる型

繰り返し項目が指定できる項目の型は次のとおりです。

- 文字列
- 数値
- 日付
- 日付時刻

モデル参照、文字列（ファイル）は繰り返し項目として指定できません。この場合は「繰り返しコンテナ」を利用します。

ルール

- 数値型の場合、"0" という値は常に画面に表示されなくなります。
- 日付型の場合、"1970-1-1"（システム時刻）という値は常に画面に表示されなくなります。
- 「追加」ボタンを押下すると、画面再描画後に入力位置まで自動スクロールが行われます。
- 「追加」ボタンを複数回押下しても、新規の入力欄は最大1つまでとなります。（複数個の入力欄は用意されません。）
- 空白のデータを登録することはできません。新規の入力欄が空白のまま保存すると、その空白分は無視されます。

データベースとの関係

繰り返し属性を指定した項目は別テーブルに分割されるため、登録できるデータ数の上限はありません。詳細...

繰り返し項目 - 初期表示数

例

繰り返し属性を指定した「電話番号」の初期表示数を "5" とした例を図に示します。

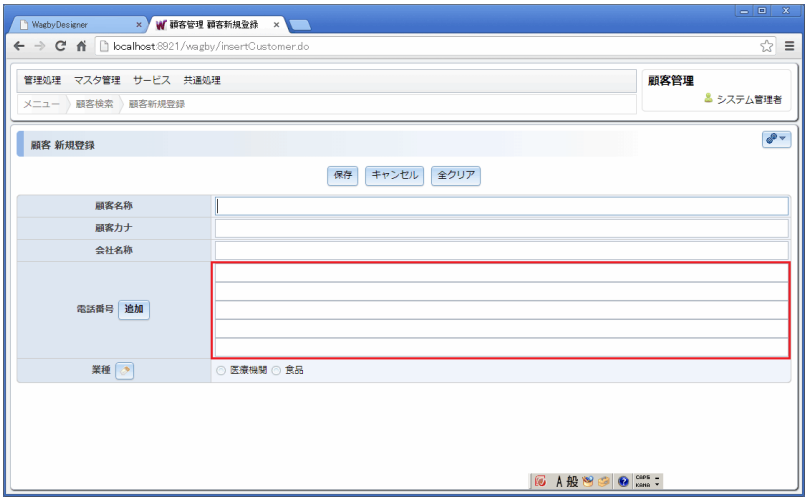


図 初期表示数5の例

定義方法

モデル項目詳細定義ダイアログの「入力制御」タブ内にある「繰り返し - 初期表示数」に数字を記述します。

図 初期表示数の設定

繰り返し項目の初期表示数を「0」と指定しても、一つの入力欄は必ず用意されます。

繰り返し項目 - 繰り返し項目の上限数

例

繰り返し属性を指定した「電話番号」の上限数を ″3″ とした例を図に示します。
入力欄が3つになると、「追加」ボタンを押すことができなくなります。

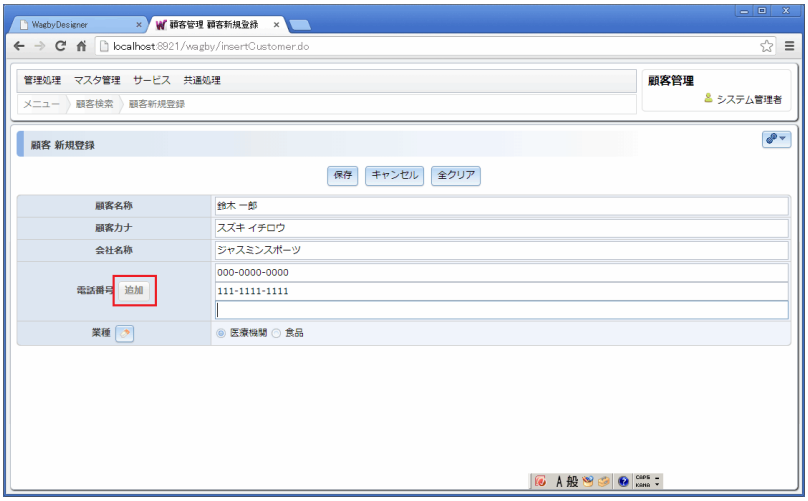


図 上限数を設定

定義方法

モデル項目詳細定義ダイアログの「入力制御」タブ内にある「繰り返し － 繰り返し項目の上限数」に数字を記述します。

図 上限数の設定

この値が空白または ″-1″ の場合、上限なし、となります。

繰り返し項目 － 区切り文字

例

繰り返し属性を指定した「電話番号」の区切り文字の標準はコンマ ″,″ です。

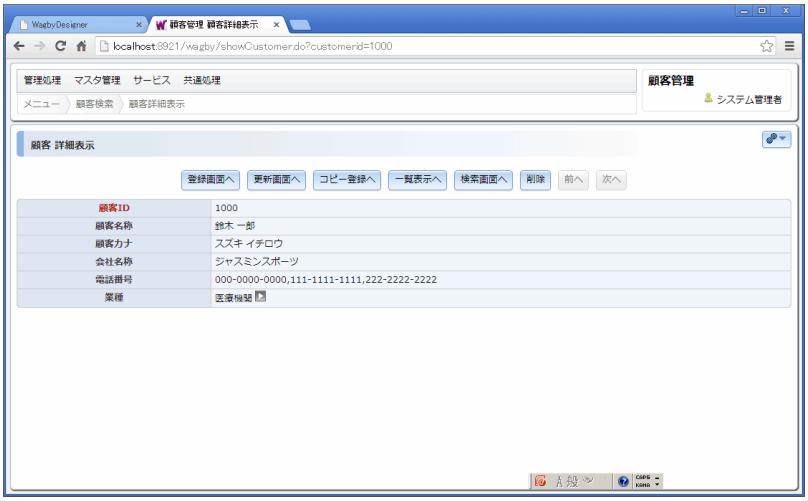


図 区切り文字の標準（詳細画面）

この区切り文字″改行″とした例を図に示します。

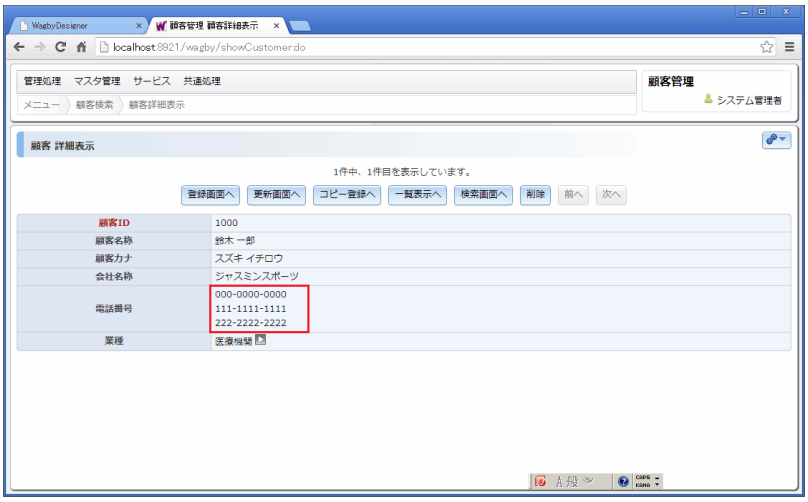


図 区切り文字を改行に設定（詳細画面）

一覧画面にも適用できます。

図 区切り文字を改行に設定（一覧画面）

定義方法

モデル項目詳細定義ダイアログの「入力制御」タブ内にある「繰り返し - 繰り返し項目の区切り文字」に文字を記述します。

ここで図に示すように、特別な表記として

```
<br/>
```

と記載すると、改行文字になります。

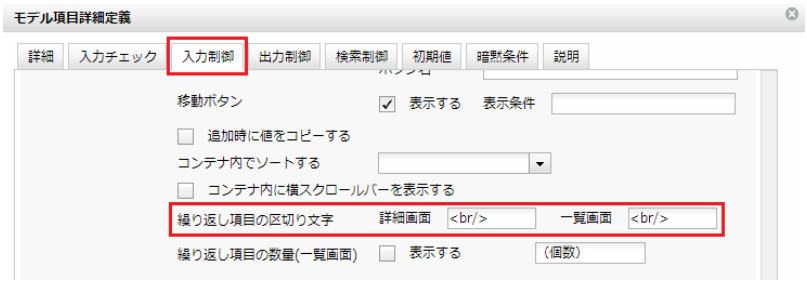


図 区切り文字の設定

応用 - 重複を除く

例

繰り返しコンテナ内のデータが重複する可能性がある場合、重複を除いた値を格納するダミー項目を用意します。

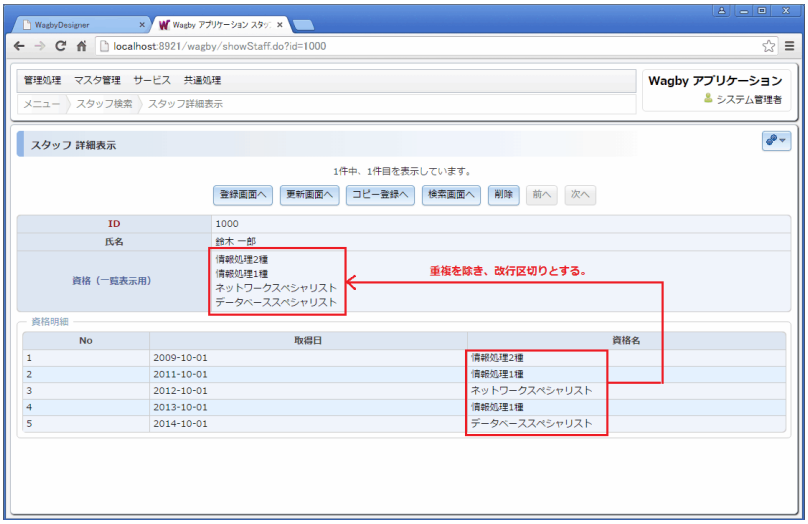


図 繰り返しコンテナの値から重複を除く

一覧表示でも改行されて表示されるようにします。

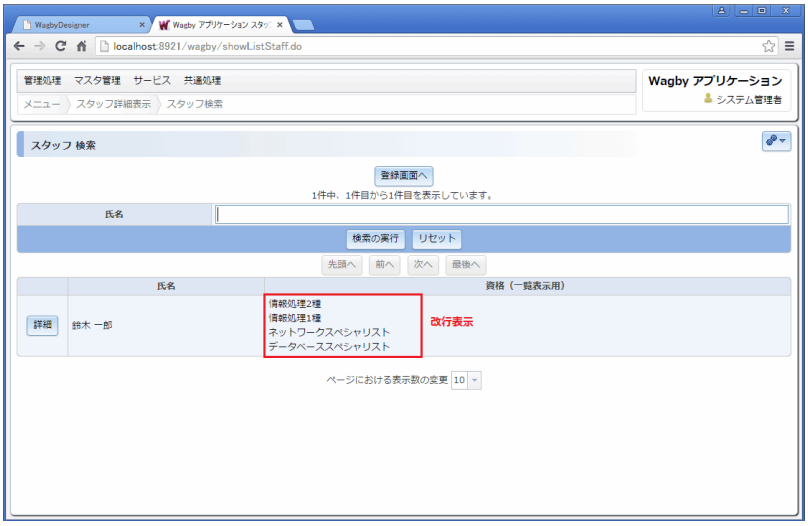


図 一覧表示

定義方法

重複を除いた値を格納するダミー項目を定義します。

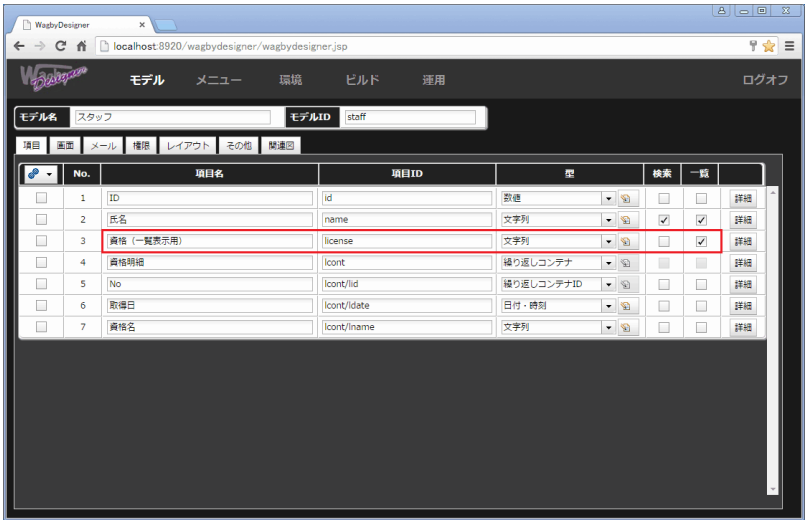


図 モデル定義

DDJOIN関数を使って、重複を覗きます。区切り文字を改行文字とします。

```
DDJOIN(${lcont/lname},"¥n")
```

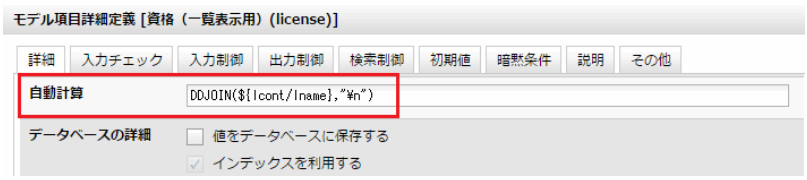


図 DDJOIN関数

改行のため、この項目をテキストエリアとします。計算項目のため自動的に読み取り専用として扱われます。

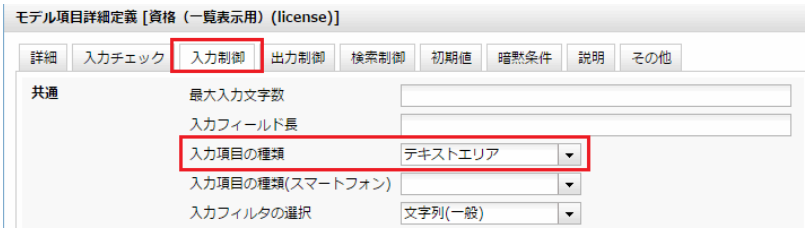


図 テキストエリアの設定

詳細表示画面、一覧表示画面それぞれで「（テキストエリアを）ラベルで表示する」を有効にします。

図 ラベルで表示する-詳細画面

図 ラベルで表示する-一覧表示

繰り返し項目 - 数量の表示

例

一覧画面で、繰り返し属性を指定した「電話番号」の内容ではなく、個数を表示した例です。

図 個数の表示

定義方法

モデル項目詳細定義ダイアログの「入力制御」タブ内にある「繰り返し　－　繰り返し項目の数量（一覧表示）」の「表示する」を有効にします。

同時に（表示上の）単位を指定することもできます。標準では〝(個数)〟となっています。

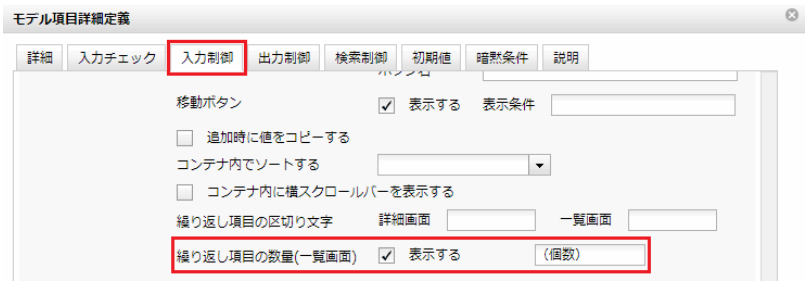


図 数量の設定

繰り返しコンテナ

例

図に繰り返しコンテナを用意した画面を示します。

追加

繰り返しコンテナはグループ（行）単位で追加することができます。「追加」ボタンを押下すると現在の行の下に、追加されます。

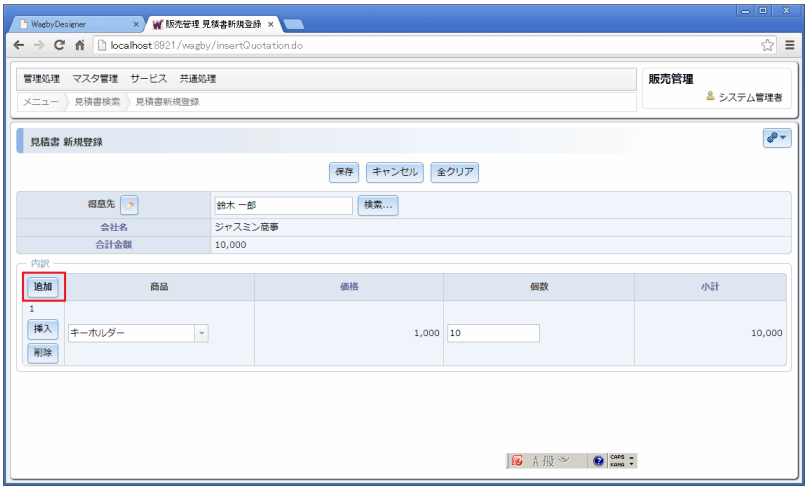


図 繰り返しコンテナの追加ボタンを押下する

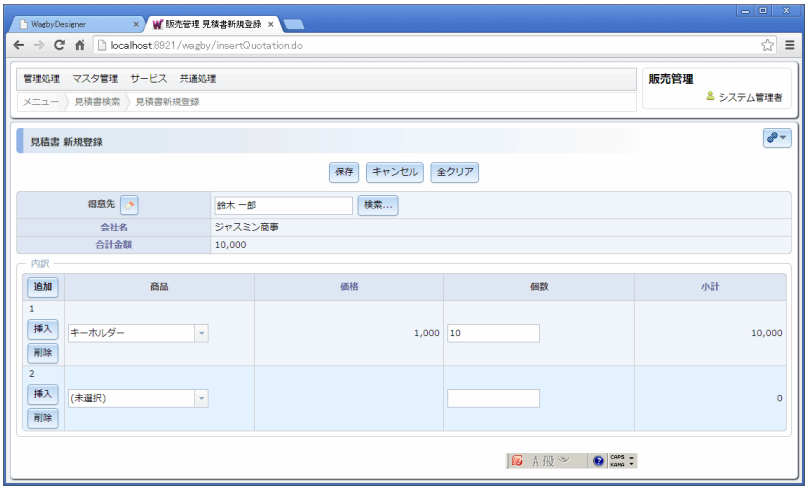


図 行が追加された

挿入

「挿入」ボタンを押下すると、行の挿入を行うことができます。

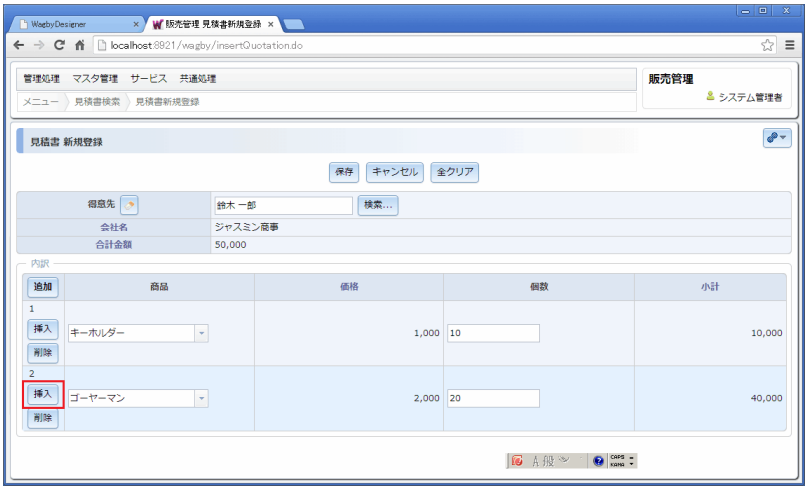


図 繰り返しコンテナの挿入ボタンを押下する

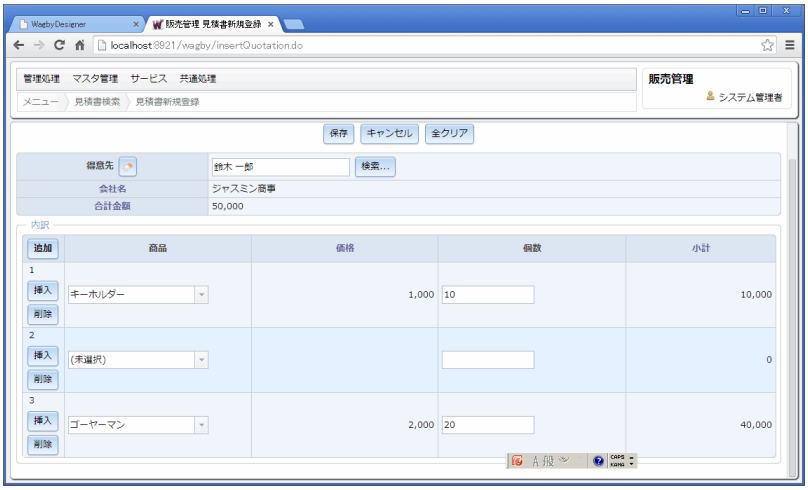


図 行が挿入された

削除

「削除」ボタンを押下すると、行の削除を行うことができます。

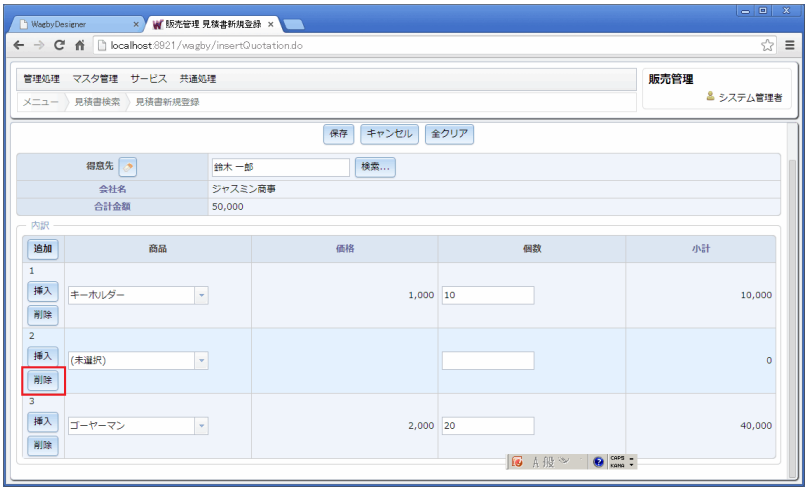


図 繰り返しコンテナの削除ボタンを押下する

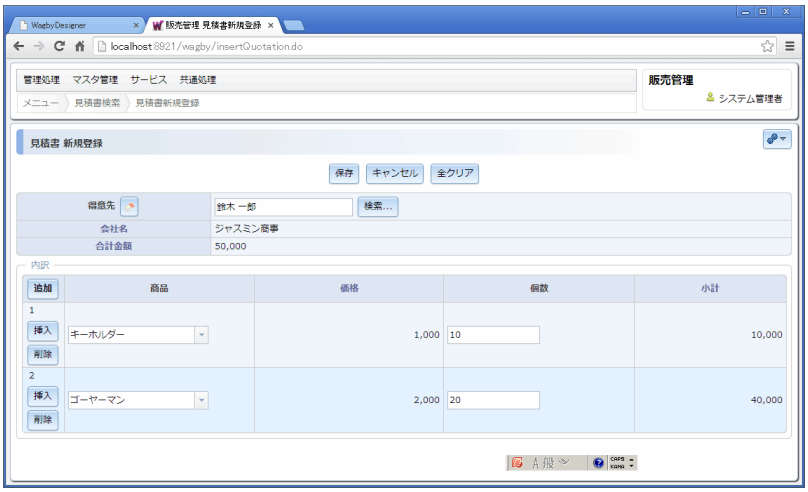


図 行が削除された

1つのモデルに、複数の「繰り返しコンテナ」を含めることができます。（上限はありません。）

定義方法

はじめに、繰り返しコンテナ型の項目を用意します。

以降、繰り返しコンテナに含まれる項目を連続で定義します。項目IDは次のようになります。

```
Container ID + "/" + Item ID
```

なお、繰り返しコンテナ型項目の直下に「繰り返しコンテナID」を必ず用意してください。図では「No」がこれに相当します。

図 繰り返しコンテナを定義する

- コンテナ名の右側の項目IDは、他の項目名と重複しないようにしてください。例えばモデル内にすでに "mdate" という項目が定義されていた場合、コンテナ内に "report/mdate" という項目を用意することはできません。
- 上に関連して、コンテナ名が異なるが、そのあとの名称が同じという設定も行えません。例えば "report1/mdate" と "report2/mdate" は、いずれも "mdate" が重複しているとみなされます。
- コンテナ内の項目に、再びコンテナを指定することはできません。具体的には "report1/report2/name" といった繰り返しの繰り返しには対応していません。
- 同様にコンテナ内の要素を繰り返しとすることもできません。具体的には "report/email" を用意した場合、この "email" を繰り返し項目とすることはできません。
- コンテナ項目は離して定義することはできません。（通常の項目を間に挟んで定義することはできません。）必ず連続して項目を定義してください。

列幅の設定

列の数が多い場合、お使いのブラウザによっては4つの列まで表示し、5つ目以降の列表示が潰れてしまうことがあります。 それぞれの列の幅を指定することで、見栄えを調節してください。

詳細は"レイアウト>繰り返しコンテナ - 列幅の調整"をお読みください。

繰り返しコンテナID

「繰り返しコンテナID」項目は必ず必要です。また、この項目を隠し設定にすることはできません。登録・更新画面で「追加」「削除」ボタンが表示されるためです。また「繰り返しコンテナID」はデータベースに保存する必要があります。データベース非保存とすると正しく動作しません。

コンテナIDの値 "1", "2", ... を隠すことはできます。詳細は"コンテナIDを非表示にする"をお読みください。

仕様・制約

繰り返しコンテナIDはデータベースには "0" から開始される連番として値が保存されています。nullは許容しません。また、飛び番号も認められません。

(52)

データベースとの関係

繰り返しコンテナは別テーブルに分割されるため、登録できるデータ数の上限はありません。詳細...

(52) Wagbyアプリケーションからデータの登録や更新を行う場合はこれらの制約を満たすように内部では動作していますので問題ありません。外部システムから直接、繰り返しコンテナへ値をセットする場合や、スクリプトを用いて繰り返しコンテナを操作する場合は、この制約を満たすようにしてください。

繰り返しコンテナ － 初期表示数

例

繰り返しコンテナの初期表示数を〝3〟とした例を図に示します。

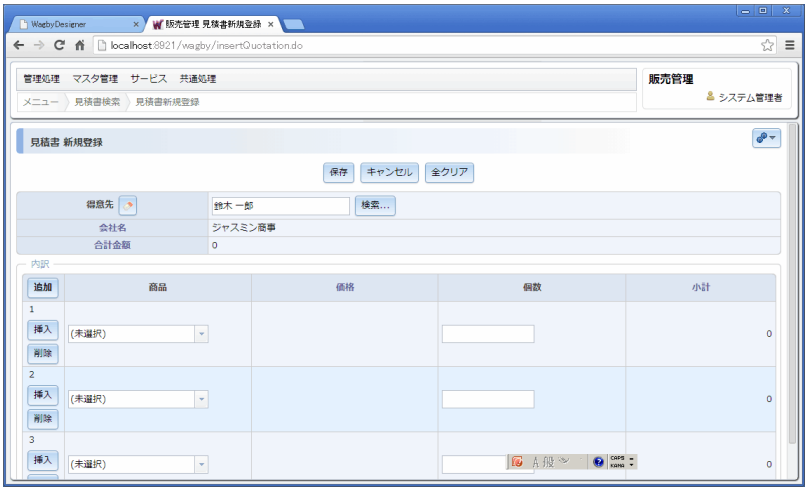


図 初期表示数3の例

定義方法

繰り返しコンテナ型の項目の詳細を設定します。 モデル項目詳細定義ダイアログの「入力制御」タブ内にある「繰り返し － 初期表示数」に数字を記述します。

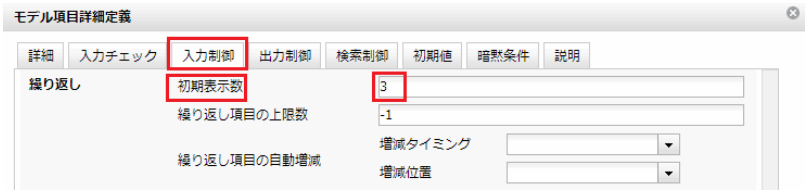


図 初期表示数の設定

初期表示数を0にする

初期表示数を「0」とすると、新規登録画面を開いた時には項目名のみが表示されます。

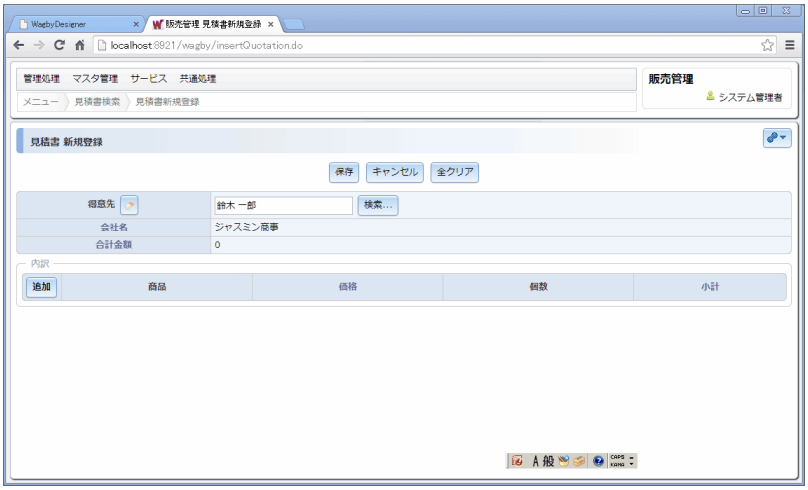


図 初期表示数が0の例

初期表示数を式で設定する

例

初期表示数として、今月の日数（2014年2月の場合、28日）を指定した例です。28個の繰り返しコンテナが用意されています。

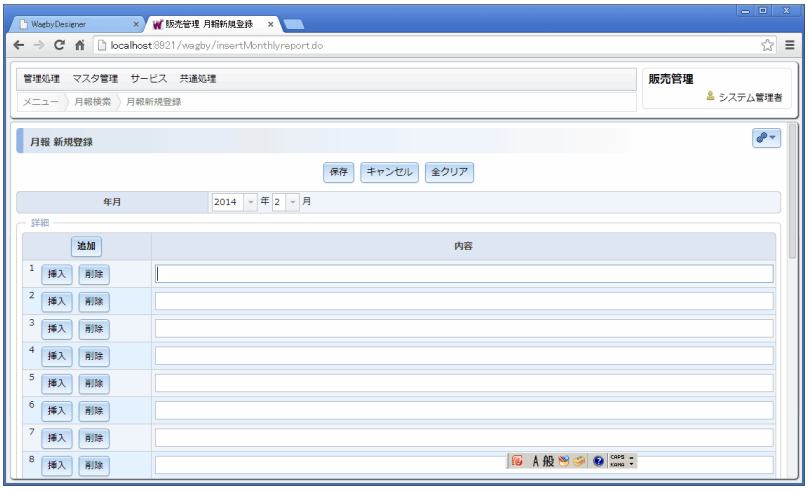


図 今月の日数を初期表示数に設定

定義

初期表示数に式を記述することができます。次の式は、mdate項目の月末の日付の日数を返します。

```
DAY(EOMONTH(${mdate}))
```

図 初期表示数に式を記述する

(53) (54) (55)

登録・更新画面の動作（8.1.0）

トリガーとなる項目の「値が変更された画面を再描画させる設定にする」を有効にすると、登録・更新画面でも（画面再描画のタイミングで）初期表示数が変化します。次のルールが適用されます。

- 初期表示数は固定値ではなく、式を記述することができます。
- 式の結果と、現時点のコンテナ行数を比較し、式の結果が大きい場合、不足行を補完します。補完する行には「初期値」が適用されます。
- 式の結果と、現時点のコンテナ行数を比較し、式の結果が小さい場合、行を削除します。削除されるのは「後ろから」となります。

登録・更新時の初期表示数（8.1.1）

″入力制御 > 繰り返し > 登録・更新時″に値または式を設定することができます。上記「登録・更新画面の動作」に似ていますが、画面再描画のタイミングではなく、登録および更新時に行数を調整します。

仕様・制約

- 繰り返しコンテナ中に他モデルのコンテナを参照する参照連動項目を1つでも定義した場合、初期表示数は 0 に固定されます。

繰り返しコンテナ - 繰り返し項目の上限数

例

繰り返しコンテナの上限数を ″3″ とした例を図に示します。

3つめの繰り返しコンテナ入力欄が表示された時点で、「追加」「挿入」ボタンを押下することはできなくなります。

(53) mdate項目は「年月」を格納する項目として用意しています。
(54) EOMONTHは月末の日付を返す関数です。
(55) DAYは日数を返す関数です。

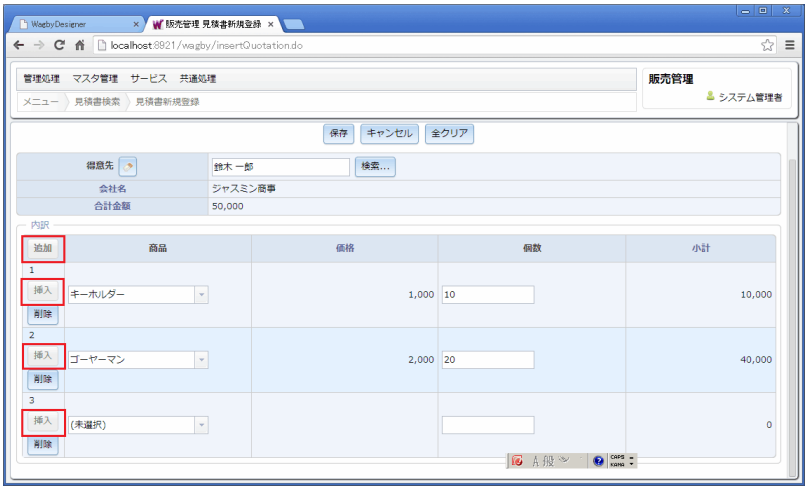


図 上限数を設定

定義方法

繰り返しコンテナ型の項目の詳細を設定します。 モデル項目詳細定義ダイアログの「入力制御」タブ内にある「繰り返し - 繰り返し項目の上限数」に数字を記述します。

図 上限数の設定

この値が空白または ″-1″ の場合、上限なし、となります。

仕様・制約

- R8.3.1まで、繰り返しコンテナに含まれるファイル型項目に複数のファイルを直接、ドラッグ&ドロップして登録したとき、上限数チェックが適用されません。R8.3.2で適用されるようになります。この場合、画面操作では登録可能なように動作しますが、保存ボタン押下時にエラーチェックがかかります。

繰り返しコンテナ - 追加時に値をコピーする

例

繰り返しコンテナ「商談履歴」の「内容」は、追加時にコピーするという設定としています。 図において、追加ボタンを押下すると、図のように「内容」に（直前の行の）コピーされた値が設定されます。

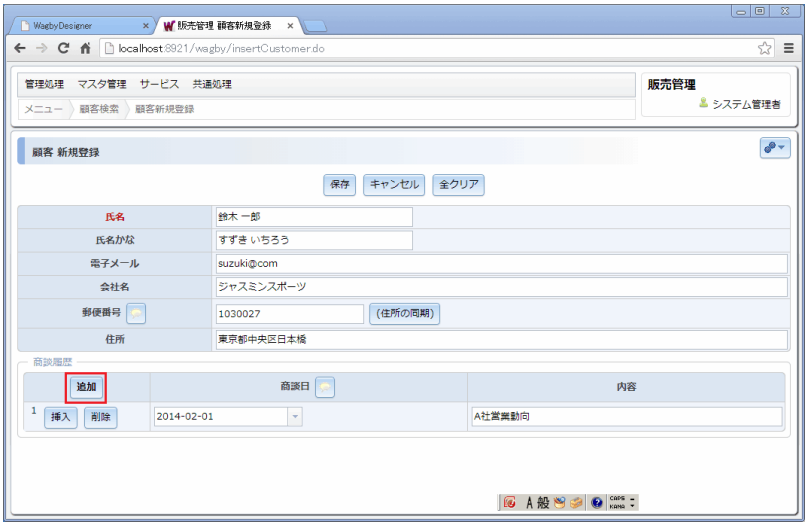


図 追加ボタンを押下する

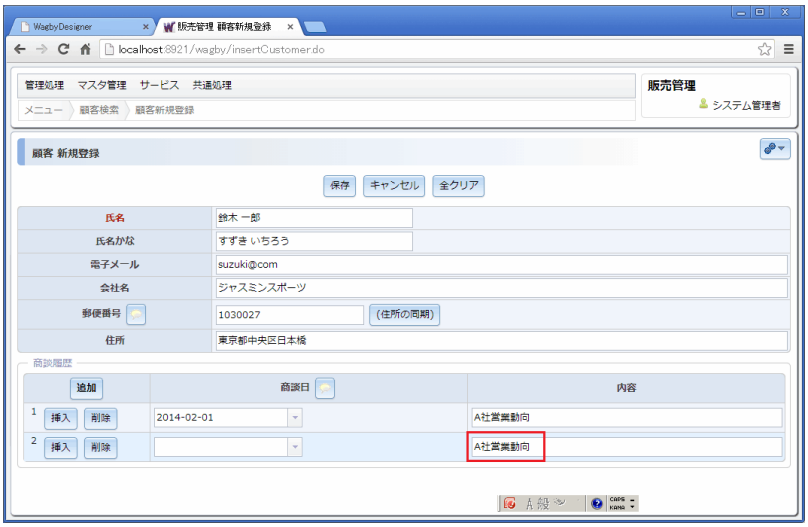


図 「内容」がコピーされた

定義方法

繰り返しコンテナ「商談履歴」を用意します。

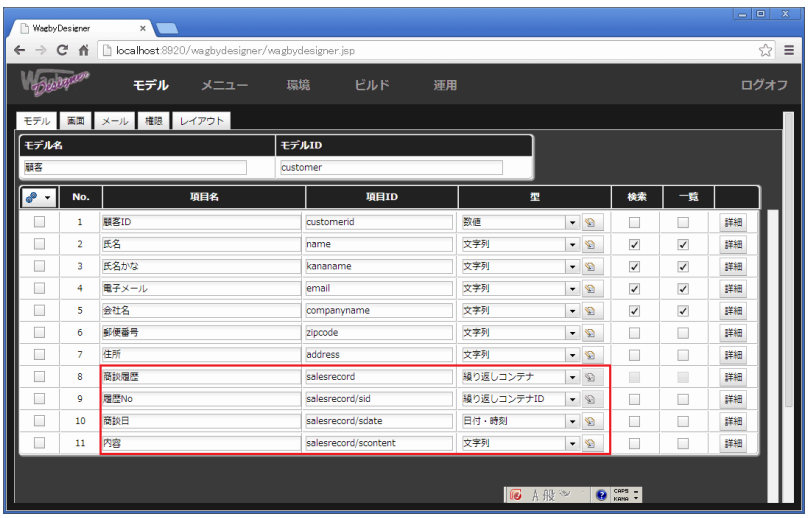


図 モデル定義

繰り返しコンテナ型の項目の詳細を設定します。 モデル項目詳細定義の入力制御タブにある繰り返しの設定で「追加時に値をコピーする」をチェックします。

図 追加時に値をコピーする

繰り返しコンテナ内の複数の項目をコピーさせたい場合は、すべての項目について図の設定を行います。

繰り返しコンテナ - ソート

定義方法

ソートの対象としたい、繰り返しコンテナ内の項目の「モデル項目詳細定義」を開きます。 入力制御タブにある繰り返しの設定で「コンテナ内でソートする」を選択します。 標準では空白となっており、ソートは行われません。

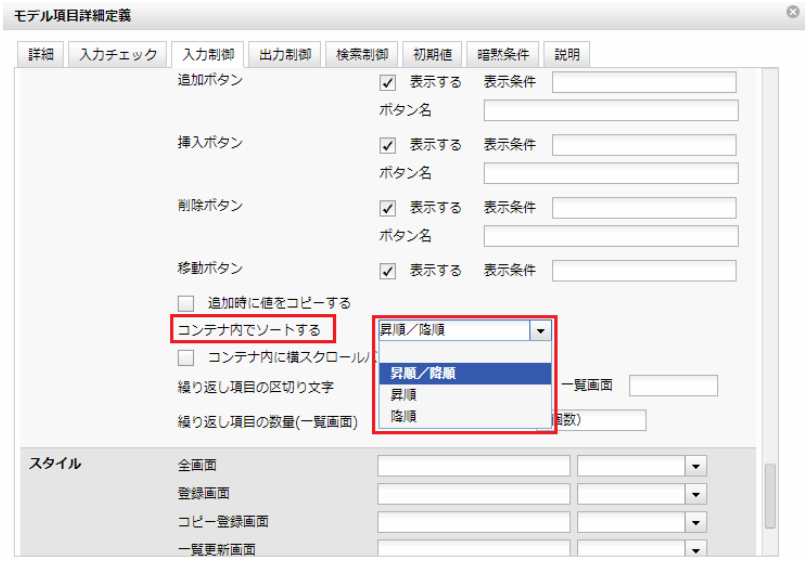


図 コンテナ内でソートする設定

以下の項目はソート対象に指定することはできません。

- 繰り返しコンテナID

・　他モデルの参照（チェックボックス）
・　他モデルの参照（参照連動）
・　計算式によって求められる項目

実行時にソートルールを決める

「昇順/降順」を選択すると、実行時に昇順と降順を切り替えることができます。図のように、ソートルールを指定するための矢印型のアイコンが表示されます。

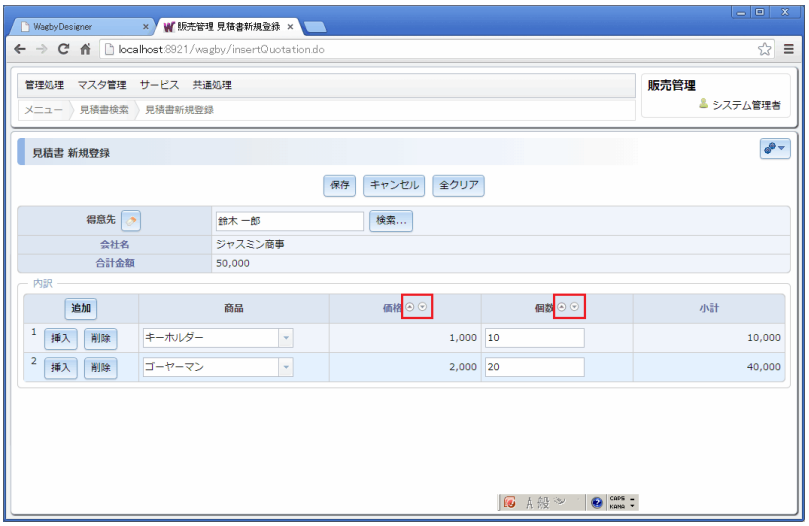


図　ソートルールを可変とする

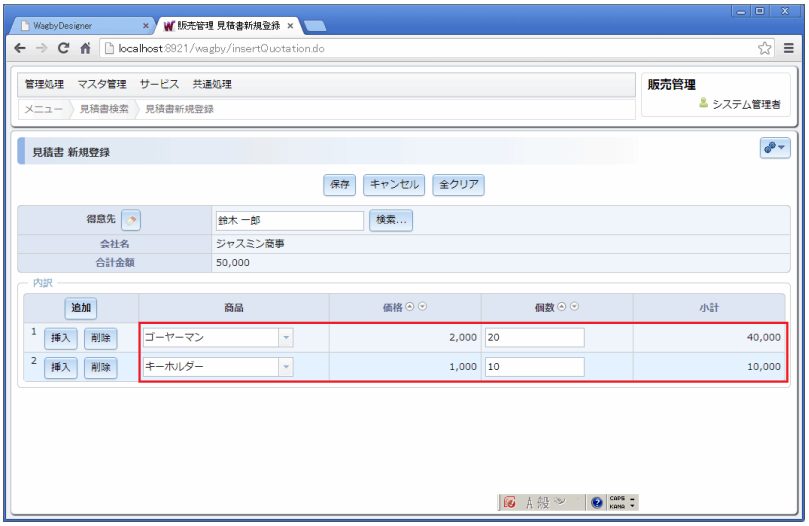


図　アイコンをクリックすることでソートされる

ソートルールを固定にする

図で「昇順」または「降順」を選択すると、コンテナの追加・挿入・削除のタイミングで自動的にソートされます。 図, 図は個数の昇順という固定ルールを適用した例です。

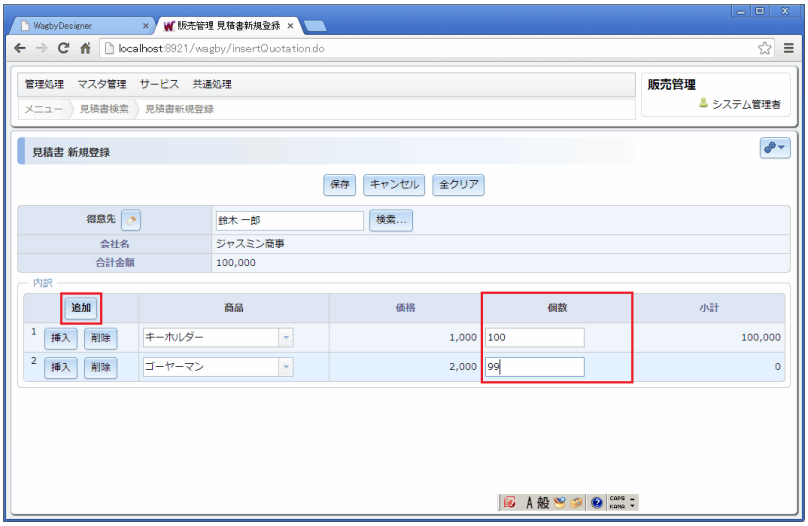


図 個数を入力して「追加」ボタンを押下する直前

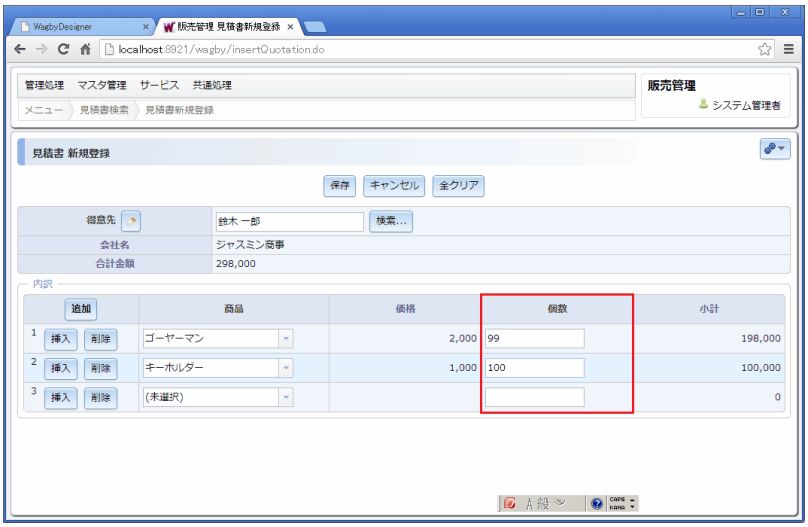


図 「追加」直後、ソートされた

「昇順/降順」と「昇順」または「降順」を同時に設定することはできません。実行時にソートルールを決める方式とするか、固定ソートルールとするか、いずれかとしてください。

追加時の動作の詳細

繰り返しコンテナにソートを設定した場合に追加を行なうと、既存のコンテナ行に対してソートを行なった後、新規行がコンテナの最終行に追加されます。　このため、ソート項目が空の行は追加ボタン押下時にソートされ、先頭に移動するようになります。 ソート項目に値を設定後、保存すると全てソートされた状態で表示されます。

繰り返しコンテナ － 並びを入れ替える

例

登録や更新時に、繰り返しコンテナの並びを入れ替えることができます。図に示すように、並び替えのためのボタンが用意されます。

図 コンテナ内を並び替えるボタン

ボタンをクリックすると、前後の行が入れ替わります。

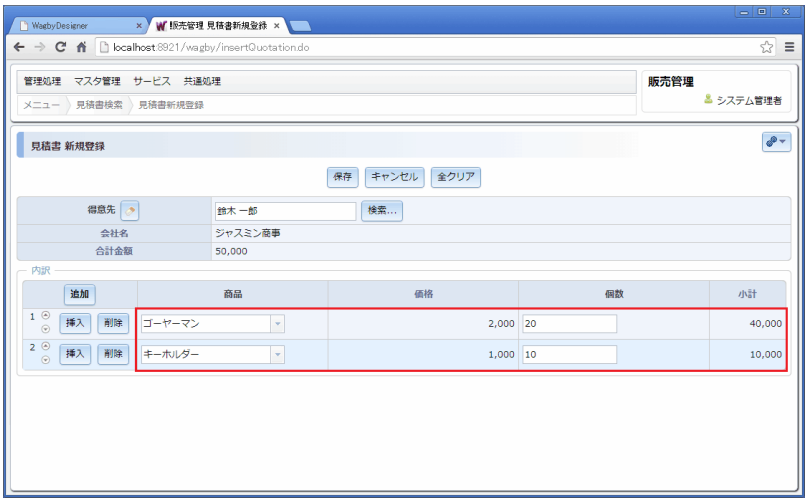


図 前後の行が入れ替わる

定義方法

繰り返しコンテナ型の項目のモデル項目詳細定義を開きます。 入力制御タブ内の繰り返し設定で、移動ボタンを表示する機能（チェック）を有効にします。標準では有効になっています。

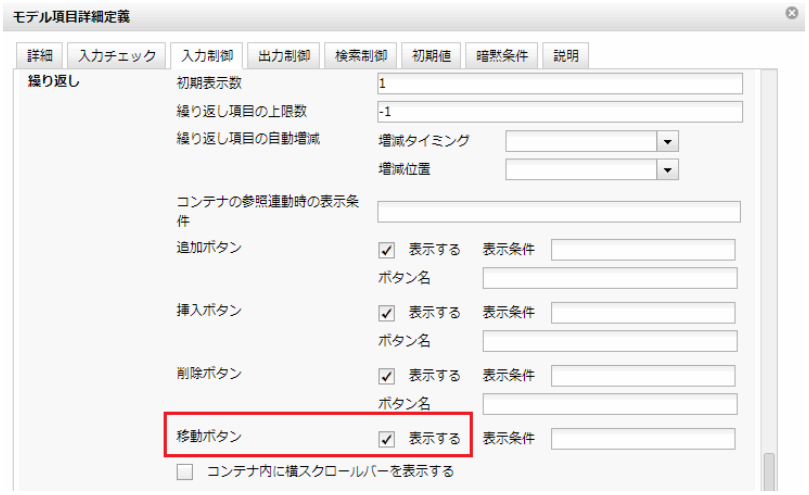


図 移動ボタンを表示する

式による制御

「画面機能>ボタン名と表示条件のカスタマイズ」で説明しています。

繰り返しコンテナ － 操作ボタンの表示制御

ボタン名を変更する

例

繰り返しコンテナの追加と挿入ボタンを「+」、削除ボタンを「-」とした例を図に示します。

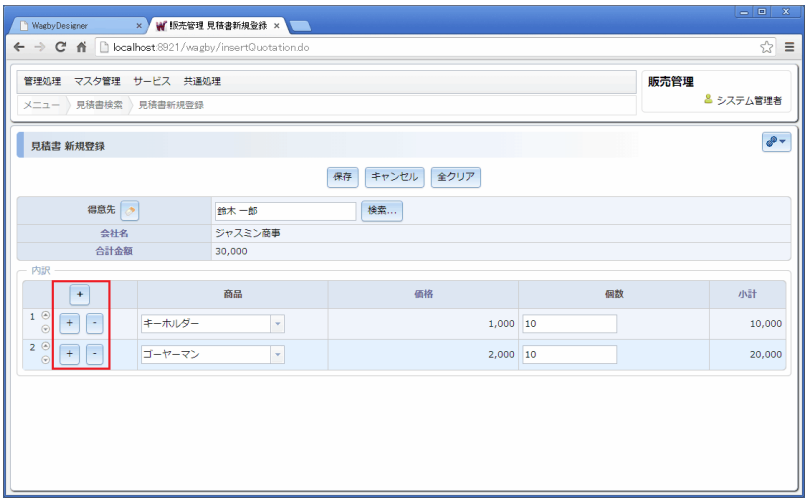


図 ボタン名を変更した例

定義方法

繰り返しコンテナ型の項目のモデル項目詳細定義を開きます。 入力制御タブ内の繰り返し設定で、追加・挿入・削除のボタン名を指定することができます。

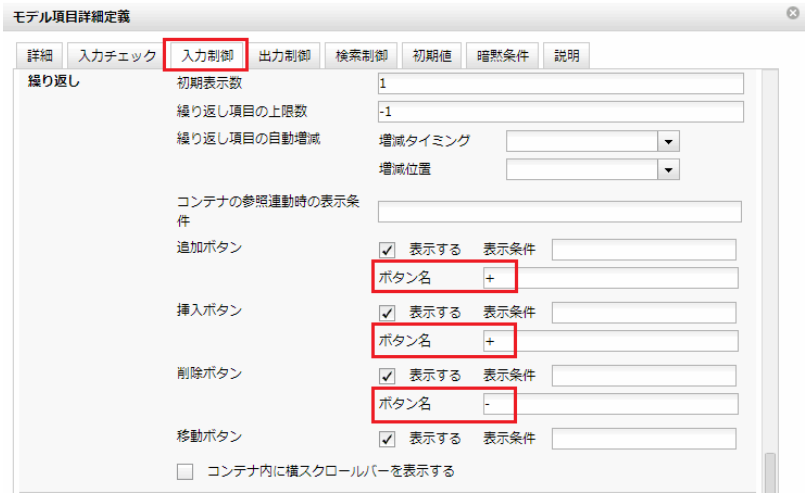


図 ボタン名の設定欄

ボタンを非表示にする

入力制御タブ内の繰り返し設定で、追加・挿入・削除・移動ボタンを表示しないという指定を行うことができます。

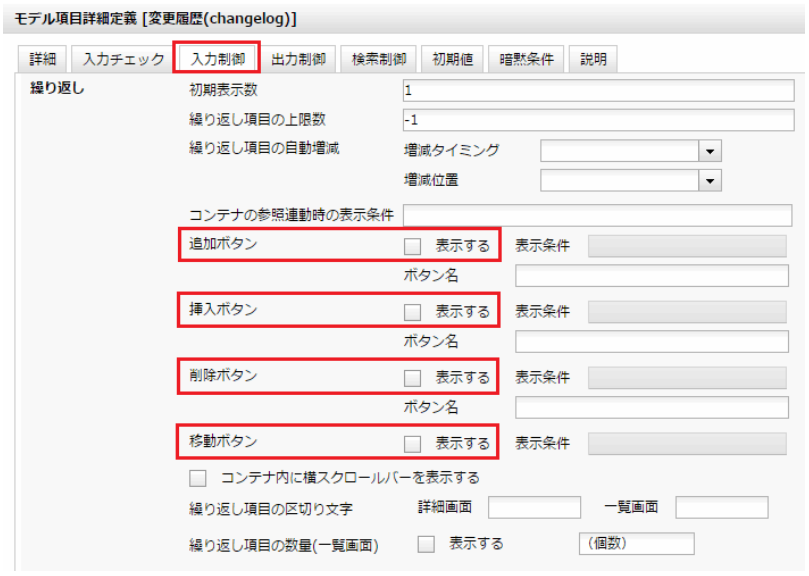


図 ボタンを表示しない

式による制御

「画面機能>ボタン名と表示条件のカスタマイズ」で説明しています。

繰り返しコンテナ - コンテナIDを非表示にする

例

「繰り返しコンテナID」項目を隠し設定にすることはできません。登録・更新画面で「追加」「削除」ボタンが表示されるためです。

コンテナIDの値 "1", "2", ... を隠すことはできます。図は繰り返しコンテナID部（数字）を非表示にした例です。

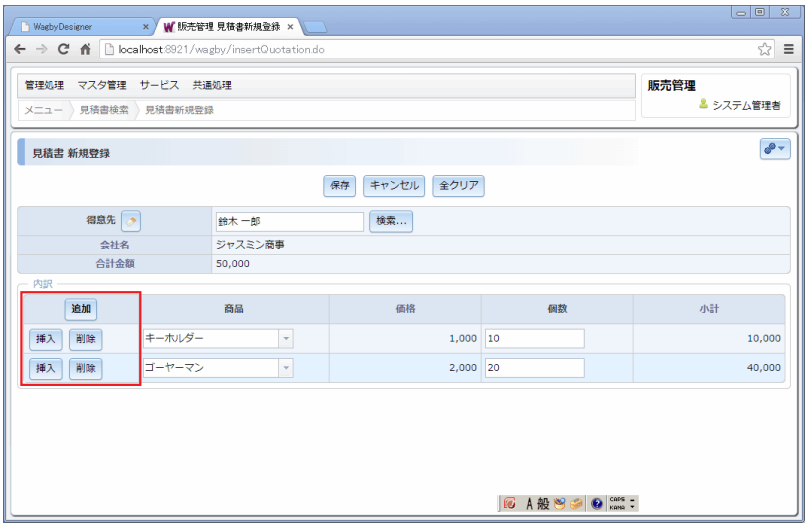


図 登録画面でコンテナIDが表示されていない

定義方法

繰り返しコンテナID項目の「モデル項目詳細定義」ダイアログを開きます。 入力制御タブ内にあるスタイルの設定で、登録画面その他に次の設定を行います。

```
display:none;
```

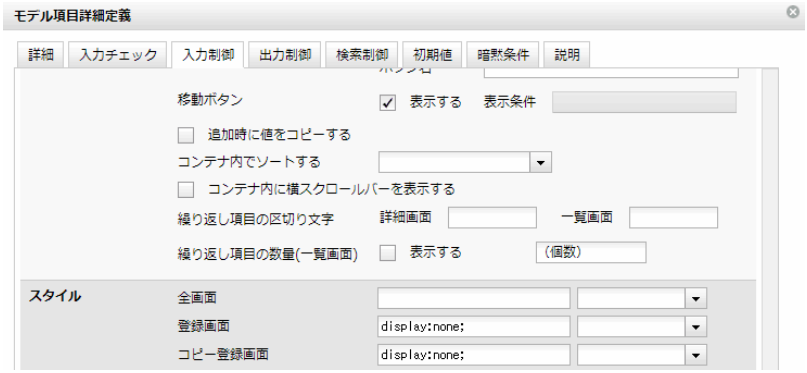


図 スタイルの設定

複数段一行明細利用時の注意点

複数段一行明細レイアウト利用時は、詳細表示画面で繰り返しコンテナID項目を非表示とすることはできません。 （全体の列が一つ左にずれてしまい、レイアウトが崩れます。）

仕様・制約

この設定は登録・更新画面で有効です。詳細画面には適用されません。

繰り返しコンテナ － 更新のタイミングで別の繰り返しコンテナの同期増減を行う

例

ある繰り返しコンテナに対して挿入・追加・削除を行うと、同一画面内の別のコンテナが同時に挿入・追加・削除されるという例を紹介します。

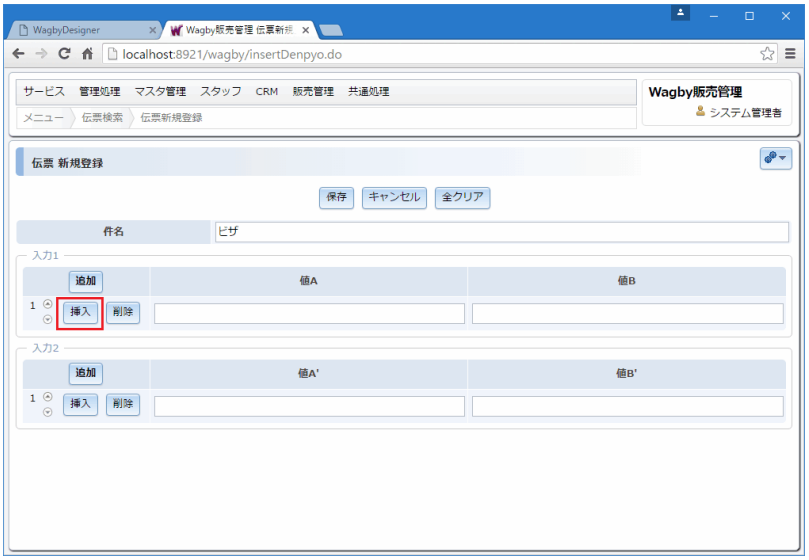


図 繰り返しコンテナの追加前

″入力1″のコンテナを追加すると、同じタイミングで ″入力2″ のコンテナも追加されます。

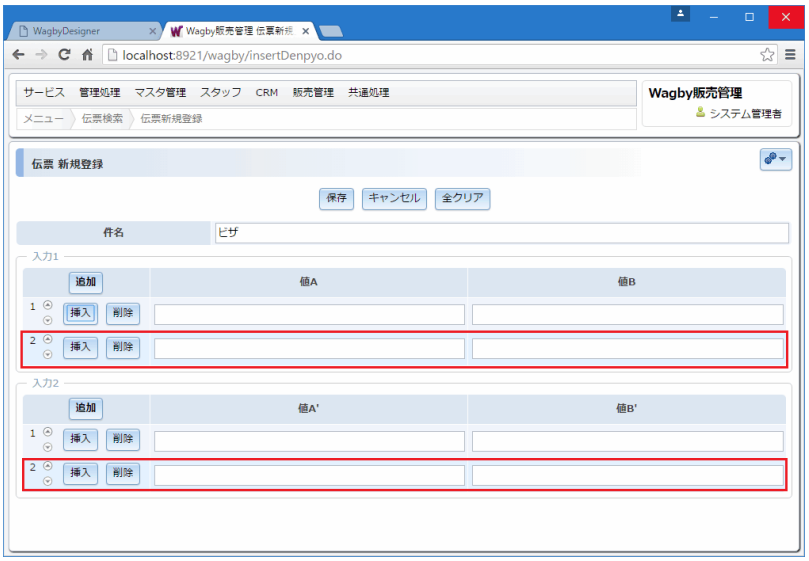


図 繰り返しコンテナの追加後

削除の同期

削除機能を同期させることもできます。″入力1″ のコンテナを削除すると、同じタイミングで ″入力2″ のコンテナも削除されます。

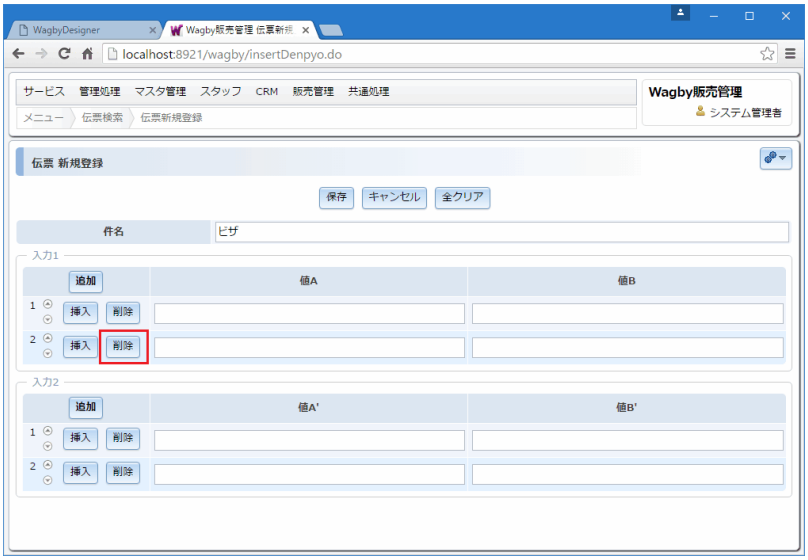


図 削除の同期(1)

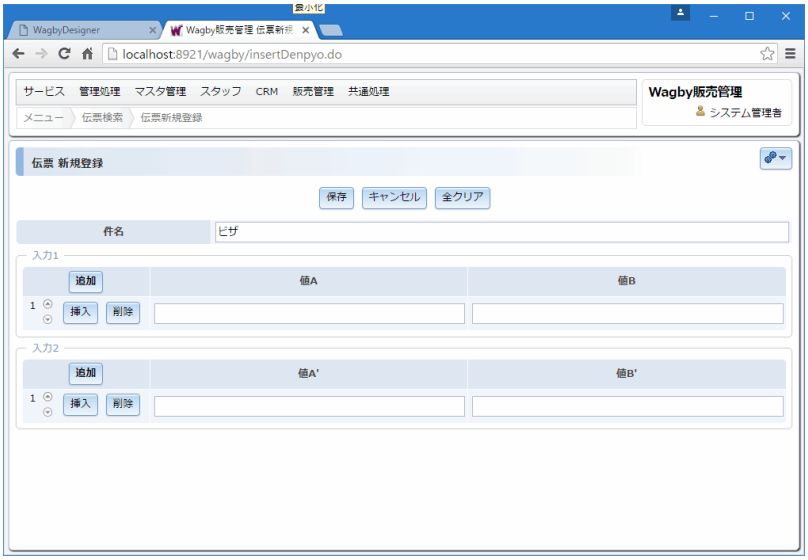


図 削除の同期(2)

定義方法

"入力1" と "入力2" という二つの繰り返しコンテナを用意します。

"入力2"の繰り返しコンテナの詳細を定義します。

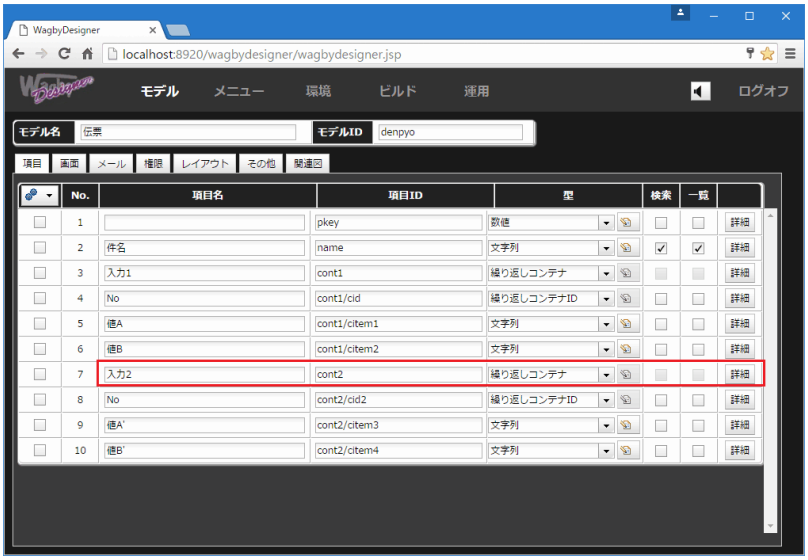


図 二つの繰り返しコンテナを定義する

繰り返しコンテナ"入力2"の「入力制御 > 繰り返し > 繰り返し項目の同期増減」を指定します。

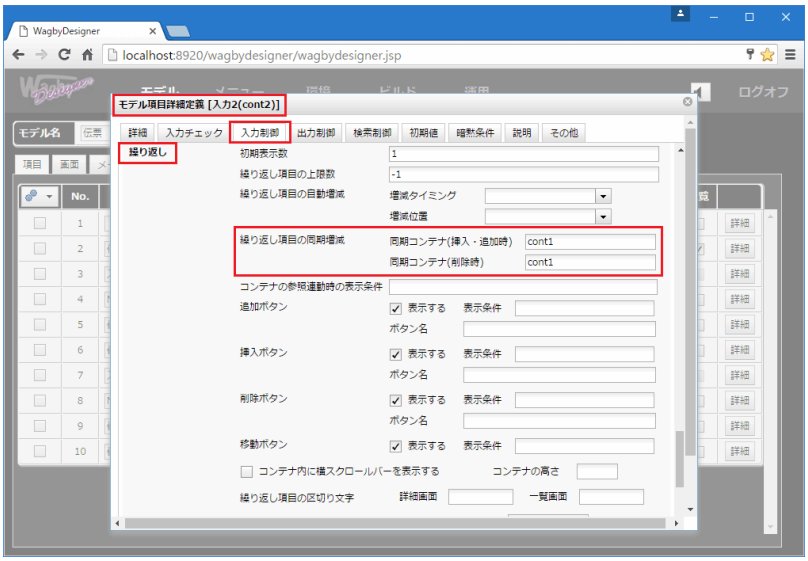


図 同期対象の繰り返しコンテナを指定する

図では「追加・挿入」と「削除」のいずれも "cont1" と記述しているため、追加・挿入・削除すべてが同期します。

繰り返しコンテナのコピー入力

値コピー機能と併用することで、"入力1" に値を入力すると、連動して "入力2" に値が転記されます。

図では、"入力1" 側に値をセットしました。ここでカーソルが次の入力欄に移動したタイミングで、"入力2" 側に同じ値がコピーされます。

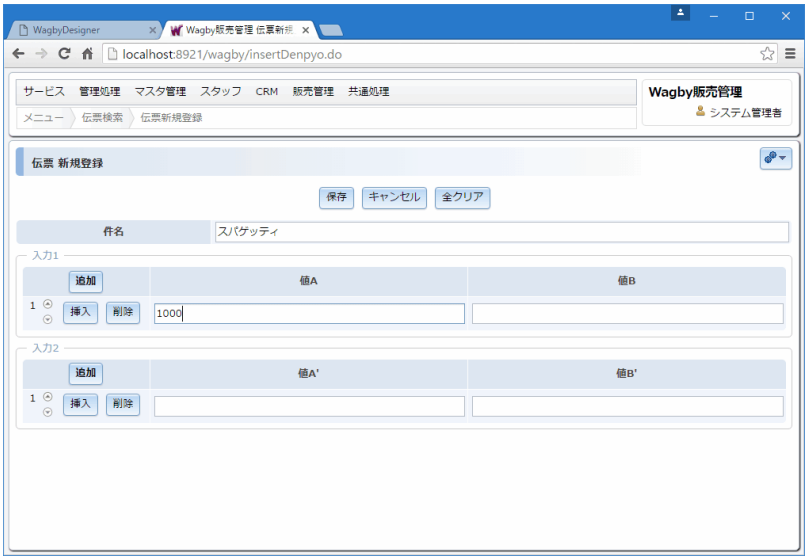


図 値の転記(1)

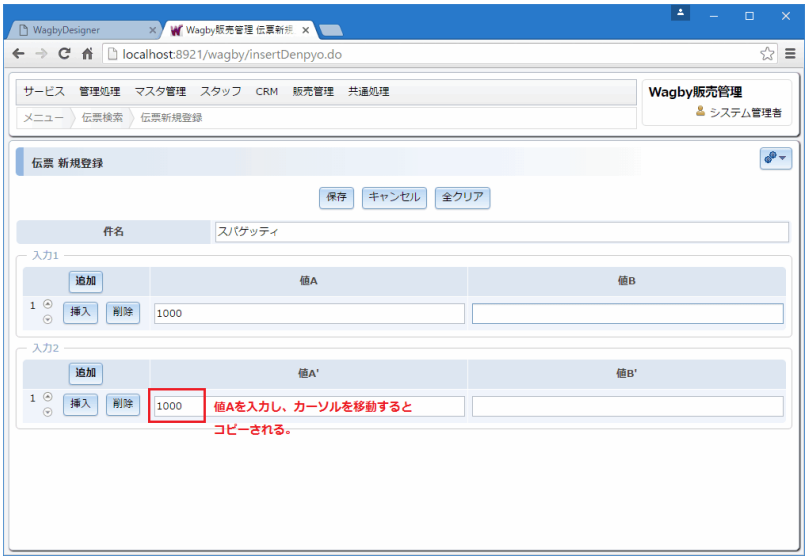


図 値の転記(2)

定義方法

「別項目の値をコピーして利用する」に、コピーする項目名を記載します。　図は、入力2側の「値A'」項目の設定です。

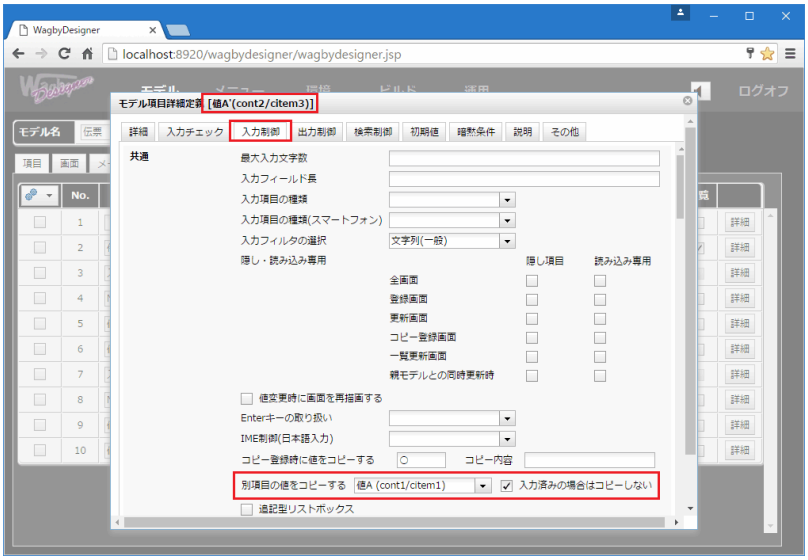


図 別項目の値をコピーして利用する設定

(56)

繰り返しコンテナ － 折りたたみ

例

この設定を行うと、繰り返しコンテナが折りたたむことができるようになります。

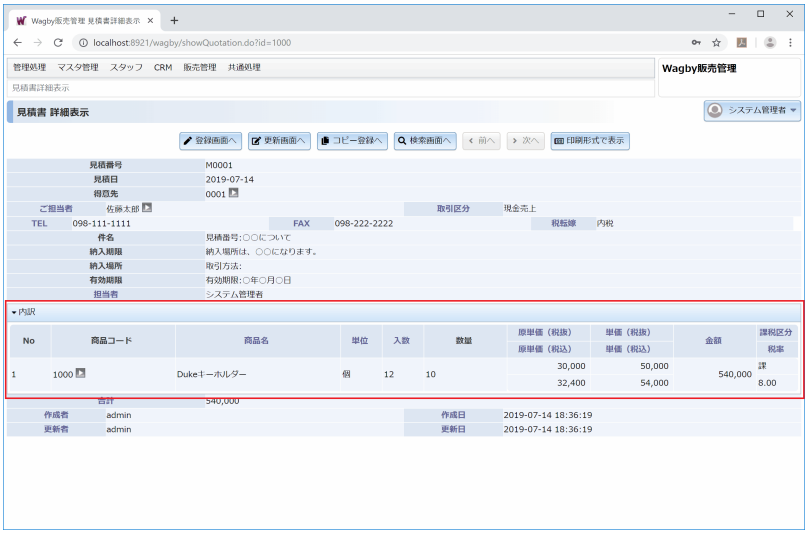


図 繰り返しコンテナ折りたたみ(1)

(56) 転記できるのは文字列型または数字型項目です。詳細は"入力制御 > 別項目の値をコピーする"をお読みください。

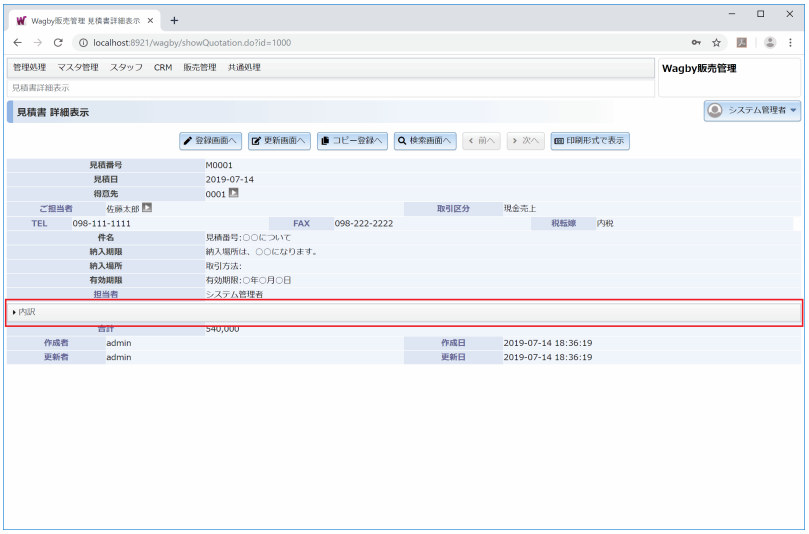


図 繰り返しコンテナ折りたたみ(2)

定義方法

″入力制御 > 繰り返し″ の ″コンテナを折り畳めるようにする″ を有効にします。

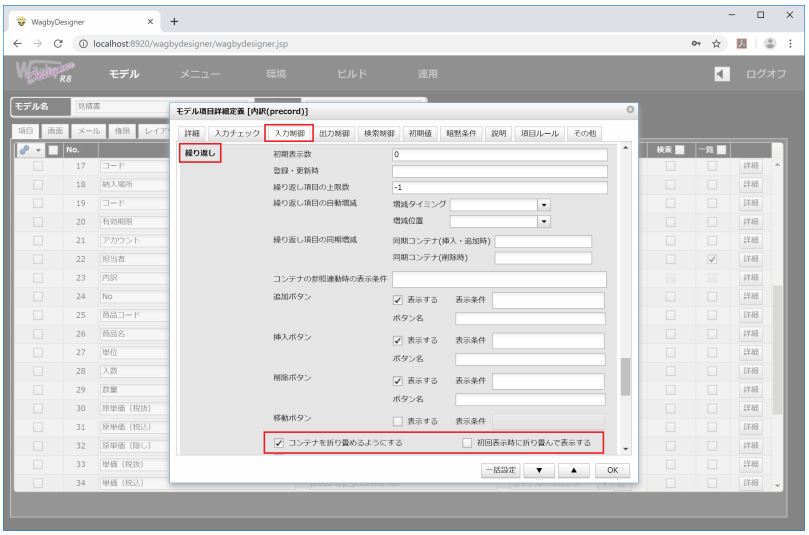


図 コンテナを折り畳めるようにする

″初回表示時に折り畳んで表示する″ を有効にすると、最初にこのモデルの登録画面時に折りたたまれた状態で表示されます。

(57)

繰り返しコンテナの参照連動

例

見積書モデルを用意しました。これを参照連動の元とします。

(57) 折り畳みされたかどうかという状態は登録、更新、詳細画面ごとに（ブラウザの）クッキーに保存されます。このため、コンテナの折り畳みを開いた状態で画面遷移すると、この画面を次に表示したとき、コンテナの折り畳みは開いた状態で表示されます。つまり″初回表示時に折り畳んで表示する″ を有効にしても、登録画面表示時に毎回、折り畳まれて表示されるということではありません。

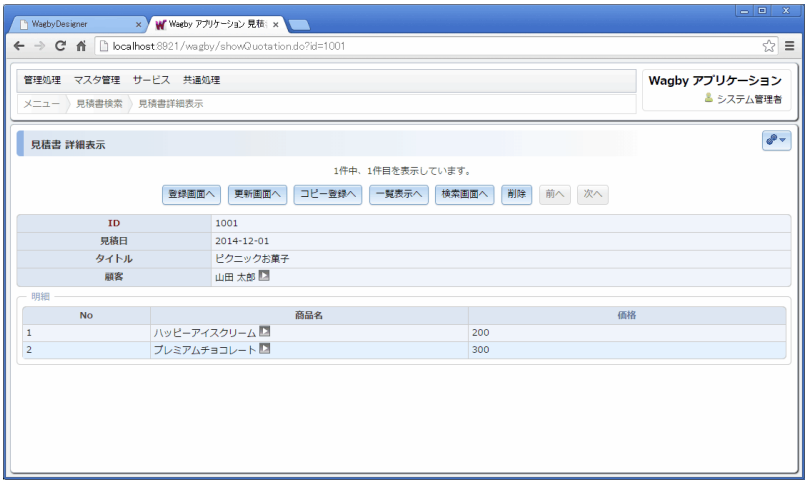


図 見積書モデル

この見積書モデルを参照する、注文書モデルを用意しました。見積書を選択しています。

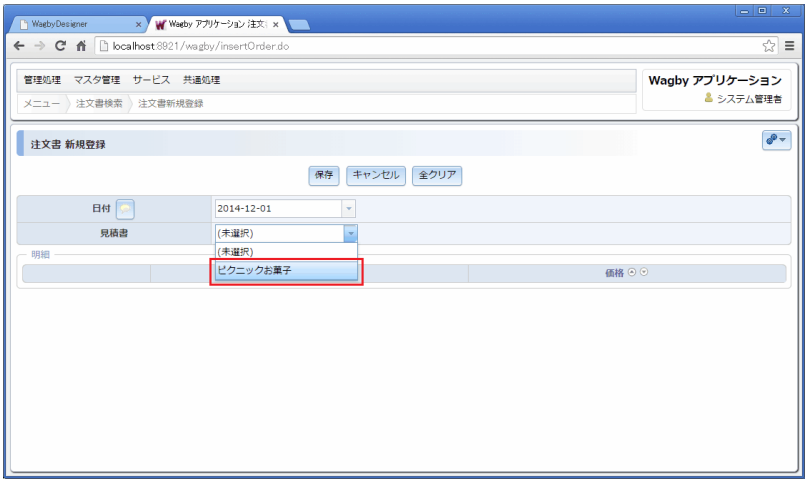


図 注文書モデル

紐づく（見積書）モデルが確定されると、繰り返しコンテナ全体が参照連動されます。

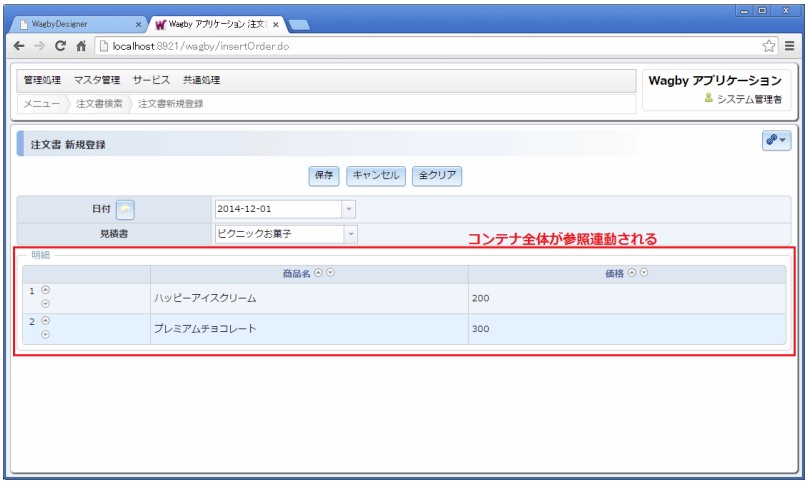


図 繰り返しコンテナの参照連動

この動作イメージを図に示します。

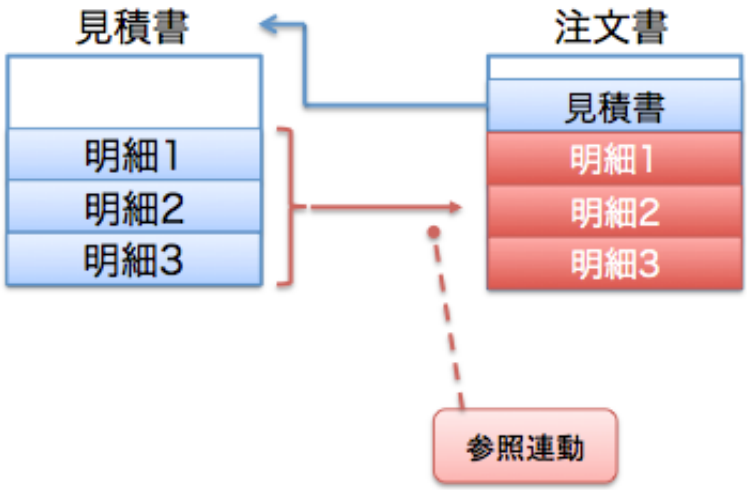


図 動作イメージ

定義方法

4つのモデルを定義しています。「顧客」と「商品」はマスタデータとして扱います。（説明を省略します。）

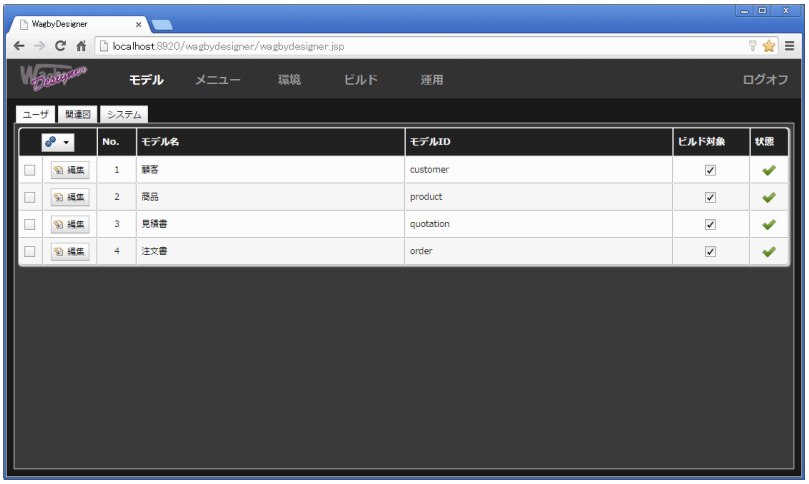


図 顧客・商品・見積書・注文書モデル

見積書モデルの定義を示します。「明細」は繰り返しコンテナになっています。

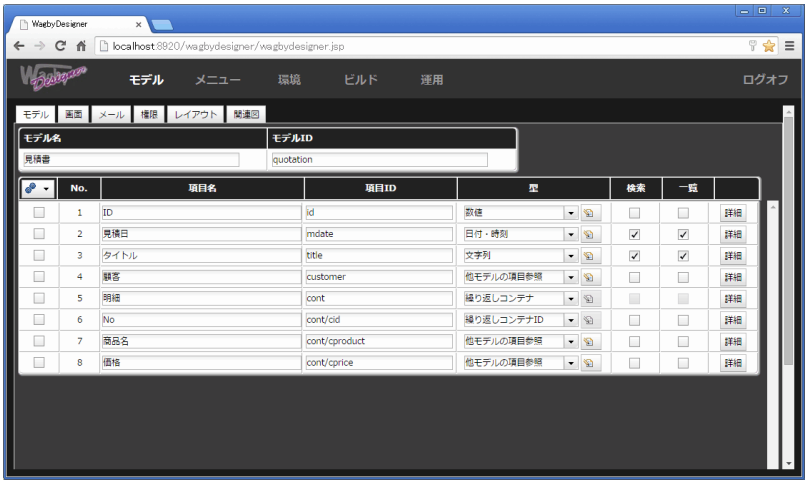


図 見積書モデルの定義

注文書モデルの定義を示します。ここで「明細」モデルに含まれる「商品名」「価格」を ″参照連動″ としています。

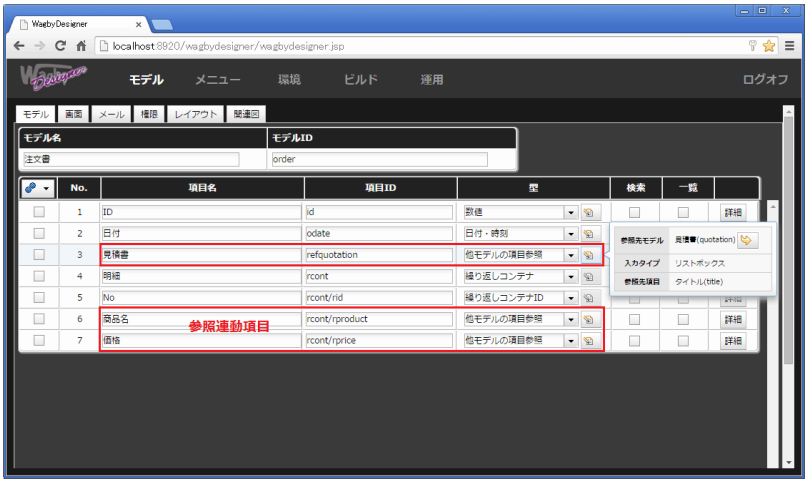


図 注文書モデルの定義

繰り返しコンテナ内の項目「商品名」は、見積書項目が確定したら（見積書モデルに含まれる）商品名項目を参照連動するように設定します。

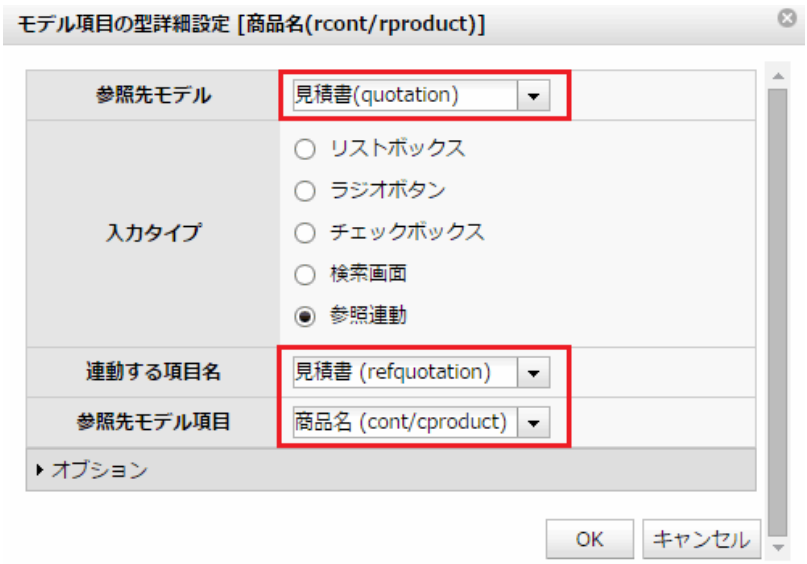


図 商品名項目の参照連動の設定

同じように「価格」も参照連動の設定を行います。

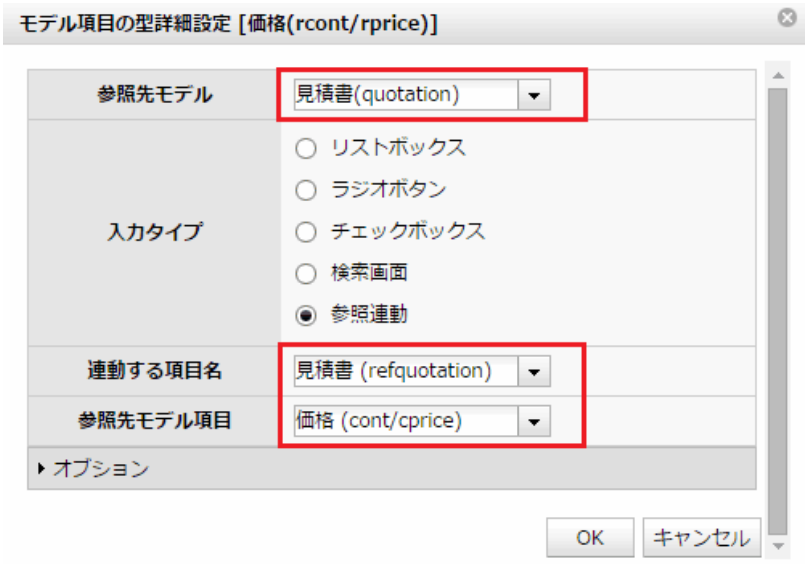


図 価格項目の参照連動の設定

画面再描画の設定

本機能を利用する場合、繰り返しコンテナ内に少なくとも一つのリストボックス/ラジオボタン/チェックボックス項目を含めるか、または連動元のモデルを選択する（モデル参照）項目に「この項目が変更されたら画面リロードする」を設定してください。画面再描画によって参照連動を行わせる必要があるためです。

今回のケースでは、繰り返しコンテナ内にリストボックス項目が含まれるため、本ルールを満たしています。

参照連動を自モデル保存する

参照連動項目を自モデル保存とすることで、参照後に値を変更して保存することもできます。 書き換えた値は、参照元とは別個に管理されます。

例

図のアプリケーションを、参照連動自モデル保存に変更した例を示します。

値を変更して保存することができます。その後、紐づく見積書モデル側の明細を変更しても、この注文書モデルには影響はありません。

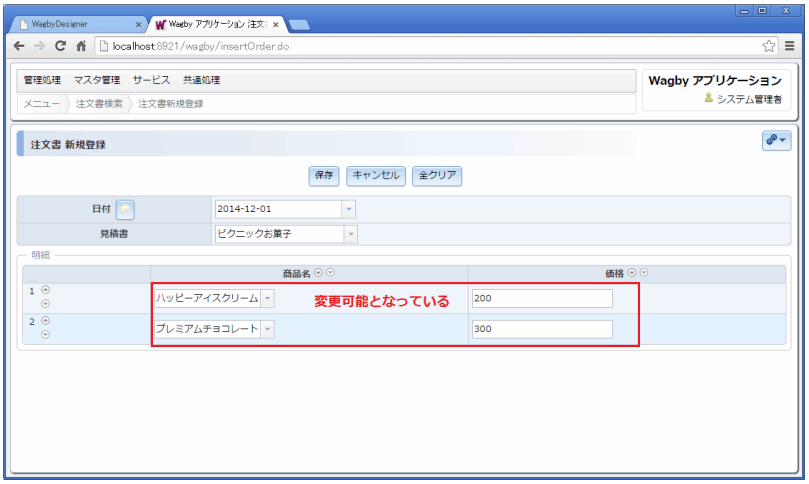


図 参照連動自モデル保存の例

(58)

定義方法

自モデル保存する項目は「値をデータベースに保存する」を有効にします。 また、「読込専用」の設定を″解除″します。

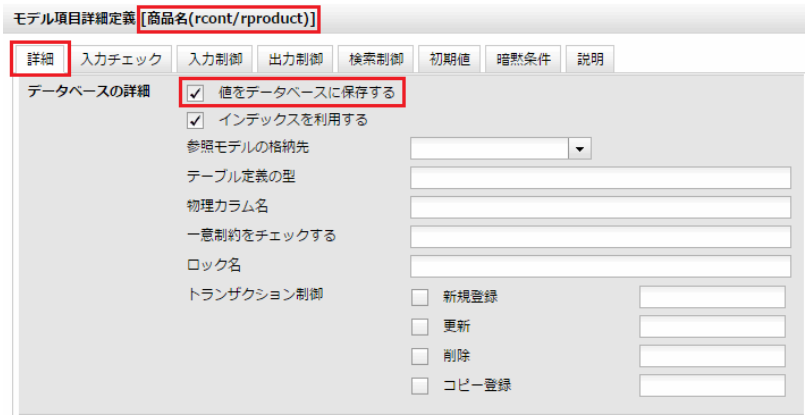


図 値をデータベースに保存する

(58) 図で、繰り返しコンテナ全体が参照連動されたあと、「商品名」を変更しても「価格」は変わりません。「価格」はあくまでも「見積書」の参照連動であり、「商品名」と紐づいているわけではありません。

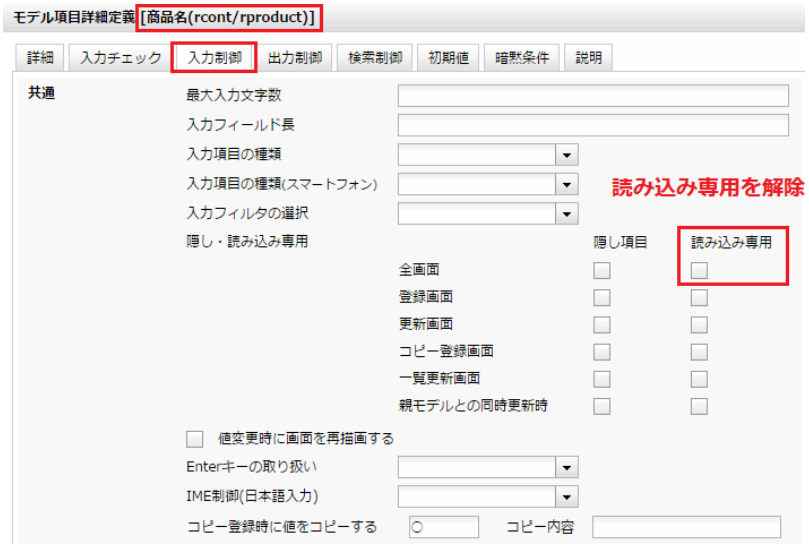


図 読込専用の設定を解除する

仕様・制約

参照連動の保存を行う場合、次のルールが適用されます。

- コンテナ初期数はゼロに設定されます。
- コンテナの追加、削除、挿入処理は行えません。

条件によって参照連動するデータを絞り込む

参照元となるモデルの繰り返しコンテナに対して、条件式を指定して（参照連動するデータを）絞込むことができます。

例

価格が100(円)より大きい場合のみ参照連動する、という条件を付与した例を示します。 見積書には3件の明細を含めています。

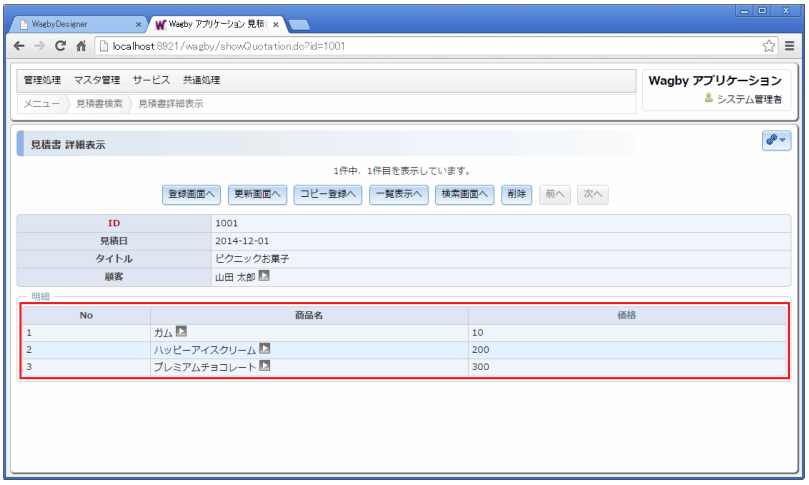


図 見積書

注文書の新規登録画面では2件の明細が参照連動されました。

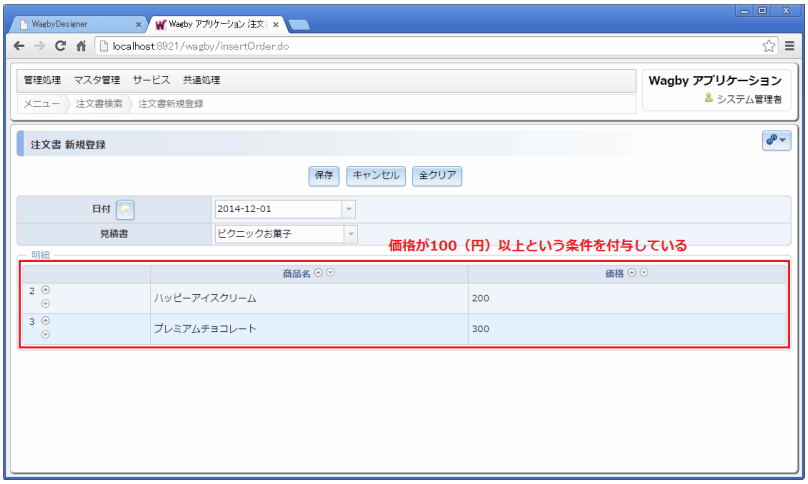


図 注文書

定義方法

「コンテナの参照連動時の表示条件」に式を記述します。この式がtrueとなった行のみが参照連動されます。

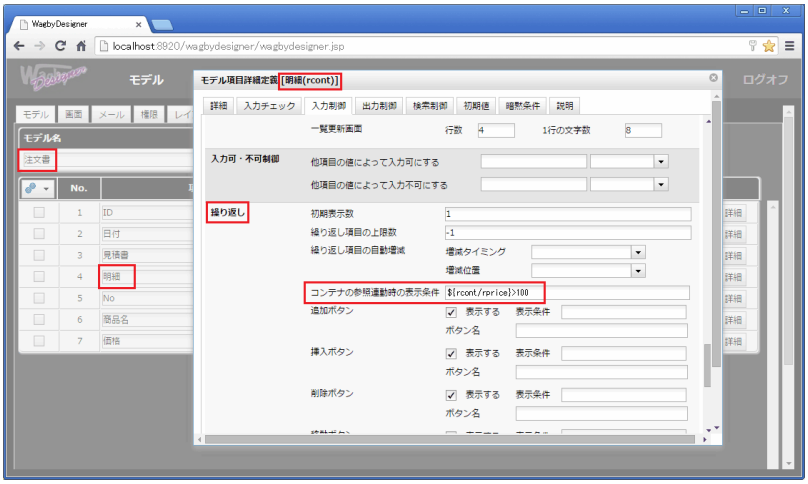


図 コンテナの参照連動時の表示条件の設定

ここでは式を次のように設定しています。

```
${rcont/rprice}>100
```

仕様

- 繰り返しコンテナID項目の次の項目は、参照連動項目となっている必要があります。そうでない場合、式は無視されます。（上の例では"No"の次の項目"商品名"が参照連動項目となっているため、この設定は有効となっています。）

式による連動（8.3.0）

表示条件の式に繰り返しコンテナ外にあるモデル参照項目が含まれていた場合、その項目の値が変更されたタイミングで画面再描画が行われると、変更された値を反映した参照連動の絞り込みが行われます。

例

繰り返しコンテナ meisai の"コンテナの参照連動時の表示条件"に次の式を設定したとします。

```
${meisai/seibetsu_no#id}==${seibetsu_list#id}
```

ここではコンテナ内項目 meisai/seibetsu_no の ID部と、コンテナ外項目 seibetsu_list の ID 部が一致するもので絞り込むとしており、両方ともモデル参照項目です。また seibetsu_list 項目は ″値変更時に画面を再描画する″ を有効としています。

このとき Wagby は式を解析し、seibetsu_list 項目が変更（かつ再描画）されたときも繰り返しコンテナ meisai の参照連動を行うようになります。そのため、コンテナ外項目 seibetsu_list の選択肢を変更すると、繰り返しコンテナ参照連動の内容が書き換わるという動きを実現できます。

トラブルシューティング

繰り返しコンテナの参照連動が動作しない

画面再描画の設定ルールを満たしているかどうかをご確認ください。

先頭の行だけが参照連動されない !-- N20180828_120147 -->

次の例をもとに説明します。

想定するモデル構成

- 親子関係のモデルがあり、親モデルのコンテナを子モデルで参照連動する。
- 親モデルの繰り返しコンテナ内にファイル型項目がある。
- 親モデルで繰り返しコンテナ内のファイル型項目が変更されても、子モデルでは元のファイルを維持させるために、参照連動した（繰り返しコンテナ内の）ファイル型項目はデータベースに保存する設定にする。

処理の流れ

1. 親モデルの登録画面を開く。
2. 繰り返しコンテナのデータが1行存在し、ファイル型項目が空（添付ファイルなし）の状態で親のデータを保存する。
3. 続いて、子モデル登録画面を開く。
4. そのまま保存する。保存されたデータは親モデルのコンテナを参照連動しているため、添付ファイルなしの1レコードのみが保存される。
5. 親モデルの更新画面を開く。繰り返しコンテナの1行目に複数ファイルをドラッグ&ドロップで登録し、保存する。例えば4つのファイルをドラッグ&ドロップすると、展開されて4行の繰り返しコンテナデータとなって保存される。
6. 子モデルの詳細画面を開く。子モデルの繰り返しコンテナは親モデルのコンテナを参照連動するために4行のデータが表示される。ただし1行目だけは手順4の時点で添付ファイルなしのデータが保存されていたため、これが優先され、1行目のデータのみ添付ファイルなしの状態で表示される。2-4行目は親モデルの添付ファイルが表示される。1行目だけ表示されないため、動作不良のように見える。

繰り返しコンテナの参照連動、かつ、繰り返しコンテナ内に自モデル保存の項目が存在する場合は、この処理は仕様となります。子モデルで一度参照連動した後に親モデルで添付ファイルが追加されても、子モデルのデータには反映されません。

対策

子モデルの登録時のスクリプトで対応する方法があります。

スクリプトでは親の繰り返しコンテナデータを取得し、子の繰り返しコンテナデータとして転記させます。このとき子モデルに既存のコンテナデータが存在していても、これを無視して上書きするようにする、というものです。

(59)

繰り返しコンテナでデータの変更履歴を管理する

変更履歴を有効にする

更新処理を行ったとき、ログファイル（logs/system.log）に次のような記録を残すことができます。

(59) この処理を親モデルの同時更新画面からの登録時のみに実行するようにするという制御も考えられます。SCREENID関数を利用して判別させることができます。

```
updateCustomer|DoUpdate:1000|
モデル'customer'の項目'顧客名'の値'鈴木　花子'が'佐藤　花子'に変更されました。
```

(60)

この機能は標準では無効になっています。有効にする場合は「画面>その他>世代管理>ログに変更履歴を残す」を有効にします。

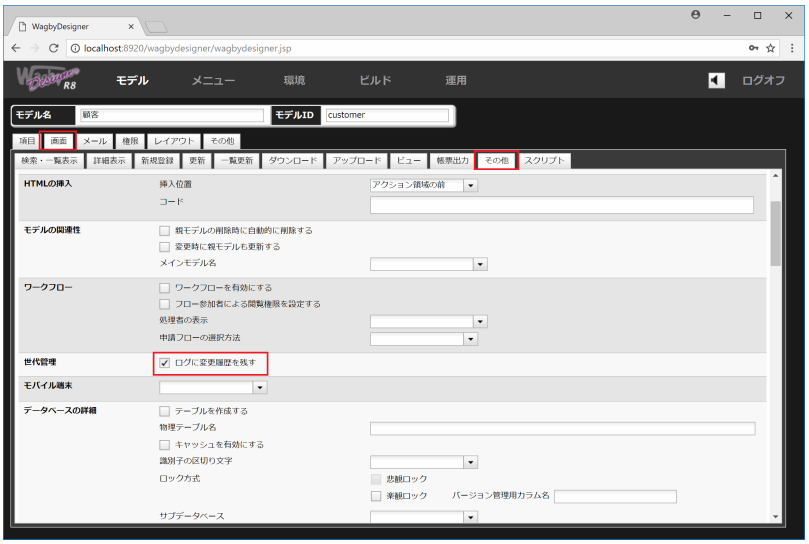


図　変更履歴機能を有効にする

本設定は変更のみならず、値の追加ならびに削除時にもログが出力されます。

モデルの詳細画面に変更履歴を表示する(1) - 先頭に追加される

例

ここでは繰り返しコンテナを用いて、任意のデータに関するすべての変更履歴を画面に表示する例を紹介します。

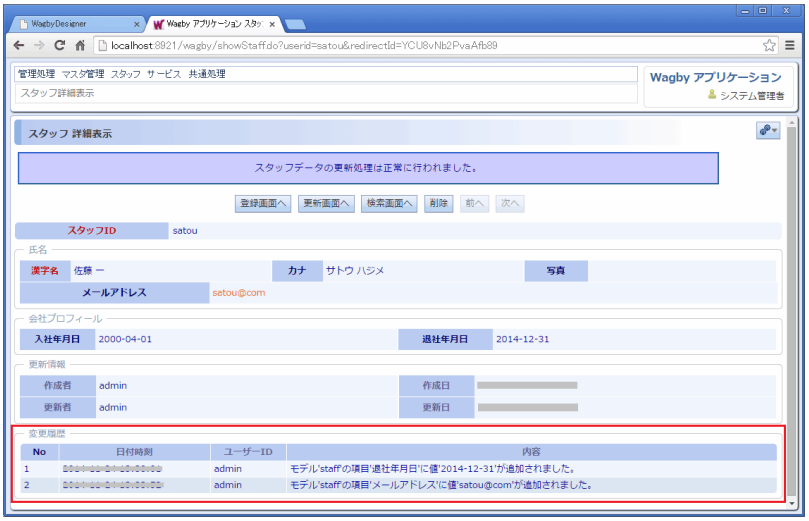


図　変更履歴が繰り返しコンテナの先頭に追加される

定義方法

(60) 実際には一行で出力されます。

変更履歴を管理する5つの項目を追加します。（項目名、項目IDはここに示した以外でもかまいません。）

項目名	項目ID	型	ラベルの幅	読み込み専用
変更履歴	changelog	繰り返しコンテナ	-	-
No	changelog/cid	繰り返しコンテナID	5%	○
日付時刻	changelog/cdate	日付時刻	20%	○
ユーザID	changelog/cuserid	文字列（ユーザID）	10%	○
内容	changelog/ccontent	文字列(テキストエリア)	65%	○

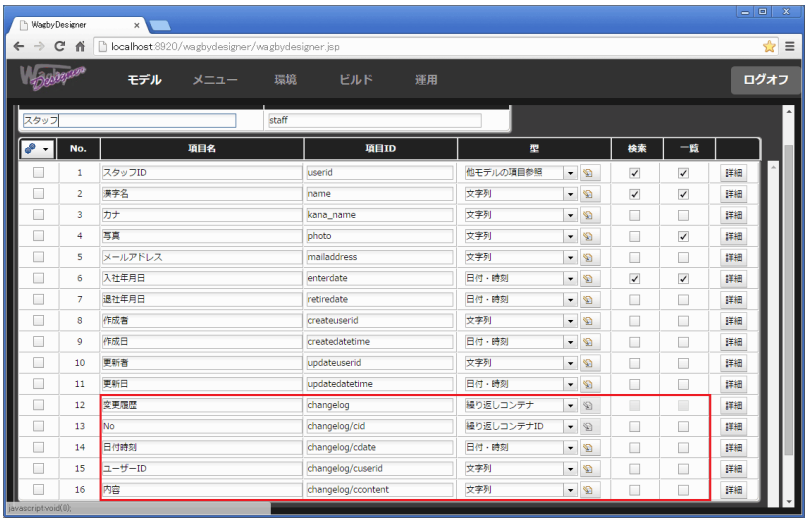


図 用意する項目

繰り返しコンテナ changelog

- 初期表示数を 0 に設定します。
- 「繰り返し項目の自動増減」を設定します。ここでは増減タイミングを「初期値設定の前」とし、増減位置を「先頭に追加」とします。
- 追加・挿入・削除・移動ボタンを表示しない、とします。

図 繰り返しコンテナchangelogの設定

コンテナの先頭に、最新の変更履歴を格納しようとしています。

繰り返しコンテナID changelog/cid

「出力制御>スタイル>ラベルの幅」を 5% とします。（以降、ラベルの幅設定の説明は省略します。）

日付時刻 changelog/cdate

日付時刻型とします。

図 日付時刻型とする

読込専用とします。

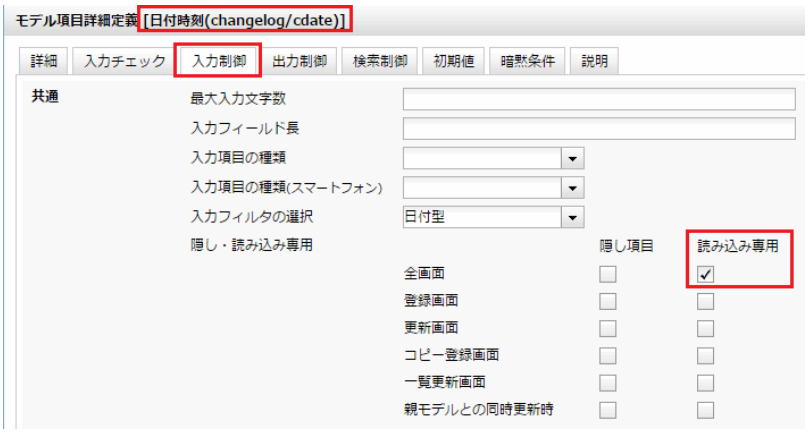


図 日付時刻項目を読取専用とする

日付時刻型の項目は標準でコンテナ内のソート機能が有効になっています。これを解除します。

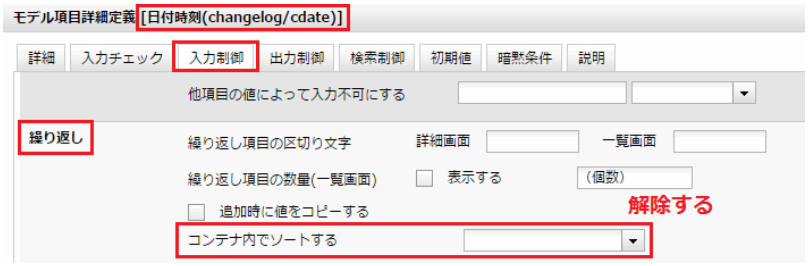


図 日付時刻項目によるコンテナ内のソート機能を解除する

登録・更新時の初期値を次のように式で設定します。

```
IF(CID()==FIRST(${changelog/cid}),NOW(),${changelog/cdate})
```

CIDは繰り返しコンテナでのみ使える特別な関数で、コンテナ内の位置を返します。この式は、先頭の行にはNOW関数の結果を適用し、それ以外は現時点の値をそのまま維持する、というものです。

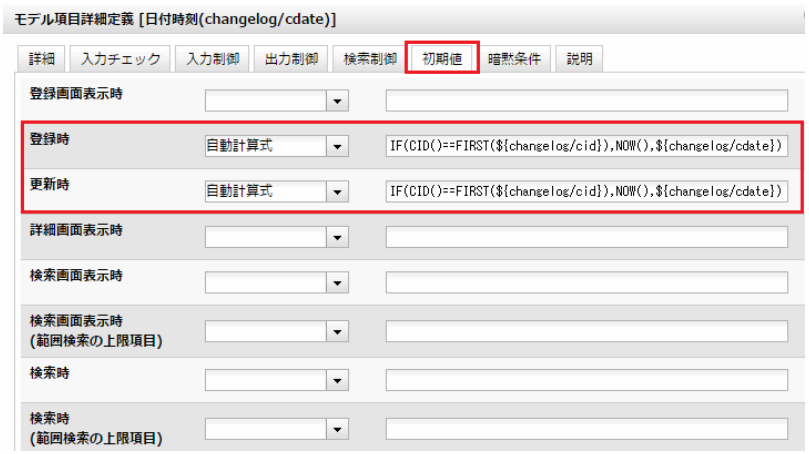


図 日付時刻項目の初期値

ユーザID changelog/cuserid

読込専用とします。

図 ユーザID項目を読取専用とする

登録・更新時の初期値を次のように式で設定します。

```
IF(CID()==FIRST(${changelog/cid}),USERINFO("userid"),${changelog/cuserid})
```

この式は、先頭の行には現在ログオンしているアカウントのuseridを適用し、それ以外は現時点の値をそのまま維持する、というものです。

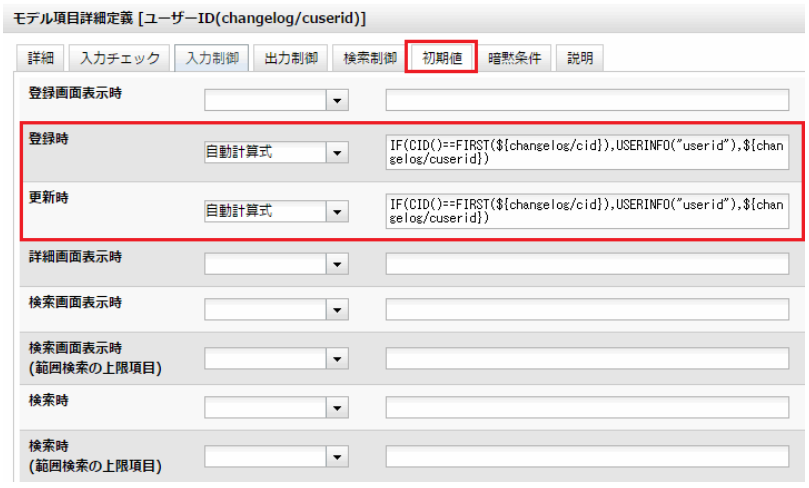


図 ユーザID項目の初期値

内容 changelog/ccontent

入力項目の種類を「テキストエリア」とします。また、読込専用とします。

図 内容項目を読取専用とする

内容の初期値を次のように式で設定します。

内容項目（新規登録）

```
IF(
  CID()==FIRST(${changelog/cid}),
  "データを新規に登録しました。",
  ${changelog/ccontent}
)
```

内容項目（更新）

```
IF(
  CID()==FIRST(${changelog/cid}),
  CHANGELOG(),
  ${changelog/ccontent}
)
```

この式は、先頭の行にはCHANGELOG関数の結果を適用し、それ以外は現時点の値をそのまま維持する、というものです。

図 内容項目の初期値にCHANGELOG関数を適用する

変更履歴内容を管理する項目の型が「テキストエリア」となっていない場合、標準で格納できる文字列の最大数は255文字に制限されます。

モデルの詳細画面に変更履歴を表示する(2) - 末尾に追加される

上の例は、繰り返しコンテナ「変更履歴」の先頭に、最新の更新履歴を追加するものでした。 これをコンテナの末尾に追加するように変更する方法を説明します。

図 変更履歴が繰り返しコンテナの末尾に追加される

設定の変更

図の「繰り返し項目の自動増減」を「先頭に追加」から「末尾に追加」に変更します。

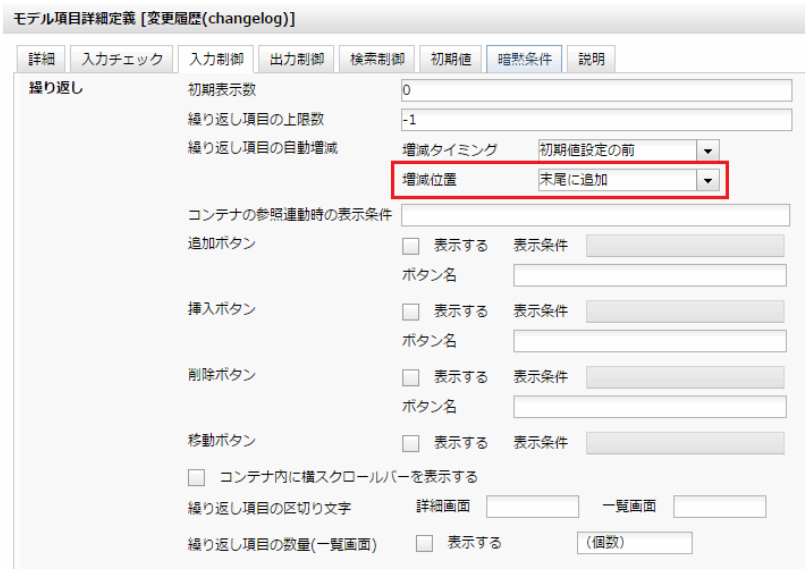


図 末尾に追加する設定

続いて、初期値の式でFIRST関数をLAST関数へ変更します。（日付時刻、ユーザID、内容項目のいずれも変更します。）

日付時刻項目

```
IF(CID()==LAST(${changelog/cid}),NOW(),${changelog/cdate})
```

ユーザID項目

```
IF(CID()==LAST(${changelog/cid}),USERINFO("userid"),${changelog/cuserid})
```

内容項目（新規登録）

```
IF(
  CID()==LAST(${changelog/cid}),
  "データを新規に登録しました。",
  ${changelog/ccontent}
)
```

内容項目（更新）

```
IF(
  CID()==LAST(${changelog/cid}),
  CHANGELOG(),
  ${changelog/ccontent}
)
```

図 内容項目の初期値にCHANGELOG関数を適用する

これでコンテナの末尾に、最新の変更履歴が格納されます。

テンプレートモデルとして用意する

複数のモデルで変更履歴を扱うような場合は、テンプレートモデルにすると便利です。 ここでは「システム」タブ内の jfcDesignerUserTemplateModelitem1 を「変更履歴」テンプレートとする例を示します。

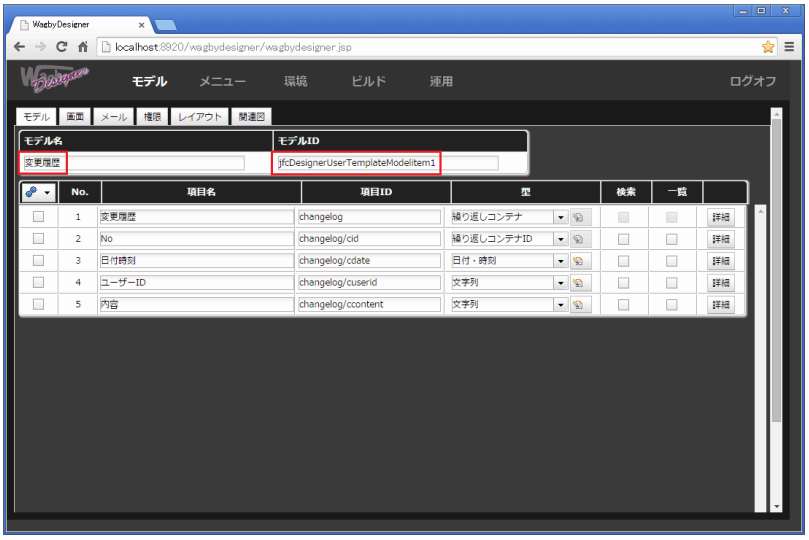


図 変更履歴テンプレートの作成

システムタブ一覧には、作成した変更履歴テンプレートを確認できます

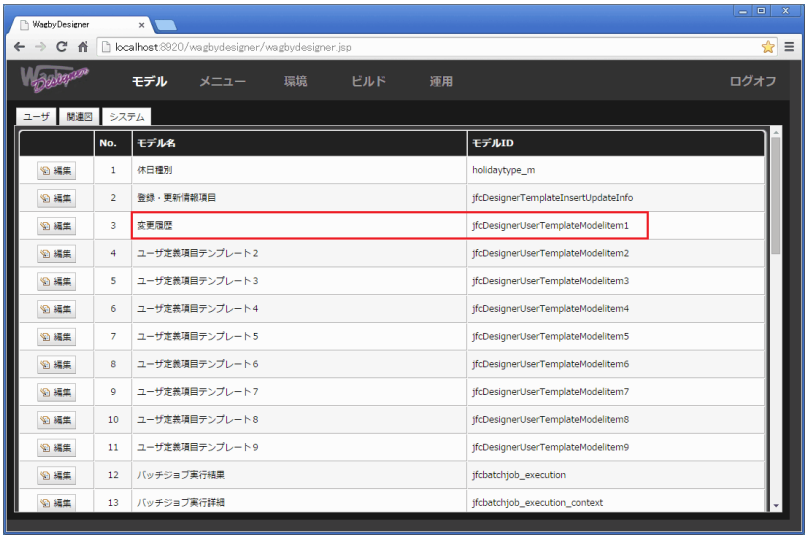


図 システムタブ内のモデル

適用例

任意のモデルに対して、ギアアイコンから「テンプレート>変更履歴」を選択してみます。

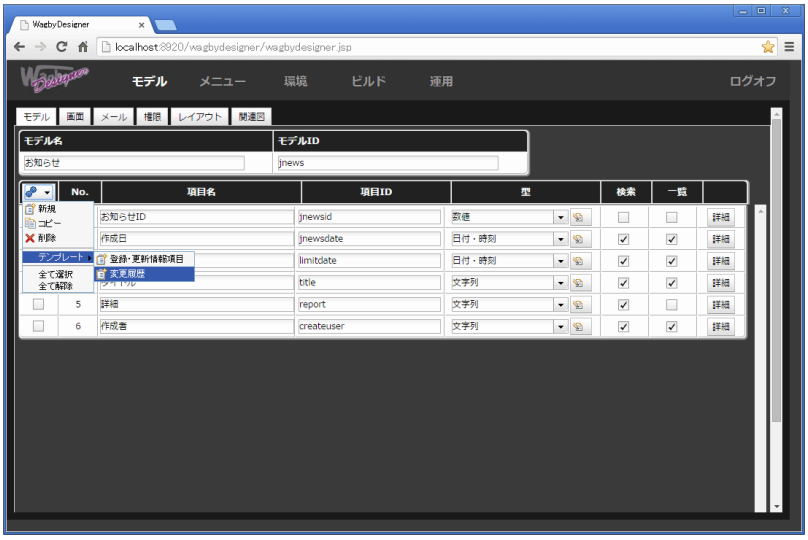


図 変更履歴テンプレート項目群を追加する

変更履歴に関する5つの項目がモデルに追加されます。

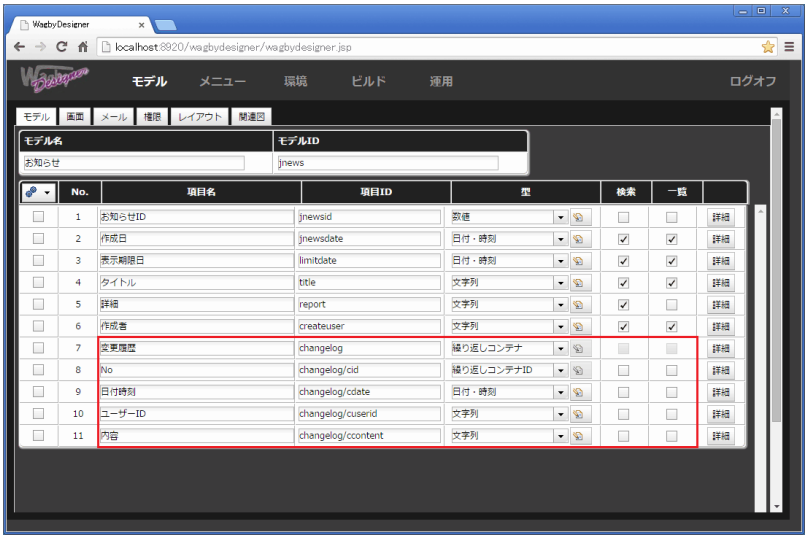


図 追加された項目

最後に、このモデルの変更履歴機能を有効にします。

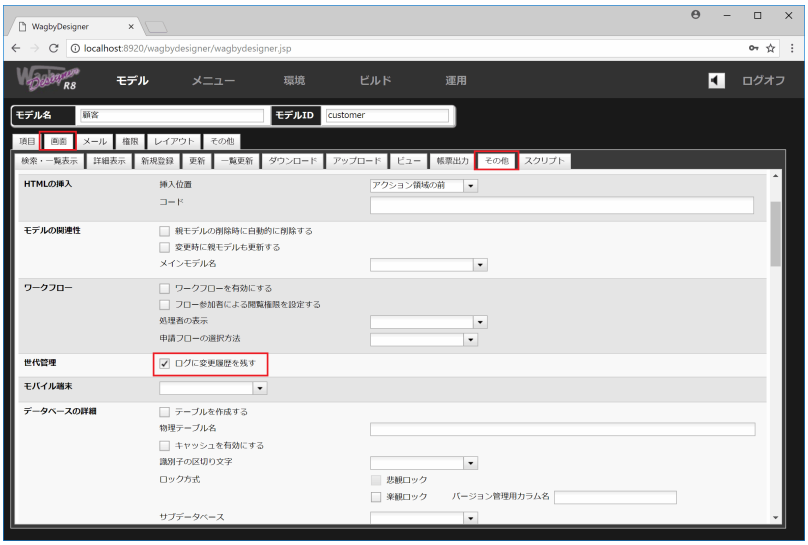


図 変更履歴機能を有効にする

仕様・制約

- 変更履歴として用いる繰り返しコンテナは、一つのモデルに「最大一つだけ」用意することができます。同じ（変更履歴を管理する目的の）コンテナを二つ以上、定義することはできません。[61]
- ファイル型項目で変更があった場合、実ファイル名部分が変更履歴に記録されます。

繰り返しコンテナでデータの既読状態を管理する(スクリプト)

例

「顧客」モデルに既読情報を扱えるデータ項目を用意した例を紹介します。登録直後では、「閲覧記録」は空となっています。

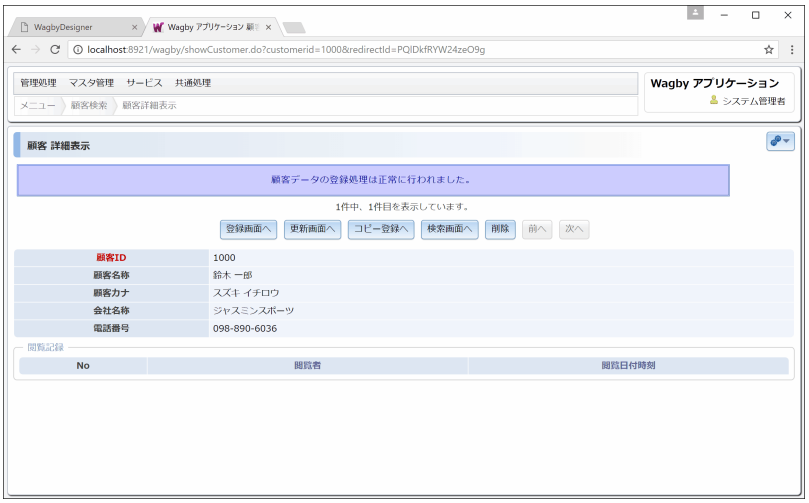
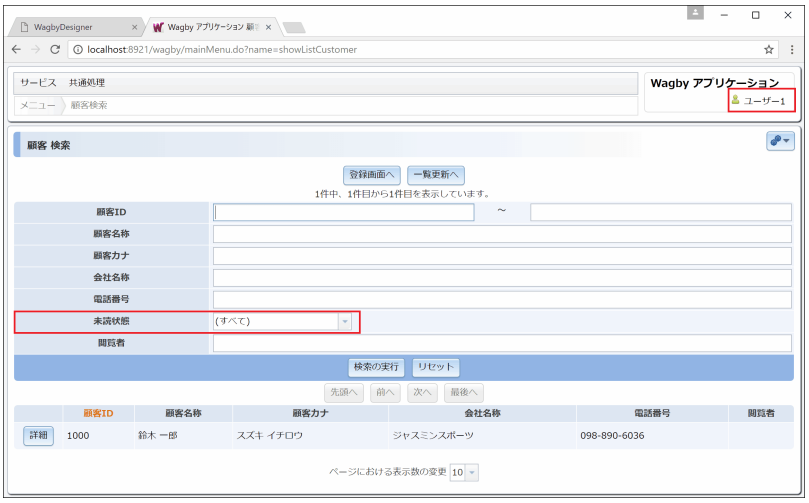


図 登録直後のデータ

「ユーザー1」でログオンする

事前に用意したアカウント「ユーザー1」でログオンします。検索条件に「未読状態」という項目が用意されています。



(61) 具体的には、初期値の式に CHANGELOG() 関数を用いている繰り返しコンテナが一つだけという判定を行っています。

図 未読状態で検索が行える

未読状態を"既読"とすると、データは見つかりません。

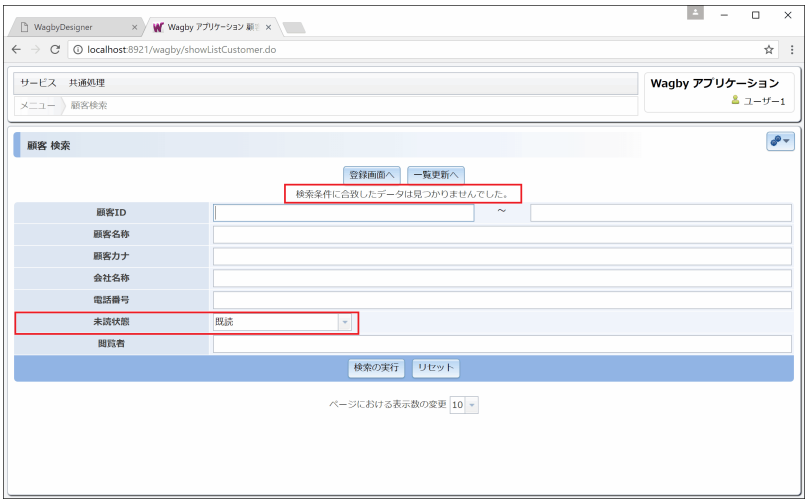


図 既読したデータがない

逆に、未読状態を"未読"とすると、データが見つかります。

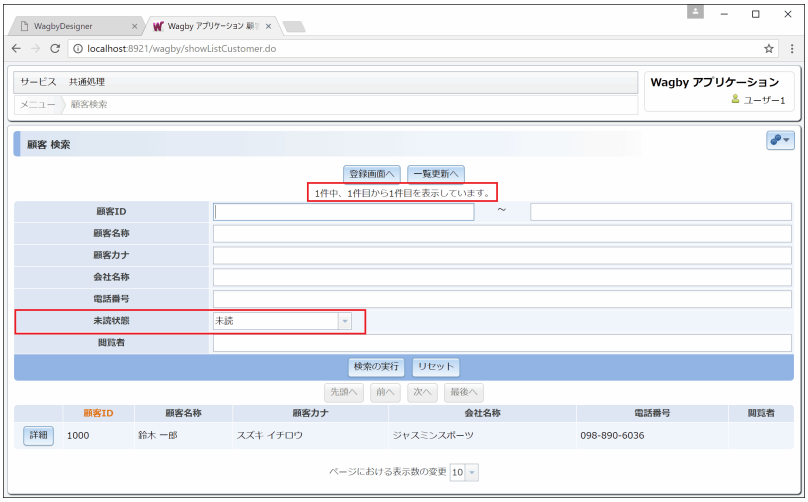


図 未読のデータが存在する

データの詳細画面を開きます。このタイミングで、閲覧記録（繰り返しコンテナ）に既読情報が登録されます。

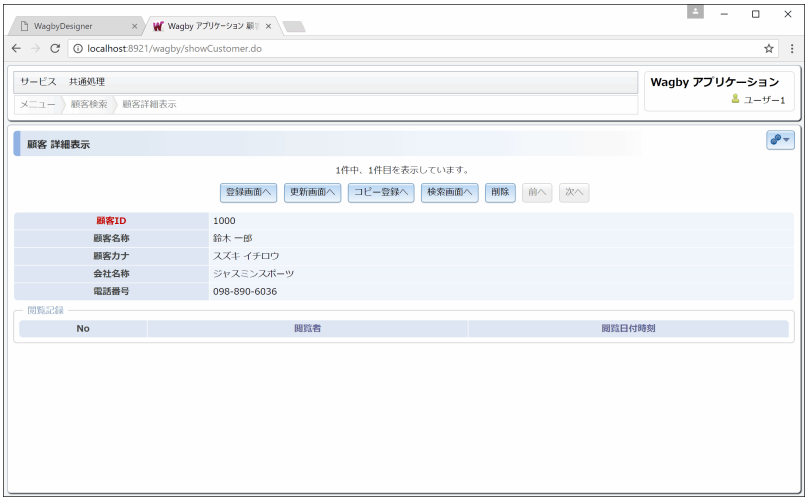


図 詳細画面を開く

(62)

再び検索を行ってみます。今度は"未読"データが存在しないことがわかります。

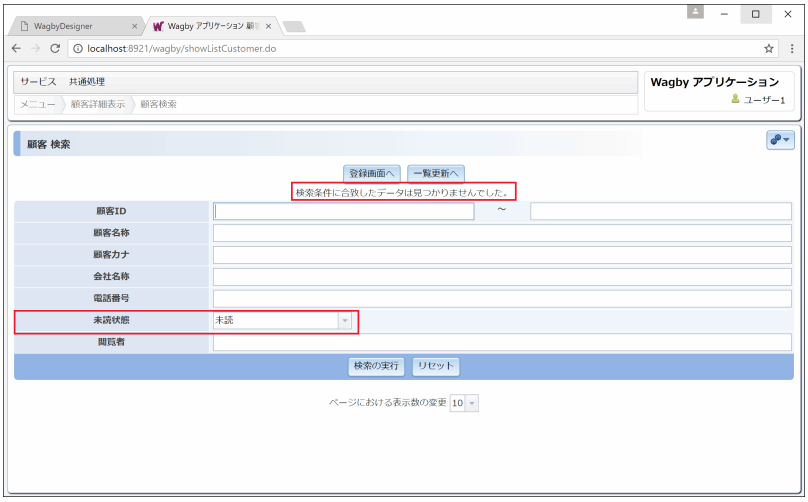
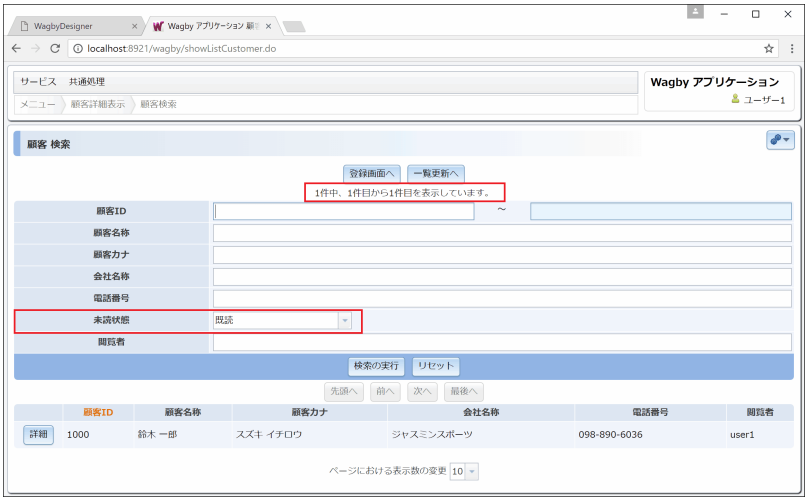


図 未読データがない

逆に、"既読"データが見つかります。



(62) 正確には、詳細表示データを画面に描画した直後に、既読情報が登録されます。そのため図の画面には既読情報がまだ表示されていません。

図 既読のデータが存在する

改めてデータを確認します。閲覧記録にアカウント名と閲覧日付時刻が加わっていることがわかります。

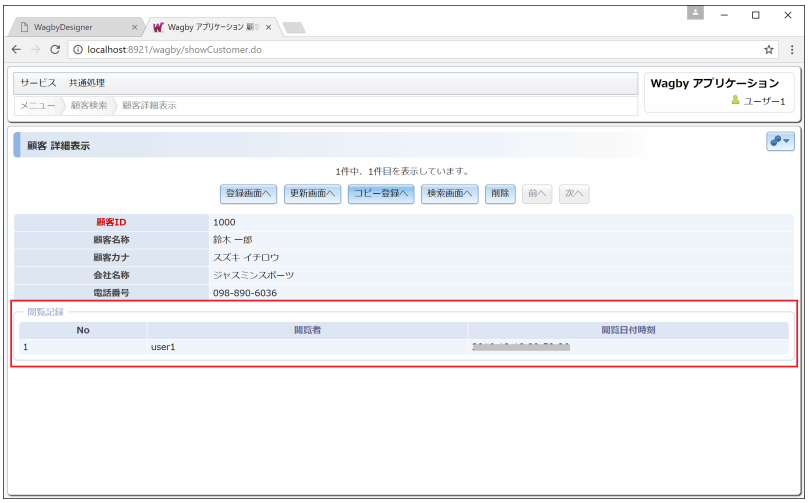


図 閲覧記録が更新されている

「ユーザー2」でログオンする

別のアカウント「ユーザー2」でログオンします。同じデータは"未読"となっていることがわかります。

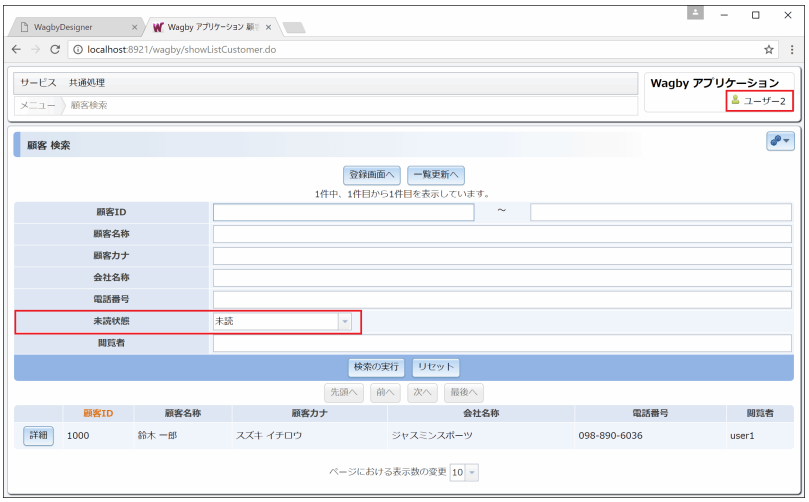


図 ユーザー2でログオンした場合

複数のユーザーでデータを閲覧する

複数のアカウントがデータを閲覧するたびに、閲覧記録が追加されていきます。

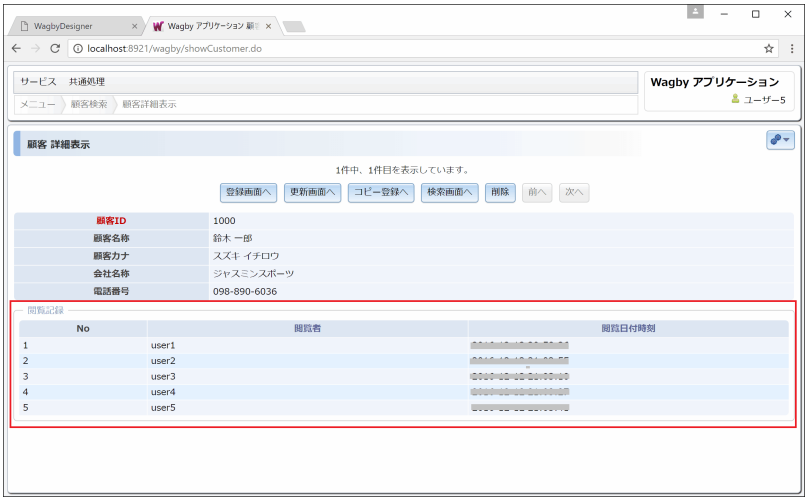


図 閲覧記録が追加された例

定義方法

設定のポイントは2つです。1つは「未読状態」で検索できるようにすること。もう1つは「閲覧記録」という繰り返しコンテナを用意することです。

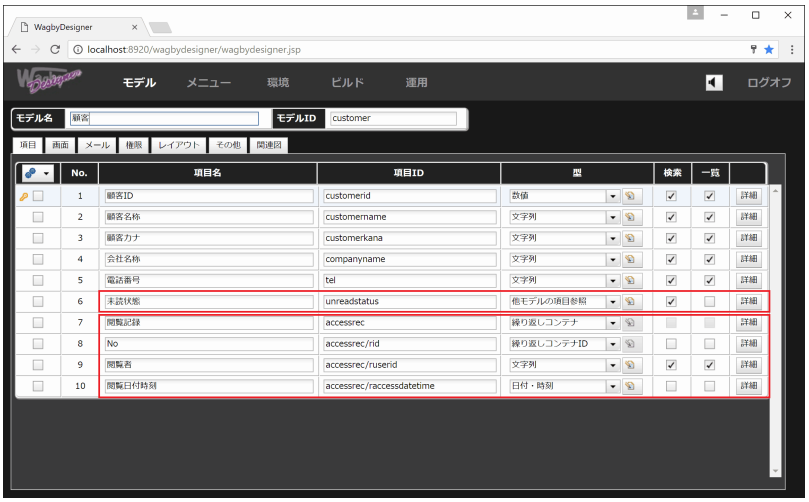


図 顧客モデルの定義

閲覧記録は次のようになっています。

項目名	項目ID	型
閲覧記録	accessrec	繰り返しコンテナ
No	accessrec/rid	繰り返しコンテナID
閲覧者	accessrec/ruserid	文字列
閲覧日付時刻	accessrec/raccessdatetime	日付時刻

未読状態

システムが提供する選択肢モデル「未読状態マスタ」を使います。この選択肢モデルはデータベースにテーブルを作成しないようになっています。

図 未読状態マスタを検索条件として用いる

「検索制御 > 既読/未読管理」で、既読/未読状態の検索に利用する項目として、繰り返しコンテナ内の閲覧者を格納している項目を指定します。

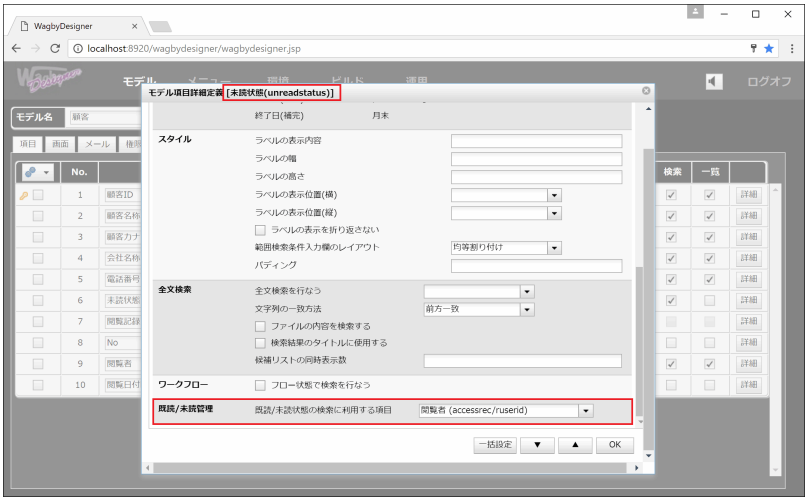


図 既読/未読状態の検索に利用する項目を設定する

この項目は検索のためだけに用いるため、DB保存なし、検索画面以外では隠し項目とします。具体的には、次の設定を行ってください。

詳細 > データベースの詳細 > 値をデータベースに保存する、を「無効」とする。

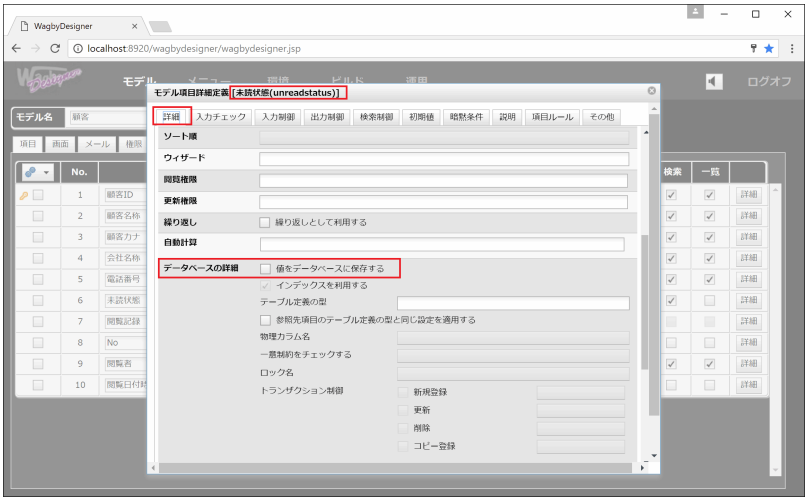


図 値をデータベースに保存する、を「無効」とする

入力制御 > 共通 > 隠し・読み込み専用、を「全画面で隠し項目」とする。

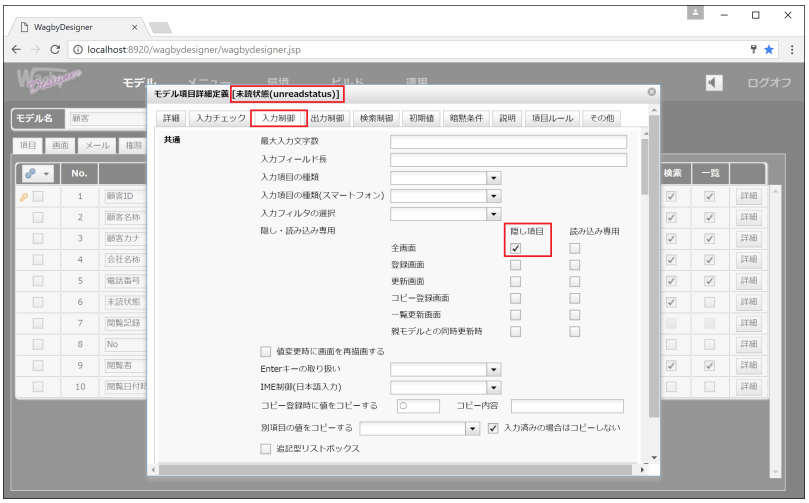


図 入力時は全画面で隠し項目とする

入力制御 > モデル参照 > 未選択を用意する、を「無効」とする。

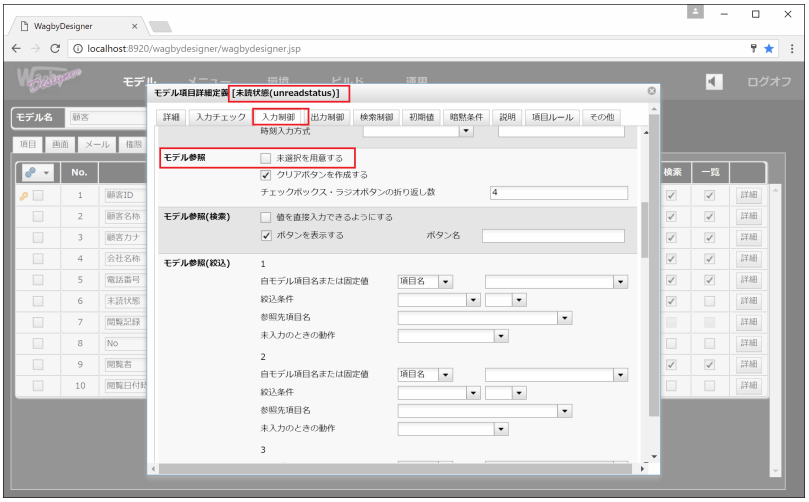


図 未選択を用意しない

出力制御 > 詳細画面 > 共通 > 隠し項目にする、を「有効」とする。

図 出力時も隠し項目とする

繰り返しコンテナ 閲覧記録

閲覧記録は読み込み専用とします。

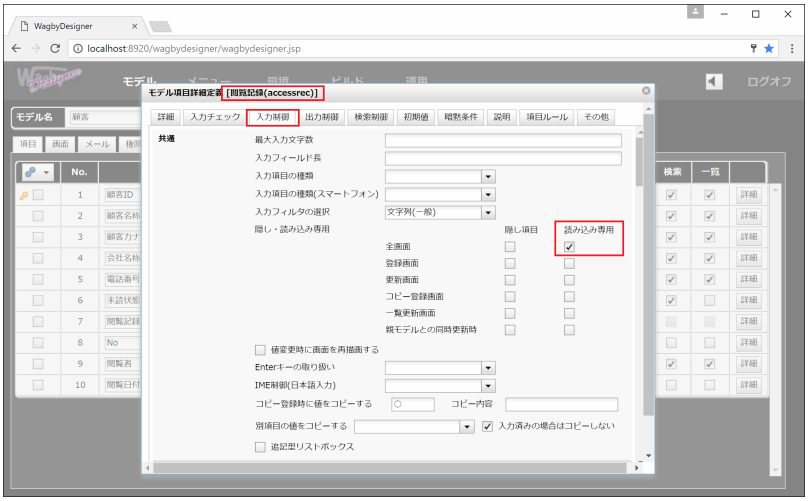


図 読み込み専用とする

初期表示数を 0 とします。追加/挿入/削除/移動ボタンはすべて無効とします。

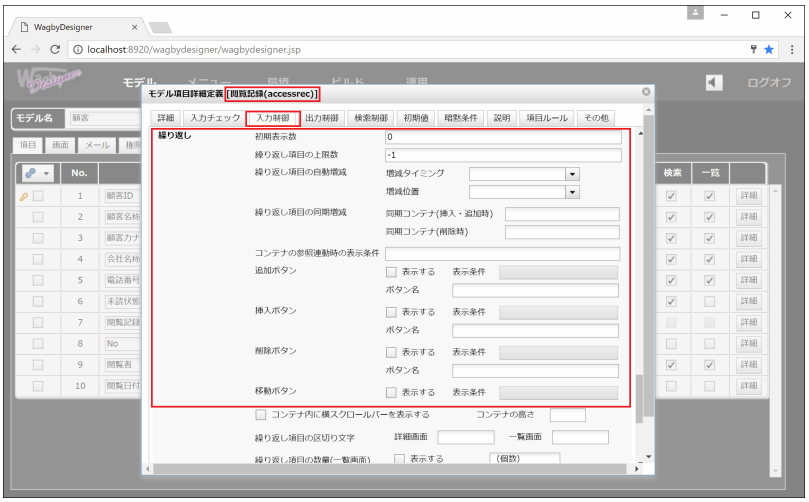


図 ボタン類を無効とする

スクリプト

「画面 > コントローラ」のスクリプトを設定します。タイミングは詳細画面の表示時です。

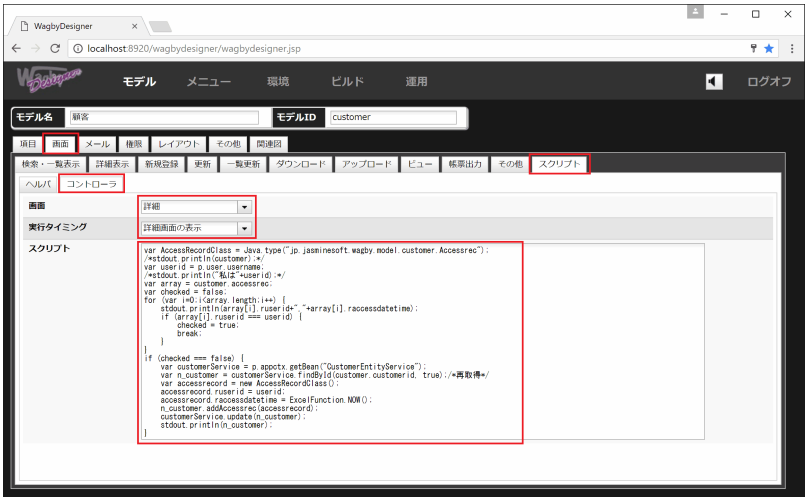


図 スクリプトの設定

スクリプトは次のようにします。モデルや項目の名称は、適切に置換して利用してください。

```
var AccessRecordClass = Java.type("jp.jasminesoft.wagby.model.customer.Accessrec");/*繰り返しコンテナ*/
/*print(customer);*/
var userid = p.user.username;
/*print("私は"+userid);*/
var array = customer.accessrec;
var checked = false;
for (var i=0;i<array.length;i++) {
    /*print(array[i].ruserid+","+array[i].raccessdatetime);*/
    if (array[i].ruserid === userid) {
        checked = true;
        break;
    }
}
if (checked === false) {
    var customerService = p.appctx.getBean("CustomerEntityService");
    var n_customer = customerService.findById(customer.customerid, true);/*再取得*/
    var accessrecord = new AccessRecordClass();/*新しい繰り返しコンテナを用意する*/
    accessrecord.ruserid = userid;
    accessrecord.raccessdatetime = ExcelFunction.NOW();
    n_customer.addAccessrec(accessrecord);/*追加*/
    customerService.update(n_customer);/*更新*/
    /*print(n_customer);*/
}
```

- 前半で、繰り返しコンテナを1件ずつ確認し、すでに自分自身 (p.user.username) が既読かどうかをチェックします。
- もし未読であれば、改めてサービスオブジェクトからオブジェクトを取得し、繰り返しコンテナを1件、追加して更新します。
- 更新処理を行うため、findById メソッドの第二引数を true としてロックしています。

URLを直接入力して閲覧する場合

URLなどを直接指定して詳細画面に遷移させる場合、リンク先は次のように action_Show=Show というパラメータを付与してください。

```
http://localhost:8921/wagby/showModel1.do?pkey=1000&action_Show=Show
```

コントローラのスクリプトを呼び出すためには、イベント名が必要となるためです。（イベント名は、action_ の後ろの文字列が利用されます。）

仕様・制約

- この機能は、通常の検索画面、サブウィドウでの検索画面、帳票画面で有効です。
- ダウンロード画面では動作しません。
- この例では閲覧したユーザーのIDを格納するための項目として繰り返しコンテナを用いましたが、それ以外にも繰返し項目またはチェックボックスを使うことができます。[後述...]

例 既読状態をチェックボックスで管理する

閲覧したユーザーのIDをチェックボックス型項目で管理する例を紹介します。"閲覧記録"（accessrec）はjuser モデルを参照するようにします。

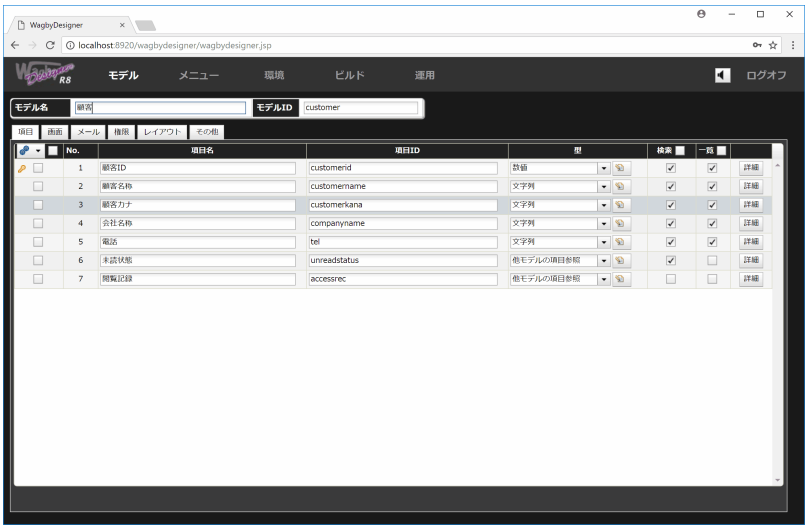


図 閲覧記録項目をチェックボックスとする(1)

この構成では、"閲覧日付時刻" 項目は含まれません。閲覧者の情報が "閲覧記録" 項目に格納されます。

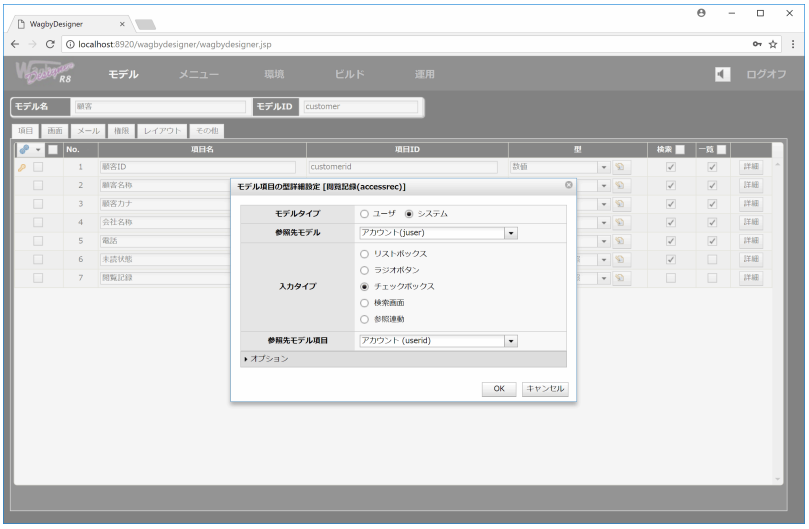


図 閲覧記録項目をチェックボックスとする(2)

スクリプトは次のようになります。

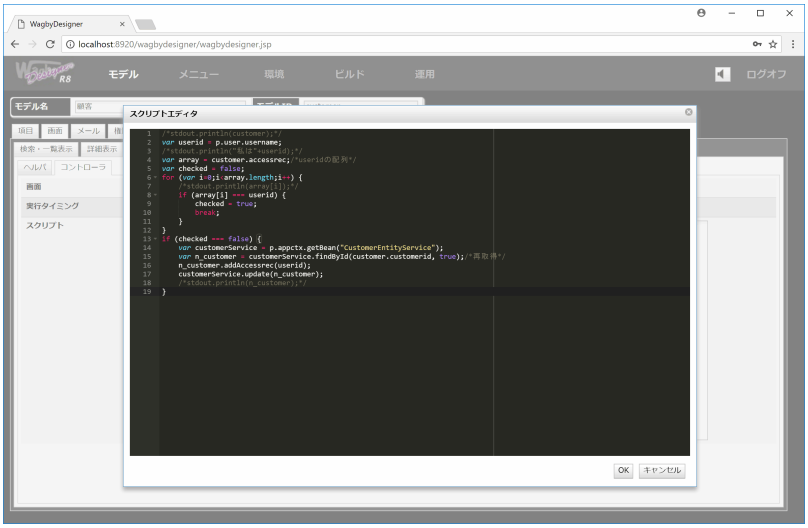


図 詳細画面を開いた時のスクリプト

```
var userid = p.user.username;
/*print("私は"+userid);*/
var array = customer.accessrec;/*useridの配列(文字列型)*/
var checked = false;
for (var i=0;i<array.length;i++) {
    /*print(array[i]);*/
    if (array[i] === userid) {
        checked = true;
        break;
    }
}
if (checked === false) {
    var customerService = p.appctx.getBean("CustomerEntityService");
    var n_customer = customerService.findById(customer.customerid, true);/*再取得*/
    n_customer.addAccessrec(userid);
    customerService.update(n_customer);
    /*print(n_customer);*/
}
```

・　閲覧記録(accessrec)項目はjuserモデルの主キーを格納します。これは文字列型項目となります。

繰り返しコンテナ - 独立したモデルとして扱う

例

「見積書」モデルに含まれる繰り返しコンテナ「内訳」を、別モデル「見積明細」として操作できるようにします。 図にイメージを示します。

繰り返しコンテナはデータベース上は別テーブルとなっており、テーブル名は次のルールで決められています。

```
Model ID + "$" + Container ID
```

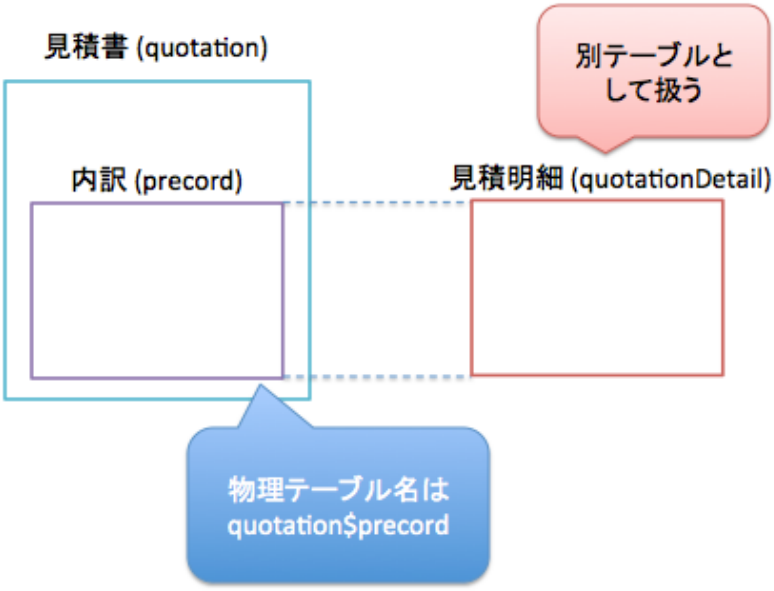


図 繰り返しコンテナを別モデルとして扱うイメージ

見積書モデルの繰り返しコンテナは図のとおりです。

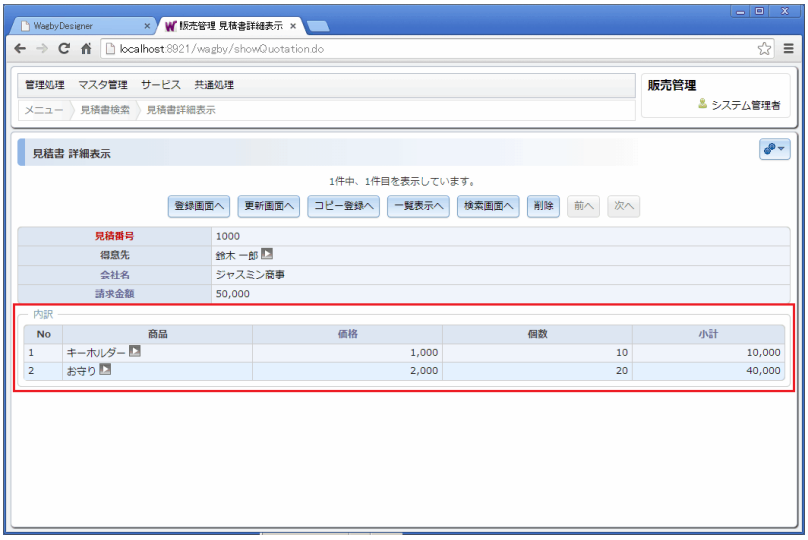


図 見積書モデル詳細画面

追加した「見積明細」モデルの検索画面を図に示します。繰り返しコンテナ部分がデータとして検索できます。

図 見積明細モデル検索画面

「見積明細」モデルの一覧画面を図に示します。内容は図と同じです。

図 見積明細モデル一覧画面

定義方法

繰り返しコンテナを定義した「見積書」モデルを図に示します。

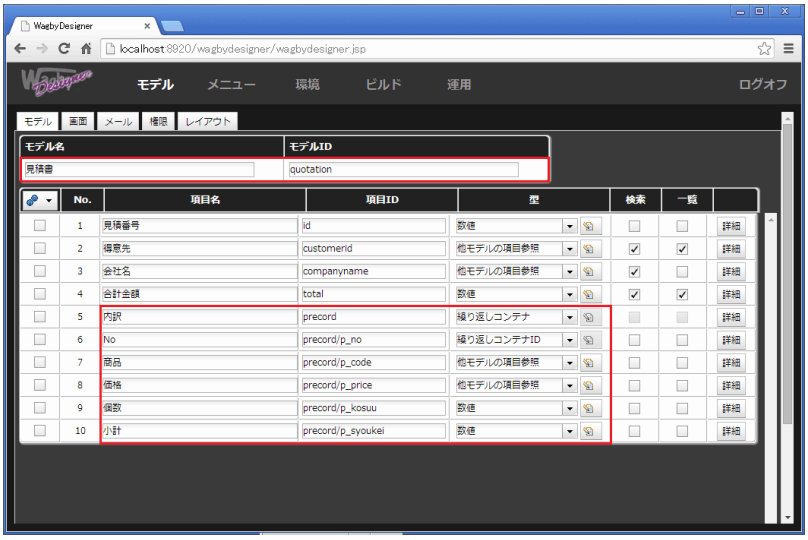


図 見積書モデルの定義

図における赤枠の繰り返しコンテナ部分を抜き出して、「見積明細」モデルとして定義したのが図です。

定義のポイントは次のとおりです。

- 主キーは「見積書」モデルと同一とします。この例では "id" です。
- 「繰り返しコンテナ名 + "jshid"」となる項目が必要です。型は整数型です。この例では "precordjshid" です。
- "id" と "precordjshid" の二つで複合主キーとします。（それぞれに「主キーとして利用する」を有効とします。）
- モデル内の項目は、参照元の繰り返しコンテナと同じ項目ID、同じ型としてください。項目名の部分は変更できます。
- 参照元の繰り返しコンテナの項目を減らすことはできますが、追加することはできません。

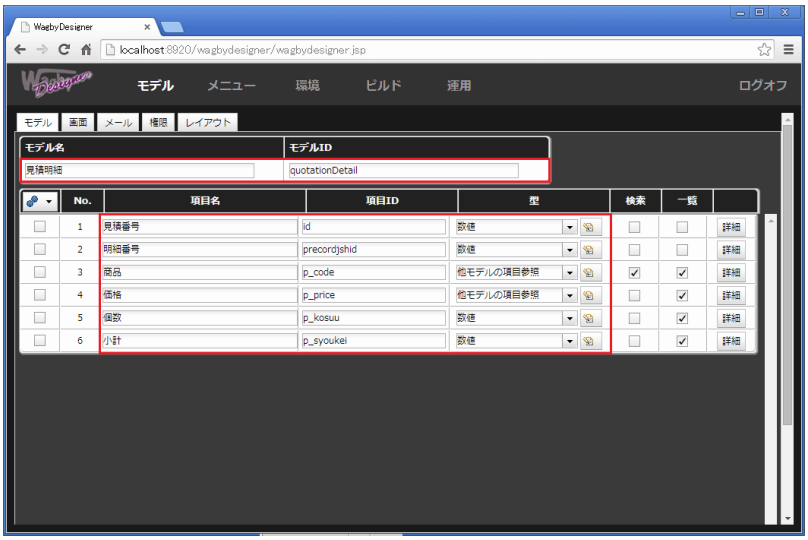


図 見積明細モデルの定義

図に、繰り返しコンテナID "precordjshid" を主キーとして利用する設定を行ったところを示します。順序は利用しません。

図 主キーの設定

見積明細モデルの「画面>その他>データベースの詳細」を開きます。「テーブルを作成する」を無効とし、「物理テーブル名」を "quotation$precord" と直接、明記します。さらに「キャッシュを有効にする」を解除します。

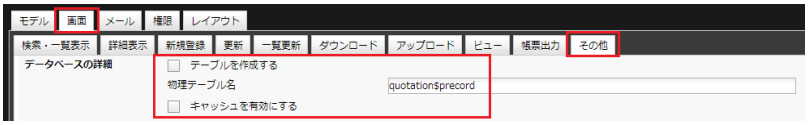


図 テーブルを作成するを解除し、物理テーブル名を指定する、キャッシュを無効化する

更新を行わないように設定する

別モデルとして用意した「見積明細」モデルは、登録・更新・削除は行なわないようにしてください。参照のみで運用することを想定しています。（データ入力は、元々の「見積書」モデルでのみ行います。）

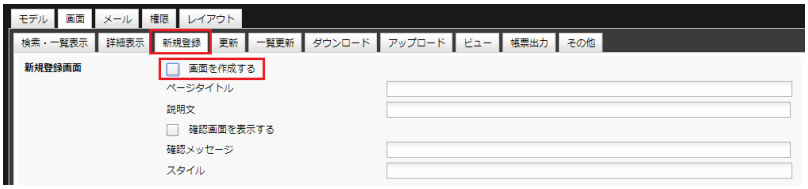


図 新規登録画面を作成しない

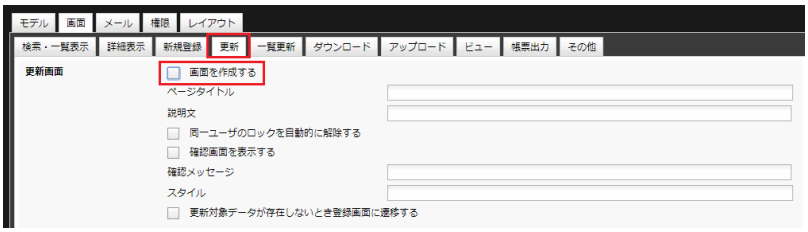


図 更新画面を作成しない

更新を行いたい場合

上の説明では「更新画面を作成しない」としていますが、その理由は次のとおりです。

1. 元のモデルと、繰り返しコンテナ独立モデルの双方で更新すると、メモリに保持されているキャッシュの整合性がとれない。
2. 両方が同時に更新処理を行ったとき、ロックキーが異なるため、ロック機構が働かない。

この問題を回避する方法を紹介します。

キャッシュを使わない

元のモデル（上の例では「見積書」）と、繰り返しコンテナ独立モデル（上の例では「見積明細」）のそれぞれで、キャッシュを無効にします。パフォーマンスは不利になります。

同時に更新されない運用とする

例えば見積書に制御フラグを用意します。このフラグ値によって、一方の更新ボタンを無効にするというボタン表示制御式を用意し、同時更新を回避します。

ロックキーを同じにする

繰り返しコンテナ独立モデル（上の例では「見積明細」）のロックキーを、元のモデル（上の例では「見積書」）に合わせます。この場合、繰り返しコンテナ独立モデルの個々のレコード単位でのロック制御は行えなくなります。（繰り返しコンテナ行全体で、一つのロックキーとなるためです。）

スクリプトで利用する

スクリプトで繰り返しコンテナ独立モデルを操作することができます。

ここでは customer モデルに favorite（趣味）という繰り返しコンテナを用意したとします。また、繰り返しコンテナ独立モデル Favorite を定義したとします。

```
var Favorite = Java.type("jp.jasminesoft.wagby.model.favorite.Favorite");
var entityService = p.appctx.getBean("FavoriteEntityService");
var key = new Favorite.Key();/*複合キーの扱いになる*/
key.id = 1000;
key.contjshid = 1;
var favorite = entityService.findById(key);
print(favorite);
```

トラブルシューティング

データインポート時や init_db 実行時に警告が発生する

system.log や initdb.log に次のような警告が出力されることがあります。

```
Invocation of init method failed; nested exception is org.hibernate.MappingException: Foreign
 key (...) must have same number of columns as the referenced primary key (...)
```

このExceptionが発生した場合は、主キーの定義が誤っている可能性があります。主キーの定義の見直しを行って下さい。

親子関係

親子関係を定義する

例

ここでは「顧客（customer）」と「サポート（support）」という2つのモデルをとりあげてみます。

顧客のデータを新規に登録します。 顧客の詳細表示画面に「サポート新規作成」ボタンが用意されています。 外部キーの関係にあるサポートモデルの新規登録は、対象データ（ここでは顧客データ）の詳細表示画面から行います。

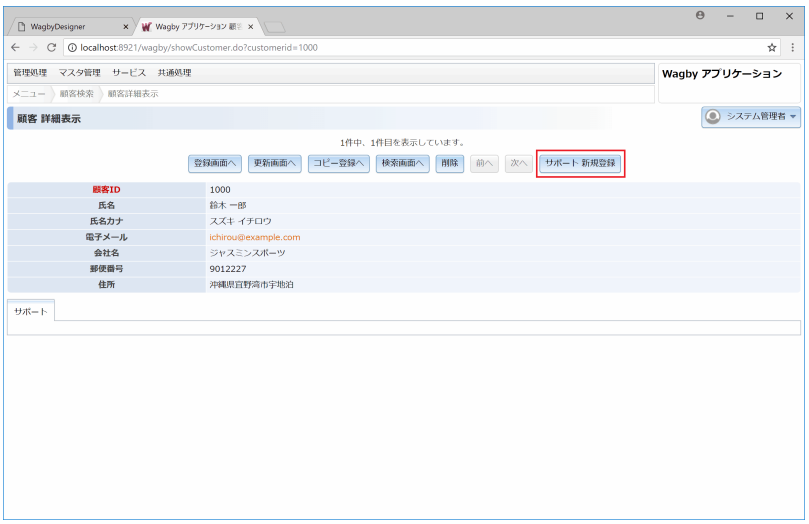


図 顧客詳細表示画面

「サポート新規作成」ボタンをクリックし、サポート新規登録画面を開いてみます。顧客ID（紐づく親モデルのID）は自動設定されています。

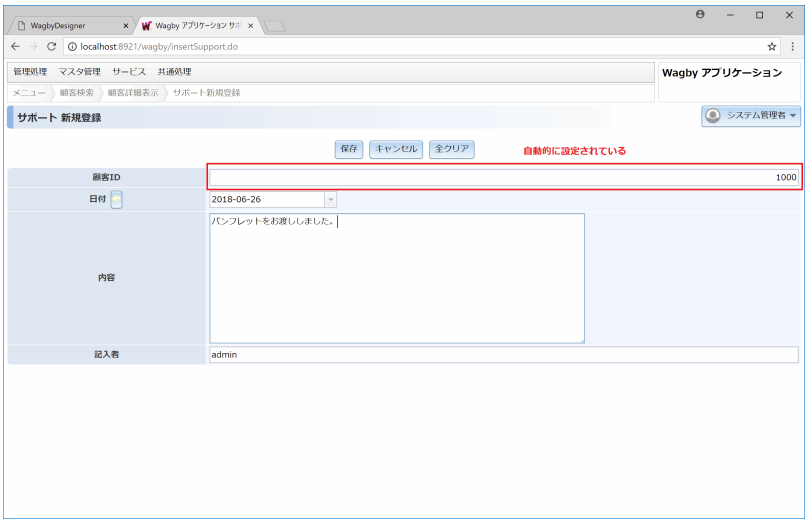


図 サポート新規登録画面

各項目を入力して登録を行います。サポート詳細表示画面から顧客モデルへ戻るための「顧客 一覧表示」「顧客 詳細表示」ボタンが用意されています。

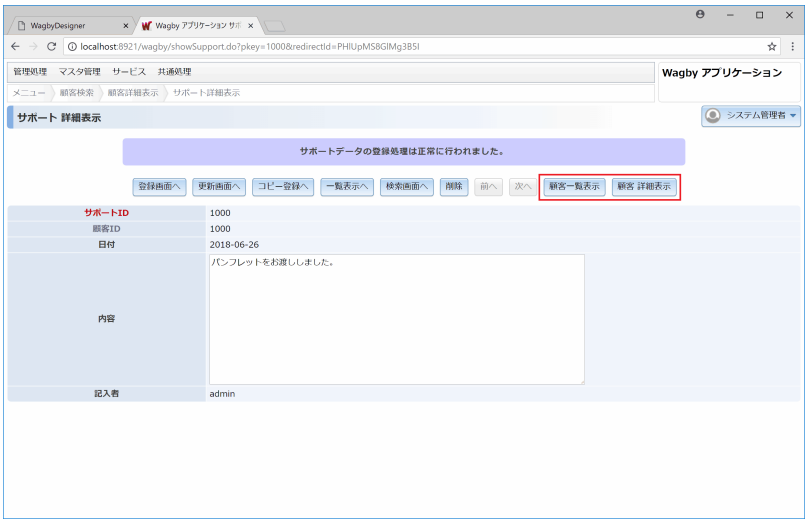


図 サポート詳細表示画面

「顧客 詳細表示」ボタンをクリックして顧客詳細表示画面を開きます。顧客モデル側でもサポート記録の一覧が表示されるようになります。

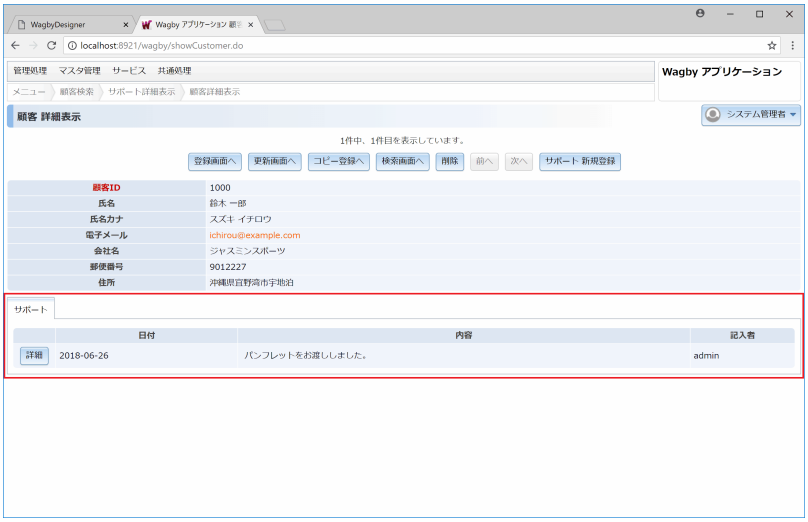


図 顧客の詳細表示画面

このように、外部キー関係にあるモデルは、画面上でも双方向のリンク関係をもちます。これによって「どの顧客についてのサポート記録だったか」を確認できるようになります。

外部キー関係では、顧客のデータが削除されたタイミングで、それに関連するサポートのデータも全て削除されます。（この動作を変更し、サポートのデータを同時に削除しない設定とすることもできます。）

定義方法

今回用意したサポートモデルの定義を図に示します。 外部キーに指定した項目の型は、参照先（親）の項目の型と同じ型で指定します。ここでは「顧客ID」を「数値」型と指定しています。

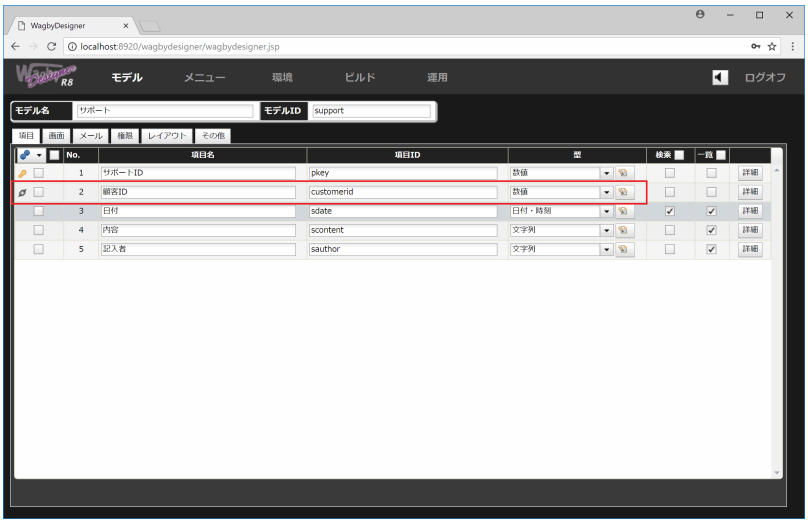


図 サポートモデルの定義

定義のポイントは「顧客ID」が外部キーであることを指定する部分です。 「外部キー>モデル名」に、親モデルを指定します。

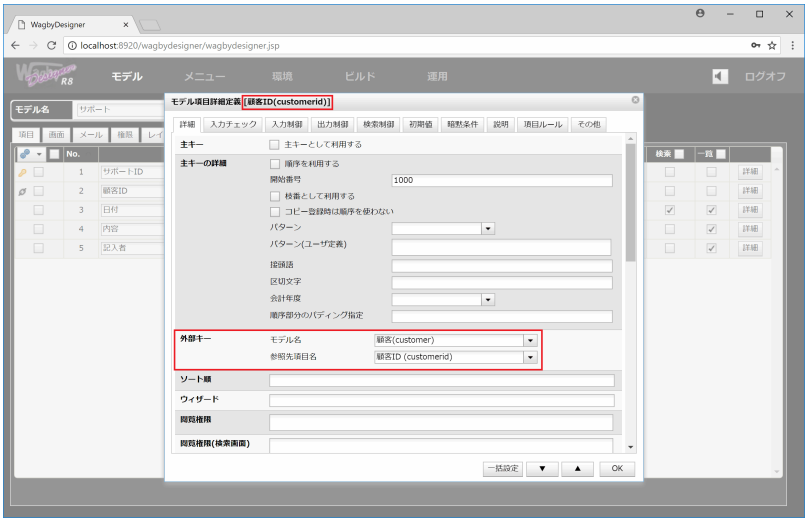


図 外部キーの定義

参照先項目名は、主キーとなっている必要があります。上の例では（参照先モデルである）″顧客″の″顧客ID″は主キーとなっています。

子モデル作成時の親の主キーの扱い

子モデルの詳細画面に用意された「登録画面へ」ボタンを押下して新規登録画面を開くと、親の主キーは転記された状態になります。つまり子は共通の親を持つことが前提となります。

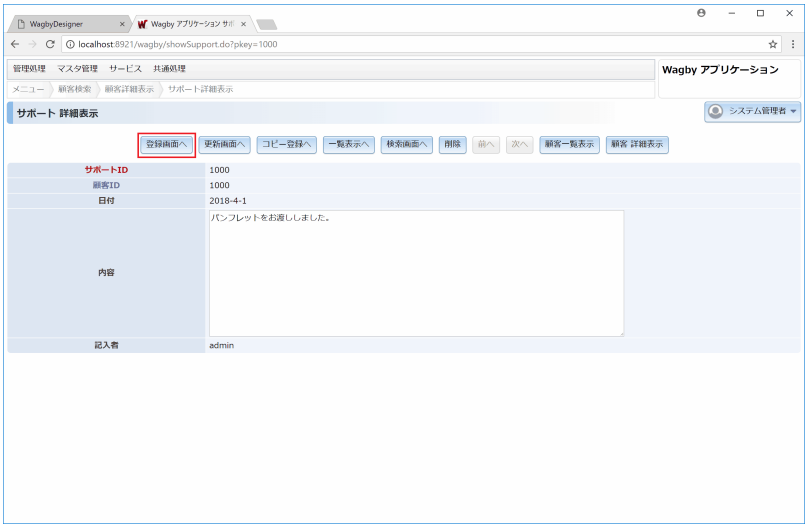


図 子モデルの詳細画面に用意された「登録画面へ」ボタンを押下する

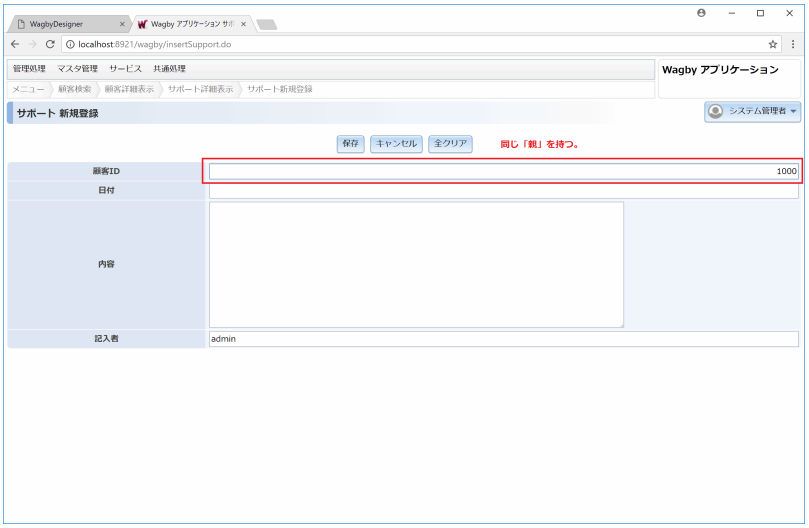


図 親の主キーが転記されている

一方、子モデルの検索・一覧表示画面に用意された「登録画面へ」ボタンを押下して新規登録画面を開くと、親の主キーは空白です。これは特定の子を起点にしていないためです。

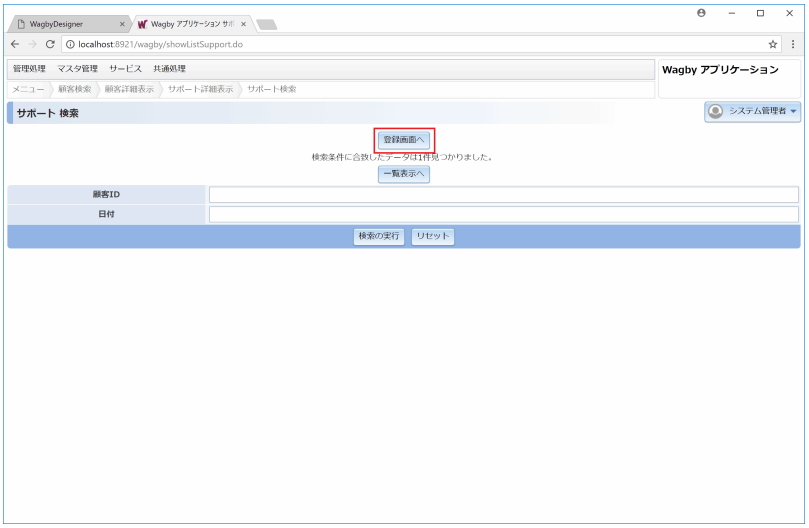


図 子モデルの検索・一覧表示画面に用意された「登録画面へ」ボタンを押下する

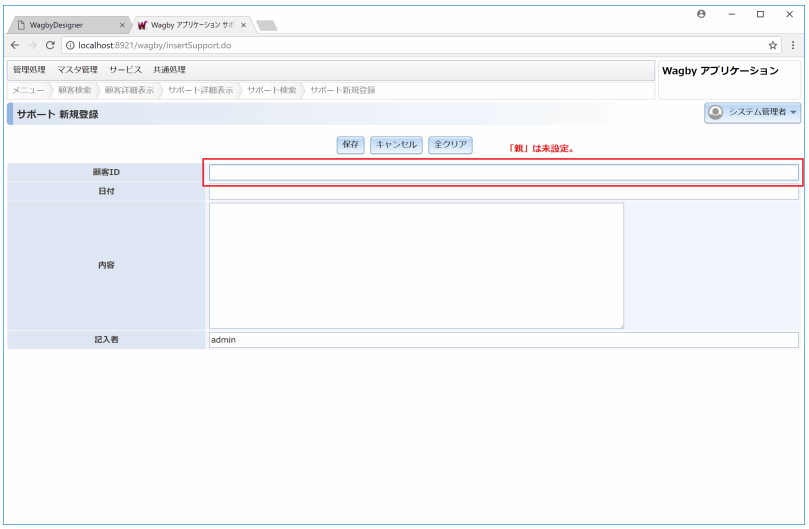


図 親の主キーは空白である

親モデルの詳細画面に用意された子モデルの新規登録を押下すると、親の主キーは転記されます。

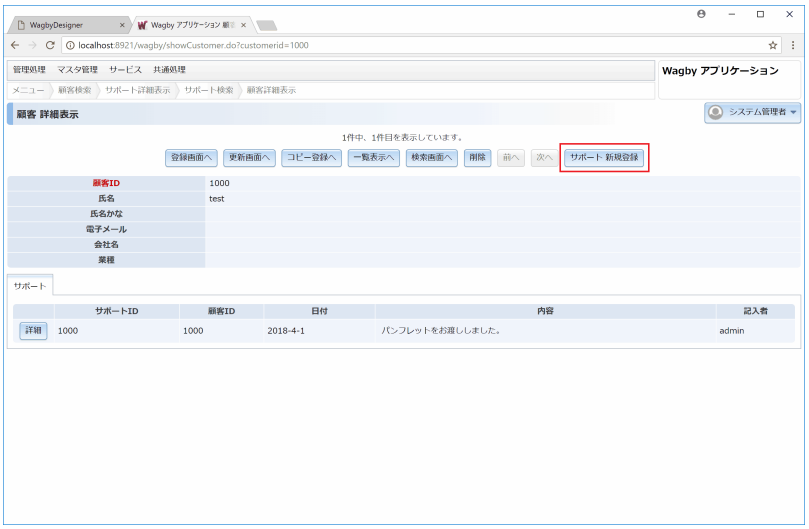


図 親の主キーは空白である

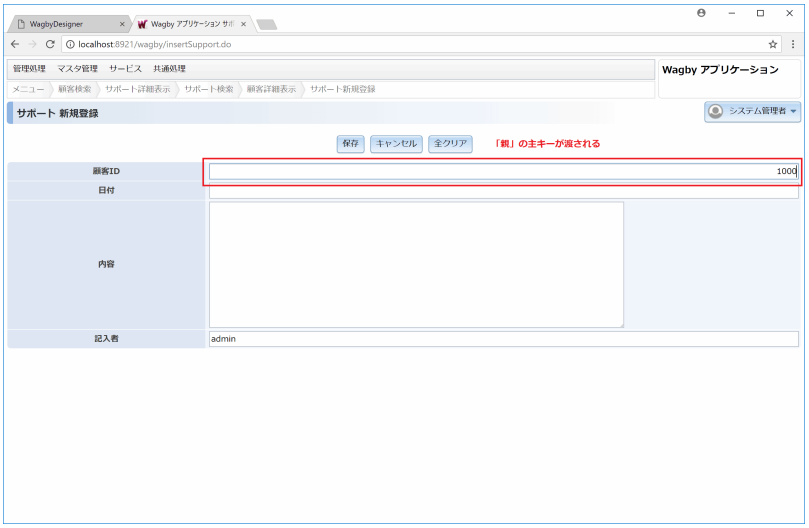


図 親の主キーが転記されている

グリッド表示

外部キーモデルの一覧表示形式がグリッドのとき、子モデル表示もグリッドになります。

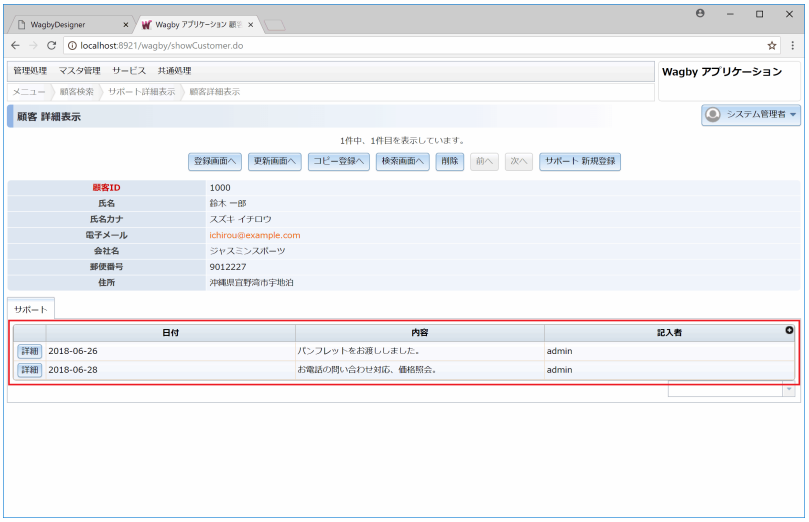


図 子モデルのグリッド表示

(63)

定義方法

子モデルが「グリッド表示」の場合、子の外部キーは検索条件を有効にしてください。

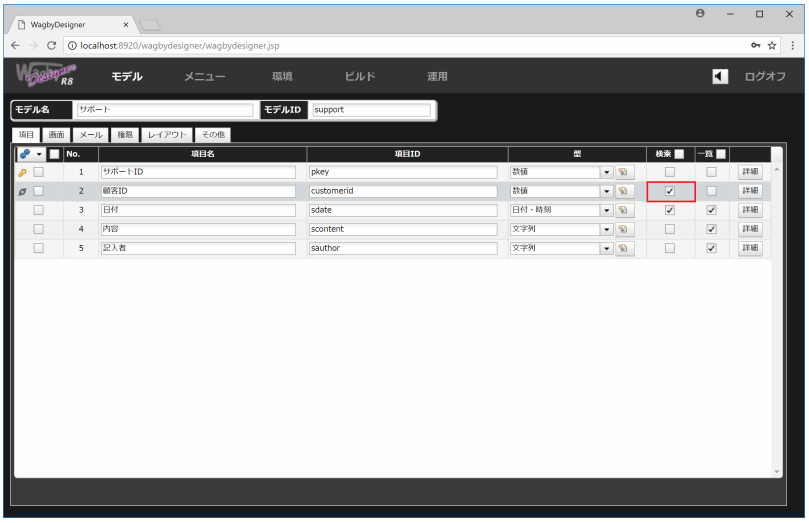


図 外部キーは検索条件を有効にする

外部キーアイコン

外部キー指定された項目は、モデル項目一覧で鎖のアイコンが表示されます。

(63) グリッド表示にはいくつかの制約があります。詳細は"画面機能 > 検索・一覧表示（一覧表示）グリッド形式"をお読みください。

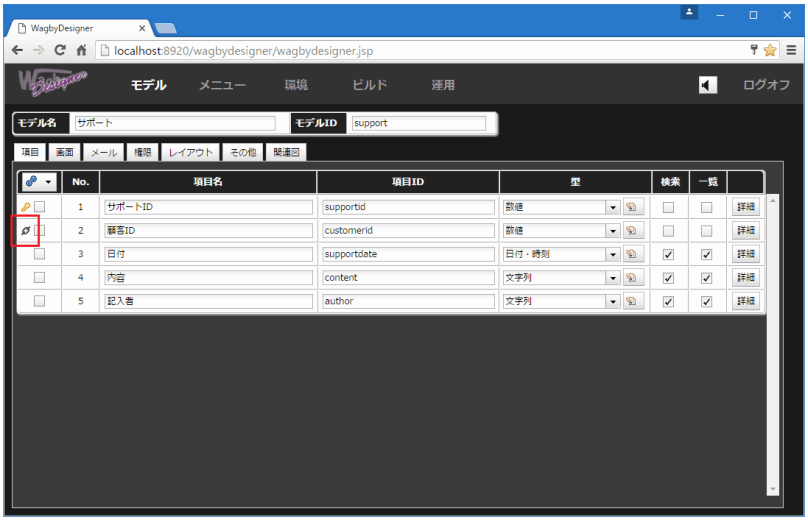


図 鎖アイコンが表示される

子モデルの表示順序を制御する

親モデルの詳細表示画面の下部に用意される子モデル一覧データの並びは、子モデル主キーの昇順となっています。 これを変更することができます。

例えば親p1に紐づく子データc1, c2, c3を作成したとき、標準では、子モデル一覧の表示順は c1, c2, c3 です。 ここで子モデル「主キー」項目のソート順に"-1"を指定するとc3, c2, c1になります。

詳細は"モデル > ソート"をお読みください。

ソートの設定は、親子モデル画面の子モデル表示だけでなく、子モデルの一覧表示画面にも同時に反映されます。（両者は区別されません）

繰り返しコンテナと外部キーの違い

「繰り返しコンテナ」と「外部キー」のいずれも、親モデルと子モデルの間で 1:N の関係を構築できます。両者の違いは次のとおりです。

- 繰り返しコンテナは、親モデル内の "明細" という意味になります。親と強い従属性をもち、親が削除されると自動的に削除されます。
- 外部キーは、親と子に何らかの関連があることを意味します。繰り返しコンテナほど強い従属性をもたず、子モデル側に対する独自の検索処理などを行うこともできます。

参照連動との組み合わせ

外部キーの値を使った参照連動を指定することもできます。ここでは「氏名」項目を表示するようにします。

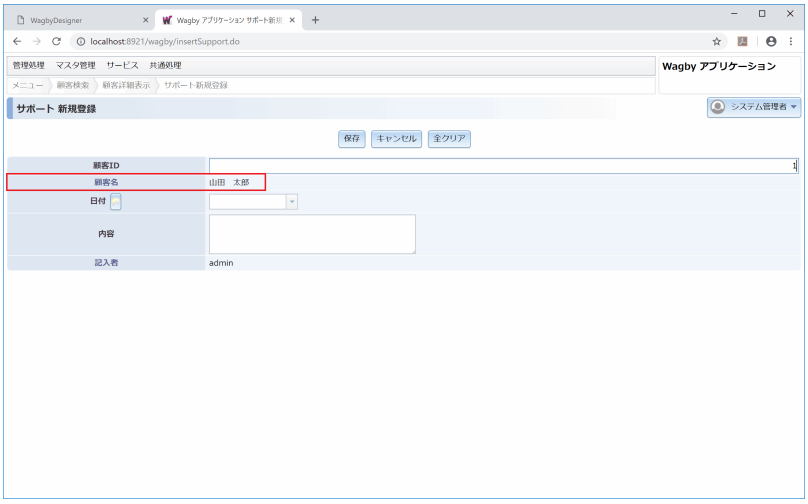


図 外部キーを連動元とした参照連動の実現

定義方法

ここでは「顧客名」項目を用意してみます。

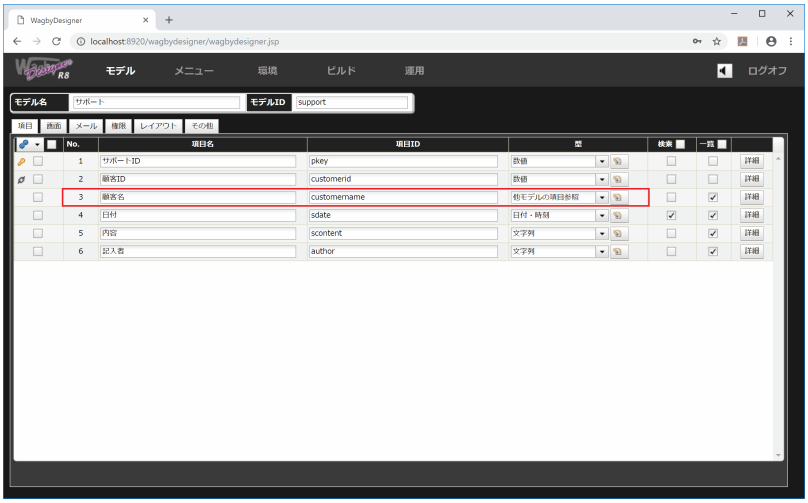


図 顧客名項目を追加する

この顧客名項目は参照連動とします。連動する項目は外部キーである"顧客ID"です。

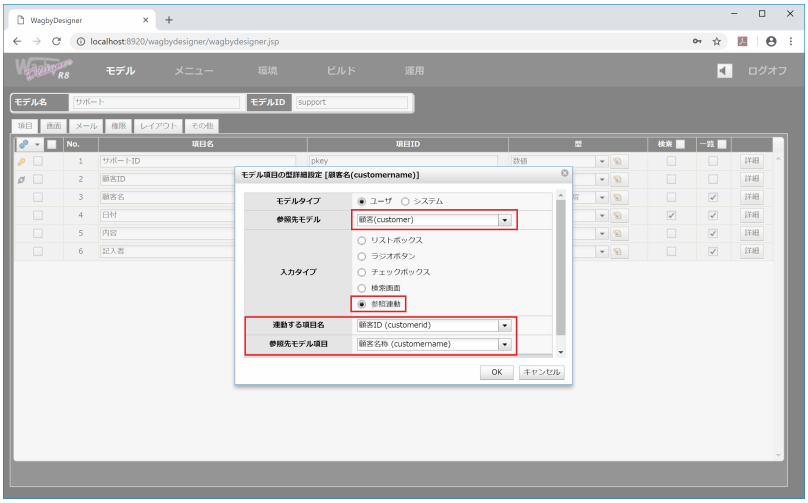


図 参照連動の設定

参照先モデルが複合キーの場合

参照先モデル（上の例では ″顧客″ モデル）の主キーが複合キーの場合、外部キー項目（上の例では ″顧客ID″ 項目）の物理カラム名にコンマ区切りで（参照先モデルの）主キー項目を列挙してください。

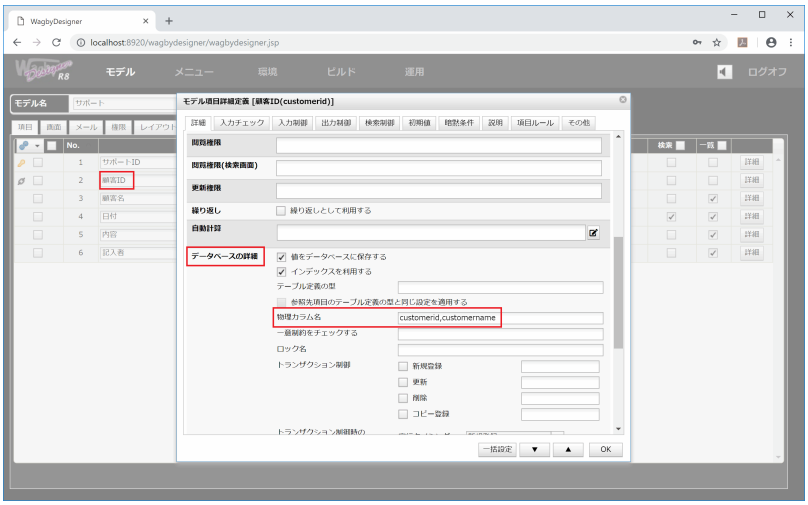


図 物理カラム名を指定する（参照先モデルの主キーがcustomeridとcustomernameだった場合）

トラブルシューティング

子モデル一覧が表示されない

「画面 ＞ 検索・一覧表示 ＞ 外部キー連携 ＞ 親モデルの詳細表示画面に表示するデーター覧」の設定が有効になっていることを確認してください。

子モデル一覧が表示されない(グリッド表示)

子モデルが「グリッド表示」の場合、子の外部キーは検索条件を有効にする必要があります。

親モデルの主キーが複合キーの場合

外部キーを用意する子モデルでは、親モデルの複合キーをすべて含める必要があります。

例えば親モデルの主キーが pkey1, pkey2 のとき、子モデルは pkey1 と pkey2 を外部キーとする項目を保持しなければなりません。どちらか片方のみを保持した子モデルを定義することはできません。

一つの親に複数の子を紐づける

例

外部キーによるモデルの連携は、子を複数指定することができます。顧客モデルを親とし、二つの子「購買履歴」「サポート」モデルを用意した例を図,2に示します。

図 親モデルの詳細画面に、複数の子モデルの新規登録ボタンが用意される

図 子モデルを登録すると（親モデルの詳細画面に）表示される

親モデルにおける表示順

モデルID（英語）の辞書順に並びます。

定義方法

今回用意した購買履歴モデルとサポートモデルの定義を図, 図に示します。 外部キーに指定した項目の型は、参照先の項目の型と同じ型で指定します。ここでは「顧客ID」を「数値」型と指定しています。

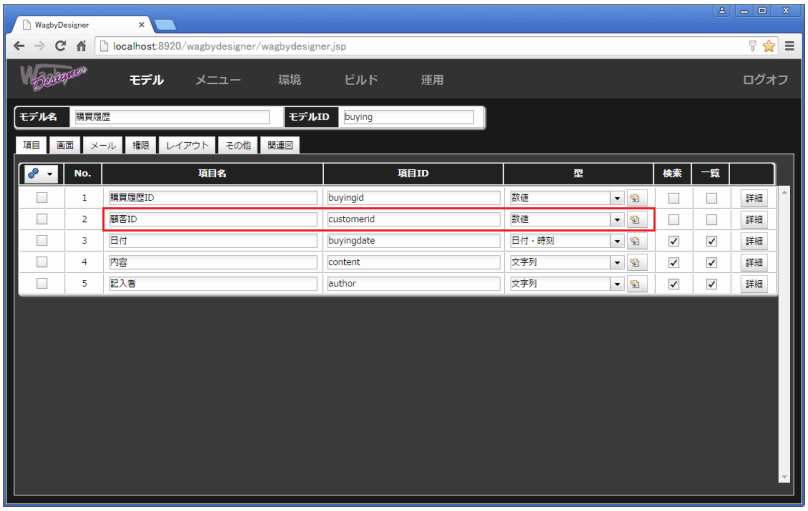


図 購買履歴モデルの定義

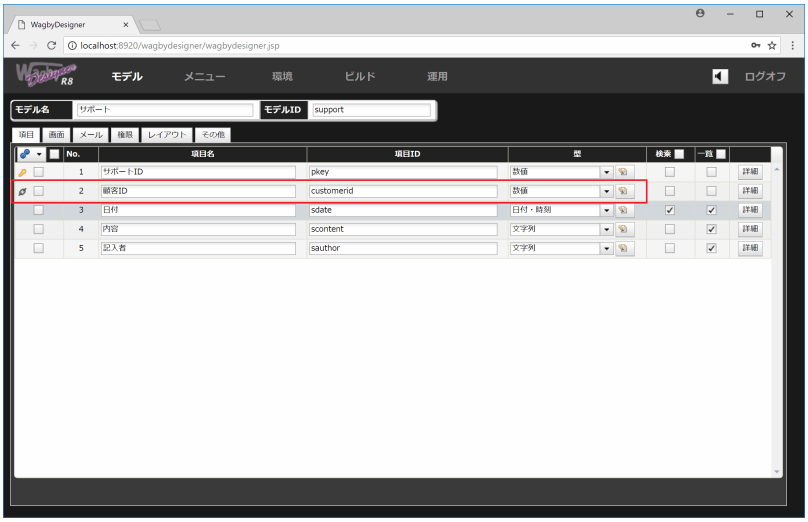


図 サポートモデルの定義

「顧客ID」項目には、外部キーの設定を行います。 「外部キー>モデル名」に、親モデルを指定します。

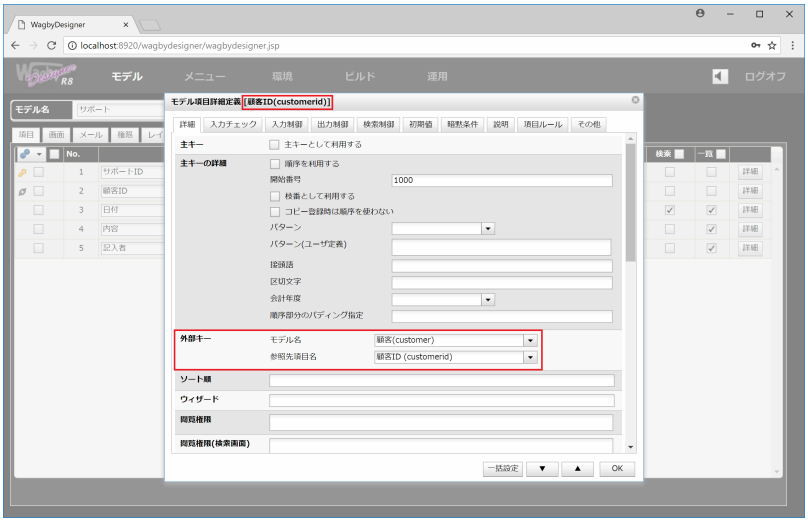


図 外部キーの定義

「参照先項目名」は、子モデル側の項目が同じ名前の場合、省略できます。（今回は同じ "customerid" としたため、省略することができます。）

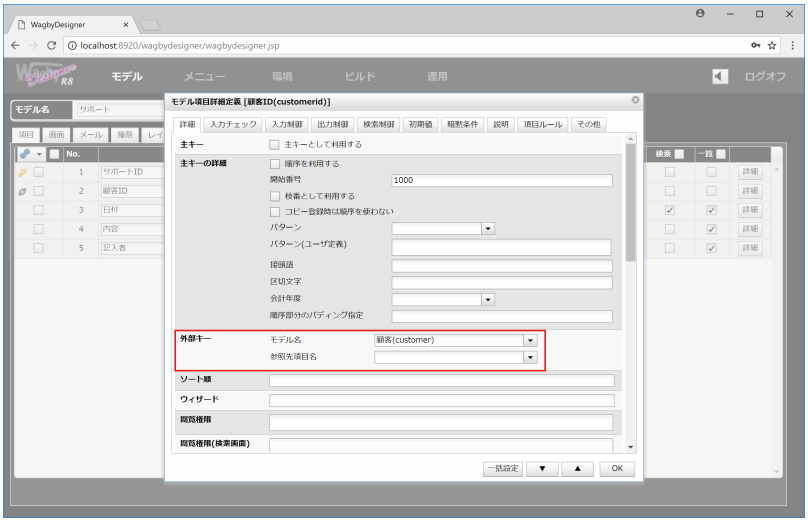


図 外部キーの定義（参照先項目名を省略したとき）

子モデル表示レイアウトの変更（タブ表示と一覧表示）

親モデル詳細画面での子モデルは標準ではタブに格納されます。 子モデルをタブではなく、下図のように縦に並べて表示させることもできます。

図 複数の子モデルを縦に並べて表示させる

定義方法

子モデルの画面定義を修正します。

「画面 ＞ 検索・一覧表示 ＞ 外部キー連携 ＞ 表示方式 ＞ タブレイアウト」のチェックをはずします。標準では、チェックされています。

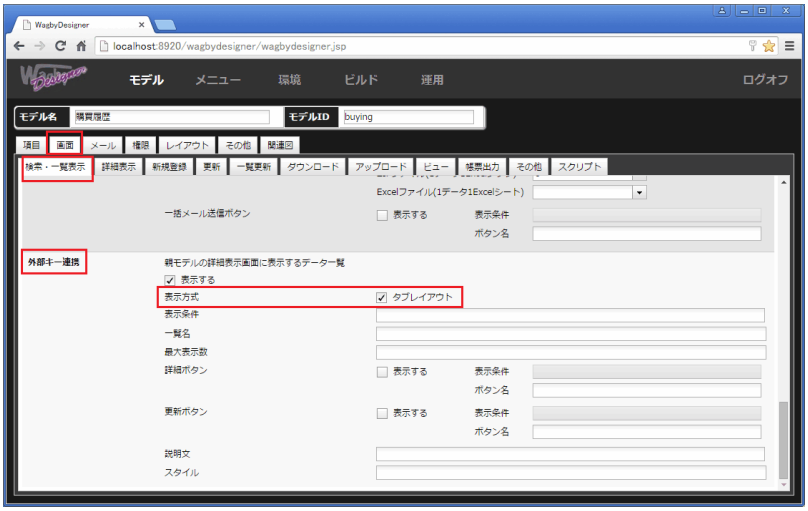


図 タブレイアウトを使う

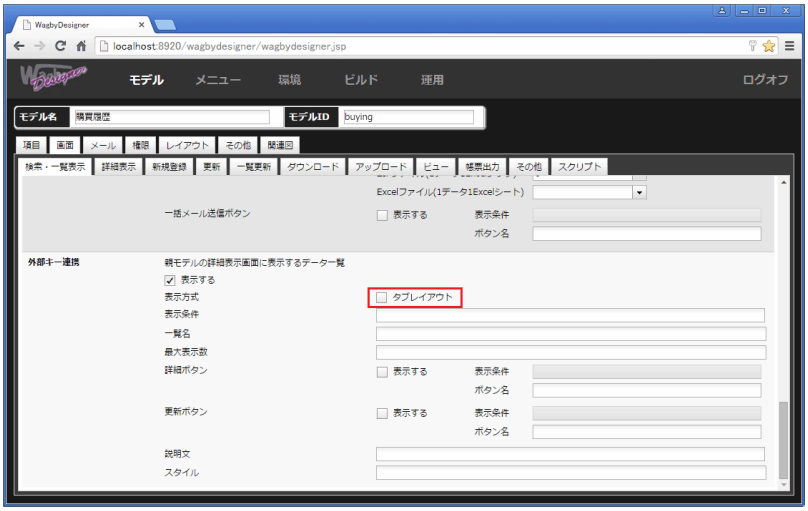


図 タブレイアウトを使わない

（今回の例のように）複数の子モデルがある場合、すべての子モデルでタブレイアウトを使うか、使わないかの設定を行ってください。混ぜることは可能ですが、通常はどちらかに統一します。

子モデル表示の並び

一つの親モデルに複数の子モデルが紐づく場合、表示順序は「子モデルのモデルIDの辞書順」となります。図は、問い合わせ(ask)と、サポート(support)というモデルIDの例です。

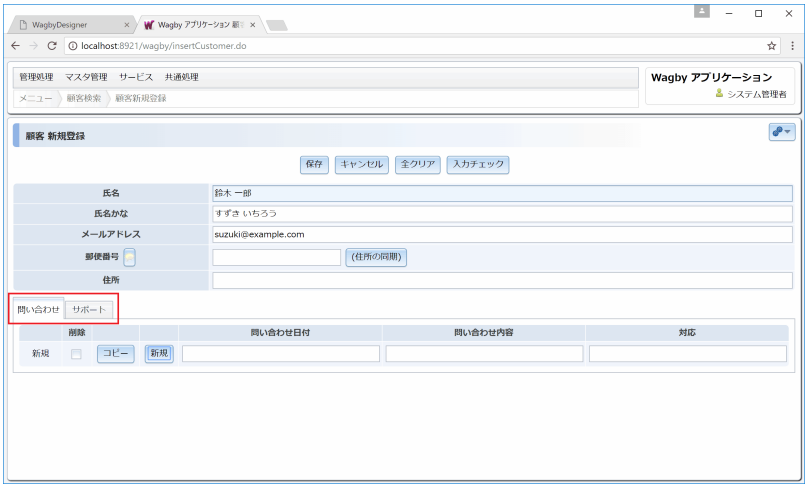


図 子モデルaskとsupportの辞書順に並ぶ

この順序を変更することができます。

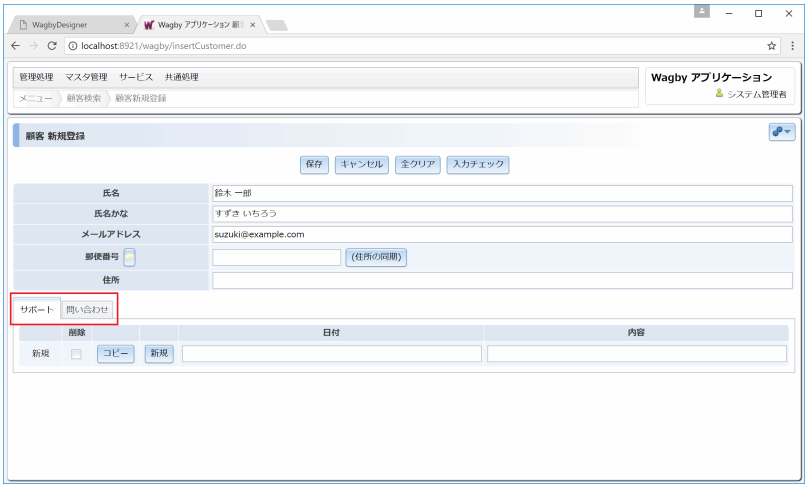


図 子モデルの並び順を変更した例

タブレイアウトを利用しない場合も、この表示順序が適用されます。

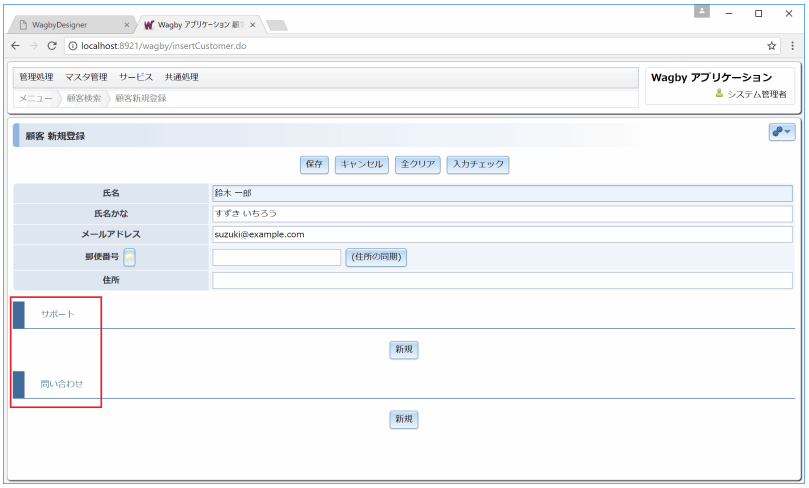


図 タブレイアウトを利用しない場合

定義方法

「画面 > 検索・一覧表示 > 外部キー連携」の「タブの表示位置」を次の書式で指定します。

```
Parent Model ID:Position
```

一つの子に複数の親を紐づける設定を行なった場合、次のようにコンマで区切って指定することができます。

```
Parent Model ID:Position,Parent Model ID:Position,...
```

位置番号は「1」から数えます。

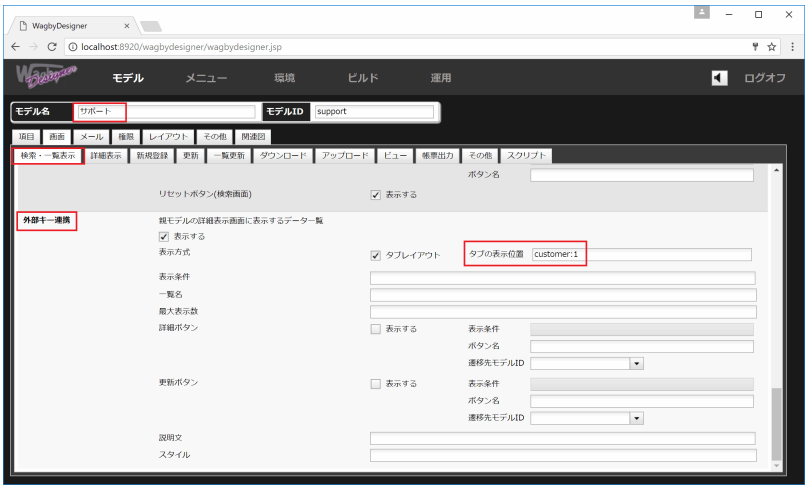


図 子モデル「サポート」の表示位置を1とする

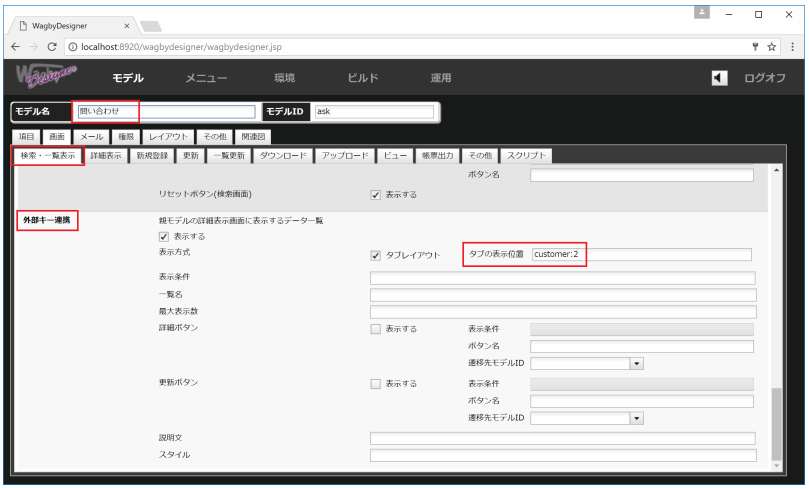


図 子モデル「問い合わせ」の表示位置を2とする

(64) (65)

親モデルが存在しないと子モデルの作成が行えないようにする例

「顧客（customer）」と「サポート（support）」という2つのモデルを題材とします。

最初に「サポート検索」からサポート検索画面を開きます。この時点では、データは1件も登録されていません。

(64) この設定を行う場合は、すべての子モデルについて指定してください。未設定の子モデルが含まれていた場合、未設定が最初に位置づけられます。

(65) 位置番号が重複した場合は、最初のルールであるモデルIDの辞書順が適用されます。

サポートモデルは、「どの顧客モデルに対してのサポートか」という関係を持っています。そのため、顧客モデルを指定せずにサポートモデルを単独で作成することはできないようになっています。

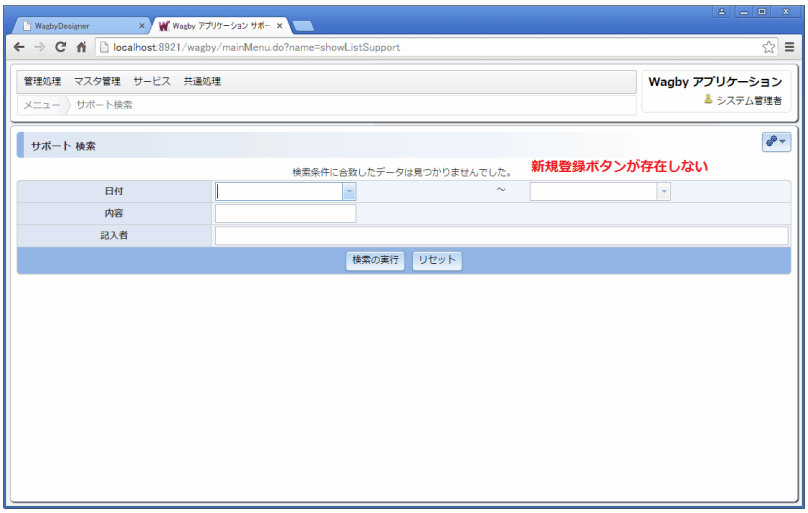


図 サポート検索画面

次に、顧客のデータを新規に登録します。 顧客の詳細表示画面に「サポート新規作成」ボタンが用意されています。 外部キーの関係にあるサポートモデルの新規登録は、対象データ（ここでは顧客データ）の詳細表示画面から行います。

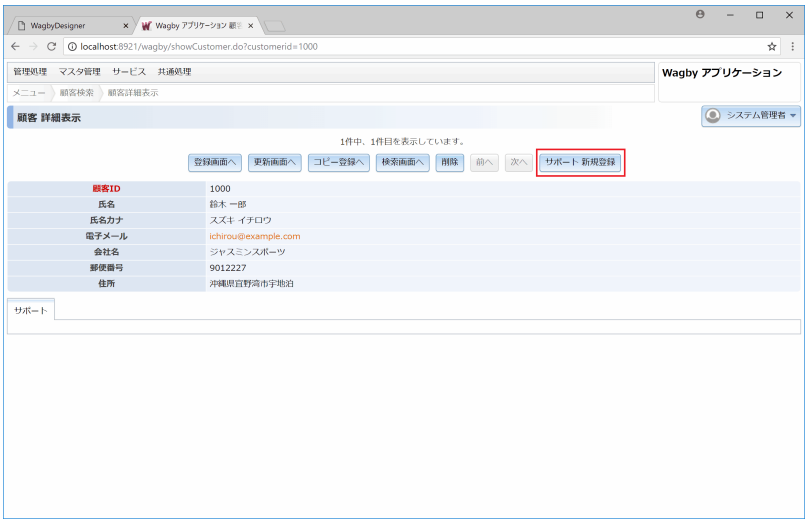


図 顧客詳細表示画面

「サポート新規作成」ボタンをクリックし、サポート新規登録画面を開いてみます。顧客ID（紐づく親モデルのID）は自動設定かつ読み込み専用としています。（親を変更することはできません。）

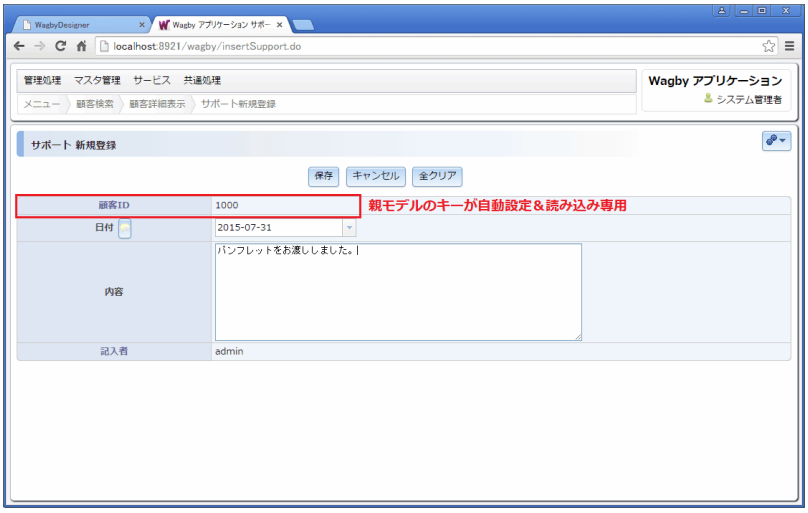


図 サポート新規登録画面

各項目を入力して登録を行います。サポート詳細表示画面から顧客モデルへ戻るための「顧客 一覧表示」「顧客 詳細表示」ボタンが用意されています。

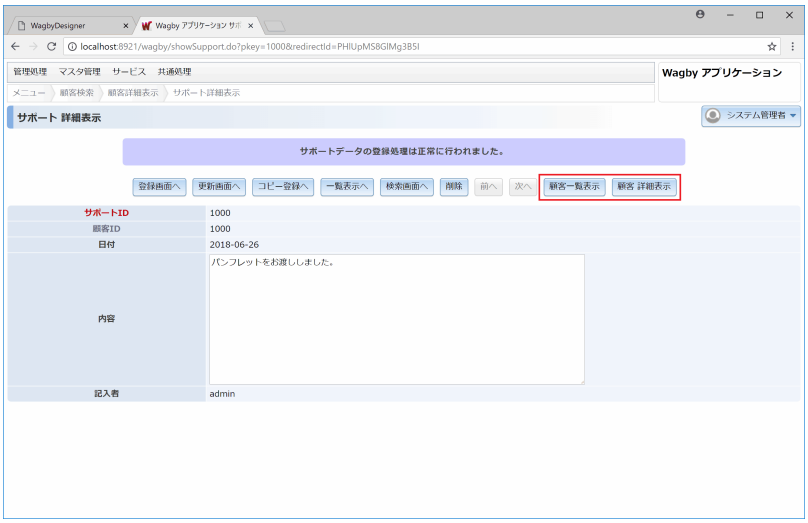


図 サポート詳細表示画面

定義方法

子モデルの検索・一覧画面

子モデル（ここではサポートモデル）の検索・一覧画面で「登録」ボタンを非表示とします。

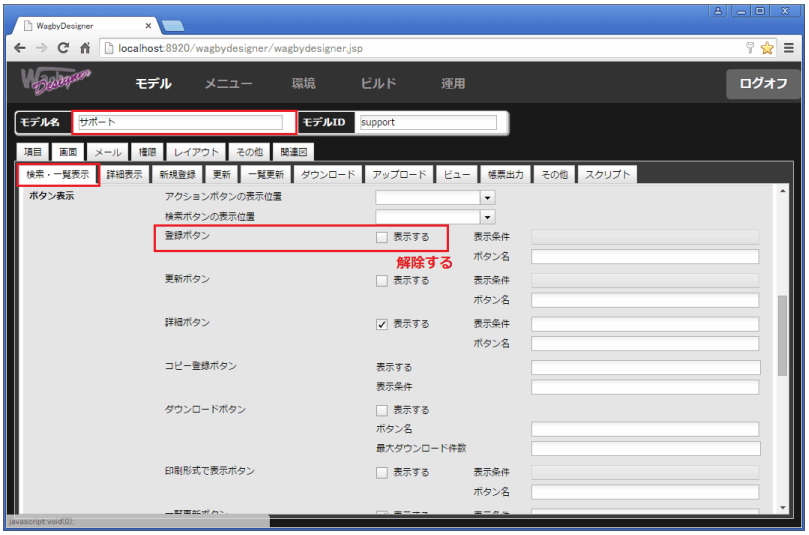


図 新規登録ボタンを表示しない

外部キーを読み込み専用にする

子モデルがもつ外部キー（顧客ID）を読み込み専用にします。

図 外部キー項目を読み込み専用とする

仕様・制約

- 本ページに記載の内容は、子モデルの新規登録ボタンを表示しないようにしていますが、CSVアップロード更新を使うことで親モデルが存在しないデータを作成することは可能です。
- 親モデルが削除された場合には子モデルだけ残り、結果として親モデルが存在しないデータが存在することになります。これを回避するためにカスケード削除を設定することができます。親モデルの削除時に自動的に子モデルも削除されます。

親モデルが存在しない子モデルの作成を許可する

例

「顧客（customer）」と「サポート（support）」という2つのモデルを題材とします。

顧客詳細画面（親）から、サポート（子）を作成する

この場合、親の主キーを子の新規画面に渡すことができます。

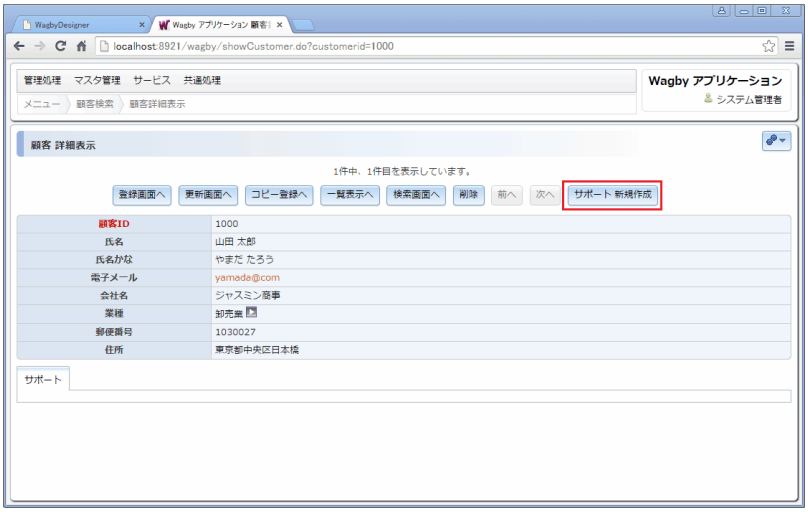


図 顧客詳細画面

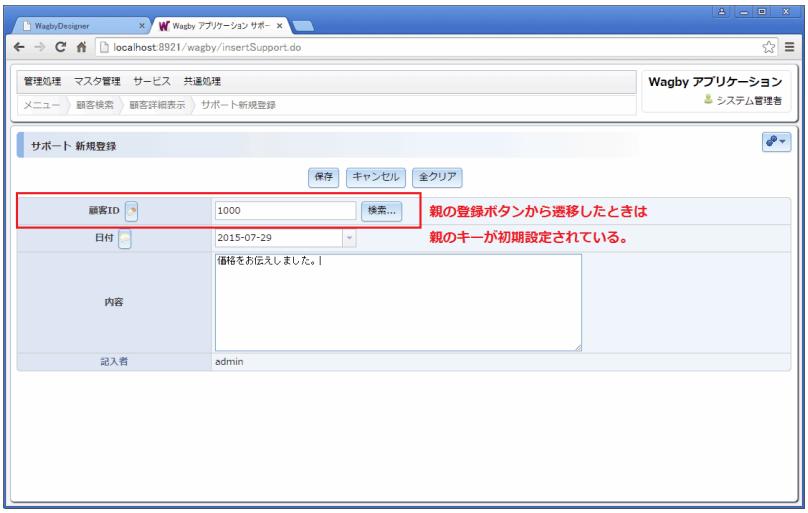


図 サポート新規登録画面

ただしサポート新規登録画面では、親の主キーを変更できるようになっています。

サポート（子）を直接、作成する

サポート検索画面で「登録画面へ」ボタンを押下します。

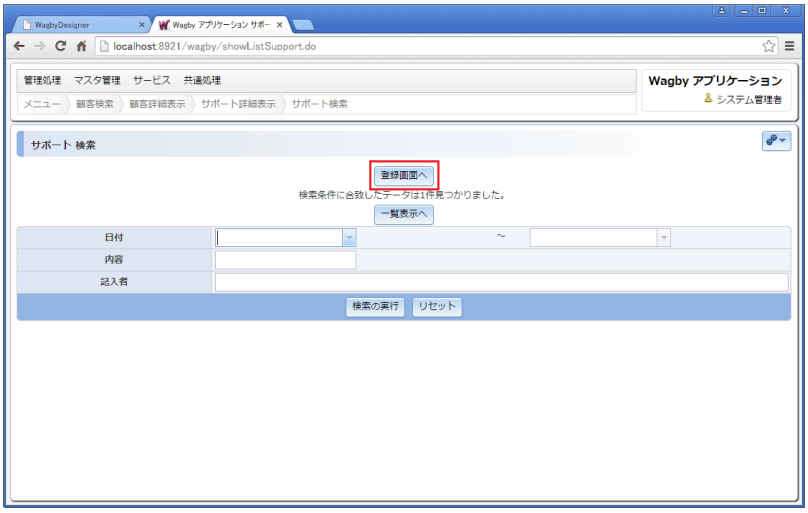


図 サポート検索画面

サポート新規登録画面では、親の主キーを未設定として登録することができます。

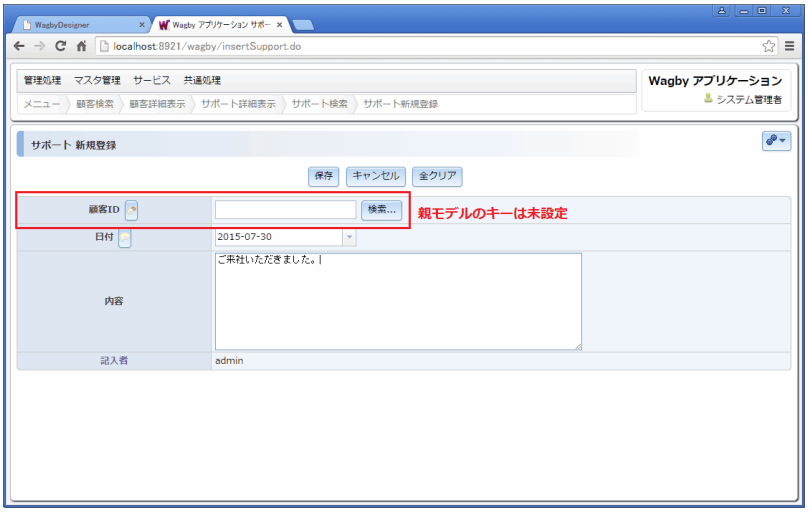


図 サポート新規登録画面

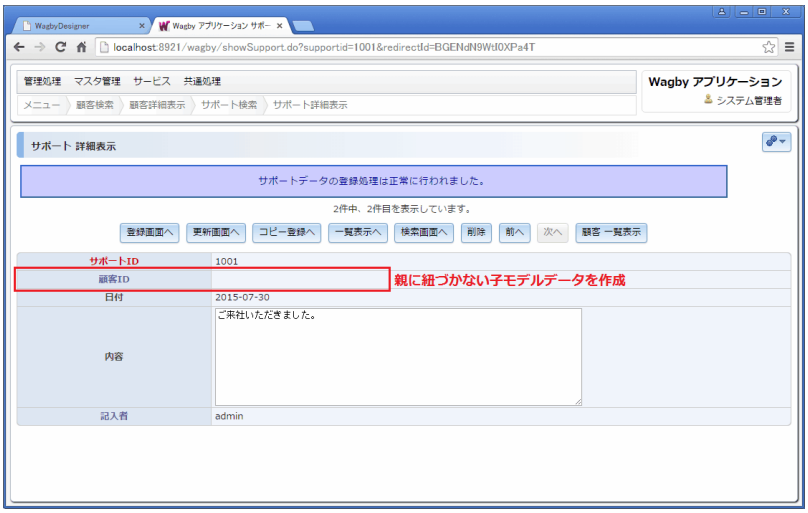


図 サポート詳細画面（親は未設定）

サポート更新画面で、親の主キーを設定することができます。ここではじめて親と紐付きます。

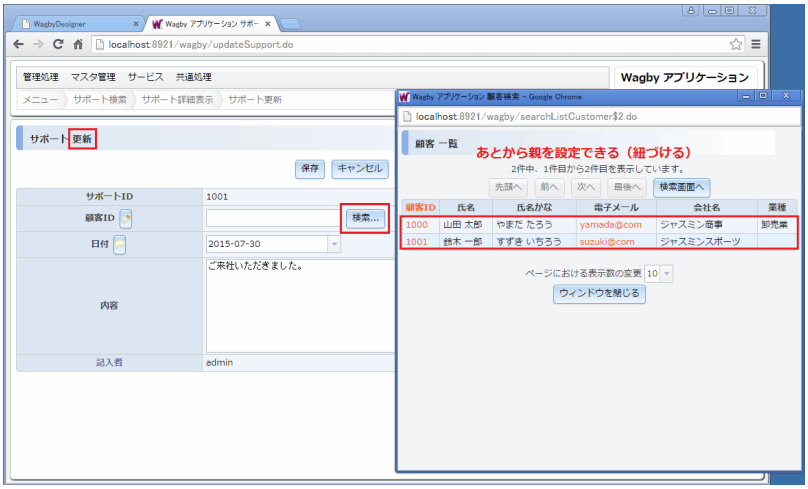


図 サポート更新画面

定義方法

外部キーの型

子モデル（サポートモデル）に含まれる外部キーの型を「他モデルの項目参照」にします。

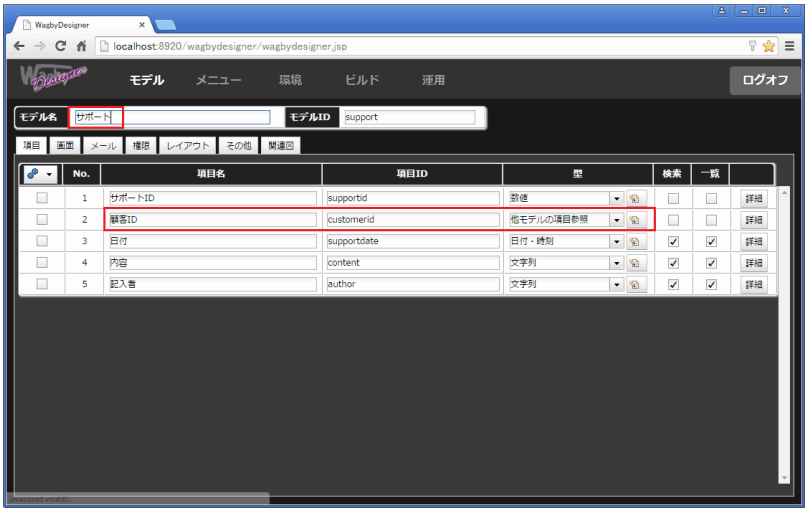


図 他モデルの項目参照を用いる

入力タイプを「検索画面」とします。ここでは、参照先モデル項目に親モデルの主キーを指定しています。

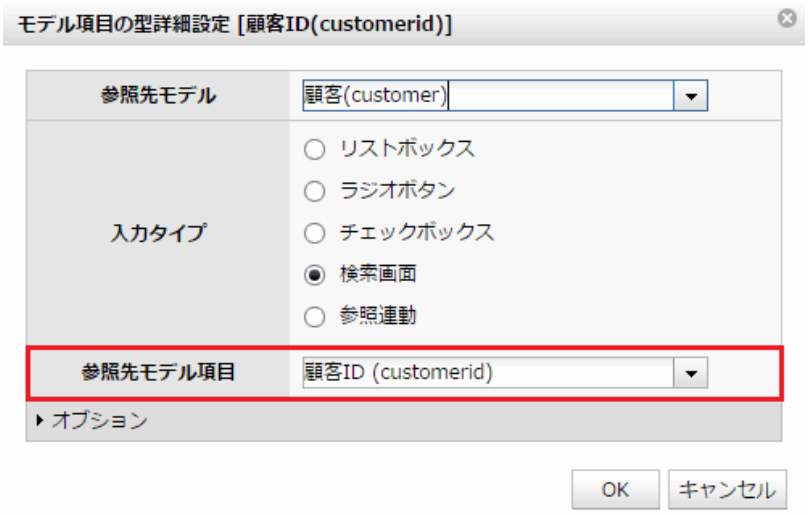


図 参照先モデル項目を親モデルの主キーとした

親の任意の項目で選択できるようにする

例

親モデルの主キー以外の項目を画面表示時に用いることもできます。ここでは「氏名」項目を表示するようにします。

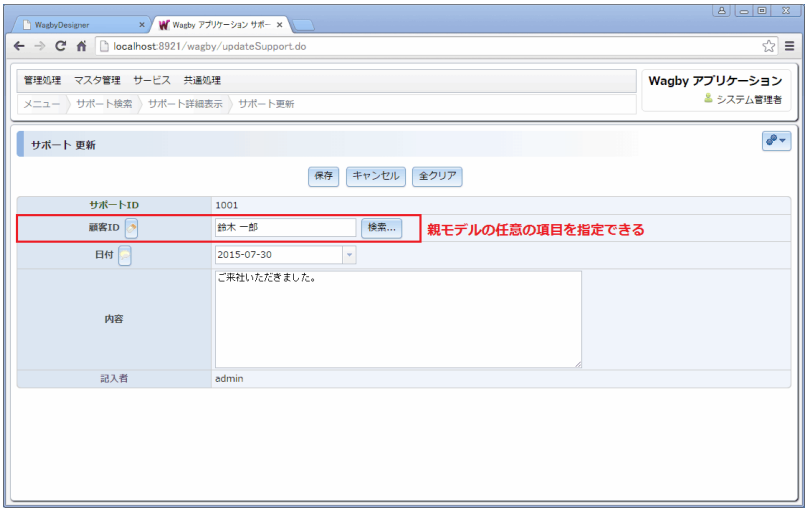


図 親モデルの氏名項目を表示した例

定義方法

図において、参照先モデル項目を親モデルの任意の項目とします。ここでは氏名項目を指定しています。

図 参照先モデル項目を親モデルの氏名項目とした

参照連動との組み合わせ

例

親モデルの主キーを選択し、参照連動を使ってその他の項目を表示させることもできます。ここでは「氏名」項目を表示するようにします。

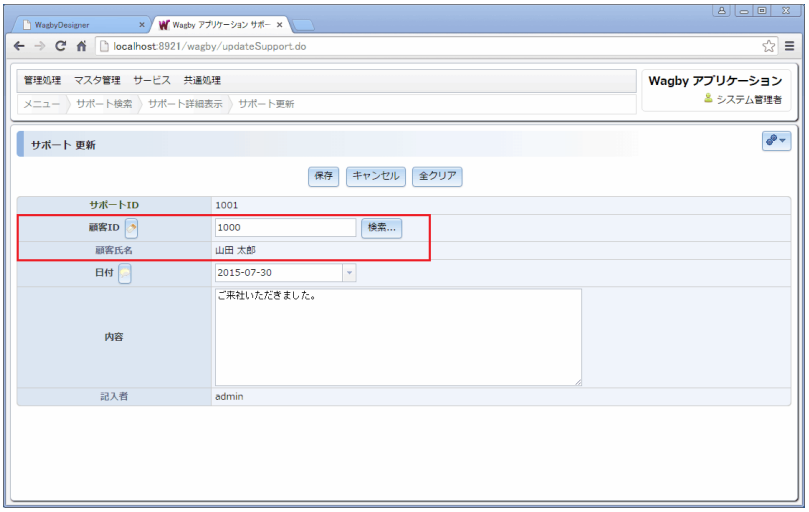


図 参照連動を使って親モデルの氏名項目を表示した例

定義方法

参照連動のための項目を追加します。

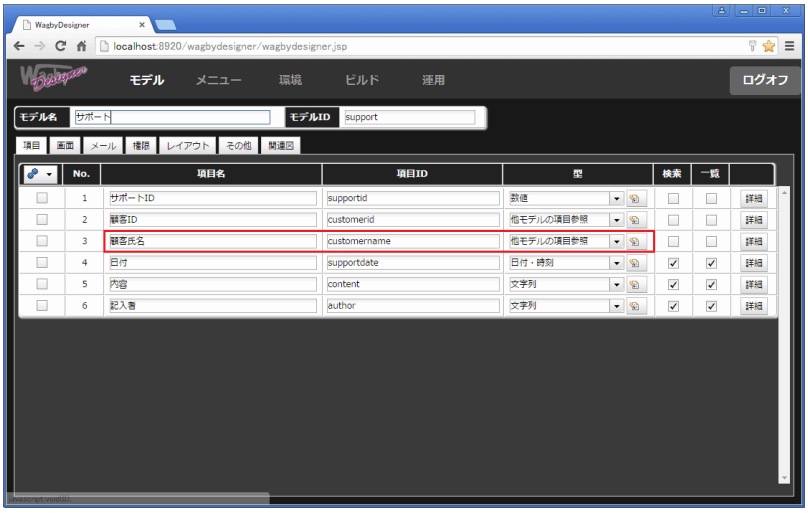


図 参照連動の項目を追加する

図 参照連動の項目を追加する(2)

参照先モデルが複合キーの場合

参照先モデル（上の例では ″顧客″ モデル）の主キーが複合キーの場合、外部キー項目（上の例では ″顧客ID″ 項目）の物理カラム名にコンマ区切りで（参照先モデルの）主キー項目を列挙してください。

仕様・制約

- 子モデルの同時更新機能と併用したとき、親の主キーが未設定のデータは同時更新の対象になりません。

一つの子に複数の親を紐づける

モデルの説明

品目モデル、倉庫モデル、倉庫在庫モデルの関係を示します。倉庫在庫モデルは親となる品目モデル、倉庫モデルの主キーを保持し、かつ、これらを自身の複合主キーとして管理します。

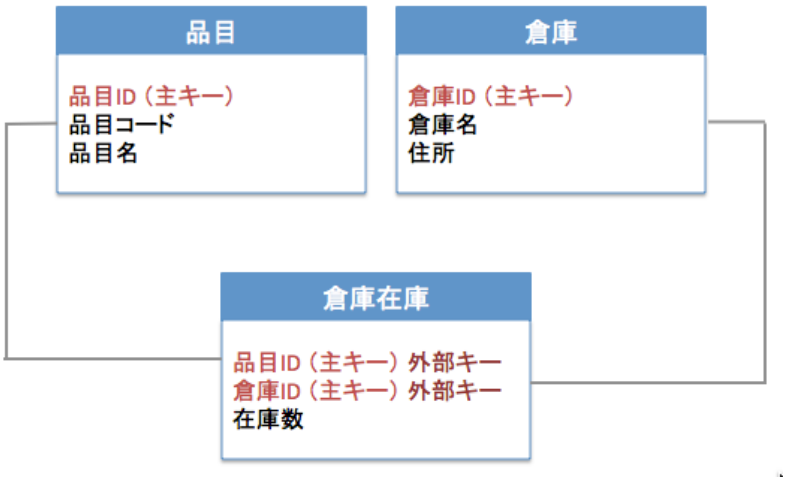


図 品目、倉庫、倉庫在庫の関係

例

倉庫モデル

倉庫モデルのデータ例を示します。

図 倉庫モデル

品目モデル

品目モデルのデータ例を示します。

図 品目モデル

倉庫在庫モデルの登録 － 倉庫から

倉庫モデルの詳細画面から（子である）倉庫在庫モデルの作成を行います。

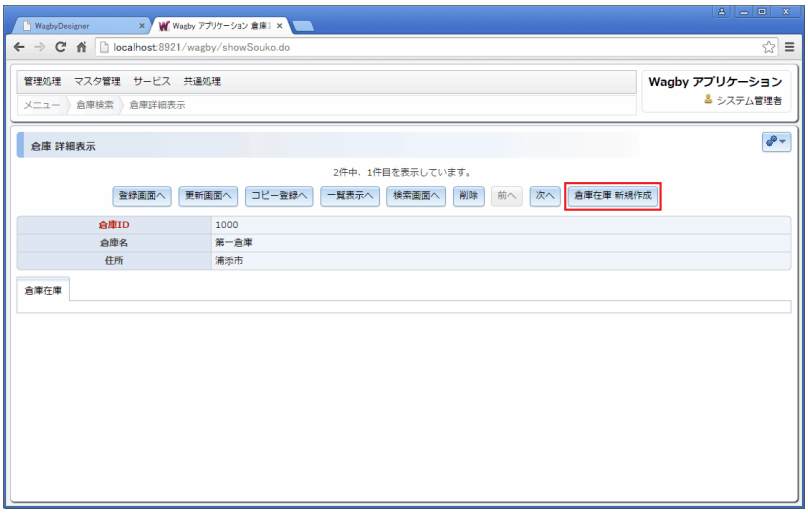


図 倉庫在庫モデルの新規作成

親となる倉庫は、この時点で確定しています。

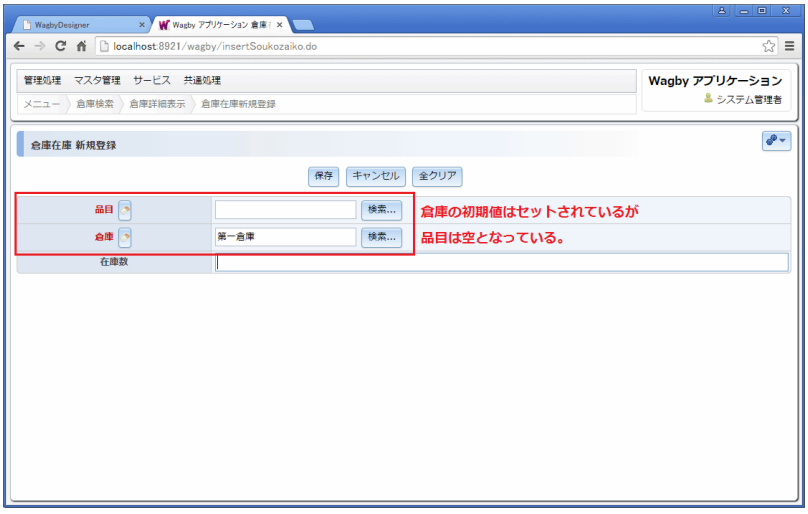


図 親（倉庫）は初期値としてセットされている

もう一方の親である、品目を手動で設定します。品目は主キーであるため、未設定で登録することはできない、というルールです。

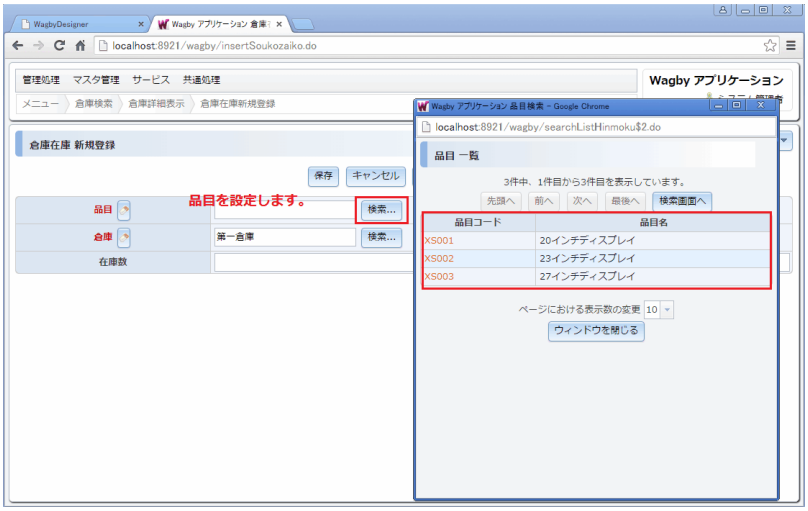


図 品目をセットする

子である倉庫在庫モデルの登録が完了しました。いずれの親の画面へも遷移することができます。

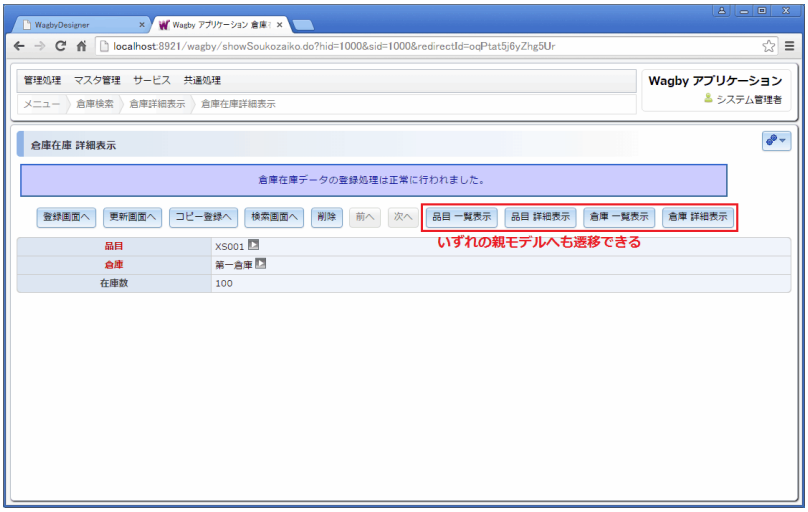


図 倉庫在庫モデルの登録が完了した

倉庫在庫モデルの登録 － 品目から

今度は、品目モデルの詳細画面から（子である）倉庫在庫モデルの作成を行います。

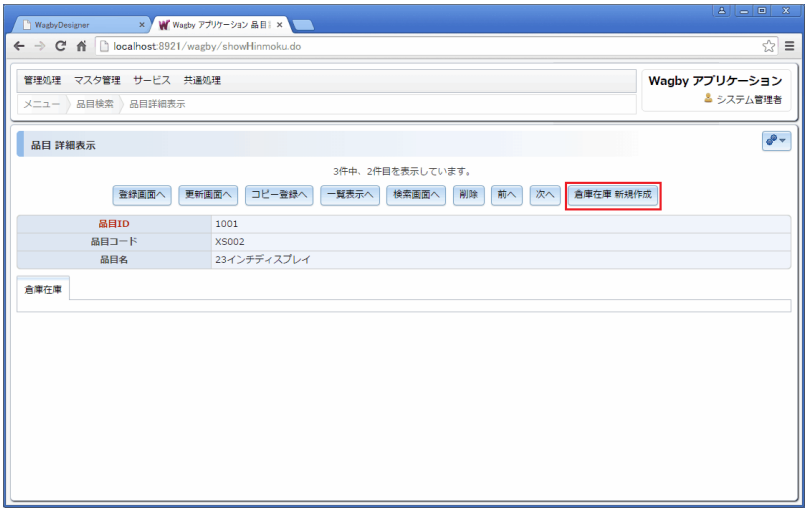


図 倉庫在庫モデルの新規作成

親となる品目は、この時点で確定しています。

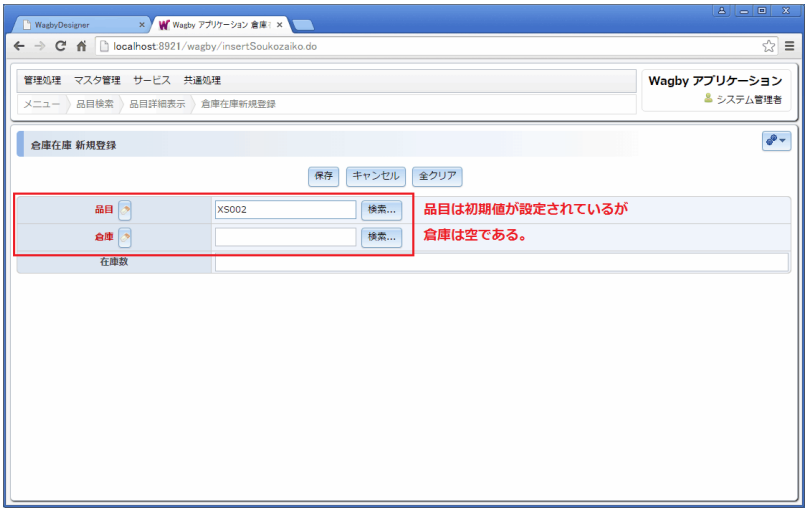


図 親（品目）は初期値としてセットされている

もう一方の親である、倉庫を手動で設定します。倉庫は主キーであるため、未設定で登録することはできない、というルールです。

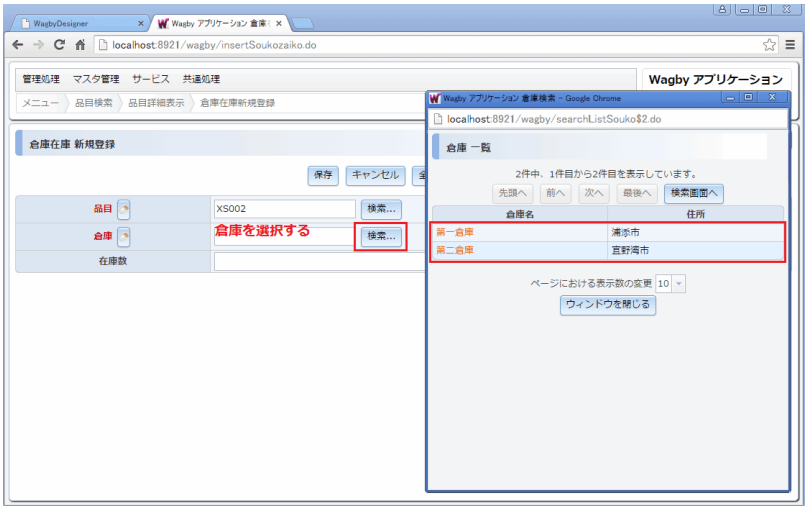


図 倉庫をセットする

子である倉庫在庫モデルの登録が完了しました。いずれの親の画面へも遷移することができます。

図 倉庫在庫モデルの登録が完了した

親の詳細画面に紐づく子の確認

品目モデルの詳細画面では、紐づく（子である）倉庫在庫モデルの一覧を確認できます。

図 品目モデルの詳細画面

同様に、倉庫モデルの詳細画面でも、紐づく（子である）倉庫在庫モデルの一覧を確認できます。

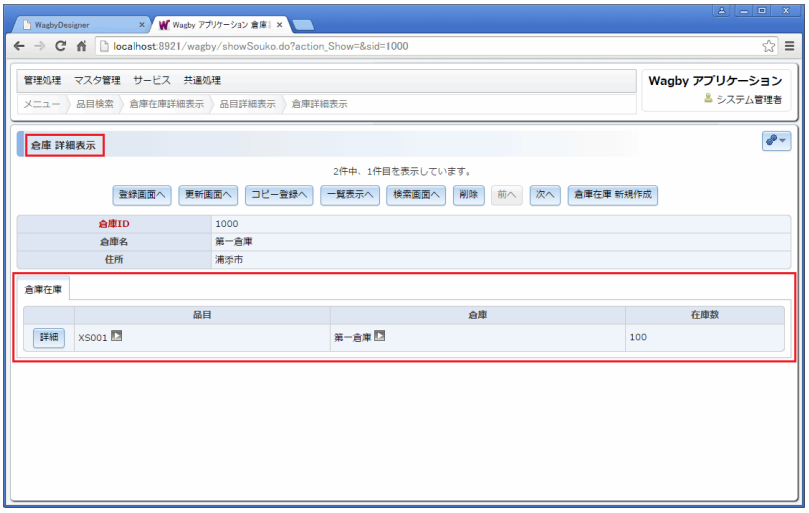


図 倉庫モデルの詳細画面

定義方法

3つのモデル

今回、用意する品目、倉庫、倉庫在庫モデルを定義します。

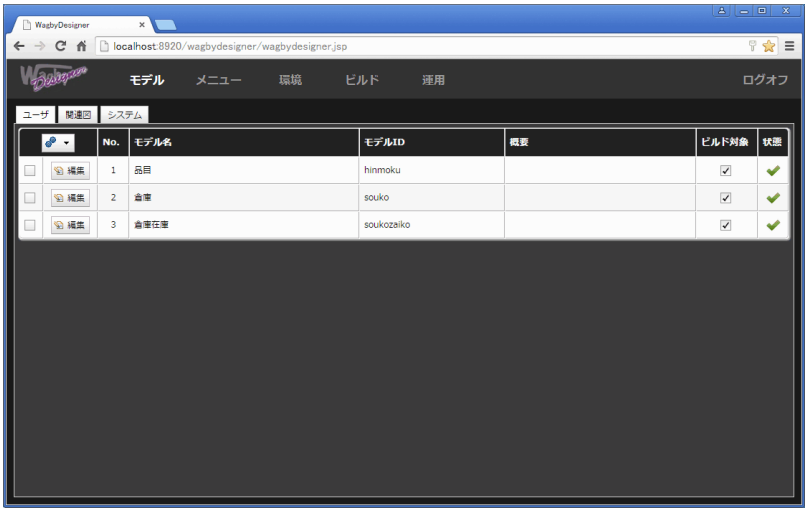


図 定義するモデル一覧

品目モデル

品目モデルは、品目コードと品目名を管理します。

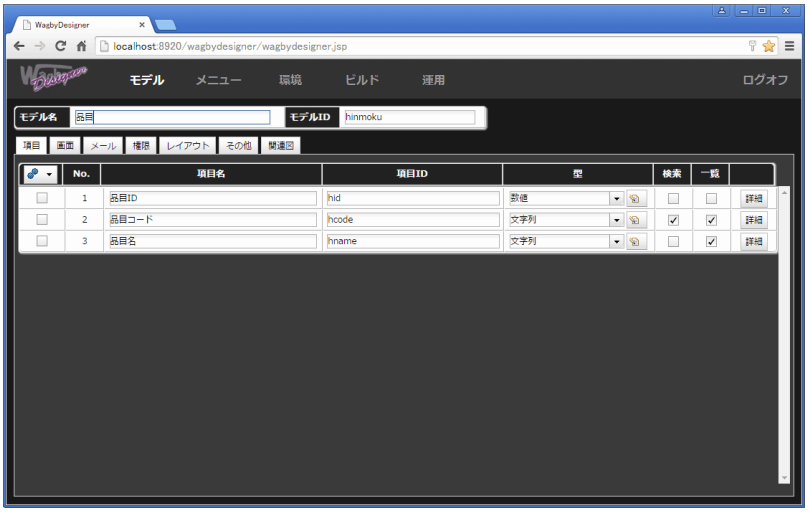


図 品目モデルの定義

倉庫モデル

倉庫モデルは、倉庫名と住所を管理します。

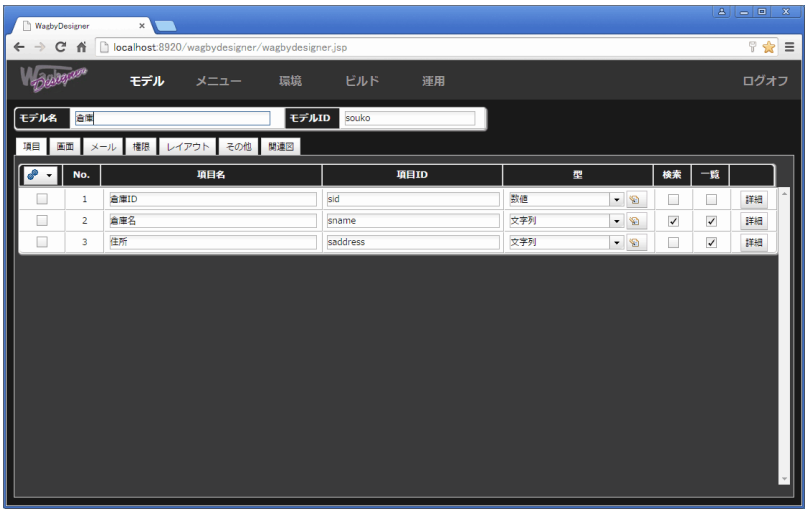


図 倉庫モデルの定義

倉庫在庫モデル

倉庫在庫モデルは、品目と倉庫を主キーかつ外部キーとして用意します。

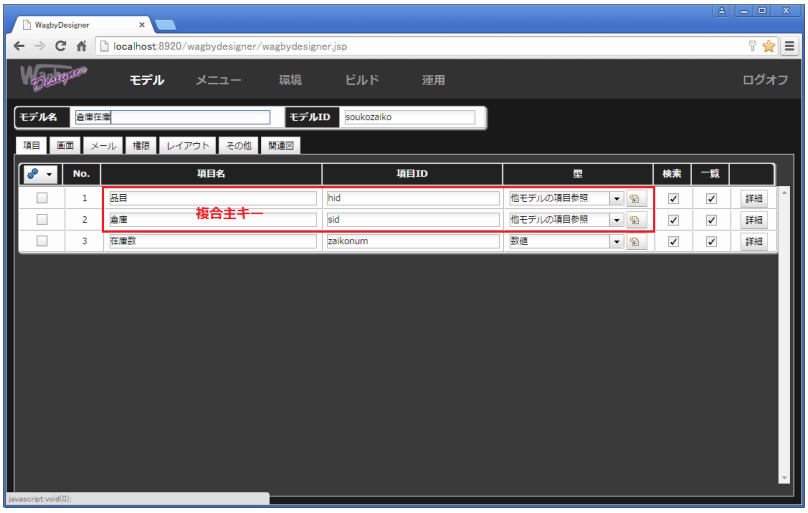


図 倉庫在庫モデルの定義

「品目」項目はモデル参照で、主キーかつ外部キーとします。

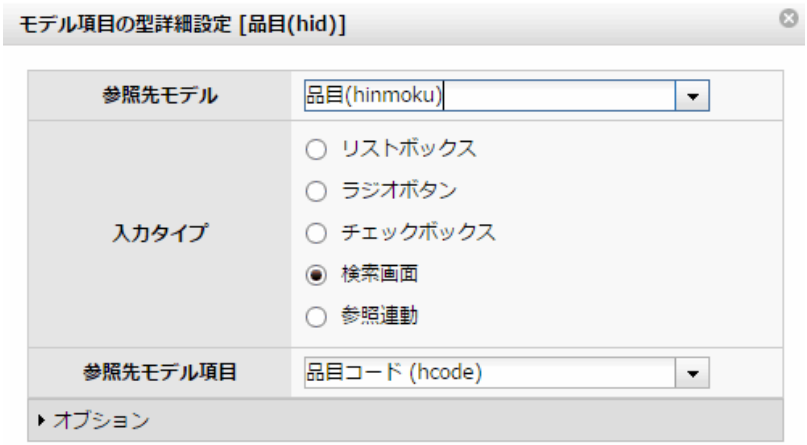


図 品目項目のモデル参照定義

図 品目項目の主キー、外部キー定義

「倉庫」項目もモデル参照で、主キーかつ外部キーとします。

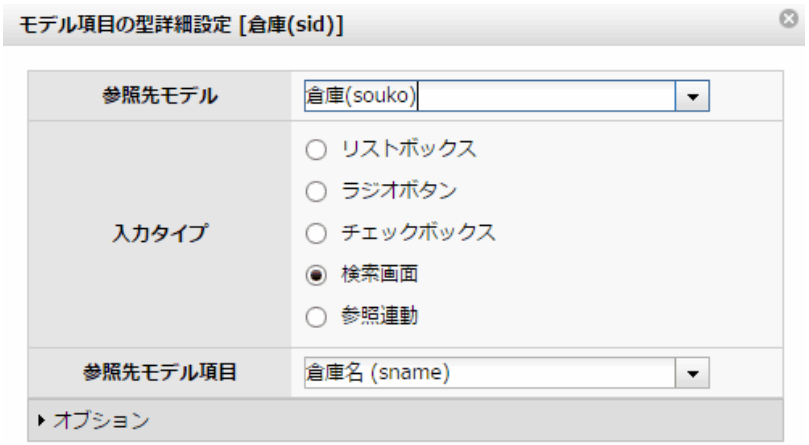


図 倉庫項目のモデル参照定義

図 倉庫項目の主キー、外部キー定義

親キー以外の項目値を用いて子モデル一覧表示を行う(スクリプト)

例

「週報」モデルを親(parent)とし、「行動計画」モデルを子(child) とした例を用いて説明します。

通常、行動計画モデルは、自分の親となる週報モデルのキーを（外部キーとして）保持します。しかし図,図からわかるように、外部キー「週報ID」を未設定としています。

実はこのサンプルアプリケーションは、行動計画（子）の「活動日」が、週報（親）の「開始日」と「終了日」の期間に含まれるデータが一覧表示されるようになっています。図のログオンアカウントは　″佐藤″で、図のログオンアカウントは ″山田″ ですが、週報IDが未設定であるため、それぞれの週報の子モデルとして（同じ行動計画データが）表示されていることがわかります。

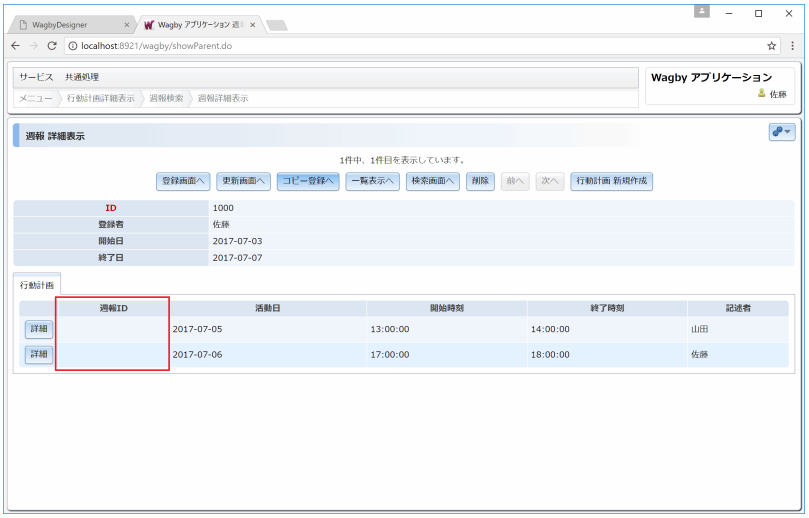


図　ログオンアカウント"佐藤"の週報

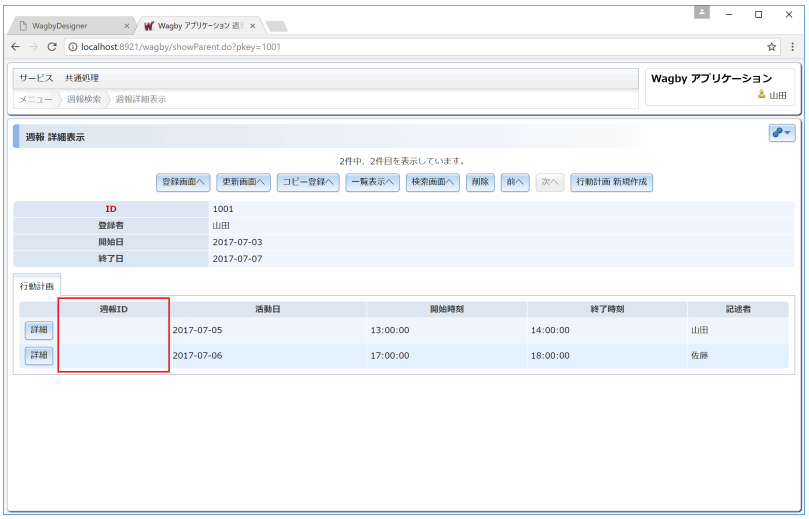


図　ログオンアカウント"山田"の週報

このため、行動計画の策定時は「どの週報に紐づくか」を未設定としたままデータの新規登録が可能です。運用しながら、紐づく親を指定しますが、変更も行うことができます。それでいて、週報（親）の子モデルとして、期間による絞り込みで行動計画（子）を表示することができます。

週報（親）を確定させる

図は、行動計画データの一つについて、週報ID（親の主キー）を設定した例です。週報IDが"1001"の週報（これは"山田"の週報です）に行動計画（子）が表示されています。

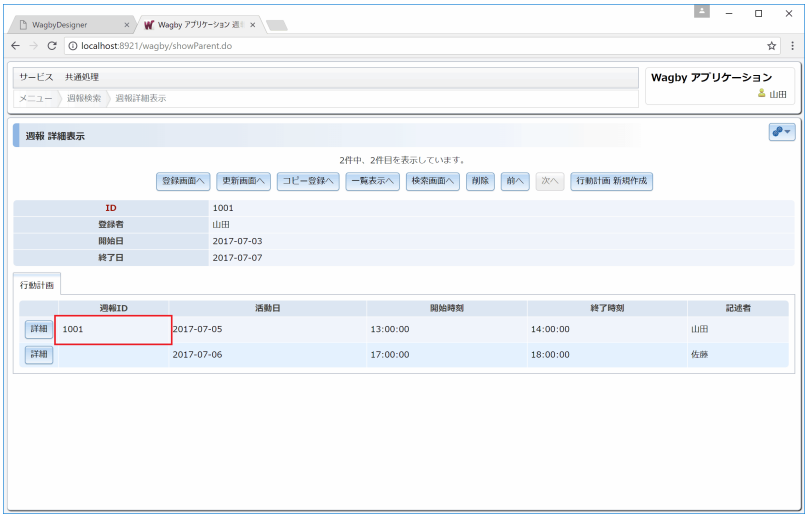


図 週報IDが決まった場合

″佐藤″の週報では、このデータは表示されないようになります。

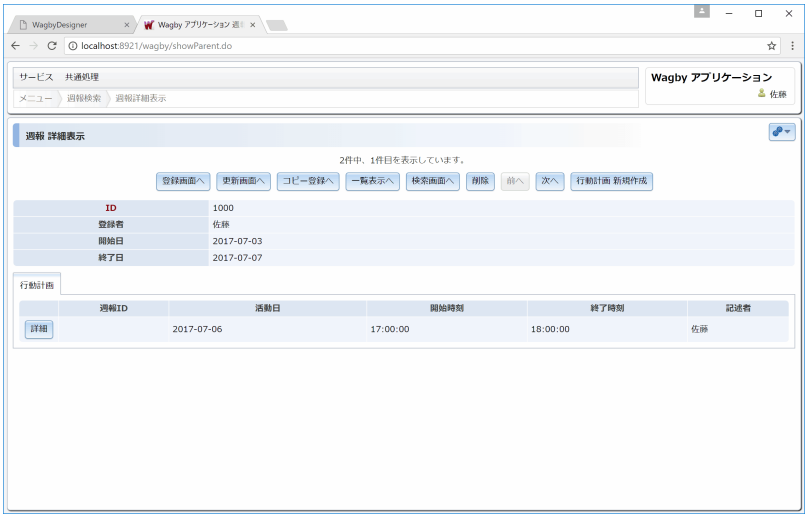


図 ″佐藤″の週報

システム管理者はすべてのデータが閲覧できる

なお、システム管理者はすべてのデータが閲覧できることに注意してください。

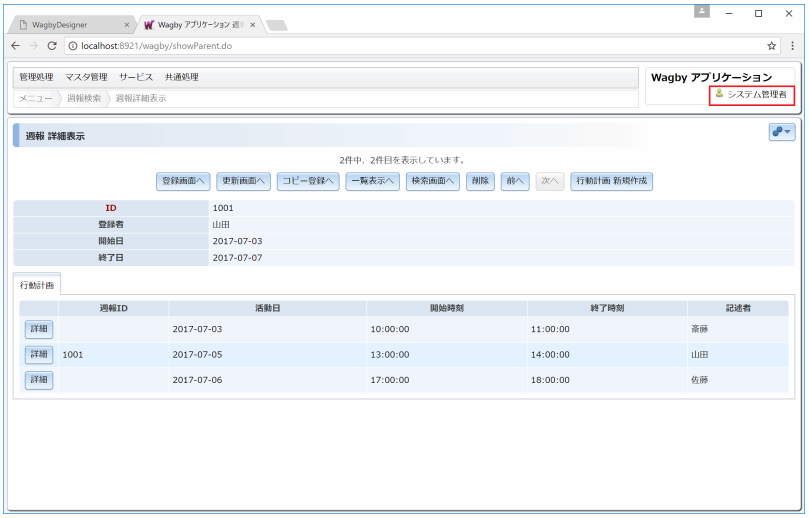


図 システム管理者の場合

定義方法

週報モデルと行動計画モデル

このアプリケーションでは、週報モデルを parent、行動計画モデルを child としています。

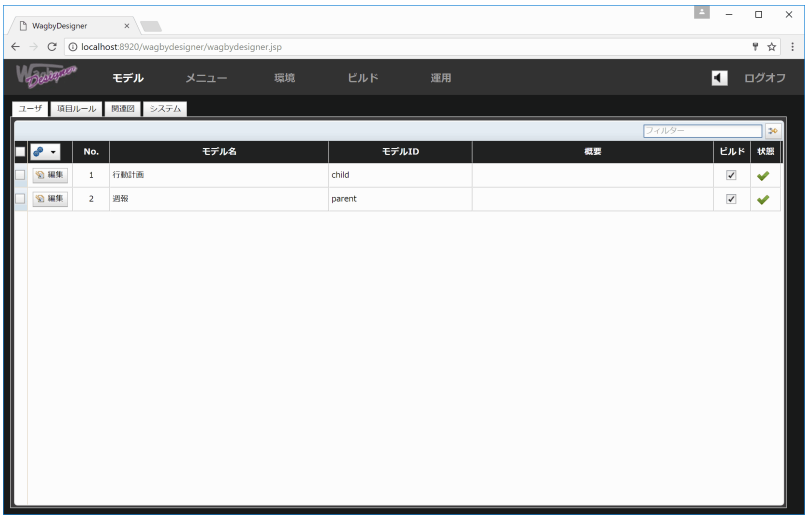


図 週報モデルと行動計画モデル

行動計画モデルでは、週報IDは外部キーとなっていることがわかります。

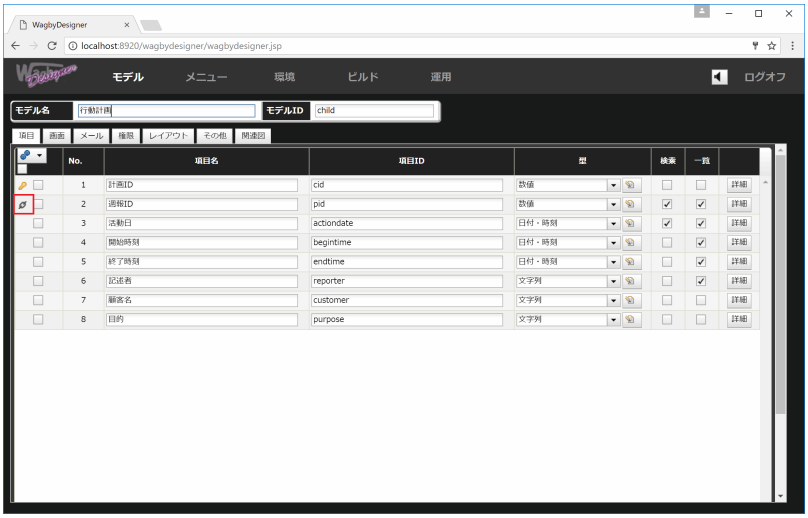


図 外部キーの設定

スクリプトの設定（親）

週報（親）の詳細表示処理時、週報の「開始日」を p.request オブジェクトにセットしています。

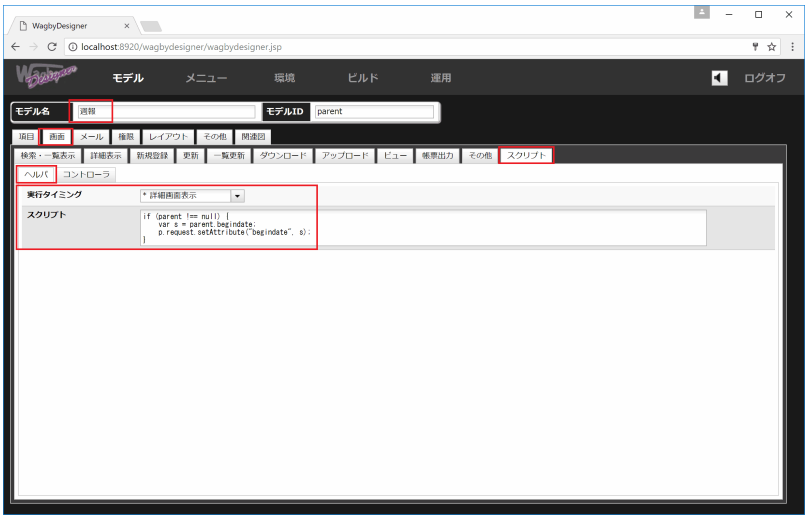


図 詳細表示のヘルパスクリプト（親）

```
if (parent !== null) {
    var s = parent.begindate;
    p.request.setAttribute("begindate", s);
}
```

暗黙条件の設定（子）

行動計画モデルでは、actiondate 項目を検索条件に設定します。このとき「範囲検索」は無効とします。

設定された値 "begindate" を使って絞り込み条件を設定しています。

```
(java.sql.Date)ATTRIBUTE("begindate")
```

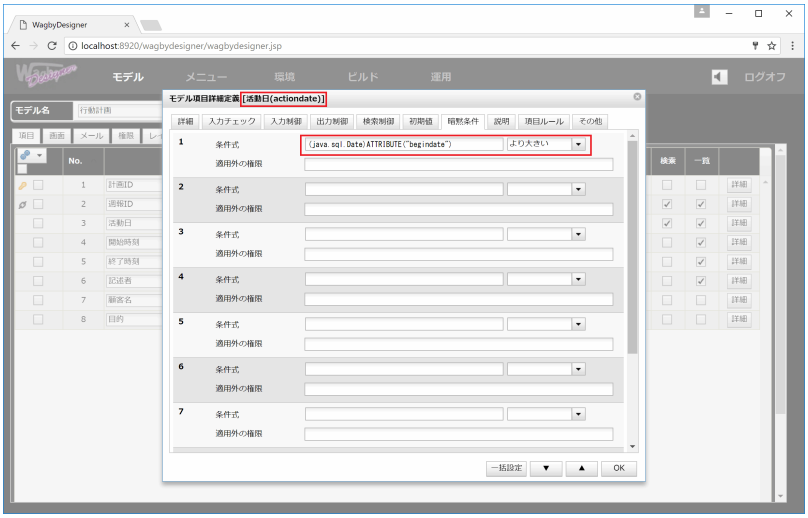


図 暗黙条件の設定（子）

仕様・制約

ATTRIBUTE関数について

今回の例では、一回の処理の間に、親のヘルパクラスの beforeShow メソッドが呼ばれ、そのあとに子の一覧表示が行われるという「順序」も重要です。この順序に従って、親の処理で p.request に「開始日」の情報をセットし、子の一覧表示（のときに用いられる暗黙条件のルール）で、その「開始日」を使っています。このとき、ATTRIBUTE 関数を使って値を取得することができます。

(66)

制約

・ この設定を行う場合、子モデルの一覧表示にグリッド形式を利用することはできません。

連携に関する設定

子モデル表示のレイアウト

親モデル詳細画面での子モデルは標準ではタブに格納されます。子モデルをタブではなく、縦に並べて表示させることもできます。 詳細は"一つの親に複数の子を紐づける > 子モデル表示レイアウトの変更（タブ表示と一覧表示）"をお読みください。

一覧のタイトルをリンクにする

「顧客」モデルに外部キーで連携するモデル、「サポート」の一覧表示部分のタイトルをリンクにすることができます。

(66) 暗黙条件に ATTRIBUTE 関数を記述できます。

図 タイトルがリンクになる

リンククリックにより「サポート」の一覧表示画面に遷移します。「顧客」の詳細表示画面で確認していた一覧を確認することができます。

図 一覧表示画面に遷移する

次に検索画面をみてみましょう。図で表示していた「顧客ID」が自動的にセットされていることがわかります。

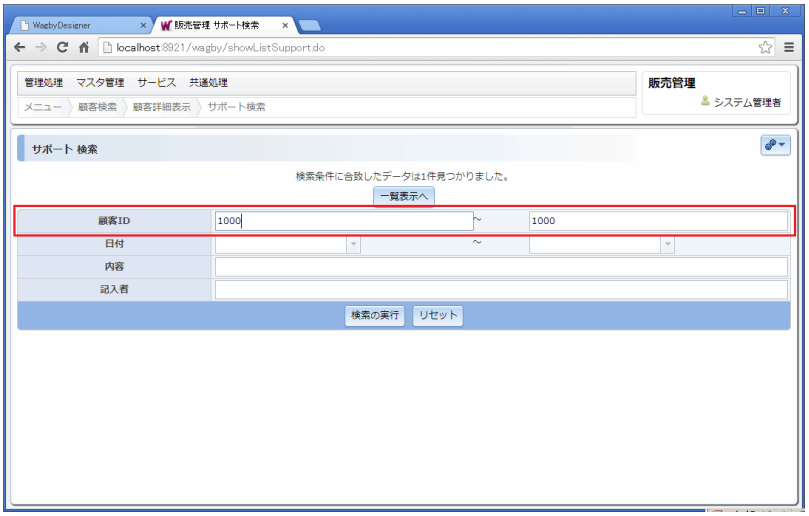


図 サポートの検索画面

定義方法

サポートモデル側（子）で、外部キーとしている「顧客ID」（顧客モデルの主キー）を検索条件に指定します。

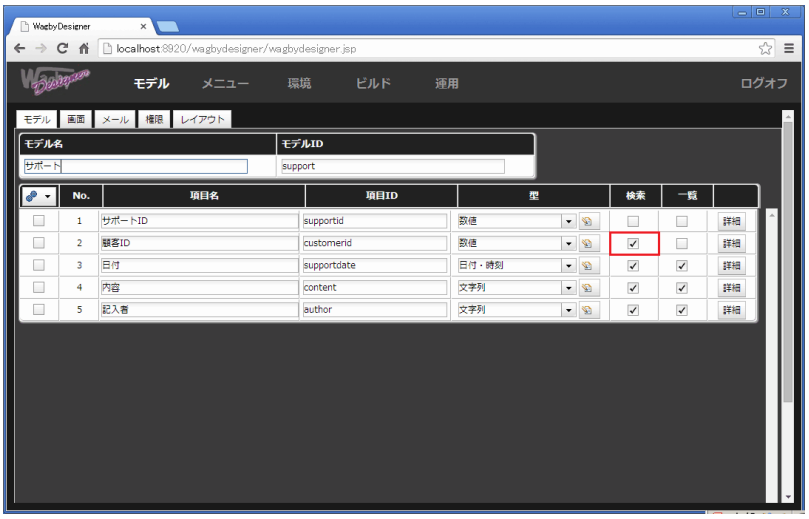


図 （サポートモデルの）顧客IDを検索条件に含める

外部キー項目の、モデル項目詳細定義ダイアログを開きます。 「データベースの詳細>登録時に値をデータベースに保存する」を有効にします。

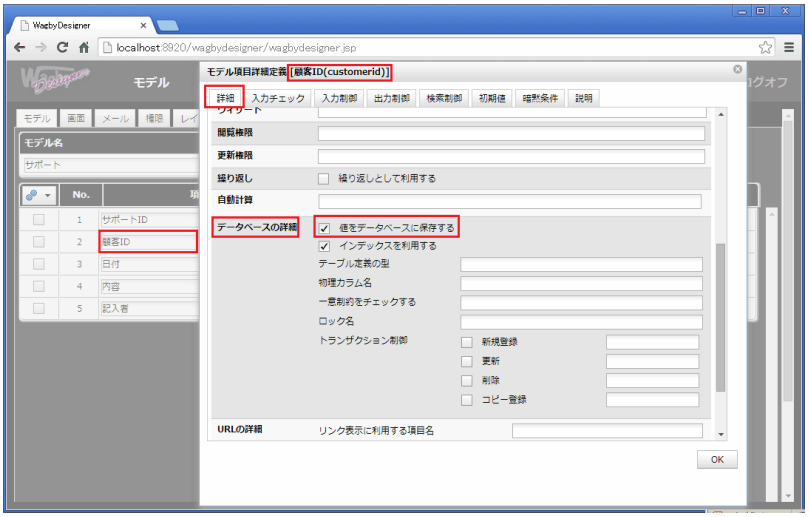


図 登録時に値をデータベースに保存する設定

設定上の注意

本設定は、下記の条件をすべて満たしている必要があります。

- 子モデルはタブによる表示を使わない。縦に並べて一覧表示する方式とすること。
- 子モデルに検索画面があること。
- 外部キーが検索項目として定義されていること。
- 外部キーの検索条件は、項目詳細定義の検索制御タブの「条件式」設定が「等しい」となっていること。
- 外部キーが文字列型の場合、標準では 「条件式」設定が「文字列の部分一致」となっているので、「等しい」に定義変更すること。
- 外部キーを除く検索項目に必須チェックが定義されていないこと。具体的には、項目詳細定義の検索制御タブの「必須チェック」が定義されていないこと。

子モデル一覧名を変更する

子モデルの一覧表示名を「顧客別サポート履歴一覧」に変更した例を示します。

図　一覧表示名の変更

定義方法

サポートモデルの「検索・一覧表示>外部キー連携>一覧名」を設定します。

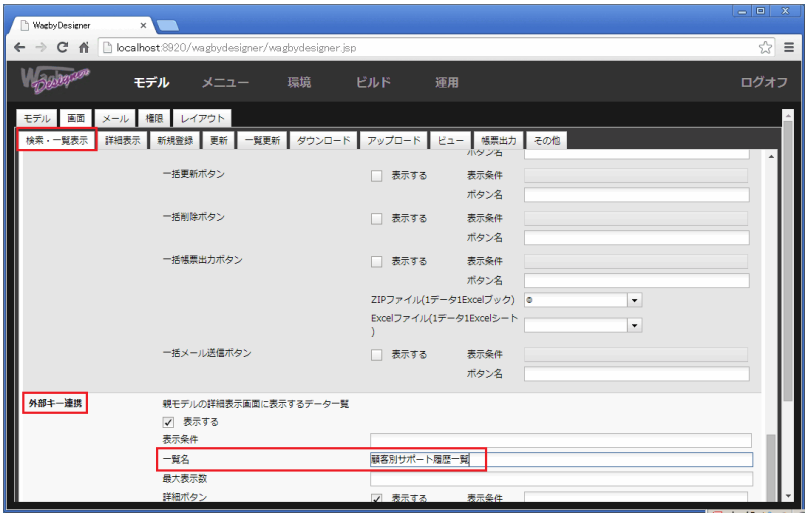


図　一覧名の設定

一覧データの表示件数を制限する

子モデルの一覧表示件数を最大1件と設定した例を示します。

図 子モデルの最大表示件数を1件とする

表示されなかったデータは、サポートモデルの検索画面で確認することができます。

図 サポートモデルの検索画面で他のデータを確認する

定義方法

サポートモデルの「検索・一覧表示>外部キー連携>最大表示件数」を設定します。この例では〝1〟としています。

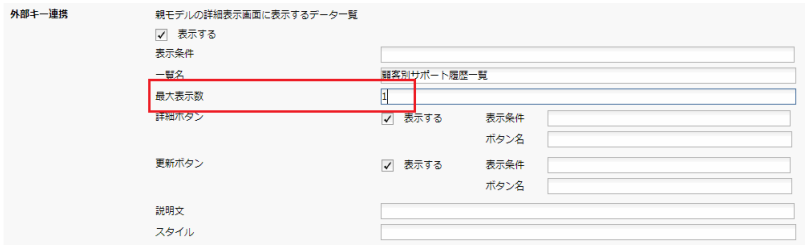


図 最大表示件数の設定

説明文を表示する

図に対して、説明文を加えた例を図に示します。

図 子モデル一覧表示に説明文を加える

定義方法

サポートモデルの「検索・一覧表示>外部キー連携>説明文」を設定します。同時に、スタイルを指定することもできます。

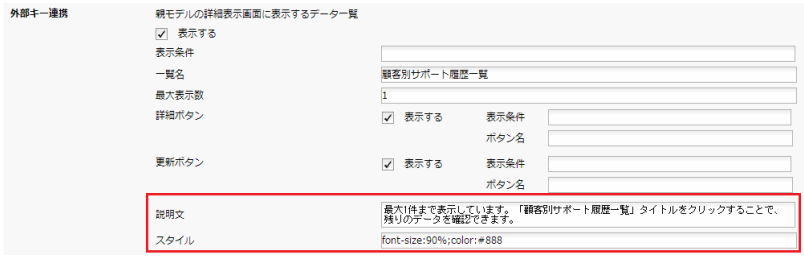


図 説明文とスタイルの指定

ボタンの表示を制御する

子モデル新規作成

外部キー連携における子モデルの作成は、親モデルの詳細表示画面に表示される「新規作成」ボタンを用いることができます。このボタンを表示しない方法を説明します。

図 新規作成ボタンを非表示とした例

サポートモデルの「新規登録>外部キー連携>親モデルの詳細表示画面に表示する新規登録ボタン」を無効にします。（標準では有効になっています。）

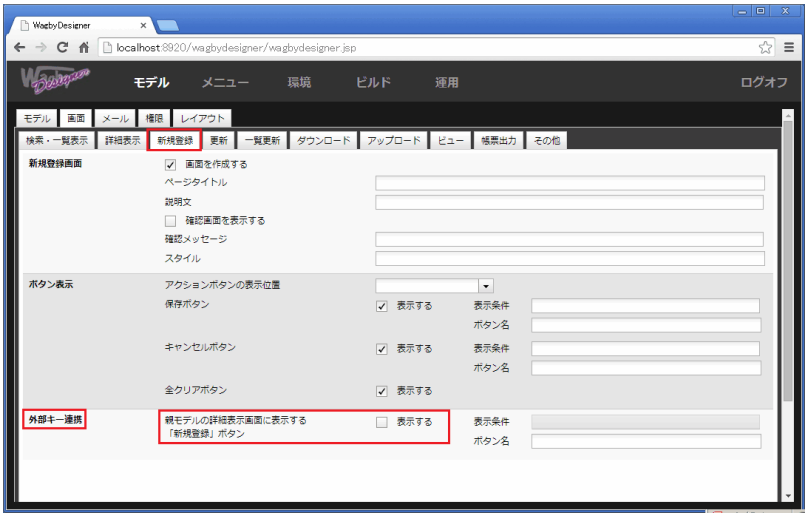


図 親モデルの詳細表示画面に表示する新規登録ボタンの設定

ボタン名や表示条件を設定することもできます。詳細は「画面機能>ボタン名と表示条件のカスタマイズ」で説明しています。

親モデルへの一覧表示・詳細表示

子モデルの詳細画面に用意される、親モデルへの一覧表示・詳細画面へ遷移するボタンを表示しないようにすることができます。

図 親モデルへの遷移用ボタンを非表示とした例

サポートモデルの「詳細表示>外部キー連携>親モデルの一覧表示ボタン」および「詳細表示>外部キー連携>親モデルの詳細表示ボタン」を無効にします。（標準では有効になっています。）

図 親モデルの一覧表示ボタン、詳細表示ボタンの設定

ボタン名や表示条件を設定することもできます。詳細は「画面機能>ボタン名と表示条件のカスタマイズ」で説明しています。

親モデルの画面に子モデルのデーター覧を表示しない

図の設定（「新規作成」ボタンの非表示）に加え、子モデル一覧表示も行わないように設定することもできます。図のように、親子モデル関係の存在が画面上からは認識できません。

図 新規作成ボタンを作成せず、子モデル一覧表示も非表示とした例

定義方法

サポートモデルの「検索・一覧表示>外部キー連携>親モデルの詳細表示画面に表示するデーター覧」の「表示する」設定を解除します。

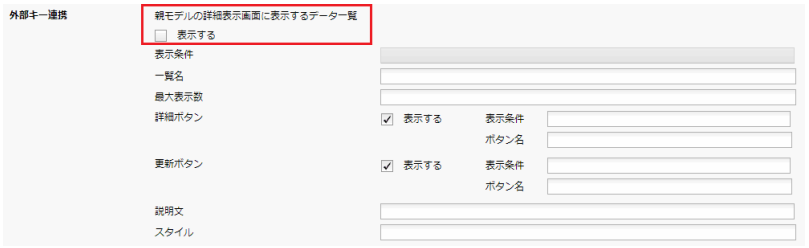


図 親モデルの詳細表示画面に表示するデーター覧の設定

ボタン名や表示条件を設定することもできます。詳細は「画面機能>ボタン名と表示条件のカスタマイズ」で説明しています。

削除レベルの指定

外部キー関係では、親モデルのデータを削除したとき、子のモデルも同時に削除されます。　図, 図で、親モデル「顧客」を削除する例を示します。

図 顧客データを削除する

図 顧客データの削除に成功

ここでサポートモデルの確認を行います。 子モデル（サポート）も削除されていることがわかります。

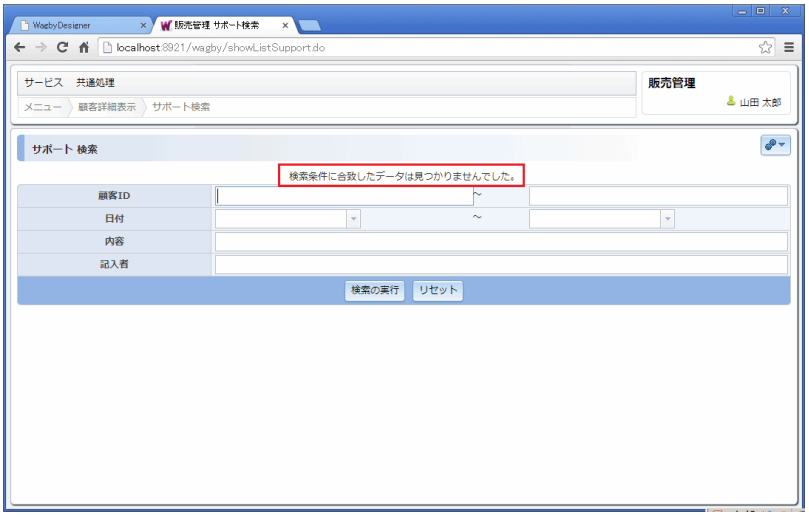


図 子モデル「サポート」も同時に削除されている

定義方法

「その他>モデルの関連性>親モデルの削除時に自動的に削除する」を確認します。標準では有効になっています。この設定を解除すると、親モデルのデータを削除しても子モデルは残ります。（ただし、このときは紐づく親のデータは存在しないという状態になります。）

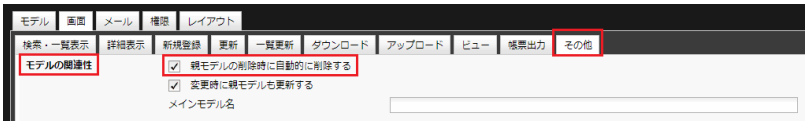


図 親モデルの削除時に自動的に削除する設定

親−子−孫という関係では、親を削除したときにそれに紐付くすべての子孫が削除されます。

再計算の指定（親）

子モデルの登録・更新・削除のタイミングで、親モデル側の項目値の再計算と再保存処理を行わせることができます。

子モデルとなるサポートの件数を、親モデルである顧客で管理した例を図に示します。

図 サポート件数項目を追加する

サポートを追加すると、「サポート件数」の値も増加します。この登録・更新・削除のタイミングで、親モデル側の項目値の再計算と再保存処理が行われます。

図 サポート件数項目が再計算される

定義方法

親となる顧客モデルに「サポート件数」項目を追加します。データベースに保存するとし、式はCOUNT関数を用います。

```
COUNT(${support_lp})
```

図 サポート件数項目の定義

サポートモデルの「画面>その他>変更時に親モデルも更新する」を有効にします。（標準で、この値は有効になっています。）

図 変更時に親モデルも更新する設定

この設定は「サポート件数」項目がデータベースに保存するとなっているため機能します。データベース非保存の場合は、表示時に常に再計算されるため、本設定に関わらず計算は実行されます。

この設定は、親がさらに数珠つなぎとなっている場合、先祖まで遡って再計算します。 なお、親に紐づく当該子データの「兄弟」データは再計算の対象となりません。

子モデル単体での登録・更新時は、本処理は行われません。例えば子モデルをメール受信により登録するような場合、親モデルの再計算は行われません。

再計算の指定（子）

親モデルの登録・更新・削除のタイミングで、子モデル側の項目値の再計算と再保存処理を行わせることができます。

次の条件が成立した場合に、子の再計算機能が有効になります。

- 子モデルの一覧更新画面が作成されている。
- 子モデルの一覧更新で外部キーの同時更新機能が有効になっている。

詳細は"子モデルの同時更新"をお読みください。

子モデルの同時更新

例と定義方法

「顧客」モデルと「サポート」モデルを例として扱います。 1人の顧客にはN件のサポート情報が紐づいています。このとき、顧客モデルの更新画面で（紐づいている）N　件のサポート情報を更新することができます。

図 顧客モデルの更新画面（子モデルも同時に更新できる）

定義方法

設定はすべて（子モデルとなる）サポートモデルで行います。親モデル側の設定はありません。

サポートモデルで、外部キーとなる「顧客ID」を検索条件に含めます。

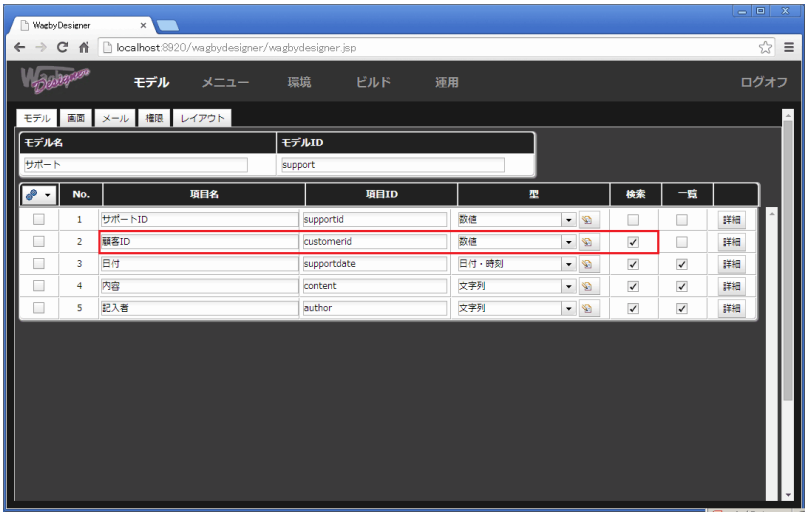


図 外部キーを検索条件に含める

「顧客ID」は、登録時に値をデータベースに保存する設定とします。

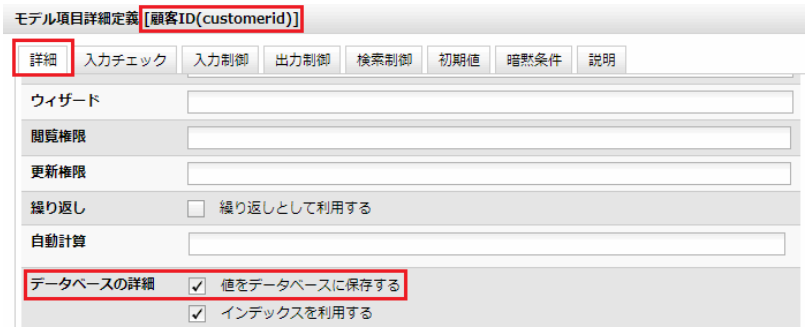


図 登録時に値をデータベースに保存する設定

同じく「顧客ID」項目の詳細定義にて、条件式を「等しい」とします。

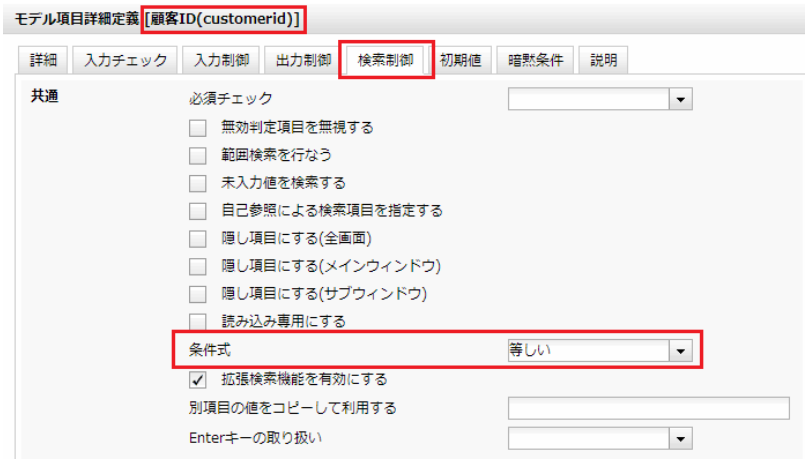


図 条件式を等しいとした場合

この設定の意図は、外部キーの参照先を特定できるようにするためです。よって例えば、外部キーが「文字型」の場合、「部分一致検索」を指定することはできません。

サポートモデルの「一覧更新画面」を作成するようにします。

図 一覧更新画面を作成する

一覧更新画面の設定欄に用意されている、外部キーの同時更新機能を有効にします。（標準では有効になっています。）

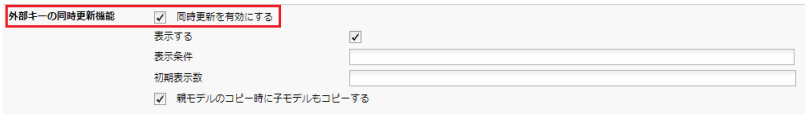


図 外部キーの同時更新機能を有効にする

制約

本機能は「一覧更新画面」を使って実現しています。そのため、本機能の仕様と制約は一覧更新画面に準じます。

初期表示数

子モデルの初期表示数を指定することができます。 親モデル（ここでは顧客モデル）の登録画面では、図のように指定した初期表示数分だけレコードが表示されるようになります。

図 顧客の登録画面。初期表示数が反映されている。

定義方法

サポートモデルの一覧更新画面設定で、「外部キーの同時更新機能>初期表示数」を設定します。

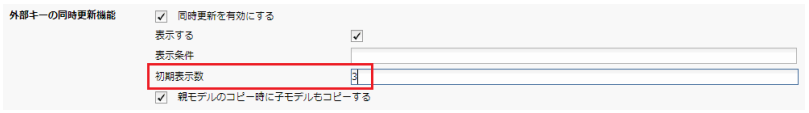


図 初期表示数の指定

子モデル一覧表示から直接、子モデルの更新画面へ遷移する

子モデルの任意のデータを選択し、更新画面へ直接、遷移することができます。

図 更新ボタンが用意された例

定義方法

サポートモデルの「検索・一覧表示>外部キー連携」で、詳細ボタンと更新ボタンをそれぞれ有効にします。（更新ボタンのみ有効にすることもできます。）

図 詳細ボタンと更新ボタンを有効にする

定義時の注意点

- 親モデルの詳細表示画面に含まれる子モデル一覧表示では、子モデルの「検索・一覧定義」のボタン設定は反映されません。ここに用意されたボタン設定を利用してください。
- 子モデル一覧表示で「詳細」ボタンが非表示の場合、行クリックも無効です。（マウスオーバーによって色は変わりますが、クリックしても詳細画面へ遷移しません。）
- 子モデルの詳細画面や更新画面を用意しない場合、これに対応してボタンも表示されません。
- 詳細表示および更新ボタンの「表示条件」に含めることができる項目は、子のモデルになります。これは、子の各レコード毎に条件を指定するためです。（一方で、「子モデル全体を表示するかどうか」という表示条件は、親モデルの値を使います。）
- ボタン名は固定表記のみとなっています。ここに式を指定することはできません。

1つの親に複数の子を紐付ける

外部キーによるモデルの連携は、複数指定することができます。このとき、子モデル同時更新も動作します。

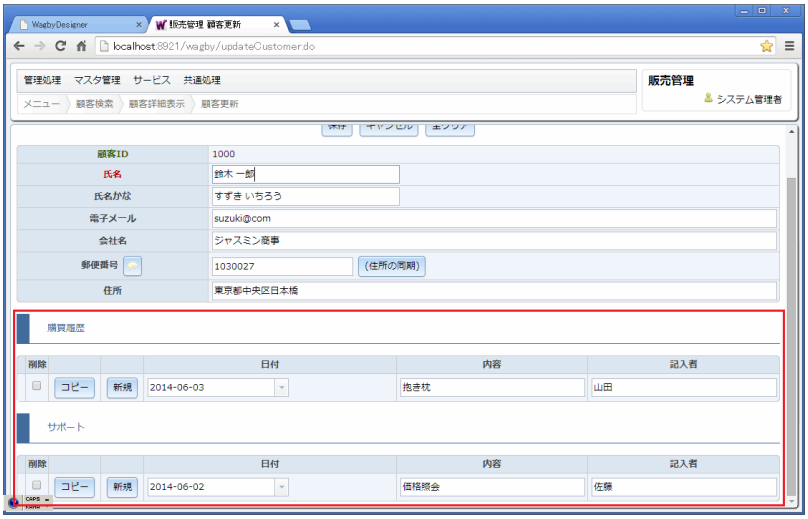


図 1つの親モデルに複数の子モデルを定義する

コピー登録

外部キー関係にあるモデルにおいて、コピー登録を行うと親モデルに紐づく子モデルもすべてコピーされます。

図では、顧客モデルにコピー登録機能を用意しました。 コピーボタンをクリックすると、コピー登録画面に遷移します。

図 顧客モデルのコピー登録ボタン

顧客モデルのコピー登録画面では、関連するサポートのデータがある場合、これも同時にコピーされていることが分かります。

図 顧客のコピー登録画面。関連するサポートのデータもコピーされている。

定義方法

顧客モデルでは、「検索・一覧表示>ボタン表示>コピー登録ボタン」を有効にします。

図 コピー登録ボタンを用意する

サポートモデルでは、「一覧更新>外部キーの同時更新機能>同時更新を有効にする」を設定するとともに、「親モデルのコピー時に子モデルもコピーする」を設定します。（標準ではこの設定は有効になっています。）

図 親モデルのコピー時に子モデルもコピーする設定

複数の外部キー子モデルをもつ場合

二つ以上の外部キー子モデルをもつ親モデルでも、本コピー機能は動作します。

スクリプトを使って子モデル毎に異なる初期値を設定する

"親子モデル同時更新機能で親モデル登録時、子モデル毎に異なる初期値を設定する"をお読みください。

仕様・制約

- ″一つの子に複数の親を紐づける″設定を行った子モデルは、同時更新機能を利用することができません。
- 子モデルの更新ではページネーション機能は動作しません。
- 子モデルの更新「コピー登録」では、「削除のチェックボックスを表示する」は設定に関わらず常に有効となります。(*1)
- 子モデルの更新を用いる場合、確認画面を設定することはできません。

(67)

入力チェック

必須・警告

必須入力

例

必須チェックを指定した項目の項目名は赤字で表示されます。 未入力のままカーソルを移動すると、図のようにリアルタイムでチェックされます。

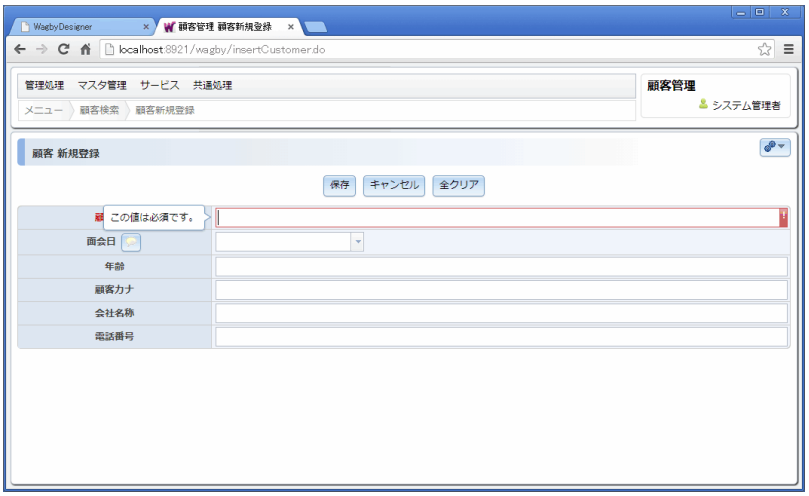


図 リアルタイム入力チェック

定義方法

モデル項目の詳細設定ダイアログにて「入力チェック-必須」を「○」とします。

(67) 1. コピー登録の場合の子モデルの一覧更新の削除処理は、実際には削除でなく、コピー登録の扱いとなります。このため、削除のチェックボックスはコピーの有無の選択となるため、常に有効となっています。同様に更新処理についても、コピー登録時に値を修正して保存する動作と同様なため、更新も可能としています。

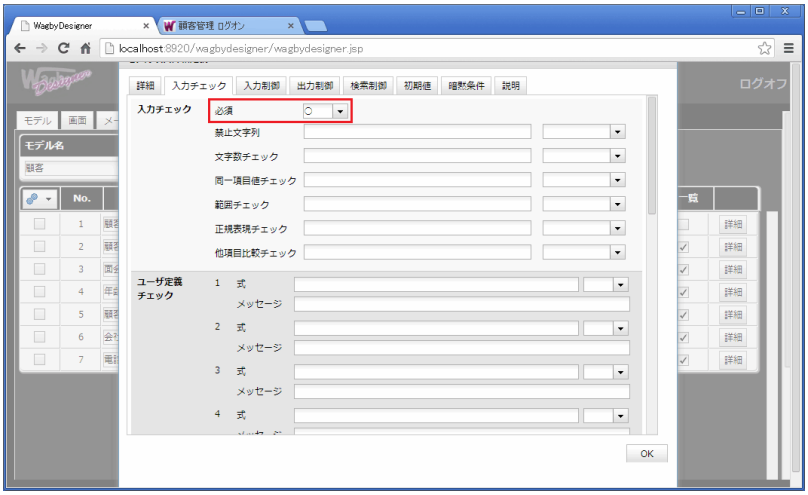


図 必須入力の指定

警告を出すが、入力を許可する

例

「警告」を設定した項目も項目名は赤字で表示されます。 入力しない場合は警告が表示されますが、もう一度「保存」ボタンをクリックすると登録することができます。

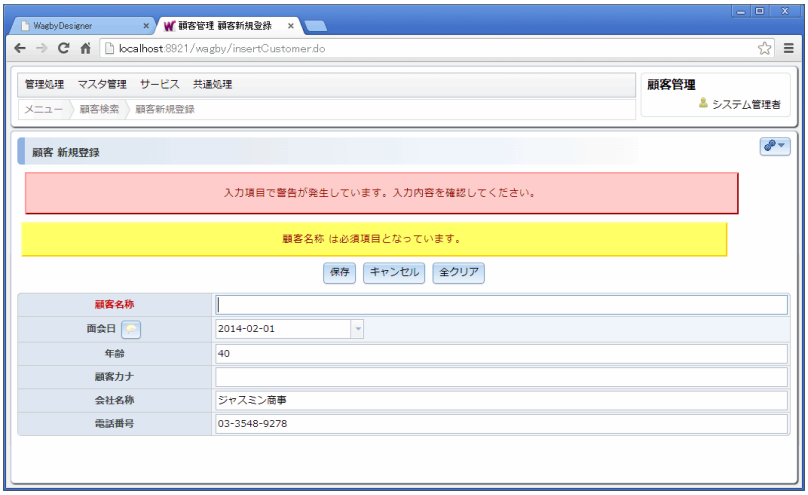


図 一回目の「保存」ボタン押下時は警告メッセージが表示される

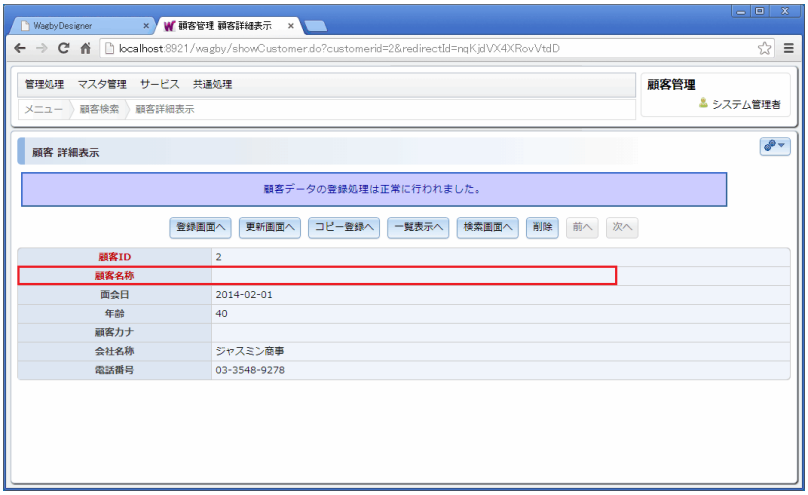


図 二回目の「保存」ボタン押下時は（空白のまま）保存される。

定義方法

モデル項目の詳細定義にて「入力チェック-必須」を「警告」とします。

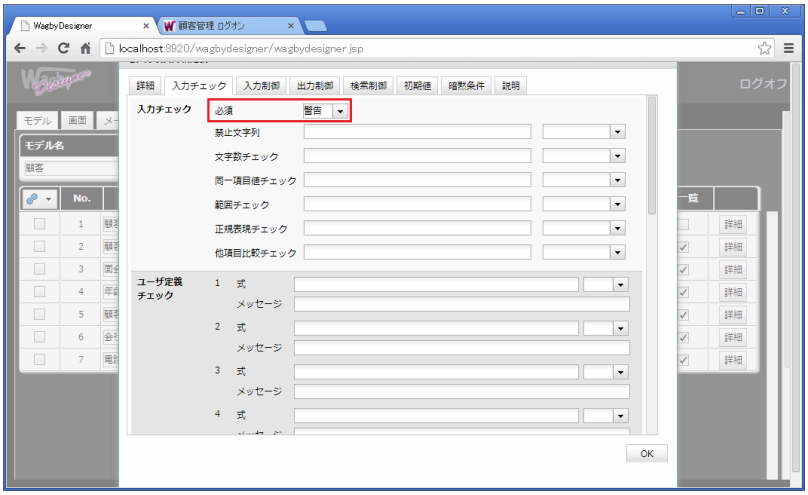


図 必須入力の指定（警告）

初期値について

項目を必須とした場合、自動的に次の初期値が設定されます。

項目の型	初期値
整数/1バイト整数/2バイト整数/4バイト整数/8バイト整数	0
4バイト浮動小数/8バイト浮動小数	0.0

データベーススキーマへの影響

この設定を行うと、テーブル定義の当該項目に NOT NULL 制約が付与されます。つまりテーブルスキーマが変更されます。

既存テーブルスキーマを再利用する場合の注意点

既存のテーブルスキーマを再利用する場合は注意が必要です。 具体的には必須項目に NULL 値が存在する場合、そのデータは Wagby からの検索でヒットしません。[詳細...]

サブモデルを定義する場合の注意点

サブモデルを定義し、かつ、サブモデル側の該当項目では必須チェックを行わないとします。 （すなわち、メインモデル側の項目には必須チェックが設定されているが、サブモデル側の項目では必須チェックが設定されていない、という状況を想定します。）

この場合、メインモデルは必須チェックを行い、サブモデルでは必須チェックを行いません。

この設定では、テーブル定義で NOT NULL 制約は設定されません。

主キーとの関係

「主キーとして利用する」設定がチェックされた場合、必須チェックが自動的に有効かつ変更不可となります。

仕様・制約

- このチェックはリアルタイムで行われます。（ブラウザのJavaScriptでチェックします。）
- 入力エラーの状態では「保存」などのボタンを押下することはできません。
- エラーとなった入力項目は、背景が赤となります。テキストボックスまたは日付時刻リストボックスが対象です。
- 閲覧権限が設定されている項目に対して「必須チェック」は指定できません。
- 入力対象ではない項目に対する入力チェックを行うことはできません。具体的には計算式で求められる項目や、読み込み専用項目、参照連動項目です。
- 一覧更新画面（親子モデルにおける、子モデル同時更新画面も含む）では、警告指定は動作しません。

関連するページ

- 他モデルの参照 > リストボックス > 必須を指定する

禁止文字列

例

禁止文字列として、携帯メールアドレスを指定した例です。

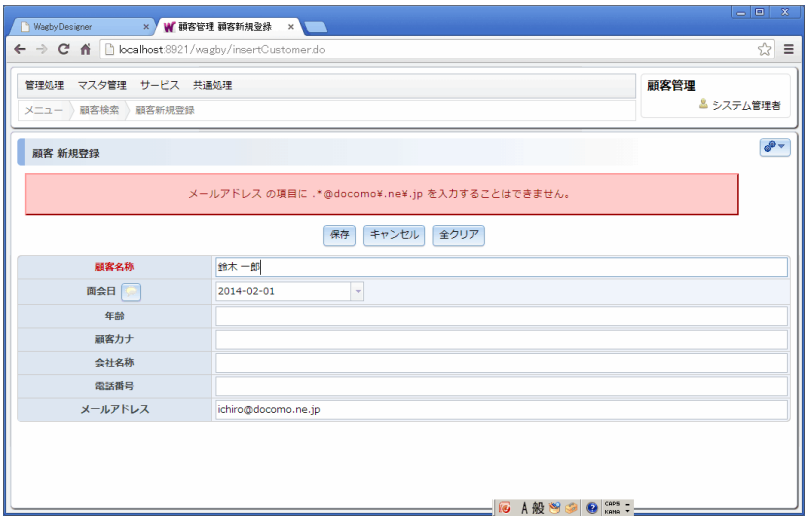


図 携帯メールアドレスが禁止された例

禁止文字列チェックはリアルタイムでは行われません。保存ボタン押下後にサーバ側でチェックされます。

定義方法

モデル項目の詳細定義「入力チェック」で、禁止文字列の欄に正規表現式を指定します。

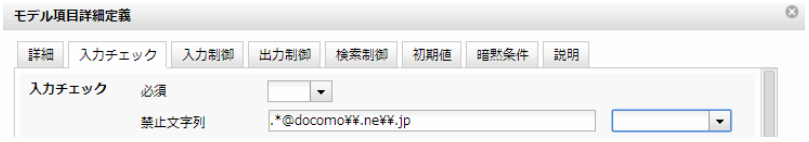


図 禁止文字列の指定

```
.*@docomo¥¥.ne¥¥.jp
```

ここで「¥¥」(半角文字)は、その次の文字「.」が正規表現ではなくピリオド文字として解釈させるためのエスケープ処理です。 Java正規表現について説明しているサイトを紹介します。

- Java正規表現の使い方 - JavaDrive（日本語）

禁止文字列の解釈

禁止文字列チェックは登録・更新を禁止する文字列全体を設定するものです。 例えば同設定欄に ″&″ を設定した場合、当該項目に ″&″ のみの登録は行えなくなりますが、″A&B&C″ のような文字は登録できます。″&″ を含む文字の登録を禁止したい場合は、正規表現で次のように設定してください。

```
.*&.*
```

ある文字を含むことを禁止する場合の設定

次の文字を含む入力を禁止する例を説明します。

- &
- $
- ″

この3文字を含んだ文字列の登録を禁止する場合、正規表現で次のように設定してください。

```
.*[&|$|″].*
```

複数の禁止文字列を定義する

複数の禁止文字列を指定するときは、各正規表現式をコンマ区切りで列挙します。

```
.*@docomo¥¥.ne¥¥.jp,.*@ezweb¥¥.ne¥¥.jp,.*@softbank¥¥.ne¥¥.jp
```

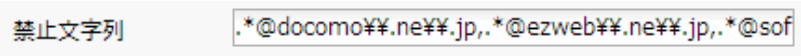


図 複数の禁止文字列を指定する

警告の設定

警告を表示しますが、ここで保存ボタンを押下すると登録自体は妨げないという設定を行う場合は、入力欄右側のリストボックスから「警告」を指定します。（空欄時は、エラーのときは登録できないようになります。）

図 警告の指定

トラブルシューティング

正規表現式が動作しない

ビルドしたWagbyアプリケーションの実行時ログ(logs/system.log)にエラーが出力されていないかどうかを確認ください。ここに ″illegal syntax pattern″ などの文字列が含まれている場合、式に誤りがあります。

仕様・制約

- このチェックは「保存」などのボタンを押下したタイミングで、サーバ側で行われます。
- エラーとなった入力項目は、背景が赤となります。テキストボックスまたは日付時刻リストボックスが対象です。
- 入力対象ではない項目に対する入力チェックを行うことはできません。具体的には計算式で求められる項目や、読み込み専用項目、参照連動項目です。
- 一覧更新画面（親子モデルにおける、子モデル同時更新画面も含む）では、警告指定は動作しません。

文字数チェック

例

指定したサイズを超えて入力したタイミングで、リアルタイム入力チェックが作動します。

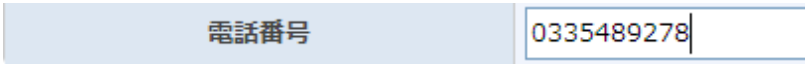


図 入力サイズを超えない

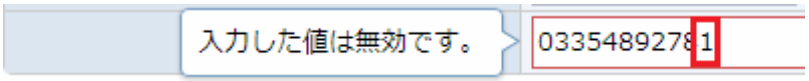


図 入力サイズを超えた

定義方法 - 数値型項目の場合

範囲チェックを指定します。例えば4桁の場合「0-9999」と設定してください。

定義方法 - それ以外

モデル項目の詳細定義にて「入力チェック-文字数チェック」に数値を設定します。

図 文字数チェックの指定

チェックの単位を設定できます。「char」は文字数で、「byte」は文字を構成するバイト数です。

「char（警告）」および「byte（警告）」を設定すると、入力値が設定した文字数を超えた場合、登録時に警告を表示します。再度「保存」ボタンをクリックすると、登録を行うことができます。

この欄には数字以外の特別表記を指定することができます。

指定方法	記述例	説明
数字のみ	5	最大5文字(6文字ならエラー)

-数字	-5	上記「数字のみ」の記述と同じ
数字-	5-	最小5文字(4文字ならエラー)
数字1-数字2	5-8	最小5文字、最大8文字

文字列項目とデータベースの関係

文字列型の項目に「文字数チェック」を指定しない場合、テーブル定義上で指定された最大文字数を超えた入力が行われてしまうとデータベースエラーが発生します。

例えば Oracle の場合、標準では文字列型項目に「varchar2(255 char)」が適用されるため、最大文字数は 255 文字です。従って、文字数チェック設定に 255 より小さい数値を指定することで、過入力によるデータベースエラーを回避できるようになります。

仕様・制約

- このチェックはリアルタイムで行われます。(ブラウザのJavaScriptでチェックします。)
- 入力エラーの状態では「保存」などのボタンを押下することはできません。
- エラーとなった入力項目は、背景が赤となります。テキストボックスまたは日付時刻リストボックスが対象です。
- 数値フォーマットを指定している場合、これも考慮して文字数を設定してください。(例 : 7桁の数値入力項目で、3桁区切りで「,」を入力させる場合、これを考慮して文字数の上限を 9 とする必要があります。)
- 入力対象ではない項目に対する入力チェックを行うことはできません。具体的には計算式で求められる項目や、読み込み専用項目、参照連動項目です。

正規表現との併用

文字数チェックは内部で正規表現式に展開されます。例えば文字数に4と設定した場合、正規表現 ".{0,4}" が設定されます。 すなわち正規表現の設定は無効になります。

文字数チェックと正規表現を併用する場合は、文字数チェックを指定せず、正規表現式で、文字数チェックも兼ねる式を個別に設定してください。

(68)

モデル参照項目での利用

Wagbyの文字数チェックは、データベースのテーブルのカラム（列）のサイズを超えないように、事前に入力チェックを行うという機能です。

そのため、モデル参照項目（検索画面など）に文字数チェックを適用した場合は、画面に表示される「内容部」の文字数ではなく、テーブルのカラムに格納される「ID部」の文字数のチェックになります。

バイト指定の動作 (8.1.1)

入力文字数チェック(byte)のとき、"環境>データベースの文字エンコーディング情報"を利用します。この設定が空白のときは "UTF-8" 扱いとなります。[この仕様を旧方式に戻す...]

ブラウザでのリアルタイムチェック

データベースの文字エンコーディング情報が "UTF-8" の場合のみ、バイト数でチェックするようになります。それ以外は文字数でチェックします。

(69)

(68) そのため数値型に文字数チェックを設定すると数字以外の文字も入力できるようになります。数値型の場合は文字数ではなく範囲チェックを指定することで対応してください。

(69) UTF-8 では原則は1文字3バイトですが、JIS X 0213の第3・4水準漢字の一部と絵文字では1文字4バイトとなるものがあります。

同一項目値チェック

例

二つの項目が同一値でない場合にエラーとなった例を図に示します。

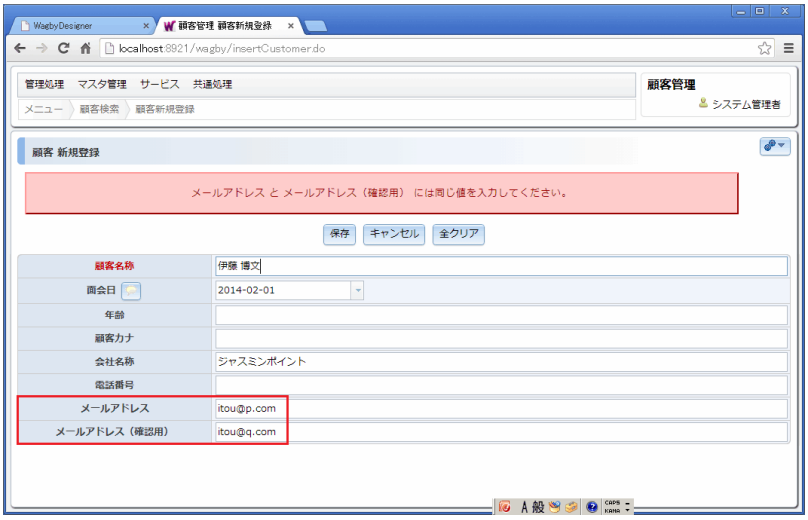


図 メールアドレスに確認用入力欄を用意する

同一項目値チェックはリアルタイムでは行われません。保存ボタン押下後にサーバ側でチェックされます。

定義方法

モデル定義で、二つの項目「メールアドレス」と「メールアドレス（確認用）」を用意します。

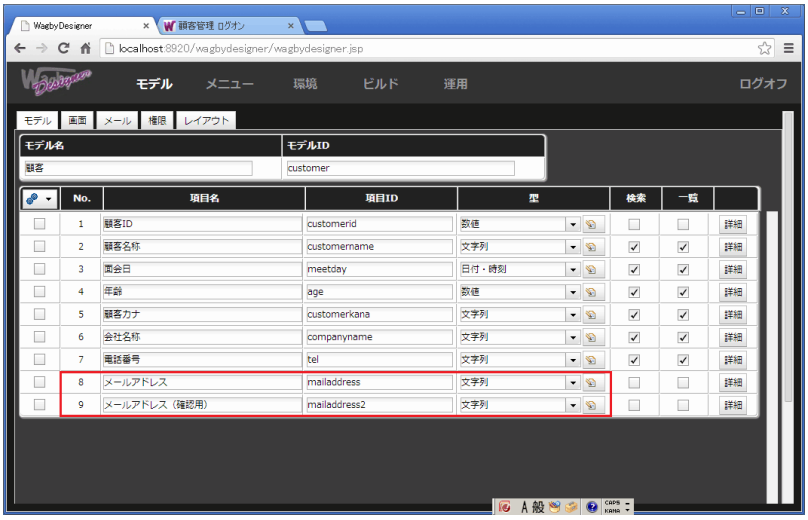


図 同一値となる項目を用意する

モデル項目の詳細定義「入力チェック」で、同一項目値の欄に項目IDを記載します。

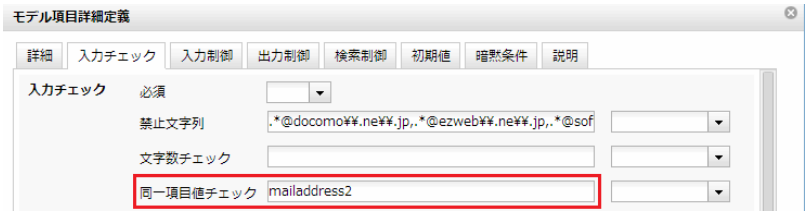


図 項目IDを記載する

警告の設定

警告を表示しますが、ここで保存ボタンを押下すると登録自体は妨げないという設定を行う場合は、入力欄右側のリストボックスから「警告」を指定します。（空欄時は、エラーのときは登録できないようになります。）

図 警告の指定

仕様・制約

- 本機能は、項目の型が「文字列」「数値」「日付・時刻」の場合に利用できます。
- ウィザードを利用する場合、同じ画面に含まれれている必要があります。
- このチェックは「保存」などのボタンを押下したタイミングで、サーバ側で行われます。
- エラーとなった入力項目は、背景が赤となります。テキストボックスまたは日付時刻リストボックスが対象です。
- 入力対象ではない項目に対する入力チェックを行うことはできません。具体的には計算式で求められる項目や、読み込み専用項目、参照連動項目です。
- 一覧更新画面（親子モデルにおける、子モデル同時更新画面も含む）では、警告指定は動作しません。

範囲チェック

数値の範囲

「年齢」項目は0から130までという範囲を設定した例を図に示します。範囲外の値はエラーになります。

図 年齢の範囲チェック

数値の範囲チェックはリアルタイムで行われます。

数値の範囲を定義する

入力チェック-範囲チェックに値を記入します。

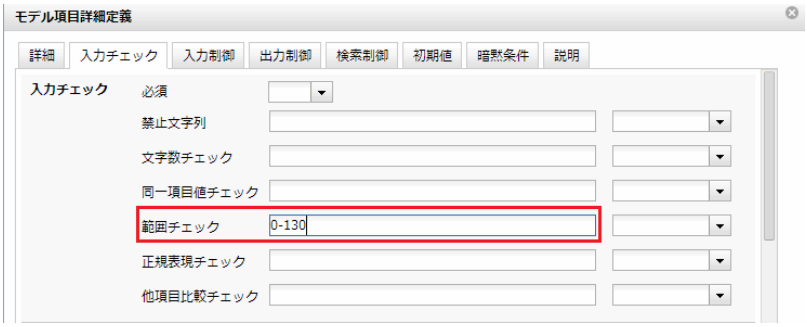


図 数値の範囲指定

正の数

二つの数値を「-」（ハイフン）で連結させて範囲を指定します。 例えば、数値が1から10までの入力のみに制限したい場合は、この項目に「1-10」と指定しておきます。

負の数

負の数を含めたい場合は、括弧″（）″で囲みます。 例えば、-10から10までの入力としたい場合は、「(-10)-10」と指定します。

文字数チェックとして使う

例えば4桁の場合、「0-9999」と指定します。

日付型の範囲

「yyyy年MM月dd日までの日付を有効」という指定を行うことができます。

図は、範囲に「2000/1/1-」と記述した例です。それ以前の値を入力するとエラーになります。

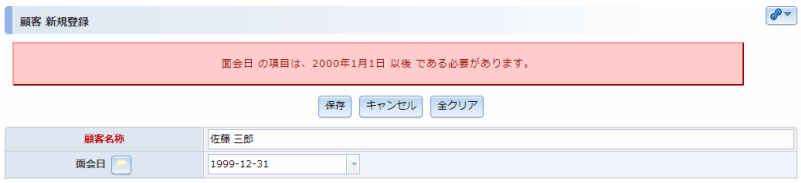


図 日付の範囲設定を超えた入力を行った

日付の範囲チェックはリアルタイムでは行われません。保存ボタン押下後にサーバ側でチェックされます。

日付の範囲を定義する

固定の日付は yyyy/MM/dd 形式で指定します。 ″now″ を含む式も記述できます。now は「当日」を意味する特別な表記方法です。

記述例	説明
(now-7)-(now-1)	7日前から1日前まで
(now-7)-now	7日前から今日まで
(now-7)-2013/1/1	7日前から2013年1月1日まで
now-2013/12/31	今日から2013年12月31日まで
2013/1/1-now	2013年1月1日から今日まで
2013/1/1-(now+7)	2013年1月1日から、今日から7日後まで
-now	今日まで

now-	今日以降
now	今日のみ
now, now+2	今日と、あさって
now, now+2, now+4	今日と、あさってと、4日後
2013/1/1, now, 2013/12/31	2013年1月1日と、今日と、2013年12月31日

なお、コンマとハイフンを混ぜた式は記述できません。次の表記は誤りです。

```
(now-7)-(now-5),(now-3),(now+2)-(now+5)
```

警告の設定

警告を表示しますが、ここで保存ボタンを押下すると登録自体は妨げないという設定を行う場合は、入力欄右側のリストボックスから「警告」を指定します。（空欄時は、エラーのときは登録できないようになります。）

図は数値型の例ですが、日付型も同様の設定です。

図 警告の指定

仕様・制約

- このチェックはリアルタイムで行われます。（ブラウザのJavaScriptでチェックします。）
- 入力エラーの状態では「保存」などのボタンを押下することはできません。
- エラーとなった入力項目は、背景が赤となります。テキストボックスまたは日付時刻リストボックスが対象です。
- 入力対象ではない項目に対する入力チェックを行うことはできません。具体的には計算式で求められる項目や、読み込み専用項目、参照連動項目です。
- 一覧更新画面（親子モデルにおける、子モデル同時更新画面も含む）では、警告指定は動作しません。

正規表現チェック

例

電話番号の項目に「数字の4桁-4桁-4桁」というルール（正規表現式）を指定した例を図に示します。

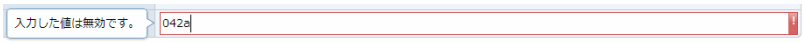


図 電話番号の正規表現ルールに合致しなかった

入力文字に文字が含まれているため、エラーになっていることがわかります。

定義方法

電話番号項目に次の正規表現式を設定した例を図に示します。

```
¥d{1,4}?-?¥d{1,4}?-?¥d{1,4}
```

「d{1,4}」は、1つから4つまでの数字が入ることを意味します。「-?」は、「-」があってもなくてもよいということを意味します。

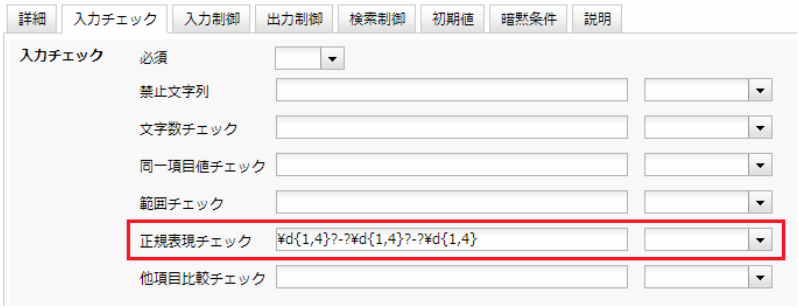


図 正規表現式の指定

警告の設定

警告を表示しますが、ここで保存ボタンを押下すると登録自体は妨げないという設定を行う場合は、入力欄右側のリストボックスから「警告」を指定します。（空欄時は、エラーのときは登録できないようになります。）

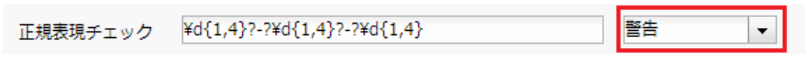


図 警告の指定

仕様・制約

- 利用できる正規表現の構文は、JavaおよびJavaScriptの両方で使える書式に限定されます。Designerで設定した式は、サーバ側のJavaと、クライアント(Webブラウザ)側のJavaScriptの両方に反映されるためです。
- Javaで使える正規表現の詳細はjava.util.regex.Patternクラスのjavadocをお読みください。
- JavaScriptで使える正規表現の詳細はMozillaのサイトが参考になります。
- このチェックはリアルタイムで行われます。（ブラウザのJavaScriptでチェックします。）(*1)
- 入力エラーの状態では「保存」などのボタンを押下することはできません。
- エラーとなった入力項目は、背景が赤となります。テキストボックスまたは日付時刻リストボックスが対象です。
- 正規表現式がブラウザのJavaScriptで解釈できない（動作しない）場合、保存ボタン押下後にサーバ側（Java）で処理されます。つまりフォームの値が送信されてからもチェック処理が適用されるという二重チェックとなっています。
- 「文字数チェック」と併用する場合は、正規表現式を工夫してください。[詳細...]
- 入力対象ではない項目に対する入力チェックを行うことはできません。具体的には計算式で求められる項目や、読み込み専用項目、参照連動項目です。
- 一覧更新画面（親子モデルにおける、子モデル同時更新画面も含む）では、警告指定は動作しません。
- カレンダを利用した日付型項目は、この項目に設定した日付フォーマットの正規表現をカレンダの入力チェックに利用します。詳細は〝モデル項目 > 日付 > カレンダの利用〟をお読みください。

(70)

利用できる構文

JavaとJavaScriptの両方で使える構文は次の通りです。

構文	説明
x	文字x
¥	次の文字をエスケープする
¥¥	バックスラッシュ文字

(70) 1. テキストエリアの場合はリアルタイムでのチェックは行われません。サーバ側でチェックされます。

¥xhh	16進値0xhhを持つ文字
¥uhhhh	16進値0xhhhhを持つ文字
¥t	タブ文字 (¥u0009)
¥n	改行文字 (¥u000A)
¥r	キャリッジ・リターン文字 (¥u000D)
¥f	用紙送り文字 (¥u000C)
¥cx	xに対する制御文字
[abc]	a、b、またはc (単純クラス)
[^abc]	a、b、c以外の文字(否定)
[a-zA-Z]	a-zまたはA-Z (範囲)
.	任意の文字(行末記号とマッチする場合もある)
¥d	数字:[0-9]
¥D	数字以外:[^0-9]
¥s	空白文字:[¥t¥n¥x0B¥f¥r]
¥S	非空白文字:[^¥s]
¥v	垂直方向の空白文字:[¥n¥x0B¥f¥r ¥x85¥u2028¥u2029]
¥w	単語構成文字:[a-zA-Z_0-9]
¥W	非単語文字:[^¥w]
^	行の先頭
$	行の末尾
¥b	単語境界
¥B	非単語境界
X?	X、1または0回
X*	X、0回以上
X+	X、1回以上
X{n}	X、n回
X{n,}	X、n回以上
X{n,m}	X、n回以上、m回以下
X??	X、1または0回
X*?	X、0回以上
X+?	X、1回以上
X{n}?	X、n回
X{n,}?	X、n回以上
X{n,m}?	X、n回以上、m回以下
XY	Xの直後にY
X|Y	XまたはY
(X)	X、前方参照を行う正規表現グループ
(?:X)	X、前方参照を行わない正規表現グループ
(?=X)	X、幅ゼロの肯定先読み
(?!X)	X、幅ゼロの否定先読み

例

次の例は、JavaとJavaScriptの両方で使えるものとなっています。

テーマ	書式
電話番号	¥d{1,4}?-¥d{1,4}?-¥d{1,4}
携帯電話番号	090-¥d{4}-¥d{4}
郵便番号 (*2)	^¥d{1,3}-¥d{1,4}$
生年月日	¥d{4}-¥d{2}-¥d{2}
アルファベットを含まない文字列	[^a-zA-Z]+
半角スペースを含まない文字列	[^]+
全角および半角スペースを含まない文字列	[^　]+
数字を含まない文字列	[^0-9]+
特定の文字のみを認める(全角50音ひらがな、全角50音カタカナ、全角数字、半角数字、半角ハイフン)	[ぁ-んァ-ンｏ－９0-9¥-]+

空白以外のすべての文字を認める	¥S+
数字以外のすべての文字を認める	¥D+[^0-9]+
アルファベットとアンダースコア以外のすべての文字を認める	¥W+[^a-zA-Z0-9_]+
半角英小文字のみを認める	[a-z]+
半角英小文字以外のみを認める（英大文字、数字、全角文字など含む）	[^a-z]+
半角英大文字のみを認める	[A-Z]+
半角英大文字以外のみを認める（英小文字、数字、全角文字など含む）	[^A-Z]+

(71)

全角文字のみ入力可能とする（日本）

Wagby の標準的な設定では「全角文字のみ入力可能」というルールは実現できません。 許容文字種の設定では全角漢字のチェック、または空白文字（スペース）が半角か全角かのチェックという指定になります。

そこで次のように設定します。

1. 入力フィルタを「文字列(最小限)」とします。
2. 次の正規表現式を指定します。

```
[^¥x20-¥x7e¥uff61-¥uff9f]*
```

他項目比較チェック

例

図は「紹介日（introducedate）」と「面会日（meetday）」という二つの日付型項目について、 「面会日」は「紹介日」と同一かそれ以降でなければいけない、という制約を指定している例です。 「面会日」が「紹介日」より以前の日付を入力した場合、エラーメッセージが表示されます。

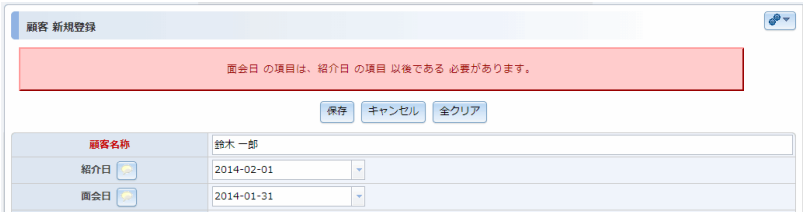


図 項目の値を比較

他項目比較チェックはリアルタイムでは行われません。保存ボタン押下後にサーバ側でチェックされます。

定義方法

モデル定義で、比較対象となる二つの項目「紹介日」「面会日」を用意します。

図 紹介日と面会日の定義

モデル項目の詳細定義ダイアログで、他項目比較チェックの欄に（比較対象となる）他項目のIDと、比較条件を指定します。

(71) 2. 郵便番号型の項目には、自動的に適用されます。ただし開発者が明示的に正規表現式を指定した場合は、その式を適用します。

図 他項目のIDと条件を指定する

比較する項目は、型が同じである必要があります。（数値型と数値型、または日付型と日付型）

指定できる比較条件は次のとおりです。

- 以上
- 以下
- より大きい
- より小さい
- 等しい
- 等しくない

警告の設定

警告を表示しますが、ここで保存ボタンを押下すると登録自体は妨げないという設定を行う場合は、入力欄右側のリストボックスから「（警告）」と付与されている選択肢を指定します。

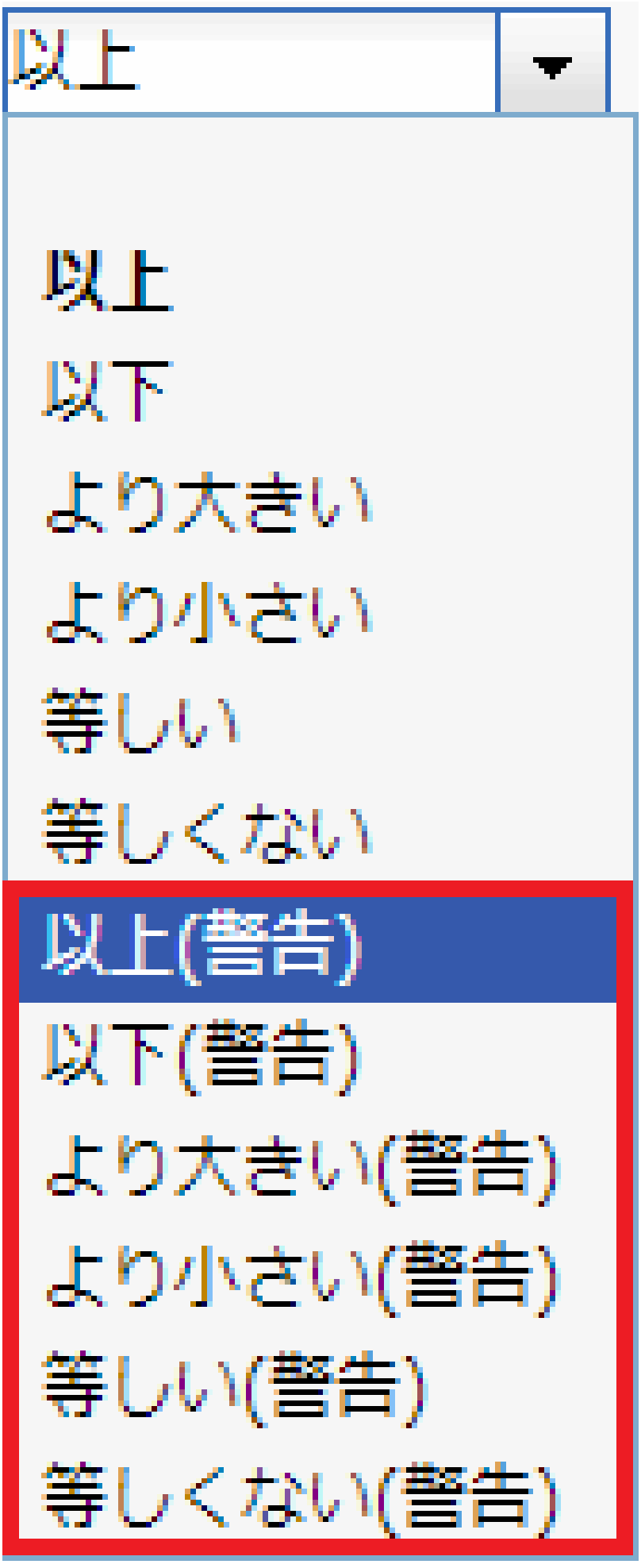

以上
以上
以下
より大きい
より小さい
等しい
等しくない
以上(警告)
以下(警告)
より大きい(警告)
より小さい(警告)
等しい(警告)
等しくない(警告)

図 警告の指定

仕様・制約

- 項目の型がモデル参照（チェックボックス）のときは、条件式は「等しい」「等しい（警告）」のいずれかのみが指定できます。
- このチェックは「保存」などのボタンを押下したタイミングで、サーバ側で行われます。
- エラーとなった入力項目は、背景が赤となります。テキストボックスまたは日付時刻リストボックスが対象です。
- 入力対象ではない項目に対する入力チェックを行うことはできません。具体的には計算式で求められる項目や、読み込み専用項目、参照連動項目です。

ユーザ定義チェック

例

図は「会社名」項目に空白が含まれていた場合にエラーとする例です。エラーメッセージも任意に用意したものです。

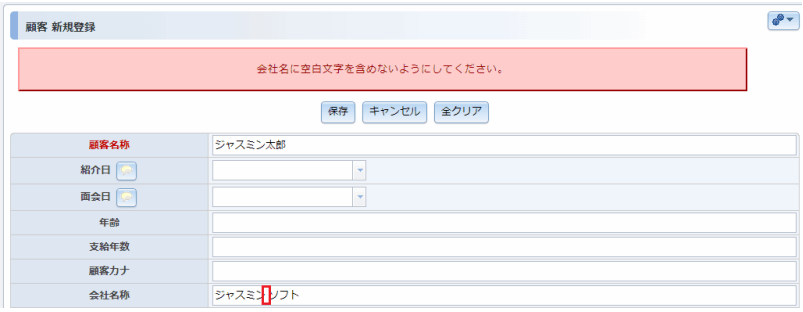


図 空白文字のチェック

ユーザ定義チェックはリアルタイムでは行われません。保存ボタン押下後にサーバ側でチェックされます。

定義方法

モデル項目の詳細定義ダイアログで、ユーザ定義に式とエラーメッセージを記述します。　式の値が「真(true)」となれば正常とします。「偽(false)」のときエラーメッセージを表示します。

項目名はプレースホルダ ${...} で括ります。式中には関数も使用できます。

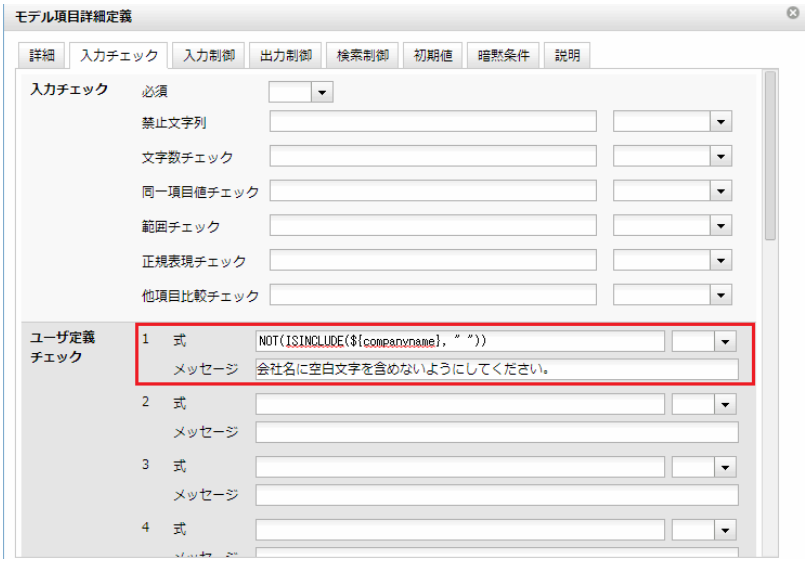


図 ユーザ定義式

式は「正常系（許可される）」内容を記述します。式の値が真(true)なら、エラーメッセージは表示されません。

警告の設定

警告を表示しますが、ここで保存ボタンを押下すると登録自体は妨げないという設定を行う場合は、入力欄右側のリストボックスから「（警告）」と付与されている選択肢を指定します。

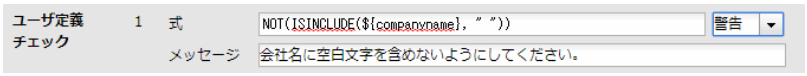


図 警告の指定

例

項目 item1 の未入力をエラーとしたい。（Wagbyの必須指定は使わない）

```
NOT(ISEMPTY(${item1}))
```

項目 item1 が数字型で、0 以上であることを判定したい。

```
${item1} >= 0
```

項目 item1 に "0" から "9" までの数字が含まれていること。そうでなければエラーとなる。

```
ISINCLUDE(${item1},"0123456789")
```

項目 item3 に半角スペースが含まれていないこと。もし含まれていればエラーとなる。

```
NOT(ISINCLUDE(${item3}, " "))
```

チェックボックス項目 item4 は 3 つまで選択できる。4 つ以上選択された場合、エラーとなる。

```
CHECKED(${item4})<=3
```

日付時刻型項目 sdate の値が、本日より30日以前の場合、入力が行えないとする。

```
OR(
 DATE(TODATE(${sdate})) >= DATE(TODAY()),
```

```
 DATE(TODATE(${sdate})) + 30 < DATE(TODAY())
)
```

sdate は日付時刻型のため、関数 TODATE を介して日付型に変換している。

今日が5月18日のとき、今日及び未来の日付、そして一ヶ月より前（4月17日より前）は入力できるが、4月18日から5月17日まではエラーになる。

複数の項目による条件式

IF 関数を組み合わせることで、複数の項目の条件を一つの式で記述することができます。

次の例は item1（文字列型）が "株式" の場合、item2（数値型）は 100 を上限とする、という式です。trueとfalseを返すような記述になります。

```
IF(
  EXACT(${item1},"株式"),
  IF(
    ${item2} < 100, true, false
  ),
  true
)
```

入力チェックスクリプトの利用

スクリプトを用いた、より複雑な入力チェックを行うこともできます。

国際化対応

エラーメッセージの国際化対応を行うことができます。詳細...

仕様・制約

- このチェックは「保存」などのボタンを押下したタイミングで、サーバ側で行われます。
- エラーとなった入力項目は、背景が赤となります。テキスト入力項目または日付時刻リストボックスが対象です。
- 繰り返し項目、繰り返しコンテナ、外部キー関連モデルに対する COUNT, SUM 等の集合処理関数を用いることはできません。
- 他の入力チェックは画面から入力された文字列に対して行われますが、この式を用いたチェックは唯一「ストアモデル」の項目を対象としています。（正確には、画面から入力された文字列から構成されたプレゼンテーションモデルからストアモデルへと変換したあと、そのストアモデルの値に対して式が適用されます。）ストアモデルとプレゼンテーションモデルの詳細は"モデルの基本"をお読みください。
- エラーメッセージに改行を含めることはできません。（スクリプトを使ってエラーメッセージを設定する場合、この制約を解除することができます。）
- 入力対象ではない項目に対する入力チェックを行うことはできません。具体的には計算式で求められる項目や、読み込み専用項目、参照連動項目です。

読み込み専用項目をチェックする方法

例として、項目 a, 項目 b, そして項目 sum を用意します。 ここで sum = a + b の関係があり、かつ sum の値は 100 でなければならない、という入力チェックを想定します。

この場合、sum 項目側に入力チェックを記述することはできません。 Wagby では読み込み専用項目(ユーザが直接編集できない項目; 自動計算項目もこれに含まれる)に対しては入力チェックを行わないようになっているためです。 [72]

[72] ユーザが直接編集できない項目に入力チェックをかけてしまった場合、常に入力エラーとなるようなデータしか作成できない場合が起きないようにしています。

そこで、項目 a の側に式を記述します。[73]

```
${sum}==100
```

許容文字種

例

「顧客カナ」項目に入力できる文字種（許容文字種）として、全角カナとスペースを指定しました。

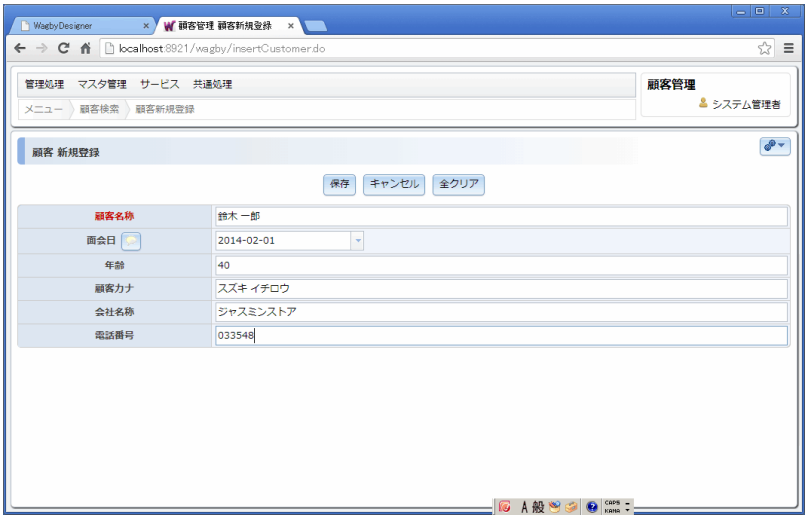


図 全角カナとスペースは入力できる

許容文字種以外（例：ひらがな）を入力するとエラーになります。

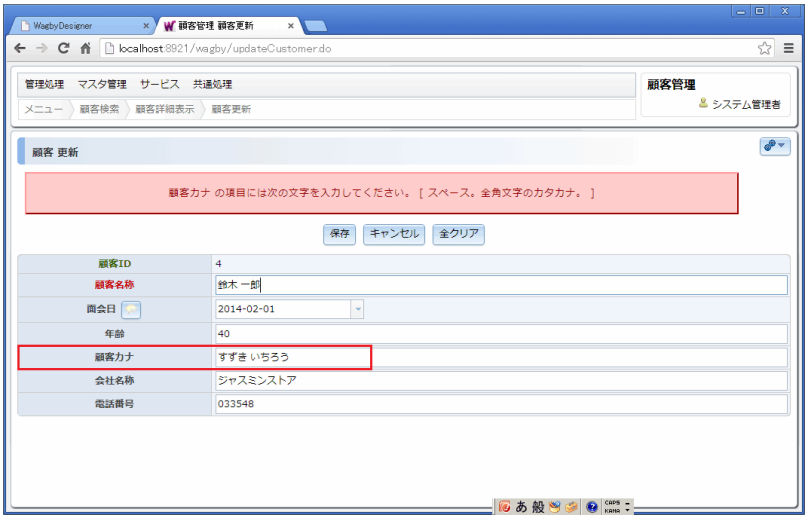


図 入力エラー

許容文字種のチェックはリアルタイムでは行われません。保存ボタン押下後にサーバ側でチェックされます。

[73] 項目 b に記述することもできます。同じ動作になります。

定義方法

モデル項目の詳細定義「入力チェック」で、許容する文字種に「○」を設定します。 一つでも「○」を設定した場合、それ以外の文字種は入力エラーになります。

図 許容文字種の設定

自動設定される型

数値型項目では、自動的に「半角全角文字（負数は許容）」が「○」になります。 日付型項目では「日付型(yyyy-MM-d, yyyy-MM形式または和暦形式）」が「○」になります。

設定の詳細

メールアドレス(「@」はなくてもよい)

入力値にメールアドレス形式の文字列が含まれているかをチェックします。「@」はなくてもよいと判断します。

日付型

次のいずれかの書式で表現された日付かどうかをチェックします。ここで yyyyは年(4桁)、MM は月(2桁)、dd は日(2桁)をそれぞれ表します。

- yyyy/MM/dd
- yyyy-MM-dd
- yyyy.MM.dd
- yyyyMMdd
- yyyy/MM
- yyyy-MM
- yyyy.MM
- yyyyMM
- yyyy年MM月dd日
- yyyy年MM月
- yyyy年
- yyyy

日付型(yyyy-MM-dd)

次のいずれかの書式で表現された日付かどうかをチェックします。ここで　yyyyは年(4桁)、MM　は月(2桁)、dd は日(2桁)をそれぞれ表します。

- yyyy-MM-dd
- yyyy/MM/dd
- yyyyMMdd

日付型(yyyy-MM)

次のいずれかの書式で表現された日付かどうかをチェックします。ここで　yyyyは年(4桁)、MM　は月(2桁)、dd は日(2桁)をそれぞれ表します。

- yyyy-MM
- yyyy/MM
- yyyyMM

数字-半角全角文字(負数は許容)

入力値が半角文字または全角文字の数字かどうかをチェックします。数字の前にマイナス符号が含まれる場合も、数値として認めます。

数字-半角文字(負数は許容しない)

入力値が半角文字の数字かどうかをチェックします。数字の前にマイナス符号は含まれないものとして扱います。

数字-全角文字(負数は許容しない)

入力値が全角文字の数字かどうかをチェックします。数字の前にマイナス符号は含まれないものとして扱います。

アルファベット-半角全角文字（大文字、小文字を含む）

入力値に半角全角文字のアルファベット（大文字、小文字を含む）が含まれているかをチェックします。

アルファベット-半角文字（大文字、小文字を含む）

入力値に半角文字のアルファベット（大文字、小文字を含む）が含まれているかをチェックします。

アルファベット-半角文字（大文字のみ）

入力値に半角文字のアルファベット（大文字のみ）が含まれているかをチェックします。

アルファベット-半角文字（小文字のみ）

入力値に半角文字のアルファベット（小文字のみ）が含まれているかをチェックします。

アルファベット-全角文字（大文字、小文字を含む）

入力値に全角文字のアルファベット（大文字、小文字を含む）が含まれているかをチェックします。

アルファベット-全角文字（大文字のみ）

入力値に全角文字のアルファベット（大文字のみ）が含まれているかをチェックします。

アルファベット-全角文字（小文字のみ）

入力値に全角文字のアルファベット（小文字のみ）が含まれているかをチェックします。

ひらがな

入力値に平仮名が含まれているかをチェックします。

カタカナ-半角全角文字

入力値に全角あるいは半角のカタカナが含まれているかをチェックします。

カタカナ-半角文字

入力値に半角のカタカナが含まれているかをチェックします。

カタカナ-全角文字

入力値に全角のカタカナが含まれているかをチェックします。

記号-半角全角文字

入力値に全角あるいは半角の記号が含まれているかをチェックします。ここで「記号」とは以下に示す文字をいいます。

!	”	#	$	%	&	¥	(	)	^	
~	[	]	.	{	}	;	:	+	*	?
_	/	−								

記号-半角文字

入力値に半角文字の記号が含まれているかをチェックします。

記号-全角文字

入力値に全角文字の記号が含まれているかをチェックします。

スペース

入力値に全角または半角スペースが含まれているかをチェックします。

警告の設定

警告を表示しますが、ここで保存ボタンを押下すると登録自体は妨げないという例を図に示します。

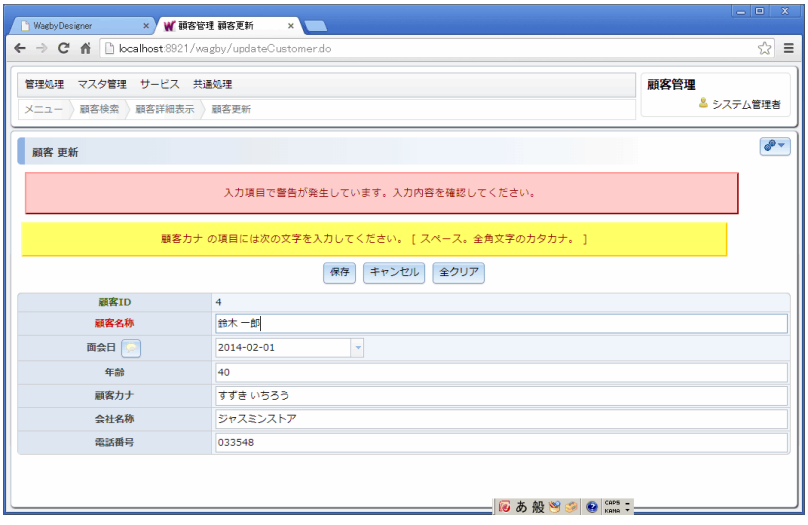


図 警告の表示

定義方法

許容文字種の設定で、「○」の代わりに「警告」を設定します。

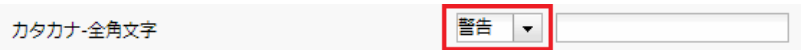


図 最大文字数の指定

複数の許容文字種を設定する場合、「○」と「警告」を混在させることはできません。いずれかに統一してください。

必要最小文字数の指定

許容文字種の必要最低文字数を指定することができます。

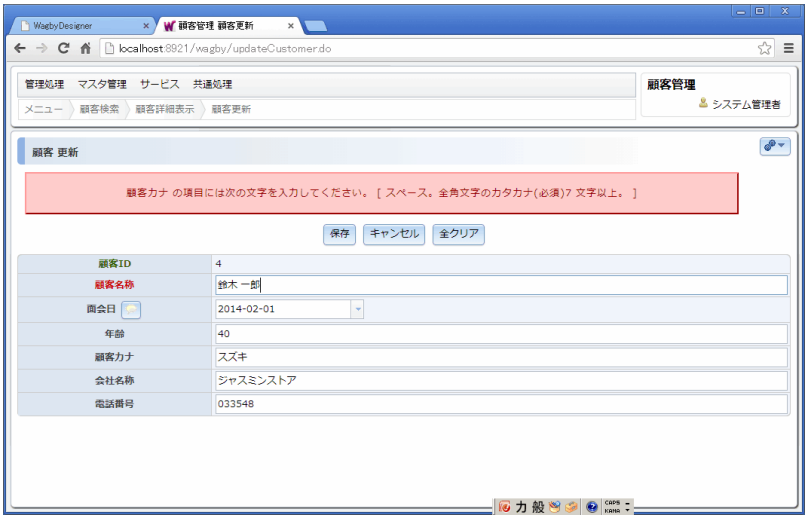


図 最低文字数制限エラー

定義方法

モデル項目の詳細定義にて「入力チェック-文字数チェック」に数値を設定します。　指定しない場合は文字数の制限はかかりません。

図 必要最小文字数を同時に指定する

フィルタとの関係

許容文字種を指定した場合は、自動的に「文字列（最小）」フィルタが適用されます。

ただし　″許容文字種が設定されていても入力フィルタの手動設定を行う″　を有効にすると、個別に入力フィルタを指定することができます。

図 必要最小文字数を同時に指定する

業務仕様によっては、フィルタを手動で変更した方がよい場合があります。

- 文字列型 － 入力フィルタと許容文字種
- 文字列型 － 入力フィルタの手動設定

[注意] 許容文字種を設定しているとき、入力フィルタとの組み合わせは動作しないパターンがあります。例えば許容文字種を半角カナとし、入力フィルタを文字列一般とすることはできません。文字列一般フィルタは常に半角カナを全角カナに変換する仕様のため、利用者は絶対に保存できなくなってしまいます。

任意の文字を指定する（8.1.0）

許容文字として、任意の文字を指定することができます。例えば次のような文字を入力させたいという要件を想定します。

- 半角英数
- 半角記号
- 半角スペース
- ″&″ と ″€″

次のように「任意の文字」の欄に、許容したい文字を直接、指定します。

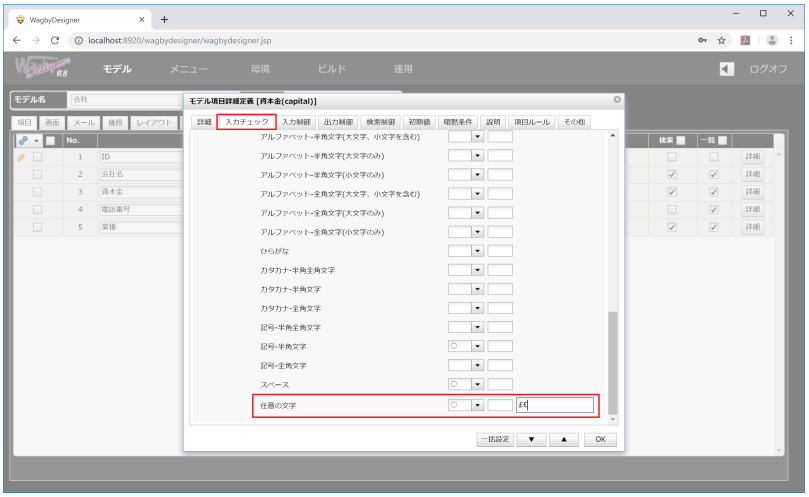


図 任意の文字を指定する

(74)

実行例

(74) スペースやコンマ、改行などは不要です。複数の文字を連続して指定してください。

許容できる文字でなければ、入力エラーになります。

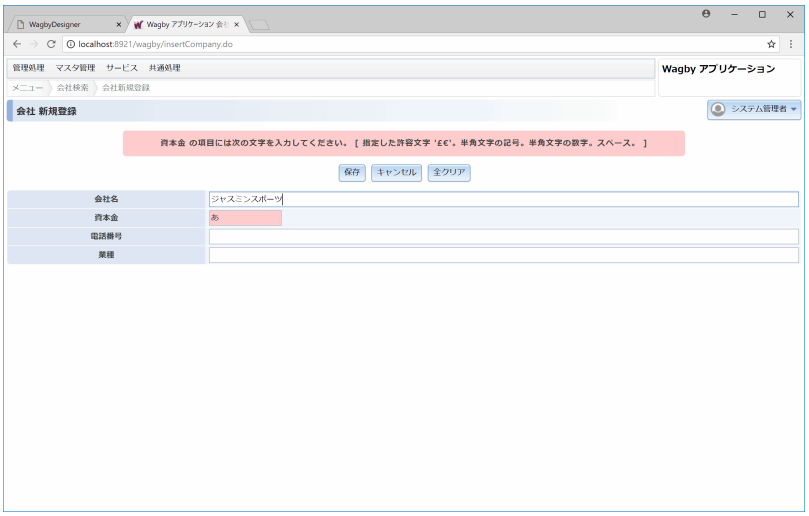


図 入力エラー画面の例

必要最小文字数を指定する（8.3.2）

″規定された記号を N 文字以上含む″という要件では、必要最小文字数を指定してください。

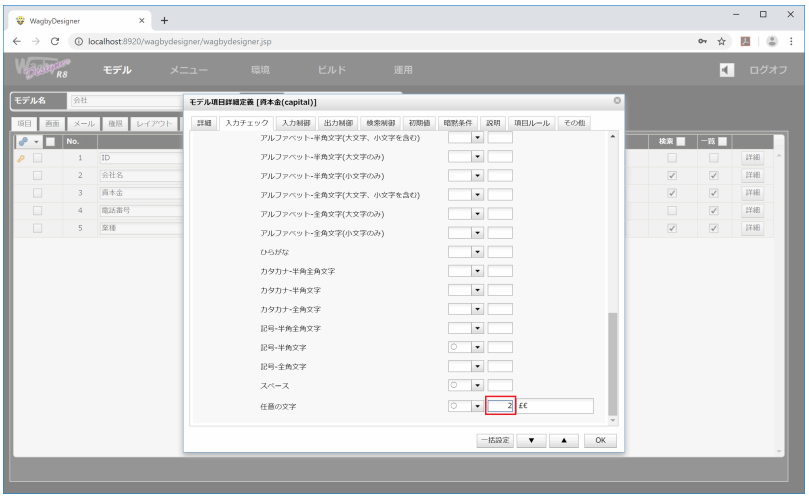


図 必要最小文字数を指定する

仕様・制約

- このチェックは「保存」などのボタンを押下したタイミングで、サーバ側で行われます。リアルタイムチェックではありません。
- エラーとなった入力項目は、背景が赤となります。テキスト入力部または日付時刻リストボックスが対象です。
- 入力対象ではない項目に対する許容文字種チェックを行うことはできません。具体的には計算式で求められる項目や、読み込み専用項目、参照連動項目です。
- 一覧更新画面（親子モデルにおける、子モデル同時更新画面も含む）では、警告指定は動作しません。

入力制御

最大入力文字数

例

入力できる最大文字数を指定します。設定した値を超える入力が行えなくなります。

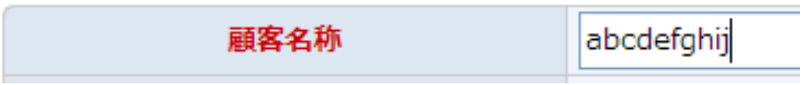


図 10文字までしか入らない例（それ以降のキー入力は無視される）

定義方法

モデル項目の詳細定義にて「入力制御-最大入力文字数」に数値を設定します。

図 最大文字数の指定

仕様・制約

実装の詳細

この設定は、input 要素の maxlength 属性に反映されます。例：<input type="text" maxlength="10">

文字数チェックとの併用 !-- N20170606_092319 -->

ブラウザによって input/@maxlength 指定の挙動が異なる場合があります。（例 : Google Chrome では日本語入力モードで数値を入力したあと、Enterキーを押さずにカーソルを移動させると入力できてしまう場合がある、など）

「文字数チェック」も併用して定義すると、より確実に指定文字数以上の入力を抑制することができます。

入力フィールド長

方法1 入力フィールド長を指定する

入力欄そのものが指定した文字数分まで用意されます。この指定では、入力できる文字数に制限はかかりません。見た目上の入力欄の長さになります。

図 入力欄の大きさが変わる

定義方法

モデル項目の詳細定義にて「入力制御 > 入力フィールド長」に数値を設定します。

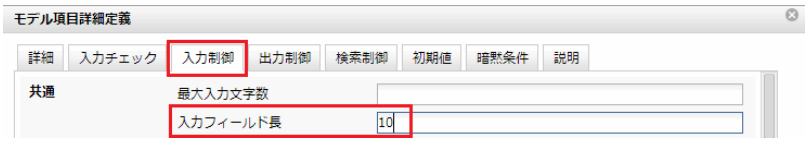


図 入力フィールド長の指定

1文字の単位

入力欄の長さの単位は、"em" となります。例えば20の場合、内部では

```
style="width:20em;"
```

と表現されます。

(75)

設定できる項目

文字列型、数字型およびモデル参照（検索画面）の入力欄に適用できます。

方法2 スタイルを指定する

リストボックス

リストボックスの入力フィールド長の標準は 200px となっています。

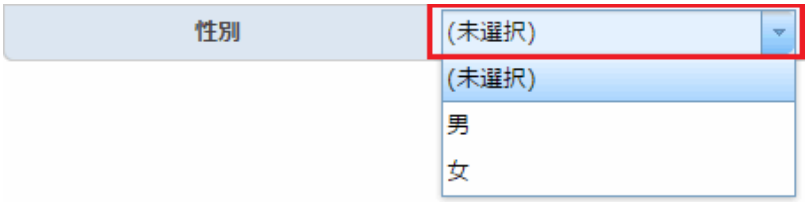


図 リストボックスの標準

この入力フィールド長を変更した例を示します。

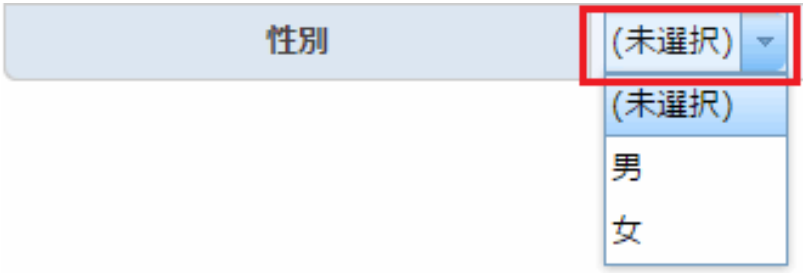


図 リストボックスの入力フィールド長を調整した例

(75) emとは CSS で定められた単位で、1文字分という意味です。その時点の1文字分の高さが 1em になります。文字サイズが14ptの場合、1emは14ptになります。

日付型（カレンダ入力）

標準では、プルダウンで表示されるカレンダーの幅がそのまま入力フィールドの幅となります。これを調整することができます。

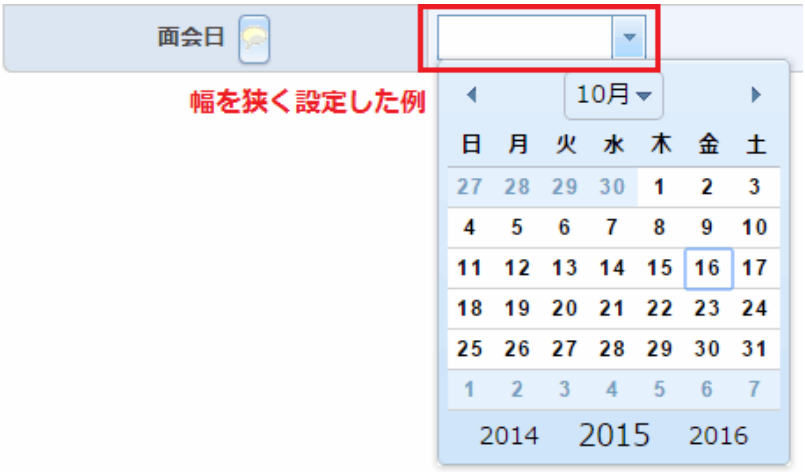


図 日付型（カレンダ入力）の入力フィールドの調整

(76)

定義方法

「モデル項目詳細定義 > 入力制御 > スタイル > 全画面」を設定します。

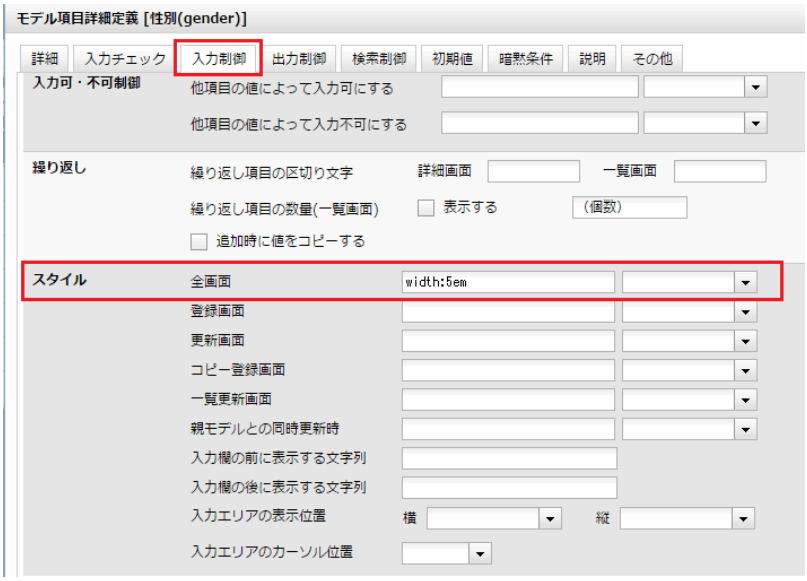


図 定義方法

ここでは、次のように設定しています。

```
width:5em
```

これは選択肢に表示される ″(未選択)″ という文字を表示できる、最小の長さです。 なお、適用箇所は空白とします。（空白時は、″入力欄のみ″ で扱われます。）

関連するマニュアル

・ 「入力欄のスタイル指定」

(76) プルダウンされるカレンダーの幅を小さくすることはできません。この設定は入力フィールドのみに適用されます。

設定できる項目

　文字列型、数字型およびモデル参照（検索画面）、リストボックス、日付型（カレンダ利用）の入力欄に適用できます。

テキストエリア

　″モデル項目 ＞ 文字列 － テキストエリア ＞ 行数を指定する″をお読みください。

隠し項目（登録・更新）

定義方法

　モデル項目の詳細設定ダイアログを開きます。「入力制御>共通」欄に用意された「隠し・読み込み専用」の隠し項目設定欄をチェックします。更新系の画面機能毎に細かく指定することもできます。

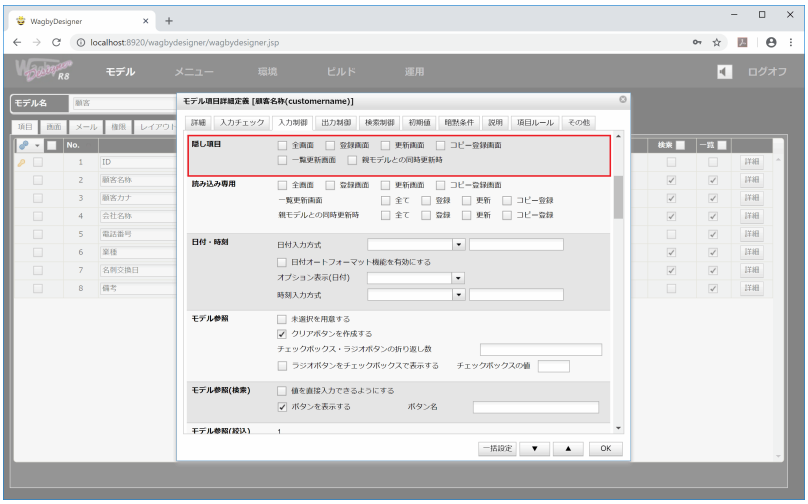


図 隠し項目の設定

　「全画面」を設定すると、他の画面設定は無効になります。（すべての画面について隠し項目となります。）

仕様

詳細画面、一覧画面との関連

　この設定は入力系画面に関わるものです。詳細画面、一覧画面には別途、設定欄が用意されています。

出力制御の隠し項目と併用すると、利用者からはこの項目は存在しないようにみえます。

帳票、メール送信機能との関連

　隠し項目に設定した項目は画面では表示されませんが（帳票出力の項目に設定した場合）帳票には出力されます。 また、メール送信機能においても同様にメール本文にデータが反映されます。

読み込み専用項目

定義方法

　モデル項目の詳細設定ダイアログを開きます。「入力制御>共通」欄に用意された「隠し・読み込み専用」の読み込み専用項目設定欄をチェックします。更新系の画面機能毎に細かく指定することもできます。

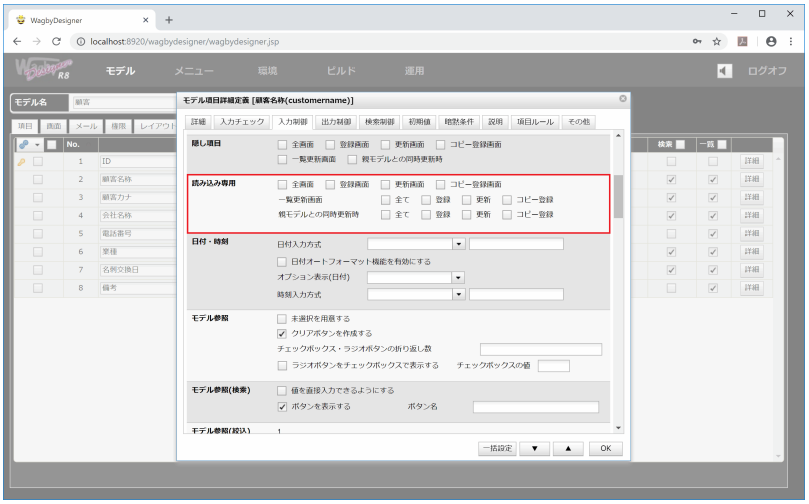


図 読み込み項目の設定

　「全画面」を設定すると、他の画面設定は無効になります。（すべての画面について読み込み専用となります。）

仕様

認証・認可

　この設定は「認証・認可」と連動していません。すべての利用者にとって無条件に読み込み専用となります。

　権限によって読み込み専用にするかどうかを制御することもできます。〝認証・認可 〉 項目単位の更新権限（更新可または閲覧のみ）〟をお読みください。

検索条件

　検索条件となる項目に本設定を適用すると、検索条件部の画面に値が表示されますが、検索条件値としては利用できません。 参照連動項目の値を（検索条件部に）参考として表示する、という場合に有効です。

日付・数値の初期値は表示しない

例

　この設定を行ったテスト用のモデルを示します。上の二つの項目は、非・繰り返し項目で本設定を有効にしています。下の二つの項目は繰り返し指定を行っています。

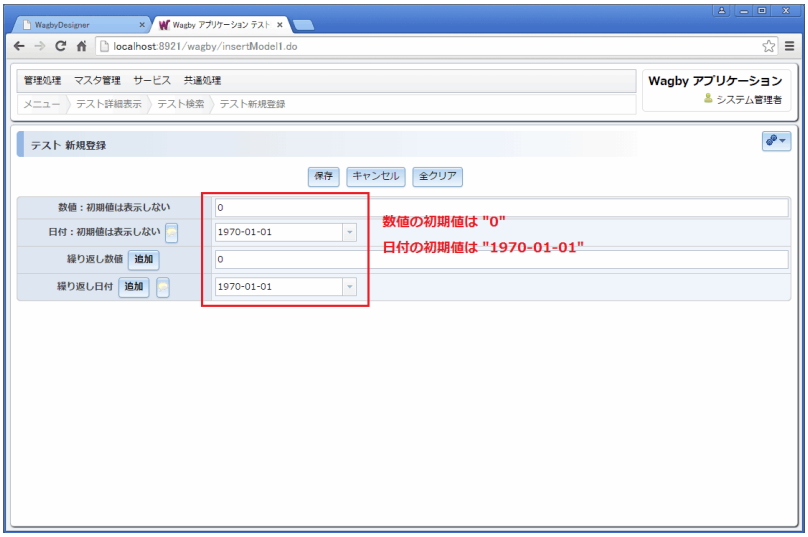


図 初期値の入力

詳細画面で値が表示されません。

図 詳細画面で値が表示されない

一覧画面でも値が表示されません。

図 一覧画面も値が表示されない

定義方法

モデル項目詳細定義の「出力制御 - 詳細画面」タブに用意された「共通」の「日付・数字の初期値は表示しない」を有効にします。

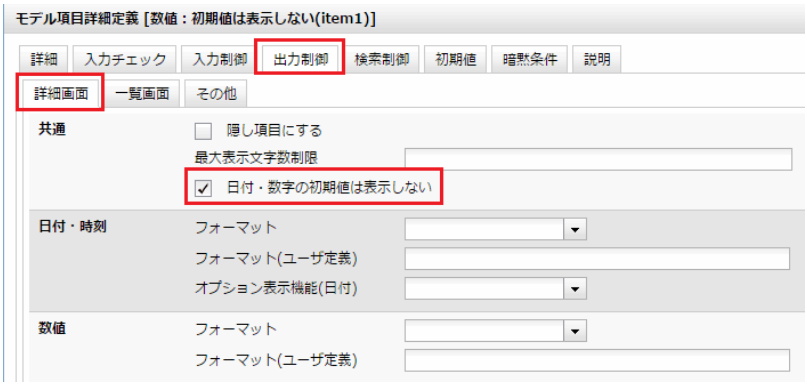


図 日付・数字の初期値は表示しないを有効にする

設定は「詳細画面」のみですが、一覧画面にも適用されます。

繰り返し項目への自動設定

数値・日付型項目を繰り返しとした場合、本設定は自動的に有効になります。

Enterキーの取り扱い

定義方法

Enterキーの取り扱いは3種類から指定できます。

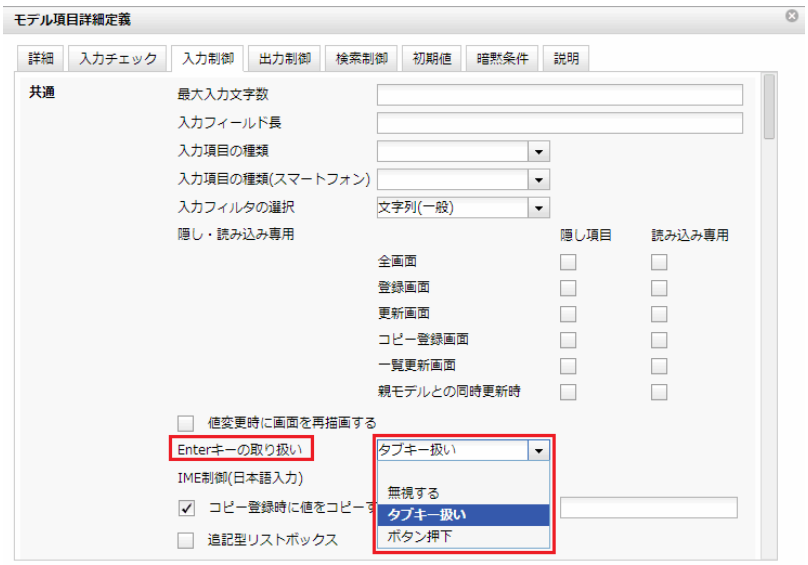


図 Enterキーの挙動を指定する

選択肢	挙動

（未設定）	フォーカスを移動させません。「無視する」と同じ。
無視する	フォーカスを移動させません。
タブキー扱い	フォーカスを他のテキストボックス、ボタンなどに移動させます。
ボタン押下	「保存」ボタンを押下することと同じ動作(サブミット)を行います。

利用例

データをキーボードで手入力する場合、最後以外の項目は「タブキー扱い」に、最後の項目は「ボタン押下」に設定することで、マウスで「保存」ボタンを押す手間がなくなり、すばやくデータの登録ができます。

制約

- 「テキストエリア」「ファイル」型では動作しません。
- 「クリア」「カレンダ」ボタンなど項目毎に用意されるボタンは、押下扱いになります。
- ご利用のブラウザによっては挙動が異なる場合があります。

IME制御

定義方法

IME制御は3種類から指定できます。

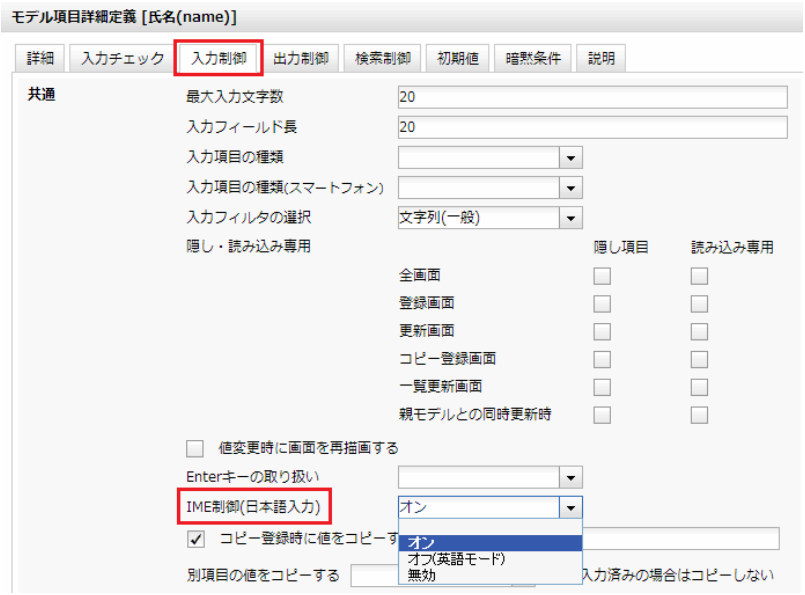


図 IME制御を指定指定する

選択肢	挙動
（空白）	何もしません。
オン	日本語入力モードになります。
オフ	英語入力モードになります。
無効	IMEが無効となり英語入力しかできません。手動で日本語入力モードに変更することもできません。

対応するWebブラウザ

この設定は Internet Explorer と Firefox で有効です。

仕様・制約

本設定はモデル参照(検索画面)で、値を直接入力する項目には適用されません。この制約は将来のWagbyで改善する予定です。

タブインデックス

定義方法

モデル項目詳細定義ダイアログの「入力制御 > 共通」欄の「タブインデックス」に数字を設定します。開始番号は1からとします。番号1の欄が最初の入力欄になります。

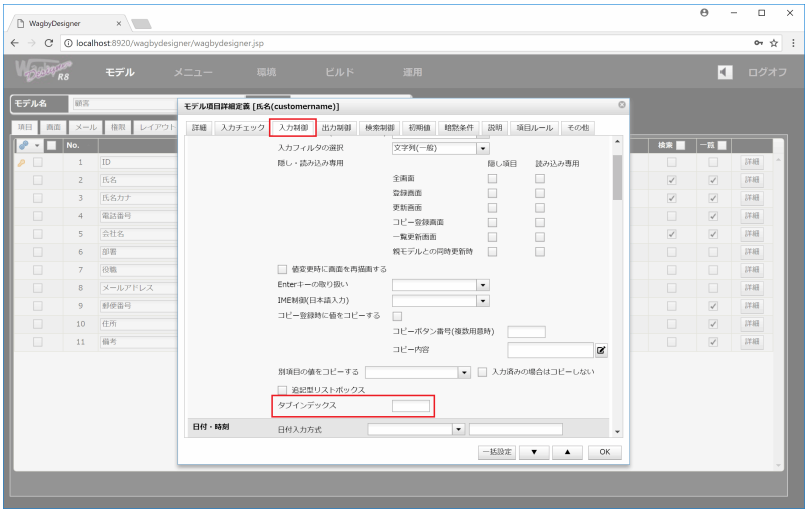


図 タブインデックスの設定

一括設定

この設定は項目全体で同じ数字が割り当てられないに配慮する必要があるため、一括設定をおすすめします。

ダイアログの ″タブインデックス″ という文字をマウスで選択した状態で「一括設定」ボタンを押すと、図の画面が表示されます。

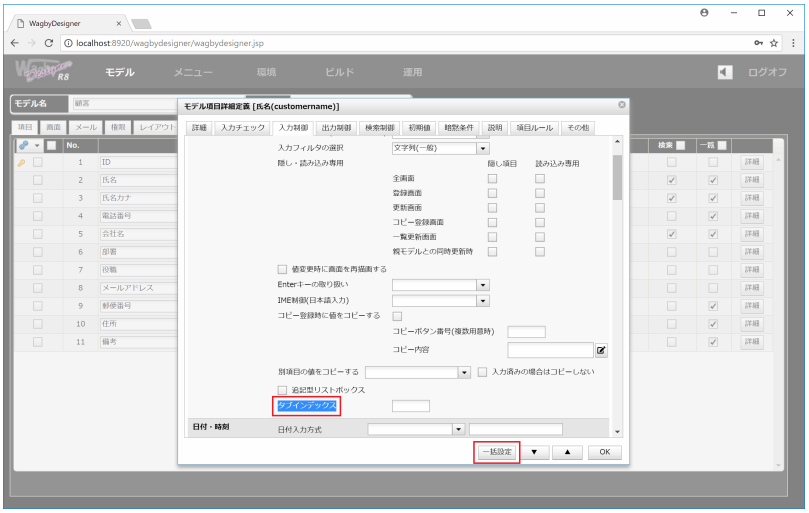


図 タブインデックスの一括設定(1)

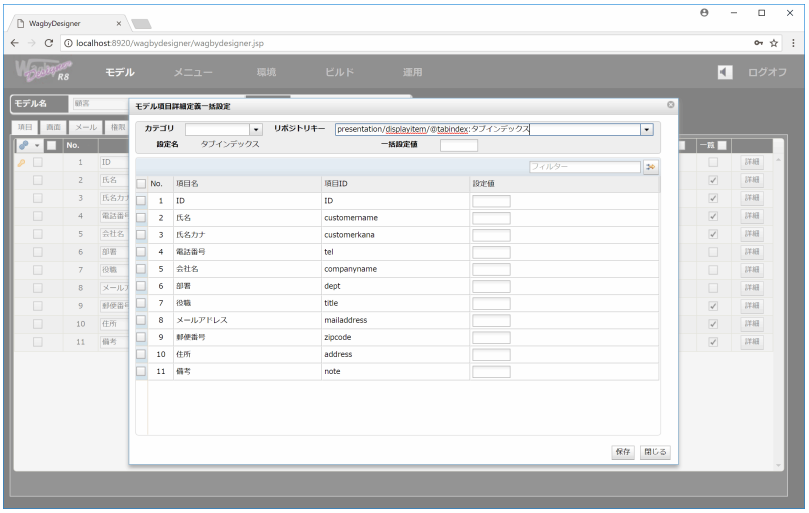


図 タブインデックスの一括設定(2)

最後の項目から先頭に向かって数字を割り当てた例を示します。なおテキストエリア項目（ここでは「備考」項目）にはタブインデックスは設定できないため、これは除いています。

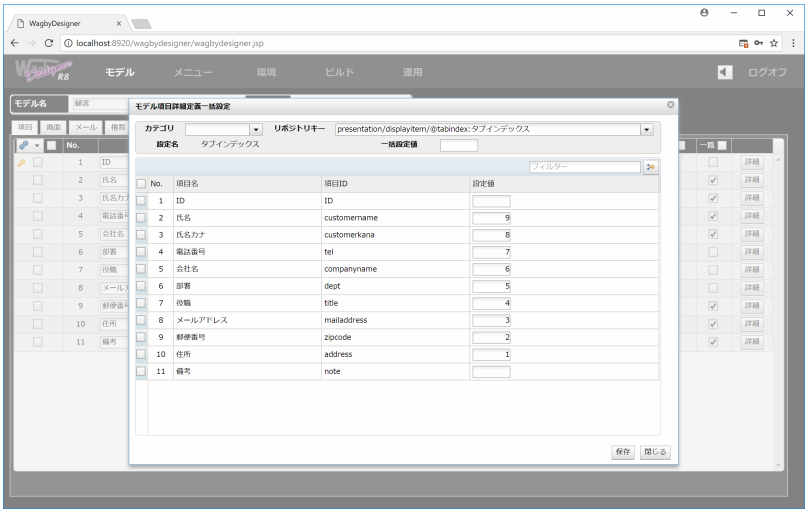


図 タブインデックスの設定例

実行例

最初の入力欄は（タブインデックスに″1″を設定した）「住所」項目になります。値を入力し、タブキーを押下すると、次の入力欄に遷移します。

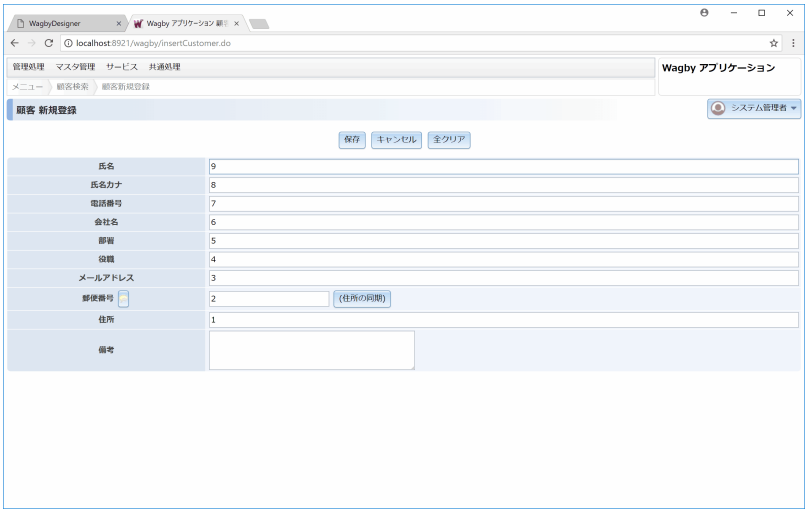


図 タブインデックスを設定した入力画面（タブの移動のつど連番を入力した例）

繰り返しコンテナの場合

繰り返しコンテナ内の項目には「すべて同じ値」を指定してください。また、繰り返しコンテナ項目およびコンテナID項目には数値を割り当てません。

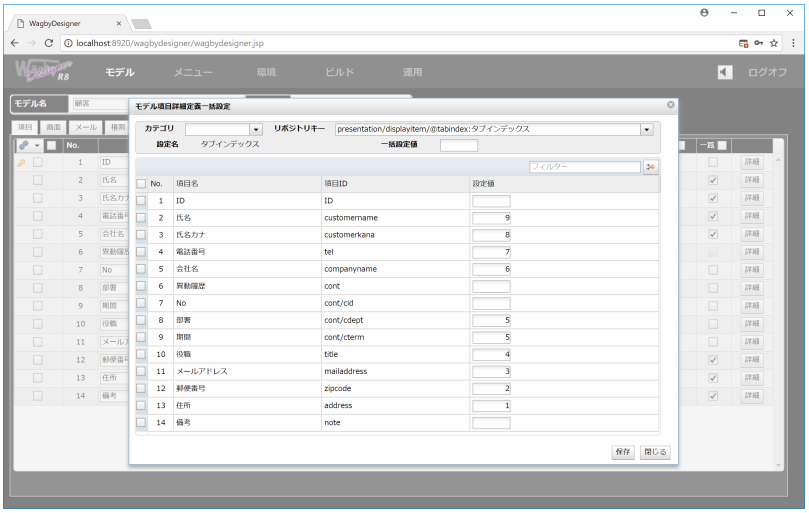


図 繰り返しコンテナの設定例

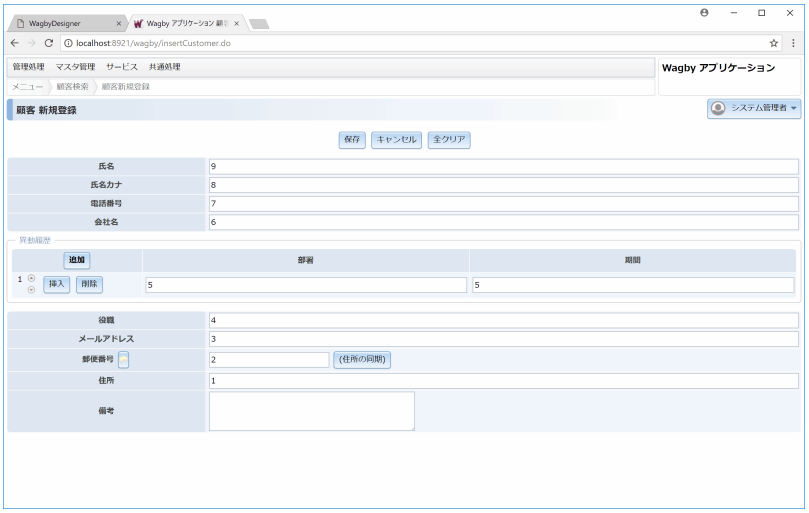


図 繰り返しコンテナを含むモデルの動作例

仕様・制約

- テキストエリア項目（リッチテキストエリアを含む）にはタブインデックスを設定することはできません。
- 読み込み専用項目や隠し項目にはタブインデックスを設定することはできません。
- 「住所の同期」など、いくつかのボタンはタブインデックスを指定できませんが、タブ移動の対象となります。
- 繰り返しコンテナ内の項目にはすべて同じ値を設定してください。
- 日付入力方式、時刻入力方式をリストボックスとした項目にはタブインデックスを指定することができます。

別項目の値をコピーする

例

伝票モデルの「購入」欄に値を入力すると、「請求先」欄に自動的に転記される例を示します。

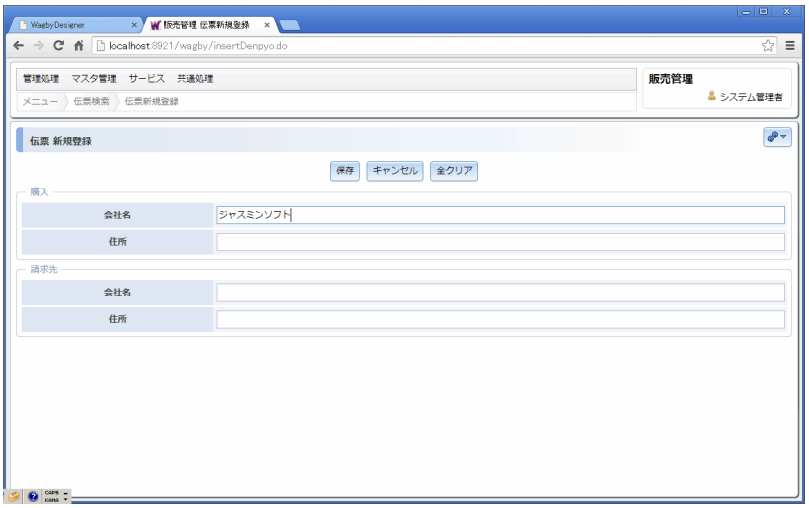


図 購入欄の会社名を入力する

カーソルを次の入力欄に移動したタイミングで、会社名が転記されます。

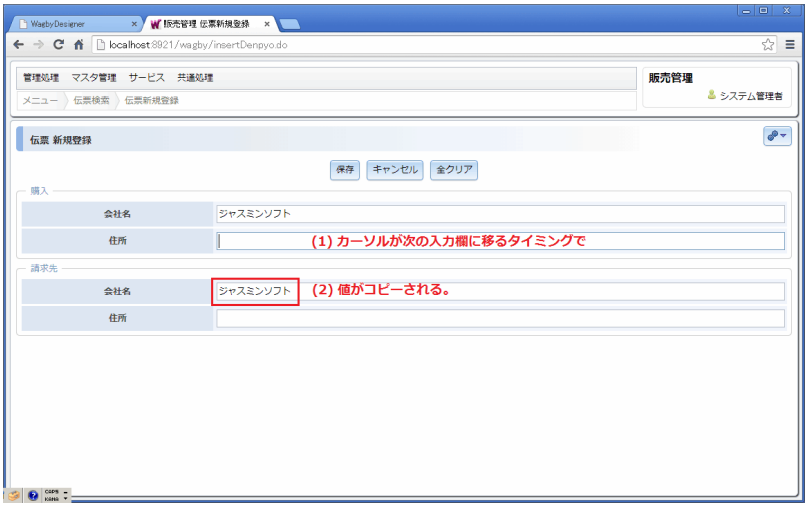


図 カーソル移動により、会社名が転記された

同様に、住所も転記させた例です。

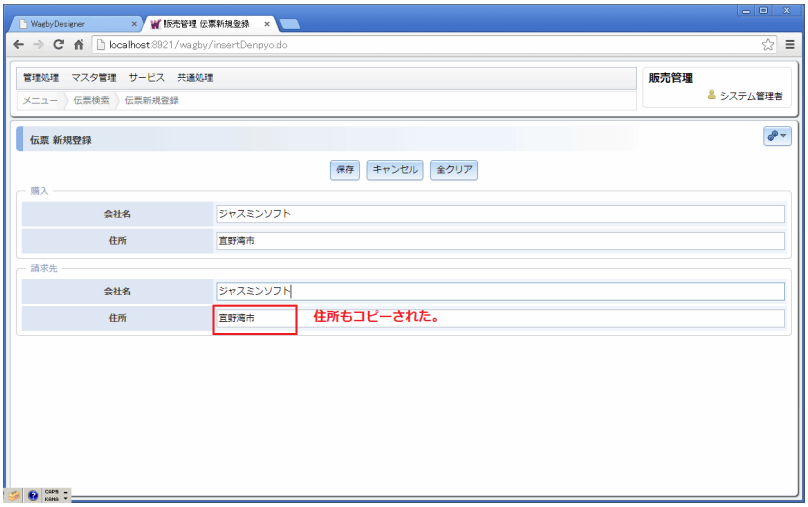


図 住所も転記させた

定義方法

コピー先の項目側で、「入力制御>共通>別項目の値をコピーする」に、コピー元の項目を設定します。

図 会社名のコピー設定

入力済みの場合はコピーしない

「入力済みの場合はコピーしない」を有効にすると、すでに値が入力されている場合は、このコピー処理を行いません。

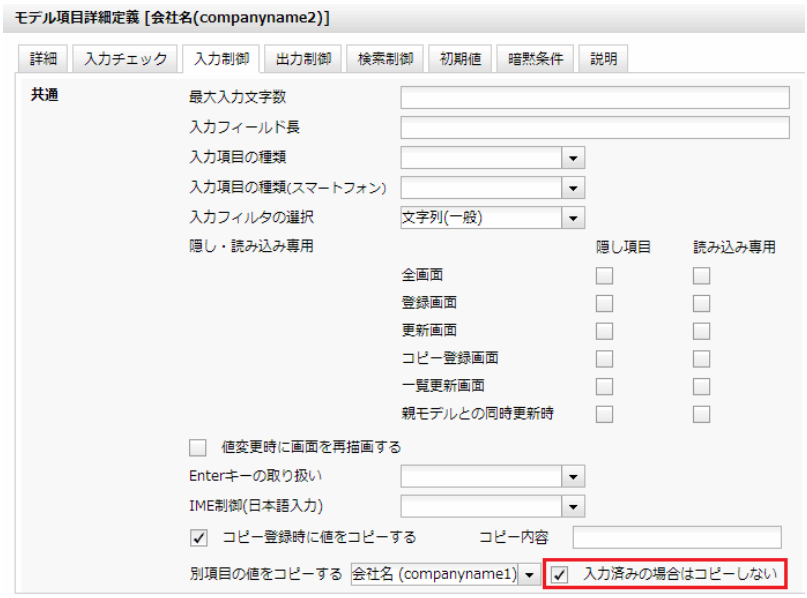


図 入力済みの場合はコピーしない設定

検索条件 － 範囲検索

数値型や日付型で範囲検索を有効としている場合、2つの設定欄が用意されます。上限と下限をそれぞれ指定できます。

図 範囲検索が有効の場合、2つの設定欄が用意される（上限および下限）

仕様・制約

- 本機能はコピー元の値が変更されたタイミングで動作します。（コピー先の入力値の有無にかかわらず、コピー元の値を変更せずにカーソルを外しても値はコピーされません。）
- 文字列型および数字型で利用できます。
- １つの項目を、同時に複数の項目にコピーすることができます。
- コピー元とコピー先は項目の型が同じである必要があります。
- 繰り返しコンテナ内の項目同士はコピーできます。コピー先側の指定では「（コピー元の）繰り返しコンテナ名/項目名」と記述します。
- コピー元が通常の項目で、コピー先が「繰り返し項目」または「繰り返しコンテナ内の項目」という指定を行うことはできません。
- コピー元が「繰り返し項目」または「繰り返しコンテナ内の項目」で、コピー先が通常の項目という指定を行うことはできません。
- コピー先に、ウィザードで別の画面にある項目を指定することはできません。
- コピー元およびコピー先に読み取り専用項目（参照連動項目や自動計算項目など）を指定することはできません。

別項目をグループとして扱い、値をまとめてコピーする

例

伝票モデルの「購入」欄に値を入力すると、「請求先」欄に自動的に転記される例を示します。

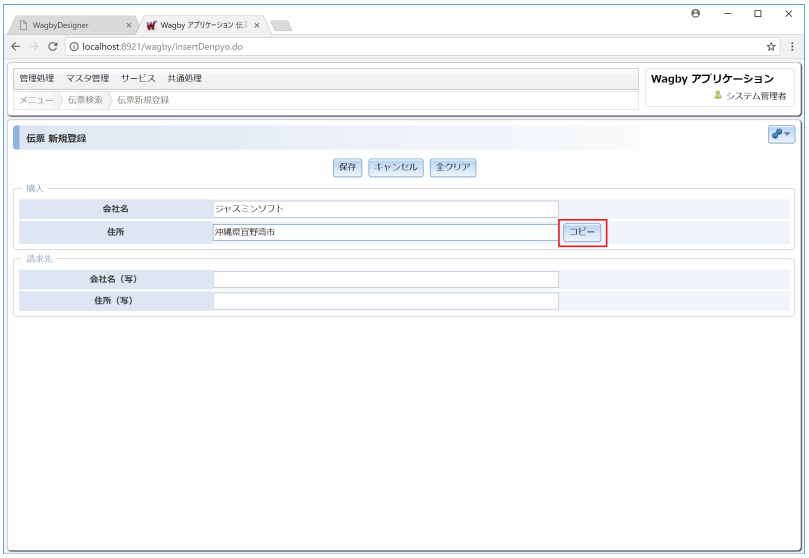


図 購入欄の会社名と住所を入力する

用意された「コピー」ボタンを押下すると、複数の項目の値がコピーされます。

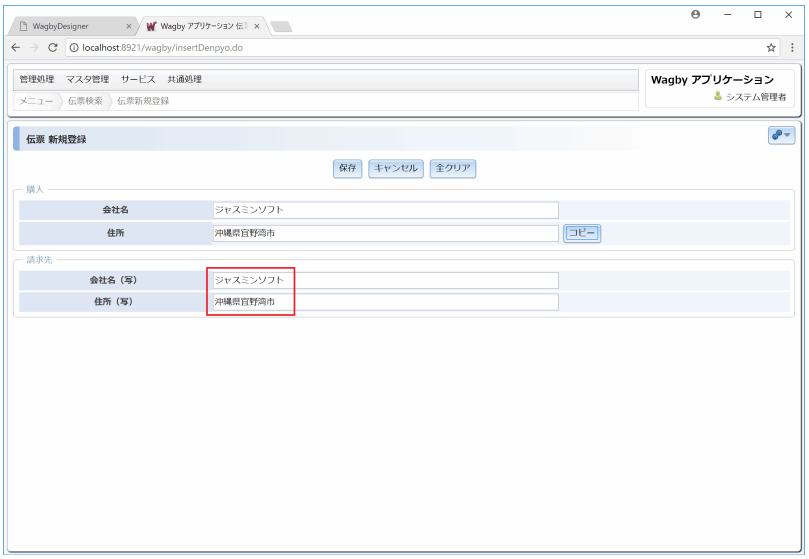


図 ボタン押下により、会社名と住所が転記された

定義方法

ここでは「住所」項目にボタンを用意した例を説明します。

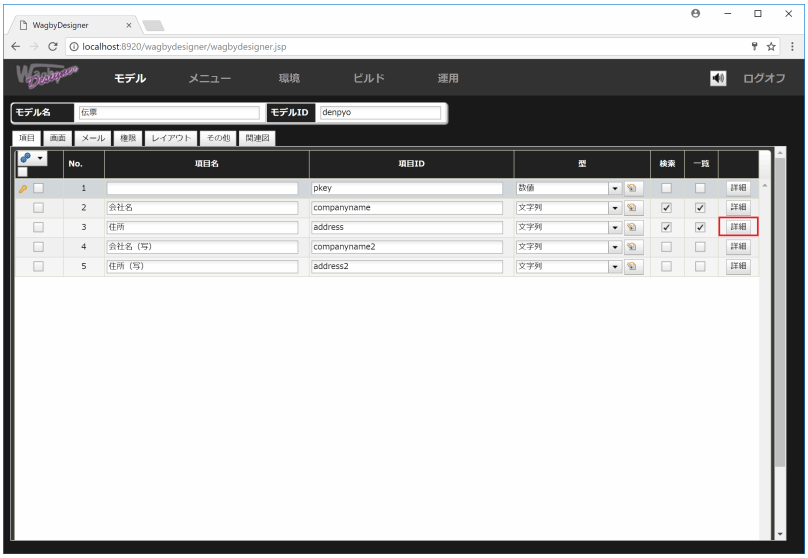


図 伝票モデルの住所項目

モデル項目の詳細設定から「その他」タブを開きます。一つの項目に最大5つまでの追加ボタンを用意することができます。

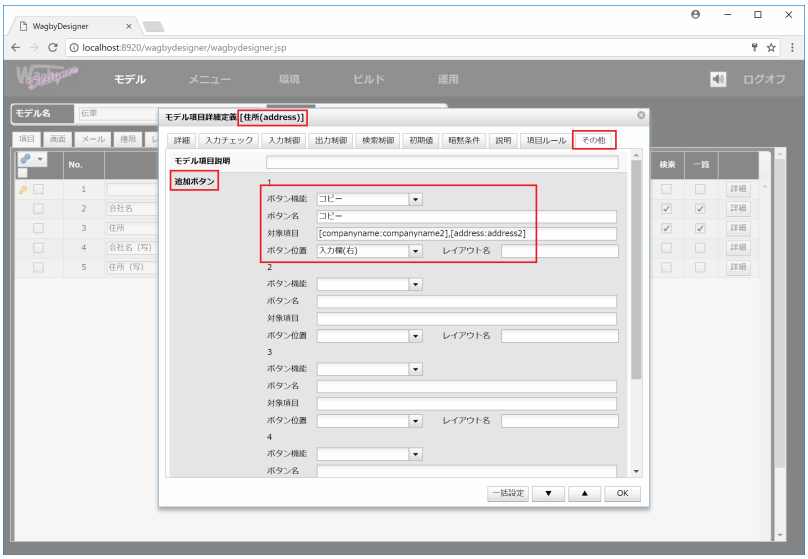


図 追加ボタンの設定

項目	説明	入力例
ボタン機能	「コピー」「クリア」「オリジナル」を選択できます。コピーとクリアはこのページで説明します。オリジナルは、任意のJavaScriptを記述するために用意されています。	コピー
ボタン名	任意の文字列です。ボタン機能を「コピー」「クリア」とした場合、ボタン名の標準も同じになります。（変更することもできます。）	コピー
対象項目	ボタン機能を「コピー」「クリア」とした場合、対象項目を指定します。[後述]	item1,item2
ボタン位置	ボタンの配置方法を指定します。「入力欄(右)」「入力	入力欄(右)

	欄(左)」「レイアウト名(右上)」「レイアウト名(左上)」から選択します。[後述]	
レイアウト名	ボタンの配置方法が「レイアウト名(右上)」「レイアウト名(左上)」のとき有効になります。	(レイアウト名)

対象項目

コピー

次のように記述します。

```
[Source item1:Destination item1],[Source item2:Destination item2],...
```

- "[","]" で囲み、コピー元項目とコピー先項目をコロン ":" で区切ります。
- コンマ "," を使って複数のコピーを指定することができます。
- 単純に項目名だけをコンマ区切りで列挙すると、コピー元項目が、この設定を行なった項目となります。つまり一つの入力値を、同時に複数の項目にコピーすることができます。

クリア

次のように記述します。

```
Clear item1,Clear item2,...
```

- コンマ "," を使って複数のコピーを指定することができます。

クリアボタンを用意する

コピーボタンとともに、クリアボタンを用意することもできます。ボタン押下により、複数の項目の値を消去します。

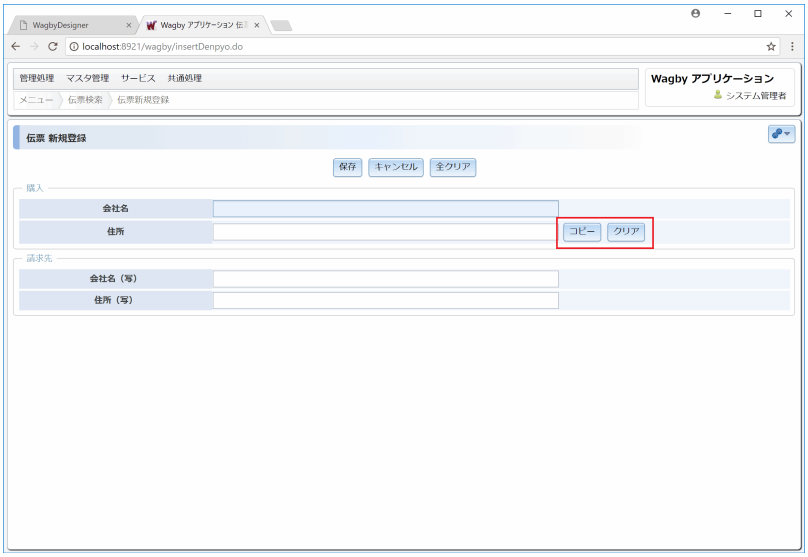


図 クリアボタン

定義方法

二つ目のボタン定義を用意します。ボタン機能を「クリア」とします。対象項目をコンマ区切りで指定します。

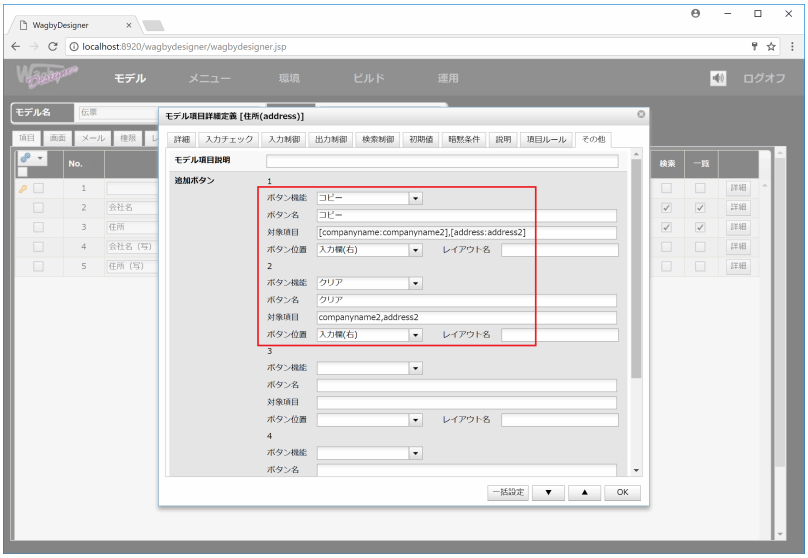


図 クリアボタンを用意する

ボタンの位置を指定する

ボタンの位置を指定した例を示します。

入力欄の左

入力欄の左側に配置した例を図に示します。

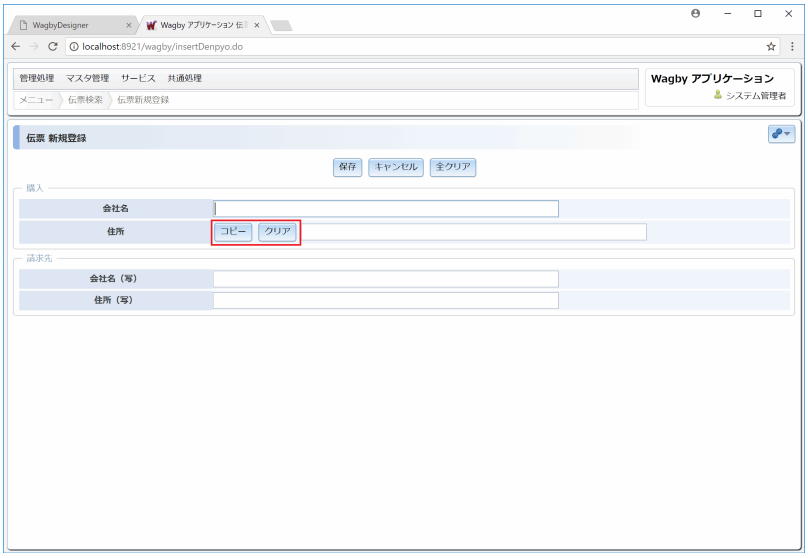


図 入力欄の左側にボタンを配置する

定義方法

ボタン位置を「入力欄(左)」とします。

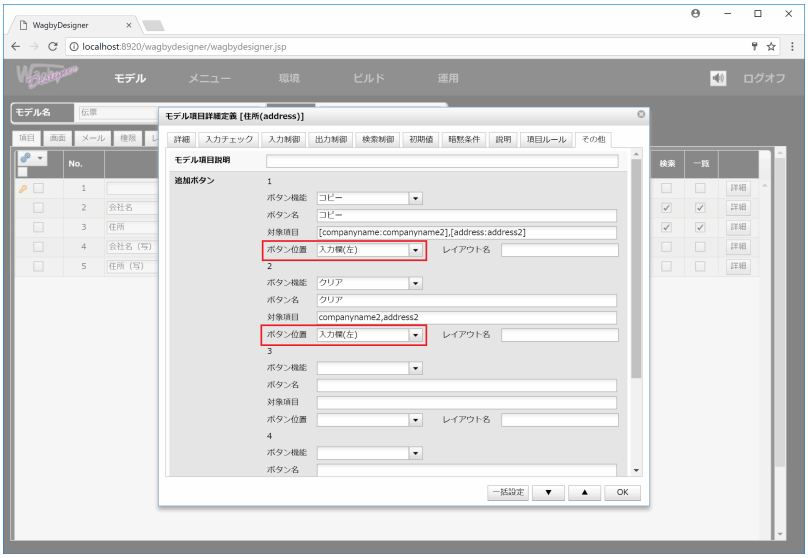


図 ボタン位置を「入力欄(左)」とする

Wagbyの標準では、入力欄の幅が標準で"100%"となっているため、上の設定だけではボタンと入力欄が横並びにならないことがあります。ボタンを左に配置する場合、モデル項目詳細定義ダイアログの入力制御タブの「入力フィールド長」に数値で長さを設定するか、スタイルの「全画面」に"width: 85%"などと設定し、入力欄の長さを調整するとよいでしょう。これによってボタンと入力欄が横並びになります。

入力欄の左と右

入力欄の左側および右側にそれぞれ配置した例を図に示します。

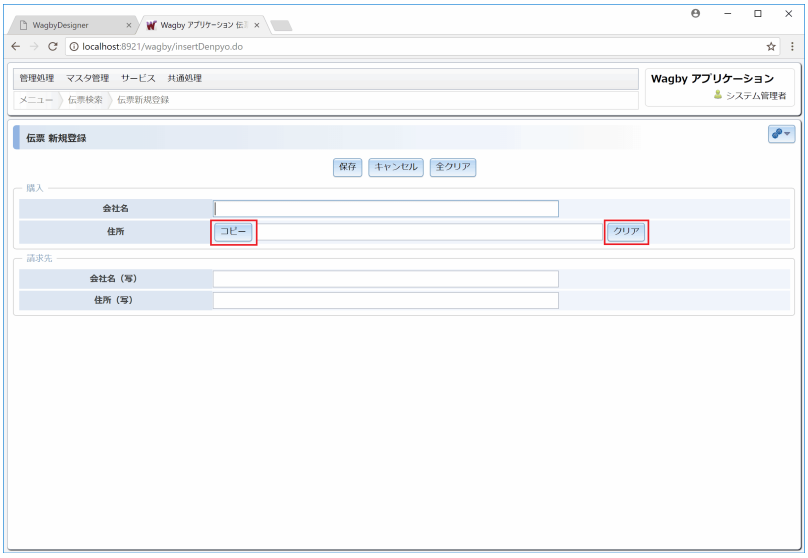


図 入力欄の左側と右側にボタンを配置する

定義方法

ボタン位置をそれぞれ「入力欄(左)」「入力欄(右)」とします。

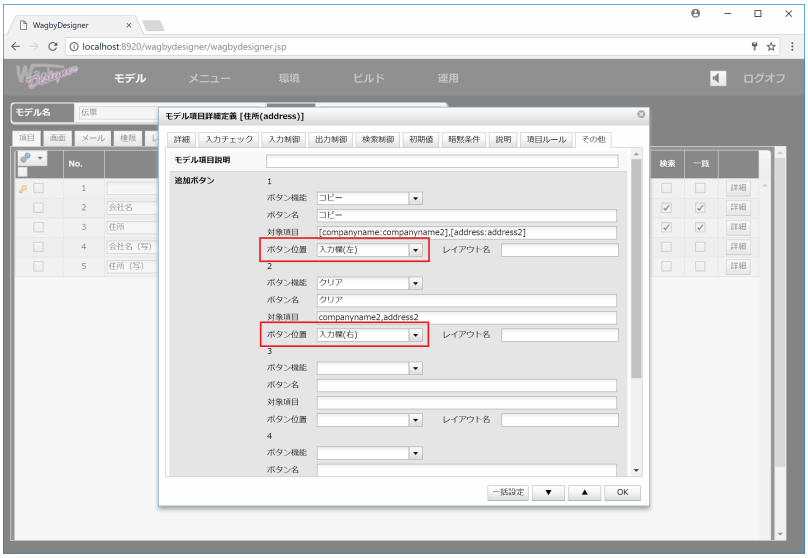


図 ボタン位置を「入力欄(左)」「入力欄(右)」とする

レイアウト名の左

レイアウト名の左側に配置した例を図に示します。

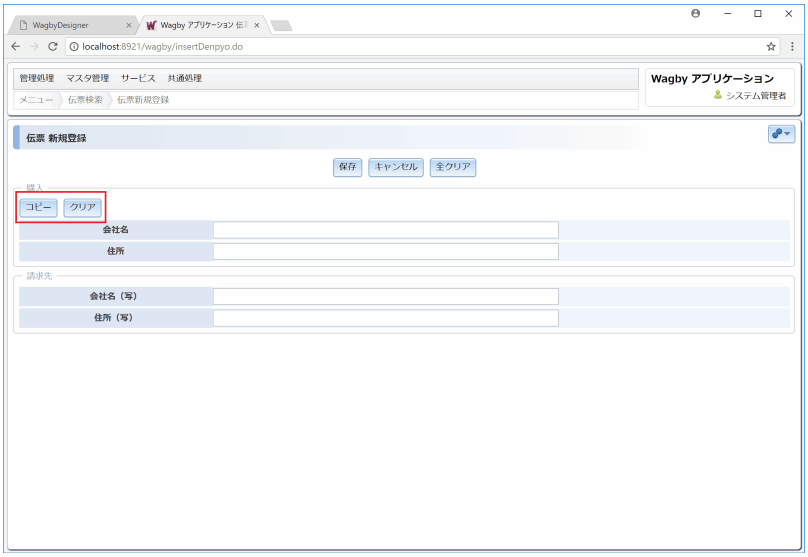


図 レイアウト名の左側にボタンを配置する

定義方法

ボタン位置を「レイアウト名(左上)」とします。また、図で設定済みの「レイアウト名」を利用します。

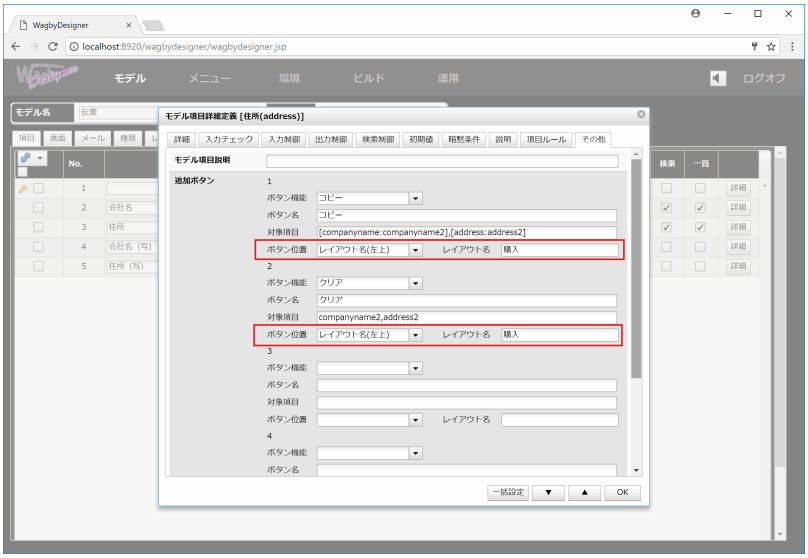


図 ボタン位置を「レイアウト名(左上)」とする

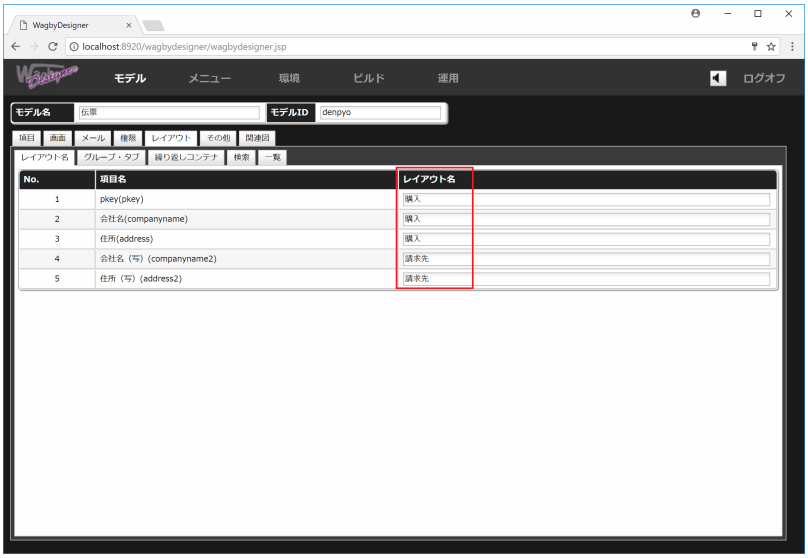


図 レイアウト名

レイアウト名の右

レイアウト名の右側に配置した例を図に示します。

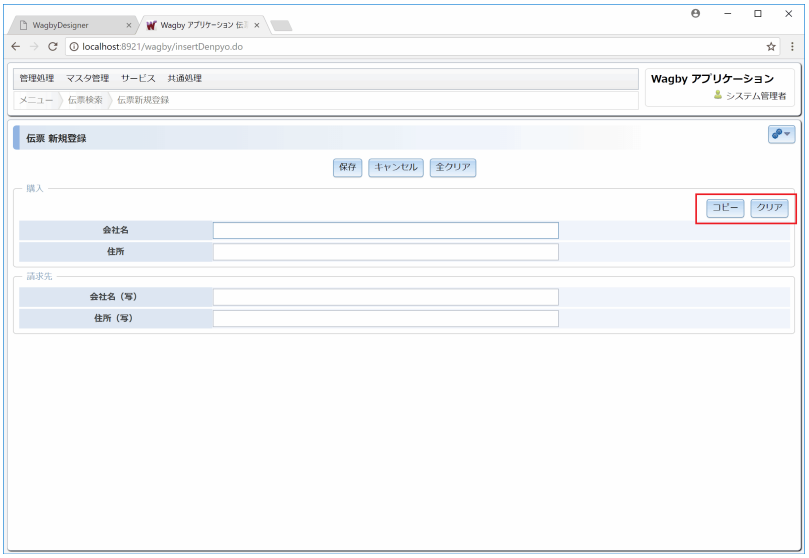


図 レイアウト名の右側にボタンを配置する

定義方法

ボタン位置を「レイアウト名(右上)」とします。

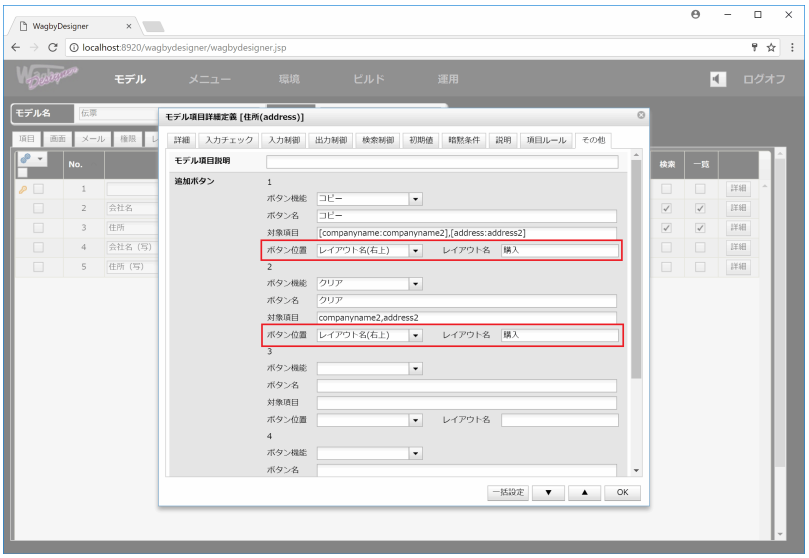


図 ボタン位置を「レイアウト名(左上)」とする

レイアウト名の左右

レイアウト名の左側および右側にそれぞれ配置した例を図に示します。

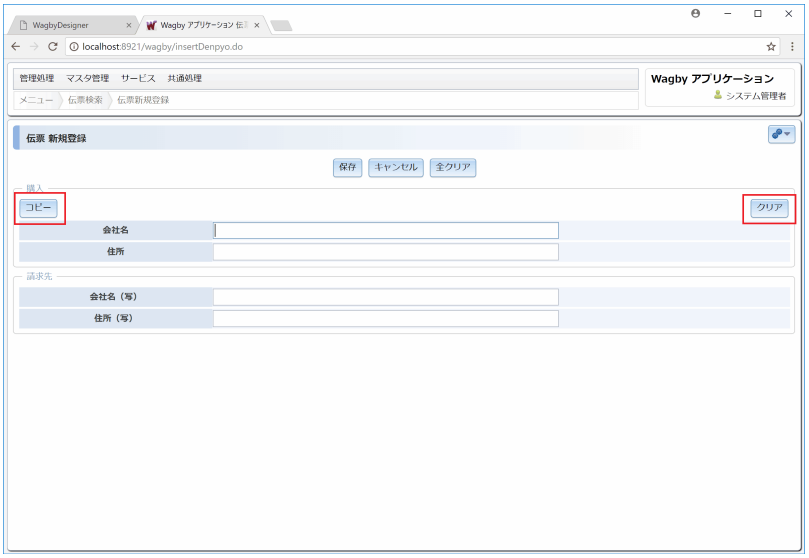


図 レイアウト名の左側と右側にそれぞれボタンを配置する

定義方法

ボタン位置をそれぞれ「レイアウト名(左上)」「レイアウト名(右上)」とします。

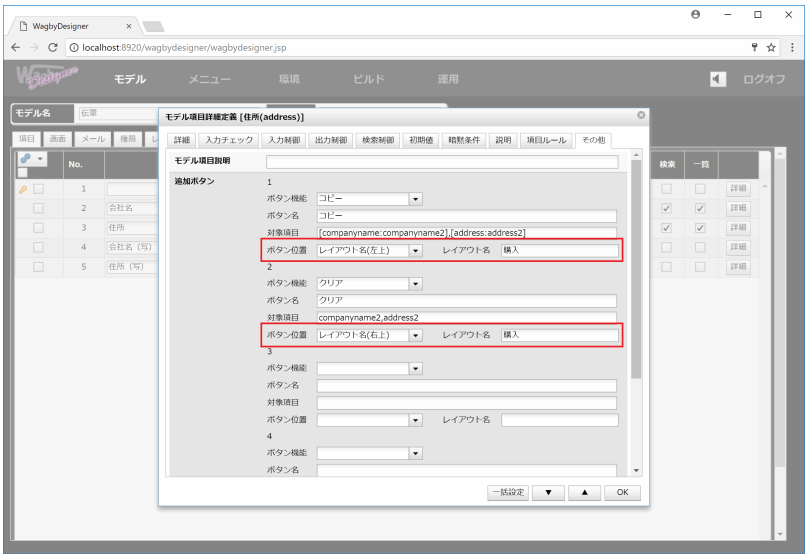


図 ボタン位置をそれぞれ「レイアウト名(左上)」「レイアウト名(右上)」とする

オリジナルボタンの処理を記述する

ボタン機能に「オリジナル」を指定し、ボタン押下時の処理を JavaScript で記述することができます。
詳細...

仕様・制約

- 本機能はボタン押下のタイミングで動作します。
- コピー先の項目に値が設定されていた場合でも、このボタン押下により上書きされます。
- この処理は Web ブラウザの JavaScript によって行われるため、サーバとのやりとりは生じません。（値の書き換えによってサーバのスクリプトが動作することはありません。）
- 一覧更新画面には対応していません。
- １つの項目を、同時に複数の項目にコピーすることができます。
- コピー元とコピー先は項目の型が同じである必要があります。

- 繰り返しコンテナ内の項目同士はコピーできます。コピー元、コピー先にはそれぞれ「繰り返しコンテナ名/項目名」と記述します。
- 繰り返しコンテナ内の項目にボタンを配置する場合は、レイアウト名への配置は行えません。入力欄への配置のみとなります。
- 繰り返し項目に本設定を適用することはできません。
- ボタンを「レイアウト名(左上)」「レイアウト名(右上)」に配置する場合、異なるレイアウトに配置することはできません。同じレイアウト内に配置してください。
- コピー元が通常の項目で、コピー先が「繰り返し項目」または「繰り返しコンテナ内の項目」という指定を行うことはできません。
- コピー元が「繰り返し項目」または「繰り返しコンテナ内の項目」で、コピー先が通常の項目という指定を行うことはできません。
- コピー元およびコピー先はテキストフィールドまたはテキストエリアである必要があります。テキストエリア（リッチ）には対応していません。
- コピー先に、ウィザードで別の画面にある項目を指定することはできません。
- コピー元およびコピー先に読み取り専用項目（参照連動項目や自動計算項目など）を指定することはできません。
- コピー元とコピー先が他モデル参照(検索)項目の場合は動作しません。 R8.3.0 で動作するようになりました。

項目のオリジナルボタン押下で JavaScript を動作させる(JavaScript)

例

項目へのボタンの配置方法は「別項目をグループとして扱い、値をまとめてコピーする」をお読みください。

ボタン機能に「オリジナル」を指定し、ボタン押下時の処理を JavaScript で記述することができます。ここでは Wagby が内部で利用している Dojotoolkit ライブラリを使ったサンプルを紹介します。

(77) (78)

ボタンを押すと計算を行う

ボタン押下により、item1 と item2 の入力値を加算したものを item3 にセットするサンプルです。Wagbyの「自動計算」設定と異なり、式を直接、JavaScript で記述します。サーバとの通信は発生せず、クライアント（Webブラウザ）だけで処理されます。

ここではモデル model1 に対して、MyInsertModel1.js を用意した例です。これを customize/webapp/model1 フォルダに保存してビルドしてください。（モデルにあわせて MyXX.js ファイル名を用意してください。）

```
require(["dijit/registry", "dojo/ready"], function(registry, ready) {
  // 画面表示時に対象の入力項目を取得してイベント処理を設定する
  ready(function() {
    var item1Widget = registry.byId("model1_p$002fitem1"); // item1 項目の入力欄
    var item2Widget = registry.byId("model1_p$002fitem2"); // item2 項目の入力欄
    var item3Widget = registry.byId("model1_p$002fitem3"); // item3 項目の入力欄
    var btnItem3 = registry.byId("item3_nop_1"); // item3 項目のボタン
    // item3 項目のボタン押下時の処理を記述
    btnItem3.on("click", function (){
      var item1Value = item1Widget.get("value"); // item1 項目の入力値を取得
      var item2Value = item2Widget.get("value"); // item2 項目の入力値を取得
      item3Widget.set("value", item1Value + item2Value); // item3 項目の入力欄へ合計値をセットする
    });
  });
});
```

(77) ここでいう "JavaScript" は、サーバサイドではなく、Webブラウザで動作するものです。

(78) Dojotoolkit は各種の入力部品を提供しているオープンソースのライブラリです。Wagby に同梱されています。

他項目の値による入力制御

入力可制御

ステータスが〝トラブル〟のとき、「原因」項目を入力できる例を説明します。 ステータスマスタは図のように用意されているとします。

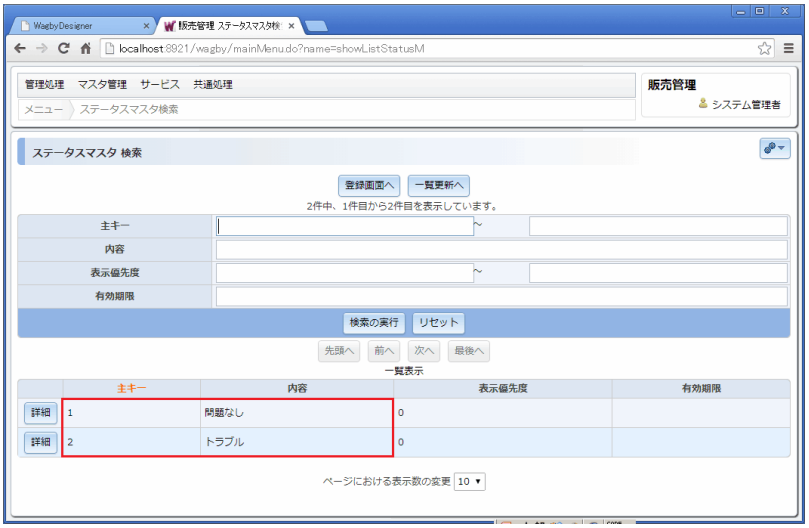


図 ステータスマスタの設定

レポート新規登録画面の初期状態は、「原因」項目の入力が行えません。

図 原因項目は入力できない

ステータスに〝トラブル〟を選択すると、「原因」項目が入力できます。

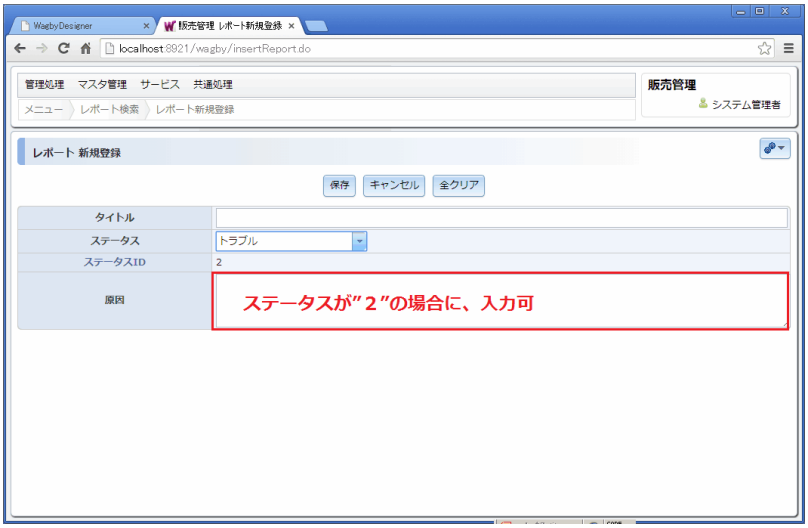


図 原因項目の入力が可能になる

定義方法

上記サンプルアプリケーションを実現する「レポート」モデルを図に示します。

図 レポートモデルの定義

「原因」項目の詳細定義ダイアログを開きます。　　「入力可・不可制御>他項目の値によって入力可にする」に〝項目名=値〟という形式で式を記述します。

```
status=2
```

図 他項目の値によって入力可にする設定

複数の値を指定する場合

一つの項目に対して複数の値を指定する場合は、","　（カンマ）区切りで指定します。以下の例は優先度が 1 または 2 の場合という意味になります。

```
status=1,status=2
```

他項目に値が入力されている場合に入力可とする

ある項目に値が入力されている場合、という条件を記述できます。他項目の名前がstatusのとき、

```
status=
```

とします。値の指定が無いため、入力値がある場合に条件が成立するようになります。

変更時の既存データの扱い

上記例において、新規登録時に「原因」を記載後、「ステータス」を "トラブル" 以外に変更したとします。このとき、すでに入力済みの「原因」の扱いをどうするかについて、二つの方法を提供しています。

選択肢	挙動
空白で上書き（標準）	入力不可となった段階で、入力値を消去（空白で上書き）します。（すでに入力済みの項目値は画面上に維持されますが、サーバには送信されません。そのため入力値は消去されます。）
値を保持	入力不可となった場合でも、すでに入力された値は残します。

(79)

この設定の切り替えは、図 に示すように選択できます。未設定時は「空白で上書き」となります。

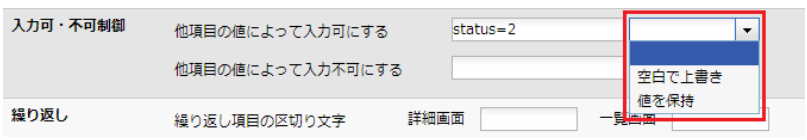


図 変更時の既存データの扱いを指定する

(79) 入力可・不可制御は、条件により入力項目の disabled 属性を true, false へ切り替えることで実現しています。 入力項目の disabled 属性が true の場合は入力欄がグレーアウトされ、入力値があってもサーバへは値が送信されません。これは HTML の仕様です。

必須指定との組み合わせ

入力可とする制御は必須チェックと組み合わせて定義することができます。 「入力可」になった場合に必須入力項目となります。

補足：「ステータスID」項目の式

上記の例で「ステータスID」という項目は本来、不要です。ID値を目視するために用意しています。 この項目は（IDを格納するため）数値型としており、次の式を適用しています。

```
${status#id}
```

式の記述方法の詳細は「計算式>モデル参照」をお読み下さい。

入力不可制御

条件に合致した場合に入力不可にする、という指定を行うこともできます。 図は、ステータスが"1"の場合に入力を不可にするという例です。

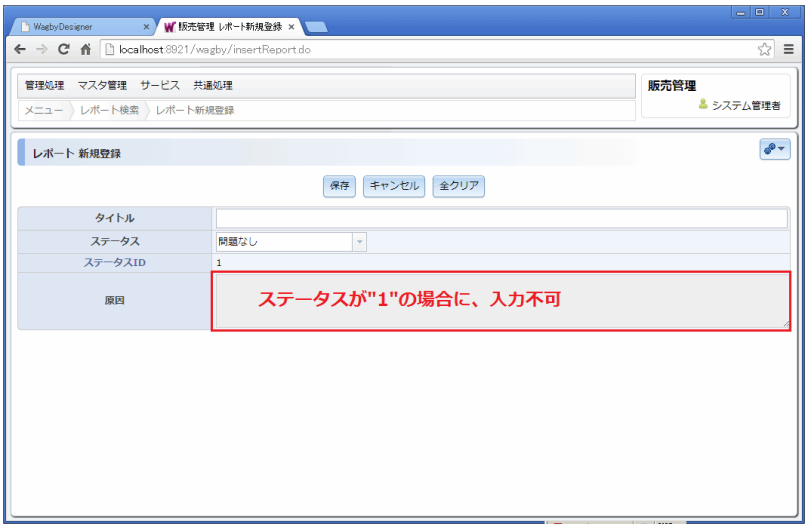


図 原因項目の入力が不可能になる

定義方法

「入力可・不可制御 > 他項目の値によって入力不可にする」の欄に条件式（項目名=値）を記述します。

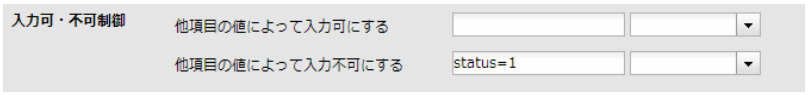


図 他項目の値によって入力不可にする設定

繰り返しコンテナ（8.3.0）

繰り返しコンテナ項目での入力可・不可制御は次の組み合わせで動作します。

制御元が通常項目、制御先が繰り返しコンテナ項目

繰り返しコンテナの列の項目の制御を行います。

図 制御元が通常項目、制御先が繰り返しコンテナ項目

制御元が繰り返しコンテナ項目、制御先も繰り返しコンテナ項目

繰り返しコンテナの行毎の項目の制御を行います。設定欄に繰り返しコンテナ項目を記述するときは「コンテナ名/項目名」とします。（例：cont/citem1）

図 制御元が繰り返しコンテナ項目、制御先も繰り返しコンテナ項目

仕様・制約

定義できない型

- 繰り返し項目
- 追記型リストボックス用いた場合
- 日付型（リストボックスを用いた場合）
- 一覧更新画面での、モデル参照(検索画面)

「他項目」として定義できない型
- 上記「定義できない型」の4パターン
- ファイル型
- モデル参照(検索画面)

条件式の記述
- 式は″項目名=値″という書式のみを受け付けます。関数を記述することはできません。またモデル参照項目では、ID部を指定してください。（上記例では ″トラブル″ という文字ではなく、″2″ というIDを使っていることを指します。）
- 条件式に複数の項目を記述することはできません。例えば gender=1,status=2 という指定を行うことはできません。

両方を同時に指定することはできません

「他項目の値によって入力可とする」「他項目の値によって入力不可とする」を両方、同時に指定することはできません。

異なるモデルで同じ項目名を用いることはできません

例として、外部キー連携により複数の子モデルを持つ親子モデルの同時更新画面を想定します。このとき、2つの子モデルで入力可・不可の制御元の項目が同じ項目IDをもっている場合は、動作しません。（2つの子モデルがメインモデルとサブモデルの関係で、同じ項目IDというケースが考えられます。）

(80)

読み込み専用項目（自動計算項目を含む）の扱い

「他項目」が読み込み専用項目の場合でも画面の初期表示時のみ「他項目の値による入力制御」が動作します。ただし表示後（式などによって）読込専用項目の値が変わっても入力可/不可の状態は変化しません。

(81)

この対応として、次のいずれかを行ってください。

- 入力可/不可の制御に関わる項目の ″値が変更された際に再読込する″ 設定を行う。画面再描画によって初期表示扱いとすることで入力制御を行えるようにする。
- 読み込み専用ではなく、隠し項目とする。

入力欄のスタイル指定

例

入力欄へのスタイル適用は、″入力欄のみ″ ″入力欄を囲う領域″ ″両方″ を指定することができます。

(80) 入力可・不可の制御は「項目ID」のみで判断しており、「モデルID」は含まれていません。そのため、このようなケースではどのモデルの項目かを判定できません。

(81) 本機能は「他項目」の値が変更された際に発生する JavaScript の change や blur イベントを利用しているためです。読込専用項目ではこれらのイベントが発生しないため、値が変更された場合に入力可/不可の状態を変化させることができません。

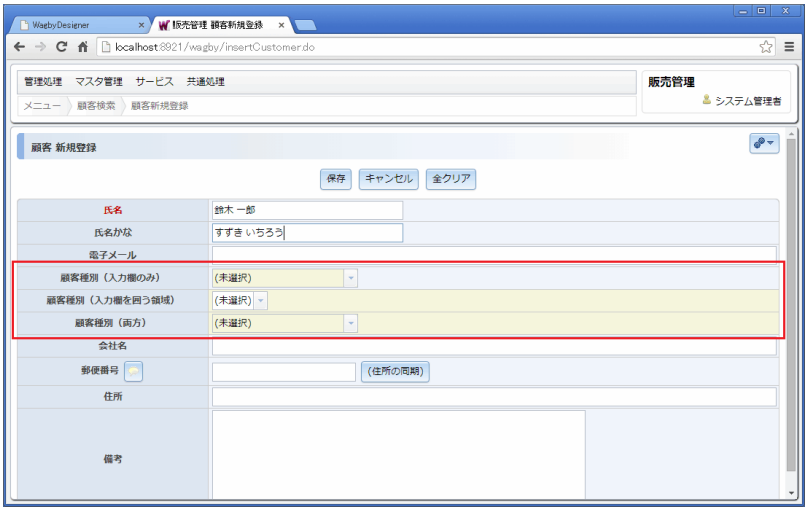


図 入力欄へのスタイル適用

″入力欄″で実際に入力するタイミングでは、背景は白色となります。カーソルが移動すると、指定した背景色に変わります。

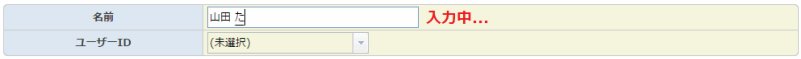


図 入力時は背景は白色になる

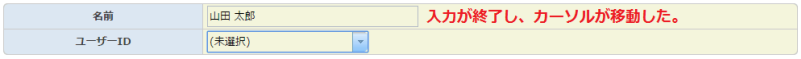


図 入力が終了すると再び背景色がセットされる

定義方法

「モデル項目詳細定義>入力制御>スタイル」を指定します。図では全画面共通に設定していますが、画面毎に個別に指定することもできます。

入力欄にはCSSを直接、記述します。右側に用意された選択肢で、適用箇所を指定することができます。空白時は″入力欄のみ″となります。

```
background:beige
```

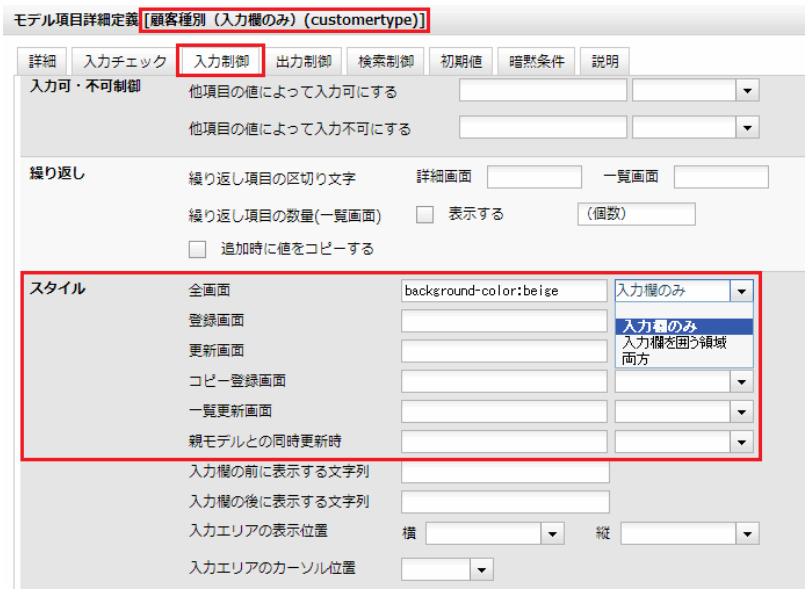


図 スタイルの適用設定

図にあるように、CSSでは色名を直接、指定することもできます。

例 リストボックスの幅を指定する

登録・更新画面でリストボックスの幅を指定する場合は、width 指定を行います。単位はピクセル(px)や1文字分の高さ(em)を持ちいることができます。

```
width:50px

width:9em
```

適用箇所は空白とします。（空白時は、"入力欄のみ" で扱われます。）

具体的なサンプルは"入力制御 > 入力フィールド長 > リストボックスの場合"をお読みください。

日付型（カレンダ入力）

この設定は日付型（カレンダ入力）にも適用できます。 具体的なサンプルは"入力制御 > 入力フィールド長 > 日付型（カレンダ利用）の場合"をお読みください。

例 ラジオボタン・チェックボックスの高さを指定する

ラジオボタンやチェックボックスを格納する div 要素へのスタイルを指定することができます。図では、高さを150pxとし、スクロールバーを用意する設定を行なった例です。

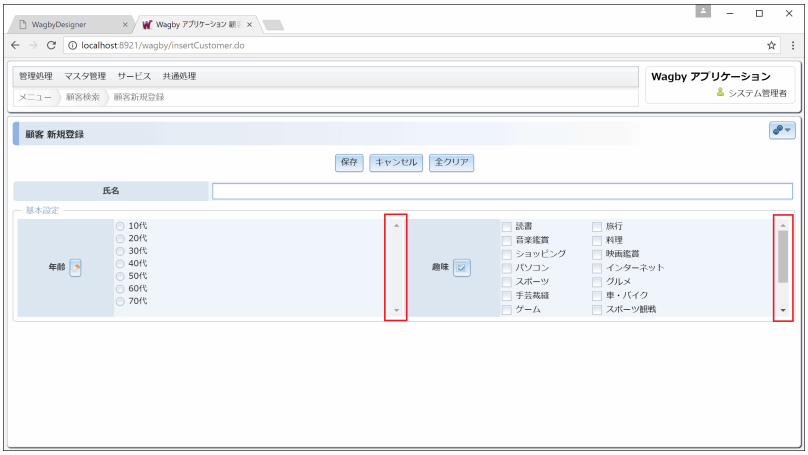


図 div要素へのスタイル指定の例

定義方法

モデル項目の「入力制御 > スタイル」に次のように div{..} で囲って記述します。

```
div{height:150px;overflow-y:scroll;}
```

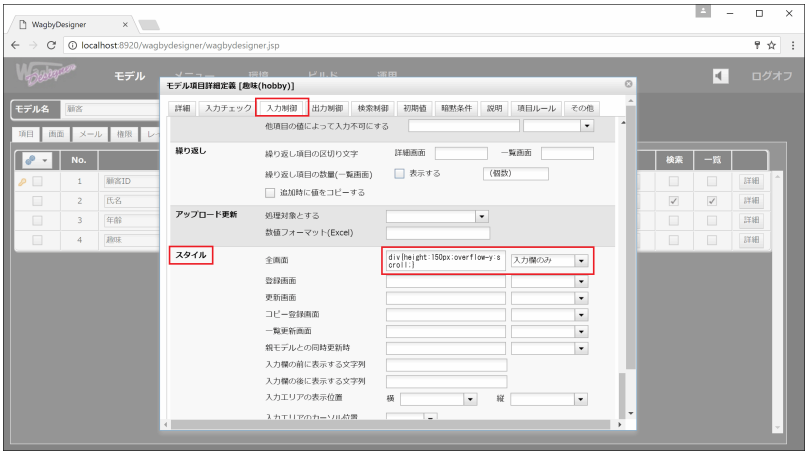


図 スタイルの指定

チェックボックス・ラジオボタンの折り返し数

図の例では、チェックボックス・ラジオボタンの折り返し数も指定しています。

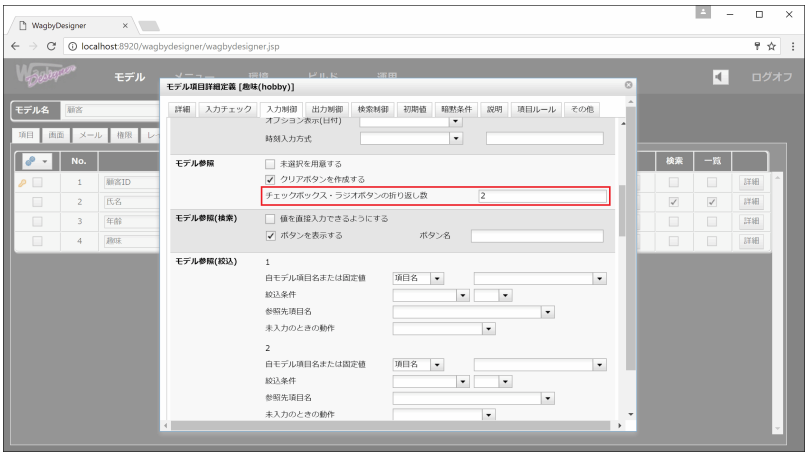


図 チェックボックス・ラジオボタンの折り返し数

レイアウト

図の例では、二つの項目を横並びレイアウトで表示するようにしています。

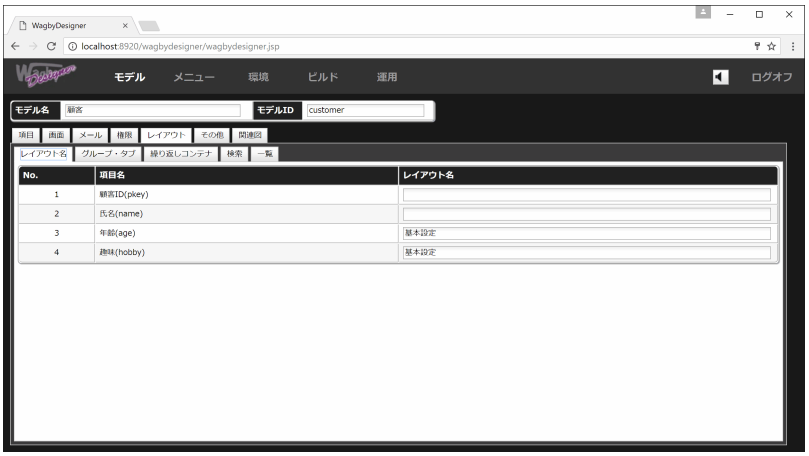


図 レイアウトの設定

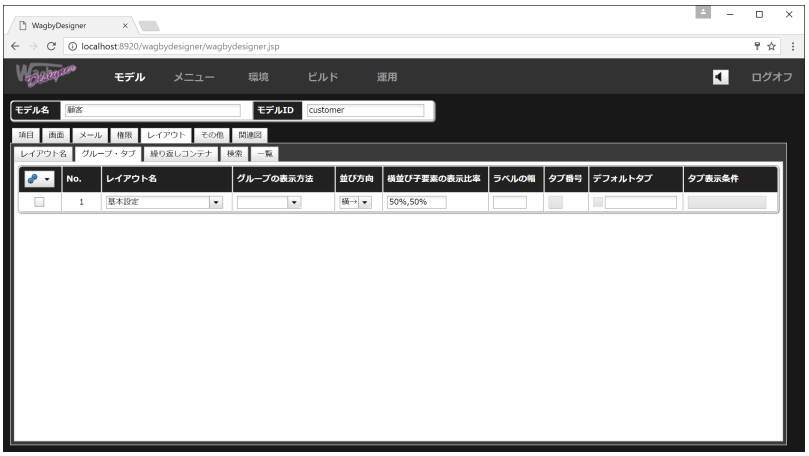


図 レイアウトの並び設定

他の設定との併用

div {...} 設定のあとに、続けて色の指定など、他のスタイルを指定することができます。div {...} で囲まれていないスタイルは、チェックボックス本体に適用されます。

(82)

次の例は文字色を赤とした例です。

```
div{height:150px;overflow-y:scroll;} color:red
```

(82) 正確には、チェックボックスを囲う span 要素にスタイルが適用されます。

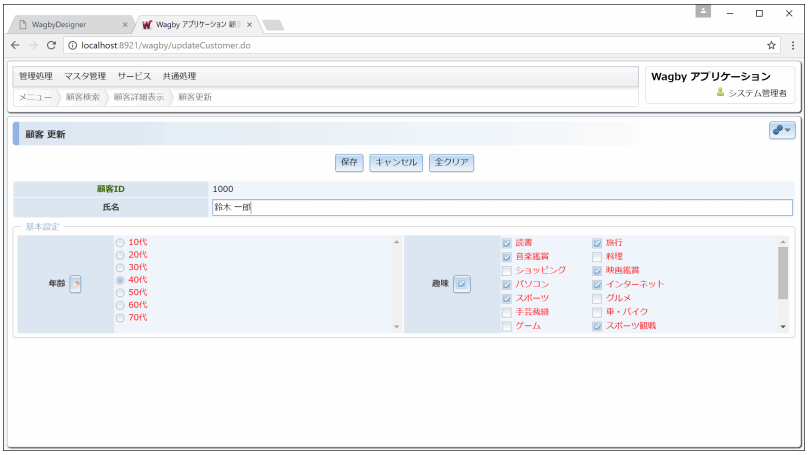


図 div要素とspan要素の両方を設定する

詳細画面への設定

"出力制御（スタイル）> CSSクラス名・スタイルの指定"をお読みください。

式を適用する

CSS記入欄に式を適用することもできます。Wagbyが用意する関数を利用できます。

```
IF(${age}>30,"background:beige","")
```

式を用いる場合の注意点は次のとおりです。

- CSS部分はダブルクォーテーションで囲ってください。上の例のように、条件が成立しなかった場合は空文字列（""）を返すようにします。
- 入力画面へ適用する場合、トリガーとなる項目について「値変更時に画面を再描画する」設定を有効にするとよいでしょう。上の例では、age 項目に適用します。これにより age 項目の値を変更すると画面が再描画され、CSS が適用されるようになります。

入力欄の前後に説明文を付与する

例

気温を入力する欄を用意します。その前に "およそ" を、その後に "℃" を表示するようにしました。

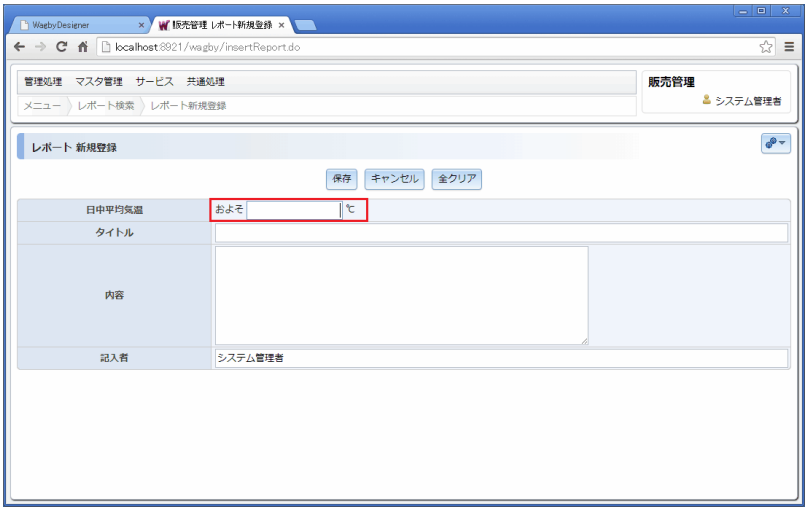


図 入力欄の前後に文字を表示する

この指定は表示画面でも有効です。

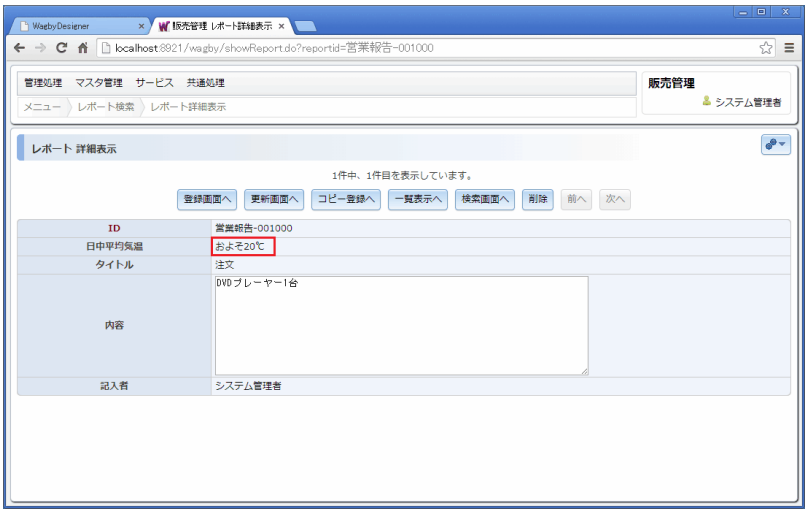


図 登録後の表示画面

定義方法

「モデル項目詳細定義>入力制御>スタイル」内に用意された「入力欄の前に表示する文字列」「入力欄の後に表示する文字列」をそれぞれ指定します。

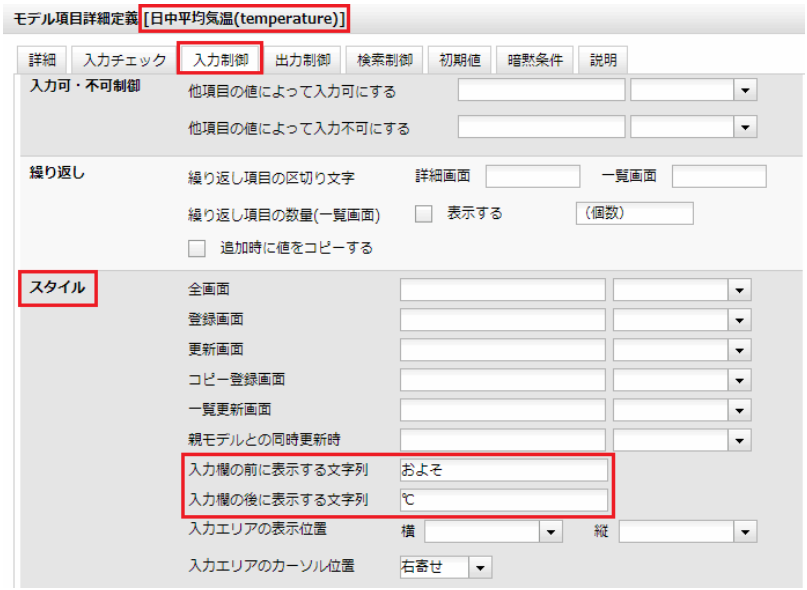


図 入力欄の前（後）に表示する文字列の設定

入力フィールド長を設定する

本設定を行う入力欄には、同時に「入力フィールド長」も指定してください。図では　"10"　としています。[83]

[83] 今回の例ではさらに、数字入力のため入力エリアのカーソル位置を"右寄せ"にしています。

図 入力フィールド長の指定

テキストエリアでの利用

入力フィールドがテキストエリアの場合も利用できます。行数ならびに1行の文字数を指定し、入力欄の前後の文字列が含まれるように調整します。

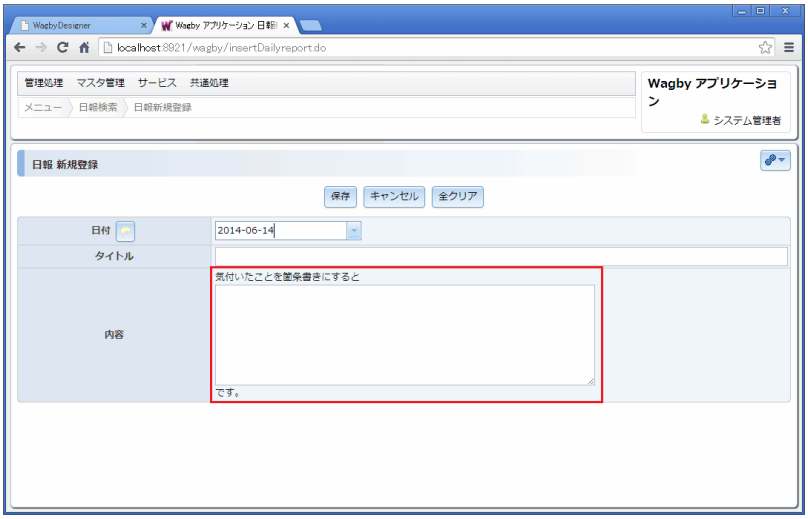


図 テキストエリアでの利用例

HTMLを含めることもできます。

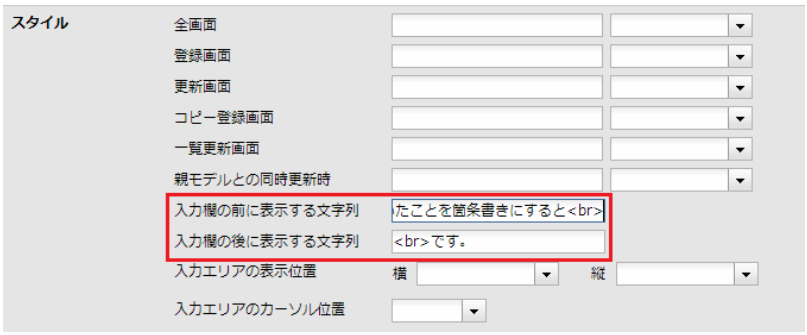


図 テキストエリアの指定例

仕様・制約

- この設定を繰り返し項目で利用することはできません。
- この設定を行なった項目を検索条件（範囲検索）で用いる場合、範囲検索条件入力欄のレイアウトを〝左寄せ〟にしてください。

入力欄の表示位置の制御

横位置の設定

画面例

図は入力欄の横の位置を中央（センタリング）とした例です。

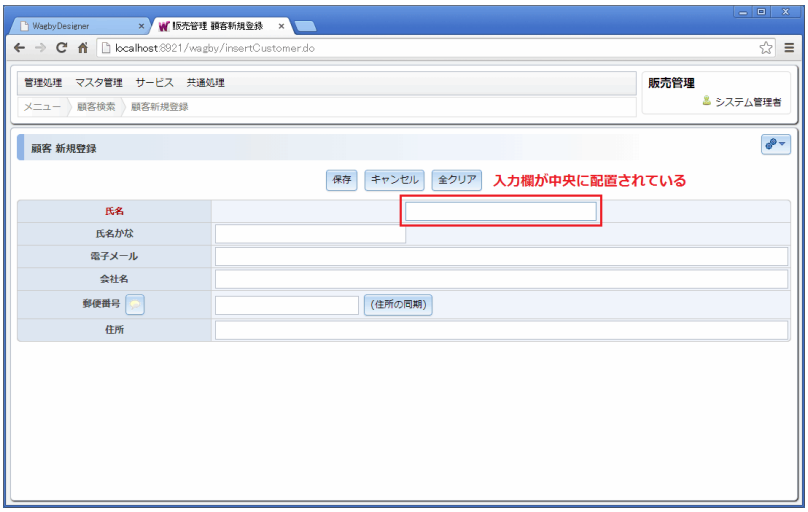


図 入力欄の横の表示位置を中央と指定した例

定義方法

モデル項目詳細定義の「入力制御>スタイル>入力エリアの表示位置　横」を指定します。″左寄せ″，″センタリング″，″右寄せ″から選択できます。空白時は″左寄せ″になります。

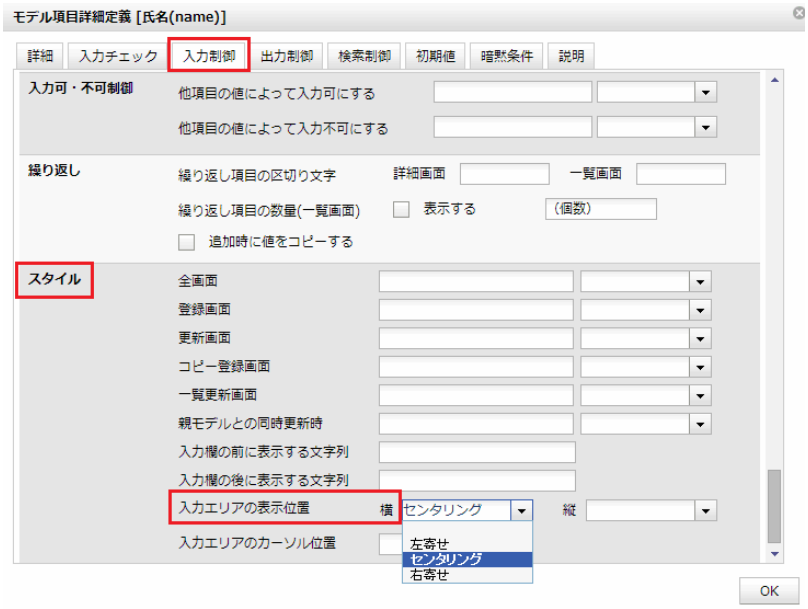


図 入力エリアの表示位置（横）

縦位置の設定

画面例

図は入力欄の縦の位置を中央（センタリング）とした例です。

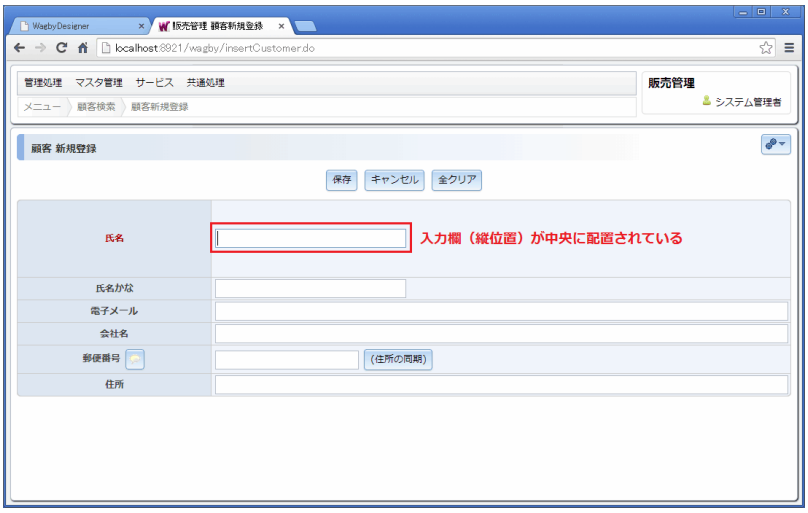


図 入力欄の縦の表示位置を中央と指定した例

定義方法

ここでは視覚的にわかりやすくするため、入力欄の高さを100ピクセルとしています。

モデル項目詳細定義の「入力制御>スタイル>入力エリアの表示位置　縦」を指定します。″上寄せ″, ″センタリング″, ″下寄せ″から選択できます。空白時は ″センタリング″ になります。

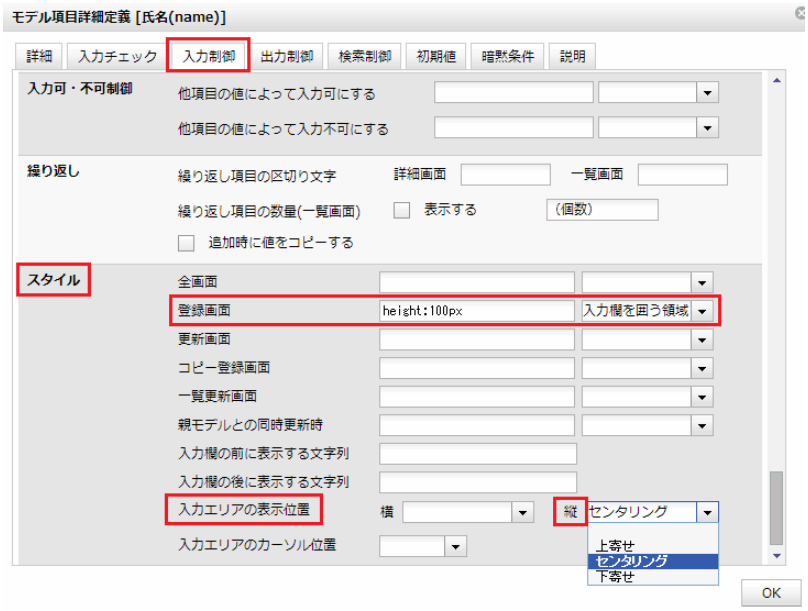


図 入力エリアの表示位置（縦）

カーソル位置の設定

画面例

図は入力欄のカーソル位置を右寄せとした例です。

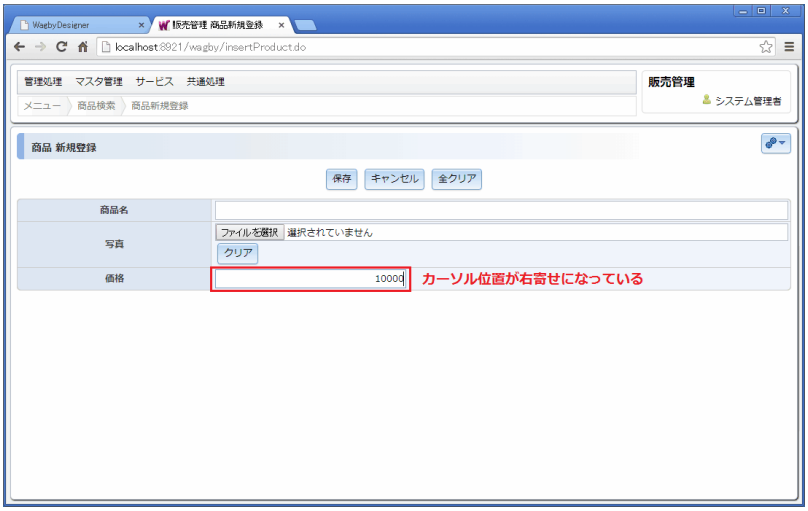


図 入力欄のカーソル位置を右寄せとした例

定義方法

モデル項目詳細定義の「入力制御>スタイル>入力エリアのカーソル位置」を指定します。″左寄せ″, ″右寄せ″から選択できます。空白時は ″左寄せ″ になります。

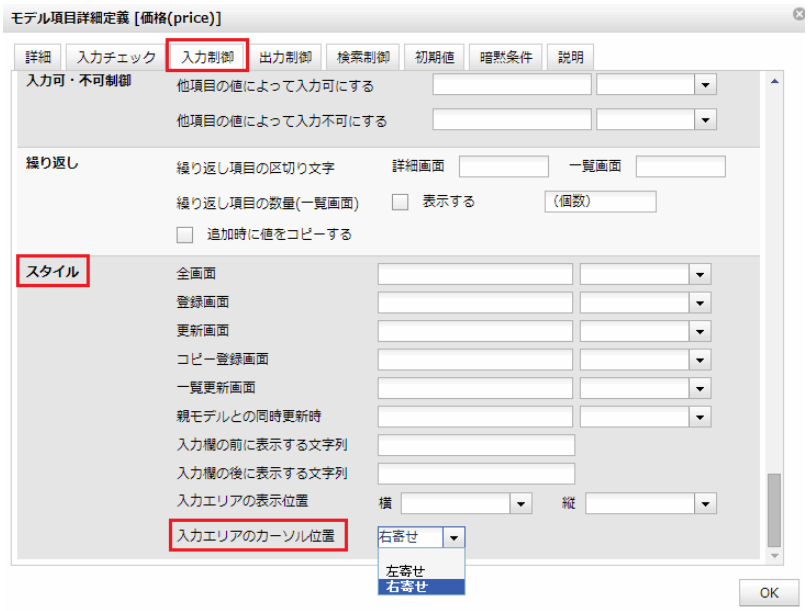


図 入力エリアのカーソル位置を右寄せと指定した例

手書き入力

実行例

手書き入力機能を有効にすると図のような入力欄が用意されます。

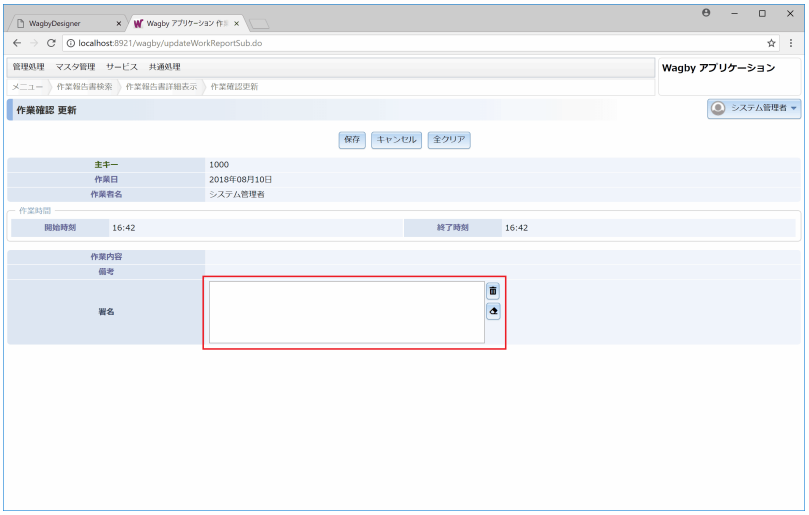


図　手書き入力欄

マウス、もしくはタブレットで直接、指を使って描くことができます。

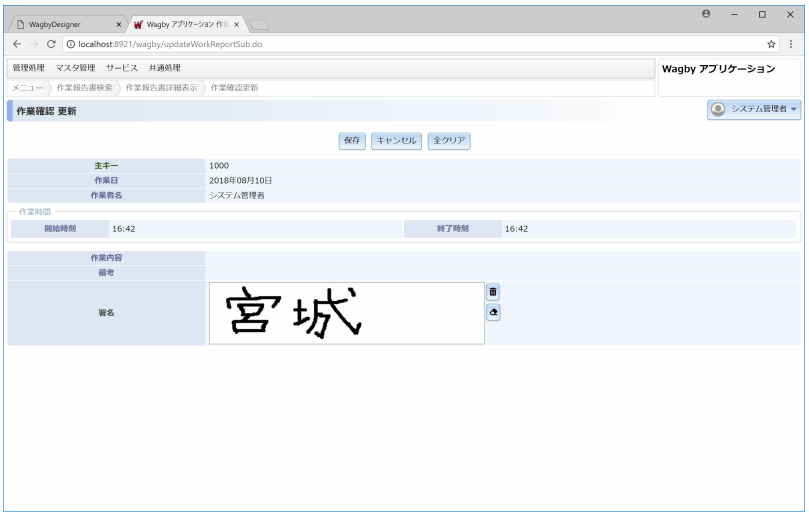


図　入力途中の例

「ゴミ箱アイコン」のボタンを押下すると、これまで描いた署名がクリアされます。　「消しゴムアイコン」を押すと消去モードに移ることができます。この消しゴムアイコンはトグル式になっており、押下後にマウスの左ボタンを押下したまま線をなぞると消去されます。再度手書きを行なう場合は切り替わった鉛筆アイコンを押下します。

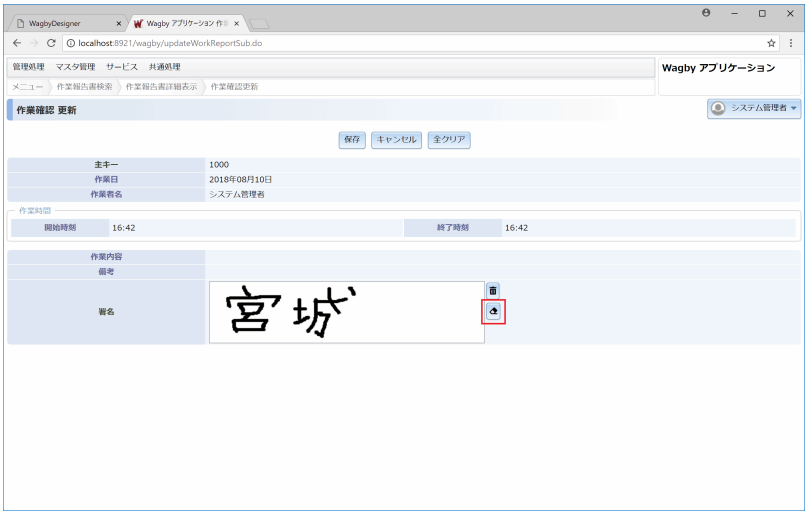


図 消しゴムを使って一部を消去した例

保存すると詳細画面でも確認できます。

図 詳細画面

定義方法

モデル項目詳細ダイアログを開きます。入力制御 > 入力項目の種類で ″手書き″ を選択します。

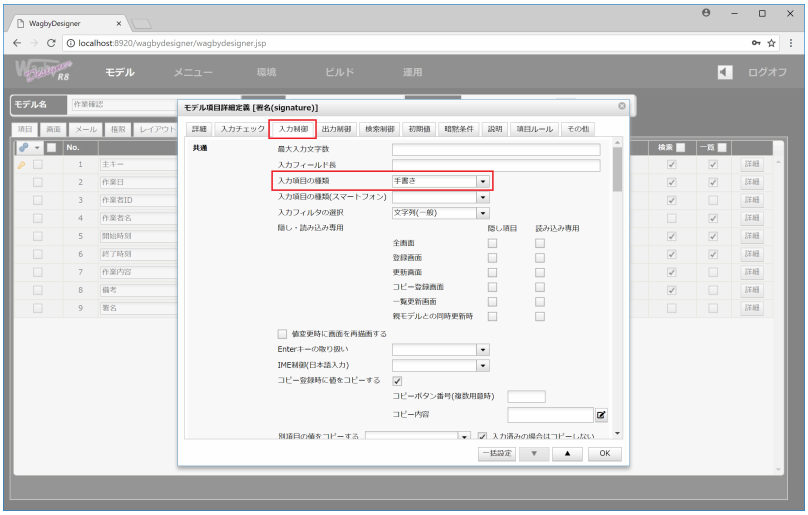


図 入力項目の種類を手書きとする

出力制御 > 詳細画面 > 手書きの設定で、幅と高さを指定することができます。

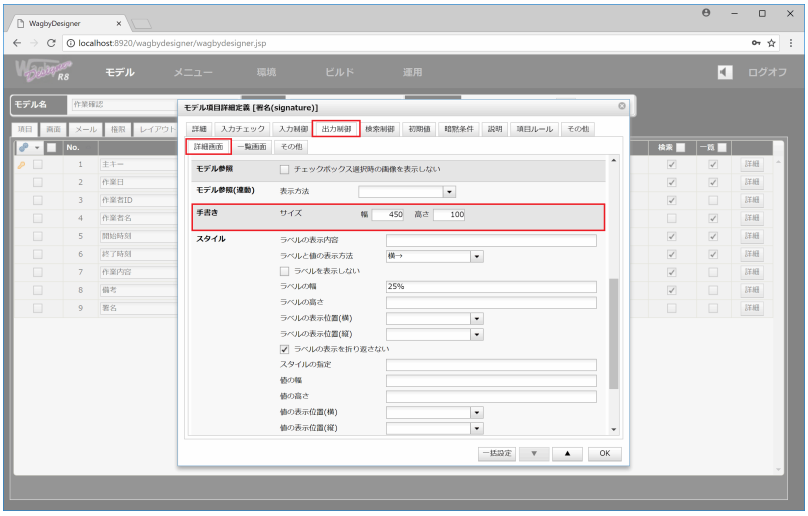


図 幅と高さを指定する

署名画像の内部での扱い

画像データは base64 エンコードされたテキスト情報としてデータベースに格納されています。そのため、この設定を行った文字列項目はデータベースの型が CLOB となっています。

JasperReportに署名画像を埋め込む

この署名画像を JasperReports のイメージとして利用することができます。

Jaspersoft Studio で base64 のテキストデータを表示させる場合は Image のプロパティの「Expression」に以下の設定を行なってください。

```
new
 ByteArrayInputStream(org.apache.commons.codec.binary.Base64.decodeBase64($F{signature}.getBytes()))
```

$F{signature} は、手書き項目のテキストデータを指します。

仕様・制約

- 手書き入力を設定する項目は文字列型としてください。
- この設定を行った項目は、検索条件に指定することはできません。
- 一覧更新画面、親子モデル同時更新画面には対応していません。
- Excel出力で画像を埋め込むことには対応していません。

電話番号入力時、自動的にハイフン区切りを行う

実行例

ハイフン区切りなしで電話番号を入力し、カーソルを（次の入力フィールドに）移動すると、そのタイミングで自動的にハイフンを設定します。

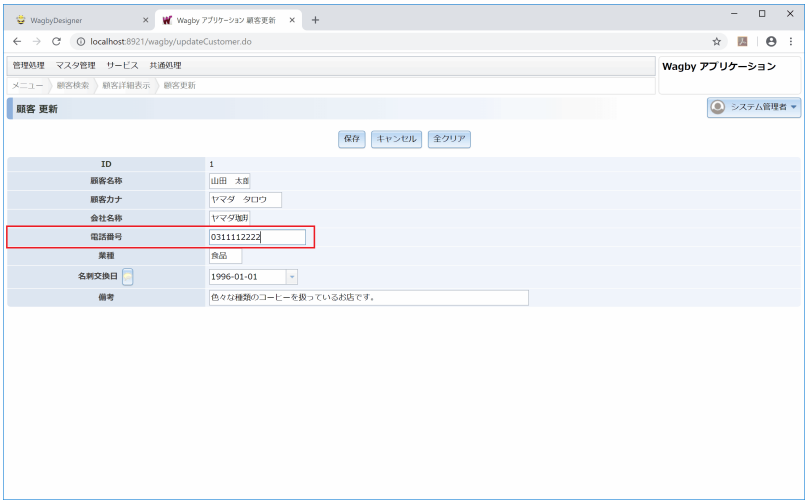


図 電話番号の入力

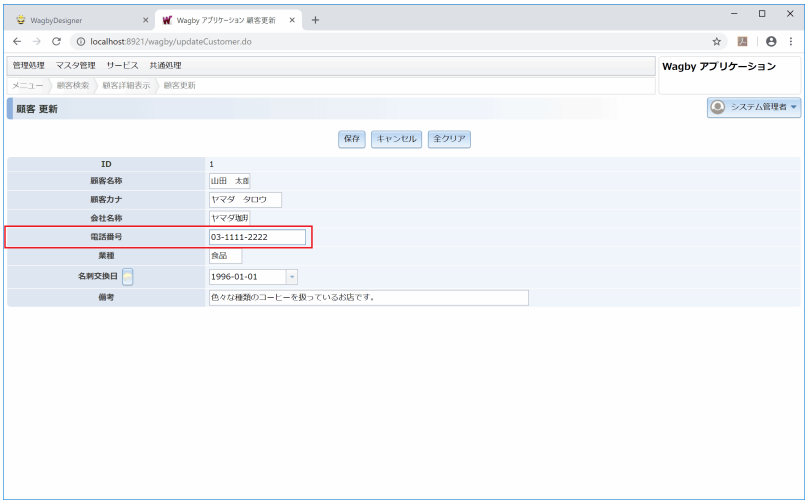


図 ハイフンがセットされた

定義方法

上の例では、"電話番号"項目についてのモデル項目詳細ダイアログを開きます。入力制御 ＞ 入力支援で"電話番号をハイフン区切りにする" 設定を有効にします。

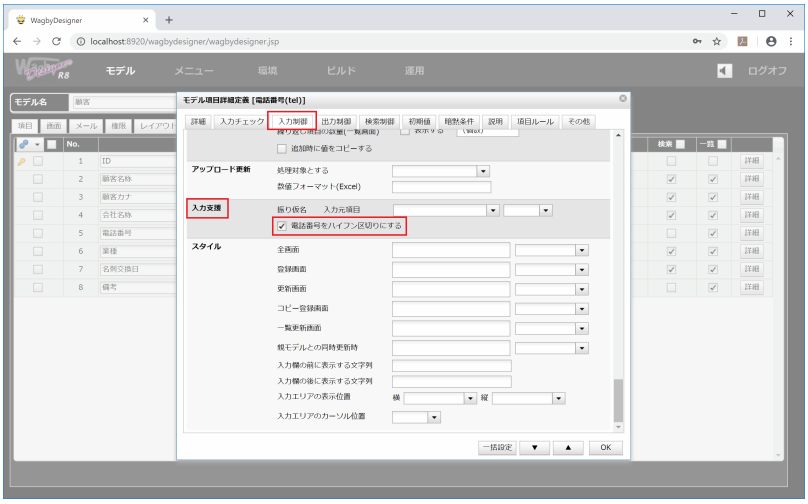


図 電話番号をハイフン区切りにする

仕様・制約

制約

・ 繰り返し項目には対応していません。 R8.3.0 より、繰り返し項目でも本機能が動作するようになりました。

謝辞

この機能は libphonenumber という JavaScript ライブラリを用いて実現しています。

振り仮名の自動入力

実行例

"顧客名称"項目に入力を行うと、自動的に"顧客カナ"項目にカナが入力されます。

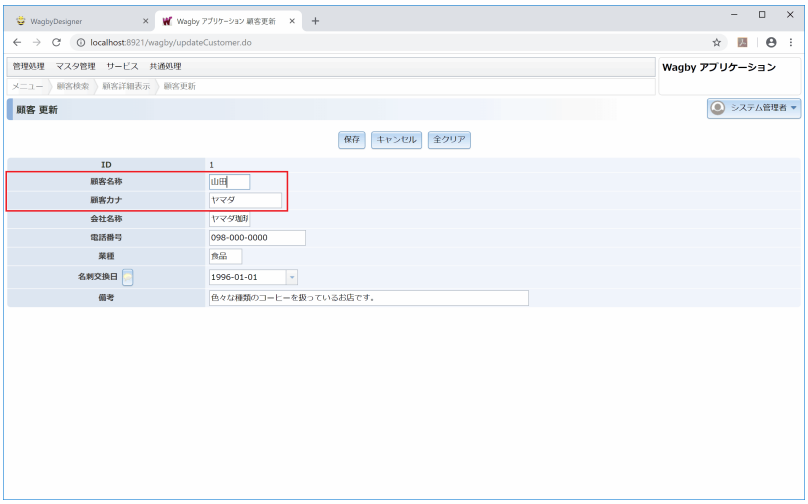


図 カナが自動で入力される(1)

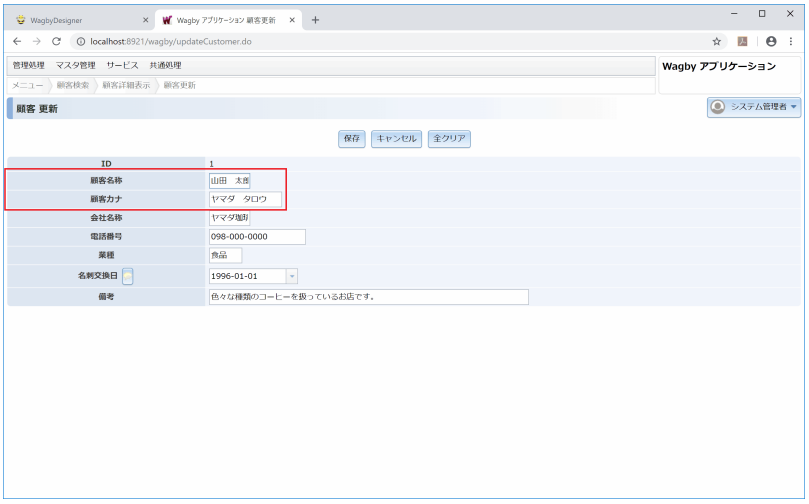


図 カナが自動で入力される(2)

定義方法

上の例では、"顧客カナ"項目についてのモデル項目詳細ダイアログを開きます。入力制御 > 入力支援で、振り仮名の "入力元項目" を指定します。また "カタカナ" か "ひらがな" を選択します。

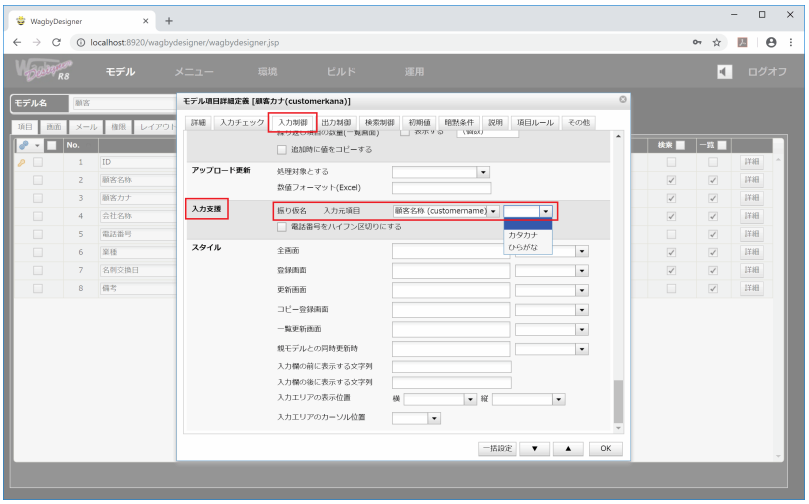


図 振り仮名の設定

仕様・制約

- この機能は vanilla-autokana という JavaScript ライブラリを用いて実現しています。
- ログオンしたときの言語が日本語でのみ利用できる機能です。

入力フォーカス

実行例

標準では画面上から辿って最初に見つかったテキスト入力欄に、入力フォーカスがセットされます。

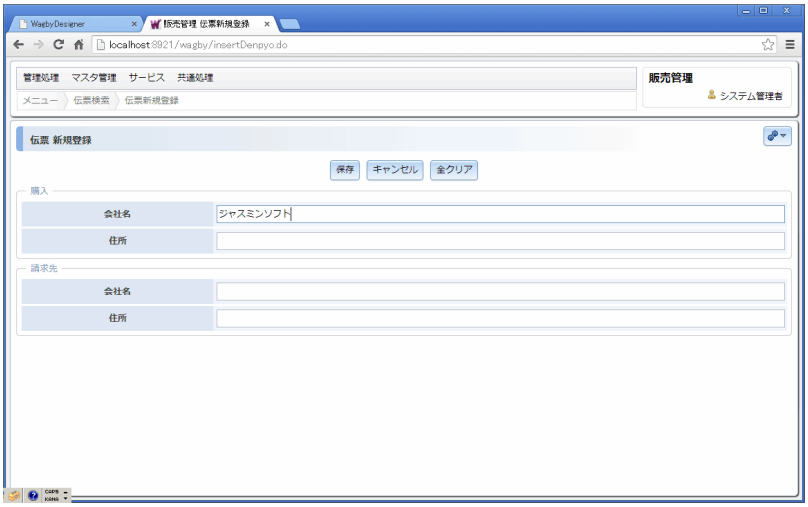


図 登録画面の入力フォーカス

フォーカスのカスタマイズ

開発者は JavaScript ファイルをカスタマイズすることで、フォーカスのセットを無効化することができます。また、任意の位置にフォーカスをセットすることもできます。

この詳細は レイアウト > JavaScript のカスタマイズ > 入力フォーカスの制御 をお読みください。

バーコード読み取り

実行例

スマートフォン用画面の登録、更新画面でカメラアイコンが表示されます。カメラを使ってバーコードを読み込むことができます。

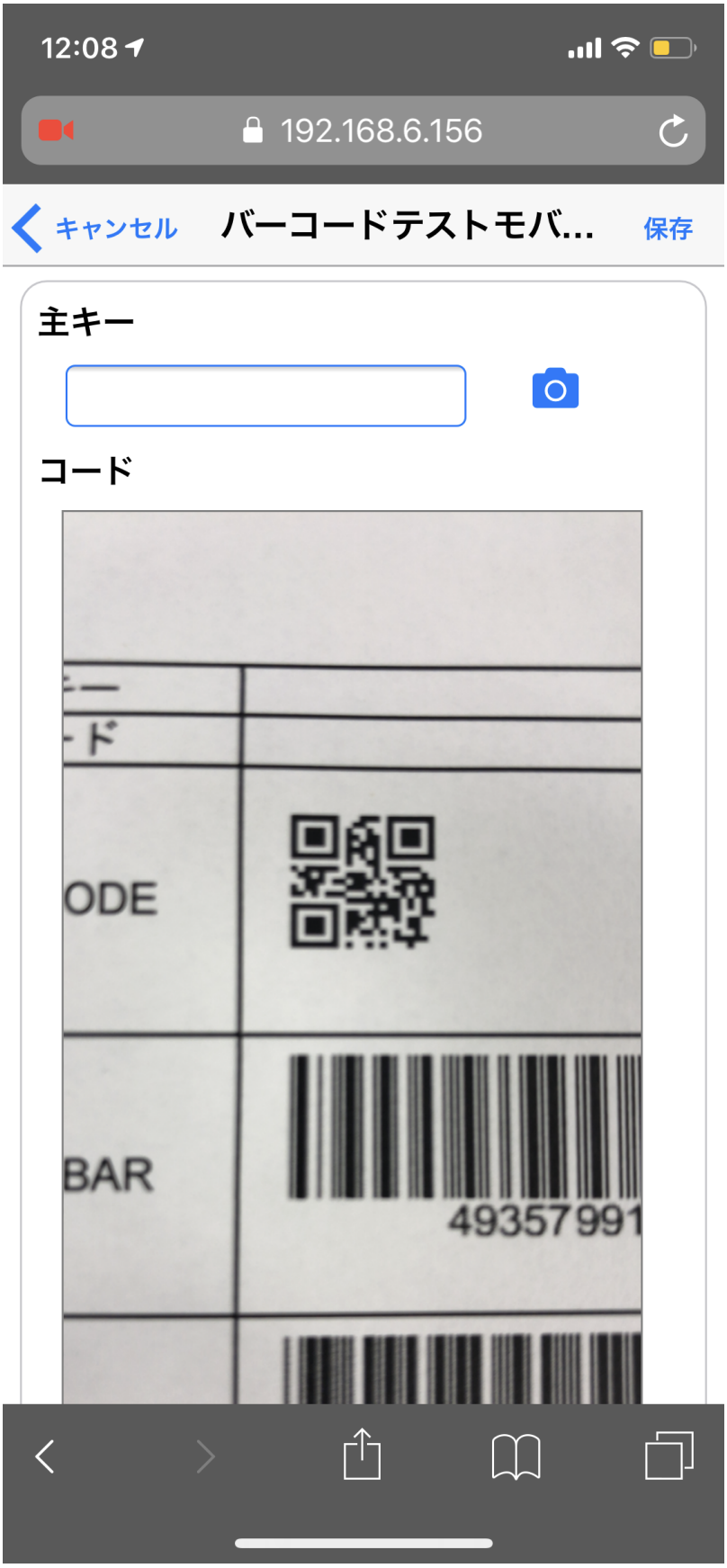


図 カメラアイコンが用意される

スマートフォン用画面の作成は、サブモデルと組み合わせるとよいでしょう。詳細はスマートフォンの説明をお読みください。

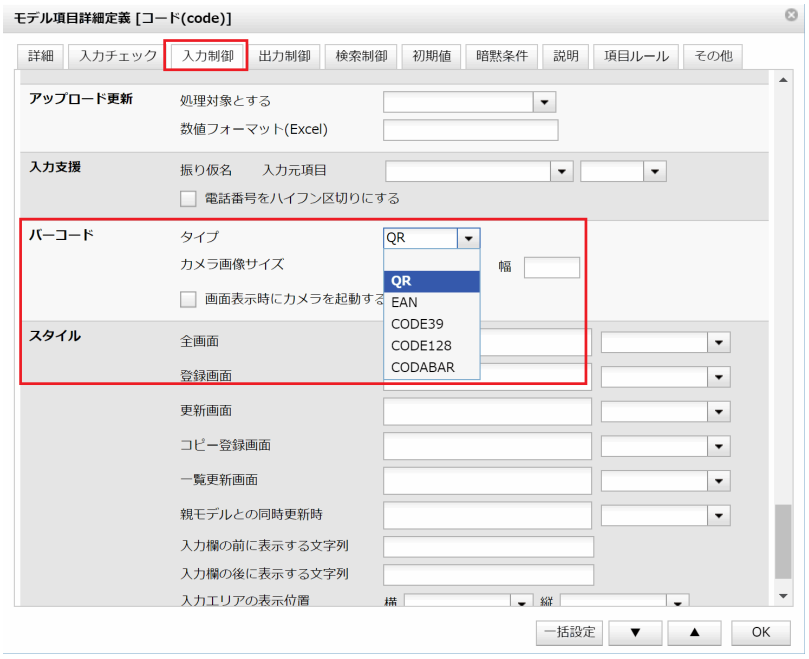
定義方法

モデル項目詳細ダイアログの「入力制御>共通>入力項目の種類」から「バーコード」を選択します。

図 入力項目の種類

「入力制御>バーコード」設定欄の「タイプ」でバーコードのタイプを選択します。1次元バーコードはEAN, CODE39, CODE93, CODE128, CODABAR, UPC, 2of5に対応します。2次元バーコードはQR, EAN, CODE39, CODE128, CODABARに対応します。

検索制御にも対応するため、検索条件入力で読み込んだバーコードでの検索が可能です。「検索制御>バーコード」の「バーコード入力を有効にする」を有効にしてください。



仕様・制約

- この機能はWebRTC技術を使います。そのためスマートフォン(iOSやAndroid)でWebRTCに対応したバージョンを使う必要があります。iOSは11以降になります。
- WebRTCのセキュリティ制約により、動作環境はhttps接続に限定されます。
- バーコード入力項目と、バーコード出力項目（BARCODE関数を適用する項目）を同一にすることはできません。それぞれ用意するようにしてください。

関連するページ

- 出力制御（スタイル）＞バーコード出力

[事例] 百万円単位での入力欄を用意する

例

プロジェクトモデルの金額項目は、入力・出力時に百万円単位の数値を扱います。しかしデータベースには ″1,000,000″ という数値が格納されるようになっています。

図 登録画面

図 表示画面

検索画面では、丸めた数値ではなく、正確な数値で検索できます。

図 検索画面

定義方法

例:プロジェクトモデル

上の例で示したプロジェクトモデルの定義を図に示します。金額項目は「入力・出力用」用と「検索」用の二つを用意しています。

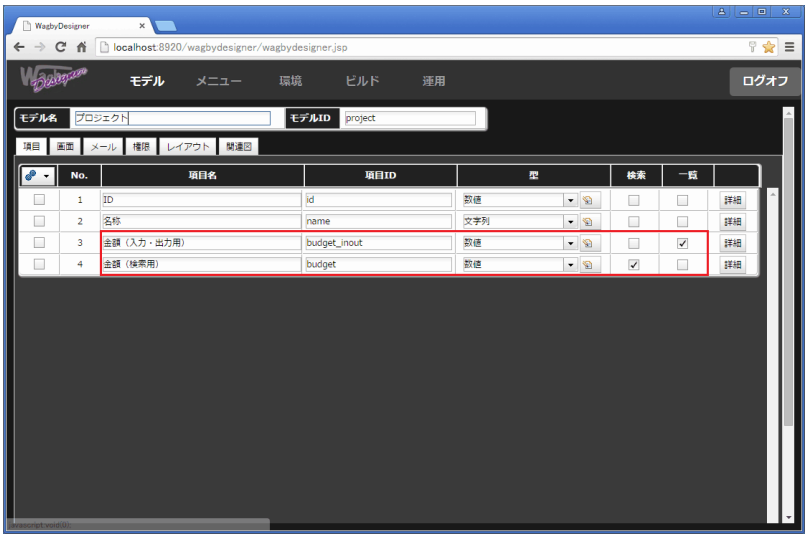


図 プロジェクトモデルの定義

金額（入力・出力用）

この項目は入力と出力で用いますが、検索では用いないとします。　はじめに、単位表示のために「入力フィールド」を設定し、入力欄の右側に余白を用意します。

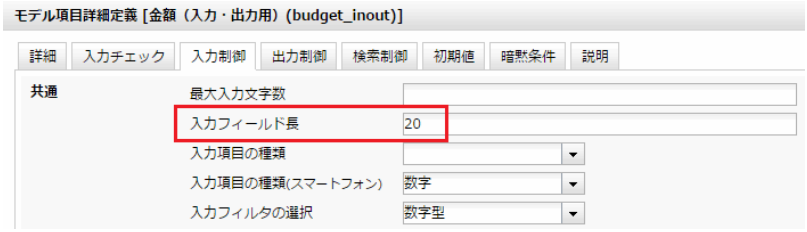


図 入力フィールド長の指定

「入力制御 > スタイル > 入力欄の後に表示する文字列」に、単位を指定します。
あわせて今回は「入力エリアのカーソル位置」を「右寄せ」とします。

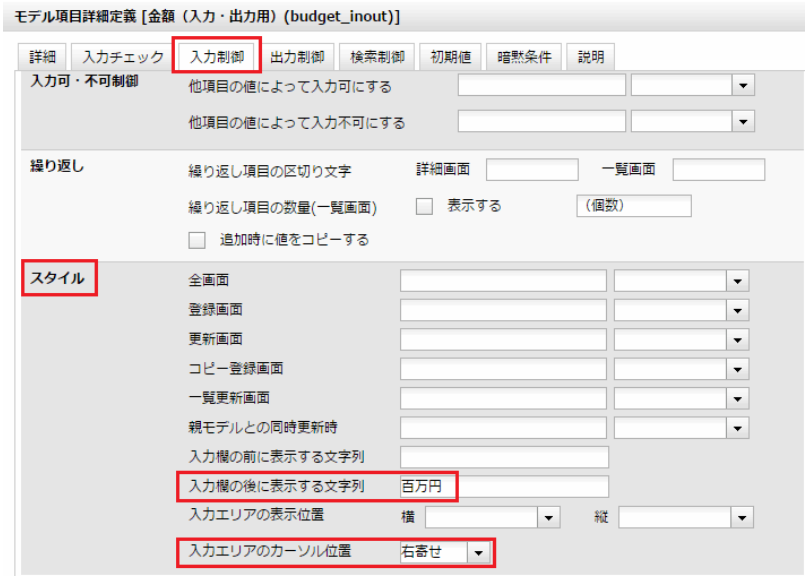


図 入力欄の後に表示する文字列とカーソル位置の設定

「出力制御 > 詳細画面 > 数値」のフォーマットを3桁区切りとします。

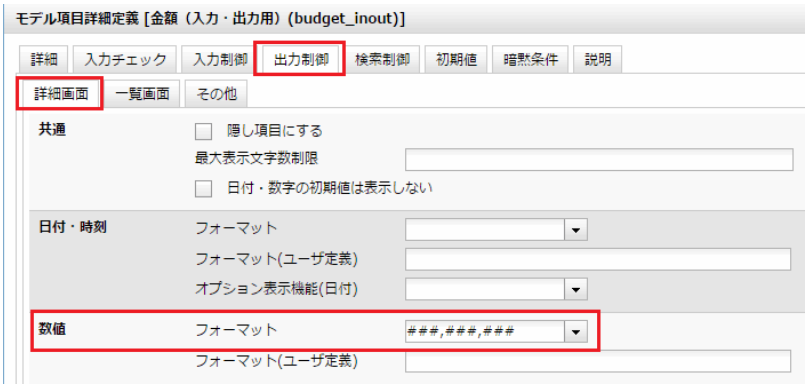


図 数値フォーマットの指定

「出力制御 > 詳細画面 > スタイル > ラベルの表示内容」を〝金額〟とします。画面に表示される項目名を〝金額〟に統一する意図です。（後述する、検索画面でも同様の対応を行います。）

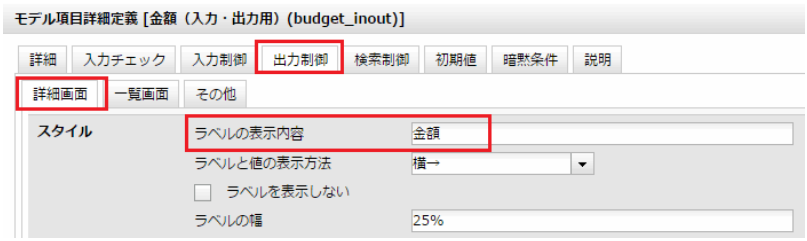


図 ラベルの表示内容の指定

金額（検索用）

この項目は検索用途です。入力された budget_input 項目の値に 1000000 倍した数値をデータベースに保存します。

図 計算式の設定

「入力制御 ＞ 隠し項目」の設定を行います。

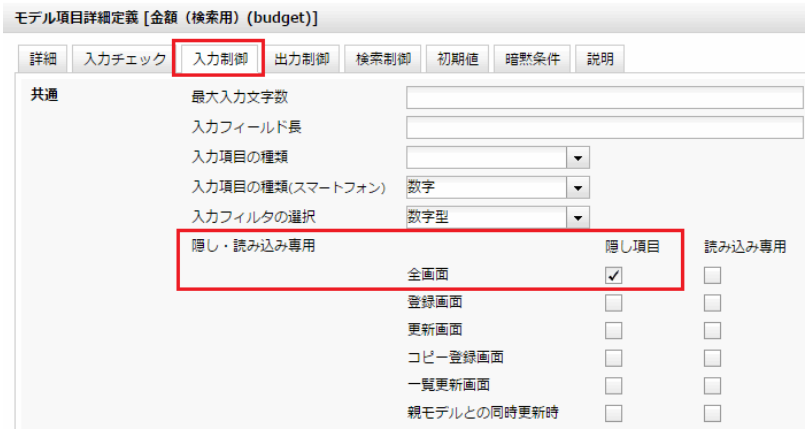


図 隠し項目の設定（入力）

同じように「出力制御 ＞ 詳細画面 ＞ 隠し項目」の設定を行います。これによってこの項目は入力、出力ともに画面には表示されません。

あわせて出力フォーマットの設定も行います。

図 隠し項目の設定（出力）と、出力フォーマットの設定

「検索制御 ＞ スタイル ＞ ラベルの表示内容」を〝金額〟とします。

図 検索画面のラベル表示内容の設定

計算式

基本的な書き方

定義方法

モデル項目詳細定義ダイアログの「詳細 > 自動計算」の欄に、計算式を記述します。

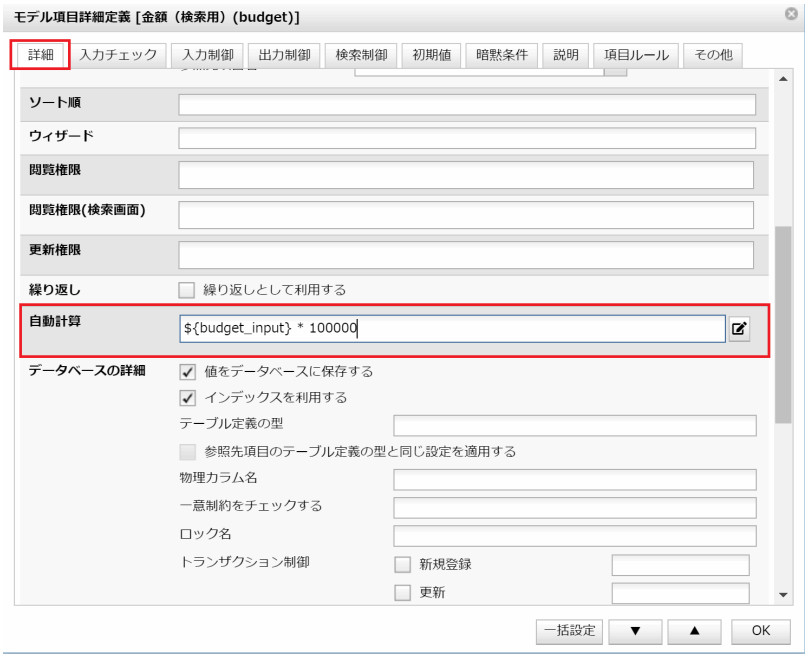


図 計算式の記述例

三項演算子

計算式に三項演算子 "?:" を含めることができます。

```
( 式 ) ? 式が真の場合の値 : 式が偽の場合の値
```

書式	説明
式 ? x : y	式が真なら x を返します。式が偽なら y を返します。x, y も式という場合もあります。

例：三項演算子を使って自動計算の結果に null を設定する

次の例は、項目 item1 の値が 0 より大きければ 1 をセットします。そうでなければ null をセットします。

```
${item1} > 0 ? 1 : null
```

三項演算子を徐算で用いる

ゼロで徐算することはできません。これを回避するために三項演算子を使うことができます。

```
( (${bunbo}!=0) ? ${bunsi}/${bunbo} : 0 )
```

これは項目 bunbo（ここでは整数型とします）の値が 0 なら "0" を、そうでなければ bunsi/bunbo の計算結果を格納するという意味になります。 なお、数値が等しくないという判定は「!=」を使います。

上の式において、項目 bunsi や bunbo は整数型だが、計算結果は小数点を格納する場合は、次のように(double)を項目IDの前に付与する強制的な型指定を行うことができます。

```
( (${bunbo}!=0) ? (double)${bunsi}/(double)${bunbo} : 0.0 )
```

初期値との関係

初期値を設定するタイミングは、自動計算の「後」になります。 そのため、初期値の値を自動計算の条件判定に使うような記述を行うことはできません。

繰り返し項目との関係

繰り返し項目は、内部では「配列」形式でデータを格納しています。 従って、配列を引数に受け取り、配列を返すような関数を使って、繰り返し項目の値を、別の繰り返し項目へセットすることができます。

例：繰り返し項目 toAddress から、メールアドレス部のみを抜き取って、同じく繰り返し項目 toEmailAddress へセットする場合、toEmailAddress へ次の式を記載します。

```
MAILADDRESS(${toAddress})
```

範囲検索項目との関係

検索条件の初期値に式を記述するとき、対象となる項目が範囲検索に対応している場合は次のルールがあります。

範囲検索の場合、内部では上限と下限を表す二つの項目が用意され、それぞれ、元の項目名に "1jshparam" "2jshparam" が付与されます。例えば、整数型の項目 num については、num1jshparam と num2jshparam が内部で用意されます。

例：検索条件項目 item1 の初期値として、同じ検索条件項目である num の上限値に 1 を加えた値を設定する場合は次のように記述します。

```
${num1jshparam}+1
```

親子モデルとの関係

親子モデル関係において、子モデルの情報が親モデル側に反映されます。

例：親モデル側の項目で、子モデルの数をかぞえるための式 COUNT(${child_lp}) を用意したとき、子モデルの登録や削除時に、親モデルの該当項目も変更されます。

モデル参照項目の扱い

式中にモデル参照項目を使うことができます。計算式 > モデル参照をお読みください。

式を設定する項目の型がモデル参照項目で、参照しているモデルが複合主キーの場合はスクリプトを利用してください。

スクリプトを利用する

計算式の設定欄に代えて、スクリプトを利用することもできます。スクリプトの「ヘルパ > 計算」のタイミングで値をセットしてください。

次のケースではスクリプトを利用します。

・　複雑な条件を記述する必要がある。
・　式を設定する項目の型がモデル参照項目で、参照しているモデルが複合主キーの場合。

四則演算式

Wagby がサポートしている四則演算式は次のとおりです。

演算子	説明
( )	括弧
*	乗算
/	徐算（注意点...）
+	加算
-	減算または符号反転
%	余り

自動計算項目は計算により値が一意に決まるため、登録・更新画面では読み取り専用項目となります。

関数

四則演算に加え、Wagbyが提供するさまざまな関数を利用することができます。関数の一覧はこちらをお読みください。

モデル項目の指定

式中に（同一モデル内の）項目を指定することができます。項目名の指定は「${...}」と記述します。（... の部分に英語項目名を指定します。）

```
${...}
```

以下に例を示します。

モデル項目 value1 に 1 を加算した値を求める。

```
${value1} + 1
```

モデル項目 kosuu とモデル項目 tanka の積に1.05を乗じた値を求める。

```
${kosuu} * ${tanka} * 1.05
```

条件判定式

条件判定式の書き方は、対象となる項目の「型」によって異なります。

型	数値型	文字列型	日付型 (*3)
一致	${v1}==${v2}	EXACT(${v1},${v2})	DATE(${v1})==DATE(${v2}) またはEXACT(${v1},${v2})
不一致	${v1}!=${v2}	!EXACT(${v1},${v2})	DATE(${v1})!=DATE(${v2})または!EXACT(${v1},${v2})
より大きい	${v1}>${v2}	-	DATE(${v1})>DATE(${v2})
より小さい	${v1}<${v2}	-	DATE(${v1})<DATE(${v2})
以上	${v1}>=${v2}	-	DATE(${v1})>=DATE(${v2})
以下	${v1}<=${v2}	-	DATE(${v1})<=DATE(${v2})
未設定	-	ISBLANK(${v1})	ISBLANK(${v1})

式の入力ダイアログ

式を入力する専用のダイアログを用意しています。図の赤枠部分に示すボタンを押下して、入力ダイアログを開きます。

図 ダイアログを開く

ダイアログでは、関数と項目名をマウスで選択できます。

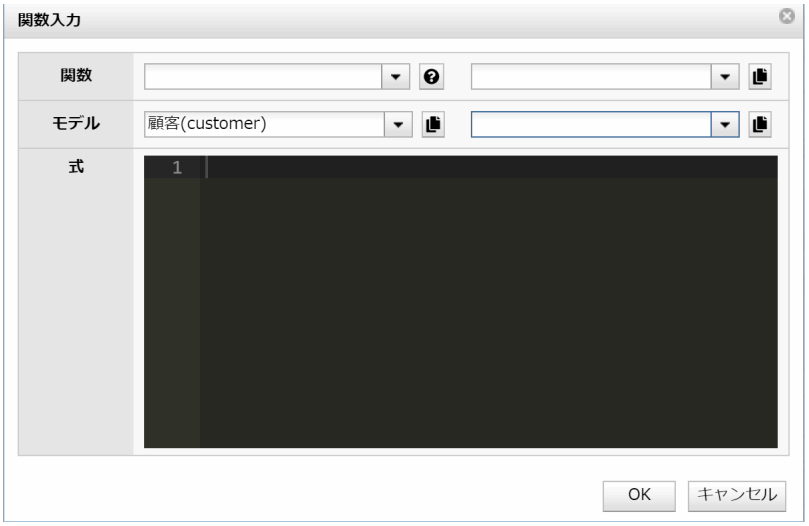


図 入力ダイアログ

はじめに関数の「種別」を指定します。種別の詳細は「関数」のページをお読みください。

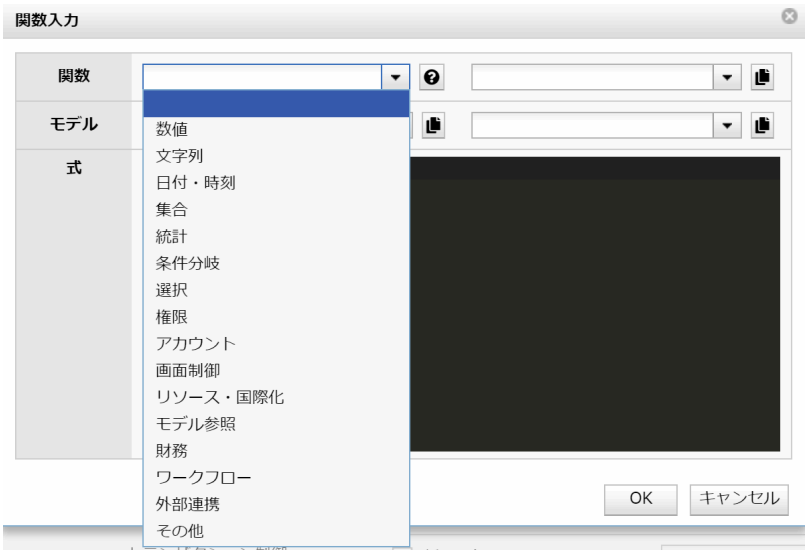


図 関数の種別を指定する

種別を指定すると、さらに利用できる関数を一覧から選択することができます。

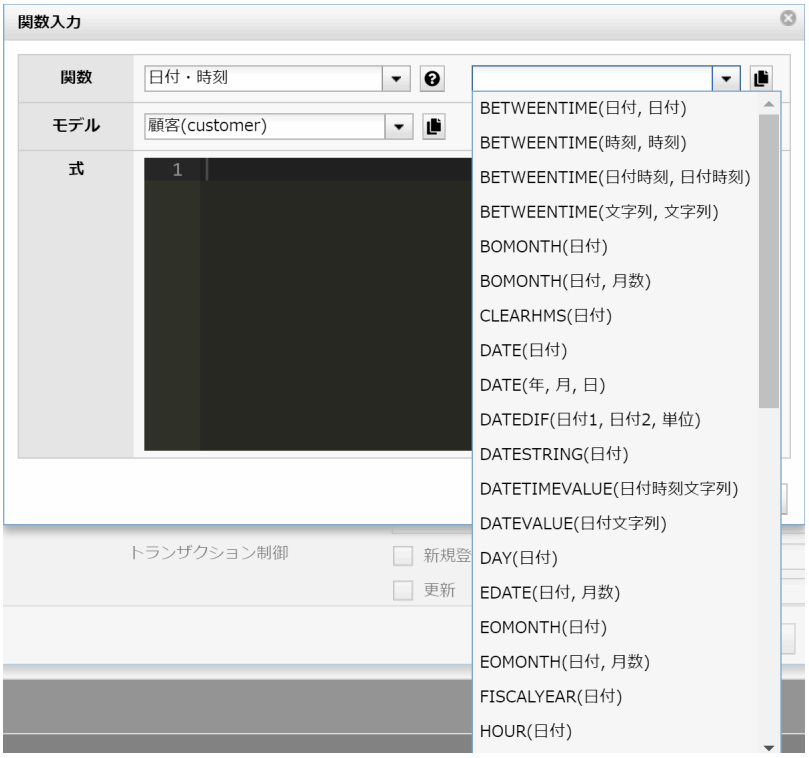


図 関数を指定する

利用したい関数を選択後、「コピー」ボタンを押します。

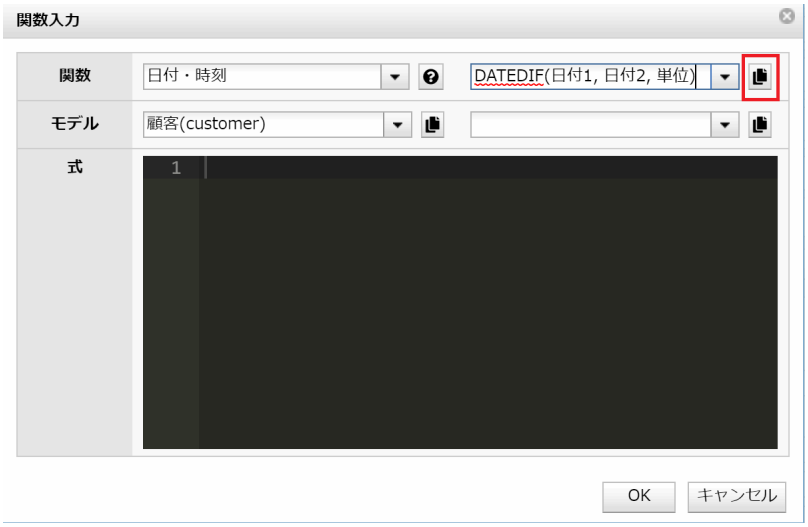


図 コピーボタン

式の入力欄で CTRL-V キーで「ペースト」します。

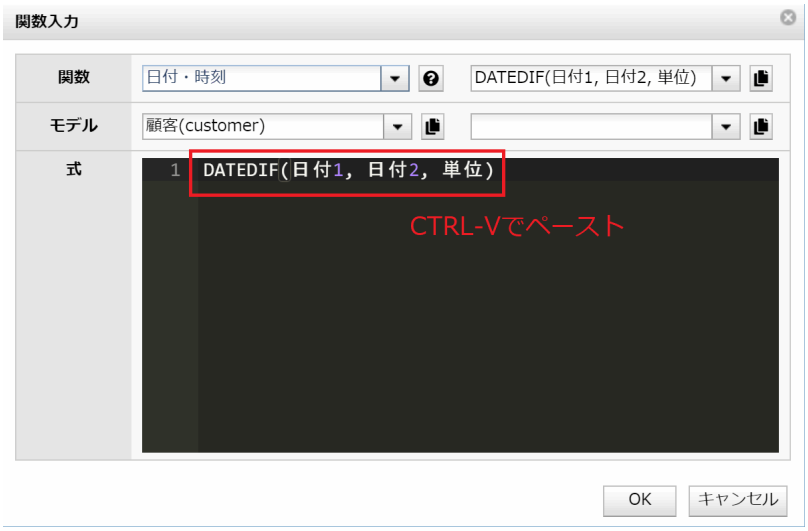


図 ペースト

さらに、モデル内の項目を選択することができます。

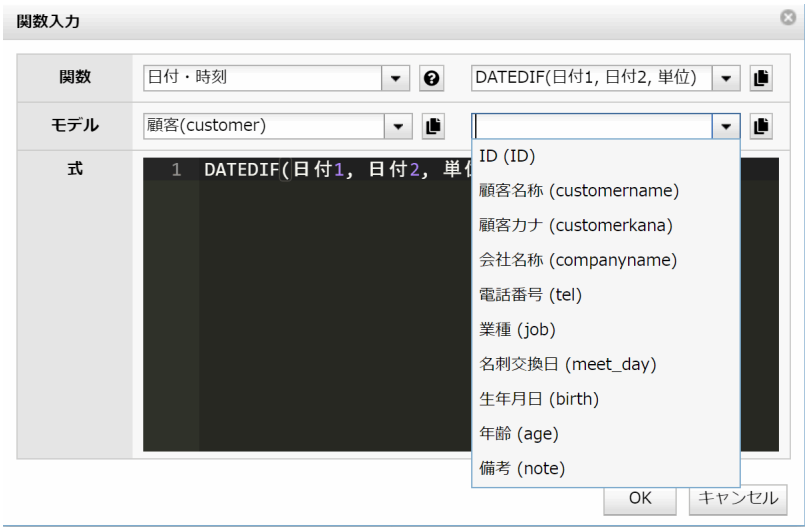


図 項目を選択する

項目を選択後、同様にコピーボタンを押下します。

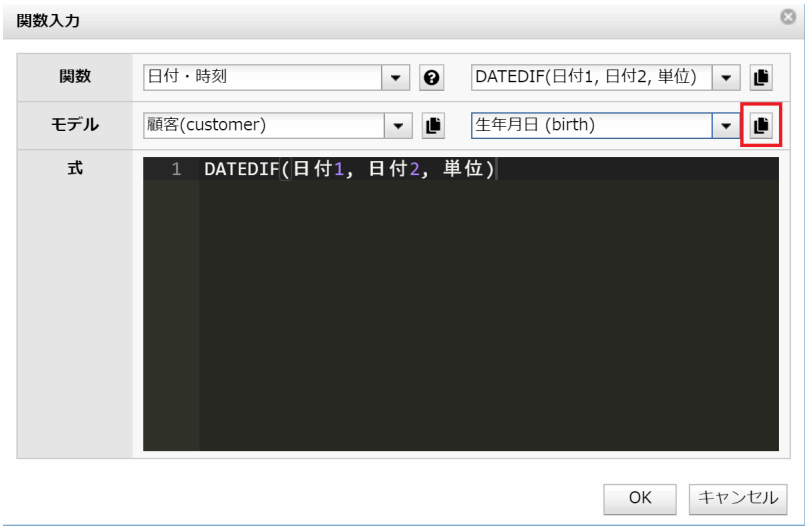


図 項目名をコピーする

先ほど入力した関数の引数部分で、この項目に相当する箇所をマウスで選択します。

図 引数部分をマウスで選択状態としておく

ペーストすることで、項目名が入ります。項目名はプレースホルダ表記となっています。（"${" と "}" で囲まれています。）

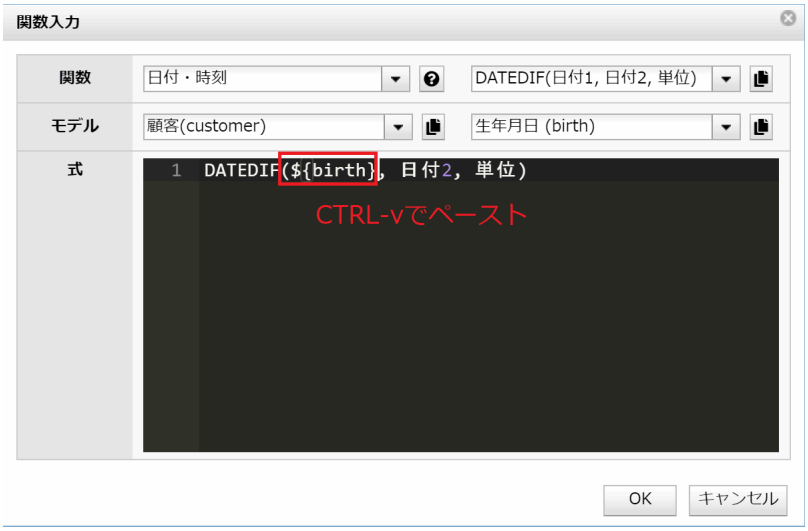
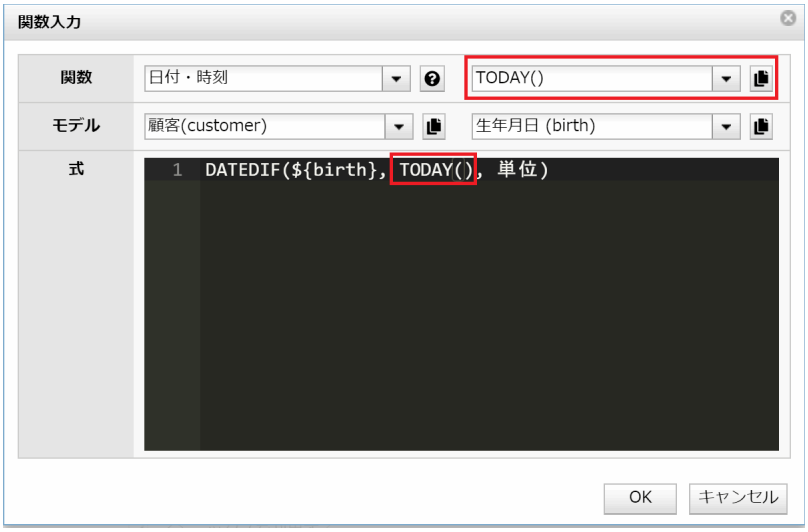


図 ペースト

別の関数を選択し、第二引数部分にペーストしてみます。



直接、キーボード操作で入力することもできます。

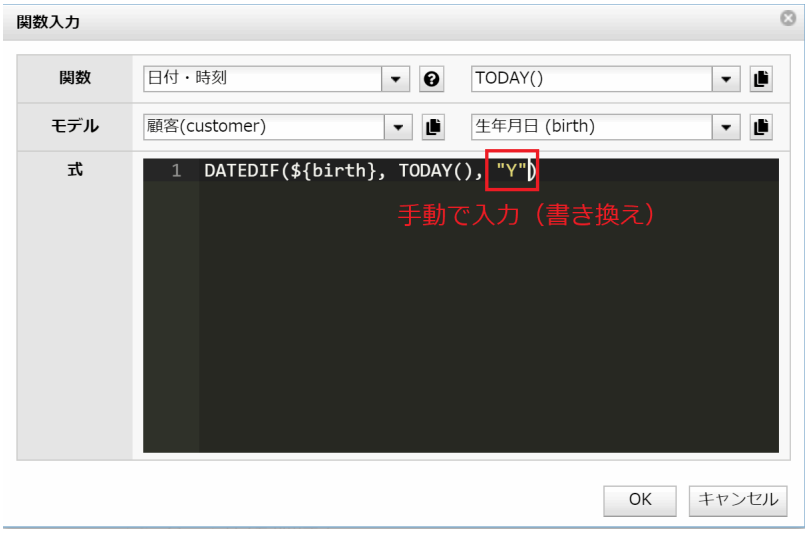


図 第三引数を手動で書き換える

OKボタンを押すことで、計算式の欄に反映されます。

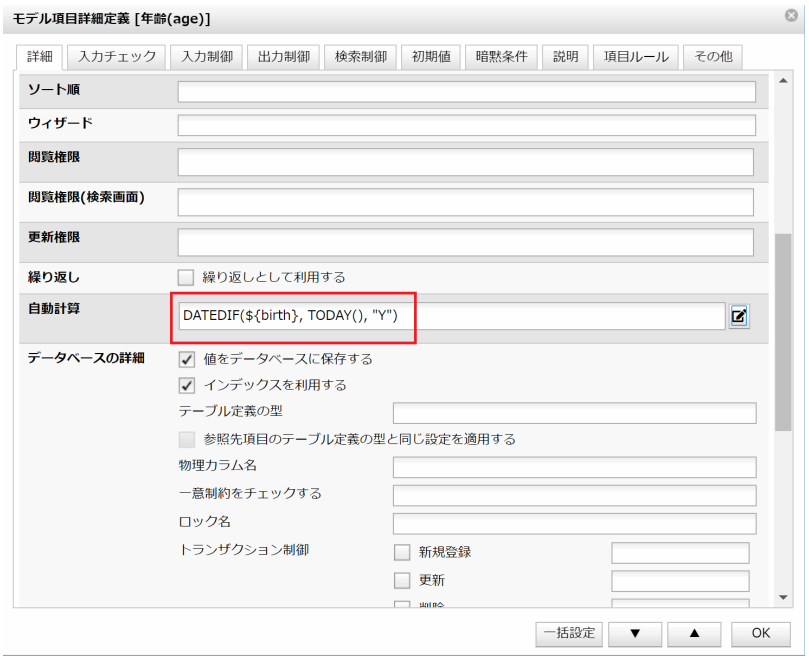


図 式を確定する

値をデータベースに保存する

標準は「値をデータベースに保存する」となっています。

図 値をデータベースに保存するが有効な状態

- 計算項目が「値をデータベースに保存する」となっていた場合、登録・更新時にのみ計算されます。表示時はデータベースに格納されている値がそのまま用いられます。
- 計算項目が「値をデータベースに保存する」となっていない場合、表示のタイミングで常に最新の値で再計算されます。

自分自身を演算する場合

数値型項目に次の式を設定した場合を想定します。

```
Me()+1
```

(84)

このとき、データベース保存の有無により計算結果は異なります。

種別	説明
DB保存あり	更新の都度、値は +1 されて保存されます。
DB保存なし	初期値 0 が +1 され、常に 1 となります。

(84) Me は自分自身を示す特別な関数です。

精度

Excelでは格納できる有効桁数は15桁となっています。そのため、例えば "11000.00000000001" は強制的に "11000.00000000000" となります。しかし Wagby は Java の浮動小数点仕様に準拠しているため、Excelと動きが異なります。（上記例では、強制的に小数部をゼロとすることはありません。）そのため、小数点を含む計算では Excel と結果が異なることがあります。ここでは、計算誤差を回避するいくつかの手法をご説明します。

- [参考] Excel のセルに桁数の多い数字を入力すると最後の桁がゼロに変更される［日本語］

金額の計算（演算誤差を考慮した四則演算）

加減乗除の演算子 (+,-,*,/) ではなく、関数 ADD,SUB,MUL,DIV を用いると、演算誤差を考慮した加減乗除を行うことができます。大きな金額を扱う場合や、途中で小数を含むような計算で有効です。

例えば ${item1} + ${item2} という式は、次のように記述できます。

```
ROUND(VALUE(ADD(${item1},${item2})),1)
```

ADD,SUB,MUL,DIV関数の戻り値は、数値の文字列表現です。例えば "123.456" となります。VALUE関数を適用することで、8バイト浮動小数点に戻すことができます。

ROUND関数は数値を四捨五入して指定された桁数にします。詳細は、各関数の説明をお読みください。

出力フォーマットの指定

出力フォーマットを指定することができます。例えばフォーマット中に「%」文字を指定すると、計算結果が自動的に100倍されます。詳細は「数値-出力フォーマット」を参照してください。

整数型の計算結果を小数点型で表現する場合

整数型同士の計算結果を小数点で表現したい場合、計算時に（整数型を）小数点型に「型変換」する方法があります。

次の計算式のように項目名の前に「(double)」と「8バイト浮動小数点（double）型」への変換を明記します。

```
(double)${uriagekingaku}/(double)${yosankingaku}
```

(85)

演算誤差を回避する式の書き方

小数点型の演算には常に誤差が含まれることに注意してください。例えば次の式において

```
ROUND(${item1}*0.12+${item2}*0.34, 1)
```

ROUND関数に渡す計算値が 10.8499999999 となった場合、ROUND によって 10.9 ではなく 10.8 になってしまいます。

ADD,SUB,MUL,DIV 関数を使うことで、このような問題を回避できます。もう一つの簡易手法として、式を工夫してできるだけ整数演算に持ち込むことで演算誤差を極小化する方法があります。

上の式は、次のように作成することができます。

```
ROUND((${item1}*12+${item2}*34)/100.0, 1)
```

計算結果が一時的に整数の範囲を超える場合には注意が必要です。上記の例で item1 が 10 億という値をとる場合、12倍すると 120 億になりますが、これは整数型の表現できる範囲（42億）を超えるため正しい結果になりません。整数型が表現できる数値の範囲は"モデル > 型"をお読みください。

(85) この方法は Java 開発者向けです。（double などの「型」は Java 文法に準拠しています。）

空文字とnull

"null" は "値がない" という意味で用いられます。null がセットされた項目は、表示時に空白となります。

"null" と "0" は違うものです。また、"null" と "" (空文字) も違うものです。 null は値がない、という特別な意味をもちます。

必須項目に null をセットすることはできません。

未入力項目の扱い

未入力項目（入力フィールドだが、何も入力していない状態）の値は、次のようになります。

型	値
数値	0
それ以外	null

このため式中で "0" という値の扱いは注意が必要です。入力値が "0" なのか、未入力状態なのかを区別できないためです。

例えば数値型項目 item1 に値が入力されていればそれを使い、未入力なら item2 の値を使うという式を次のように記述した場合、意図に反して item1 が未入力でも値 "0" が返されます。

```
IF(${item1} >= 0, ${item1}, ${item2})
```

項目 item1 に格納される値は常に "0" より大きいという場合、等号 (=) を除くことで意図通りの動作になります。

```
IF(${item1} > 0, ${item1}, ${item2})
```

ISEMPTY関数

ISEMPTY関数を使って項目に値が入っているかどうかを判定することができます。未入力の場合は true を返します。入力済の場合は false を返します。

数値型項目を例に説明します。

型と性質	ISEMPTY関数の戻り値	説明
数値型-必須	false	必須項目の場合、初期値0がセットされているため。
数値型-非必須:未入力	true	未入力の判定が可能。
数値型-非必須:入力あり	false	入力された時点でfalseになる。

モデル参照

モデル参照項目の指定方法

モデル参照項目は ID 部と内容部から構成されます。次に示すような形式で扱うことができます。

例：スタッフモデルの構成

ID部	内容部
1000	山田 太郎
1001	鈴木 一郎

ここで、上記のスタッフモデルを参照する項目 staffname を用意したとします。 自動計算式の中では、次のような指定を行うことができます。

```
${staffname}
```

内容部を取得します。戻り値の型はスタッフモデルへの「参照先モデル項目」に指定した項目の型になります。 上の例では "山田 太郎" "鈴木 一郎" という文字列が返されます。

```
${staffname#id}
```

ID部を取得します。戻り値の型はスタッフモデルへの「参照モデルの主キー」になります。型はスタッフモデルの主キー項目の型と同じになります。 上の例では 1000, 1001 という数値が返されます。

```
${staffname#content}
```

${staffname} と同じ意味になります。

ユーザー定義チェックにおける内容部の取得

ユーザー定義チェックの場合、内容部を取得するためには常に #content 指定が必要になります。 上の例では${staffname} ではなく、${staffname#content} をお使いください。

リストボックス項目の扱い

リストボックスとして参照している項目を式中に指定した場合、新規登録画面を開いたときの初期値は null となります。これは選択肢の先頭が「(未選択)」となっているためです。未選択項目を作成しない場合の初期値は、表示されている選択肢の先頭の値を（#id　や　#content　を指定して）取得することができます。

選択肢の絞り込み指定を行った場合、未選択項目を作成しない場合でも初期値は null となります。（絞り込み操作を行うことで、はじめて選択値がセットされます。）

参照連動項目

参照連動項目は #id, #content 指定を行うことはできません。転記された値をそのまま取り扱います。

参照連動元の項目は図に示すように ID 値が格納されます。

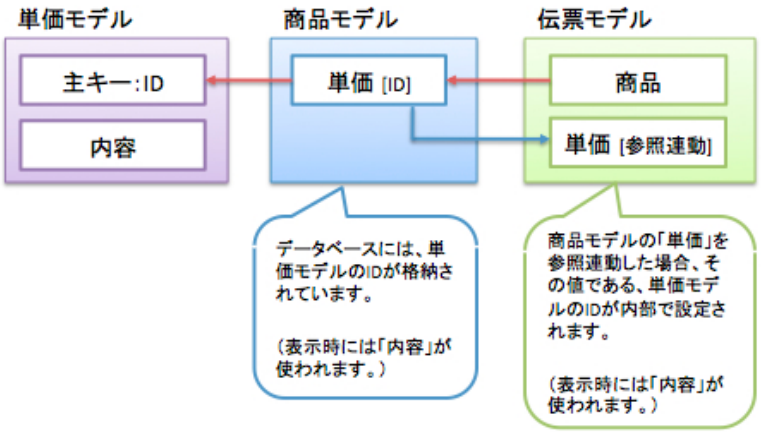


図 参照連動項目の取り扱い

ここで「単価モデル」の「内容」を用いたい場合は、図のように内容部を参照連動してください。

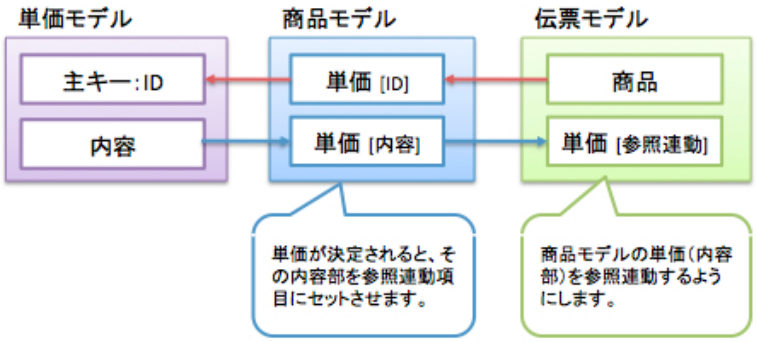


図 参照連動をつなぐ

関数と組み合わせたときの型情報

${XXX#id}の型は参照先モデルの主キーの型となります。

このため、関数内で利用する場合は必要に応じてTOSTRを用いて文字列に変換してください。またはVALUEを用いて数値に変換してください。

例：モデル参照（リストボックス）の項目 item1 が選択済みのとき、選択された ID 値を返す。そうでなければ 文字列「(未選択)」を返す。（この演算結果は文字列型の項目に格納する、としています。）

```
IF(SELECTED(${item1}), TOSTR(${item1#id}), "(未選択)")
```

item1#id は整数型であるため関数 TOSTR で囲み、文字列型へ変換しています。

主キーが複合キーのモデルを参照する項目

モデル model1 の項目 item1 はモデル参照であり、参照先モデル model2 が複合主キー pkey1, pkey2 となっている構成を想定します。

設計情報の上では "item1" という項目は一つですが、Wagbyの内部では二つの値を保持するフィールド（値を保持する入れ物）が用意されています。フィールドの名前は "item1_pkey1", "item1_pkey2" となっています。

(86)

この "item1" に計算式を設定する場合はスクリプトを利用します。スクリプトの「ヘルパ > 計算」のタイミングで、値をセットしてください。例えば次のようなスクリプトになります。

```
model1.item1Pkey1 = (pkey1の型に準拠した値);
model1.item1Pkey2 = (pkey2の型に準拠した値);
```

スクリプトでは、項目IDはキャメル記法を使います。

参照連動との組み合わせ

上記の例で、さらに item1 の値を連動元とした参照連動項目がある場合、スクリプトで参照連動の解決を行うコードも記述するようにしてください。[詳細...]

ジョブにおける式記述の制約

ジョブで用いる式（メール送受信ジョブにおける起動条件、成功時・失敗時の式など）記述では、モデル参照項目に対して #content 指定を行うことができません。#id のみ、利用できます。

(86) この命名規則の詳細は Appendix > 複合キーの内部的な表現をお読みください。

計算式の例

空白・改行の除去

文字列型の「備考 (note)」項目に含まれる文字列から余分なスペース（半角、全角）、タブ、改行文字を取り除くには、次の式を設定します。

```
TRIM(${note})
```

詳細はTRIMの説明をお読み下さい。

条件によって算出値を変える

数値型項目 source の値により、同モデル内の数値型項目 cvalue を変えるという例を紹介します。 次のようなルールを実現するものとします。

source の値	cvalue の値
10より小さい	100
20より小さい	200
それ以外	300

このとき、項目 cvalue の式は次のように記述することができます。

```
IF(${source}<10,
  100,
  IF(${source}<20,
    200,
    300
  )
)
```

(87)

計画日と実績日の差異

「計画日(plandate)」と「実績日(realdate)」を含むモデル定義があるものとします。ここで、「遅延」（整数型）の値を次のルールによって算出するものとします。

条件	遅延の算出ルール
実績日が未入力	今日の日付と、計画日(plandate)との差
実績日が入力済	実績日(realdate)と、計画日(plandate)との差

このとき、遅延項目（整数型）の式は次のように記述することができます。

```
IF(${realdate}==null,
  DATEDIF(${plandate},TODAY(),"D"),
  DATEDIF(${realdate},${plandate},"D")
)
```

ここで "${realdate}==null" とは、この値がセットされていないという意味です。 null と比較できる項目の型は文字列型、日付型、時刻型、日付時刻型です。

(88)

(87) 改行は入力しなくとも結構です。（ここでは見やすくするために入れています。）

(88) 改行は入力しなくとも結構です。（ここでは見やすくするために入れています。）

文字列の連結

文字列項目 sitem1 と sitem2 を連結した、sitem3 を自動計算として記述することもできます。

次のような式になります。

```
${sitem1}+":"+${sitem2}
```

ところで上記式の場合、sitem1 または sitem2 の値が未設定(null)の場合や、空白の場合を考慮していません。 sitem1 と sitem2 のいずれもが有効（未設定でなく、空白でもない）の場合にのみ sitem3 を出力する場合は、次のような式になります。

```
IF(
  AND(
    !ISBLANK(${sitem1}), !ISBLANK(${sitem2})
  ),
  ${sitem1}+":"+${sitem2},
  ""
)
```

(89) (90)

条件付きSUM関数

繰り返しコンテナ payment 内に、日付型項目 pdate と金額 amount が用意されています。 このとき、payment/amount の合計値を（この繰り返しコンテナではない）total 項目にセットするようにします。ただし、次のルールを適用します。

- payment/pdate が本日の日付より前であれば payment/amount の値を用いる。
- そうでなければ 0 とする。

集計値を格納する項目 total に条件付き SUM 関数を適用します。

```
SUM(${payment/amount},
  "DATE(${payment/pdate})<=DATE(TODAY())")
```

(91)

日付の比較を行う場合、DATE 関数を使って日付を数値化する必要があります。

以上を整理すると、次のような定義になります。（必要な項目のみ抜粋しています。）

項目名（日本語）	項目名（英語）	項目の型	自動計算式
集計	total	8バイト整数	SUM(${payment/amount}, "DATE(${payment/pdate})<=DATE(TODAY())")
支払	payment	繰り返しコンテナ	
支払回数	payment/pid	繰り返しコンテナID	
支払日	payment/pdate	日付	
支払金額	payment/amount	8バイト整数	

数値の按分

「売上」項目（数値）を、繰り返しコンテナに按分する方法を説明します。 ただし、余りが生じた場合は先頭の繰り返しコンテナに加えるというルールとします。

例えば売上値が"100"のとき、繰り返しコンテナが二つなら "50","50" になります。三つなら "34","33","33" になります。八つに分割したときの例を図に示します。

(89) 改行は入力しなくとも結構です。（ここでは見やすくするために入れています。）

(90) "+" 演算子の代わりに、CONCATENATE 関数を使うこともできます。

(91) 改行は入力しなくとも結構です。（ここでは見やすくするために入れています。）

図 売上の按分

売上項目を uriage とし、繰り返しコンテナの数を記憶する項目を count とします。 count 項目には次の式を適用します。

```
COUNT(${cont})
```

繰り返しコンテナ項目「各売上」の式を次のように設定します。 [92]

```
IF(CID()==1,
  ROUND(${uriage}/${count},0)+MOD(${uriage},${count}),
  ROUND(${uriage}/${count},0)
)
```

入力前の値との差

"SQL式 > 保存前の値を保持する項目を用意する" をお読みください。

数値項目にnullをセットする

"計算式 > 基本的な書き方 > 参考演算子" をお読みください。

加入日からの経過月数

日付型の「加入日 (joinday)」項目に入力された日から現在までの経過月数を計算します。

```
DATEDIF(${joinday}, TODAY(), "M")
```

詳細はDATEDIFの説明をお読み下さい。

生年月日から年齢

「顧客 (customer) 」モデルにおいて、「生年月日 (birth) 」項目と「年齢 (age) 」項目を用意した例を示します。

年齢項目 (整数型) の自動計算式を次のように記載します。

```
DATEDIF(${birth}, TODAY() ,"Y")
```

「何年何ヶ月」という表記を行なわせることもできます。この場合、年齢項目は文字列型とします。

[92] CID関数は繰り返しコンテナの行数を返します。 ROUND関数は数値を四捨五入して指定された桁数にします。 MOD関数は数値を除数で割った余りを求めます。

```
DATEDIF(${birth}, TODAY() ,"Y") +
  "年" +
  DATEDIF(${birth}, TODAY() ,"YM") +
  "ヶ月"
```
(93)

詳細はDATEDIFの説明をお読み下さい。

所要時間 (1)

サポートモデルの「開始時間（date1）」と「終了時間（date2）」から、「所要時間」を自動算出させる例を示します。 ここで「所要時間」の方は時刻型としてください。なお、24 時間を超過した分は切り捨てられます。

```
BETWEENTIME(${date1}, ${date2})
```

詳細はBETWEENTIMEの説明をお読み下さい。

所要時間 (2)

退勤時間(endtime)と出勤時間(starttime)の差を求めます。ただし、休憩時間(breaktime)を考慮します。 各項目はすべて時刻型として定義しています。

```
TIMEVALUE(
  TEXT(
    TIME(${endtime})-
    TIME(${starttime})-
    TIME(${breaktime}),
    "HH:mm:ss"
  )
)
```
(94)

所要時間 (3)

上の計算式は、次のように記述することもできます。

```
BETWEENTIME(
  TOTIME(
    MOVEMINUTE(${starttime},
      HOUR(${breaktime})*60 +
      MINUTE(${breaktime})
    )
  ),
  ${endtime}
)
```
(95)

所要時間 (4)

繰り返しコンテナ内の項目同士で所要時間を求める例を示します。

(93) 改行は入力しなくとも結構です。（ここでは見やすくするために入れています。）
(94) 改行は入力しなくとも結構です。（ここでは見やすくするために入れています。）
(95) 改行は入力しなくとも結構です。（ここでは見やすくするために入れています。）

図 動作イメージ

用意する項目

次のように繰り返しコンテナを定義します。

コンテナ	kks	繰り返しコンテナ
No	kks/id	繰り返しコンテナID
時間 1	kks/komoku1	日付・時刻
時間 2	kks/komoku2	日付・時刻
時間差分	kks/keisan	日付・時刻

図 繰り返しコンテナを用意する

「時間差分」(kks/keisan)項目に次の式を設定します。

```
TIMEVALUE(
  TEXT(
    TIME(${kks/komoku2})-
    TIME(${kks/komoku1}),
    "HH:mm:ss"
  )
)
```

日付から曜日

日付型項目「出荷日 (syukka_date)」の曜日を求める例を示します。

```
CHOOSE(WEEKDAY(${syukka_date}),
  "日","月","火","水","木","金","土")
```

(96)

詳細はWEEKDAYの説明をお読み下さい。

日付から年、月、日を切り出す

日付型項目「出荷日 (syukka_date)」から、年、月、日の部分をそれぞれ文字列として別項目に格納する例を示します。

```
TEXT(${syukka_date},"{1,number,00}")

TEXT(${syukka_date},"MM")

TEXT(${syukka_date},"dd")
```

ゼロ詰め文字の作成

製品モデルの整数型項目「製品ID (productid)」（主キー）をゼロ詰めの 4 桁と定義する例を示します。

```
PADDING("0", 4, ${productid})
```

(96) 改行は入力しなくとも結構です。（ここでは見やすくするために入れています。）

詳細はPADDINGの説明をお読み下さい。

計算式かつ編集可能な項目を用意する

例

「見積書」モデルの「請求金額」は計算式で求められますが、かつ、入力用の編集項目も用意しています。

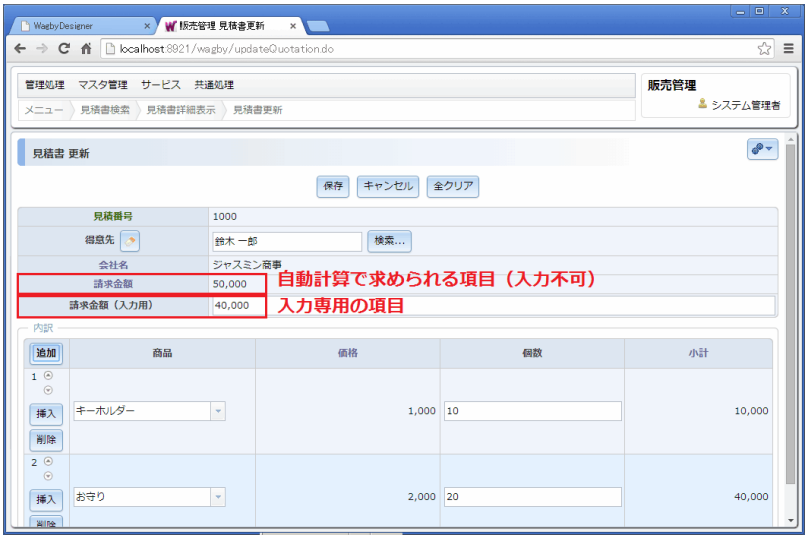


図 請求金額項目を自動計算かつ入力可能とする

入力欄に値が入っていた（すなわち手動入力された）場合はその値を優先します。入力欄が空の場合は、計算値をそのまま使います。

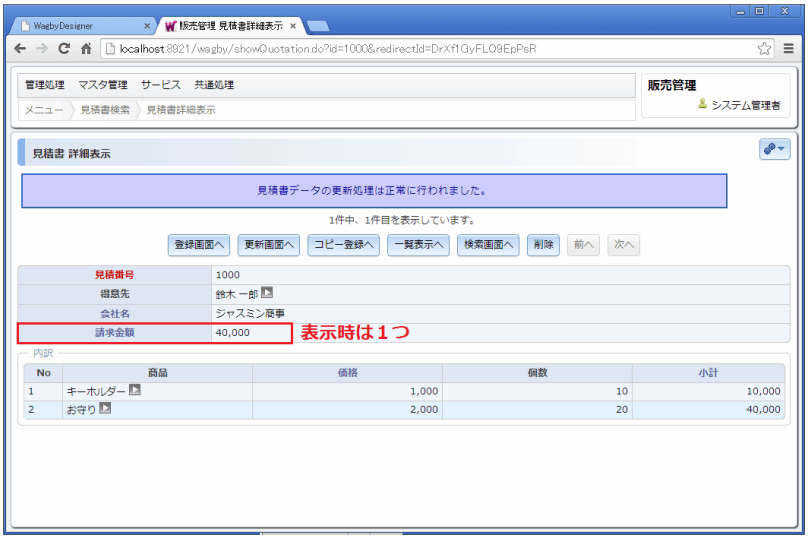


図 表示時は一つのみとなる

定義方法

「請求金額」の項目を3つ用意します。それぞれ次の特徴があります。

項目ID	用途	計算式	DB保存	表示時に隠す

total_calc	登録更新画面で自動計算値を保持する	SUM(${precord/p_syoukei})	しない	○
total_input	登録更新画面で入力値を保持する	なし	しない	○
total	詳細画面用	IF(${total_input}, ${total_input}, ${total_calc})	する	-

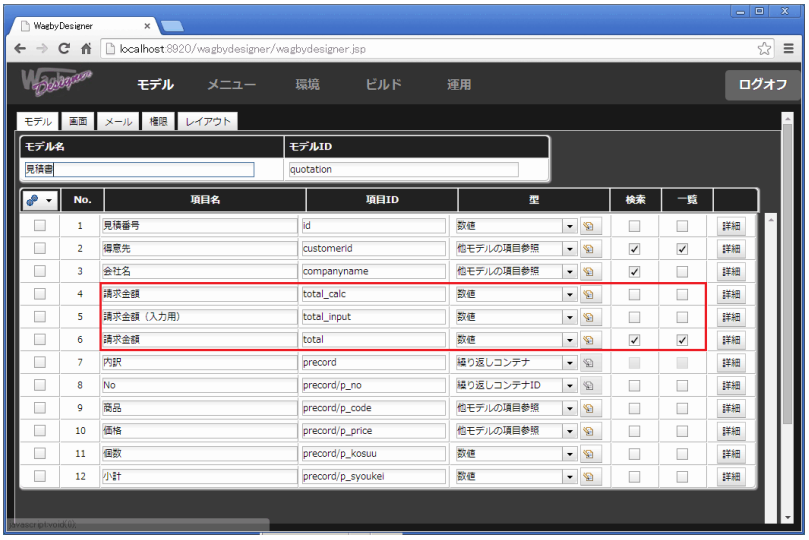


図 請求金額項目の定義

total_calc項目

登録更新画面で自動計算値を保持する目的で用意します。値はデータベースに保存しません。 表示時は隠し項目とします。

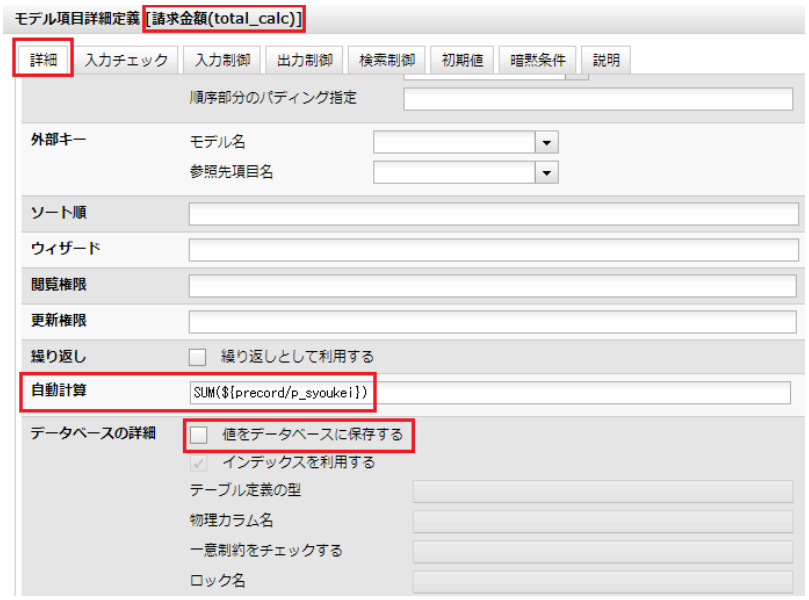


図 total_calc項目の定義(1)

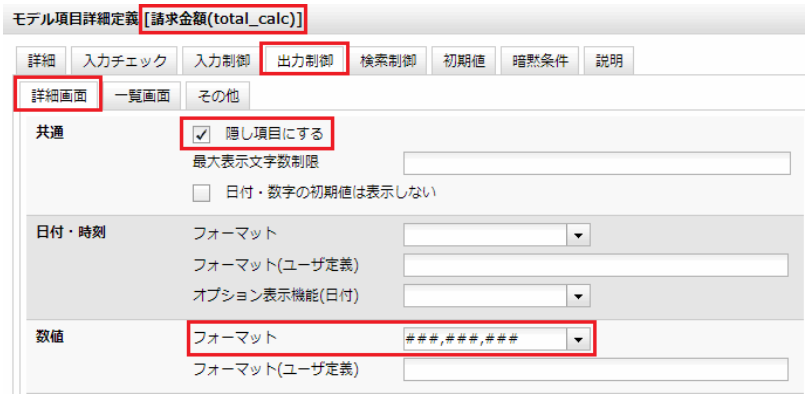


図 total_calc項目の定義(2)

total_input項目

登録更新画面で入力用項目を用意します。値はデータベースに保存しません。　表示時は隠し項目とします。

図 total_input項目の定義(1)

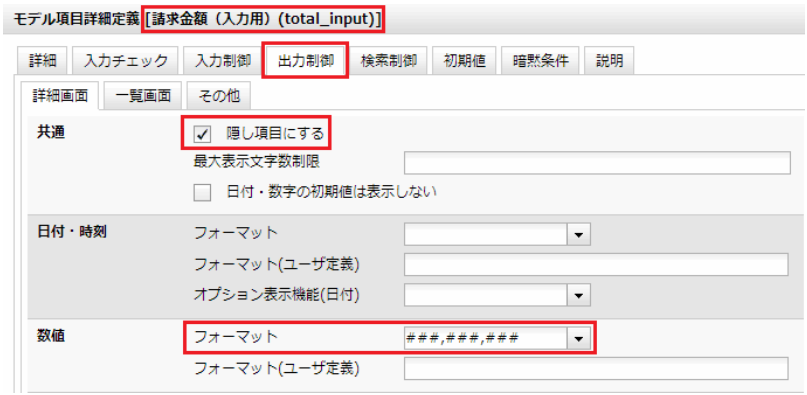


図 total_input項目の定義(2)

total項目

total_input項目に入力があれば、こちらを正とします。そうでなければ自動計算値(total_calc項目)の値をデータベースに保持します。検索時はこの項目を使って行います。

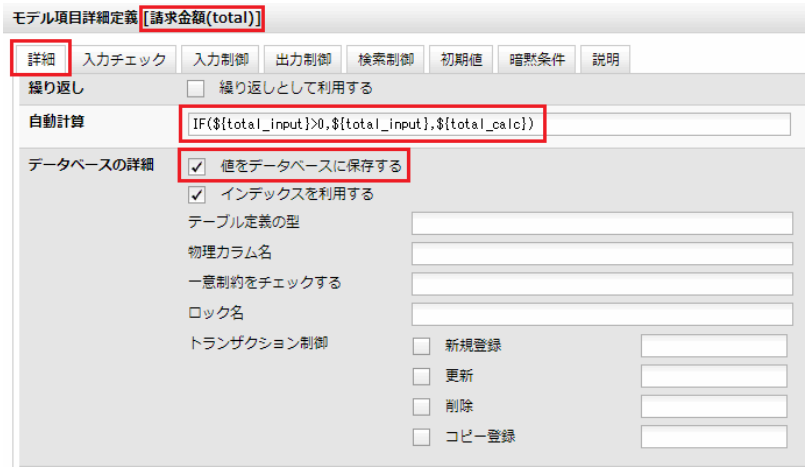


図 total項目(1)

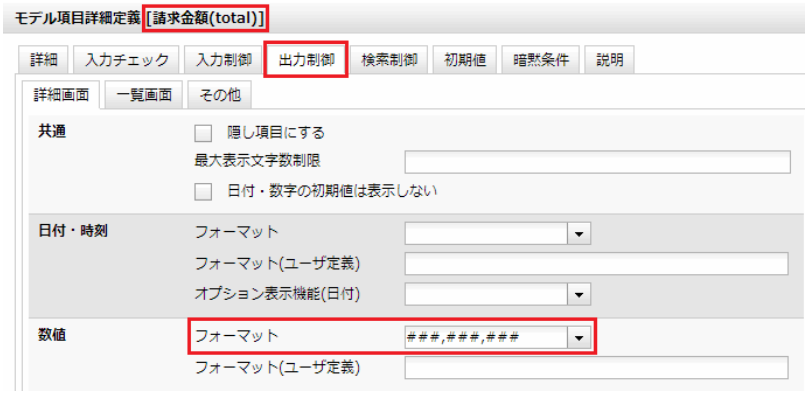


図 total項目(2)

JavaScriptを利用する

「追加ボタン」を用意し、この押下によって計算を行い、値を上書きする方法もあります。この場合は入力可能項目に対して直接、値を書き換えます。

詳細は"項目のオリジナルボタン押下で JavaScript を動作させる"をお読みください。

関数

数値関数

COSH

COSH(双曲線余弦を求める実数)

数値の双曲線余弦（ハイパーボリック コサイン）を返します。

例

```
COSH(${item1})
```

DEGREES

DEGREES(数値)

ラジアンを度に変換します。戻り値の型は 8 バイト浮動小数点型です。

例

```
DEGREES(${item1})
```

例	動作
DEGREES(PI())	180

DIV

DIV(数値,数値,スケール[,丸めモード])

DIV(数値の文字列表現,数値の文字列表現,スケール[,丸めモード])

引数の数を徐算します。内部ではJavaが提供するBigDecimalクラスを使った演算を行います。浮動小数を含む厳密な計算に向いています。戻り値は数値の文字列表現になります。

例

```
VALUE(DIV(ADD(MUL(10000, "0.12"),MUL(10000, "0.34")),100,2,"HALFUP"))
```

上記例のように、引数に数値と文字列表現を混在させることもできます。0.12という浮動小数を表現した段階で、すでに誤差が含まれてしまうため、より確実な演算を行う場合は、浮動小数部は "0.12" というように文字列を引数とする方がよいでしょう。

スケールは小数点以下の桁数です。

丸めモードは次の表記を指定できます。省略時は HALFUP（四捨五入）になります。

指定	説明
CEILING	正の無限大に近づくように丸めるモード
DOWN	ゼロに近づくように丸めるモード
FLOOR	負の無限大に近づくように丸めるモード
HALFDOWN	五捨六入
HALFEVEN	末尾が偶数のほうに丸める
HALFUP ※省略時	四捨五入
UP	0 から離れるように丸めるモード

DIV関数の戻り値にVALUE関数を適用することで、8バイト浮動小数値に変換することができます。

EVEN

EVEN(数値)

最も近い偶数に切り上げた値を返します。戻り値の型は整数型です。

例

```
EVEN(${item1})
```

例	動作
EVEN(1.5)	2
EVEN(3)	4
EVEN(-1)	-2
EVEN(0)	0

EXP

EXP(数値)

e を底とする数値のべき乗を返します。戻り値の型は 8 バイト浮動小数点型です。

例

```
EXP(${item1})
```

FACT

FACT(数値)

数値の階乗を返します。数値に整数以外の値を指定すると、小数点以下が切り捨てられます。戻り値の型は 8 バイト整数です。

例

```
FACT(${item1})
```

例	動作
FACT(5)	120

FIXED

FIXED(数値, 桁数, 桁区切り)

数値を四捨五入し、ピリオド(".") とカンマ(",") から構成される文字列に変換します。 桁数は四捨五入の位置を示します。省略すると小数第二位とみなします。 桁区切り部分は "true" か "false" を指定します。省略すると false とみなします。

例：3桁目で四捨五入とし、桁区切り(3桁毎に "," を挿入する）を行う。

```
FIXED(${item1}, 3, true)
```

例	動作
FIXED(123.4, 0, false)	"123"
FIXED(1234567, -1, true)	"1,234,570"
FIXED(1234.567, 5)	"1234.56700"
FIXED(1234.567)	"1234.57"

FLOOR

FLOOR(数値, 基準値)

数値を基準値の倍数に近い値に切り下げます。戻り値の型は 8 バイト浮動小数点型です。

例

```
FLOOR(${item1}, 1)
```

例	動作
FLOOR(2.5, 1)	2
FLOOR(-2.5, -2)	-2
FLOOR(-2.5, 2)	-2.5
FLOOR(1.5, 0.1)	1.5

時間にも適用できます。

例	動作
TEXT(FLOOR(${time}, "1:00"), "hh:mm")	"11:00" ※ ${time} 項目の型が時刻型で、値が "11:30:00" のとき
DATETIMEVALUE(TEXT(DATE(${datetime}) + FLOOR(TOTIME(${datetime}), "1:00"), "yyyy-MM-dd HH:mm:ss"))	2008-3-14 11:00:00 ※ ${datetime} 項目の型が日付時刻型で、値が "2008-3-14 11:30:00" のとき

HYPOT

HYPOT(数値1, 数値2)

中間のオーバーフローやアンダーフローなしに sqrt(x2+y2) が返されます。

例

```
HYPOT(${x}, ${y})
```

IEEEremainder

IEEEremainder(数値1, 数値2)

IEEE 754 標準に従って、2 個の引数について剰余を計算します。

例

```
IEEEremainder(${item1}, ${item2})
```

ABS

ABS(数値)

引数の数値の絶対値を返します。戻り値の型は「整数」です。

例

```
ABS(${item1})
```

例	動作
ABS(10)	10
ABS(-10)	10
ABS(-3.14)	3.14

INT

INT(数値)

指定した数値を超えない最大の整数を返します。

例

```
INT(${item1})
```

例	動作
INT(8.9)	8
INT(-8.1)	-9

LN

LN(数値)

数値の自然対数を返します。戻り値の型は 8 バイト浮動小数点型です。

例

```
LN(${item1})
```

LOG

LOG(数値)

数値の自然対数を返します。戻り値の型は 8 バイト浮動小数点型です。

例

```
LOG(${item1})
```

LOG(数値, 底)

指定された数を底とする数値の対数を返します。戻り値の型は 8 バイト浮動小数点型です。

例

```
LOG(${item1}, ${item2})
```

LOG1P

LOG1P(数値)

引数と 1 の合計の自然対数を返します。

例

```
LOG1P(${num})
```

LOG10

LOG10(数値)

10 を底とする数値の対数（常用対数）を返します。戻り値の型は 8 バイト浮動小数点型です。

例

```
LOG10(${item1})
```

例	動作
LOG(10)	1.0
LOG(8,2)	3.0
LOG10(10)	1.0

MOD

MOD(数値, 除数)

数値を除数で割った余りを求めます。戻り値の型は整数型です。

例

```
MOD(${item1}, ${item2})
```

例	動作
MOD(10,3)	1

MROUND

MROUND(数値, 基準値)

数値を基準値の倍数に丸めます（四捨五入）。戻り値の型は 8 バイト浮動小数点型です。

例

```
MROUND(${item1}, 1)
```

例	動作
MROUND(10, 3)	9
MROUND(-10, -3)	-9
MROUND(1.3, 0.2)	1.4
MROUND(5, -2)	5

時間にも適用できます。

例	動作
TEXT(MROUND(${time}, "0:30"), "hh:mm")	"11:30" ※ ${time} 項目の型が時刻型で、値が "11:31:00" のとき
DATETIMEVALUE(TEXT(DATE(${datetime}) + MROUND(TOTIME(${datetime}), "0:30"), "yyyy-MM-dd HH:mm:ss"))	2008-3-14 11:30:00 ※ ${datetime} 項目の型が日付時刻型で、値が "2008-3-14 11:31:00" のとき

MUL

MUL(数値,数値)

MUL(数値の文字列表現,数値の文字列表現)

引数の数を乗算します。内部ではJavaが提供するBigDecimalクラスを使った演算を行います。浮動小数を含む厳密な計算に向いています。戻り値は数値の文字列表現になります。

例：

```
VALUE(MUL("123.456", "456.789"))
VALUE(MUL(123.456, 456.789))
```

(97)

```
VALUE(MUL(1000, "0.1"))
```

このように、引数に数値と文字列表現を混在させることもできます。

MUL関数の戻り値にVALUE関数を適用することで、8バイト浮動小数値に変換することができます。

ODD

ODD(数値)

最も近い奇数に切り上げた値を返します。戻り値の型は整数型です。

例

(97) 上の二つは厳密には同じではありません。123.456という浮動小数を表現した段階で、すでに誤差が含まれてます。より確実な演算を行う場合は、浮動小数部は "123.456" というように文字列を引数とする方がよいでしょう。

```
ODD(${item1})
```

例	動作
ODD(3)	3
ODD(2)	3
ODD(1.5)	3
ODD(0)	1
ODD(-2)	-3

PI

PI()

円周率πの近似値である数値 3.14159265358979 を返します。この数値の精度は 15 桁です。戻り値の型は 8 バイト浮動小数点型です。

例

```
PI()
```

ACOS

ACOS(求める角度のコサインの値)

数値のアークコサインを返します。アークコサインとは、そのコサインが数値であるような角度のことです。

例

```
ACOS(${item1})
```

入力値 item1 は、求める角度のコサインの値を、-1 ～ 1 の範囲で指定します。 戻り値の角度は、0（ゼロ）～ PI の範囲のラジアンとなります。

POWER

POWER(数値, 指数)

数値の指数乗を返します。数値、指数には整数または小数値を指定できます。戻り値の型は 8 バイト浮動小数点型です。

例

```
POWER(${item1}, 2)
```

例	動作
POWER(5,2)	25.0

QUOTIENT

QUOTIENT(分子, 分母)

除算の商の整数部を返します。戻り値の型は整数型です。

例

```
QUOTIENT(${item1}, ${item2})
```

例	動作

QUOTIENT(5, 2)	2
QUOTIENT(4.5, 3.1)	1
QUOTIENT(-10, 3)	-3

RADIANS

RADIANS(数値)

度をラジアンに変換します。戻り値の型は 8 バイト浮動小数点型です。

例

```
RADIAN(${item1})
```

例	動作
RADIANS(270)	4.712389

RAND

RAND()

0 以上で 1 より小さい乱数を発生させます。戻り値の型は 8 バイト浮動小数点型です。 (98)

例

```
RAND()
```

RANDBETWEEN

RANDBETWEEN(最小値, 最大値)

指定された範囲で一様に分布する整数の乱数を返します。戻り値の型は整数型です。 (99)

例

```
RANDBETWEEN(${min}, ${max})
```

RINT

RINT(数値)

引数の値にもっとも近く、計算上の整数に等しい double 値を返します。

例

```
RINT(${item1})
```

ROUND

ROUND(数値,桁数)

数値を四捨五入して指定された桁数にします。戻り値の型は「4バイト浮動小数点」または「8バイト浮動小数点」になります。（引数の数値の型によって、戻り値の型も自動的に決まります。）

(98) ある項目 A で RAND() を適用し、別の項目 B が A の値を参照する場合、B の値と A の値は同じになります。 ある項目 A で RAND() を複数回用いた場合、最初の RAND() の結果と二つ目以降の RAND() の結果は同じ値となります。

(99) ある項目 A で RAND() を適用し、別の項目 B が A の値を参照する場合、B の値と A の値は同じになります。 ある項目 A で RAND() を複数回用いた場合、最初の RAND() の結果と二つ目以降の RAND() の結果は同じ値となります。

・　桁数に正の数を指定すると、数値は小数点の右側 （小数点以下） で四捨五入され、小数点以下の桁数が桁数に等しくなります。
・　桁数に 0 を指定すると、数値は最も近い整数として四捨五入されます。
・　桁数に負の数を指定すると、数値は小数点の左側（整数部分）の指定した桁（1 の位を 0 とする）に四捨五入されます。

例 小数点2桁に丸める場合

```
ROUND(${item1}/${item2}, 2)
```

例	動作
ROUND(2.15, 1)	2.15 を小数点第 1 位に四捨五入します (2.2)
ROUND(2.149, 1)	2.149 を小数点第 1 位に四捨五入します (2.1)
ROUND(-1.475, 2)	-1.475 を小数点第 2 位に四捨五入します (-1.48)
ROUND(21.5, -1)	21.5 を小数点の左側 1 桁の 1 の位で四捨五入します (20)

この関数は、演算誤差をなくすものではありません。 ROUNDと演算精度についての詳細は"計算式>基本的な書き方>演算精度"をお読みください。

ROUNDDOWN

ROUNDDOWN(数値,桁数)

数値を切り捨てて指定された桁数にします。戻り値の型は「4バイト浮動小数点」または「8バイト浮動小数点」になります。（引数の数値の型によって、戻り値の型も自動的に決まります。）

例 小数点2桁で切り捨てる場合

```
ROUNDDOWN(${item1}/${item2}, 2)
```

この関数は、演算誤差をなくすものではありません。 ROUNDと演算精度についての詳細は"計算式>基本的な書き方>演算精度"をお読みください。

ROUNDUP

ROUNDUP(数値,桁数)

数値を切り上げて指定された桁数にします。戻り値の型は「4バイト浮動小数点」または「8バイト浮動小数点」になります。（引数の数値の型によって、戻り値の型も自動的に決まります。）

例 小数点2桁に切り上げる場合

```
ROUNDUP(${item1}/${item2}, 2)
```

この関数は、演算誤差をなくすものではありません。 ROUNDと演算精度についての詳細は"計算式>基本的な書き方>演算精度"をお読みください。

SIN

SIN(数値)

指定した角度のサインを返します。サインを求める角度をラジアンを単位として指定します。戻り値の型は 8 バイト浮動小数点型です。

例

```
SIN(${item1})
```

例	動作
SIN(PI()/2)	1

SIN(30d*PI()/180d)	0.5 ※ 実数計算の場合、数字の後ろに "d" を付与してください。

ADD

ADD(数値,数値)

ADD(数値の文字列表現,数値の文字列表現)

引数の数を加算します。内部ではJavaが提供するBigDecimalクラスを使った演算を行います。浮動小数を含む厳密な計算に向いています。戻り値は数値の文字列表現になります。

例：

```
VALUE(ADD("0.001", "123.456"))
VALUE(ADD(0.001, 123.456))
```

(100)

```
VALUE(ADD(1000, "0.1"))
```

このように、引数に数値と文字列表現を混在させることもできます。

ADD関数の戻り値にVALUE関数を適用することで、8バイト浮動小数値に変換することができます。

SINH

SINH(双曲線正弦を求める実数)

数値の双曲線正弦（ハイパーボリック サイン）を返します。

例

```
SINH(${item1})
```

SIGN

SIGN(数値)

引数の数値が正／負／ゼロかどうかを判定します。 数値が正の数のときは 1, ゼロのときは 0, 負の数のときは -1 を返します。戻り値の型は「整数」です。

例

```
SIGN(${item1})
```

例	動作
SIGN(100)	1
SIGN(0)	0
SIGN(-100)	-1

SIGNUM

SIGNUM(数値)

引数の符号要素を返します。

(100) 上の二つは厳密には同じではありません。0.001という浮動小数を表現した段階で、すでに誤差が含まれてます。より確実な演算を行う場合は、浮動小数部は "0.001" というように文字列を引数とする方がよいでしょう。

例

```
SIGNUM(${item1})
```

SQRT

SQRT(数値)

正の平方根を返します。なお数値に負数を指定すると、NaN（無効な演算結果を意味する特殊表記）が返されます。戻り値の型は 8 バイト浮動小数点型です。

例

```
SQRT(${item1})
```

例	動作
SQRT(16)	4.0
SQRT(-2)	Double.NaN

SUB

SUB(数値,数値)

SUB(数値の文字列表現,数値の文字列表現)

引数の数を減算します。内部ではJavaが提供するBigDecimalクラスを使った演算を行います。浮動小数を含む厳密な計算に向いています。戻り値は数値の文字列表現になります。

例

```
VALUE(SUB("123.456", "0.001"))
VALUE(SUB(123.456, 0.001))
```

(101)

```
VALUE(SUB(1000, "0.1"))
```

このように、引数に数値と文字列表現を混在させることもできます。

SUB関数の戻り値にVALUE関数を適用することで、8バイト浮動小数値に変換することができます。

TAN

TAN(数値)

指定した角度のタンジェントを返します。タンジェントを求める角度をラジアンを単位として指定します。戻り値の型は 8 バイト浮動小数点型です。

例

```
TAN(${item1})
```

例	動作
TAN(45d*PI()/180d)	1 ※ 実数計算の場合、数字の後ろに "d" を付与してください。

(101) 上の二つは厳密には同じではありません。0.001という浮動小数を表現した段階で、すでに誤差が含まれてます。より確実な演算を行う場合は、浮動小数部は "0.001" というように文字列を引数とする方がよいでしょう。

TANH

TANH(双曲線余弦を求める実数)

数値の双曲線正接（ハイパーボリック タンジェント）を返します。

例

```
TANH(${item1})
```

TRUNC

TRUNC(数値)

数値の小数部を切り捨てて、整数に変換します。戻り値の型は整数型です。

例

```
TRUNC(${item1})
```

例	動作
TRUNC(8.9)	8
TRUNC(-8.1)	-8

ATAN

ATAN(求める角度のタンジェントの値)

数値のアークタンジェントを返します。アークタンジェントとは、そのタンジェントが数値であるような角度のことです。

例

```
ATAN(${item1})
```

入力値 item1 には、求める角度のタンジェントの値を指定します。戻り値の角度は、-PI/2 ～ PI/2 の範囲のラジアンとなります。

ATAN2

ATAN2(x座標値, y座標値)

指定された x-y 座標のアークタンジェントを返します。 アークタンジェントとは、x 軸から、原点 0 と x 座標、y 座標で表される点を結んだ直線までの角度のことです。

例

```
ATAN2(${item1}, ${item2})
```

入力値 item1, item2 には、それぞれ x 点の x 座標および y 点の y 座標を指定します。 戻り値の角度は、-PI ～ PI（ただし -PI を除く）の範囲のラジアンとなります。

ASIN

ASIN(求める角度のサインの値)

数値のアークサインを返します。アークサインとは、そのサインが数値であるような角度のことです。

例

```
ASIN(${item1})
```

入力値 item1 は、求める角度のサインの値を -1 ～ 1 の範囲で指定します。戻り値の角度は、-PI/2 ～ PI/2 の範囲のラジアンとなります。

CBRT

CBRT(数値)

引数の数値の立方根を返します。

例

```
CBRT(${item1})
```

CEILING

CEILING(数値, 基準値)

数値を基準値の倍数に近い値に切り上げます。戻り値の型は 8 バイト浮動小数点型です。

例

```
CEILING(${item1}, 1)
```

例	動作
CEILING(2.5, 1)	3
CEILING(-2.5, -2)	-4
CEILING(-2.5, 2)	-2.5
CEILING(1.5, 0.1)	1.5

時間にも適用できます。

例	動作
TEXT(CEILING(${time}, "1:00"), "hh:mm")	"12:00" ※ ${time} 項目の型が時刻型で、値が "11:30:00" のとき
DATETIMEVALUE(TEXT(DATE(${datetime}) + CEILING(TOTIME(${datetime}), "1:00"), "yyyy-MM-dd HH:mm:ss"))	2008-3-14 12:00:00 ※ ${datetime} 項目の型が日付時刻型で、値が "2008-3-14 11:30:00" のとき

COS

COS(数値)

指定した角度のコサインを返します。コサインを求める角度をラジアンを単位として指定します。戻り値の型は 8 バイト浮動小数点型です。

例

```
COS(${item1})
```

例	動作
COS(60d*PI()/180d)	0.5 ※ 実数計算の場合、数字の後ろに "d" を付与してください。

文字列関数

A1FORMAT

A1FORMAT(数字)

引数の数字をExcelの「A1形式」文字列に変換します。戻り値は文字列型です。

入力値	変換後の文字列
0	""（空文字）
1	A
25	Y
26	Z
27	AA
28	AB
701	ZY
702	ZZ
703	AAA
704	AAB

ISBLANK

ISBLANK(文字列)

ISBLANK(日付型)

ISBLANK(時刻型)

ISBLANK(日付時刻型)

引数が NULL または空白かどうかを調べます。NULL または空白なら真(true)を返します。戻り値の型は真偽値(trueまたはfalse)となるため、IF関数の内部などで利用できます。

```
ISBLANK(${item1})
```

例	動作
ISBLANK(" ");	true
ISBLANK(" A");	false

ISEMPTY関数との違い

ISBLANKの方が実行パフォーマンスが良いです。単純に値がNULLかどうかを判定する場合はISBLANKの利用を推奨します。ISEMPTYは数値型項目の未入力判定や参照連動項目にも対応していますが、その分、内部処理が複雑になっています。

ISINCLUDE

ISINCLUDE(文字列1,文字列2)

第一引数の値に、第二引数のいずれかの文字が含まれていれば true を返します。 戻り値の型は真偽値(trueまたはfalse)となるため、IF関数の内部などで利用できます。

```
ISINCLUDE(${item1},${item2})
```

例	動作
ISINCLUDE("Hello,World!", "aiueo");	true

ISINCLUDE("Hello,World!", "xyz");	false

JIS

JIS(入力文字列)

半角の文字列を全角の文字に変換します。 具体的には「アルファベット」「数字」「記号」「カタカナ」が対象になります。

JOIN

JOIN(文字列配列)

JOIN(文字列配列, 区切り文字)

文字列配列の各要素を、区切り文字で連結した文字列を返します。 区切り文字を指定しない場合は、","が区切り文字となります。 区切り文字が null の場合、長さ0の文字列 "" を区切り文字として使用します。 文字列配列が null 場合、長さ0の文字列 "" を返します。[102]

```
JOIN(${array_of_string_item}, "#")
```

JOINと対になる関数がSPLITです。

JPKANANORM

JPKANANORM(カナ文字が格納された文字列)

濁音、拗音を清音に正規化します。例えば「ヤマダ」という文字列が「ヤマタ」になります。戻り値の型は文字列型です。

```
JPKANANORM(${kananame})
```

詳細な変換ルールは次のとおりです。

入力文字：ヴガギグゲゴザジズゼゾダヂヅデドバビブベボパピプペポァィゥェォャュョヵヶッ 出力文字：ウカキクケコサシスセソタチツテトハヒフヘホハヒフヘホアイウエオヤユヨカケツ

LEFT

LEFT(文字列, [文字数])

文字列の先頭（左端）から指定された文字数の文字を返します。文字数を省略した場合は 1 になります。戻り値の型は文字列型です。

```
LEFT(${item1}, ${num})
```

例	動作
LEFT("abcde", 3)	"abc"

LEN

LEN(文字列)

文字列に含まれる文字数を返します。戻り値の型は整数型です。

[102] null は空白に置換されます。また null 要素に対する区切り文字は出力されません。 例えば引数が "北海道","青森",null,"秋田" という4つの要素だった場合、JOIN の結果は "北海道,青森,秋田" となります。

```
LEN(${item1})
```

例	動作
LEN("abcde")	5

LENB (8.2.0)

LENB(文字列, 文字エンコーディング)

LENと同じく文字列長を返しますが、第二引数に文字エンコーディングを指定できます。

例	動作
LEN("こんにちは");	5
LENB("こんにちは", "Shift-JIS");	10
LENB("こんにちは", "UTF-8");	15

LOWER

LOWER(文字列)

アルファベットの大文字を小文字に変換します。戻り値の型は文字列型です。

```
LOWER(${item1})
```

例	動作
LOWER("HELLO, WORLD.");	"hello, world."

LPAD

LPAD(文字列, 指定桁数)

LPAD(文字列, 指定桁数, 埋め込み文字)

文字列の左側に、指定桁数になるまで埋め込み文字を挿入します。埋め込み文字の指定がない場合は、空白を挿入します。

```
LPAD(${item1}, 6, "0")
```

例	動作
LPAD("ABC", 6);	" ABC"
LPAD("ABC", 6, "0");	"000ABC"

ACCOUNTKANANORM

ACCOUNTKANANORM(半角カナ文字が格納された文字列)

半角カナで表現された口座名義の正規化処理を行います。検索揺れや、名寄せに適用できます。引数の文字列は半角カナ文字列で、戻り値の型は正規化された半角カナ文字列となります。[(103)]

```
ACCOUNTKANANORM(${hankaku-kananame})
```

詳細な変換ルールは次のとおりです。

- ハイフン等が連続で複数ある場合は、一つにまとめる。

(103) 本関数を適用する文字列型項目の入力フィルタは、標準の入力フィルタである「文字列（一般）」から「全角->半角」「全角->半角（空白を含む）」のいずれかに変更してください。

・　「ヴ，ヂ，ヅ」を「ブ，ジ，ズ」に正規化する。
・　"株式会社"、"有限会社"等の略号を除去する。[104]
・　濁点、半濁点を除去する。
・　小文字(拗音・促音)を大文字に変換する。
・　ハイフンを母音に変換する。
・　文字列中に"エ"("オ")段の文字が連続し、かつ二文字目が"エ"("オ")の場合、"エ"を"イ"へ、"オ"を"ウ"へ変換する。
・　半角ブランク(スペース)を削除する。

LTRIM

LTRIM(文字列)

文字列の左側に含まれる余分なスペース（半角、全角）、タブ、改行文字を取り除きます。戻り値の型は文字列型です。

```
LTRIM(${item1})
```

例	動作
LTRIM(" ABC");	"ABC"

MAILADDRESS

MAILADDRESS(文字列)

MAILADDRESS(文字配列)

RFC822フォーマットに準拠したメールアドレスを表す文字列から、メールアドレス部を抽出します。 繰り返し項目（文字列）の項目を引数に指定することもできます。この場合、戻り値も文字配列になります。

入力文字列	結果
山田 太郎 <yamada@foo.bar.com>	yamada@foo.bar.com

MATCHRE

MATCHRE(検索文字列, 対象文字列)

対象文字列から検索文字列に一致する部分文字列を、文字列配列として返します。 検索文字列には、正規表現を用いることができます。

```
MATCHRE(${searchitem}, ${targetitem})
```

式の例	JOIN(MATCHRE("(S|E)[0-9]+_[0-9]+$", ${item}), ",")
入力の例	item = "ジャスミン太郎 S2010_0401¥n" + "E2010_0402 くばがさ次郎¥n" + "さんぴん花子 S2010_0403"
結果	"S2010_0401,S2010_0403"
説明	引数 item の値から「先頭が 'S' または 'E' ではじまり、1つ以上の連続した数字とアンダースコア、そして数字が続いた文字列、そして行末」という条件に合致した文字列を配列として受け取る。さらに、これを JOIN 関数の引数とし、コンマ区切りの文字列にする。 ここで $ は "行末" を表します。上記例における "E2010_0402 くばがさ次郎¥n"

[104] (*2) 対応する略号は次のとおりです。 "カ." "ユ." "メ." "シ." "キ." ".カ" ".ユ" ".メ" ".シ" ".キ"

	は、"E2010_0402" の後が行末ではなく、任意の文字が含まれているため条件に合致していないことがわかります。

MATCHRE1

MATCHRE1(検索文字列, 対象文字列, [インデックス], [グループ番号])

対象文字列から検索文字列に一致する部分文字列を返します。検索文字列にヒットする部分文字列が複数存在する場合は、インデックスを指定して任意の部分文字列を取得できます。(インデックスの開始番号は 1)

また、正規表現中に括弧で検索パターンをグループ化した場合、グループ番号を指定して、任意のグループの文字列を取得することができます。インデックスを省略した場合は 1、グループ番号を省略した場合は 0 を指定されたものとして扱います。

例：モデル内の body 項目の文字列内容が次の場合

```
顧客ID: A0001

ご担当者様へ

いつもお世話になっております。
Wagbyのライセンスについて質問があります。

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
カスタマーID : A0002

以上、よろしくお願いします。
```

例	動作
MATCHRE1("^(顧客|カスタマー)ID[ ]*:[ ]*(.*)$", ${body})	顧客ID: A0001
MATCHRE1("^(顧客|カスタマー)ID[ ]*:[ ]*(.*)$", ${body}, 0)	顧客ID: A0001
MATCHRE1("^(顧客|カスタマー)ID[ ]*:[ ]*(.*)$", ${body}, 1)	カスタマーID : A0002
MATCHRE1("^(顧客|カスタマー)ID[ ]*:[ ]*(.*)$", ${body}, 1, 0)	カスタマーID : A0002
MATCHRE1("^(顧客|カスタマー)ID[ ]*:[ ]*(.*)$", ${body}, 1, 1)	カスタマー
MATCHRE1("^(顧客|カスタマー)ID[ ]*:[ ]*(.*)$", ${body}, 1, 2)	A0002

MID

MID(文字列, 開始値, 文字数)

MID(文字列, 開始値)

文字列の任意の位置から指定された文字数の文字を返します。先頭文字の位置が 1 になります。戻り値の型は文字列型です。 第三引数（文字数）を省略した場合は、開始値以降のすべての文字列が返されます。

```
MID(${item1}, ${begin}, ${length})
```

例	動作
MID("abcde", 2, 3)	"bcd"
MID("abcde", 2)	"bcde"

NUMTOJTEXT

NUMTOJTEXT(数字, 区切りパターン, 文字種)

NUMTOJTEXT(数字を表す文字列, 区切りパターン, 文字種)

数字を日本語のテキストに変換します。戻り値の型は文字列です。

区切りパターンは次の4種類があります。

- 0 : 区切りパターンなし
- 1 : 3桁毎（千単位）
- 2 : 万および億単位
- 3 : 十、百、千、万および億単位

文字種パターンは次の4種類があります。

- 0 : 半角数字
- 1 : 全角数字
- 2 : 漢数字（一、二、三）
- 3 : 旧漢字（壱、弐、参）

例	返される文字列
NUMTOJTEXT(123456789, 0, 0)	"123456789"
NUMTOJTEXT(123456789, 1, 0)	"123,456,789"
NUMTOJTEXT(123456789, 2, 0)	"1億2345万6789"
NUMTOJTEXT(123456789, 3, 0)	"1億2千3百4十5万6千7百8十9"
NUMTOJTEXT(123456789, 0, 1)	"１２３４５６７８９"
NUMTOJTEXT(123456789, 1, 1)	"１２３，４５６，７８９"
NUMTOJTEXT(123456789, 2, 1)	"１億２３４５万６７８９"
NUMTOJTEXT(123456789, 3, 1)	"１億２千３百４十５万６千７百８十９"
NUMTOJTEXT(123456789, 0, 2)	"一二三四五六七八九"
NUMTOJTEXT(123456789, 1, 2)	"一二三,四五六,七八九"
NUMTOJTEXT(123456789, 2, 2)	"１億二三四五万六七八九"
NUMTOJTEXT(123456789, 3, 2)	"一億二千三百四十五万六千七百八十九"
NUMTOJTEXT(123456789, 3, 3)	"壱億弐千参百四拾五万六千七百八拾九"
NUMTOJTEXT("59234271", 3, 3)	"五千九百弐拾参万四千弐百七拾壱"

制約

- 動作する数値は13桁までとなっています。
- 国際化には未対応です。日本語のみを対象としています。

PADDING

PADDING(埋める1文字, 桁数, 文字列)

引数の文字列の前に「埋める1文字」を加えます。全体で「桁数」の範囲内の文字列にします。 「埋める1文字」はダブルクォートで囲みます。（二文字以上が指定された場合、先頭の文字が有効になります。）

```
PADDING(埋める1文字, 桁数, ${item1})
```

例	動作
PADDING("0", 6, ${item1})	"000123" ※ ${item1} の値が "123" の場合

PATTERN

PATTERN(入力文字列, パターン文字列)

入力文字列に、パターン文字列がいくつ含まれているかを返します。

```
PATTERN(${item1}, "東")
```

例	動作
PATTERN("東京と京都", "京")	2
PATTERN("東京と京都", "東京")	1

PROPER

PROPER(文字列)

アルファベットの 1 文字目を大文字に、2 文字目以降を小文字に変換します。戻り値の型は文字列型です。

```
PROPER(${item1})
```

例	動作
PROPER("hello, world.")	"Hello, world."

RANDSTR

RANDSTR(桁数)

RANDSTR(桁数,除外する文字列)

英大文字、小文字、数字から構成されるランダムな文字列を生成します。 第二引数は「除外する文字列」を指定します。この引数を省略した場合、除外する文字列として標準で "iIl10o0" が指定されます。これは似たような文字を除くことで視認性を高める措置です。[105]

```
RANDSTR(8)
```

例	動作
RANDSTR(8, "0123456789")	数字を除く 8 桁のランダムな文字列を生成します。
RANDSTR(8, "")	除外文字列を空と指定したため、すべての文字列を対象としたランダム文字列を生成します。

ANORMPART

ANORMPART(住所文字列, 構成要素)

住所正規化コンバータと連携することで、住所文字列から都道府県・市区町村などに分割された要素を取り出します。 具体的な使い方は"モデル項目 > 文字列 > 住所正規化コンバータとの連携"をお読みください。

```
ANORMPART(${address}, "PREF")
```

構成要素の部分は次の指定を行うことができます。（ダブルクォーテーションで構成要素名を囲ってください。）

構成要素の指定	説明
CTV_CODE	市町村コード
AREA_CODE	郵便番号
PREF	都道府県名
CTV	市町村名
AREA	町域名

[105] ある項目 A で RANDSTR() を適用し、別の項目 B が A の値を参照する場合、B の値と A の値は同じであることが保証されます。 ある項目 A で RANDSTR() を複数回用いた場合、最初の RANDSTR() の結果と二つ目以降の RANDSTR() の結果は同じ値となります。

KOAZA	小字・丁目
BANCHI_GO	番地号
AREA_NUMBER	小字・丁目+番地号
NAME	建物名
CLASS	建物階数
PREF_K	都道府県名(カナ)
CTV_K	市町村名(カナ)
AREA_K	町域名(カナ)
KOAZA_K	小字名(カナ)
C_CTV_CODE	市町村コード(候補)
C_AREA_CODE	郵便番号(候補)
C_PREF	都道府県名(候補)
C_CTV	市町村名(候補)
C_AREA	地名(候補)
C_PREF_K	都道府県名(カナ)(候補)
C_CTV_K	市町村名(カナ)(候補)
C_AREA_K	町域名(カナ)(候補)
C_KOAZA_K	小字名(カナ)(候補)
C_KOAZA	小字・丁目(候補)
C_BANCHI_GO	番地号(候補)
C_AREA_NUMBER	小字・丁目+番地号(候補)
C_NAME	建物名(候補)
C_CLASS	建物階数(候補)

・この関数は住所正規化コンバータ R5/R6 API オプションが必要です。 ・「候補」は入力された住所の解釈が複数見つかった場合に、次点の住所候補として設定されます。つまり候補は存在しないこともあります。 ・AREA_NUMBER の値は KOAZA の値と BANCHI_GO の値をハイフン (-) で結んだものと同じです。すなわち KOAZA と BANCHI_GO を一括りに扱いたい場合に利用できます。 ・住所情報の取り扱いの詳細については、住所正規化コンバータ製品に付属のマニュアルをお読みください。

REMOVEHTMLTAG

REMOVEHTMLTAG(対象文字列)

REMOVEHTMLTAG(対象文字列, 改行コード)

文字列中に含まれるHTMLタグを除去します。戻り値の型は文字列型です。 brタグは改行コードに置換されます。第二引数を省略した場合、改行コードは "¥r¥n" が利用されます。(Windows標準)。個別に "¥r" (Mac) や "¥n" (Linux) と指定することもできます。

```
REMOVEHTMLTAG(${richtextitem})
```

REPLACE

REPLACE(対象文字列, 開始位置, 文字数, [置換文字列])

文字列中の指定された文字数の文字を除去します。置換文字列は省略することができます。戻り値の型は文字列型です。

```
REPLACE(${target}, ${start}, ${len}, ${replace})
```

例	動作
s = "あいうえお" のとき REPLACE(s,1,1)	"いうえお"
s = "かきくけこ" のとき REPLACE(s,1,2)	"くけこ"
s = "さしすせそ" のとき REPLACE(s,2,2)	"させそ"

REPT

REPT(文字列, 繰り返し回数)

文字列を指定回数繰り返し表示します。戻り値の型は文字列型です。

```
REPT(${item1}, ${count})
```

例	動作
REPT("*", 5)	"*****"

RIGHT

RIGHT(文字列, [文字数])

文字列の末尾（右端）から指定された文字数の文字を返します。文字数を省略した場合は 1 になります。戻り値の型は文字列型です。

```
RIGHT(${item1}, ${num})
```

例	動作
RIGHT("abcde", 3)	"cde"

RPAD

RPAD(文字列, 指定桁数)

RPAD(文字列, 指定桁数, 埋め込み文字)

文字列の右側に、指定桁数になるまで埋め込み文字を挿入します。埋め込み文字の指定がない場合は、空白を挿入します。

```
RPAD(${item1}, 6, "0")
```

例	動作
RPAD("ABC", 6);	"ABC "
RPAD("ABC", 6, "0");	"ABC000"

RTRIM

RTRIM(文字列)

文字列の右側に含まれる余分なスペース（半角、全角）、タブ、改行文字を取り除きます。戻り値の型は文字列型です。

```
RTRIM(${item1})
```

例	動作
RTRIM("ABC ");	"ABC"

SLICE

SLICE(文字配列, インデックス)

引数の文字列配列の中から、インデックスで指定された要素を返します。戻り値は文字列型です。 入力の文字列配列が null、またはインデックスが文字列配列を超えた値を示した場合は、長さ0の文字列 "" を返します。

```
SLICE(${array_of_string_item}, 0)
```

例	動作
array = "北海道", "青森", "秋田" SLICE(array, 0)	"北海道"
array = "北海道", "青森", "秋田" SLICE(array, -1)	""
array = "北海道", "青森", "秋田" SLICE(array, 3)	""
array = null SLICE(array, 0)	""

SPLIT

SPLIT(対象文字列)

SPLIT(対象文字列, 区切り文字)

対象文字列を、区切り文字で分割します。区切り文字を省略したときは、半角スペースを区切り文字とします。戻り値の型は文字列の配列です。

区切られた個々の文字列には、LTRIMとRTRIMが適用されます。このため、前後の空白は削除されます。

```
SPLIT(${target})
SPLIT(${target}, ",")
```

例	動作
SPLIT(" a , bc, d , efg ", ",")	abcdefg

SPLITと対になる関数がJOINです。

区切り文字を複数指定する

区切り文字を複数指定することができます。次の例は「,」「;」のいずれも区切り文字として扱います。

```
SPLIT(${target}, ",;")
```

SUBSTITUTE

SUBSTITUTE(対象文字列, 検索文字列, 置換文字列)

文字列中から検索文字列を探し、置換文字列に置き換えます。戻り値の型は文字列型です。

```
SUBSTITUTE(${target}, ${str}, ${replace})
```

例	動作
s = "沖縄県那覇市" のとき LEN(s)- LEN(SUBSTITUTE(s, "県", ""))	1
s = "沖縄県沖縄市" のとき LEN(s)- LEN(SUBSTITUTE(s, "沖", ""))	2

STR2BYTEA (8.3.1)

STR2BYTEA(対象文字列、文字コード、固定長サイズ)

STR2BYTEA(対象文字列、文字コード、固定長サイズ、固定長に満たないときに埋める文字)

対象文字列を、指定した文字コードの固定長サイズに変換します。固定長サイズに満たないときの動作は次の通りです。

- 第四引数がない場合、対象文字列をそのまま指定の文字コードでバイト配列にした状態とします。例えば対象文字列が1文字(2バイト)で、固定長サイズを10とした場合、2バイトの配列が返戻されます。
- 第四引数が指定されていた場合、不足分を指定した文字で埋めます。例えば対象文字列が1文字(2バイト)で、固定長サイズを10とした場合、10バイトの配列が返戻されます。うち8バイトは、指定された文字(バイト)で埋められています。

```
STR2BYTEA(${item1}, "MS932", 10)

STR2BYTEA(${item1}, "MS932", 10, 0x20)
```

この関数の利用例に "固定長ファイルのダウンロードとアップロード更新" があります。

ASC

ASC(入力文字列)

全角の文字列を半角の文字に変換します。 具体的には「アルファベット」「数字」「記号」「カタカナ」が対象になります。

TEXT

TEXT(数値)

TEXT(数値, 書式)

TEXT(日付, 書式)

数値（または日付のシリアル値）および日付型を書式設定した文字列に変換します。

第二引数の書式を指定しなかった場合は、TOSTR関数と同じ動きになります。

```
TEXT(${item1}, "yyyy-MM-dd")
```

TEXT(1234, "#,###")	"1,234"
TEXT(3.33333, "0.0")	"3.3"
TEXT(0.1234, "0.0%")	"12.3%"
TEXT(0.00001, "0.0E0")	"1.0E-5"
TEXT(1234, "¥¥#,####")	"¥¥1,234"
TEXT(${date}, "yyyy-MM-dd");	"2000-01-01" ※ ${date} の値が 2000-1-1 の場合。
TEXT(${date}, "yyyy年M月d日")	"2000年1月1日" ※ ${date} の値が 2000-1-1 の場合。
TEXT(${date}, "{0}{1,number,integer}年{2,number,integer}月{3,number,integer}日")	"平成28年1月1日" ※ ${date} の値が 2016-1-1 の場合。（詳細は「和暦についての補足」をお読みください。）
TEXT(${time}, "HH:mm")	"15:05" ※ ${time} の値が 15:05:55 の場合。時刻を24時間表記するときは大文字の"H"を使います。[106]
TEXT(${time}, "h:m:s")	"3:5:55" ※ ${time} の値が 15:05:55 の場合。
TEXT(${time}, "ahh:mm")	"午後12:00" ※ ${time} の値が 00:00:00 の場合。"a"によって"午前/午後"を表示できます。

[106] Excel には 24 時間以上を表記するための指定 "[h]:mm" がありますが、Wagby ではサポートしていません。

和暦についての補足

和暦書式のルールは次のとおりです。

指定方法	説明
{0} または {5}	年号(明治、大正、昭和、平成)
{6}	年号(明、大、昭、平)
{7}	年号(M、T、S、H)
{1,number,integer}	年
{2,number,integer}	月
{3,number,integer}	日

応用例 : 日付の「年度」部分を "Hxx" 形式の文字列として取得する

FISCALYEAR関数と組み合わせます。item1 を日付型項目とします。

```
TEXT(DATEVALUE(CONCATENATE(TOSTR(FISCALYEAR(${item1})),"-1-1")),"{7}{1,number,integer}")
```

TOBYTE

TOBYTE(文字列)

文字列表現された数字を1バイト整数へ変換します。具体的には "1" という文字列を 1 へ変換します。

TOBYTE_ARRAY

TOBYTE_ARRAY(文字列の配列)

文字列表現された数字の配列を1バイト整数の配列へ変換します。具体的には ["1"] という文字配列を1バイト整数配列の [1] へ変換します。

TOSHORT

TOSHORT(文字列)

文字列表現された数字を2バイト整数へ変換します。

TOSHORT_ARRAY

TOSHORT_ARRAY(文字列の配列)

文字列表現された数字の配列を2バイト整数の配列へ変換します。具体的には ["1"] という文字配列を2バイト整数配列の [1] へ変換します。

TOINT

TOINT(文字列)

文字列表現された数字を整数型 (4バイト整数) へ変換します。

TOINT_ARRAY

TOINT_ARRAY(文字列の配列)

文字列表現された数字の配列を4バイト整数の配列へ変換します。具体的には［"1"］という文字配列を4バイト整数配列の［1］へ変換します。

TOLONG

TOLONG(文字列)

文字列表現された数字を8バイト整数へ変換します。

TOLONG_ARRAY

TOLONG_ARRAY(文字列の配列)

文字列表現された数字の配列を8バイト整数の配列へ変換します。具体的には［"1"］という文字配列を8バイト整数配列の［1］へ変換します。

TOFLOAT

TOFLOAT(文字列)

文字列表現された数字を4バイト浮動小数へ変換します。

CONCATENATE

CONCATENATE(文字列1, 文字列2, 文字列3,...)

引数の文字列を連結した文字列を返します。 引数の文字列は、コンマ区切りで複数個、記述することができます。（個数の制限はありません）[107]

```
CONCATENATE(${str1}, ${str2})
```

例	動作
CONCATENATE("A")	"A"
CONCATENATE("A", "B")	"AB"
CONCATENATE("A", "", null, "B", "", "C")	"ABC"

TOFLOAT_ARRAY

TOFLOAT_ARRAY(文字列の配列)

文字列表現された数字の配列を4バイト浮動小数の配列へ変換します。具体的には ［"1.0"］ という文字配列を4バイト浮動小数配列の［1.0］へ変換します。

TODOUBLE

TODOUBLE(文字列)

文字列表現された数字を8バイト浮動小数へ変換します。

TODOUBLE_ARRAY

[107] CONCATENATEの引数に繰り返し項目や繰り返しコンテナ項目を指定することはできません。代わりにJOINを使うことができます。

TODOUBLE_ARRAY(文字列の配列)

文字列表現された数字の配列を8バイト浮動小数の配列へ変換します。具体的には ["1.0"] という文字配列を8バイト浮動小数配列の [1.0] へ変換します。

TOSTR

TOSTR(数値)

TOSTR(日付)

数値型/日付型の値を文字列型に変換します。TEXT関数と異なり、書式指定を行うことはできません。 計算結果を文字型項目にセットするといった目的に使うことができます。

```
TOSTR(${num}+1)
```

TOSTR_ARRAY

TOSTR_ARRAY(数値型の配列)

基本型配列を文字列型配列に変換します。繰り返し項目や、チェックボックス項目などに適用できます。

TRIM

TRIM(文字列)

文字列から余分なスペース（半角、全角）、タブ、改行文字を取り除きます。先頭の空白はすべて削除されます。文字間の空白は 1 個残します。戻り値の型は文字列型です。[108]

```
TRIM(${item1})
```

例	動作
TRIM(" ABC");	"ABC"
TRIM(" A B C ")	"A B C"
TRIM(" A B C ")	"A B C"

UPPER

UPPER(文字列)

アルファベットの小文字を大文字に変換します。戻り値の型は文字列型です。

```
UPPER(${item1})
```

例	動作
UPPER("hello, world.");	"HELLO, WORLD."

VALUE

VALUE(文字列)

VALUE(文字列, デフォルト値)

[108] 処理の流れは LTRIMとRTRIM を行い、かつ、複数の空白文字を 1 文字にまとめるものです。

数値を表す文字列を数値に変換します。日付、時間を表す文字列は日付（時間）シリアル値に変換します。戻り値の型は 8 バイト浮動小数点型です。

第一引数の文字列が　　null　　または解析不可能な値の場合、デフォルト値が用いられます。省略時は　Double.NAN（不定）となります。デフォルト値は第二引数で与えることができます。型は8バイト浮動小数です。

```
VALUE(${item1})

VALUE(${item1}, 0.0)
```

VALUE("0.12")	0.12
VALUE("1,234")	1234
VALUE("1,234.567")	1234.567
VALUE("1,23E-4")	0.000123
VALUE("12%")	0.12
VALUE("15:30")	0.645833
VALUE("2000-01-01 12:34:56")	36526.52426
VALUE("2000年1月1日")	36526

VALUE_ARRAY

VALUE_ARRAY(文字列の配列)

VALUEと同じ動作をしますが、引数には文字列の配列を渡します。戻り値の型は 8 バイト浮動小数点型の配列になります。

COUNT

COUNT(文字列配列)

引数の文字列配列の長さを返します。引数が null の場合は 0 を返します。

```
COUNT(${array_of_string_item})
```

EXACT

EXACT(文字列1, 文字列2)

二つの文字列の比較を行います。同一なら真(true)を返します。戻り値の型は真偽値(trueまたはfalse)となるため、IF関数の内部などで利用できます。

```
EXACT(${item1}, ${item2})
```

例	動作
EXACT("ABC", "ABC")	true
EXACT("ABC", "ABD")	false

EXACT(日付1, 日付2)

二つの日付の比較を行います。同一なら真(true)を返します。戻り値の型は真偽値(trueまたはfalse)となるため、IF関数の内部などで利用できます。

```
EXACT(${date1}, ${date2})
```

Wagbyは内部で Java を使います。Java では文字列および日付の一致判定を「==」演算子で行うことはできません。例えば item1 == "山田" という式は常にfalseとなります。この問題を回避するため、Wagbyでは EXACT 関数を用意しています。文字列および日付の一致判定は必ず EXACT 関数を用いるようにしてください。

FIND

FIND(検索文字列, 対象, [開始位置])

検索文字列が先頭から何番目にあるかを返します。戻り値の型は整数型です。1 から数えます。 見つからなかった場合は -1 を返します。

開始位置は省略できます。省略時の開始位置は 1 となります。 英字の大文字と小文字は区別します。ワイルドカードを使うことはできません。

```
FIND(${item1}, ${target}, ${pos})
```

例	動作
FIND("市", "沖縄県うるま市州崎5番地1", 4)	7
FIND("う", "沖縄県うるま市州崎5番地1")	4

FINDTAIL

FINDTAIL(検索文字列, 対象)

検索文字列が対象文字列の末尾から何番目にあるかを返します。末尾からみて、最初にみつかった文字の位置を返します。 戻り値の型は整数型です。 見つからなかった場合は -1 を返します。

```
FINDTAIL(${item1}, ${target})
```

例	動作
文字列型の項目 urlitem の値が「file:///// server001/fileshare/catalog001.doc」 のとき IF(NOT(ISBLANK(${urlitem})), MID(${urlitem},FINDTAIL("/", ${urlitem})+1),"")	catalog001.doc
文字列型の項目 urlitem の値が「file:///// server001¥fileshare¥catalog001.doc」 のとき IF(NOT(ISBLANK(${urlitem})), MID(${urlitem},FINDTAIL("¥¥", ${urlitem})+1),"")	catalog001.doc

FINDRE

FINDRE(検索文字列, 対象, [開始位置])

検索文字列が先頭から何番目にあるかを返します。戻り値の型は整数型です。 検索文字列部分に正規表現が指定できます。（正規表現の適用ルールは MATCHRE と同等です。）

開始位置は省略できます。 英字の大文字と小文字は区別します。

```
FINDRE(${item1}, ${target}, ${pos})
```

例	動作
FINDRE("w.+gbyR[0-9]¥¥.[0-9]¥¥.[0-9]","あいうえお12345w3gbyR8.abc")	11
FINDRE("^[A-Za-z0-9!¥"?$&()^~¥¥[¥¥].{};:¥¥¥+_/-]{8,}$", ${passwd}) > 0	passwd項目に8文字以上の A-Za-z0-9!"? $&()^~{};:+_/-¥ のいずれかの文字のみが指定されているかどうかを判定する。R8.4.0 以上で利用できる正規表現。
FINDRE("^[A-Za-z0-9!¥"¥"?$&()^~¥¥[¥¥].{};:¥¥¥+_/-]{8,}$", ${passwd}) > 0	passwd項目に8文字以上の A-Za-z0-9!"? $&()^~{};:+_/-¥ のいずれかの文字のみが指定されているかどうかを判定する。R8.3.2 までで利用できる正規表現。(ダブルクォーテーションを2つ指定しているのは、Designerの入力チェッ

	クを回避するため。R8.4.0では、このような回避策は不要となった。）

日付時刻関数

EDATE

EDATE(日付, 月数)

指定した月数分だけ前後した日付を返します。戻り値の型は日付型です。

```
EDATE(${date1}, 3)
```

EOMONTH

EOMONTH(日付)

月末の日付を返します。請求日項目に対する支払い期限の設定などに利用できます。戻り値の型は日付型です。

```
EOMONTH(${date1})
```

EOMONTH(日付, 月数)

月末の日付を返します。指定した月数分だけ前後させます。（第二引数に負数を指定することで、前に移動させることができます。）戻り値の型は日付型です。

```
EOMONTH(${date1}, 1)
```

FISCALYEAR

FISCALYEAR(日付)

入力日付（日付型）に対する「年度」を返します。戻り値の型は整数型です。

```
FISCALYEAR(${date1})
```

例	動作
FISCALYEAR(DATEVALUE("2009-1-1"))	2008
FISCALYEAR(DATEVALUE("2009-3-31"))	2008
FISCALYEAR(DATEVALUE("2009-4-1"))	2009

HOUR

HOUR(日付)

引数の日付型の値から、時間を求めます。

```
HOUR(${date1})
```

MINUTE

MINUTE(日付)

引数の日付型の値から、分を求めます。

```
MINUTE(${date1})
```

MONTH

MONTH(日付)

引数の日付型の値から、月を求めます。（月は 1 から 12 までの範囲です。）戻り値の型は整数型です。

```
MONTH(${date1})
```

MOVEDAY

MOVEDAY(日付, 日数)

指定した日数分だけ前後した日付を返します。（負数を与えると前に戻ります。） 戻り値の型は日付型です。

```
MOVEDAY(${dateitem}, 1)
```

MOVEHOUR

MOVEHOUR(日付時刻, 時)

指定した時間（時）だけ前後した日付時刻を返します。（負数を与えると前に戻ります。） 戻り値の型は日付時刻型です。

```
MOVEHOUR(${datetimeitem}, 1)
```

MOVEMINUTE

MOVEMINUTE(日付時刻, 分)

指定した時間（分）だけ前後した日付時刻を返します。（負数を与えると前に戻ります。） 戻り値の型は日付時刻型です。

```
MOVEMINUTE(${datetimeitem}, 1)
```

MOVEMONTH

MOVEMONTH(日付, 月数)

指定した月数分だけ前後した日付を返します。（負数を与えると前に戻ります。） 戻り値の型は日付型です。 この関数は EDATE と同じ動作となります。

```
MOVEMONTH(${dateitem}, 1)
```

BETWEENTIME

BETWEENTIME(日付型, 日付型)

二つの日付型項目の時間差をとります。戻り値の型は時刻型です。なお、24 時間を超過した分は切り捨てられます。

```
BETWEENTIME(${date1}, ${date2})
```

BETWEENTIME(時刻型, 時刻型)

二つの時刻型項目の時間差をとります。戻り値の型は時刻型です。なお、24 時間を超過した分は切り捨てられます。

```
BETWEENTIME(${time1}, ${time2})
```

BETWEENTIME(日付時刻型, 日付時刻型)

二つの日付時刻型項目の時間差をとります。戻り値の型は時刻型です。なお、24 時間を超過した分は切り捨てられます。

```
BETWEENTIME(${datetime1}, ${datetime2})
```

BETWEENTIME(文字型, 文字型)

文字列表記された二つの日付型項目の時間差をとります。戻り値の型は時刻型です。なお、24 時間を超過した分は切り捨てられます。

```
BETWEENTIME(${datestr1}, ${datestr2})
```

MOVESECOND

MOVESECOND(日付時刻, 秒)

指定した時間（秒）だけ前後した日付時刻を返します。（負数を与えると前に戻ります。） 戻り値の型は日付時刻型です。

```
MOVESECOND(${datetimeitem}, 1)
```

MOVEYEAR

MOVEYEAR(日付, 年数)

指定した年数分だけ前後した日付を返します。（負数を与えると前に戻ります。） 戻り値の型は日付型です。

```
MOVEYEAR(${dateitem}, 1)
```

NETWORKDAYS

NETWORKDAYS(開始日,終了日)

指定された2つの日付の期間に含まれる稼働日(土・日・祝日を除外した営業日)の日数を返します。

休日はWagbyに標準で同梱されている休日モデル(jholiday)を利用します。

NOW

NOW()

今日の日付時刻を返します。戻り値は日付時刻型です。[109]

[109] 同一モデル内の複数項目に NOW() を適用した場合、すべての NOW() の戻り値は同じとなることが保証されます。（描画のタイミングによって、ある項目の NOW() は 00 秒で、別の項目の NOW() は 01 秒になるということはありません。）

```
NOW()
```

SECOND

SECOND(日付)

引数の日付型の値から、秒を求めます。

```
SECOND(${date1})
```

TIME

TIME(時, 分, 秒)

指定された時刻に対応する時間シリアル値（小数）を返します。これは「1」を 24 時間とみなした小数値です。戻り値の型は 8 バイト浮動小数点型です。 23 を超える時間指定は、24 で徐算された剰余が用いられます。 59 を超える分指定は、時と分に変換されます。 59 を超える秒指定は、時、分、秒に変換されます。

```
TIME(${hour}, ${minute}, ${second})
```

TIME(時刻型)

時刻型項目の値を、時間シリアル値（小数）に変換します。戻り値の型は 8 バイト浮動小数点型です。

```
TIME(${timevalue})
```

TIMEVALUE

TIMEVALUE(時刻を表す文字列)

時刻を表す文字列を時刻型に変換します。"HH:mm" や "HH:mm:ss" といった表記が利用できます。戻り値の型は時刻型です。

```
TIMEVALUE(${timestr})
```

TODATE

TODATE(日付時刻型)

日付時刻型を日付型に変換します。

```
TODATE(${datetime})
```

TODAY

TODAY()

今日の日付を返します。戻り値は「日付型」です。 (110)

```
TODAY()
```

(110) 同一モデル内の複数項目に TODAY() を適用した場合、すべての TODAY() の戻り値は同じとなることが保証されます。（描画のタイミングによって、ある項目の TODAY() は 1月1日で、別の項目の TODAY() は 1月2日になるということはありません。）

TOTIME

TOTIME(日付時刻型)

日付時刻型を時刻型に変換します。

```
TOTIME(${datetime})
```

BOMONTH

BOMONTH(日付)

月初の日付を返します。戻り値の型は日付型です。

```
BOMONTH(${date1})
```

BOMONTH(日付, 月数)

月初の日付を返します。指定した月数分だけ前後させます。（第二引数に負数を指定することで、前に移動させることができます。）戻り値の型は日付型です。

```
BOMONTH(${date1}, 1)
```

WEEKDAY

WEEKDAY(日付)

日付を曜日に対応する数値に変換します。日曜日が 1 で、土曜日が 7 になります。戻り値の型は整数型です。

```
WEEKDAY(${date1})
```

WEEKDAY(日付, 種類)

日付を曜日に対応する数値に変換します。戻り値の型は整数型です。 種類に 1 を指定すると、日曜日が 1 で、土曜日が 7 になります。 種類に 2 を指定すると、月曜日が 1 で、日曜日が 7 になります。 種類に 3 を指定すると、月曜日が 0 で、日曜日が 6 になります。

引数の日付が null の場合は、-1 を返します。

```
WEEKDAY(${date1}, 1)
```

例	動作
WEEKDAY(d,1)	4
WEEKDAY(d,2)	3
WEEKDAY(d,3)	2

WEEKNUM

WEEKNUM(日付)

日付がその年の第何週目に当たるかを返します。週の始まりを日曜日とします。戻り値の型は整数型です。

```
WEEKNUM(${date1})
```

WORKDAY

WORKDAY(開始日,日数)

ある日付から、指定された日数だけ前後した日付を返します。土・日・祝日を除外し、営業日だけで計算します。戻り値の型は日付型です。

休日はWagbyに標準で同梱されている休日モデル(jholiday)を利用します。

YEAR

YEAR(日付)

引数の日付型の値から、年を求めます。戻り値の型は整数型です。

```
YEAR(${date1})
```

CLEARHMS

CLEARHMS(日付)

日付の時分秒をゼロクリアします。

```
CLEARHMS(${dateitem})
CLEARHMS(TODAY())
```

DATE

DATE(日付)

DATE(年,月,日)

年月日で表現される日付を、シリアル値に変換します。ここでシリアル値とは、1900年1月1日を起点（1）とした経過日数です。戻り値の型は整数型です。

例：日付型項目のシリアル値を求める

```
DATE(${date1})
```

例	動作
DATE(1900,1,1)	1
DATE(2000,1,1)	36526

なお Excel は 1900 年をうるう年とみなすという仕様です。この関数もその仕様に合わせています。

この関数を応用して、二つの日付の差を求めることができます。具体的には次のように記述します。
DATE(${date1})-DATE(${date2})

DATEDIF

DATEDIF(日付1, 日付2, 単位)

開始日と終了日との日数差を調べて、指定した単位で表示します。単位は "Y"（年数）, "M"（月数）, "D"（日数）, "YM"（年数表示での端数の月数）, "YD"（年数表示での端数の日数）, "MD"（月数表示での端数の日数）です。戻り値の型は整数型です。

```
DATEDIF(${date1}, ${date2}, "Y")
```

例	動作
DATEDIF(${birth}, TODAY() ,"Y")	37 (生年月日が 1970/12/9 で、TODAY() が 2008/2/25 の場合)
DATEDIF(${birth}, TODAY() ,"YM")	2 (生年月日が 1970/12/9 で、TODAY() が 2008/2/25 の場合)

この関数を使うと、「何年何ヶ月」という自動計算を行わせることができます。具体的には次のような式になります。 DATEDIF(${birth}, TODAY() ,"Y") + "年" + DATEDIF(${birth}, TODAY() ,"YM") + "ヶ月"

単位が "D", "YD", "MD" の場合、Excel とは違った値を返すことがあります。これは Excel 側の不具合であることが知られていますので、まったく同じにならないことに留意してご使用ください。

DATESTRING

DATESTRING(日付)

日付型の値を和暦に変換します。和暦の書式は「(元号)NN年M月D日」です。戻り値の型は文字列型です。

```
DATESTRING(${date1})
```

DATETIMEVALUE

DATETIMEVALUE(日付時刻を表す文字列)

日付時刻を表す文字列を日付時刻型に変換します。"yyyy-MM-dd HH:mm:ss" および "yyyy/MM/dd HH:mm:ss" 表記が利用できます。戻り値の型は日付時刻型です。

```
DATETIMEVALUE(${timestr})
```

DATEVALUE

DATEVALUE(日付を表す文字列)

日付を表す文字列を日付型に変換します。"yyyy-MM-dd" や "yyyy/MM/dd" といった表記が利用できます。戻り値の型は日付型です。

```
DATEVALUE(${datestr})
```

DAY

DAY(日付)

引数の日付型の値から、日を求めます。 (月は 1 から 31 までの範囲です。) 戻り値の型は整数型です。

```
DAY(${date1})
```

集合処理

DJOIN

DJOIN(繰り返し項目, [連結文字])

DJOIN(繰り返しコンテナ項目, [連結文字])

DJOIN(外部キー子モデル項目, [連結文字])

DJOIN(チェックボックス型の項目, [連結文字])

連結対象項目の重複を許さない join です。

例えば、連結対象の文字列項目が "aaa", "bbb", "aaa", "ccc" のとき、区切り文字「,」で連結すると "aaa,bbb,ccc" という文字列が生成されます。

DDJOIN

DDJOIN(繰り返し項目, [連結文字])

DDJOIN(繰り返しコンテナ項目, [連結文字])

DDJOIN(外部キー子モデル項目, [連結文字])

DDJOIN(チェックボックス型の項目, [連結文字])

連結対象項目の重複を許さない join です。さらに、連結対象項目内に区切り文字があれば、それを分割して重複チェックを行います。

例えば、連結対象の文字列項目が "a,b,c", "a,b,d" のとき、区切り文字「,」で連結すると "a,b,c,d" という文字列が生成されます。

RANK

RANK(繰り返し項目, 繰り返し項目(範囲), [順序])

RANK(繰り返しコンテナ項目, 繰り返しコンテナ項目(範囲), [順序])

RANK(外部キー子モデル項目, 外部キー子モデル項目(範囲), [順序])

引数の集合の順位を返します。第一引数が対象項目、第二引数が範囲を示す項目ですが、通常は第一引数と第二引数の項目名は同じものになります。第三引数の順序は省略できます。省略時は大きい方からの順（降順）となります。"1" を指定すると昇順になります。

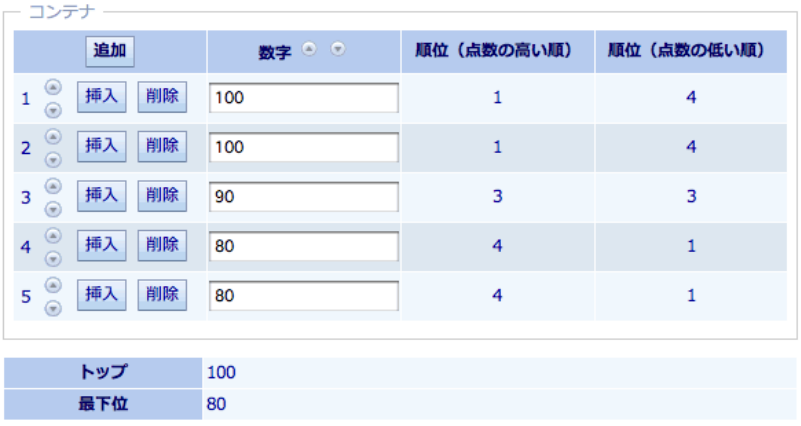


図 RANK関数の指定例

例：繰り返しコンテナ項目 staff/tensuu を対象とする。昇順とする。

```
RANK(${staff/tensuu}, ${staff/tensuu}, 1)
```

例：外部キー子モデルとして紐づいている staff モデルの tensuu 項目を対象とする。降順とする。

```
RANK(${staff_lp.tensuu}, ${staff_lp.tensuu})
```

LARGE

LARGE(繰り返し項目, 順位)

LARGE(繰り返しコンテナ項目, 順位)

LARGE(外部キー子モデル項目, 順位)

大きい方からの順位を返します。第二引数の順位は数値（一位,二位..）を指定します。RANK関数の利用例における「トップ」の項目が相当します。

例：繰り返しコンテナ項目 staff/tensuu を対象

```
LARGE(${staff/tensuu}, 1)
```

例：外部キー子モデルとして紐づいている staff モデルの tensuu 項目を対象

```
LARGE(${staff_lp.tensuu}, 1)
```

SMALL

SMALL(繰り返し項目, 順位)

SMALL(繰り返しコンテナ項目, 順位)

SMALL(外部キー子モデル項目, 順位)

小さい方からの順位を返します。第二引数の順位は数値（一位,二位..）を指定します。RANK関数の利用例における「最下位」の項目が相当します。

例：繰り返しコンテナ項目 staff/tensuu を対象

```
SMALL(${staff/tensuu}, 1)
```

例：外部キー子モデルとして紐づいている staff モデルの tensuu 項目を対象

```
SMALL(${staff_lp.tensuu}, 1)
```

GCD

GCD(繰り返し項目)

GCD(繰り返しコンテナ項目)

GCD(外部キー子モデル項目)

GCD(整数型の配列)

引数の整数型配列から、最大公約数を返します。戻り値は整数型です。

例：以下の式は "6" を返します。

```
GCD(ARRAY_INT(12,18))
```

例：繰り返しコンテナ項目 cont/value を対象。型は整数型のみ。

```
GCD(${cont/value})
```

LCM

LCM(繰り返し項目)

LCM(繰り返しコンテナ項目)

LCM(外部キー子モデル項目)

LCM(整数型の配列)

引数の整数型配列から、最小公倍数を返します。戻り値は整数型です。

例：以下の式は "6" を返します。

```
LCM(ARRAY_INT(2,3))
```

例：繰り返しコンテナ項目 cont/value を対象。型は整数型のみ。

```
LCM(${cont/value})
```

ARRAY

ARRAY_INT(整数型の数値...)

ARRAY_LONG(8バイト整数型の数値...)

ARRAY_FLOAT(4バイト浮動小数点型の数値...)

ARRAY_DOUBLE(8バイト浮動小数点型の数値...)

ARRAY(文字列...)

引数の整数型数値（コンマ区切りで複数個、記述できます）から、整数型の配列を生成します。 MAX, MIN 関数と組み合わせることで、最大値や最小値を返すことができます。

引数がない場合は、空の配列を生成します。ARRAY_INT(), ARRAY_LONG(), ARRAY_FLOAT(), ARRAY_DOUBLE(), ARRAY() が利用できます。

例：

```
MAX(ARRAY_INT(10, 20, 30))
```

文字列項目を、文字列の配列へと変換することもできます。

例：

```
ARRAY(USERNAME())

ARRAY(${item1},${item2},${item3})
```

ARRAY_OBJ (8.3.0)

ARRAY_OBJ(数値型配列)

基本型の数値型配列を、オブジェクトの配列に変換します。戻り値の型は基本型のラッパークラスとなります。例えばint配列を引数に渡すと、Integer配列が返ります。

SORT

SORT(数値型配列)

SORT(文字列型配列)

引数の配列を昇順で並び替えます。戻り値の型は、引数と同じ型です。

COUNT

COUNT(繰り返し項目, [条件])

COUNT(繰り返しコンテナ名, [条件])

COUNT(外部キー子モデル名, [条件])

レコードの総数を求めます。計算結果を格納する項目の型は「整数」としてください。　条件に合致するデータが存在しなかった場合、"0" 件と表示されます。[制約...]

例：繰り返し項目 number の件数を求める。

```
COUNT(${number})
```

例：繰り返しコンテナ report の件数を求める。

```
COUNT(${report})
```

(111)

例：外部キー子モデル support の件数を求める。(外部キーの指定方法の詳細...)

```
COUNT(${support_lp})
```

条件の指定

二重引用符（ダブルクォーテション）の中に、条件式を記述します。項目名は${...} で囲みます。

例：繰り返し項目 number について、正の数となっている件数を求める。

```
COUNT(${number}, "${number}>0")
```

例：繰り返しコンテナ report の件数を求める。ただし同コンテナ内の項目である report/date が今年のもののみとする。

```
COUNT(${report}, "YEAR(${report/date})==YEAR(TODAY())")
```

例：外部キー子モデル support の項目 kingaku の件数を求める。ただし子モデルの項目 name に値がセットされているもののみとする。(外部キーの指定方法の詳細...)

```
COUNT(${support_lp}, "${support_lp.name}!=null")
```

例：条件部にモデル参照項目を指定する。ID 部で判断する場合。

```
COUNT(${report}, "${report/type#id}==1")

COUNT(${support_lp}, "${support_lp.type#id}==1")
```

(111) コンテナ名だけを指定します。コンテナ内のいずれの項目名も含める必要はありません。

例：条件部にモデル参照項目を指定する。内容部で判断する場合は文字列一致比較のため EXACT 関数と併用します。[112]

```
COUNT(${report}, "EXACT(${report/type#content}, "男")")
COUNT(${support_lp}, "EXACT(${support_lp.type#content}, "男")")
```

COUNT 関数は "自分のモデルの検索結果" を扱うものではありません。この場合は RESULTSIZE 関数を利用します。

REVERSE

REVERSE(数値型配列)

REVERSE(文字列型配列)

引数の配列を降順で並び替えます。戻り値の型は、引数と同じ型です。

CONTAINS

CONTAINS(数値型配列, 数値)

CONTAINS(文字列型配列, 文字列)

第一引数の配列に、第二引数の値が含まれているかどうかを返します。戻り値は boolean 型（真/偽）です。

ISEMPTY

ISEMPTY(配列)

第一引数の配列が空かどうかを返します。戻り値は boolean 型（真/偽）です。

[113]

ADD

ADD(配列, 値)

第一引数の配列の末尾に、値を加えた新しい配列を生成します。戻り値は配列です。数値型、文字列型、日付型で利用できます。

```
ADD(${array},1)
```

項目arrayが繰り返し項目で値 "1,0" を持っている場合、"1,0,1" という配列を新たに生成します。

GET

GET(配列, 添字)

第一引数の配列の「何番目」の値を返します。配列の添字は0から数えます。第二引数の添字が不適切の場合（例：添字が負数、添字が配列の大きさを超えている）、添字は 0 として扱われます。すなわち、配列の0番目の値を返します。

---

(112) 参照連動項目には対応していません。

(113) ISEMPTY(項目名)という使い方もできます。

INDEXOF

INDEXOF(配列, 見つける値)

第一引数の配列中に、第二引数で指定した値が含まれている場合、先頭から何番目に含まれているかを返します。配列の添字は0から数えます。見つからなかった場合は -1 を返します。

LASTINDEXOF

LASTINDEXOF(配列, 見つける値)

第一引数の配列中に、第二引数で指定した値が含まれている場合、末尾から何番目に含まれているかを返します。配列の添字は0から数えます。見つからなかった場合は -1 を返します。

ISEQUALS

ISEQUALS(配列, 配列)

第一引数の配列と、第二引数の配列の値が同じかどうかを調べます。戻り値は boolean 型（真/偽） です。数値型、文字列型、日付型で利用できます。

ISSAMELENGTH

ISSAMELENGTH(配列, 配列)

第一引数の配列と、第二引数の配列の要素数が同じかどうかを調べます。戻り値は boolean 型（真/偽） です。数値型、文字列型、日付型で利用できます。（配列がnullの場合、要素数は0個として比較します。）

REMOVE

REMOVE(配列, 添字)

第一引数の配列から、何番目の要素（添字として表現された数値）を削除した配列を返します。配列の添字は0から数えます。数値型、文字列型、日付型で利用できます。

SUM

SUM(繰り返し項目, [条件])

SUM(繰り返しコンテナ項目, [条件])

SUM(外部キー子モデル項目, [条件])

指定した集合値の加算を行います。 計算結果を格納する項目の型、ならびに計算対象項目の型は「整数」（あるいはその他、数値を表現する型）としてください。[制約...]

例：繰り返し項目 number の合計を求める。

```
SUM(${number})
```

例：繰り返しコンテナ report/number の合計を求める。

```
SUM(${report/number})
```

例：外部キー関連モデル support の項目 kingaku の合計を求める。(外部キーの指定方法の詳細...)

```
SUM(${support_lp.kingaku})
```

条件の指定

二重引用符（ダブルクォーテション）の中に、条件式を記述します。項目名は${...} で囲みます。

例：繰り返し項目 number について、正の数の合計を求める。

```
SUM(${number}, "${number}>0")
```

例：繰り返しコンテナ report/number の合計を求める。ただし同コンテナ内の項目である report/date が今年のもののみとする。

```
SUM(${report/number}, "YEAR(${report/date})==YEAR(TODAY())")
```

例：外部キー子モデル support の項目 kingaku の合計を求める。ただし子モデルの項目 name に値がセットされているもののみとする。(外部キーの指定方法の詳細...)

```
SUM(${support_lp.kingaku}, "${support_lp.name}!=null")
```

例：条件部にモデル参照項目を指定する。ID 部で判断する場合。

```
SUM(${report/number}, "${report/type#id}==1")

SUM(${support_lp.kingaku}, "${support_lp.type#id}==1")
```

例：条件部にモデル参照項目を指定する。内容部で判断する場合は文字列一致比較のため EXACT 関数と併用します。

(114)

```
SUM(${report/number}, "EXACT(${report/type#content}, "男")")

SUM(${support_lp.kingaku}, "EXACT(${support_lp.type#content}, "男")")
```

演算精度 (8.2.0)

R8.2.0より集合関数 SUM は java.math.BigDecimal を使います。

REMOVEELEMENT

REMOVEELEMENT(配列, 削除する値)

第一引数の配列から、最初にみつかった「削除する値」を除いた配列を返します。指定された値が含まれていなかった場合は、第一引数の配列と同じ配列が戻り値になります。削除される要素は1つだけです。

例えば配列の要素が "1,2,3,2,1" で、削除する値が "2" のとき、戻り値の配列は "1,3,2,1" になります。

REMOVEELEMENTALL

REMOVEELEMENTALL(配列, 削除する値)

第一引数の配列から、みつかった「削除する値」をすべて除いた配列を返します。指定された値が含まれていなかった場合は、第一引数の配列と同じ配列が戻り値になります。

例えば配列の要素が "1,2,3,2,1" で、削除する値が "2" のとき、戻り値の配列は "1,3,1" になります。

UBOUND

UBOUND(配列)

配列の大きさ（サイズ）を返します。戻り値は整数型です。

(114) 参照連動項目には対応していません。

AGCOUNT

AGCOUNT("項目名")

「出力制御 > 一覧画面 > 集計値 > 個数」の項目名を指定します。一覧表示画面の集計機能で求めた「個数」の値を取得できます。戻り値の型は8バイト整数(long)です。

例：

```
AGCOUNT("kosuu")
```

項目名が存在しない場合、または一覧表示画面の集計機能が設定されていない場合は -1 が返ります。

AGSUM

AGSUM("項目名")

「出力制御 > 一覧画面 > 集計値 > 合計値」の項目名を指定します。一覧表示画面の集計機能で求めた「合計値」の値を取得できます。戻り値の型は8バイト整数(long)です。

例：

```
AGCOUNT("goukei")
```

項目名が存在しない場合、または一覧表示画面の集計機能が設定されていない場合は -1 が返ります。

AGAVG

AGAVG("項目名")

「出力制御 > 一覧画面 > 集計値 > 平均値」の項目名を指定します。一覧表示画面の集計機能で求めた「平均値」の値を取得できます。戻り値の型は8バイト浮動小数点(double)です。

例：

```
AGAVG("heikin")
```

項目名が存在しない場合、または一覧表示画面の集計機能が設定されていない場合は -1 が返ります。

AGMAX

AGMAX("項目名")

「出力制御 > 一覧画面 > 集計値 > 最大値」の項目名を指定します。一覧表示画面の集計機能で求めた「最大値」の値を取得できます。

例：

```
AGMAX("saidai")
```

戻り値の型は java.lang.Object であるため、利用する場合は明示的な型変換が必要です。例えば日付型の場合、次のようになります。

```
(java.util.Date)AGMAX("salesday")
```

数値型（整数）の場合、次のようになります。

```
(Integer)AGMAX("salestotal")
```

項目名が存在しない場合、または一覧表示画面の集計機能が設定されていない場合は null が返ります。

AGMIN

AGMIN("項目名")

「出力制御 ＞ 一覧画面 ＞ 集計値 ＞ 最小値」の項目名を指定します。一覧表示画面の集計機能で求めた「最小値」の値を取得できます。

例：

```
AGMIN("saisyou")
```

戻り値の型は java.lang.Object であるため、利用する場合は明示的な型変換が必要です。例えば日付型の場合、次のようになります。

```
(java.util.Date)AGMIN("salesday")
```

数値型（整数）の場合、次のようになります。

```
(Integer)AGMIN("salestotal")
```

項目名が存在しない場合、または一覧表示画面の集計機能が設定されていない場合は null が返ります。

モデル参照項目の扱い

コンテナ内に含まれているモデル参照の ID 部と内容部を指定することができます。 単に項目名のみを指定した場合は、その「内容」が対象となります。

外部キー子モデルの指定方法

- 外部キー関連モデルのモデル名に接尾語 "_lp" を付与してください。
- 外部キーの項目を指定する場合、その項目は一覧表示の対象となっている必要があります。
- 親モデルの更新時に、紐づく子モデルの値も再計算＆再保存されます。
- 子モデルが親モデルの値を参照するような式の場合、子モデルを保存すると関連する親モデルの値も再計算＆再保存されます。

COUNT,SUMなどの条件部に指定する項目

子モデルの値で判定する

外部キー子モデルのCOUNT,SUMなどを扱う場合、条件部に指定する項目に接頭語 "子モデル名_lp." を指定することで「子モデル」に含まれるものとなります。

例えば

```
SUM(${support_lp.kingaku}, "${support_lp.type#id}==1")
```

の場合、条件部に記述した type 項目は、子モデル側（support）に含まれるものを使います。このときの条件判定は、子モデルの1データ毎に行われます。

親モデルの値で判定する

接頭語を付与しない場合、「親モデル」に含まれる項目を使います。

例えば

```
COUNT(${child_lp}, "${child_lp.status#id}==${status#id}")
```

の場合、親モデルの status 項目の値と一致する子モデルのみが対象となります。

(115) (116)

(115) この例でわかるように、接頭語 "子モデル名_lp." を省略し、かつ、条件部の項目が親モデルと子モデルの両方に含まれる場合は、親モデルの項目が使われます。

(116) 子モデル名を明示しない場合で、指定項目（上の例では status 項目）が親モデルに存在せず、子モデルに存在した場合は、自動的に接頭語 "子モデル名_lp." があると解釈されます。つまり子モデルの項目 status で判定されます。

AVERAGE

AVERAGE(繰り返し項目, [条件])

AVERAGE(繰り返しコンテナ項目, [条件])

AVERAGE(外部キー子モデル項目, [条件])

平均値を算出します。

整数型の項目に対する平均値を求める場合、計算結果を格納する項目の型は「8バイト浮動小数点」とすることを推奨します。（「4バイト浮動小数点」よりも演算精度が高くなります。） その他の記述例、制約については SUMと同様です。

演算精度 (8.2.0)

R8.2.0より集合関数 AVERAGE は java.math.BigDecimal を使います。

制約

- 計算対象となる繰り返し項目、コンテナ、外部キー項目が存在しないか、または sum, count 演算子に指定した条件を満たすものが一つもなかった場合、計算結果は格納されません。（表示部分は空白となります。） このような場合でも強制的に「0」をセットする場合は、計算式を記述した項目を「必須」としてください。
- 検索条件の初期値設定に、集合関数を用いることはできません。
- 「入力方式 > ユーザ定義チェック」に集合関数を用いることはできません。
- 「メール送受信 > 時間指定によるメール送信」「時間指定によるメール受信（データ取り込み）」の「ジョブ定義」-「起動条件」に集合関数を用いることはできません。
- 「帳票 > ファイル名」に集合関数を用いることはできません。
- 集合関数で指定する「項目部」および「条件部」に（COUNT,SUM 等の）集合関数を含めることはできません。例えば次の式はエラーになります。

```
JOIN(JOIN(${item1}, ","), ",")
```

JOIN関数の項目部にJOIN関数が指定されています。 このような場合は、それぞれ別の項目単位で計算するようにしてください。

MAX

MAX(繰り返し項目)

MAX(繰り返しコンテナ項目)

MAX(外部キー子モデル項目)

MAX(数値1, 数値2), MAX(数値が格納された配列)

MAX(日付1, 日付2), MAX(日付が格納された配列)

MAX(文字列1, 文字列2), MAX(文字列が格納された配列)

複数個の引数のうち、最大値を返します。 引数が二つ以上の場合は、ARRAY関数と組み合わせて利用できます。

引数に繰り返し項目、繰り返しコンテナ項目、外部キー子モデルを指定できます。 また項目の型に日付型、文字列型を指定できます。

例：multiple_item は繰り返し項目

```
MAX(${multiple_item})
```

例：繰り返しコンテナ項目 report/number を対象

```
MAX(${report/number})
```

例：外部キー子モデルとして紐づいている support モデルの kingaku 項目を対象

```
MAX(${support_lp.kingaku})
```

例：二つの項目 item1, item2 は同じ型

```
MAX(${item1}, ${item2})
```

例：複数の項目 item1, item2, item3 は同じ型

```
MAX(ARRAY_INT(${item1}, ${item2}, ${item3})
```

MIN

MIN(繰り返し項目)

MIN(繰り返しコンテナ項目)

MIN(外部キー子モデル項目)

MIN(数値1, 数値2), MIN(数値が格納された配列)

MIN(日付1, 日付2), MIN(日付が格納された配列)

MIN(文字列1, 文字列2), MIN(文字列が格納された配列)

複数個の引数のうち、最小値を返します。　引数が二つ以上の場合は、ARRAY関数と組み合わせて利用できます。[117]

例：multiple_item は繰り返し項目

```
MIN(${multiple_item})
```

例：繰り返しコンテナ項目 report/number を対象

```
MIN(${report/number})
```

例：外部キー子モデルとして紐づいている support モデルの kingaku 項目を対象

```
MIN(${support_lp.kingaku})
```

例：二つの項目 item1, item2 は同じ型

```
MIN(${item1}, ${item2})
```

例：複数の項目 item1, item2, item3 は同じ型

```
MIN(ARRAY_INT(${item1}, ${item2}, ${item3})
```

FIRST

FIRST(繰り返し項目, [条件])

[117] 引数に繰り返し項目、繰り返しコンテナ項目、外部キー子モデルを指定できます。 また項目の型に日付型、文字列型を指定できます。

FIRST(繰り返しコンテナ項目, [条件])

FIRST(外部キー子モデル項目, [条件])

先頭のレコードの値を取得します。

文字列型/整数型/日付型など、さまざまな型で用いることができます。計算結果を格納する項目の型と、計算対象となる項目の型を同じとしてください。（ただしモデル参照については参照先の型を指定してください。）

この設定を行うときは繰り返しコンテナ内のすべての項目について、ソートボタンを非表示にすることを推奨します。（ソートされてしまうと、最後のレコードが変わってしまうため。）

その他の記述例、制約については SUMと同様です。

LAST

LAST(繰り返し項目, [条件])

LAST(繰り返しコンテナ項目, [条件])

LAST(外部キー子モデル項目, [条件])

最後のレコードの値を取得します。

文字列型/整数型/日付型など、さまざまな型で用いることができます。計算結果を格納する項目の型と、計算対象となる項目の型を同じとしてください。（ただしモデル参照については参照先の型を指定してください。）

この設定を行うときは繰り返しコンテナ内のすべての項目について、ソートボタンを非表示にすることを推奨します。（ソートされてしまうと、最後のレコードが変わってしまうため。）

その他の記述例、制約については SUMと同様です。

SLICE

SLICE(繰り返し項目, 開始レコード番号または項目名, [代替値])

SLICE(繰り返しコンテナ項目, 開始レコード番号または項目名, [代替値])

SLICE(外部キー子モデル項目, 開始レコード番号または項目名, [代替値])

N番目のレコードの値を取得します。

文字列型/整数型/日付型など、さまざまな型で用いることができます。計算結果を格納する項目の型と、計算対象となる項目の型を同じとしてください。（ただしモデル参照については参照先の型を指定してください。）

開始レコード番号は項目名の横に（コンマ区切りの第二引数として）指定します。レコード番号は「0」から数えます。また、指定したレコードが存在しなかった場合は、空白が表示されます。

開始レコード番号は固定値を指定します。また、開始レコード番号を示す特別な表記 "first" と "last" を使うことができます。"first" は先頭のレコードを、"last" は末尾のレコードを意味します。

指定したレコードが存在しない場合に、セットする値を指定することができます。コンマ区切りの第三引数に代替値を指定してください。

例：繰り返し項目 number の最初の項目の値を取得する。

```
SLICE(${number},0)
```

例：繰り返しコンテナ report/number の先頭から 1 番目の項目の値を取得する。ただしレコードが存在しなかった場合は "なし" を取得する。（取得する項目の型は文字列型であること）

```
SLICE(${report/number}, first+1, "なし")
```

例：外部キー連携モデル support の項目 kingaku の末尾から 2 つ前の値を取得する。ただしレコードが存在しなかった場合は 0 を取得する。(外部キーの指定方法の詳細...)

```
SLICE(${support_lp.kingaku}, last-2, 0)
```

開始レコード番号に代わって項目名を指定する

第二引数の開始レコード番号に、項目名を指定できます。項目名は ${...} で囲みます。

```
SLICE(${report/number}, ${item}+1)
```

項目名に簡単な四則演算を加えることはできますが、他の関数を含めることはできません。

JOIN

JOIN(繰り返し項目, [連結文字])

JOIN(繰り返しコンテナ項目, [連結文字])

JOIN(外部キー子モデル項目, [連結文字])

JOIN(チェックボックス型の項目, [連結文字])

すべての項目値を連結した文字列を作成します。文字列型/整数型/日付型など、さまざまな型で用いることができます。計算結果を格納する項目の型は「文字列型」としてください。

第二引数に連結時の文字を指定することができます。省略時は「,」（コンマ）が用いられます。

チェックボックス型の項目に対して適用した場合は、"チェックされたデータ" を連結対象とします。

例：繰り返し項目 number の値を連結した文字列を取得する。区切り文字はデフォルト値「,」（コンマ）を用いる。

```
JOIN(${number})
```

例：繰り返しコンテナ　report/staff　の値を連結した文字列を取得する。区切り文字は「;」（セミコロン）を用いる。

```
JOIN(${report/staff},';')
```

例：外部キー連携モデル support の項目 companyname の値を連結した文字列を取得する。区切り文字はデフォルト値「,」（コンマ）を用いる。(外部キーの指定方法の詳細...)

```
JOIN(${support_lp.staff})
```

統計処理

MEDIAN

MEDIAN(数字の配列)

MEDIAN(数字型の繰り返し項目)

MEDIAN(数字型の繰り返しコンテナ項目)

MEDIAN(数字型の外部キー子モデル項目)

中央値を求めます。戻り値は「8バイト浮動小数点型」です。

```
MEDIAN(ARRAY_INT(80,55,90,55,77,64))
```

上記式は "70.5" を返します。

MODE

MODE(数字の配列)

MODE(数字型の繰り返し項目)

MODE(数字型の繰り返しコンテナ項目)

MODE(数字型の外部キー子モデル項目)

最頻値を求めます。戻り値の型は、引数に渡した数字型になります。 いずれも最頻値とならない(すべての個数が一つずつ)の場合は null を返します。

```
MODE(ARRAY_INT(80,55,90,55,77,64))
```

上記式は "55" を返します。

STDEV

STDEV(数字の配列)

STDEV(数字型の繰り返し項目)

STDEV(数字型の繰り返しコンテナ項目)

STDEV(数字型の外部キー子モデル項目)

標準偏差を求めます。戻り値は「8バイト浮動小数点型」です。 引数の個数は2つ以上の値が必要です。これを下回った場合、値は不定 (Double.NaN) となります。

```
STDEV(ARRAY_DOUBLE(5.0, 6.0, 7.0, 8.0))
```

上記式は "1.2909944487358056" を返します。

STDEVP

STDEVP(数字の配列)

STDEVP(数字型の繰り返し項目)

STDEVP(数字型の繰り返しコンテナ項目)

STDEVP(数字型の外部キー子モデル項目)

母集団の標準偏差を求めます。戻り値は「8バイト浮動小数点型」です。 引数の個数は2つ以上の値が必要です。これを下回った場合、値は不定 (Double.NaN) となります。

```
STDEVP(ARRAY_DOUBLE(5.0, 6.0, 7.0, 8.0))
```

上記式は "1.118033988749895" を返します。

VAR

VAR(数字の配列)

VAR(数字型の繰り返し項目)

VAR(数字型の繰り返しコンテナ項目)

VAR(数字型の外部キー子モデル項目)

分散値を求めます。戻り値は「8バイト浮動小数点型」です。 引数の個数は2つ以上の値が必要です。これを下回った場合、値は不定 (Double.NaN) となります。

```
VAR(ARRAY_DOUBLE(5.0, 6.0, 7.0, 8.0))
```

上記式は "1.6666666666666667" を返します。

VARP

VARP(数字の配列)

VARP(数字型の繰り返し項目)

VARP(数字型の繰り返しコンテナ項目)

VARP(数字型の外部キー子モデル項目)

母集団の分散値を求めます。戻り値は「8バイト浮動小数点型」です。 引数の個数は2つ以上の値が必要です。これを下回った場合、値は不定 (Double.NaN) となります。

```
VARP(ARRAY_DOUBLE(5.0, 6.0, 7.0, 8.0))
```

上記式は "1.25" を返します。

条件分岐

AND

AND(式...)

引数の条件式の AND 演算を行います。すべての「式」が真(true)のとき、AND演算の結果は真(true)を返します。戻り値の型は真偽値(trueまたはfalse)となるため、IF関数の内部などで利用できます。

```
AND(式1, 式2, 式3,...)
```

例	動作
IF(AND(${x}>0, ${y}>0), "Yes", "No")	(X と Y の値が両方とも 0 より大きければ "Yes" という文字がセットされます。)

IF

IF(条件式, 条件式が真の場合, 条件式が偽の場合)

条件式が満たされている場合（真）と、そうでない場合（偽）の判定を行います。 条件式に使う記号（比較演算子）は次のとおりです。Excelと全く同じではないのでご注意ください。

記述	説明
X==Y	XとYが等しい
X>Y	XがYよりも大きい
X<Y	XがYよりも小さい
X>=Y	XがY以上
X<=Y	XがY以下
X!=Y	XとYは等しくない
!X	Xの否定（Xがtrueなら、falseになる）
X==null	Xは未入力である。null という表記は文字型または日付/時刻/日付時刻型でのみ利用できます。

なお、IF 関数の組み合わせの数の制限はありません。

```
  IF(式, ${value1}, ${value2})
IF(式, IF（式, ${value1}, ${value2}), ${value3})
```

例	動作
IF(${score} >= 80, "目標達成", "未達")	(score 値が 80 以上であれば "目標達成" という文字がセットされます。)
IF(EXACT(${str1}, ${str2}), "同じ", "違う")	(文字列の比較については、EXACT 関数を組み合わせてください。)

(118) (119)

N

N(値)

引数の値を数値に変換します。

```
  N(${item1})
```

例	動作
N(数値)	数値
N(日付)	シリアル値
N(true)	1
N(false)	0
N(その他)	0

NOT

NOT(式...)

引数の boolean 値を反転させます。 戻り値の型は真偽値(trueまたはfalse)となります。

```
  NOT(式)
```

(118) IF 関数内における"条件式が真の場合", "条件式が偽の場合" の演算結果は、その式を記述した項目の型となります。

(119) 数値型項目に null をセットする場合は IF ではなく参考演算子を使います。"計算式 > 基本的な書き方 > 参考演算子" をお読みください。

例	動作
NOT(ISBLANK(${x}))	(X が空白の場合は false が戻ります。)

OR

OR(式...)

引数の条件式の OR 演算を行います。いずれかの「式」が真(true)のとき、OR演算の結果は真(true)を返します。戻り値の型は真偽値(trueまたはfalse)となるため、IF関数の内部などで利用できます。

```
OR(式1, 式2, 式3,...)
```

例	動作
IF(OR(${x}>0, ${y}>0), "Yes", "No")	(X と Y の値のどちらか一方が 0 より大きければ "Yes" という文字がセットされます。)

TRUE

TRUE()

真(true)を返します。

FALSE

FALSE()

偽(false)を返します。

選択

CHOOSE

CHOOSE(インデックス, 文字列, 文字列,...)

CHOOSE(インデックス, 文字列の配列)

CHOOSE_BYTE(インデックス, 1バイト整数, 1バイト整数,...)

CHOOSE_BYTE(インデックス, 1バイト整数の配列)

CHOOSE_SHORT(インデックス, 2バイト整数, 2バイト整数,...)

CHOOSE_SHORT(インデックス, 2バイト整数の配列)

CHOOSE_INT(インデックス, 4バイト整数, 4バイト整数,...)

CHOOSE_INT(インデックス, 4バイト整数の配列)

CHOOSE_LONG(インデックス, 8バイト整数, 8バイト整数,...)

CHOOSE_LONG(インデックス, 8バイト整数の配列)

CHOOSE_FLOAT(インデックス, 4バイト浮動小数, 4バイト浮動小数,...)

CHOOSE_FLOAT(インデックス, 4バイト浮動小数の配列)

CHOOSE_DOUBLE(インデックス, 8バイト浮動小数, 8バイト浮動小数,...)

CHOOSE_DOUBLE(インデックス, 8バイト浮動小数の配列)

index に指定した番号に基づいた値を返します。戻り値の型は文字列型です。 該当する要素が含まれない場合は null を返します。

```
CHOOSE(${index}, 値1, 値2,...)
```

例	動作
CHOOSE(1, "a", "b", "c")	"a"
CHOOSE(3, "a", "b", "c")	"c"
CHOOSE(0, "a", "b", "c")	null
CHOOSE(4, "a", "b", "c")	null
CHOOSE(2.5, "a", "b", "c")	"b"

CID

CID()

初期値の設定時に利用できる特別な関数です。繰り返しコンテナ内の項目について、現在操作しているコンテナ行の番号を返します。番号は1から開始します。

CID関数を用いた例を紹介します。

- 数値を繰り返しコンテナに按分する
- 繰り返しコンテナでデータの変更履歴を管理する

権限

HASMODELPERMISSION

HASMODELPERMISSION(権限名, モデル名(英語))

利用者が、引数で指定された権限を有している場合はtrueを返します。そうでなければfalseを返します。

```
HASMODELPERMISSION("perm_boss", "staff")
```

HASPRINCIPAL

HASPRINCIPAL(プリンシパル名)

利用者が、引数のプリンシパルを有している場合はtrueを返します。 そうでなければfalseを返します。
(120)

(120) 管理者権限でログオン時は HASPRINCIPAL 関数は常に true を返します。

```
HASPRINCIPAL("principal_manager")
```

アカウント

USERNAME

USERNAME()

ログオンアカウント名を返します。

例: ログオンアカウントが "admin" なら "管理者" という文字列を、そうでなければ "一般" という文字列をセットする。

```
>IF(EXACT(USERNAME(),"admin"), "管理者","一般")
```

USERINFO

USERINFO(項目名)

USERINFO(項目名#id)

ログオンアカウントに関する情報を返します。引数には juser モデルの項目名（英語）を指定します。

戻り値の型は文字列になります。

繰り返し項目の場合は item.0, item.1 と指定します。 繰り返しコンテナの場合は cont.0.str と指定します。

チェックボックスなど複数個の値をもつ場合は「空白」を区切り文字とした一つの文字列を返します。

対象項目がモデル参照の場合、「項目名」は内容部を、「項目名#id」はID部を返します。（なおID部も常に文字列型になります。）

例1: ログオンアカウントの氏名情報を取得する。

```
USERINFO("name")
```

例2: juserモデルに、別モデル "company" を参照する項目 "companyname" を用意した。その companyname 項目の ID 部を取得する。

```
USERINFO("companyname#id")
```

(121)

詳細

USERINFO関数の戻り値はログオン時にセットされる

USERINFO関数の戻り値は、ログオン時に一度 juser テーブルの値を読み込んで用意します。この方式のため、juser の更新後に USERINFO 関数が（更新前の）古い値を返すことがあります。

戻り値にnullは返されない

USERINFO関数の戻り値がnullとなることはありません。値がない場合は、空文字 "" を返します。

IF関数の条件部に用いるときは USERINFO("name") == null とするのではなく、次のようにISBLANK関数を利用して判定してください。

```
IF(ISBLANK(USERINFO("name")), ...)
```

(121) 参照項目が参照する先のモデルが複合主キーの場合、最初の主キーは "#id" ですが、二番目の主キーは "#id2" で参照します。以降、"#id3", "#id4",... となります。

JGROUPID

JGROUPID()

ログオンアカウントが所属するグループIDの配列（整数型）を返します。値は昇順に並んでいます。

例：利用者の所属するグループが「ない」か、またはグループID 1000 に所属している場合は null を、そうでなければ（所属している）先頭のグループIDを返す。

```
IF(
  OR(
    ISEMPTY(JGROUPID()),
    CONTAINS(JGROUPID(),1000)
  ),
  null,
  CHOOSE_INT(1,JGROUPID())
)
```

グループ「本社、A事業所、B事業所、C事業所」を想定します。本社のグループIDを1000とします。　本社所属の社員はA,B,C全ての事業所に所属させているとします。　グループで検索できるよう、検索画面にグループ(jgroupを参照する項目）を用意します。参照型はリストボックスとします。ここで検索画面の初期値として上記式を設定すると、所属グループに本社が含まれている場合は、検索条件の初期値が　「(すべて)」(null）になります。本社が含まれていない場合は先頭の1件をコンボボックスにセットします。

・　利用例:他モデルの参照 > 他項目の値で選択肢を絞り込む > 固定値に関数を利用する

画面制御

BROWSERTYPE

BROWSERTYPE()

利用している Web ブラウザの種類を返します。

```
BROWSERTYPE()
```

戻り値は次のような文字列です。

戻り値	ブラウザの種類
Gecko	FireFox
Opera	Opera
BIZBROWSER	Biz/Browser
MSIE6	Internet Explorer 6
MSIE7	Internet Explorer 7
MSIE8	Internet Explorer 8
MSIE9	Internet Explorer 9
MobileSafari	iPhone, iPod touch
Safari	Safari
iPad	iPad
docomo	DoCoMo 携帯電話
SoftBank	SoftBank 携帯電話
au	KDDI 携帯電話

iPhone/iPod touch で操作するときに非表示にしたい項目には、条件付き閲覧権限を以下のように指定することができます。

```
select;NOT(EXACT(${browsertype},"MobileSafari"))
```

iPad指定について

"iPad" はデフォルトで無効となっています。

この設定を有効にする場合「スマートフォンの利用 > iPad利用時のデフォルトをPC画面とする」の設定を行ってください。iPadでアクセスしたときにブラウザタイプが "MobileSafari" ではなく "iPad" となります。

EVENT

EVENT()

利用者が押下した画面上のボタン名を取得します。[122]

RESULTSIZE

RESULTSIZE()

検索・一覧表示画面でのみ有効な関数です。直近の検索処理結果数を返します。

ボタン制御の式として用いることで「検索結果が1件以上の場合にのみ一覧更新ボタンを表示する」ことや、「検索処理を行った場合のみ帳票出力ボタンを表示する」といった運用を行うことができます。

SCREENID

SCREENID()

ユニークな画面名を返します。戻り値は文字列型になります。 具体的には「SCREENTYPE + 画面名」となります。

モデル名（英語）が "customer" の場合の SCREENID() の戻り値は次のとおりです。

画面種別	戻り値	備考
詳細表示画面	showCustomer	モデル名の先頭は大文字になります。
コピー登録画面	copyCustomerC1	コピー登録画面では、末尾に "C1" などが付与されます。（コピー画面数に応じて変動します）
ウィザード詳細画面	showCustomerP	ウィザードを用いた画面では、末尾に "P" が付与されます。
カスタマイズした画面	myShowCustomerP	カスタマイズした画面は "my" で開始されます。
（画面なし）	unknown	メール送信、メール受信ジョブなどをいいます。

SCREENTYPE

SCREENTYPE()

画面種別を返します。戻り値は文字列型になります。

大文字・小文字は区別します。以下の表に記載の通りに使ってください。（例： "SHOW" ではなく "show" です。"showlist"ではなく"showList"です。）

具体的な画面種別は次のとおりです。

画面種別	戻り値
詳細表示画面	show
一覧表示画面	showList

[122] ボタン名は、各画面を構成する JSP ファイル内のフォーム要素に含まれる "action_XXX" という文字列における "XXX" の部分が該当します。

新規登録画面	insert
コピー登録画面	copy
更新画面	update
CSV・Excelダウンロード画面	download
CSV・Excelアップロード更新画面	uploadUpdate
一覧更新画面	updateList
カレンダビュー	showListCalendar
(画面なし)	unknown

※ 「画面なし」は、メール送信、メール受信ジョブなどをいいます。

WIZARDPAGE

WIZARDPAGE()

ウィザード画面の画面番号を返します。戻り値は整数型です。ウィザードでない画面で使った場合、0 が返されます。

リソース・国際化

LOCALE

LOCALE()

Web ブラウザのロケール情報が返されます。例えば日本語ロケールの場合は "ja" となります。

条件付き権限管理と組み合わせることで、一つのモデルに複数のモデル項目を用意し、ロケールによって表示を切り替えるという使い方ができるようになります。例を示します。

項目名（日本語）	項目名（英語）	閲覧権限
名前(日)	name_ja	select;EXACT(LOCALE(),"ja")
名前(英)	name_en	select;EXACT(LOCALE(),"en")

MESSAGE

MESSAGE(リソースキー)

MESSAGE(リソースキー, 引数...)

リソースファイル jfcapp のリソースキーに合致したメッセージを取得します。戻り値は文字列型です。ロケールは考慮されていますので、ログオンしている利用者が選択したロケールにあわせて適切なメッセージが表示されます。（ロケール毎のメッセージは事前に用意しておく必要があります。）

```
MESSAGE("M0001")
MESSAGE("M0002", ${item1})
```

開発者が独自のリソースキーと、それに対応するメッセージを用意することができます。　詳細は"メッセージの変更・国際化 > リソースファイルの編集"をお読みください。

(123)

ERRORMESSAGE

ERRORMESSAGE(リソースキー)

(123) リソースファイル jfcapp に記述するメッセージに HTML タグを含めることはできません。

ERRORMESSAGE(リソースキー, 引数...)

リソースファイル errormsg のリソースキーに合致したメッセージを取得します。戻り値は文字列型です。ロケールは考慮されていますので、ログオンしている利用者が選択したロケールにあわせて適切なメッセージが表示されます。（ロケール毎のメッセージは事前に用意しておく必要があります。）

```
ERRORMESSAGE("E0001")
ERRORMESSAGE("E0002", ${item1})
```

開発者が独自のリソースキーと、それに対応するメッセージを用意することができます。 詳細は"メッセージの変更・国際化 > リソースファイルの編集"をお読みください。

(124) (125)

モデル参照

ISEMPTY

ISEMPTY(項目名)

項目に値が入っているかどうかを判定します。未入力の場合は true を返します。入力済の場合は false を返します。

数値型で非必須項目かつ未入力の場合の判定が行えます。参照連動型項目にも対応しています。

例：参照連動型項目 refitem1 が未入力かどうかをチェックする。

```
IF(ISEMPTY(${refitem1}), "未入力です","値があります")
```

(126)

ISBLANK関数との違い

ISBLANKの方が実行パフォーマンスが良いです。単純に値がNULLかどうかを判定する場合はISBLANKの利用を推奨します。ISEMPTYは数値型項目の未入力判定や参照連動項目にも対応していますが、その分、内部処理が複雑になっています。

SELECTED

SELECTED(リストボックス/ラジオ型項目名)

リストボックス、ラジオボタンの未選択を判定します。 引数に指定した項目の値が選択されていた場合に true を返します。そうでなければ false を返します。

モデルに含まれるリストボックス/ラジオ型項目、またはSUMやCOUNTなどの条件部に用いることができます。条件部のみ、外部キー子モデルの要素を指定することができます。

例：リストボックス項目 mrefmodel の選択状態を判定する

```
IF(SELECTED(${mrefmodel}), "選択されています","未選択です")
```

例：繰り返しコンテナ内のリストボックス項目 mrefmodel の選択状態を判定する

```
IF(SELECTED(${cont/cmrefmodel}), "選択されています","未選択です")
```

例：外部キーの子モデルの条件部にて、選択されていないものを数える

```
COUNT(${Child_lp}, "SELECTED(${Child_lp.mrefmodel})==false")
```

---

(124) ユーザ定義チェックのメッセージに ERRORMESSAGE 関数を用いた場合、引数は固定文字列のみとなります。${...} による項目名の指定を行うことはできません。

(125) リソースファイル errormsg に記述するメッセージに HTML タグを含めることはできません。

(126) ISEMPTY(配列)という使い方もできます。

CHECKED

CHECKED(チェックボックス型項目名)

チェックボックス項目の、選択されている数をかぞえます。戻り値は整数型です。

モデルに含まれるチェックボックス型項目、またはSUMやCOUNTなどの条件部に用いることができます。条件部のみ、外部キー子モデルの要素を指定することができます。

例：チェックボックス型項目の選択肢を数える。

```
CHECKED(${checkboxitem})
```

例：親モデル側で、外部キー子モデルのチェックボックスが一つ以上選択されている子モデルのみを数える。ここではCHECKED関数を直接、記述しない。[127]

```
COUNT(${Child_lp}, "${Child_lp.checkednum}>0")
```

EXACT

EXACT(チェックボックス型項目名, 値)

第一引数のチェックボックス型項目で最初に選択された値が第二引数の値と等しいかどうかを返します。戻り値は boolean 型（真/偽）です。

例：チェックボックス型項目でチェックされた最初の選択肢の（内容部が）A かどうかを調べる。

```
EXACT(${checkboxitem}, "A")

EXACT(${checkboxitem#content}, "A")
```

例：チェックボックス型項目でチェックされた最初の選択肢の（ID部が）1000 かどうかを調べる。

```
EXACT(${checkboxitem#id}, 1000)
```

この関数は文字列処理でも説明しています。

CONTAINS

CONTAINS(チェックボックス型項目名#array, 取得パターン, 値)

第一引数のチェックボックス型項目で複数の選択された値が第三引数の値と等しいかどうかを返します。戻り値は boolean 型（真/偽）です。第一引数は項目名の後ろに "#array" を付与する必要があります。第二引数には "id" または "content" を指定します。それぞれ ID 部か内容部かを指定します。

例：チェックボックス型項目でチェックされた選択肢に（ID部が）1000　が含まれているかどうかを調べる。

```
CONTAINS(${checkboxitem#array}, "id", 1000)
```

例：チェックボックス型項目でチェックされた選択肢に（内容部が）"A"　が含まれているかどうかを調べる。

```
CONTAINS(${checkboxitem#array}, "content", "A")
```

ISNULL

ISNULL(項目名)

指定した項目の、値が存在しない(未設定)かどうかを調べます。

この関数はSUMやCOUNTなどの条件部にのみ、用いることができます。

---

[127] 親モデル側で、外部キー子モデルのチェックボックスを直接、引数に指定することはできません。 このため次の例では子モデル側に、チェックボックスの数を記録する項目 checkednum を用意しています。 また、同項目を一覧表示に加え、かつ隠し項目としています。

例：親モデル側で、外部キー子モデルのストア参照項目(参照先は日付型)attendance_date　が未設定のデータを数える。

```
COUNT(${attendance_lp}, "ISNULL(${attendance_lp.attendance_date})")
```

例：繰り返しコンテナ内のストア参照項目(参照先は日付型)cont/cdate が未設定のデータを数える。

```
COUNT(${cont}, "ISNULL(${cont/cdate})"})
```

制約

- 参照先モデルが複合主キーの場合、これらの関数は利用できません。
- 入力チェック（ユーザー定義式）ではCOUNT,SUMなどの集合演算を行うことができません。そのためCOUNT,SUM関数の条件部にSELECTEDやISNULL関数を指定するような使い方もできません。

財務

FV

FV

積立金の計算を行います。一定の利率、支払い額、支払い期間を元にした、初期投資の将来価値を算出します。

毎月の積み立て金額	50000
年利（%）	1.0
積み立て期間(年)	10
積み立て金額	¥6,307,494

図 FV関数の例

```
ROUND(FV(${interestRates}/100/12,${periods}*12,${payment}),0)
```

NPV

NPV

正味現在価値の計算を行います。期間中の利率を固定と想定して、毎回異なる支払い額を定期的に受け取る場合の純現在価値を算出します。

年間収益

No	収益額
1	300000
2	420000
3	680000

この投資の正味現在価値	¥118,844

図 NPV関数の例

```
ROUND(NPV(${interestRates}/100,${loan_amount},${cont/value}),0)
```

PMT

PMT

ローン返済の計算を行います。借入元金を指定された利率、指定された期間で返済する際に必要な各期間の返済額を算出します。

ID	1000
タイトル	PMT
年利（%）	4.0
返済期間(年)	30
借入金	10000000
毎月の返済金額	¥47,742
総返済金額	¥17,186,950

図 PMT関数の例

```
PMT(${interestRates}/100/12,${periods}*12,${payment})
```

PV

PV

ローン借入の計算を行います。期間中の利率を固定金利と想定して、毎回同じ支払い額を定期的に受け取る場合の純現在価値を算出します。

ID	1000
タイトル	PV
毎月の返済金額	50000
年利（%）	4.0
返済期間(年)	30
借り入れ金額	¥10,473,062

図 PV関数の例

```
ROUND(PV(${interestRates}/100/12,${periods}*12,${payment}),0)
```

ワークフロー

WF_NEXT_GROUPID_MAIL

WF_NEXT_GROUPID_MAIL("jgroupモデル名", "jgroupモデルの主キー名", "jgroupメールアドレスを格納している項目名")

次のノードのメールアドレスを返します。次のノードとは、申請時であれば次の承認者へ、承認時であれば次の承認者（または決裁者）という意味です。差し戻しの場合は、差し戻されたときの承認者になります。グループを使うことがわかっている場合に利用できます。

第一引数のモデル名と第二引数名の主キー名から、第三引数のメールアドレス格納項目の値を返します。戻り値は文字列型（メールアドレス）です。複数ある場合はコンマ区切りの文字列となります。

```
WF_NEXT_GROUPID_MAIL("jgroup", "jgroupid", "jgroupmailaddress")
```

(128) (129)

(128) この関数はジョブ制御では利用できません。

(129) 差し戻し先のノードが合議の場合、合議ルールに含まれる全員が対象となります。（合議で承認した人のみ、ではなく全員、です。）

WF_NEXT_MAIL

WF_NEXT_MAIL("juserモデル名", "juserモデルの主キー名", "juserメールアドレスを格納している項目名","jgroupモデル名", "jgroupモデルの主キー名", "jgroupメールアドレスを格納している項目名")

次のノードのメールアドレスを返します。次のノードとは、申請時であれば次の承認者へ、承認時であれば次の承認者（または決裁者）という意味です。差し戻しの場合は、差し戻されたときの承認者になります。ユーザあるいはグループの両方の可能性がある場合に利用できます。

第一引数から第三引数まではjuserに関するもので、第四引数から第六引数まではjgroupに関するものです。戻り値は文字列型（メールアドレス）です。複数ある場合はコンマ区切りの文字列となります。

```
WF_NEXT_MAIL("juser", "userid", "mailaddress", "jgroup", "jgroupid", "jgroupmailaddress")
```

(130) (131)

WF_GROUPID

WF_GROUPID()

次のノードに関する処理保留者（次に承認アクションを起こすべき人）のグループIDを返します。WF_NEXT_GROUPID()関数の、引数なし版です。当該データに関する値を返します。戻り値の型は整数型の配列です。 値が存在しない場合は、nullを返します。

この関数を利用する項目はデータベースに保存しない設定にしてください。常に最新の値を表示する必要があります。

```
WF_GROUPID()
```

(132)

WF_IS_ADMITUSER

WF_IS_ADMITUSER()

当該データについて、ワークフローを保留させている人(処理保留者)かどうかを返します。

具体的には、自分自身（ログオンアカウント）が当該データについての処理保留者または処理保留グループIDに属しているとき、trueを返します。そうでなければfalseを返します。

この関数を利用する項目はデータベースに保存しない設定にしてください。常に最新の値を表示する必要があります。

```
WF_IS_ADMITUSER()
```

(133)

WF_APPLICANT

WF_APPLICANT()

当該データについて、ワークフロー申請者のユーザIDを返します。

この関数を利用する項目はデータベースに保存しない設定にしてください。常に最新の値を表示する必要があります。

```
WF_APPLICANT()
```

---

(130) この関数はジョブ制御では利用できません。

(131) 差し戻し先のノードが合議の場合、合議ルールに含まれる全員が対象となります。（合議で承認した人のみ、ではなく全員、です。）

(132) この関数はジョブ制御では利用できません。

(133) この関数はジョブ制御では利用できません。

(134)

WF_APPLICANT_MAIL

WF_APPLICANT_MAIL("モデル名", "モデルの主キー名", "メールアドレスを格納している項目名")

当該データについて、申請者のメールアドレスを返します。

第一引数のモデル名と第二引数名の主キー名から、第三引数のメールアドレス格納項目の値を返します。戻り値は文字列型（メールアドレス）です。

```
WF_APPLICANT_MAIL("juser", "userid", "mailaddress")
```

(135)

WF_LASTPROCESSDATE

WF_LASTPROCESSDATE()

当該データについて、最後にワークフローに何らかの処理（承認、決裁など）を行った日付時刻を返します。

この関数を利用する項目はデータベースに保存しない設定にしてください。常に最新の値を表示する必要があります。

```
WF_LASTPROCESSDATE()
```

(136)

WF_LASTPROCESS_USERID

WF_LASTPROCESS_USERID()

当該データについて、最後にワークフローに何らかの処理（承認、決裁など）を行った処理者の「ユーザID」(juser の userid項目)を返します。

この関数を利用する項目はデータベースに保存しない設定にしてください。常に最新の値を表示する必要があります。

```
WF_LASTPROCESS_USERID()
```

(137)

WF_LASTPROCESS_USERNAME

WF_LASTPROCESS_USERNAME()

当該データについて、最後にワークフローに何らかの処理（承認、決裁など）を行った処理者の「名前」(juser の name項目)を返します。

この関数を利用する項目はデータベースに保存しない設定にしてください。常に最新の値を表示する必要があります。

```
WF_LASTPROCESS_USERNAME()
```

(138)

---

(134) この関数はジョブ制御では利用できません。
(135) この関数はジョブ制御では利用できません。
(136) この関数はジョブ制御では利用できません。
(137) この関数はジョブ制御では利用できません。
(138) この関数はジョブ制御では利用できません。

FLOWSTATUS

FLOWSTATUS("モデル名",モデルの主キー項目(文字列型))

ワークフローの現在のノード名を文字列で返します。ノード名は通常「申請、承認、決裁」などとなっていますが、これはフローパターンの設定時に変更できます。第二引数は文字列型である必要があります。

```
FLOWSTATUS("leave", TOSTR(${appid}))
```

FLOWSTATUS関数が空文字を返す場合は、未申請を意味します。

> 上記は主キーを整数型と仮定し「TOSTR」関数で文字列に変換しています。 主キーが文字列型の場合は下記のように直接、項目名を指定できます。 FLOWSTATUS("leave", ${appid})

(139) (140) (141)

WF_EVENT

WF_EVENT("イベント名")

ワークフローの処理内容（新規登録、申請、承認、差し戻し、保留、却下、決裁、（申請者による）取り消し）を文字列で返します。[利用例...]

```
WF_EVENT("Start")
```

(142)

国際化対応

戻り値の文字列は国際化に対応しています。例えば日本語環境では "申請" を、英語環境では "Application" を返します。

引数の文字列	日本語	英語
Start	新規登録	Start
Application	申請	Application
Admit	承認	Admit
Back	差し戻し	Back
Suspend	保留	Suspend
Reject	却下	Reject
End	決裁	End
Cancel	取り消し	Cancel

WF_LASTEVENT

WF_LASTEVENT("モデル名",モデルの主キー項目(文字列型))

ワークフローの直近の処理内容（新規登録、申請、承認、差し戻し、保留、却下、決裁、（申請者による）取り消し）を文字列で返します。第二引数は文字列型である必要があります。

```
WF_LASTEVENT("leave", TOSTR(${appid}))
```

(143) (144)

---

(139) FLOWSTATUS関数を適用した項目はデータベースに保存しないようにしてください。（保存する、を設定しても内部では無視されるようになっています。）

(140) この関数はジョブ制御では利用できません。

(141) FLOWSTATUS関数を適用した項目を検索条件に指定する場合、特別な設定が必要です。 詳細は"モデルのワークフロー状態で検索する"をお読みください。

(142) この関数はジョブ制御では利用できません。

(143) この関数はジョブ制御では利用できません。

(144) 内部では WF_EVENT 関数を使っています。そのため国際化に対応しています。

WF_CURRENT_USERID

WF_CURRENT_USERID("モデル名", モデルの主キー項目(文字列型))

現在のノードに関する処理保留者（現在、アクションを起こすべき人）のログオンIDを返します。戻り値の型は文字列型です。

ワークフローのデータを新規登録したが申請していない場合、この値はデータを作成したログオンIDが入ります。

この関数を利用する項目はデータベースに保存しない設定にしてください。常に最新の値を表示する必要があります。

```
WF_CURRENT_USERID("leave", TOSTR(${appid}))
```

上記は主キーを整数型と仮定し「TOSTR」関数で文字列に変換しています。 主キーが文字列型の場合は下記のように直接、項目名を指定できます。 WF_CURRENT_USERID("leave", ${appid})

(145)

WF_NEXT_USERID

WF_NEXT_USERID("モデル名", モデルの主キー項目(文字列型))

次のノードに関する処理保留者（次に承認アクションを起こすべき人のログオンID）を返します。戻り値の型は文字列型です。 モデル名とモデルの主キー項目の値を指定することができます。

ワークフローのデータを申請した場合、この値は次のノードの承認者のログオンIDが入ります。（承認者が個人のとき）

この関数を利用する項目はデータベースに保存しない設定にしてください。常に最新の値を表示する必要があります。

```
WF_NEXT_USERID("leave", TOSTR(${appid}))
```

上記は主キーを整数型と仮定し「TOSTR」関数で文字列に変換しています。 主キーが文字列型の場合は下記のように直接、項目名を指定できます。 WF_NEXT_USERID("leave", ${appid})

(146)

WF_NEXT_USERID_MAIL

WF_NEXT_USERID_MAIL("モデル名", "モデルの主キー名", "メールアドレスを格納している項目名")

次のノードのメールアドレスを返します。次のノードとは、申請時であれば次の承認者へ、承認時であれば次の承認者（または決裁者）という意味です。差し戻しの場合は、差し戻されたときの承認者になります。グループを使わないことがわかっている場合に利用できます。

第一引数のモデル名と第二引数名の主キー名から、第三引数のメールアドレス格納項目の値を返します。戻り値は文字列型（メールアドレス）です。

```
WF_NEXT_USERID_MAIL("juser", "userid", "mailaddress")
```

(147) (148)

(145) この関数はジョブ制御では利用できません。

(146) この関数はジョブ制御では利用できません。

(147) この関数はジョブ制御では利用できません。

(148) 差し戻し先のノードが合議の場合、合議ルールに含まれる全員が対象となります。（合議で承認した人のみ、ではなく全員、です。）

WF_USERID

WF_USERID()

次のノードに関する処理保留者（次に承認アクションを起こすべき人のログオンID）を返します。戻り値の型は文字列型です。 WF_NEXT_USERID()関数の、引数なし版です。当該データに関する値を返します。

この関数を利用する項目はデータベースに保存しない設定にしてください。常に最新の値を表示する必要があります。

```
WF_USERID()
```

(149)

WF_CURRENT_GROUPID

WF_CURRENT_GROUPID("モデル名", モデルの主キー項目(文字列型))

現在のノードに関する処理保留者（現在、承認アクションを起こすべき人）のグループIDを返します。 戻り値の型は整数型です。 値が存在しない場合は、nullを返します。

この関数を利用する項目はデータベースに保存しない設定にしてください。常に最新の値を表示する必要があります。

```
WF_CURRENT_GROUPID("leave", TOSTR(${appid}))
```

上記は主キーを整数型と仮定し「TOSTR」関数で文字列に変換しています。 主キーが文字列型の場合は下記のように直接、項目名を指定できます。 WF_CURRENT_GROUPID("leave", ${appid})

(150)

WF_NEXT_GROUPID

WF_NEXT_GROUPID("モデル名", モデルの主キー項目(文字列型))

次のノードに関する処理保留者（次に承認アクションを起こすべき人）のグループIDを返します。 戻り値の型は整数型の配列です。 値が存在しない場合は、nullを返します。 モデル名とモデルの主キー項目の値を指定することができます。

この関数を利用する項目はデータベースに保存しない設定にしてください。常に最新の値を表示する必要があります。

```
WF_NEXT_GROUPID("leave", TOSTR(${appid}))
```

上記は主キーを整数型と仮定し「TOSTR」関数で文字列に変換しています。 主キーが文字列型の場合は下記のように直接、項目名を指定できます。 WF_NEXT_GROUPID("leave", ${appid})

(151)

外部連携

ISREST

ISREST()

(149) この関数はジョブ制御では利用できません。

(150) この関数はジョブ制御では利用できません。

(151) この関数はジョブ制御では利用できません。

現在のやりとりが、REST API 経由の場合はtrueを返します。そうでない場合（画面からの操作またはアップロード更新、ダウンロードなど）はfalseを返します。

HTTPSEND

HTTPSEND(urlを示す文字列)

HTTPSEND(urlを示す文字列, "ユーザID:パスワード") (8.0.4)

HTTPSEND(urlを示す文字列, 接続タイムアウト, 読み込みタイムアウト, リトライ回数) (8.0.4)

HTTPSEND(urlを示す文字列, 接続タイムアウト, 読み込みタイムアウト, リトライ回数, "ユーザID:パスワード") (8.0.4)

urlに示した（外部）サイトへアクセスし、ページ情報を取得します。

```
HTTPSEND("http://wagby.com")
HTTPSEND("http://wagby.com", "scott:tiger")
HTTPSEND("http://wagby.com", 10000, 10000, 2)
HTTPSEND("http://wagby.com", 10000, 10000, 2, "scott:tiger")
```

url部に JSONを返す REST API サーバを指定することで、外部の値を取得することに活用できます。後述するJSONPATH関数との組み合わせを想定しています。

接続タイムアウト、読み込みタイムアウト、リトライ回数を指定できます。省略時はそれぞれ接続タイムアウト30秒、読み込みタイムアウト30秒、リトライ回数3回となります。

BASIC認証のユーザ名とパスワードを指定できます。"ユーザID:パスワード" と記述します。

(152)

JSONPATH

JSONPATH(JSONオブジェクトが格納された文字列, JSON Path 文字列)

JSON形式の文字列を解析して、JSON Path で指定された値を取得します。

```
JSONPATH(${jsondata},"$.Items[0].Item.title")
```

上記の例では、jsondata項目に設定されたJSONデータを解析して、タイトルを取得するものです。

JSON Path は「Java JsonPath」というライブラリを使っています。http://jsonpath.curiousconcept.com/で、JSON Pathの動作を試すことができます。

HTTPSENDとJSONPATHを使った例

Wagby のスクリプトで外部サービスのREST APIを利用する例を示します。ここではPostcodeJP APIを使って、郵便番号から住所の変換を行います。

PostcodeJP が返すJSONのイメージ

次の URL にアクセスしたときに戻される JSON は次のとおりです。 https://apis.postcode-jp.com/api/v2/postcodes?postcode=1000001

```
{"data": [
    {
        "allCode": "13101", "stateCode": "13", "townCode": "101",
```

(152) このメソッドは HTTP GET のみに対応しています。POST や PUT などの用途にはお使い頂けません。またリクエストヘッダの指定なども行うことはできません。

```
        "zip": "1000001", "state": "東京都",
        ...
        "allAddress": "東京都千代田区千代田",
        ...
        "general": true, "office": false
    }
]}
```

スクリプト

customer モデルの postcode 項目に郵便番号がセットされているものとします。上の JSON 形式データの data[0].allAddress が住所に相当しますので、この値を customer モデルの address 項目にセットしてみます。

```
var ExcelFunction = Java.type("jp.jasminesoft.util.ExcelFunction");
var data = ExcelFunction.HTTPSEND(
    "https://apis.postcode-jp.com/api/v2/postcodes?postcode="
    + customer.postcode);
//print(data); // 取得したデータを表示
var allAddress = ExcelFunction.JSONPATH(data, "data[0].allAddress");
customer.address = allAddress;
//print(customer.address); // データの確認
```

REQUEST

REQUEST()

REQUEST("名前")

この関数は、Wagby とは異なるシステムで用意された Web フォーム（例：PHP や Perl などで用意したフォーム）から Wagby へ POST するといったケースで用いられることを想定しています。

指定した名前で、Java が管理するリクエストオブジェクトから値を取得します。 Wagby 以外のページから遷移されたとき、HTTP GET/POST パラメータを直接、受け取ることができます。戻り値の型は文字列型です。

```
REQUEST("id")
```

Webフォームから送信されてくる名前が、対象項目の名前（英語）と一致している場合、REQUEST関数の引数は省略できます。

REQUEST関数の戻り値の型は文字列型です。この値を数字型や日付型項目にセットする場合は、TEXT関数などを用いて文字列から数値、日付へ変換してください。

SESSION

(型)SESSION()

(型)SESSION("名前")

この関数はJavaによるカスタマイズ開発を行う方向けに用意されました。

指定した名前で、Java　が管理するセッションオブジェクトから値を取得します。戻り値の型は任意のため、必ず型を明示してください。

```
(String)SESSION("item1")
```

セッションオブジェクトのキー名が、対象項目の名前（英語）と一致している場合、SESSION関数の引数は省略できます。

SESSION関数は開発者がJavaコードをカスタマイズし、セッションオブジェクトに値を格納したものを「受け取る」ために使うことができます。値を受け取る場合、必ずセッションオブジェクトに格納したオブ

ジェクトの型でキャストしてください。（型の指定をしないか、誤った型を指定するとコンパイル時にエラーとなります。）

型についての補足

格納するオブジェクトと「型」の関係は次のとおりです。

格納するオブジェクト	明示する「型」の記述方法
整数型	(Integer)
文字列型	(String)
日付型	(java.sql.Date)
時刻型	(java.sql.Time)
日付時刻型	(java.sql.Timestamp)
ファイル型(ファイル名)	(String)
1バイト整数	(Byte)
2バイト整数	(Short)
4バイト整数	(Integer)
8バイト整数	(Long)
4バイト浮動小数点数	(Float)
8バイト浮動小数点数	(Double)
固定値（作成日/更新日）	(java.sql.Timestamp)
その他（URL型/メールアドレス型など）	(String)

ATTRIBUTE

(型)ATTRIBUTE("名前")

一回の処理（ユーザがボタン押下など何らかの操作を行なった結果サーバとの通信が生じ、サーバでの処理が終わった後にブラウザの画面が書き換わるまで）のたびに「p.request」というオブジェクトが生成されます。処理が終わると破棄されます。つまり p.request オブジェクトの「生存期間」は一回の処理の開始から終了まで、となります。

ATTRIBUTE　関数は、この　p.request　に格納された値を取得するものです。戻り値の型を明示するため、ATTRIBUTE 関数の前に型を指定してください。オブジェクトと型の関係は SESSION 関数の説明と同じです。

p.requestに値をセットする

この p.request オブジェクトに「値」を格納する場合は p.request.setAttribute メソッドを用います。引数はキー名（文字列）と、格納したいオブジェクトです。この関数は提供されていません。スクリプトを使って値をセットするようにしてください。"親キー以外の項目値を用いて子モデル一覧表示を行う"にこの例を紹介しています。

XML2JSON (8.3.1)

XML2JSON(XML形式の文字列)

XML形式の文字列を、JSON形式の文字列に変換します。Wagbyのストアモデルやプレゼンテーションモデルは toString メソッドを呼び出すことでXML形式の文字列になります。これを引数に渡すことで、JSON形式の文字列へ変換することができます。

なお、次の動作は仕様となります。

- チェックボックスや繰り返し項目など、同一の項目名は JSON の配列となります。このときXML形式の表示順序は保証されません。(JSONの仕様では、順序は不定となっているため。)
- 繰り返しコンテナを含むモデルでは、コンテナのデータ件数によって JSON の構造が変わります。
  - コンテナの件数が0件の場合、JSON にコンテナIDのキーも存在しない。
  - コンテナの件数が1件の場合、JSON にコンテナIDのキーが追加され、値はコンテナデータの連想配列となる。
  - コンテナの件数が2件以上の場合、JSON にコンテナIDのキーが追加され、値はコンテナデータの連想配列の配列となる。

  このため、JSONを利用する側では、コンテナIDのキーが存在するか、値が配列かどうかの判定を行なうようにしてください。

(153)

その他

BARCODE (8.3.0)

BARCODE(文字列, 出力方式)

バーコードを出力します。第二引数は "codabar", "code39", "code128", "ean-13", "qr", "jppost" という方式を指定することができます。

使い方は「出力制御（スタイル）>バーコード出力」をお読み下さい。

CLIENTHOSTNAME (8.3.0)

CLIENTHOSTNAME()

利用者(Webブラウザ)が動作しているPCのホスト名を返します。戻り値は文字列型です。ホスト名が未設定の場合、IPアドレスを返します。

(154) (155)

CHANGELOG

CHANGELOG()

更新履歴を取得します。使い方は「繰り返しコンテナでデータの変更履歴を管理する」をお読み下さい。

この関数は上記ページに記載のとおり、繰り返しコンテナ内での利用を前提としています。それ以外の箇所では使わないようにしてください。

EXEC

EXEC(実行ディレクトリ, コマンド, 引数...)

第一引数のディレクトリ位置で、第二引数のコマンドを実行します。コマンドライン引数は第三引数以降、コンマ区切りで複数指定できます。戻り値は実行結果（文字列）です。

Wagbyがインストールされたフォルダのディレクトリリストを取得する

```
EXEC("../..", "cmd", "/C", "dir")

EXEC("../..", "ls", "-al")
```

(156)

Wagbyアプリケーションが実行されたときのカレントディレクトリは wagbyapp/bin となっています。上の例は、相対パス表記でディレクトリを指定しています。

実行時フォルダの指定方法

カレントフォルダは "." で表現します。"" と空文字を渡すことはできません。 フォルダの区切り文字 "/" で終わらないようにしてください。例えば "../" ではなく "../." と記述します。

フォルダを作成する

(153) 内部では json.org で公開されている Java 用の JSON ライブラリ（JSON-java）を用いています。

(154) この値は、ログファイル（system.log）に記録されるクライアントの情報（ホスト名またはIPアドレス）と同じです。

(155) ジョブやアップロード更新画面での利用時は、この値は空白となります。

(156) 最初の例は Windows OS 用で、次の例は Linux OS 用です。cmd.exe は Windows OS に標準で含まれている DOS コマンドです。

Wagbyがインストールされたフォルダに work1 という名前のフォルダを新しく用意する例です。

```
EXEC("../..", "cmd", "/C", "mkdir", "work1")
EXEC("../..", "mkdir", "work1")
```
(157)

SERVERHOSTNAME (8.3.0)

SERVERHOSTNAME()

Wagbyアプリケーションが動作しているサーバのホスト名を返します。戻り値は文字列型です。ホスト名が未設定の場合、IPアドレスを返します。

Me

Me()

擬似関数 Me（注:二文字目の 'e' は小文字）は、この式を記述した項目名を指し示します。例えば item1 項目の式で Me() を使うと、Me() の部分が ${item1} に置換されます。

例

```
IF (Me() > 0; 100, 0)
```

Me() の用途は、同じ式を複数の項目に設定する場合に、項目名部分を統一できることです。コピー&ペーストした式を再利用できます。

初期値

基本的な書き方

定義方法

項目毎に初期値を設定できます。画面と、値の種類、具体的な設定値を指定します。

(157) 最初の例は Windows OS 用で、次の例は Linux OS 用です。cmd.exe は Windows OS に標準で含まれている DOS コマンドです。

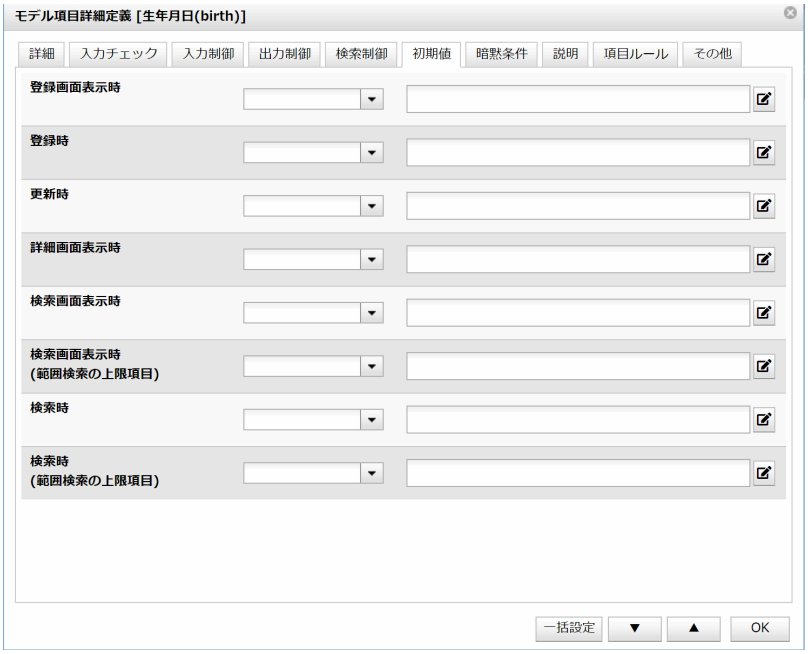


図 初期値の設定欄

画面の指定

各画面における初期値設定のタイミングは次のとおりです。

タイミング	説明
登録画面表示時	「登録画面へ」ボタンをクリックしたときに新規登録画面に初期値が設定されます。
登録時	新規登録画面で、データを登録するタイミングで（「保存」ボタンがクリックされた時に）初期値が設定されます。データベースには新規登録画面での入力にかかわらず、初期値として指定された値が格納されます。
更新時	更新画面で、データを更新するタイミングで（「保存」ボタンがクリックされた時に）初期値が設定されます。データベースには更新画面での入力にかかわらず、初期値として指定された値が格納されます。
詳細画面表示時	詳細画面の表示直前に初期値がセットれます。詳細画面では、データベースに格納されている値にかかわらず、初期値が設定された状態で表示されます。（なお、更新画面を開いた直後も本設定が反映されます。）[注意点...]
検索画面表示時	検索画面で検索条件の項目に初期値が設定されます。範囲検索が指定されている場合は、検索条件の下限項目（「～」の左側）に初期値が設定されます。
検索画面表示時（範囲検索の上限項目）	検索画面において範囲検索が指定されている場合、検索条件の上限項目（「～」の右側）に初期値が設定されます。
検索時	検索画面からデータを検索するタイミングで初期値を設定します。範囲検索が指定されている場合は、検索条件の下限項目（「～」の左側）の初期値となります。
検索時（範囲検索の上限項目）	検索画面からデータを検索するタイミングで初期値を設定します。範囲検索が指定されている場合、検索条件の上限項目（「～」の右側）の初期値となります。

種類の指定

指定できる初期値の種類は次のとおりです。

種類	説明
（省略時）	このモデル項目の型になります。（文字列、数値、日付...）
文字列（ランダム）	モデル項目の方が文字列の場合に利用できます。（詳細...）
ログオンID	モデル項目の方が文字列の場合に利用できます。（詳細...）
所属グループ	（準備中）
四則演算	計算式を使って値を設定します。項目名はWagby ルールによって置換されます。関数を用いることもできます。
Javaコード	Javaコードを直接、埋め込みます。代入の右辺に相当するコードになります。

関数を用いる場合、必ず「四則演算」を指定してください。これを忘れるとコンパイルエラーになります。

図 関数利用時の設定

初期値が設定されるタイミング

登録・詳細・検索画面表示時

次の順序で処理されます。

1. 初期値の設定
2. モデル参照関係の解決
3. 自動計算

そのため、初期値の値を自動計算式に適用することができます。逆に、参照関係を解決した後の値や、自動計算結果を初期値としてセットすることはできません。(158)

登録時・更新時

「登録時」「更新時」では、初期値を設定するタイミングはデータベースに保存する直前となります。

そのため初期値を指定した項目に対して自動計算が適用されたり、CSVアップロード更新が値がセットされた場合でも、最終的にこれらの値は初期値によって上書きされます。

式にモデル参照項目を含める場合

計算式中にモデル参照項目を用いる場合、この項目がDB保存有りとなっているかどうかで取得できる値が変わります。

DB保存の項目を式に含めたとき、DBに格納されている値を利用します。DB保存しない項目の場合、整数型項目は0、文字列型項目は空白として扱われます。

(158) このような場合は、生成されたコードをカスタマイズしてください。カスタマイズ方法の詳細は「Wagby Developer Network」をお読みください。

別画面または別システムとの連携

REQUESTやSESSION関数を使うと、別画面（または別システム）からパラメータを受け取り、初期値として設定することができるようになります。

この表記は、Wagby とは異なるシステムで用意された Web フォーム（例：PHP や Perl などで用意したフォーム）から Wagby へ POST するといったケースで用いられることを想定しています。

スクリプトを利用する

初期値の設定欄に代えて、スクリプトを利用することもできます。スクリプトの「ヘルパ > 登録」および「ヘルパ > 更新」のタイミングで値をセットしてください。

次のケースではスクリプトを利用します。

- 初期値を計算で求めたいが、複雑な条件を記述する必要がある。
- モデル参照項目で、参照先モデルの主キーが複合キーである。

文字列型で指定できるオプション

関数利用時の注意点

初期値に関数を適用する場合、入力フィルタとの関係に注意してください。

例えば、文字列型の項目にLPAD関数を用いて空白を付与した初期値を作成したとしても、登録時に入力フィルタの動作によって空白が除去されます。

この場合は、フィルタを「文字列（一般）」から「文字列（最小限）」に変更することで対応できます。

URLの初期値

URL型項目では、初期値の省略時は「http://」が設定されています。

応用例：ランダム文字列をセットする

初期値に英大文字、小文字、数字から構成されるランダムな文字列を指定します。値の部分には桁数を指定します。

想定用途は、初期パスワードの自動生成です。

(159)

応用例：ログオンIDをセットする

ログオンしているアカウントのIDが設定されます。（例：admin）

日付・時刻・日付時刻型で指定できる特殊表記 now

定義方法と例

初期値に ”now” と表記することで、現在の日付時刻を指定することができます。

(159) 内部では RANDSTR 関数を利用しています。

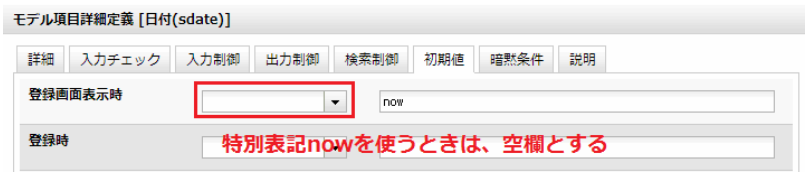


図 now表記の設定

「now」を使う場合、種類の欄は空欄としてください。

「now+1」は翌日を表現します。「now-1」は前日を表現します。その他、以下に示すような指定を行うことができます。

種類	説明
now	当日
now+1	翌日
now-1	前日
now+1M	翌月
now-1M	前月
now:firstday	当月の最初の日を表します。例）本日が2016-02-14の場合、 "2016-02-01" となります。
now:lastday	当月の最後の日を表します。例）本日が2016-02-14の場合、"2016-02-28" となります。
now:everyXX	次の XX 日を表します。例）本日が2016-02-14 で XX が 25 の場合、"2016-02-25" となります。本日が 2016-02-14 で XX が 10 なら"2016-03-10" となります。
now+1M:firstday	翌月の最初の日を表します。本日が2016-02-14の場合、 "2016-03-01" となります。

例を図に示します。

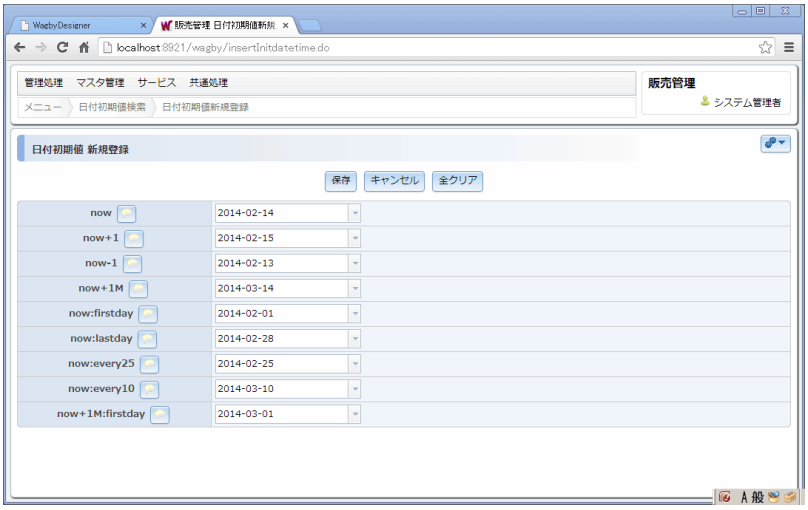


図 now指定の例

Wagby は now 表記とは別にNOW関数が提供されています。（またはTODAY関数もあります。この場合、NOW関数は日付時刻型で、TODAY関数は日付型で指定します。（異なる組み合わせはエラーになります。）

(160)

(160) now+1やnow-1のような動作を(Wagbyが提供する)日付時刻関数の組み合わせで実現することもできます。

関数との組み合わせ

now 表記を用いた場合の時刻部分は、実際に動作したタイミングの時刻がセットされます。 時刻部分をゼロにする場合は、CLEARHMS関数を適用してください。

```
CLEARHMS(now:firstday)
```

型とnowの関係

型が「日付」「時刻」「日付時刻」によって、nowの扱いが異なります。

型	now が指す内容	例
日付	本日の日付 2014/02/14　など	「now+1」は翌日（本日日付の1日後）。 「now+1M」は本日日付の1ヶ月後。
時間	現在の時刻 11:46:13　など	「now+1」は現在の時刻の1秒後。 「now+3600」は現在時刻の1時間後（3600秒後）。
日付と時間	現在の日付と時刻 2014/02/14 11:46:13　など	「now+1」は1日後の現在時刻を指す。左の例では「2014/02/15 11:46:13」となる。

チェックボックス

すべてを選択する

初期値に "ALL" と指定すると「すべてを選択した」状態になります。（「種類」の欄は空白にしてください。）

図 初期値にALLを指定する。

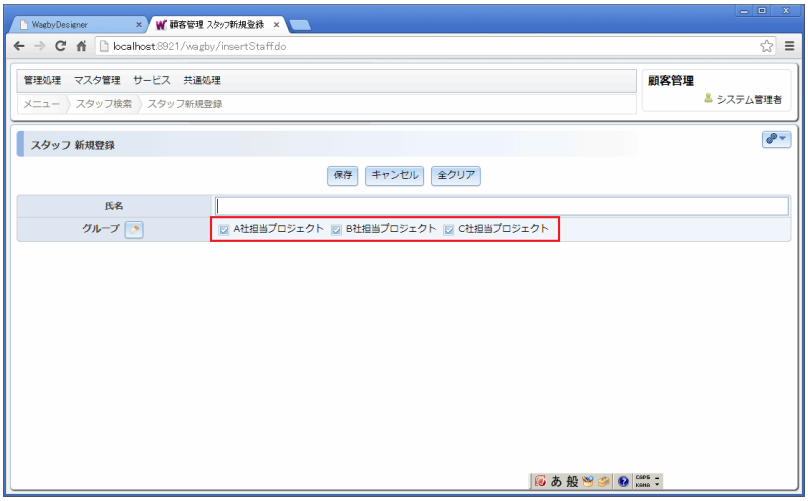


図 例

本設定は次の制約があります。

・　自動計算式との併用はできません。"ALL" のみの指定となります。
・　参照先モデルの主キーが日付型の場合、この指定を行うことができません。
・　絞り込み設定は無視されます。（すべて選択された状態になります。）

「所属グループ」の利用

初期値の種類に「所属グループ」を指定すると、利用者が所属するグループをチェック状態とすることができます。

ここでは、モデルに項目「グループ」を用意した例を示します。

項目名（日本語）	項目名（英語）	型	参照先モデル	参照モデルの内容
グループ	itemgroup	他のストアモデルの参照（チェックボックス）	jgroup	jgroupname

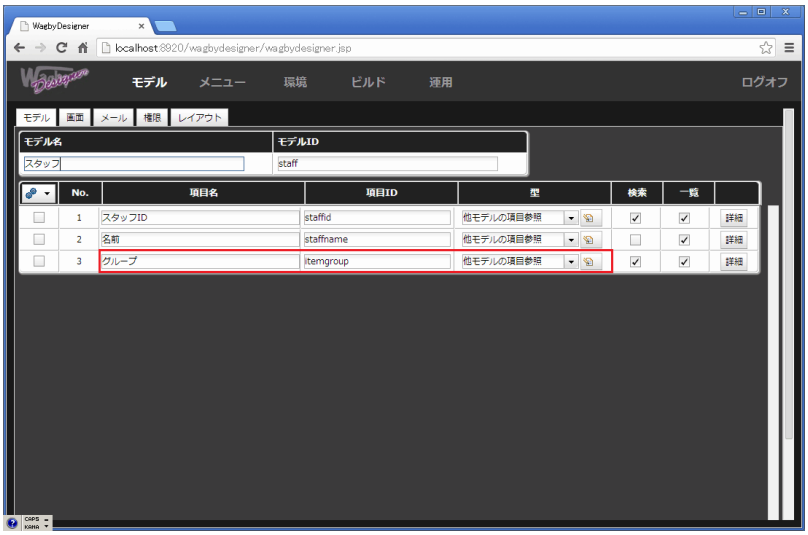


図　「グループ」項目の定義

新規登録時ならびに検索画面表示時において、同項目の初期値を「所属グループ」とします。

図　初期値を「所属グループ」とする。

グループ "総務部" に所属しているアカウントでログオンしたときの、検索条件部の表示例を図に示します。 所属しているグループがチェックされていることがわかります。

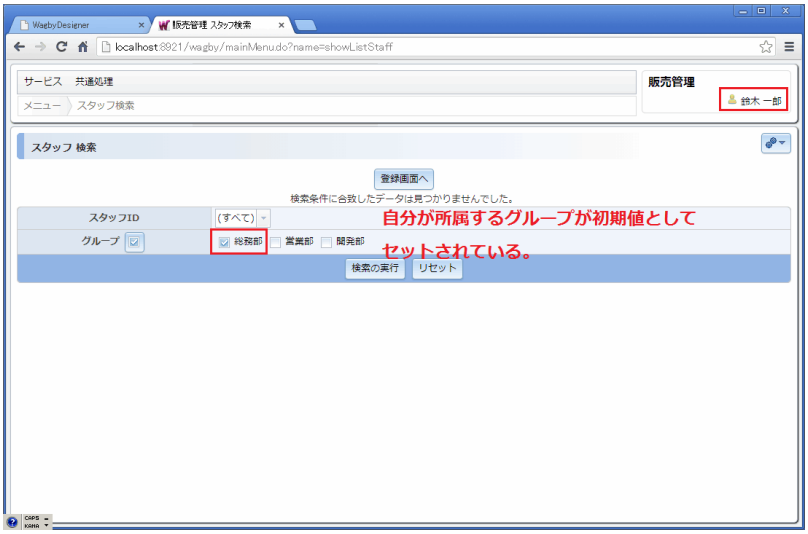


図 所属グループのみチェックされている

関数を適用する

初期値に関数を適用する場合、「初期値の種類」を「四則演算」としてください。（初期値の種類を省略すると、関数を使うことはできません。）

関数を用いる場合は配列の値を返すことが必要です。（チェックボックスは複数の値を保持するため、内部では配列形式でデータを扱います。）そのため単一の値を返す場合であっても ARRAY_INT 関数などで括ってください。

例：権限によって初期値を変える。

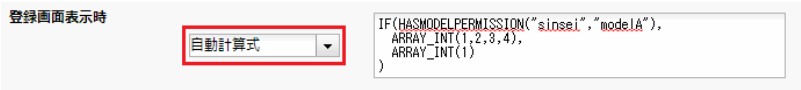


図 サンプル

```
IF(HASMODELPERMISSION("sinsei","modelA"),
  ARRAY_INT(1,2,3,4),
  ARRAY_INT(1)
)
```

REQUEST関数利用時の注意点

初期値に REQUEST 関数を適用すると、URLのパラメータに値を埋め込むことができます。 次のように指定することで、チェックボックスをすべて選択した状態にすることができます。

```
http://localhost:8921/wagby/insertTest.do?checkboxitem=ALL&action_New=
```

複数の項目を選択状態とする、という指定には対応していません。 次の指定は、最初の "10" だけがチェックされた状態になります。

```
http://localhost:8921/wagby/insertTest.do?item=10&item=30&item=50&action_New=
```

計算式によって初期値を求める

ラジオボタンへの適用例（三項演算子との組み合わせ）

次の例は、ラジオボタンの項目の初期値に式を記述した例です。複数の条件を満たすことができれば "1" をセットします。そうでなければ null をセットし、ラジオボタンを未選択状態とします。三項演算子を使うことで1行で記述しています。

以下の例は繰り返し項目from_addressの文字列を結合した中に "info@jasminesoft.co.jp" が含まれ、かつ項目user_accountが空白でなく、かつ項目user_accountに "@jasminesoft.co.jp" が含まれておらず、かつ項目subjectに "添付ファイル" という文字列が含まれていないという条件をすべて満たした場合は 1 を初期値とする。そうでなければ null すなわち初期値は設定しない、というものです。

```
AND(
  ISINCLUDE(JOIN(${from_address}), "info@jasminesoft.co.jp"),
  NOT(ISBLANK(${user_account})),
  NOT(ISINCLUDE(${user_account}, "@jasminesoft.co.jp")),
  NOT(ISINCLUDE(${subject}, "添付ファイル"))
) ? 1: null
```

関連するページ

- 空文字とnull
- 三項演算子

チェックボックスへの適用例

チェックボックスは複数の値を配列として管理しています。 そのため式中に関数を用いる場合は配列の値を返すようにしてください。 単一の値を返す場合であっても ARRAY_INT 関数などで括ってください。

権限によってチェックボックスの初期値を変える例を示します。

```
IF(HASMODELPERMISSION("sinsei","modelA"),
  ARRAY_INT(1,2,3,4),
  ARRAY_INT(1)
)
```

関連するページ

- ARRAY関数ファミリー
- HASMODELPERMISSION関数

主キーが複合キーのモデルを参照する項目

モデル model1 の項目 item1 はモデル参照であり、参照先モデル model2 が複合主キー pkey1, pkey2 となっている構成を想定します。

設計情報の上では "item1" という項目は一つですが、Wagbyの内部では二つの値を保持するフィールド（値を保持する入れ物）が用意されています。フィールドの名前は "item1_pkey1", "item1_pkey2" となっています。

(161)

この "item1" に初期値を設定する場合はスクリプトを利用します。スクリプトの「ヘルパ > 登録」および「ヘルパ > 更新」のタイミングで、値をセットしてください。例えば次のようなスクリプトになります。

```
model1.item1Pkey1 = (pkey1の型に準拠した値);
model1.item1Pkey2 = (pkey2の型に準拠した値);
```

スクリプトでは、項目IDはキャメル記法を使います。

(161) この命名規則の詳細は Appendix > 複合キーの内部的な表現をお読みください。

版の管理

例

図は、モデルに「版」という項目を用意した例です。ここでいう「版」とは、更新の度に版数がカウントアップするものです。

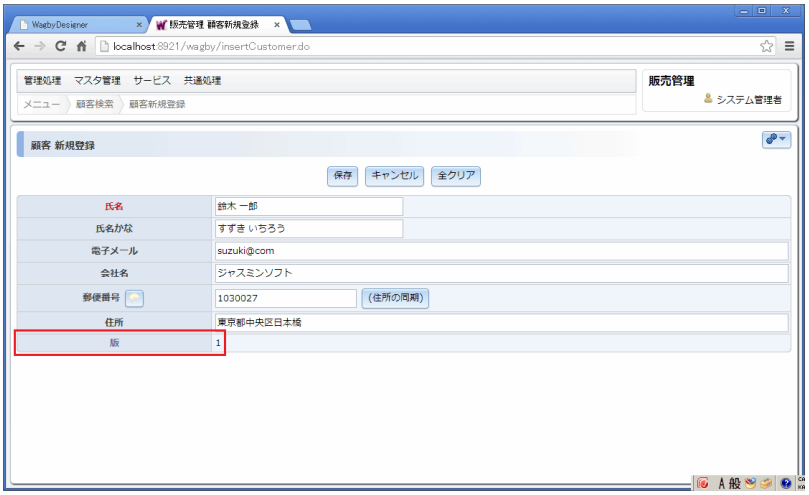


図 新規登録時は、版の初期値が' 1' となっている

データを更新すると、版が' 2' とカウントアップする。

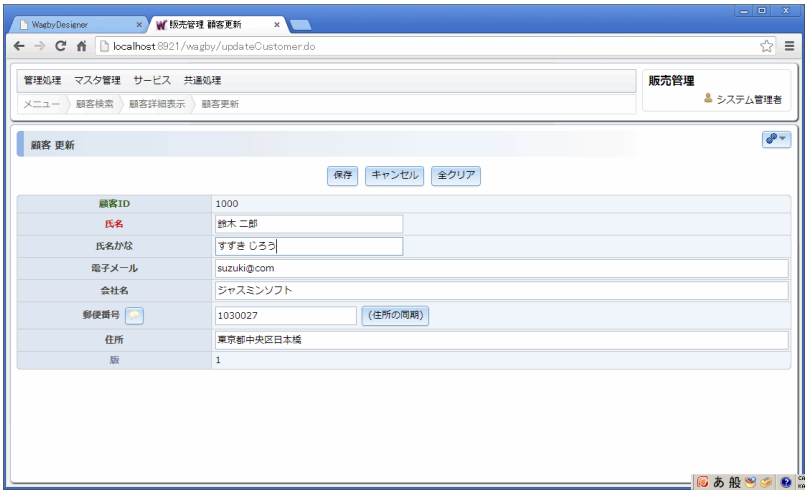


図 更新画面を開く

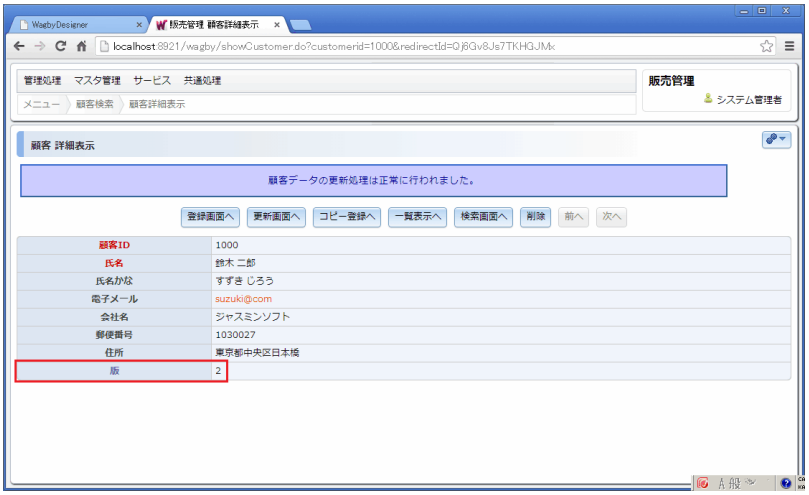


図 更新直後の状態。版がカウントアップされている。

定義方法

項目名を「版」（英語名 revision）とします。型は数値型です。

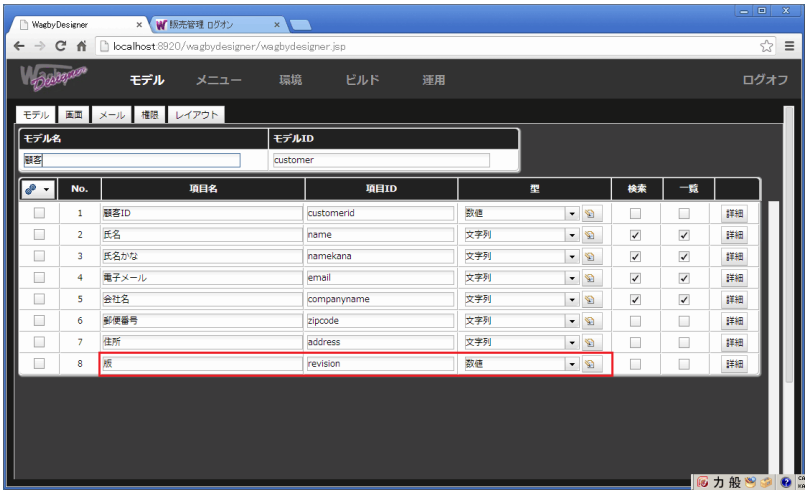


図 項目「版」を用意する

初期値を次のように設定します。

タイミング	種類	値
登録画面表示時	整数	1
更新時	自動計算式	${revision}+1

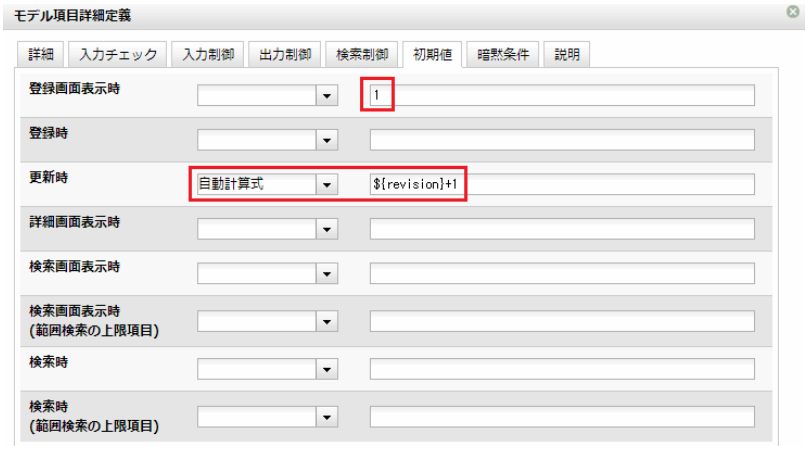


図 初期値の設定

入力チェックの設定で「全画面」を「読み込み専用」に設定します。

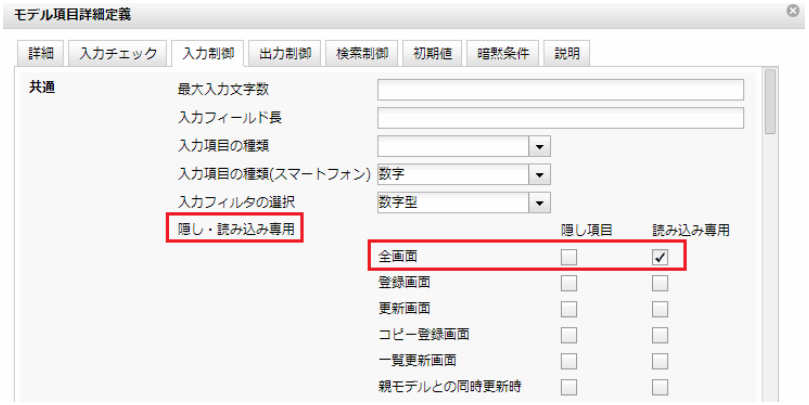


図 読み込み専用を設定する

自動計算に指定した項目は自分自身を参照することはできませんが、初期値は可能です。

主キーオプション

用意されたパターン

0 詰めの順序

例

0詰め6桁のキー構成を説明します。

図 0詰め6桁のキー

定義方法

図のようなモデルを用意します。主キー項目 ″ID″ は文字列型とします。

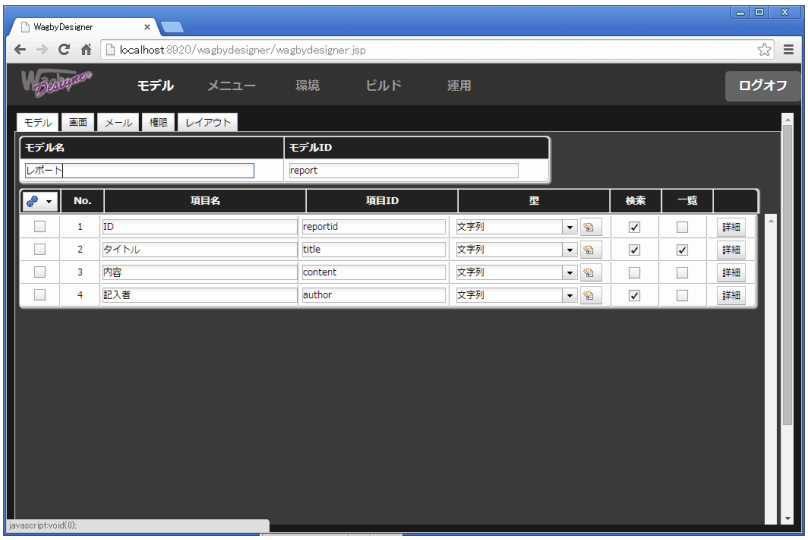


図 レポートモデルの定義

主キーを次のように設定します。

指定	内容
順序	使用する
開始番号	(数値)
順序部分のパディング指定	0, 6 (″0″詰め6桁)

補足

- 文字列型項目にも順序を適用できます。
- パディング指定では埋める文字（上の例では ″0″）は1文字のみを指定します。桁数の制限はありません。

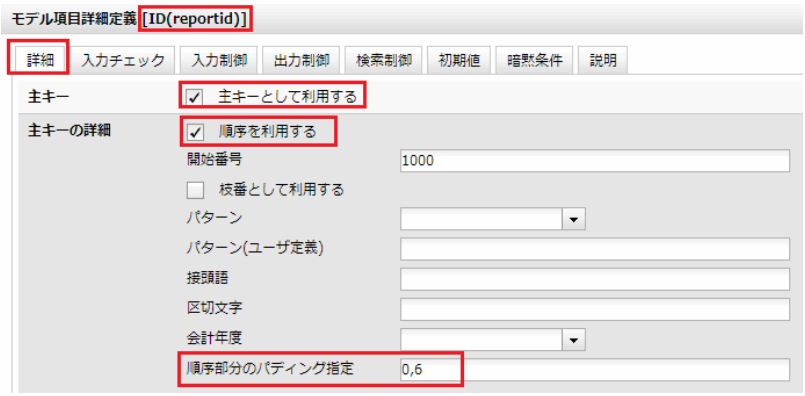


図 順序の定義

接頭語+区切文字+順序

例

「任意の接頭語」＋「区切文字」＋「自動採番」の具体例として〝営業報告-001000〟のようなキー構成を説明します。

図 接頭語+区切文字+自動採番主キー

定義方法

主キーを次のように設定します。

指定	内容
順序	使用する
開始番号	(数値)
パターン	[接頭語][区切文字][順序]
接頭語	(任意の文字列)
区切文字	(任意の文字列)
順序部分のパディング指定	0,6 (〝0〟詰め6桁)

補足

- 文字列型項目にも順序を適用できます。
- 接頭語、区切り文字は1文字以上指定できます。
- パターンに〝[接頭語][区切文字][順序]〟を指定してください。

・　パディング指定では埋める文字（上の例では ″0″）は1文字のみを指定します。桁数の制限はありません。

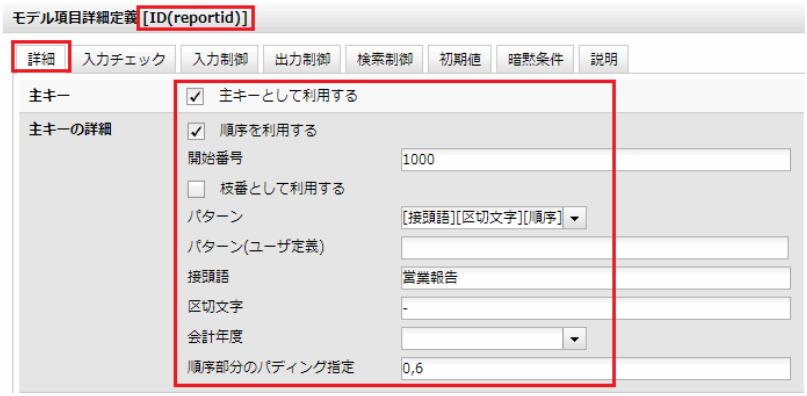


図 順序の定義

日付+区切文字+順序

例

「日付」＋「区切文字」＋「自動採番」の具体例として ″2014-001000″ のようなキー構成を説明します。

図 日付+区切文字+自動採番主キー

定義方法

主キーを次のように設定します。

指定	内容
順序	使用する
開始番号	(数値)
パターン	平成〇〇[区切文字][順序]平〇〇[区切文字][順序]H〇〇[区切文字][順序]YYYY[区切文字][順序]
接頭語	(任意の文字列)
区切文字	(任意の文字列)
会計年度	(空白)4月開始
順序部分のパディング指定	0,6 (″0″詰め6桁)

補足

- 文字列型項目にも順序を適用できます。
- パターンは接頭語が"平成","平","H"もしくは西暦4桁を選択できます。
- 会計年度が空白の場合は1月～12月となります。"4月開始"を選択すると4月～3月となります。8.1.1
- パディング指定では埋める文字（上の例では "0"）は1文字のみを指定します。桁数の制限はありません。

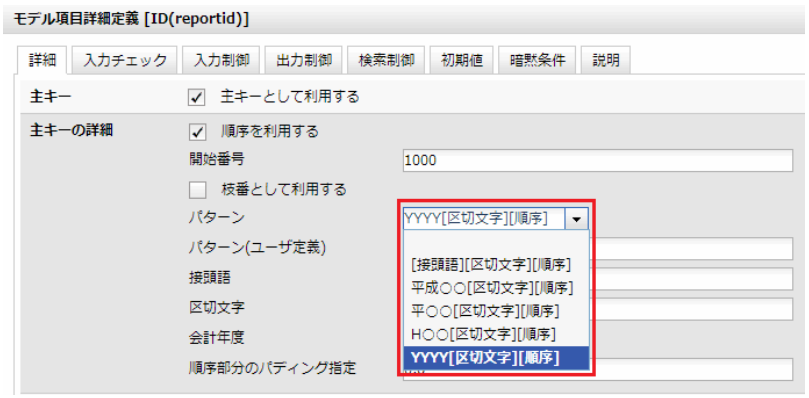


図 順序の定義

順序に関する注意

順序は単調に増加し続けます。年が変わってもリセットされることはありません。　例えば　"平成26年-12345" という表記を設定した状態で、翌年になった場合、次のキーは "平成27年-12346" となります。

年毎に順序を変更する場合はカスタマイズコードの作成を行ってください。

任意の書式を設定する（ユーザ定義パターン）

ステータスコード + 年（西暦下二桁）+ 月（ゼロ詰め）+ 連番（ゼロ詰め四桁）

例

主キーに次のルールを適用します。

ステータスコード + 年（西暦下二桁）+ 月（ゼロ詰め）+ 連番（ゼロ詰め四桁）

図 新規顧客のレポート,主キーが114041000となっている

定義方法

「レポート」モデルの全体構成を図に示します。

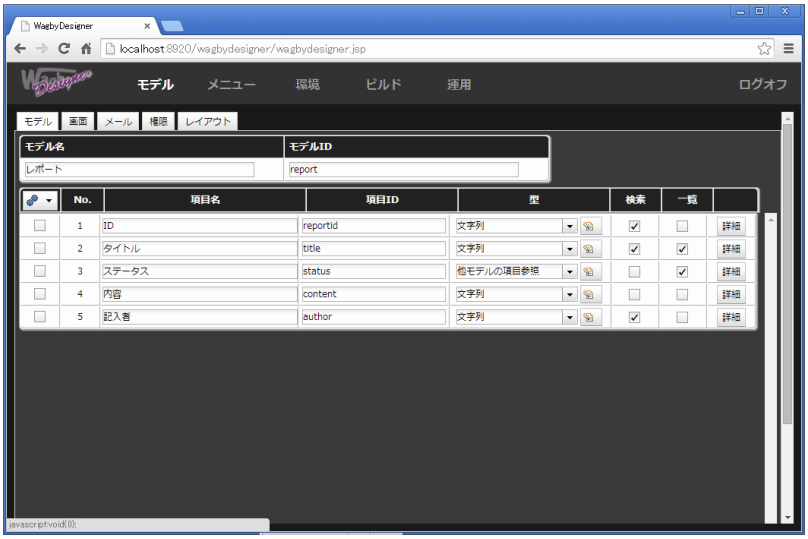


図 レポートモデル

主キー項目「ID」は文字列型で、かつ、順序を用いるとします。 ユーザ定義式に次のように設定します。

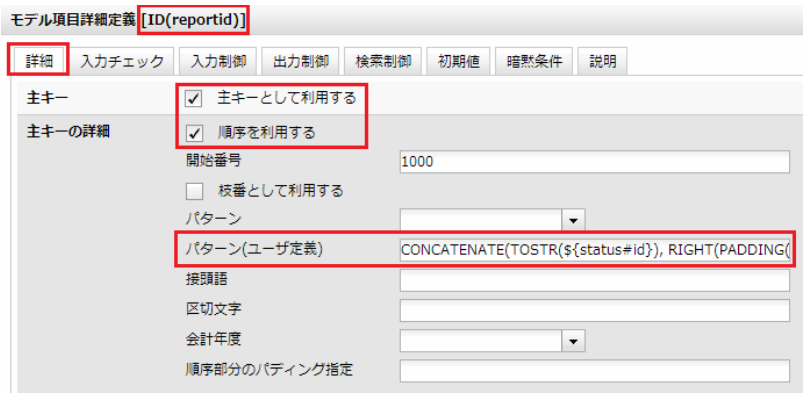


図 主キーの詳細設定

```
CONCATENATE(
  TOSTR(${status#id}),
  RIGHT(
    PADDING("0",2,YEAR(TODAY())),
    2),
  RIGHT(
    PADDING("0",2,MONTH(TODAY())),
    2),
  SEQ()
)
```

- ${status#id}と記述することで、ステータス項目のID値を取得することができます。
- CONCATENATE 関数は、コンマ区切りで指定された複数の文字列を連結します。
- CONCATENATE 関数への引数はすべて文字列であるため、TOSTR 関数で数字を文字列に変更することを行っています。
- RIGHT 関数は、文字列の右からN桁までを取り出します。
- PADDING 関数は、文字列に指定文字を詰めて、切り出します。
- YEAR 関数は、日付型の値から「年」の部分を取り出します。
- MONTH 関数は、日付型の値から「月」の部分を取り出します。

・　TODAY 関数は本日の日付を返します。
・　SEQ()　は、この主キー指定部だけで使える特別な関数です。自動採番された値（順序）を意味します。

SEQ()は自動計算式や、初期値で利用することはできません。

「ステータス」モデル(status_m)の初期値を示します。"新規" はコード1を、"既存" はコード2をそれぞれ割り当てます。

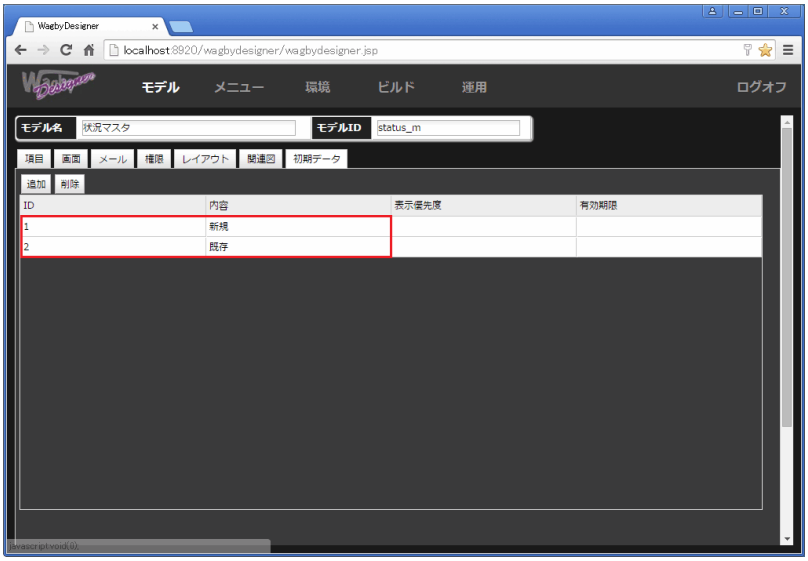


図 ステータスモデルの初期値

固定コード + 年度（西暦四桁）+ 連番（ゼロ詰め六桁）

例

主キーに次のルールを適用します。

固定コード（"PJ"）+ 年度（西暦四桁）+ 連番（ゼロ詰め六桁）

主キー項目「ID」は文字列型で、かつ、順序を用いるとします。 ユーザ定義式に次のように設定します。

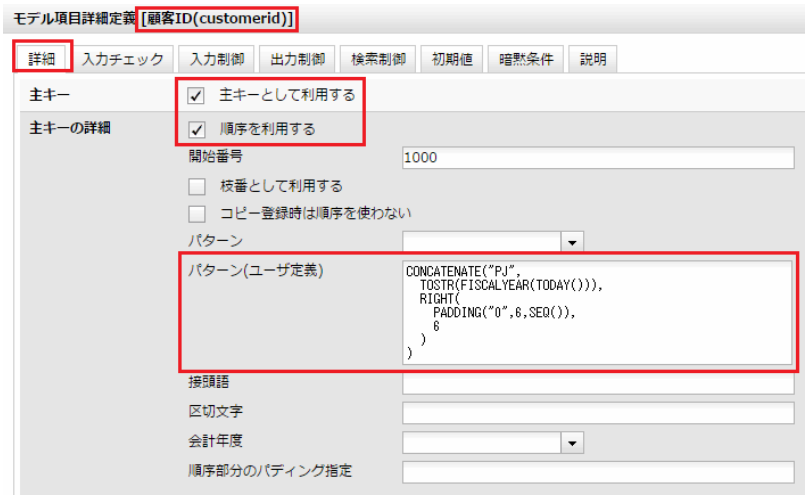


図 主キーの詳細設定

```
CONCATENATE("PJ",
 TOSTR(FISCALYEAR(TODAY())),
 RIGHT(
```

```
    PADDING("0",6,SEQ()),
    6
  )
)
```

- CONCATENATE 関数は、コンマ区切りで指定された複数の文字列を連結します。
- CONCATENATE 関数への引数はすべて文字列であるため、TOSTR 関数で数字を文字列に変更することを行っています。
- FISCALYEAR 関数は、年度を返します。
- PADDING 関数は、文字列に指定文字を詰めて切り出します。
- SEQ() は、この主キー指定部だけで使える特別な関数です。自動採番された値（順序）を意味します。

複合キーによる枝番

例1 主キーの手動入力

動作

「顧客ID」と「枝番」を用意した例を図に示します。（それぞれが主キーとなっています。） 枝番の初期値は 1 としています。

図 顧客IDと枝番を用意した例

ここでコピー登録機能を使い、同じ顧客IDで登録してみます。

主キー「顧客ID」は入力する方式となっており、同じ顧客IDを入力したときに枝番が付与されるという動作を実現しています。

登録・更新画面では枝番は自動的に非表示となります。

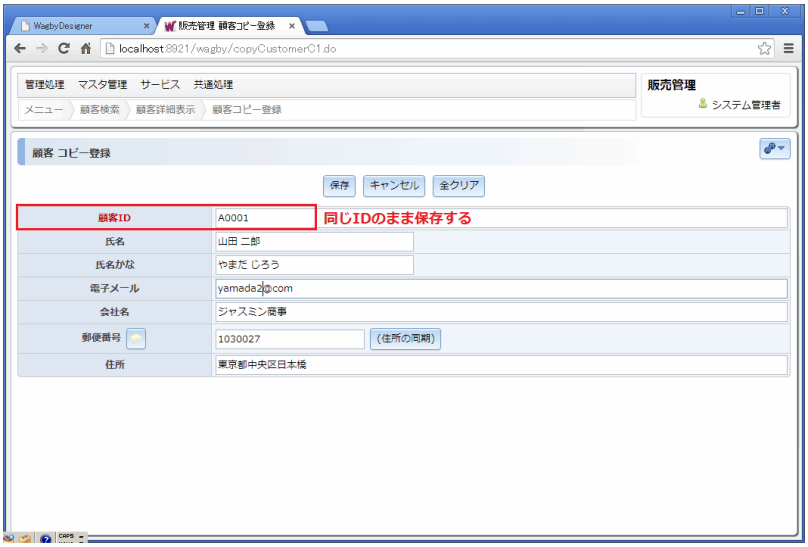


図 同じ顧客IDでデータを登録する

枝番が増加します。

図 枝番が2になった

枝番は、データ1件毎に管理されています。顧客IDが変われば、枝番は1になります。

図 異なる顧客IDの例

定義方法

顧客モデルに「顧客ID」と「枝番」項目を用意した例を図に示します。

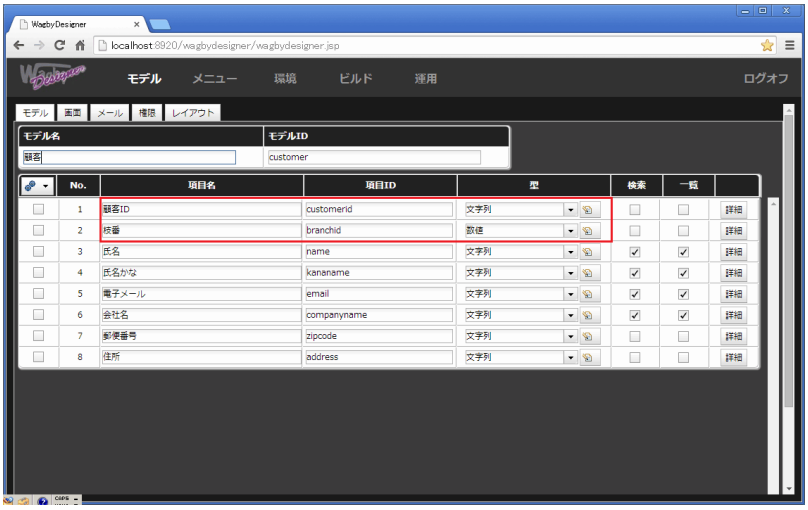


図 顧客モデル

顧客IDは主キーとします。この例では文字列とし、入力できるようにしました。（自動採番のための ″順序″ は使わないようにしています。）

図 顧客IDの定義（順序を用いない）

枝番も同じく主キーとしています。かつ「枝番として利用する」を有効にしています。開始番号は1としました。

図 枝番の定義

この枝番指定は「主キーとして利用する」が有効でないと機能しません。（枝番を利用するモデルは、必ず複合キーになります。）

例2 コピー登録時に枝番を付与する

動作

主キーに順序を用いる（自動採番）が、コピー登録時は順序を使わない設定と併用することで「コピー登録時のみ（主キーの重複が発生するため）枝番が付与される」という例を説明します。

新規登録画面です。「顧客ID」と「枝番」はいずれも非表示です。

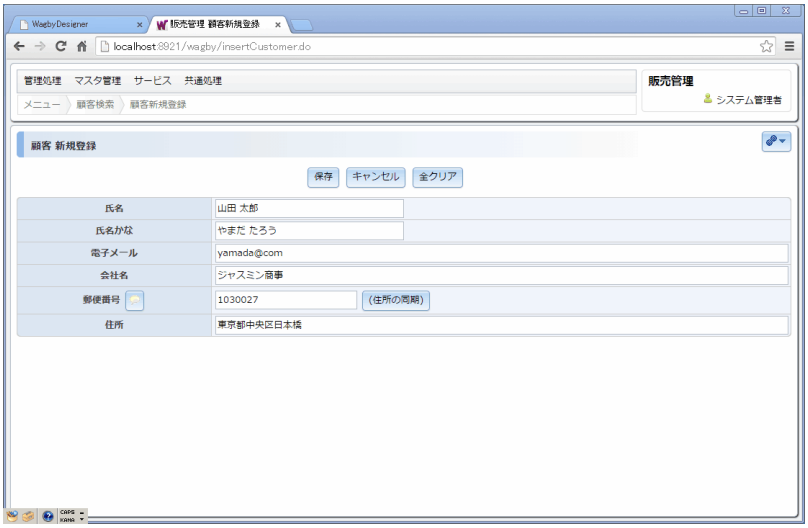


図 新規登録画面（順序を利用）

新規登録直後の画面です。枝番は ″1″ となっています。

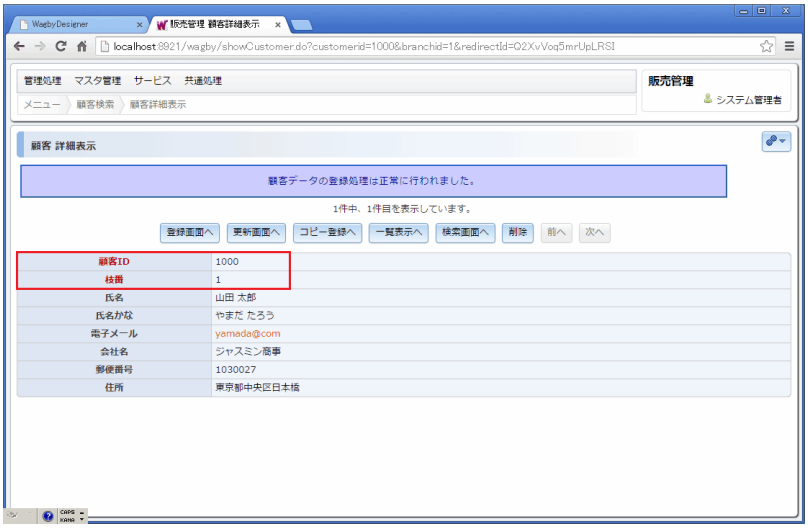


図 新規登録直後の詳細画面

次にコピー登録を行ってみます。コピー登録では順序を使わない設定としているため、「顧客ID」は重複することになります。

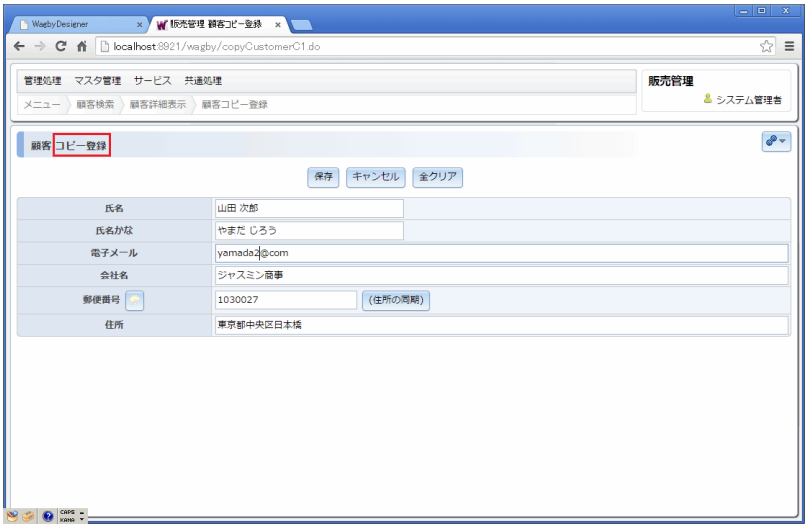


図 コピー登録

枝番が増加します。

図 コピー登録直後の詳細画面

定義方法

顧客IDは順序を用いますが、同時に「コピー登録時は順序を使わない」設定を有効にしています。

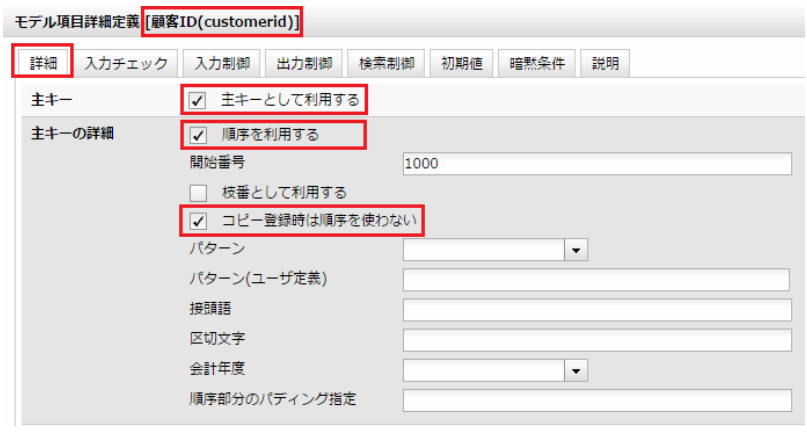


図 顧客IDの定義（順序を用いる）

枝番の定義は図と同様です。枝番指定は「主キーとして利用する」を有効とします。

図 枝番の定義

枝番をアルファベット表記とする

例

枝番を 1,2,3,... という整数値ではなく、A,B,C,... というアルファベット表記にする例を示します。 図は主キー "A-0001" のデータを登録した直後です。ここでは枝番は含まれていません。

図 主キーA-0001のデータを登録した

コピー登録機能を用いて、同じ主キーでデータを再登録してみます。

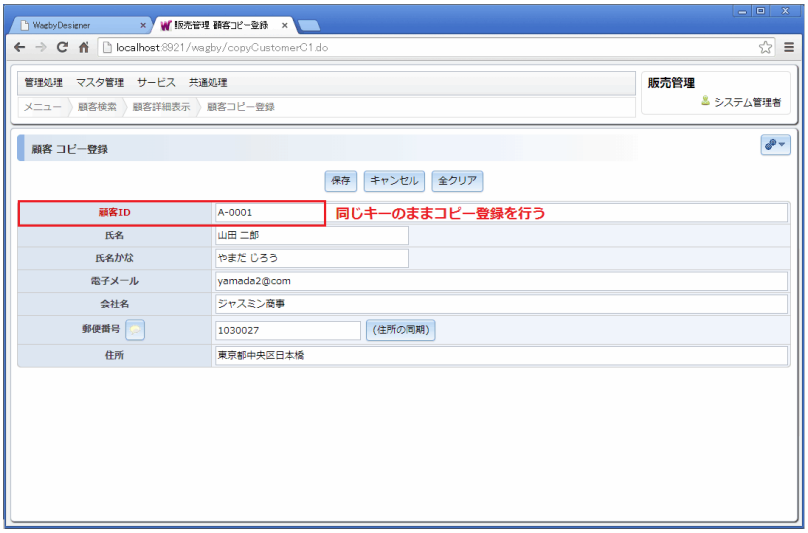


図 同じ主キーA-0001でデータを再登録する

キーの末尾に ″A″ が付与されました。これが枝番です。

図 枝番Aが付与された

再び同じ主キー ″A-0001″ で登録すると、今度は枝番 ″B″ が付与されます。 以降、枝番は A,B,C,...Z と付与され、Z の次は AA となります。

図 枝番Bが付与された

定義方法

3つの項目「顧客ID」「枝番（隠し）」「顧客ID」を用意します。最初の「顧客ID」は登録・更新画面で表示されます。もう一つの「顧客ID」は表示用で、計算式によって枝番を末尾に結合した形で表示させるようにします。

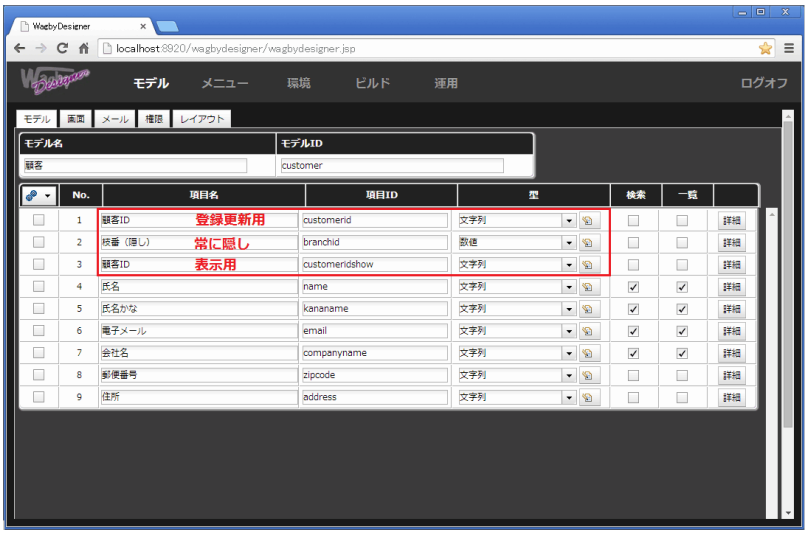


図 モデル定義

各項目の詳細な設定は次のとおりです。

項目名	項目ID	型	主キー	枝番	順序	開始番号	DB保存	入力時隠し	出力時隠し
顧客ID	customel	文字列	○	－	－	－	○	－	○
枝番（隠し）	branchi	数値	○	○	－	1	○	○	○
顧客ID	customel	文字列	－	－	－	－	－	○	－

表示時に用いる「顧客ID」（customeridshow）項目には、次の自動計算式を適用します。

```
CONCATENATE(${customerid},A1FORMAT(${branchid}-1))
```

関数A1FORMATは、Excelの列の命名規則に準拠したアルファベットを返します。

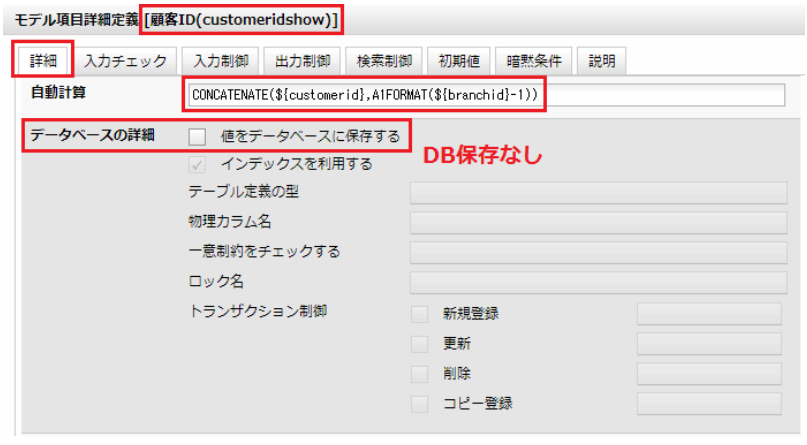


図 表示時の式の設定

複合キーによる制約を回避したい場合

Wagbyは一部の機能で複合キーモデルを利用できないことがあります。　この制約を回避するために「単一キーで絞込機能を応用した枝番」を設定することもできます。

単一キーで絞込機能を応用した枝番

この説明を読む前に

「複合キーによる枝番」を先に読んでおいてください。「枝番」を複合キーの一部として設定する方式は、このページに記載している内容よりも設定が容易です。

ここでは複合キーではなく、単一キーで枝番機能を設定する方法を説明します。「他モデルの参照 > 他項目の値で選択肢を絞り込む」機能を応用します。

仕様と例

仕様

ここで説明する設定は次の仕様に基づいています。

- 顧客モデルの「顧客ID」は文字列型で、かつ順序を利用します。
- 通常の新規登録処理では、顧客IDは "1000", "1001", "1002",... と増加します。
- コピー登録機能では順序を利用しない設定とします。このためコピー登録ではキーが重複します。そこで枝番を割り当てます。
- 枝番は "1000-1" と、末尾にハイフンと数字を付与して表現します。
- 同じデータを何度もコピー登録すると、その都度 "1000-2", "1000-3",... と枝番が増加します。
- 枝番は主キーの一部には含めません。

動作

顧客モデルの新規登録を行います。最初の顧客IDは "1000" となっています。これは順序を使って割り当てています。

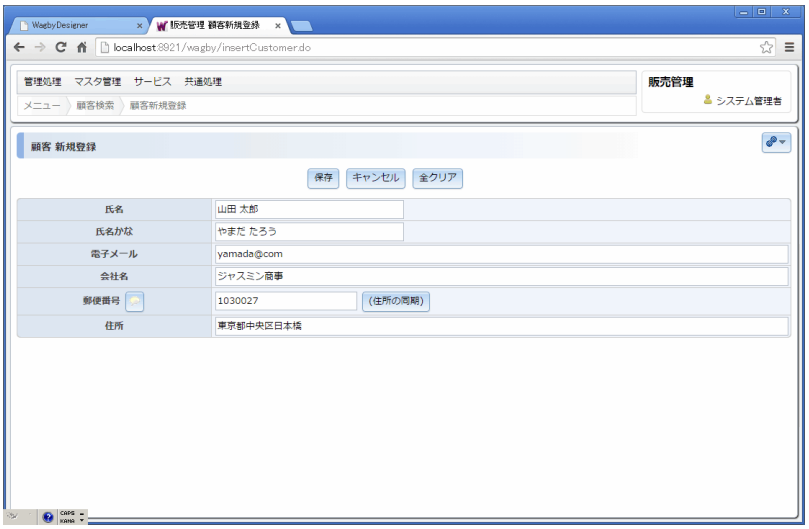


図 顧客の登録

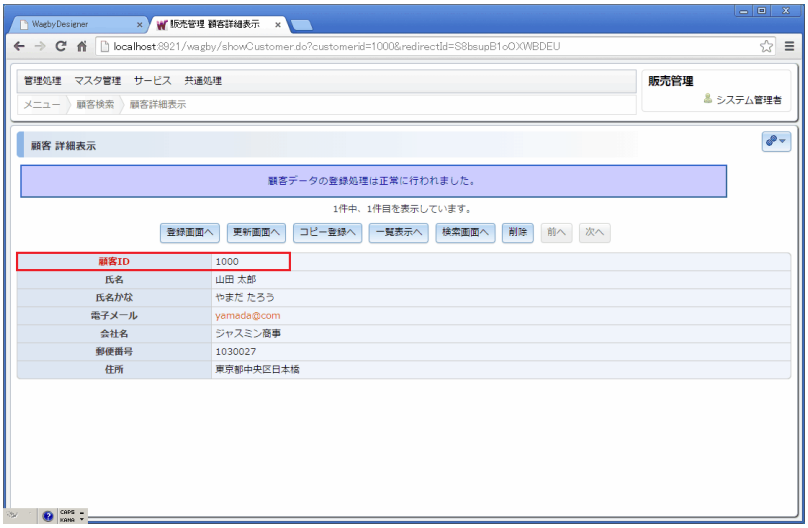


図 顧客IDに枝番は付与されていない

コピー登録機能を使います。

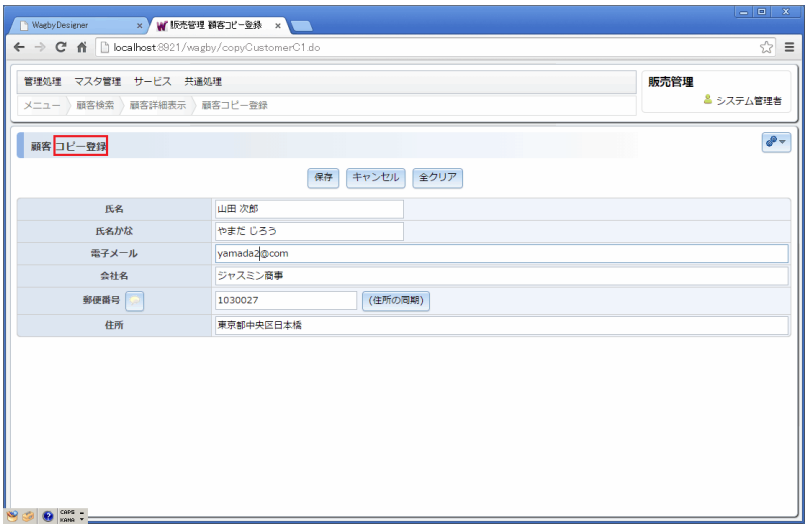


図 コピー登録

枝番が付与されます。以降、顧客IDが ″1000″ のデータに対してコピー登録を行うと、枝番が増加していきます。

図 顧客IDに枝番が付与された

図 さらにコピー登録を行った（枝番が増加）

定義方法

主キーと枝番に関係する項目は4つあります。

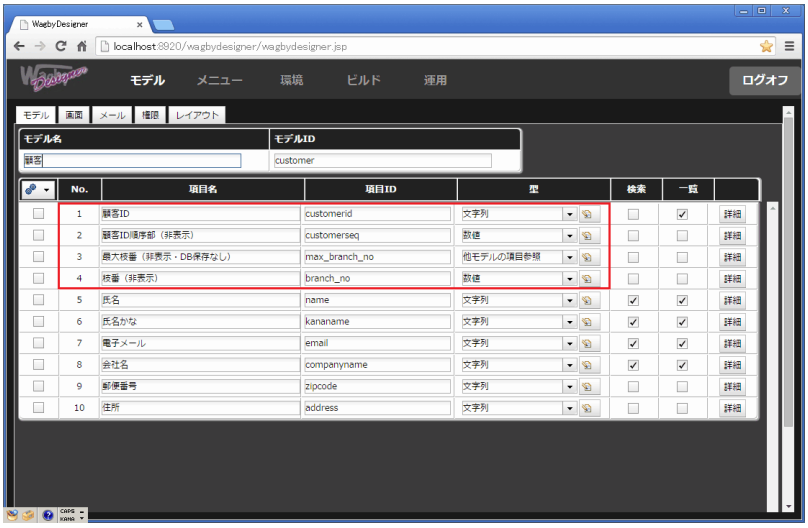


図 顧客モデルの定義

項目名	項目ID	型	主キー	順序	DB保存	入力時隠し	出力時隠し	登録時の計算	その他
顧客ID	customer	文字列	○	○	○	－	－	[後述]	コピー登録時は順序を使わない
顧客ID順序部	customer	数値	－	－	○	○	○	[後述]	－

最大枝番	max_bra	モデル参照（自分自身のbranch_	－	－	－	○	○	－	絞込[後述]
枝番	branch_	数値	－	－	○	○	○	[後述]	－

顧客ID

顧客IDは主キーとします。この例では文字列とし、かつ順序を用います。開始番号は1000から、としています。

主キーであるため自動的に必須指定となります。また、入力時は隠し項目になります。

図 顧客IDの定義（文字列型かつ順序を用いる）

登録時の式を次のようにします。コピー登録の場合は枝番を付与し、新規登録の場合は標準の順序値を利用するものです。

```
IF(EXACT(SCREENTYPE(),"copy"),
  CONCATENATE(
    TOSTR(${customerseq}),
    IF(${max_branch_no}>0,
      CONCATENATE("-",TOSTR(${max_branch_no})),
      ""
    )
  ),
  ${customerid}
)
```

customerseq 項目の値をベースとし、max_branch_no の値が 0 より大きい場合、"-" の後に max_branch_no の値を付与します。これが枝番部分になります。

CONCATENATE は引数の文字列を結合した文字列を返します。 TOSTR は数字を文字列に変換します。

図 主キーの式

顧客ID順序部

顧客ID順序部は、顧客IDの順序部分だけを切り出して保存する項目として用意します。DB保存とし、入力時・出力時とも隠し項目とします。

登録時の式を次のようにします。コピー登録の場合は現在、保持している順序をそのまま使い、新規登録の場合は順序部分（数値部）を切り出します。

```
IF(EXACT(SCREENTYPE(),"copy"),
  ${customerseq},
  TOINT(
    LEFT(${customerid},
      IF(FIND("-", ${customerid})>0,
        FIND("-", ${customerid})-1,
        LEN(${customerid})
      )
    )
  )
)
```

顧客ID値を数字に変換したものを保持します。このとき、枝番部（"-" で始まる）があれば、それを取り除きます。

TOINT は引数の文字列を数値に変換します。 LEFT は引数の文字列の左から何番目までを取得します。FIND は指定された文字が何番目に含まれているかを返します。 LEN は文字列の長さを返します。

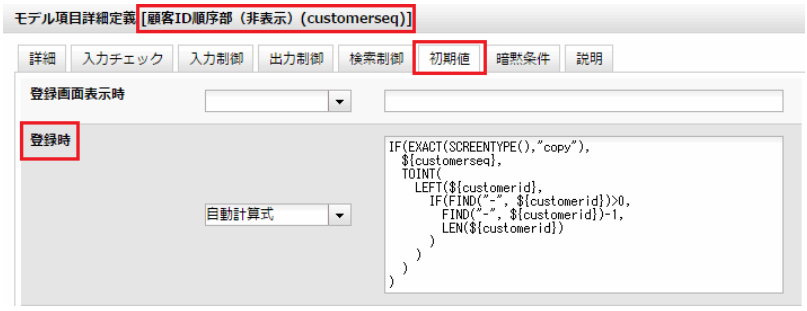


図 顧客ID順序部の式

最大枝番

特定の「顧客ID順序部」をもったデータ群について、最大枝番の値を保持するように設定します。

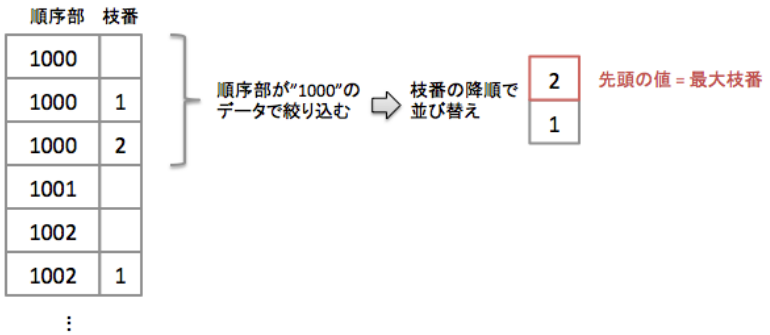


図 絞込のイメージ

自モデル（自己）参照とし、枝番の値をリストボックスで表示するとしますが、さらに「順序部で絞込み」を行い、「枝番の降順で並び替え」し、「先頭の項目のみを出力する」とすることで、現在の枝番の最大値を取得するようにします。

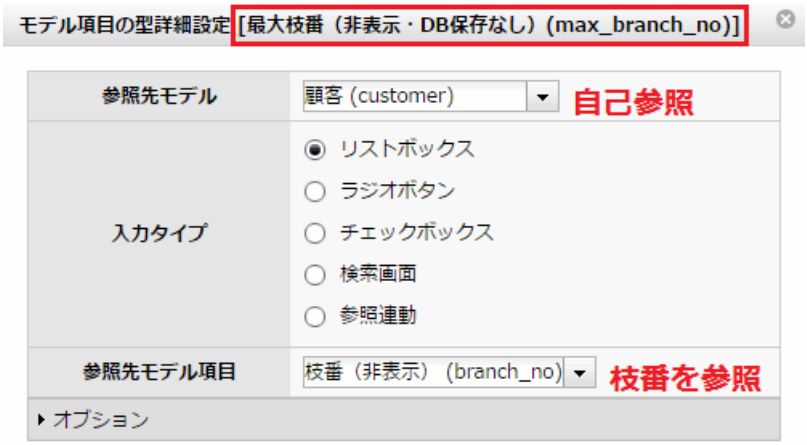


図 モデル参照の設定

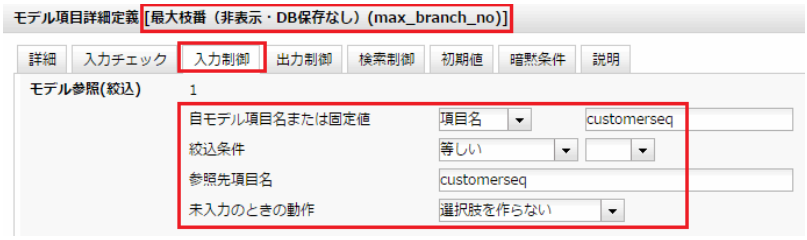


図 絞込の設定

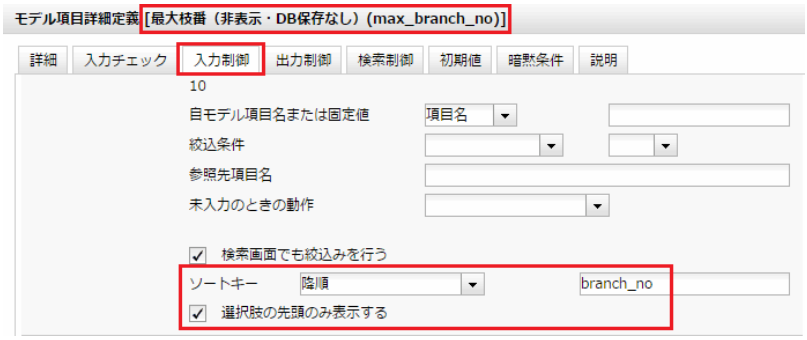


図 枝番の降順で並び替え

枝番

現在の枝番の値を保持します。

登録時の式を次のようにします。

```
IF(ISBLANK(${max_branch_no}),
  1,
  ${max_branch_no#content}+1
)
```

最大枝番が空白の場合は 1 を、すでに存在する場合は +1 した値を保存するようにします。

ISBLANK は引数が空白かどうかを判定します。

図 枝番の式

最初の設定では、システムが内部で利用する上記項目を「隠し」ではなく「読み取り専用」とするとよいでしょう。テストによって枝番の動きが把握できます。動作確認できたところで「隠し」としてください。

枝番をゼロ詰め3桁で表現する

画面例

枝番部をゼロ詰め3桁表記とした例を図に示します。

図 枝番部がゼロ詰め3桁表記となっている

顧客IDの設定

顧客IDの登録時の式を次のようにTEXT関数を使うように変更します。

```
IF(EXACT(SCREENTYPE(),"copy"),
  CONCATENATE(
    TOSTR(${customerseq}),
    IF(${max_branch_no}>0,
      CONCATENATE("-",TEXT(${max_branch_no}, "000")),
      ""
    )
  ),
  ${customerid}
)
```

TEXT は書式付きの変換を行います。

枝番をアルファベット表記かつハイフン区切りなしとする

画面例

枝番部をアルファベット表記（"A", "B", "C",...）かつハイフンなしとした例を図に示します。

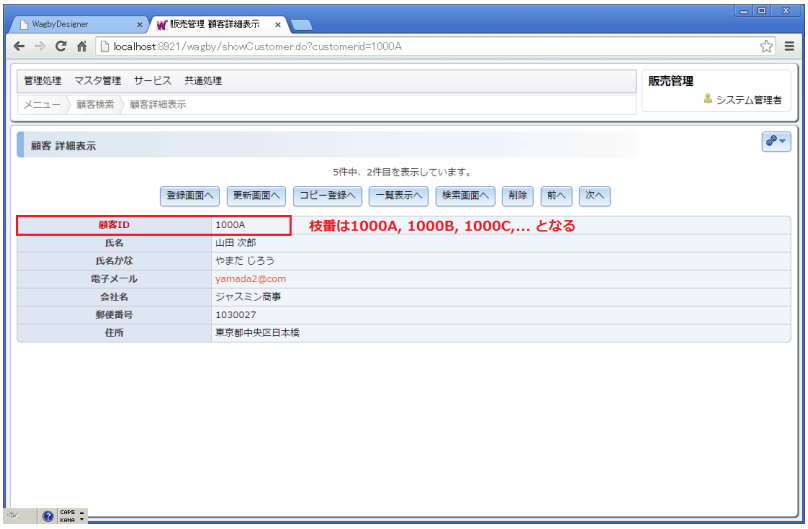


図 枝番部がアルファベット表記かつハイフンなしとなっている

顧客IDの設定

顧客IDの登録時の式を次のようにA1FORMAT関数を使うように変更します。

```
IF(EXACT(SCREENTYPE(),"copy"),
  CONCATENATE(
    TOSTR(${customerseq}),
    IF(${max_branch_no}>0,
      A1FORMAT(${max_branch_no}),
      ""
    )
  ),
  ${customerid}
)
```

A1FORMAT はExcelの列名ルールに準拠して数値をアルファベットに置き換えます。

顧客ID順序部の設定

顧客ID順序部の登録時の式を次のように正規表現で判定するように変更します。

```
IF(EXACT(SCREENTYPE(),"copy"),
  ${customerseq},
  TOINT(
    LEFT(${customerid},
      IF(FINDRE("[A-Z]", ${customerid})>0,
        FINDRE("[A-Z]", ${customerid})-1,
        LEN(${customerid})
      )
    )
  )
)
```

FINDRE は正規表現にマッチした文字が含まれていた場合、その位置を返します。

検索制御

文字列のAND/OR検索

例

一つの単語で検索する

検索入力欄に一つの単語を入力した場合は、その単語を含む部分一致検索になります。

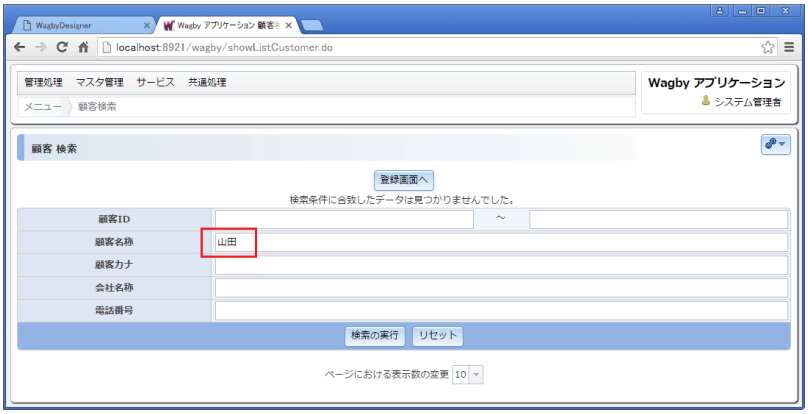


図 山田を含むデータを検索する（標準は部分一致検索）

複数の単語をすべて含むデータを検索する（AND 検索）

複数の単語による AND 検索を行うことができます。検索入力欄に半角スペース区切りで、複数の単語を入力します。

図 山田かつ太郎を含むデータを検索する

複数の単語のいずれか含むデータを検索する（OR 検索）

複数の単語による OR 検索を行うことができます。はじめに半角大文字で "OR" を指定し、そのあとに半角スペース区切りで、複数の単語を入力します。

図 山田または鈴木を含むデータを検索する

定義方法

「検索制御 > 拡張検索機能を有効にする」を有効にします。標準では、本設定は有効となっています。

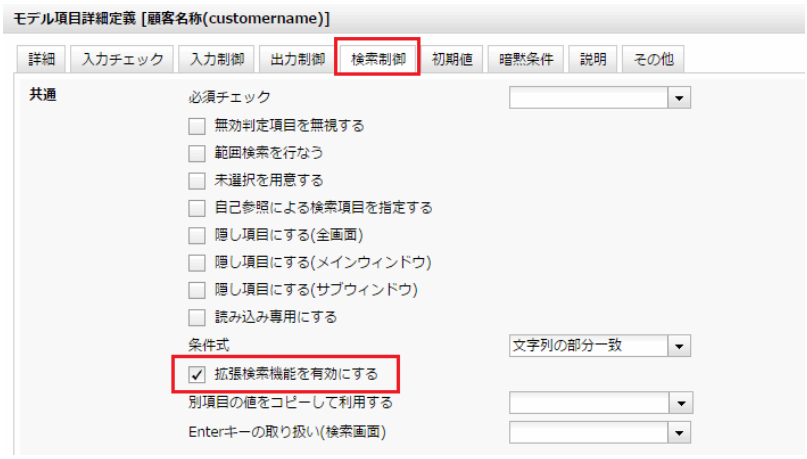


図 拡張検索機能の設定

仕様・制約

- 検索文字列が「OR」で開始する場合のみ OR 検索を行い、その他の場合は通常の完全/前方/後方一致検索を行います。これは完全一致、前方一致、後方一致は AND 検索では意味をもたないためです。
- 他モデル参照(参照連動は除く)では拡張検索は無視します。
- 「等しくない」(notEqual) 検索では、OR 検索を行いません。検索文字列に空白が含まれていた場合、分割して AND 検索を行います。これは「等しくない」検索では OR は意味をもたないためです。
- 「以上」「以下」「より大きい」「より小さい」検索は AND・OR いずれも意味をもたないため、拡張検索は行いません。

標準を OR 検索にする

フィルタスクリプトを利用すると、検索文字列に「OR」を付与する仕掛けを設定できます。

文字列の一致ルールを指定する（ワイルドカードの利用）

定義方法

「検索制御 ＞ 条件式」で判定ルールを選択します。文字列型の場合、次のルールを適用できます。

- 等しい（完全一致）
- 文字列の部分一致
- 文字列の前方一致
- 文字列の後方一致
- 文字列の部分一致（実行時指定）

標準は〝文字列の部分一致〟となっています。

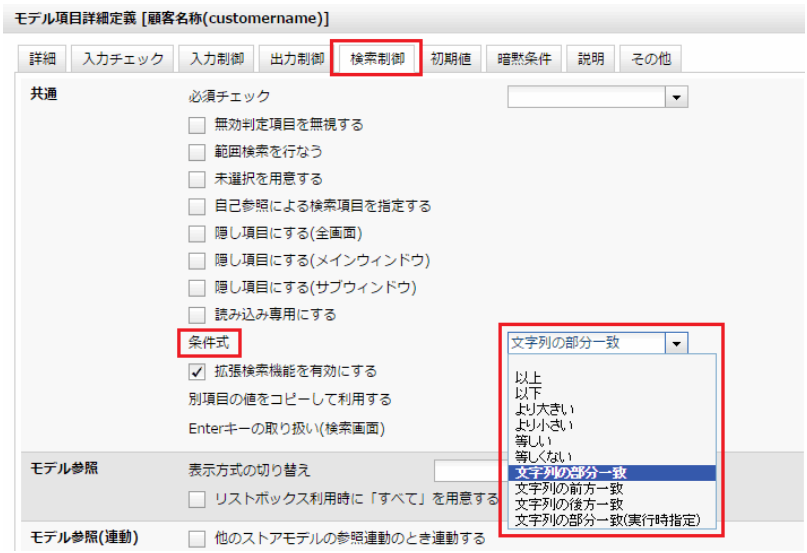


図 検索制御>条件式の設定

ワイルドカード

条件式に「文字列の部分一致（実行時指定）」を指定すると、検索語にワイルドカードを指定することができます。

書式	説明	例
?	任意の文字	「あい?」と検索すると「あいう」がヒットします。
*	任意の長さの任意の文字	「あい*」と検索すると「あいうえお」がヒットします。

検索条件を必須入力にする

例

図は氏名項目を必須とした例です。値を入力しなければ検索を行うことができません。

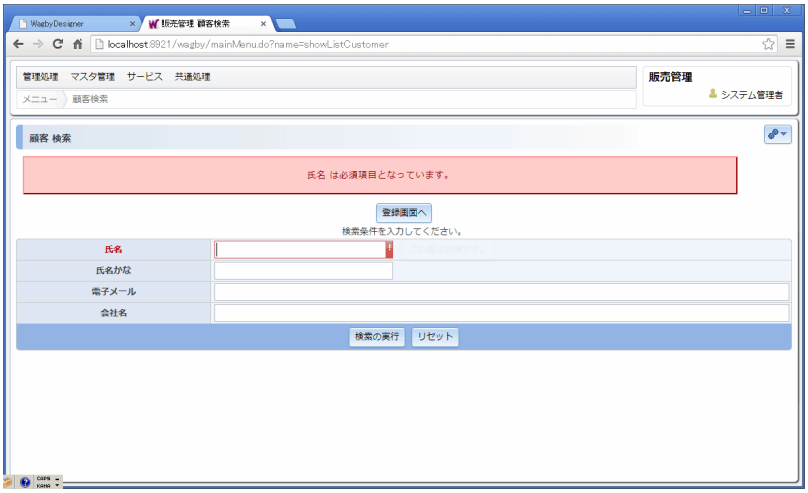


図 氏名項目を必須とした

定義方法

モデル項目詳細定義の「検索制御>共通」内にある「必須チェック」を有効にします。

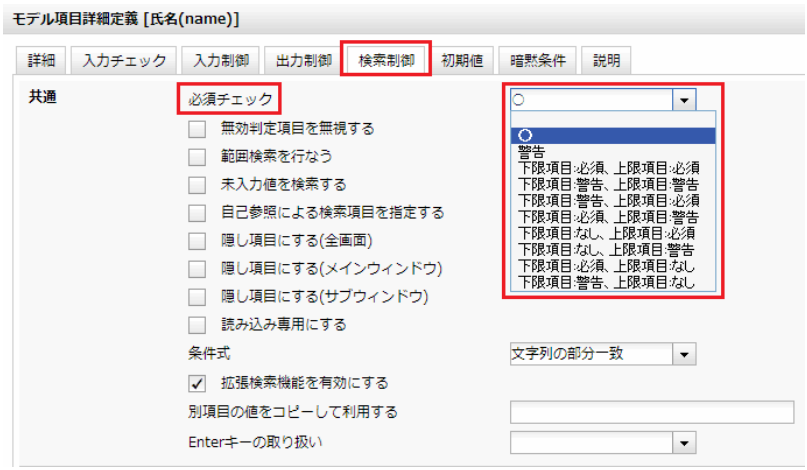


図 必須チェックに「○」を設定する

「○」ではなく「警告」と設定することもできます。「警告」と設定した場合にも検索画面にメッセージは表示されますが、再度「検索の実行」ボタンをクリックすると、検索処理は行われます。

上限と下限の必須チェック

数字型や日付型の項目で範囲検索を行う場合には「下限項目：警告、上限項目：必須」などのパターンで設定することもできます。（「範囲検索を行う」が有効になっている必要があります。）

検索画面遷移直後の検索処理を止める

Wagbyは標準で、検索画面に遷移したタイミングで、検索処理を行います。しかし本設定を有効にすると、必ずエラーとなってしまいます。これを避けるため、本設定を行います。

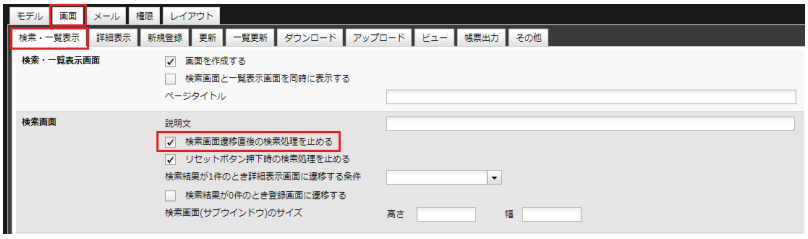


図 検索画面遷移直後の検索処理を止める設定を有効にする

代案 ： 初期値を設定する

検索画面遷移直後の検索処理は行わせますが、当該項目の検索条件の初期値を設定する方法もあります。いずれかの方法で、エラーメッセージを回避できます。

仕様・制約

- 型が「モデル参照（リストボックス）」の項目に本設定を適用すると、選択肢 ″すべて″ が用意されません。
- 型が「モデル参照（ラジオボタン）」の項目に本設定を適用すると、クリアボタンが用意されません。
- カレンダビューの日付の項目は、この指定を行うことができません。
- 範囲検索が指定されていないにも関わらず、検索制御の必須チェックが範囲検索（上限、下限）を設定することはできません。
- 範囲検索を指定した項目では、必須チェックは「〇」ではなく上限、下限といった選択肢から選択してください。検索条件入力部を一つとしても範囲検索を指定した場合も同様に、必須チェックは「〇」ではありません。
- 本設定を行ったときは、同時に「検索時の初期値を設定する」か、「検索画面遷移直後の検索処理を止める」と「リセットボタン押下時の検索処理を止める」を設定して下さい。

無効判定項目を無視する

例

商品モデル

商品モデルに、無効判定用の項目「有効期限」を用意しました。図にあるように、商品名″マンゴー″のみ、無効としています。 (162)

(162) Wagbyでは、無効判定用の項目に何らかの値が設定されていた場合に、本データは無効とみなすという仕組みが備わっています。 詳細は「他モデルの参照>選択肢を無効にする」をお読み下さい。

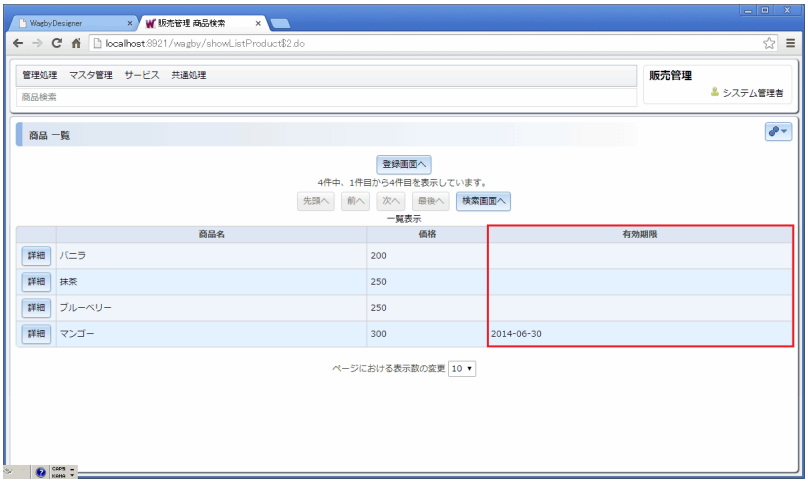


図 商品データー覧

見積書モデル

見積書モデルは、上記の商品モデルを参照しています。

図に示すように、登録・更新画面では無効となったデータは選択肢として表示されません。

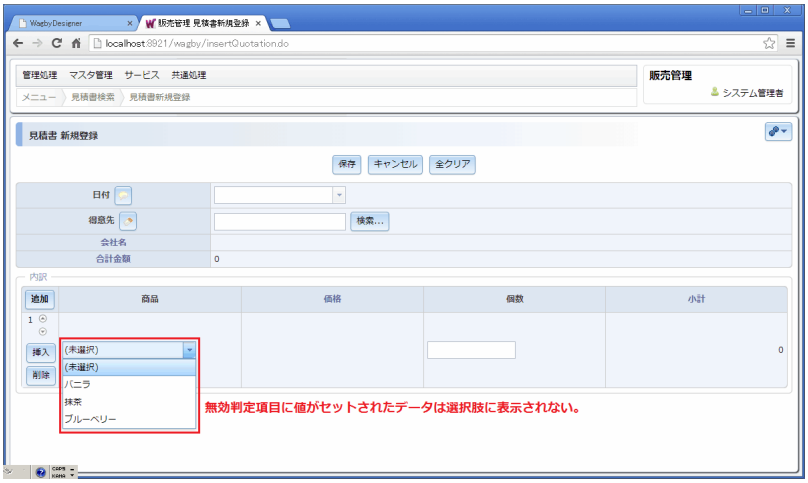


図 無効データは選択肢に表示されない

しかし検索画面では無効データが表示されています。これによって、無効データについての検索が可能となっています。

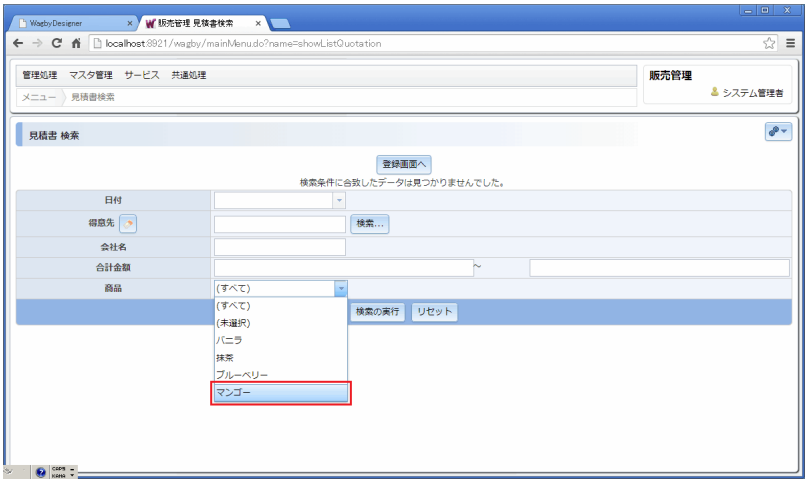


図 検索画面では無効データも検索できる

定義方法

商品モデル

「有効期限」項目を（文字列型として）用意します。

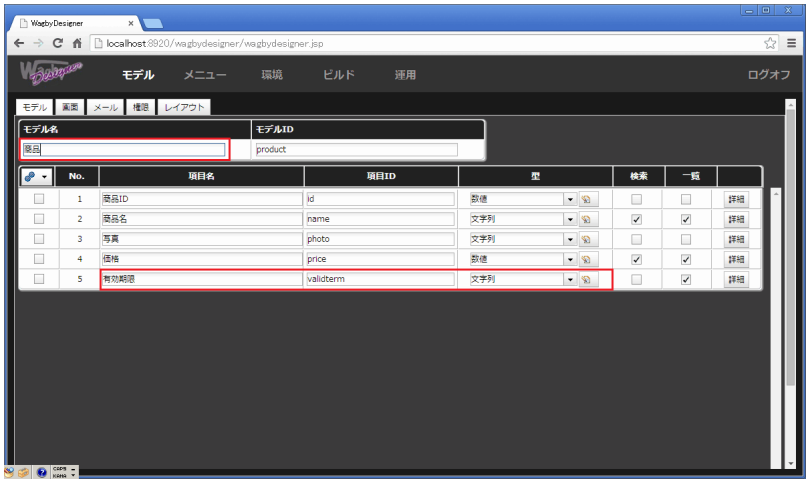


図 商品モデルの定義

見積書モデル

商品モデルを参照する項目では、無効判定項目として図で用意した「有効期限」項目を指定しています。

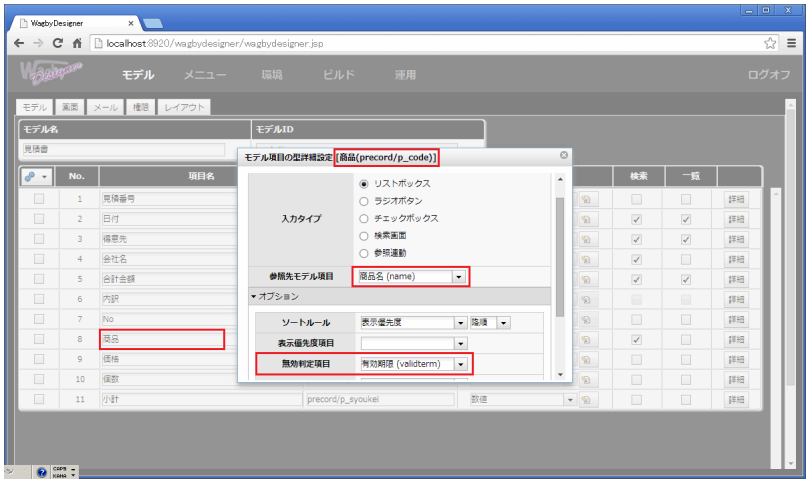


図 見積書モデルにおける商品モデルの参照定義

モデル項目詳細定義の検索制御タブ内にある「共通>無効判定項目を無視する」をチェックすると、無効判定項目も検索対象となります。（標準ではチェックされていません。）

図 無効判定項目を無視する設定

範囲検索を行う

例

図は数値型項目の範囲検索が有効になった画面例です。

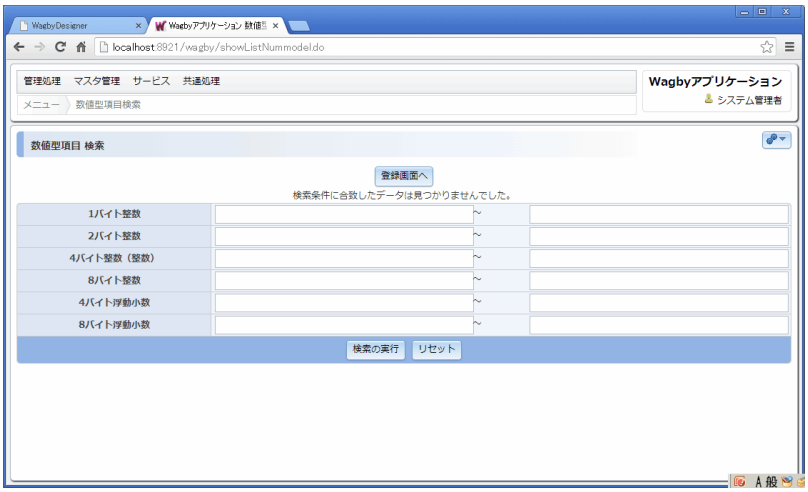


図 検索条件画面

範囲検索を無効にすると図のようになります。この場合は指定した値に合致するかどうか、という検索になります。

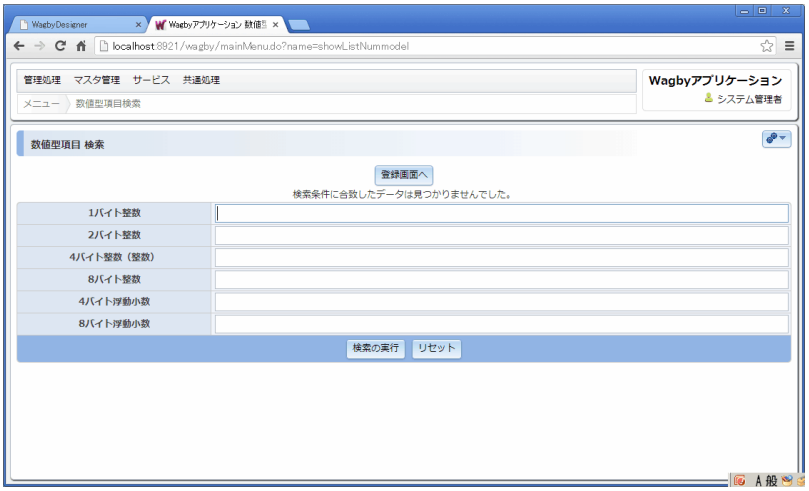


図 範囲検索を無効にした検索画面

定義方法

モデル項目詳細定義の検索制御タブ内にある「共通>範囲検索を行う」のチェックを解除すると、範囲検索ではなくピンポイントの検索になります。

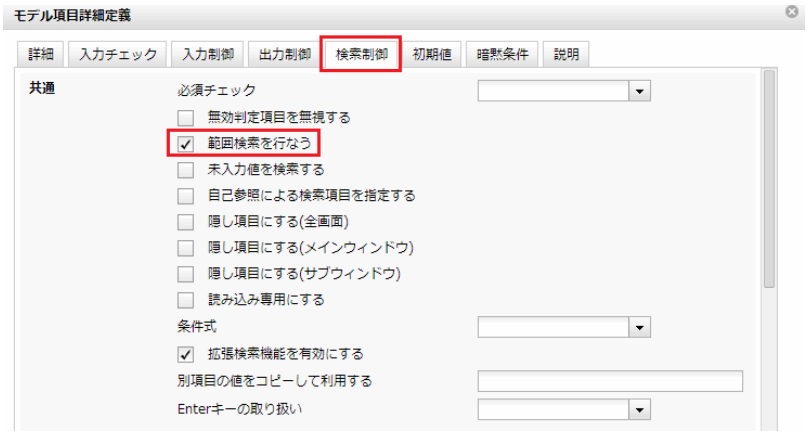


図 範囲検索を無効にする

日付型項目の開始および終了の初期値の設定

検索範囲の初期値に「月初」「月末」を指定することができます。

図 初期値に月初から月末までと指定する

定義方法

開始部分を「初期値 ＞ 検索画面表示時」に、終了部分を「初期値 ＞ 検索画面表示時（範囲検索の上限項目）」に指定します。それぞれ、特別表記 now を使います。

```
now:firstday

now:lastday
```

図 初期値の設定

月日の自動補完

日付型項目で、年（または年度）を指定すると、自動的に月（および月・日）を補完させることができます。

例

月日の自動補完機能は、検索条件の日付型項目をリストボックスとして利用する場合に使うことができます。 「年」または「年度」によって補完されるパターンが変わります。

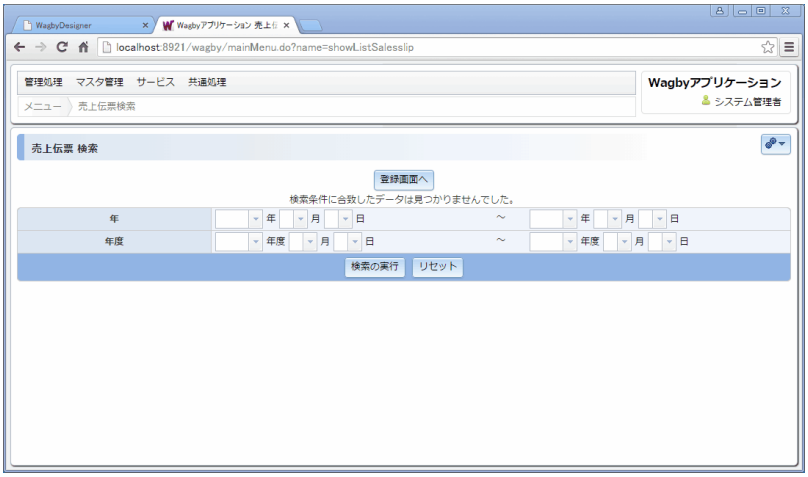


図 検索条件の日付項目（リストボックス形式）

「年」の場合を説明します。開始部分では、年を指定すると自動的に〝1月1日〟が補完されます。

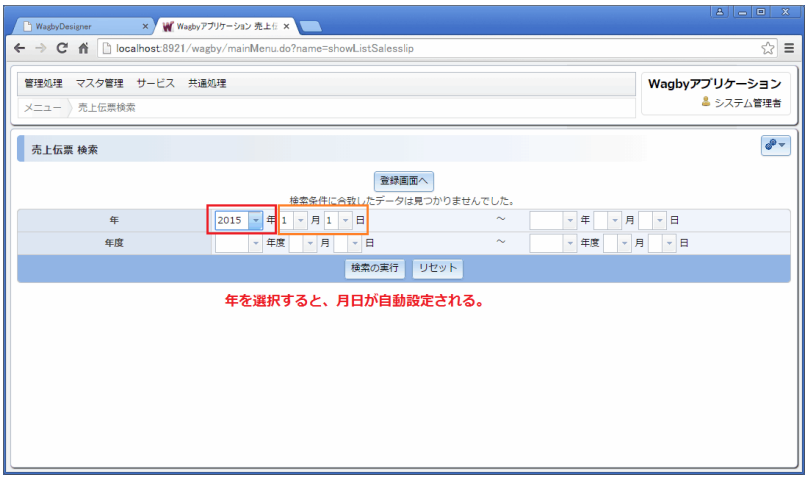


図 開始年月日の設定（月日は自動補完）

同じく終了部分では、自動的に〝12月31日〟が補完されます。

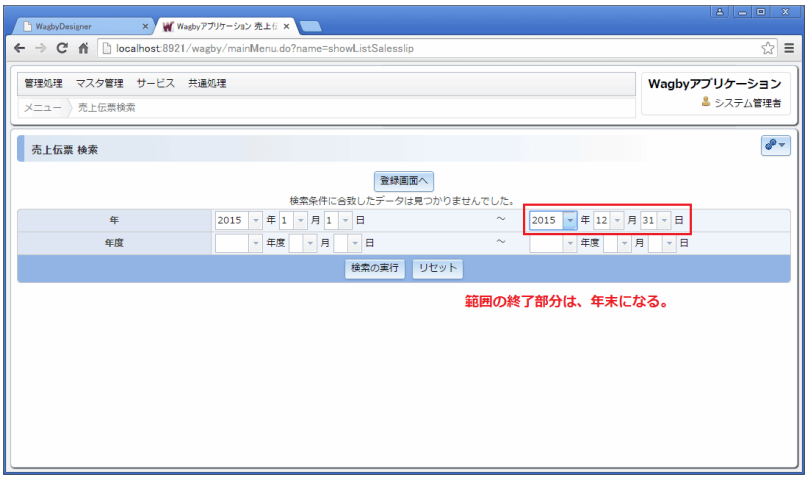


図 終了年月日の設定（月日は自動補完）

「年度」の場合、開始が〝4月1日〟となり、終了が〝3月31日〟となります。年度の開始月は4月固定ではなく、自由に設定できます。

図 年度の場合

定義方法

年の場合

「検索制御 > 日付・時刻」で、「日付補助入力」を「リストボックスの利用」とします。上の例では〝年月日〟を使いましたが、〝年月〟とすることもできます。（日の部分は入力できませんが、自動的に月初と月末扱いになります。）

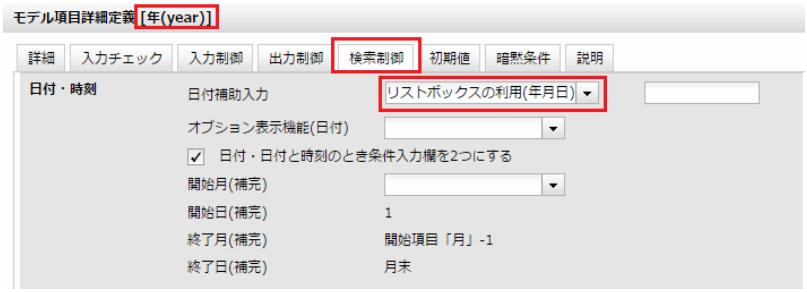


図 年の場合の設定

年度の場合

開始項目「月」（補完）を指定すると、「年」が「年度」扱いとなります。例では 〝4〟 としていますので、年度切り替えは4月となります。

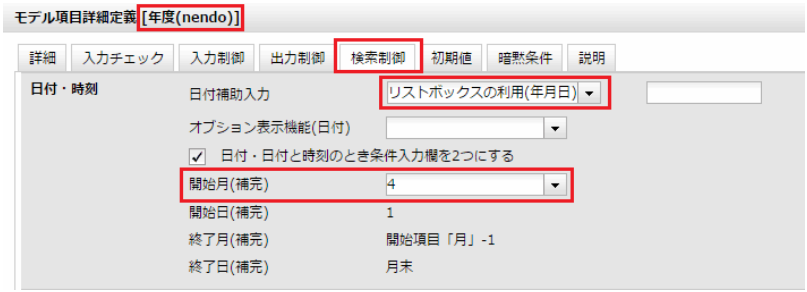


図 年度の場合の設定

「開始日」「終了月」「終了日」は自動的にセットされます。これらを指定することはできません。

日付型項目の開始のみを設定する

開始項目だけを指定して範囲検索を行うこともできます。

例えば ″201411″ と指定すると、″2014年11月1日～11月30日″ で検索を行います。

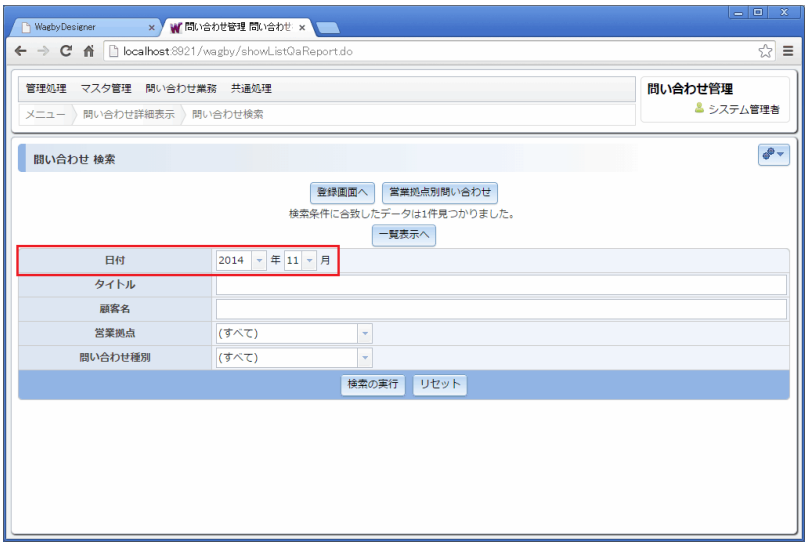


図 開始項目だけを指定した範囲検索

定義方法

日付補助入力に「リストボックスの利用（年／年月／年月日）」を設定します。（「カレンダの利用」設定時には機能しません。）

あわせて「日付・日付と時刻型のとき条件入力欄を2つにする」の設定を解除します。（標準では有効になっています。）

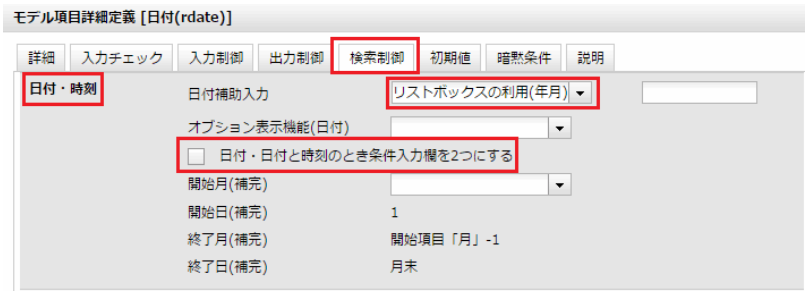


図 開始項目だけを指定した範囲検索

リストボックスのパターンにより検索範囲は次のようになります。

日付補助入力のパターン	入力値	検索範囲
年	年のみ指定	指定年の1月～12月
年月	年月指定	指定した月の1日～月末日※ 月は必ず補完されます。空白にすることはできません。
年月日	年月日指定	指定した日付※ 月、日は必ず補完されます。空白にすることはできません。

「日付と時間」型の場合、時間まで指定することで一致するデータを検索することができます。

テキストフィールドを用いた年、年度、年月日の検索

検索画面で、テキストフィールドを用いた日付入力を行うことができます。4つのパターンがあります。

年のみを指定する

年の入力フィールドは最大入力文字数 4 で用意されます。

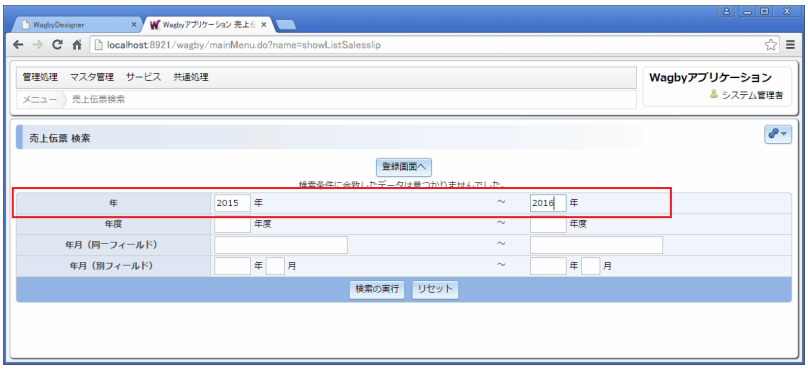


図 年のみを指定

設定は「検索制御 ＞ 日付・時刻 ＞ 日付補助入力機能」で「リストボックスの利用(年)」を選択し、入力部分に ″text″ と指定します。

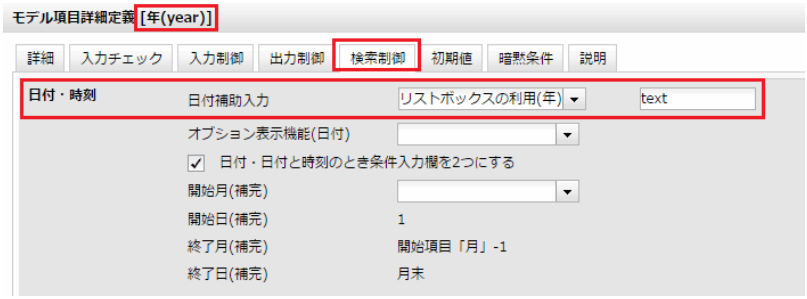


図 年のみを指定する設定

年のみを指定する

年度の入力フィールドも最大入力文字数 4 で用意されます。

図 年度のみを指定

設定は「検索制御 ＞ 日付・時刻 ＞ 日付補助入力機能」で「リストボックスの利用(年)」を選択し、入力部分に ″text″ と指定します。 さらに「開始項目「月」(補完)」に年度の開始月を指定します。

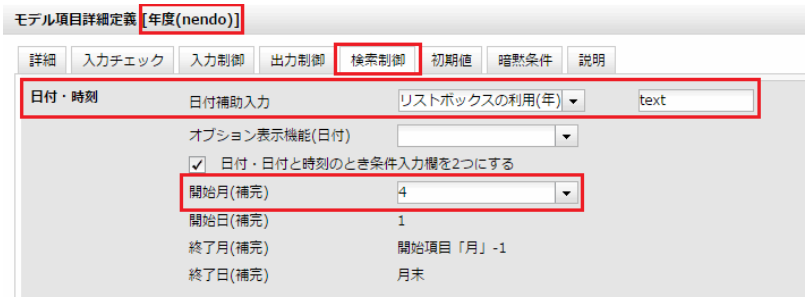


図 年度のみを指定する設定

年月を指定する（同一フィールド）

年月の入力フィールドは最大入力文字数 6 で用意されます。 入力フォーマットは "yyyyMM", "yyyy/MM", "yyyy-MM" のいずれも入力できます。入力後に "yyyy-MM" に変換されます。

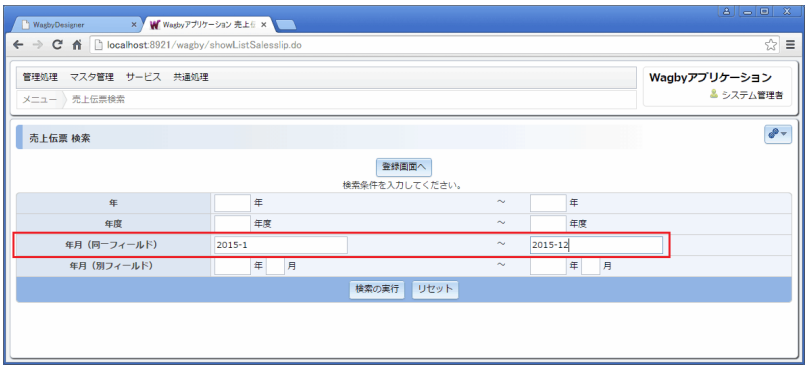


図 年月指定（同一フィールド）

設定は「検索制御 ＞ 日付・時刻 ＞ 日付補助入力機能」で「リストボックスの利用(年月)」を選択し、入力部分に "input" と指定します。

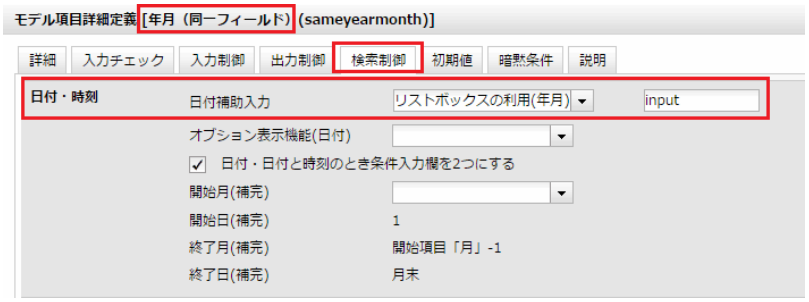


図 年月指定（同一フィールド）の設定

年月を指定する（別フィールド）

年の入力フィールドは最大入力文字数 4 で用意されます。月の入力フィールドは最大入力文字数 2 で用意されます。 年を入力すると月は自動的に開始部に "1" が、終了部には "12" が補完されます。

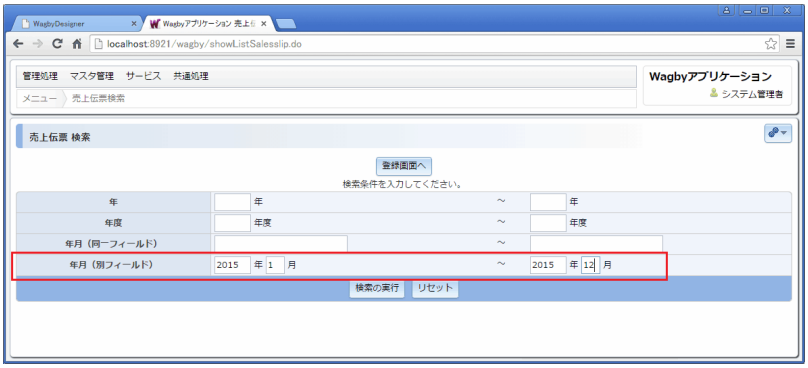


図 年月指定（別フィールド）

設定は「検索制御 ＞ 日付・時刻 ＞ 日付補助入力機能」で「リストボックスの利用(年月)」を選択し、入力部分に ″text_ym″ と指定します。

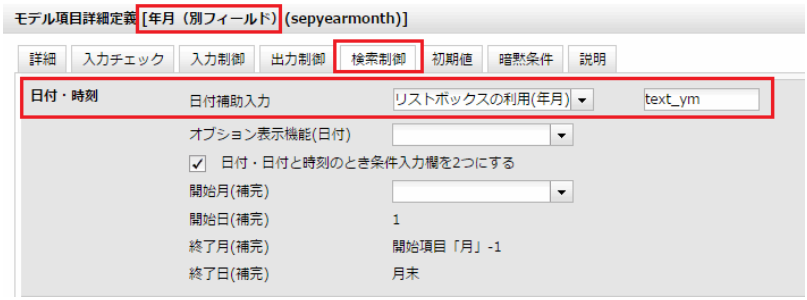


図 年月指定（別フィールド）の設定

入力支援・入力チェック

- ″平成xx年″, ″平成xx″, ″平x″ と入力すると、自動的に西暦に変換されます。″明治″,″大正″,″昭和″が利用できます。
- 年の部分に数値4桁の数値ではないものを入力すると、エラーダイアログが表示されます。

仕様・制約

- 範囲検索は、時刻型では機能しません。（設定を行っても検索画面に入力欄は１つしか用意されません。）
- 本設定を繰り返し項目に適用することはできません。
- 本設定と暗黙条件は同時に指定できません。
- 範囲検索が機能しない場合は「検索制御 ＞ 範囲検索を行う」が有効になっているかどうかを確認してください。
- 「日付・日付と時刻型のとき条件入力欄を2つにする」を設定したときには、範囲検索も有効となっている必要があります。

未入力値の検索

例

文字列型、数字型、日付型の項目について、「未入力」を検索することができます。

文字列型への適用

図は「氏名かな」項目に ″未入力を検索″ オプションを適用した例です。

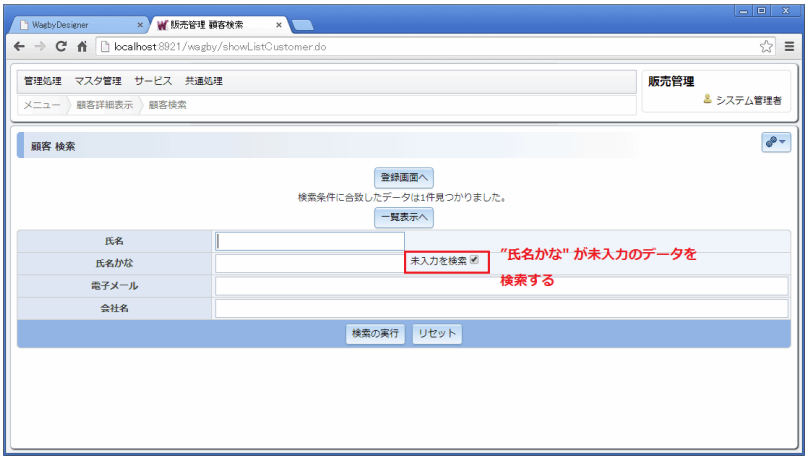


図 文字列型項目に未入力を検索オプションを適用した例

数値型項目への適用

図は「価格」項目に ″未入力を検索″ オプションを適用した例です。範囲検索と併用することができます。

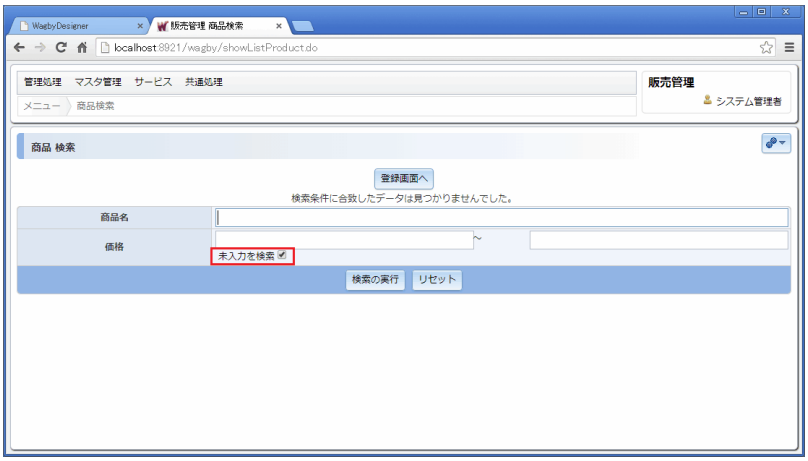


図 数値型項目に未入力を検索オプションを適用した例

定義方法

モデル項目詳細定義の「検索制御>共通」内にある「未入力値を検索する」を有効にします。

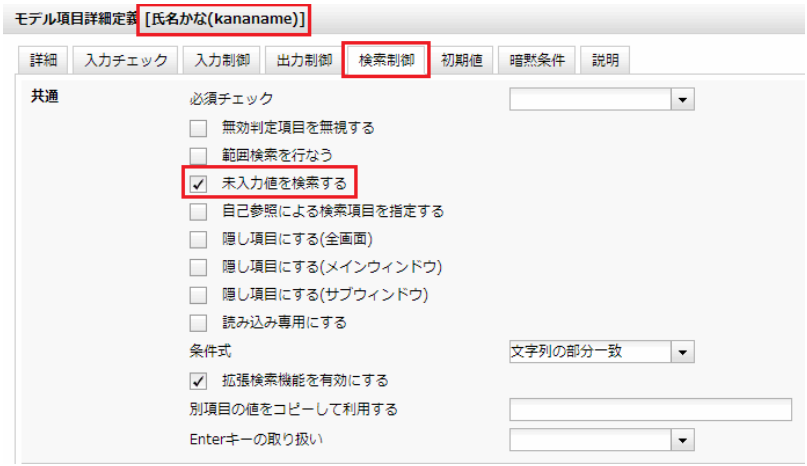


図 未入力値を検索する設定（文字列型）

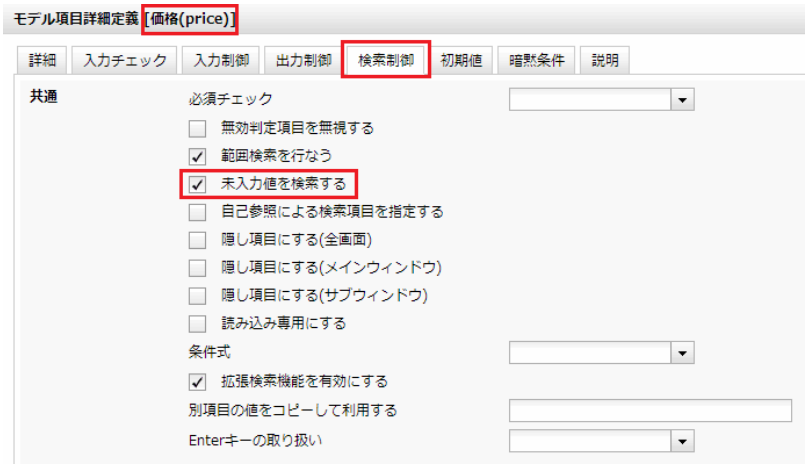


図 未入力値を検索する設定（数値型）

検索値の初期値を未入力値とする

未入力値のデータ(つまりnull)を検索条件の初期値に設定することができます。初期検索でnullのデータを抽出できます。

定義方法

検索条件表示時の初期値に関数 NULLSEARCH() と記述してください。

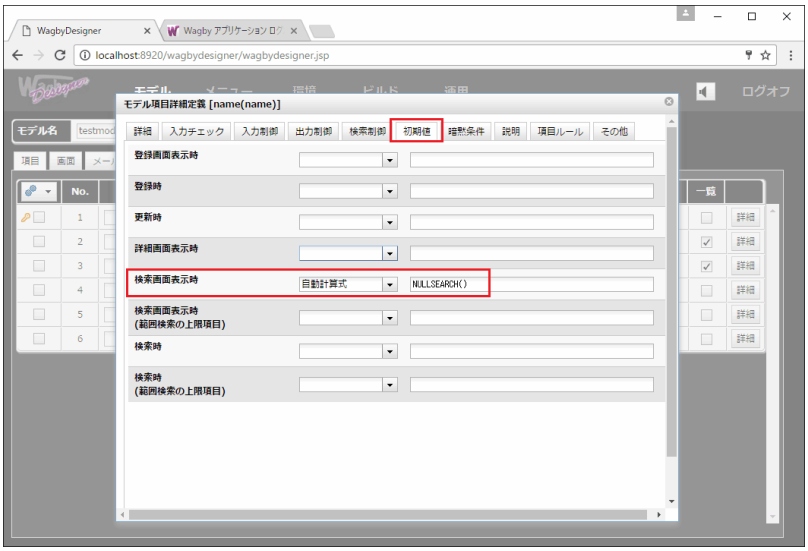


図 NULLSEARCH関数

制約

- この関数は検索条件の初期値でのみ指定できます。外の設定欄（自動計算など）では利用できません。
- 他の関数との組み合わせはできません。（例えば IF 関数との組み合わせは行えません。）単独でご利用ください。
- 数値、日付型の場合、範囲検索との併用はできません。範囲検索を解除してご利用ください。

仕様・制約

本機能は次の場合には利用できません。

- 必須入力項目
- 繰り返し項目
- モデル参照項目(チェックボックス)

モデル参照項目（リストボックス、ラジオボタン）には「未選択項目を用意する」を設定してください。

検索条件を入力した場合

″未入力値を検索する″ を有効にした状態で検索条件に値を指定することは矛盾します。 両方を満たす結果はありえないため、この場合は「データなし（0件)」となります。

自己参照による検索項目を指定する

例

商品モデルの「商品名」（文字列型の項目）に自己参照検索を適用した例を示します。入力したデータが選択できるようになっています。

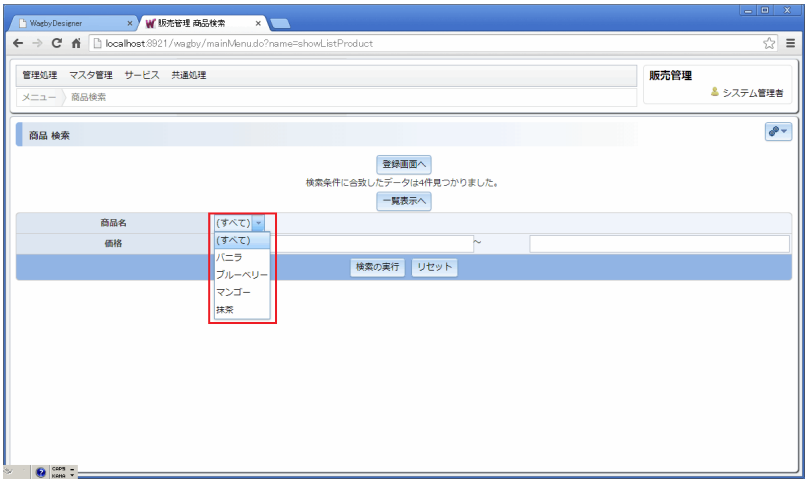


図 商品名項目に自己参照検索を適用した例

定義方法

モデル項目詳細定義の「検索制御>共通」内にある「自己参照による検索項目を指定する」を有効にします。

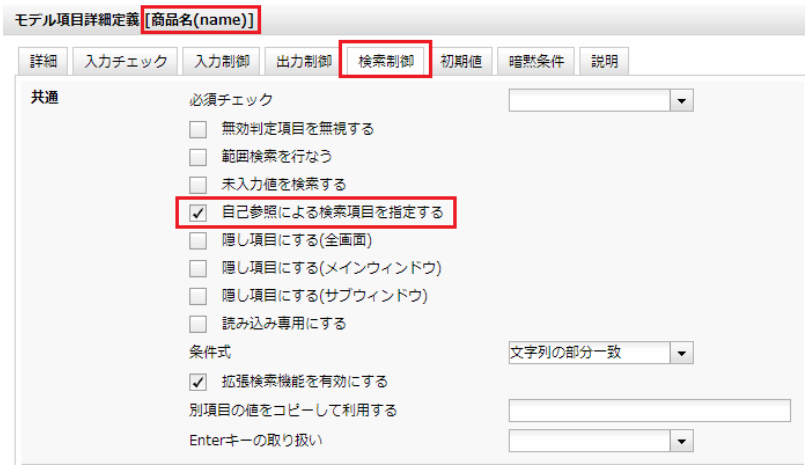


図 自己参照による検索項目を指定する

入力フィールド長の設定

リストボックスの長さを調整するために、「入力フィールド長の設定」を行うことを推奨します。

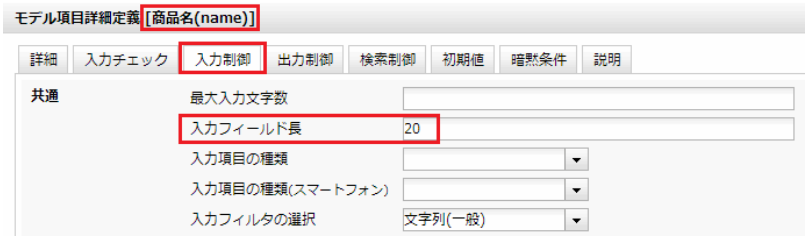


図 入力フィールド長の設定

仕様・制約

- 本設定はデータベースに格納されているすべての該当データを選択できるようにします。そのため、入力パターンが限られるような項目で用いるようにしてください。
- 本設定を繰り返し項目に適用することはできません。

数十を超えるような異なるデータを格納する項目には不向きです。

隠し項目・読み込み専用項目にする

隠し項目にする

営業日報モデルを用意する

図, 図はそれぞれ "yamada" と "suzuki" が営業日報を作成した例です。 データベースには2件のデータが格納されていますが、検索時は自分が作成した日報しか検索することができません。そのため、検索結果は1件となっています。

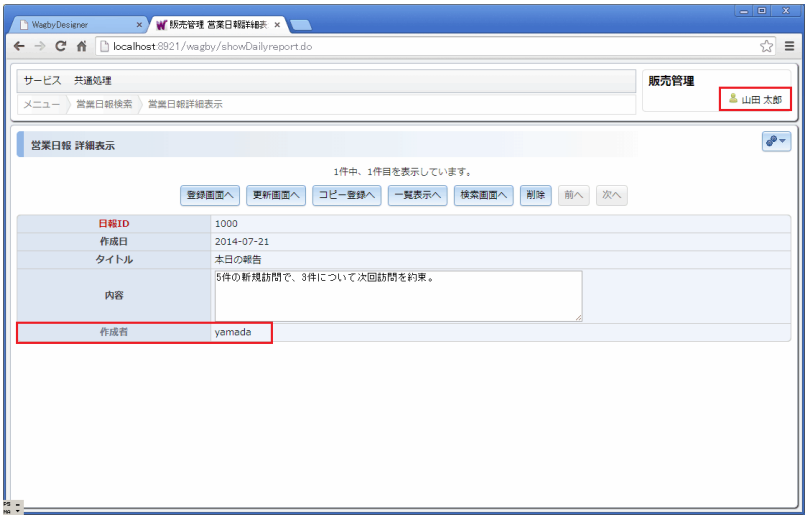


図 アカウントyamadaが作成した営業日報

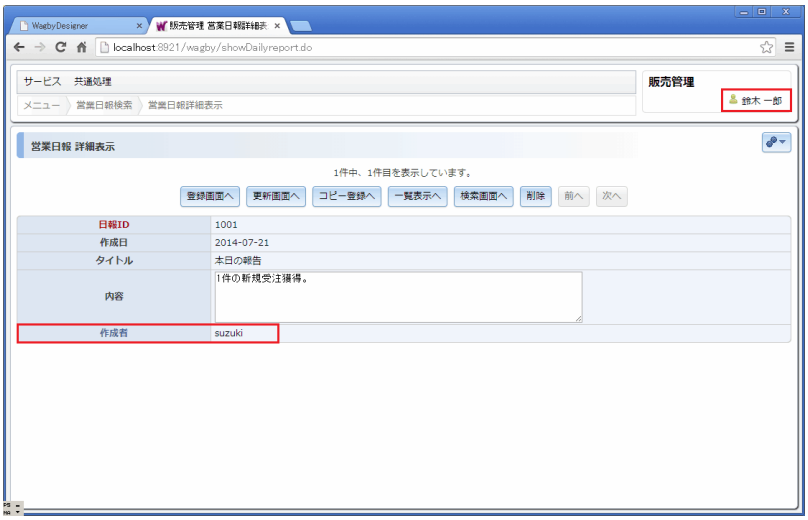


図 アカウントsuzukiが作成した営業日報

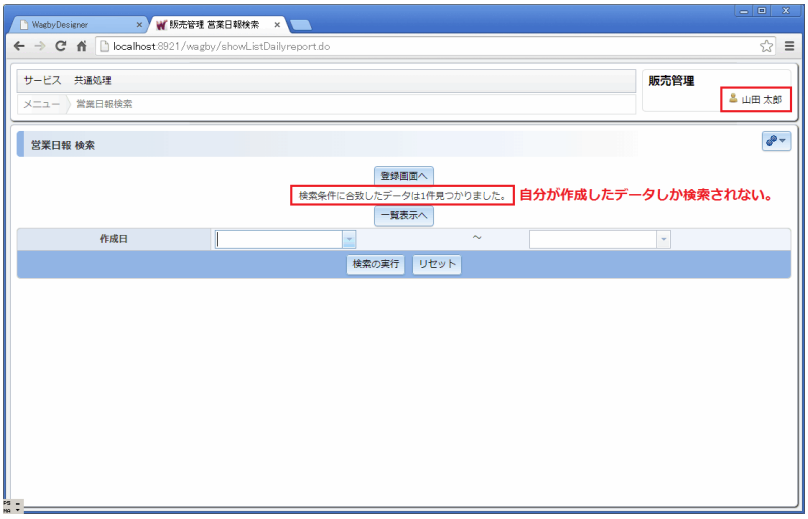


図 自分が作成した営業日報しか検索されない

定義方法

営業日報モデルの定義

営業日報モデルに、「作成者」という項目を用意します。

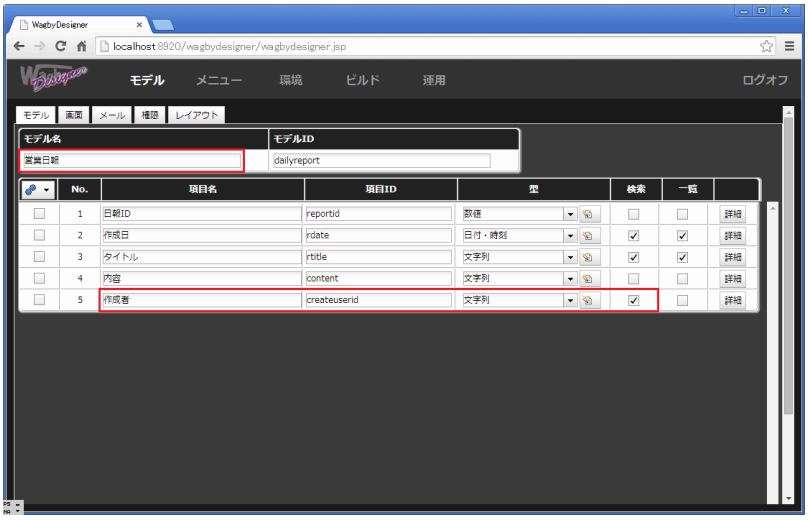


図 営業日報モデルに「作成者」項目を用意する

初期値の設定

「作成者」項目は登録画面表示時ならびに検索時にログオンIDをセットするようにします。

図 初期値にログオンIDをセットする

「検索時」とは、検索条件に用いる値という意味です。仮にこの項目に値が入力された場合でも、実際の検索時には、ここで指定された初期値が用いられます。

登録・更新画面では読み込み専用にする

「作成者」項目は登録・更新画面では読み込み専用にします。

図 登録・更新画面では読み込み専用にする

検索画面では隠し項目にする

モデル項目詳細定義の「検索制御>共通」内にある「隠し項目にする（全画面）」を有効にします。

この設定により、図にあるように、同項目は検索条件部には表示されなくなります。

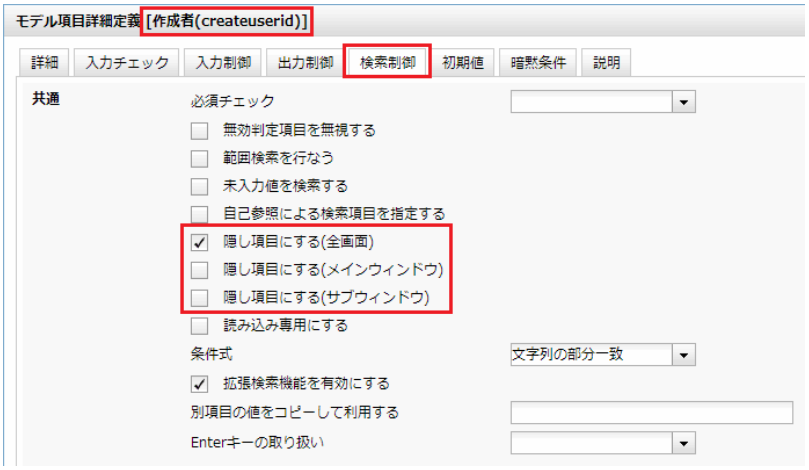


図 検索画面では隠し項目にする

この設定は「全画面」「メインウィンドウ」「サブウィンドウ」を個別に指定できます。

読み込み専用項目にする

図では、検索画面で作成者項目を表示した例です。ただし読み込み専用とし、入力は行えないとします。

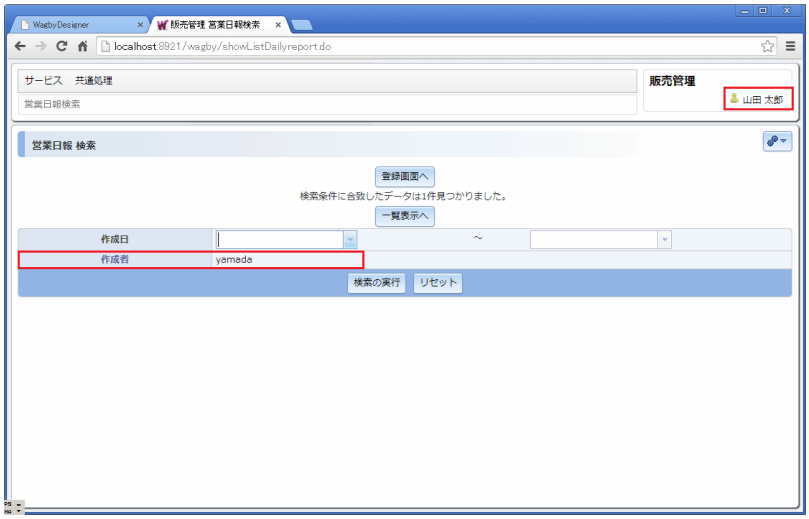


図 作成者項目が検索画面で表示されている。

定義方法

検索画面では読み込み専用にする

モデル項目詳細定義の「検索制御>共通」内にある「読み込み専用にする」を有効にします。

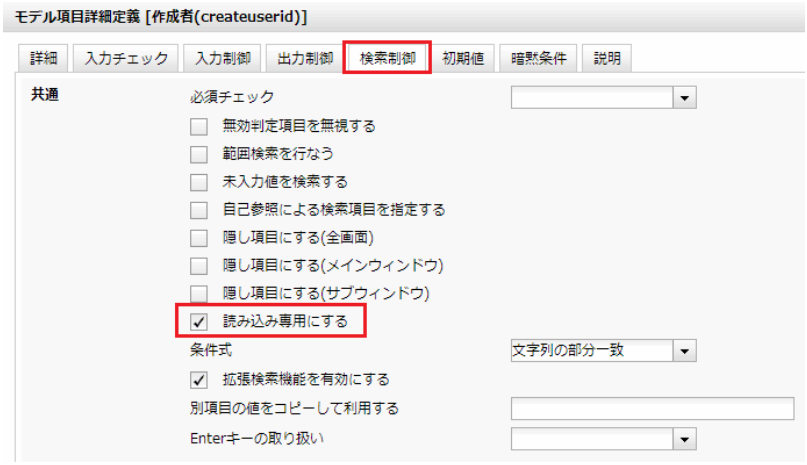


図 検索画面では読み込み専用にする

検索画面に、検索条件を表示させる

検索画面表示時の初期値を設定します。これによって、どの値で検索しようとしているかを明示します。

図 検索画面表示時の初期値を設定する

参照連動項目を検索条件に指定する

例

「会社」モデルを親とし、それに紐づく「顧客」モデルを子とした例

図, 図はそれぞれ〝ジャスミンソフト〟と〝ジャスミン商事〟に顧客が紐付いた（外部キー関係の）定義を行っています。

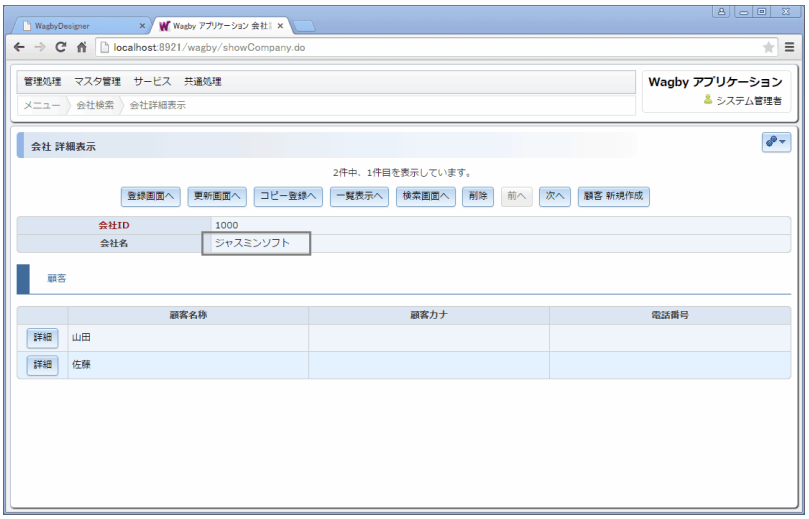


図 会社ジャスミンソフトに紐づく2件の顧客

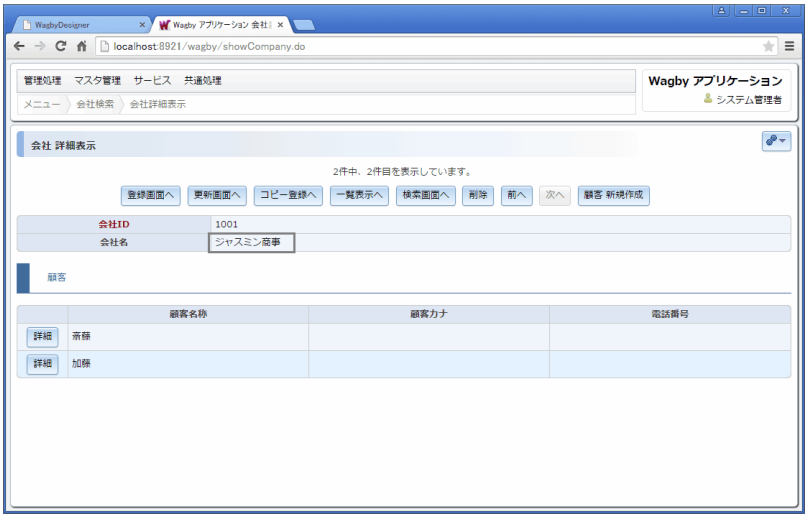


図 会社ジャスミン商事に紐づく2件の顧客

（子である）「顧客」モデルの検索画面を開きます。会社名は参照連動項目であり、顧客モデルのテーブルには（データベース定義として）用意されていません。（親である）「会社」モデルと結合して、検索を行うことができます。

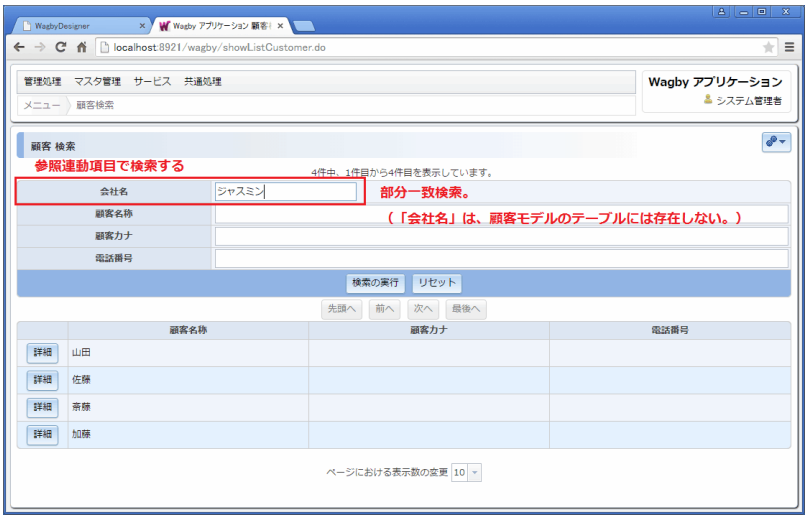


図 参照連動項目を使った部分一致検索を行った例(1)

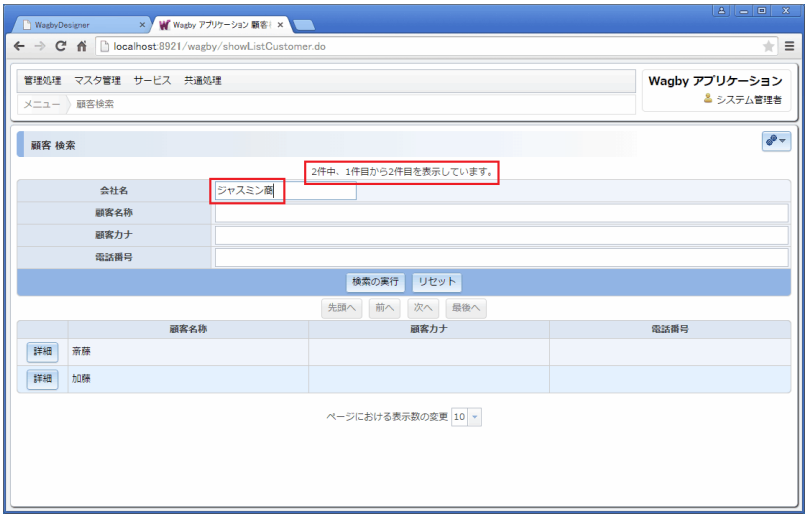


図 参照連動項目を使った部分一致検索を行った例(2)

定義方法

親子モデル関係の定義

この例では「会社」モデルを親と、「顧客」モデルを子として定義しています。

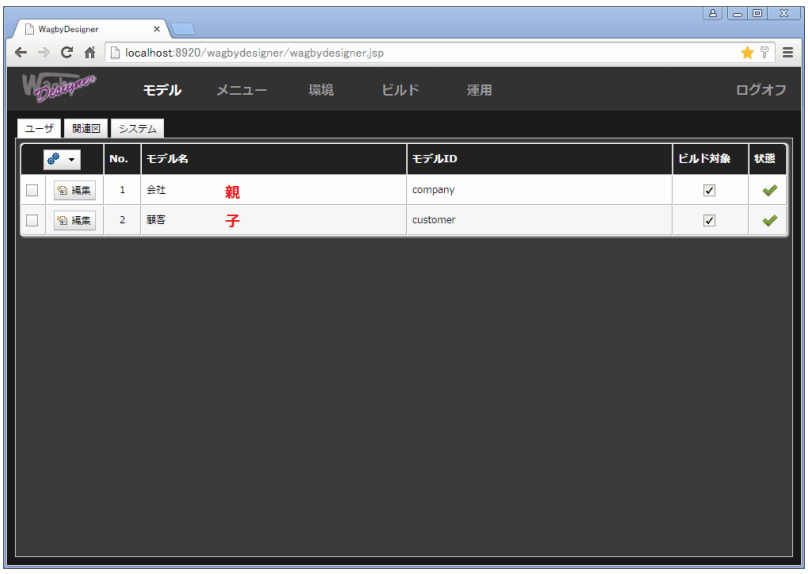


図 会社モデルと顧客モデル

会社モデル

「会社名」項目をデータベースに保存するようにします。（項目を定義すると、標準ではデータベースに保存されます。）

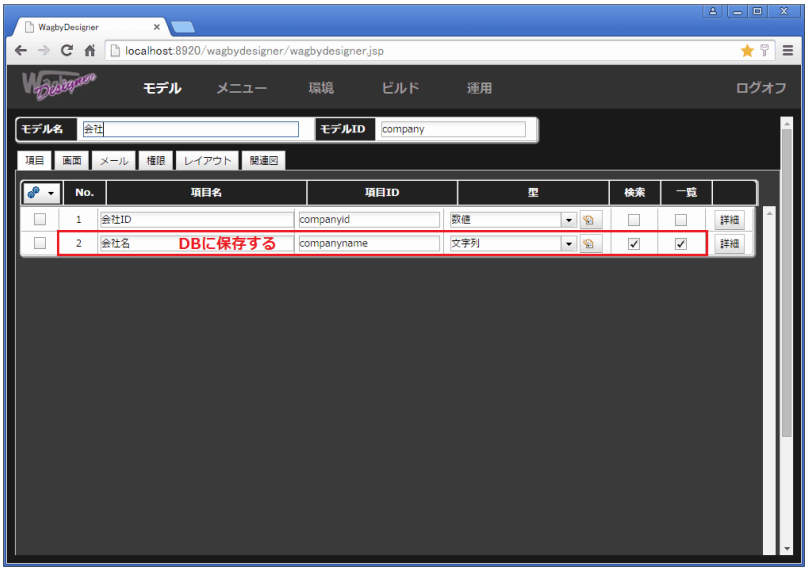


図 会社モデルの定義

顧客モデル

「会社ID」は外部キーとなります。会社IDの参照連動項目として「会社名」を用意します。

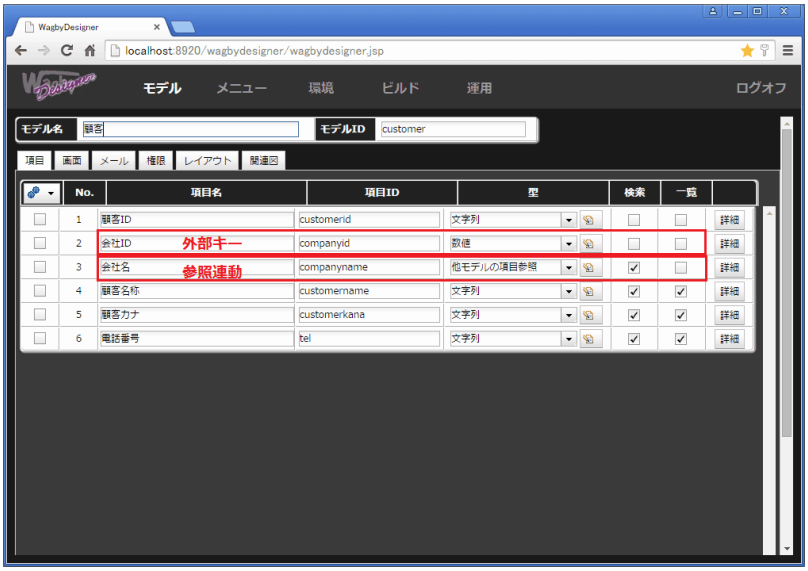


図 顧客モデル

会社ID

会社モデルの主キーを、外部キーとして指定します。

図 外部キーの設定

会社名

（顧客モデルの）会社IDをトリガーとして、（会社モデルの）会社名を参照連動します。

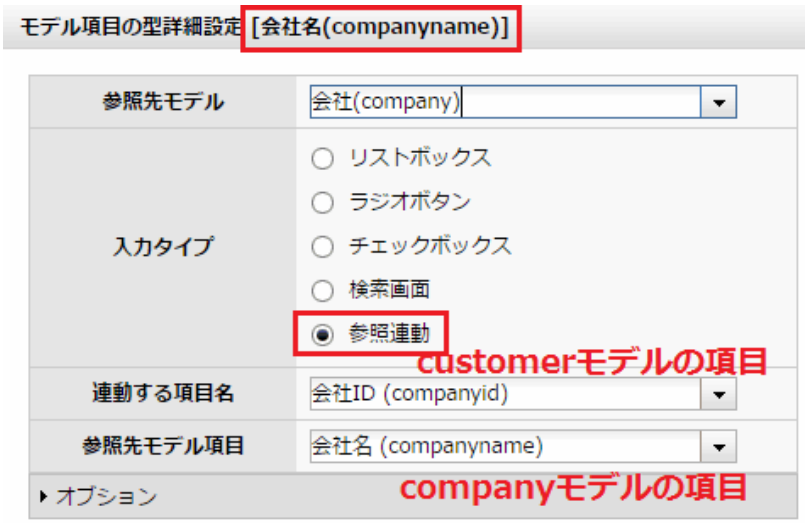


図 参照連動の設定

この項目はデータベースに保存しないとします。これによって会社モデルに対応したテーブル定義には、この会社名項目は含まれなくなります。

図 データベースに保存しない

データベースに保存するという設定を行うこともできます。この場合、親の会社モデルの会社名が変更された場合でも、子である顧客モデルのデータは古い会社名のまま維持されます。

検索条件 － 条件式

文字列型

条件式は標準で空白となっています。参照連動項目を検索条件として用いる場合、完全一致検索となります。

図 条件式が未設定

「文字列の部分一致」を選択すると、部分一致検索となります。（上の例で示した動作は、この設定を行った場合となります。）

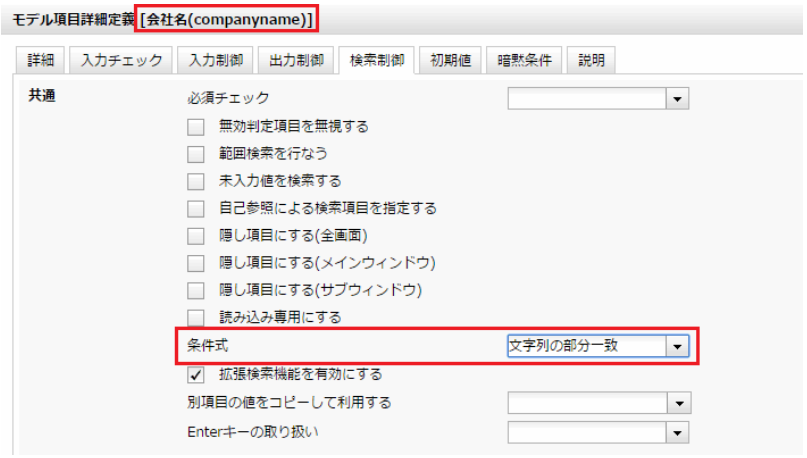


図 文字列の部分一致検索

数値、日付型

範囲検索を有効にすることで、範囲による検索条件を設定できるようになります。

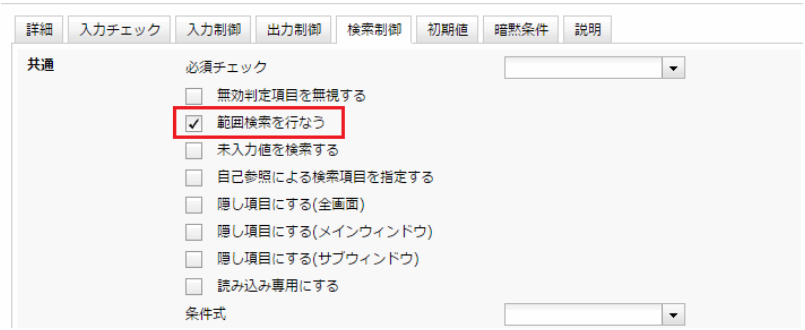


図 範囲検索を指定する

仕様・制約

参照先項目はデータベース保存されていること

参照先の項目は、"データベースに保存する" ようにしてください。（上の例における「会社モデル」の「会社名」はデータベースに保存する設定となっている必要があります。）

連動元項目はデータベース保存されていること

連動元の項目も、"データベースに保存する" ようにしてください。（上の例における「顧客モデル」の「会社ID」はデータベースに保存する設定となっている必要があります。）

指定できない場合

下記の指定は行えません。

- 参照先項目（上の例では「会社」モデル）の型が「郵便番号」である。
- 参照元項目（上の例では「顧客」モデル）に「リストボックス／チェックボックス／ラジオボタン切替」の指定を行っている。
- 参照元項目（上の例では「顧客」モデル）に「追記型リストボックス」の指定を行っている。

暗黙条件

例

商品モデルの「価格」が「0より大きい」ものだけを検索対象とする、という暗黙条件を加えた例を示します。図で、価格が0円の商品を登録します。

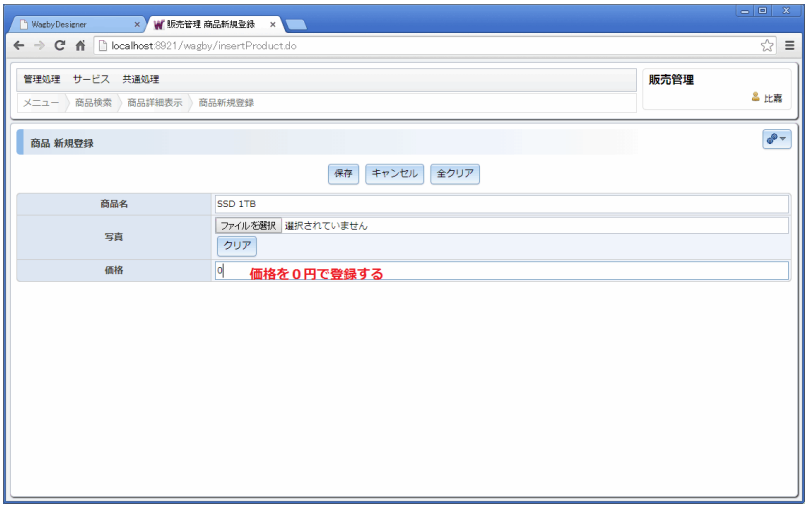


図 価格が0円の商品を登録する

図に示すように、検索結果に表示されません。これは隠れた検索条件項目「価格」が用意されており、「0より大きい」という条件が常に有効になっているためです。

図 価格が0円の商品は検索されない。

暗黙条件が適用されない権限を用意する

暗黙条件を設定すると、条件に漏れたデータは誰も閲覧できなくなります。そこで、この条件が適用されない権限を設定することができます。 図では「商品管理者」というプリンシパルを用意した例を示します。

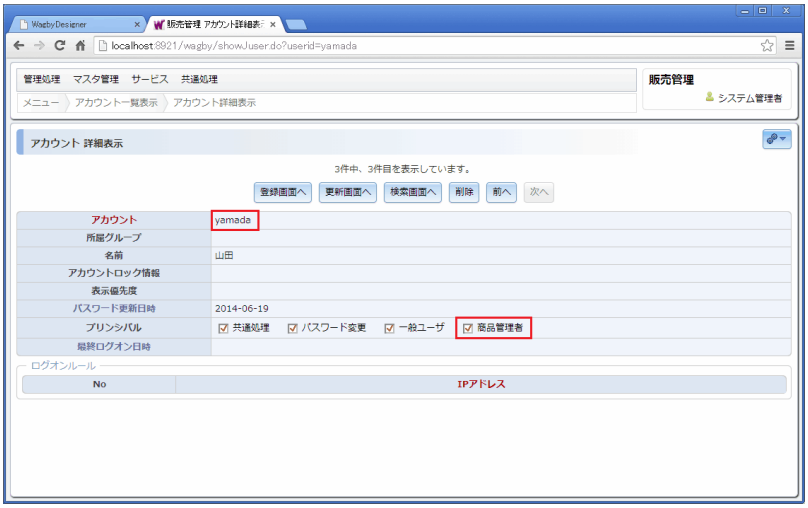


図 商品管理者プリンシパルを用意する

このプリンシパルをもったアカウントでログオンした場合、商品データはすべて検索されます。

図 すべての商品データ（暗黙条件を適用しない）

定義方法

シンプルな定義

暗黙条件を適用したいモデル項目の詳細定義ダイアログを開きます。「暗黙条件」タブで、条件式を設定します。

図の例では、「0より大きい」と指定しています。

暗黙条件は複数、指定することができます。すべての条件を満たすデータだけが検索対象になります。

図 暗黙条件の設定

図の設定では、どのアカウントも（条件に適合しなかったデータを）検索することはできません。これを回避するために、「適用外の権限」を同時に指定することができます。[後述]

適用外の権限を設定する

はじめに、このモデルに新しい権限（パーミッション）を作成します。

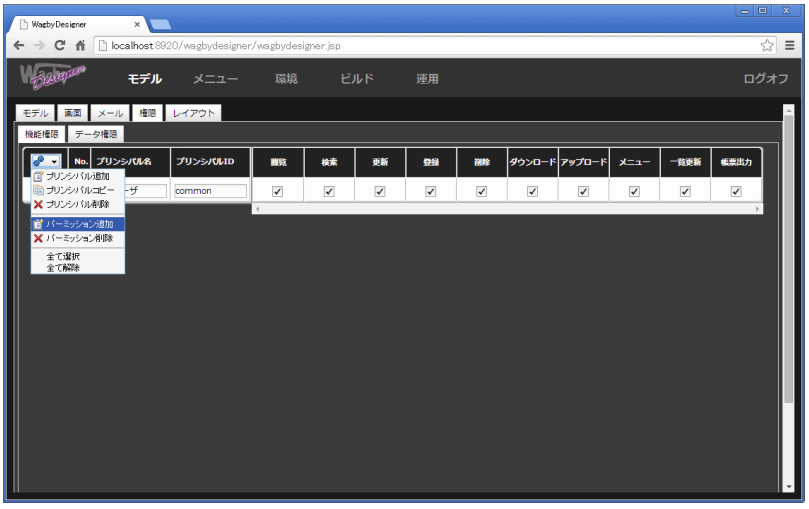


図 パーミッションの作成(1)

ここでは「除外（p_except)」という名前の権限を用意しました。

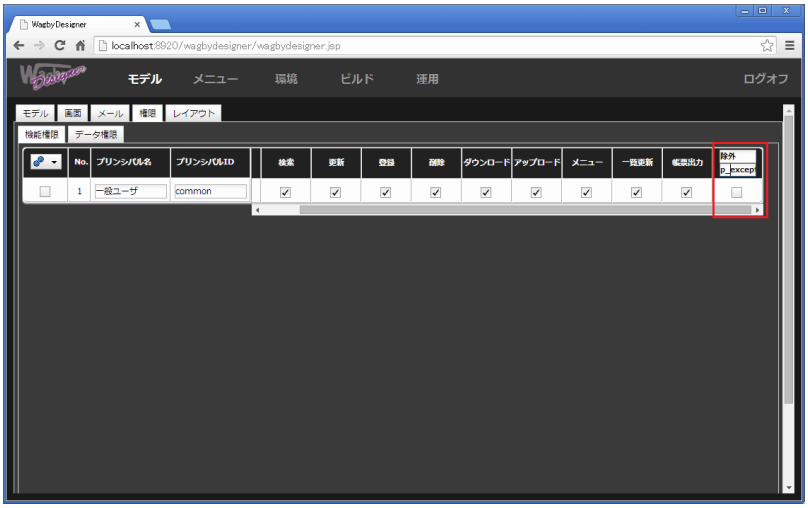


図 パーミッションの作成(2)

追加した権限を管理するためのプリンシパルを作成します。

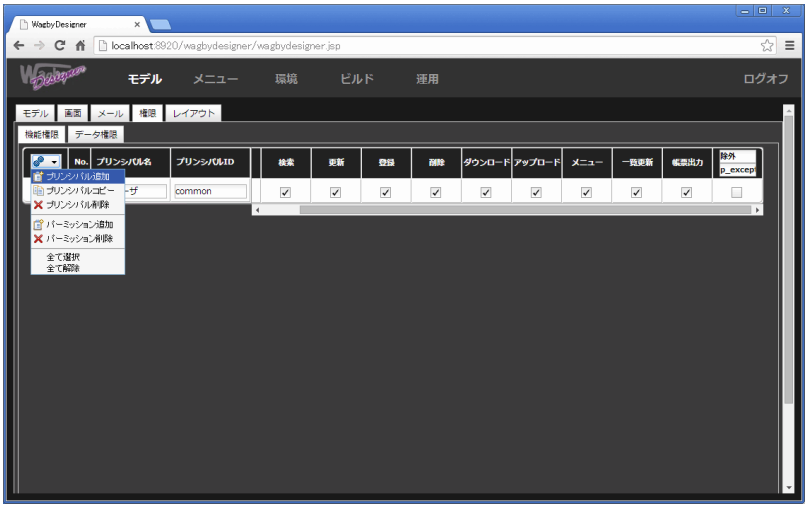
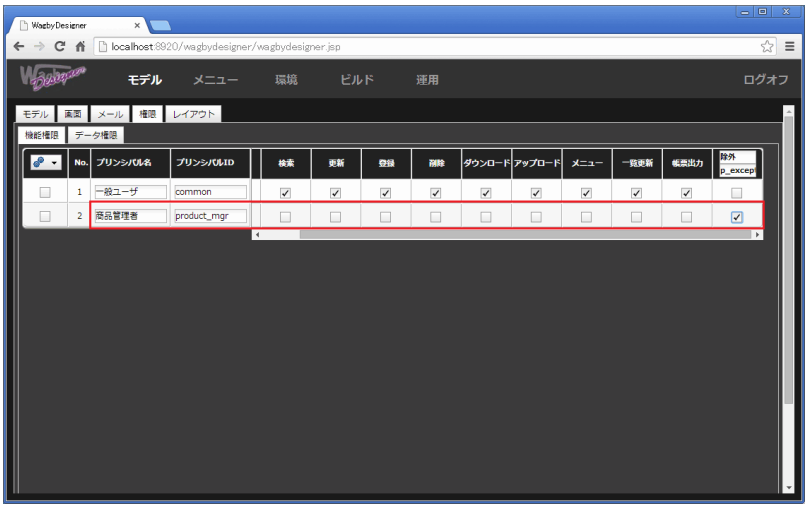


図 プリンシパルの作成(1)

ここでは「商品管理者 (product_mgr)」というプリンシパルを用意しました。除外パーミッションを持ちます。



暗黙条件の設定で、作成した"p_except"（除外）パーミッションを「適用外の権限」欄に設定します。

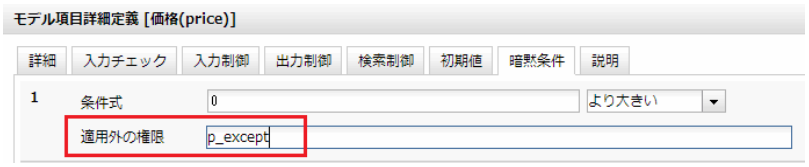


図 適用外の権限を指定する

ここではパーミッションを設定します。プリンシパルではありません。

図に示すように、このプリンシパルをもったアカウントは、暗黙条件がスキップされます。

暗黙条件の仕組み

暗黙条件は、データベース問合せ時に SQL の where 句に自動的に付与されます。 このため、データベース自体にはデータが格納されます。システム利用時には、検索されることはありません。（適用外の権限を除く）

関数を利用する

文字列型項目の場合、条件値の部分に項目を参照せずに計算結果を取得できる関数を利用することができます。

ログオンアカウントを暗黙条件とする

```
USERNAME()
```

履修年度を暗黙条件とする

```
FISCALYEAR(NOW())
```

USERINFO関数を使ってjuserの値を取得する

```
USERINFO("name")

TOINT(USERINFO("age"))
```

USERINFO関数の戻り値は文字列型です。整数型の値を使いたい場合は、TOINTなどの関数と併用してください。

利用できない関数の考え方

暗黙条件には、関数の引数に特定モデルの項目が紐付かないものは利用できます。 例えば絶対値をとる ABS という関数を例にとって説明します。

```
ABS(-1)
```

は利用できますが

```
ABS(${item1})
```

は利用できません。

特別表記nowを利用する

「日付」「時間」「日付と時間」項目の暗黙条件に "now" 表記を使用することができます。

関数を利用した場合

関数の結果が NULL または 空文字となる場合は、暗黙条件による絞込は行われません。

OR検索

暗黙条件で OR 検索を利用する場合、暗黙条件の設定に代わってスクリプトで実現します。

仕様・制約

システム管理者の扱い

システム管理者には暗黙条件が適用されません。

通常の検索条件との併用は不可

暗黙条件と、通常の検索条件を併用することはできません。この場合は暗黙条件が優先され、検索条件に値を入力しても無視されます。

範囲検索との併用は不可

検索制御の「範囲検索」と暗黙条件は同時に指定できません。

検索・一覧表示画面が必要

そのモデルの検索・一覧表示画面を用意しなかった場合、暗黙条件は利用できません。（設定は無効となります。）

トラブルシューティング

暗黙条件が効かないという場合の確認方法を説明します。

- システム管理者(admin)でログオンしていませんか？　システム管理者には暗黙条件が適用されません。
- ″適用外の権限″ を定義したときは ″権限 > 機能権限″ の設定で、これを同名のパーミッションとして用意する必要があります。

全文検索

全文検索とは

全文検索機能は、利用するリレーショナルデータベースとは別に検索専用のインデックスファイルを利用します。次のようなアプリケーションに向いています。

- 大量の文字情報を高速に検索する。
- 添付ファイルの内容を高速に検索する。

例と定義方法

例

顧客モデルの「氏名」「備考」項目に全文検索を適用した例です。検索画面は「通常検索」と「全文検索」の二つのタブが用意されます。

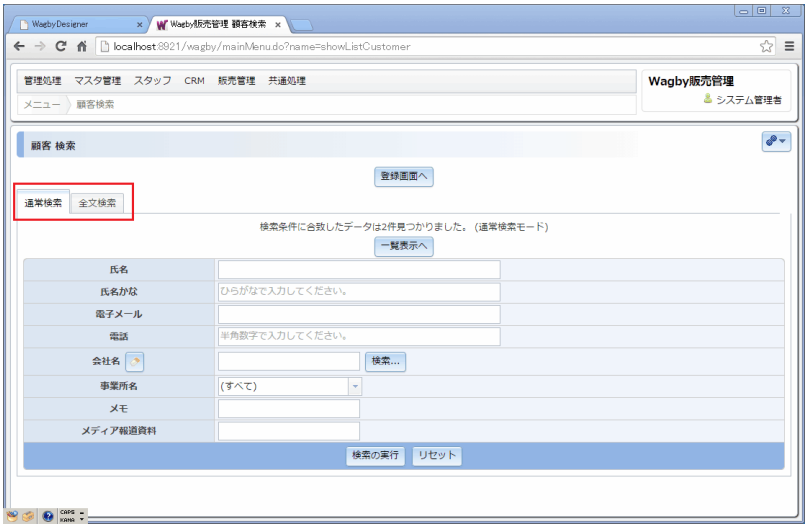


図 通常検索と全文検索

全文検索タブを開いたところを図に示します。入力欄は1つのみとなります。これでモデル内の（全文検索を指定した）項目を横断的に検索します。

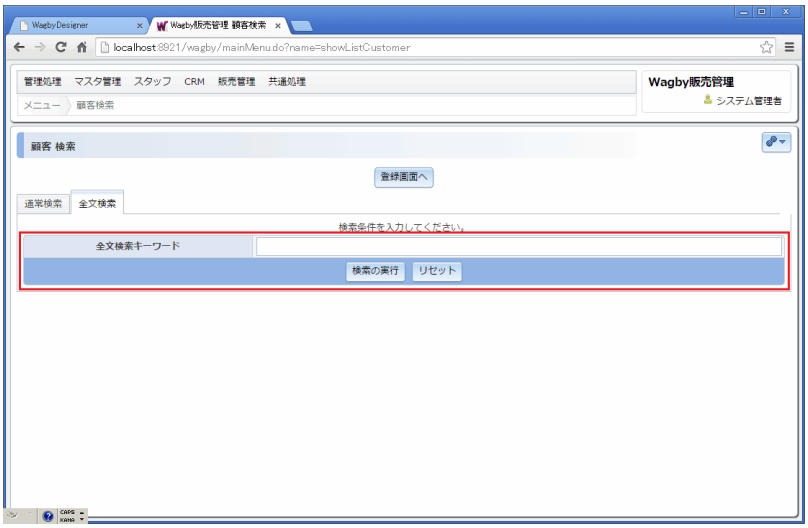


図 全文検索タブ

検索結果の一覧表示画面を図に示します。全文検索では一覧表示の方法も異なっています。全文検索の対象とした項目の内容が表示され、かつキーワードの部分は強調（太字）となります。

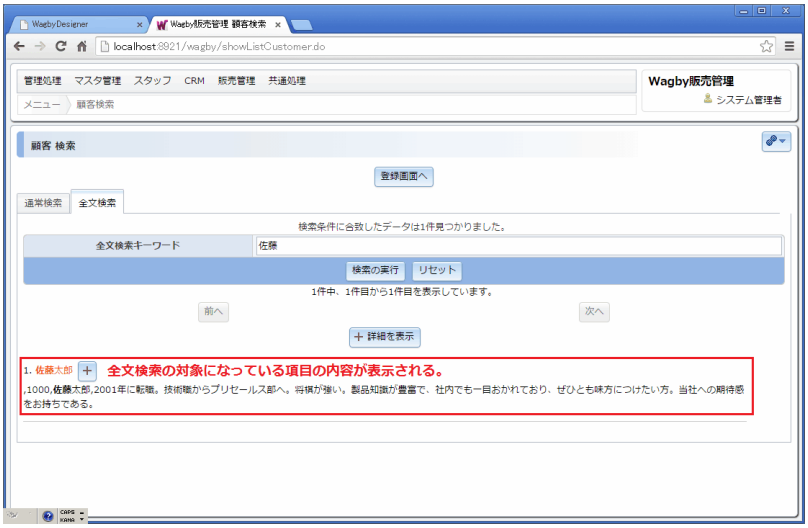


図 全文検索の一覧表示

定義方法

項目「氏名」は検索条件に使う、という設定を行っています。 この状態で「モデル項目詳細定義 〉 検索制御 〉 全文検索」にある「全文検索を行う」を有効にします。

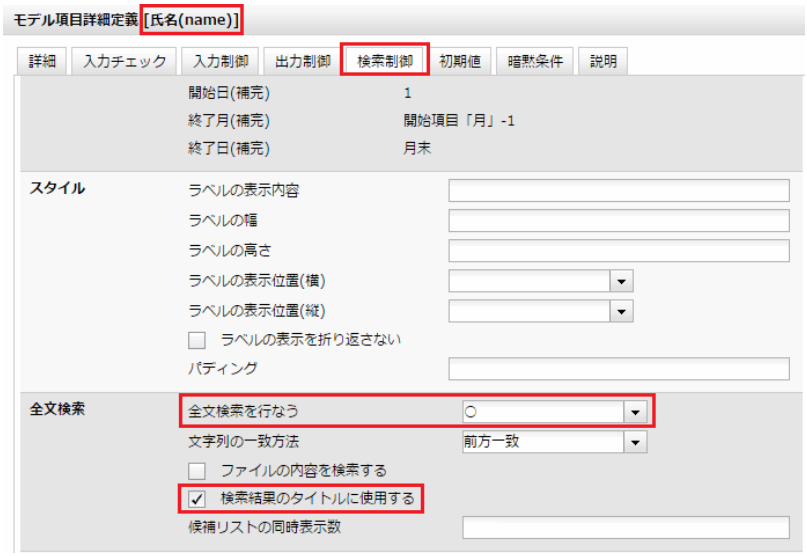


図 全文検索の設定

全文検索に関するその他の設定は次のとおりです。

検索結果のタイトルに使用する

全文検索に使用する項目のうちいずれか一つに本設定を適用してください。検索結果表示時のタイトルとして利用します。

(163)

文字列の一致方法

候補リストの同時表示数

この二つはサジェストの設定で利用します。

ファイルの内容を検索する

(163) どの項目にも、本設定を行っていなかった場合、最初にみつかった「全文検索を行う」モデル項目をタイトルとして使用します。8.4.0

次節で説明します。

添付ファイルの内容を検索する

添付ファイルの内容も全文検索の対象とすることができます。Word, Excel, PowerPoint, PDF といったバイナリファイルおよび TEXT, HTML, RTF, XML といったテキストファイルに対応しています。

PDFファイルに含まれる文字列で検索した例を図に示します。

図 PDFファイルの全文検索

画面上の「+」ボタンを押下すると、ファイル内容の詳細が表示されます。ボタンは「-」に変わります。

図 PDFファイルの内容（一部）が表示される

項目の設定

ファイル型項目を用意し、検索対象とします。さらに全文検索を有効にすると同時に「ファイルの内容を検索する」を有効にします。

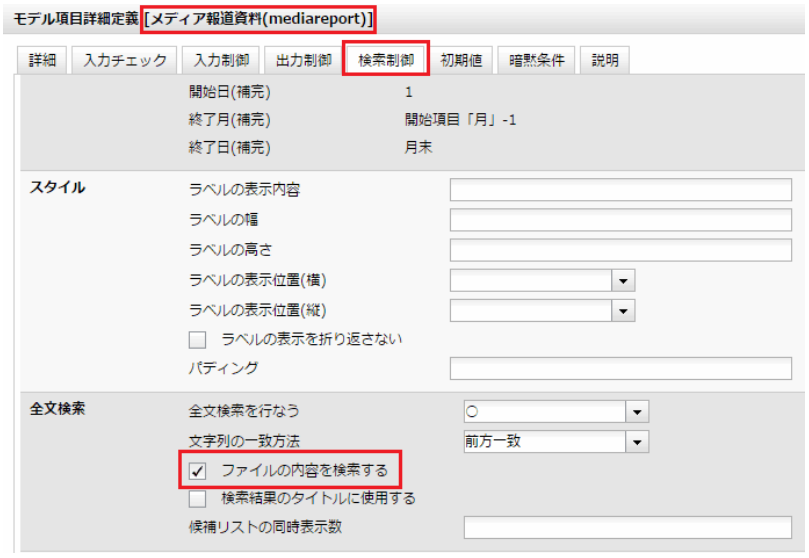


図 ファイルの内容を検索する設定

文字抽出プログラムの指定

「環境 ＞ アプリケーション」タブを開きます。「全文検索 ＞ 文字抽出プログラム」に "Tika" を指定します。(*1) [164]

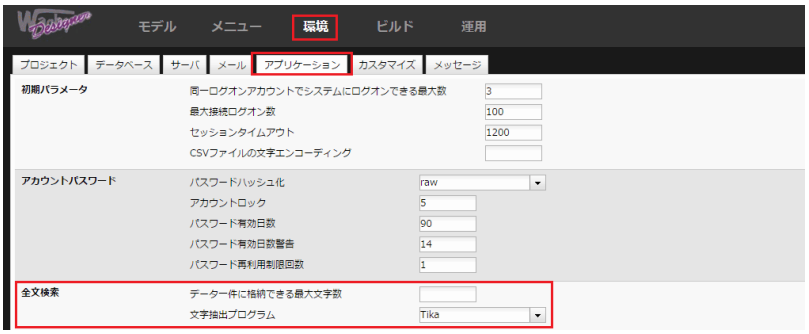


図 文字抽出プログラムの設定

データ一件に格納できる最大文字数

「データ一件に格納できる最大文字数」を指定することができます。未指定時は330　万文字となります。この値を超える文字は、インデックス作成の対象とならないため、検索されません。

あまり大きな文字数を指定すると、インデックス作成に時間がかかり、メモリ不足となる可能性もあります。 業務要件に合わせた数字を設定してください。以下、設定時の目安を示します。

- 新聞の朝刊がおよそ 40 ～ 50 万文字であると言われています。（標準値はおおよそ朝刊 7 日分に相当します。）
- 繰り返しコンテナにファイル添付ができるようなモデルを定義した場合、複数ファイルの合算文字数の上限となります。

仕様・制約

検索キーワードの入力ルール

全文検索タブにおける検索キーワードの入力ルールは次のとおりです。

(164) 1. Apache Tika はオープンソースのテキスト文字列抽出ライブラリです。Wagbyに標準で同梱されています。

ルール	説明	例
AND 検索	複数のキーワードがすべて含まれるデータを検索します。	東京 ニューヨーク ロンドン
OR 検索	いずれかのキーワードがすべて含まれるデータを検索します。先頭に OR と指定します。	OR 東京 ニューヨーク ロンドン

- 複数キーワードを指定する場合は「半角スペース」または「全角スペース」で区切ります。
- 全角スペースそのものを検索する場合、二重引用符で括ってください。
- "OR" は半角文字をお使いください。大文字、小文字は区別しません。
- キーワードは二文字以上が必要です。一文字のキーワードでは検索できません。

表示順序

全文検索機能を用いた場合は「検索語の一致度の高さ」を基準にして検索結果の一覧が表示されます。 通常の Wagby による一覧表示とは異なり、項目毎の並び替え機能は動作しません。

指定できる項目

全文検索機能の対象となる項目は以下の通りです。

項目の形態	項目の型
(繰り返しでない)単一項目	文字列、ファイル名
繰り返し項目	文字列
繰り返しコンテナ内の項目	文字列、ファイル名

さらに次の条件を満たす必要があります。

- 参照権限(条件付き)が設定されていないこと。
- 一覧表示画面で隠し項目に設定されていないこと。

この条件を満たさない項目は全文検索機能の対象となりません。（「全文検索機能を使用する」を有効にしても無視されます。）

詳細画面は必要 !-- N20170425_140554 -->

全文検索を有効にしたモデルは、詳細画面を作成するようにしてください。一覧表示から詳細表示画面へのリンクが常に用意されます。

通常検索との関係 !-- H20190226_191452, aloha 234951 -->

暗黙条件やデータ権限、検索条件の初期値設定などがあった場合、通常検索の結果と全文検索の結果はマージされます。具体的には通常検索と全文検索のそれぞれの結果の積集合が、検索結果となります。

さらに通常検索の検索条件として利用者が設定した値（数値、日付、全文検索の対象となっていない文字列型）の絞り込み条件も適用されます。例えば通常検索で日付範囲指定を行い、かつ全文検索で特定のキーワードを指定することができます。(*2) 8.2.2

これらのルールから、Wagby の全文検索は単なる文字列検索ではなく、Wagbyの暗黙条件やデータ権限を考慮し、かつ通常検索とのAND検索を行なった結果を表示します。

(165)

検索結果数の上限 !-- N20190319_210314 -->

全文検索結果の上限は 1000 件となっています。1000件以上の結果となる場合は、検索語を追加する、または通常検索の条件を追加するなどして、検索結果を絞り込むようにしてください。

(165) 2.全文検索と通常検索の検索結果をマージする動作の詳細は次の通りです。　全文検索: 全文検索対象項目に対して指定されたキーワードで検索を行う。　通常検索: 全文検索対象項目を除いた検索項目を使ってRDB検索を行う。 これらの結果より、主キーが同じもの(積集合)が検索結果として表示されるようになります。

インデックスファイルについて

全文検索・サジェストのために利用する検索用インデックスファイルはデータベースとは別に作成されます。詳細...

サジェスト機能

全文検索の応用例に「サジェスト検索」があります。 詳細は"他モデルの参照 > サジェスト"をお読みください。

検索画面のみに表示され、検索条件には使われない項目を用意する

例

検索画面において「対応期限」という項目の検索を実行するにあたり、操作を容易にするために "3日後", "5日後", "7日後" といったラジオボタンを選択することで「対応期限」の検索条件の日付を自動設定する例を説明します。

図に検索画面を示します。ラジオボタンで対応期限を選択できる操作となっています。

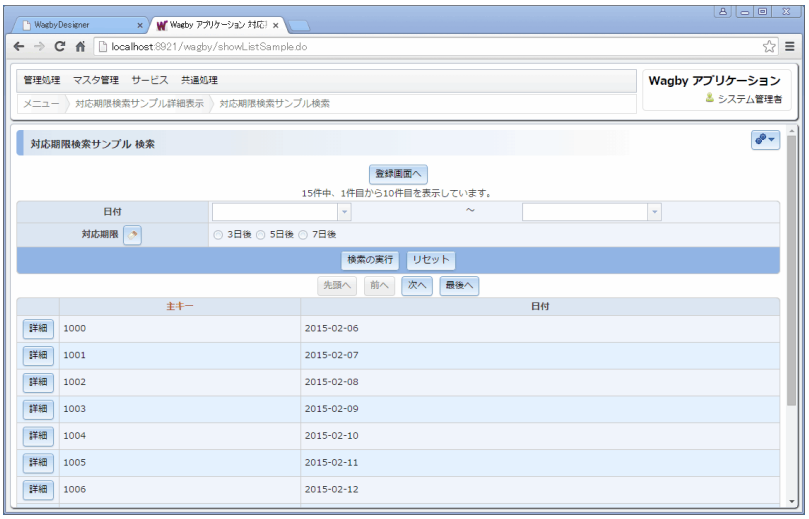


図 検索画面

ラジオボタンを選択することで、絞り込み検索ができます。

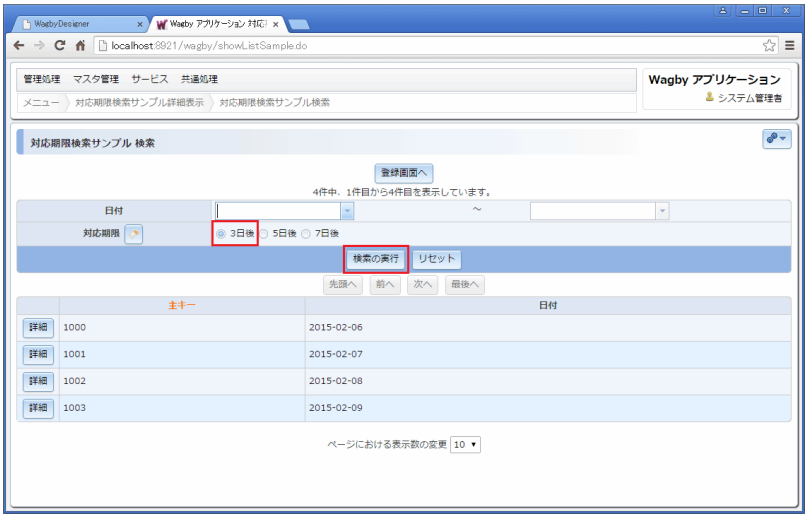


図 検索の実行(1)

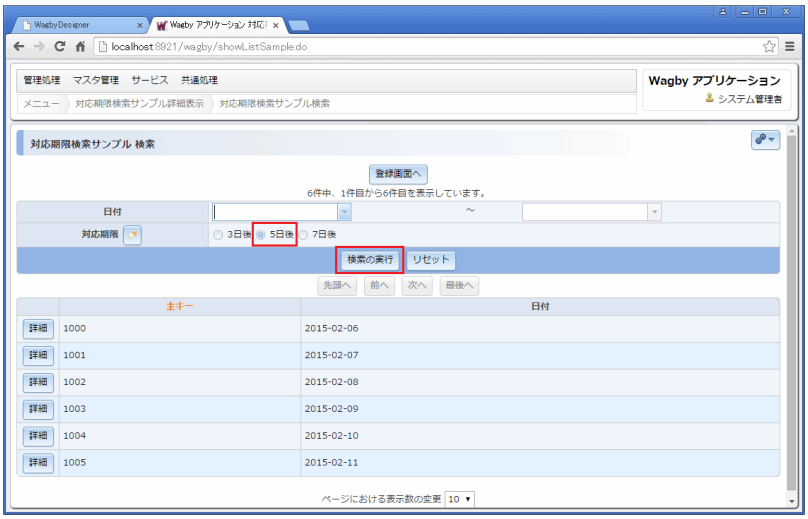


図 検索の実行(2)

定義方法

上で紹介したサンプルアプリケーションを実現するモデルを図に示します。

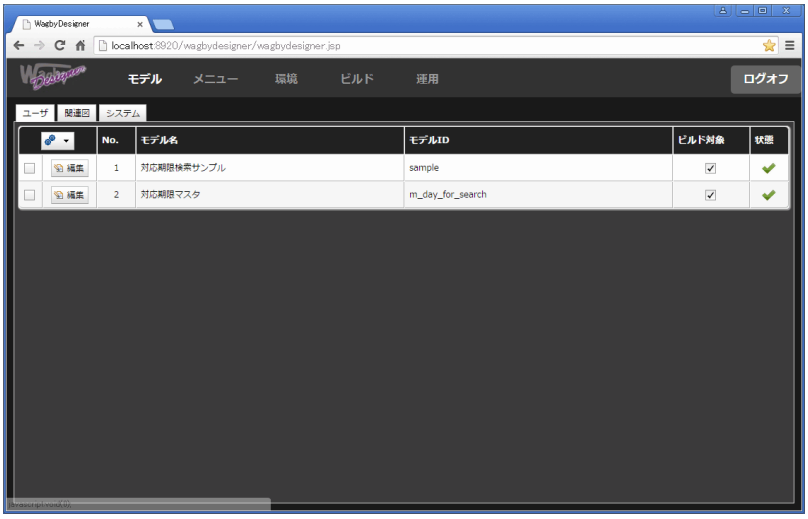


図 モデル一覧

「対応期限」で用いる選択肢は次のとおりです。

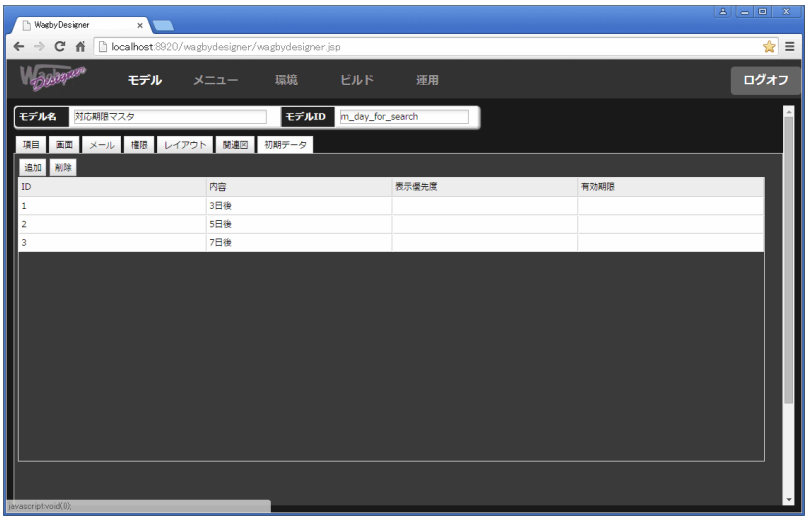


図 選択肢の一覧

「検索用日付」項目

ポイントは、"3日後"，"5日後"，"7日後" を表示するために「日付」項目とは別に「検索用日付」と「対応期限」という二つの項目を設定することです。「検索用日付」項目はラジオボタンとして用意された「対応期限」の値によって自動設定されるようにします。

「検索用日付」項目の初期値に式を設定します。「対応期限」の設定値によって検索対象日を自動設定します。

図 初期値の設定

検索時:

```
IF(
  NOT(SELECTED(${day_for_search})),
  null,
  TODAY()
)
```

検索時（範囲検索の上限項目）:

```
IF(
  NOT(SELECTED(${day_for_search})),
  null,
  IF(
    ${day_for_search#id}==1,
    MOVEDAY(TODAY(), 3),
    IF(
      ${day_for_search#id}==2,
      MOVEDAY(TODAY(), 5),
      MOVEDAY(TODAY(), 7)
    )
  )
)
```

「検索用日付」項目は、同じ検索画面に用意された「日付」項目に値がセットされていた場合、その値を転記するようにします。

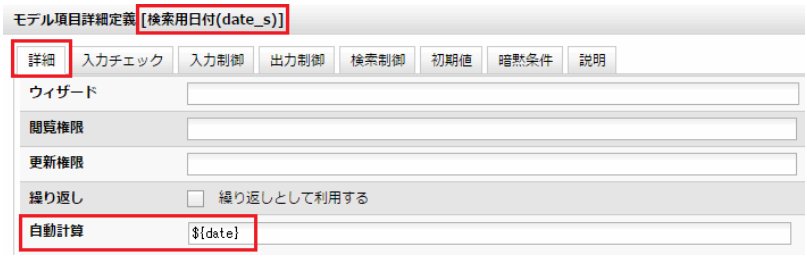


図 計算式の設定

```
${date}
```

「検索用日付」項目は入力時に隠し項目とします。

図 入力制御-隠し項目の設定

「検索用日付」項目は出力時にも隠し項目とします。

図 出力制御-隠し項目の設定

さらに「検索用日付」項目は検索画面でも隠し項目とします。

(166)

また、検索画面でもラジオボタンとして操作できるよう「モデル参照 > 表示形式の切り替え」を空白にします。

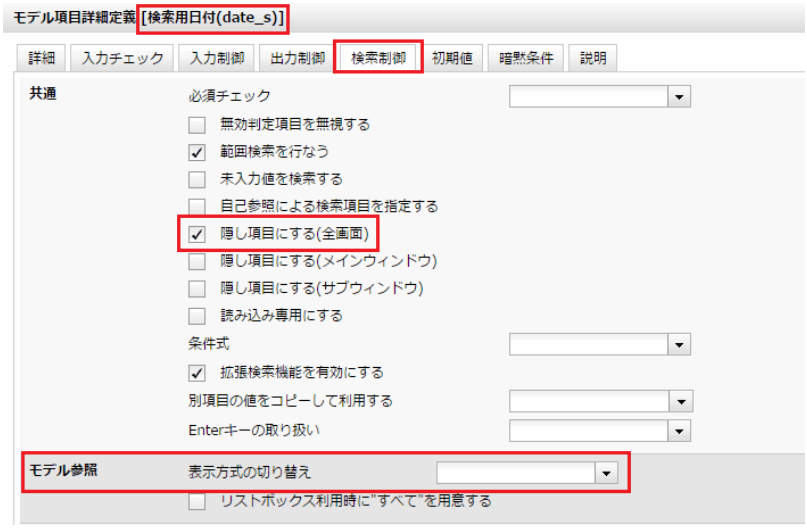


図 検索制御-隠し項目の設定

「対応期限」項目

「対応期限」項目は「検索条件」に「○」をつけていますが、実際の検索処理には用いません。ここで選択した値は、「検索用日付」項目に反映されるためです。よって「値をデータベースに保存する」の設定を解除します。

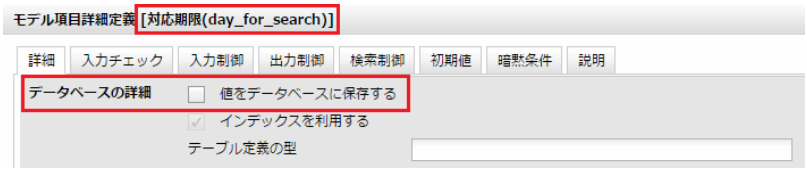
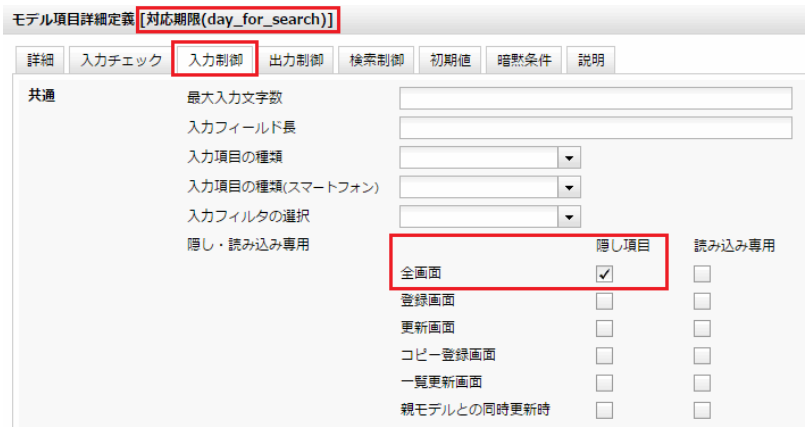


図 値をデータベースに保存しない

検索条件に指定かつ、データベースに保存しない項目は、検索用のSQLに反映されません。検索条件の入力欄としてのみ利用できます。

「対応期限」項目も入力、出力で隠し項目とします。



(166) 隠し項目ですが、実際には検索絞り込みに用いられます。検索値が「日付」や「対応期限」といった項目から自動設定されます。

図 入力制御-隠し項目の設定

図 出力制御-隠し項目の設定

濁音や拗音の違いを考慮したカナ検索の実現

例

この設定を行っておくことで濁音、清音のいずれでも検索ができます。 ここでは顧客名「山田」のカナ名が「ヤマダ」か「ヤマタ」か不明な場合でも、ひとまずどちらかで登録し、後でどちらでも検索できる例を示します。

氏名「山田 太郎」、カナ氏名「ヤマダ タロウ」と登録します。

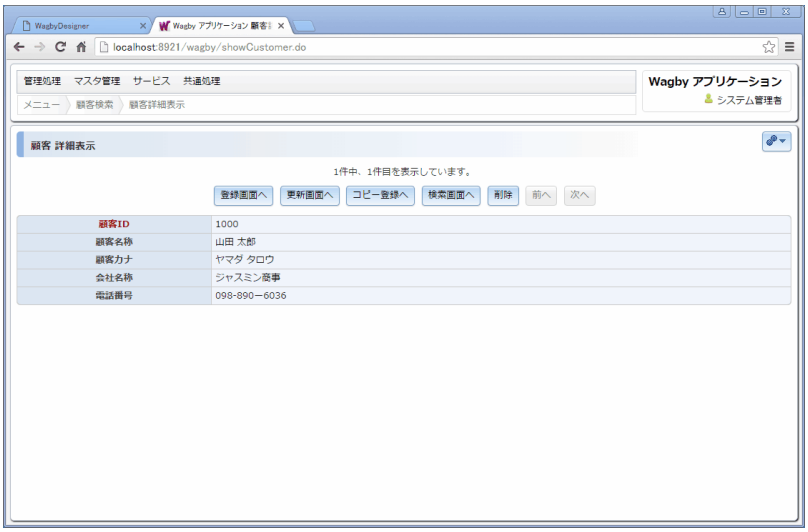


図 顧客詳細表示画面

カナ氏名「ヤマダ」で検索します。

図 「ヤマダ」で検索

カナ氏名に「ヤマタ」と指定して検索しても、同じデータを見つけることができます。

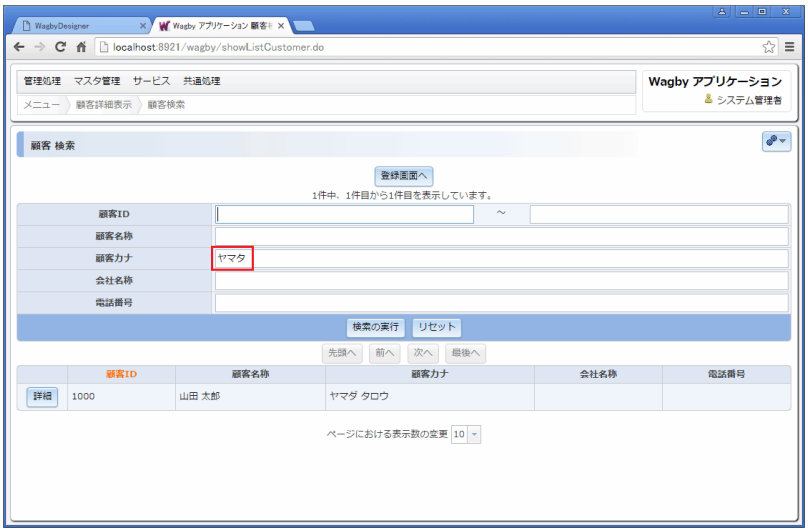


図 「ヤマタ」で検索

定義方法

定義のポイントは「カナ氏名」を2項目用意することです。「customerkana」は通常の入力及び一覧表示用です。もう一方の「customerkana2」は入力画面では隠し項目にし、検索条件用として使用します。

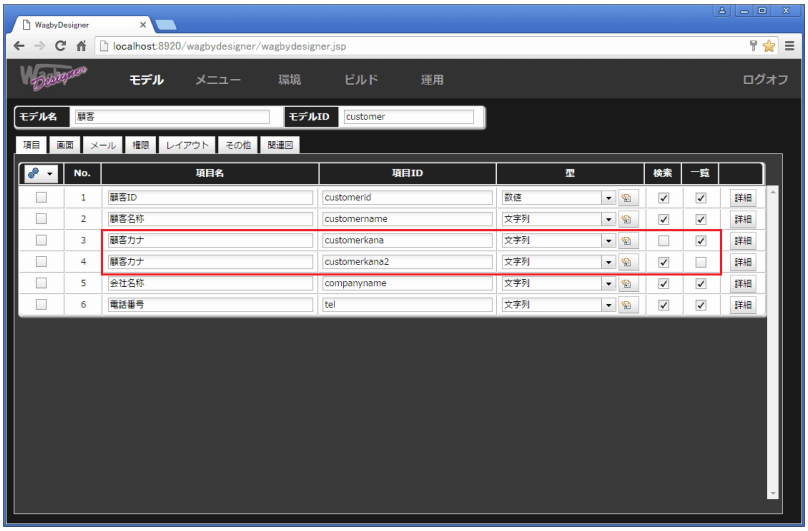


図 モデル項目一覧

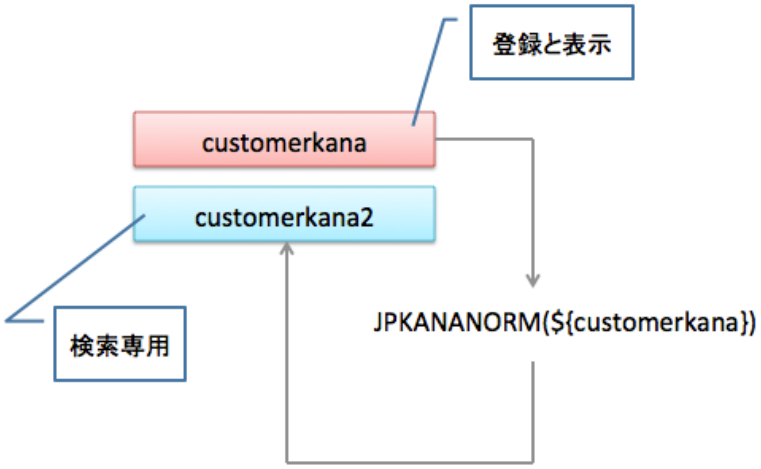


図 customerkana項目とcustomerkana2項目の関係

以降の設定は、すべて「customerkana2」項目に対して行います。

計算式とデータベース保存の設定

次のように式を設定します。

```
JPKANANORM(${customerkana})
```

図 自動計算の設定

入力された customerkana の値を JPKANANORM 関数を使って濁音や拗音を正規化します。また、この customerkana2 項目で検索を行うため、データベースに保存します。

隠し項目にする

入力制御で、隠し項目とします。

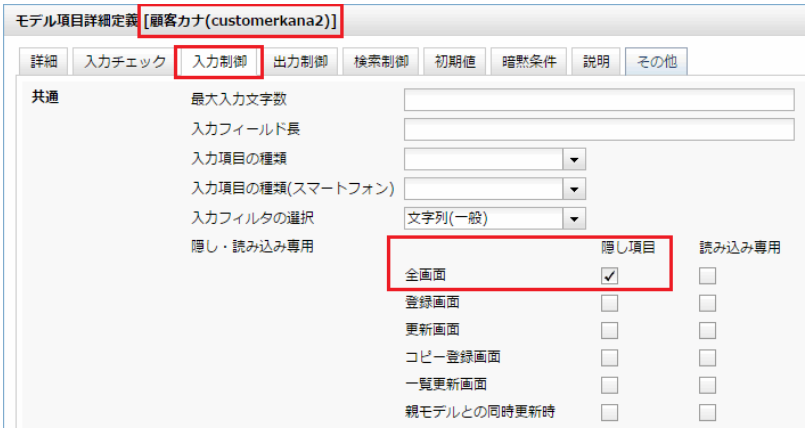


図 入力時に隠し項目とする

同じく出力制御でも、隠し項目とします。

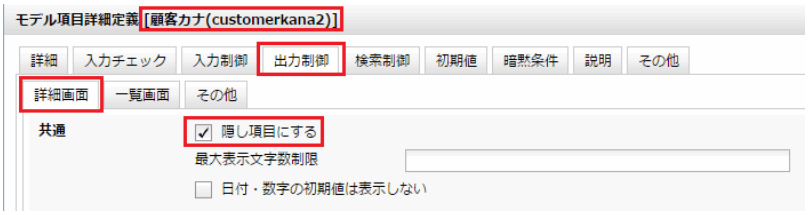


図 出力時に隠し項目とする

CSV出力を行わないようにします。この設定で、CSVダウンロード機能を用いた場合でも、この項目は含まれないようになります。

図 CSV出力を行わない

検索条件の設定

検索の初期値として、検索画面で入力された文字（検索条件文字列）を清音化します。データベースに格納された文字列も同じように清音化されているため、必ずヒットするようになります。

```
JPKANANORM(${customerkana2})
```

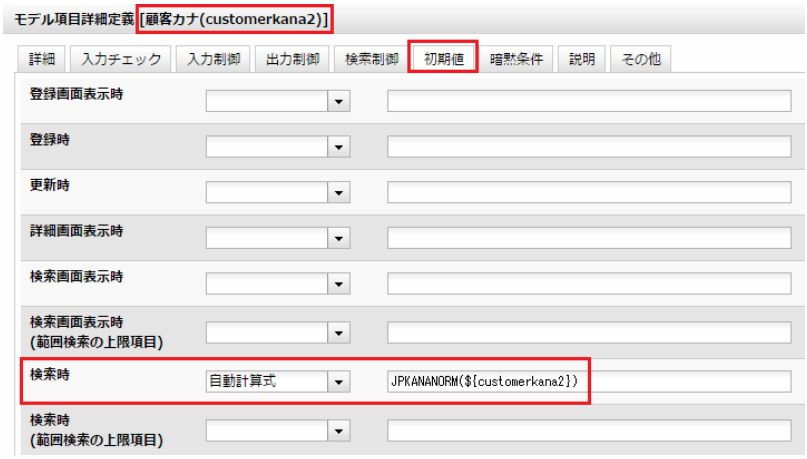


図 「初期値」の設定

入力フィルタを使わずに全角半角文字が混在した文字列を検索する

例

Wagby は標準で「入力フィルタ」という機能を提供しています。この機能を用いると文字列型の項目はデータベースに保存されるときに全角文字が半角文字に自動的に置換されます。

しかしここでは入力フィルタ機能を使わず、全角半角文字が混在した文字列項目を想定します。 このような状態で、例えば「ＪａｓｍｉｎｅＳｏｆｔ」（全角文字かつ大文字小文字が混在）で登録された項目に対して、検索文字列「jasminesoft」（半角文字かつすべて小文字）で検索できるようにします。

ここでは「会社名称」項目を例に説明します。会社名「ＪａｓｍｉｎｅＳｏｆｔ」と登録します。全角文字です。

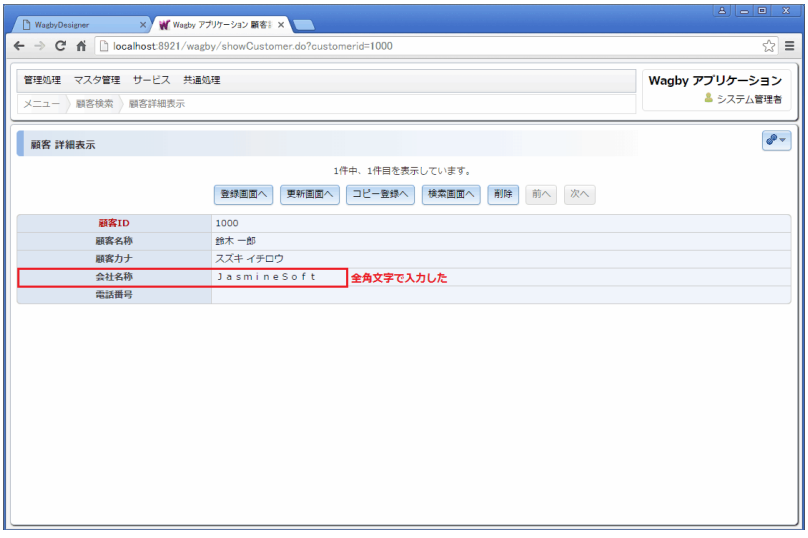


図 顧客詳細表示画面

半角文字で検索してもヒットします。

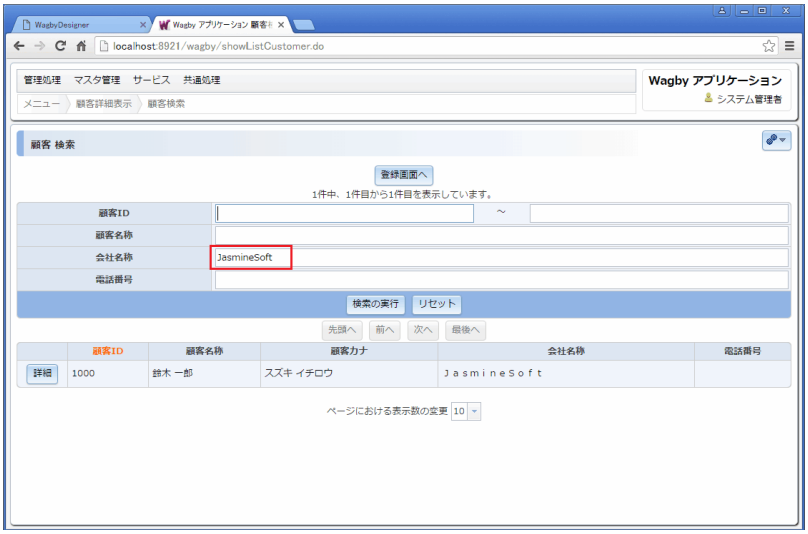


図 半角文字での検索

大文字小文字の違いがあってもヒットします。

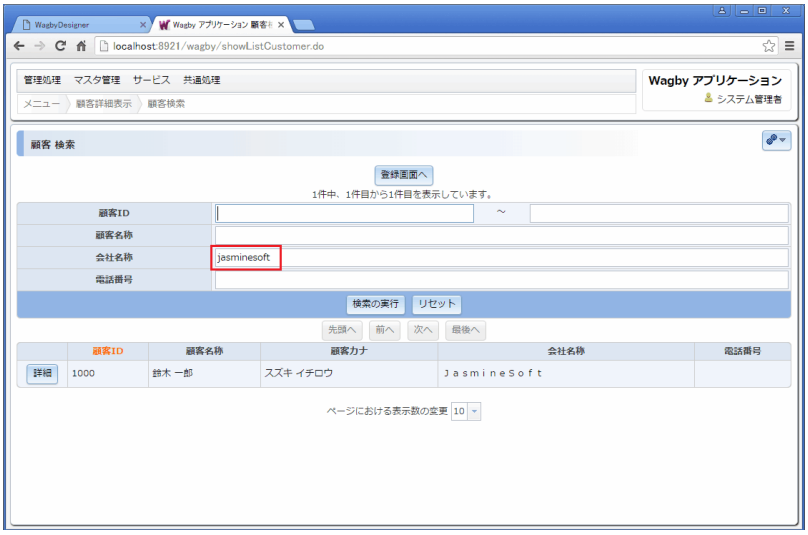


図 すべて小文字での検索

すべて大文字でもヒットします。

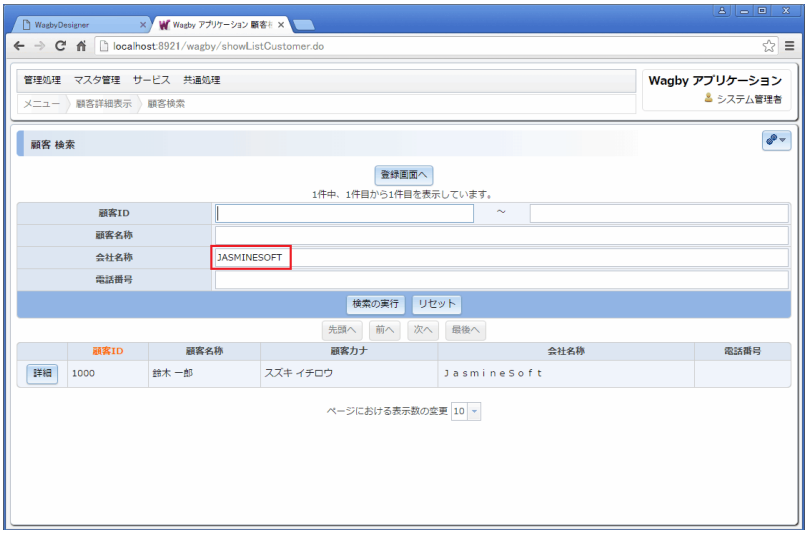


図 すべて大文字での検索

途中の空白も無視して検索します。

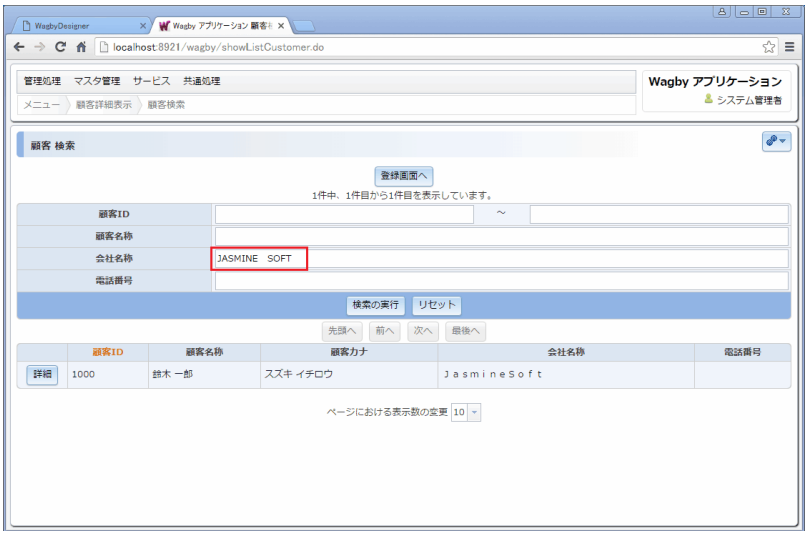


図 途中の空白も無視する

定義方法

定義のポイントは「会社名称」を2項目用意することです。「companyname」は通常の入力及び一覧表示用です。もう一方の「companyname2」は入力画面では隠し項目にし、検索条件用として使用します。

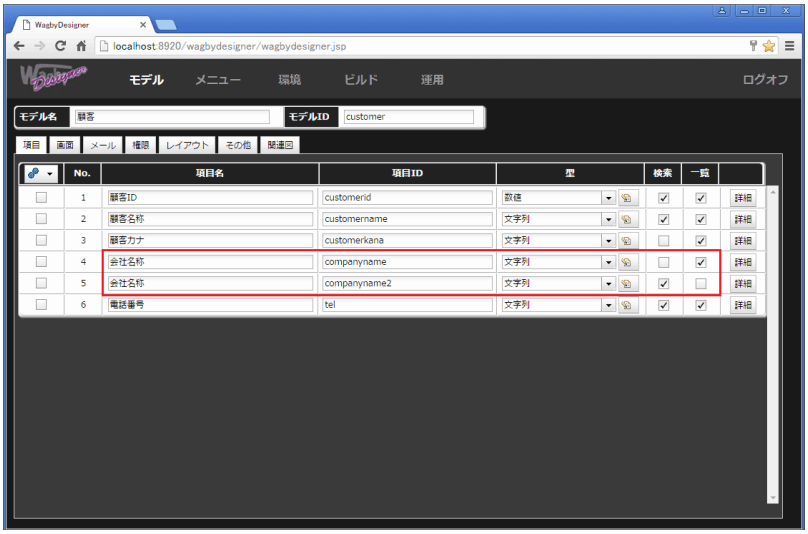


図 モデル項目一覧

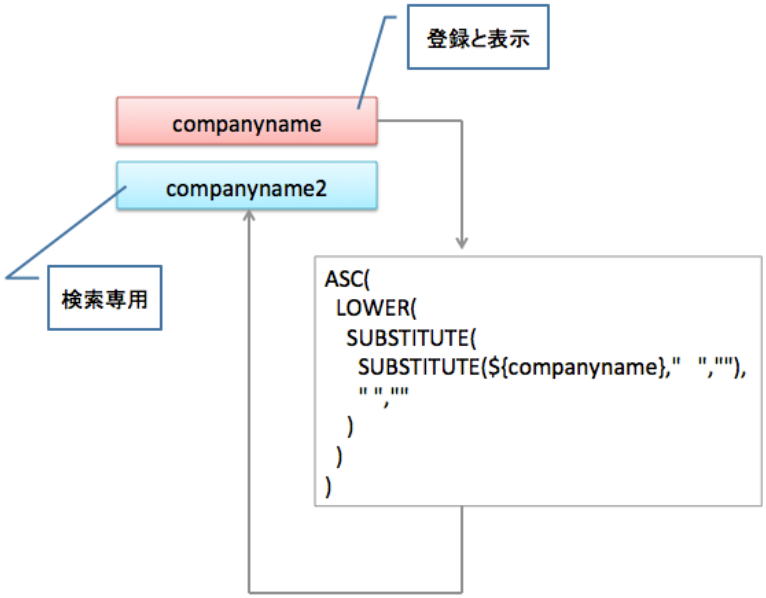


図 companyname項目とcompanyname2項目の関係性

入力フィルタの設定

「companyname」「companyname2」とも、入力フィルタは「文字列（最小限）」としています。これによって全角で入力された文字列はそのままデータベースに格納されます。

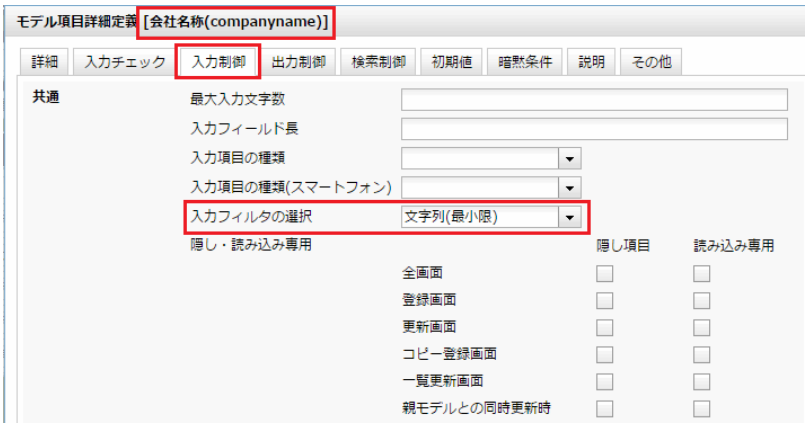


図 入力フィルタを文字列（最小限）にする(1)

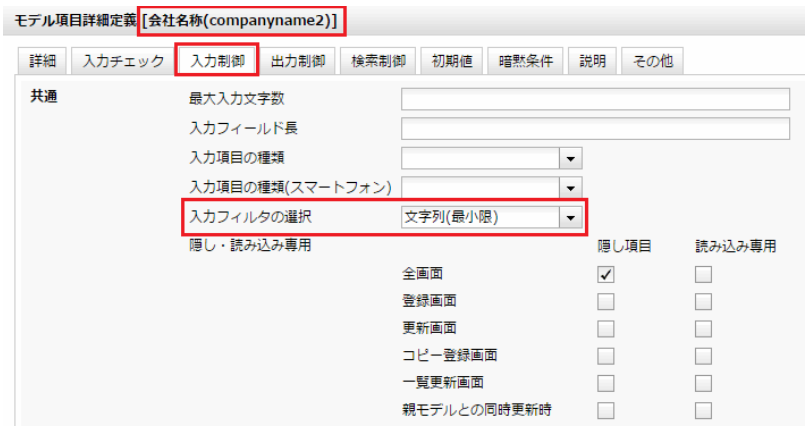


図 入力フィルタを文字列（最小限）にする(2)

以降の設定は、すべて「companyname2」項目に対して行います。

計算式とデータベース保存の設定

次のように式を設定します。

```
ASC(
  LOWER(
    SUBSTITUTE(
      SUBSTITUTE(${companyname},"　",""),
      " "," "
    )
  )
)
```

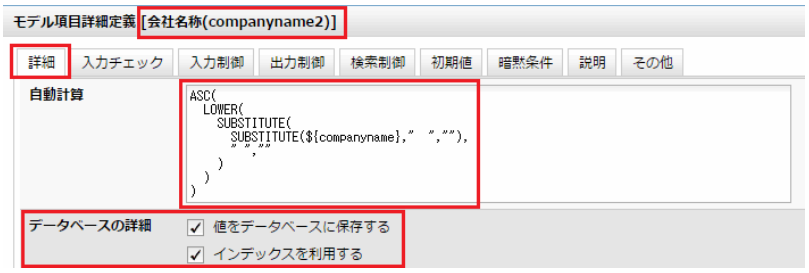


図 計算式の設定

ASC 関数は全角文字を半角文字に変換します。 LOWER 関数はアルファベットを小文字に変換します。 最初の SUBSTITUTE 関数は半角空白を空文字に置換します。もう一つの SUBSTITUTE 関数は全角空白を空文字に置換します。

また、この companyname2 項目で検索を行うため、データベースに保存します。

隠し項目にする

入力制御で、隠し項目とします。

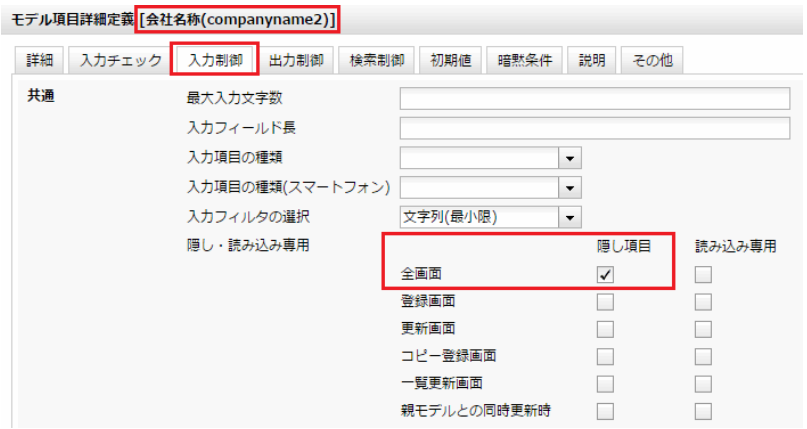


図 入力時に隠し項目とする

同じく出力制御でも、隠し項目とします。

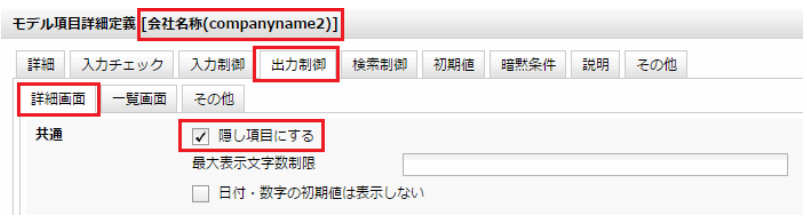


図 出力時に隠し項目とする

CSV出力を行わないようにします。この設定で、CSVダウンロード機能を用いた場合でも、この項目は含まれないようになります。

図 CSV出力を行わない

検索条件の設定

検索の初期値として、検索画面で入力された文字（検索条件文字列）を正規化します。データベースに格納された文字列も同じように正規化されているため、必ずヒットするようになります。

```
ASC(
  LOWER(
    SUBSTITUTE(
      SUBSTITUTE(${companyname2},"　",""),
      " ",""
    )
  )
)
```

図 検索時の「初期値」の設定

通常の検索画面とサブウインドウ検索画面で検索条件の表示・非表示を個別に指定する

検索条件の隠し項目設定

設定

「顧客」モデルに、「会社1」「会社2」「会社3」という3つの項目を用意してみます。いずれも検索条件として用いますが、次の性質があります。

・　「会社1」は常に非表示。

・　「会社2」はメイン画面（通常検索画面）で非表示。
・　「会社3」はサブウィンドウ検索画面で非表示。

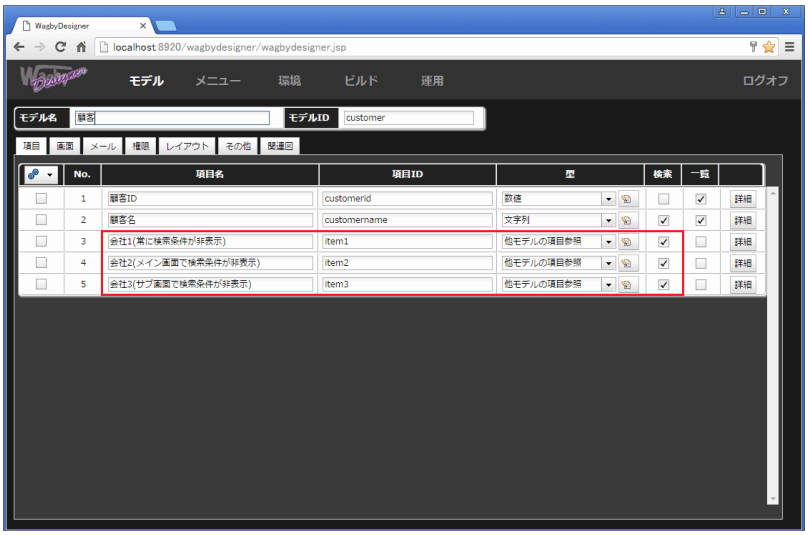


図 顧客モデル

「会社1」項目

「会社1」項目の検索制御の設定を図に示します。

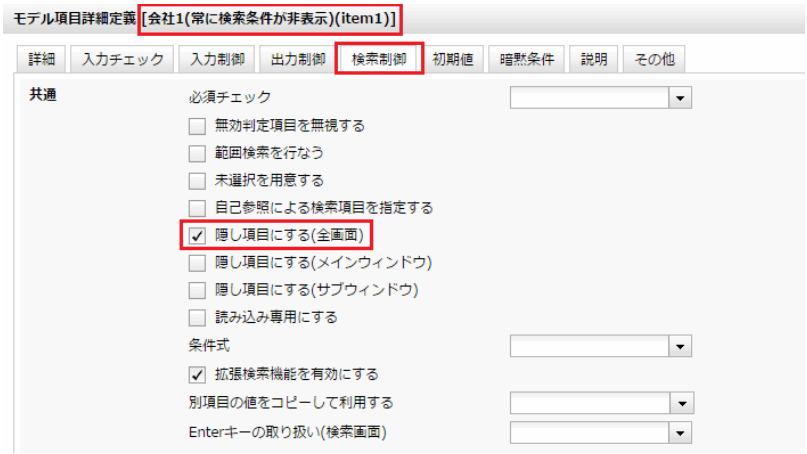


図 会社1項目の設定

「会社2」項目

「会社2」項目の検索制御の設定を図に示します。

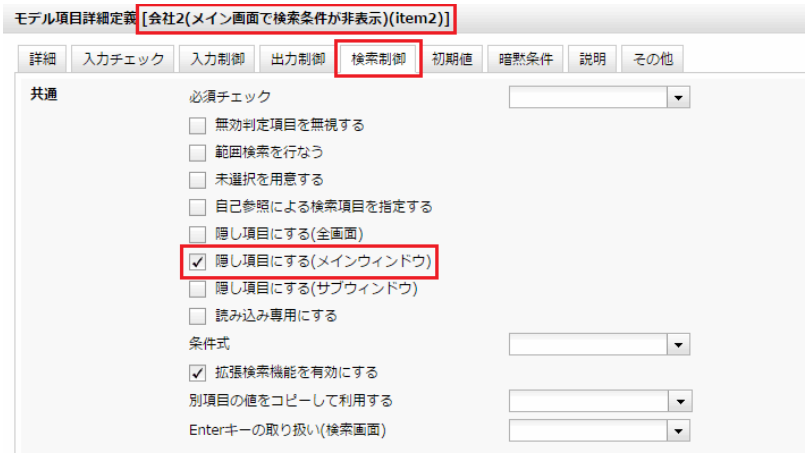


図 会社2項目の設定

「会社3」項目

「会社3」項目の検索制御の設定を図に示します。

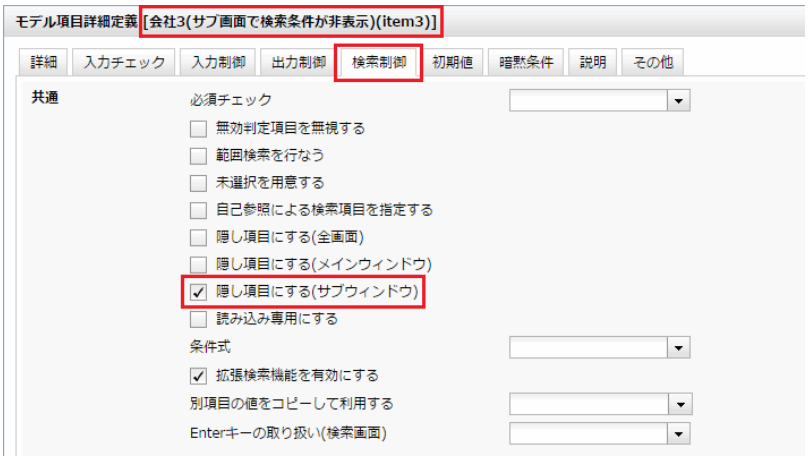


図 会社3項目の設定

例

通常の検索画面（メインウィンドウ）を図に示します。「会社1」項目と「会社2」項目がそれぞれ非表示になっていることがわかります。

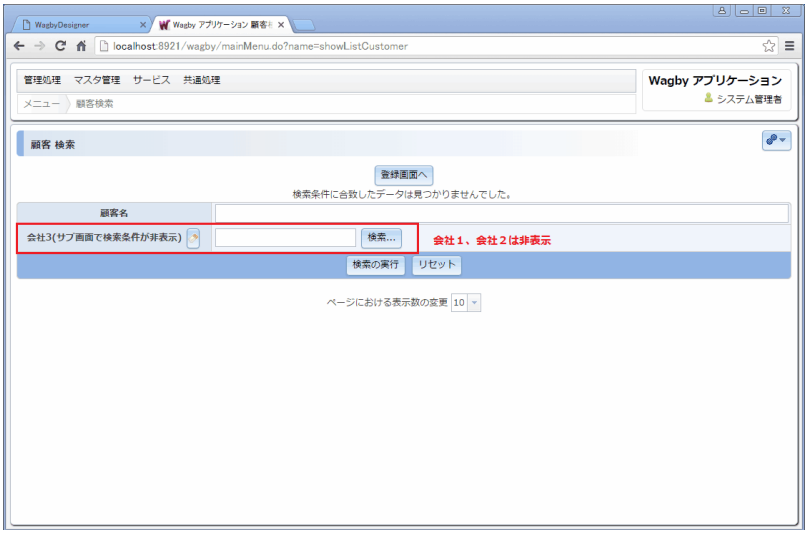


図 通常の検索画面

この顧客モデルを参照する「営業日報」モデルを用意してみました。モデル参照（検索）で、サブウィンドウを開きます。サブウィンドウでは「会社1」項目と「会社3」項目がそれぞれ非表示になっていることがわかります。

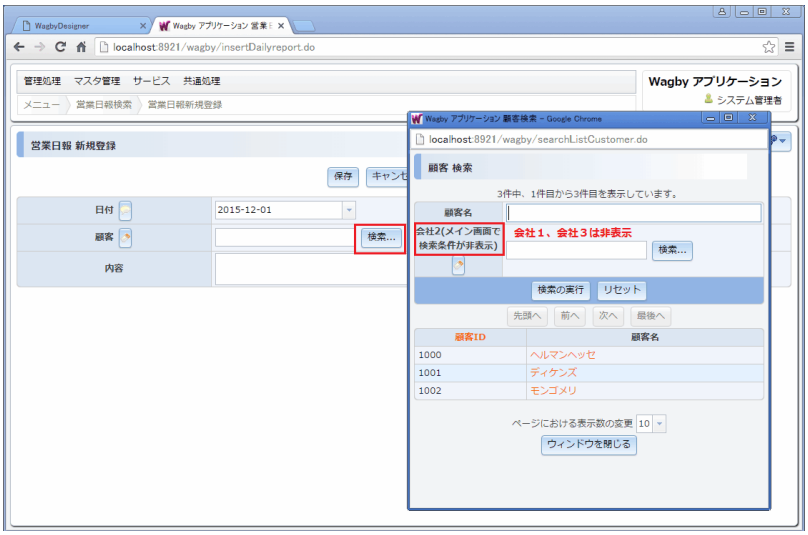


図 顧客モデルを参照する営業日報モデル

絞り込み設定との併用

例

通常の検索画面では自由に検索条件を変更できるが、サブウインドウ検索の場合はメインウインドウから指定された初期値を隠し項目にセットして検索範囲を制限する例を紹介します。

（顧客モデルを参照する）営業日報モデルで、「会社1へ」という項目を用意しました。この項目は顧客モデルをモデル参照するのですが、そのとき「会社」項目で指定された値で絞込むように設定されています。

図 あらかじめ用意された顧客リスト

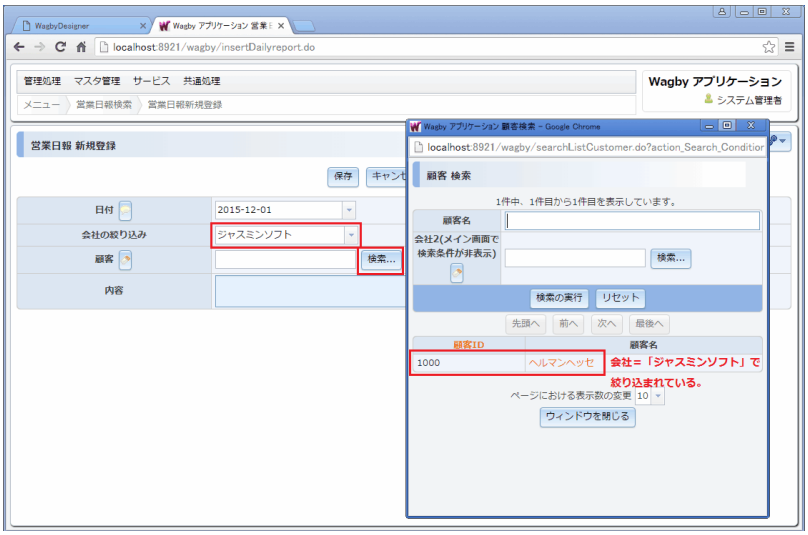


図 営業日報モデルのサブウィンドウ検索への絞り込みの適用

設定

営業日報モデルの定義を図に示します。

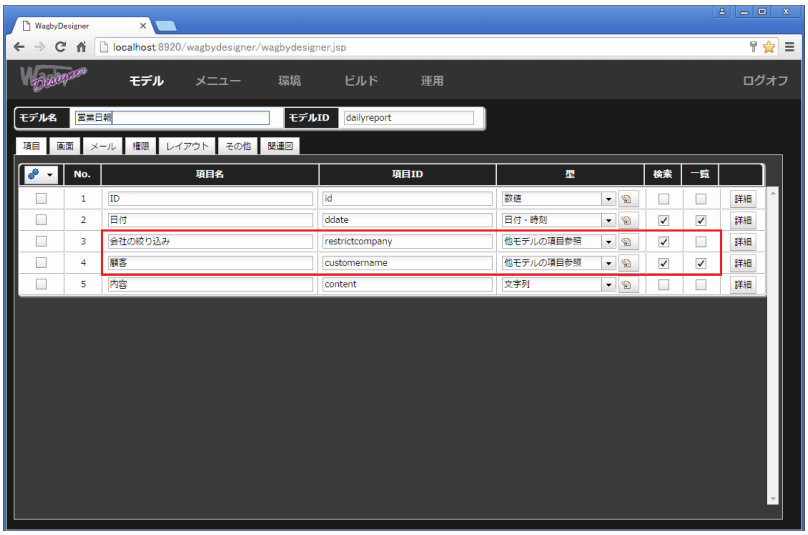


図 営業日報モデル

「会社の絞り込み」項目は、会社モデルを参照します。

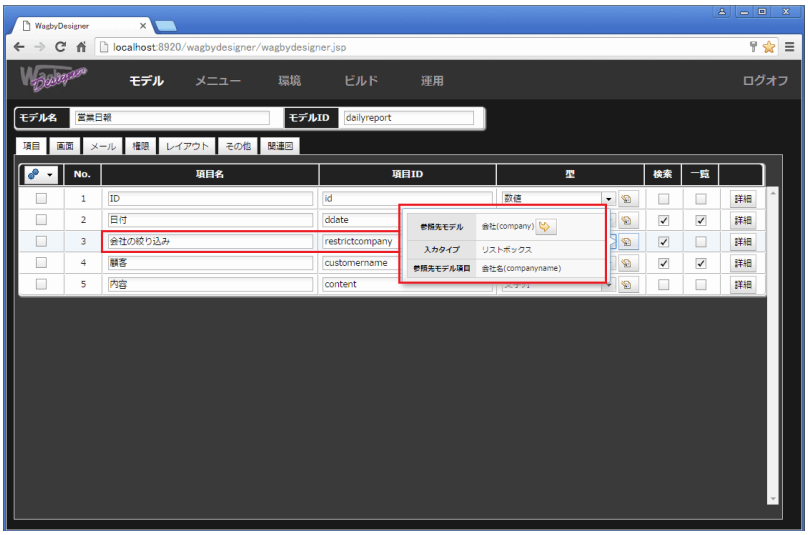


図 「会社の絞り込み」項目

「顧客」項目は、顧客モデルを参照します。

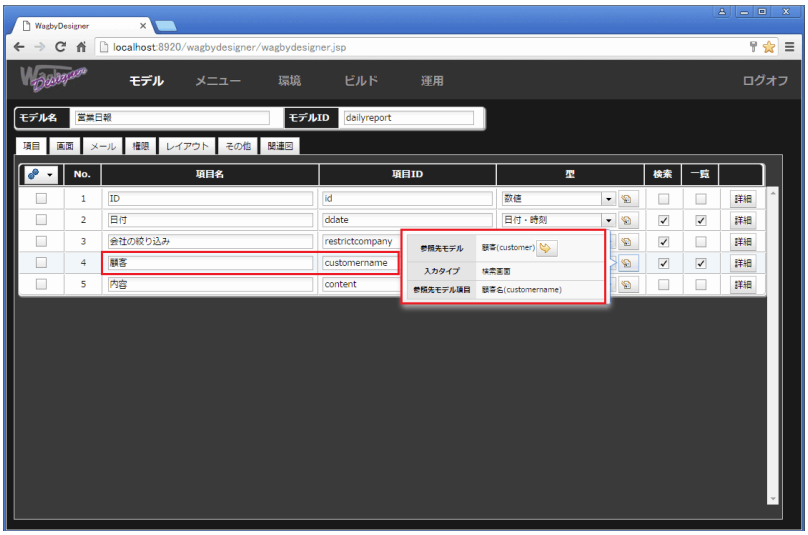


図 「顧客」項目

ここで「顧客」項目に図に示す絞り込み条件を設定します。

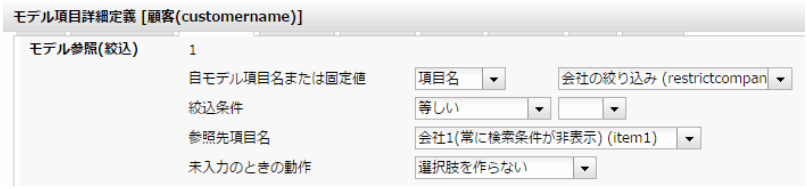


図 絞り込み条件の設定

検索条件テンプレート

例

「検索テンプレート」は、複数の検索条件の組み合わせに名前を付け、この名前によって検索条件を自動セットするという機能です。

図に例を示します。「検索テンプレート名称」を選択していない状態では、検索項目「業務名」「登録区分」はともに未選択です。

図 検索テンプレート名称を選択していない初期状態

「検索テンプレート名称」から任意の検索処理パターンを選択します。

図 検索テンプレート名称を選択する

図に示すように、検索条件が自動的にセットされます。

図 検索条件が自動的にセットされた

ここで「検索の実行」ボタンを押下すると、セットされた検索条件で絞り込みを行うことができます。

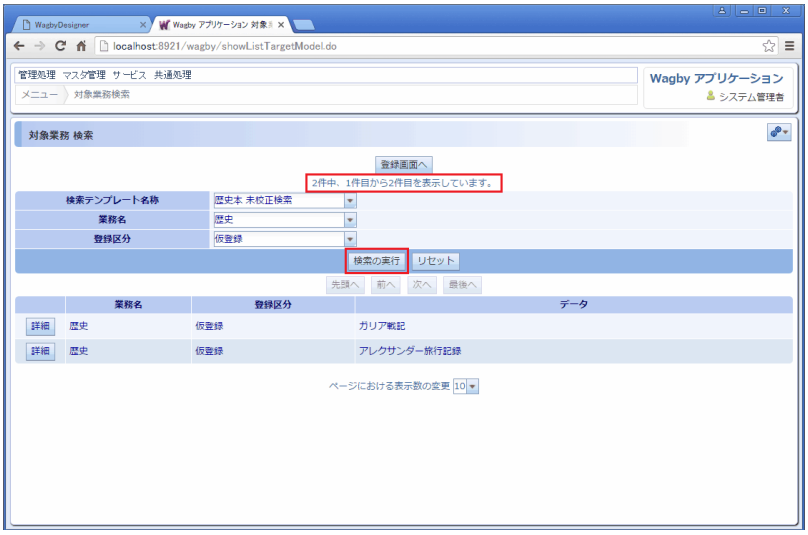


図 検索条件を用いる

定義方法

上の例を実行するために用意したモデルを図に示します。

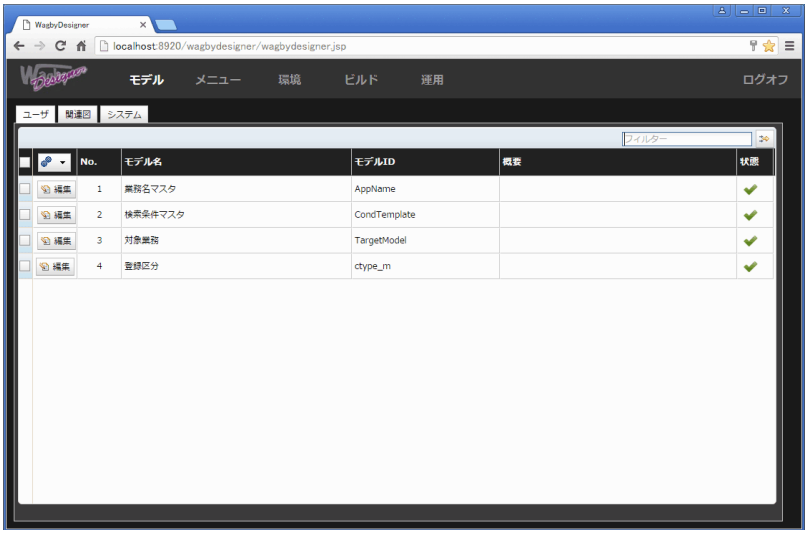


図 モデル一覧

「業務区分」モデル

このモデルおよび値（″仮登録″ ″本登録″）に意味はありません。サンプルとして扱っています。

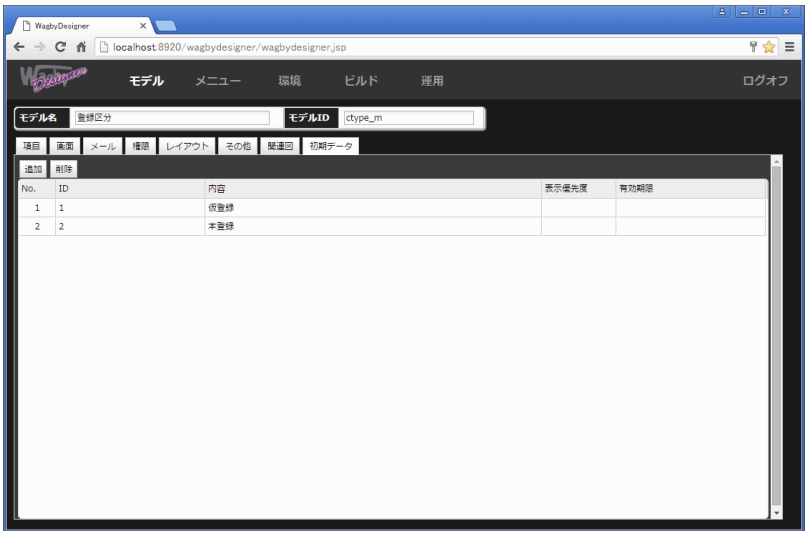


図 業務区分モデル

「業務名マスタ」モデル

このモデルにも意味はありません。サンプルデータとして図を用意しています。

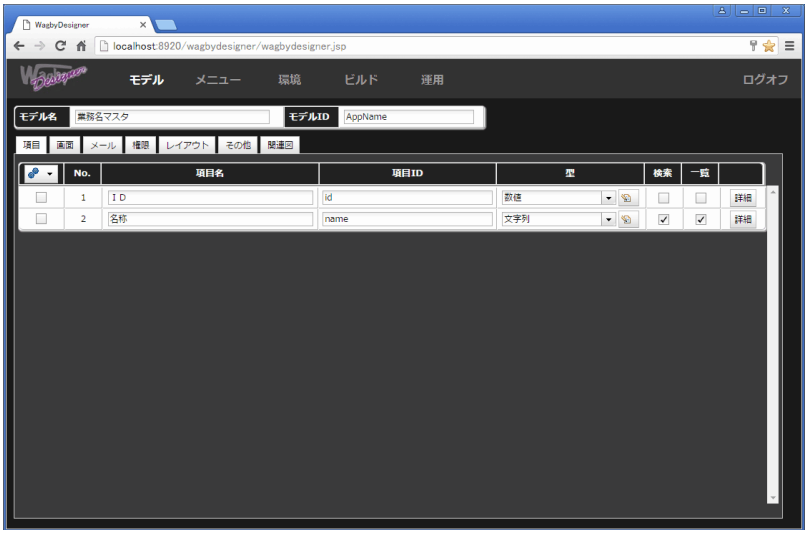


図 業務名マスタモデル

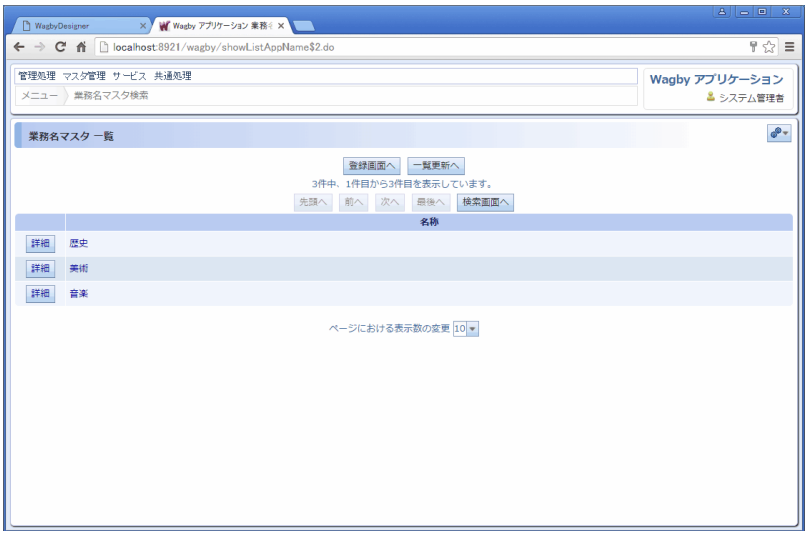


図 業務名マスタモデルのサンプルデータ

「検索条件マスタ」モデル

「検索条件テンプレート」を格納するためのモデルです。　このモデルに含まれる項目は、図, 図に示した「業務区分」と「業務名マスタ」モデルの値を保持させています。

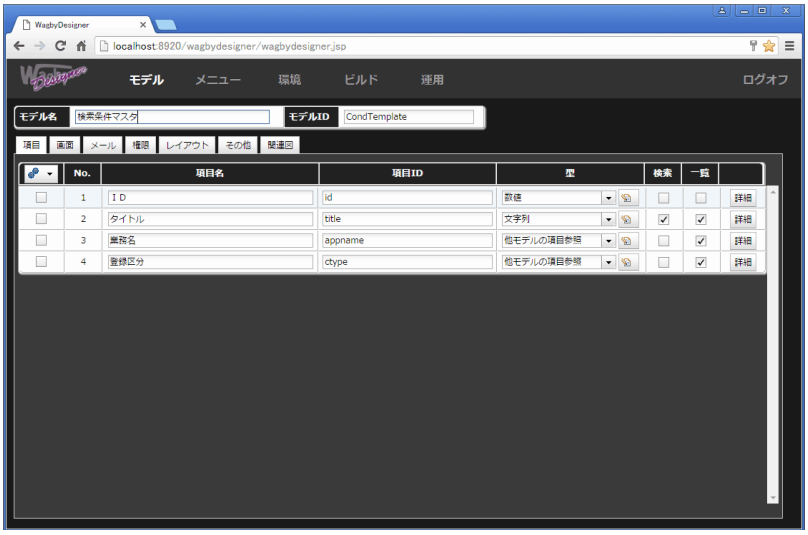


図 「検索条件テンプレート」モデルの定義

例えば「業務名」という項目は、図の「業務名マスタ」モデルを参照しています。

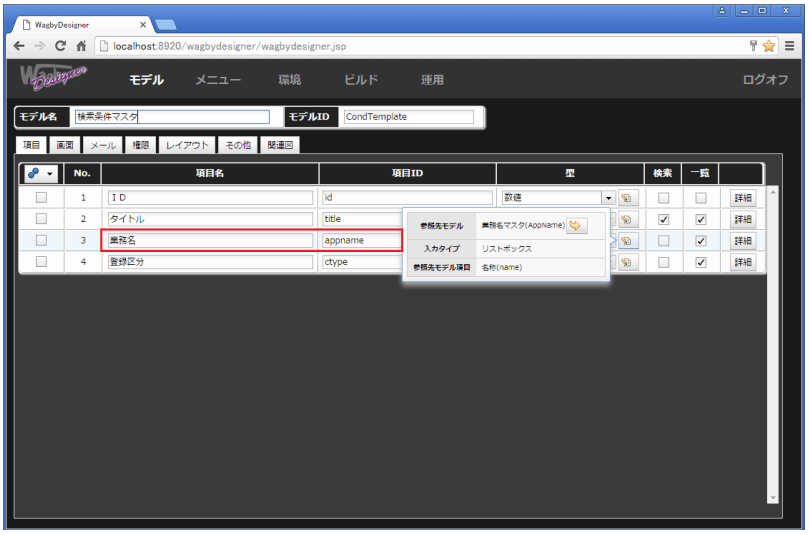


図 「業務名」項目の定義

今回のサンプルで用いた検索条件テンプレートの例を図に示します。「業務名」「業務区分」が実際の検索項目で、これに名前をつけています。

図 「検索条件テンプレート」の例

「対象業務」モデル

検索条件テンプレートを適用するモデルです。ポイントは、検索条件テンプレートモデルの名称を選択することで、検索条件を（参照連動で）引っ張ってくることです。これが図の赤枠部分になります。

ただし、この参照連動項目そのものは検索条件ではありません。実際の検索項目は図の青枠部分になります。

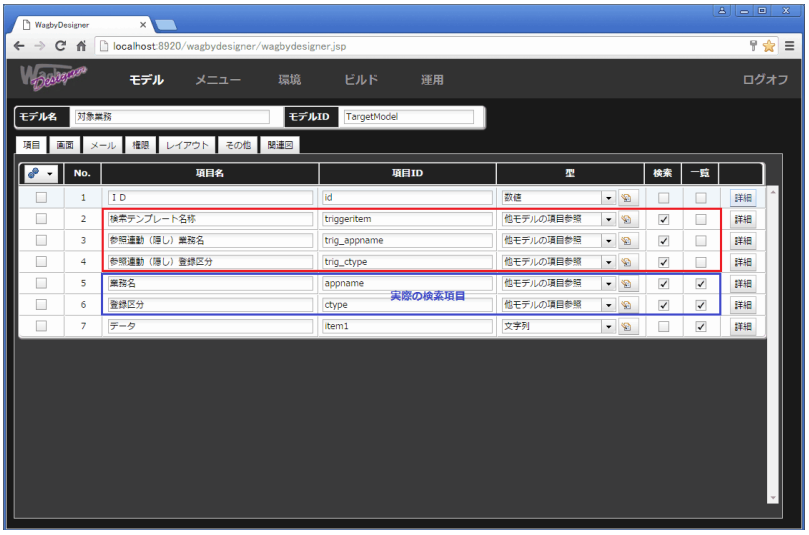


図 対象業務モデル

「検索テンプレート名称」項目は、図の「検索条件テンプレート」モデルを参照します。

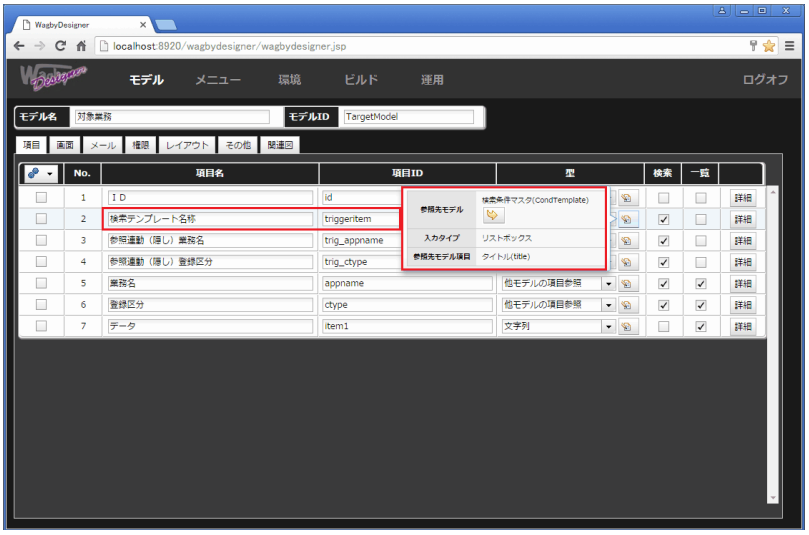


図 「検索テンプレート名称」項目の定義

「参照連動（隠し）業務名」「参照連動（隠し）業務区分」項目は、それぞれ検索テンプレートモデルの値を参照連動で取得しています。

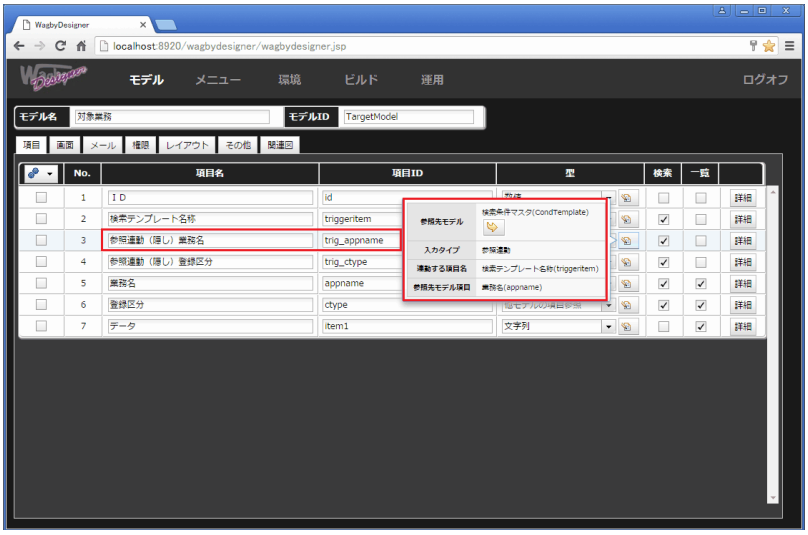


図　「参照連動（隠し）業務名」の関連

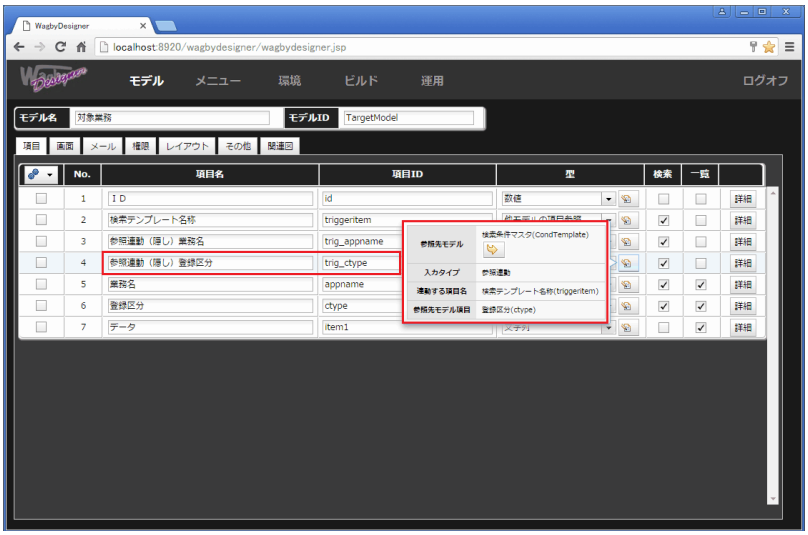


図　「参照連動（隠し）業務区分」の関連

「検索テンプレート名称」「参照連動（隠し）業務名」「参照連動（隠し）業務区分」（赤枠部分）

　図の赤枠部分となる「検索テンプレート名称」「参照連動（隠し）業務名」「参照連動（隠し）業務区分」は、検索条件に用いるという設定を行っていますが、データベースに保存しません。（これは検索画面で利用するという意図のためで、実際の検索条件には使いません。）

図　「検索テンプレート名称」はデータベースに保存しない

図　「参照連動（隠し）業務名」「参照連動（隠し）業務区分」もデータベースに保存しない

「参照連動（隠し）業務名」「参照連動（隠し）業務区分」は「入力制御-隠し項目にする(全画面)」を設定します。

図 「入力制御-隠し項目にする(全画面)」

同じく「参照連動（隠し）業務名」「参照連動（隠し）業務区分」は「出力制御(詳細画面)-隠し項目にする」を設定します。

図 「出力制御(詳細画面)-隠し項目にする」

同じく「参照連動（隠し）業務名」「参照連動（隠し）業務区分」は「検索制御-隠し項目にする(全画面)」および「検索制御-読み込み専用にする」を設定します。

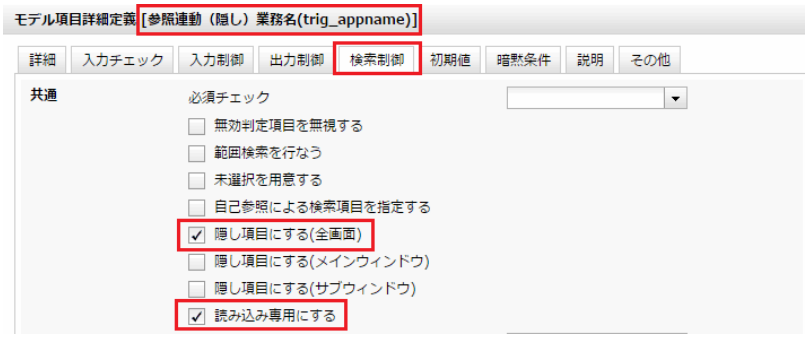


図 「検索制御-隠し項目にする(全画面)」および「検索制御-読み込み専用にする」

「業務名」「登録区分」（青枠部分）

図の青枠部分で定義した実際に検索時に用いる項目には、「検索制御-別項目の値をコピーして利用する」に、対応する参照連動項目名を指定します。これによって参照連動された値をコピーして、検索条件として利用することができます。

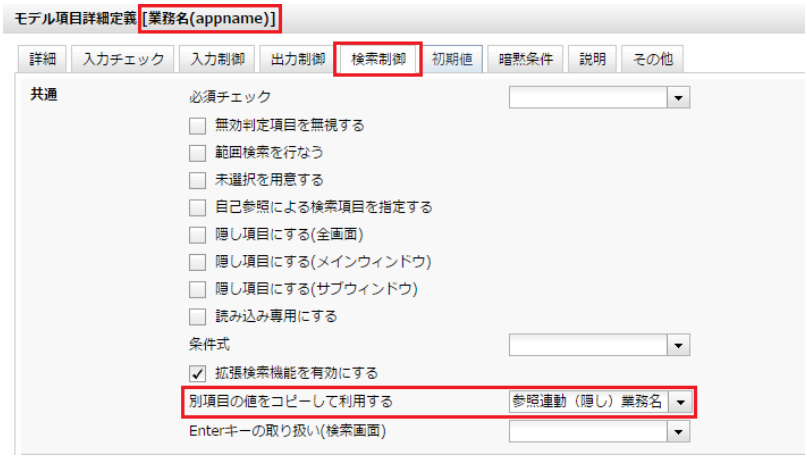


図 別項目の値をコピーして利用する(1)

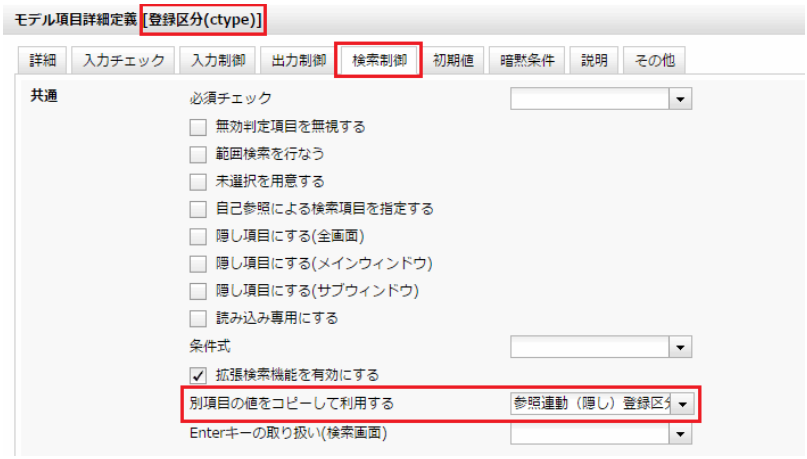


図 別項目の値をコピーして利用する(2)

値変更時に画面を再描画する

最後に、対象業務モデルの「検索テンプレート名称」項目について、「入力制御-値変更時に画面を再描画する」を有効ににします。これによって、テンプレート選択時に（画面には表示されませんが）検索値が参照連動され、その参照連動された値をコピーした状態で画面再描画が行われるようになります。

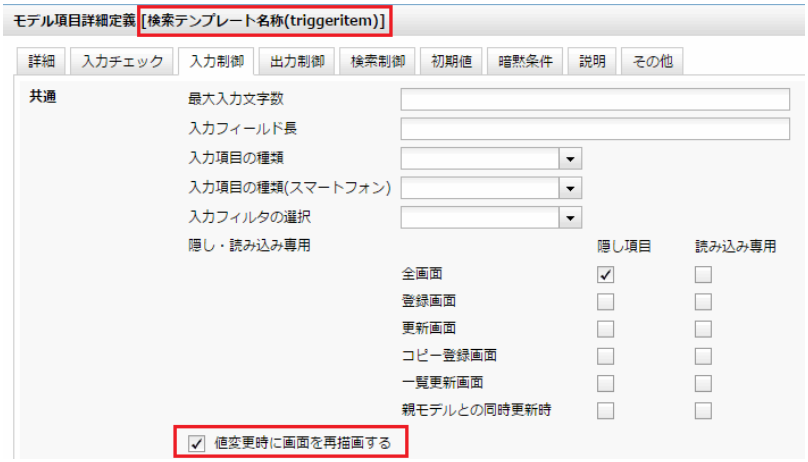


図 値変更時に画面を再描画する

スクリプトを利用して、テンプレートを選択するとすぐに検索を行うようにする

図の設定により、検索テンプレート名称を変更すると画面再描画されるようになります。しかし検索処理そのものは実行されていません。 ここではスクリプトを使って、再描画処理から検索処理を呼び出す方法を紹介します。(*1)

(167)

スクリプトを用意する

"コントローラ > 一覧 > 画面再描画" のタイミングで次のスクリプトを設定します。

```
return "redirect:showListTargetModel.do?action_Search";
```

"TargetModel" の部分はモデルIDに相当します。

"redirect:" の説明は "フォワードとリダイレクト" をお読みください。

外部から検索条件をパラメータとして指定する

基本形

画面の命名規則

Wagbyの検索画面は次の命名規則があります。

```
showList<ModelID>.do
```

「モデルID」部の表記はキャメル記法に従います。これは先頭を大文字とします。また、モデルIDにアンダースコア "_" が含まれていた場合、これを除去し、その次の文字を大文字とします。例えばモデルID が customer_order の場合は CustomerOrder になります。

(168)

例

モデル customer の検索画面を開く場合、次の URL となります。

```
http://localhost:8921/wagby/showListCustomer.do
```

(169)

メール文中に上記 URL を記載して送信すると、受信者はそのリンクをクリックすることで、当該画面を直接、開くことができます。（正確には最初にログオン画面が開きます。認証後、メニュー画面ではなく直接、上記で指定した画面が開きます。）

パラメータを指定する

モデル定義で検索条件を有効にした項目について、URLパラメータで指定することができます。

検索項目の命名規則

```
モデルID_cp$002f項目ID=値
```

ここで「モデルID_cp」と、接尾語 "_cp" が付与された表記は、検索条件を格納したことを意味します。

---

(167) 1. スクリプトを使うのがはじめての方は、スクリプトに関する説明をお読みください。

(168) 拡張子 .do は、過去の Wagby との互換性のためのルールです。将来の Wagby で廃止される可能性があります。

(169) "localhost:8921" および "wagby" の部分は、ご利用の環境にあわせて適切に読み替えてください。

また、$002f は "/" 文字を意味します。これはモデルIDと、項目IDの区切りとなります。

(170)

例1 値を設定する

モデルIDが customer の、検索条件 customerid に、値 1000 を指定する場合は次の表記となります。

```
http://localhost:8921/wagby/showListCustomer.do?customer_cp$002fcustomerid=1000
```

例2 日本語の値を設定する

日本語（を含む、英語以外の文字）をパラメータに渡す場合は、UTF-8　でエンコードした文字を渡します。 例えば customername が "やまだ" を検索する場合は、次のようになります。

```
http://localhost:8921/wagby/showListCustomer.do?customer_cp$002fcustomername=%e3%82%84%e3%81%be%e3%81%a0
```

UTF-8　でエンコードされた表記は、インターネット上に公開されているさまざまなサービスを用いて作成することができます。検索エンジンで "urlエンコード" で検索するとよいでしょう。

・　例 Web便利ツール/URLエンコード・デコードフォーム - TAG index（日本語）

自動生成されたファイルから、パラメータ名を確認する

パラメータ名は、自動生成された JSP ファイルで確認し、コピー&ペーストするとよいでしょう。

wagbyapp/webapps/プロジェクト名/モデルID/ フォルダの中に「condition + "モデルID" + Cp.jsp」というファイルがあります。これをテキストエディタで開くと、次のようなコードが見つかります。

```
<tags:input_text name="customer_cp$002fcustomername" ... />
```

この name="..." の部分が、パラメータ名になります。

範囲指定

数値型項目や日付型項目の場合、開始と終了を指定できます。それぞれ次のようになります。

```
モデルID_cp$002f項目ID1jshparam
モデルID_cp$002f項目ID2jshparam
```

項目IDに接尾語 "1jshparam" が付与されたものが開始になります。"2jshparam" が付与されたものが終了になります。

モデル参照

モデル参照項目の場合は、内容部ではなくID部を渡します。パラメータ名の命名規則は次のとおりです。

```
モデルID_cp$002f項目ID$005b1$005d$002f$0040id
```

これは内部では「モデルID_cp/項目ID[1]/@id」という表記として扱われます。

例えば、顧客区分（のID部）が "1" の顧客を検索する場合のパラメータは次のようになります。

```
http://localhost:8921/wagby/showListCustomer.do?customer_cp$002fkubun$005b1$005d$002f$0040id=1
```

(170) Javaによるカスタマイズを行う開発者は「コンディションモデルの、プレゼンテーション表記」と解釈してください。

複数のパラメータを指定する

　複数のパラメータは「&」で区切って表記します。この場合、内部では″かつ″（AND条件）として検索されます。

```
http://localhost:8921/wagby/showListCustomer.do?customer_cp$002fcustomername=%e3%82%84%e3%81%be%e3%81%a0&customer_cp$002fkubun$005b1$005d$002f$0040id=1
```

(171)

検索条件に名前をつけて再利用する

設定方法

　検索画面の設定で「検索設定を利用する」を有効にします。

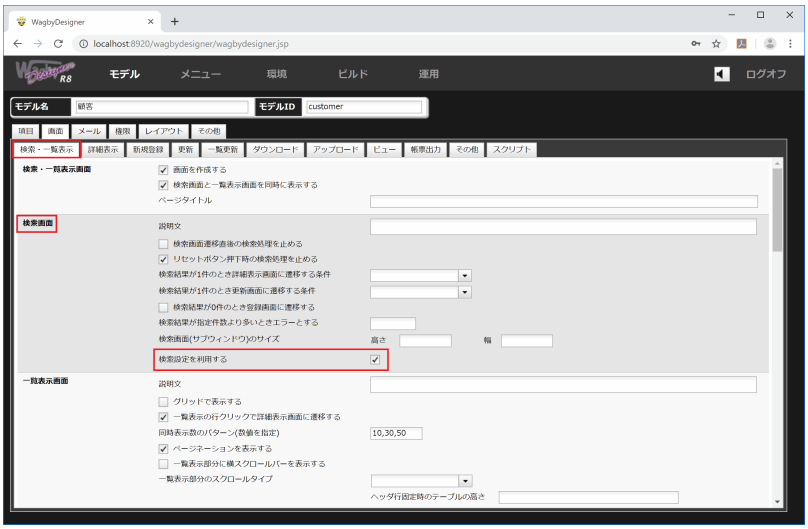


図 検索設定を利用する

ユーザ単位での利用

　この設定を有効にすると、検索条件入力欄の下に「検索設定」欄が用意されます。

(171) ″または″（OR条件）として扱う場合はスクリプトを用いたカスタマイズを行なってください。詳細...

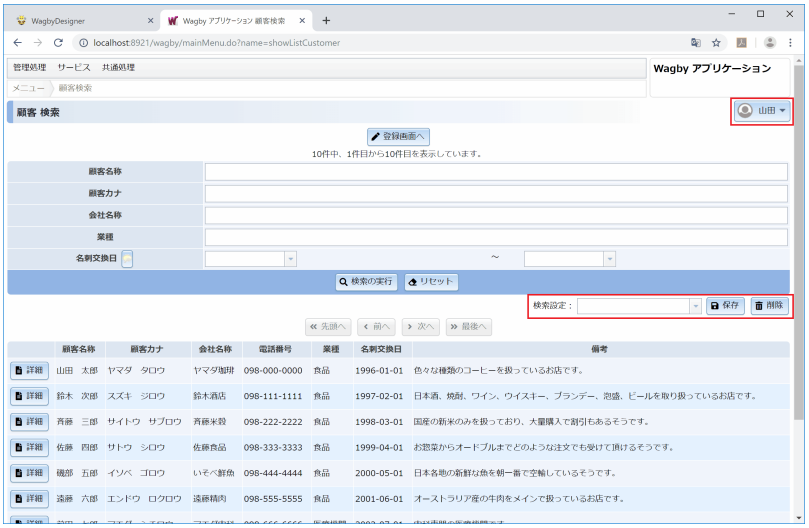


図 検索設定欄

ある検索条件で絞り込みます。

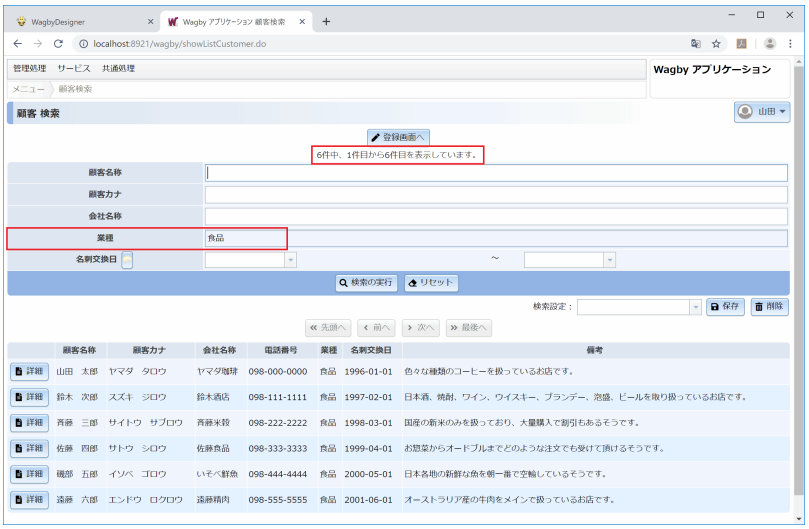


図 業種食品で検索絞り込みを行う

検索後、検索設定欄に任意の名前を入力し、保存ボタンを押します。現在の検索条件値をこの名前で記憶します。

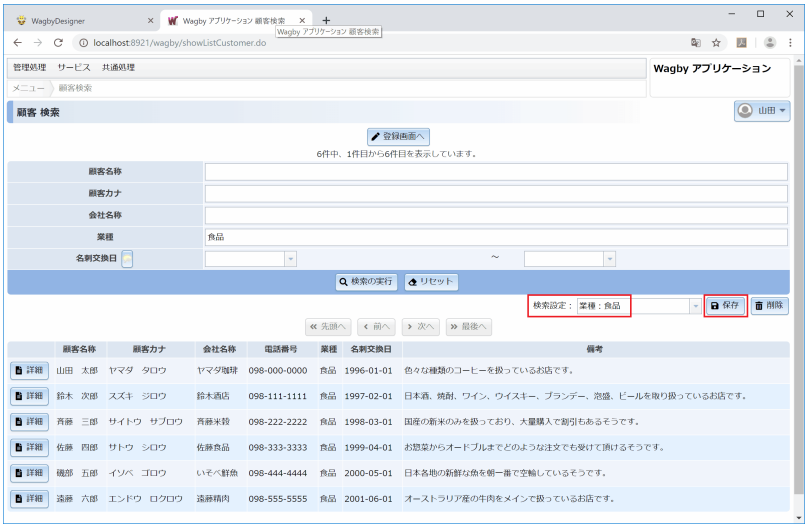


図 検索設定を保存する

動作確認のため、いったん検索条件をリセットします。

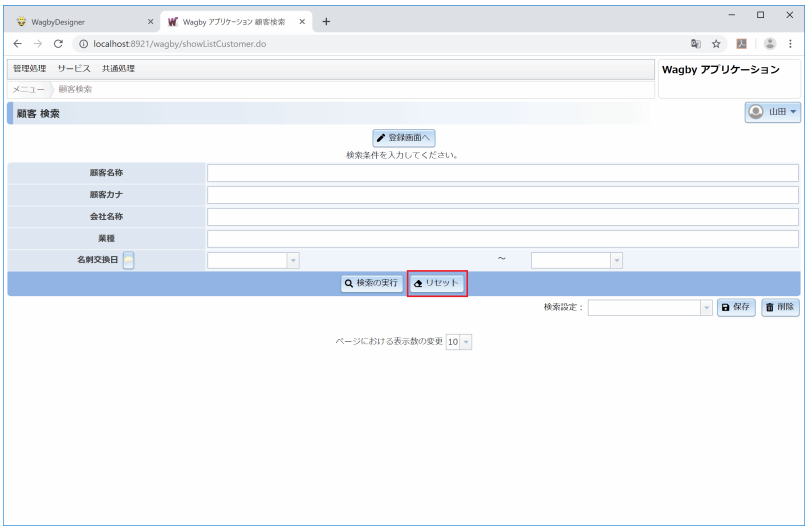


図 検索条件をリセットする

検索設定欄を確認します。先ほど保存した条件が選択できるようになっています。

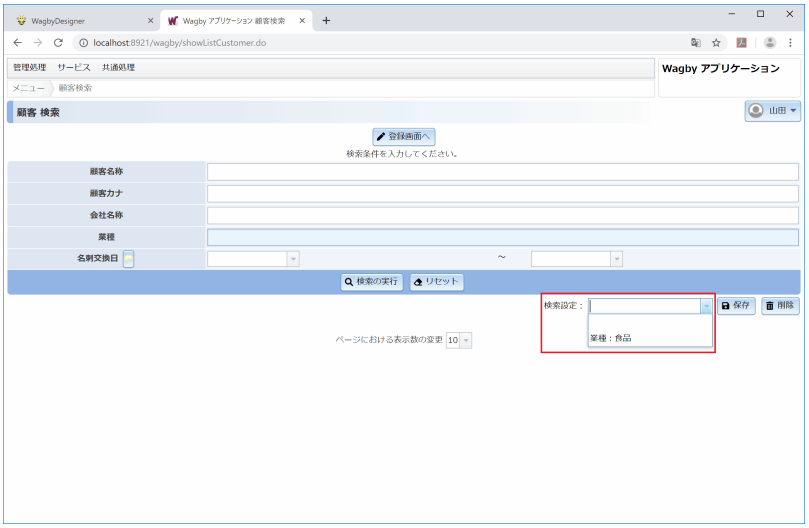


図 検索設定を選択する(1)

検索設定を選択すると、すぐに検索処理が実行されます。

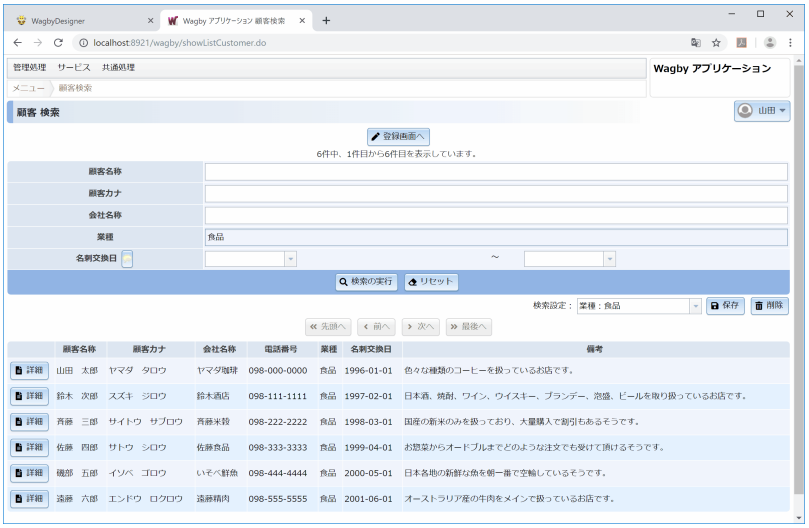


図 検索設定を選択する(2)

複数の検索設定を保存することができます。これは検索画面ごとに用意することができます。

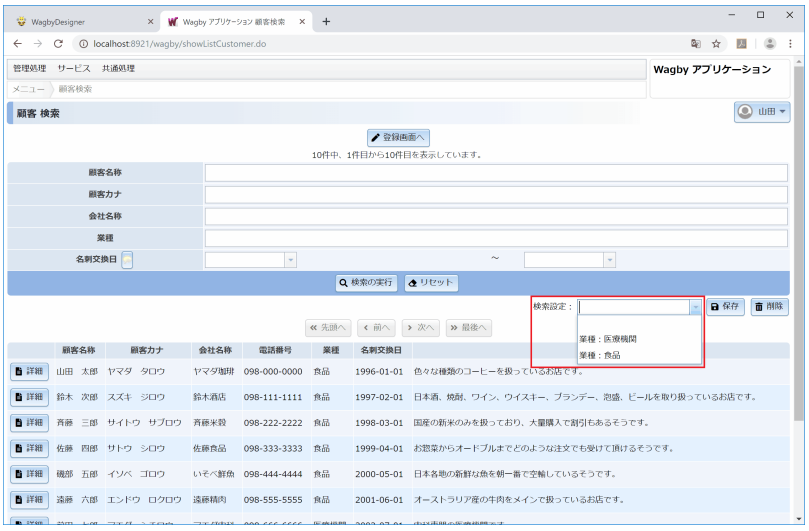


図 検索画面ごとに検索設定を保存できる

ここで保存した検索設定は、自分だけが利用できます。別ユーザでログオンしたときの検索設定には反映されません。

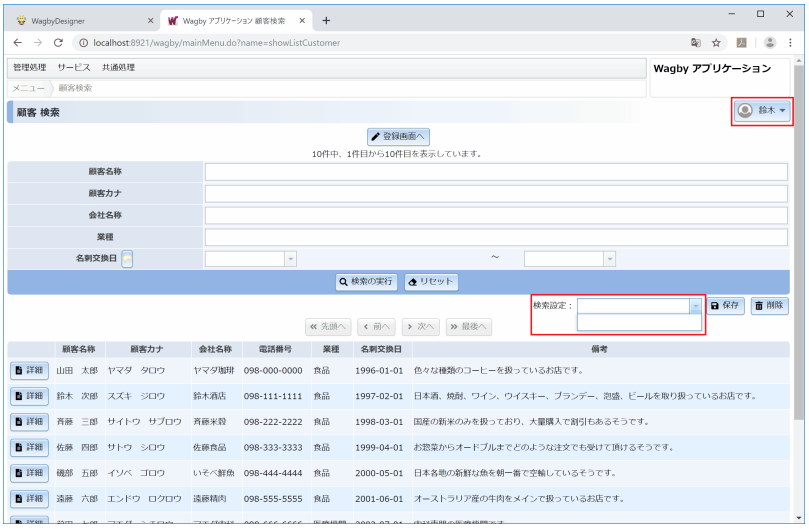


図 検索設定はユーザごとに異なる

検索条件を共通利用する

検索設定を共通に管理できる権限をもったアカウントは、共通設定を行うことができます。ここではシステム管理者を例に説明します。

共通設定は、検索設定名の先頭に ″*″（アスタリスク）を付与します。

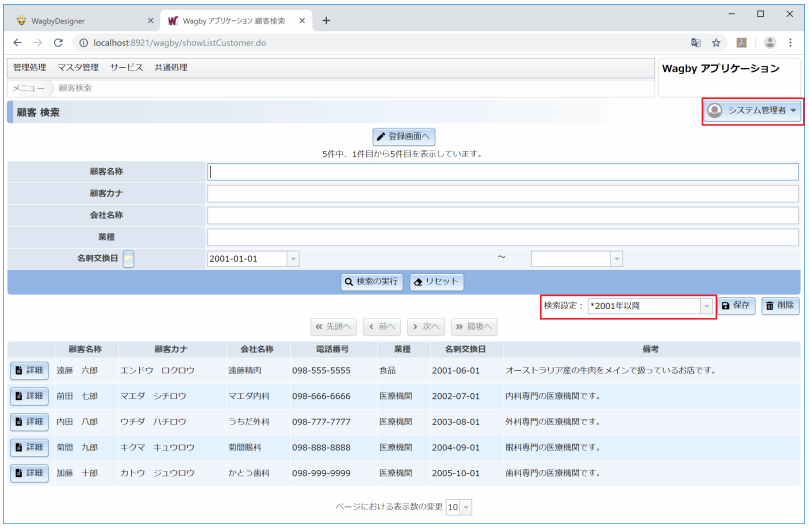


図 共通で利用できる検索設定

共通設定は、すべての利用者で再利用することができます。

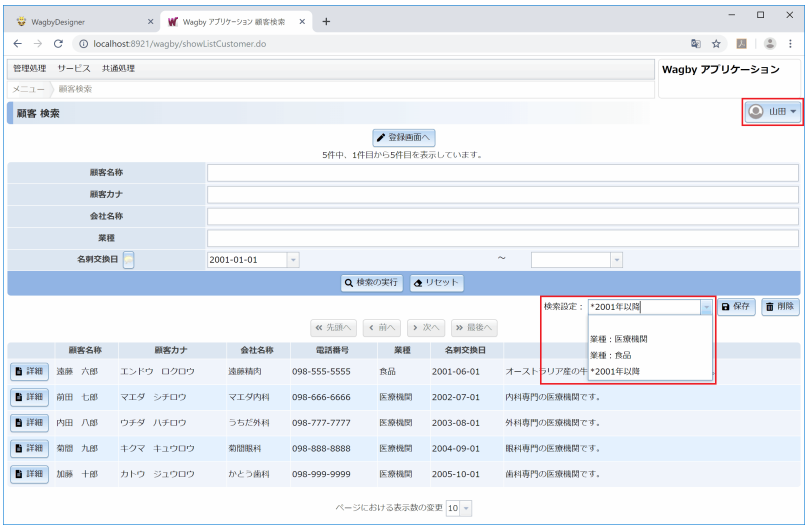


図 共通設定を利用する

共通検索設定管理プリンシパル

システム管理者以外で、共通検索設定管理プリンシパルをもったアカウントは、共通設定として保存することができるようになります。[プリンシパルの詳細...]

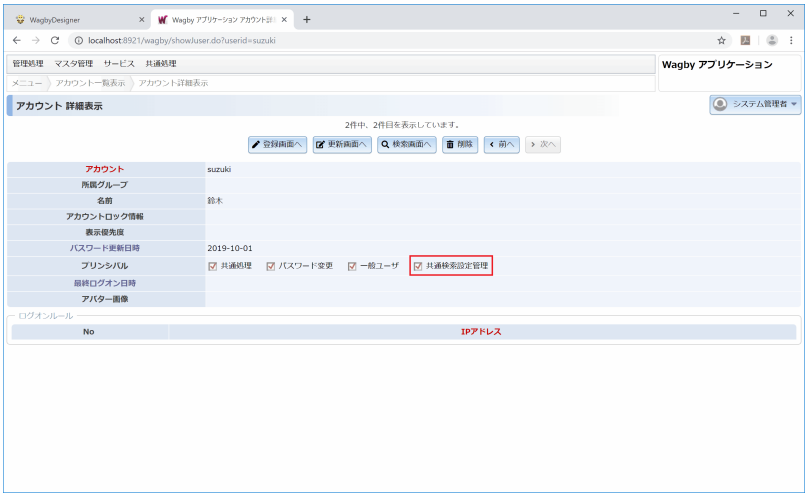


図 共通検索設定管理プリンシパルを付与されたアカウント

削除

検索設定を選択した状態で、「削除」ボタンを押下すると、その検索設定を削除することができます。

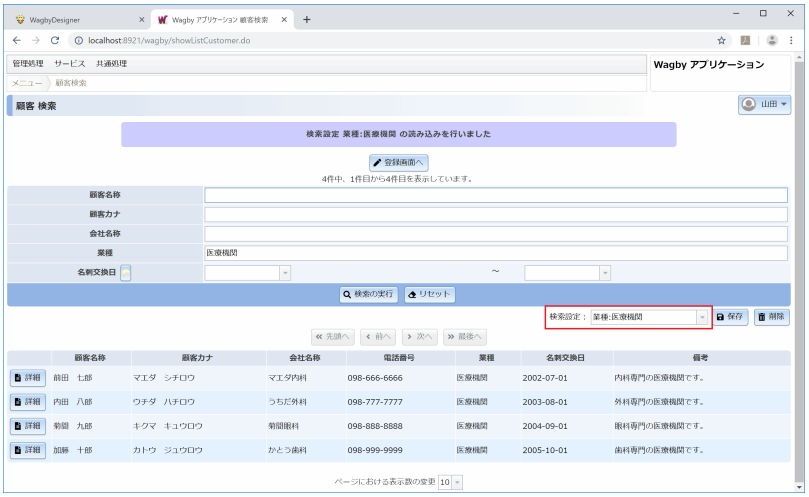


図 検索設定を削除する(1)

削除ボタン押下後、メッセージが表示されます。選択肢からは除かれています。

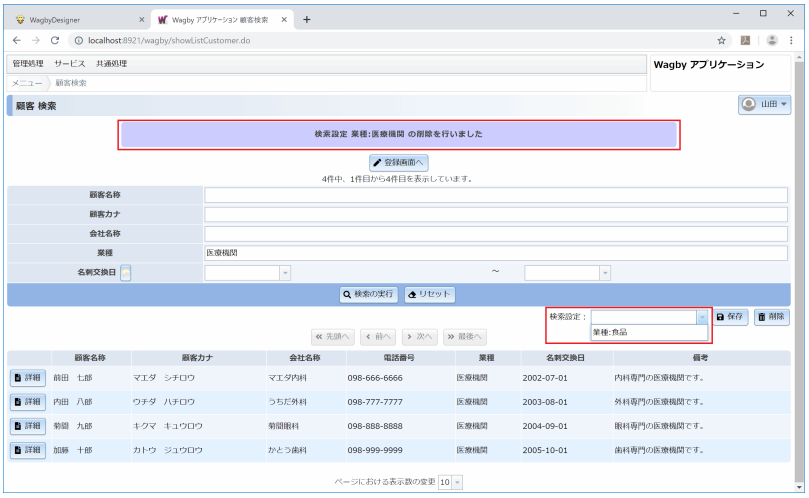


図 検索設定を削除する(2)

仕様・制約

- 個人別検索設定はシステムテーブルである jfcpreferenceitem テーブルに保存されます。キー名の先頭部分に "jfcuserconditionsetting_" が含まれます。
- 共通検索設定はシステムテーブルである jfccspreferenceitem テーブルに保存されます。キー名の先頭部分に "jfcconditionsetting_" が含まれます。
- アカウントモデル（juser）のインポート・エクスポート時に個人別検索設定（jfcpreferenceitem）も処理されます。
- 共通検索設定（jfccspreferenceitem）は明示的にインポート・エクスポートを指定してください。
- 本機能はスマートフォン用画面には対応していません。
- 本機能はカレンダービュー画面の検索条件部には対応しません。

クライテリアを直接操作して、検索条件をカスタマイズする(スクリプト)

クライテリア

「クライテリア」は、Wagby内部で利用している、検索条件を管理するクラスです。SQL に相当する表現能力があり、利用するデータベースに依存しないことや、コンパイル時に型をチェックしエラーを検出できるといったメリットがあります。

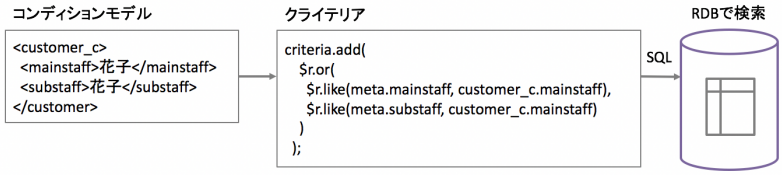


図 クライテリアの位置づけ

図にあるように、ユーザが入力した「検索条件」は、Webフォームから「コンディションモデル」という形で Wagby に取り込まれます。コンディションモデルはさらに「クライテリア」に変換されます。このクライテリアが SQL に変換され、データベースへの問い合わせを行います。

(172)

クライテリアをカスタマイズするメリット

Wagby の標準では、複数項目の検索条件はすべて AND 結合となります。例えば、顧客モデル(customer) の mainstaff 項目が "花子" または substaff 項目が "花子" という検索は、標準機能では実現できません。（標準機能では、mainstaff が "花子" かつ substaff も "花子" という AND 結合になります。）

クライテリアをカスタマイズすることで、複数項目の OR 検索を実現できます。

例 OR検索

要件

検索フォームで指定された担当者項目の検索条件値を主担当者、副担当者との OR 検索とする例を考えます。

(172) クライテリアは Wagby が同梱している O/R Mapper ミドルウェア Hibernate が提供するものです。クライテリアから SQL への変換は、Hibernate が行います。

OR

customerid	name	mainstaff	substaff	...
1000	鈴木太郎	ジャスミン四郎	ジャスミン五郎	
1001	佐藤次郎	ジャスミン花子	ジャスミン六郎	
1002	渡辺三郎	ジャスミン七郎	ジャスミン花子	

図 要件イメージ

具体的には、顧客モデル(customer) の mainstaff 項目が "ジャスミン花子" または substaff 項目が "ジャスミン花子" という検索を実現できるようにします。

用意するスクリプト

「画面 > スクリプト > ヘルパ > 検索」を用意します。

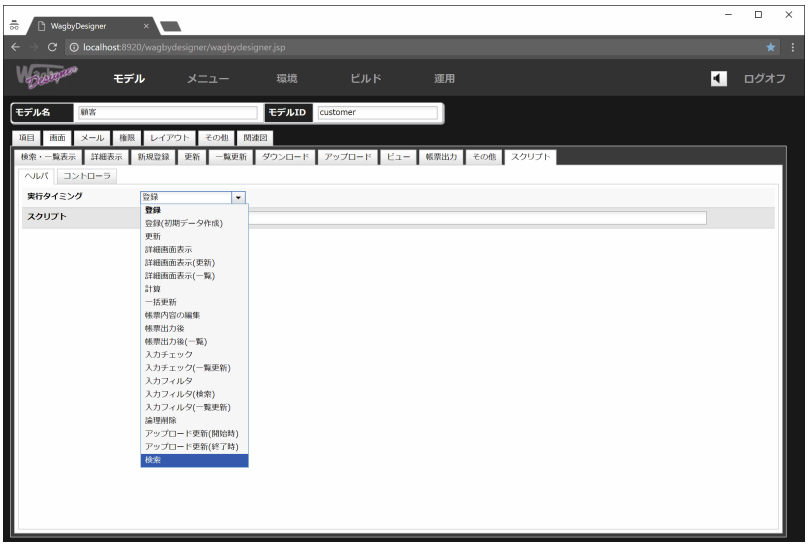


図 スクリプトの作成

(173)

```
// customerモデルを定義すると CustomerMeta クラスは自動生成されます。
var meta = new Packages.jp.jasminesoft.wagby.model.customer.CustomerMeta();

// スクリプトでは criteria.like() 等を利用すると
// Can't unambiguously select between fixed arity signatures
// エラーが発生することがあるため、$r を使って検索条件を組み立てます。
// criteria のメソッドは add() のみを使って下さい。

var $r = criteria.restrictions();
// "mainstaff" like '%ジャスミン花子%' OR "substaff" like '%ジャスミン花子%'
criteria.add(
  $r.or(
    $r.like(meta.mainstaff, customer_c.mainstaff),
    $r.like(meta.substaff, customer_c.mainstaff)
  )
```

(173) WEB-INF/script/<モデルID> フォルダに <モデルID>CriteriaConverter_convert.js が生成されます。

```
);

customer_c.mainstaff = null; // 元の検索条件は消去しておく。(*1)
```

- スクリプト内で暗黙オブジェクト criteria を利用できます。最初に criteria.restrictions() の返戻値を $r で受けるようにします。この $r を使って検索条件を組み立てます。
- 暗黙オブジェクト criteria に対しては add メソッドのみが利用できます。
- criteria を用いた検索条件の指定では、対象項目名は文字列ではなく、Wagbyが提供するメタクラスを使います。例えばcustomerモデルを定義するとメタクラス CustomerMeta が自動生成されます。これは項目名と型の情報（メタデータ）を保持する特別なクラスです。メタクラスを使うことでスペルミスによる実行時エラーといった、デバッグしにくい問題を回避することができます。

元の検索条件を消去する

上記スクリプトで (*1) の処理を行わないと、次の SQL が実行されてしまいます。

```
WHERE
    mainstaff LIKE  '%ジャスミン花子%'
AND (
       mainstaff LIKE  '%ジャスミン花子%'
    OR substaff LIKE  '%ジャスミン花子%'
)
AND...
```

WHERE 句の最初の mainstaff LIKE... は自動生成コードが付与したものですが、この要件では不要です。そのため、元の検索条件を null で上書きして、この SQL が生成されないようにします。コンディションモデルからクライテリアに変換されたとき、内部では criteria.like(meta.mainstaff, null) というコードと解釈されますが、第二引数が null の criteria は無視されるため期待どおりの動作となります。

クライテリアの詳細

クライテリアの定義方法について、具体的なコード例で説明します。

```
// criteria.add() を使うと AND で検索条件が追加されます。
// WHERE "mainstaff" LIKE '%花子%' AND "substaff" LIKE '%花子%'
criteria.add($r.like(meta.mainstaff, "花子"));
criteria.add($r.like(meta.substaff, "花子"));

// $r.and() でも同様に AND で検索条件が追加されます。
// 次の記述方法でも同じ SQL が生成されます。
// WHERE "mainstaff" LIKE '%花子%' AND "substaff" LIKE '%花子%'
criteria.add(
    $r.and(
      $r.like(meta.mainstaff, "花子"),
      $r.like(meta.substaff, "花子")
    )
);

// $r.like(), $r.and() の戻り値を変数に格納することも可能です。
// 次の記述方法でも同じ SQL が生成されます。
// WHERE "mainstaff" LIKE '%花子%' AND "substaff" LIKE '%花子%'
var likeCriterion01 = $r.like(meta.mainstaff, "花子");
var likeCriterion02 = $r.like(meta.substaff, "花子");
var andCriterion = $r.and(likeCriterion01, likeCriterion02);
criteria.add(andCriterion);

// $r.or() を使うことで OR 検索を行うことができます。
// WHERE "mainstaff" LIKE '%花子%' OR "substaff" LIKE '%花子%'
criteria.add(
    $r.or(
      $r.like(meta.mainstaff, "花子"),
      $r.like(meta.substaff, "花子")
    )
```

```
  );

  // 引数が null の場合はその条件は無視されます。
  // WHERE "substaff" LIKE '%花子%'
  criteria.add(
      $r.or(
        $r.like(meta.mainstaff, null),
        $r.like(meta.substaff, "花子")
      )
  );

  // 単一項目への OR 検索は引数を追加するだけで実現可能です。
  // WHERE "mainstaff" LIKE '%花子%' OR "mainstaff" LIKE '%太郎%' OR "mainstaff" LIKE '%次郎%'
  criteria.add($r.like(meta.mainstaff, "花子", "太郎", "次郎"));

  // 複雑な検索条件。
  // "住所が「沖縄」のお客様で、かつ、主担当が「花子」
  //  または、
  //  住所が「北海道」のお客様で、かつ、主担当が「次郎」"
  // となっている顧客データを抽出する。
  // WHERE (("address" LIKE '%沖縄%' AND "mainstaff" LIKE '%花子%') OR ("address" LIKE '%北海
道%' AND "mainstaff" LIKE '%次郎%' ))
  criteria.add(
      $r.or(
        $r.and(
          $r.like(meta.address, "沖縄"),
          $r.like(meta.mainstaff, "花子")
        ),
        $r.and(
          $r.like(meta.address, "北海道"),
          $r.like(meta.mainstaff, "次郎")
        )
      )
  );
```

利用できる検索条件

$r では次の関数が用意されています。 これらを $r.or() や $r.and() でつなげていくことになります。

提供されるメソッド	説明	コードの記述方法
eq	等しい	eq(meta, values)
ne	等しくない	ne(meta, values)
like	文字列の部分一致	like(meta, values)
likePrefix	文字列の前方一致	likePrefix(meta, values)
likeSuffix	文字列の後方一致	likeSuffix(meta, values)
ge	以上	ge(meta, value)
gt	より大きい	gt(meta, value)
le	以下	le(meta, value)
lt	より小さい	lt(meta, value)
between	範囲検索	between(meta, lowValue, highValue)
isNull	nullかどうか	isNull(meta)
isNotNull	nullでないかどうか	isNotNull(meta)

(174) (175)

(174) 1. 引数が value ではなく、values となっている関数は複数の値を指定することが可能です。

(175) 2. 複数の値を指定した場合 eq(), like(), likePrefix(), likeSuffix()は OR 条件となり、ne() は AND 条件となります。

仕様・制約

- ダウンロード処理では、このクライテリアは使われません。ダウンロード処理で検索式をカスタマイズする場合は、生成された Java コードを直接、修正する必要があります。 R8.3.3 で改善されました。ダウンロード処理で Download<モデルID>CriteriaConverter_convert.js スクリプトが呼び出されます。R8.3.4 で、このスクリプトをDesignerで直接、作成できるようになりました。
- 全文検索機能を有効にしたとき、このクライテリアは使われません。（全文検索機能を利用した場合、クライテリアとは別方式の検索処理となっています。）

レイアウト

基本構成

標準レイアウト

ここでは図のモデル定義を用いて、レイアウトの定義方法を学びます。

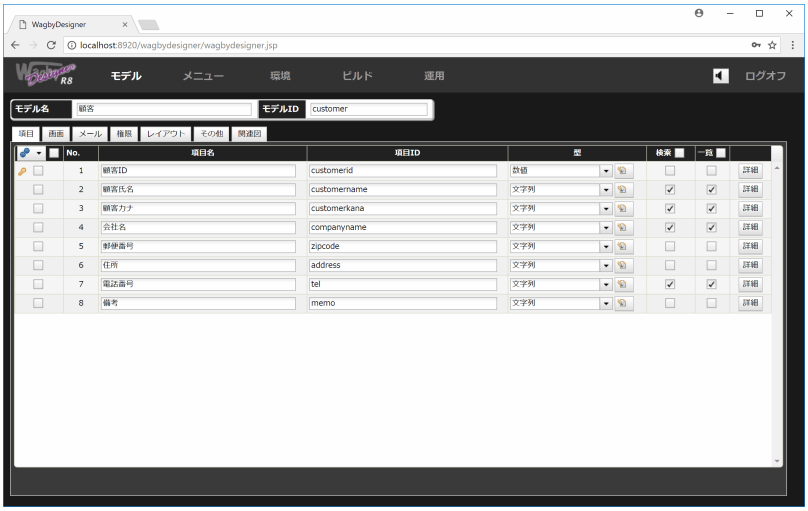


図 サンプルのモデル定義

レイアウトの定義

画面レイアウトの設計は、"レイアウト" タブで行います。

タブ	説明
検索	検索画面のレイアウトイメージを編集するスクリーンエディタです。[後述]
一覧	一覧表示画面のレイアウトイメージを編集するスクリーンエディタです。[後述]
詳細	詳細画面のレイアウトイメージを編集するスクリーンエディタです。このレイアウトは登録、更新画面でも用いられます。このページで説明します。
リポジトリ	検索、一覧、詳細タブでレイアウトした情報は、すべてリポジトリに反映されます。リポジトリを直接、操作してレイアウトを定義することもできます。つまりレイアウト定義はスクリーンエディタもしくはリポジトリ編集の両方を行き来しながら設計できる、2-Way方式となっています。

ここではレイアウトタブの詳細タブを開きます。標準では、モデル項目の並びがそのまま画面に反映されています。

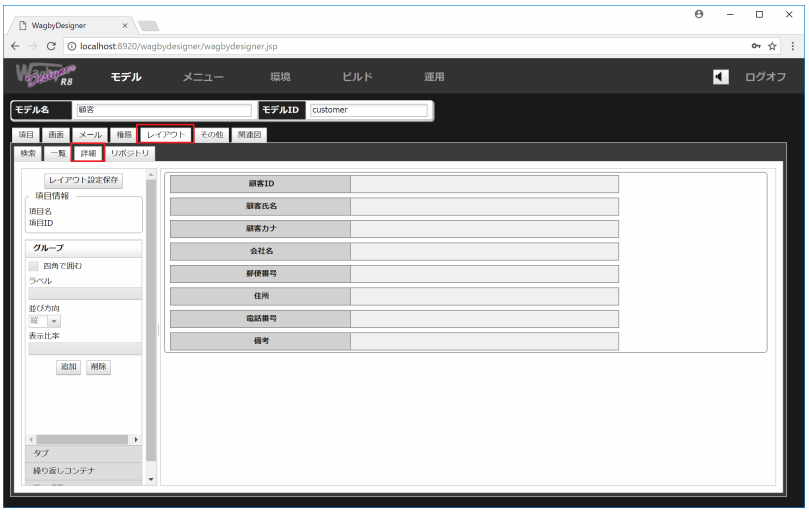


図 レイアウト定義のためのスクリーンエディタ

ビルドしたアプリケーションは、図の画面と同様のイメージとなっています。

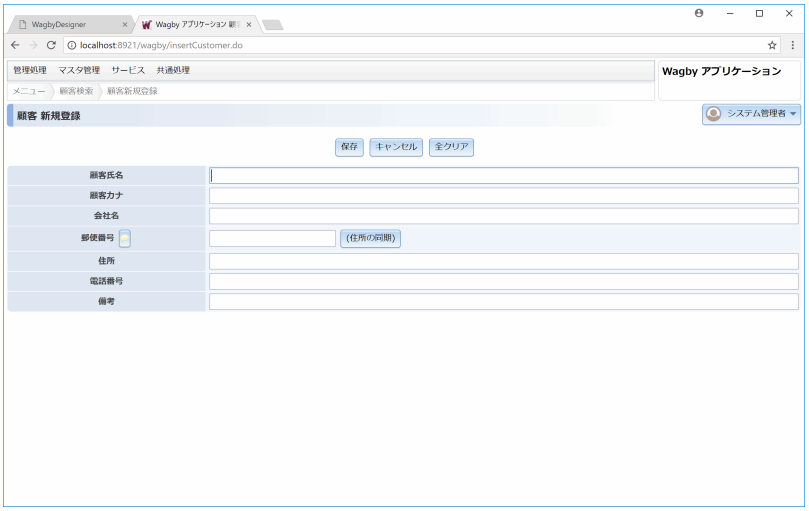


図 ビルドしたアプリケーションの画面

スクリーンエディタの画面と、実際にビルドしたアプリケーション画面との違い

スクリーンエディタは、各項目の「配置」に特化して編集できるようになっています。　そのため、配色(CSS)　や各種ボタン、そして項目の入力フィールド長などはビルドしたアプリケーションで確認するようになっています。

項目を横に並べる

複数の項目を「グループ」内に配置し、これを横に並べるというレイアウトデザインです。

図では「顧客氏名」と「顧客カナ」項目が「個人情報」というグループの中で横に並んでいます。

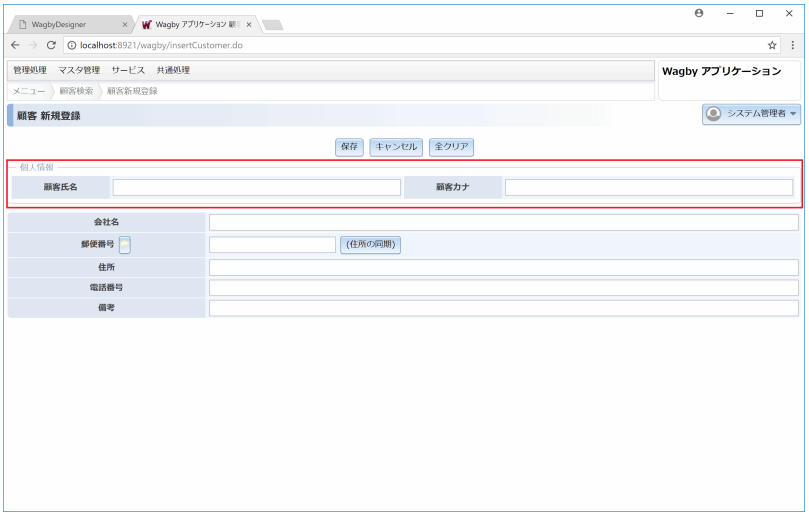


図 横並びの例

レイアウト設計

「レイアウト ＞ 詳細」タブを開きます。左側の設定欄にある「グループ」の「追加」ボタンを押下します。

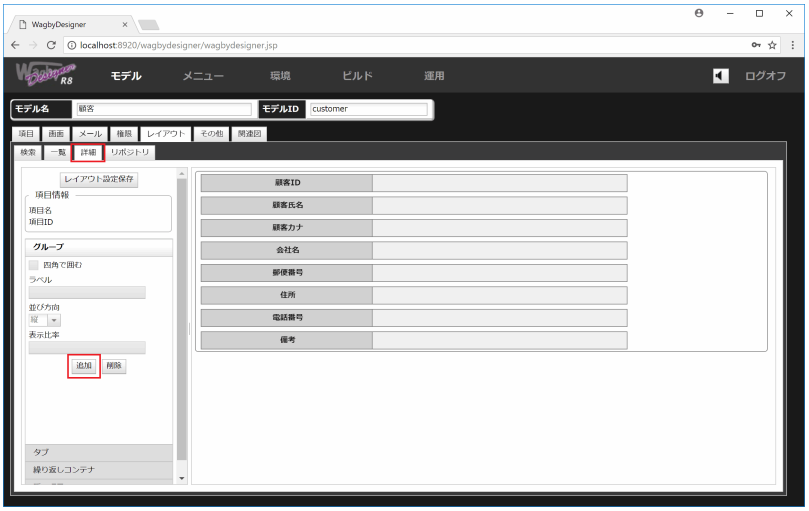


図 グループの追加

右側のレイアウト画面部に、グループ（パネル）が用意されます。

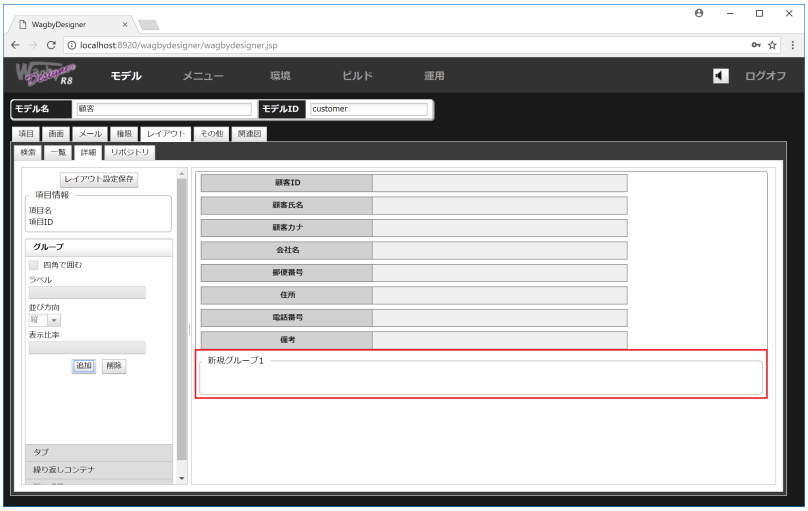


図 グループが用意された

レイアウト画面部のグループをマウスで選択します。すると左側の設定欄にある「グループ」の詳細を入力できるようになります。

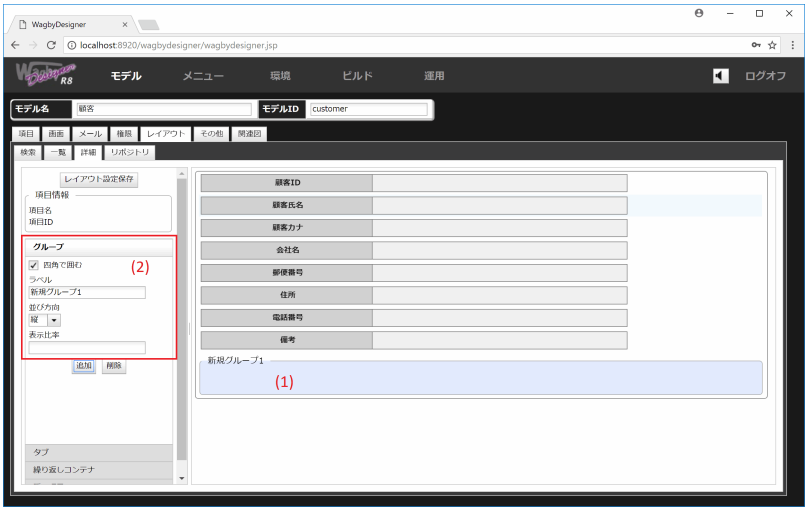


図 グループを選択する

設定欄の「ラベル」を変更し、タブキーを押下します。そのタイミングでレイアウト画面部の名称が変更されます。ここでは「個人情報」というグループ名に変更してみます。

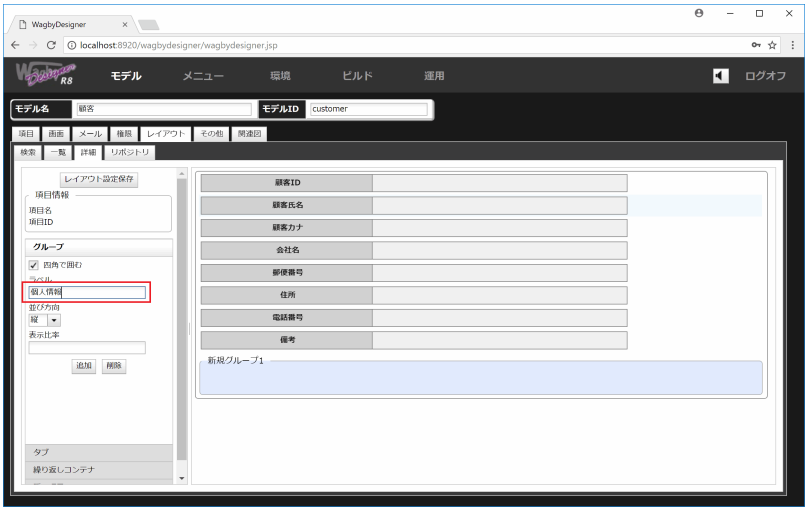


図 グループ名を変更する(1)

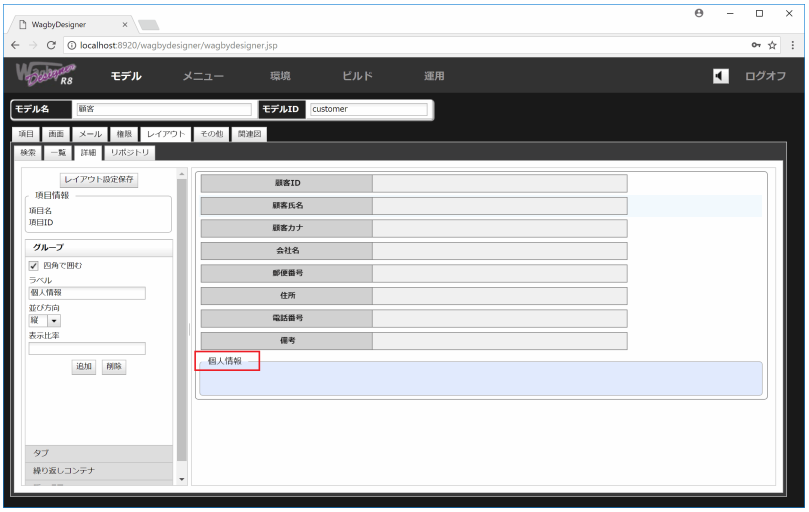


図 グループ名を変更する(2)

レイアウト画面部に配置された項目およびグループは、ドラッグ&ドロップ操作で配置を変更することができます。ここでは「個人情報」グループを上部に配置してみます。

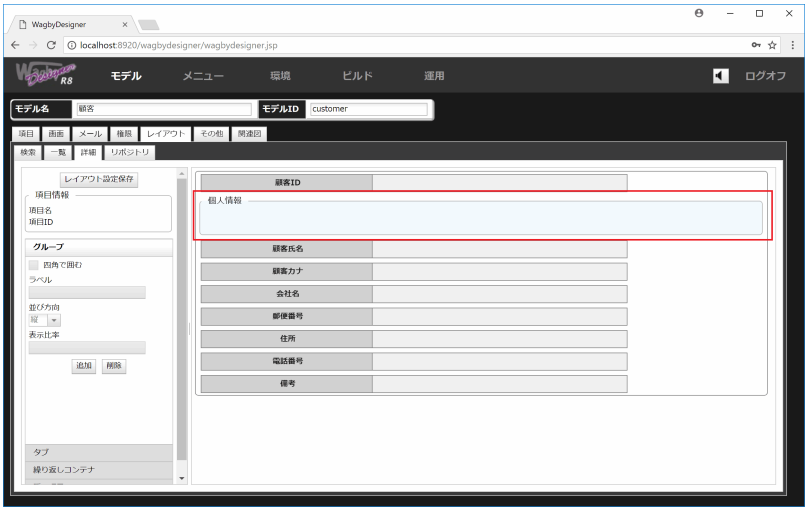


図 ドラッグ&ドロップで配置を入れ替える

さらに、「顧客氏名」と「顧客カナ」項目をドラッグ&ドロップで「個人情報」グループ内に配置します。

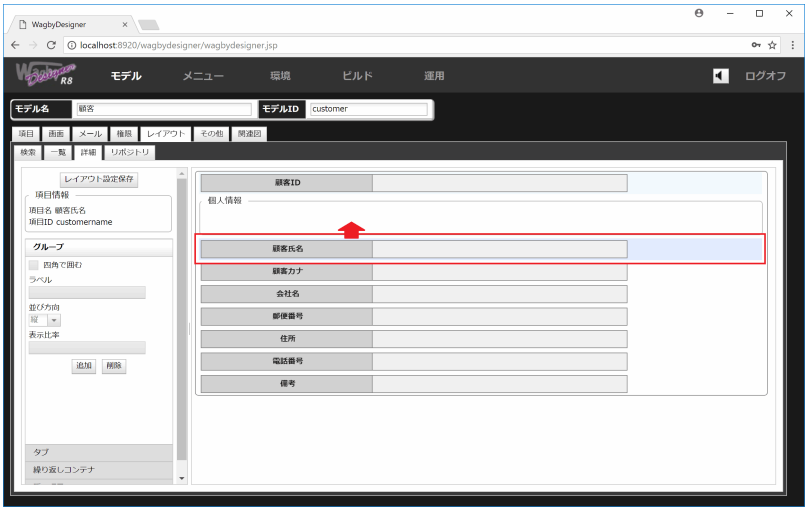


図 項目をグループ内に配置する(1)

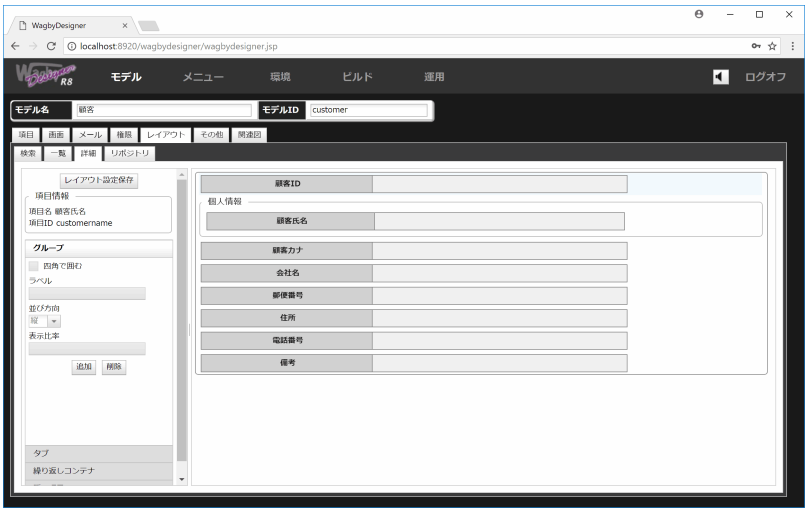


図 項目をグループ内に配置する(2)

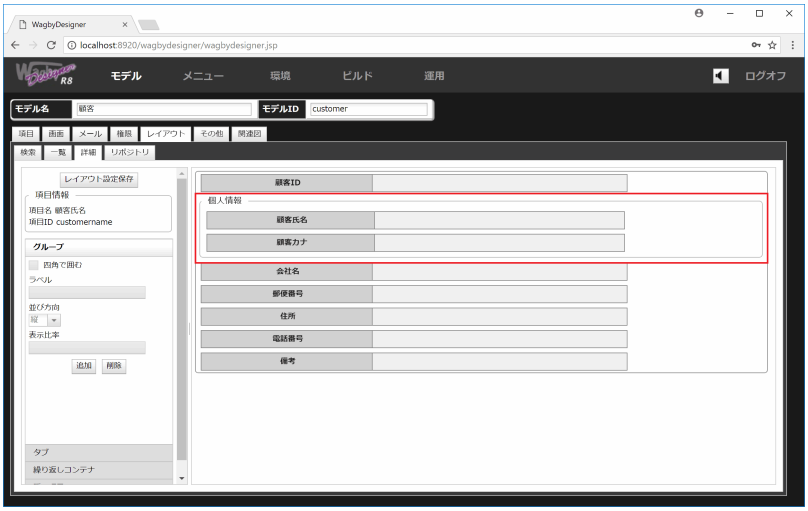


図 項目をグループ内に配置する(3)

再び「個人情報」グループ全体をマウスで選択します。左側の設定欄が有効になります。

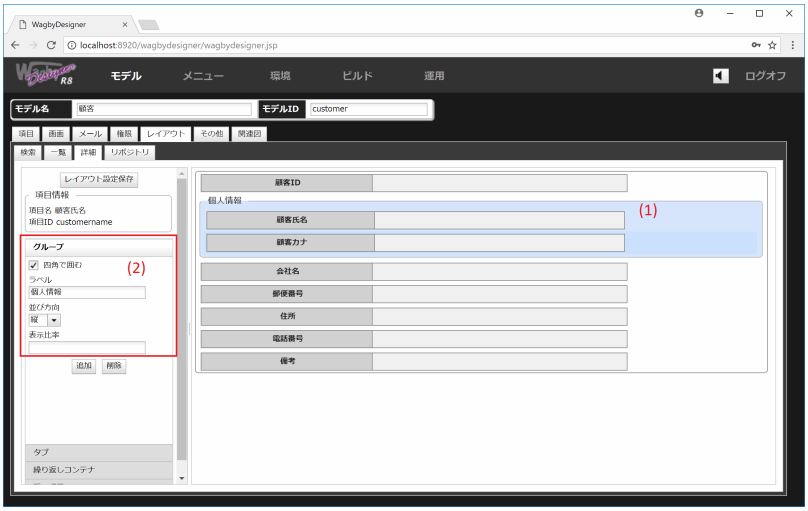


図 グループを選択する

「並び方向」を「横」とします。表示比率が自動設定されます。（もし自動設定されていない場合は、表示比率を手動で設定してください。例えば ″50%,50%″ と合計値が100%となるように入力します。）

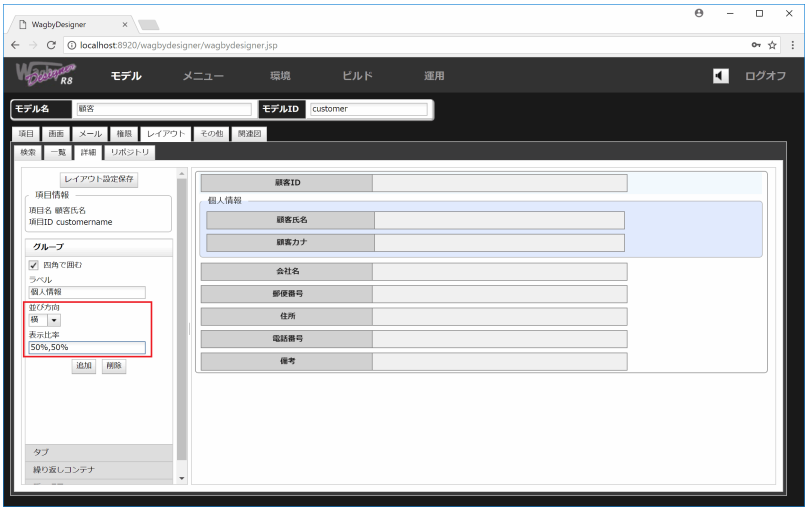


図 並び方向と比率を指定する(1)

Wagbyの項目配置は絶対座標ではなく、比率で設定します。

表示比率が未設定のときは縦並びとして解釈されます。表示比率の設定を忘れないようにしてください。

表示比率を入力してタブキーを押すことで、レイアウト画面に反映されます。

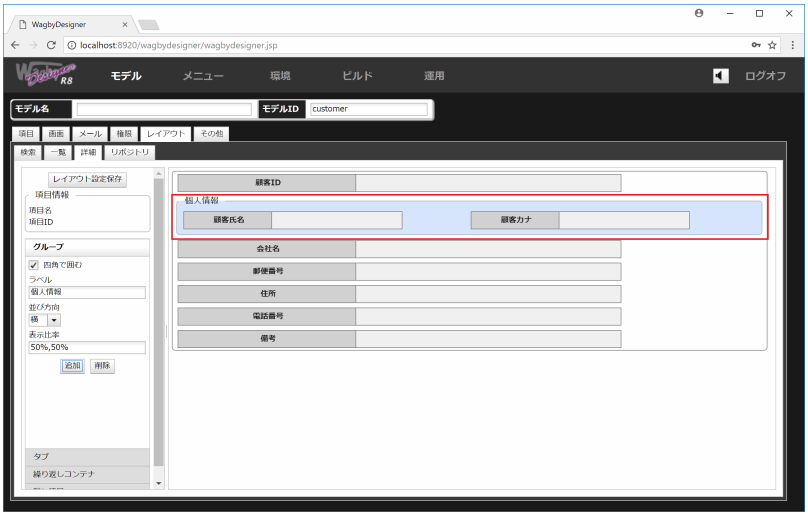


図 並び方向と比率を指定する(2)

スクリーンエディタの編集を保存する

左側の設定欄の上部にある「レイアウト設定保存」ボタンを押下し、編集内容を保存します。この操作により、リポジトリに反映されます。

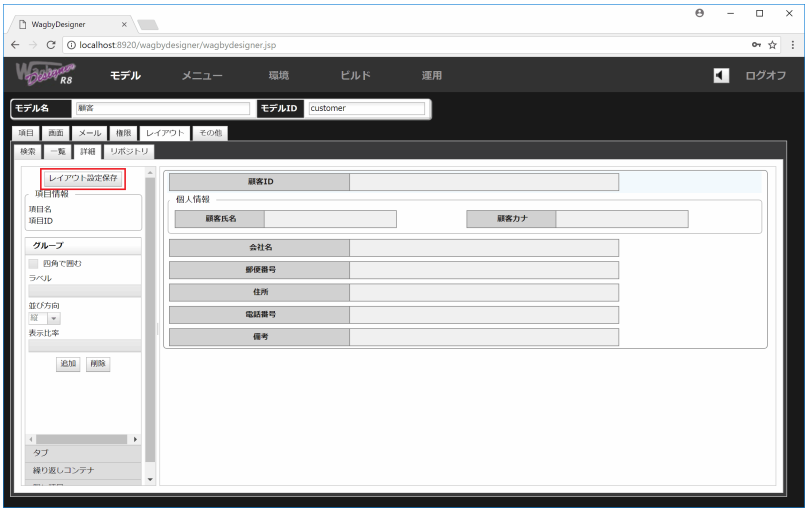


図 レイアウト設定を保存する

> ご注意ください。「レイアウト設定保存」ボタンを押下せずに他の画面に切り替えた場合、これまでの編集内容は反映されません。

リポジトリの確認

ここまでの設定が、どのようにリポジトリに反映されたかを確認します。「リポジトリ」タブを開きます。「レイアウト名」タブには、グループ名として入力した値がレイアウト名に反映されていることがわかります。「表示順」は、レイアウト画面での項目の並び順序が反映されています。

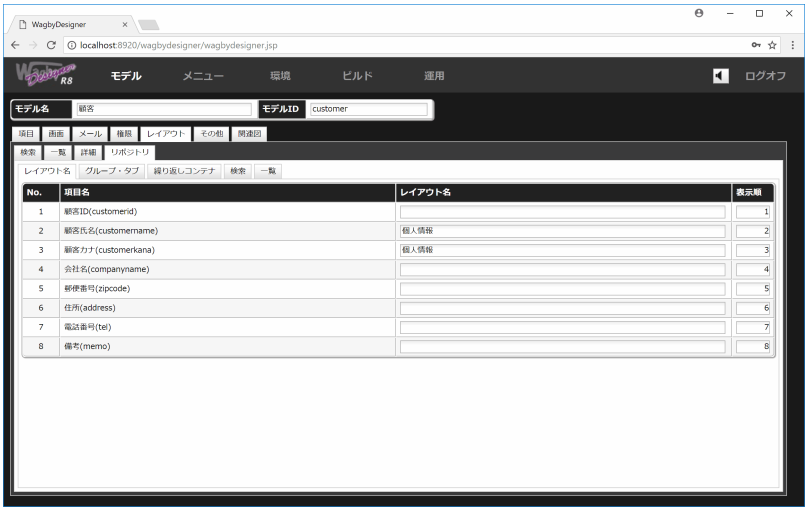


図 リポジトリの確認(1)

さらに「グループ・タブ」を確認します。レイアウト名「個人情報」の設定欄が反映されていることがわかります。

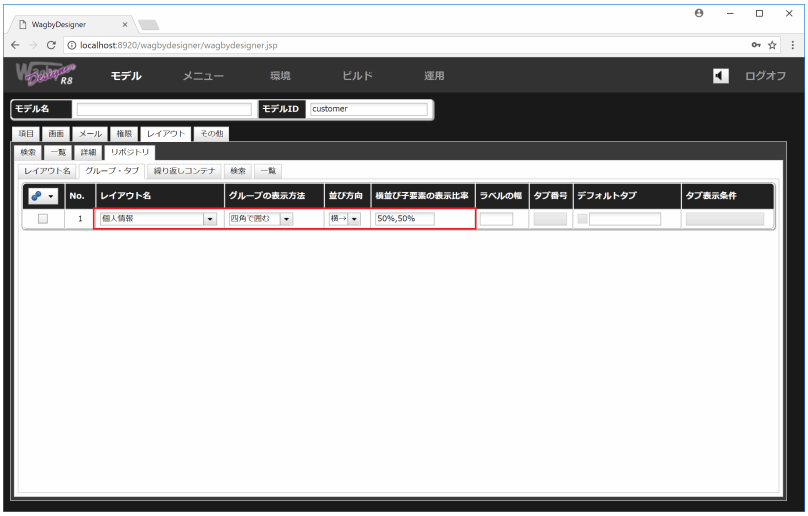


図 リポジトリの確認(2)

開発者はこのリポジトリを直接、入力することもできます。ここで入力した値は、「詳細」タブを開くとレイアウト画面に反映されています。

項目を縦に並べる

項目を縦に並べた例を図に示します。

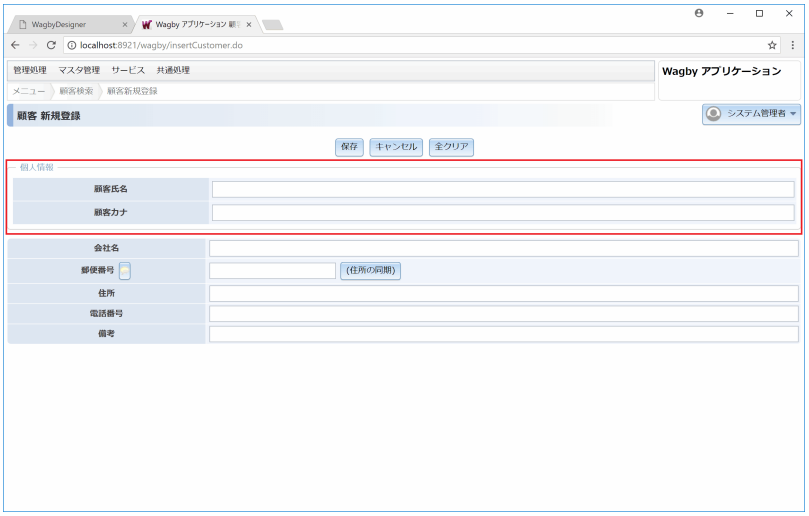


図 項目の縦並び

レイアウト設計

グループの並び方向を「縦」とします。縦並びの場合、表示比率は無視されます。

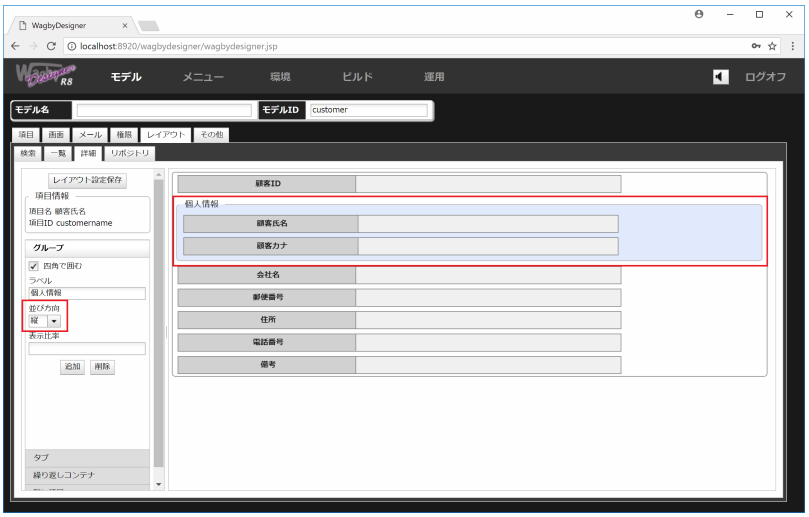


図 並び方向の指定

並び方向の標準は「縦」となっています。

横並びと縦並びを組み合わせる

一つの画面に、横並びと縦並びの項目を混在させることができます。

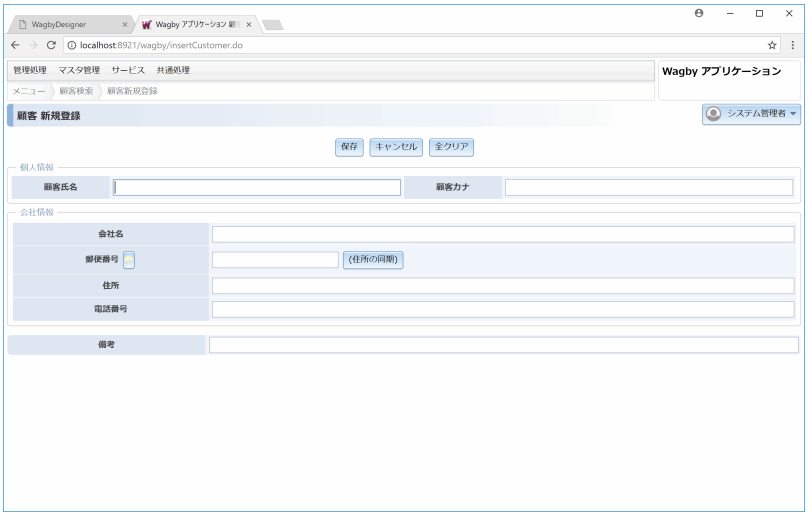


図 横並びレイアウトと縦並びレイアウトの混在

レイアウト設計

図で、「個人情報」グループに項目を横並びで設定しました。この続きとして説明します。「会社情報」を格納する新しいグループを用意します。ラベル名は「会社情報」とします。

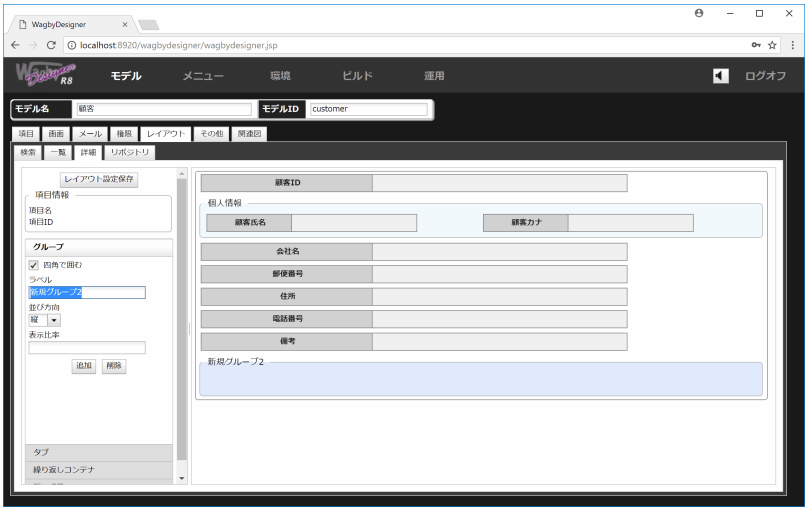


図 新しいグループ「会社情報」を用意する(1)

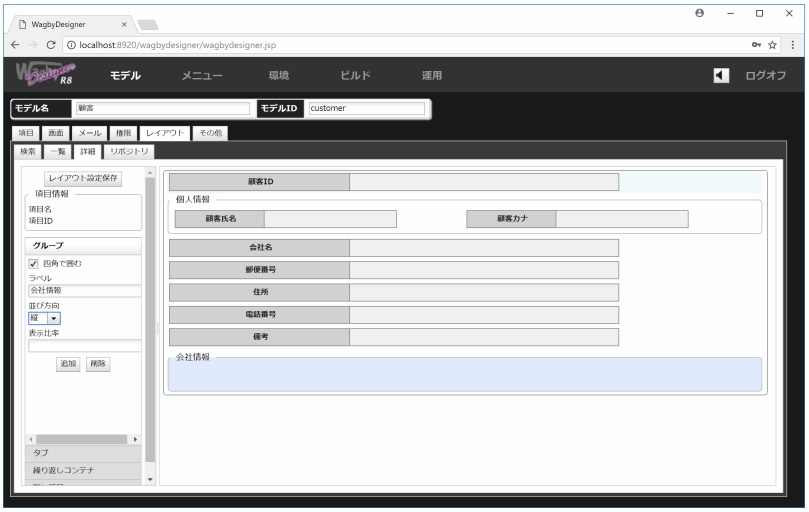


図 新しいグループ「会社情報」を用意する(2)

項目をドラッグ&ドロップし、会社情報グループ内に配置します。

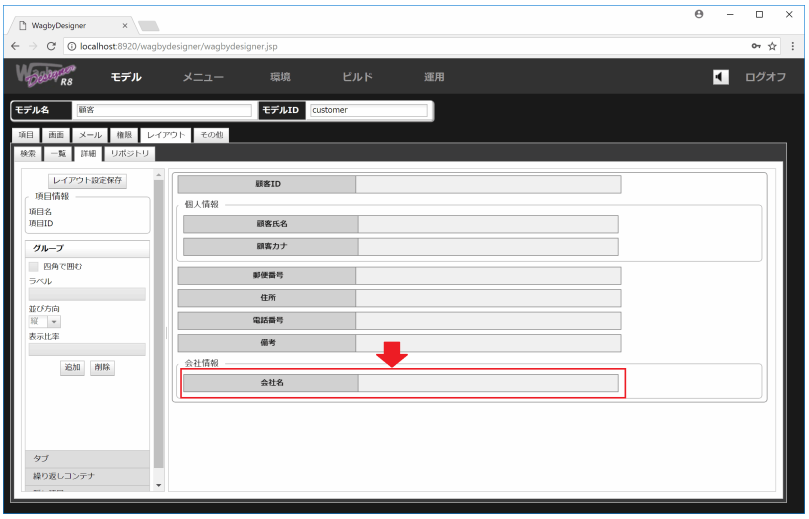


図 ドラッグ&ドロップで項目を配置する(1)

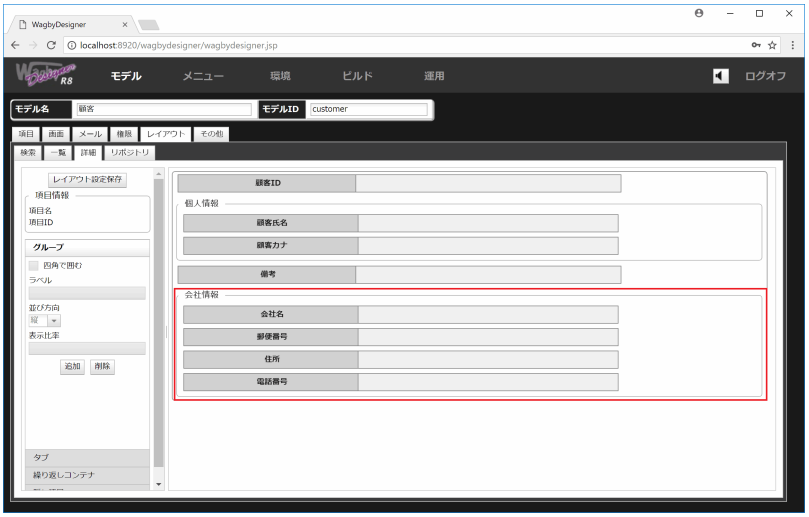


図 ドラッグ&ドロップで項目を配置する(2)

「会社情報」グループを選択し、ドラッグ&ドロップして位置を変えます。

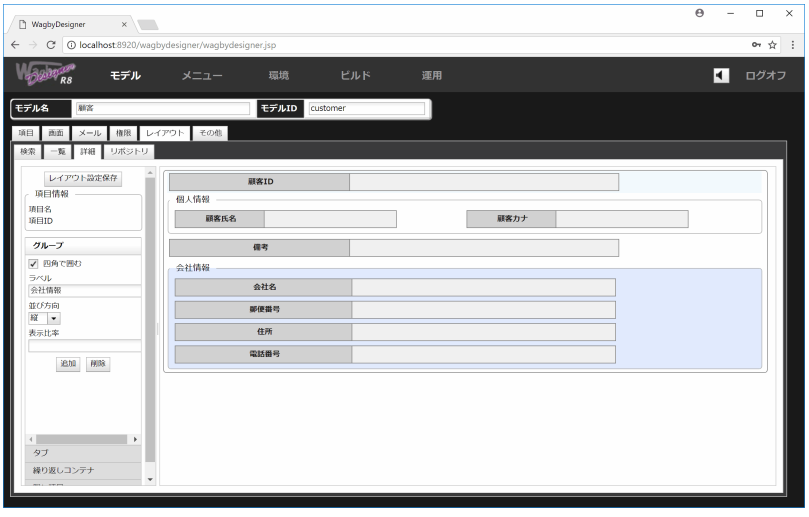


図　ドラッグ&ドロップで会社情報グループの位置を変更する(1)

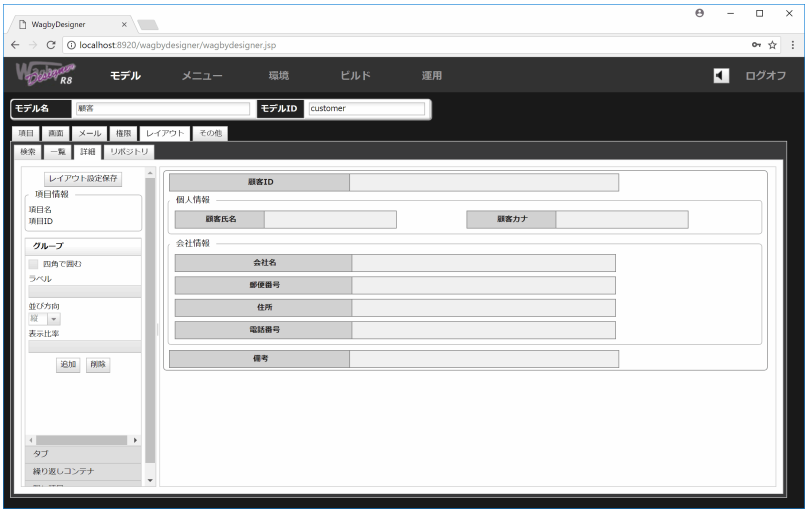


図　ドラッグ&ドロップで会社情報グループの位置を変更する(2)

スクリーンエディタの編集を保存する

左側の設定欄の上部にある「レイアウト設定保存」ボタンを押下し、編集内容を保存します。この操作により、リポジトリに反映されます。

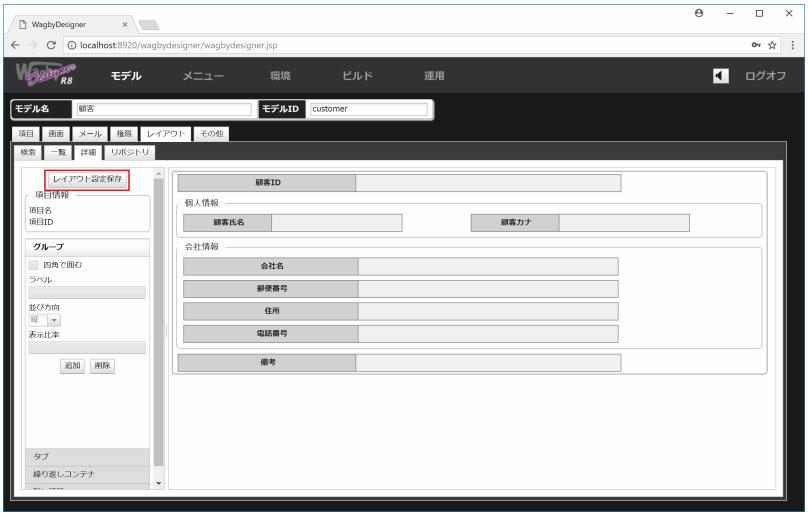


図　レイアウト設定保存ボタン

> ご注意ください。「レイアウト設定保存」ボタンを押下せずに他の画面に切り替えた場合、これまでの編集内容は反映されません。

リポジトリの確認

ここまでの設定が、どのようにリポジトリに反映されたかを確認します。「レイアウト名」には「個人情報」と「会社情報」がそれぞれ設定されています。

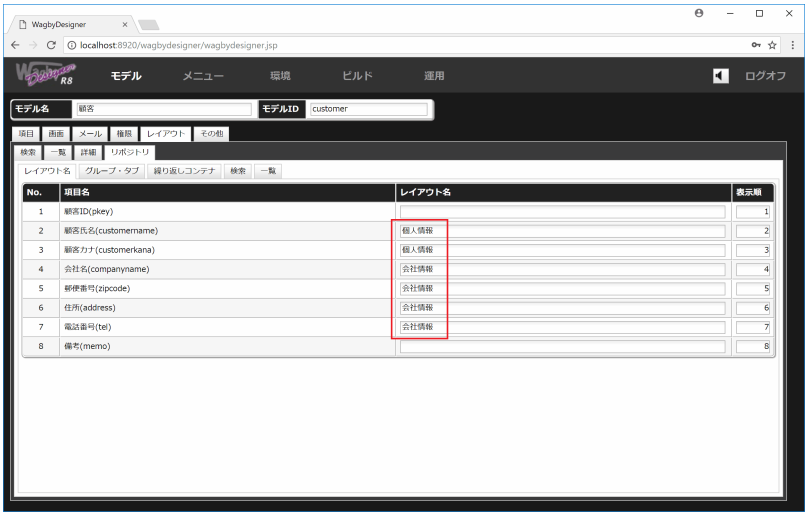


図 リポジトリの確認(1)

「グループ・タブ」を確認します。レイアウト名「個人情報」「会社情報」の設定がそれぞれ反映されていることがわかります。

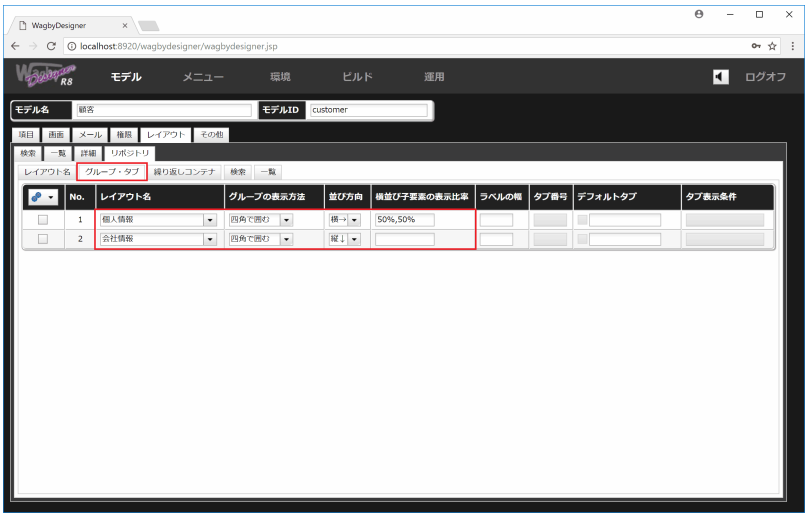


図 リポジトリの確認(2)

画面サイズを変えてみる

画面レイアウトは比率で設定するため、画面サイズを拡大・縮小してもレイアウトイメージは変わりません。

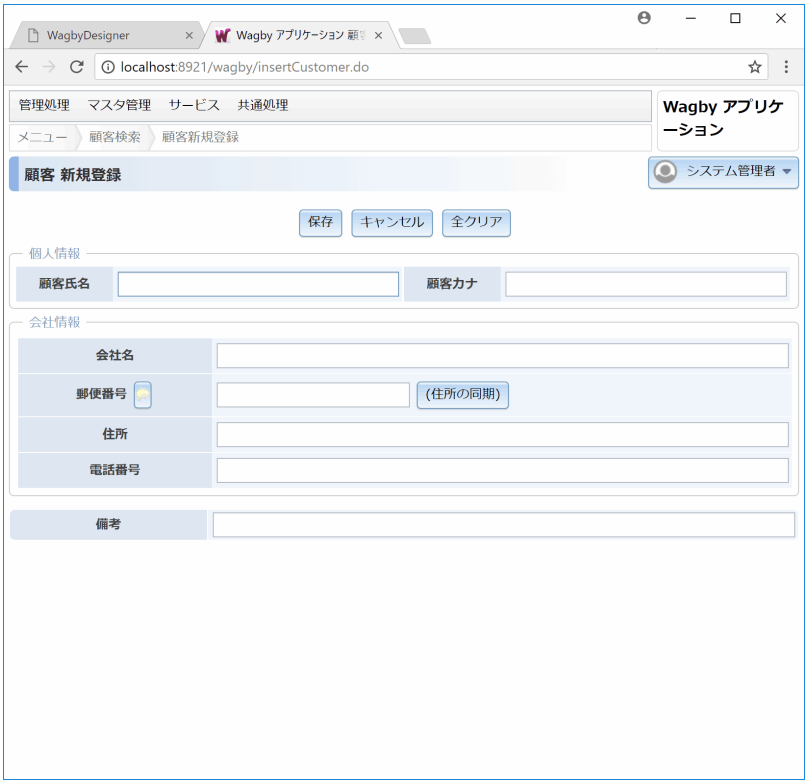


図 Webブラウザの画面サイズを小さくした例

このため、異なる画面解像度のPCやタブレットで閲覧しても、ほぼ同じ画面イメージで操作することができます。

隠し項目の扱い

隠し項目の設定は詳細画面と登録・更新系画面でそれぞれ用意されています。

- 出力制御>詳細画面>隠し項目にする
- 入力制御>隠し項目

ここで出力制御のみ隠し項目とした場合、ビルド後のアプリケーションでは登録・更新系画面では隠し項目となりません。

操作に関するTips

複数のグループを作成後、グループを移動する場合のテクニック !-- N20190218_092531 -->

グループをドラッグしたまま、他のグループの上を通過するとグレーの枠線内のエリアがドロップ可能なエリアと認識されなくなり、ドラッグしているグループのアイコンが赤になります。このとき、ドラッグしているグループや項目がドロップできません。

グループをドラッグしたまま、いったんグレーの枠線の外に出して戻すとグレーの枠線内が黄色で表示されるようになります。あわせて、ドラッグしているグループのアイコンがドロップ可能であることを示す緑になります。

入れ子レイアウトと隠しレイアウト

縦並びの中に横並び

縦並びレイアウト内に、横並びレイアウトを組み合わせる例を図に示します。

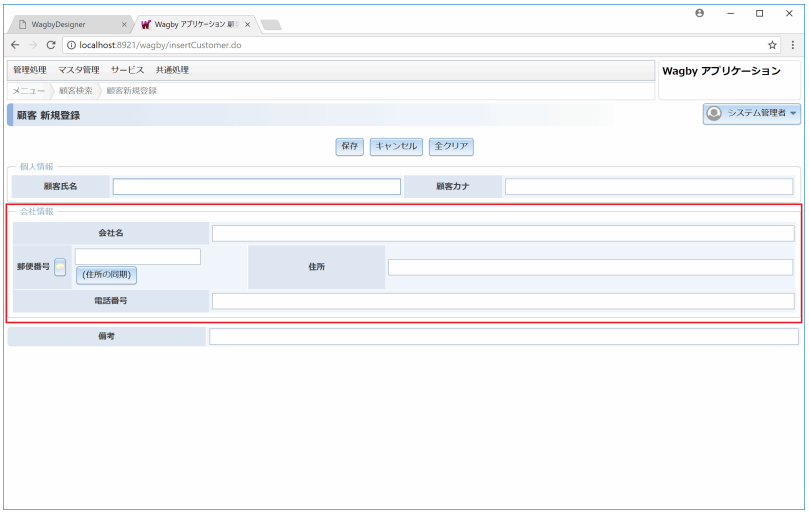


図 縦並びレイアウト内の横並びレイアウト

レイアウト設計

横並びと縦並びを組み合わせたレイアウトから続けます。

新しいグループを用意し、「郵便番号」と「住所」項目をドラッグ&ドロップで移動します。

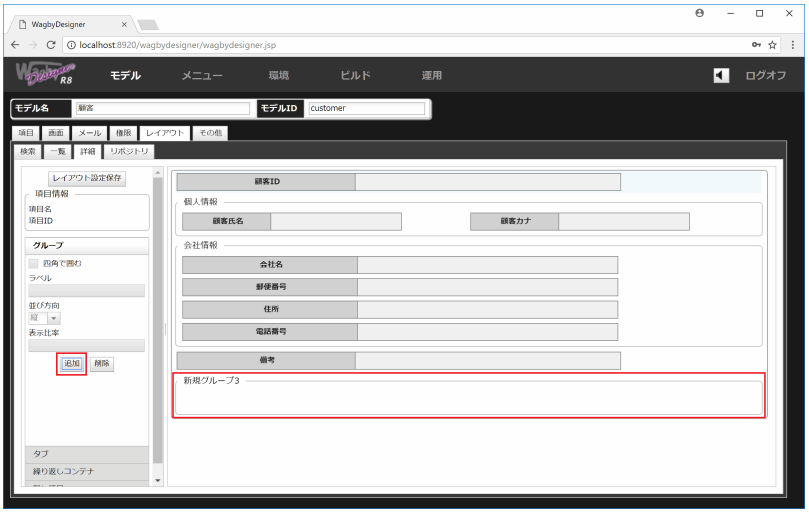


図 新しいグループを用意する

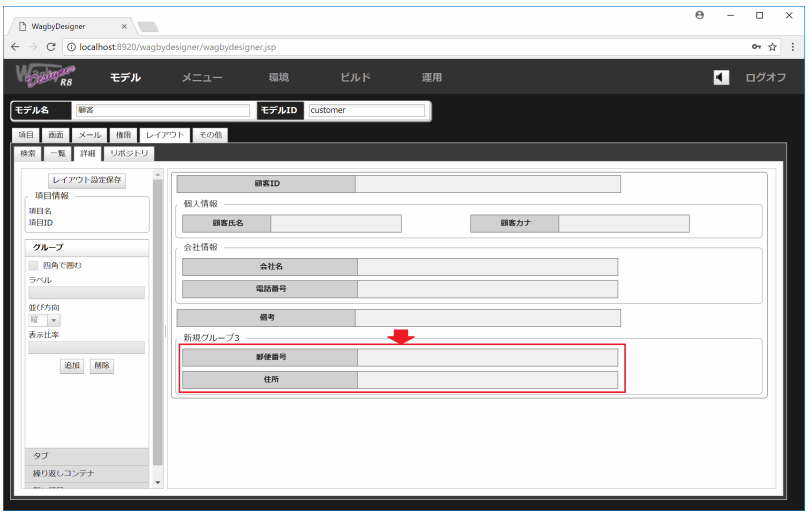


図 ドラッグ&ドロップで項目を移動する

このグループをマウスで選択し、ドラッグ&ドロップで「会社情報」グループの中へ配置します。

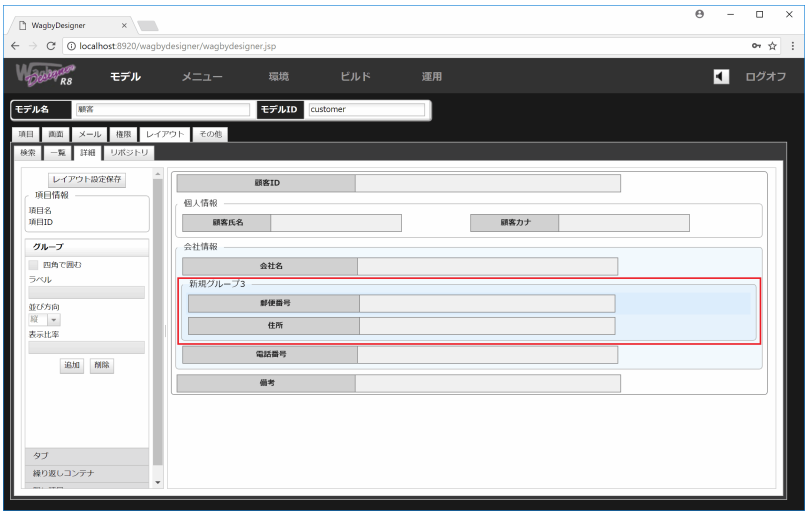


図 グループの中にグループを配置する

このグループを選択し、左側の設定欄を修正します。

ラベルを空欄とし、並び方向を横にします。表示比率は「30%, 70%」と入力します。

その後、「四角で囲む」を解除します。

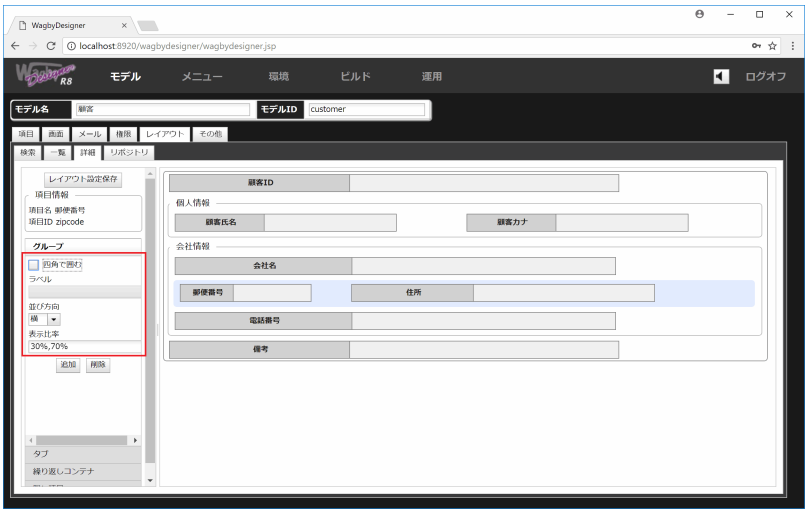


図 設定欄の値を修正する

「四角で囲む」を解除すると、ラベルの変更が行えなくなります。そのため最初にラベルを空欄とし、最後に「四角で囲む」を解除するとよいでしょう。

スクリーンエディタの編集を保存する

左側の設定欄の上部にある「レイアウト設定保存」ボタンを押下し、編集内容を保存します。この操作により、リポジトリに反映されます。

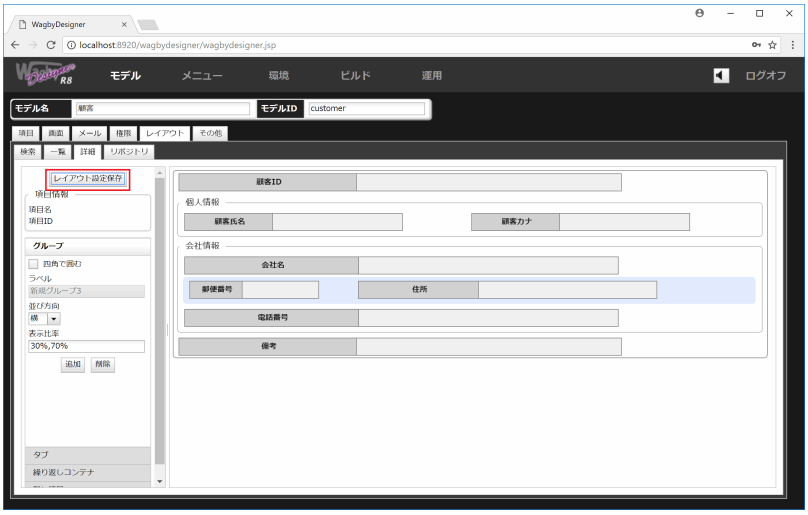


図 レイアウト設定を保存する

ご注意ください。「レイアウト設定保存」ボタンを押下せずに他の画面に切り替えた場合、これまでの編集内容は反映されません。

リポジトリの確認

ここまでの設定が、どのようにリポジトリに反映されたかを確認します。

入れ子の構造

「会社情報」というレイアウト内に、さらにレイアウト「*P3」が設定されていることがわかります。 レイアウトの入れ子は「>」文字を使って表現されます。

```
レイアウト>レイアウト
```

レイアウトの入れ子は何層にわたって行うことができます。

隠しレイアウト

ラベルを空欄とし、さらに「四角で囲む」を解除したとき、レイアウト名の先頭に「*」が付与されます。これは隠しレイアウトとなり、表示時にパネルが用意されません。レイアウトの並びを縦、横、縦...と入れ子にして組み合わせるときに、隠しレイアウトは効果を発揮します。

アスタリスク「*」のあとに続くレイアウト名は英数字の組み合わせで表現します。標準では "P" が用いられます。

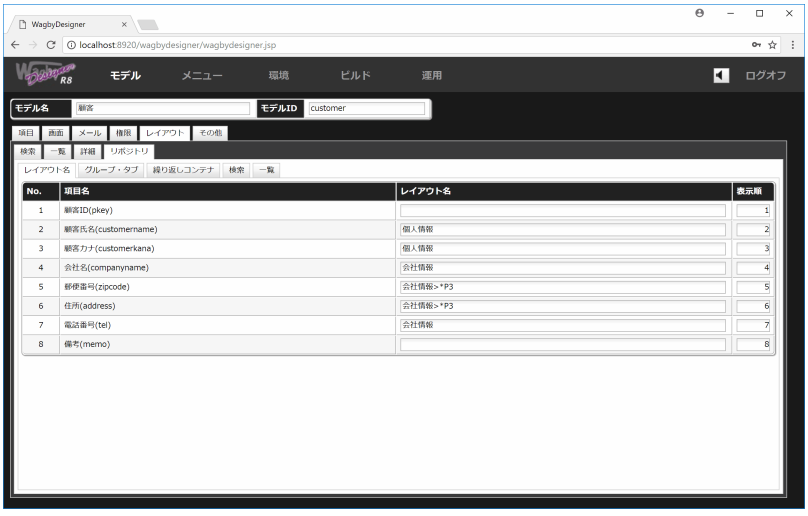


図 リポジトリの確認(1)

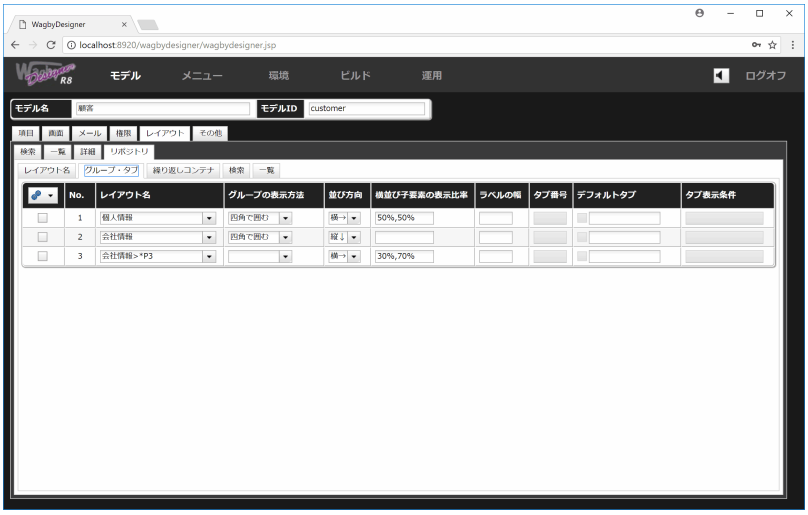


図 リポジトリの確認(2)

横並びの中に縦並び

全体を横並びとし、入れ子内のレイアウトを縦並びとした例を図に示します。

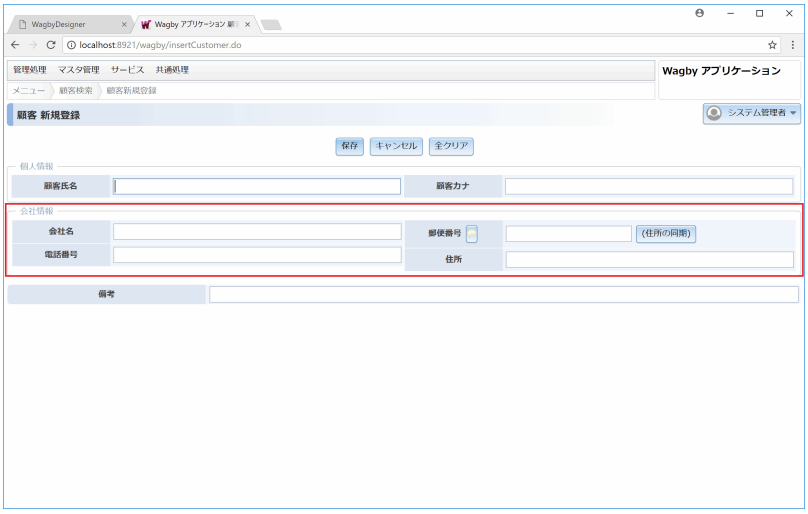


図 横並びと縦並びの組み合わせ

レイアウト設計

「横並びと縦並びを組み合わせたレイアウト」から続けます。

図の会社情報グループを変更していきます。

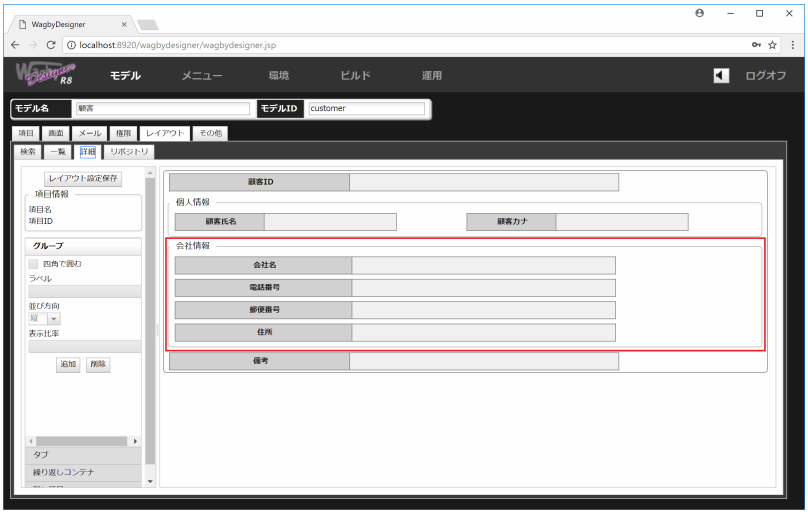


図 会社情報グループ

新しいグループを二つ用意します。

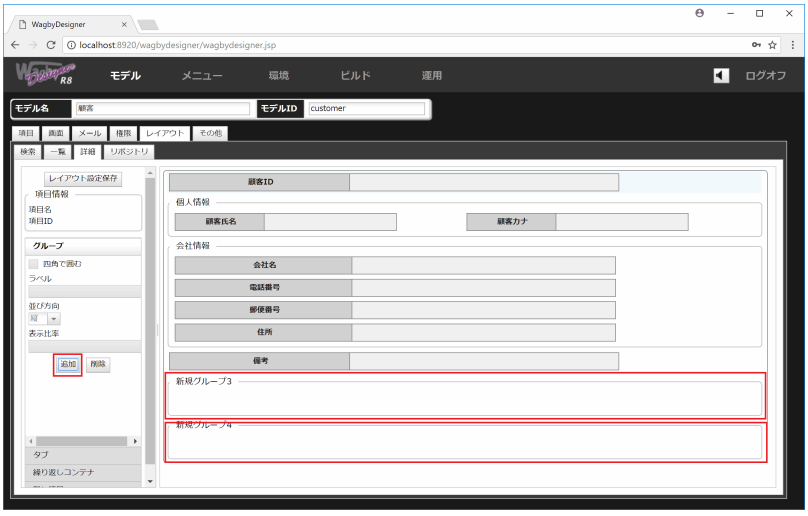


図 新しいグループを用意する

会社情報グループにあった4つの項目を、新しいグループ内に移動します。

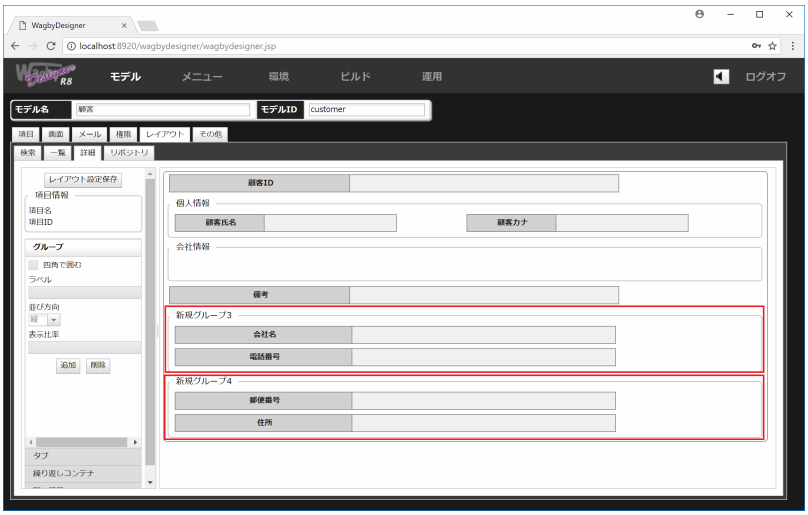


図 4つの項目を移動する

会社情報グループ内の入れ子グループとします。

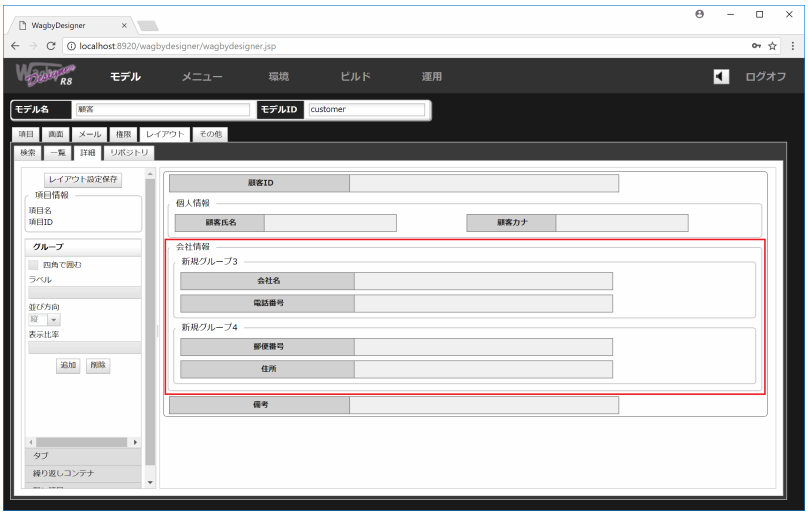
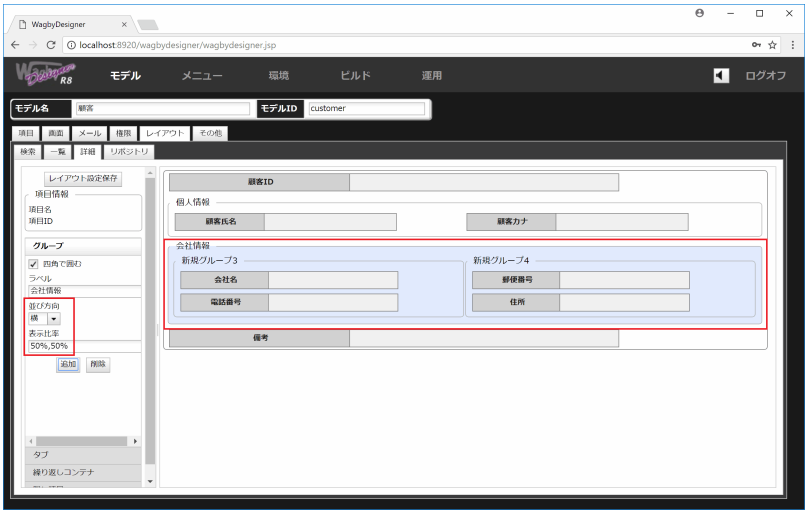


図　ドラッグ&ドロップで入れ子構造とする

会社情報グループ全体を選択し、横並びとします。比率は「50%, 50%」とします。



入れ子内の新規グループを選択し、ラベルを空白とします。

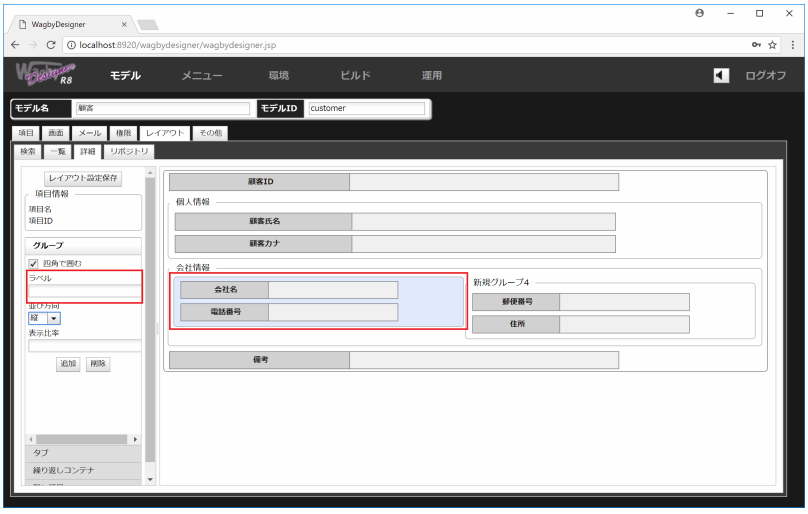


図　ラベルを空白とする

「四角で囲む」を解除します。

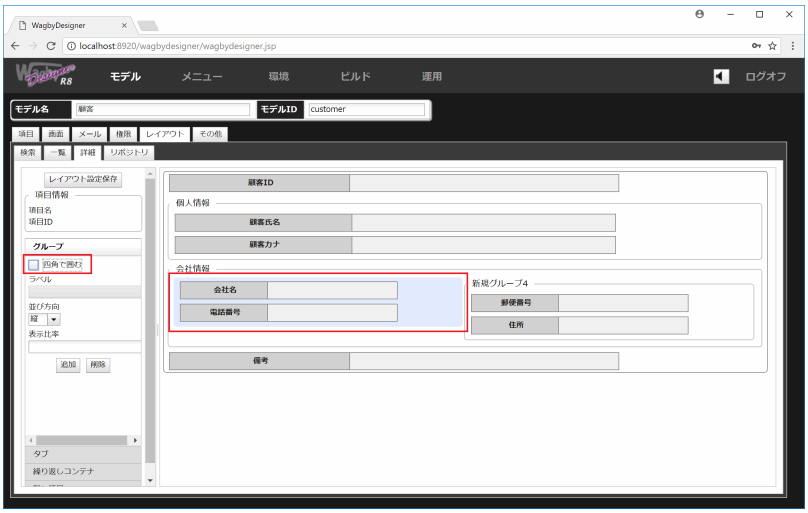


図 四角で囲むを解除する

もう一つのグループも同様にします。すべての設定が終わったら「レイアウト設定保存」を行います。

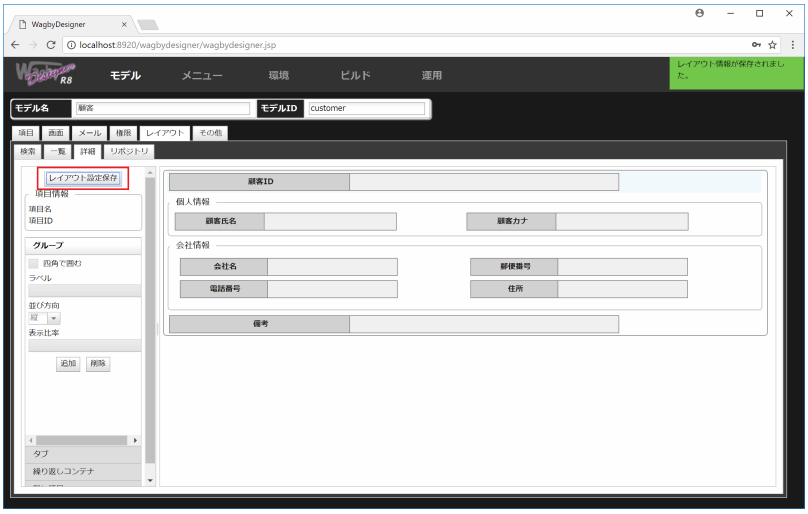


図 レイアウト設定を保存する

リポジトリの確認

ここまでの設定が、どのようにリポジトリに反映されたかを確認します。

二つの隠しレイアウト *P3 と *P4 が用意されていることがわかります。アスタリスク「*」に続くレイアウト名は表示時にパネルが用意されません。

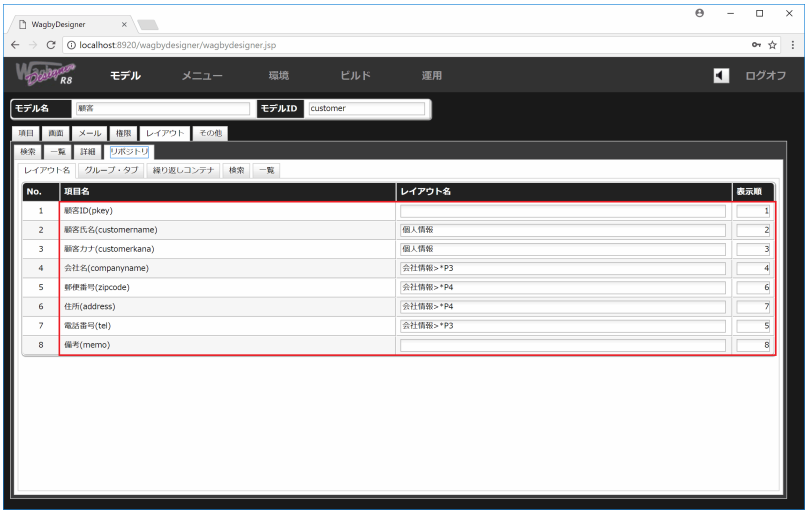


図 リポジトリの確認(1)

隠しレイアウトは「四角で囲む」も解除されています。

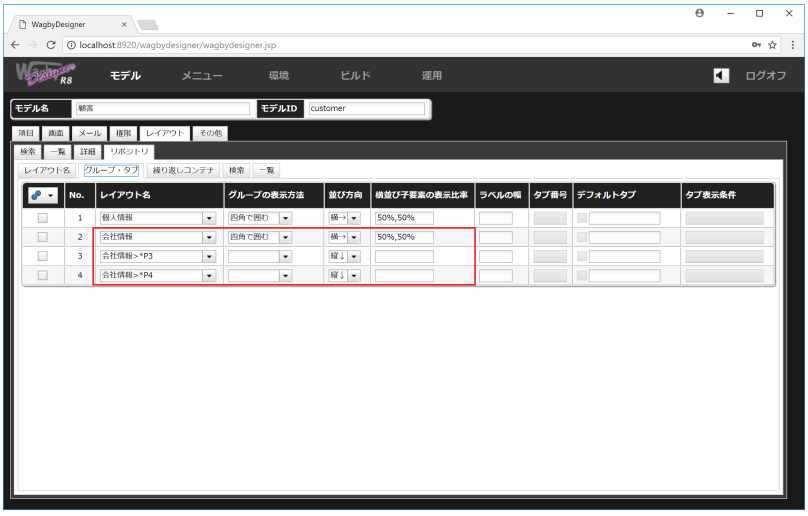


図 リポジトリの確認(2)

隠しレイアウトの混在

「顧客」レイアウトの中に隠しレイアウトと「会社」レイアウトを混在させた例を説明します。

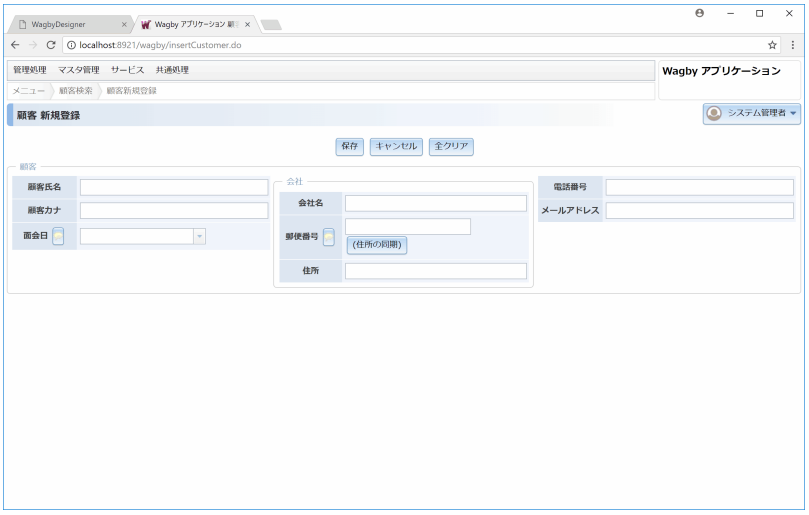


図 隠しレイアウトの混在

レイアウト設計

今回用いるモデルを図に示します。

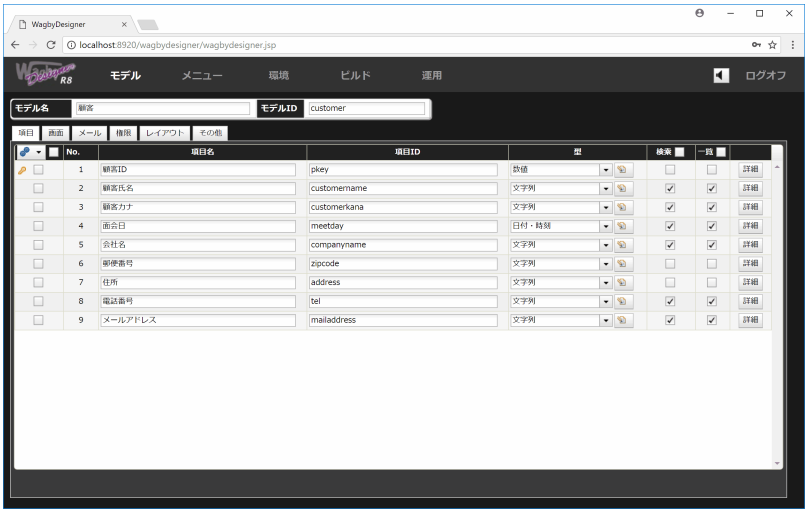


図 サンプルモデル

初期状態は、項目が順序よく整列されています。

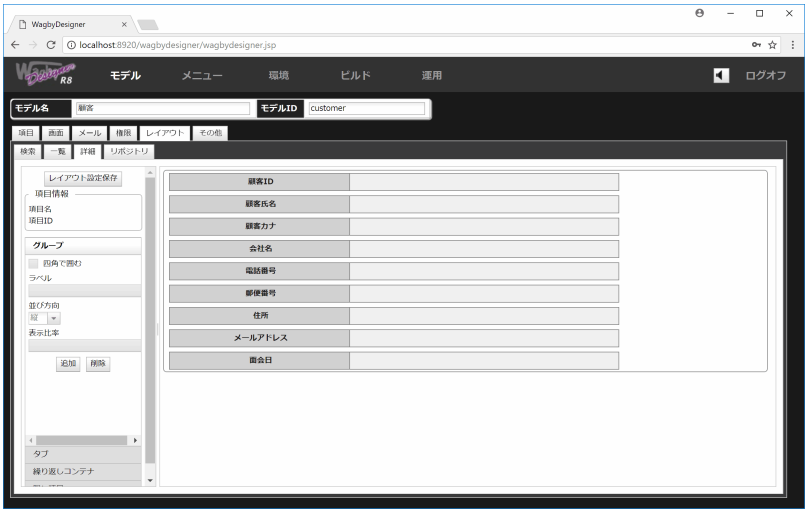


図 初期のレイアウト

3つのグループを用意します。

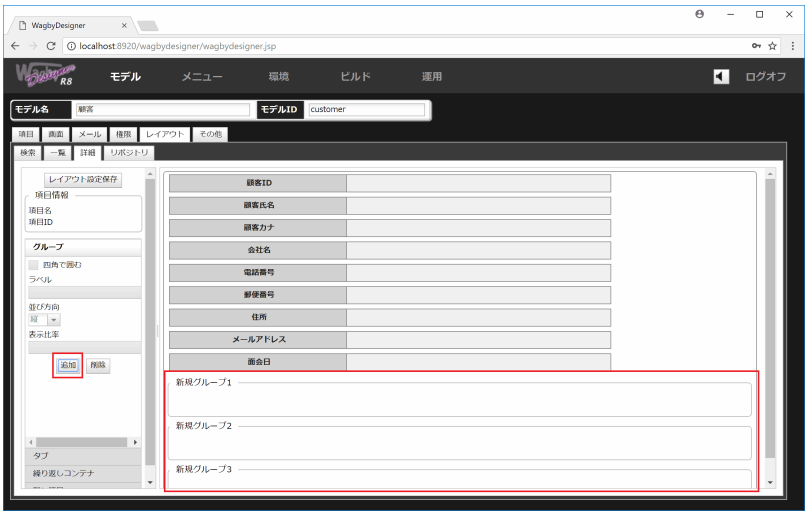


図 3つのグループを用意する

それぞれのグループに項目を配置します。

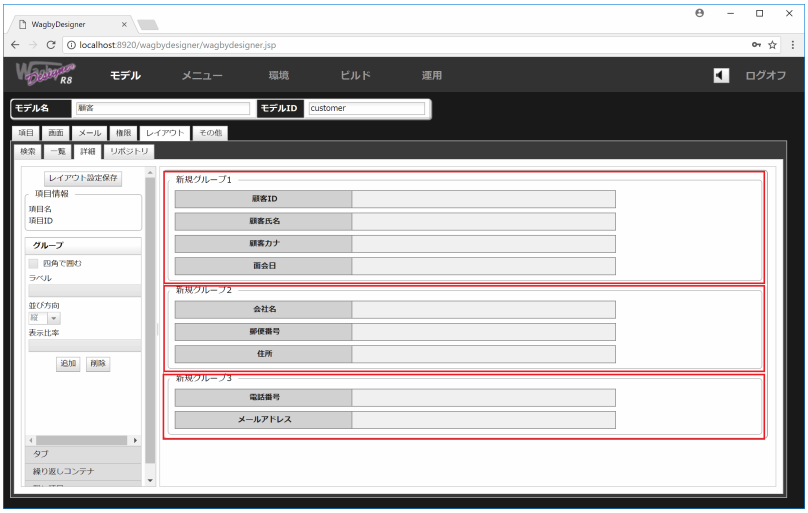


図 グループに項目を配置する

「新規グループ1」と「新規グループ3」は、ラベルを空白とし、「四角で囲む」を解除します。「新規グループ2」は、ラベルを「会社」に変更します。

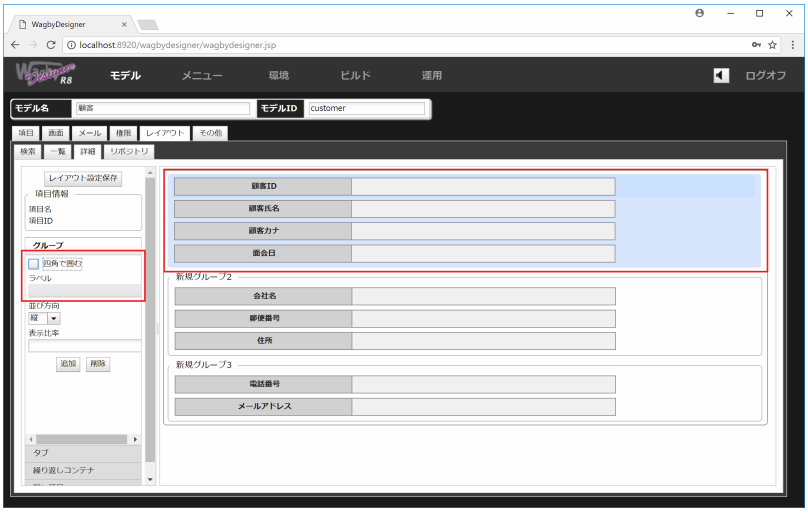


図 新規グループ1の設定

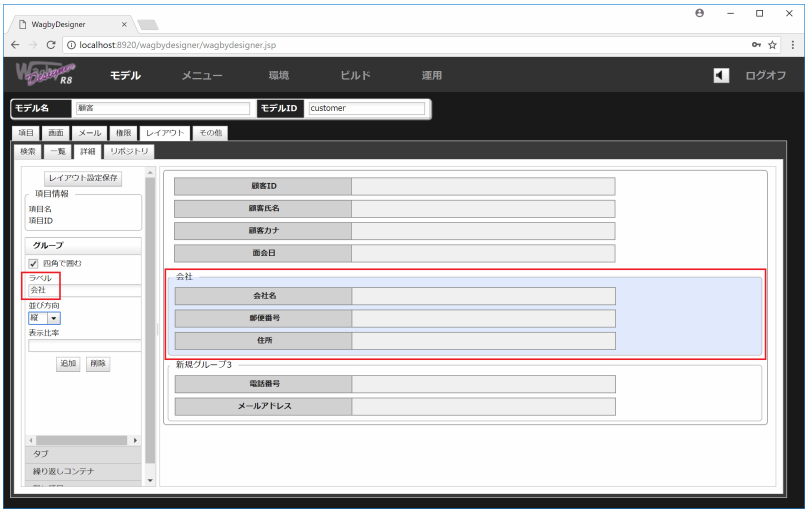


図 新規グループ2の設定

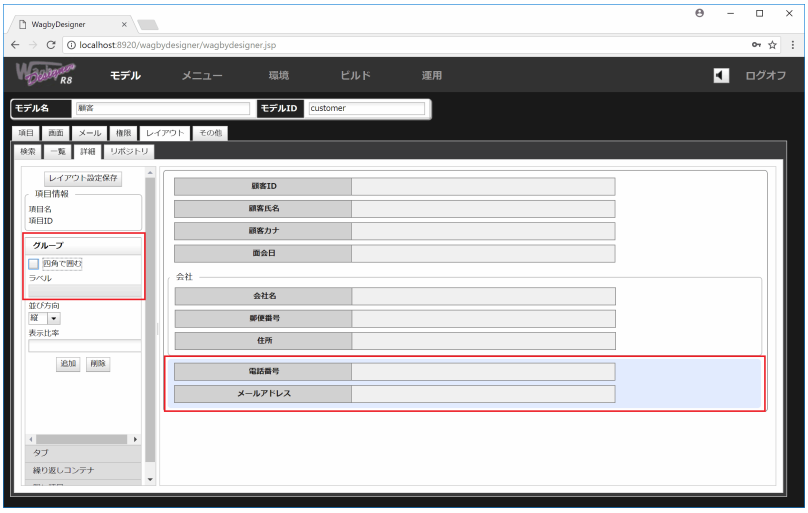


図 新規グループ3の設定

全体を囲むグループ「顧客」を用意します。

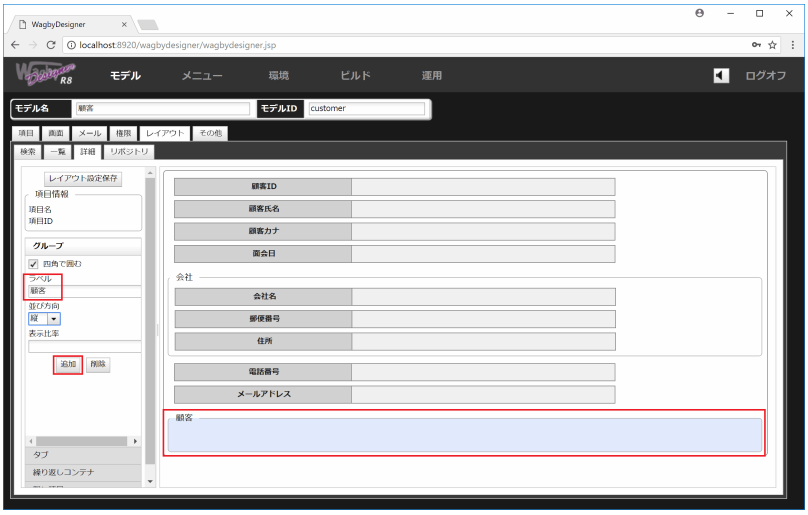


図 「顧客」グループを用意する

すでに用意した3つのグループを「顧客」グループの内部に配置します。

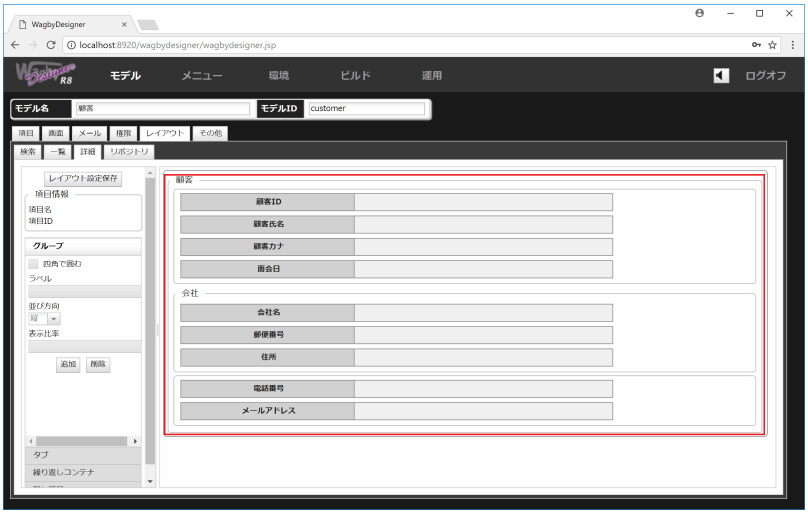


図 顧客グループへの配置

並び方向を横とし、表示比率を33%, 34%, 33%とします。ここまでの設定を保存します。

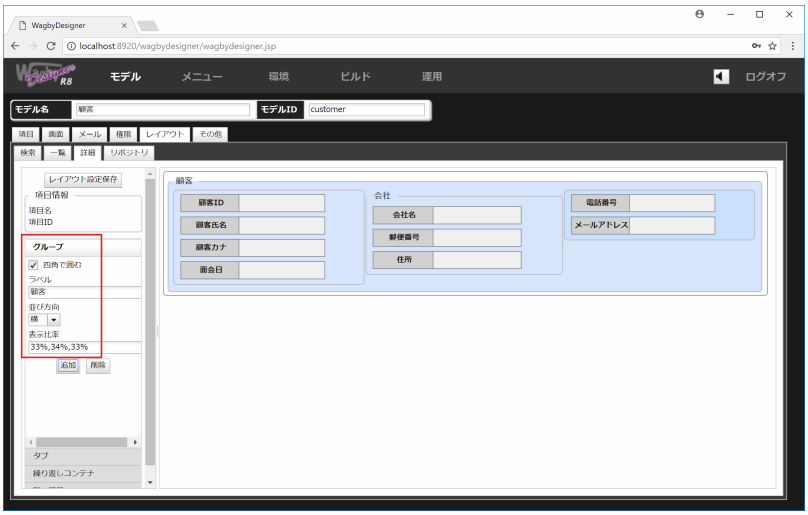


図 並び方向と表示比率の設定

リポジトリの確認

ここまでの設定が、どのようにリポジトリに反映されたかを確認します。レイアウト名は隠しレイアウトが含まれていることがわかります。

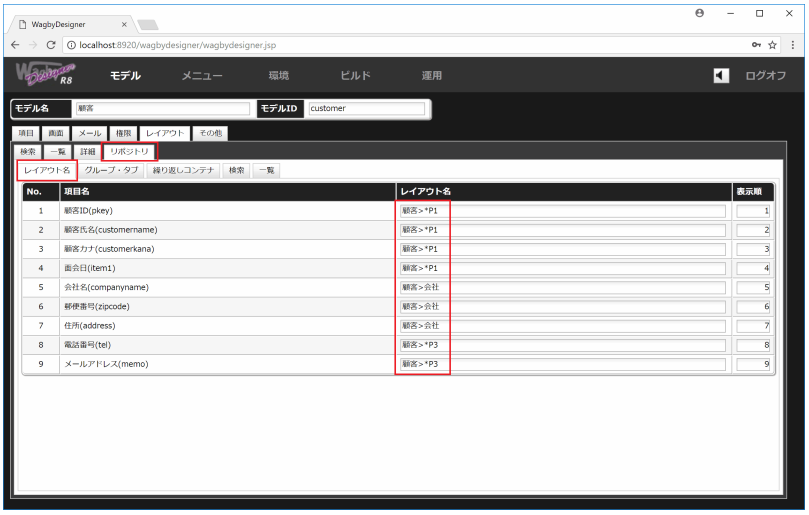


図 レイアウト名の確認

グループの表示方法と並び方向も設定が反映されていることを確認できます。

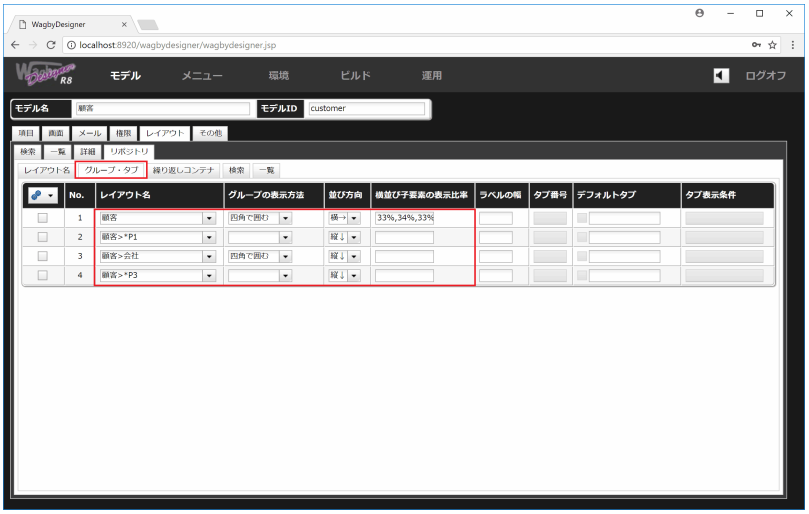


図 グループの表示方法の確認

隠しレイアウトへの入れ子レイアウト

隠しレイアウトを横並び（比率50%, 50%）とし、その中にレイアウトを入れ子にして設定しています。

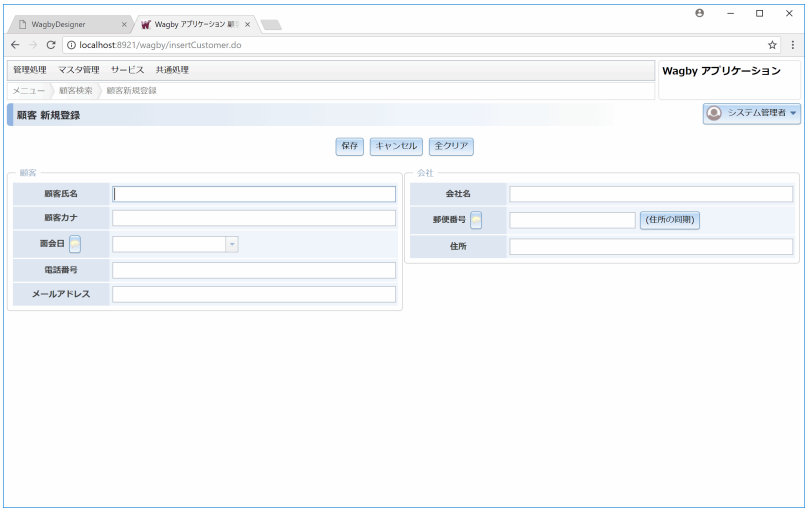


図 隠しレイアウトの中に入れ子構造でレイアウトを横並びで配置する

レイアウト設計

初期状態のレイアウトを図とします。

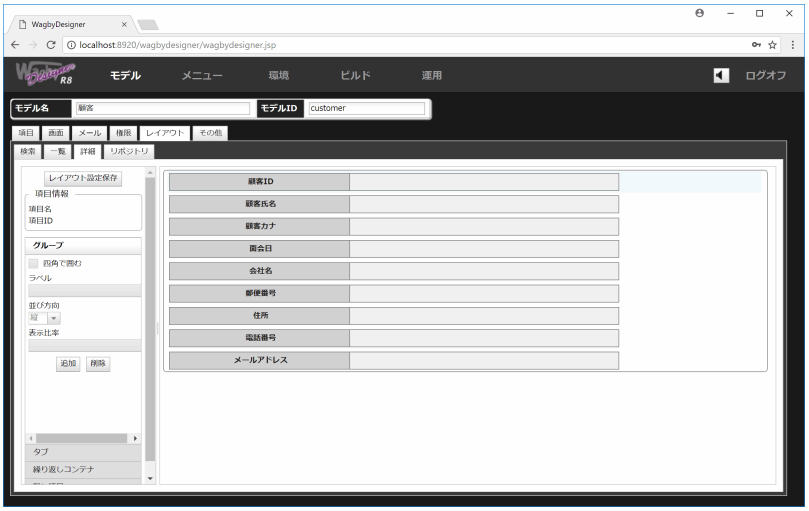


図 初期状態のレイアウト

「顧客」グループと「会社」グループを用意します。

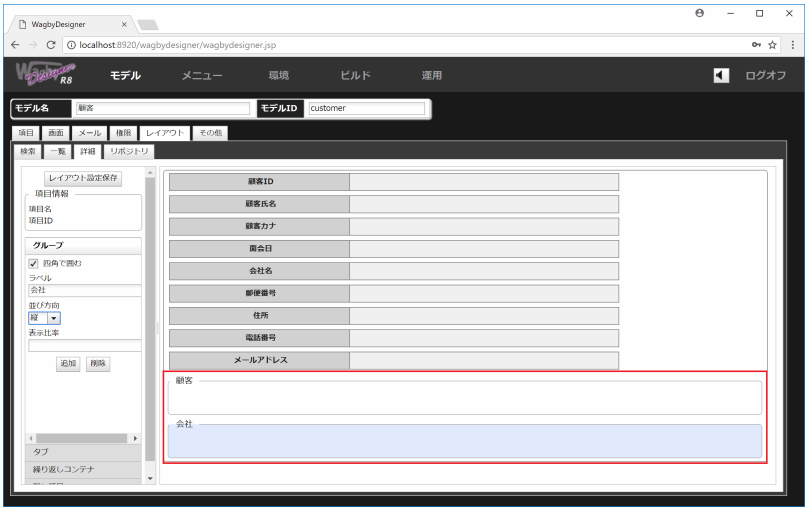
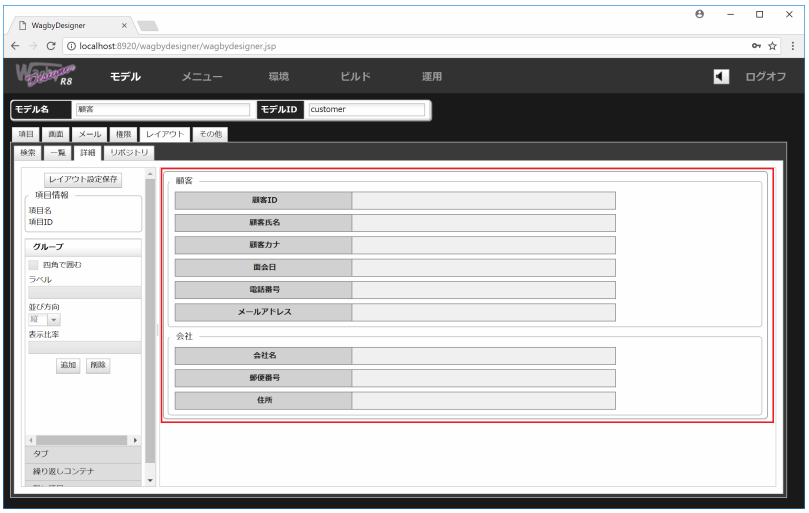


図 「顧客」グループと「会社」グループを用意する



それぞれのグループに項目を配置します。

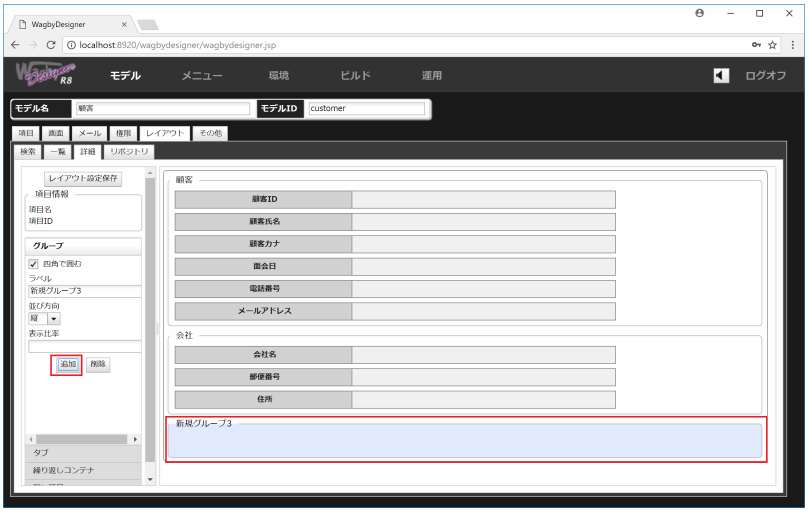


図 グループ内に項目を配置する

全体を収容するグループを作成します。

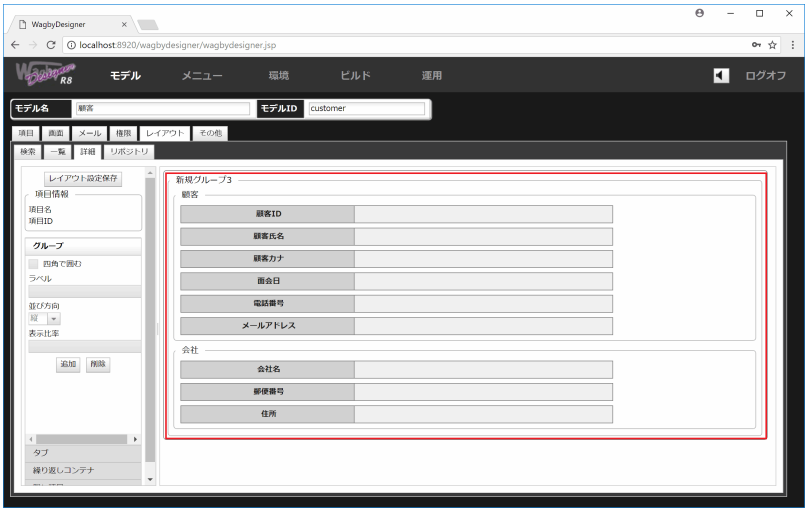


図 もう一つのグループを用意する

全体を収容するグループのラベルを空白とします。横並びで表示比率を50%, 50%とします。

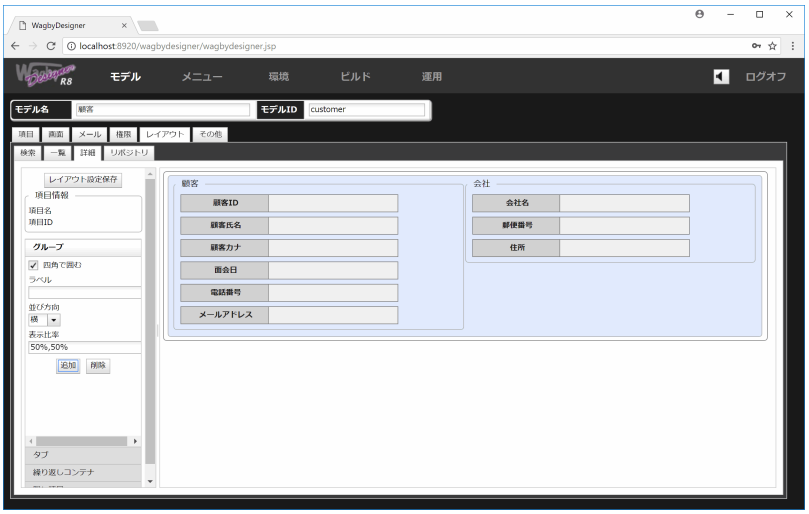


図 ラベル、並び、表示比率の設定

スクリーンエディタの編集を保存する

左側の設定欄の上部にある「レイアウト設定保存」ボタンを押下し、編集内容を保存します。この操作により、リポジトリに反映されます。

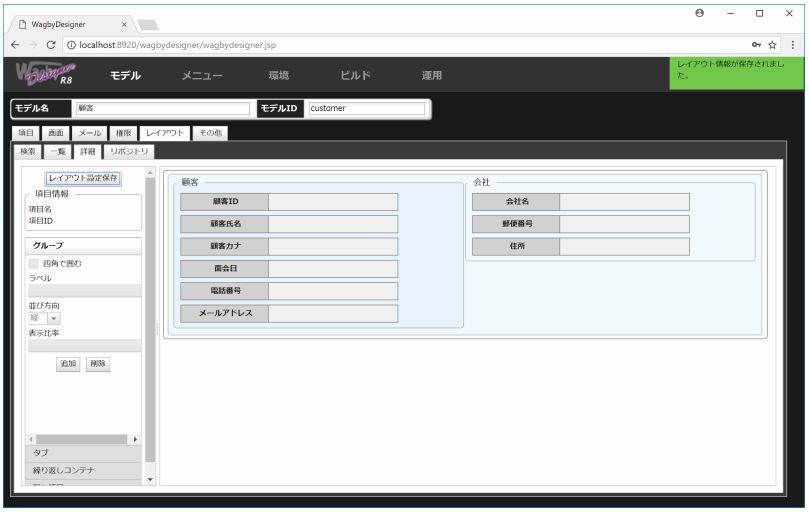


図 レイアウト設定を保存する

もっとも外側のレイアウトを隠しレイアウトにする

今回の設定は、さらにこのあとリポジトリを直接、編集する箇所があります。「リポジトリ」タブを開きます。レイアウト名はそのまま利用します。

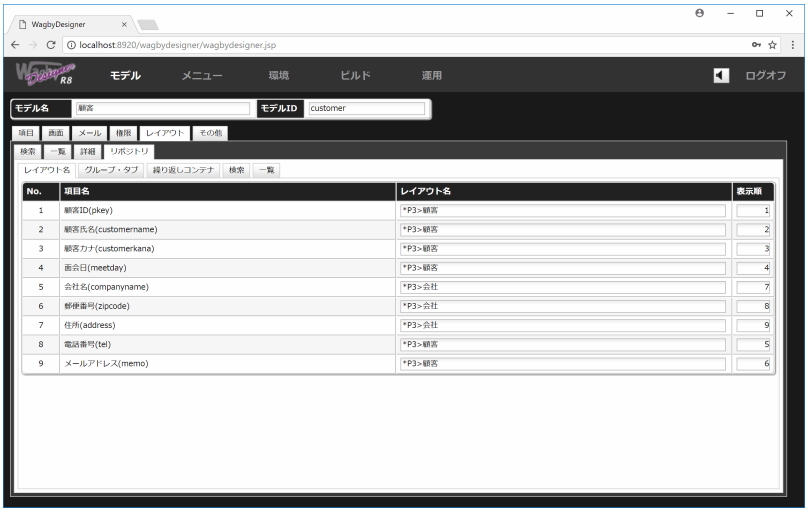


図 リポジトリの内容を確認する(1)

「グループ・タブ」タブを開きます。図のようになっています。

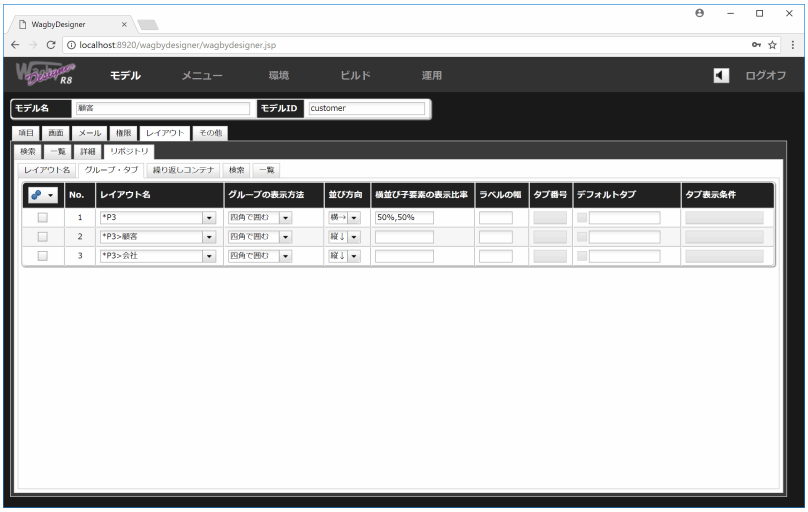


図 リポジトリの内容を確認する(2)

ラベル幅を「0%」と明記します。

(176)

(176) 隠しレイアウト（ここでは *P3）の内部のレイアウトがすべて「ラベル幅0%」の場合、隠しレイアウトを囲う枠が非表示になります。

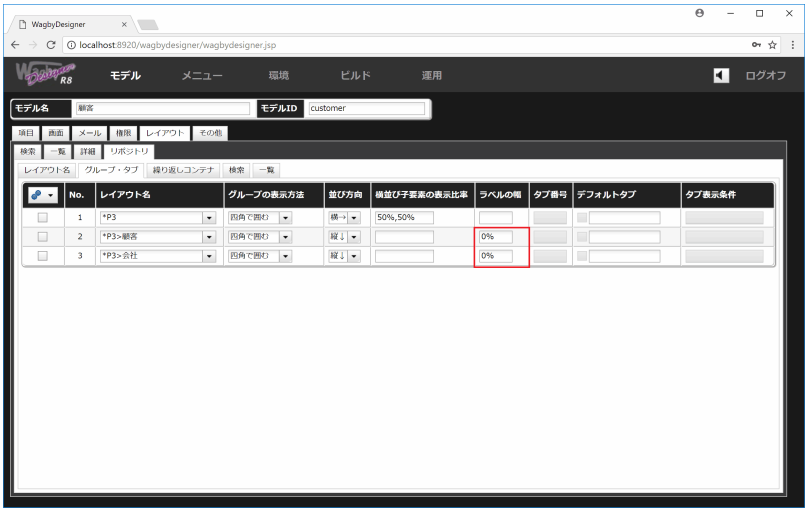


図 グループのラベル幅を手動で修正する

レイアウトラベルを独立して用意する

縦並び

レイアウトのラベルを独立して用意した例を図に示します。

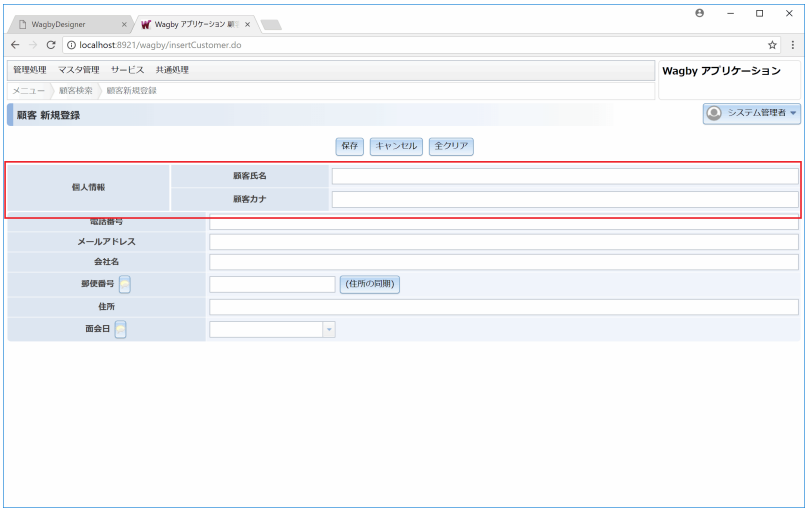


図 レイアウトラベルを独立して用意する

レイアウトの定義

「個人情報」というレイアウトを用意します。

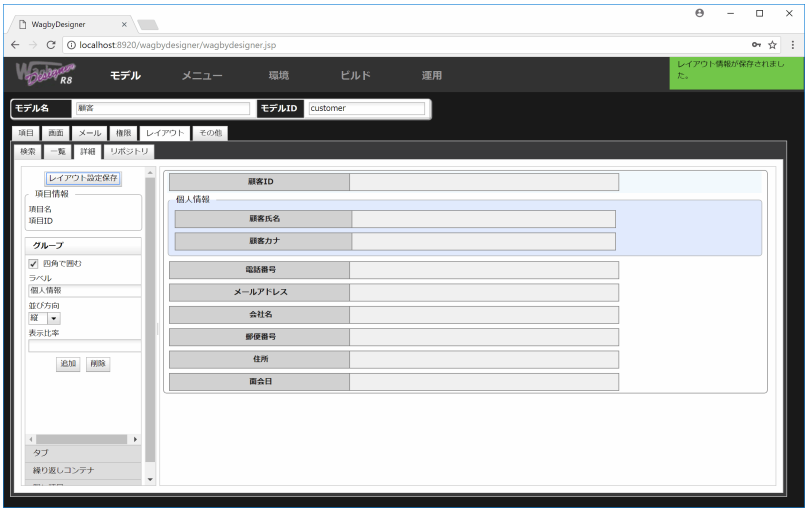


図 レイアウトを用意する

リポジトリの編集

「リポジトリ」タブを開きます。「レイアウト名」タブには、用意した「個人情報」が含まれています。ここはそのまま利用します。

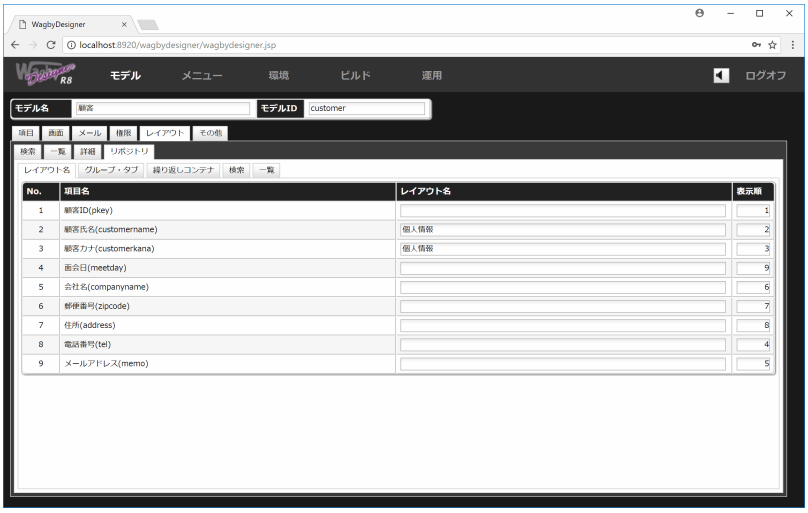


図 レイアウト名の確認

「グループ・タブ」を開きます。グループの表示方法を「縦方向にグループ化」に変更します。

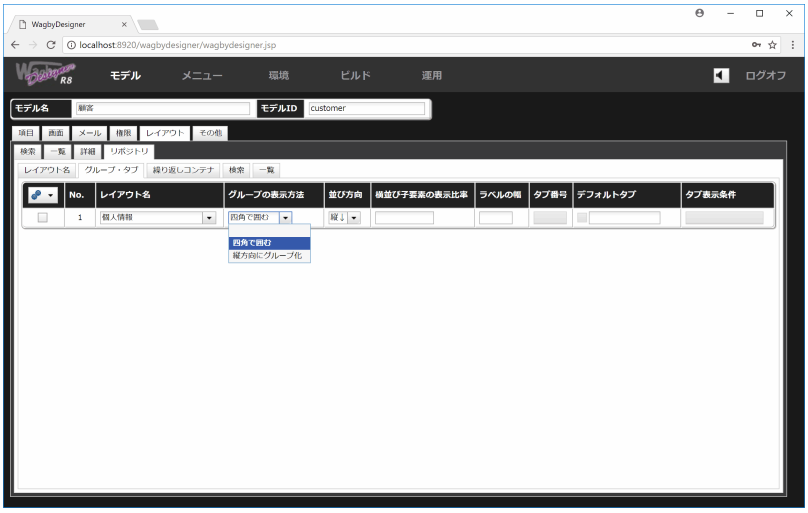


図 「縦方向にグループ化」を設定する(1)

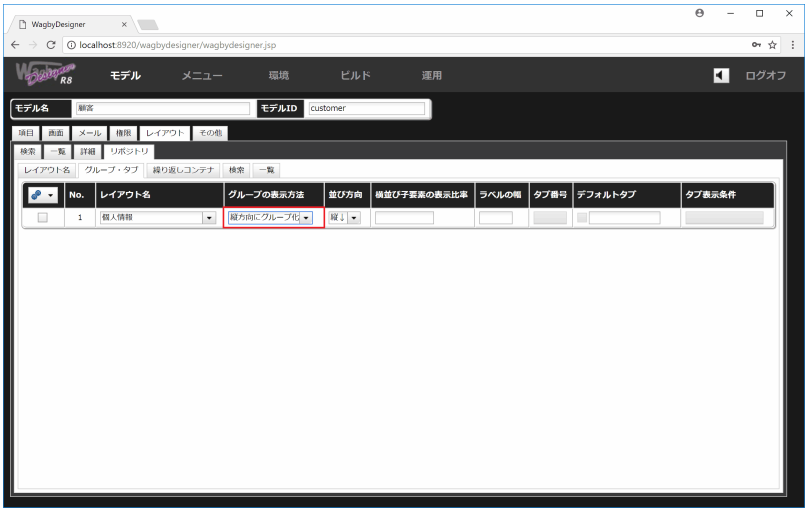


図 「縦方向にグループ化」を設定する(2)

横並び

レイアウトのラベルを独立し、横並びレイアウトを適用した例を図に示します。

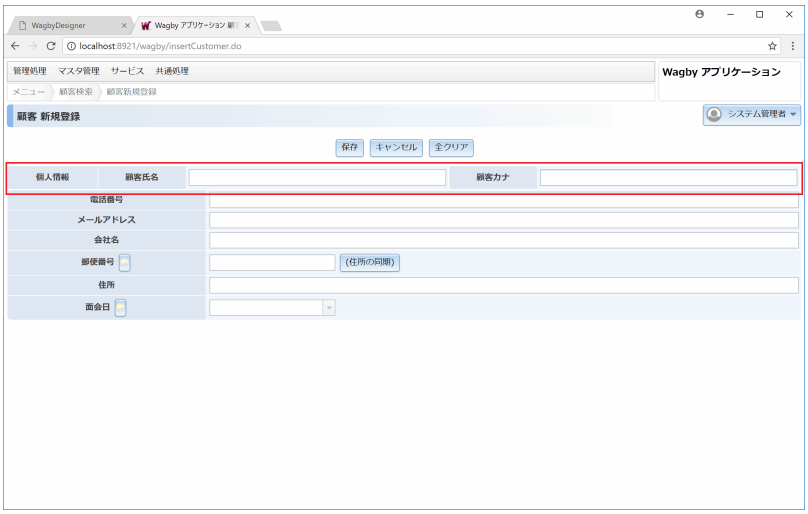


図 レイアウトラベルを独立させ、横並びレイアウトを適用する

リポジトリの編集

「個人情報」レイアウトのグループの表示方法を「縦方向にグループ化」とします。また、並び方向を「横」とし、表示比率を指定します。並び比率は50%, 50%としています。

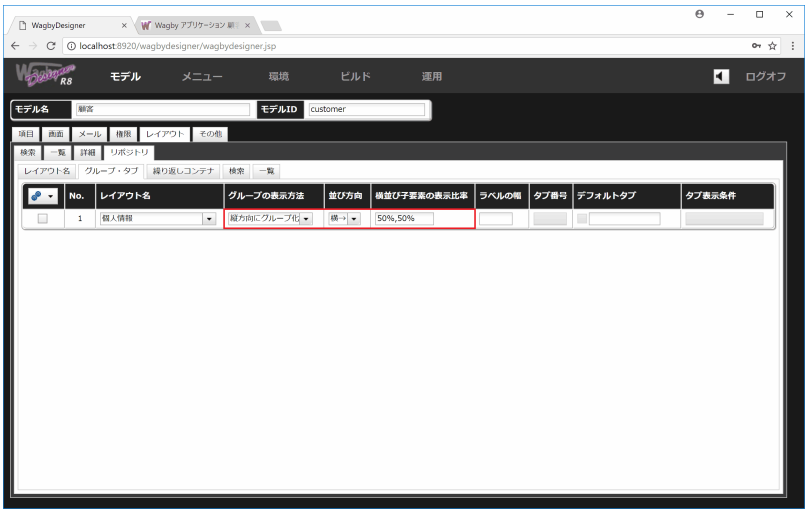


図 リポジトリを編集する

組み合わせ

横並びと縦並びのグループを用意し、それぞれレイアウトのラベルを独立した例を図に示します。

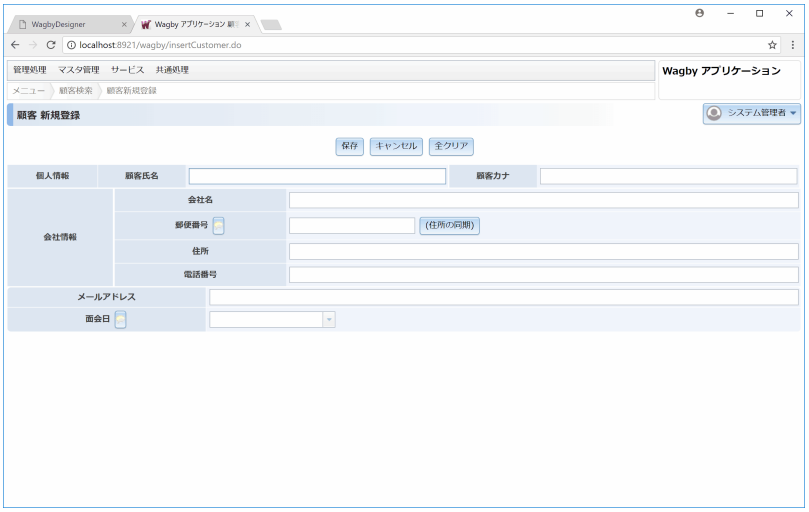


図 二つのレイアウトを並べて表示

レイアウトの定義

「個人情報」と「会社情報」の二つのレイアウトを用意します。

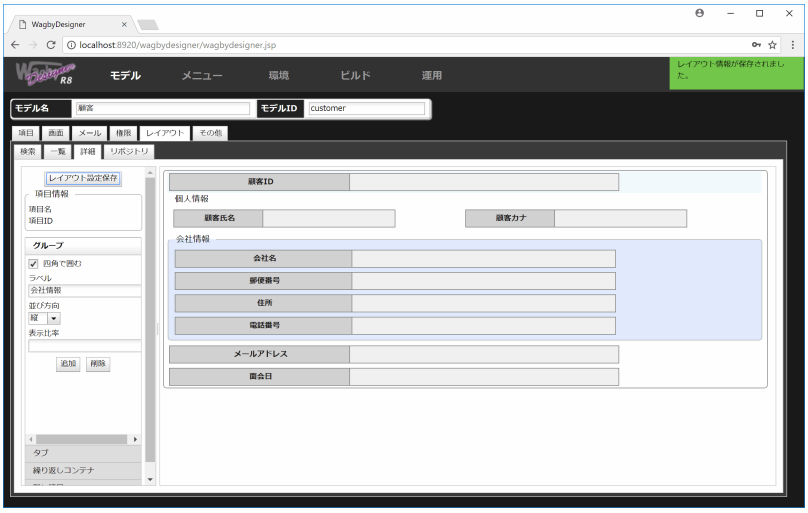


図 レイアウトの定義

リポジトリの編集

それぞれのレイアウトを「縦方向にグループ化」とします。

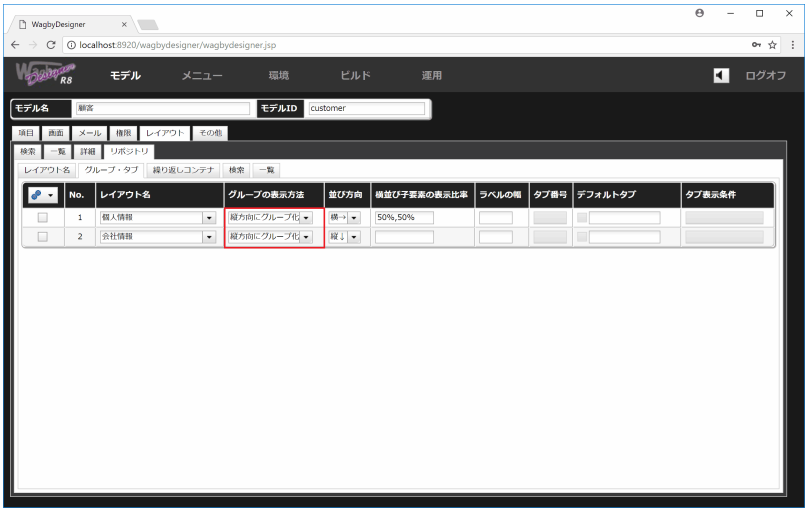


図 縦方向にグループ化を指定する

ラベル幅を合わせる

ラベル幅を揃えた例を図に示します。ラベル幅を100(px)としています。

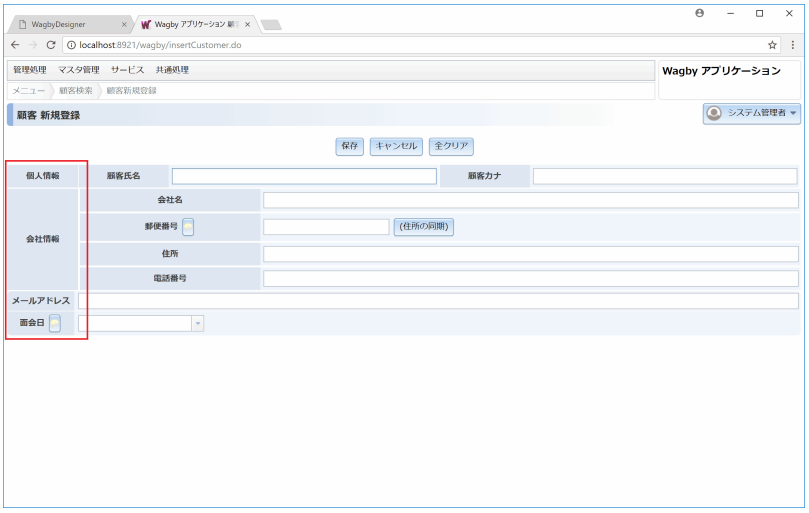


図 ラベル幅を揃える

リポジトリの編集

レイアウトのラベル幅を指定します。ここでは「100」とします。数値のみを記述してください。単位はピクセル(px)です。

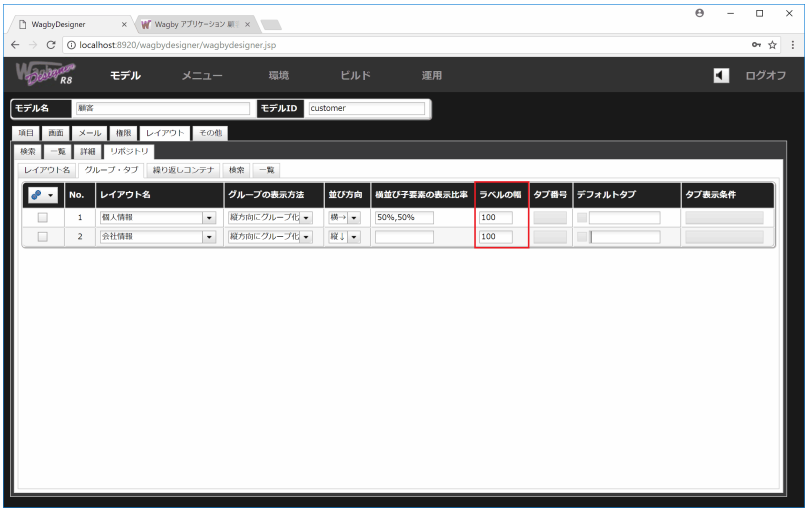


図 レイアウトのラベル幅

あわせて、レイアウト外の「面会日」「メールアドレス」項目のラベル幅も調整します。 「出力制御>詳細画面」タブにある「スタイル>ラベルの幅」を指定します。数値のみを指定すると、単位がピクセル(px)と解釈されます。

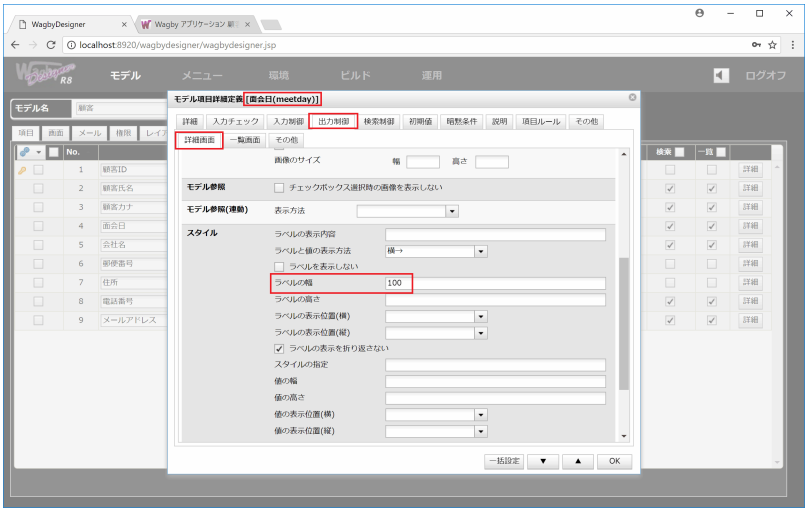


図 モデル項目詳細定義

ウィザードとタブレイアウト

ウィザードとタブレイアウトの違い

ウィザード

顧客モデルにウィザードを適用し、二画面に分割した例を図に示します。画面移動ボタン（「前画面へ」「次画面へ」）で前後のページに移動します。さらに画面指定ボタン（「画面1」「画面2」）を使って直接、特定の画面を開くこともできます。

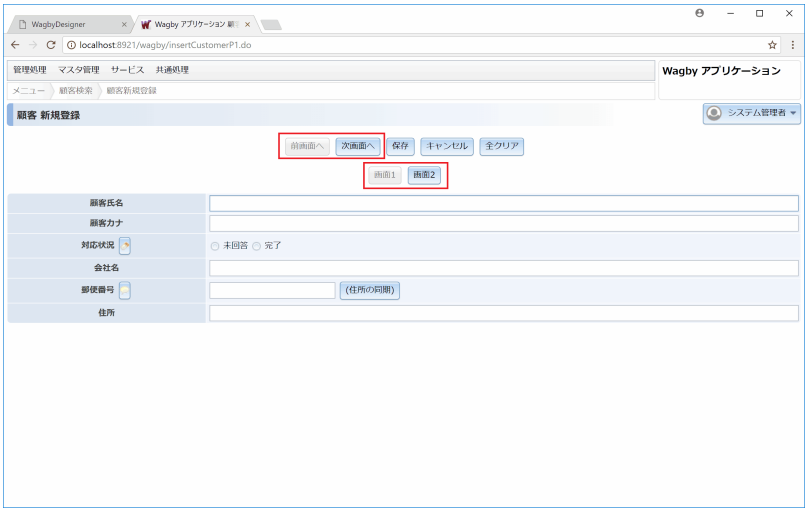


図 ウィザードの画面例

タブレイアウト

2つのタブを用意した例を示します。

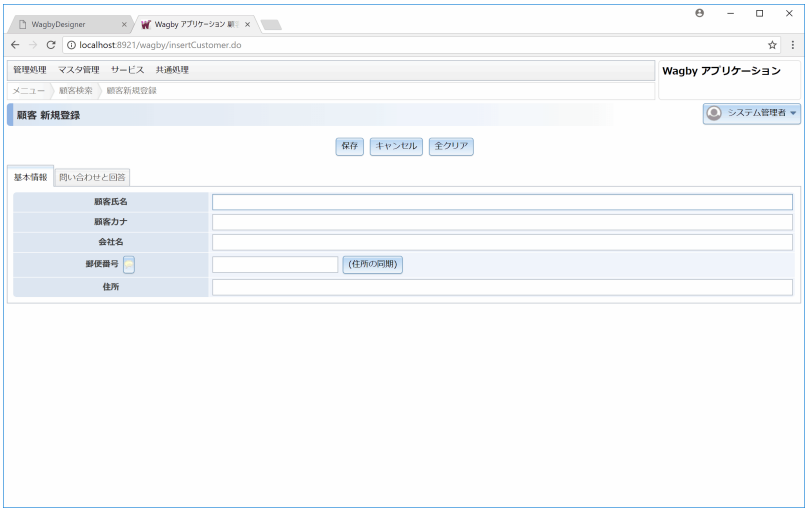


図 タブレイアウトの利用(1)

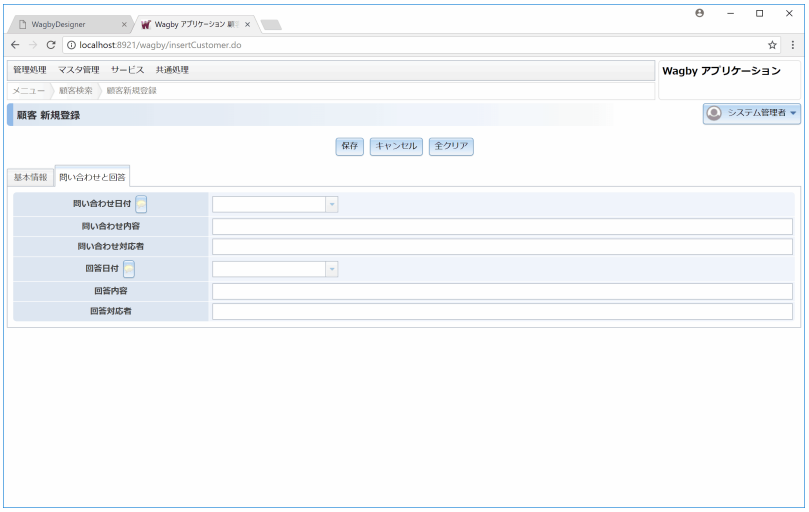


図 タブレイアウトの利用(2)

ウィザードとタブレイアウトの違い

ウィザード

順番に従って入力を行なう方式です。順番に、業務のルール（意図）が含まれます。ある画面に必須項目が用意されていた場合、同項目を入力しなければ、次の画面に進むことができません。

タブレイアウト

入力項目をグループ化したものです。入力の順番を意識する必要はありません。

ウィザードの詳細および定義方法は"画面機能 > ウィザード"をお読みください。 以降はタブレイアウトについて説明します。

タブレイアウトを設定する

図に示す、サンプルの顧客モデル定義で説明します。

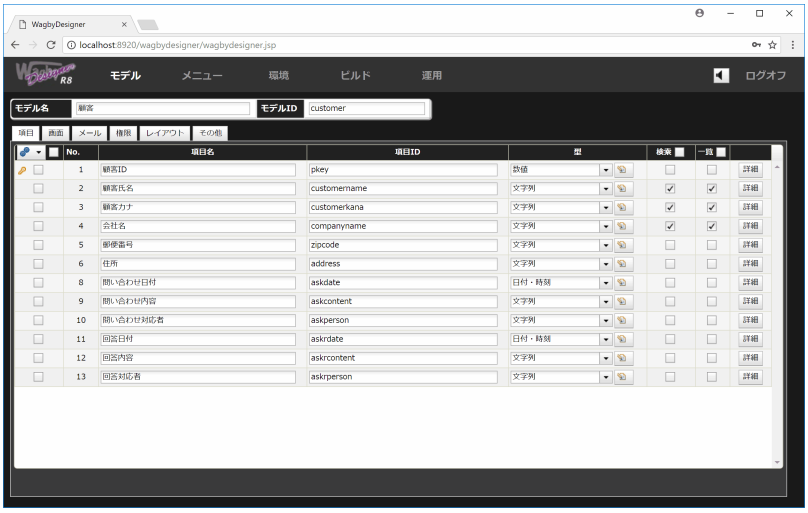


図 顧客モデル（サンプル）

「レイアウト>詳細」タブを開きます。初期状態は定義順に項目が並んでいます。

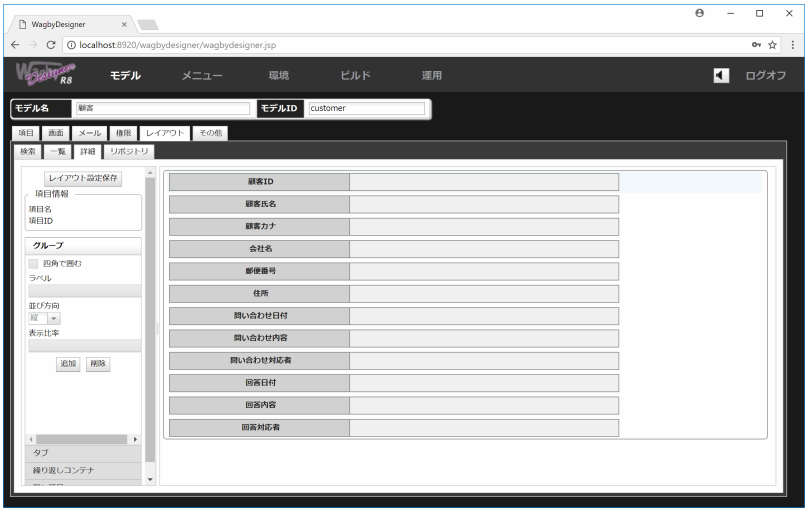


図 初期のレイアウト

左側の欄から「タブ」を選択し、その中にある「タブコンテナ追加」ボタンを押下します。画面右側のスクリーンエディタ部に新しいタブが用意されます。

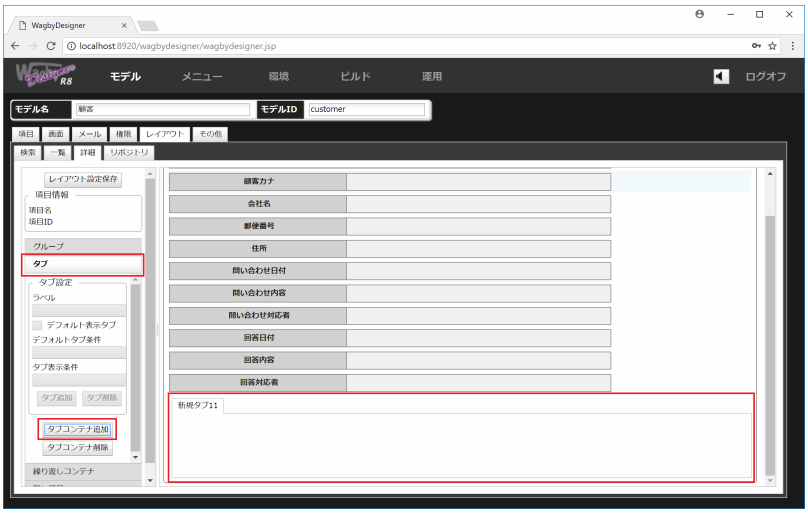


図 タブコンテナを追加する

右側のスクリーンエディタからタブを選択し、ラベル（タブの名称）を入力します。ここでは″基本情報″としました。

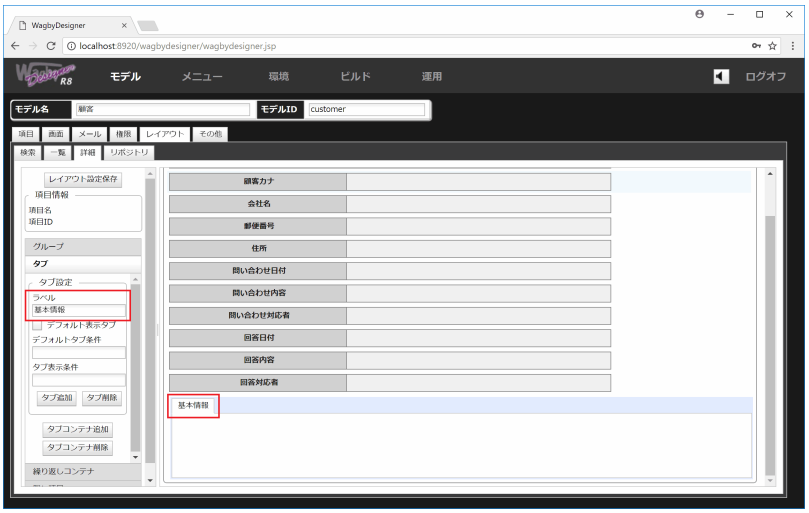


図 ラベルを指定する

項目をドラッグ&ドロップして、タブ内に項目を配置することができます。

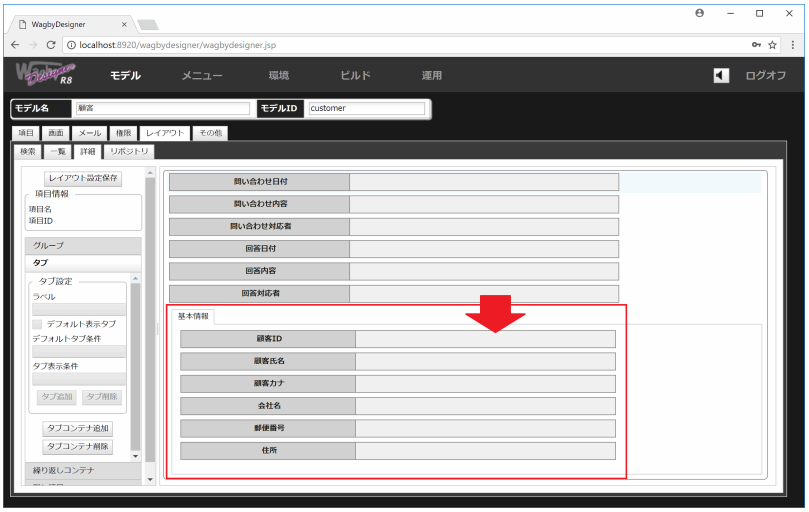


図 タブ内に項目を配置する

このタブコンテナに、複数のタブを用意することができます。スクリーンエディタでタブコンテナを選択後、「タブ追加」ボタンを押下します。

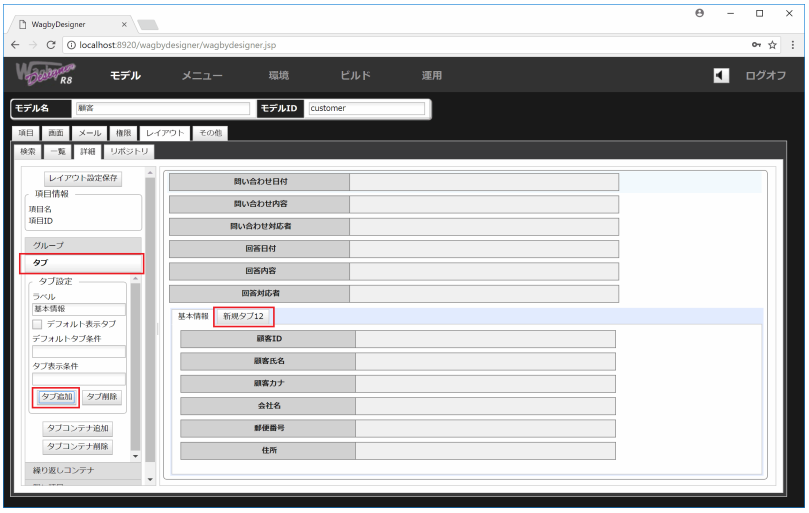


図 タブを追加する

タブ毎にラベルを指定できます。

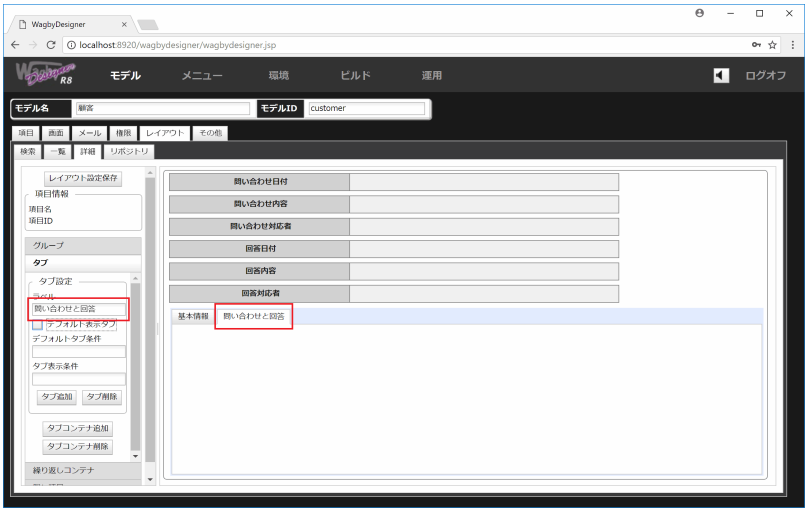


図 追加したタブにラベルを設定する

同じように項目をドラッグ&ドロップして、タブ内に配置します。

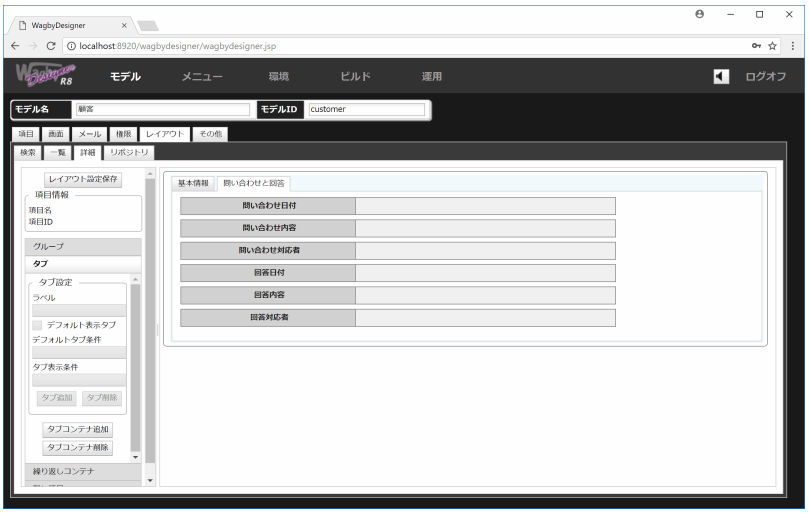


図 新しいタブに項目を配置する

タブからタブへの項目の配置を行うことはできません。いったんタブの外へ項目を移し、それから移動先のタブを選択し、項目を移動してください。

スクリーンエディタの編集を保存する

左側の設定欄の上部にある「レイアウト設定保存」ボタンを押下し、編集内容を保存します。この操作により、リポジトリに反映されます。

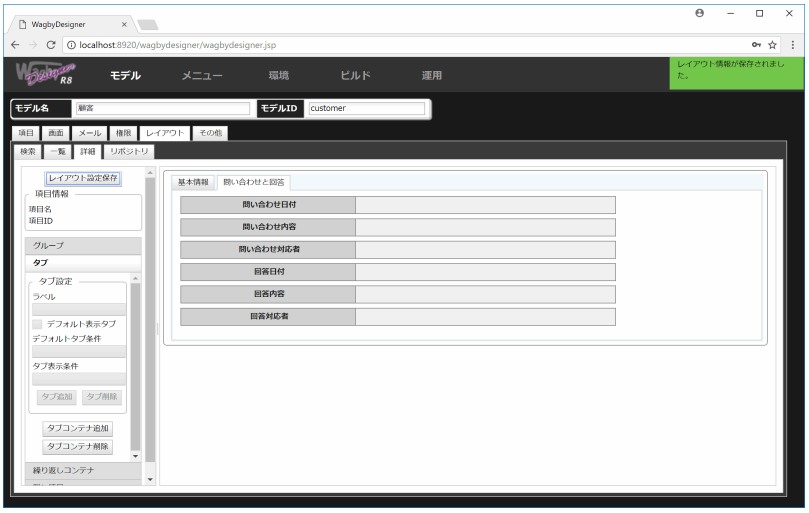


図 レイアウト設定を保存する

ご注意ください。「レイアウト設定保存」ボタンを押下せずに他の画面に切り替えた場合、これまでの編集内容は反映されません。

リポジトリの確認

ここまでの設定が、どのようにリポジトリに反映されたかを確認します。「リポジトリ」タブを開きます。「レイアウト名」タブには、タブのラベル名として入力した値が反映されていることがわかります。

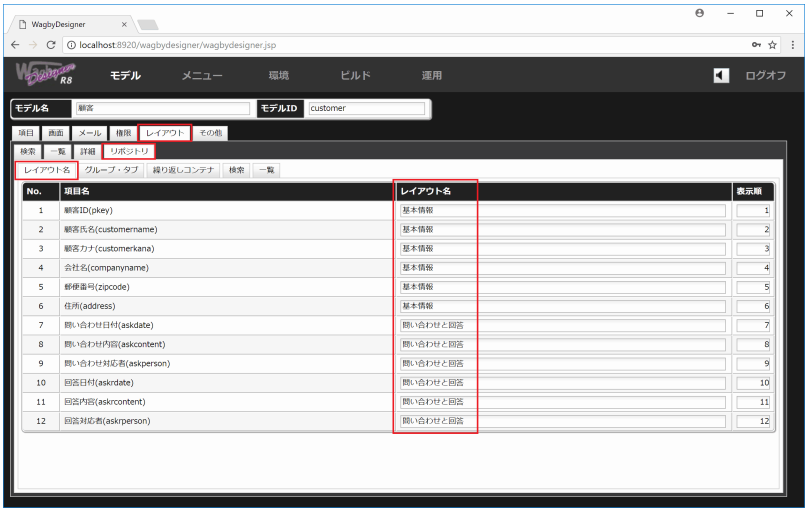


図 リポジトリの確認(1)

「グループ・タブ」の内容を確認します。並び方向が「タブ」となっています。

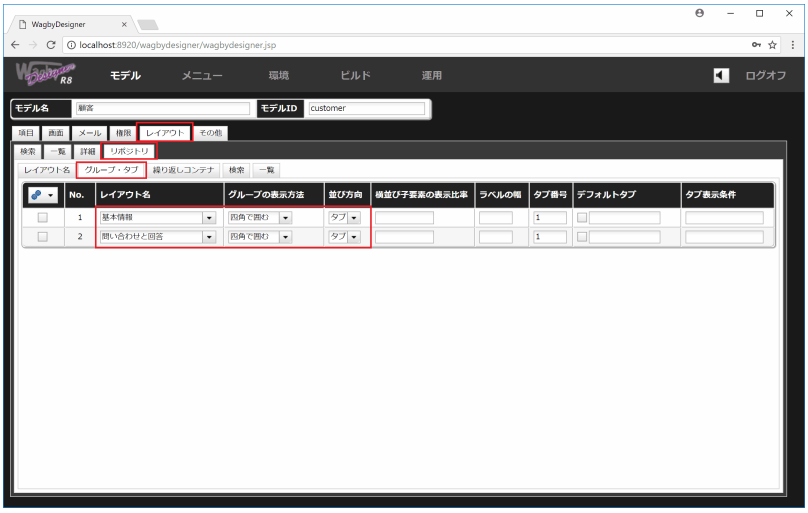


図 リポジトリの確認(2)

開発者はリポジトリを直接、編集することもできます。ここで編集した内容は、スクリーンエディタに反映されます。

デフォルトタブの指定

デフォルトのタブを指定することができます。図は新規登録画面を開いた例です。デフォルトタブ「問い合わせと回答」が最初に表示されています。

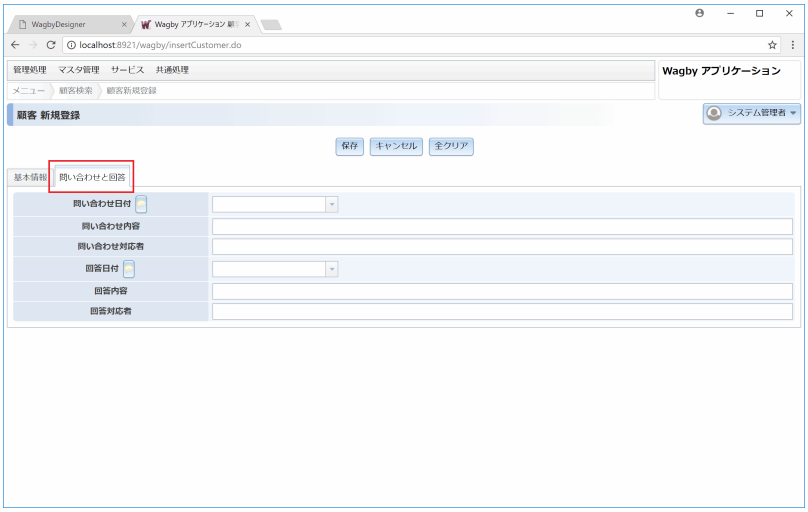


図 デフォルトタブが有効になっている

レイアウト設計

デフォルトタブをチェックします。デフォルトを指定できるタブは一つです。

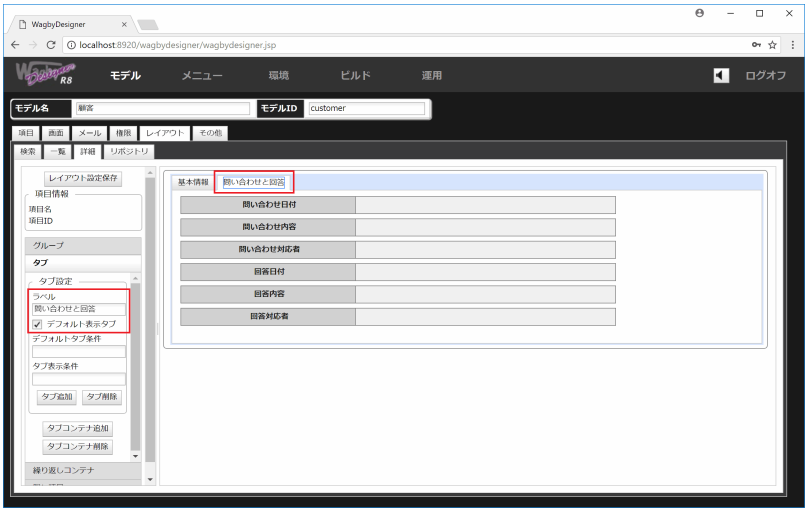


図 デフォルトタブの設定

リポジトリの確認

リポジトリを確認します。「デフォルトタブ」がチェックされています。

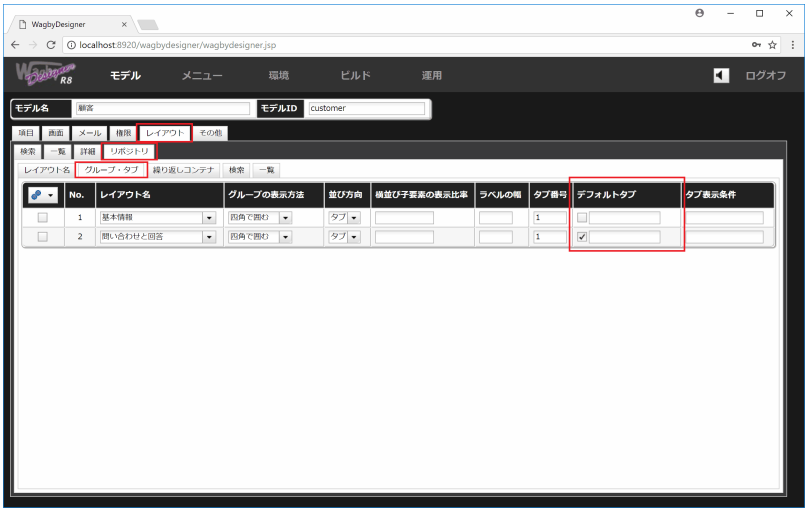


図 リポジトリの確認

項目の値によってデフォルトタブを制御する

条件に合致した場合にデフォルトタブとする指定を行うことができます。優先度は 1. 条件に合致したタブ、2. デフォルトタブ、となります。

図に例を示します。このモデルは「問い合わせ」タブをデフォルトタブと指定しています。新規登録画面を開くと、問い合わせタブが最初に選択されます。

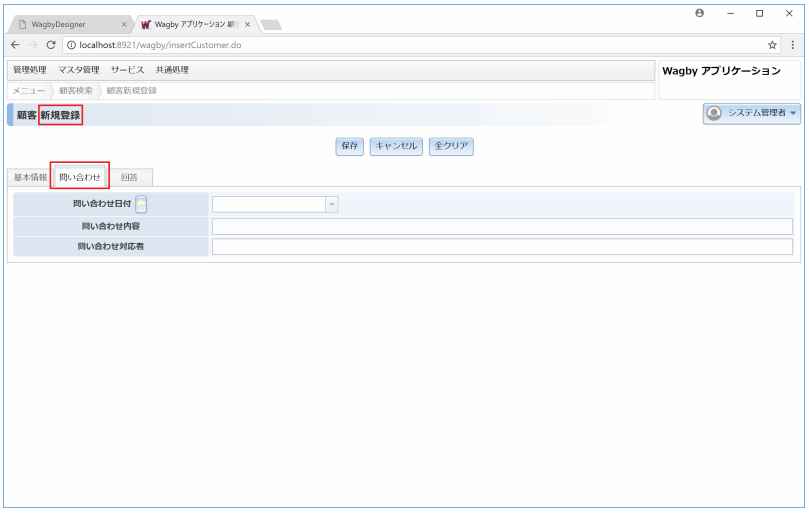


図 デフォルトタブの動作(1)

ここでは「対応状況」項目が〝未回答〟のとき、デフォルトタブを「回答」とするように設定しました。

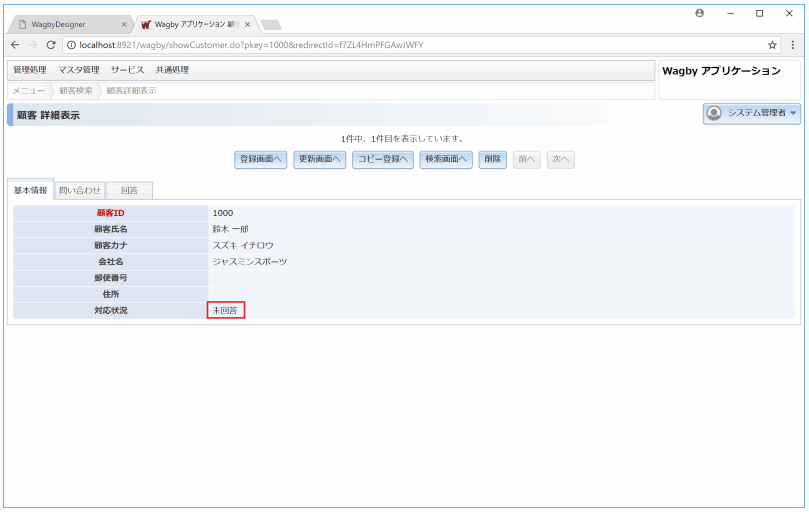


図 デフォルトタブの動作(2)

更新画面を開くと、デフォルトタブが「回答」に切り替わったことがわかります。

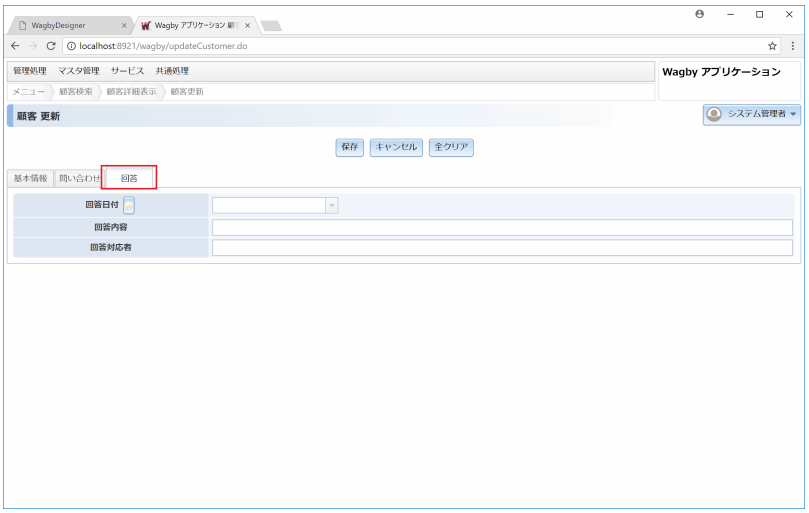


図 デフォルトタブの動作(3)

準備

ここではラジオボタンの値 （ID部） によってデフォルトタブを制御する例を説明します。図の項目を追加します。

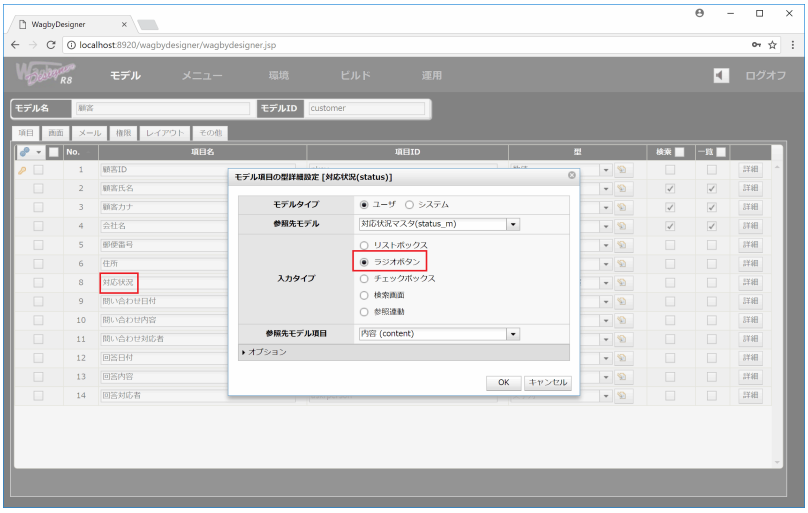


図 対応状況項目を追加する

参照先の選択肢モデル「対応状況マスタ」は次のように設定しました。

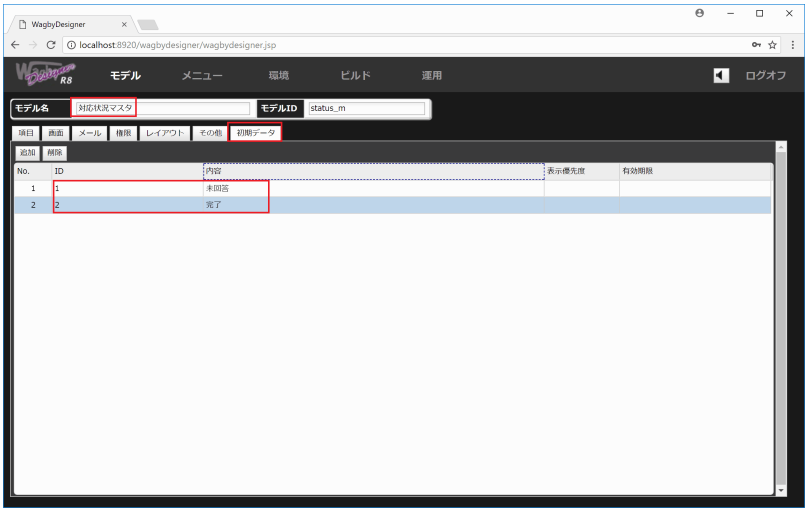


図 対応状況マスタモデル

レイアウト設計

″基本情報″,″問い合わせ″,″回答″ という3つのタブを用意します。

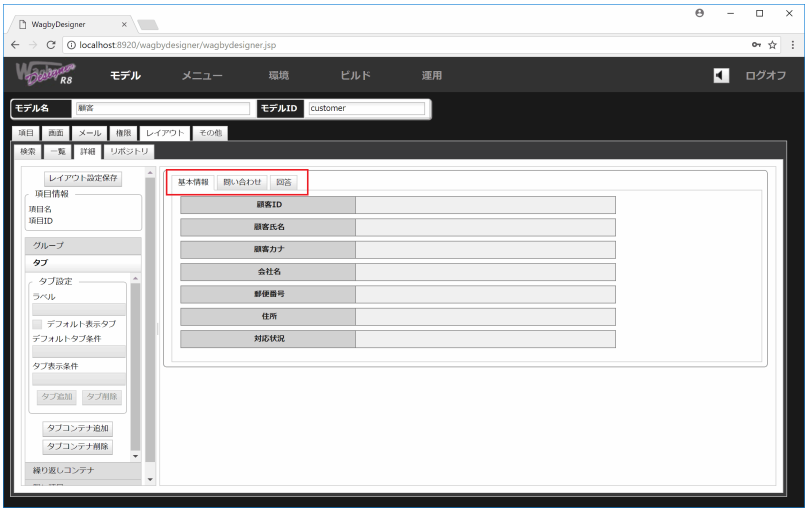


図 3つのタブを用意する

″問い合わせ″　タブをデフォルトと指定しておきます。条件が合致しない場合、この設定が有効になります。

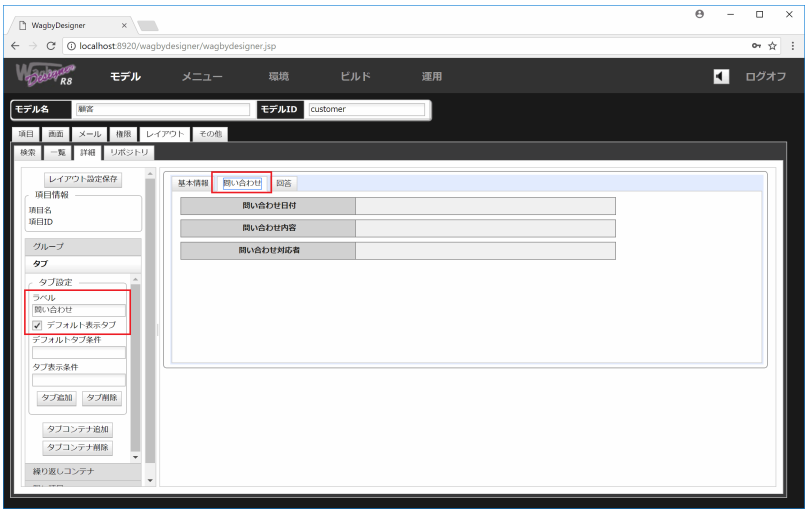


図 デフォルトタブを指定する（条件が合致しない場合）

「回答」タブに条件を指定します。書式は次のようになります。

```
項目ID=値
```

今回は、status 項目の値 （ID部） が 1 のとき、すなわち″未回答″のときは「回答」タブをデフォルトとするようにしました。

このようにモデル参照項目を用いる場合、ID部のみが利用できます。

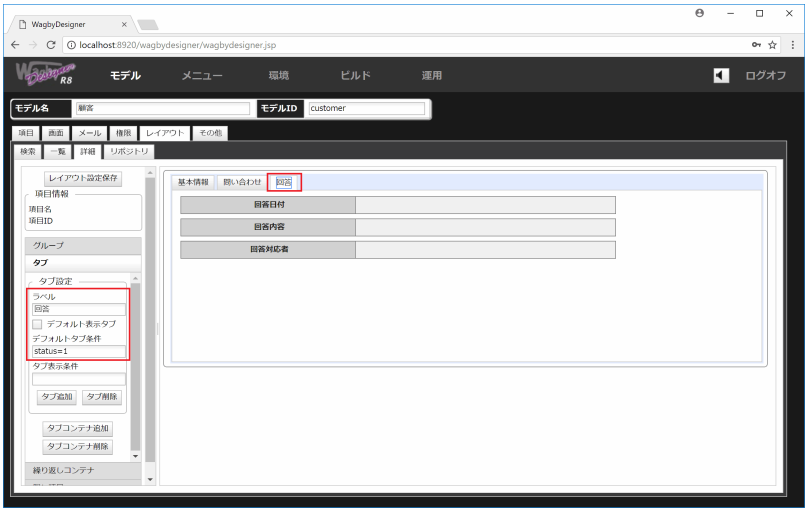


図 デフォルトタブ条件の指定

設定後、リポジトリを保存します。

リポジトリの確認

リポジトリを確認します。「デフォルトタブ」に表示条件が含まれています。

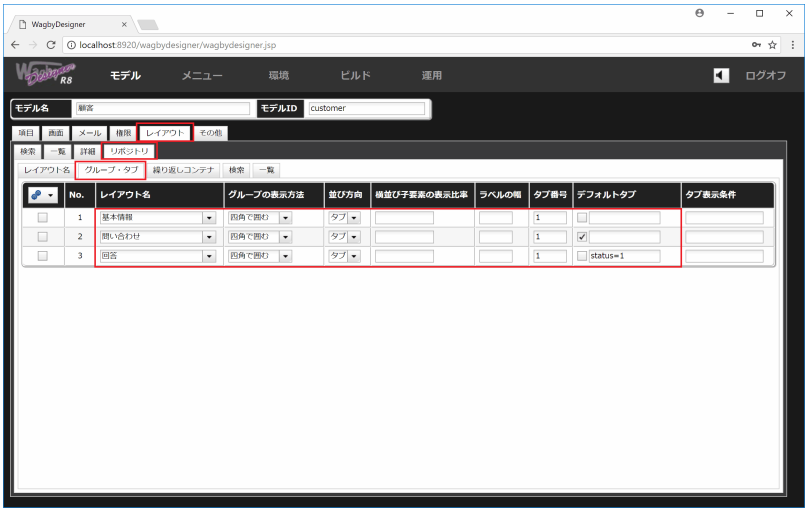


図 リポジトリの確認

タブの入れ子

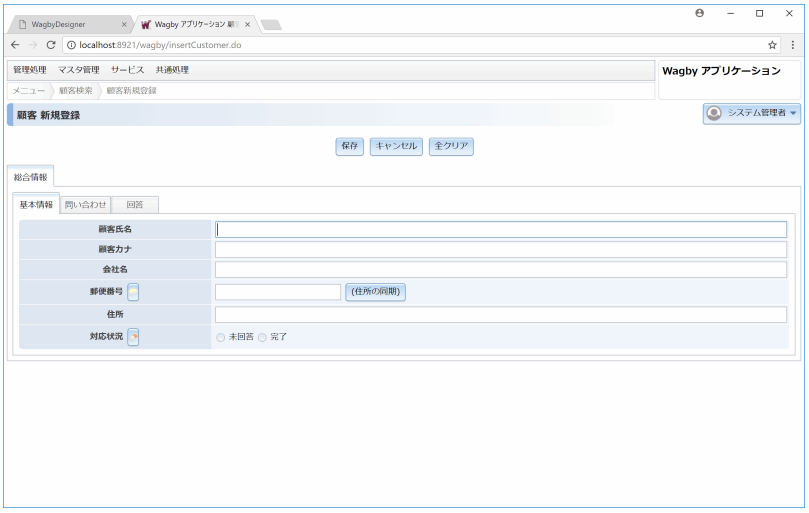


図 タブレイアウトの入れ子

レイアウト設計

スクリーンエディタで、タブの中にドラッグ&ドロップ操作でタブを含めることができます。

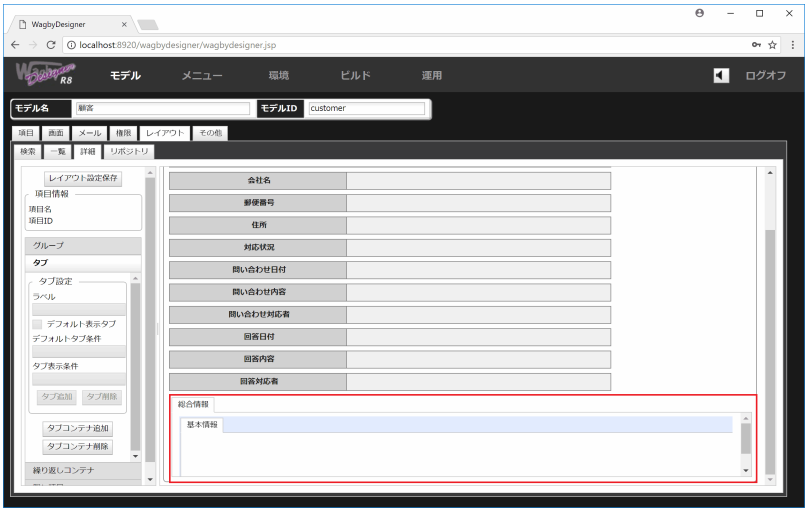


図 タブの入れ子(1)

入れ子のタブに、ドラッグ&ドロップ操作で項目を配置することができます。

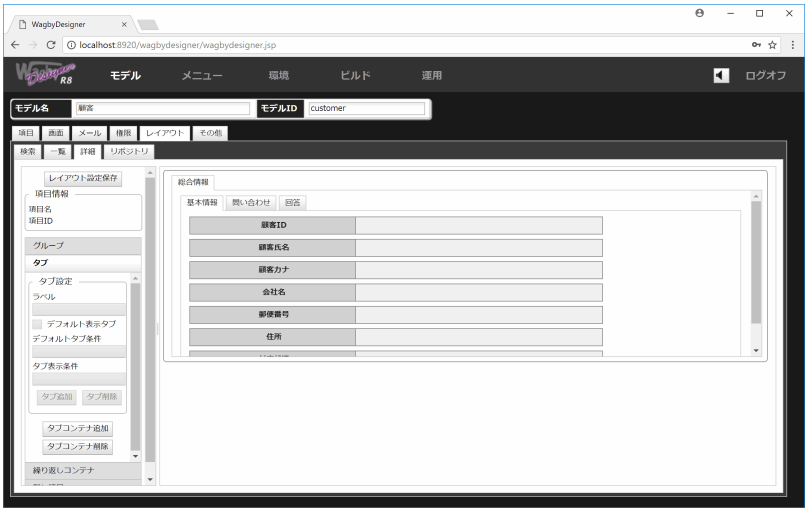


図 タブの入れ子(2)

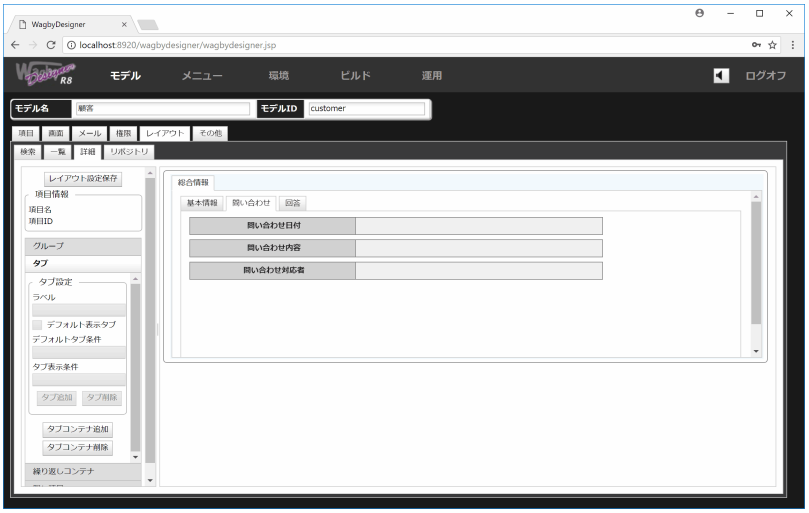


図 タブの入れ子(3)

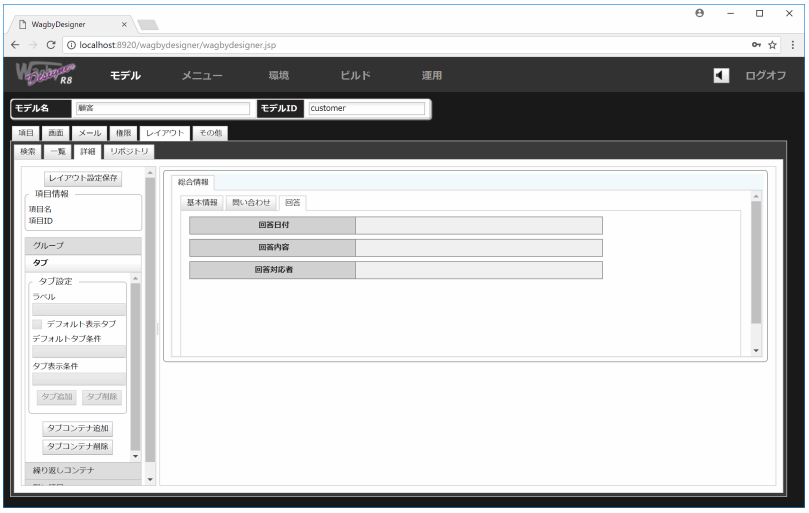


図 タブの入れ子(4)

リポジトリの確認

レイアウトの入れ子設定と同様に「>」で区切ります。

```
レイアウト>レイアウト
```

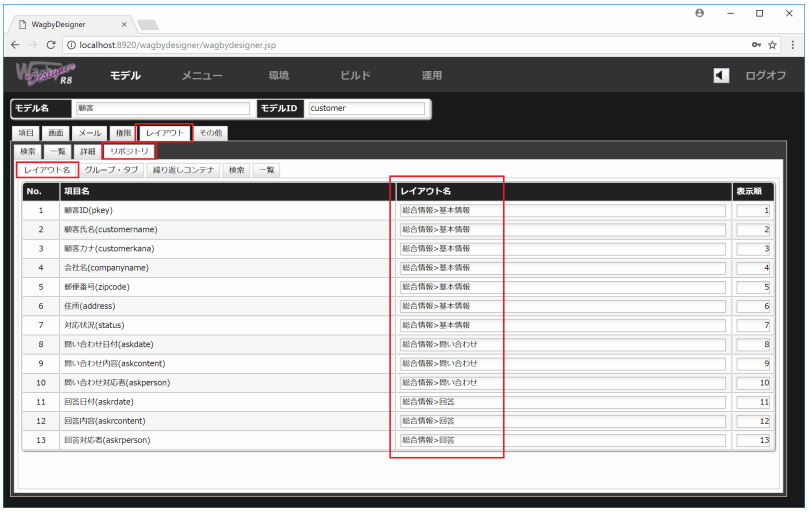


図 リポジトリの確認(1)

入れ子の場合、異なるタブ番号が割り当てられます。

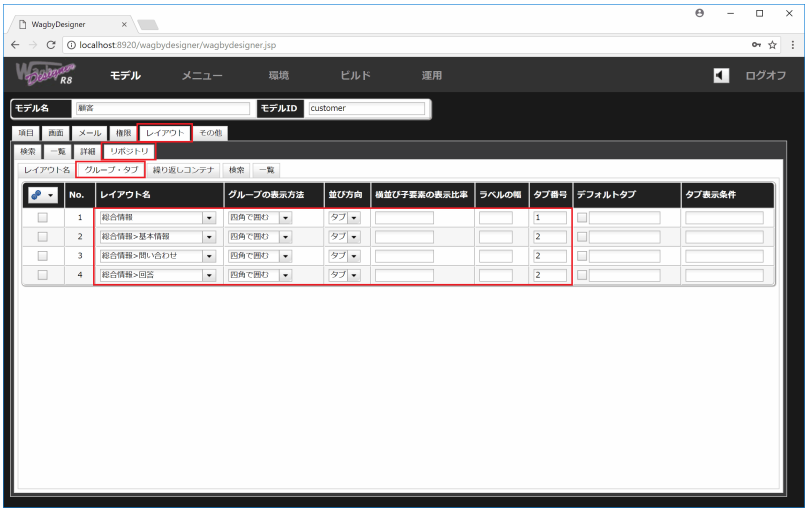


図 リポジトリの確認(2)

(177)

タブと他レイアウトとの組み合わせ

タブレイアウト内の項目に対して、縦並びや横並びといったレイアウト指定を行うことができます。

(177) 入れ子となったタブに「デフォルトタブ」を指定することもできます。 同じ階層のタブ同士でデフォルトタブをそれぞれ設定することができます。

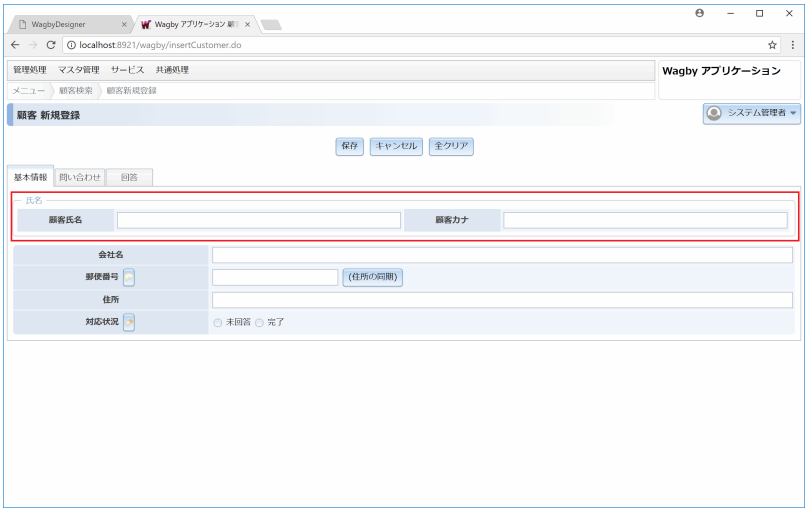


図 タブレイアウト内に横並びレイアウトを指定した例

レイアウト設計

スクリーンエディタで、タブの中にグループを作成することができます。

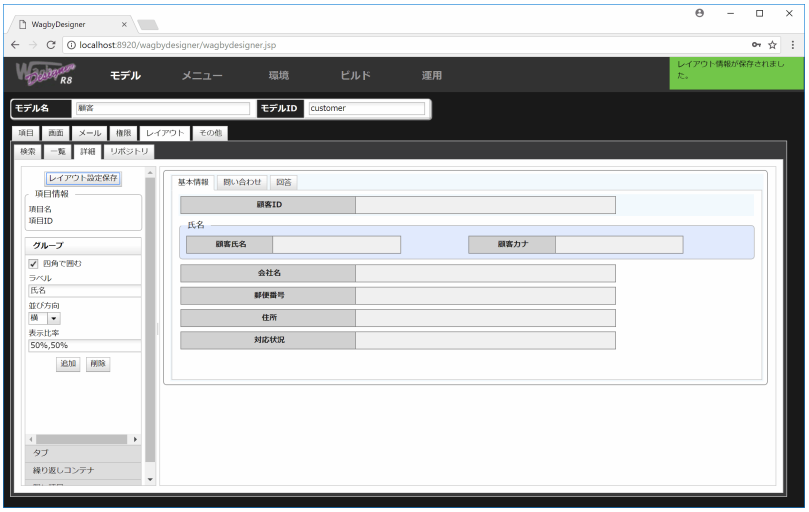


図 タブの中にグループを作成し、横並び設定を行った

リポジトリの確認

タブ名のあとに「>」文字を続けてグループ名がつながっています。

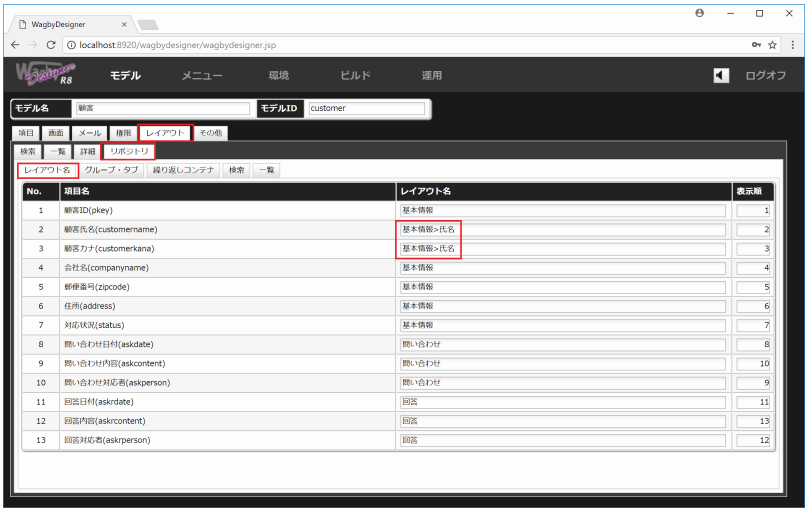


図 リポジトリの確認(1)

「グループの表示方法」と「並び方向」もそれぞれ反映されています。

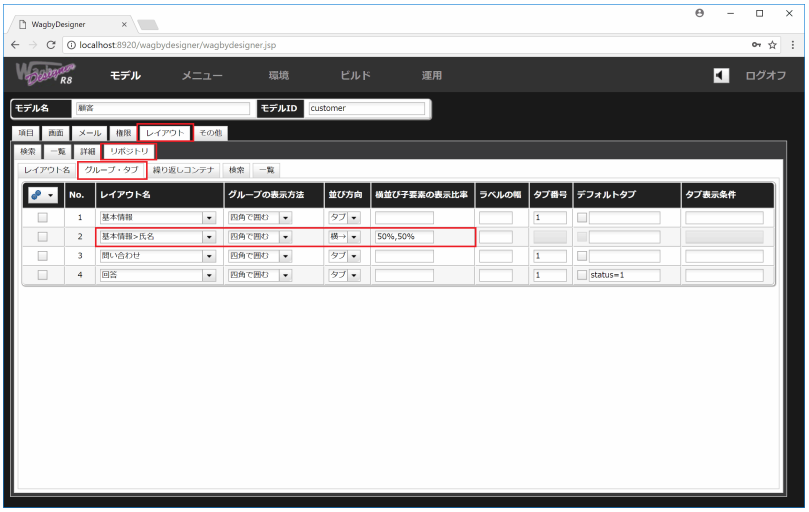


図 リポジトリの確認(2)

複数のタブコンテナを用意する

複数のタブコンテナを用意することができます。

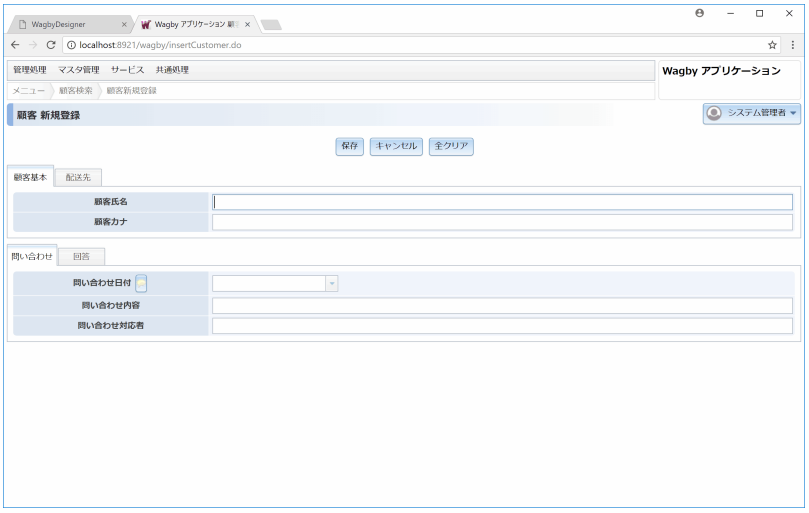


図 複数のタブコンテナを用意する

レイアウト設計

一つ目のタブコンテナを用意します。

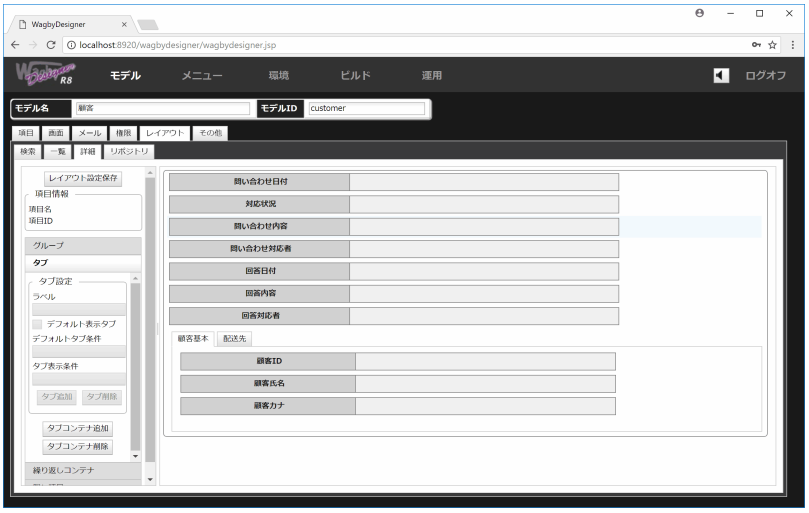


図 一つ目のタブコンテナ

二つ目のタブコンテナを作成します。

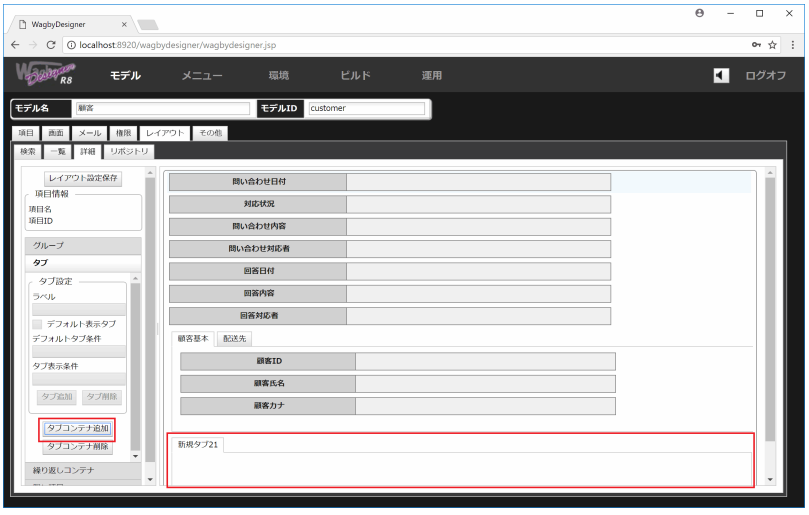
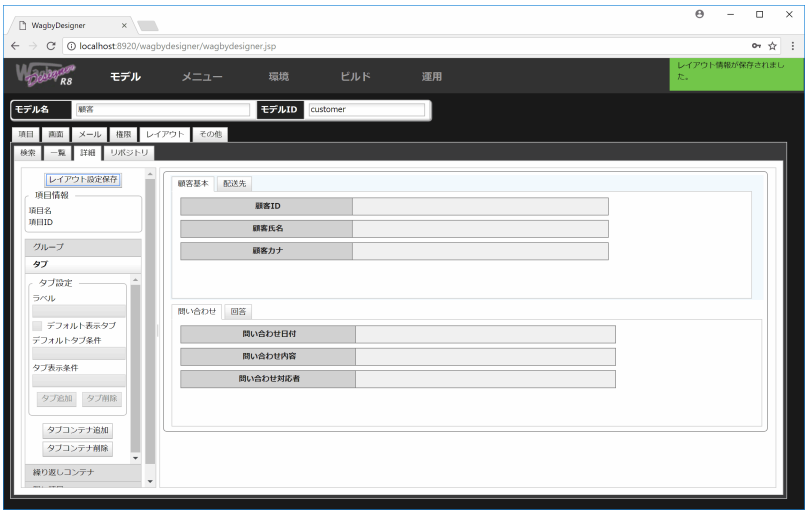


図 二つ目のタブコンテナを作成する

二つ目のタブコンテナに項目をドラッグ&ドロップして配置することができます。



「レイアウト設定保存」ボタンを押下して、設定を保存します。

リポジトリの確認

それぞれのタブを区別するために「タブ番号」が割り当てられます。同じ番号は、同じタブ（グループ）として扱われます。

(178)

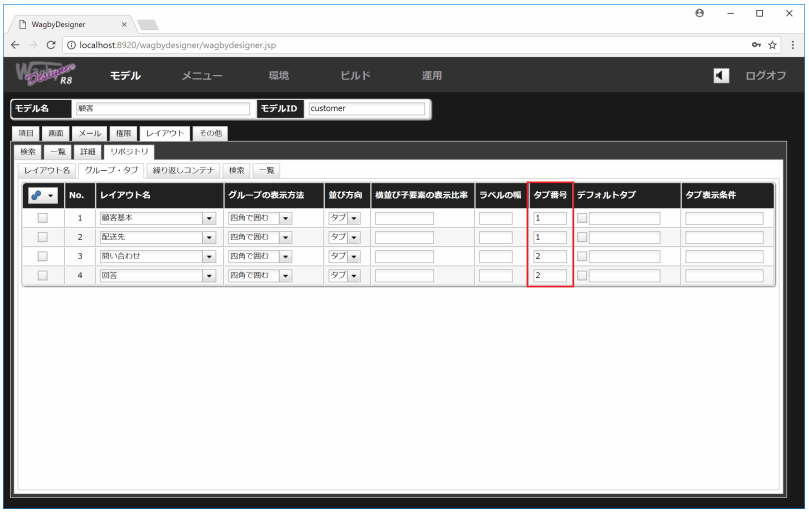


図 リポジトリの確認

タブの表示位置

タブレイアウトの表示位置をドラッグ&ドロップで変更することができます。

(178) タブ番号は自動的に割り当てられます。

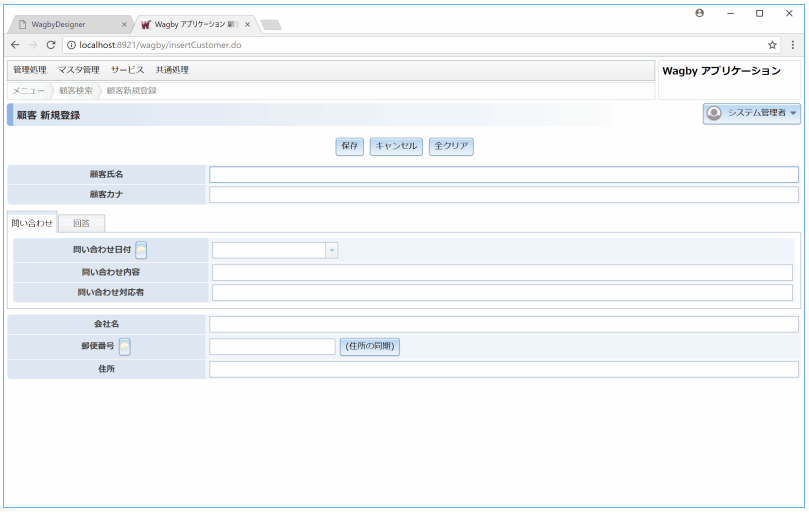


図 タブレイアウトの位置を変更した例

レイアウト設計

タブコンテナを選択し、ドラッグ&ドロップで位置を移動させてください。

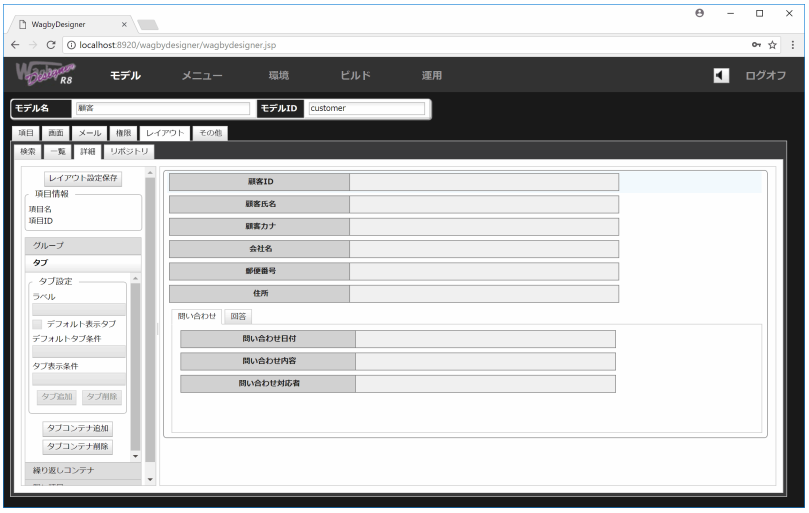


図 タブレイアウトの位置を移動する(1)

移動後は「レイアウト設定保存」を行なってください。

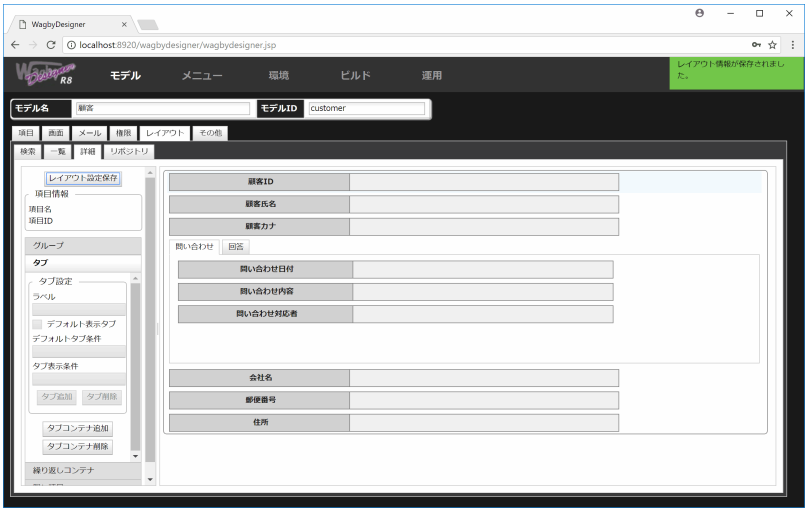


図 タブレイアウトの位置を移動する(2)

タブ表示制御

ラジオボタンによる制御

例

ラジオボタンを使ったタブの表示制御の例を示します。図は「更新記録タブ表示」を選択しています。これに連動して「更新記録」タブが表示されています。

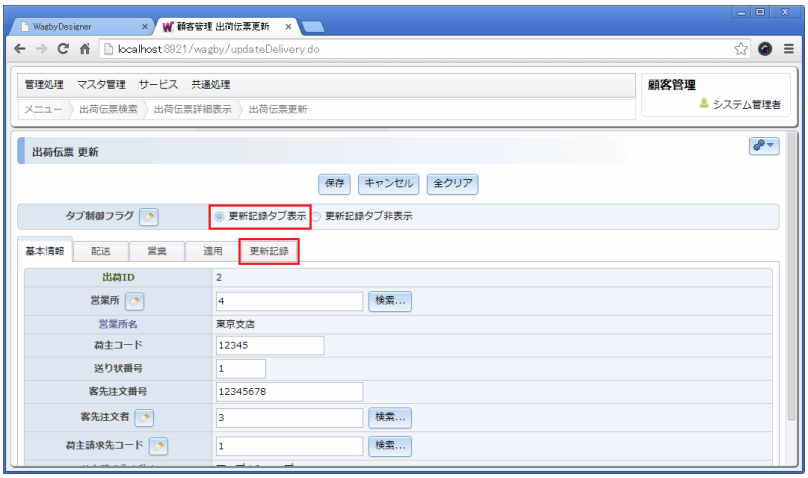


図 更新記録タブが表示されている

図は「更新記録タブ非表示」を選択した例です。このとき「更新記録」タブは表示されません。

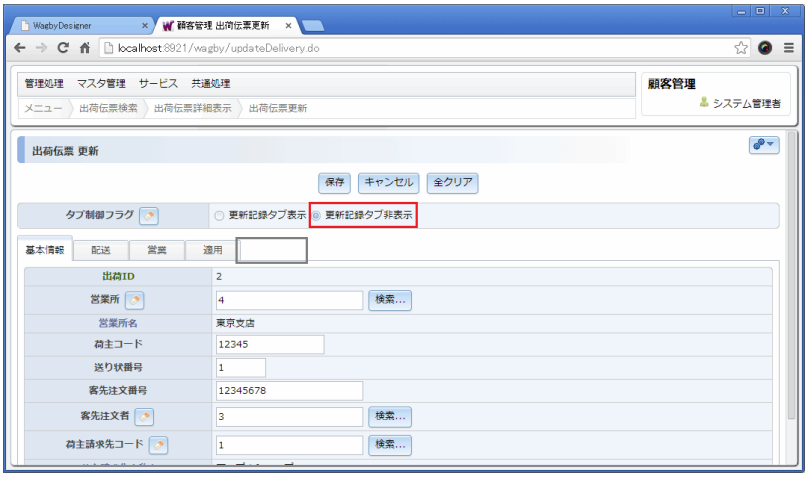


図 更新記録タブが表示されていない

定義方法

新たに「タブ制御フラグ」項目を用意します。

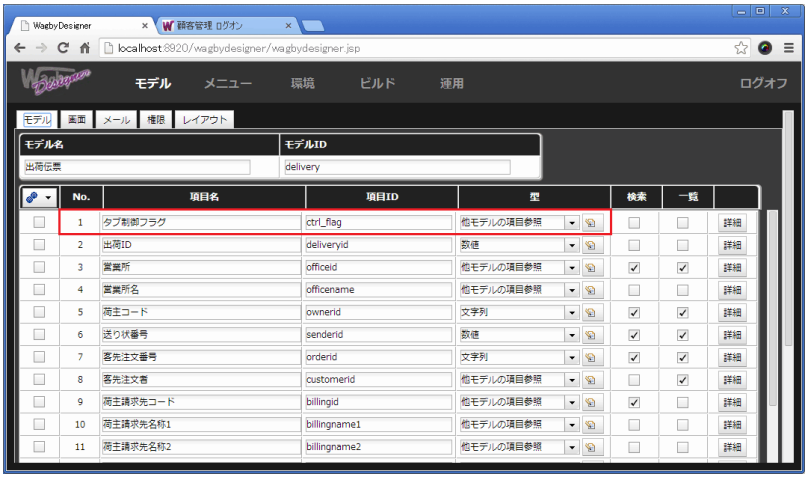


図 タブ制御フラグ項目を定義する

選択肢モデルとして「タブ表示制御フラグ」モデルを用意し、図に示す初期値を設定しておきます。このモデルをラジオボタンで参照します。

図 モデル連携の設定

「更新記録」タブレイアウトの設定で「タブ表示条件」に、条件式「項目ID=値」を記述します。

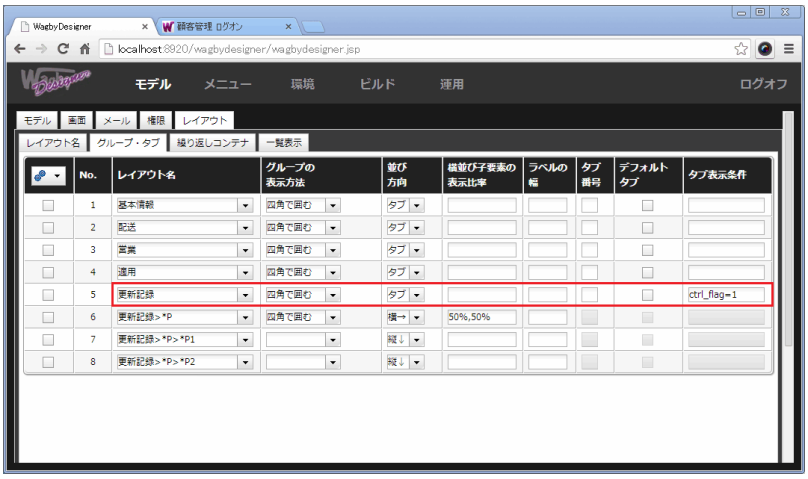


図 タブ表示条件の記述

項目ID=値

図の式は「ctrl_flag=1」となっています。 これは選択肢モデルのID値が "1" の場合（上記の例では「更新記録タブ表示」という選択肢）にタブを表示します。

仕様・制約

- 値と等しい、という記述になります。「ctrl_flag > 1」といった、不等号を用いた式を記述することはできません。
- ウィザードと併用した場合、別画面に配置された項目を指定することはできません。表示条件に用いる項目は、同一画面に存在する必要があります。
- Wagbyが提供する関数は利用できません。

チェックボックスによる制御

例

タブに対応した5つの選択肢を用意しました。最初はいずれもチェックしていないため、タブも表示されません。

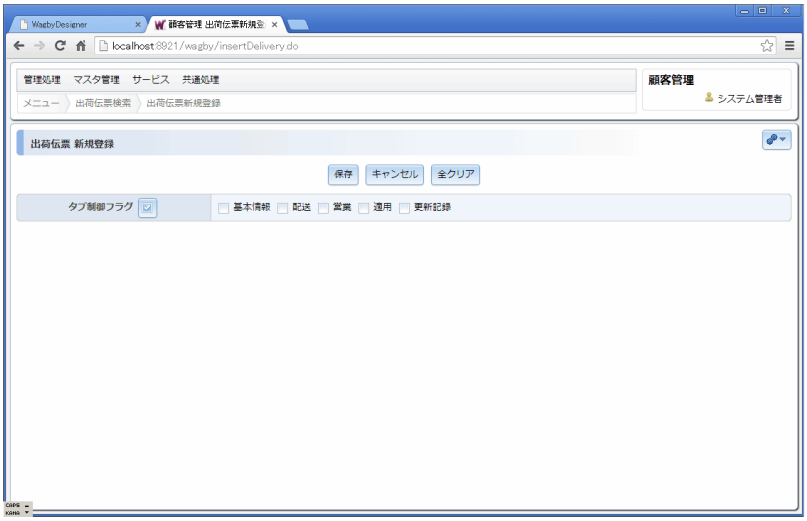


図 すべてのタブが非表示

1つをチェックすると、それに対応したタブが表示されます。

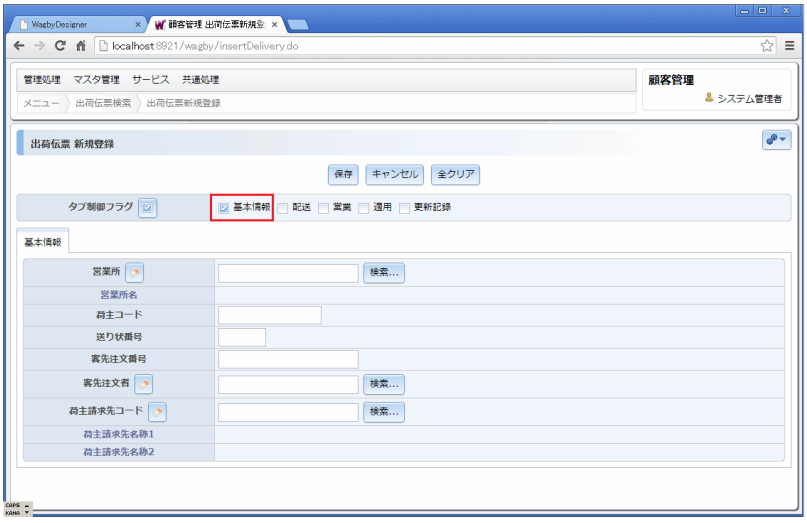


図 基本情報タブが表示された

すべてをチェックすると、すべてのタブが表示されます。

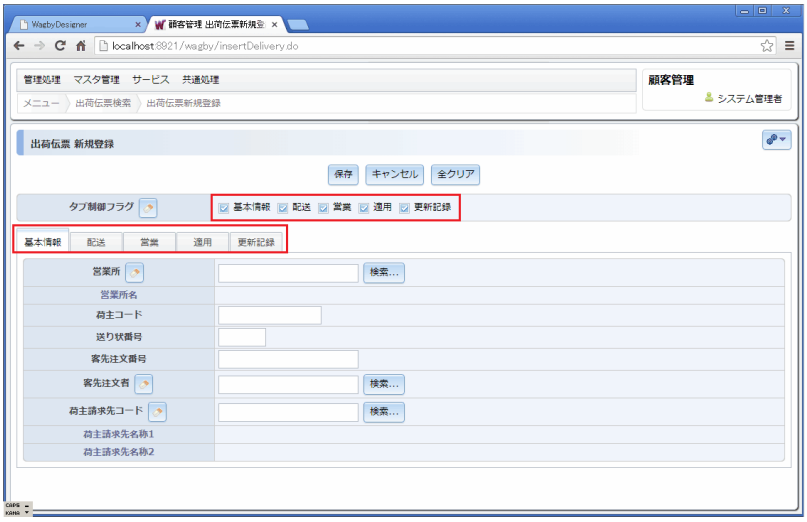


図 すべてのタブが表示された

無効になったタブは、詰めて表示されます。

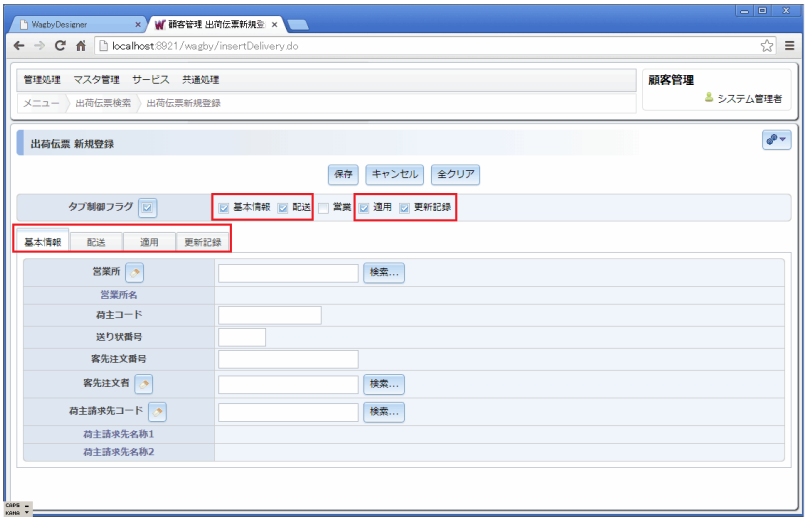


図 途中のタブが無効になった例

定義方法

タブ制御フラグの初期値を5つ用意します。

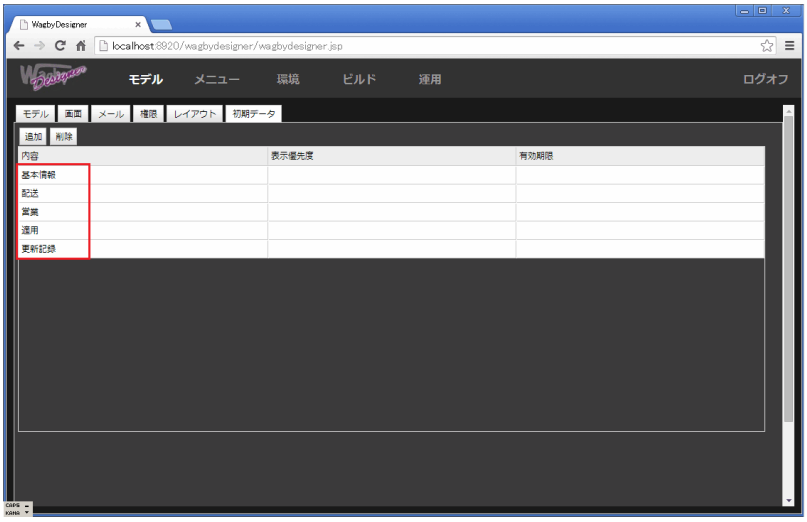


図 タブ制御フラグの初期値

タブ制御フラグをチェックボックスとします。

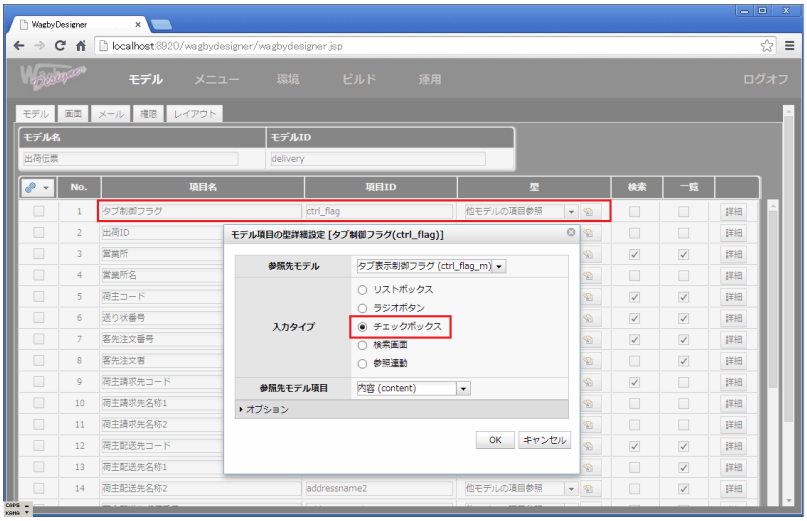


図 タブ制御フラグをチェックボックスに切り替える

各タブに、表示条件を設定します。

```
ctrl_flag=1
ctrl_flag=2
ctrl_flag=3
ctrl_flag=4
ctrl_flag=5
```

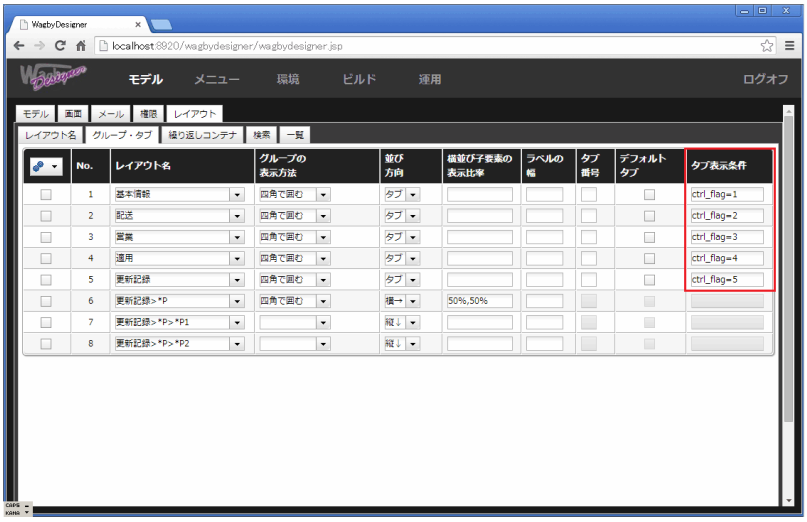


図 タブ表示条件の設定

仕様・制約

″ラジオボタンによる制御″をお読みください。

値による制御

例

登録時は表示されるが、ある条件を満たさないと詳細画面で非表示となるよう、「基本情報」タブを設定します。 ここでは「荷主コード」の値が未入力の場合、詳細画面に表示されません。

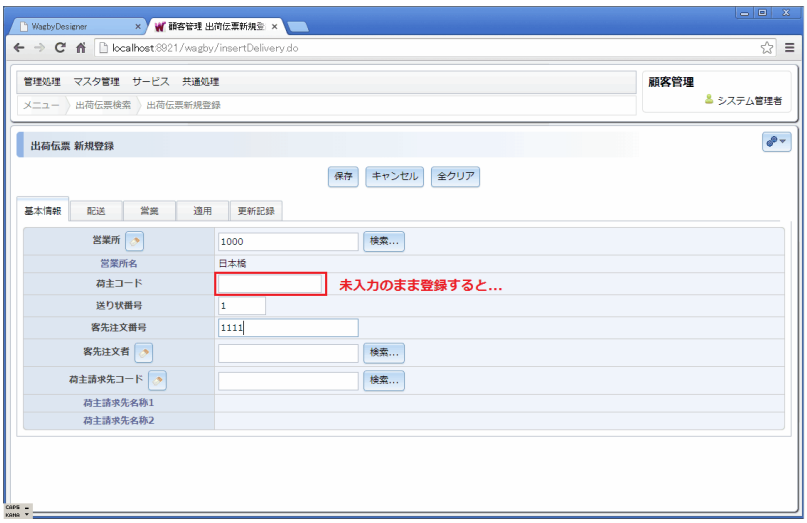


図 登録画面で「荷主コード」を未入力にする

図 詳細画面で基本情報タブは表示されない

定義方法

タブ制御フラグを二つ用意します。それぞれ、コード1が〝表示〟で、コード2が〝非表示〟です。（選択肢モデルでは、コードは1から順に割り当てられます。）

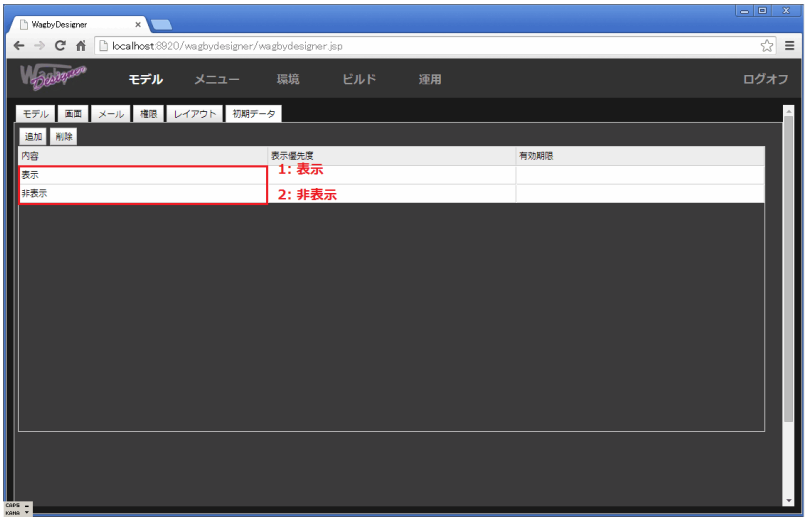


図 タブ制御フラグの設定

出荷伝票モデルで、タブ制御フラグ項目を隠し項目とします。（入力制御と出力制御のそれぞれで隠し設定とします。設定図は省略します。

値を一つしか持たせないため、ここではラジオボタンとします。

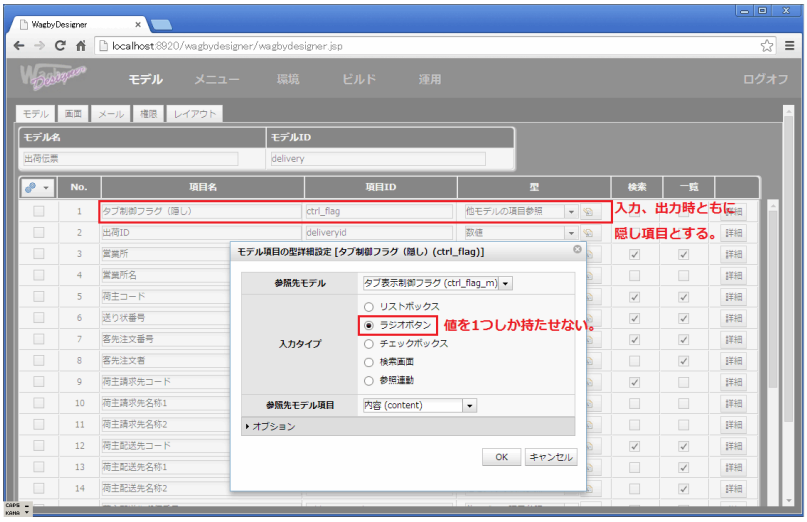


図 タブ制御フラグを隠し項目とする

この制御フラグは入力ではなく式によって値を自動設定します。

```
IF(
  AND(
    EXACT(SCREENTYPE(),"show"),
    ISBLANK(${ownerid})
  ),2,1
)
```

上式は、詳細画面でかつ荷主コード(owerid)項目が空白なら "2"（非表示）を、そうでなければ "1"（表示）をセットします。

図 制御フラグの式

基本情報タブの表示条件を設定します。"1" のときに表示する、としています。

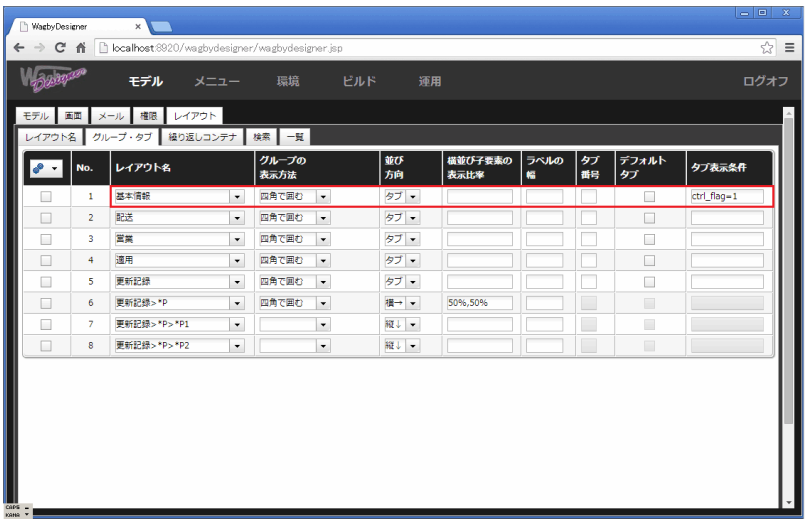


図 タブの表示条件の設定

仕様・制約

- コンテナを含むタブは表示・非表示の切り替えに対応していません。具体的には「複数の繰り返しコンテナを一つのタブにまとめる」設定との併用は行えません。
- 複数のタブの表示・非表示を連鎖させることはできません。ここで"連鎖"とは、タブAの中にある項目Xによってタブの表示制御を行い、さらにタブBの中にある項目Yによってタブの表示制御を...という動きを指します。ただし制御のトリガとなる項目Xや項目Yに対して「項目の値が変更された場合に画面の再描画を行なう」設定を併用することで、この制約を回避することができます。（新しい条件でタブの表示制御が行われた画面が再描画されます。）
- タブ表示条件に複数の異なる制御元を指定することはできません。具体的には、項目Xの値または項目Yの値のいずれも、共通のタブAの表示制御を行う、といったことはできません。項目Xの値で、それぞれ別のタブA,Bを制御することは可能です。（例：タブAはX=1,X=2で表示、およびタブBはX=1で表示）
- 非表示となったタブ内の項目の値はクリアされません。みかけ上、非表示となっていますが、タブ内の項目値は維持されています。タブが再び表示可能となったとき、以前の状態で表示されます。
- 登録・更新画面においてタブレイアウトの表示制御を行なっている項目が読み込み専用(または、更新権限で更新不可)の場合は本機能は動作しません。（ダミーの隠し項目を作成し、読み込み専用項目の値を隠し項目に転記した上で制御を行うようにすることで、この制約を回避できます。）

タイトルペイン

実行例

複数の項目を「タイトルペイン」の内部に格納した例を示します。

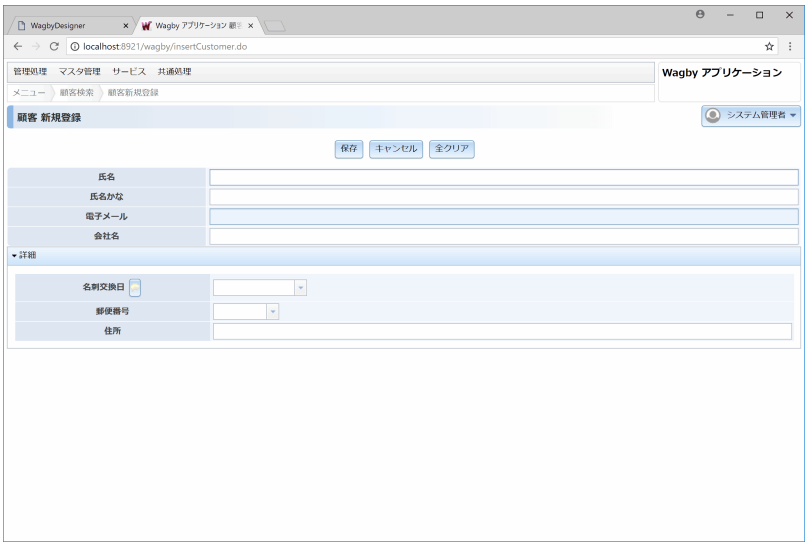


図 タイトルペインレイアウト

タイトル部をクリックすると、閉じられるようになります。再びクリックすると開くことができます。

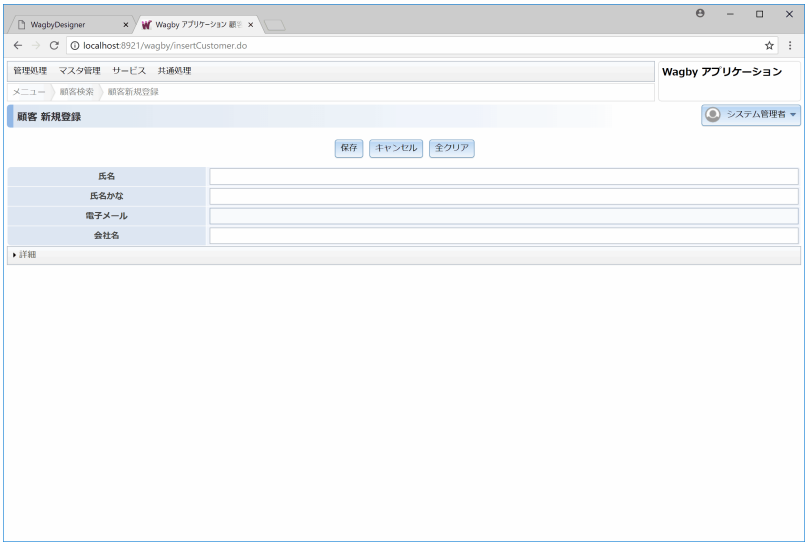


図 タイトルペインを閉じた状態

タイトルペインレイアウトは詳細画面、登録画面、更新画面に反映されます。

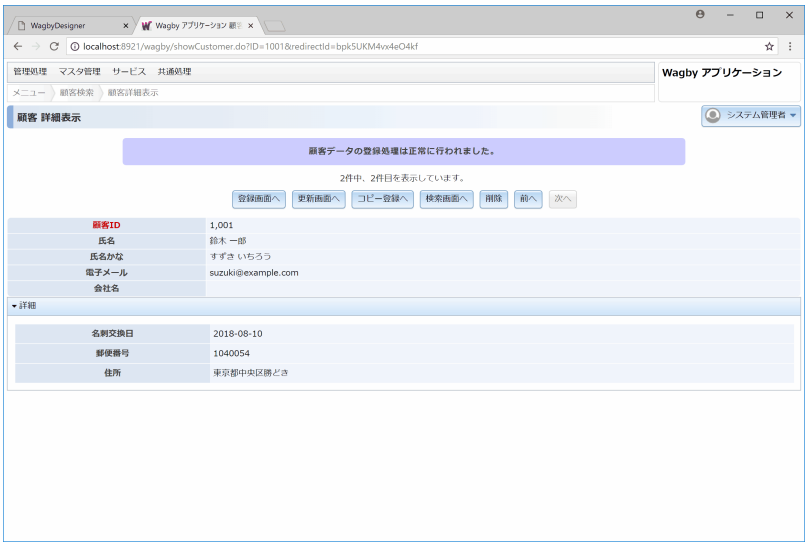


図 詳細画面の例

設定方法

レイアウト設定で「タイトルペイン」を選択します。

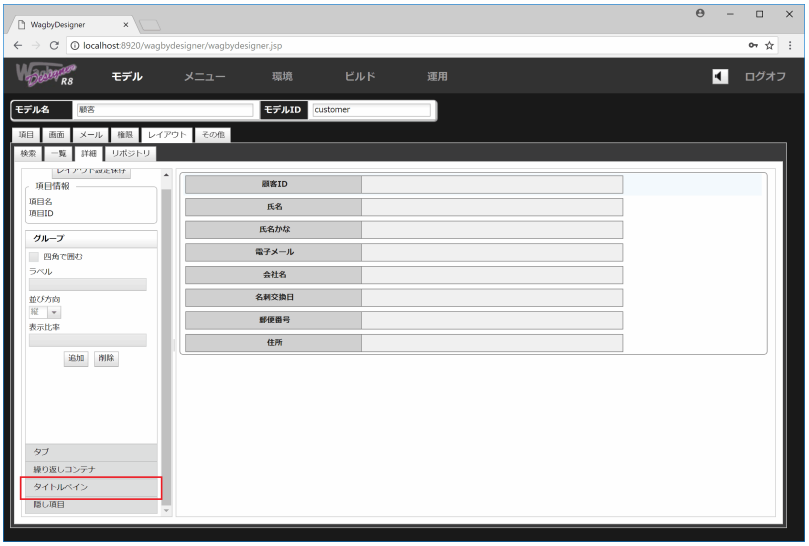


図 レイアウト設定画面

タイトルペインを追加します。

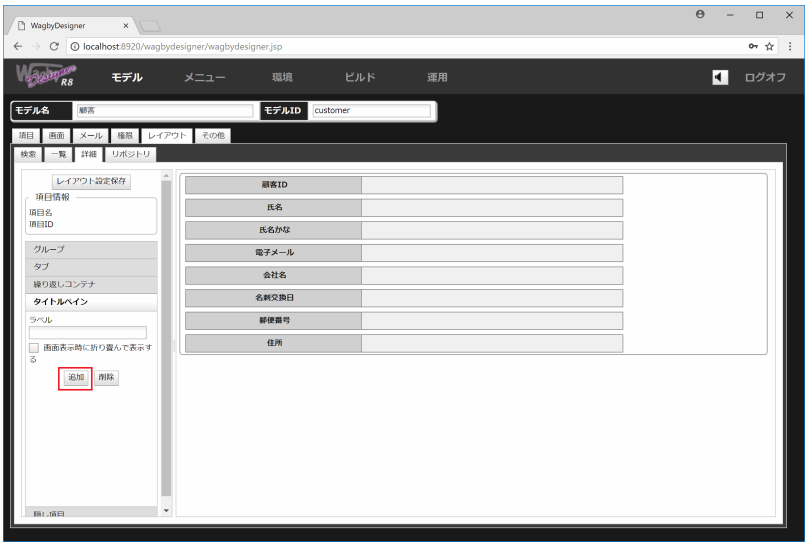


図 タイトルペインを追加する

画面右側の欄にタイトルペインが用意されます。追加されたタイトルペインを選択し、ラベルを変更します。

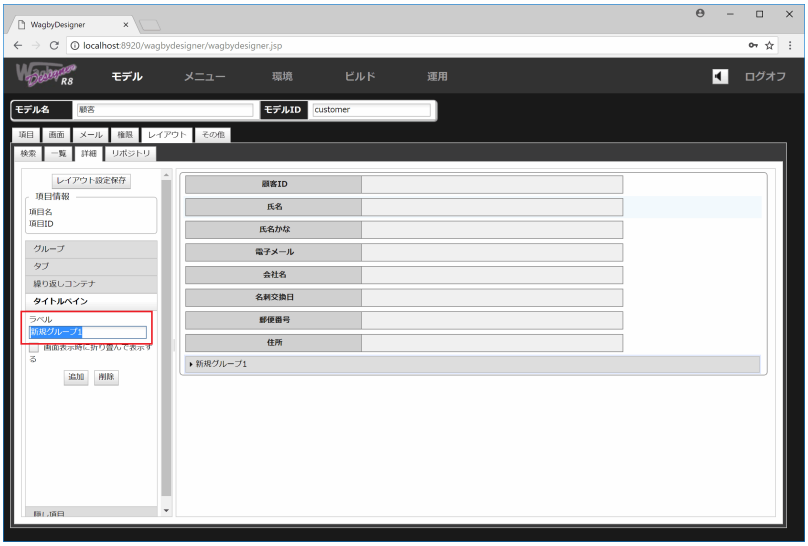


図 ラベルを変更する

ラベルが変更されました。

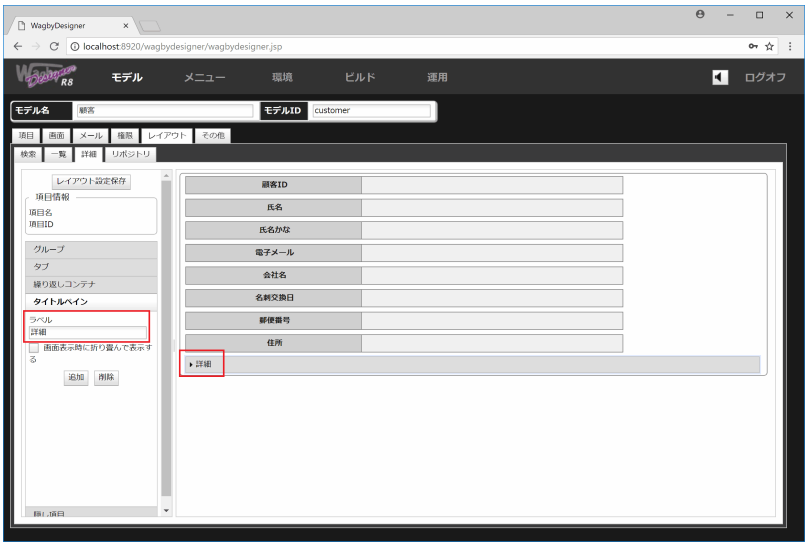


図 ラベルが変更された

項目をドラッグ&ドロップし、タイトルペイン内に移動します。

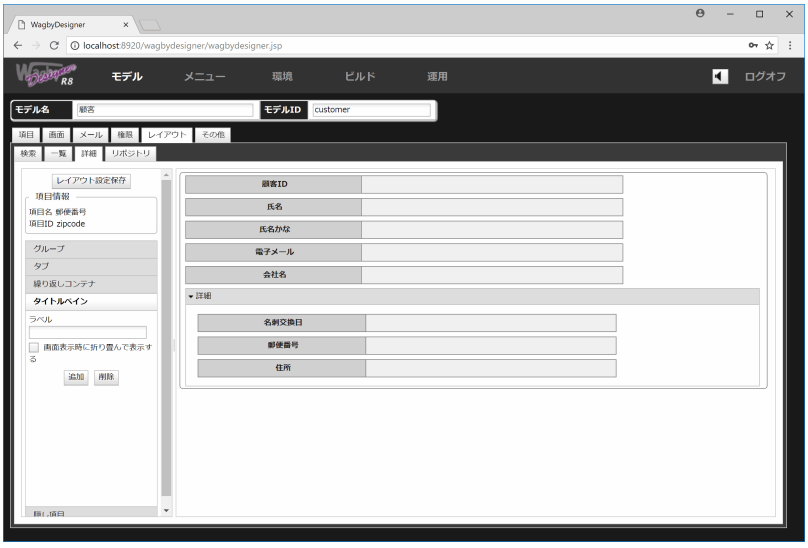


図 タイトルペインに項目を設定する

レイアウト設定を保存します。

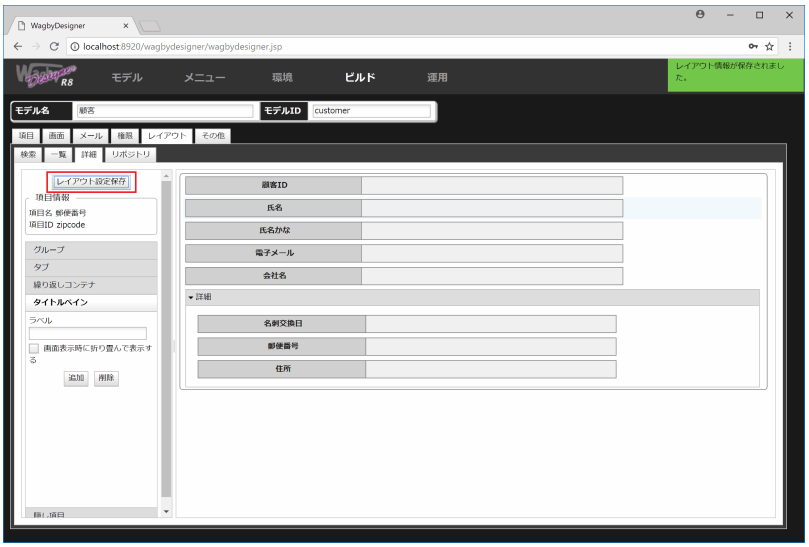


図 レイアウト設定保存

リポジトリを直接、編集する

リポジトリを直接、編集することもできます。タイトルペインに含めたい項目に「レイアウト名」を設定します。

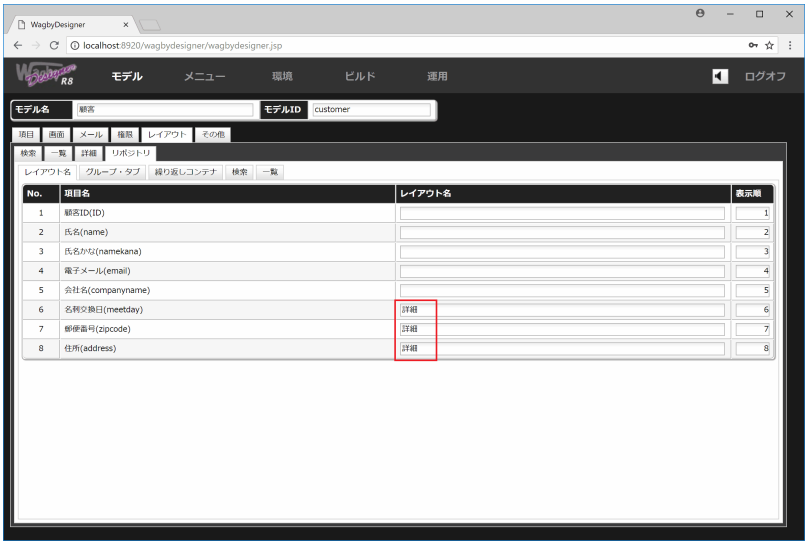


図 リポジトリに直接レイアウト名を指定する

「グループ・タブ」で、グループの表示方法を「タイトルペイン」とします。

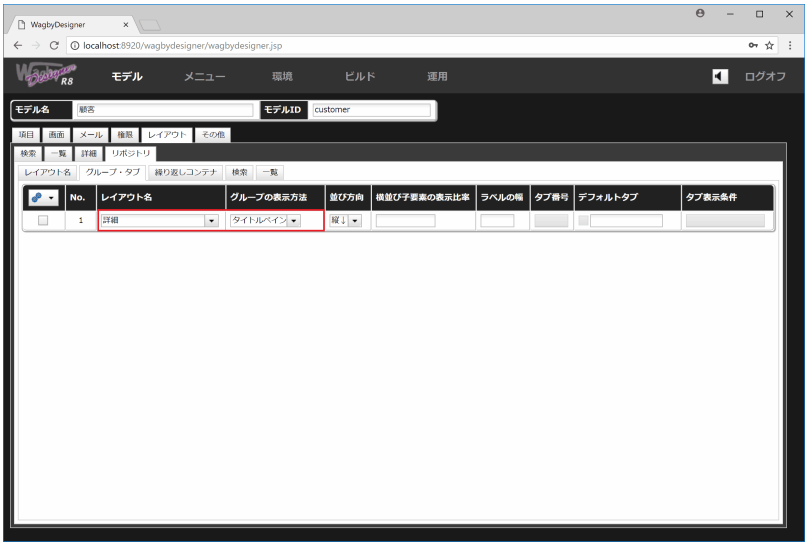


図 グループ表示方法の設定

初期状態の設定

「画面表示時に折り畳んで表示する」を有効にすると、初期表示時にはタイトルペインが閉じた状態となります。

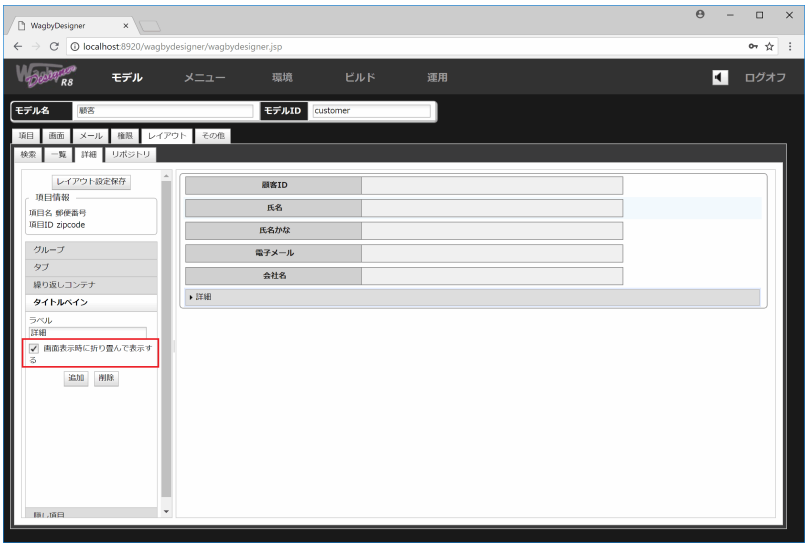


図 画面表示時に折り畳んで表示する

仕様・制約

- タイトルペインは入れ子にできません。
- タイトルペイン内にタブを含めることはできません。
- 開いているか閉じているかの状態は、ブラウザのクッキーに保存されます。
- 「画面表示時に折り畳んで表示する」を有効にしても、その後の操作で開いているか閉じているかという状態変更は（ブラウザのクッキーに）保存されます。そのため画面を再表示した際には直前の表示状態が維持されます。ブラウザ終了後に再ログオンすると初期状態で表示されるようになります。

検索条件の並び順

定義方法

モデル定義上の項目の並びを図に示します。

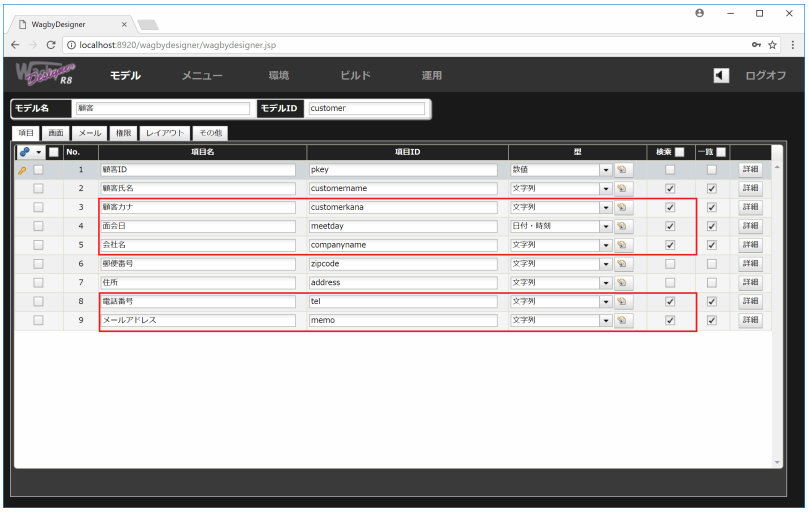


図 モデル定義

「レイアウト」タブ内の「検索」を選択します。

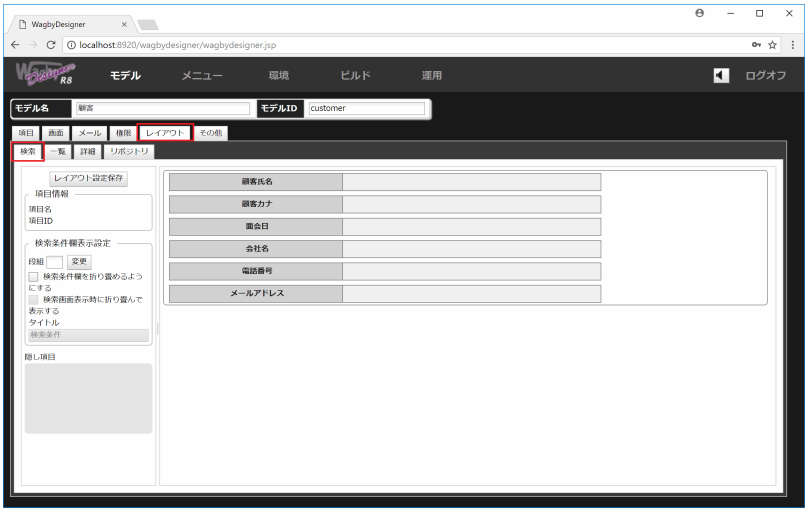


図 表示順の設定(1)

ドラッグ&ドロップで項目の並び順を入れ替えることができます。最後に「レイアウト設定保存」ボタンを押してください。

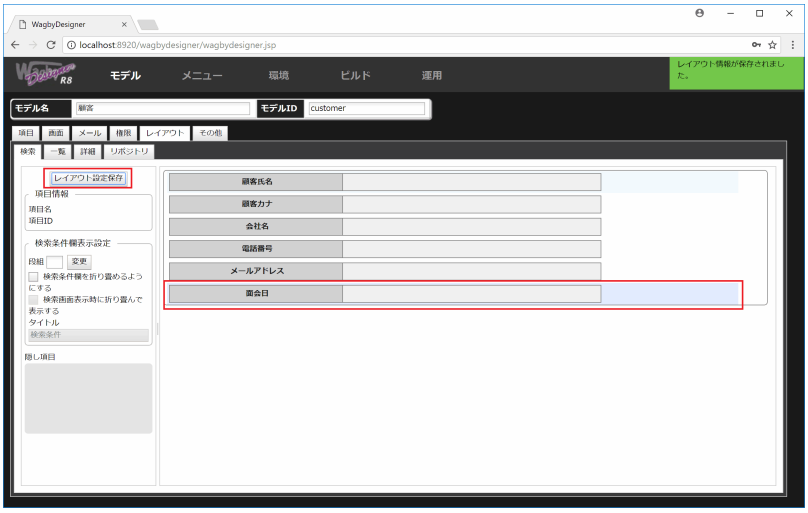


図 表示順の設定(2)

ご注意ください。「レイアウト設定保存」ボタンを押下せずに他の画面に切り替えた場合、これまでの編集内容は反映されません。

リポジトリの確認

ここまでの設定が、どのようにリポジトリに反映されたかを確認します。「リポジトリ」タブの中にある「検索」タブを開きます。「表示順」は、レイアウト画面での項目の並び順序が反映されています。

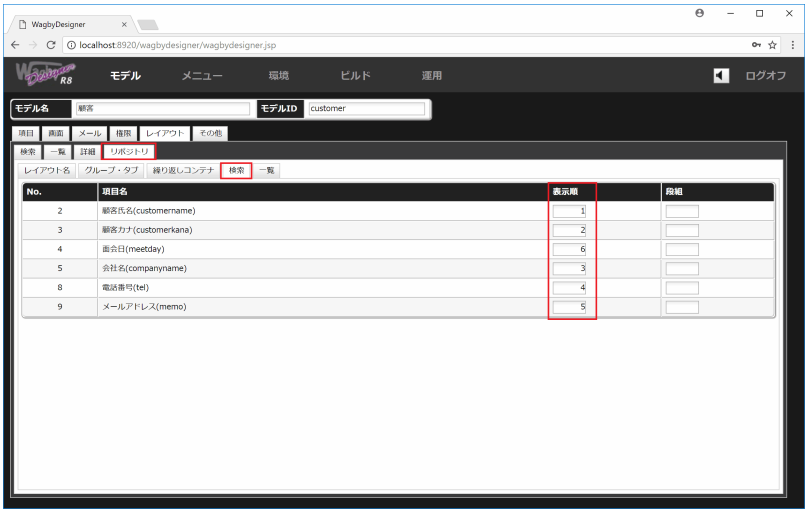


図 リポジトリの内容を確認する

例

実行例を示します。図の表示順に従った検索画面となっていることがわかります。

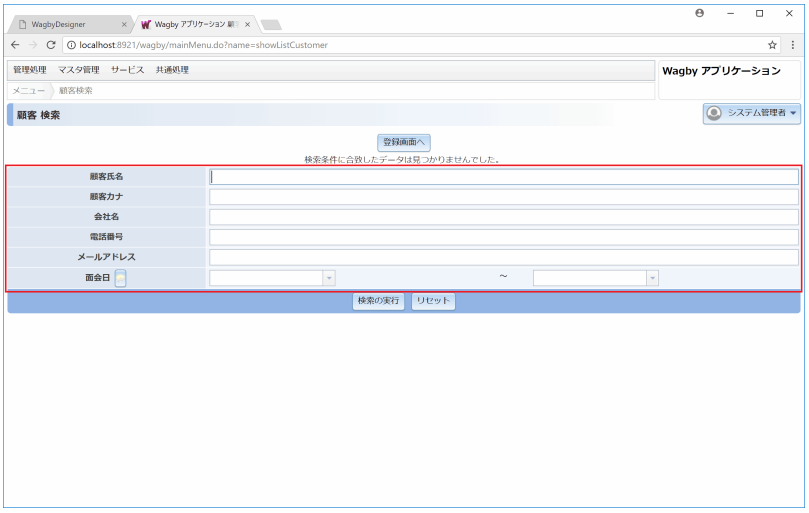


図 表示順を変更した検索画面

表示順の設定に関する詳細

- 指定した数字の昇順に画面表示されます。
- 未指定が混在した場合、数字が優先されます。そのあとに未指定が表示されます。未指定の順序は定義の出現順となります。
- 数字は飛び番号も指定できます。ゼロやマイナスも指定できます。
- 数字の範囲は整数型に準拠します。小数点は指定できません。
- 同じ数字を指定することはできません。（この場合は、同じ数字の最後の検索項目のみが表示されます。）

範囲検索条件の入力欄のレイアウト

定義方法

数値型または日付・時刻型項目を範囲検索としたとき、表示レイアウトを指定することができます。

モデル項目詳細設定の「検索制御 ＞ スタイル ＞ 範囲検索条件入力欄のレイアウト」を指定します。標準は″均等割り付け″です。

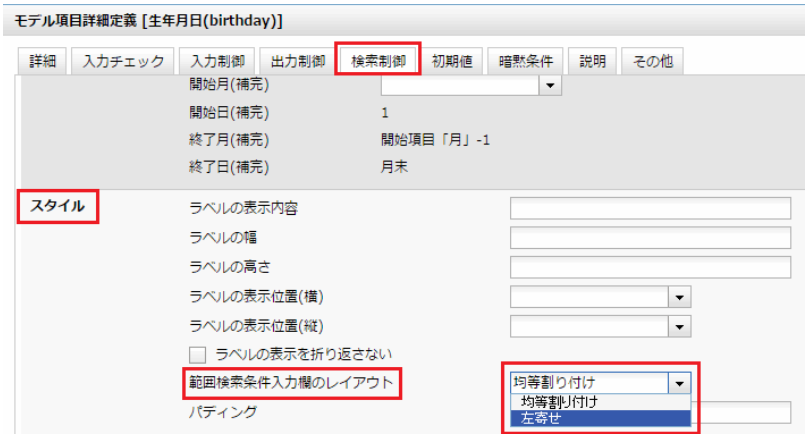


図 範囲検索条件入力欄のレイアウトの設定

例

均等割り付け

均等割り付けでは、入力欄と ″～″ 文字の表示比率が 45%, 10%, 45% で調整されています。

図 均等割り付けの例

左寄せ

左寄せの例を図に示します。

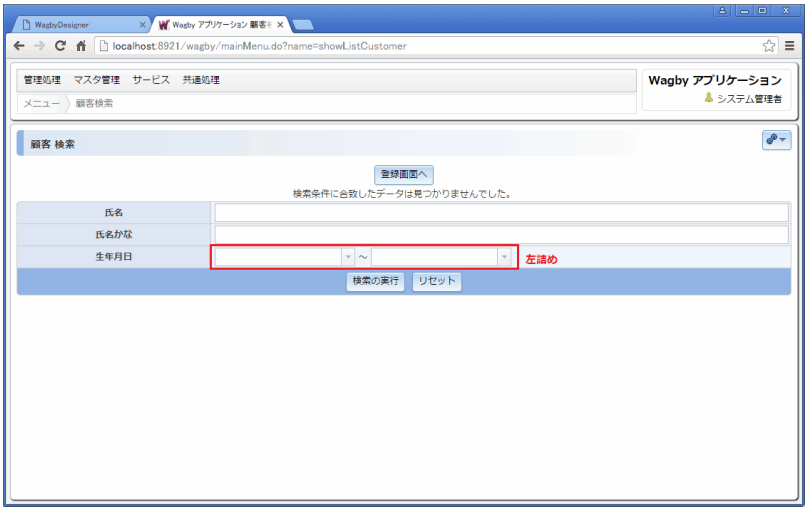


図 左寄せの例

検索条件部の段組レイアウト

例

図は、検索条件の項目を二段で表示した例です。

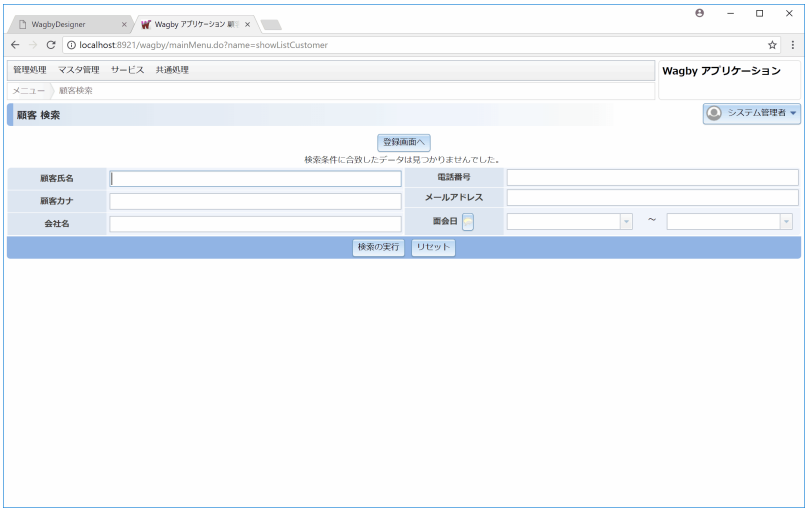


図 多段レイアウトを用いた例

定義方法

レイアウトタブ内の検索タブを開きます。「検索条件欄表示設定」パネル内にある「段組」に 2 以上の数字を指定します。

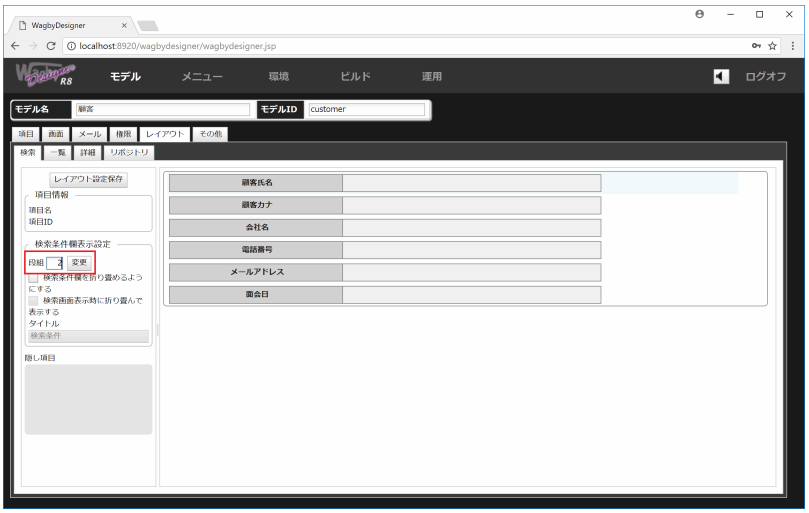


図 段組の設定

段組が用意されます。透明となっていますが、項目を配置することができます。

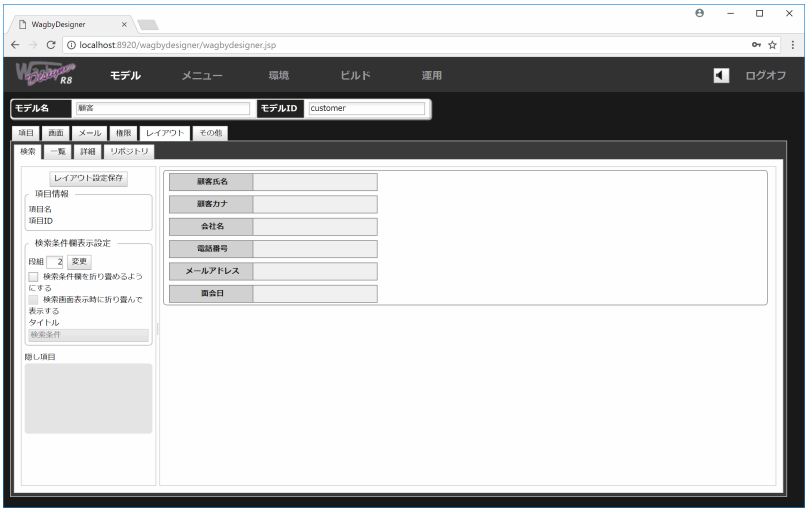


図 検索項目を配置する(1)

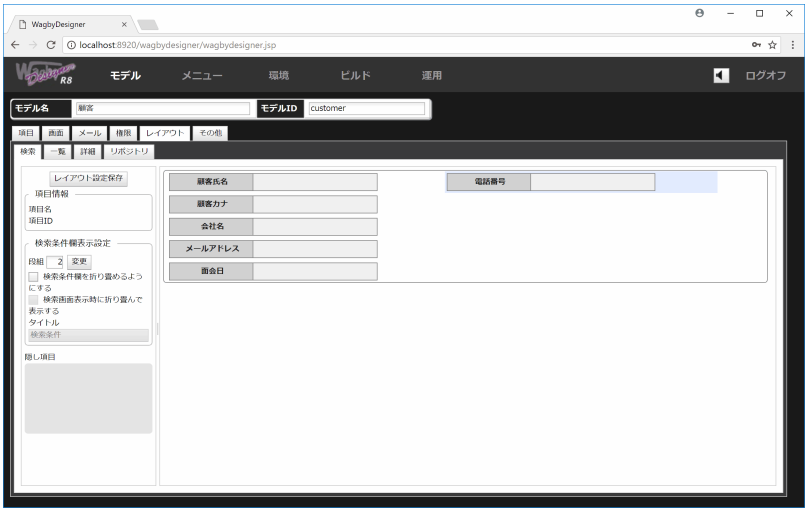


図 検索項目を配置する(2)

スクリーンエディタの編集を保存する

配置が終わったら、左側の設定欄の上部にある「レイアウト設定保存」ボタンを押下して編集内容を保存します。この操作により、リポジトリに反映されます。

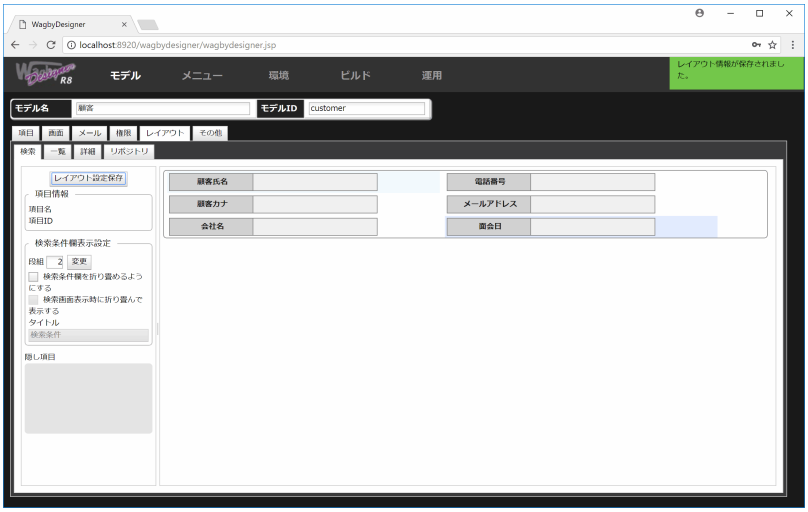


図 レイアウト設定を保存する

ご注意ください。「レイアウト設定保存」ボタンを押下せずに他の画面に切り替えた場合、これまでの編集内容は反映されません。

リポジトリの確認

リポジトリタブ内の検索タブで、表示順と段組を確認できます。段組欄の数値は 1 から開始されます。左から順に1段目,2段目に配置されるようになります。

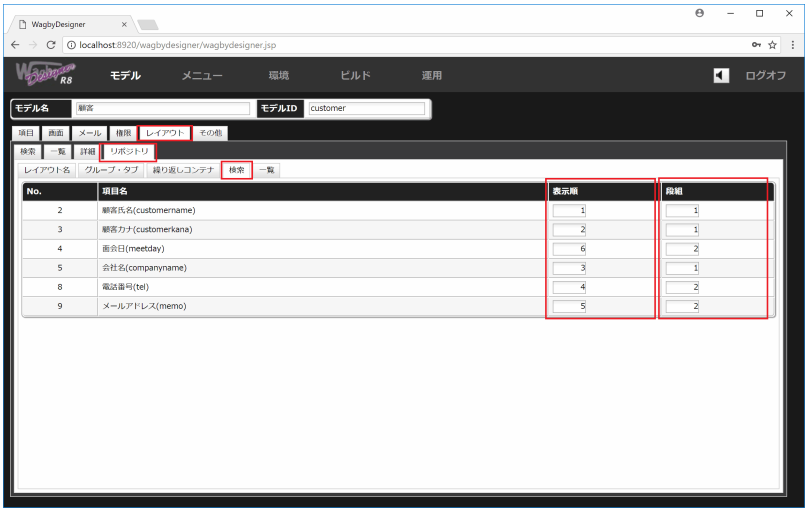


図 リポジトリの内容

仕様・制約

- 項目a,b,c,d,e がそれぞれ1,1,1,2,2となった場合、1段目にa,b,cが、2段目にd,eが配置されます。
- 表示順の設定は、全体を通して重複しないように設定してください。項目 a,b,c,d,e が二段 (a,b,c と d,e) のとき、表示順が 1,2,3,4,5 であれば 1段目が a,b,c、2段目が d,e と解釈されます。
- 表示順に同じ数字を指定することはできません。（この場合は、同じ数字の最後の検索項目のみが表示されます。）
- 数値が入っている場合と入っていない場合が混在した場合、未設定部分はすべて 1 とみなされます。
- 飛び数字はすべて詰められます。1,2,4 と設定された場合、1,2,3 と解釈され、3段組となります。
- 設定できる段数に制限はありません。

検索条件欄を折り畳む

例

顧客モデルに適用した例を示します。検索条件欄がパネルにおさまっています。

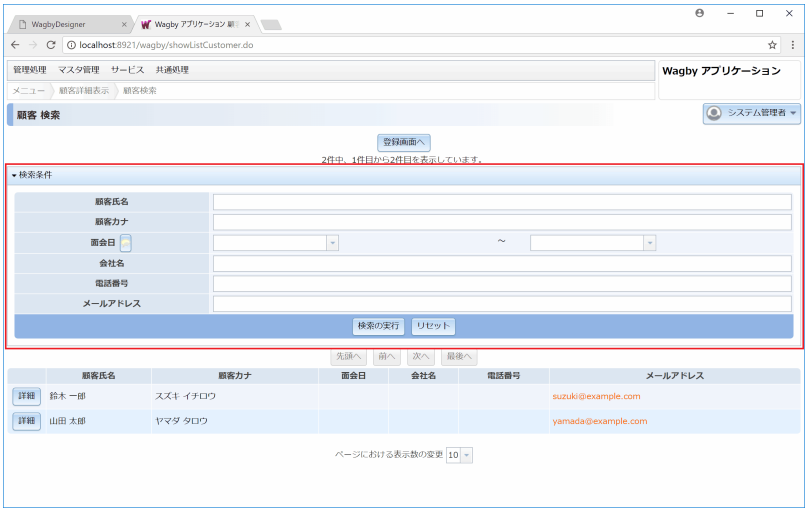


図 検索条件欄がパネルレイアウトにおさまっている

パネルのタイトル部に用意されたトグルスイッチを押下することで、検索条件欄を折り畳むことができます。

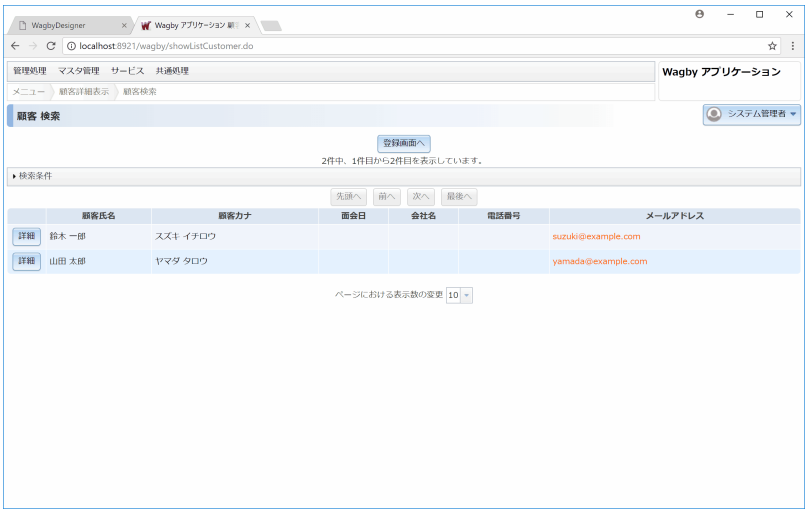


図 折り畳まれた状態

仕様

- 折り畳まれた状態は、ログオンしている間、記憶されます。ログアウトすると初期設定に戻ります。
- 例では検索画面と一覧表示画面を同時に表示していましたが、これを別々とした場合でも、折り畳み設定を行うことができます。

定義方法

「レイアウト」タブ内の「検索」を選択します。「検索条件欄表示設定」で、「検索条件欄を折り畳めるようにする」を有効にします。

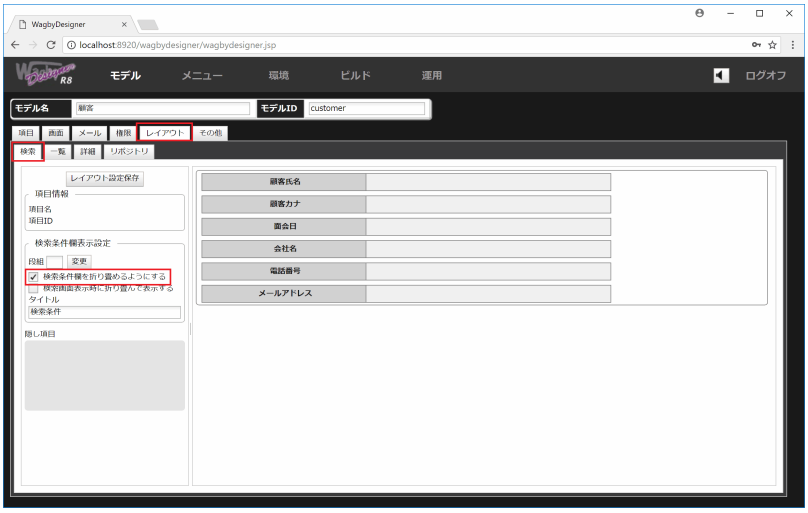


図 レイアウト定義

表示設定の詳細

検索画面表示時に折り畳んで表示する

検索画面表示時、最初から折り畳んで表示させることができます。 「検索条件欄を折り畳めるようにする」が有効な場合のみ、設定できます。

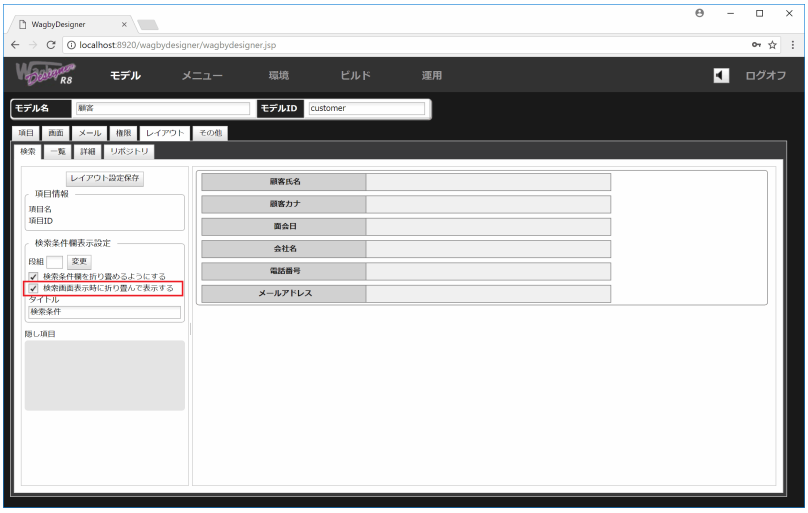


図 検索画面表示時に折り畳んで表示する設定

タイトル

タイトルを変更することができます。

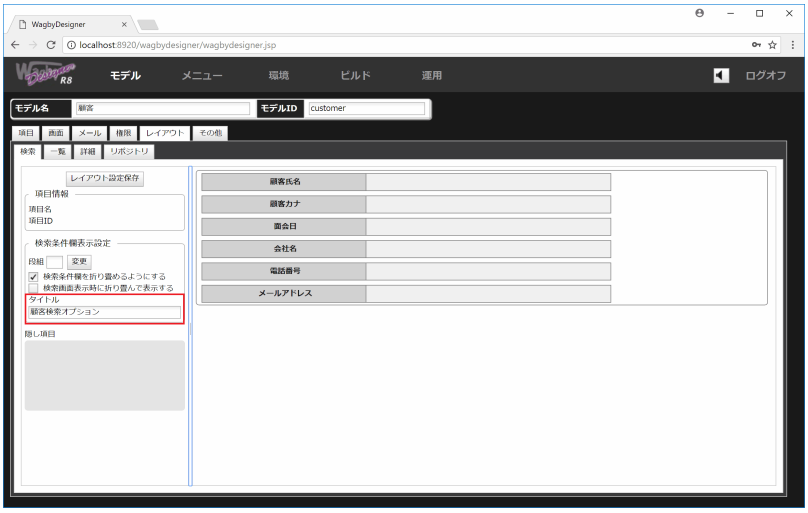


図 タイトルを変更する

タイトルを変更した例を示します。

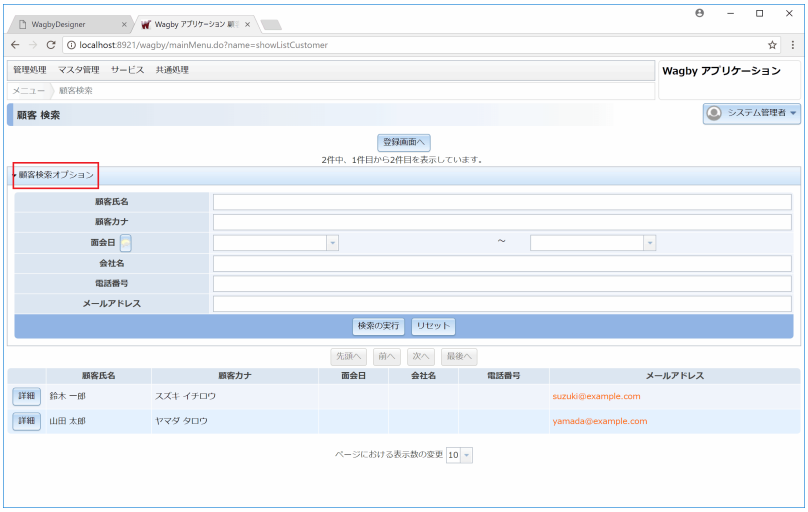


図 タイトルを変更した画面

一覧表示の並び順

定義方法

モデル定義上の項目の並びを図に示します。

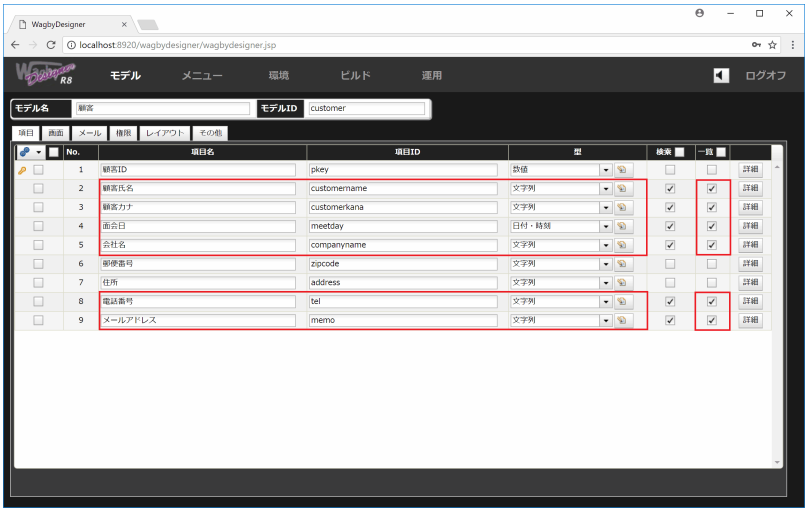


図 モデル定義

「レイアウト」タブ内の「一覧」を選択します。一覧表示の項目をドラッグ&ドロップ操作で並べ換えることができます。

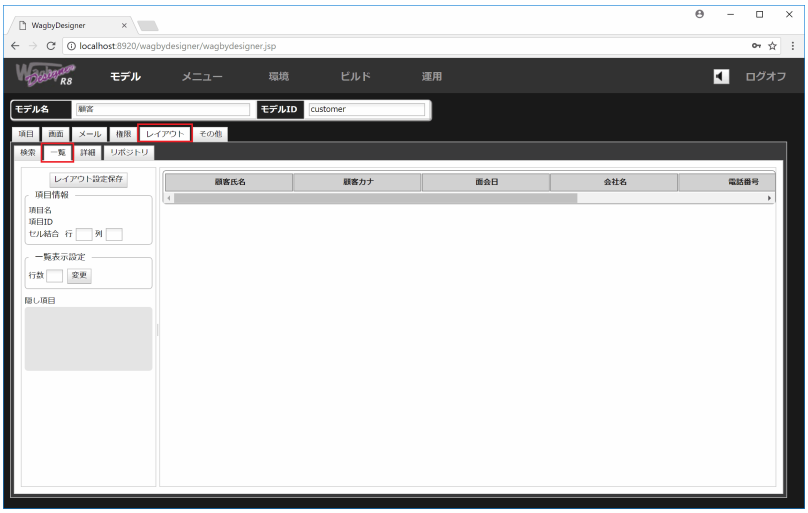


図 項目の並び替え(1)

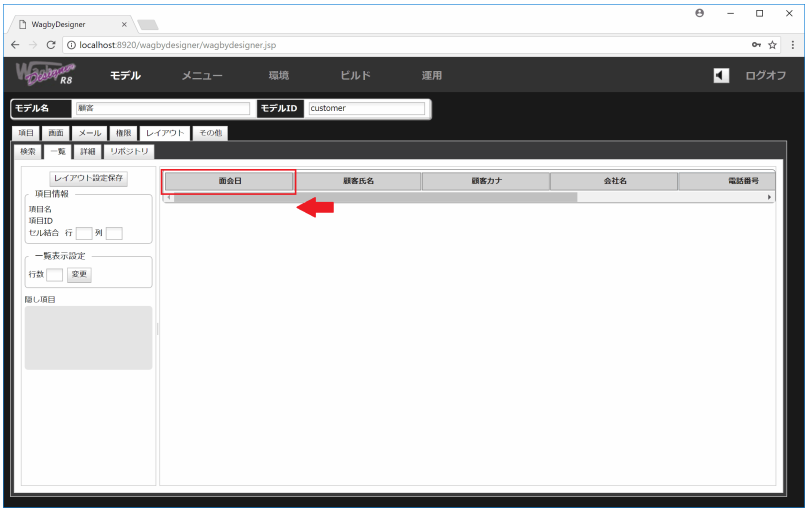


図 項目の並び替え(2)

スクリーンエディタの編集を保存する

左側の設定欄の上部にある「レイアウト設定保存」ボタンを押下し、編集内容を保存します。この操作により、リポジトリに反映されます。

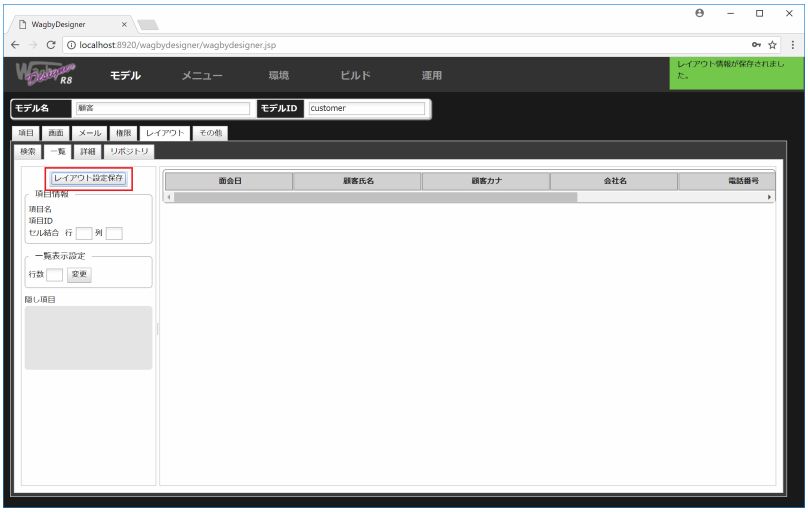


図 レイアウト設定を保存する

ご注意ください。「レイアウト設定保存」ボタンを押下せずに他の画面に切り替えた場合、これまでの編集内容は反映されません。

リポジトリの確認

ここまでの設定が、どのようにリポジトリに反映されたかを確認します。「リポジトリ」タブを開きます。「表示順」は、レイアウト画面での項目の並び順序が反映されています。

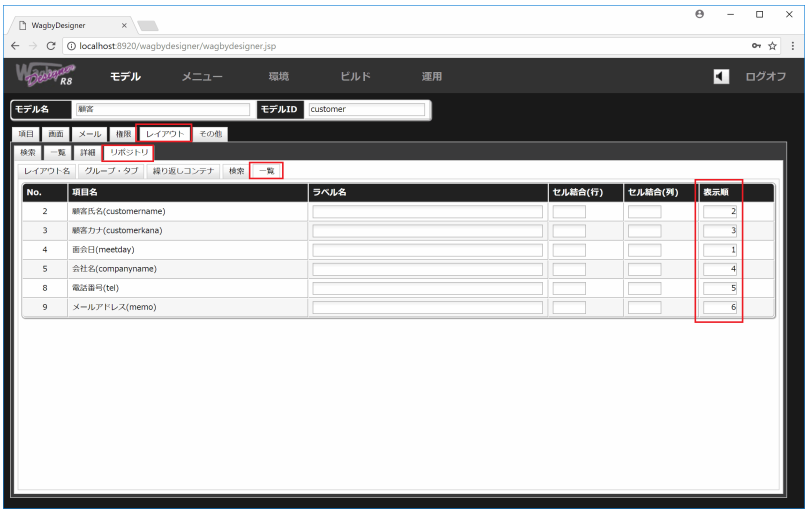


図 リポジトリの確認

図の画面で、表示順を直接、編集することもできます。

実行例

例を図に示します。図の表示順に従った一覧表示画面となっていることがわかります。

図 表示順を変更した一覧表示画面

表示順の設定に関する詳細

- 指定した数字の昇順に画面表示されます。
- 未指定が混在した場合、数字が優先されます。そのあとに未指定が表示されます。未指定の順序は定義の出現順となります。
- 数字は飛び番号も指定できます。ゼロやマイナスも指定できます。
- 数字の範囲は整数型に準拠します。小数点は指定できません。

一覧表示 － 複数段一行明細(多段)レイアウト

基本的な設定

図は、一覧表示項目を二段で表示した例です。

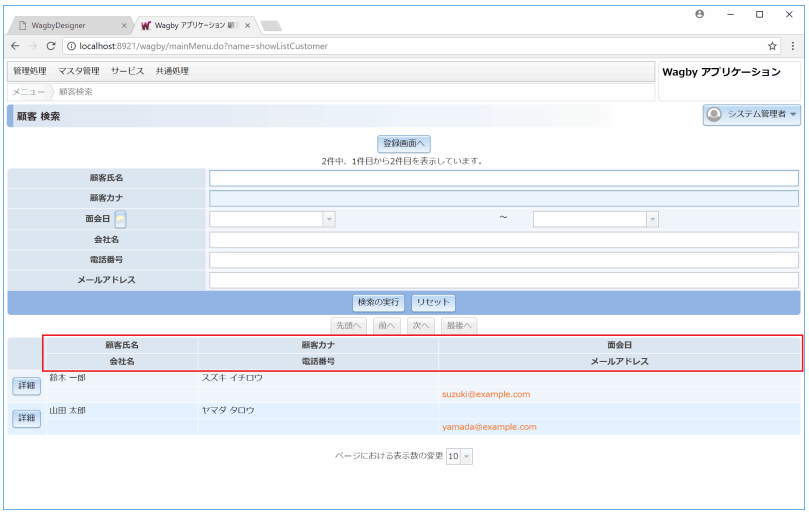


図 複数段一行明細を用いた例

レイアウトの定義

「レイアウト>一覧」を開きます。左側の「一覧表示設定」で「行数」を指定します。ここでは"2"とします。

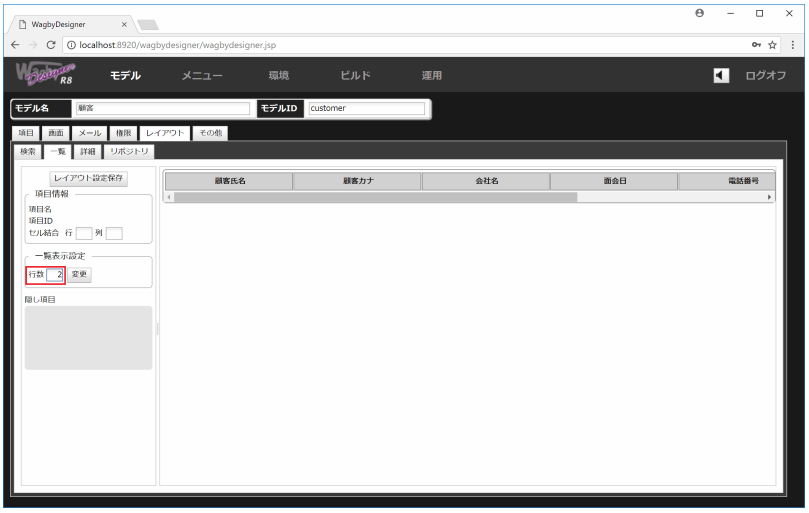


図 行数の指定(1)

「変更」ボタンを押下すると、段組が用意されます。

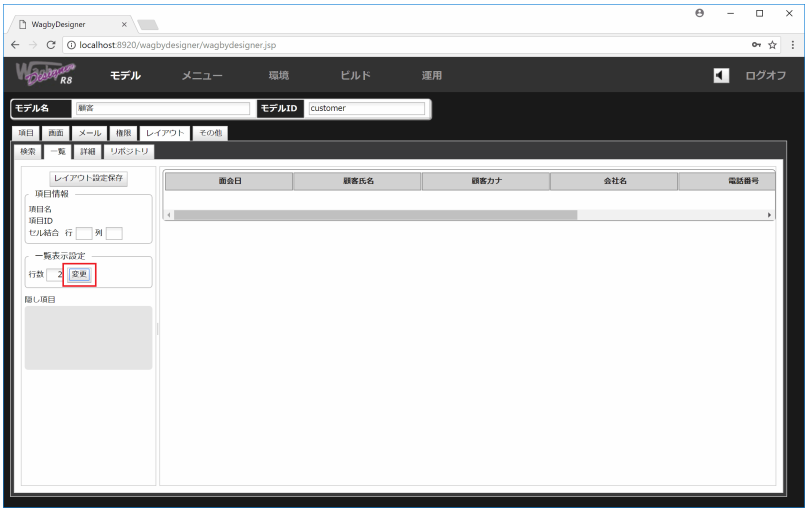


図 行数の指定(2)

ドラッグ&ドロップ操作で、項目を段組の内部へ配置することができます。

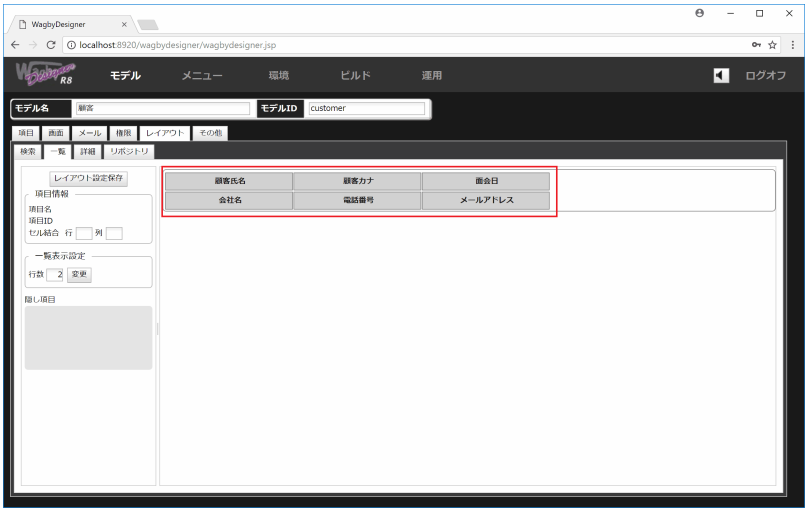


図 段組レイアウトの設定

(179)

スクリーンエディタの編集を保存する

左側の設定欄の上部にある「レイアウト設定保存」ボタンを押下し、編集内容を保存します。この操作により、リポジトリに反映されます。

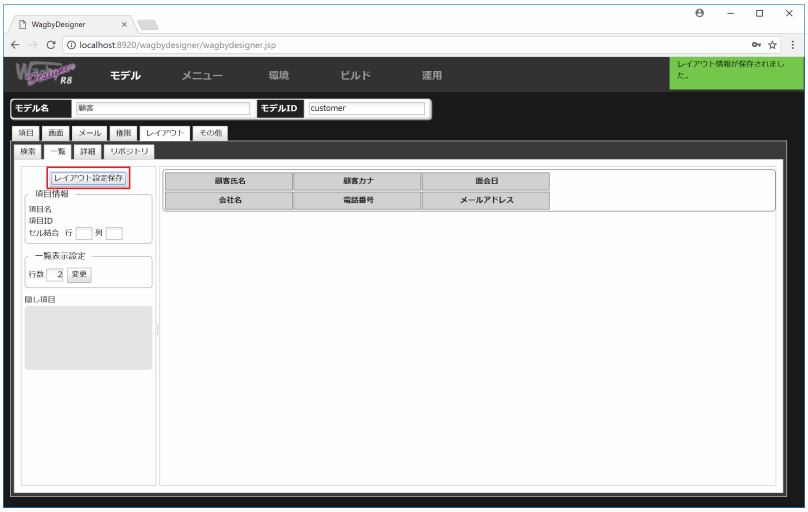


図 設定情報を保存する

リポジトリの確認

ここまでの設定が、どのようにリポジトリに反映されたかを確認します。「リポジトリ」タブを開きます。「ラベル名」に ″L1″ と ″L2″ が指定されています。これがそれぞれ1段目、2段目を意味します。

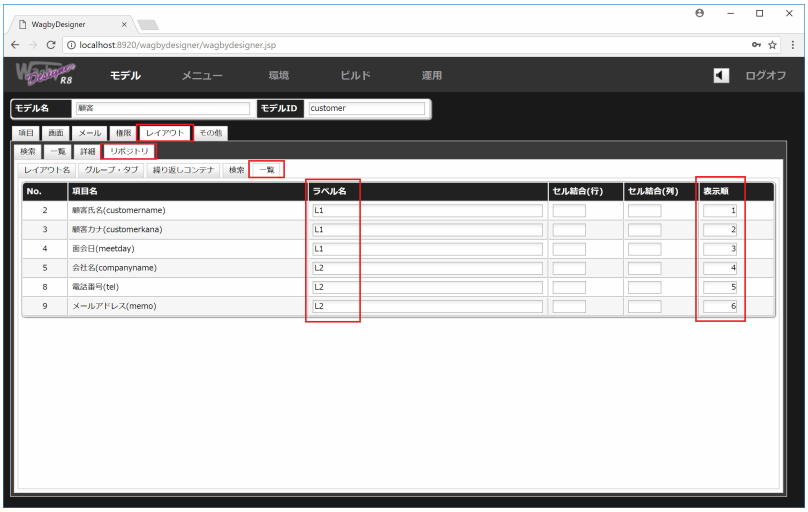


図 リポジトリの確認

ラベル ″L1″ と ″L2″ は図のように解釈してください。

(179) ご注意ください：図の状態で、行数を″1″または空白とすると、2行目が不可視となり、2行目のレイアウト変更が行えなくなります。行数を減らす場合は、いったん項目を減らした状態にしたあと、行数を変更するようにしてください。

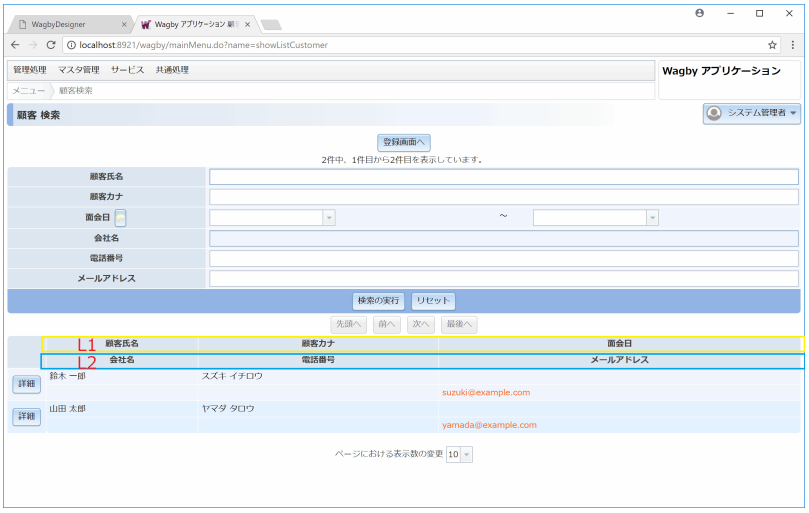


図 段組ラベルの解釈方法

セル結合（列）

三段目を用意した例を図に示します。ここで追加した「備考」欄は3つの列をまとめています。

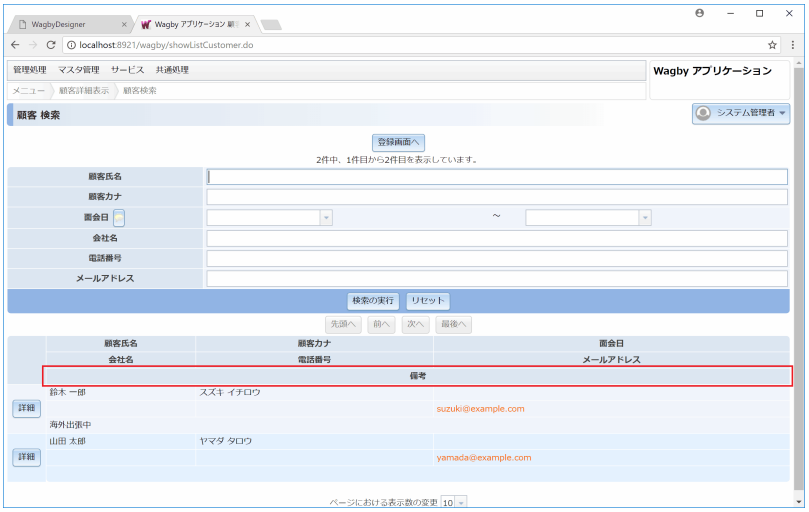


図 三段目を用意した例

レイアウトの定義

「レイアウト>一覧」を開きます。左側の「一覧表示設定」で「行数」を ″3″ に変更します。「備考」を3段目に配置します。

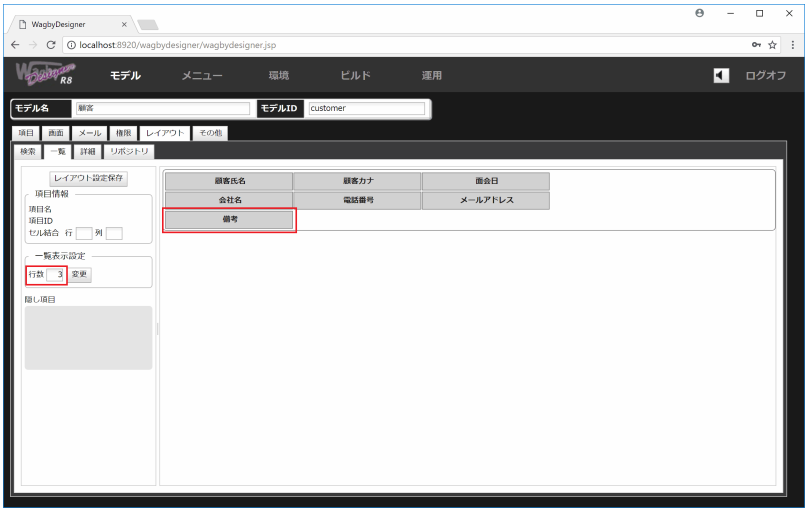


図 行数を3とし、備考項目を配置する

「備考」をマウスで選択し、列を〝3〟とします。

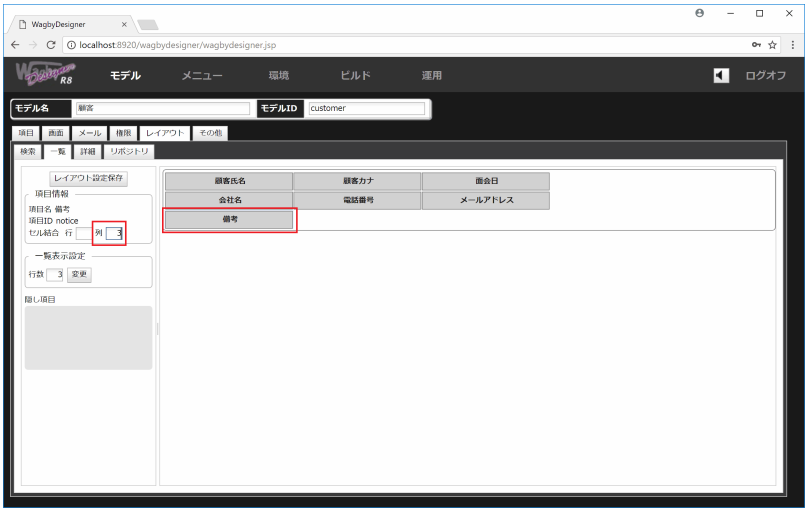


図 セル列を設定する(1)

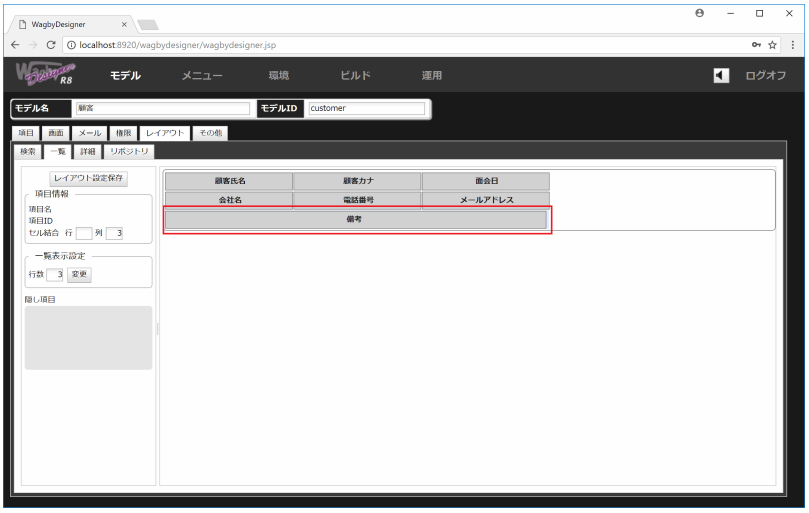


図 セル列を設定する(2)

スクリーンエディタの編集を保存する

左側の設定欄の上部にある「レイアウト設定保存」ボタンを押下し、編集内容を保存します。この操作により、リポジトリに反映されます。

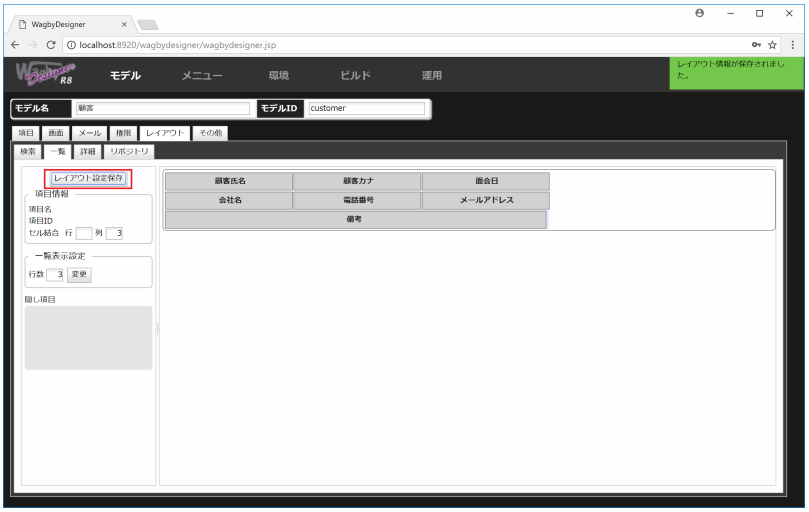


図 レイアウト設定を保存する

リポジトリの確認

ここまでの設定が、どのようにリポジトリに反映されたかを確認します。「セル結合（列）」に ″3″ が設定されています。

「セル結合（列）」は、結合を適用する項目に数値を指定します。図の設定では、自分自身　(L3に配置) の 1つ上の段である L2 の、3つのセル（列）をまとめた位置に配置される、という意味になります。

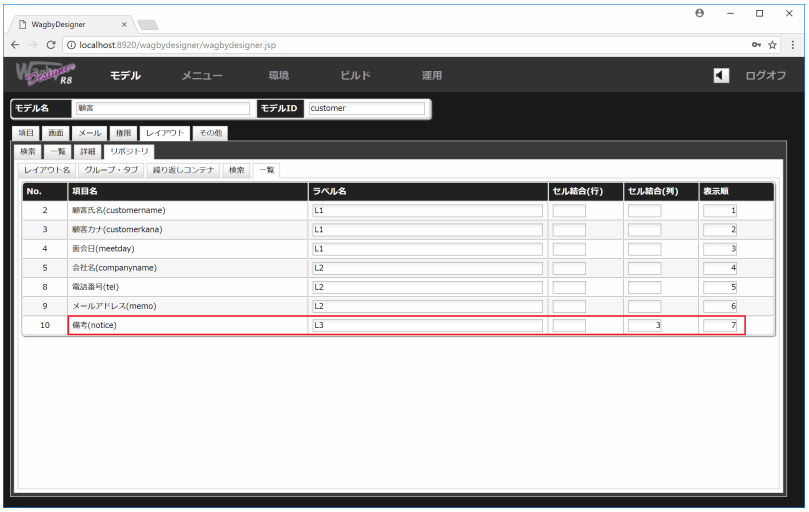


図 リポジトリの確認

セル結合（行）

セル結合（行）は、自分自身を次の行と連結させるための指定です。

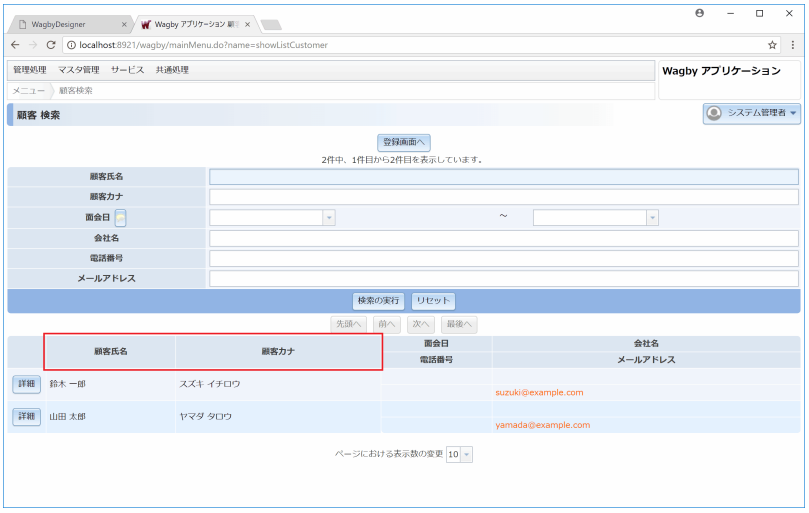


図 セル結合（行）を指定した例

レイアウト設計

行数を2とし、二行目に「電話番号」「メールアドレス」項目を配置した例を図に示します。

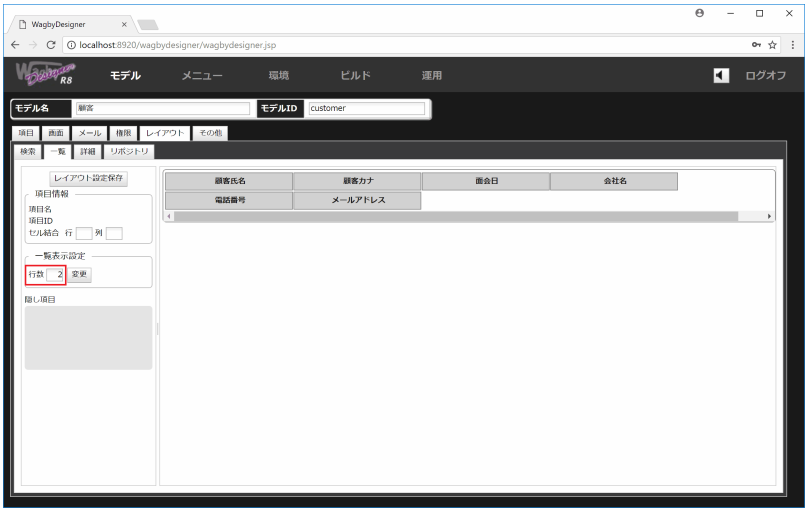


図 二行で表示する例

「顧客氏名」と「顧客カナ」項目について、セル結合（行）を ″2″ と設定します。

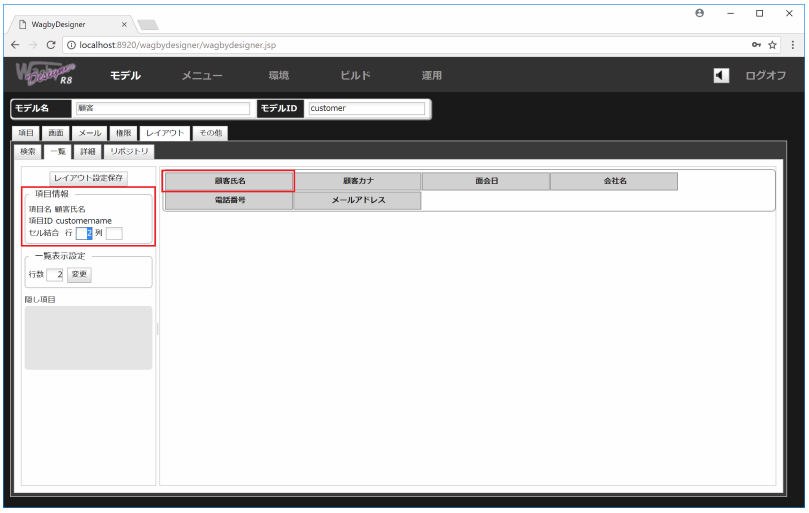


図 顧客氏名項目へのセル結合（行）の設定

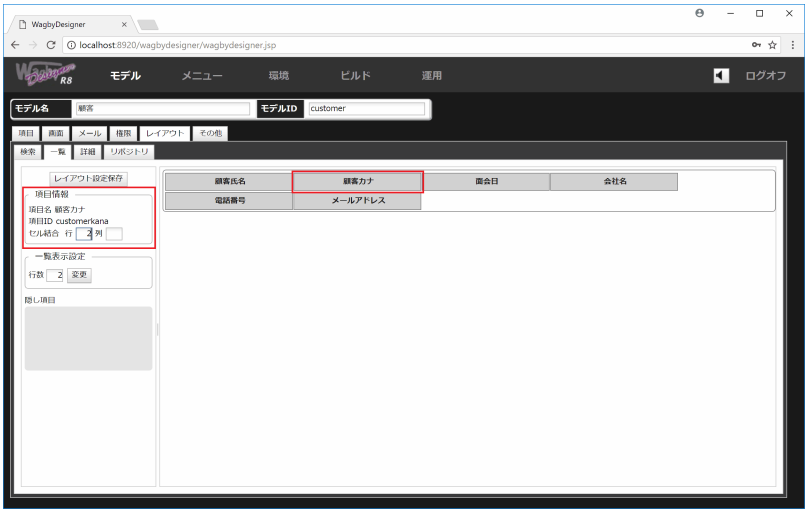


図 顧客カナ項目へのセル結合（行）の設定

設定後、「レイアウト設定保存」ボタンを押下して保存します。

リポジトリの確認

ここまでの設定が、どのようにリポジトリに反映されたかを確認します。「セル結合（行）」に ″2″ が設定されています。

「セル結合（行）」は、結合を適用する項目に数値を指定します。図の設定では、自分自身は L1 と L2 の二行にまたがった位置に配置される、という意味になります。

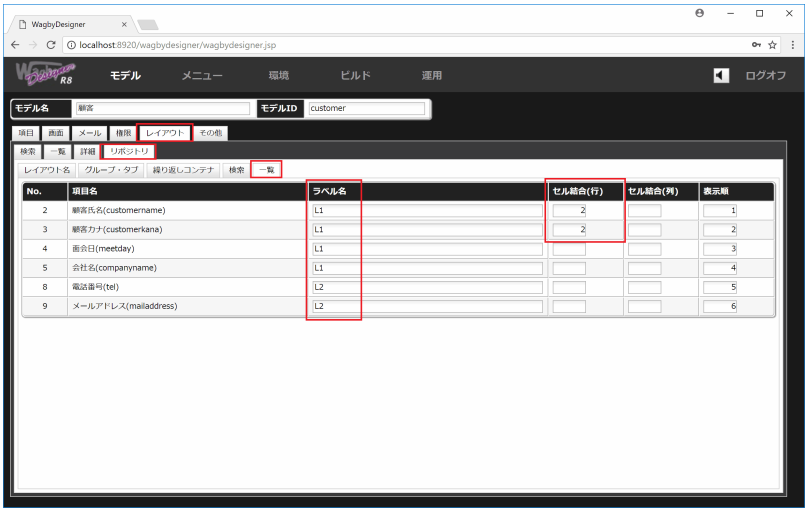


図 リポジトリの確認

一覧表示（集計）利用時の制約

一覧表示（集計）と併用する場合、集計対象の項目は上段に配置してください。上の例では、L1 と指定した行、となります。

一覧更新への適用

一覧更新機能を有効にした場合、複数段一行明細設定も適用されます。

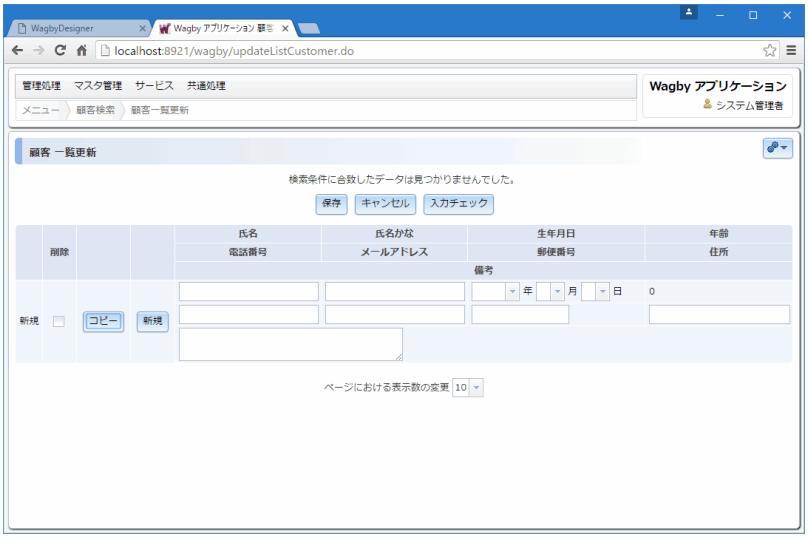


図 一覧更新画面での複数段一行明細

- テキストエリアの行数は、一覧更新画面専用の設定が用意されています。
- 一覧表示部分のスクロールタイプ（列固定、ヘッダ行固定）の指定と併用することはできません。スクロールタイプを指定するか、複数段一行明細を使うかのいずれかになります。

外部キー子モデル － 複数段一行明細(多段)レイアウト

例

外部キー関係で、子モデルを（親モデルの画面で同時に）更新することができます。このとき、複数段一行明細レイアウト指定を行うと見栄えがよくなります。

複数段一行明細レイアウトを用いない場合

図のように、横に項目が並びます。Webブラウザは可能な限り画面内にまとめようとするため、「種別」や「性別」の選択肢が縦に長く表示されています。それでも入りきれないため、横スクロールバーが表示されています。

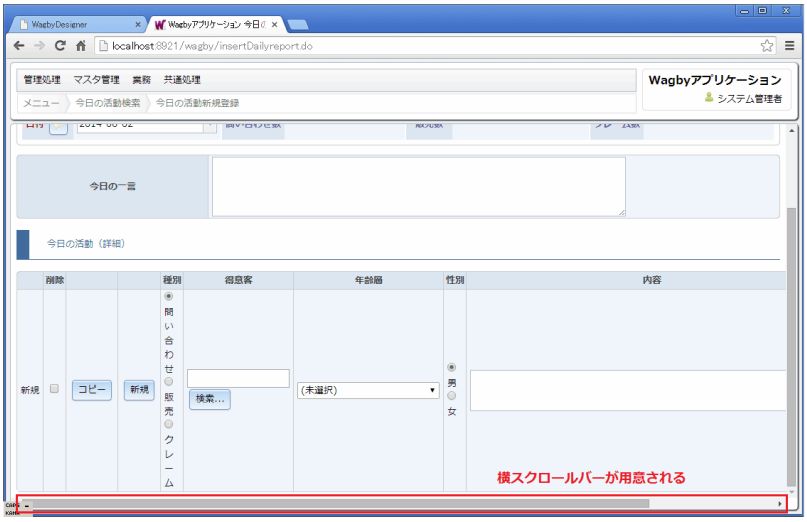


図 複数段一行明細レイアウトを用いない場合

複数段一行明細レイアウトを用いた場合

二段の設定を行った例を図に示します。「内容」と「売上」項目を二段目としています。

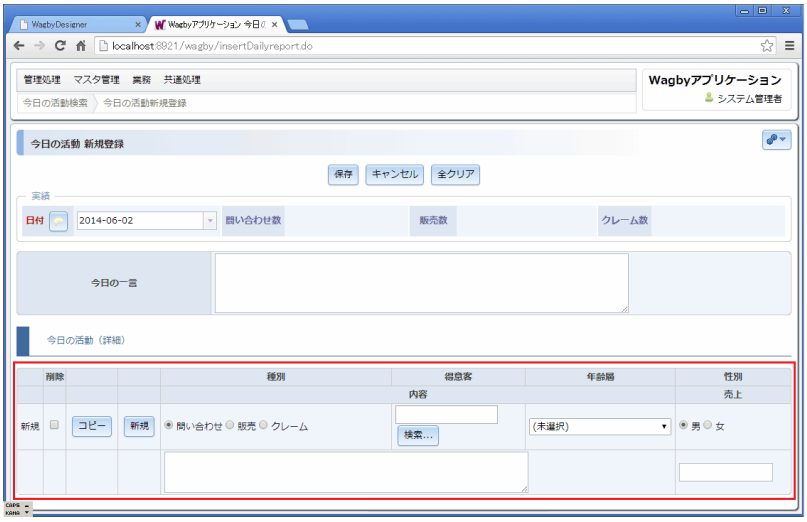


図 複数段一行明細レイアウトを用いた場合

定義方法の説明で後述しますが、1段目を "L1" とし、2段目を "L2" と扱っています。

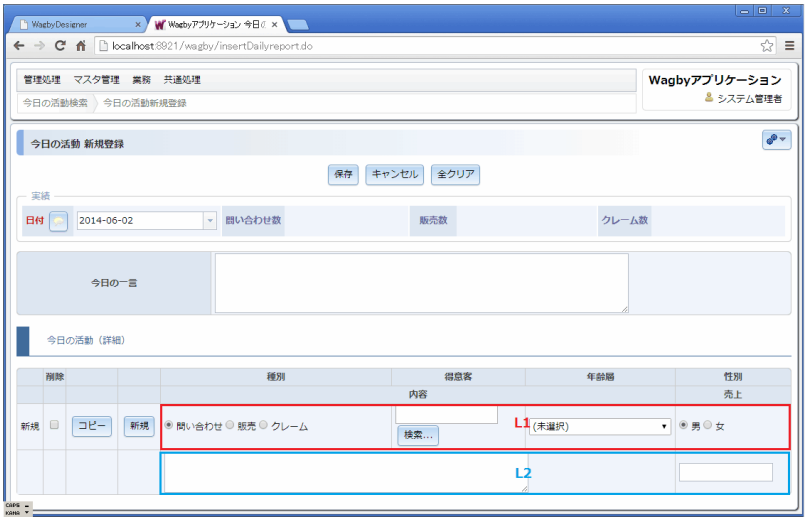


図 ラベルの設定

さらにセル結合設定も行っています。図に示すように、2段目では3つのセルを1つにまとめています。

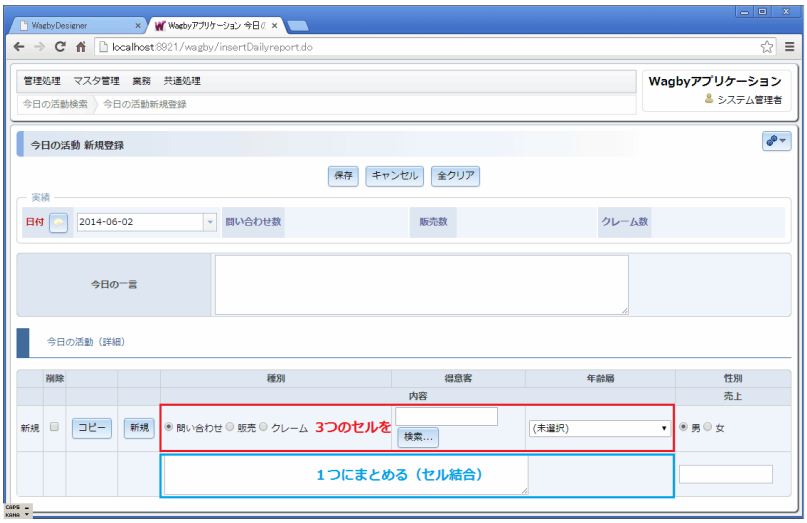


図 セル結合の設定

定義方法

上の例で用意したモデル定義を図に示します。

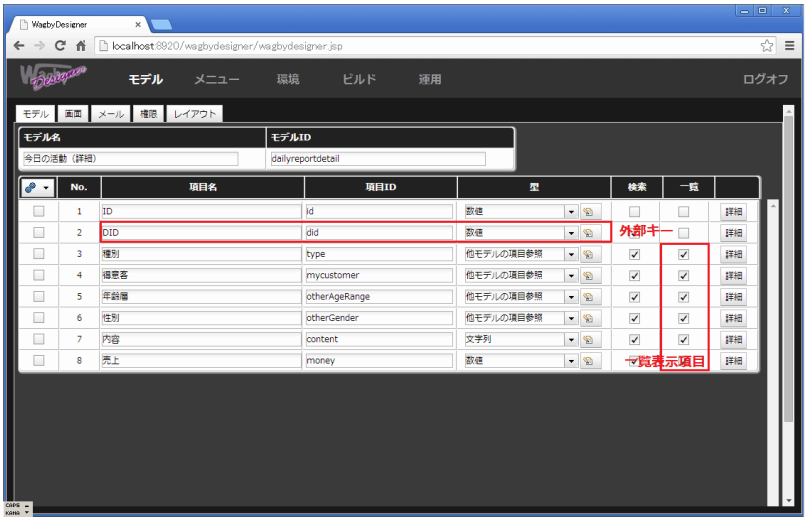


図 モデル定義

「レイアウト>一覧」を設定します。「ラベル名」は重複しない英数字の識別子とします。「セル結合」は、結合を適用する項目に数値を指定します。

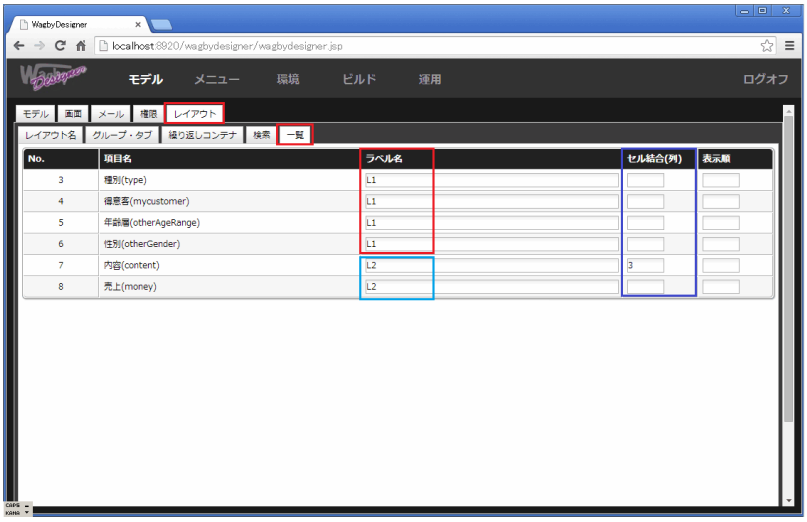


図 一覧レイアウトの設定

繰り返しコンテナ - 列幅の調整

例

繰り返しコンテナの列幅を15%, 25%, 20%, 20%, 20%と指定した例を図に示します。

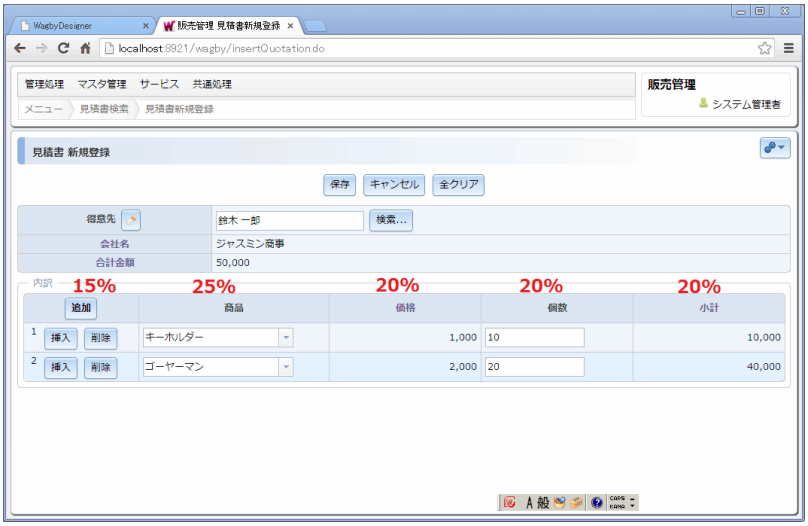


図 列幅を設定した例

定義方法

繰り返しコンテナの各項目について、モデル項目詳細定義ダイアログを開きます。 出力制御タブの詳細画面で「スタイル>ラベルの幅」を指定します。"%" を付与すると比率になります。"%" を除くと固定値（ピクセル）になります。

この設定は詳細表示画面だけでなく、新規・コピー・更新画面すべてに反映されます。

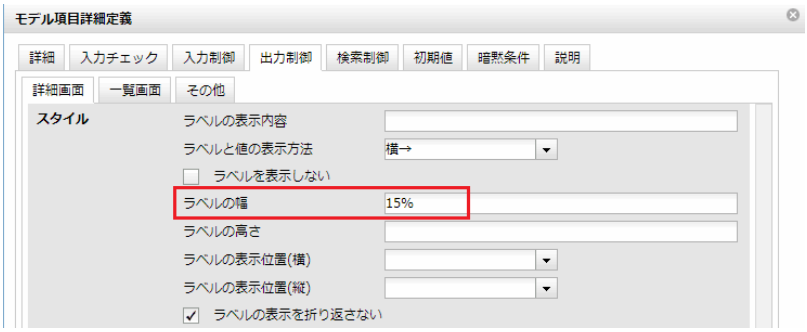


図 15%と指定した例

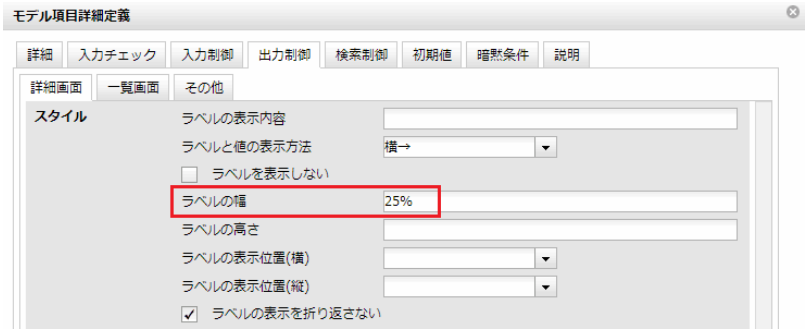


図 25%と指定した例

「繰り返しコンテナのID」列を含め、各列の合計が 100% となるように調整してください。

設計上の注意点

各列の合計が 100% となるように調整しても、画面に横スクロールバーが表示される場合は、次の点をご確認ください。

入力フィールド長

入力フィールド長の設定によって、列の幅が押し広げられることがあります。一般的な入力項目に加え、リストボックス、検索画面、日付（カレンダ利用）、テキストエリアなどでそれぞれ定義方法が異なります。

チェックボックス、ラジオボタン項目の選択肢

「ラジオボタン ＞ 折り返し数の設定」「チェックボックス ＞ 折り返し数の設定」を変更、列の幅を調整することができます。例えばこの値を ″1″ とすると、選択肢毎に折り返されます。

繰り返しコンテナ項目のソート

テーブルのヘッダに繰り返しコンテナ項目のソートアイコンが、列幅を押し広げる要因になります。

ソートが不要な項目がありましたら「繰り返しコンテナ － ソート」の設定を空にして下さい。

テーブルのヘッダのラベル長

項目名が長い場合は「ラベルの表示の折り返し」を行うようにするとよいでしょう。

繰り返しコンテナ － スクロールバーの表示（と幅、高さの設定）

横スクロールバー

例

画面サイズ（幅）が小さいと、自動的に横スクロールバーが表示される設定を行った例です。

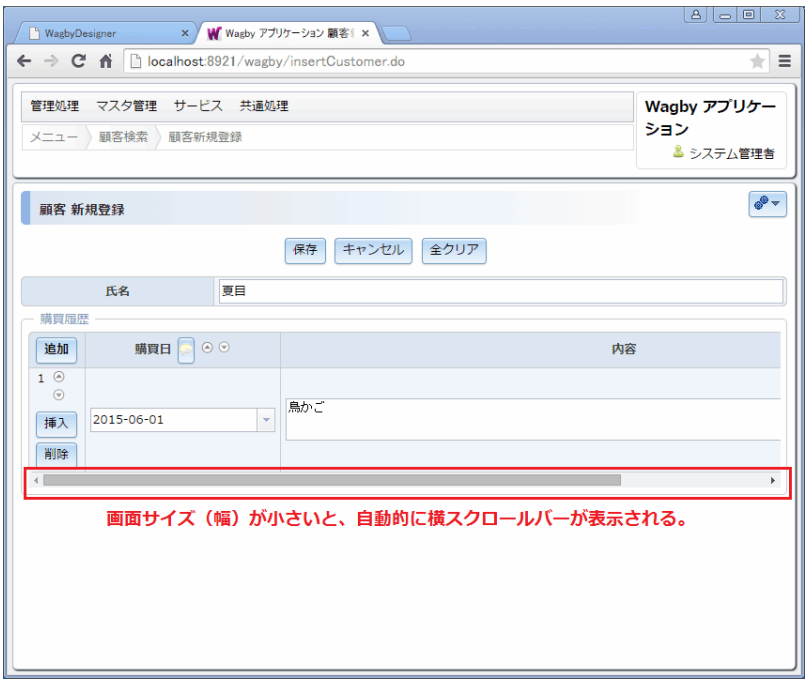


図 横スクロールバーが表示された例

定義方法

例として用意したモデルを図に示します。

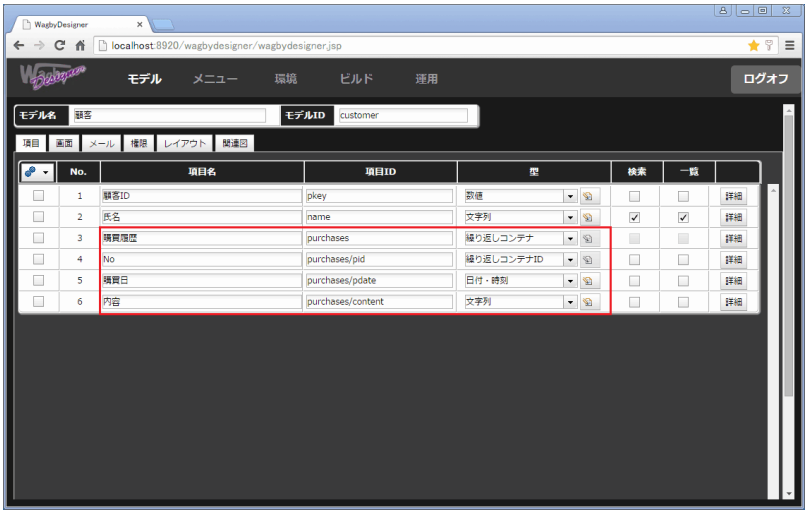


図 繰り返しコンテナを含む顧客モデルのサンプル

繰り返しコンテナ項目のモデル項目詳細定義ダイアログを開きます。

「入力制御 > 繰り返し > コンテナ内に横スクロールバーを表示する」を有効にします。

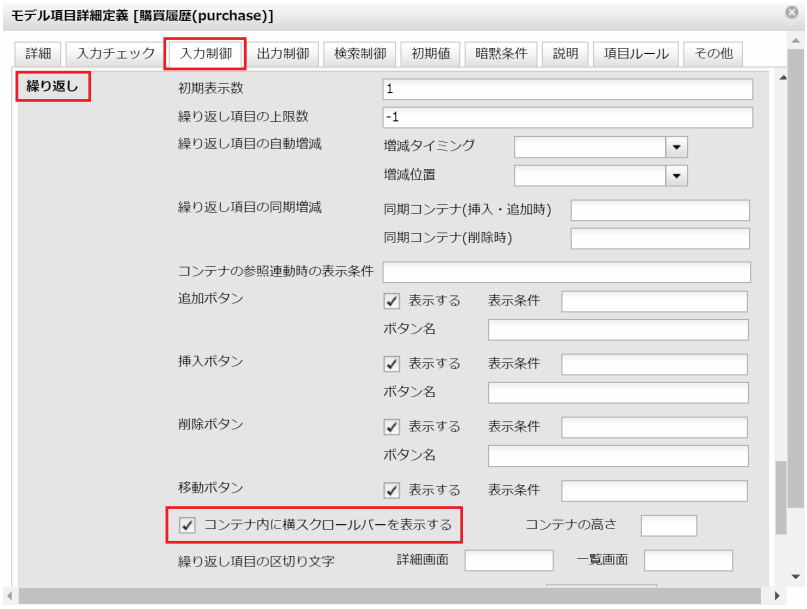


図 コンテナ内に横スクロールバーを表示する設定

高さと幅の設定 (8.3.3)

コンテナの高さと幅を指定することができます。指定した高さを超えた場合、自動的に縦スクロールバーが表示されます。

「入力制御 > 繰り返し > コンテナのサイズ」の「高さ」および「幅」に、ピクセル単位で値を設定します。単位は必須です。通常は "px" としてください。(例 300px)

スクロールバーを表示させたいときは「入力制御 > 繰り返し > コンテナ内に横スクロールバーを表示する」を有効にしてください。

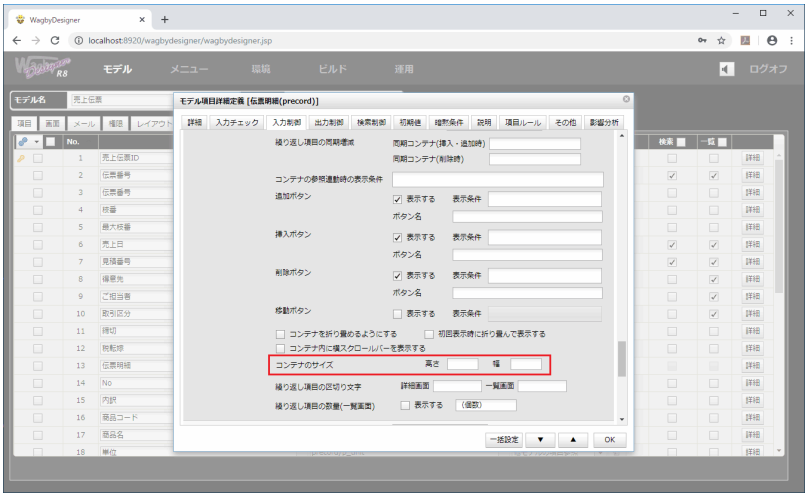


図 コンテナの高さ設定

変更履歴

・　R8.3.2までは「高さ」のみ設定できました。R8.3.3から「幅」も設定できるようになりました。

繰り返しコンテナ － タブ

例

タブを用いない場合

図のように、複数の繰り返しコンテナが（画面の下方向に）並んで表示されます。

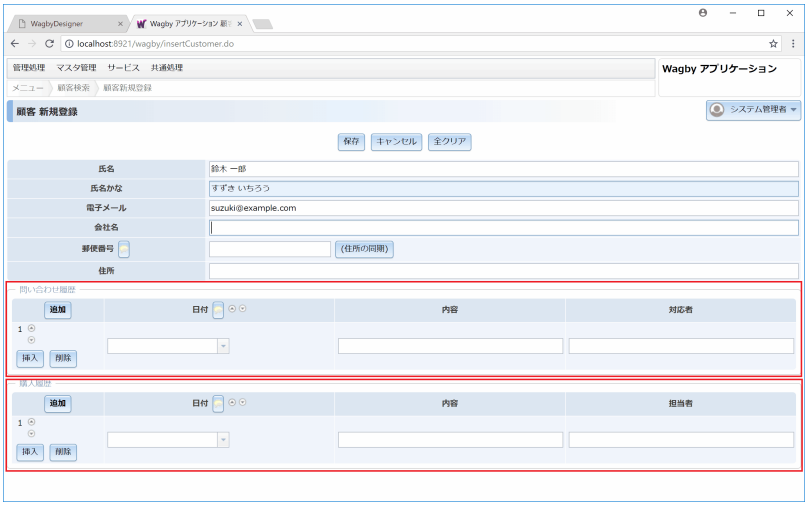


図 複数の繰り返しコンテナを用意する

タブを用いた例

図のように、複数の繰り返しコンテナがタブで管理されます。繰り返しコンテナ名がタブ名になります。

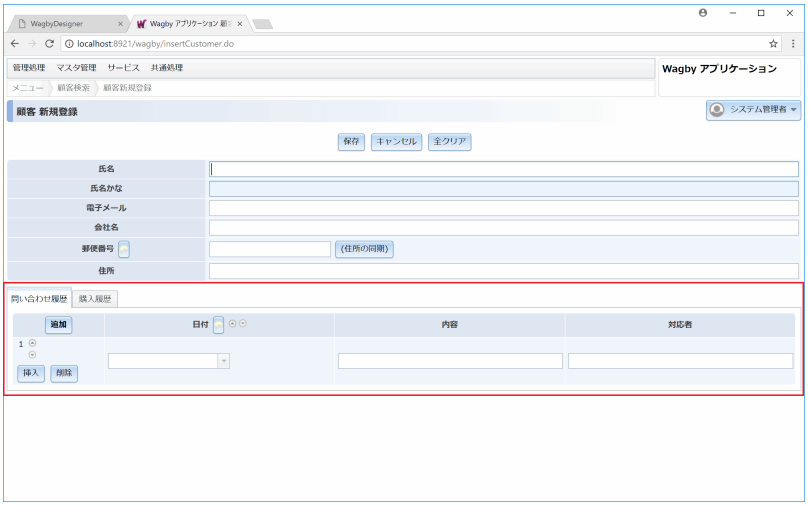


図 複数の繰り返しコンテナをタブで管理する

定義方法

図の例に用いたモデル定義を図に示します。

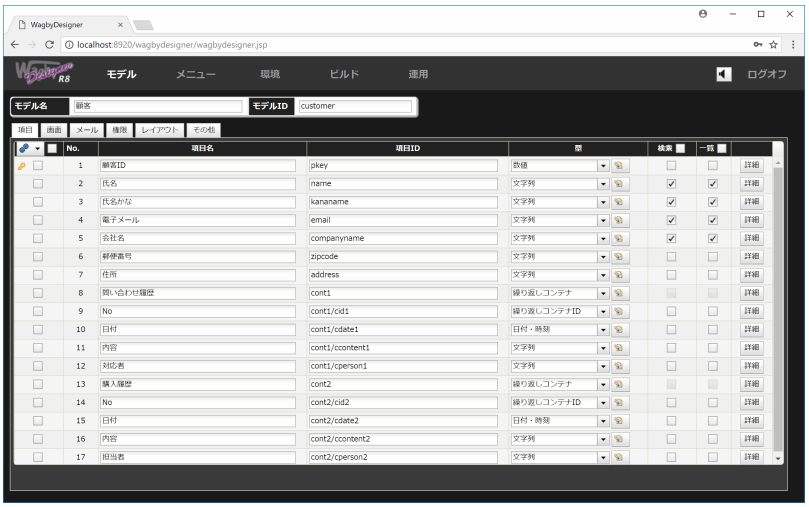


図 モデル定義

レイアウト名の書式は次の通りです。

[*レイアウト識別子:繰り返しコンテナ名]

レイアウト識別子は任意の英字です。（ここでは”T”としています。）同じタブに入る繰り返しコンテナに、同じ文字を割り当てます。

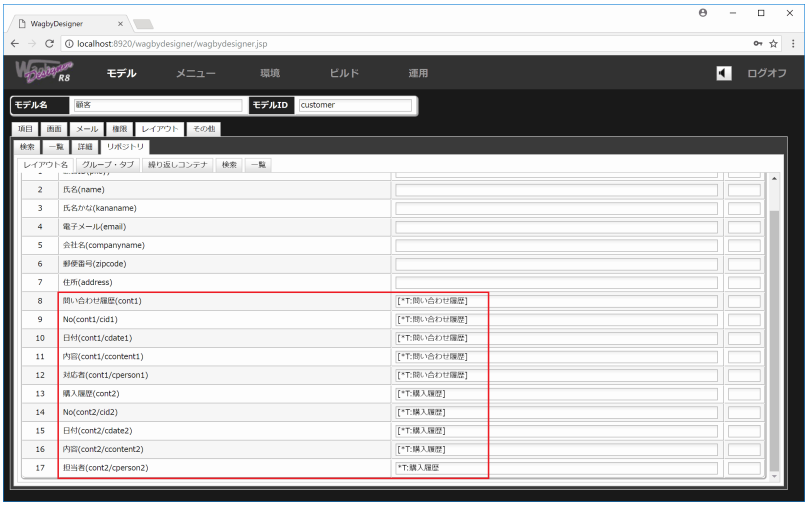


図 レイアウトの指定

レイアウト″[*T:問い合わせ履歴]″, ″[*T:購入履歴]″ をそれぞれ「タブ」と指定します。

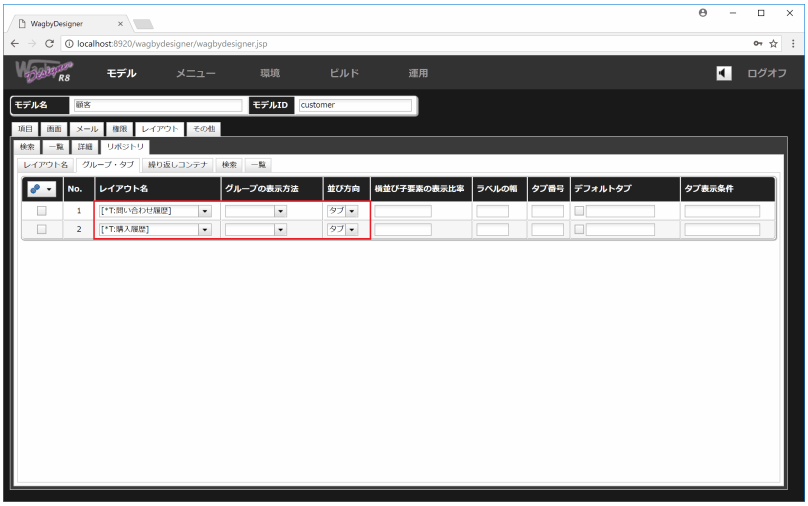


図 タブの指定

デフォルトタブを指定する

複数の繰り返しコンテナをタブで管理したとき、初期表示時は一番左側のタブがデフォルトとして選択されます。これを変更することができます。

デフォルトタブの欄をチェックします。（一つだけチェックしてください。）

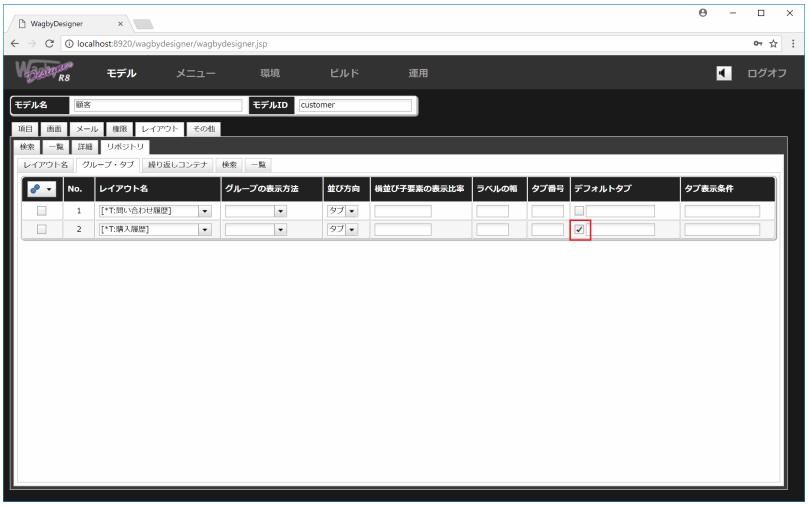


図 デフォルトタブの指定

繰り返しコンテナ － 複数段一行明細(多段)レイアウト

複数段一行明細レイアウトの設定

繰り返しコンテナの複数段一行明細レイアウトの例を図に示します。

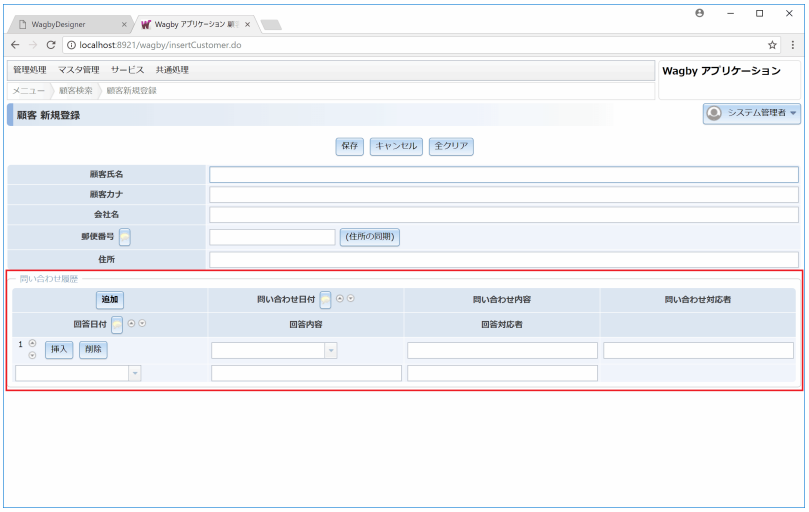


図 繰り返しコンテナの複数段一行明細レイアウト

レイアウト設計

繰り返しコンテナを構成するモデル項目を図に示します。

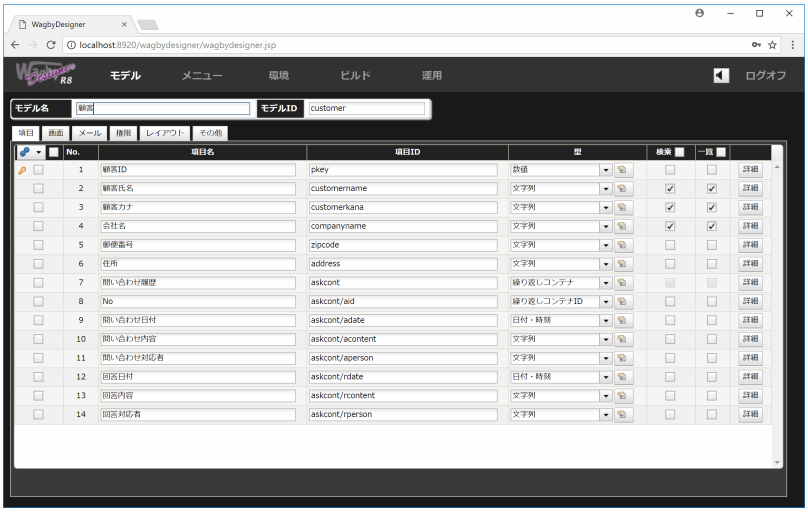


図 繰り返しコンテナの項目群

レイアウトタブ内の詳細タブを開きます。繰り返しコンテナを選択すると、画面左側の「繰り返しコンテナ」設定が開きます。

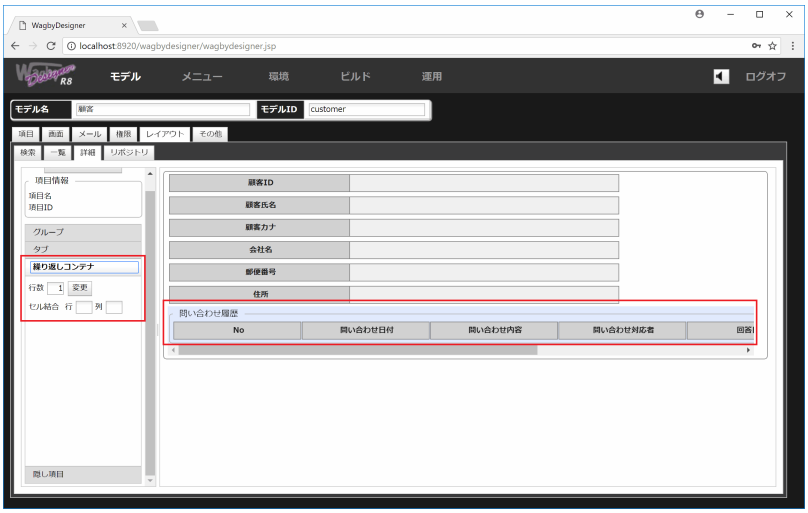


図 繰り返しコンテナの情報設定欄

行数を"2"とし、変更ボタンを押下します。二行目が用意されます。

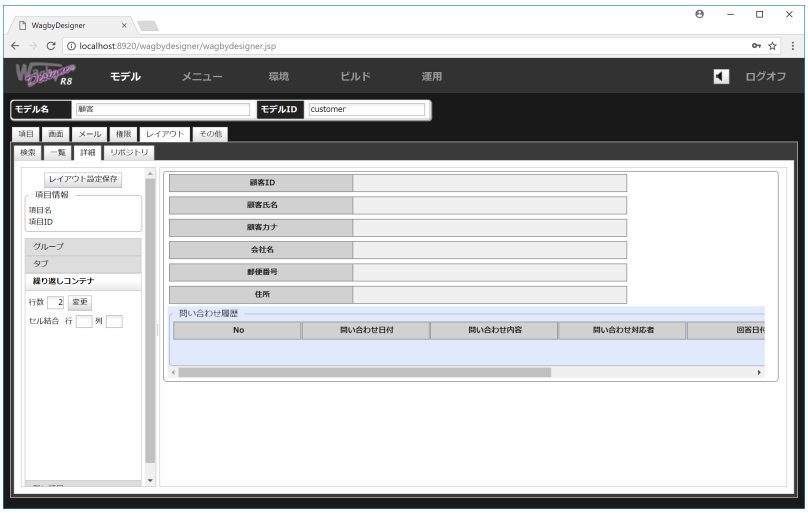


図 行数を変更する

二行目に表示したい項目群をドラッグ&ドロップで設定します。「レイアウト設定保存」ボタンを押下して、このレイアウトを保存します。

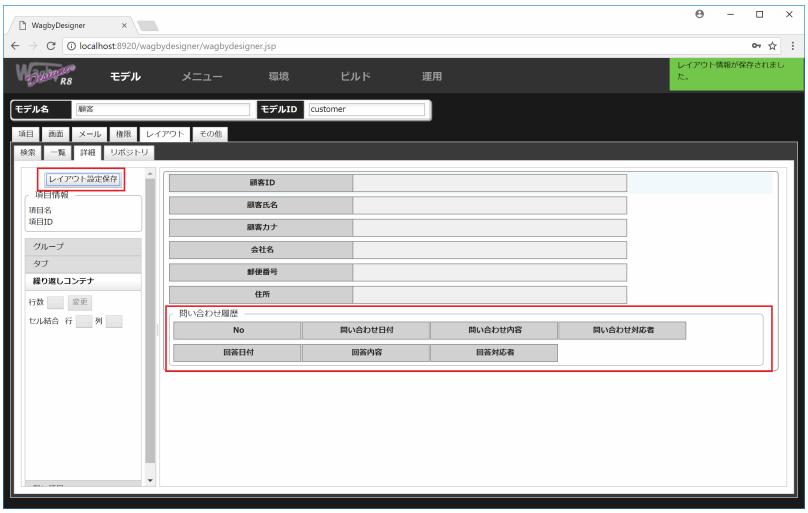


図　ドラッグ&ドロップで項目の表示並びを変更する

ご注意ください。「レイアウト設定保存」ボタンを押下せずに他の画面に切り替えた場合、これまでの編集内容は反映されません。

リポジトリの確認

ここまでの設定が、どのようにリポジトリに反映されたかを確認します。「リポジトリ」タブを開きます。「レイアウト名」タブには、"*C"ではじまる隠しレイアウト名が含まれています。文字Cに続く最初の数字は繰り返しコンテナごとに1,2,..と割り当てられます。アンダースコアに続く数字は段数になります。(180)

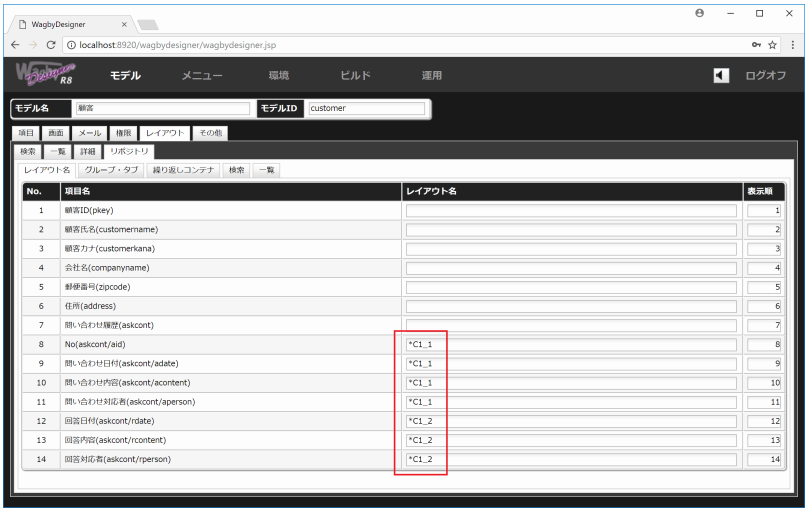


図　リポジトリの確認

列の並びを合わせる

「問い合わせ日付」項目と「回答日付」項目を並べて表示する例を説明します。

(180) 繰り返しコンテナ項目には、レイアウト名を設定しません。繰り返しコンテナの名称を除く、すべての項目名にレイアウト名（隠しラベル）が含まれるようになります。

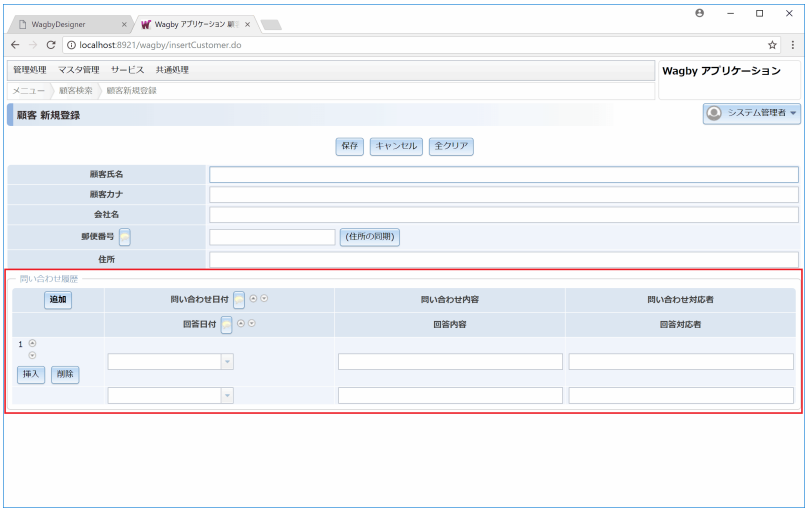


図 列を並べて表示する

リポジトリの編集

この設定は、リポジトリを直接、編集します。「レイアウト ＞ リポジトリ ＞ 繰り返しコンテナ」を開きます。

任意のラベル名を、項目の並びを合わせたい場合に指定します。ラベル名は1文字以上の英数字を使います。 ラベル位置を示す文字が同じ項目同士で位置合わせが行われます。

図の例では、″a″ ″b″ ″c″ のラベル名でそれぞれ位置合わせが行われています。

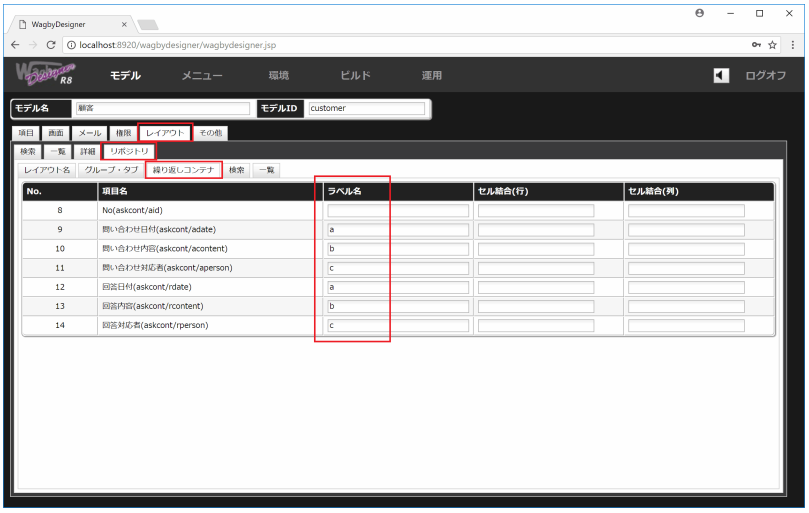


図 リポジトリを編集する

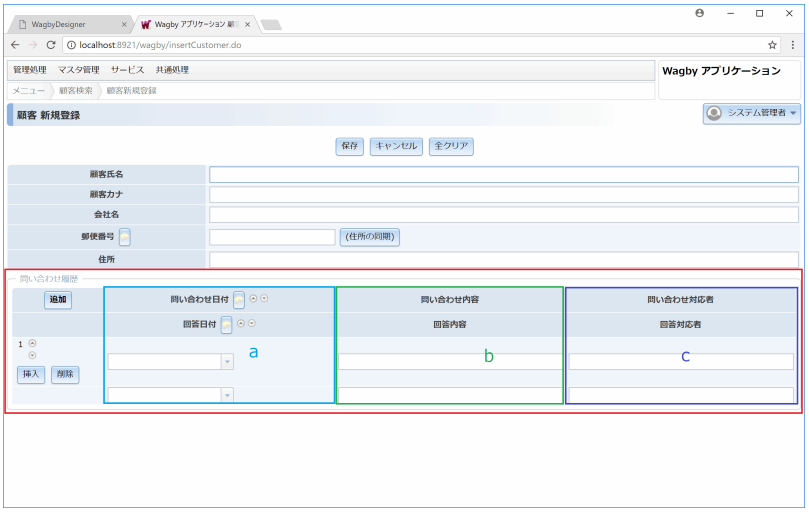


図 ラベル名による位置合わせイメージ

セル結合（行）

繰り返しコンテナIDを表現する「No」項目を、二行にまたがって表示させる例を説明します。

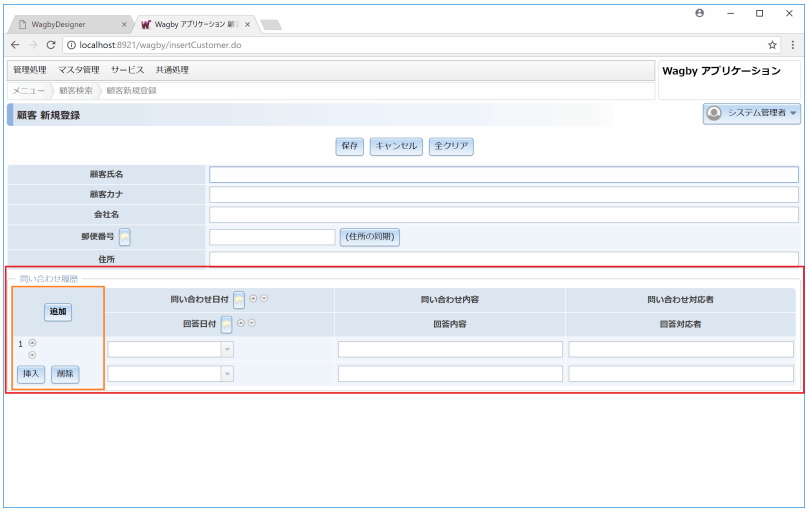


図 セル結合（行）の例

レイアウト設計

「レイアウト 〉 詳細」タブを開きます。画面右側のスクリーンエディタ部で、繰り返しコンテナ内の項目「No」を選択します。その後、画面左側の情報設定部で、この項目の「セル結合 行」に"2"を設定します。

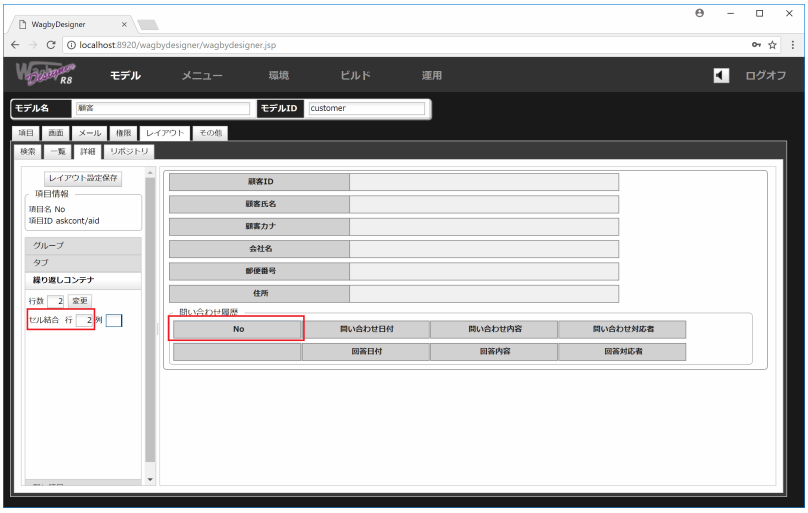


図 セル結合（行）の設定

リポジトリの確認

「No」項目の「セル結合（行）」が"2"となっています。

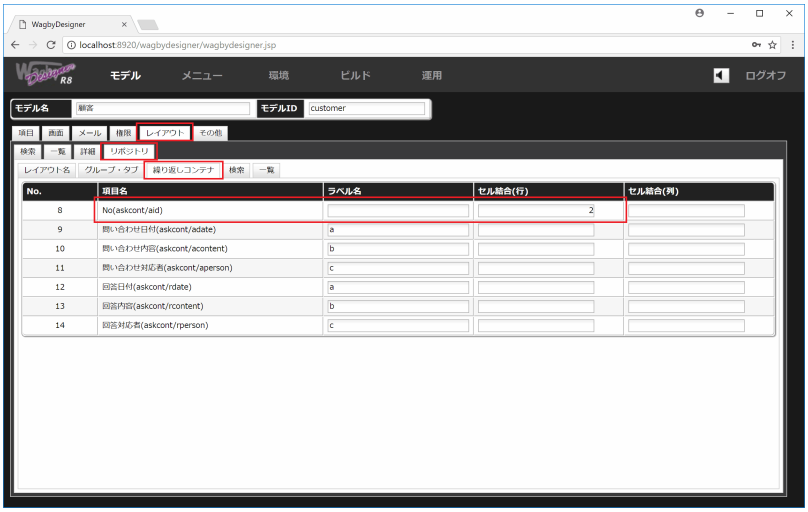


図 リポジトリの確認

ラベル名を削除する

今回の例では、先頭の「No」項目の表示が二行にまたがるため、図で設定した「ラベル名」はなくても、表示結果は同じとなります。設定を単純にするため、以降はラベル名を削除した状態とします。

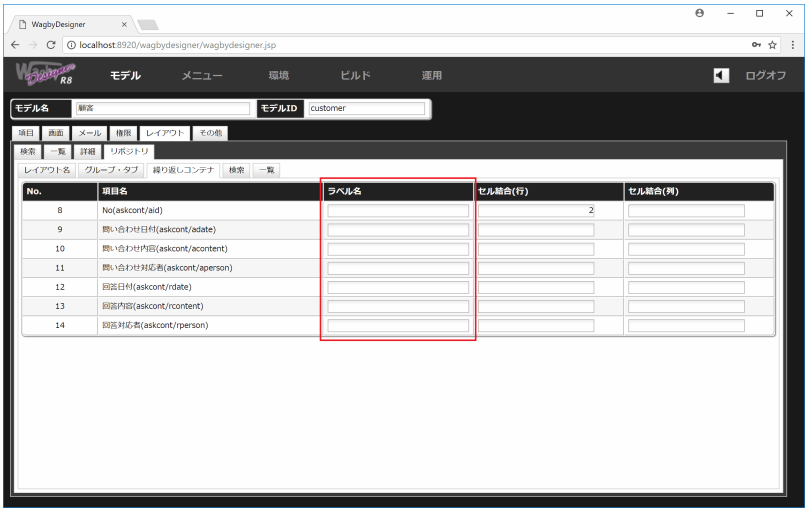


図 ラベル名を手動で消去しておく

(181)

セル結合（列）

繰り返しコンテナに新しい項目「備考」を追加した例を示します。この「備考」列は 3 列にまたがるようになっています。

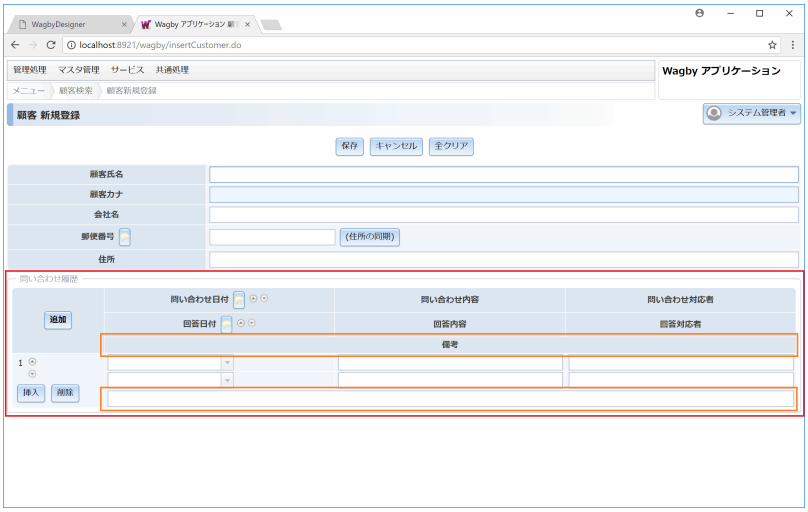


図 セル結合（列）の例

レイアウト設計

追加した項目「備考」は次のとおりです。

(181) セル結合を指定したとき、いくつかの項目が画面に表示されなくなった、などの問題が生じた場合は、ラベル名を付与してお試しください。

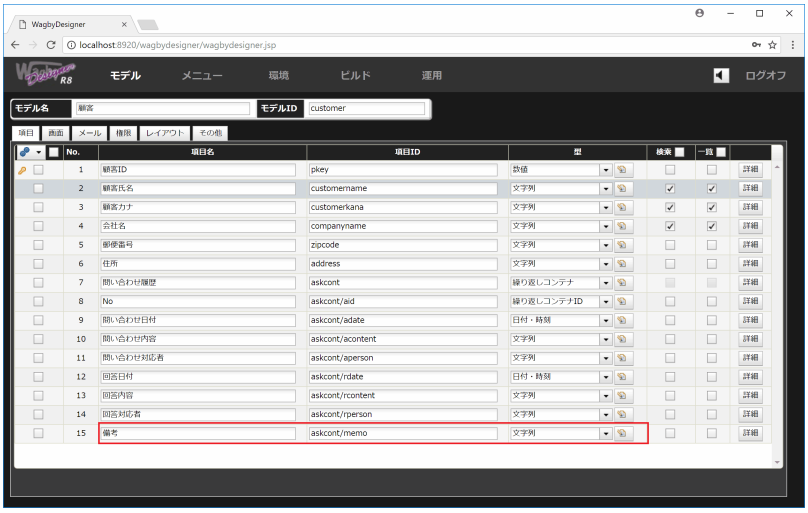


図 備考項目の追加

行数を"3"に変更します。

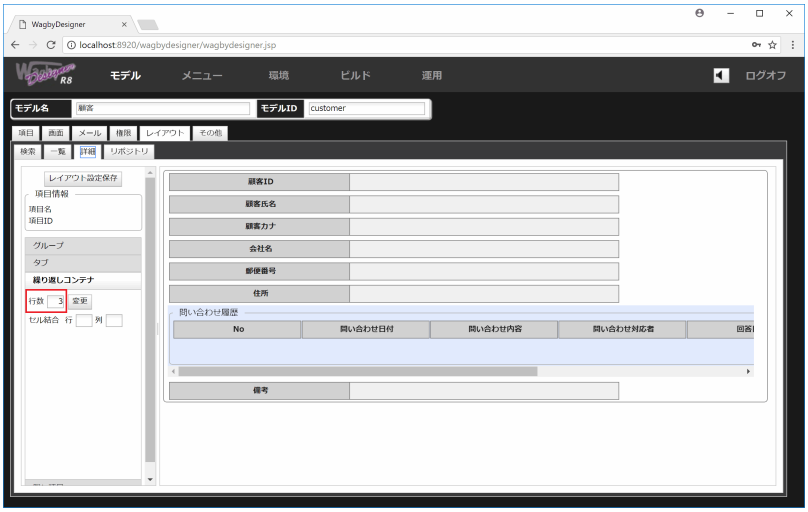


図 行数の変更

「リポジトリ ＞ レイアウト名」を直接、編集します。追加した「備考」項目のレイアウト名に手動で三行目に加えるためのラベル名を指定します。今回は"*C1_"ではじまるレイアウト名へ三行目という意味で"*C1_3"と記述します。

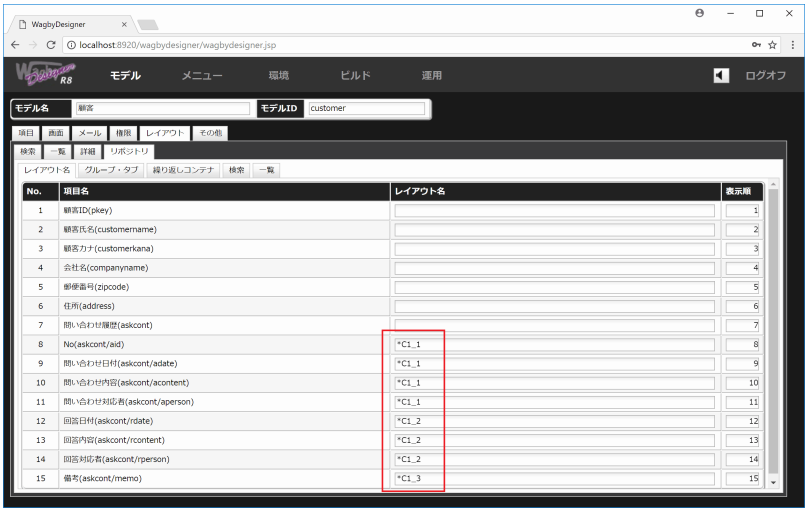


図 レイアウト名の編集

または、既存のレイアウト名をすべて削除し、再び図の画面でドラッグ&ドロップで再配置することもできます。

「No」項目にセル結合（行）を指定します。今回は3行となったので"3"とします。

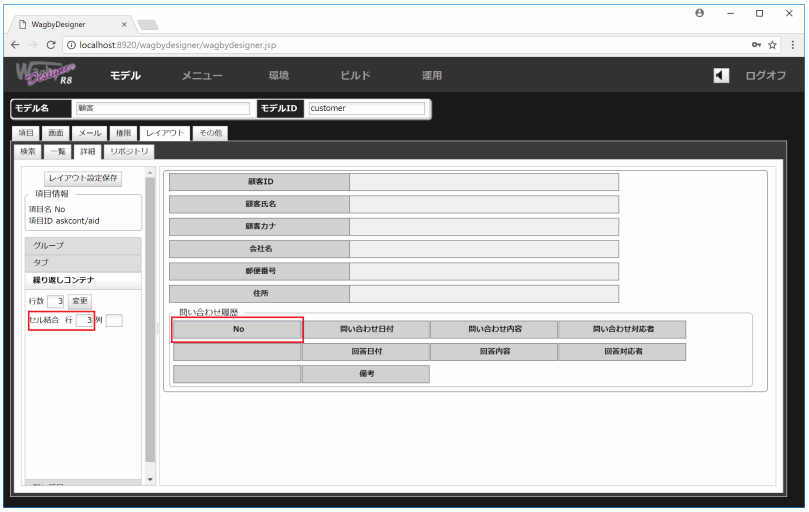


図 セル結合（行）の指定

「備考」項目にセル結合（列）を指定します。この例では3列分の表示エリアを使うため"3"とします。

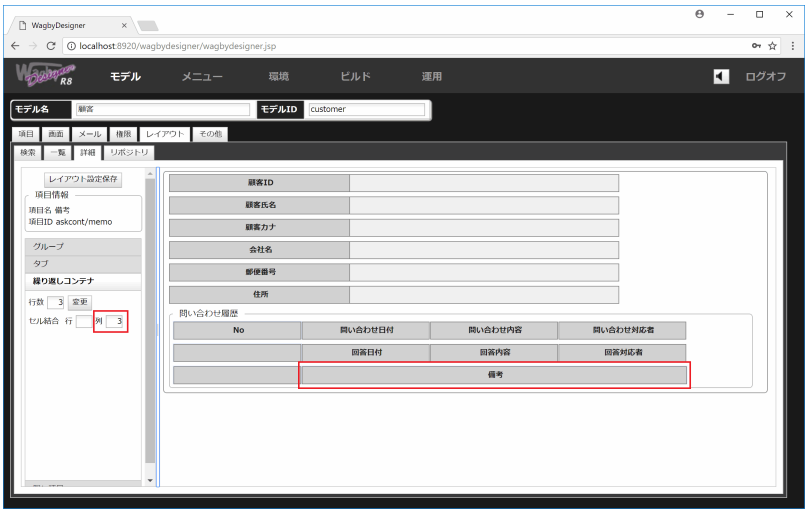


図 セル結合（列）の指定

これまでの設定を保存します。

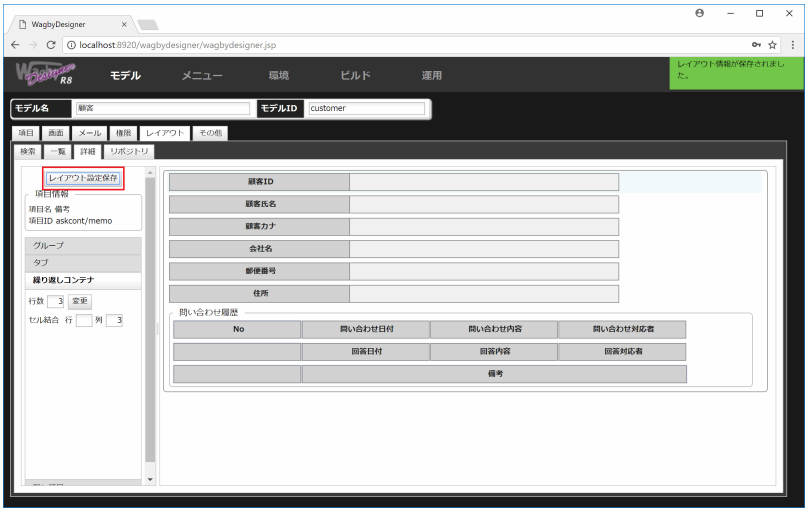


図 レイアウト設定を保存する

リポジトリの確認

ここまでの設定が、どのようにリポジトリに反映されたかを確認します。セル結合（行）とセル結合（列）に指定した値が設定されています。

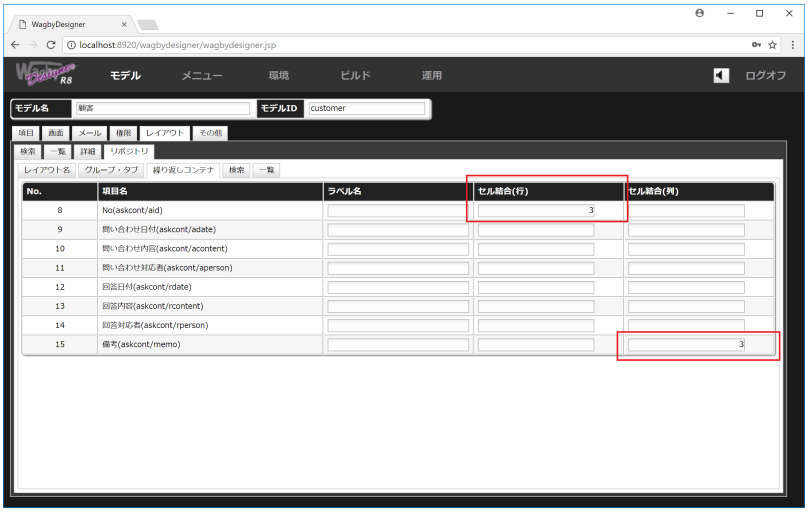


図 リポジトリの確認

トラブルシューティング

セル結合を指定したとき、いくつかの項目が画面に表示されなくなった

図のように「ラベル名」を同時に設定してお試しください。

セル結合に数値を指定するとビルドエラーになる

図や図のようにコンテナのレイアウト名を指定するとき、レイアウトを指定する項目群は一つのかたまりとしてください。途中でレイアウト名を指定しない項目を挿入することはできません。

繰り返しコンテナ － タブと複数段一行明細(多段)レイアウト

例

複数段一行明細レイアウトを適用した複数の繰り返しコンテナをタブにまとめた例を示します。

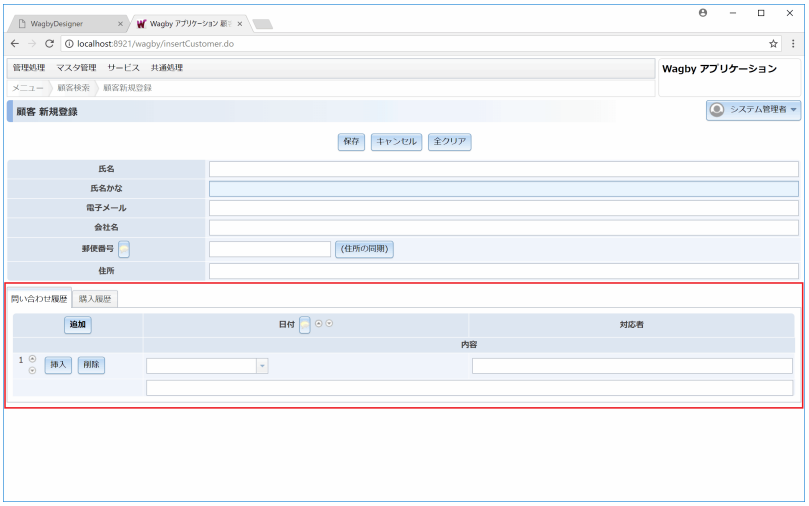


図 複数段一行明細レイアウトを適用した複数の繰り返しコンテナをタブにまとめた例

定義方法

項目の定義

複数の繰り返しコンテナ「問い合わせ履歴」「修正履歴」を含むモデル項目を図に示します。

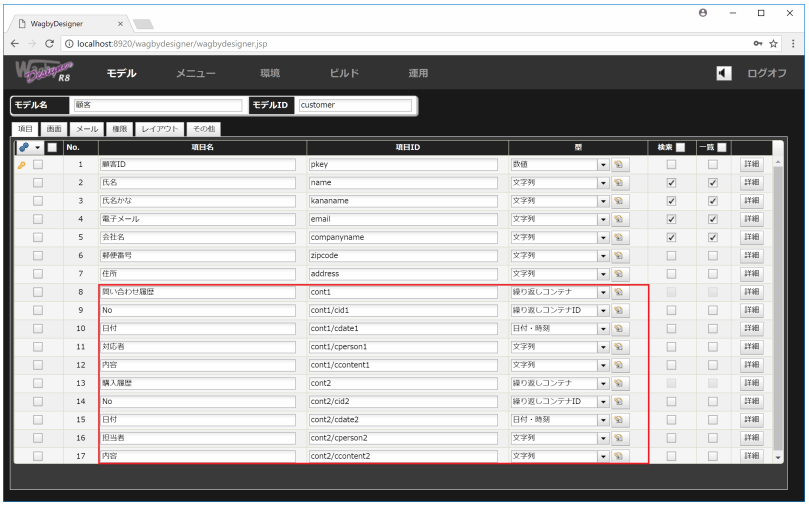


図 複数の繰り返しコンテナを定義する

レイアウト名

ここではすべての繰り返しコンテナを一つのタブにまとめています。タブ名を「*T」としています。

(182)

(182) レイアウト名の書式は［*レイアウト識別子:繰り返しコンテナ名］となります。詳細は「繰り返しコンテナ － タブ」をお読みください。

さらに、それぞれの繰り返しコンテナで複数段一行明細レイアウトを設定しています。繰り返しコンテナ毎に、それぞれ「*C1,*C2」「*D1,*D2」としています。

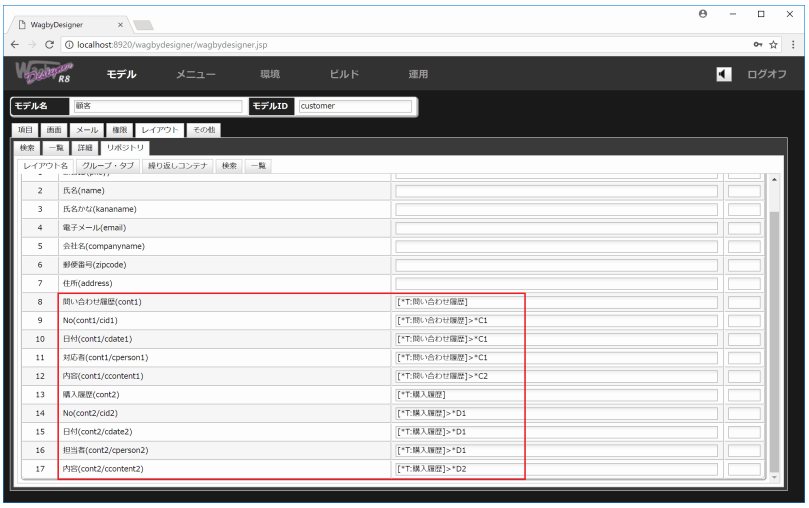


図 レイアウト名の設定

グループ・タブ

図で用意した"[*T:問い合わせ履歴]", "[*T:購入履歴]" をそれぞれ「タブ」と指定します。

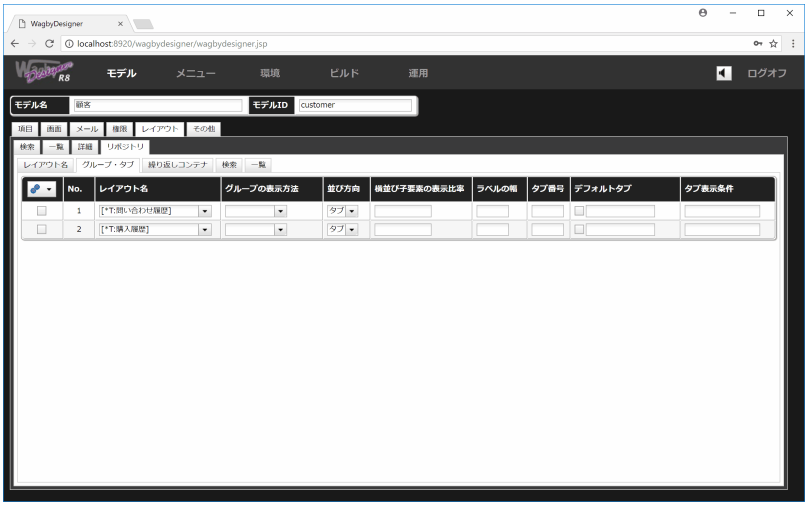


図 タブの設定

繰り返しコンテナのレイアウト - ラベル名

各コンテナにラベル名「a」「b」を指定します。

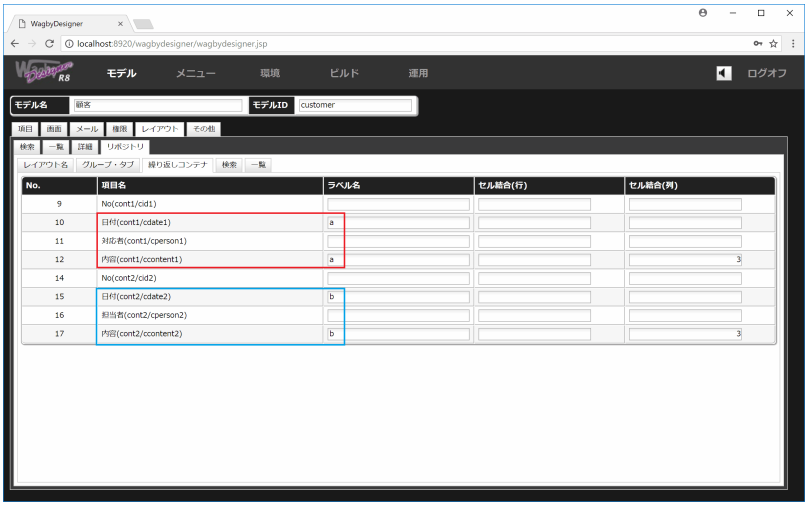


図 ラベル名の設定

ラベル名「a」の位置合わせイメージを図に示します。「日付」項目と「内容」項目の位置を合わせています。

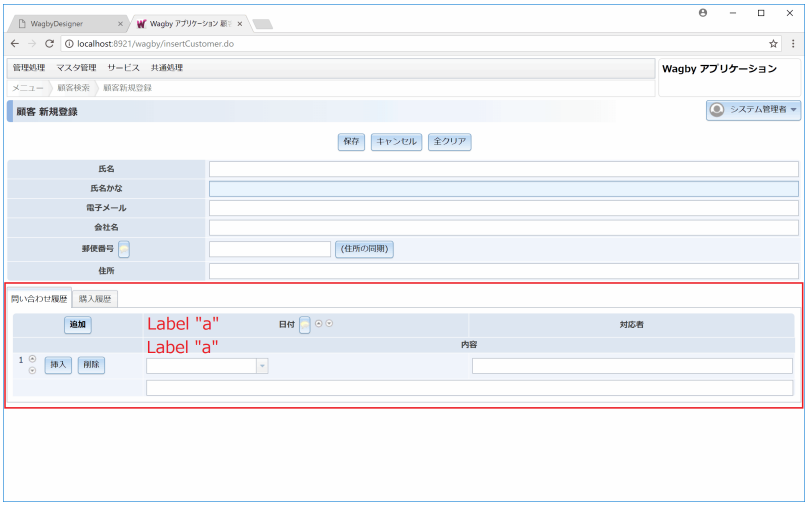


図 ラベル名の位置合わせイメージ

繰り返しコンテナのレイアウト - 項目の結合

項目「内容」は、その上の2つの項目「日付」「対応者」と同じ領域をとるものとします。図のように「3」と記載します。 この指定により、水平方向の列を一つにまとめることができます。

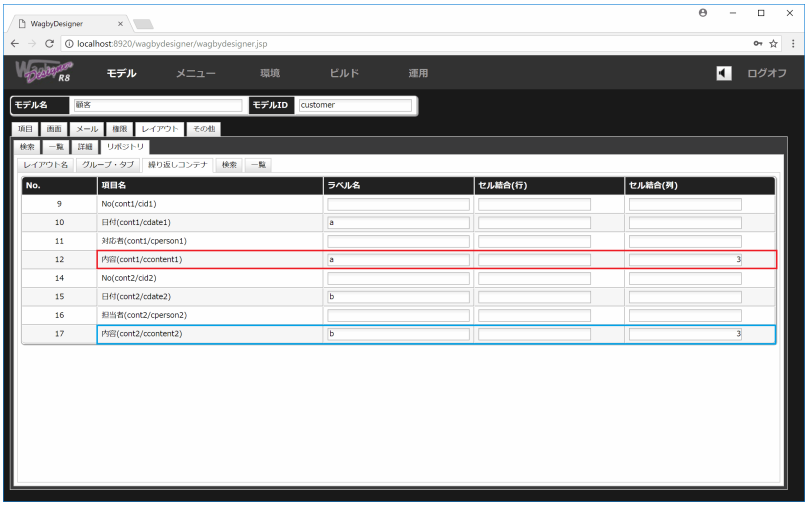


図 セル結合の設定

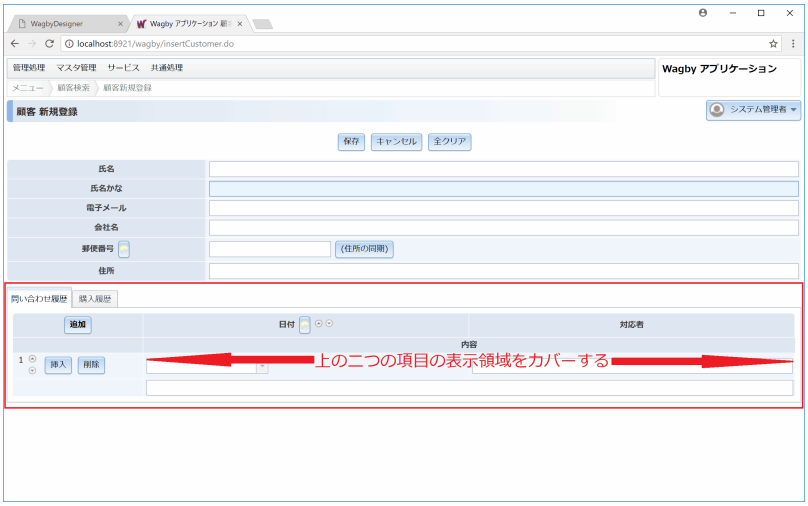


図 水平方向の結合イメージ

繰り返しコンテナ － ヘッダ部のみの複数段一行明細(多段)レイアウト

例1

繰り返しコンテナのヘッダ部の複数段一行明細レイアウトの例を図に示します。 ここでは「寸法」カラムの中に「縦」「横」というカラムが用意されています。なお、データ部分は複数段一行明細レイアウトではありません。

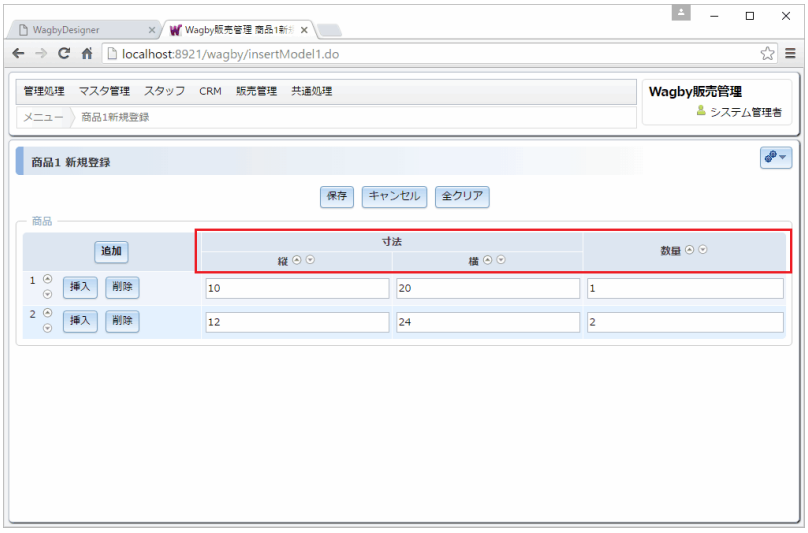


図 繰り返しコンテナヘッダ部の複数段一行明細例レイアウト(1)

定義方法1

項目の定義

繰り返しコンテナ「商品」を構成するモデル項目を図に示します。赤枠の項目に着目します。

図 項目の定義

段組みの設定

「出力制御 > 詳細画面 > スタイル > ラベルの表示内容」に特別な表記ルール

[寸法/縦]

と記述します。同じく、隣の項目には

[寸法/横]

と記述します。すると〝寸法〟の部分がまとまります。

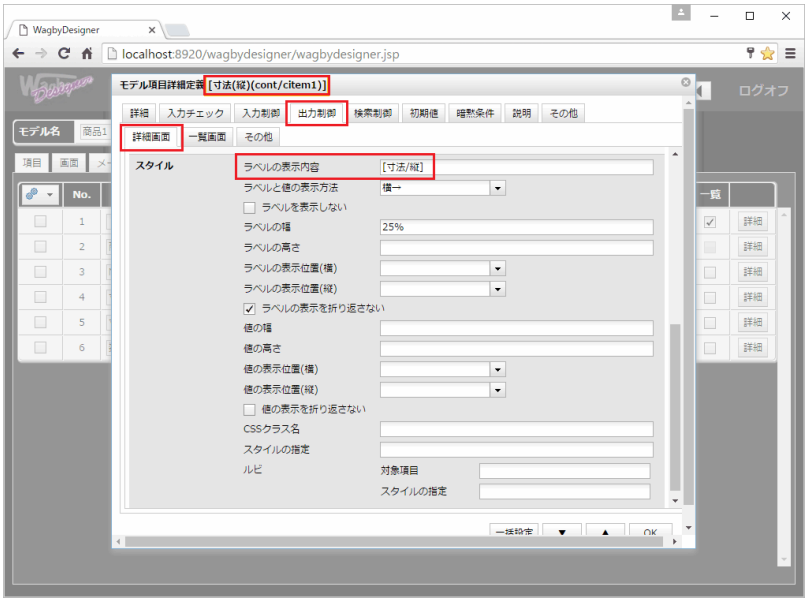


図 「寸法(縦)」項目の設定

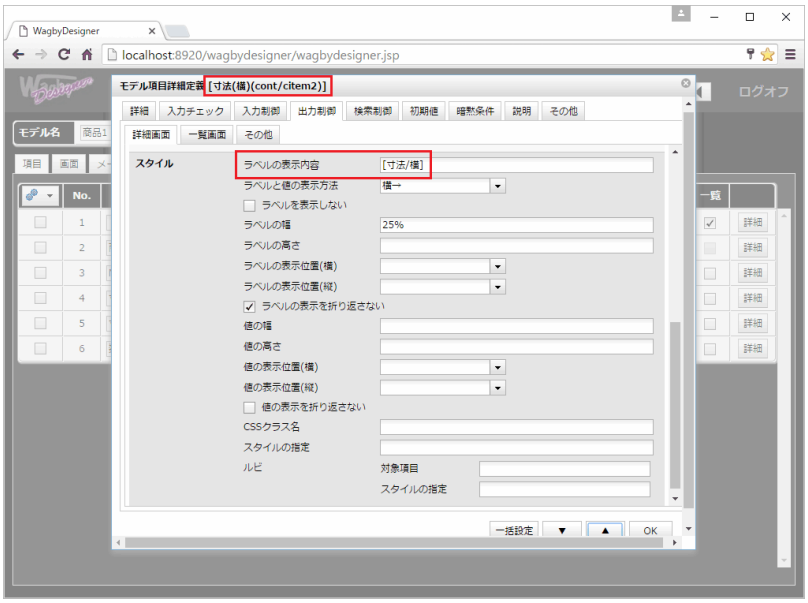


図 「寸法(横)」項目の設定

例2

ここでは「規格」カラムの中に「縦」「横」「高さ」という3つのカラムを用意した例を説明します。

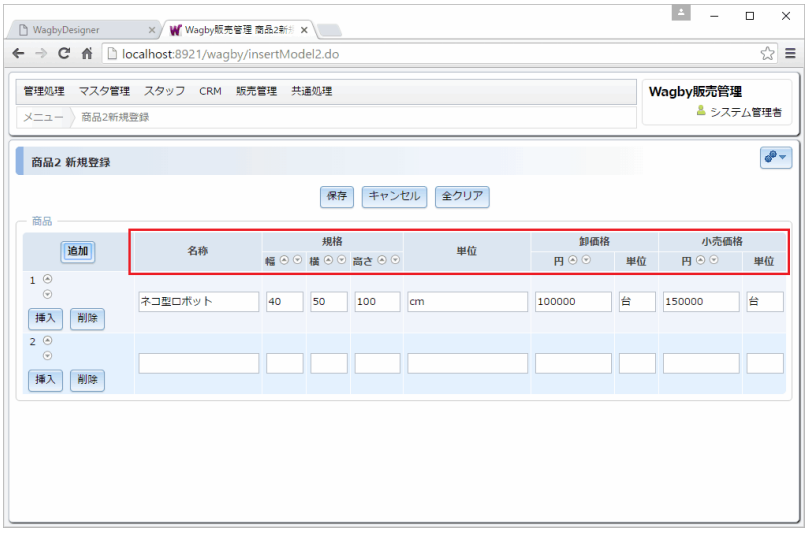


図 繰り返しコンテナヘッダ部の複数段一行明細レイアウト(2)

定義方法2

項目の定義

繰り返しコンテナ「商品」を構成するモデル項目を図に示します。赤枠の項目に着目します。

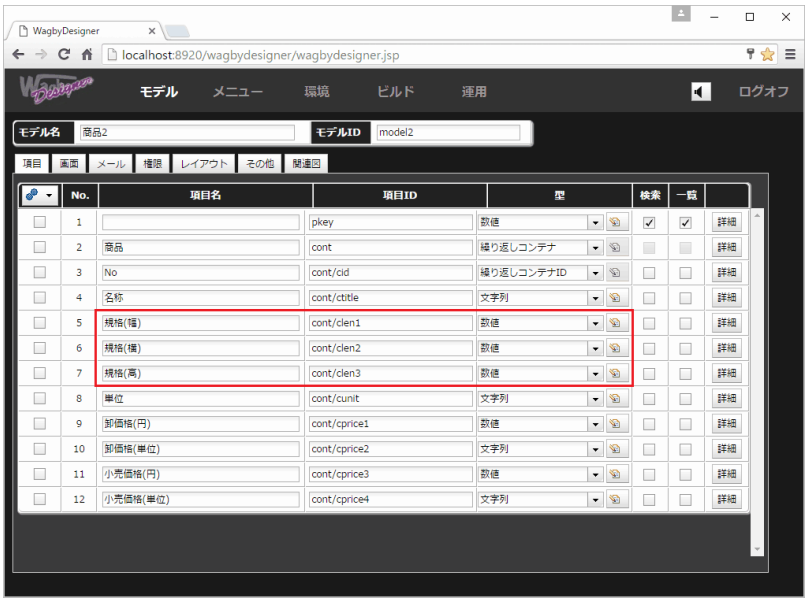


図 項目の定義

段組みの設定

「出力制御 > 詳細画面 > スタイル > ラベルの表示内容」に

```
[規格/幅]
[規格/横]
[規格/高]
```

をそれぞれ指定します。すると〝規格〟の部分がまとまります。

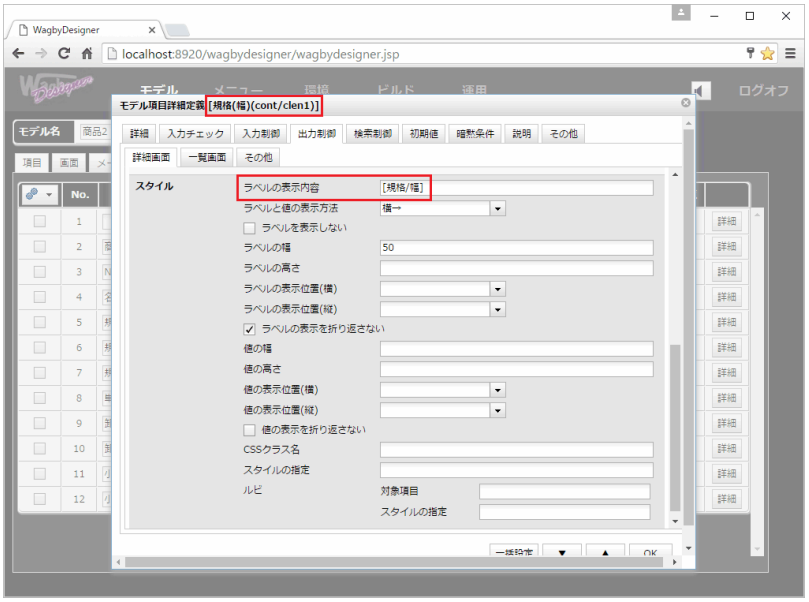


図 「規格(幅)」項目の設定

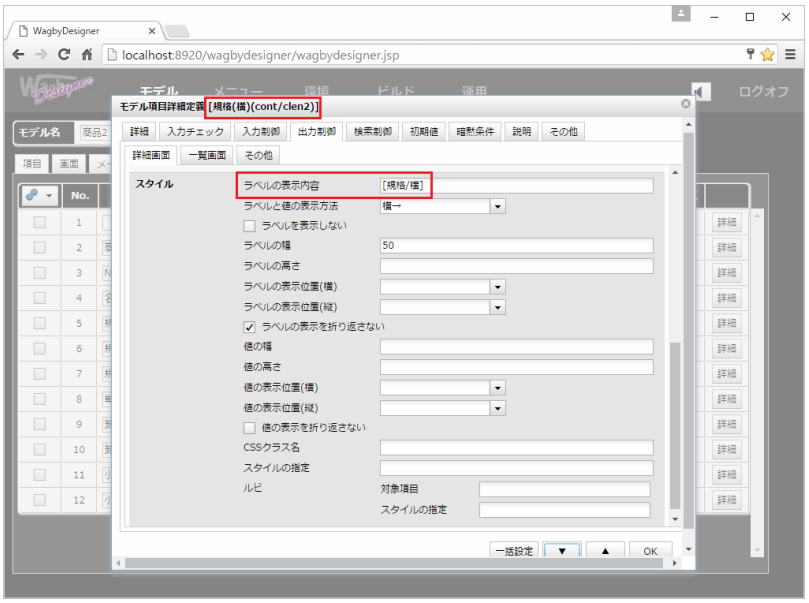


図 「規格(横)」項目の設定

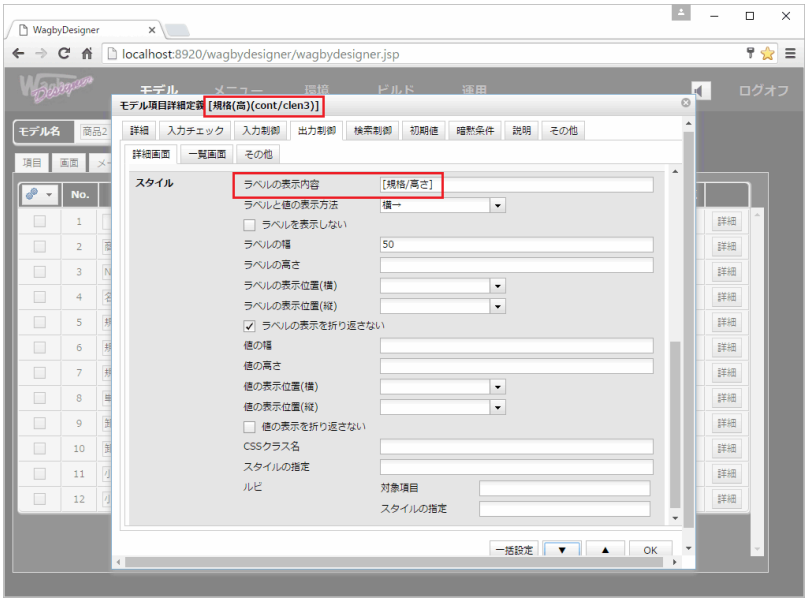


図 「規格(高)」項目の設定

例3

ここでは三段のレイアウト例を説明します。

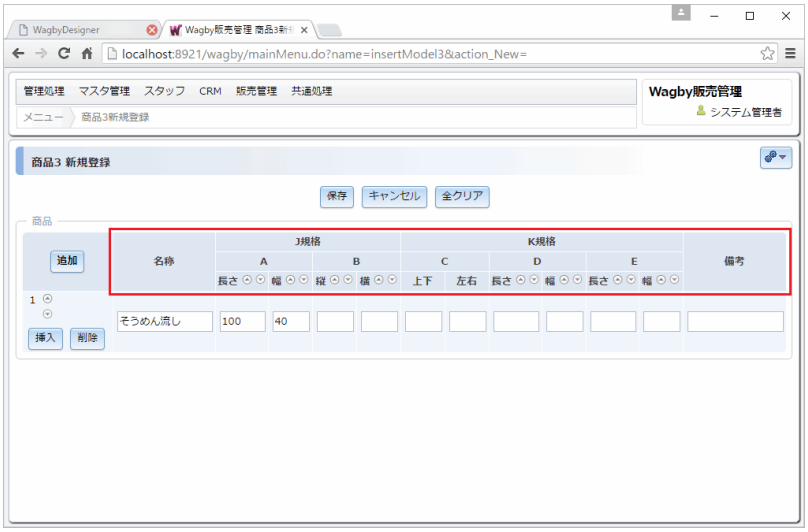


図 繰り返しコンテナヘッダ部の複数段一行明細レイアウト(3)

定義方法3

項目の定義

繰り返しコンテナ「商品」を構成するモデル項目を図に示します。赤枠の項目に着目します。

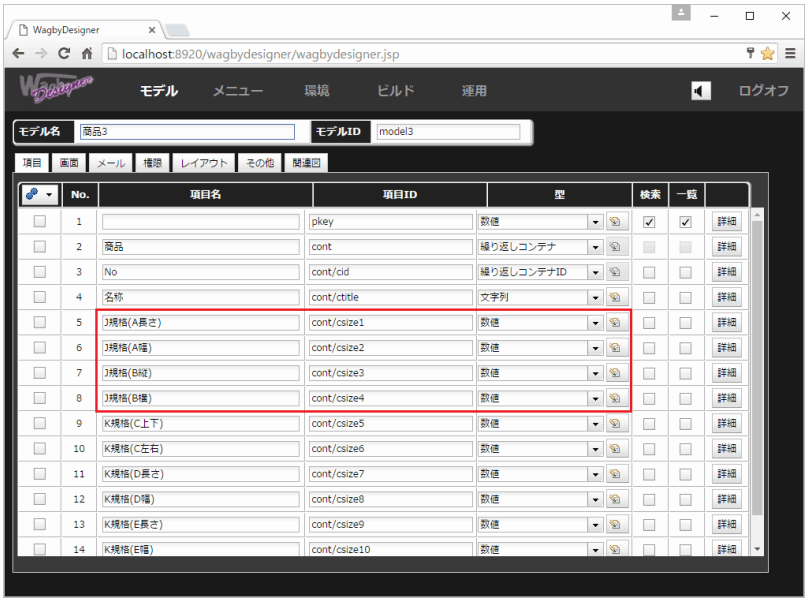


図 項目の定義

段組みの設定

「出力制御 > 詳細画面 > スタイル > ラベルの表示内容」に

```
[J規格/A/長さ]
[J規格/A/幅]
[J規格/B/縦]
[J規格/B/横]
```

をそれぞれ指定します。すると "J規格" の部分がまとまり、さらに "A" と "B" でそれぞれまとまります。

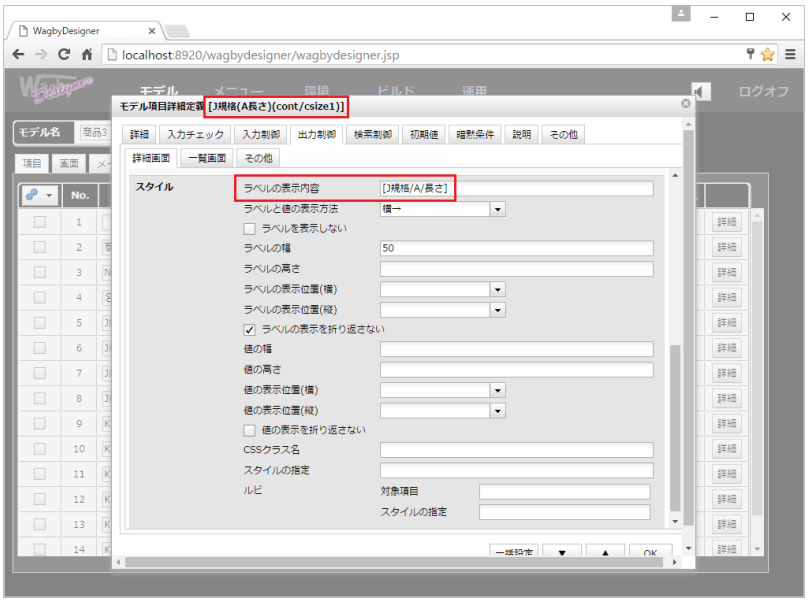


図 「J規格(A長さ)」項目の設定

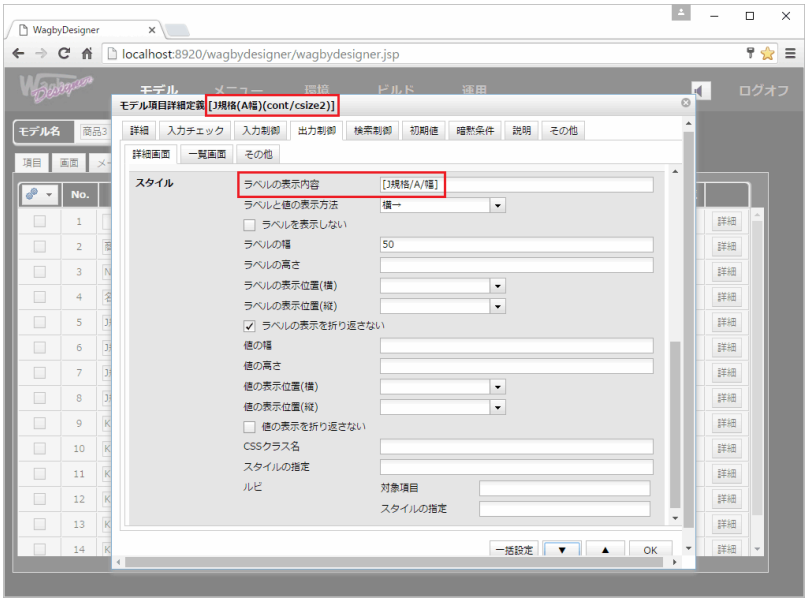


図 「J規格(A幅)」項目の設定

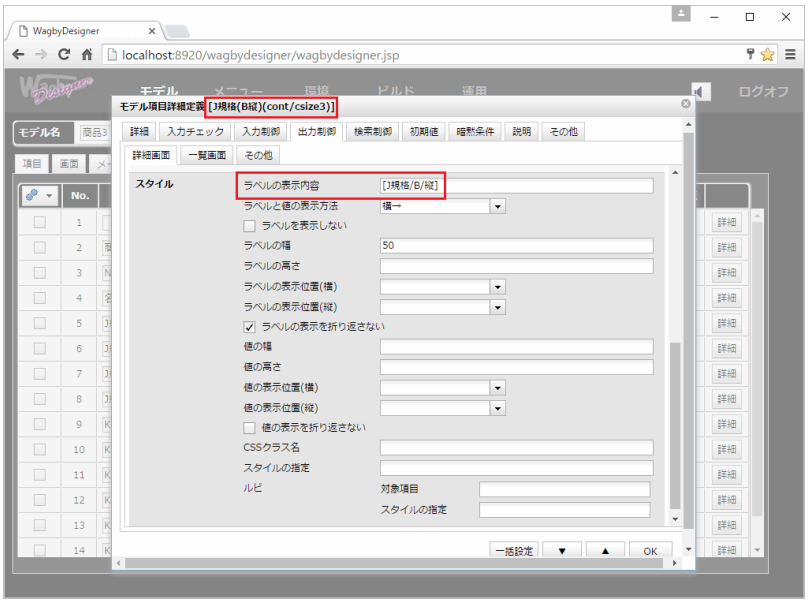


図 「J規格(B縦)」項目の設定

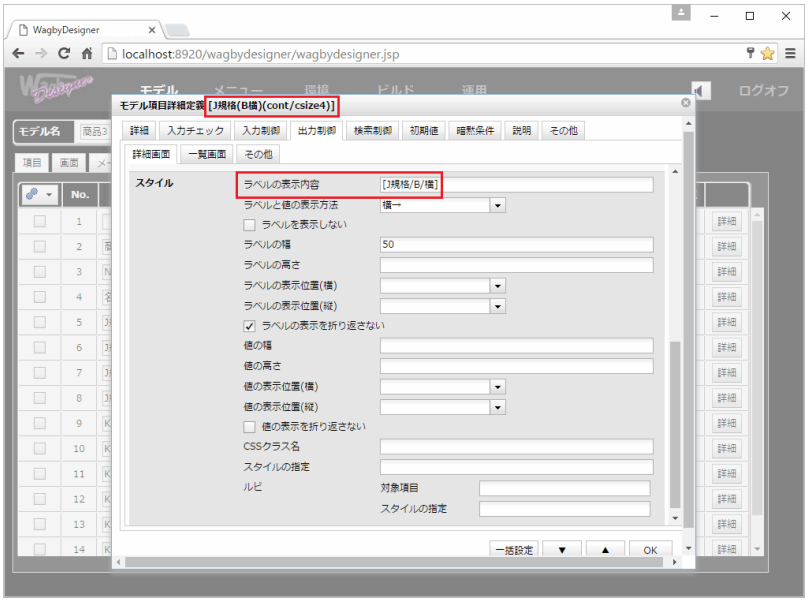


図 「J規格(B横)」項目の設定

仕様・制約

- この設定は〝繰り返しコンテナ 〉 複数段一行明細レイアウト〟と併用することはできません。ヘッダ部を複数段一行明細レイアウトとしたときは、データ部は1行となります。

繰り返しコンテナ － 縦並び

縦並びレイアウト

例

繰り返しコンテナ内の項目を縦並びにした例を図に示します。

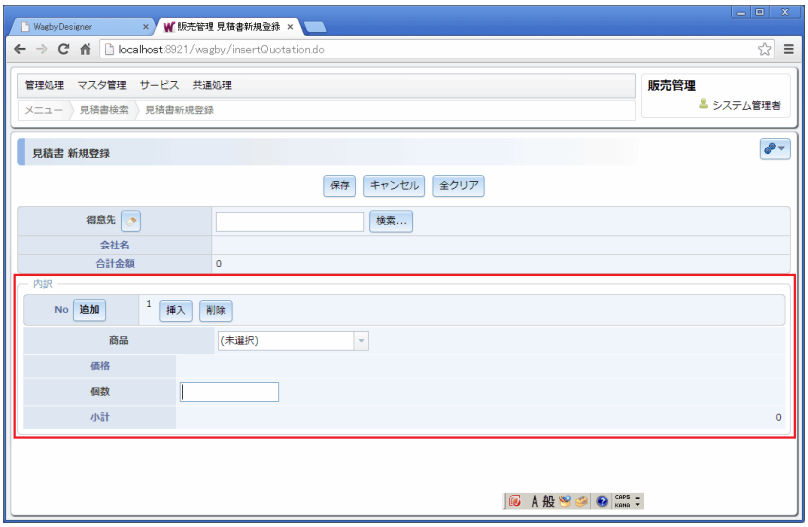


図 繰り返しコンテナの縦並びレイアウト

定義方法

繰り返しコンテナの定義を図に示します。

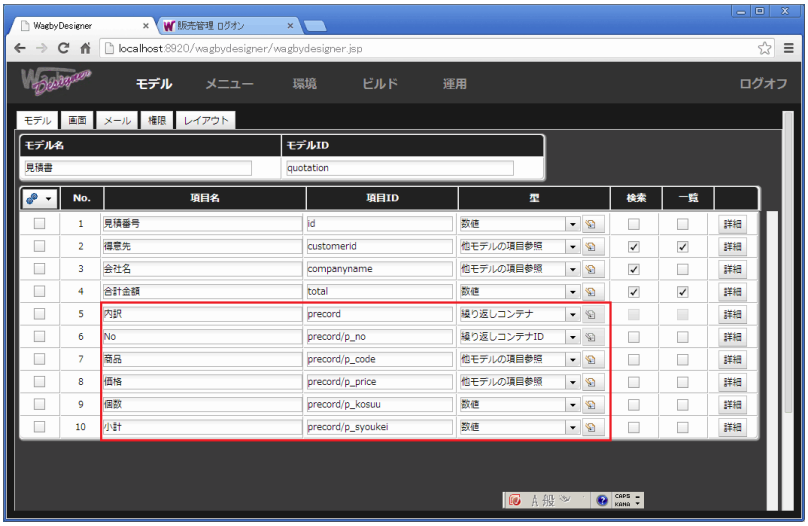
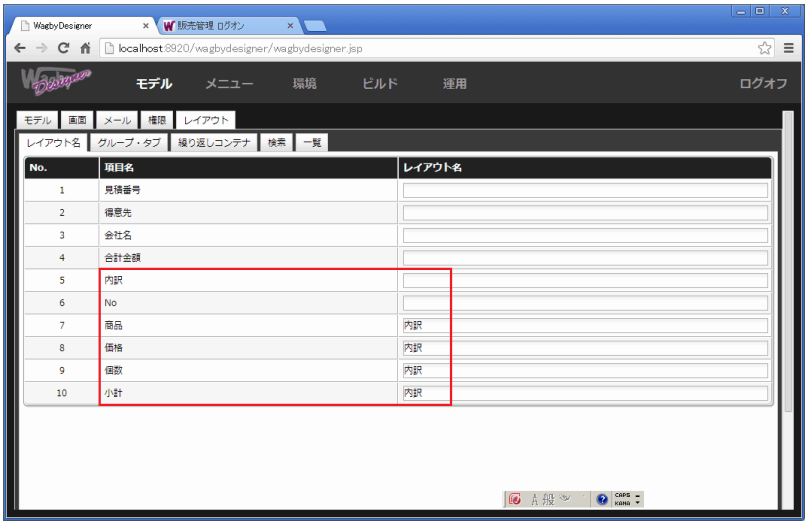


図 モデル項目

繰り返しコンテナのレイアウトを縦並びにする場合、コンテナ内の各項目名の先頭に「コンテナ名」を付与します。 すなわちコンテナ名を、レイアウト名のように扱います。

「繰り返しコンテナ」「繰り返しコンテナID」の項目には指定しません。



横並びレイアウト（標準方式）との違いを下表に示します。

	横並びの繰り返しコンテナ（標準）	縦並びの繰り返しコンテナ
コンテナ名	内訳	内訳
コンテナID	No	No
コンテナ内の項目	商品	内訳>商品

レイアウトタブ内の「グループ・タブ」レイアウトで、レイアウト名「内訳」の並び方向を「縦」とします。

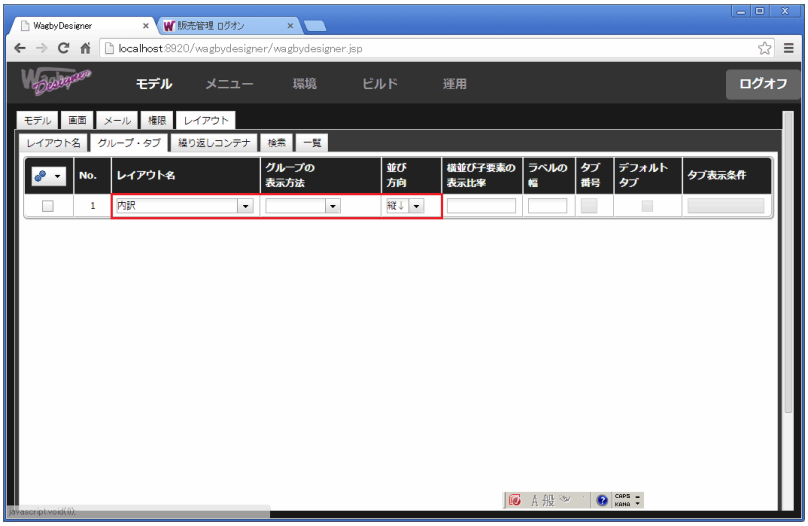


図 並び方向の設定

「繰り返しコンテナID」には「コンテナ名>」を含めないようにしてください。 繰り返しコンテナ以外の項目に「コンテナ名」をレイアウト名として定義することはできません。

縦並びと横並びの組み合わせ

例

「価格」「個数」「小計」の項目を横並びとした例を図に示します。

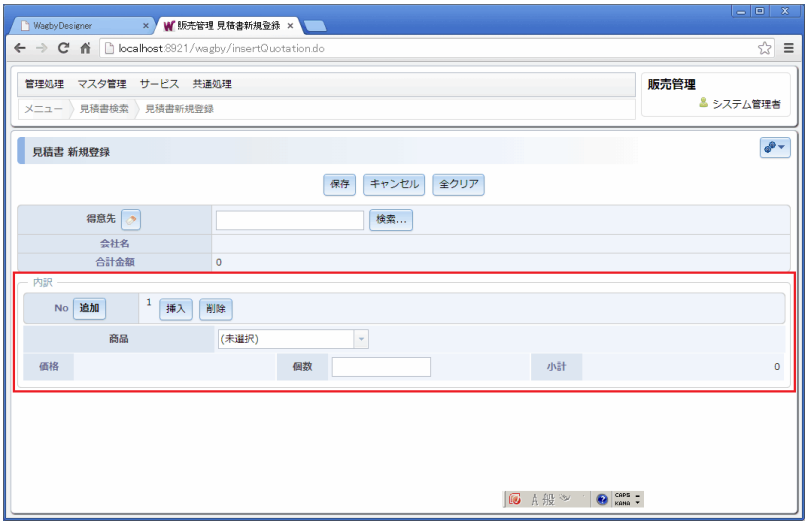


図 縦並びレイアウトと横並びレイアウトの組み合わせ

定義方法

「価格」「個数」「小計」のレイアウト名を「コンテナ名>*隠しレイアウト名」としています。

アスタリスク「*」のあとに続くレイアウト名は英数字の組み合わせとしてください。レイアウト名は任意です。

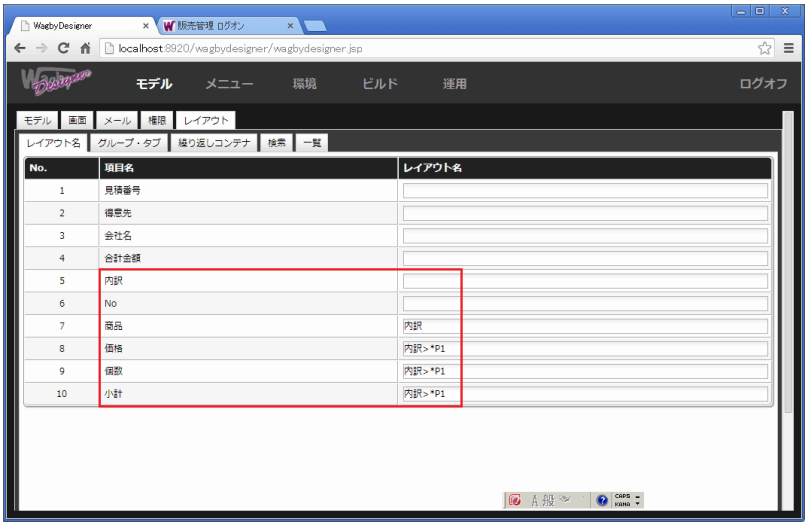


図 隠しレイアウトの設定

隠しレイアウトに対する「横並び子要素の表示比率」を指定します。

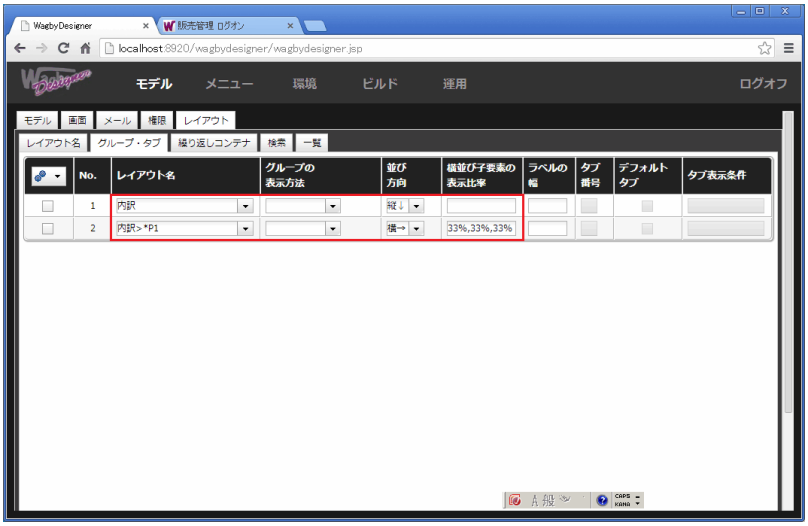


図 横並び子要素の表示比率の設定

仕様・制約

　このレイアウトは「繰り返しコンテナID」の項目に「追加」「挿入」「削除」ボタンが用意されます。そのため、一件もデータが含まれない場合、更新画面でボタンが表示されません。

繰り返しコンテナ － 複数の繰り返しコンテナの横並び

横並びレイアウト

図は、繰り返しコンテナを横並びで二段表示とした例です。

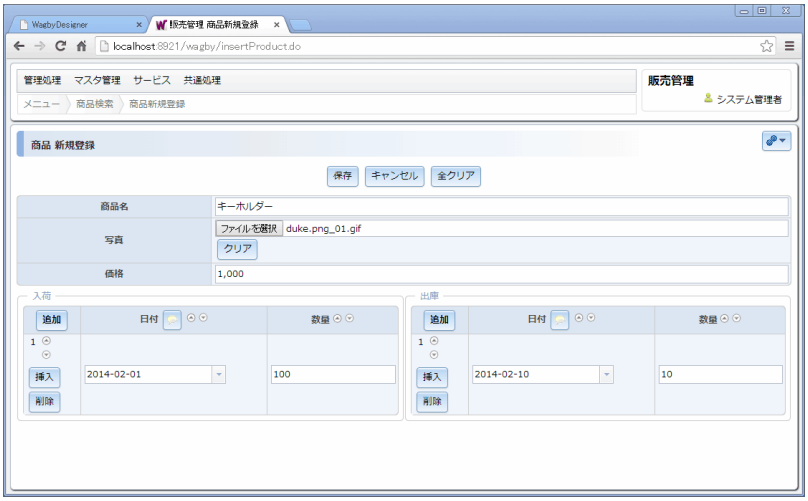


図 二つの繰り返しコンテナの横並びレイアウト

詳細画面の例を図に示します。コンテナに含まれる数は異なっていてもかまいません。

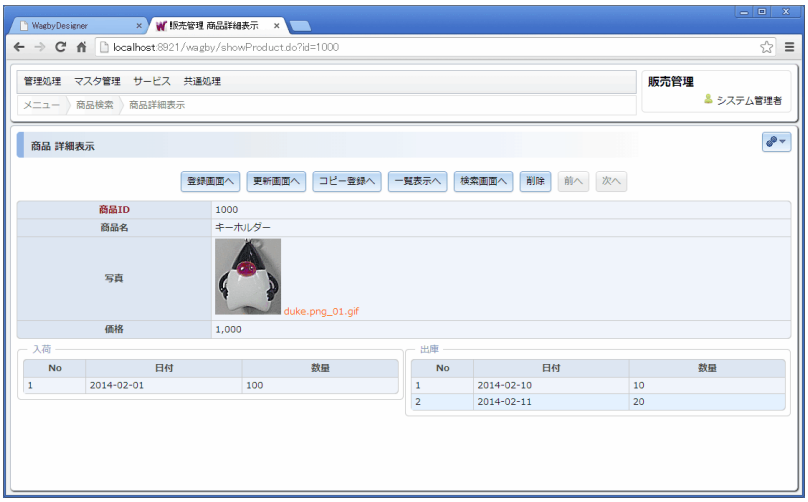


図 詳細画面

定義方法

二つの繰り返しコンテナ「入荷」と「出庫」の定義を図に示します。

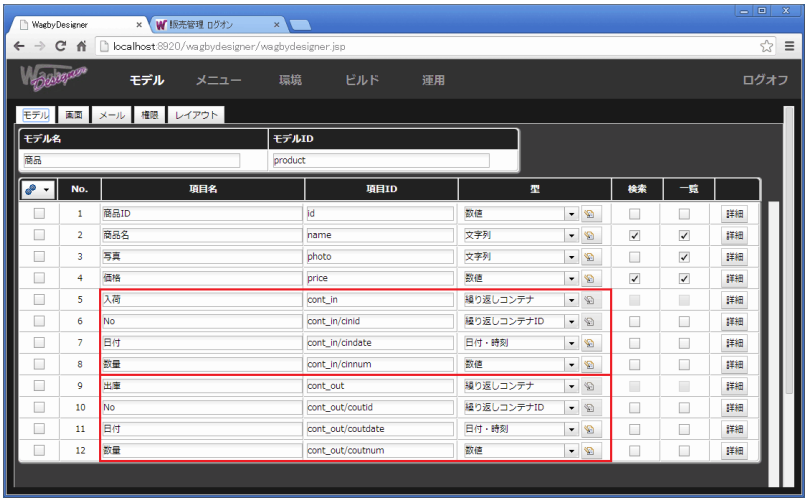


図 モデル項目の定義

「入荷」と「出庫」に隠しレイアウト指定（ここでは *P1）を行います。

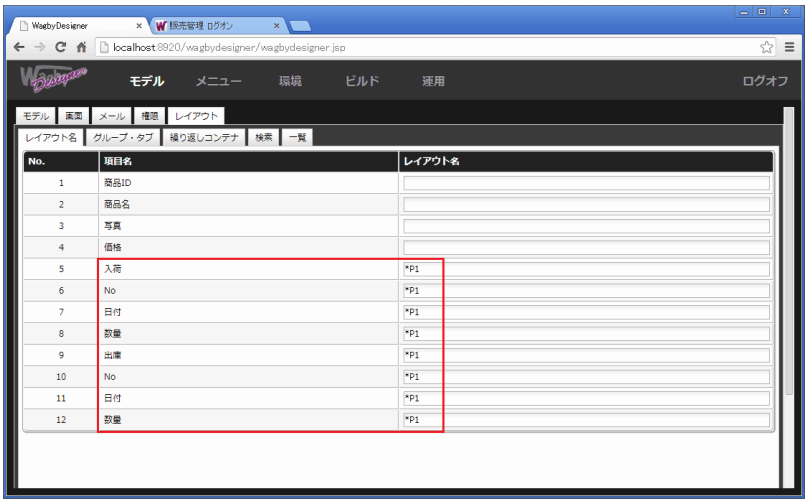


図 隠しレイアウトの指定

図で用意したレイアウトを横並びとし、比率を指定します。

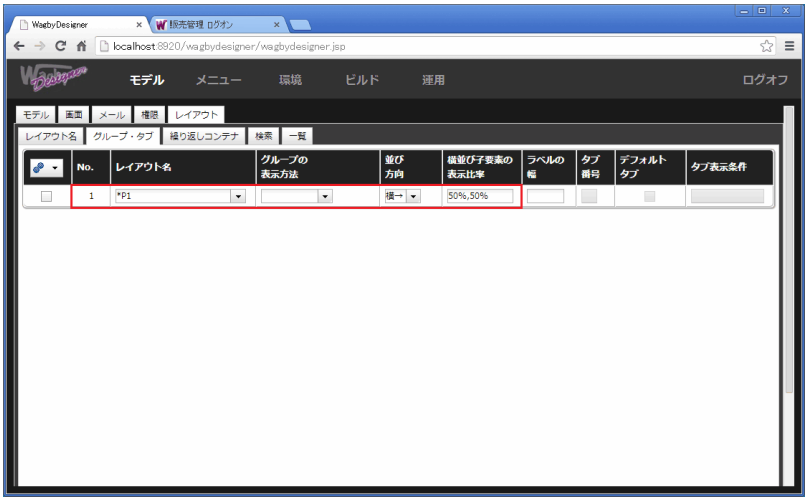


図 横並び比率の設定

繰り返しコンテナ － 複雑なタブ設定

タブの中に通常項目と繰り返しコンテナを含める

例

タブの中に通常項目と繰り返しコンテナを含めた例を図に示します。 タブに含まれない項目は、タブの上に配置されます。

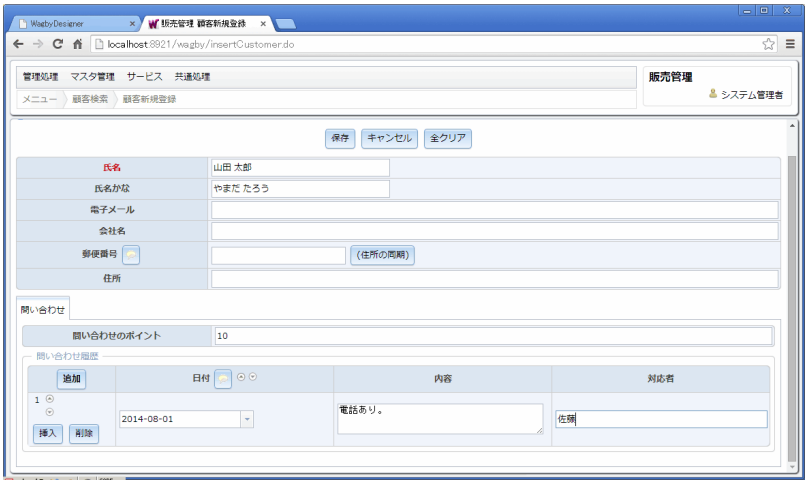


図 タブの中に通常項目と繰り返しコンテナを含める

定義方法

レイアウトの配置を指定するのは、図の赤枠で示した項目です。

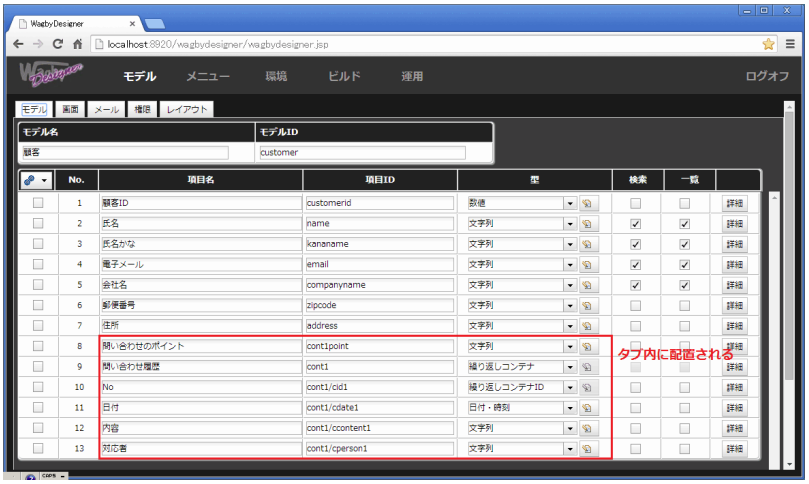


図 顧客モデルの定義

繰り返しコンテナを格納できる特別なタブを次のように定義します。

[*タブ識別子:名称]

識別子は1文字の英字を使います。

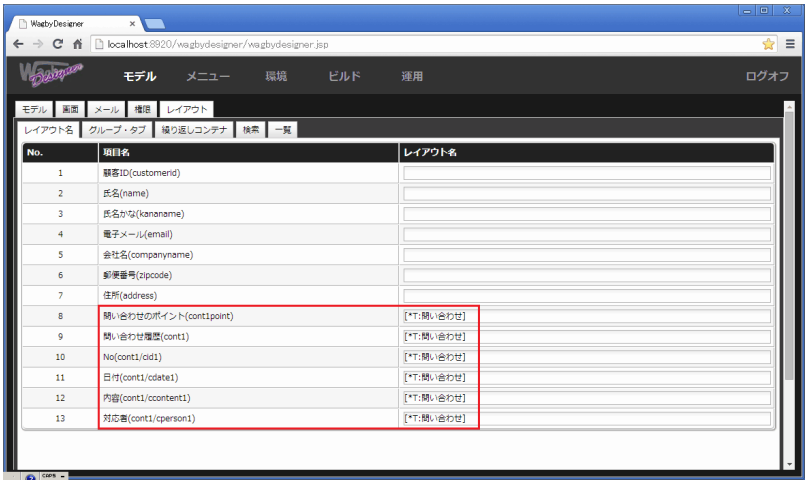


図 レイアウト名の指定

この識別子は「タブ」であることを指定します。

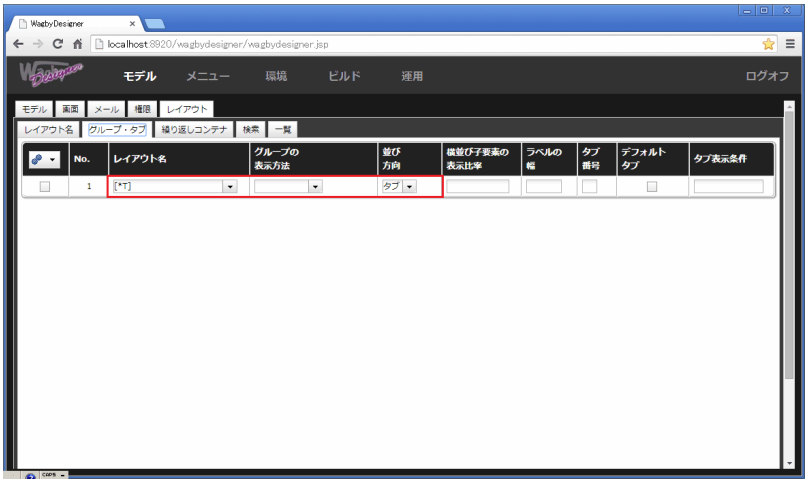


図 このレイアウトがタブであることを指定する

レイアウト名は選択肢から選ぶのではなく、手動で '[*T]' と入力してください。

タブの中に通常項目と繰り返しコンテナを含め、通常項目を横並びとする

例

タブの中の通常項目を横並びとした例を図に示します。

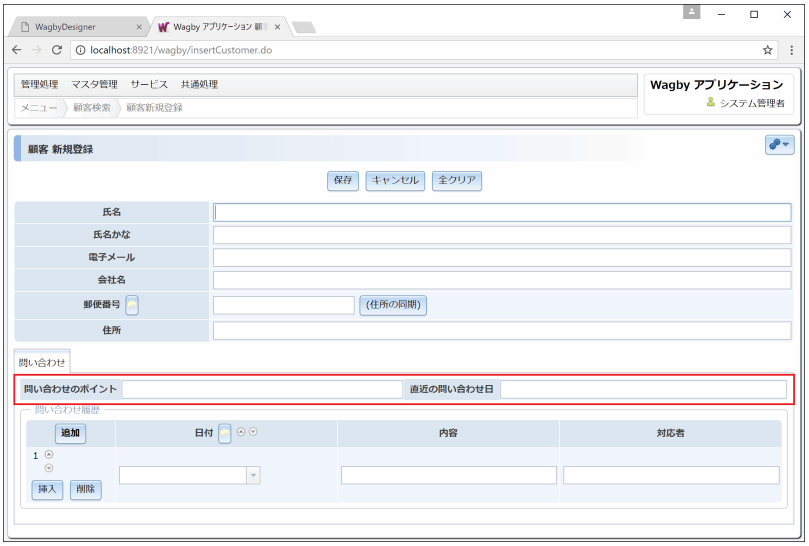


図 タブの中の通常項目を横並びとする

定義方法

レイアウトの配置を指定するのは、図の赤枠で示した項目です。

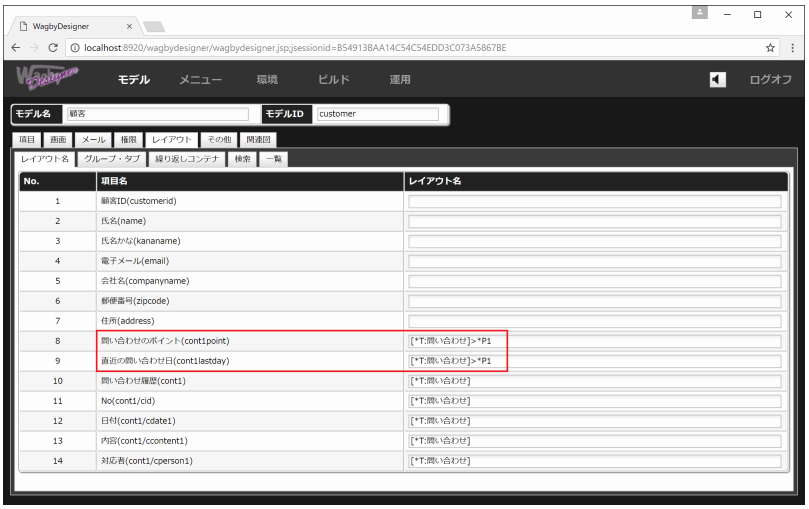


図 レイアウトの定義

ここで、タブ内にレイアウトを含めます。

[*タブ識別子:名称]>レイアウト

ここでは、隠しレイアウト ’*P1’ を指定しました。このレイアウトに、横並びを指定します。

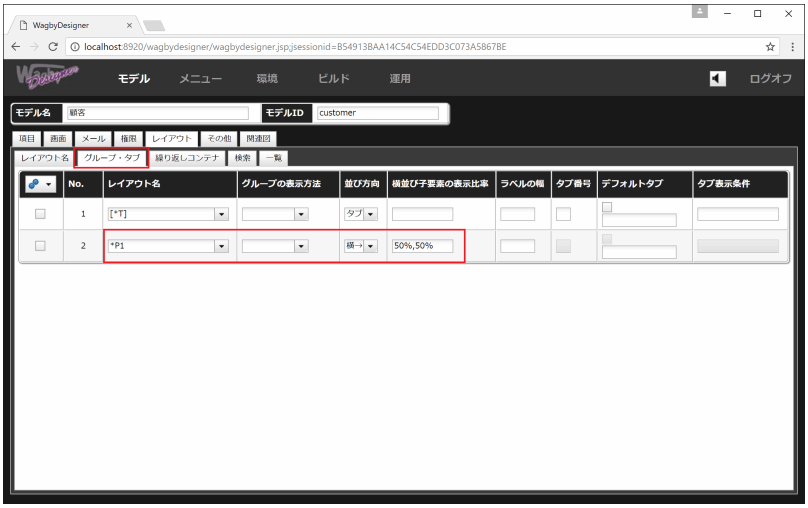


図 このレイアウトが横並びであることを指定する

レイアウト名は選択肢から選ぶのではなく、手動で ’*P1’ などと入力してください。タブ名称は含めません。

一つのタブに繰り返しコンテナを丸ごと格納する

例

タブの中に繰り返しコンテナを丸ごと格納することもできます。繰り返しコンテナの枠が表示されていません。

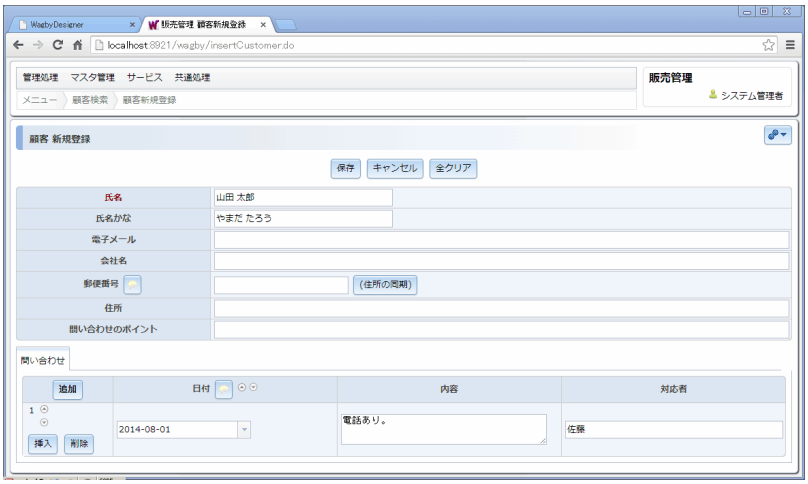


図 繰り返しコンテナ全体がタブの中に配置される

定義方法

繰り返しコンテナ名称をタブ名称と同じにすると、繰り返しコンテナの枠を除去することができます。

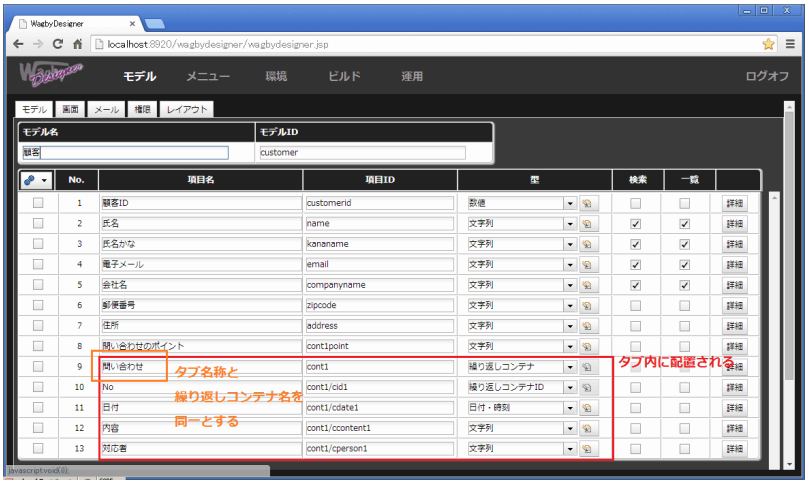


図 繰り返しコンテナ名称をタブ名称と同じにする

繰り返しコンテナ横並びをタブに格納する

例

タブの中に繰り返しコンテナを横並びで格納することもできます。

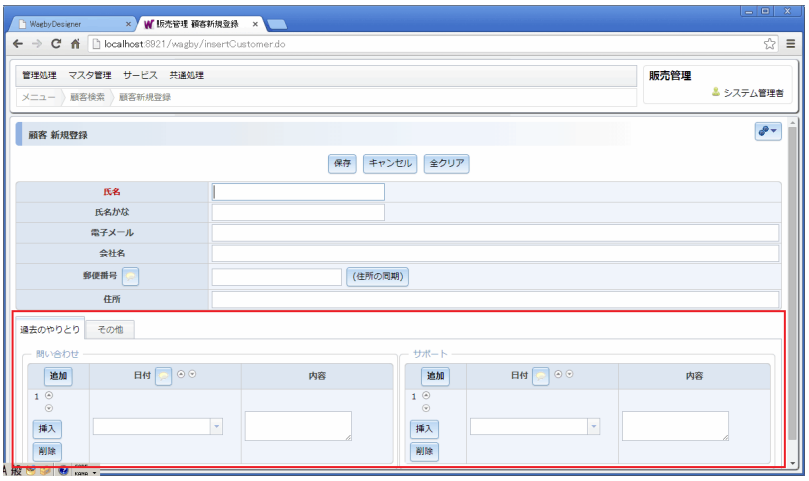


図 繰り返しコンテナの横並びレイアウトをタブに格納する

定義方法

繰り返しコンテナ横並びレイアウト項目と、それ以外の項目を二つのタブに格納します。

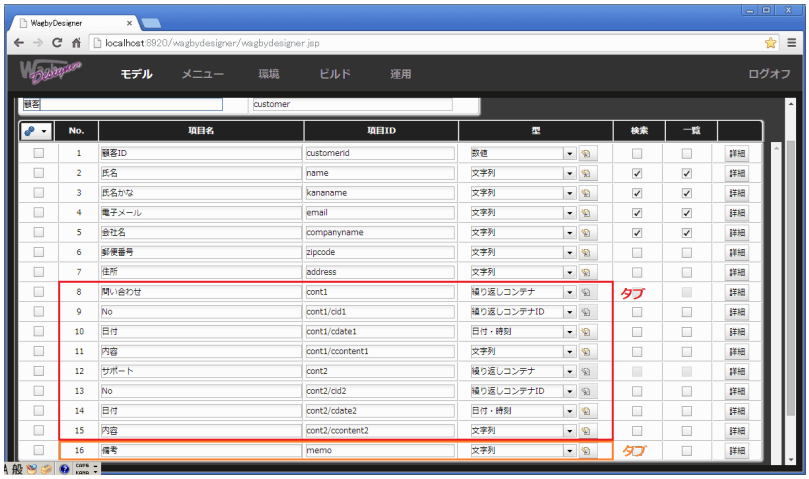


図 繰り返しコンテナ横並びレイアウト項目と、それ以外の項目

タブ指定のあとに繰り返しコンテナの横並び定義を行います。

[*タブ識別子:名称]>*繰り返しコンテナのための隠しパネル識別子

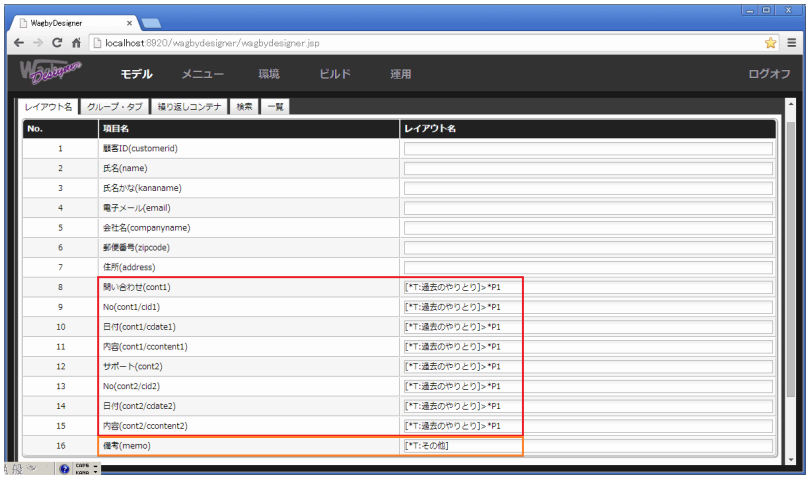


図 レイアウト名を指定する

二つのレイアウト［*T］と *P1 をそれぞれ指定します。

図 レイアウトの詳細定義

制約 ： 本レイアウト利用時は、2つ以上のタブを用意するようにしてください。さらに、本レイアウト設定の後にタブが必要です。

繰り返しコンテナ複数段一行明細レイアウトをタブに格納する

例

タブの中に複数段一行明細レイアウトの繰り返しコンテナを格納することもできます。

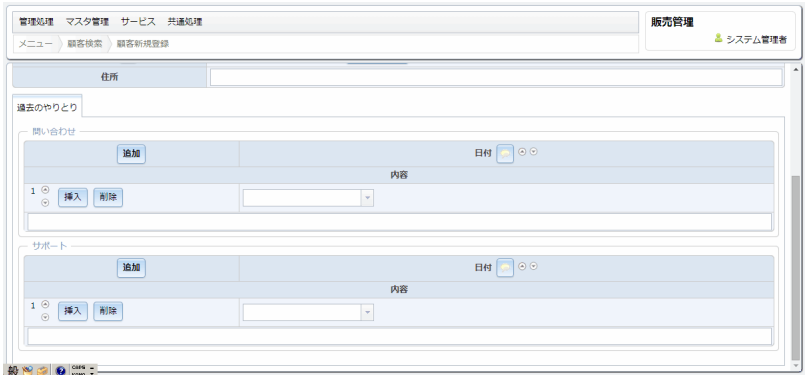


図 繰り返しコンテナ複数段一行明細レイアウトをタブに格納する

定義方法

図の例では、二つの繰り返しコンテナcont1,cont2を用意しています。

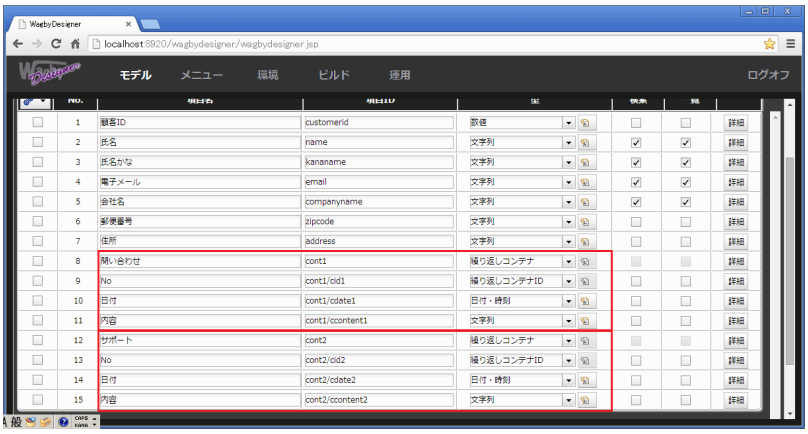


図 繰り返しコンテナの定義

タブ指定のあとに繰り返しコンテナの複数段一行明細レイアウトの定義を行います。

[*タブ識別子:名称]>*繰り返しコンテナのための隠しパネル識別子

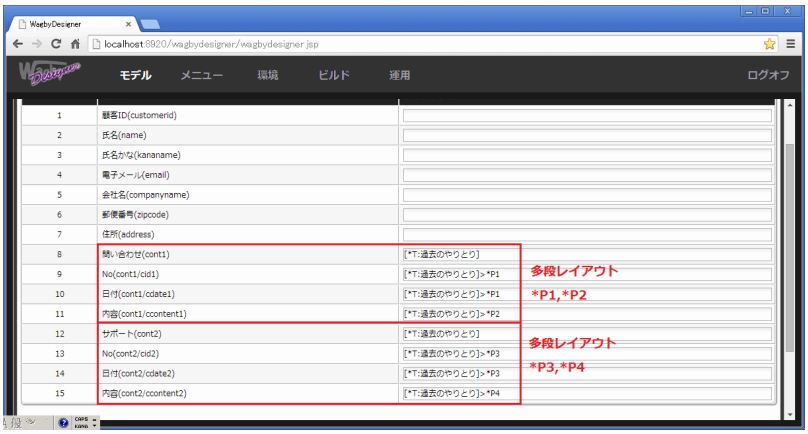


図 レイアウト名を指定する

レイアウト *T を指定します。

図 レイアウトの詳細定義

繰り返しコンテナのレイアウトで、複数段一行明細レイアウト利用時のセル結合の指定を（必要に応じて）行います。

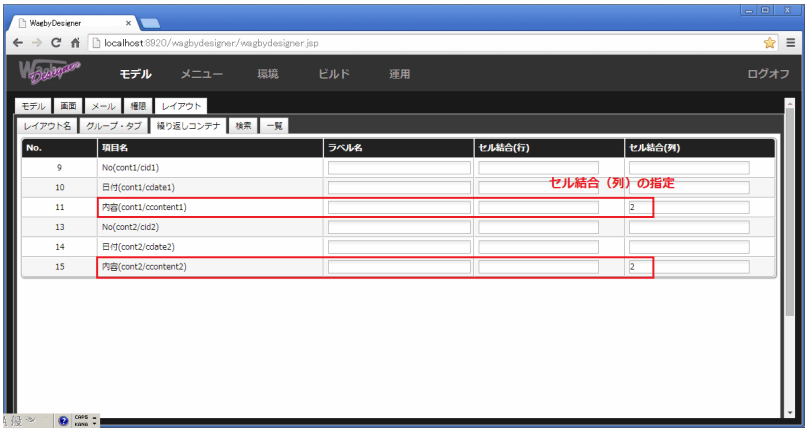


図 繰り返しコンテナ複数段一行明細レイアウトのセル結合指定

外部キー子モデルおよび繰り返しコンテナを親モデルのタブにまとめる

外部キー子モデル

親モデルを「顧客」とし、子モデル「サポート」を用意した例で説明します。図は、標準の画面レイアウトです。親モデル（顧客）は、タブを使って複数の項目をまとめています。子モデル（サポート）は親モデルの下部に表示されます。

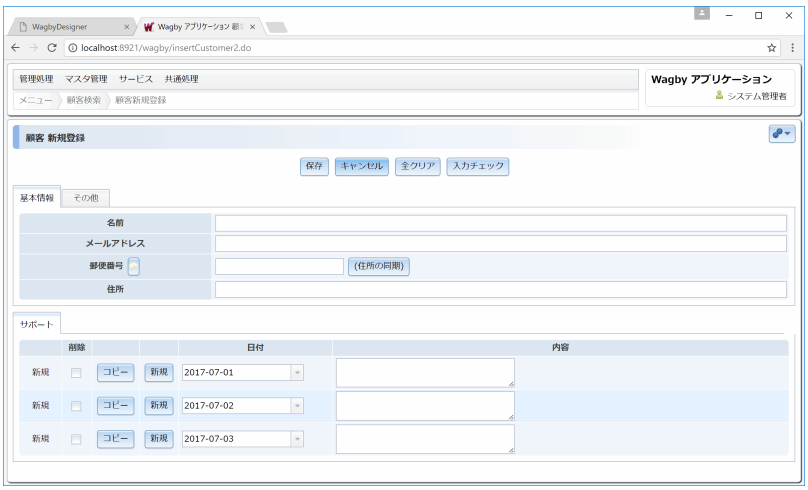


図 親子モデル同時更新画面の、標準のレイアウト

ここで紹介する「タブをまとめる」設定を行うと、下図のように親モデル側のタブにまとめて表示することができます。

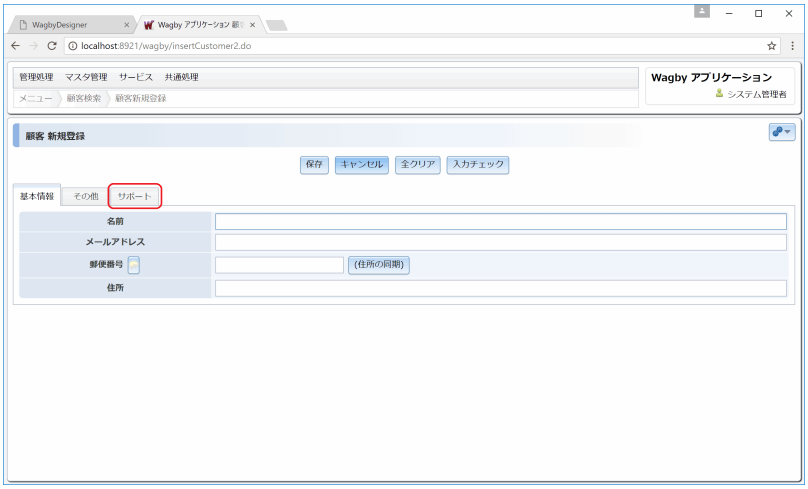


図 子モデルのタブを、親モデルにまとめる(1)

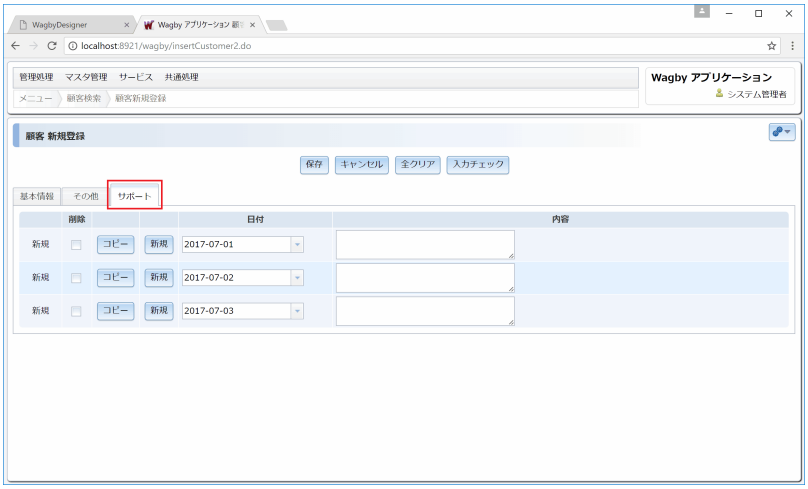


図 子モデルのタブを、親モデルにまとめる(2)

ここで子モデルは一覧更新画面を有効にしています。詳細は〝親子関係 > 子モデルの同時更新〟をお読みください。

定義方法 - 親

親モデル（顧客）の項目群にレイアウト名を定義します。

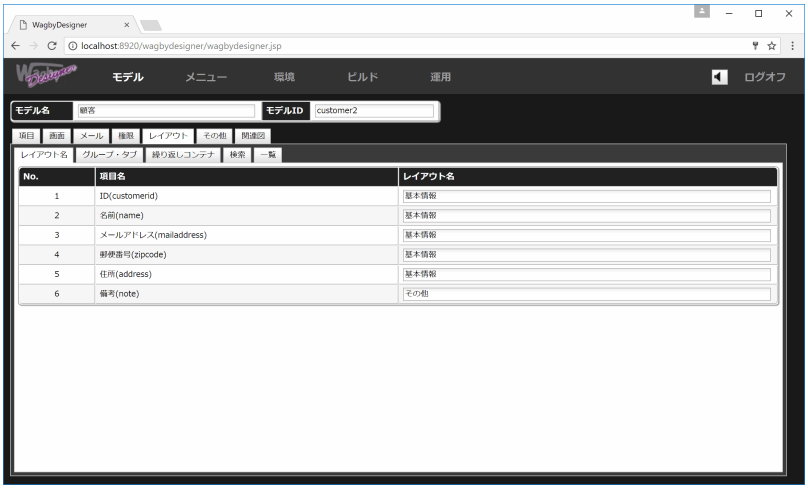


図 レイアウト名を設定する

このレイアウトを「タブ」とします。さらに「タブ番号」を次のようにします。

タブ番号:表示位置

- タブ番号は「1」から開始します。
- 後述する外部キー子モデルや繰り返しコンテナをタブにまとめる場合、対象となるタブ番号は「1」のみとなります。
- タブ番号および表示位置は、1から開始し、連続する値としてください。飛び番号などは許容されません。

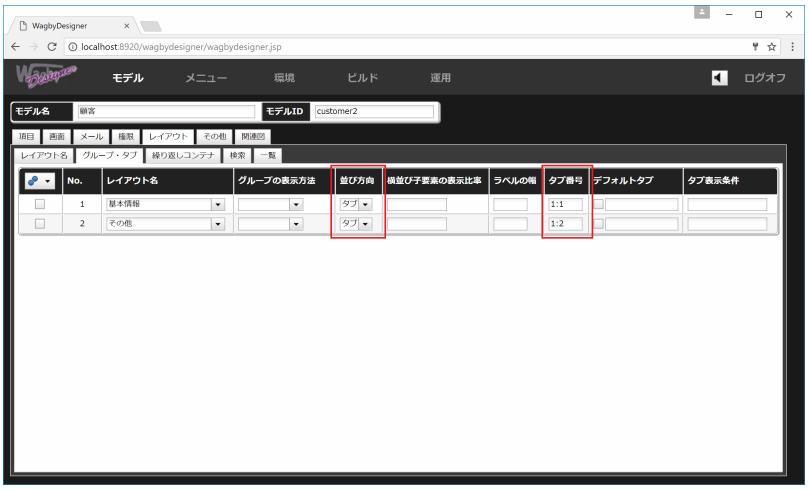


図 タブ番号の設定

定義方法 - 子

子モデル（サポート）では「画面 > 検索・一覧表示」タブ内の「外部キー連携」の設定欄で「タブレイアウト」を選択し、「タブの表示位置」に次の値を指定します。

親モデルID:タブ番号("1"で固定):位置

この例では customer:1:3 と記述しています。これにより親モデルID customer の、3番目のタブ位置に配置されます。

(183) (184)

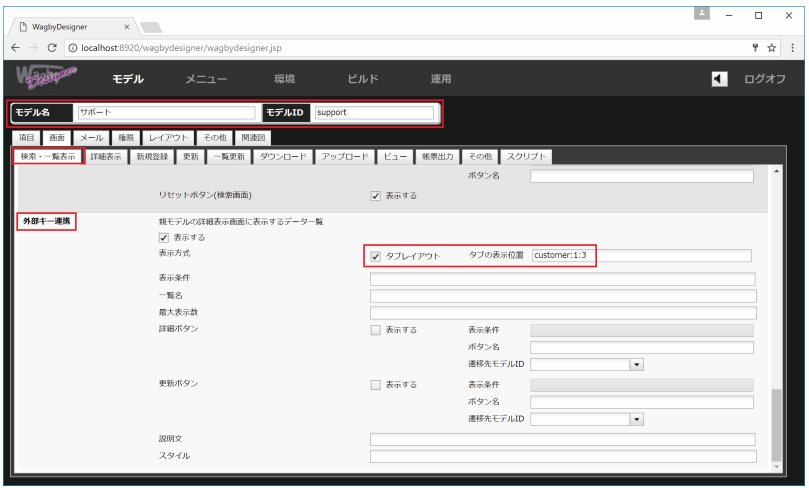


図 子モデルにおけるタブ表示位置の指定

複数の親と紐づく場合

(183) 親のタブ番号は「1」を指定してください。複数のタブグループがあった場合でも、最初のタブグループのみを対象とします。

(184) この例では、親のタブは2つしかないため、3番目 = 末尾のタブ、となります。

一つの子に複数の親を紐づける設定を行なった場合、次のようにコンマで区切って指定することができます。

```
親モデルID:1:位置,親モデルID:1:位置,...
```

タブの表示位置を変える

上の例で、customer:1 とすると、タブの先頭に子モデルが配置されます。

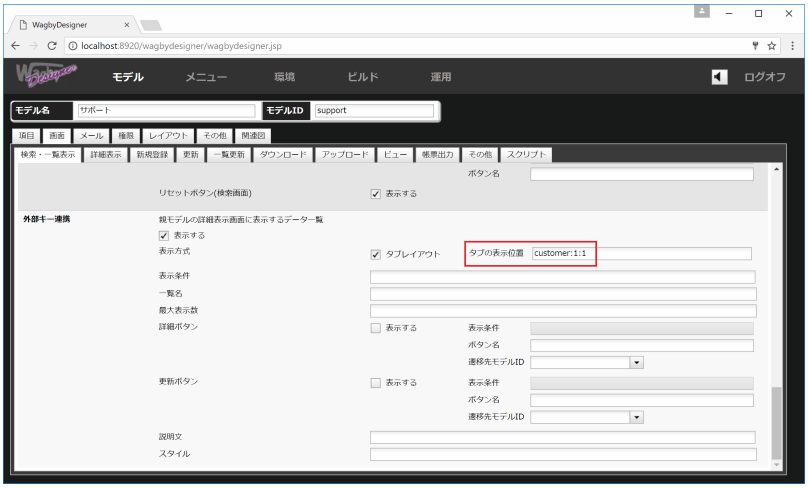


図 タブ位置を先頭にする

画面イメージは次のとおりです。

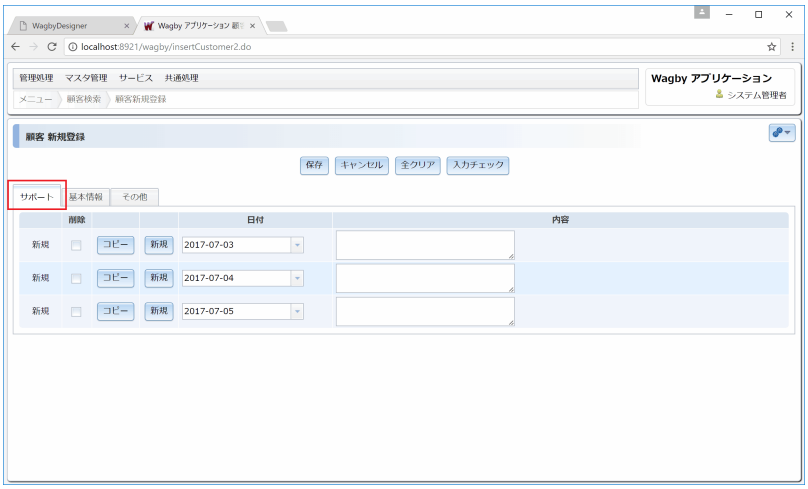


図 タブ位置を先頭にした例

繰り返しコンテナ

「顧客」モデルに、繰り返しコンテナ「趣味」を用意してみます。このコンテナのタブ表示をまとめた例を示します。

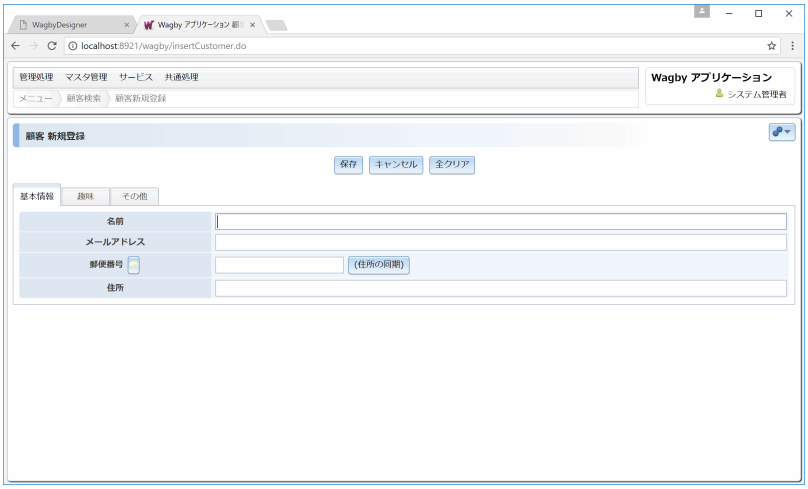


図 繰り返しコンテナ「趣味」タブをまとめた例(1)

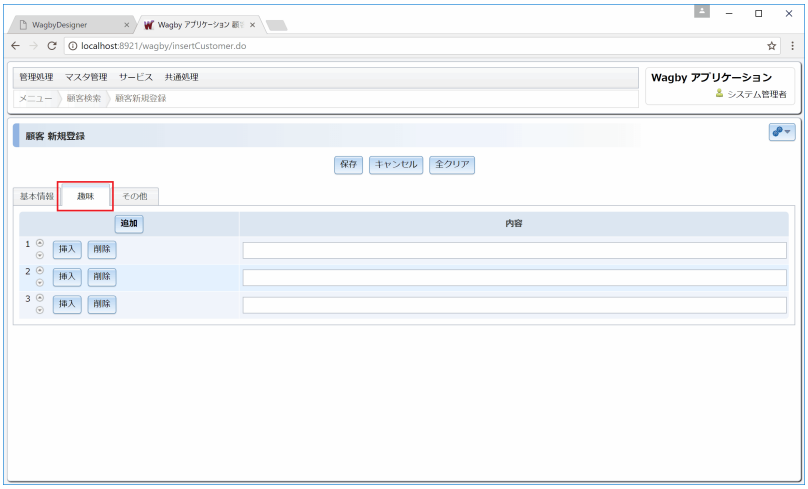


図 繰り返しコンテナ「趣味」タブをまとめた例(2)

定義方法

繰り返しコンテナのレイアウト名を、繰り返しコンテナ名と同じにします。

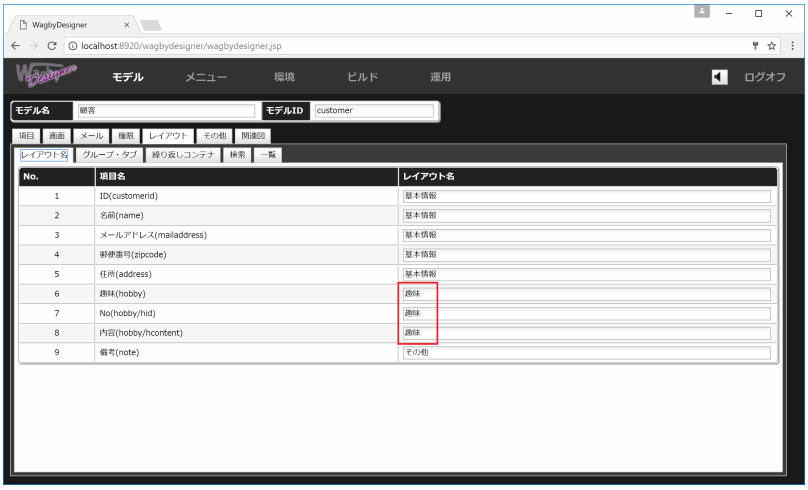


図 レイアウト名を、繰り返しコンテナ名とする

各レイアウトを「タブ」とし、タブ番号を1とします。コロンのあとに、表示位置を指定します。

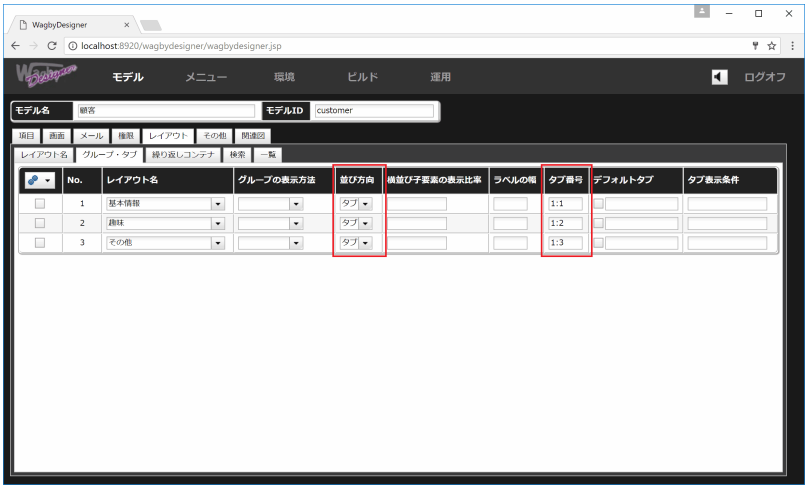


図 タブ番号の設定

外部キー子モデルと繰り返しコンテナ

これまでの例を一つの定義にまとめてみます。「顧客」モデルの繰り返しコンテナ「趣味」と、子モデル「サポート」をすべてタブにまとめることができます。

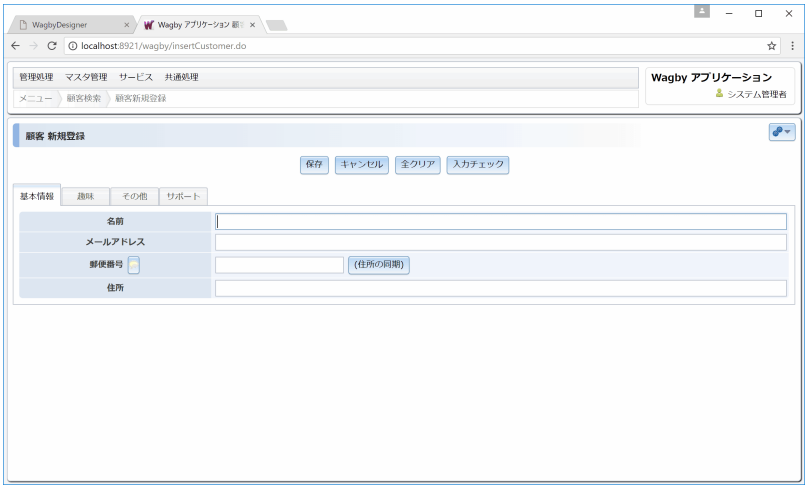


図 外部キー子モデルと繰り返しコンテナをまとめて一つのタブにする(1)

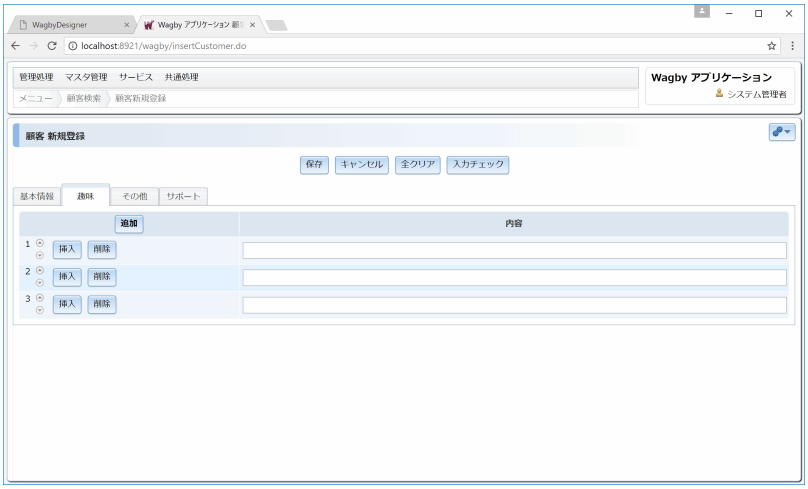


図 外部キー子モデルと繰り返しコンテナをまとめて一つのタブにする(2)

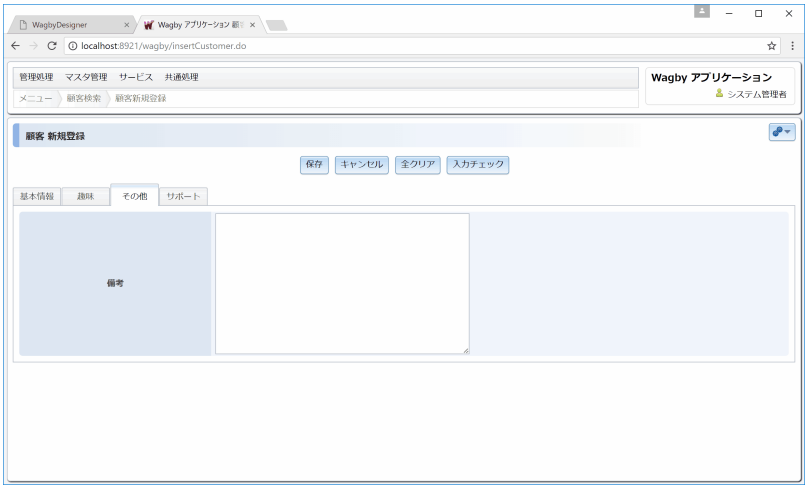


図 外部キー子モデルと繰り返しコンテナをまとめて一つのタブにする(3)

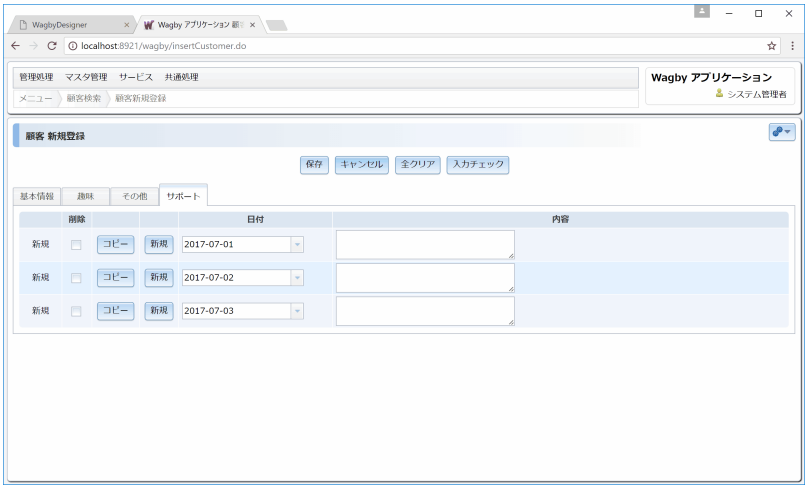


図 外部キー子モデルと繰り返しコンテナをまとめて一つのタブにする(4)

定義方法

親モデル（顧客）のレイアウトは、図と同じです。

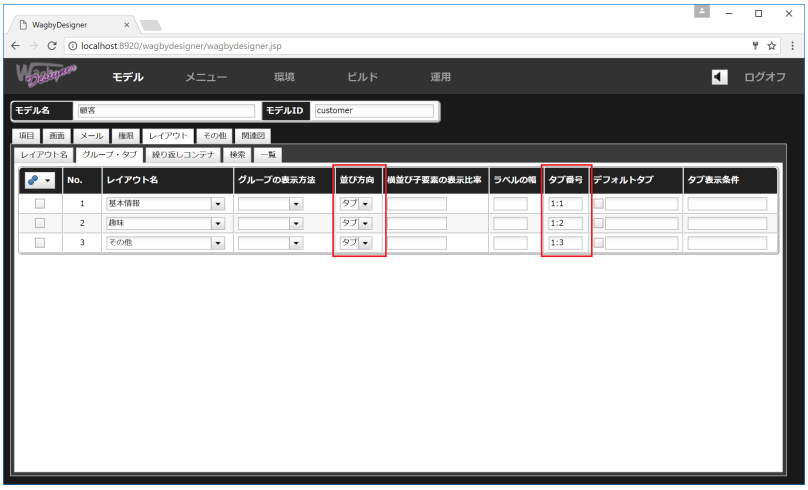


図 親モデルの設定-タブ番号

子モデル（サポート）は、図と同じです。（ここでは表示位置を4としています。）

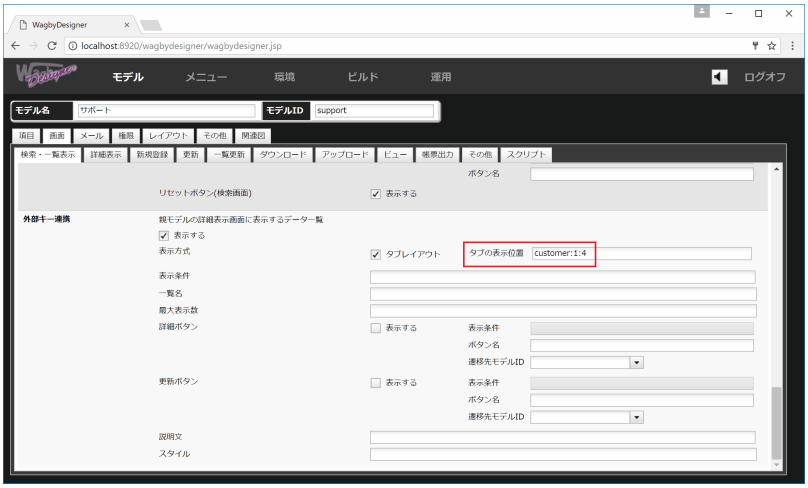


図 子モデルの設定-タブ表示位置

仕様・制約

- タブの中に通常項目と繰り返しコンテナを含める設定と併用することはできません。つまり、レイアウト名に〝[*T:コンテナ名]〟という記述を行うことはできません。
- 四角で囲むレイアウトとタブレイアウトを混在させることはできません。例えば図の設定で 〝基本情報〟のレイアウトを「四角で囲む」とし、それ以外を「タブ」とすることはできません。
- 外部キー子モデルの一覧更新機能を用いて、親モデルの登録・更新画面で同時に外部キー子モデルも更新させる場合、親モデルで確認画面を有効にすることはできません。

繰り返しコンテナ、外部キー子モデル一覧を任意の位置に配置する

繰り返しコンテナの配置

実行例

図は繰り返しコンテナを「会社」レイアウトの中に含めた例です。

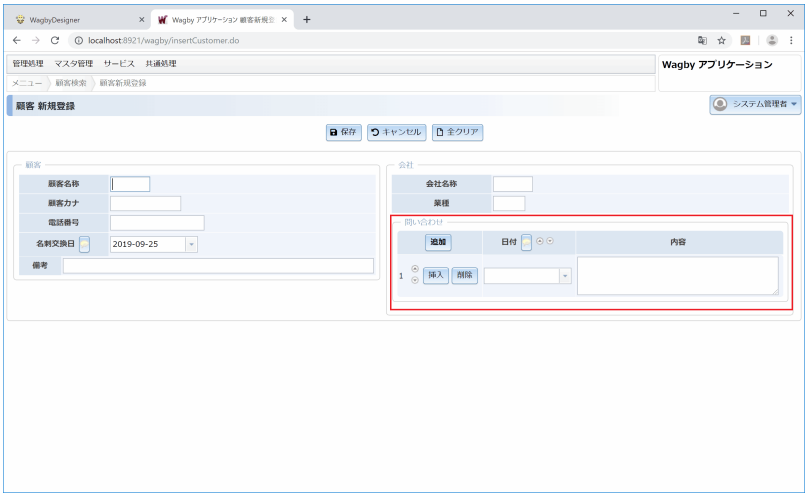


図 説明文の挿入（入力画面）

設定方法

繰り返しコンテナを配置する「位置」を示す項目を用意します。名前は任意です。型は文字列型とします。この項目はダミー項目であり、実際にはこの項目の位置に、繰り返しコンテナが表示されることになります。

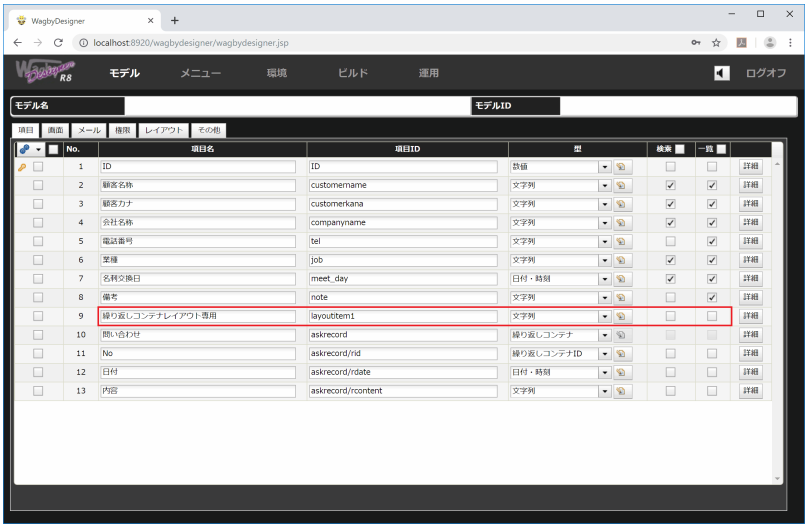


図 レイアウトのために用意するダミー項目

″出力制御>その他>レイアウト″ の ″レイアウト用項目とする″ を有効にします。″レイアウト対象ID″ に、繰り返しコンテナの項目名を指定します。

レイアウト用項目と指定された場合、自動的に次のような動作になります。

- データベースに保存されない。（対応するテーブルに、項目は作成されない）
- 自身は隠し項目となる。
- ラベル部分は非表示となる。

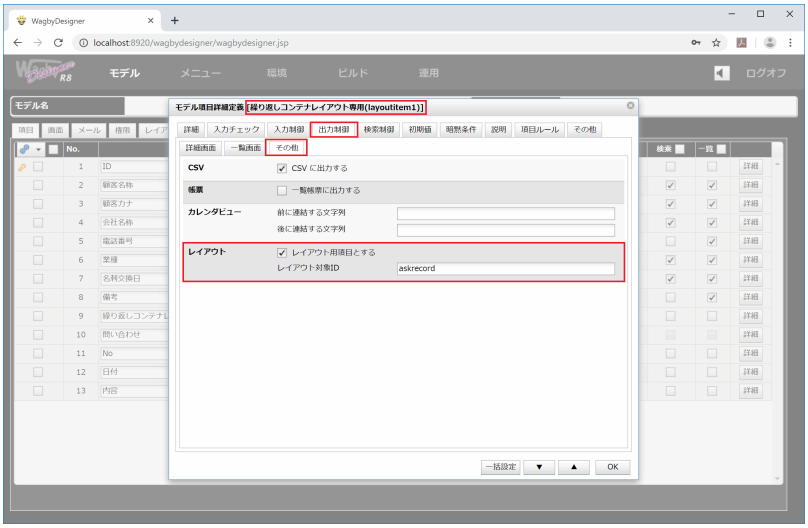


図 レイアウト用項目とする

画面レイアウトを設計します。ここでは「隠しレイアウトへの入れ子レイアウト」をベースとしています。設定の詳細はこちらのページの説明をお読みください。

レイアウト設計で ″繰り返しコンテナレイアウト専用″ の項目の位置に、繰り返しコンテナ（ここでは ″問い合わせ″）が配置されます。

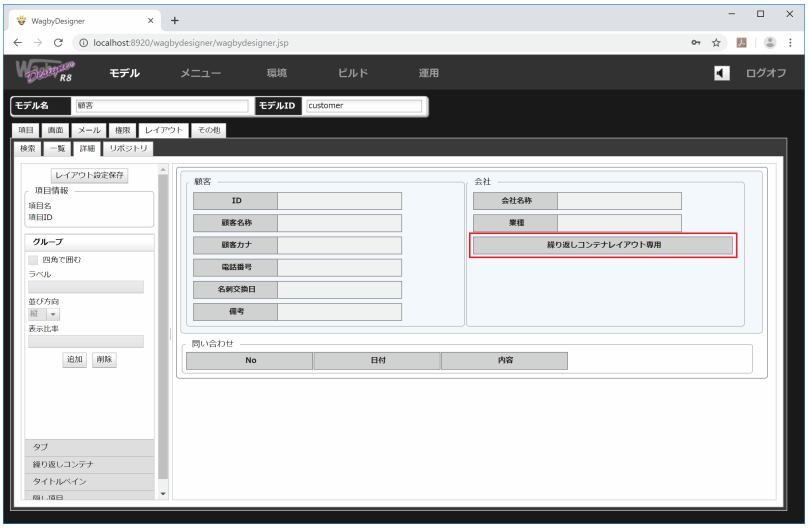


図 レイアウトを設計する

外部キー子モデル一覧の配置

実行例

外部キー子モデル一覧表示を任意の位置に配置することもできます。ここでは親モデルを ″顧客″ とし、子モデルを ″サポート″ とした例で説明します。

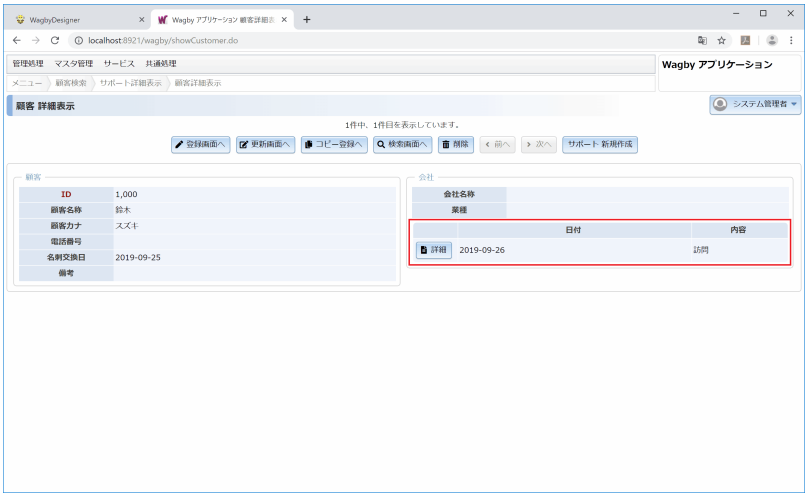


図 サポートモデル一覧表示の位置を設定した例

設定方法

顧客モデルとサポートモデルを用意します。

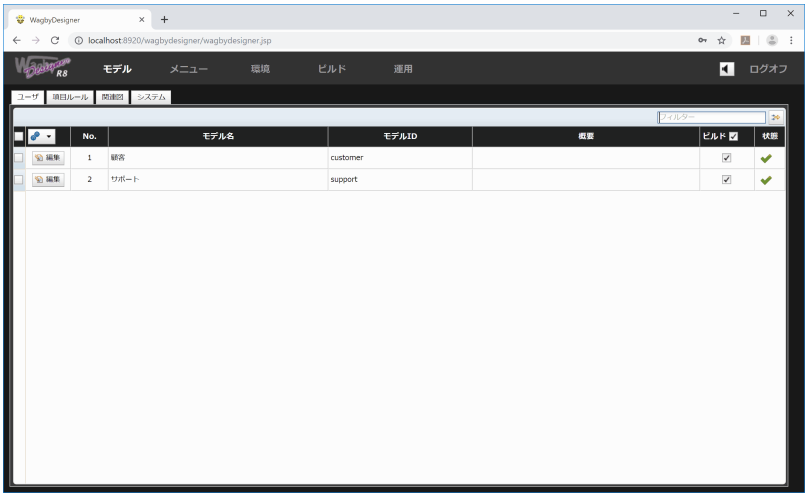


図 親子関係を定義する

レイアウト専用項目 ″外部キー子モデルサポート配置位置″ を用意します。型は文字列とします。

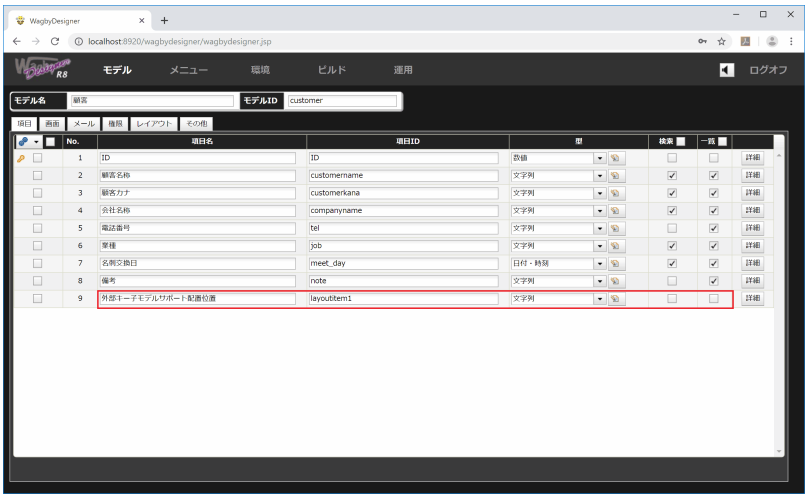


図 レイアウト専用項目を用意する

″出力制御>その他>レイアウト″ の ″レイアウト用項目とする″ を有効にします。″レイアウト対象ID″ に、子モデルのモデルIDを指定します。

レイアウト用項目と指定された場合、自動的に次のような動作になります。

- データベースに保存されない。（対応するテーブルに、項目は作成されない）
- 自身は隠し項目となる。
- ラベル部分は非表示となる。

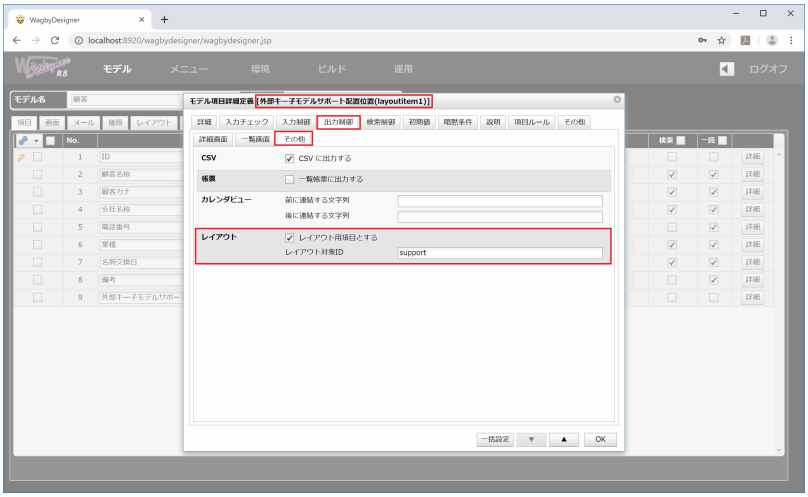


図 レイアウト用項目とする

画面レイアウトを設計します。ここでは「隠しレイアウトへの入れ子レイアウト」をベースとしています。設定の詳細はこちらのページの説明をお読みください。

レイアウト設計で ″外部キー子モデルサポート配置位置″ の項目の位置に、外部キー子モデル一覧表示（ここでは ″サポート″）が配置されます。

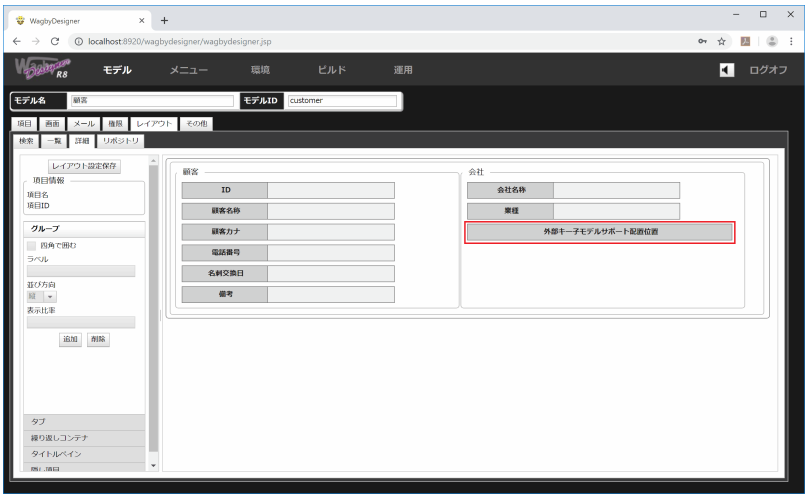


図 レイアウトを設計する

説明文・脚注を挿入する

例

図は「顧客」入力欄と「メモ」入力欄の間に説明文を挿入した例です。

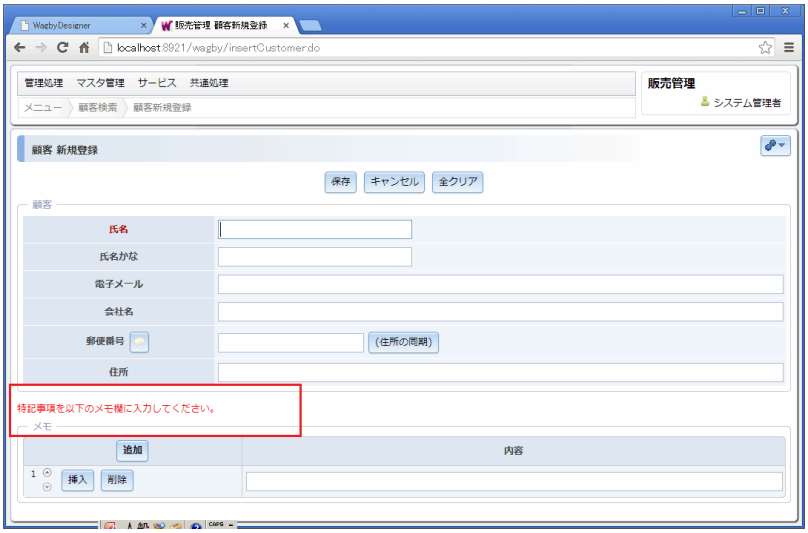


図 説明文の挿入（入力画面）

詳細画面にも反映されます。ここでは色を変えてみました。

図 説明文の挿入（詳細画面）

定義方法

図に示すように、表示する文字列を格納する項目を用意します。データベースに保存しないため、項目名は管理しやすいように定義します。

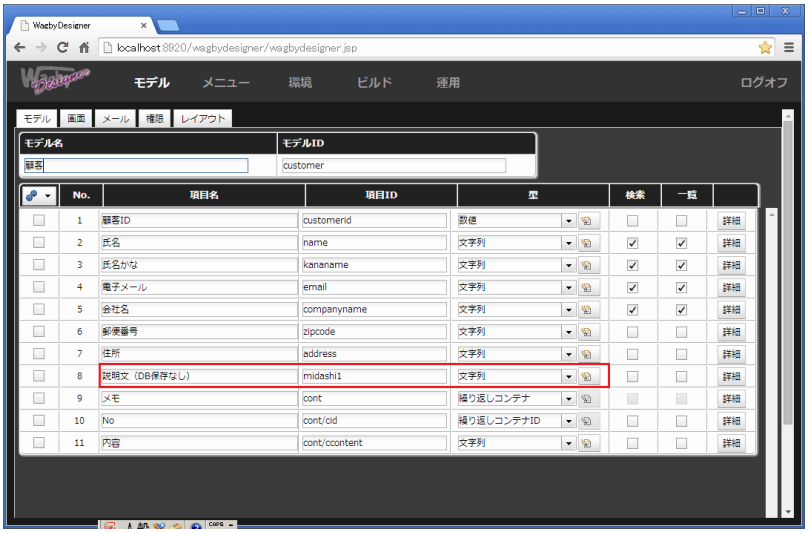


図 モデル定義

データベースに保存しない設定とします。

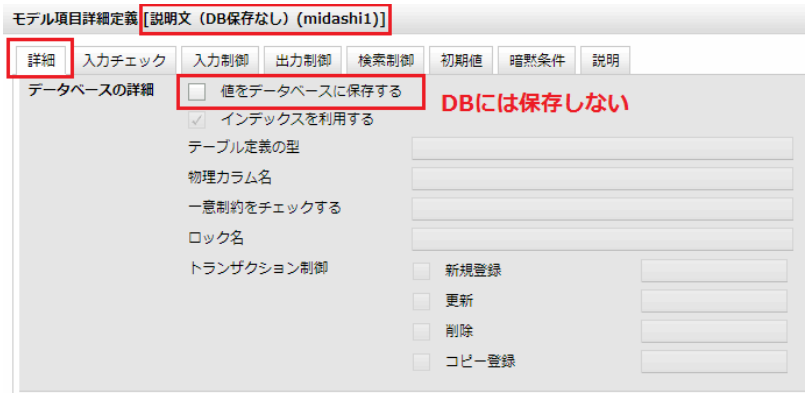


図 データベースに保存しない

読み込み専用とします。

図 読み込み専用設定とする

項目の値の初期値として、表示用の文字列を指定します。ここでは登録画面表示時と詳細画面表示時にそれぞれ表示用の文字列を設定しています。（これはデータベースに格納されません。画面表示のみで用いられます。）

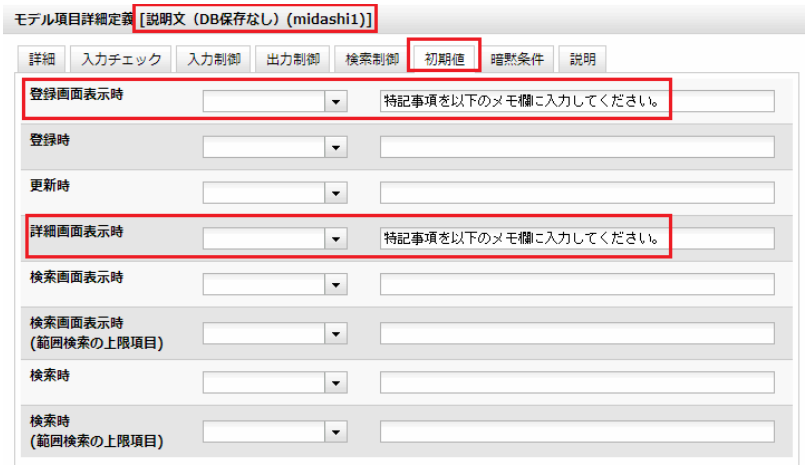


図 初期値として表示文字を設定する

入力画面（登録・更新）におけるスタイルを指定します。ここでは背景を白、文字色を赤としました。

```
background:#FFF;color:red;padding:0px;margin:0px;
```

スタイルの適用範囲は〝両方〟（入力欄および入力欄を囲う領域）とします。

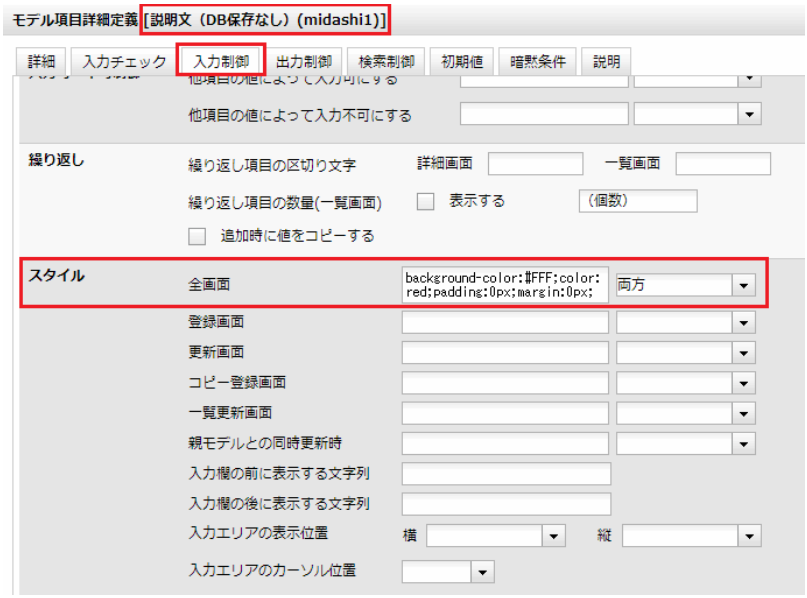


図 入力画面のスタイル設定

出力画面におけるスタイルを指定します。ここでは背景を白、文字色を青としました。 また「ラベルを表示しない」設定を有効にします。

```
background:#FFF;color:blue;padding:0px;margin:0px;
```

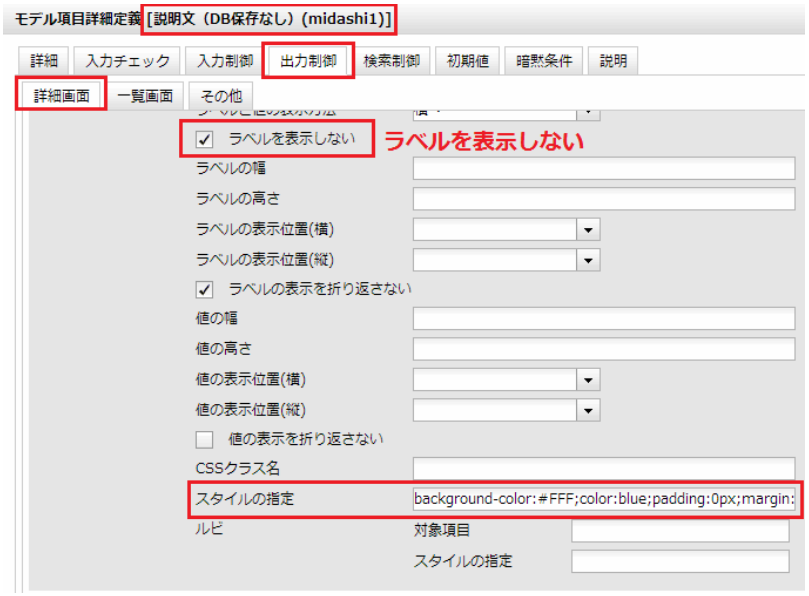


図 出力画面のスタイル指定

レイアウト

説明文を管理する項目に単独の隠しレイアウトを指定します。

図 隠しレイアウトの指定

グループの表示方法を〝縦方向にグループ化〟とします。これによって枠は表示されません。

図 レイアウトグループの表示方法の設定

説明文・脚注を枠で囲う

説明文を枠で囲った例を図に示します。

図 説明文を枠で囲って表示する

定義方法

図において、グループの表示方法を "四角で囲む" とします。

CSS レイアウトを利用する (CSS)

CSS レイアウトとは

CSS レイアウトとは、画面に表示される要素 (Wagbyの「モデル項目」) の配置を CSS だけで実現する技術です。さまざまな規格が提案されていますが、Wagby では CSS Flexbox という規格を採用しています。CSS Flexbox は Wagby がサポート対象とするすべての Web ブラウザで対応しています。

CSS Flexbox のメリットは次のとおりです。

- 自動生成される HTML が単純かつ小サイズになる。
- 画面レイアウトのカスタマイズが CSS だけで行えるようになる。
- すべてのブラウザで同じ書式である。

なお互換性のため、これまで Wagby が提供していた HTML テーブルによる画面レイアウトも選択できます。出荷時の標準は (従来の) HTML テーブル方式です。

(185)

定義方法

"画面 > その他 > 表示スタイル" で "CSS layout" を指定することができます。

(185) R8.1.0からR8.2.3まで、CSS Grid という別の規格を試していましたが、IE/Edge では書式が異なるという問題がありました。そこで R8.3.1 から CSS Flexbox を採用することにしました。

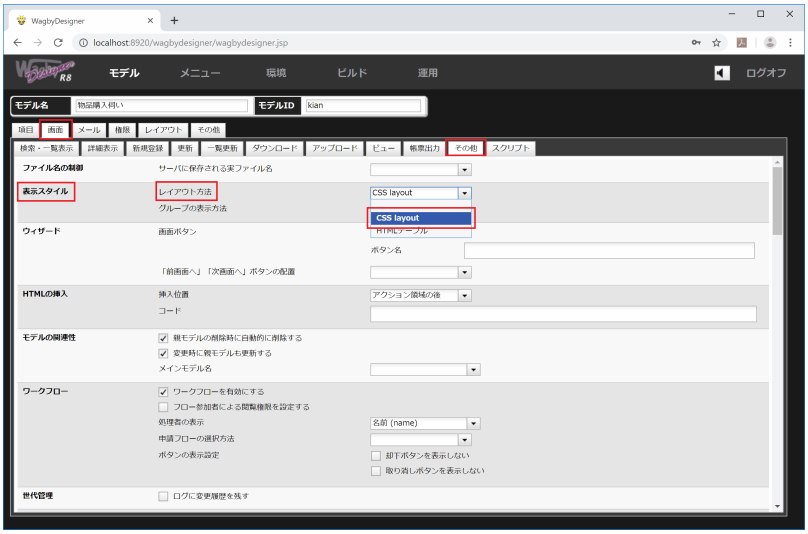


図 レイアウト方法の選択

カスタマイズ

生成される JSP では、各項目に id 属性が付与されます。この JSP に対応した CSS ファイルも生成されます。（例えば showJuser.css といったように、画面ごと、かつモデルごとに CSS ファイルが生成されます。）

レイアウトのカスタマイズを行う場合、この CSS ファイルに記述された設定を上書きした My ファイルを用意するとよいでしょう。（例えば MyShowJuser.css に、カスタマイズしたい内容だけを記述します。）

仕様・制約

- 従来のHTMLテーブルレイアウトと完全に同じ見た目ではありません。旧版からバージョンアップしてCSS layoutレイアウトを利用する設定に変えた場合、差異があることを念頭に画面レイアウトの確認を行なってください。
- CSS レイアウトでは Wagby のレイアウト機能が提供する「表示比率」の扱いが変わります。項目の数と、表示比率で指定したパラメータの数が合わない場合（例えば 3 つの項目に対応した表示比率の設定が "50%,50%" と二つしか指定していなかった、など）これまでの HTML テーブル方式と見た目が合わなくなることがあります。このような場合は表示比率の設定を見直してください。

ログオン画面の背景画像を変更する

背景画像を保存する

ログオン画面の背景画像は wagbyapp¥webapps¥$APPNAME¥img¥logon_bg.png として用意されています。

差し替えたいファイルを PNG 形式画像として用意します。これを次の場所に保存します。(*)

```
customize¥webapp¥img¥logon_bg.png
```

(186)

スタイルの設定（8.3.0）

標準で含まれている logon.css に次のような設定があります。

```
.logonBgImg {
```

(186) インストール直後は、customize¥webappフォルダまでが用意されています。開発者ご自身でimgフォルダの作成を行ってください。

```
/*  height: 100%;*/
/*
  height: auto;
  max-width: 100%;
*/
}
```

この .logonBgImg を mycommon.css で上書き設定することができます。

例えば次のようにすることで、ブラウザウィンドウの高さに合わせて背景画像のアスペクト比を保持した状態で伸縮されます。

```
.logonBgImg {
  height: 100%;
}
```

CSSのカスタマイズ（CSS）

変更するファイル

Wagbyに同梱されているCSSファイルを上書きするのではなく、「追加ファイル」を用意する方法を提供しています。これを使って標準の動作を変更することができます。

変更する範囲	開発者が作成するファイル	保存フォルダ（*1）
アプリケーション全体	mycommon.css	customize¥webapps¥css
モデル単位	My.css	customize¥webapps¥<モデルID>
画面単位	My<画面識別子>.css（*2）	customize¥webapps¥<モデルID>
システム管理	system¥My.css	customize¥webapps¥system

(187) (188)

CSSファイルの適用ルール

CSSファイルの読み込み順は次のようになります。同じ設定があった場合、後に読み込まれた設定に上書きされます。これによって適用範囲を限定したカスタマイズが行えるようになっています。

1. (製品に同梱されている)標準の common.css
2. mycommon.css
3. <モデルID>/My.css
4. My<画面識別子>.css

管理処理タブ内のモデルの扱い

管理処理など、以下の画面については system/My.css が共通で適用されます。作成した My.css を customize¥webapp¥system/My.css として保存してください。

- メニュー
- 管理処理 - インポートとエクスポート,統計情報,システムログ閲覧,このアプリケーションについて,オンラインライセンス認証,ログオンユーザ管理,メンテナンスモード切り替え
- 共通処理 - プレファレンス

テーマごとのカスタマイズ

カスタマイズしたファイル（例 mycommon.css）に記載した内容は、どのテーマ(claro,soria,tundra,nihilo)にも適用されます。テーマ毎に指定したい場合は、wagbydesigner¥bin¥webpage¥css 内のテーマ別ファイルを直接、修正してください。

---

(187) 1. インストール直後は、customize¥webappフォルダまでが用意されています。開発者ご自身で必要なフォルダの作成を行ってください。

(188) 2. 画面識別子は "画面種別" + "モデルID" の組み合わせです。SCREENID 関数の戻り値になります。

Wagby のバージョンアップ時は、開発者はこの修正を（バージョンアップしたWagbyにも）適用してください。

例 ページタイトルの背景色を変更する

model1のページタイトルを変更する

model1 のすべての画面に反映させる場合は customize¥webapp¥model1¥My.css を用意します。

```
.pagetitle {
  border-left: 10px solid #6BBF3F;
  filter:
 progid:DXImageTransform.Microsoft.gradient(GradientType=1,startColorstr="#95FF65",endColorstr="#FFFFFF");
  background: -webkit-linear-gradient(left, #95FF65, #FFFFFF);
  background: -moz-linear-gradient(right, #95FF65, #FFFFFF);
  background: -ms-linear-gradient(right, #95FF65, #FFFFFF);
  background: -o-linear-gradient(right, #95FF65, #FFFFFF);
  background: linear-gradient(to right, #95FF65, #FFFFFF);
}
```

model1の登録画面のページタイトルを変更する

画面を限定させる場合は画面識別子を指定します。登録画面は Insert とモデルIDで表現します。customize¥webapp¥model1¥MyInsertModel1.css を用意します。

```
.pagetitle {
  border-left: 10px solid #CC0000;
  filter:
 progid:DXImageTransform.Microsoft.gradient(GradientType=1,startColorstr="#FF2819",endColorstr="#FFFFFF");
  background: -webkit-linear-gradient(left, #FF2819, #FFFFFF);
  background: -moz-linear-gradient(right, #FF2819, #FFFFFF);
  background: -ms-linear-gradient(right, #FF2819, #FFFFFF);
  background: -o-linear-gradient(right, #FF2819, #FFFFFF);
  background: linear-gradient(to right, #FF2819, #FFFFFF);
}
```

model1の更新画面のページタイトルを変更する

更新画面は Update とモデル ID で表現した画面識別子を使います。customize¥webapp¥model1¥MyUpdateModel1.css を用意します。

```
.pagetitle {
  border-left: 10px solid #E49B0F;
  filter:
 progid:DXImageTransform.Microsoft.gradient(GradientType=1,startColorstr="#FBE731",endColorstr="#FFFFFF");
  background: -webkit-linear-gradient(left, #FBE731, #FFFFFF);
  background: -moz-linear-gradient(right, #FBE731, #FFFFFF);
  background: -ms-linear-gradient(right, #FBE731, #FFFFFF);
  background: -o-linear-gradient(right, #FBE731, #FFFFFF);
  background: linear-gradient(to right, #FBE731, #FFFFFF);
}
```

例 フォーカスセット時に背景色と入力欄を囲む枠を表示する

テキストボックス、日付ドロップダウンカレンダ、リストボックス、テキストエリアにフォーカスがセットされた時に背景色と入力欄を囲む枠を表示することができます。

```
@charset "UTF-8";
```

```
div.dijitTextBoxFocused.dijitTextBox,
table.dijitSelectFocused.dijitSelect,
textarea.dijitTextBoxFocused.dijitTextBox {
 /* 背景色を変える */
 background: #FFF8DC;
 /* 入力欄を囲む枠を表示する */
 outline: 3px solid #CBE6F3;
}
.soria table.dijitSelectFocused.dijitSelect > tbody > tr >
 td.dijitReset.dijitStretch.dijitButtonContents,
.tundra table.dijitSelectFocused.dijitSelect > tbody > tr >
 td.dijitReset.dijitStretch.dijitButtonContents,
.nihilo table.dijitSelectFocused.dijitSelect > tbody > tr >
 td.dijitReset.dijitStretch.dijitButtonContents
{
  /* リストボックスの背景色を変える */
  background: #FFF8DC;
}
```

CSS テーマ soria, tundra, nihilo はリストボックスのテーブルに背景画像が設定されているため、その対応を行っています。

例 一覧表示の項目間を黒色にする

CSS によって罫線の色を変更することができます。

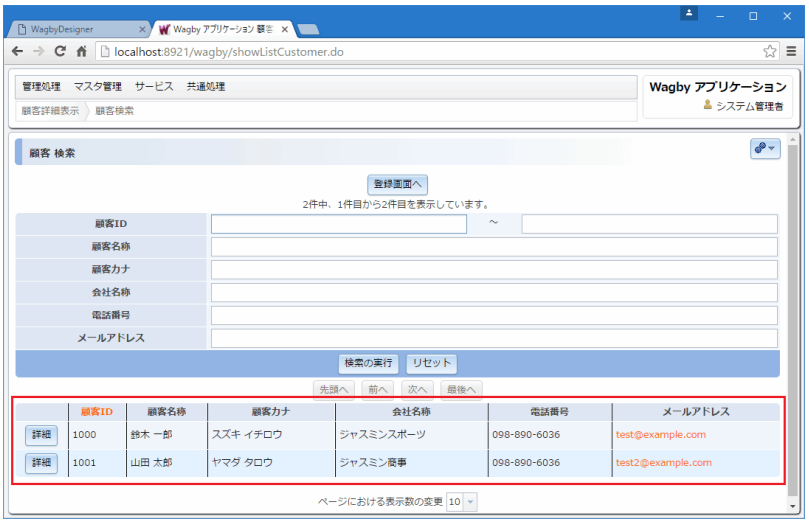


図 項目間の色を変更した例

```
@charset "UTF-8";

.display_label_list:first-child
{
  border-style:none;
}
.display_label_list
{
  border-style:solid;
  border-width:0px 0px 0px 1px;
  border-color:#000;
}

.display_field_list td:first-child,
.display_field2_list td:first-child,
.display_field_list_selected td:first-child,
.display_field2_list_selected td:first-child
```

```
{
  border-style:none;
}
.display_field_list td,
.display_field2_list td,
.display_field_list_selected td,
.display_field2_list_selected td
{
  border-style:solid;
  border-width:0px 0px 0px 1px;
  border-color:#000;
}
```

例 モダンカレンダビューの文字サイズを固定する

Wagbyの文字フォントサイズを「大きいフォントサイズ」にすると、モダンカレンダビューの日付とカレンダデータが重なって表示される場合があります。これを防ぐため、モダンカレンダビューに表示される文字（タイトル部をマウスで押下した時に表示されるダイアログを含む）のサイズを固定する設定を示します。

この設定は mycommon.css に用意します。

```
div.dojoxCalendar *,
#jfcCalendarItemDialog *,
#jfcCalendarViewMiniCalendar * {
  font-size: small;
}
```

任意のCSSを追加する

開発者が独自に用意した CSS ファイルを読み込ませることができます。

変更する範囲	開発者が修正するファイル
アプリケーション全体	customize¥webapp¥tiles2-base_classic2.jsp
画面単位	customize¥webapp¥<モデルID>¥<画面識別子><モデルID>_javascript.jsp

例 すべての画面で任意のCSSファイルを読み込ませる

wagbyapp¥webapps¥$APPNAME¥tiles2-base_classic2.jspを修正します。このファイルは先頭でさまざまなCSSファイル読み込みを行なってるので、そこに開発者が（読み込みの処理を）挿入します。挿入位置は任意に決めてください。

具体的には、独自で開発した CSS ファイルを読み込むためのscriptタグを指定します。

JavaScriptのカスタマイズ（JavaScript）

変更するファイル

Wagbyが生成したJavaScriptファイルを上書きするのではなく、「追加ファイル」を用意する方法を提供しています。これを使って標準の動作を変更することができます。

変更する範囲	開発者が作成するファイル	保存フォルダ（*1）
アプリケーション全体	mycommon.js	customize¥webapps¥system
モデル単位	My.js8.1.0	customize¥webapps¥<モデルID>
画面単位	My<画面識別子>.js8.1.0	customize¥webapps¥<モデルID>

メニュー	menu.js (*2)	customize¥webapps¥system

(189) (190)

任意のJavaScriptを追加する

開発者が独自に用意した JavaScript ファイルを読み込ませることができます。

変更する範囲	開発者が修正するファイル
アプリケーション全体	customize¥webapp¥tiles2-base_classic2.jsp
画面単位	customize¥webapp¥<モデルID>¥<画面識別子><モデルID>_javascript.jsp

例 すべての画面で任意のJavaScriptファイルを読み込ませる

wagbyapp¥webapps¥$APPNAME¥tiles2-base_classic2.jspを修正します。このファイルは先頭でさまざまなJavaScriptファイル読み込みを行なってるので、そこに開発者が（読み込みの処理を）挿入します。挿入位置は任意に決めてください。

具体的には、独自で開発した JavaScript ファイルまたは （jQueryなどの） ファイルを読み込むためのscriptタグを指定します。

個々の画面で任意のJavaScriptファイルを読み込ませる

Wagby が生成した XXX_javascript.jsp というファイルがあります。これにscriptタグを追加するとよいでしょう。

例えば customer モデルの詳細画面のために showCustomer_javascript.jsp というファイルが生成されています。これを修正し、customize¥webapp¥customer フォルダに保存します。

例 ボタン押下時にダイアログを表示する

モデル model1 の登録画面で、保存ボタンを押下したときに確認ダイアログを表示する例を示します。

customize¥webapp¥customer¥MyInsertModel1.js を作成します。

(191)

```
//
// 画面表示時に保存ボタンの押下時に確認ダイアログを
// 表示する処理をセットする
//
require(["dijit/registry", "dojo/ready"], function(registry, ready) {
  // 画面表示時に実行する
  ready(function() {
    // 保存ボタン(id が btnSend)のボタンを取得する
    var btnSend = registry.byId("btnSend");
    // 保存ボタンのデフォルトの click イベントを書き換える
    btnSend.onClick = function(e) {
      // 確認ダイアログを表示する
      showConfirmDialog("処理を続行します。よろしいですか？", function() {
        // OK ボタン押下時の処理
        // this の参照先が変わるため、click イベントオブジェクトから
        // form を取得する
        doSubmitForm(e.target.form, 'action_Send');
      });
    }
  });
```

(189) 1. インストール直後は、customize¥webappフォルダまでが用意されています。開発者ご自身で必要なフォルダの作成を行ってください。

(190) 2. menu.js は「上書き」になります。本節の後半にカスタマイズ事例を掲載しています。

(191) ビルドすると wagbyapp¥webapps¥wagby¥model1 にファイル MyInsertModel1.js が生成されています。中身は空です。これを使うとよいでしょう。

```
});
```

- 保存ボタン押下時の処理をOKボタン押下時の関数内に移すため、form の参照先(this の参照先)がボタンからダイアログに変わります。上のサンプルコードでは、ボタンクリック時のイベントオブジェクトから formを取得するようにしています。
- 独自ボタンなど、他のボタンにも同じ考え方を適用できます。

例 特定のボタンを非表示にする

モデル model1 の詳細画面で、集計ビューに関する（複数の）ボタンを非表示にする例を示します。

customize¥webapp¥customer¥MyShowModel1.js を作成します。

(192)

```
require(["dijit/registry", "dojo/dom-style", "dojo/ready"], function(registry, domStyle, ready)
 {
  // 画面表示時に以下の関数が実行される
  ready(function() {
    // show<モデルID>_body.jsp 内の集計ビューのボタンのIDを配列に設定する
    // ボタンのIDの命名規則は、btnSendShowList<モデルID>Totalview
    // 集計ビューが複数ある場合、2番目以降のボタンのIDは、
    // btnSendShowList<モデルID>T<集計ビュー番号>Totalview
    var totalviewButtons = [
      "btnSendShowListModel1Totalview",   // 1番目の集計ビューボタンのID
      "btnSendShowListModel1T2Totalview", // 2番目の集計ビューボタンのID
    ];

    // 上記集計ビューボタンのIDからボタンのオブジェクトを取得し、ボタンの
    // DOM要素の style 属性に display: none を設定し非表示にする
    for (var i=0,l=totalviewButtons.length; i<l; i++) {
      var btnShowTotalview = registry.byId(totalviewButtons[i]);
      if (!btnShowTotalview) {
        continue;
      }
      domStyle.set(btnShowTotalview.domNode, "display", "none");
    }
  });
});
```

- 集計ビューのボタンの命名規則は btnSendShowList<モデルID>Totalview となっています。また集計ビューが複数ある場合、2番目以降のボタンのIDは、btnSendShowList<モデルID>T<集計ビュー番号>Totalview となっています。
- 上のサンプルコードでは、配列にボタンのIDを設定し、これらボタンのDOM要素に display:none を設定することで非表示化しています。

例 任意のボタンにTipsを用意する

title属性を使う

customer モデルの一覧表示画面に用意される「登録画面へ」というボタンに Tips を用意してみます。ここでは HTML 5 規格で用意された title 属性を使います。

(192) ビルドすると wagbyapp¥webapps¥wagby¥model1 にファイル MyShowModel1.js が生成されています。中身は空です。これを使うとよいでしょう。

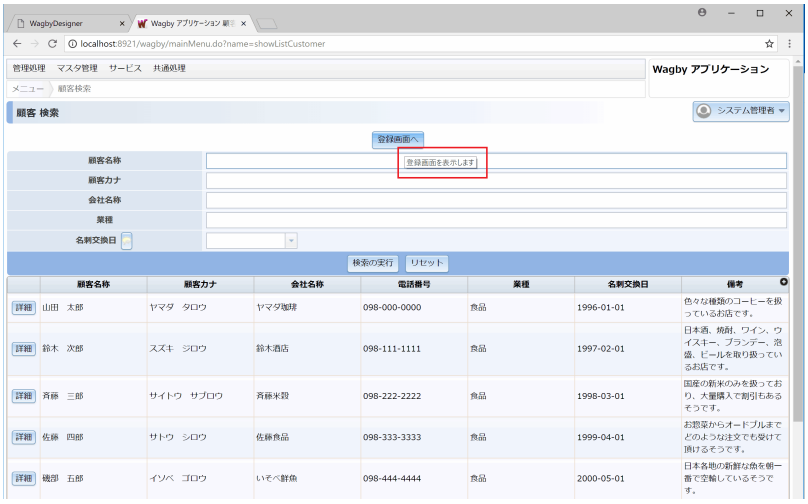


図 登録画面にTipsを用意する

customize¥webapp¥customer¥MyShowListCustomer.js を作成します。

(193)

```
require(["dijit/registry", "dojo/ready"], function(registry, ready) {
 ready(function() {
   // 「登録画面へ」ボタンを id 指定で取得する
   var btnRegistration = registry.byId("btnNewInsertCustomer");
   // 「登録画面へ」ボタンの title 属性を設定する
   btnRegistration.set("title", "登録画面を表示します");
 });
});
```

・ 対象とするボタンの id は、自動生成された JSP ファイルの内容をテキストエディタで確認してください。

Dojotoolkit の Tooltip を使う

(Wagbyが内部で利用しているJavaScriptライブラリである) Dojotoolkit が提供する Tooltip を使うこともできます。

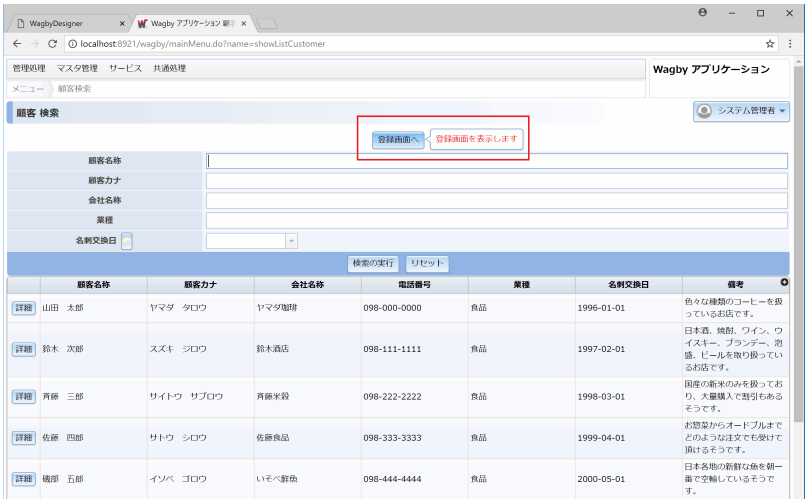


図 登録画面にTipsを用意する(2)

上と同じように customize¥webapp¥customer¥MyShowListCustomer.js を作成します。

(193) ビルドすると wagbyapp¥webapps¥wagby¥customer にファイル MyShowListCustomer.js が生成されています。中身は空です。これを使うとよいでしょう。

```
require(["dijit/registry", "dijit/Tooltip", "dojo/ready"], function(registry, Tooltip, ready) {
 ready(function() {
   new Tooltip({
     // ツールチップを表示する対象となる「登録画面へ」ボタンの id を設定する
     connectId: ["btnNewInsertCustomer"],
     // ツールチップの文言を設定
     label: "<span style='color: red;'>登録画面を表示します</span>",
   });
 });
});
```

- 対象とするボタンの id は、自動生成された JSP ファイルの内容をテキストエディタで確認してください。
- 上の例のように、ツールチップの文言には HTML を含めることができます。

メッセージの国際化

メッセージを国際化対応する場合は、Dojotoolkit の国際化の枠組みを使います。

(194)

ビルドしたアプリケーション wagbyapp¥webapps¥wagby¥nls フォルダにある common.js を編集します。 他と重複しないリソースキーを割り当てて文を登録します。以下は TIPS_CUSTOMER_BTN_INSERT というリソースキーを使った例です。

```
TIPS_CUSTOMER_BTN_INSERT: "登録画面を表示します",
```

設定したリソースは JavaScript 内で res オブジェクトで参照できます。以下のようになります。

HTML 5 の title 属性を利用した場合:

```
btnRegistration.set("title", res.TIPS_MODEL1_BTN_INSERT);
```

Dojotoolkit の Tooltip を利用した場合:

```
label: "<span style='color: red;'>" + res.TIPS_MODEL1_BTN_INSERT + "</span>",
```

入力フォーカスの制御

フォーカスを無効にする

新規登録画面のフォーカスを無効にする場合、MyInsert<モデルID>.js ファイルを用意し、標準の focusFirstElement 関数を空（処理なし）で上書きしてください。

```
function focusFirstElement() {
}
```

任意のボタンのIDにフォーカスをセットする

次の JavaScript コードは、設定したボタンのIDにフォーカスをセットします。

```
function focusFirstElement() {
 require(["dijit/registry", "dojo/ready"], function(registry, ready) {
   ready(function() {
     var btn = registry.byId("<ボタンのID>");//ボタンIDを指定してください。
     if (btn) {
       btn.focus();
     }
   });
 });
}
```

(194) サーバ側ではなく、クライアント（Webブラウザで動作するJavaScript）の国際化になります。

メニュー表示エリアの高さを変更する

標準では、ポータル表示エリアおよびメニュー表示エリアの高さは 600 ピクセルで固定されています。これを変更するスクリプトを紹介します。(*6)

このスクリプトを customize¥webapp¥system フォルダに menu.js として保存します。(標準の system¥menu.js を上書きします。)

```
function init() {
 var tabindex = getCookie("__jfc_cookie_tabindex");
 var tabindex_head = getCookie("__jfc_cookie_tabindex_head");
 if (tabindex == null || tabindex_head == null){
  tabindex = 1;
  tabindex_head = tabindex;
 }
 seltab2('box','head',eval(tabindex),eval(tabindex_head));
}

Initializer.addInit(init);

require(["dojo/dom", "dojo/dom-style"], function(dom, domStyle) {
 var mainMenuContainer = dom.byId("mainMenuContainer");
 if (mainMenuContainer) {
   domStyle.set(mainMenuContainer, "height", "900px");
 }
});
```

- 1行目から11行目までは、標準で同梱されている system/menu.js と同じ内容です。13行目以降が追加したスクリプトになっています。
- 上の例では 900 ピクセルとしています。この値を適切に変更してください。(*7)

(195) (196)

出力制御（スタイル）

隠し項目（詳細・一覧）

詳細画面

モデル項目詳細定義の「出力制御 － 詳細画面」タブに用意された「共通」の「隠し項目」を有効にします。

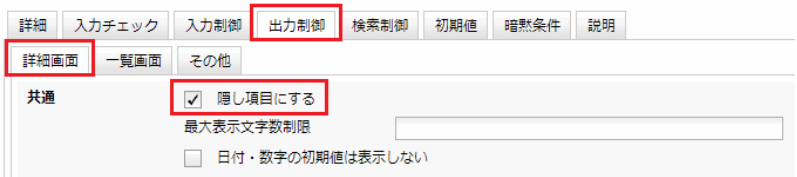


図 詳細画面の設定

この設定は入力制御の隠し項目と併用できます。いずれも設定すると、利用者からは存在しないようにみえます。

(195) 6. ポータルはメニューのタブに含まれているため、メニューのタブの高さを変更すると、ポータルの表示エリアの高さも同時に変わります。

(196) 7. JavaScriptでウインドウの高さを取得することはできますが、Wagbyは「サブメニュー」など、上部にさまざまなコンテンツを含めた画面レイアウトとなっているため、ブラウザのウインドウの高さをそのままセットすることはできません。

一覧画面

モデル項目詳細定義の「出力制御 － 一覧画面」タブに用意された「共通」の「隠し項目にする」を有効にします。

このモデルが外部キー子モデルとなっている場合、親モデル側に子モデルの一覧表示を行うことができます。「隠し項目にする（親モデルとの同時表示時）」を設定すると、親モデル内に表示される場合のみ隠し項目とします。

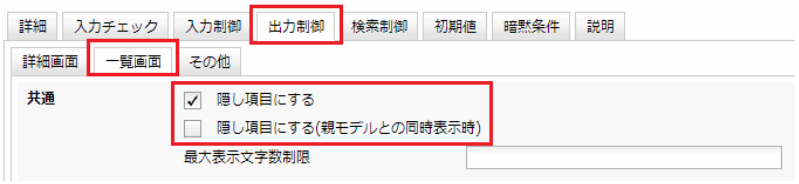


図 詳細画面の設定

この設定は入力制御の隠し項目と併用できます。いずれも設定すると、利用者からは存在しないようにみえます。

ラベルの表示内容

登録・更新・詳細画面

画面例

詳細表示画面のラベル名を「顧客」から「顧客（詳細）」に変更した例を図に示します。

図 詳細表示画面

定義方法

モデル項目詳細定義の「出力制御 － 詳細画面」タブに用意された「スタイル」の「ラベルの表示内容」を設定します。

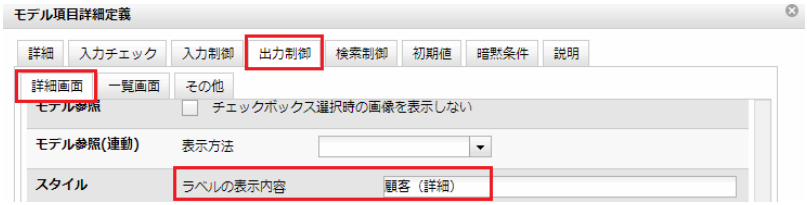


図 ラベルの表示内容の設定

この設定は詳細表示画面、登録画面、更新画面に反映されます。

一覧表示画面

画面例

詳細表示画面のラベル名を「顧客」から「顧客（一覧）」に変更した例を図に示します。

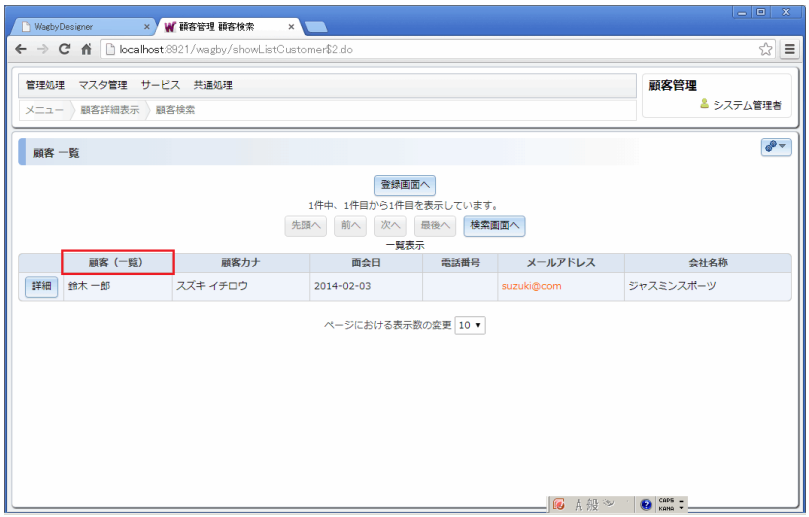


図 一覧表示画面

定義方法

モデル項目詳細定義の「出力制御 - 一覧画面」タブに用意された「スタイル」の「ラベルの表示内容」を設定します。

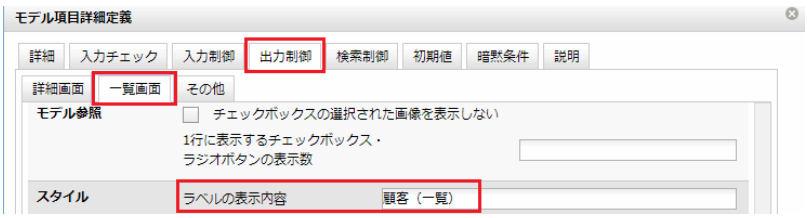


図 ラベルの表示内容の設定

検索画面

画面例

詳細表示画面のラベル名を「顧客」から「顧客（検索）」に変更した例を図に示します。

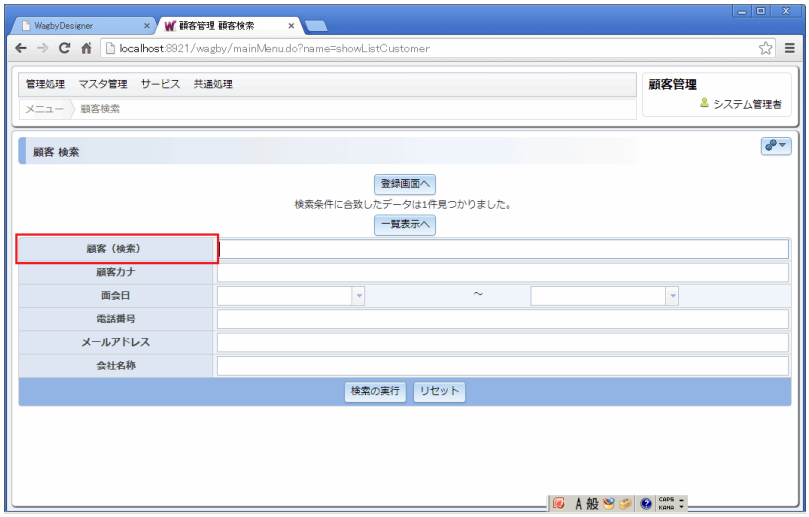


図 検索画面

定義方法

モデル項目詳細定義の「検索制御」タブに用意された「スタイル」の「ラベルの表示内容」を設定します。

図 ラベルの表示内容の設定

色のルール

ラベルの文字色は次のルールに従います。

色	説明	CSSクラス名
黒	標準	display_label
赤	必須項目	display_label_notnull
青	読み込み専用	display_label_readonly
緑	必須かつ読み込み専用	display_label_notnull_readonly

ラベルと値の表示方法

画面例

顧客モデルの「備考」項目のラベルと値の表示方法を「縦」とした例を図に示します。

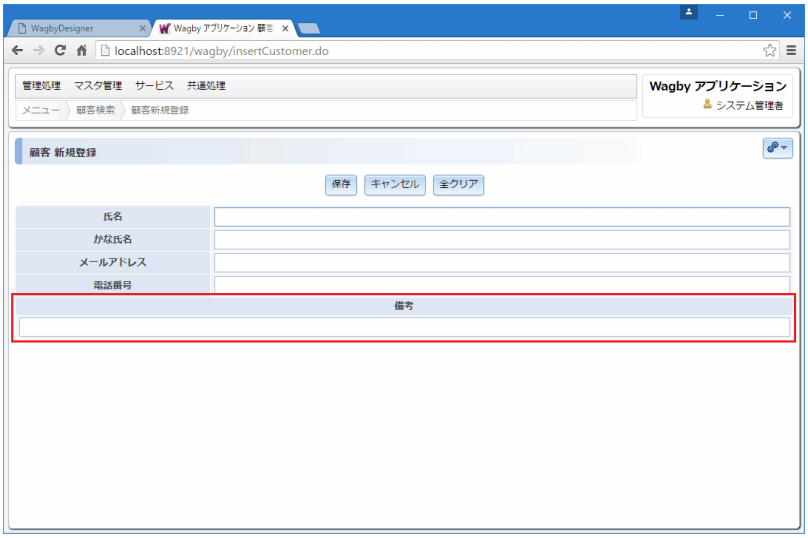


図 新規登録画面

定義方法

モデル項目詳細定義の「出力制御 － 詳細画面」タブに用意された「スタイル」の「ラベルと値の表示方法」を設定します。

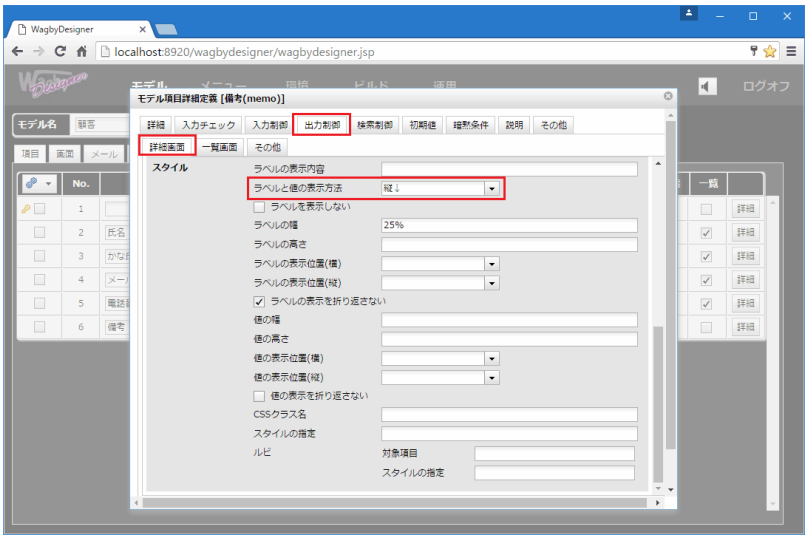


図 ラベルと値の表示方法の設定

項目の一括設定を使うと、全体をまとめて確認できます。

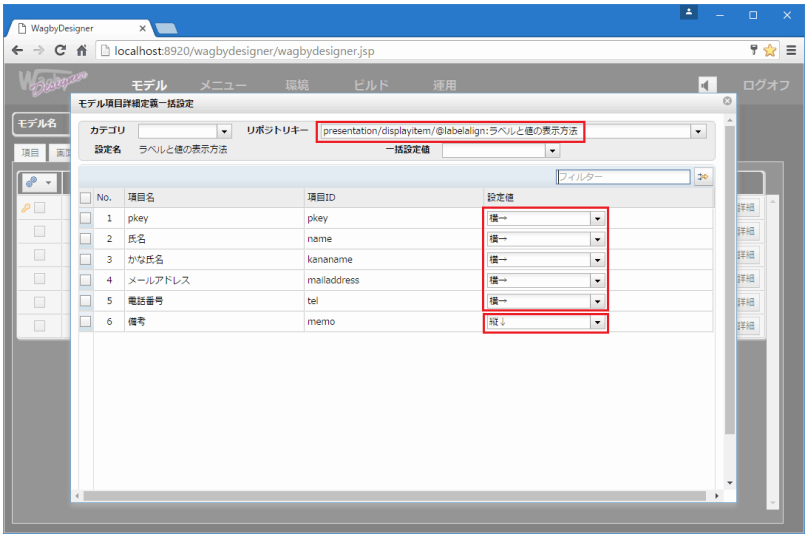


図 一括設定

この設定は詳細表示画面、登録画面、更新画面に反映されます。

すべてを縦とした場合

すべての項目を「縦」とした例を図と図に示します。

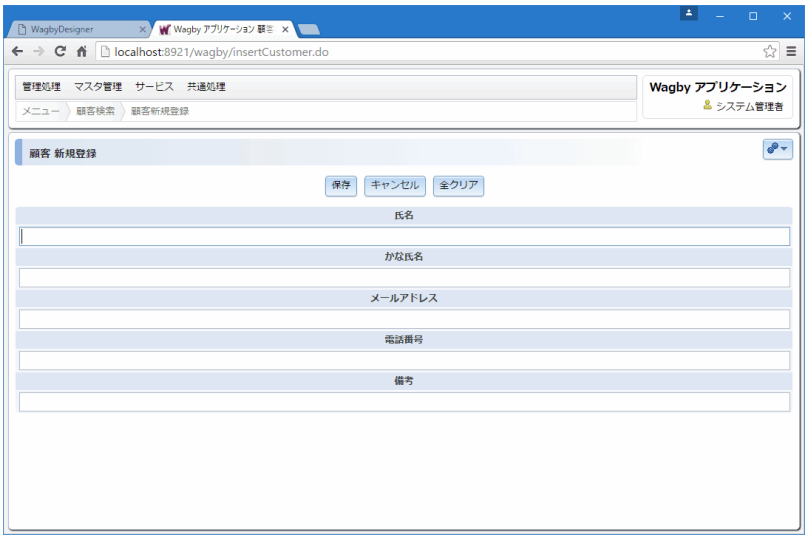


図 登録画面

図 詳細画面

ラベルを表示しない

画面例

顧客モデルの「氏名」項目のラベルを非表示とした例を図、図に示します。

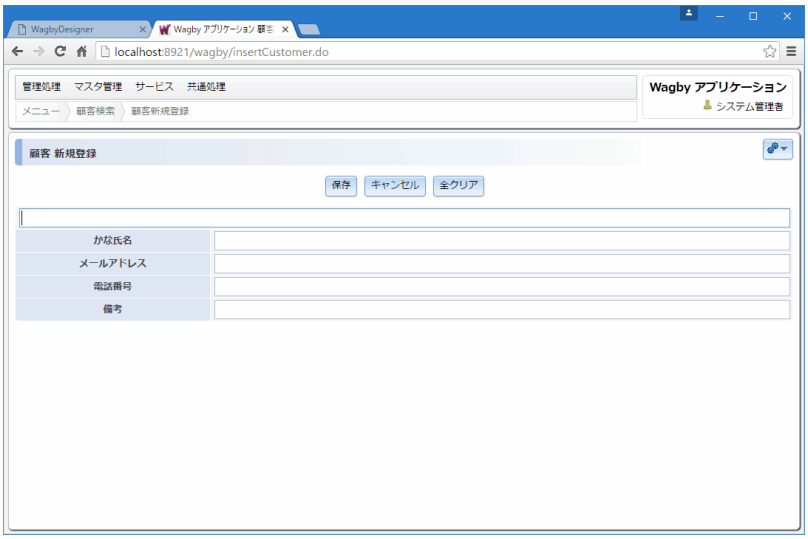


図 新規登録画面

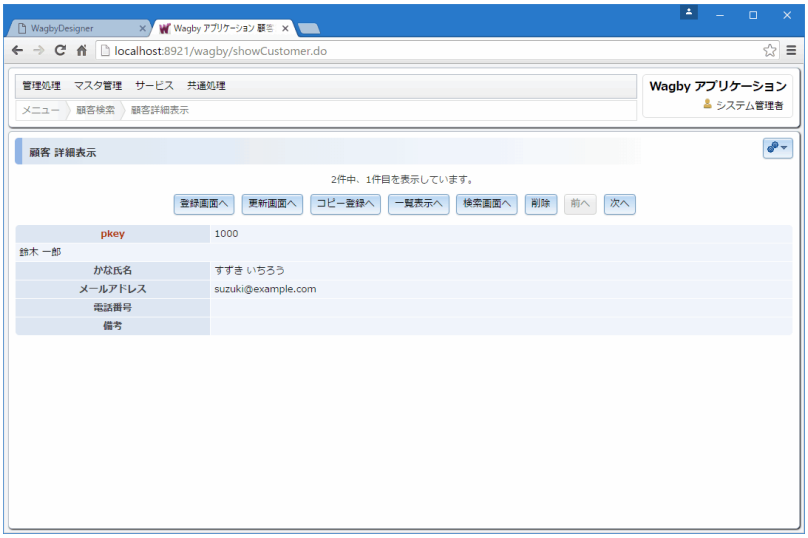


図 詳細画面

定義方法

モデル項目詳細定義の「出力制御 － 詳細画面」タブに用意された「スタイル」の「ラベルを表示しない」設定を有効にします。

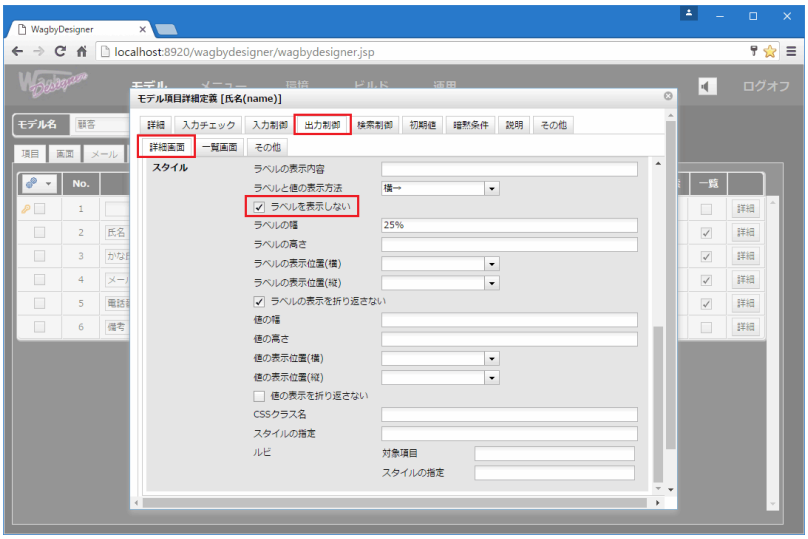


図 ラベルを表示しない

この設定は詳細表示画面、登録画面、更新画面に反映されます。

仕様・制約

日付、時刻、郵便番号型項目は標準で（ラベル部分に）ヘルプボタンが表示されるため、本設定「ラベルを表示しない」を有効にしてもラベルが表示されます。

これらの型を利用した項目でラベルを非表示にするためには、本設定「ラベルを表示しない」を有効にし、かつ、「説明」タブの「登録画面」「更新画面」「検索画面」「詳細画面」にそれぞれ用意された、「日付・時刻・郵便番号型 ＞ ヘルプボタンを表示する」設定のチェックを外してください。

ラベルの幅

標準レイアウトでラベル幅を変更する

画面例

図は、ラベル幅を10%に変更した例です。（ラベル幅の標準値は25%となっています。）

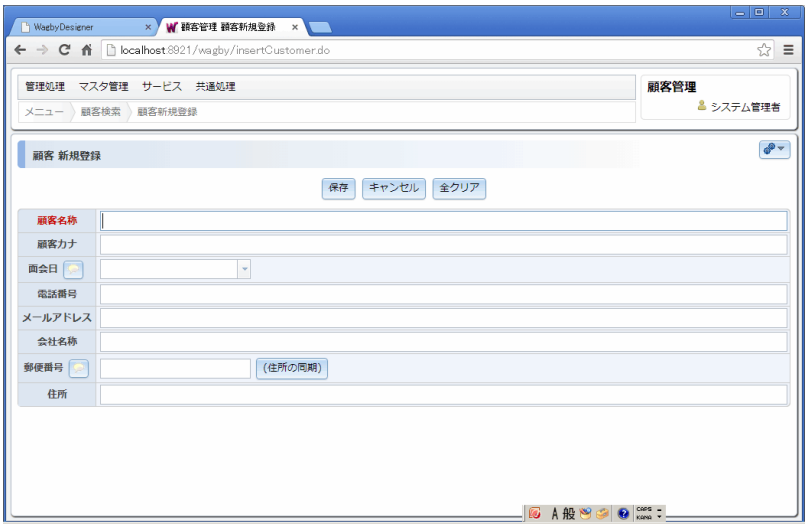


図 ラベル幅を10%に変更した例

定義方法

ラベル幅を標準値25%よりも狭める場合は、すべての項目についてモデル項目詳細定義の「出力制御 － 詳細画面」タブに用意された「スタイル」の「ラベルの幅」を設定します。

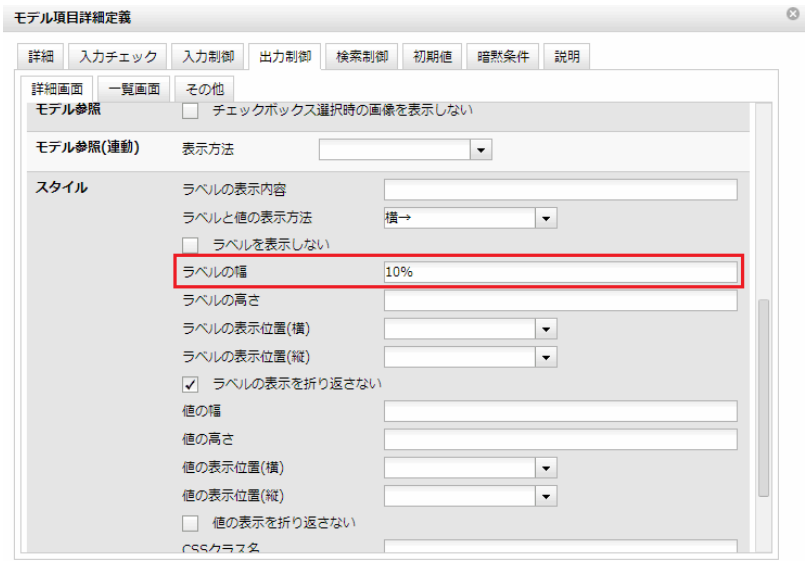


図 ラベル幅の設定

注意

すべての項目のラベル幅を変更してください。一つでも設定漏れがあると図に示すようにもっとも大きい値（ここでは標準値の25%）に合わせられます。

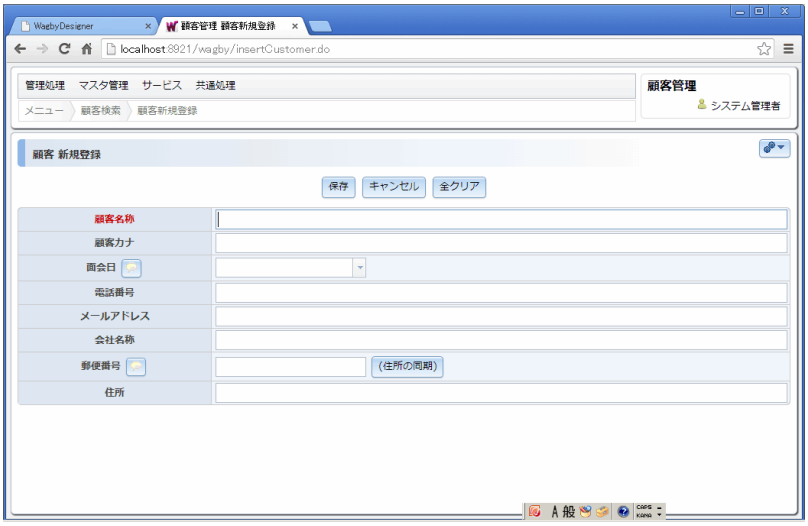


図 設定漏れのため（もっとも大きな値である）標準値の25%に合わせられた

逆に標準値25%よりも大きなラベル幅とする場合は、1つの項目について設定すればよいです。

図 一つの項目のみラベル幅を50%としても、全体がこれに合わせられる

縦並びレイアウトでラベル幅を変更する

画面例

「顧客名称」と「顧客カナ」を縦並びのレイアウトとして用意します。 その上で「顧客名称」のラベル幅を50%、「顧客カナ」のラベル幅を10%とした例を図に示します。

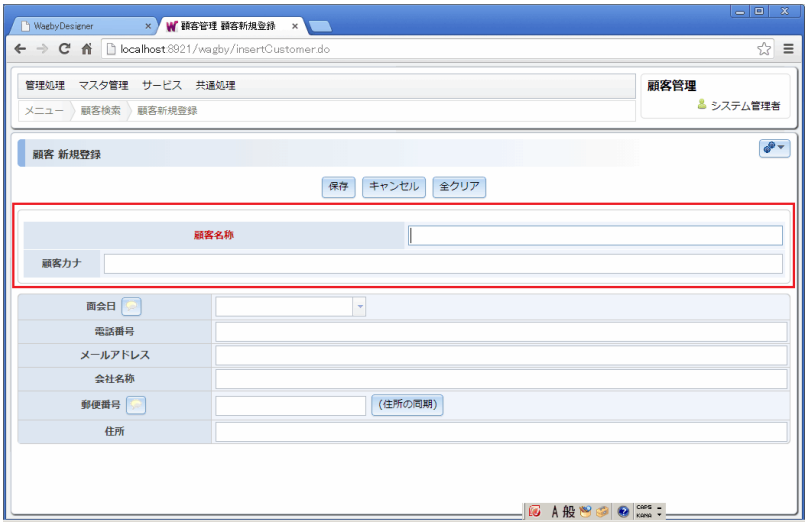


図 各項目のラベル幅を変更

定義方法

それぞれの項目のラベル幅を設定します。

横並びレイアウトでラベル幅を変更する

画面例

「顧客名称」と「顧客カナ」を横並びのレイアウト（両者の表示比率は50%ずつ）として用意します。 その上で「顧客名称」のラベル幅を50%、「顧客カナ」のラベル幅を10%とした例を図に示します。 それぞれの項目のラベル幅が異なっていることがわかります。

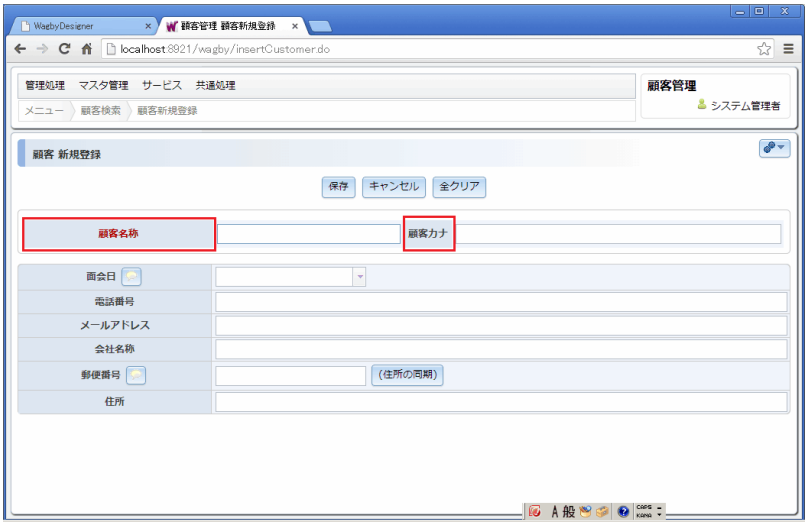


図 各項目のラベル幅を変更

定義方法

それぞれの項目のラベル幅を設定します。

比率ではなくピクセル値を指定する

ラベル幅の値に「%」を除いた数値のみを設定すると、ピクセル値による指定となります。　図は、各ラベル幅をそれぞれ「120」と設定した例です。

図 ラベル幅を120（ピクセル値）と設定した

ここでブラウザのウィンドウ幅を縮小してみます。全体のレイアウトは横並び50%,50%を維持していますが、ラベル幅は120ピクセル値で固定化されていることがわかります。

図 ウィンドウサイズを変更してもラベル幅は変わらない

ラベル幅を固定と設定することで、全体の画面レイアウトはブラウザのウィンドウサイズ変更に対応させつつ、見た目のイメージは変わらないという画面を設計することができます。

固定値の場合、Webブラウザによって見た目が変わることがあります。小さい値（例:10以下の数値）を指定したとき、ブラウザによってはこの数値を無視して自動調整する場合があります。

一覧表示画面への設定

一覧表示画面のラベル幅設定は、モデル項目詳細定義の「出力制御 - 一覧画面」タブに用意された「スタイル」の「ラベルの幅」を設定します。

図 一覧表示画面のラベル幅の設定

(197)　(198)

比率で指定する場合の注意点

一覧表示項目の列幅は、各設定値の合計が100%となるように設定してください。

ただし一覧表示画面の行ごとに「コピー登録」や「詳細」といったボタンを用意したとき、ボタンの表示がずれることがあります。　これらのボタンの列幅を直接、指定することはできないため、このような場合は一覧表示の対象としている項目の列幅の合計を100%より小さめに設定して、回避してください。

ピクセル値で指定する場合の注意点

一覧表示項目の列幅をピクセル値で設定する場合は、次のいずれかの対応を同時に行なってください。

・　テーブルの幅も同時にピクセル値で指定し、固定する。

(197) ブラウザに IE を用いた場合、一覧表示画面のラベル幅を固定にした場合のみスタイルに "table-layout:fixed" が付与されます。

(198) 一覧表示画面の「ラベル幅」と「折り返さない」設定を同時に設定した場合に IE とその他のブラウザで表示が異なることがあります。

・　ヘッダ行固定等を追加設定し、テーブルを固定レイアウトとする。

検索画面への設定

検索画面のラベル幅設定は、モデル項目詳細定義の「検索制御」タブに用意された「スタイル」の「ラベルの幅」を設定します。

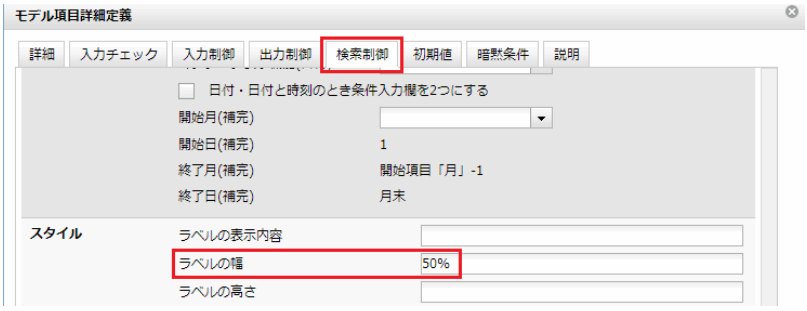


図 検索画面のラベル幅の設定

入力ルール（8.3.2）

″幅″ に入力可能な値は数字とパーセント,px,em,remです。″高さ″ に入力可能な値は数字とpx,em,rem です。

これらの入力項目に全角文字を入力したとき、内部で半角文字に変換されてリポジトリに書き込まれます。（リポジトリ再読み込み後、半角文字になっています。）

この変換処理が行われる項目は次のとおりです。

・　出力制御>詳細画面>スタイル
　・　ラベルの幅
　・　ラベルの高さ
　・　値の幅
　・　値の高さ
・　出力制御>一覧画面>スタイル
　・　ラベルの幅
　・　ラベルの高さ
　・　値の幅
　・　値の高さ
・　検索制御>スタイル
　・　ラベルの幅
　・　ラベルの高さ

ラベルの高さ

画面例

顧客モデルの「氏名」項目のラベルの高さを 50 px に設定した例を図に示します。

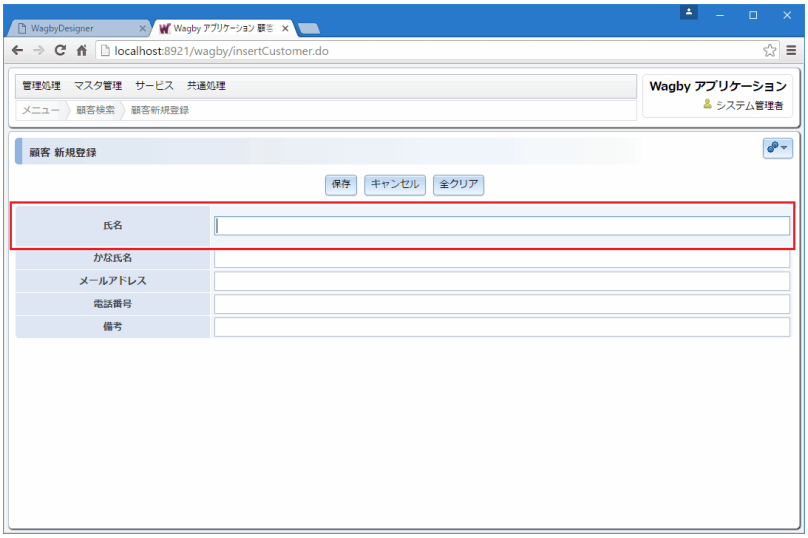


図 新規登録画面

定義方法

モデル項目詳細定義の「出力制御 － 詳細画面」タブに用意された「スタイル」の「ラベルの高さ」に数値と単位を指定します。

(199)

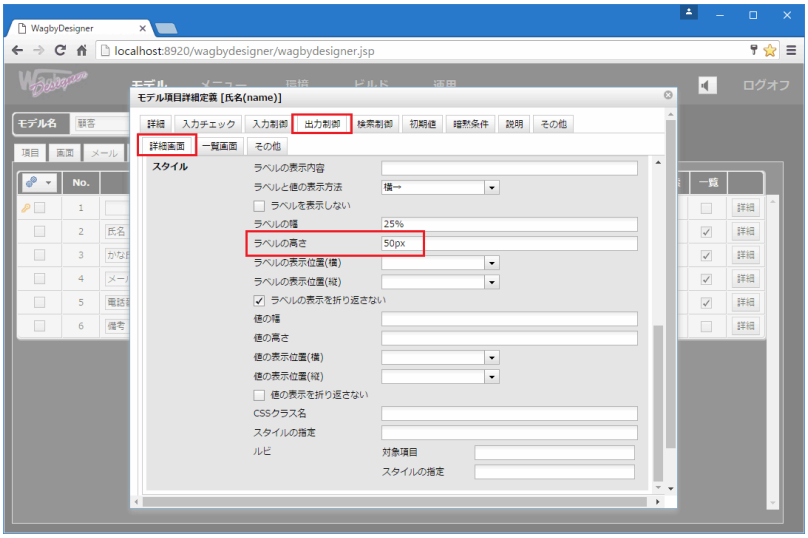


図 ラベルの高さを設定する

この設定は詳細表示画面、登録画面、更新画面に反映されます。

値の高さとの関係

「値の高さ」と併用した場合、通常のレイアウトではどちらか大きい方に合わせられます。

(199) 数値のあとの単位は省略することもできます。省略時は "px"（ピクセル）と解釈されます。

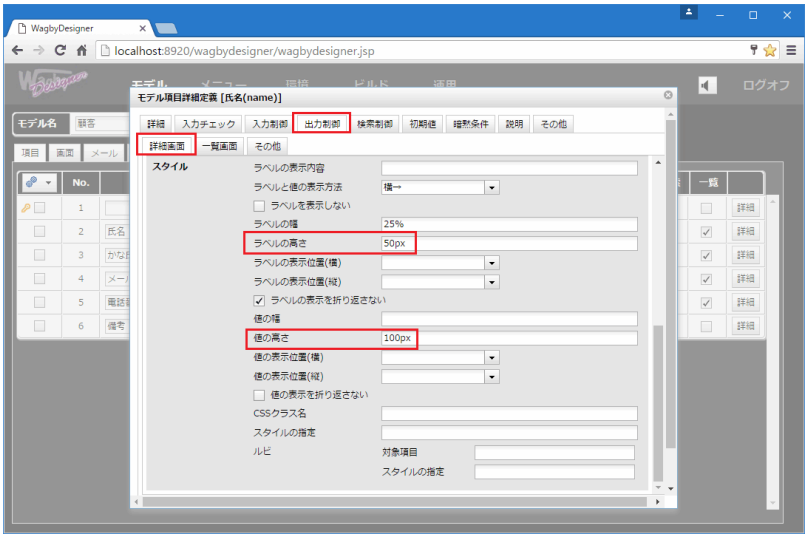


図 ラベルと値の両方の高さを設定する(1)

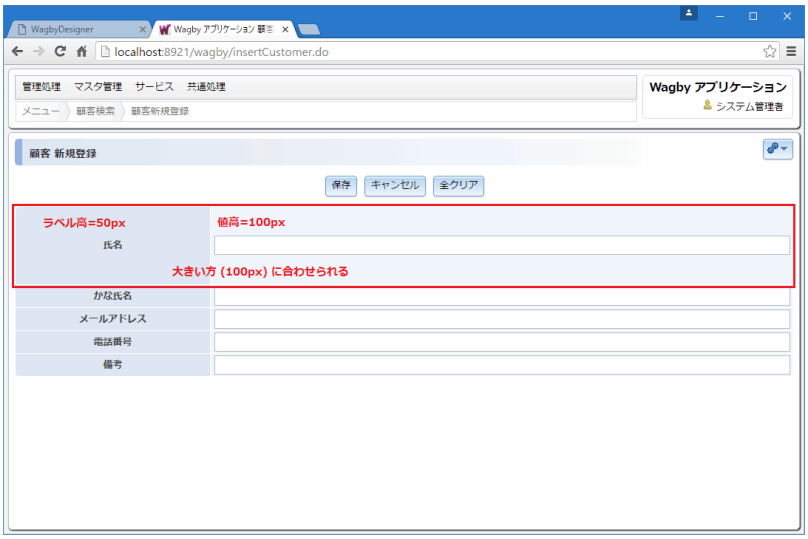


図 ラベルと値の両方の高さを設定する(2)

繰り返しコンテナでは、ラベルの高さと値の高さがそれぞれ反映されます。（繰り返しコンテナの場合、いずれかひとつの項目に設定すると、その他の列も自動的に合わされます。）

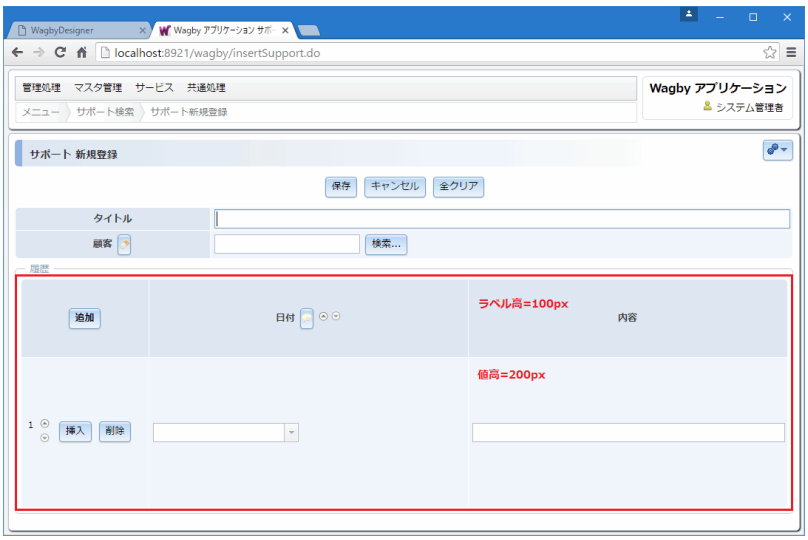


図 繰り返しコンテナに設定した場合

表示位置の制御（横）

詳細画面

画面例

図はラベルと値の部分をそれぞれ〝左寄せ〟, 〝センタリング（中央）〟, 〝右寄せ〟とした詳細画面の例です。

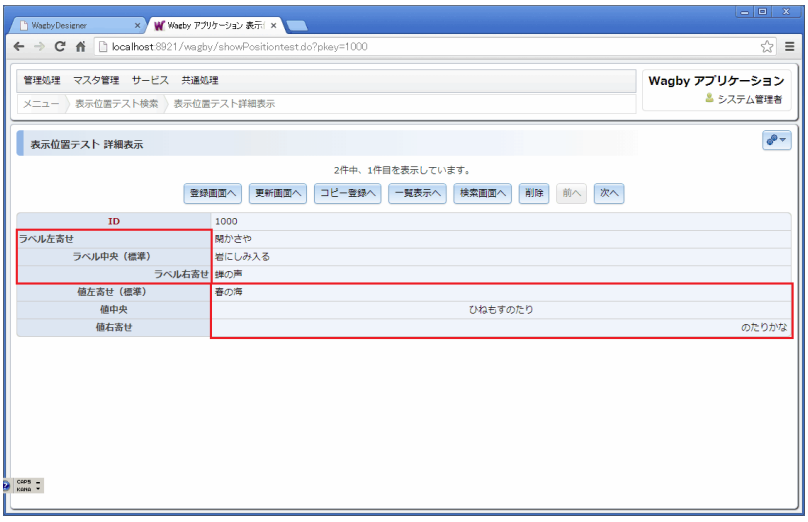


図 表示位置を指定した詳細画面

定義方法

テストで用いた項目定義を図示します。

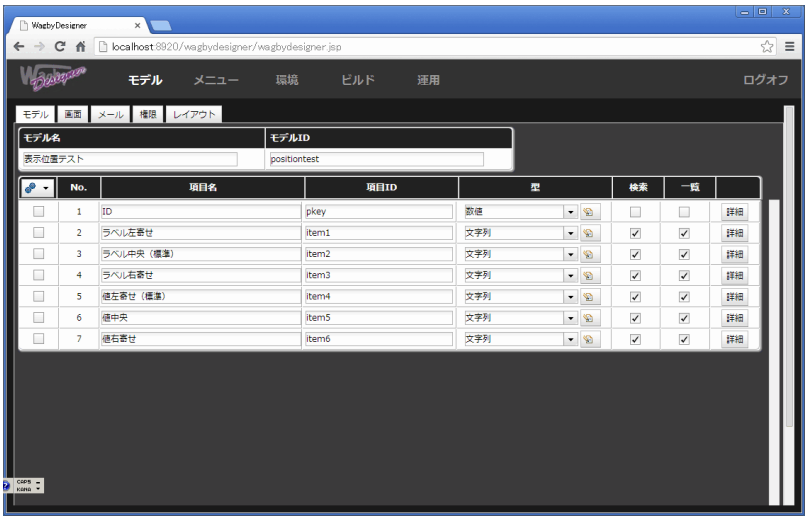


図 テスト定義

ラベルの表示位置

各項目の詳細定義で、出力制御>詳細画面>スタイルを開きます。「ラベルの表示位置（横）」を指定します。空白時は〝センタリング（中央）〟になります。

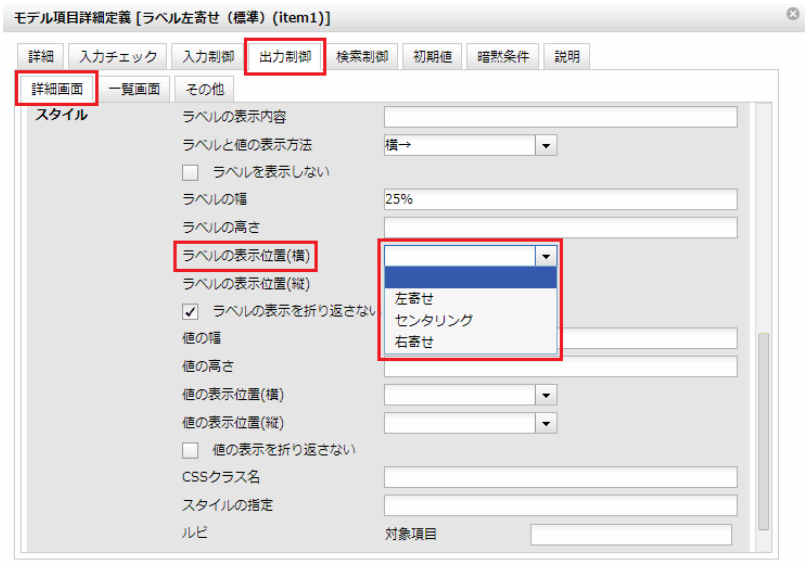


図 ラベルの表示位置（横）

値の表示位置

各項目の詳細定義で、出力制御>詳細画面>スタイルを開きます。「値の表示位置（横）」を指定します。空白時は″左寄せ″になります。

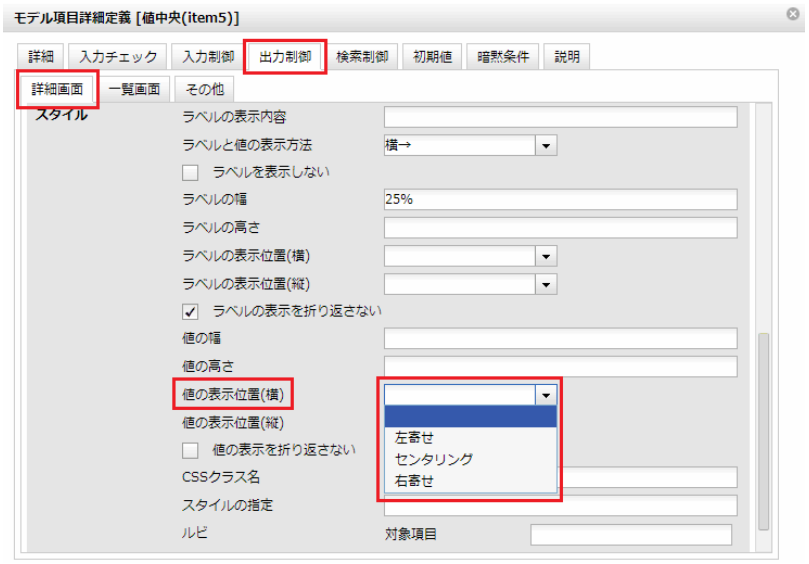


図 値の表示位置（横）

一覧画面

画面例

図はラベルと値の部分をそれぞれ″左寄せ″, ″センタリング（中央）″, ″右寄せ″とした一覧画面の例です。

図 表示位置を指定した一覧画面

ラベルの表示位置

各項目の詳細定義で、出力制御>一覧画面>スタイルを開きます。「ラベルの表示位置（横）」を指定します。空白時は〝センタリング（中央）〟になります。

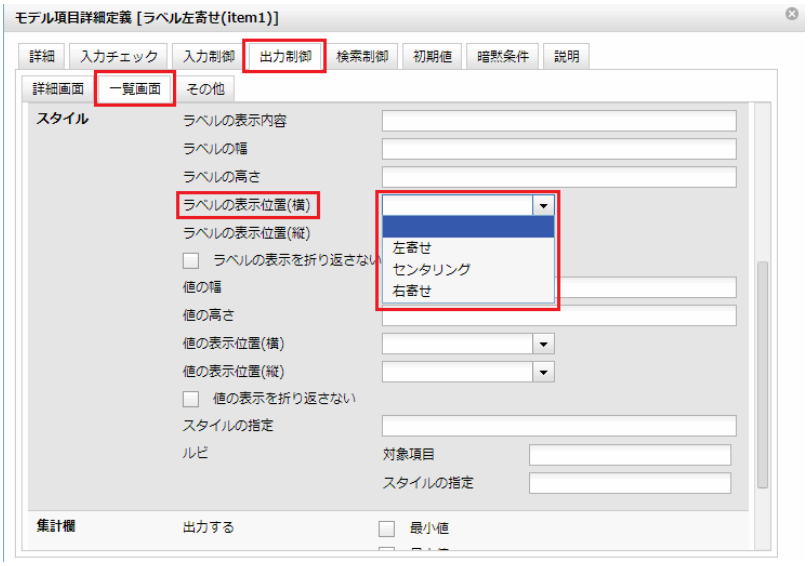


図 ラベルの表示位置（横）

値の表示位置

各項目の詳細定義で、出力制御>一覧画面>スタイルを開きます。「値の表示位置（横）」を指定します。空白時は〝左寄せ〟になります。

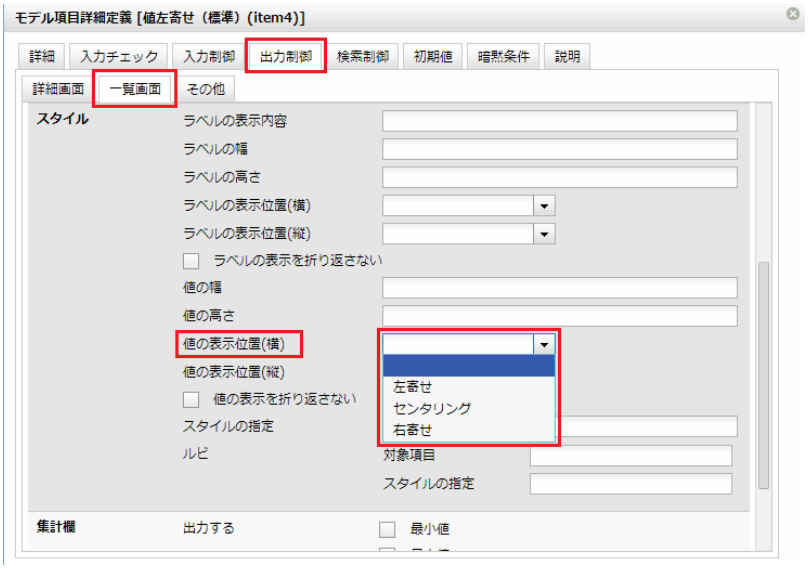


図 値の表示位置（横）

表示位置の制御（縦）

例

図はラベルと値の部分をそれぞれ〝上寄せ〟、〝センタリング〟、〝下寄せ〟とした登録画面です。 登録・更新画面ではラベル部分に反映されます。

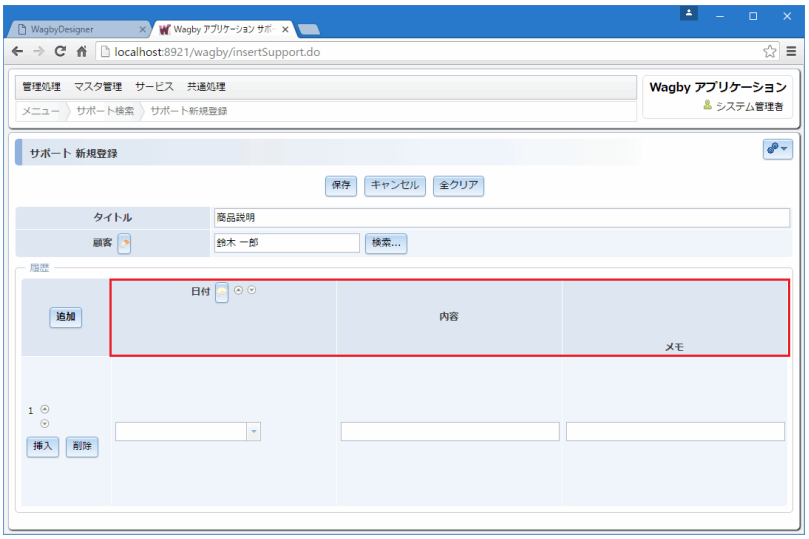


図 表示位置を指定した登録画面

詳細画面では、ラベル部分と値部分の両方の設定が反映されます。

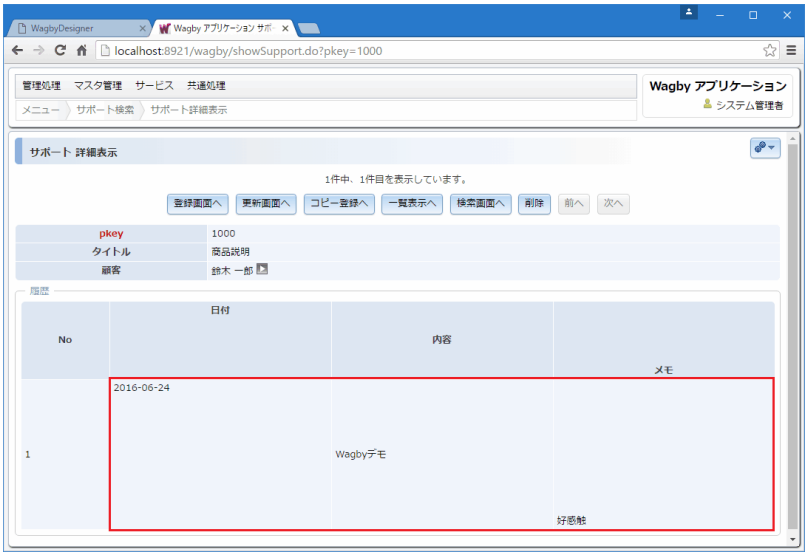


図 表示位置を指定した詳細画面

定義方法

ラベルの表示位置

各項目の詳細定義で「出力制御>詳細画面>スタイル」を開きます。「ラベルの表示位置（縦）」を指定します。空白時は〝センタリング〟になります。

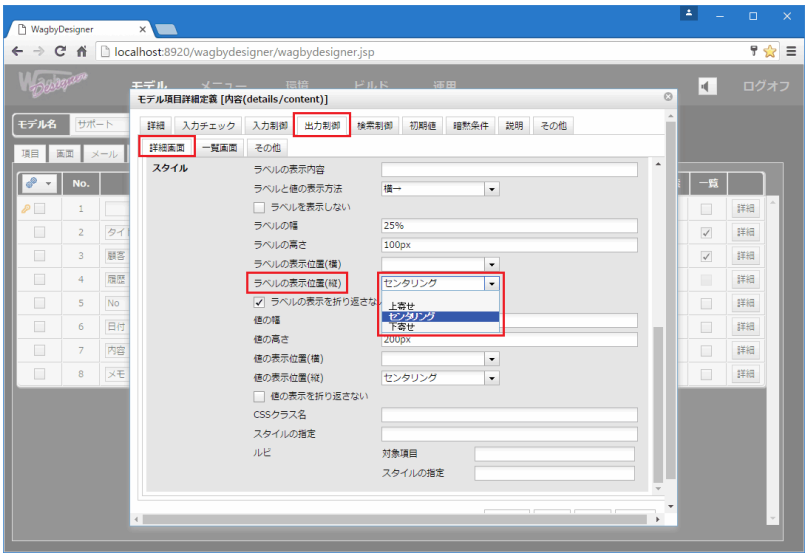


図 ラベルの表示位置の設定

値の表示位置

各項目の詳細定義で「出力制御>詳細画面>スタイル」を開きます。「値の表示位置（縦）」を指定します。空白時は〝センタリング〟になります。

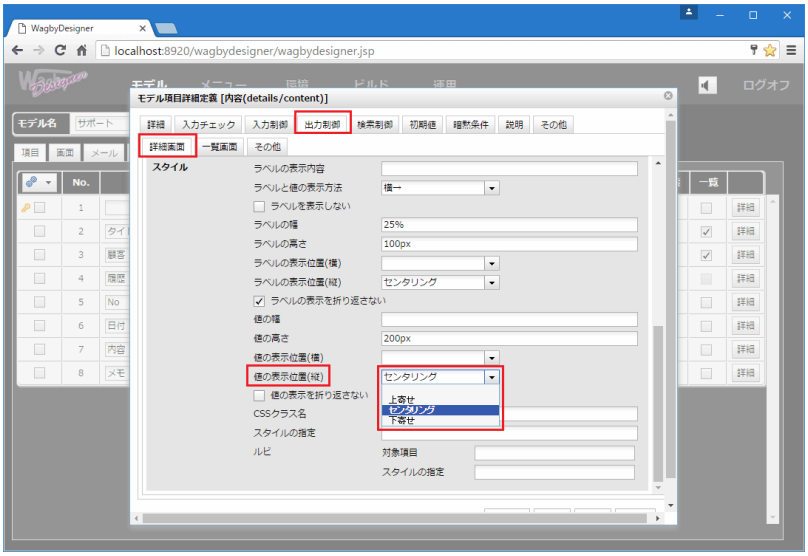


図 値の表示位置の設定

表示の折り返し

詳細画面

ラベル

図はラベルの折り返しを行わない場合です。ラベル文字が長い場合、ラベル幅設定より折り返さない設定が優先されます。標準は「（ラベルを）折り返さない」となっています。

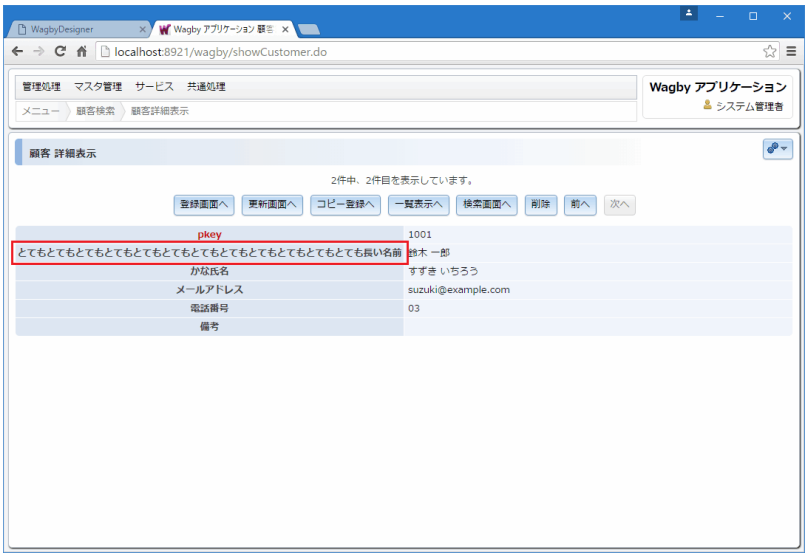


図 ラベルを折り返さない（標準）

図はラベルの折り返しを行った場合です。

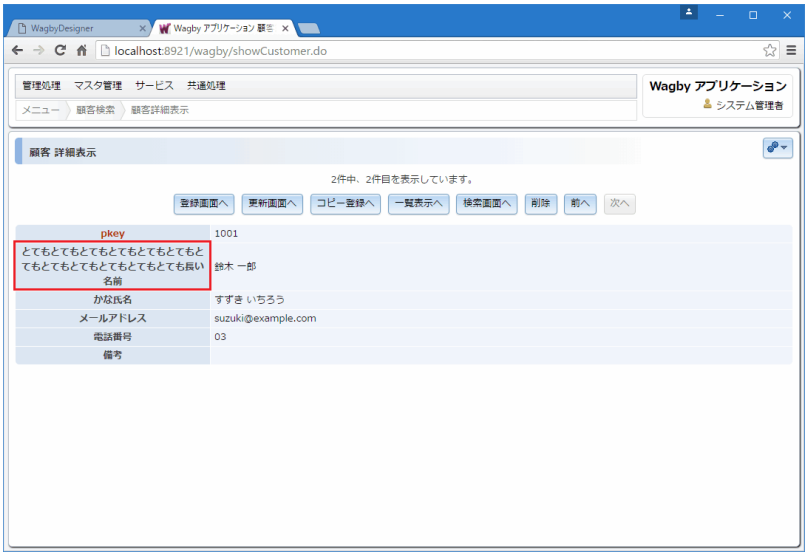


図 ラベルを折り返す

値

図は値の折り返しを行った場合です。標準は「（値を）折り返す」となっています。

図 値を折り返す（標準）

図は値の折り返しを行わない場合です。出力文字が長いとき、ブラウザの横スクロールバーが表示されます。

(200)

(200) この例では「かな氏名」は折り返さないという指定ですが、同一のレイアウトに属するため、「氏名」の動作に合わせられています。

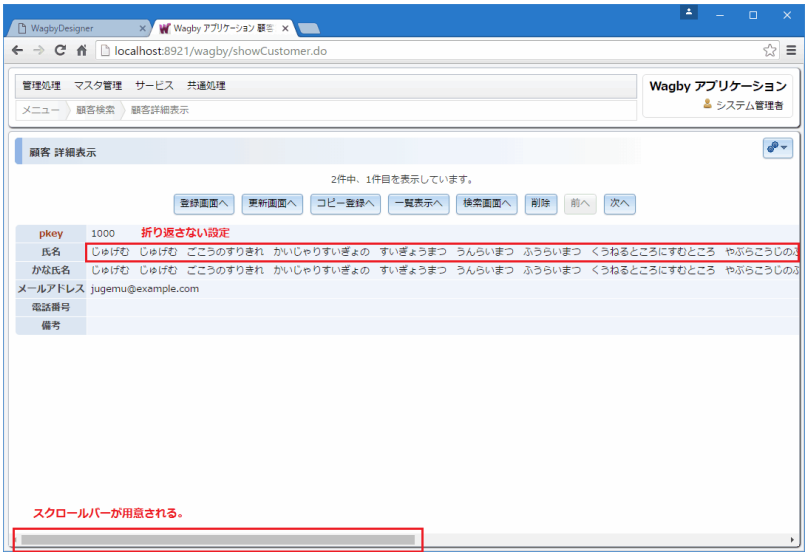


図 値を折り返さない

定義方法

各項目の詳細定義で「出力制御>詳細画面>スタイル」を開きます。「ラベルの表示を折り返さない」「値の表示を折り返さない」をそれぞれ指定します。

標準ではいずれも「ラベルの表示を折り返さない」が有効で、「値の表示を折り返さない」が無効(=折り返す)です。

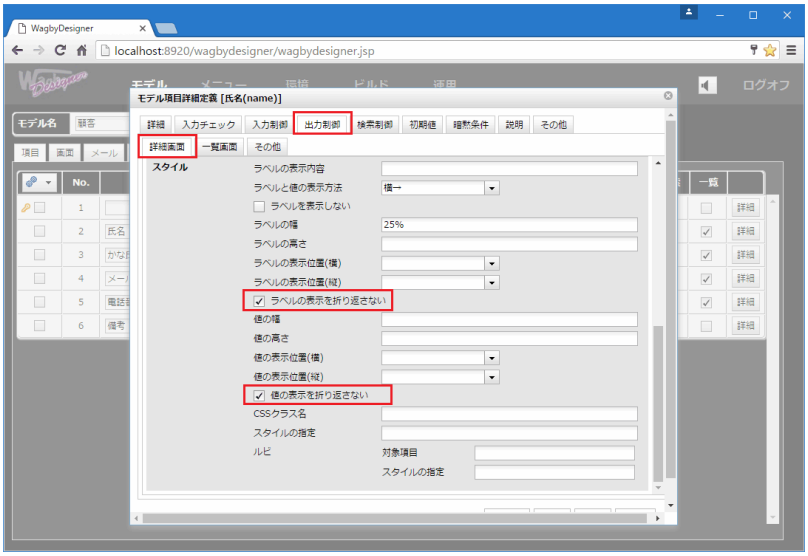


図 折り返しの設定（詳細画面）

一覧画面

ラベル

図はラベルの折り返しを行う場合です。標準は「（ラベルを）折り返す」となっています。

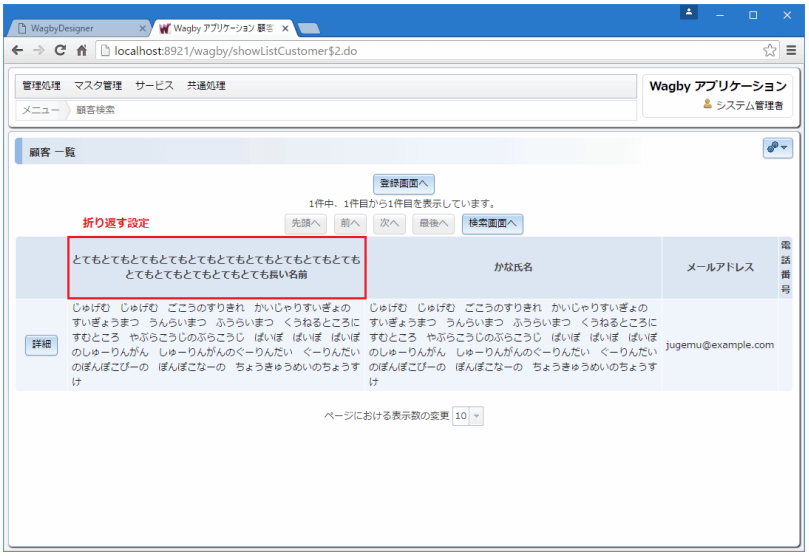


図 ラベルを折り返す（標準）

図はラベルの折り返しを行なわない場合です。ラベル文字が長い場合、ラベル幅設定より折り返さない設定が優先されます。

図 ラベルを折り返さない

値

図は値の折り返しを行った場合です。標準は「（値を）折り返す」となっています。

図 値を折り返す（標準）

図は値の折り返しを行わない場合です。出力文字が長いとき、ブラウザの横スクロールバーが表示されます。

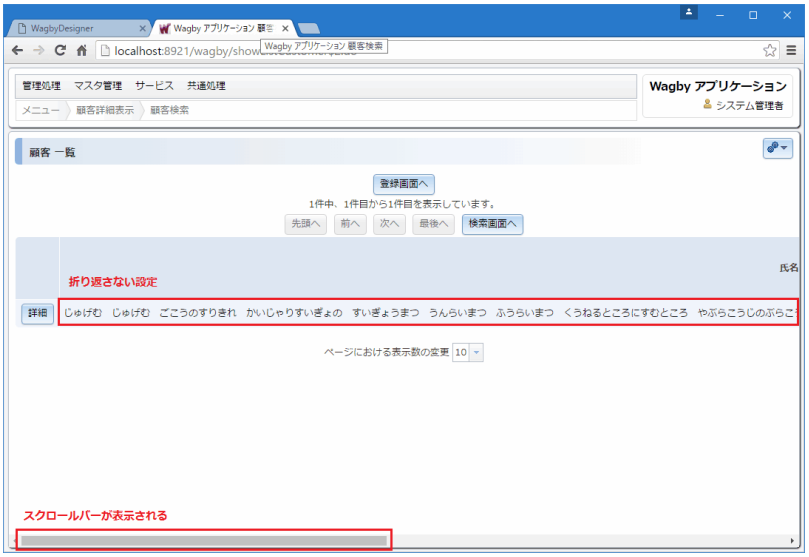


図 値を折り返さない

定義方法

各項目の詳細定義で「出力制御>一覧画面>スタイル」を開きます。「ラベルの表示を折り返さない」「値の表示を折り返さない」をそれぞれ指定します。

標準ではいずれも「ラベルの表示を折り返さない」「値の表示を折り返さない」いずれも無効(=折り返す)です。

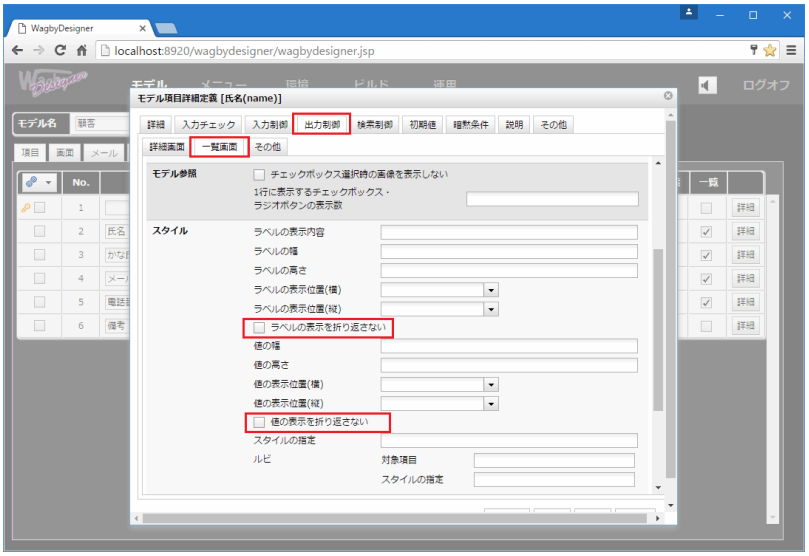


図 折り返しの設定（一覧画面）

折り返されない場合の対応方法

　折り返し処理は単語のスペース区切り部での改行を行うようになっています。そのためスペースがない長い文字列を途中で折り返すことはありません。

　このようなデータが登録される場合は、対象項目の詳細画面および一覧画面の「スタイルの指定」設定に次のCSSを指定してください。

```
word-break: break-all
```

CSSクラス名・スタイルの指定

CSSクラス名

　開発者によるカスタマイズで独自の CSS 設定を追加した際、そのクラス名を設定することで（値部分に）反映させるための設定欄です。 詳細は レイアウト 〉CSSのカスタマイズ をお読みください。

ラベルのスタイル

　「顧客名称」項目の色を緑にした例を示します。

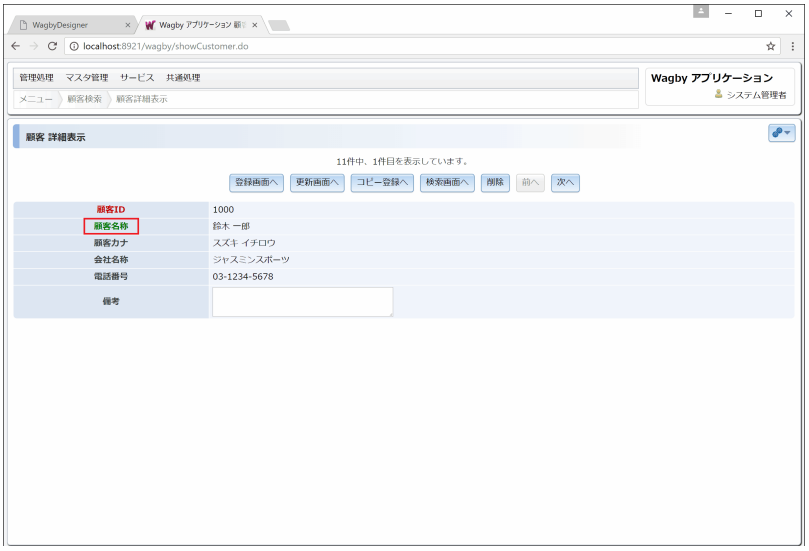


図 例

設定

「氏名」項目の詳細定義を開きます。「出力制御>詳細画面>スタイルの指定」を次のように指定します。

```
color:green
```

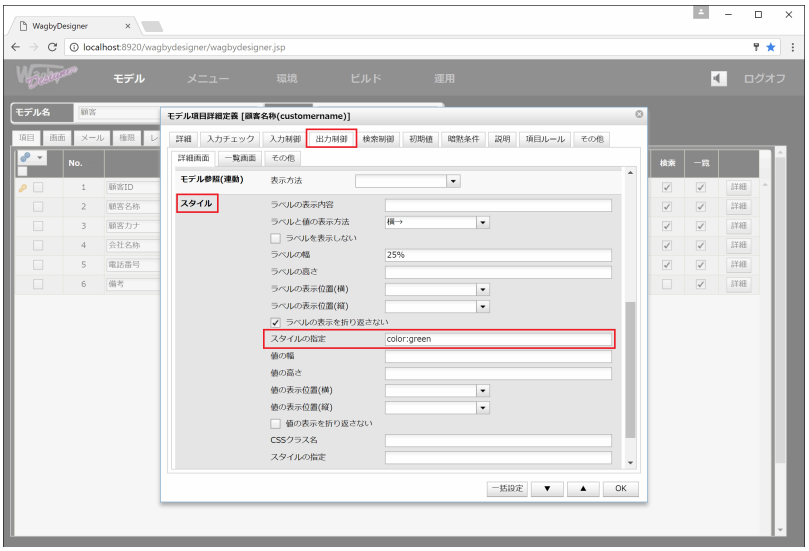


図 スタイルの設定

この設定は詳細表示画面で有効になります。一覧表示画面、検索画面もそれぞれ設定できます。

CSS の詳細については各種 Web Site における解説や、参考書をお読みください。

値のスタイル

「氏名」項目のフォントサイズを 14 ポイントとし、かつ、色を赤にした例を示します。

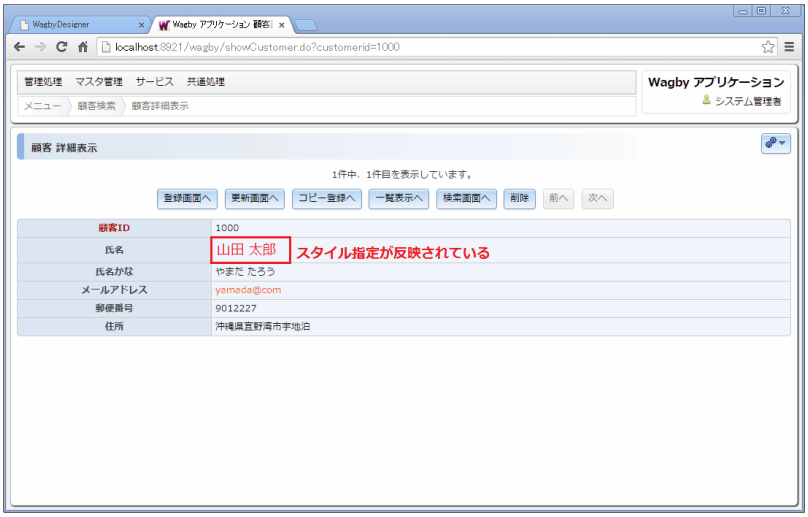


図 例

設定

「氏名」項目の詳細定義を開きます。「出力制御>詳細画面>スタイルの指定」を次のように指定します。

```
font-size:14pt; color:#F00;
```

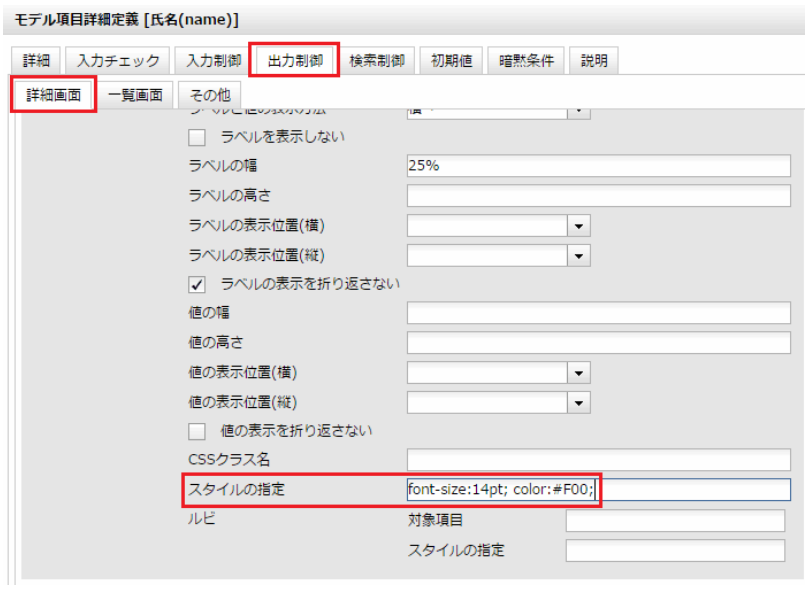


図 スタイルの設定

この設定は詳細表示画面で有効になります。

CSS の詳細については各種 Web Site における解説や、参考書をお読みください。

例 ラジオボタン・チェックボックスの高さを指定する

ラジオボタンやチェックボックスを格納する div 要素へのスタイルを指定することができます。図では、高さを150pxとし、スクロールバーを用意する設定を行なった例です。

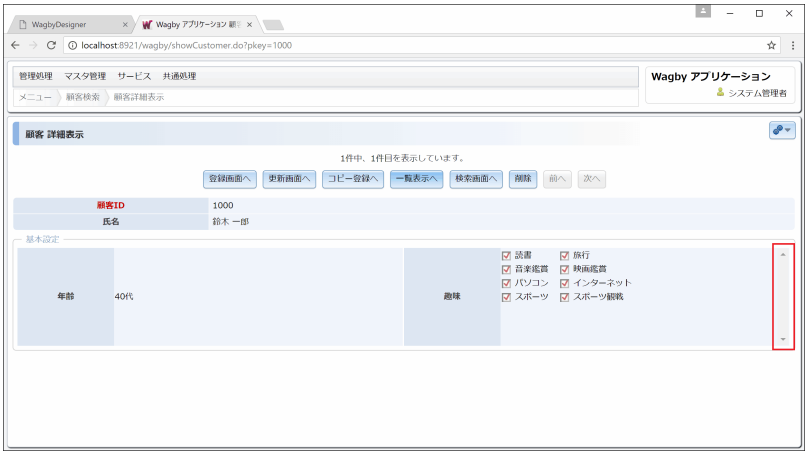


図 div要素へのスタイル指定の例

定義方法

モデル項目の「出力制御 ＞ 詳細画面の設定 ＞ スタイルの設定」に次のように div{..} で囲って記述します。

```
div{height:150px;overflow-y:scroll;}
```

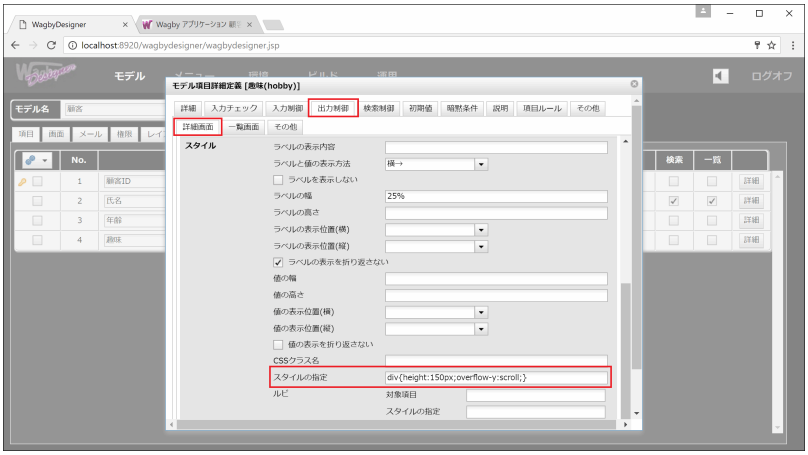


図 スタイルの指定

他の設定との併用

div {...} 設定のあとに、続けて色の指定など、他のスタイルを指定することができます。div {...} で囲まれていないスタイルは、チェックボックス本体に適用されます。

(201)

次の例は文字色を赤とした例です。

```
div{height:150px;overflow-y:scroll;} color:red
```

(201) 正確には、チェックボックスを囲う span 要素にスタイルが適用されます。

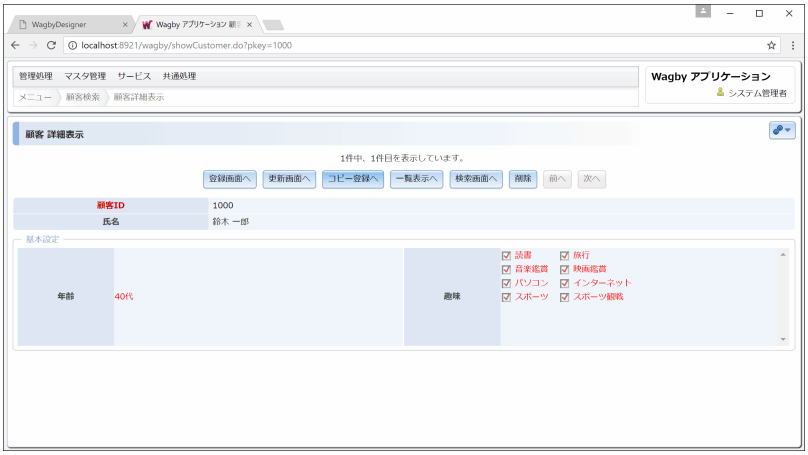


図 div要素とspan要素の両方を設定する

登録・更新画面への設定

"入力制御 > 入力欄のスタイル指定"をお読みください。

一覧表示への設定

文字色、背景色の指定

一覧表示のラベル、値部に文字色と背景色などのスタイルを指定することができます。

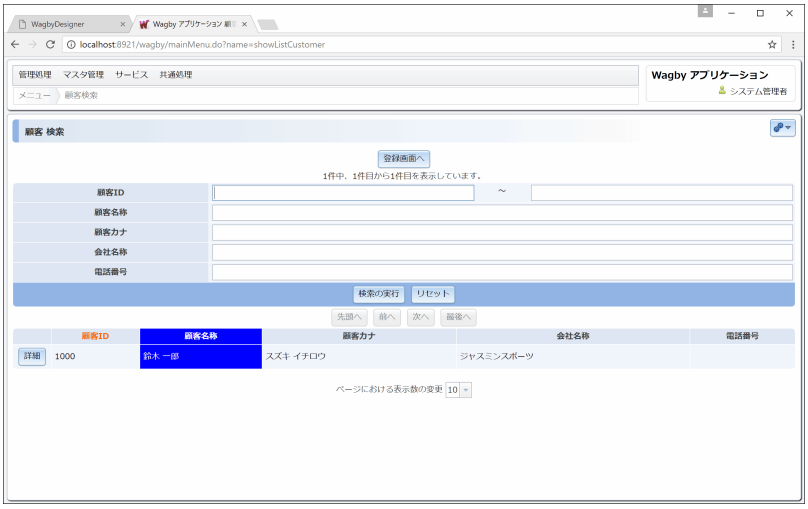


図 一覧表示のラベル、値部に文字色と背景色を指定した

「出力制御 > 一覧画面 > スタイル」に CSS を指定します。この例では次の指定を行なっています。

```
color:white;background:blue
```

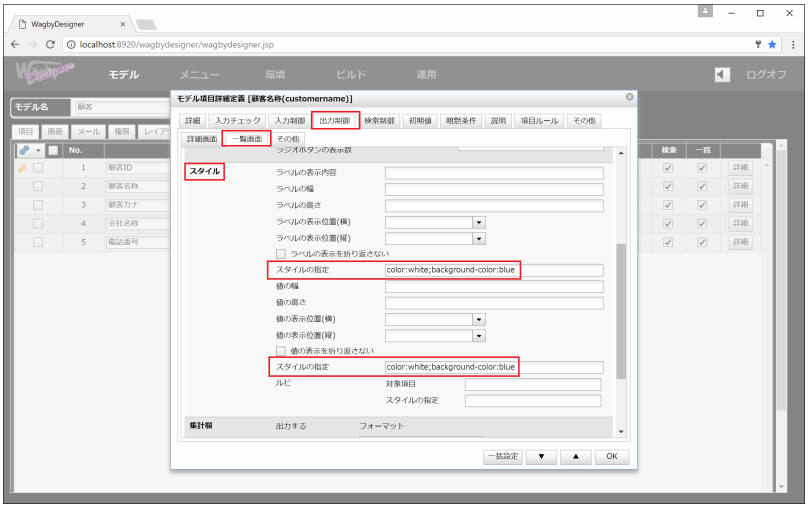


図 出力制御>一覧画面>スタイル

一覧表示項目の折り返し幅を指定する

Wagbyは標準で「空白文字があればそこで改行する」ようになっています。そのため、表示する列の内容に空白がない場合、折り返しが効きません。

この場合は「出力制御 > 一覧画面 >スタイル」の「スタイルの指定」設定に

```
max-width:50px
```

などと設定してください。単語の区切り（空白文字）がないデータについても指定幅で折り返されるようになります。

条件によってスタイルを変更する

式と関数

値部分のスタイル指定には、IF 条件式を用いることができます。次の例は tyuui 項目の値が 0 の場合にスタイルを適用するというものです。

```
IF(${tyuui}==0,"font-size:14pt; color:#F00","")
```

モデル項目名（英語）は "${" と "}" で囲みます。また指定する CSS はダブルクォーテーション ("") で囲みます。

モデル参照項目への適用

モデル参照項目の場合は「ID 部」を使うか「内容部」を使うかで、記述式が異なります。 詳細は"計算式 > モデル参照"をお読み下さい。

次の例は、productname 項目（モデル参照）の「内容部」で判定しています。

```
IF(EXACT(${productname#content},"MSX-001"),"font-size:14pt; color:#F00","")
```

文字列内容の一致判定は EXACT 関数を用います。

繰り返しコンテナ項目への適用

式中に繰り返しコンテナ項目を使うこともできます。

```
IF(EXACT(${report/pcon#content},"SR003"),"font-size:14pt; color:#F00","")
```

仕様・制約

- 一覧表示画面のラベル部分、値部分の CSS はグリッド形式表示では利用できません。
- 条件によってスタイルを変更する設定は、値部分のみで利用できます。ラベル部分の設定では式を含めないようにしてください。

改行

改行指定が必要な例

「会社名」が長いという事例を図、図に示します。図のように日本語文字の場合は Web ブラウザが適切に改行します。

図 長い会社名（日本語）

しかし英数時で構成された場合は、改行されず、横スクロールバーが表示されます。

図 長い会社名（英語）

これは Wagby の標準では、英数字の場合は空白を改行の区切り文字として認識するようになっているためです。 図のように空白が存在しない場合、適切な改行箇所が見つかりません。

定義方法

出力時のスタイルを指定することで、改行位置を調整します。

「会社名」項目の詳細定義を開きます。「出力制御>詳細画面>スタイル」を次のように指定します。

```
word-break:break-all
```

詳細画面

図 詳細画面のスタイル設定

一覧画面

図 一覧画面のスタイル設定

設定後の画面

図および図の設定後は、詳細画面、一覧画面とも改行されるようになります。

図 詳細画面の改行例

図 一覧画面の改行例

スタイルシート

標準では、Wagbyが提供するスタイルシートは次のようになっています。

ブラウザ	スタイル
IE,Firefox,Safari	word-break: normal;word-wrap: break-word;
Chrome	word-break: break-word;word-wrap: break-word;

上で紹介したスタイルシートの書き換えは、標準設定では折り返しがうまくいかない、という場合に行ってください。

ルビ

詳細画面

「氏名かな」の値をルビとして表示した例を図に示します。

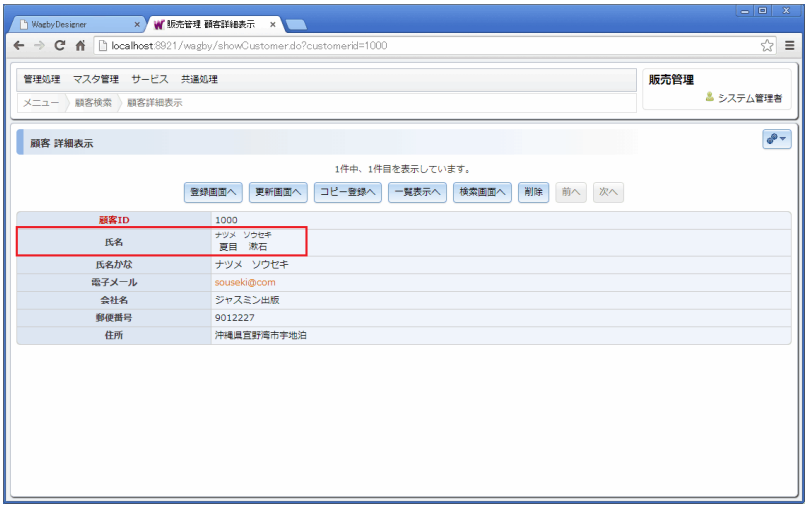


図 ルビの利用

定義方法

「氏名」項目の詳細定義を開きます。「出力制御>詳細画面>ルビ」を設定します。

図 ルビの設定

対象項目

ルビとして用いる項目ID（英語表記）を指定します。

スタイルの指定

ルビ表示部分に適用するスタイルを指定します。フォントサイズを小さめに設定するとよいでしょう。例を示します。

```
font-size:70%
```

詳細画面 - 隠し項目指定との併用

図で、ルビとして用いた項目（ここでは「氏名かな」）は二重に表示されています。これを防ぐため、「氏名かな」項目を隠し項目とする方法を紹介します。

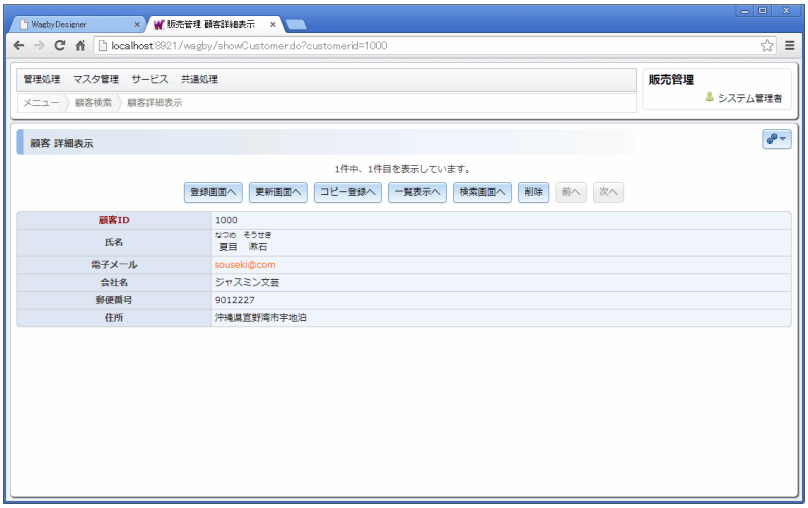


図 氏名かな項目を隠し項目とする

定義方法

「氏名かな」項目の詳細定義を開きます。「出力制御>詳細画面>隠し項目にする」を有効にします。

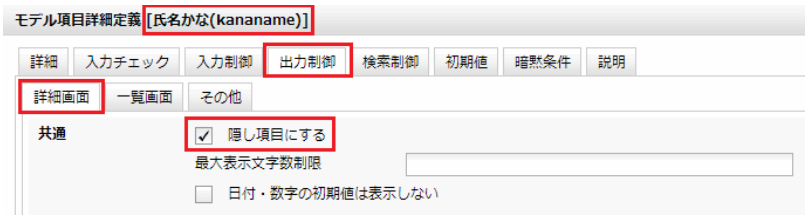


図 隠し項目の設定

一覧画面

一覧表示時も同様にルビとして扱ってみます。図に例を示します。

図 ルビを有効にした一覧画面

ルビの指定

「氏名」項目について、一覧画面について同様にルビの設定を行います。

図 一覧画面の設定（氏名項目）

隠し項目の設定

詳細画面と同様に、氏名かな項目にも隠し設定を行います。

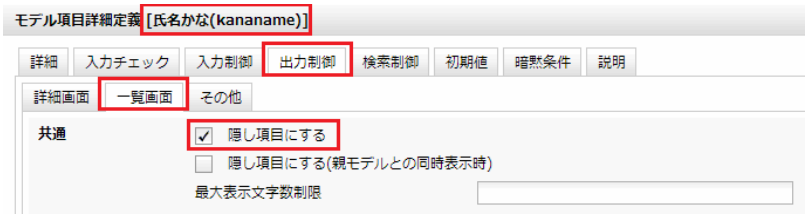


図 一覧画面の設定（氏名かな項目）

ヘルプメッセージ

ヘルプボタン

画面例

項目名（ラベル部）の右横にボタンを配置することができます。図は "Help!" というボタンを置いた例です。

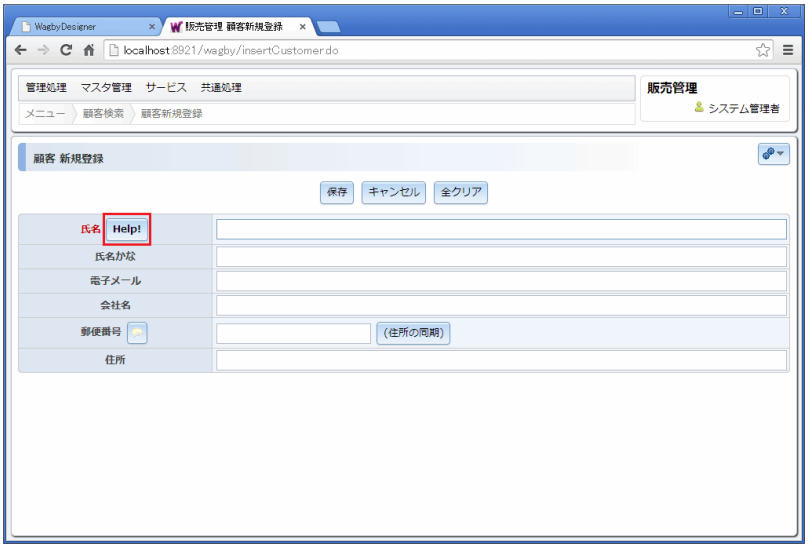


図 Help!ボタンを配置する

ボタンの上にマウスをのせるとメッセージが表示されます。

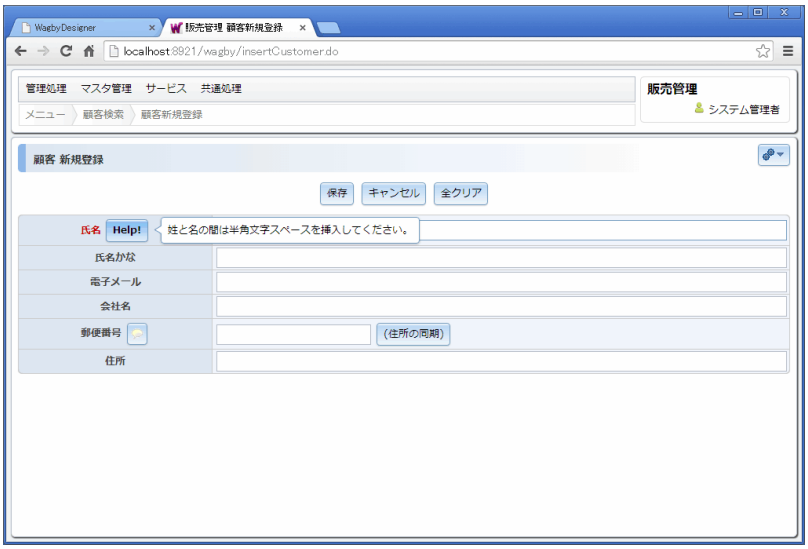


図 ボタンの上にマウスをのせる

定義方法

モデル項目詳細定義の「説明」タブを開きます。登録画面・更新画面・検索画面・詳細画面毎に設定できます。ここでは登録画面を指定します。

「ヘルプボタン」の欄にボタンのラベル（表示文字）を、「内容」の欄にメッセージをそれぞれ記載します。

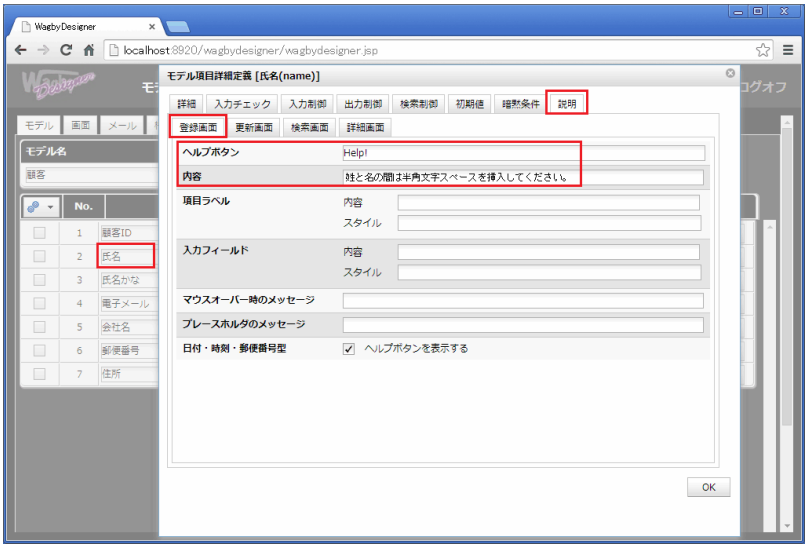


図 ヘルプボタンの設定

項目ラベルへのメッセージ

画面例

項目ラベル部にメッセージを記述することもできます。「項目名」の下に表示されます。

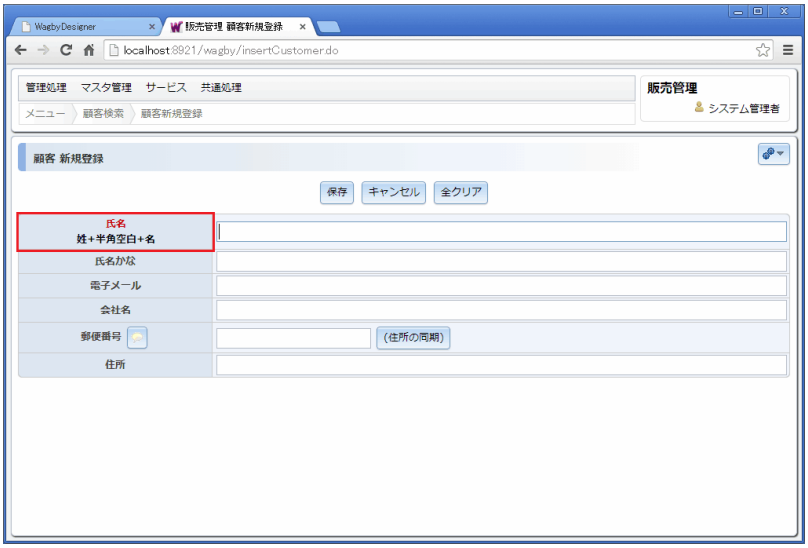


図 項目名の下にメッセージが表示される

定義方法

説明タブ内の「項目ラベル>内容」欄にメッセージを記載します。

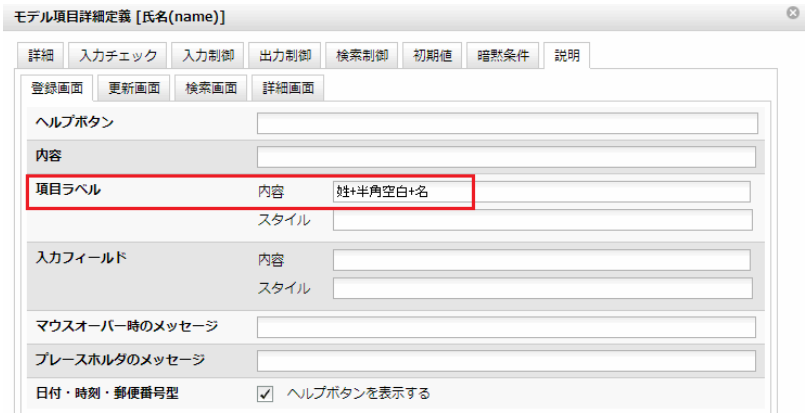


図 項目ラベルの内容

画面例 - スタイル指定

スタイル（文字の大きさや色など）を指定することもできます。

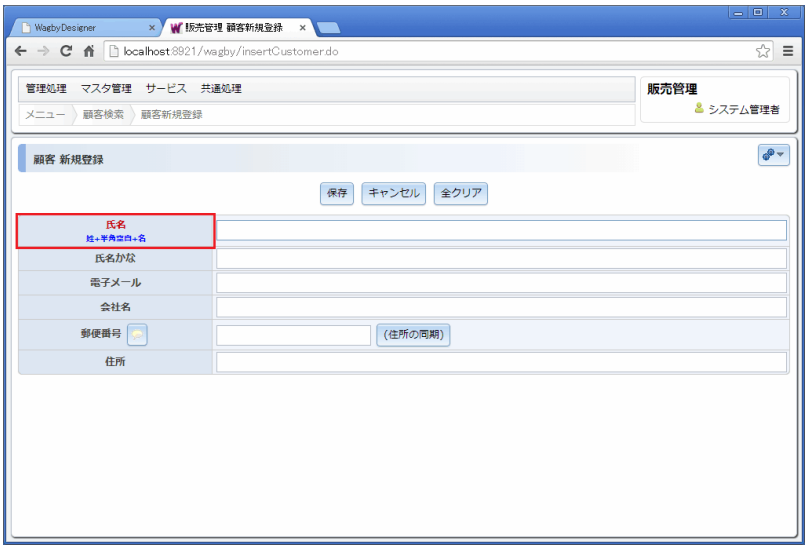


図 文字サイズを小さくし、文字色を青に設定した

定義方法

説明タブ内の「項目ラベル>スタイル」欄にCSSを記載します。具体的なCSS文法の説明は割愛します。（CSSに関する他の文献を参照してください。）

図 スタイルの設定

入力フィールドへのメッセージ

画面例

図は入力フィールドにヘルプメッセージを指定した例です。入力フィールドの幅が100%のとき、メッセージは入力フィールドの下に表示されます。

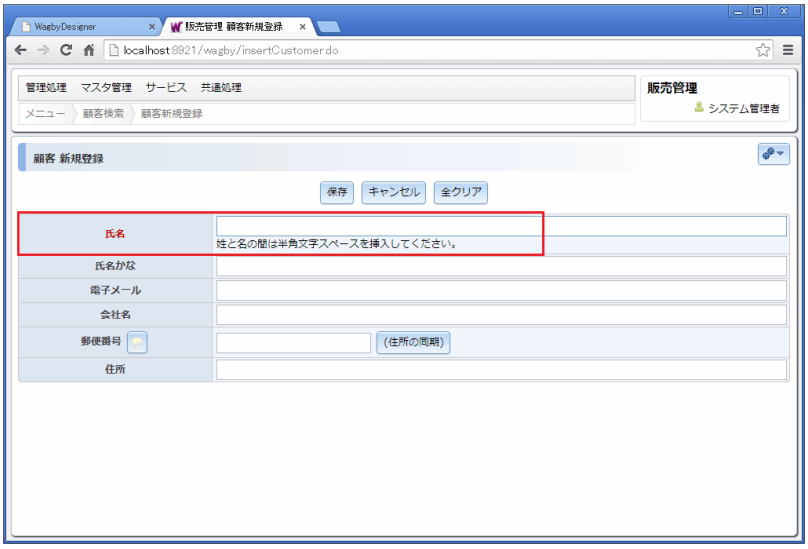


図 入力フィールドの下にメッセージが表示される

定義方法

説明タブ内の「入力フィールド>内容」欄にメッセージを記載します。

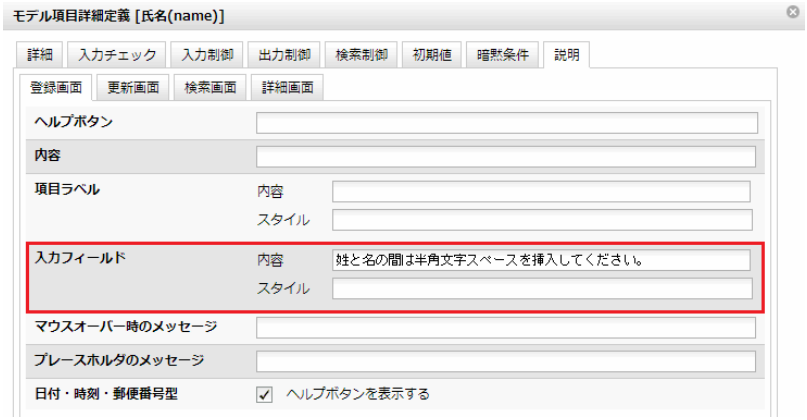


図 入力フィールドの内容

画面例 - スタイル指定

スタイル（文字の大きさや色など）を指定することもできます。

図 文字色を青に設定した

定義方法

説明タブ内の「入力フィールド>スタイル」欄にCSSを記載します。具体的なCSS文法の説明は割愛します。（CSSに関する他の文献を参照してください。）

図 スタイルの設定

マウスオーバー時のメッセージ（ツールチップ）

画面例

マウスを入力欄にのせる（オーバーラップ）と、メッセージが表示されます。これをツールチップと呼びます。

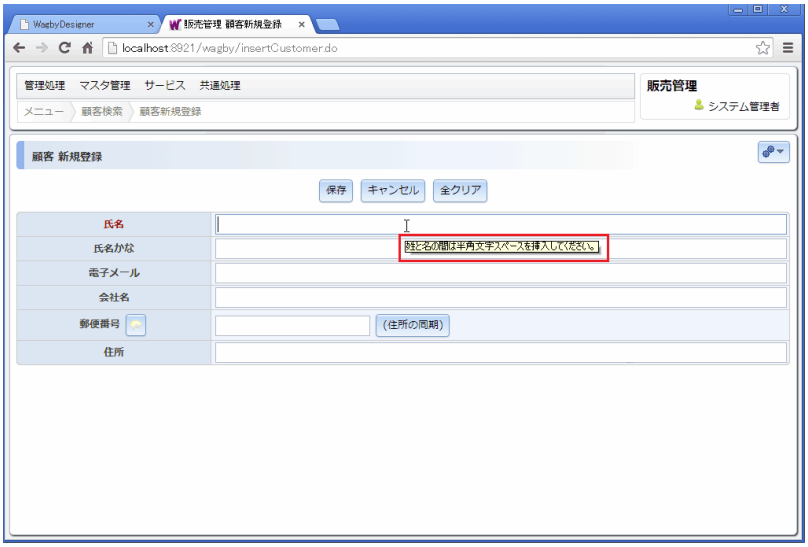


図 ツールチップ

定義方法

説明タブ内の「マウスオーバー時のメッセージ」欄にメッセージを記載します。

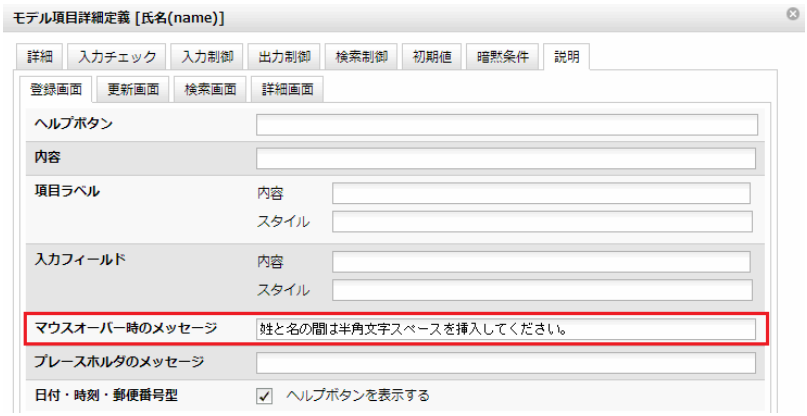


図 マウスオーバー時のメッセージを指定する

プレースホルダのメッセージ

画面例

プレースホルダとは、入力欄に値が入力されていない時に表示される文字です。

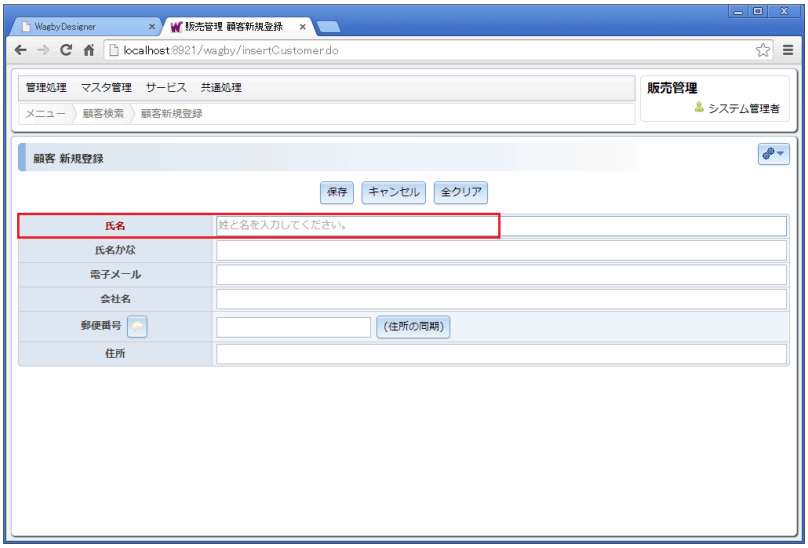


図 プレースホルダ

プレースホルダは「初期値」ではありません。この状態で保存しても、データベースにはプレースホルダの文字は保存されません。（より正確には、何も保存されません。）

この入力欄に一文字でも入力を行なうと、プレースホルダの文字は消えます。

定義方法

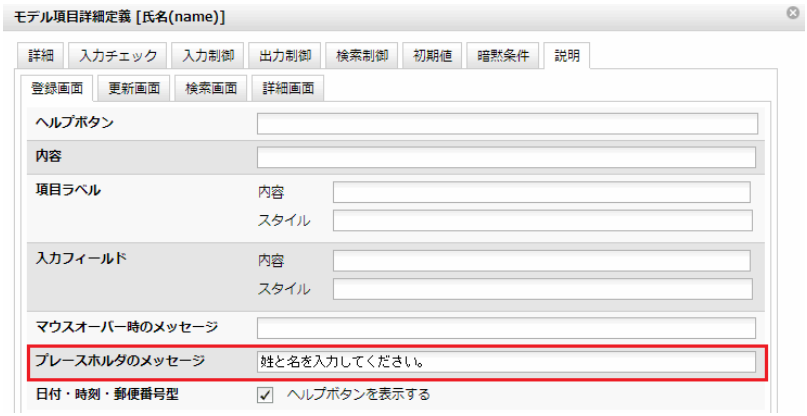


図 プレースホルダのメッセージを指定する

プレースホルダの表示ルールについての補足

- 本機能は HTML の placeholder 属性を利用して実現しています。そのため本機能に未対応のWebブラウザではご利用になれません。
- プレースホルダが消えるタイミングは、Web ブラウザによって異なります。
- テキストエリアには設定できません。

改行の扱い（8.3.0）

「ヘルプボタンの内容」「項目ラベルの内容」「入力フィールドの内容」「マウスオーバー時のメッセージ」は改行を行うことができます。

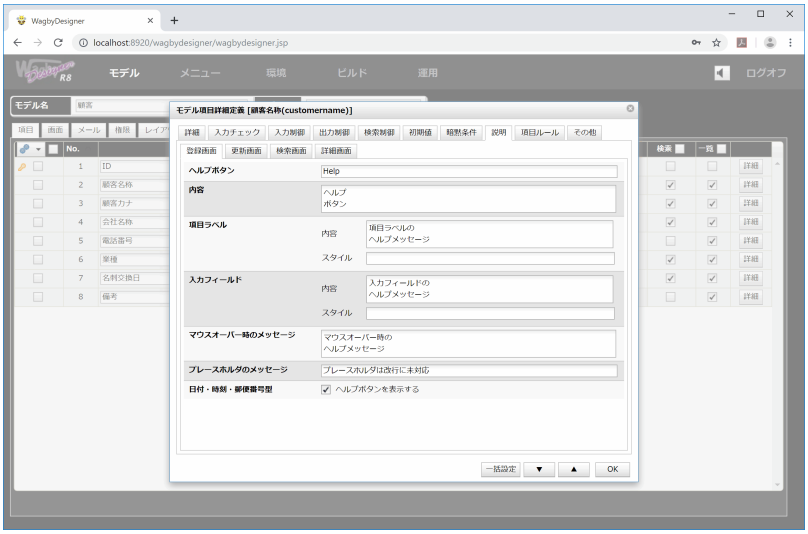


図 改行を含むメッセージの設定

アプリケーションでは改行された箇所が有効になります。

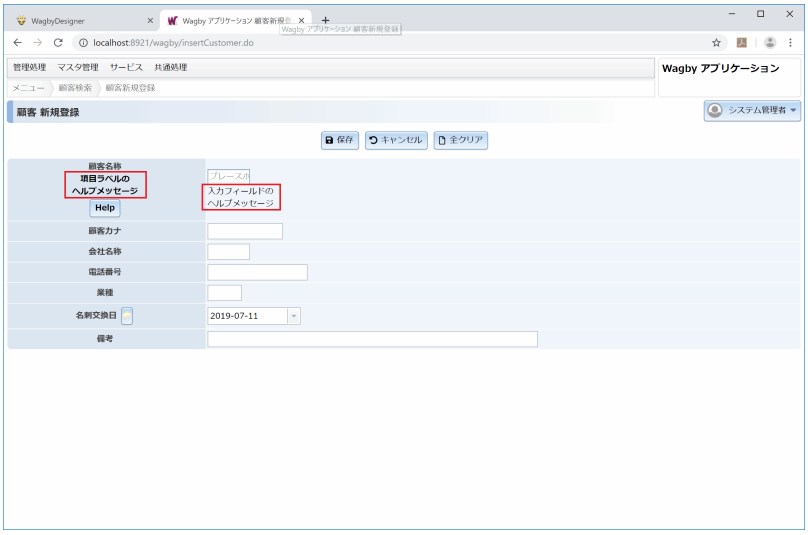


図 改行を含むメッセージが表示される

バーゲージ

実行例

図はバーゲージを使っていない例です。数値はそのまま表示されています。

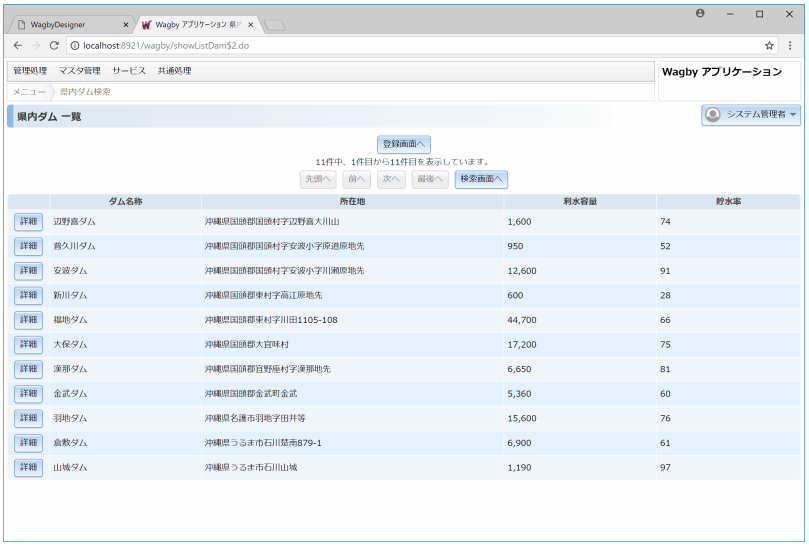


図 バーゲージを使っていない

図はバーゲージで表示した例です。数値を視覚的に把握することができます。

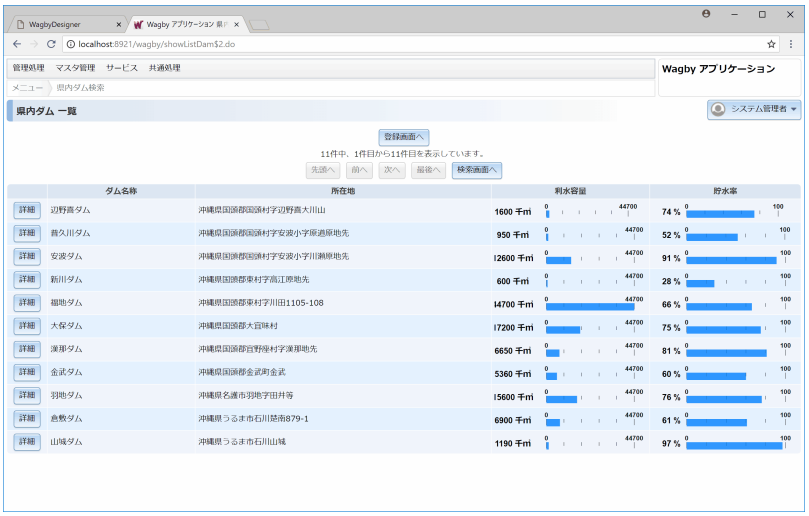


図 バーゲージを利用した

詳細画面でもバーゲージ表示を利用することができます。

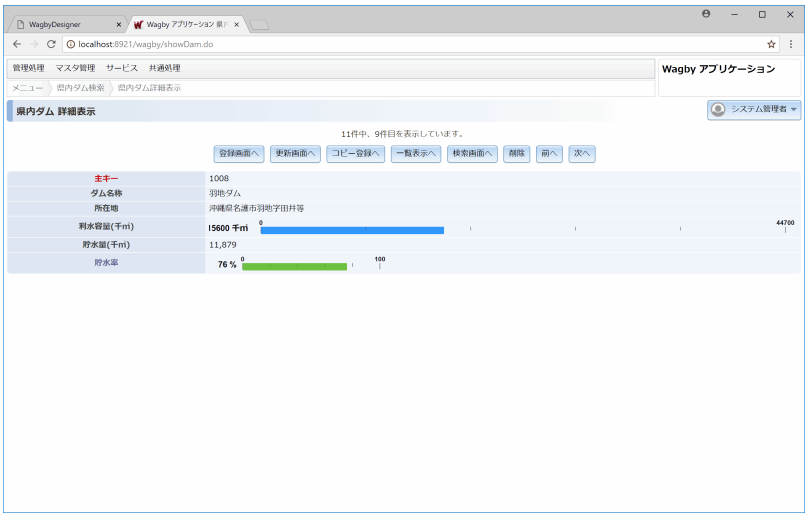


図 詳細画面の例

一覧表示（グリッド形式）でもバーゲージ表示を利用することができます。

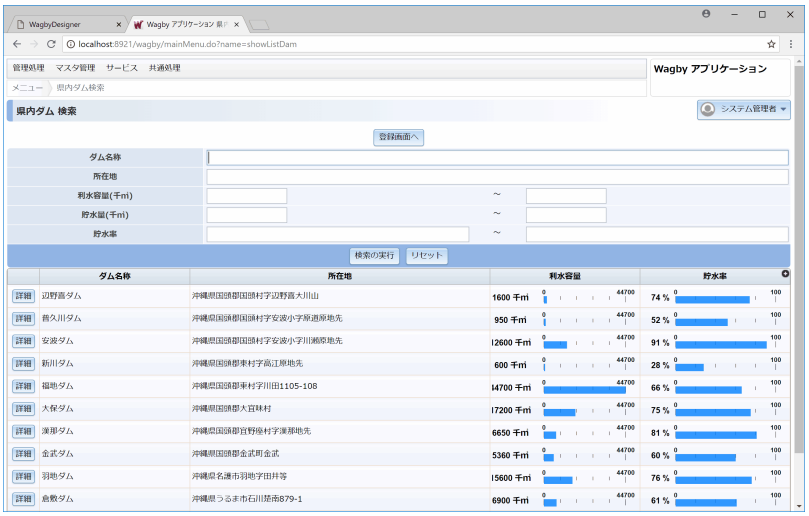


図 一覧表示グリッド形式

定義方法

モデル項目詳細定義ダイアログを開きます。出力制御　　>　　一覧表示　　>　　数値の設定を行います。 表示時の単位、および最大値、色を指定することができます。色はHTMLカラーコード（先頭がシャープ文字ではじまる16進数表記）です。

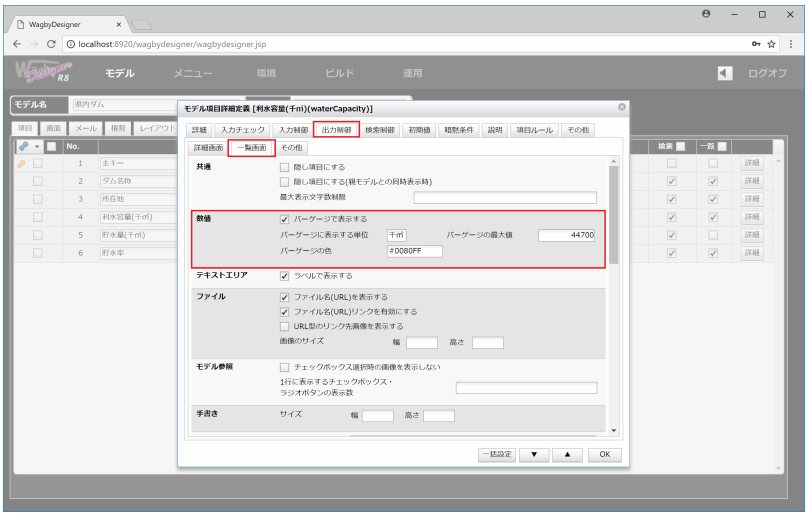


図 バーゲージで表示する設定(1)

最大値の省略時は 100 として扱われます。色の省略時は緑色になります。

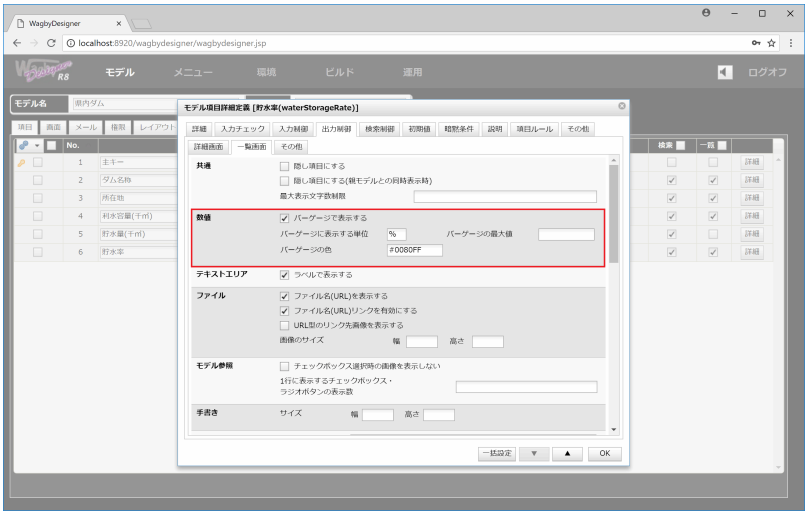


図 バーゲージで表示する設定(2)

図,6は一覧表示の設定を行っていますが、同様に詳細画面の設定を行うこともできます。

一覧表示の幅を調整する

バーゲージ表示項目の幅（ピクセル）を指定することで、見た目を調整することができます。

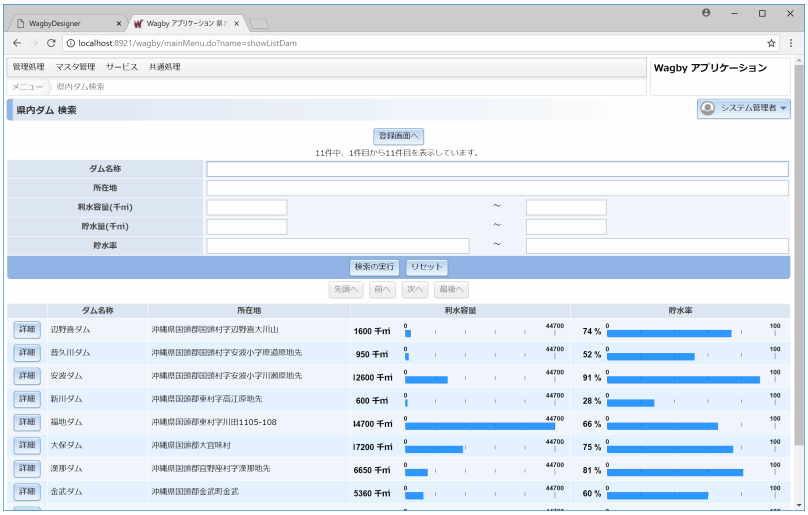


図 バーゲージ表示項目の幅を350ピクセルと指定した例

定義方法

出力制御 > 一覧画面 > スタイルの「スタイルの指定」に CSS を記述することができます。ここで width 指定を行います。次の例は 350ピクセルを指定したものです。

```
width:350px
```

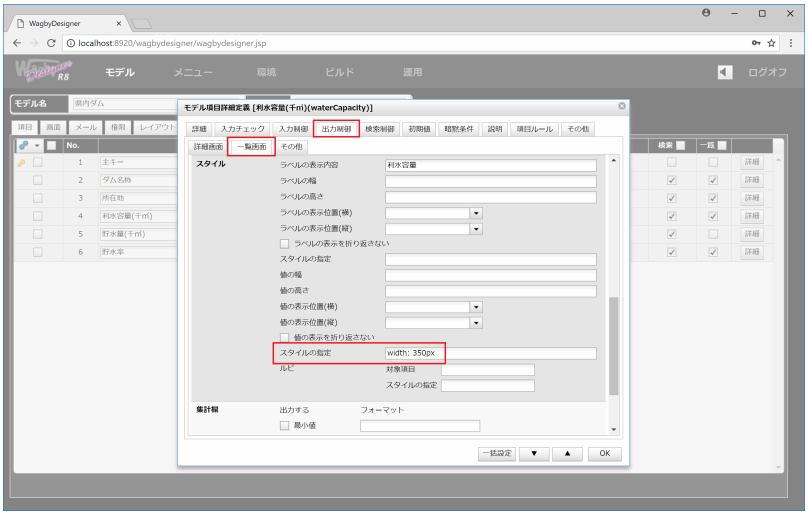


図 スタイルの設定

[カスタマイズ] JavaScriptを使って色を変更する

JavaScriptカスタマイズファイル MyShowList<モデルID>.js を用意することができます。

次のようなスクリプトを作成することができます。関数 getBarGaugeColor の第一引数にバーゲージの値が引き渡されます。

```
function getBarGaugeColor(value, barGaugeColor) {
  var color = barGaugeColor;
  if (value < 20) {
    color = "#C7243A";
  } else if (value < 40) {
    color = "#FFE920";
  }
  return color;
}
```

バーコード出力

実行例

詳細表示画面でバーコードを表示した例です。

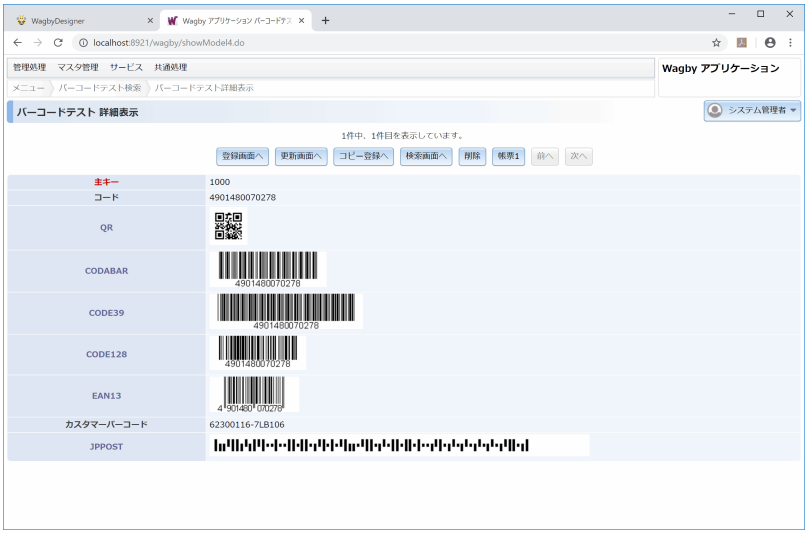


図 詳細画面にバーコードを表示する

一覧画面でもバーコードを表示することができます。

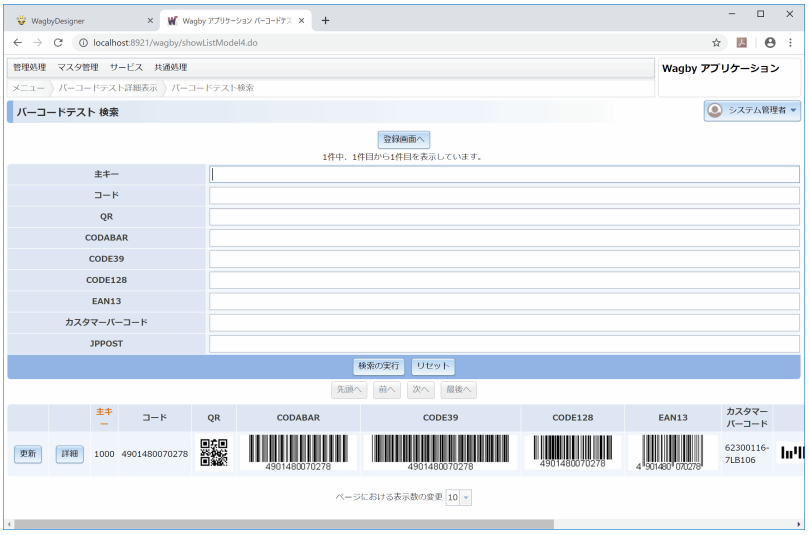


図 一覧画面にバーコードを表示する

定義方法

サンプルで用意したリポジトリは次の通りです。バーコードを出力する項目は「文字列型」として用意します。

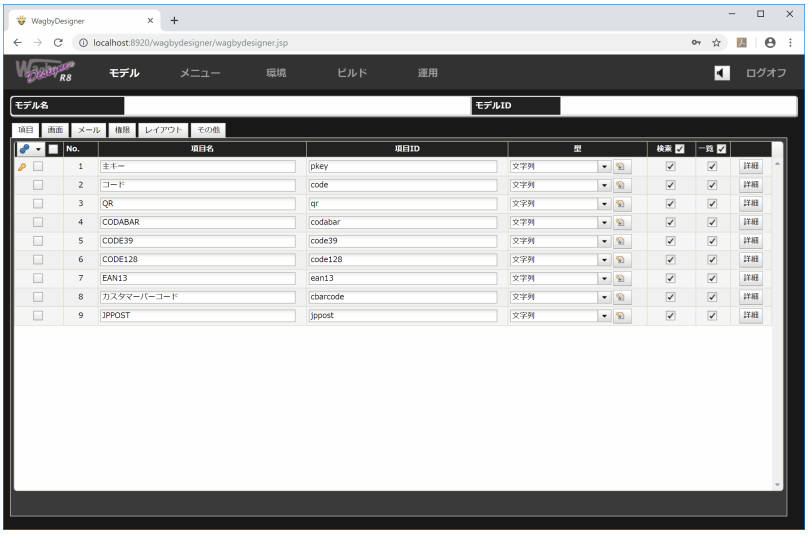


図 サンプルのリポジトリ

モデル項目詳細定義ダイアログを開きます。詳細 > 自動計算 > に BARCODE 関数を指定します。

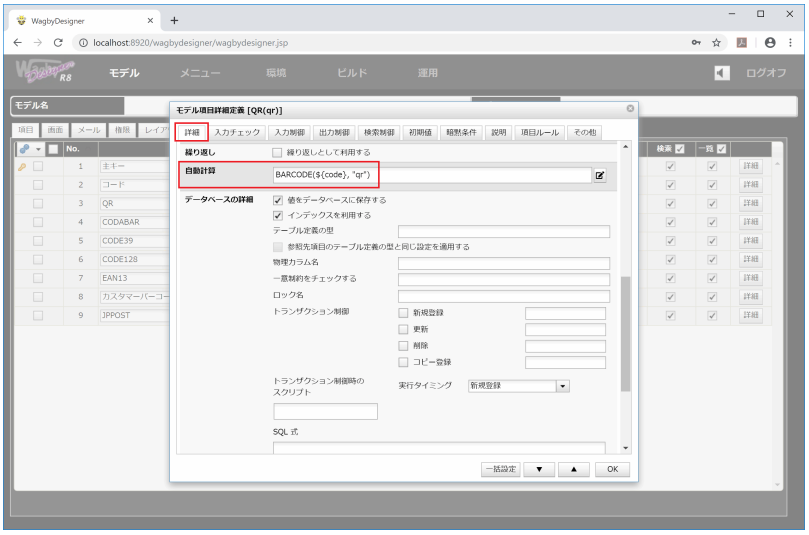


図 BARCODE関数を指定する

書式は次の通りです。第二引数は "codabar", "code39", "code128", "ean-13", "qr", "jppost" という方式を指定することができます。"jppost" は次節で説明します。

```
BARCODE(文字列, 出力方式)
```

カスタマーバーコードを出力する

BARCODE関数の第二引数である出力方式に "jppost" を指定することで、カスタマーバーコードを出力することができます。しかしこの場合、第一引数の入力文字列は、住所文字列ではなくカスタマーバーコード用のコードを指定する必要があります。

- https://www.post.japanpost.jp/zipcode/zipmanual/p17.html

この対応のために、別製品「住所正規化コンバータ」を使うことができます。住所文字列を、住所正規化コンバータを使って住所要素へ分解し、この要素の組み合わせでカスタマーバーコード用の入力文字列を作成する方法を説明します。

定義方法

サンプルとして、郵便番号と住所文字列をもつ顧客モデルを用意します。

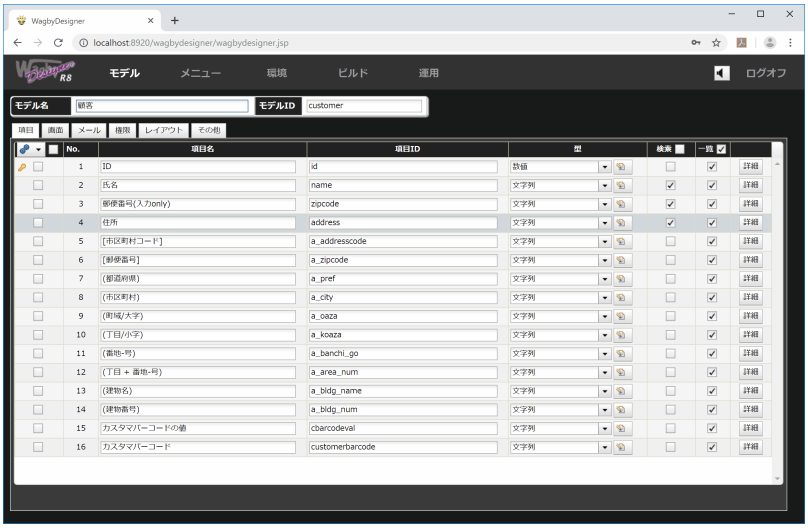


図 顧客モデル

このモデルは郵便番号から住所文字列を求める機能を有効にしています。番地以降の住所は手入力されるものとします。

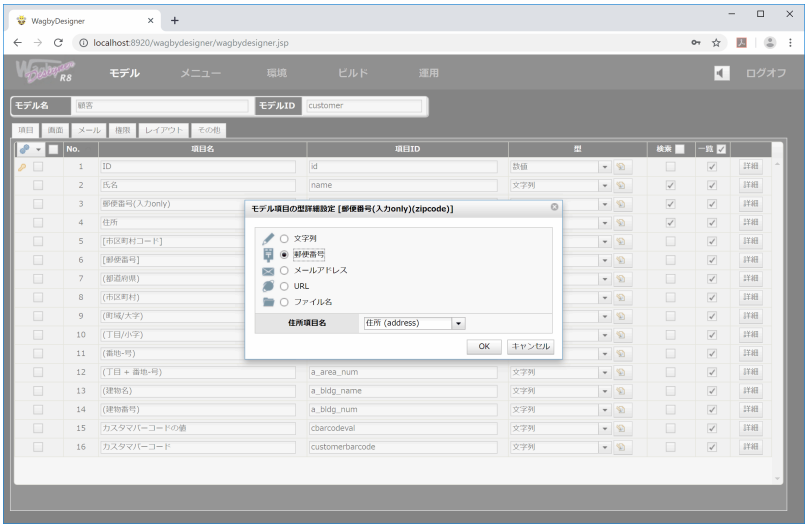


図 郵便番号から住所を求める

住所の各要素を保持する項目群を用意しています。ANORMPART 関数を使っています。

このうちカスタマーバーコードで利用するのは、"郵便番号","丁目、番地、号","建物番号" になります。

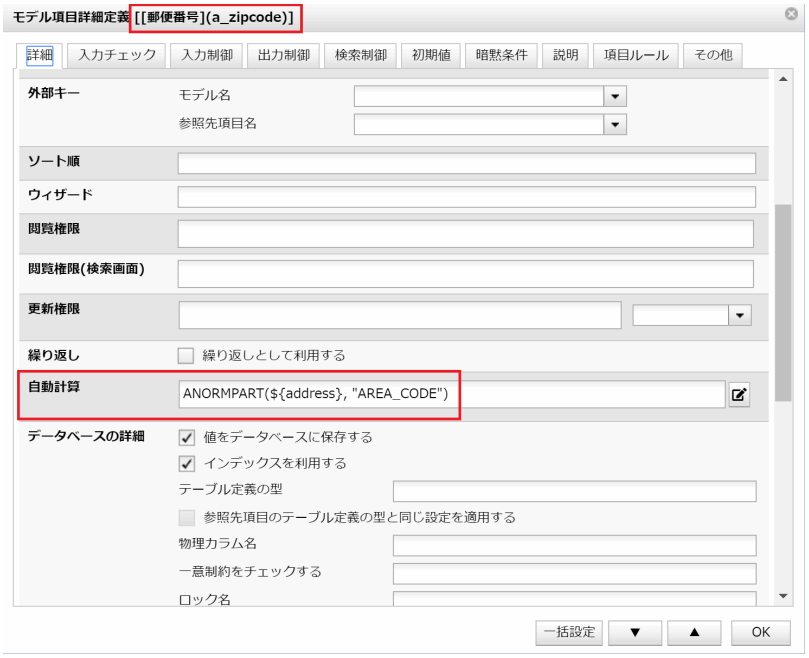


図 郵便番号部

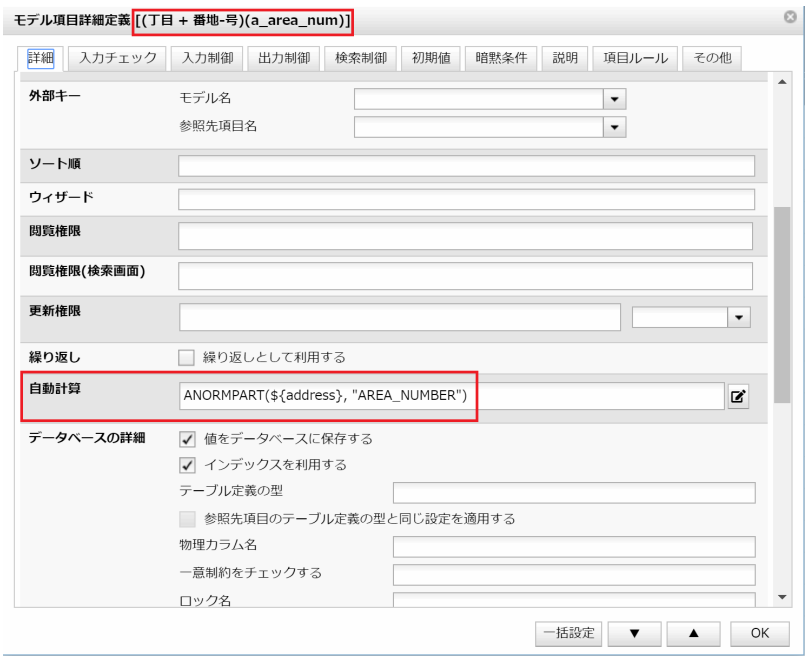


図 丁目、番地、号部

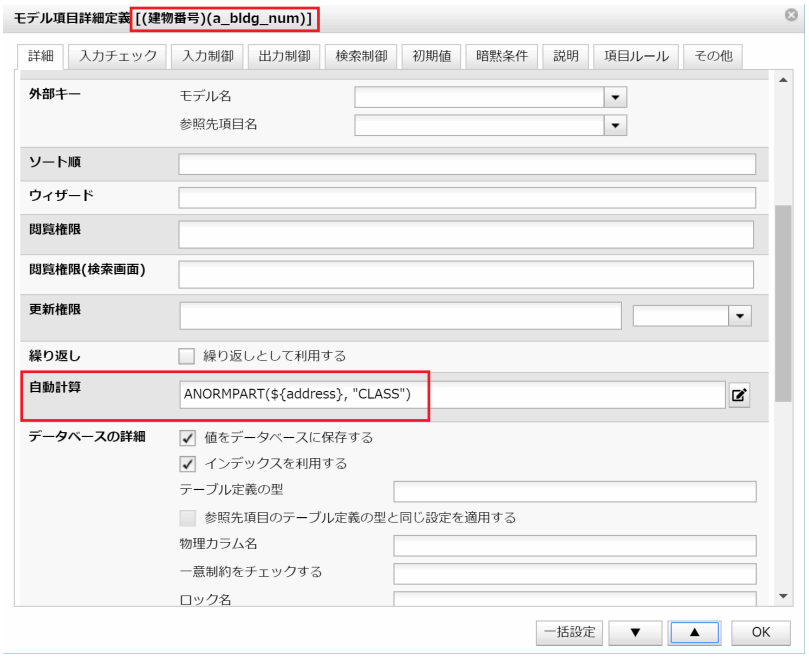


図 建物番号部

上記の要素を結合した文字列を用意します。これがカスタマーバーコードになります。

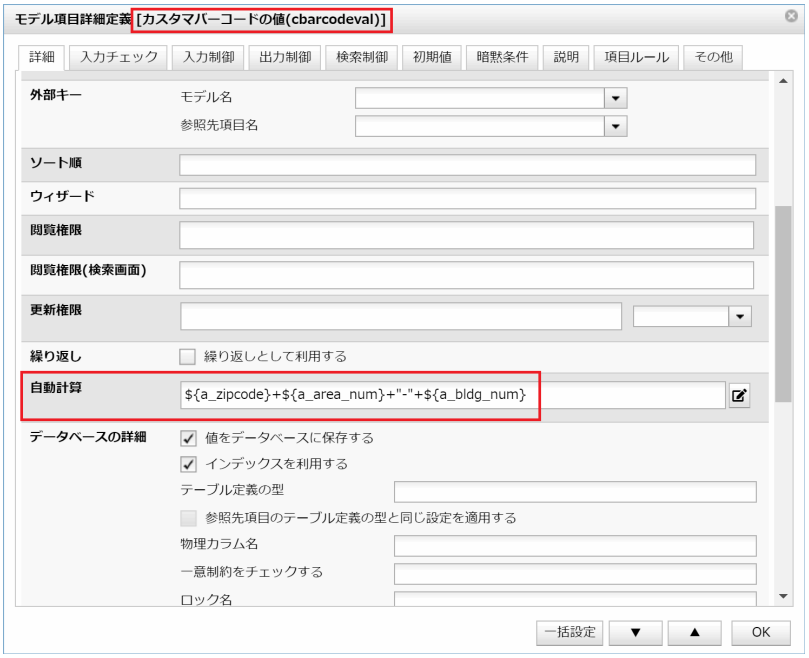


図 カスタマーバーコードを求める

BARCODE関数では、上記入力文字列を指定します。

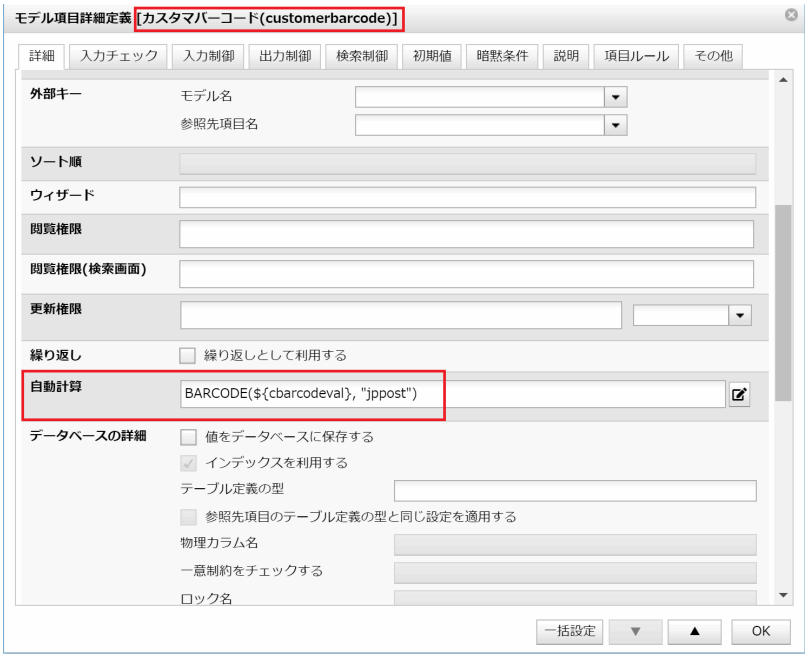


図 BARCODE関数

このANORMPART関数を使うために、住所正規化コンバータの設定を行います。URLは、住所正規化コンバータのREST APIサーバが稼働するURLを指定します。

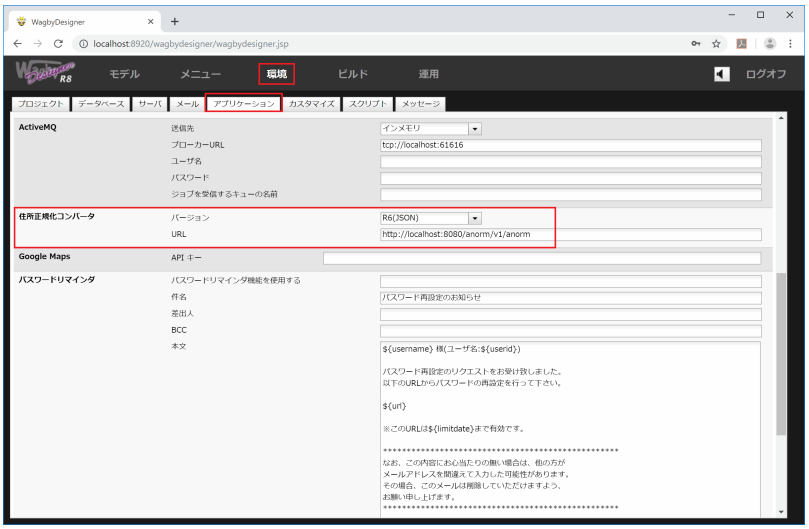


図 住所正規化コンバータの設定

動作

住所に"東京都千代田区霞が関1丁目3番2号 郵便プラザ503室"を入力した例を示します。

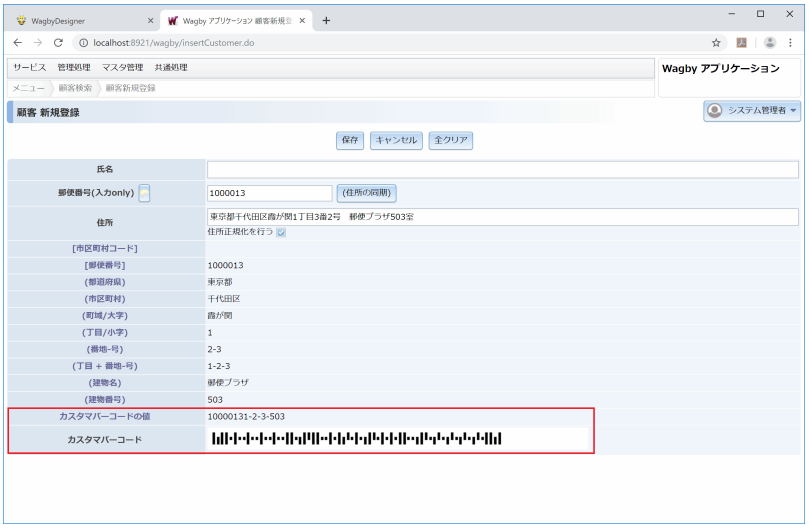


図 カスタマーバーコードの出力

帳票での利用

Excel帳票の場合

Excel帳票でもバーコード出力を行うことができます。

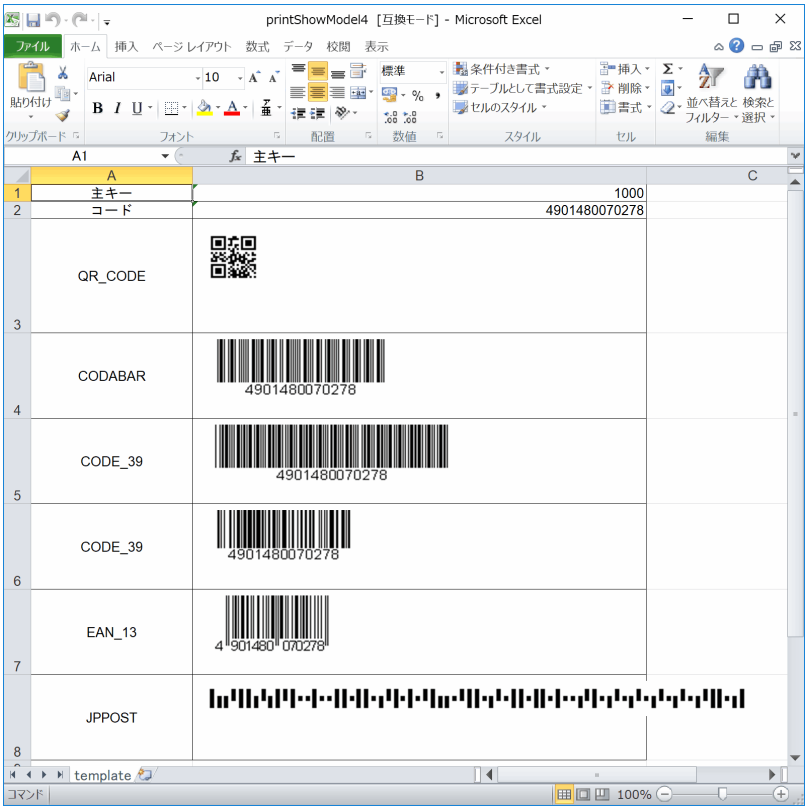


図 Excel帳票でバーコードを出力する

帳票テンプレートのプレースホルダは次のようになります。

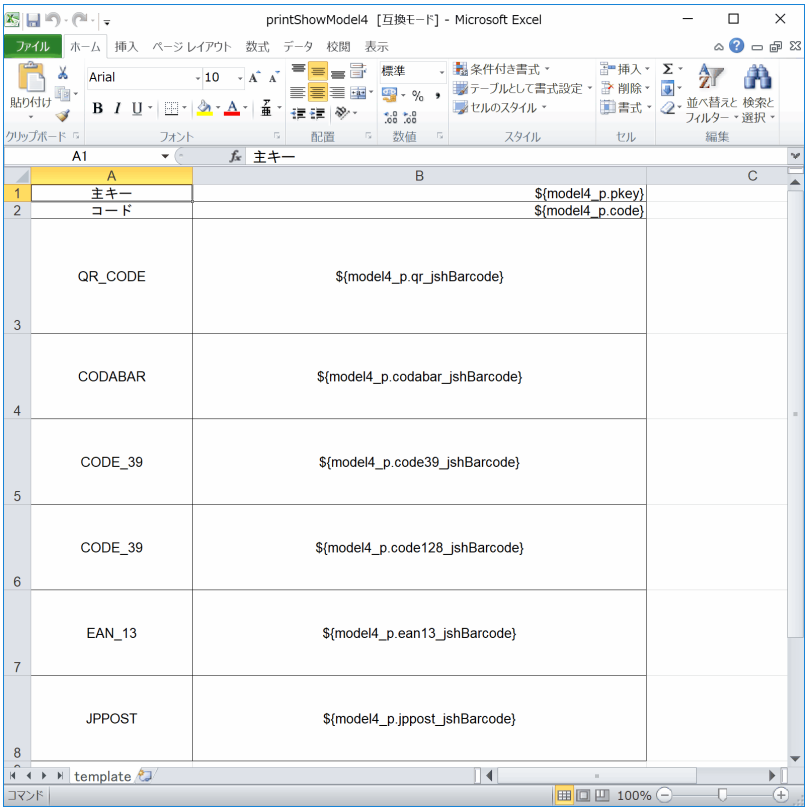


図 Excel帳票のプレースホルダの設定

```
${モデルID_p.項目ID_jshBarcode}
```

項目IDの末尾に "_jshBarcode" を付与してください。

JasperReportの場合

JasperReports の場合は画像オブジェクトの Expression 欄に次の設定を記述します。Base64エンコードされたバーコードが画像で表示されるようになります。

```
new
 ByteArrayInputStream(org.apache.commons.codec.binary.Base64.decodeBase64($F{qrcode}.getBytes()))
```

関連するページ

- 入力制御 > バーコード読み取り

ラベルの表示内容を式で制御する(スクリプト)

実行例

登録画面の "テストラベル" という表示は、項目ラベルのように見えますが、実際は値部分になっています。（元々のラベル部は非表示としています。）

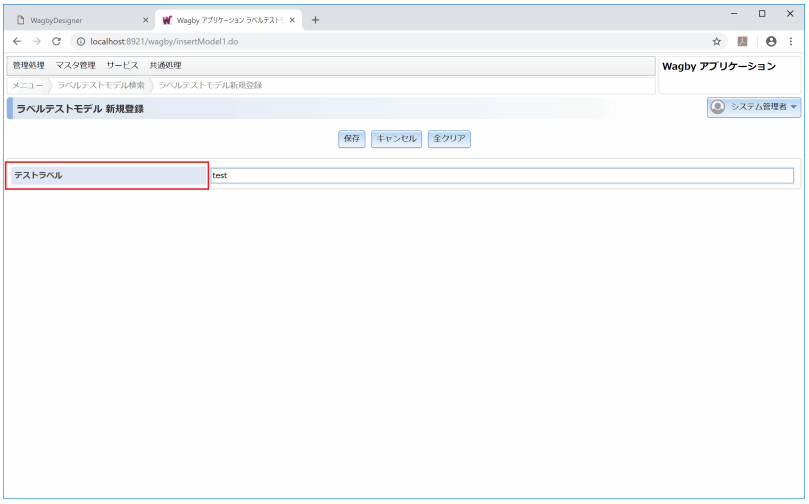


図 値部分をラベルのように表示させている

詳細画面も同じような設定を行なっています。

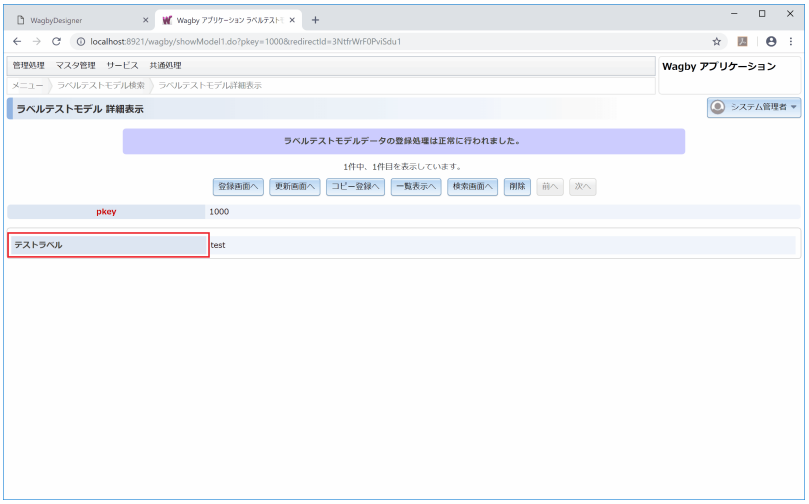


図 詳細画面

定義方法

サンプルのモデル定義を示します。ここで item1 は自身のラベルを非表示とし、内容部を （item2の） のラベルのように振舞わせます。item2 もラベルを非表示とします。

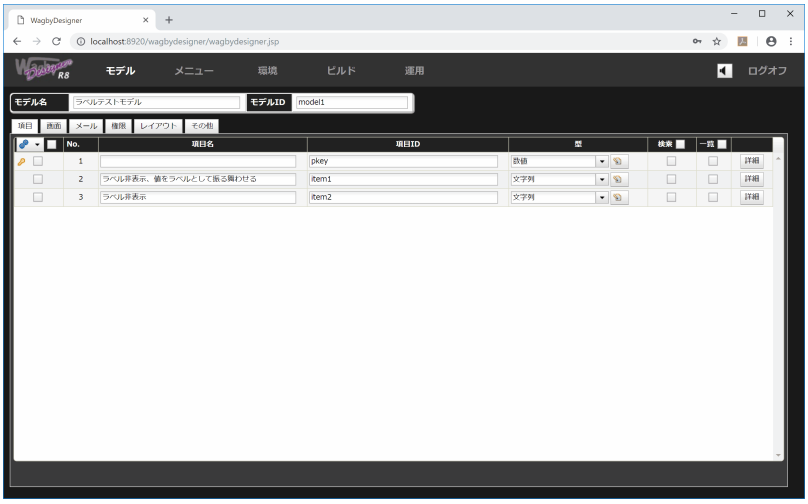


図 ラベルの代用となるitem1と、値が格納されるitem2を定義する

モデル項目詳細定義ダイアログを開きます。"詳細 > 自動計算" でラベル文字を返すようにします。ここでは単なる文字列を返していますが、ここで式を適切に指定することで、条件によって文字列の内容を変更することができます。

また、この項目はラベルの代用となるため、データベースに保存する必要はありません。

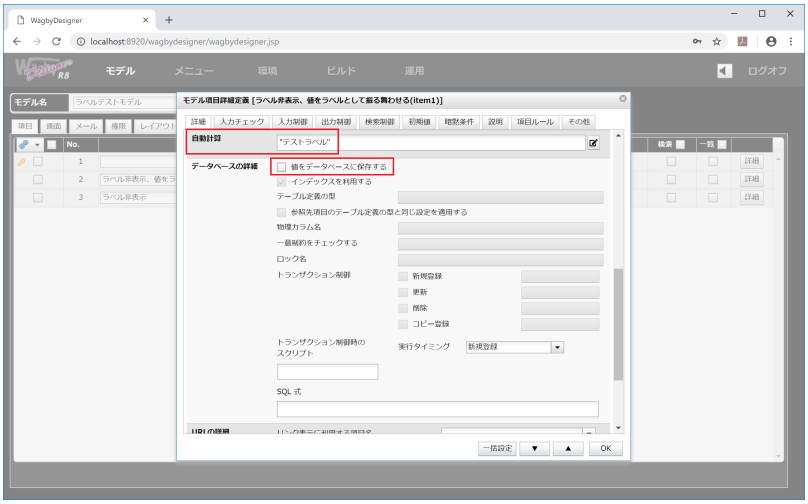


図 計算式でラベル文字を設定する

item1 項目(ラベルの代用) は "出力制御 > 詳細画面 > スタイル" で、元々のラベルを表示しないとします。また CSS クラス名に "display_label" と指定します。これは Wagby の標準でラベル部に適用される CSS です。

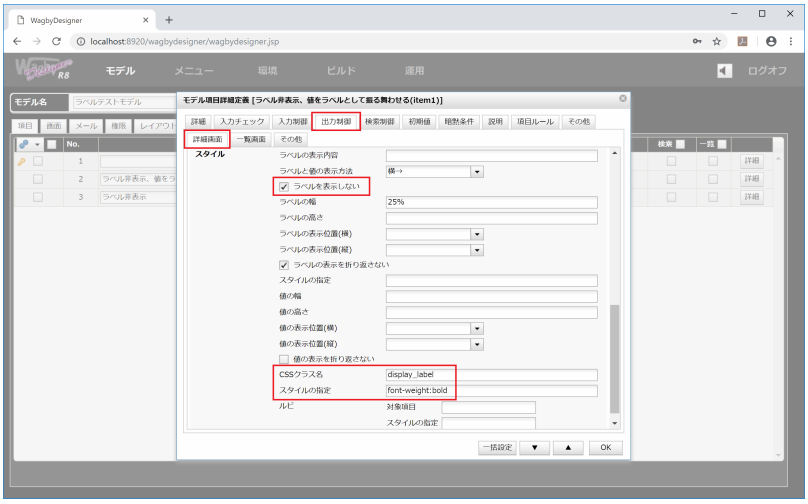


図 出力制御のスタイル設定

同じく入力制御のスタイルも設定します。ラベル部分の文字を太字とするため、CSS で "font-weight:bold" を指定します。

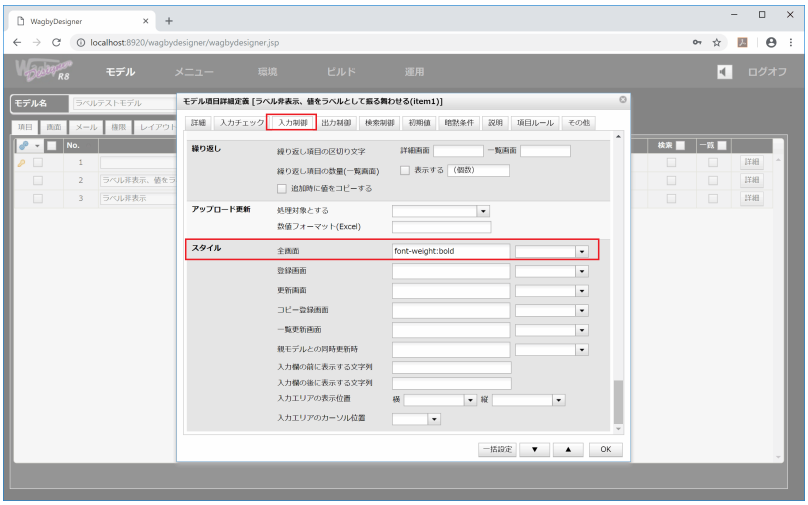


図 入力制御のスタイル設定

item1 をラベルとして使うため、item2 項目のラベルを非表示とします。

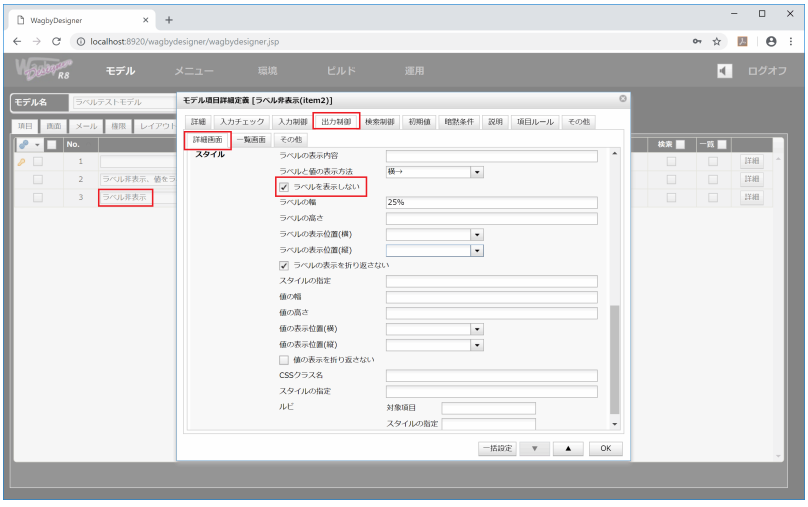


図 item2のラベルを非表示とする

レイアウトを設定します。item1とitem2をグループに格納します。最初に「追加」ボタンを押下してグループを作成し、ドラッグ&ドロップ走査で、項目をグループに含めます。両者を横並びとし、標準的な表示比率である "25%,75%" を指定します。

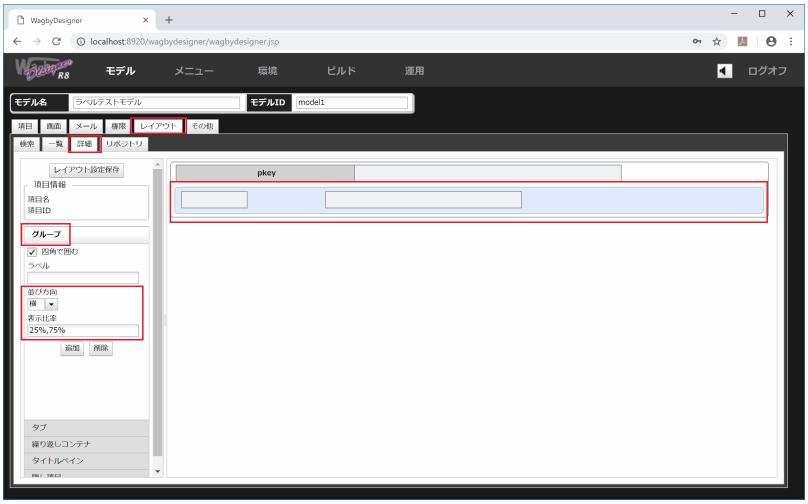


図 レイアウトの設定

画面機能

検索・一覧表示（検索）

検索と一覧を同時に表示する

画面例

検索画面と一覧表示画面を同時に表示した例です。

図 検索画面と一覧表示画面を同時に表示する

定義方法

画面タブ内の検索・一覧表示タブを開きます。「検索画面と一覧表示画面を同時に表示する」設定を有効にします。

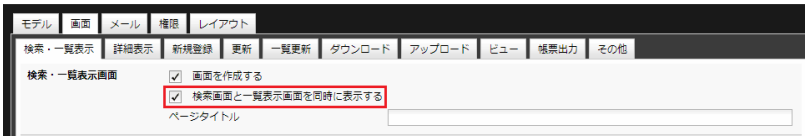


図 検索画面と一覧表示画面を同時に表示する設定

検索条件の保持ルール

パンくずのリンクを用いて（メニュー画面以外の）他の画面に遷移し、その後に検索画面に戻った場合、さきの検索条件が復元します。ただし入力チェックでエラーとなっていた検索条件は復元されません。

ページタイトル

画面例

ページタイトルを〝お得意様検索〟へ変更した例を示します。

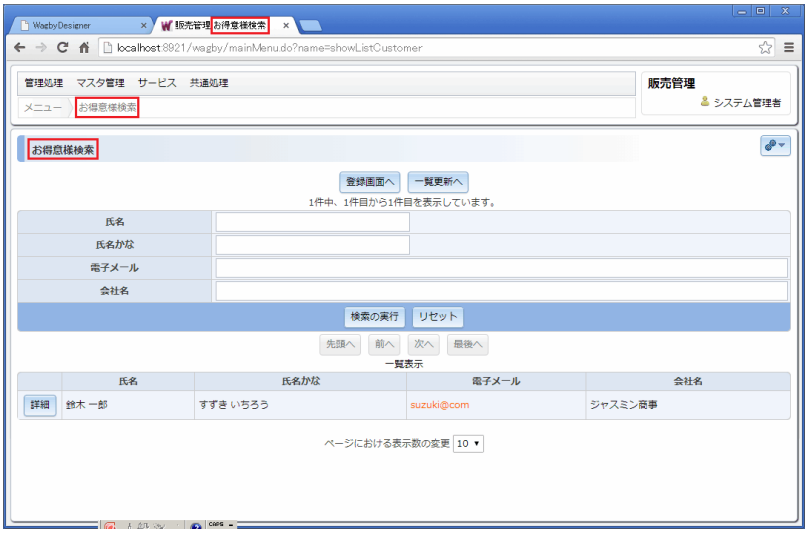


図 ページタイトルを変更

定義方法

画面タブ内の検索・一覧表示タブを開きます。「ページタイトル」の欄に文字を入力します。

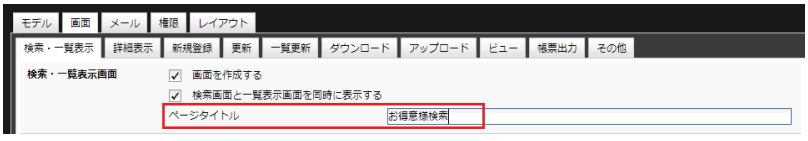


図 ページタイトルの設定

説明文

画面例

説明文を設定した例を示します。

図 説明文を表示する

定義方法

画面タブ内の検索・一覧表示タブを開きます。「説明文」の欄に文字を入力します。

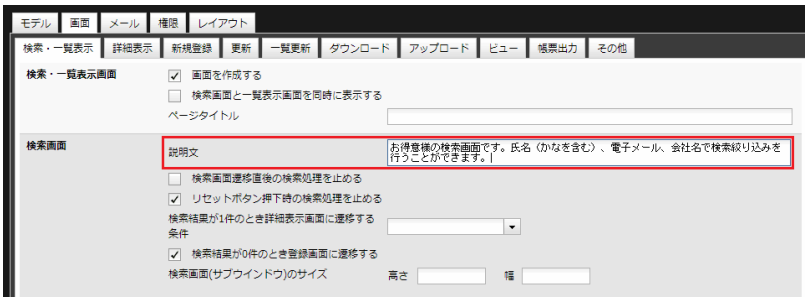


図 説明文の設定

HTMLタグを加える

説明文にHTMLタグを加えることもできます。

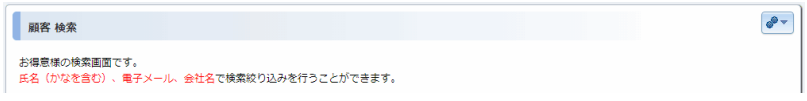


図 装飾した説明文の例

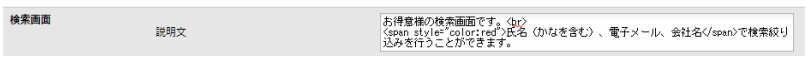


図 説明文にHTMLタグ（
,）を加えた

検索画面と一覧表示画面を同時に表示した場合

検索画面の説明文の設定が反映されます。（一覧表示画面の説明文の設定は無視されます。）

検索画面遷移直後の検索処理

検索画面に遷移した直後は検索処理が行われます。これを抑制することもできます。下記の場合に有効です。

- 初期検索処理の該当件数が多く、処理に時間がかかる。
- 検索条件に入力チェックを設定したため、検索画面遷移直後の検索処理で必ず入力エラーが発生してしまう。

図 初期検索を抑制する

適用範囲

初期検索抑制の適用範囲は次のとおりです。

- メニューまたはパンくずから遷移されてきた場合。
- 「他モデルの項目参照(検索画面)」項目からサブウィンドウで検索画面を表示した場合。ただし「サブウィンドウの検索条件をメインウィンドウの値から取得する」設定時を除く。（後述の注意点をお読み下さい。）

注意点

「サブウィンドウの検索条件をメインウィンドウの値から取得する」設定は動作の仕組み上、サブウィンドウを開いた時点で必ず検索処理を行う必要があります。この場合、本設定は行わないようにしてください。

検索項目に入力チェックが設定されていた場合、サブウィンドウを開いた際に入力チェックエラーが発生することがあります。この場合は、入力チェックを外したサブモデルを作成することで問題を回避することができます。

リセットボタン押下時の検索処理

検索画面の「リセット」ボタンを押下すると、検索条件の値が初期状態にリセットされます。 また、押下前に検索結果が表示されていた場合、結果もクリアされます。

このリセットボタンは標準では押下時に検索処理を行いません。これによって体感上の応答速度を向上させています。

リセットボタン押下時、検索条件をリセットすると同時に再検索を行いたい場合にはチェックをはずします。

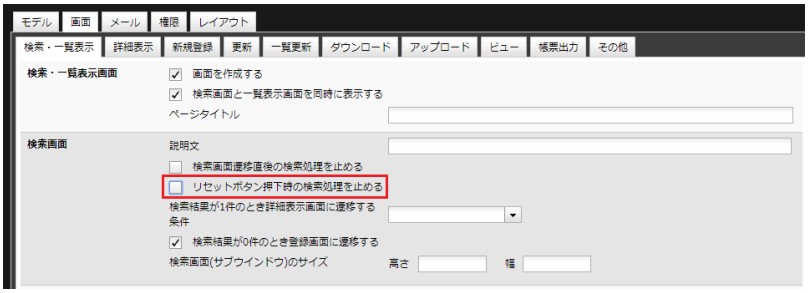


図 リセットボタン押下時の検索抑制を解除する

検索結果が1件のとき詳細画面へ

検索結果が1件のとき詳細表示画面に遷移させることができます。（標準では、一覧表示画面へ遷移します。）

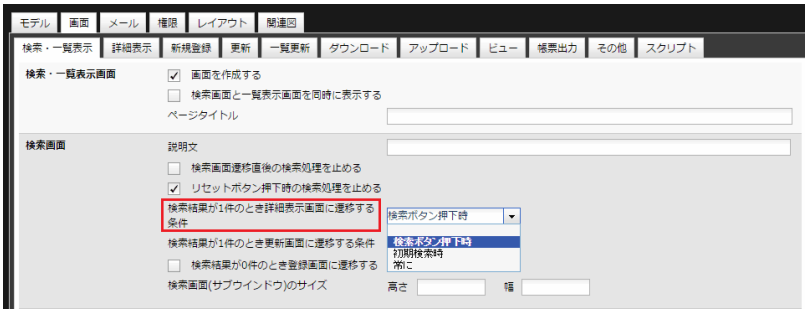


図 検索結果が1件のとき詳細表示画面に遷移する設定

以下の設定を行うことができます。

選択肢	説明
検索ボタン押下時	検索ボタン押下時のみ有効。
初期検索時	メニューから遷移時の初期検索時のみ有効
常に	検索ボタン押下時、初期検索時の両方で有効

Enterキーの取り扱いとの組み合わせ

検索項目を一つだけ用意した場合を想定します。

この画面で、本設定を"検索ボタン押下時"とし、さらに「検索制御」タブの「Enterキーの取り扱い」設定で「ボタン押下」を選択することで、検索条件入力後、Enterキーを押すとただちに詳細画面が表示されるようになります。（データが存在した場合）

検索結果が1件のとき更新画面へ

検索結果が1件のとき更新画面に遷移させることができます。

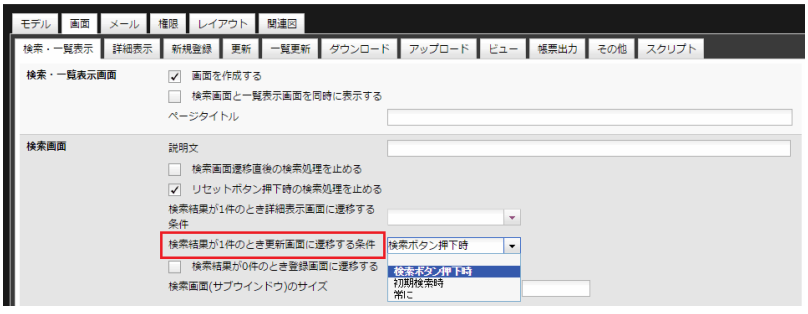


図 検索結果が1件のとき更新画面に遷移する設定

以下の設定を行うことができます。

選択肢	説明
検索ボタン押下時	検索ボタン押下時のみ有効。
初期検索時	メニューから遷移時の初期検索時のみ有効
常に	検索ボタン押下時、初期検索時の両方で有効

なお、この設定を行うときは、同時に「ボタン表示>更新ボタン」を有効にします。この設定がない場合、本機能は動作しません。

図 一覧表示画面に更新ボタンを用意する

検索結果が0件のとき登録画面へ

例

検索結果が0件のとき登録画面に遷移させることができます。　図は、検索条件に値を入力して検索を実行させようとしている例です。

図 検索条件を指定して検索する

データが見つからないとき、自動的に新規登録画面に遷移します。このとき、検索条件に入力した値はそのまま新規登録画面の初期値として再利用されます。

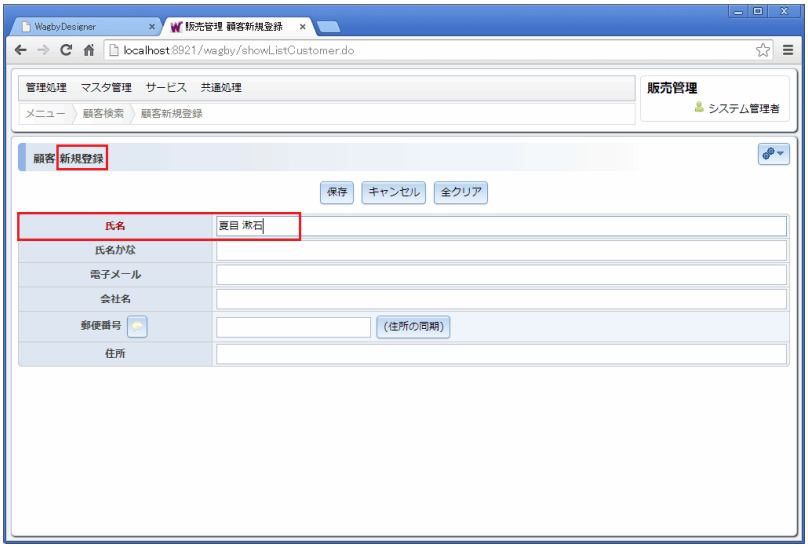


図 新規登録画面へ遷移する

定義方法

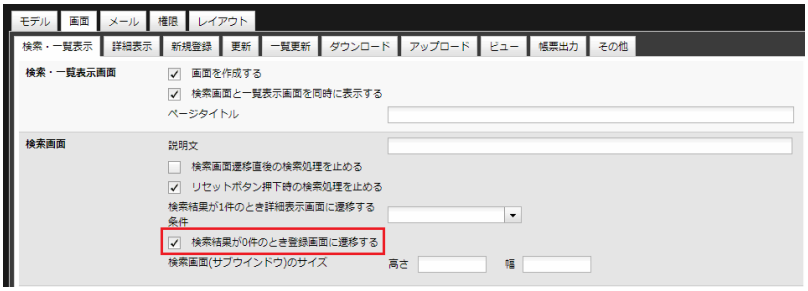


図 検索結果が0件のとき登録画面に遷移する設定

動作の詳細

- 検索ボタンを押下したときに有効です。
- 検索結果が0件のとき、新規登録画面へ遷移します。　このとき、検索条件に入力された値が、新規登録画面の初期値として自動的に反映されます。ただし値が″0″または″-1″は転記されません。
- この検索条件入力値の転記処理は、新規登録画面の初期値設定より優先されます。
- 繰り返し項目、繰り返しコンテナは、第一レコードに転記されます。
- 検索条件にエラーがあった場合(必須入力ミス、範囲チェックエラーなど)は、新規登録画面へ遷移しません。

検索結果数の上限指定

検索結果の上限を指定することができます。「検索結果が指定件数より多いときエラーとする」設定欄に、数値を指定します。検索処理の結果が指定した件数を超えた場合はエラーメッセージが表示され、一覧表示を行うことができなくなります。

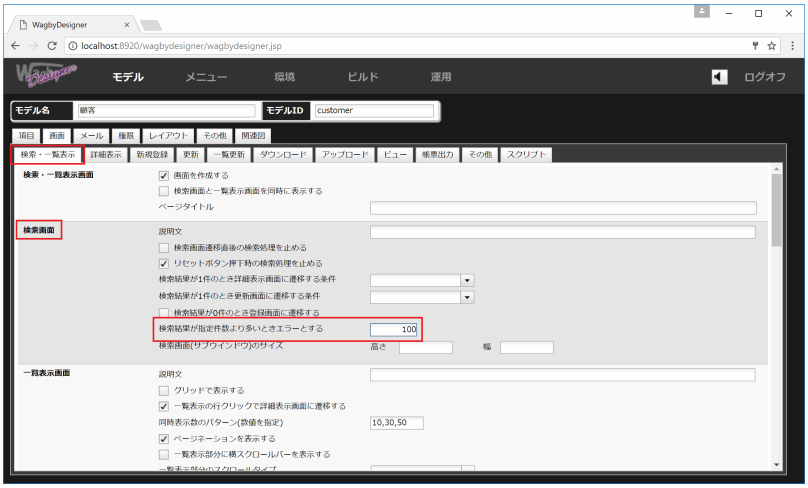


図 検索結果が指定件数より多いときエラーとする

仕様

- この設定は検索画面（サブウィンドウ）、一覧更新画面にも同時に適用されます。
- ダウンロード画面には適用されません。

検索ボタンの表示位置

検索条件部のボタンの位置を変更することができます。

上部に配置する

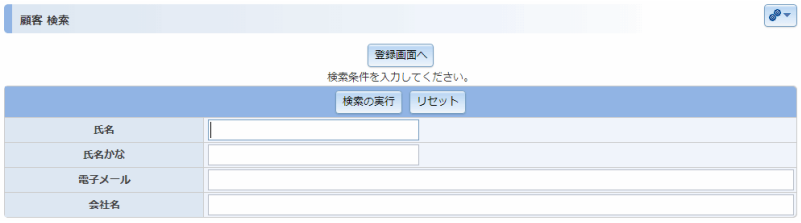


図 検索ボタンを上部に配置した例

下部に配置する

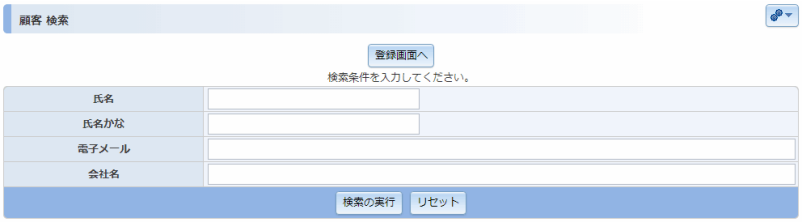


図 検索ボタンを下部に配置した例（標準）

上部と下部の両方に配置する

図 検索ボタンを上部と下部の両方に配置した例

定義方法

画面タブ内の検索・一覧表示タブに用意された「ボタン表示>検索ボタンの表示位置」を設定します。空白時は「下」として扱います。

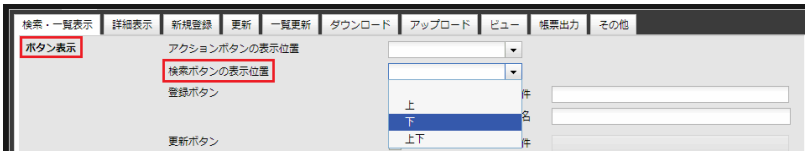


図 検索ボタンの表示位置の設定

検索・一覧表示（一覧表示）

同時表示数

一覧表示の下部に、「ページにおける表示数の変更」を指定するリストボックスが表示されます。

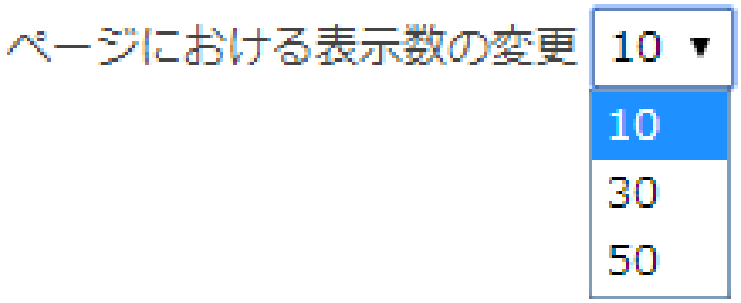


図 ページにおける表示数の変更

設定

画面タブ内の検索・一覧表示タブに用意された「同時表示数のパターン」が反映されます。コンマ区切りで複数の数値を指定することができます。

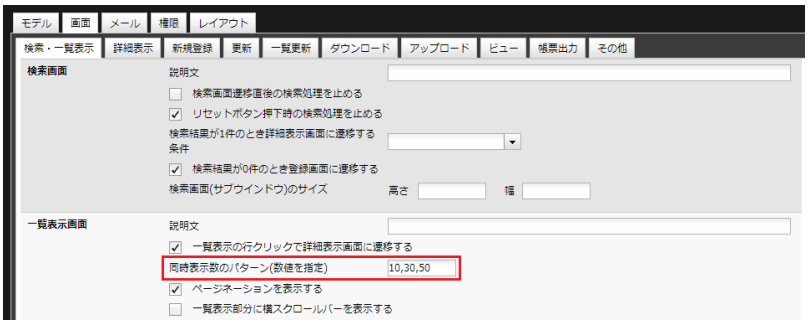


図 同時表示数のパターンを設定する

(202) (203)

値の保存ルール

- 選択した値はサーバに記憶されます。再ログオンしても前回の「表示数」が維持されます。
- 検索の「リセット」ボタンを押下すると、この値は初期化されます。（先頭の選択肢が初期値となります。）
- 運用後、この値を変更して再ビルドし、アプリケーションを入れ替えたときでも、最初にログインしたときは以前に選択した値が残っています。「リセット」ボタンを押すことで保存されていた値を消去することができます。

ページネーション

ページネーションは一覧表示結果のページを直接、指定して画面遷移を行う機能です。図のようにページ番号が表示されます。ページ番号をマウスでクリックすることで直接、そのページに遷移します。

(202) この値を一つ（例 100）とすることで、図のリストボックスは表示されません。（変更不可となります。）

(203) この値を空白にした場合、"10" と扱われます。（単一の値のため、リストボックスも非表示となります。）また、負数を指定することはできません。

図 ページネーションの表示

表示されるページネーションの最大数は20となります。これ以降もデータが存在する場合は"..."という文字が表示されます。

ページネーションを非表示にする

画面タブ内の検索・一覧表示タブに用意された「ページネーションを表示する」を無効にすると、ページネーションは非表示となります。

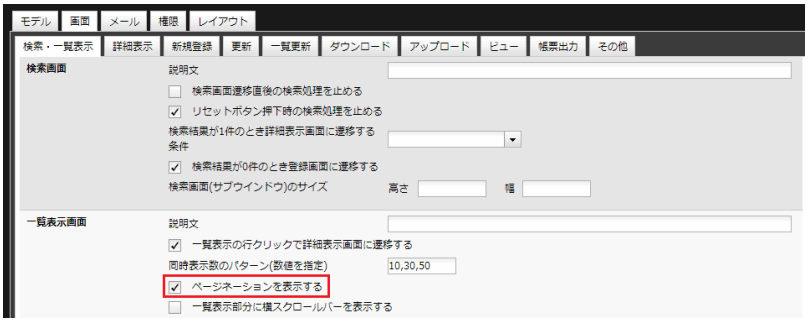


図 ページネーションの表示制御

ページネーション非表示時も、「先頭へ」「前へ」「次へ」「最後へ」ボタンを使うことでページを切り替えることはできます。

図 前へ、次へボタンを利用してページを切り替える

ページネーションの位置の記憶

現在、どのページを閲覧しているかという（ページネーションの）位置は、再検索を行なうまで保持されます。

テーブルの幅

標準ではテーブル幅はブラウザの表示領域にあわせて自動設定されます。各列の幅もこれにあわせて自動設定されます。

テーブル幅をピクセル単位で指定すると、各列の幅もあわせてピクセル列で指定することができます。

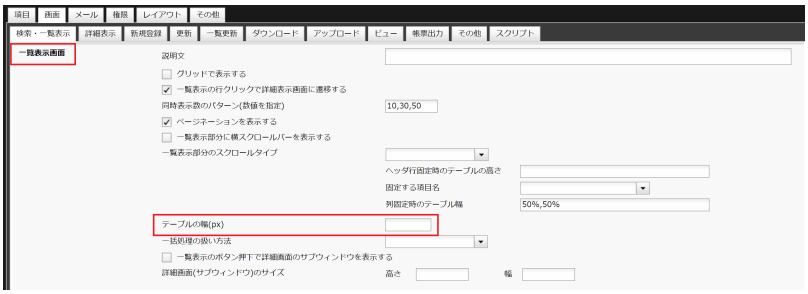


図 テーブル幅の指定

横スクロールバー表示

一覧表示の項目数が多い場合、Webブラウザが自動的に各項目の幅を調整し、できるだけ画面幅に収まるように描画します。

本設定を有効にすると、画面幅に収めるのではなく、横スクロールバーを出すようになります。　このとき、各項目の幅（「ラベルの幅」または「値の幅」）を設定してください。本設定は、この幅指定とセットで機能します。

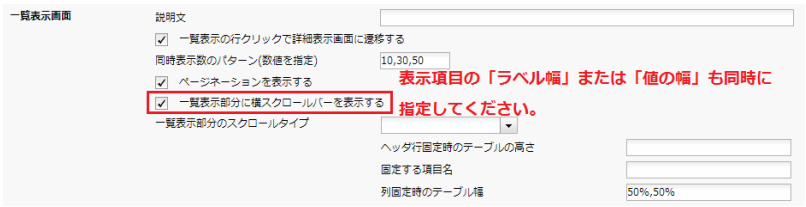


図 横スクロールバーの表示制御

スクロールタイプ

「一覧更新>一覧表示部分のスクロールタイプ（列固定）」「一覧更新>一覧表示部分のスクロールタイプ（ヘッダ行固定）」でそれぞれ説明します。

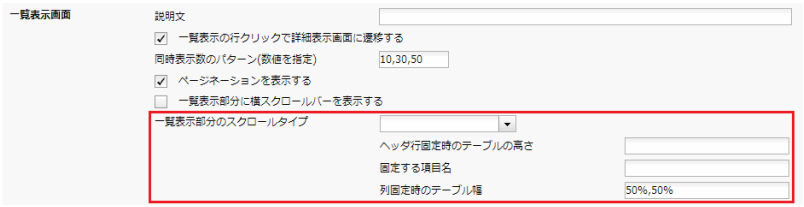


図 スクロールタイプの設定

アクションボタンの表示位置

（画面遷移を行うための）アクションボタンの位置を変更することができます。

上部に配置する

図 アクションボタンを上部に配置した例（標準）

下部に配置する

図 アクションボタンを下部に配置した例

上部と下部の両方に配置する

図 アクションボタンを上部と下部の両方に配置した例

定義方法

画面タブ内の検索・一覧表示タブに用意された「ボタン表示>アクションボタンの表示位置」を設定します。空白時は「上」として扱います。

図 アクションボタンの表示位置の設定

ボタン（機能）

検索・一覧表示画面に用意できるボタン（機能）を説明します。

検索画面

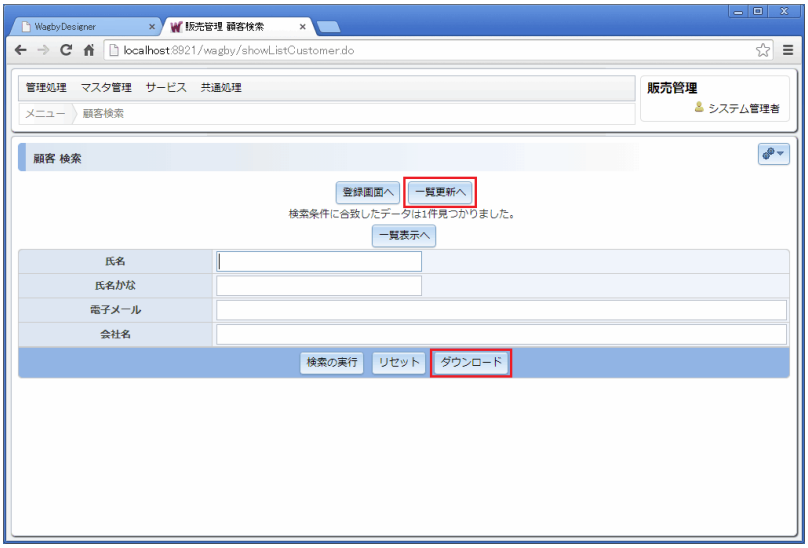


図 検索画面に用意できるボタン

一覧表示画面

図 一覧表示画面に用意できるボタン

定義方法

画面タブ内の検索・一覧表示タブに用意された「ボタン表示」を有効にします。

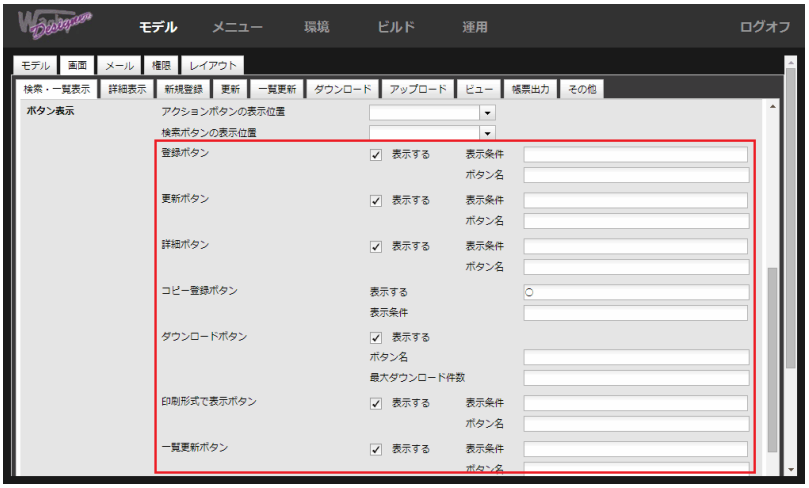


図 ボタン表示の設定

各ボタンの表示を有効にする場合、関連する画面機能が作成されている必要があります。 画面機能が作成されない設定では、ボタンも作成されません。

種別	配置	初期状態	関連する画面
登録	検索画面のアクション部一覧表示画面のアクション部	○	登録画面
更新	一覧表示画面の各行	−	更新画面
詳細	一覧表示画面の各行	○	詳細表示画面
コピー登録	一覧表示画面の各行。 「表示する」設定欄に「○」と入力するとボタンが一つ用意される。 同欄にコンマ区切りで複数のボタン名を記載することもできる。	−	新規登録画面
ダウンロード	検索画面の検索部。一覧表示画面のアクション部。	−	ダウンロード画面
印刷形式で表示	一覧表示画面のアクション部。 検索結果が存在する場合のみ表示される。	−	−
一覧更新	検索画面のアクション部。一覧表示画面のアクション部。	−	一覧更新画面
一括更新	一覧表示画面のアクション部。 一覧表示画面の左側にデータを選択するためのチェックボックスが用意される。 検索結果が存在する場合のみ表示される。	−	−
一括削除	一覧表示画面のアクション部。 一覧表示画面の左側にデータを選択するためのチェックボックスが用意される。 検索結果が存在する場合のみ表示される。	−	−

一括帳票出力	一覧表示画面のアクション部。 一覧表示画面の左側にデータを選択するためのチェックボックスが用意される。 検索結果が存在する場合のみ表示される。	-	（帳票定義）
一括メール送信	一覧表示画面のアクション部。 一覧表示画面の左側にデータを選択するためのチェックボックスが用意される。 検索結果が存在する場合のみ表示される。	-	（メール送信定義）

ボタンの表示制御

「ボタン名と表示条件のカスタマイズ」で説明します。

ダウンロード

定義方法

図で「ダウンロードボタン」を「表示する」とし、かつ、「画面 〉 ダウンロード 〉 ダウンロード画面」を作成することで、検索・一覧表示画面に「ダウンロード」ボタンが用意されます。

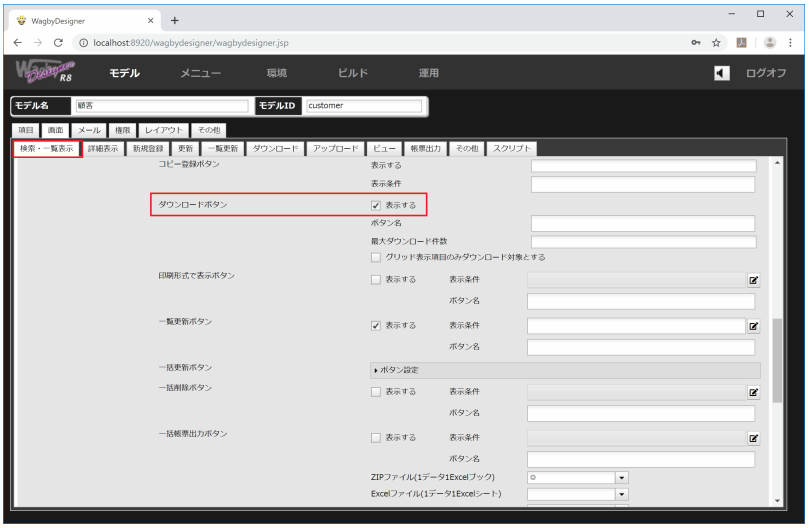


図 ダウンロードボタンを用意する

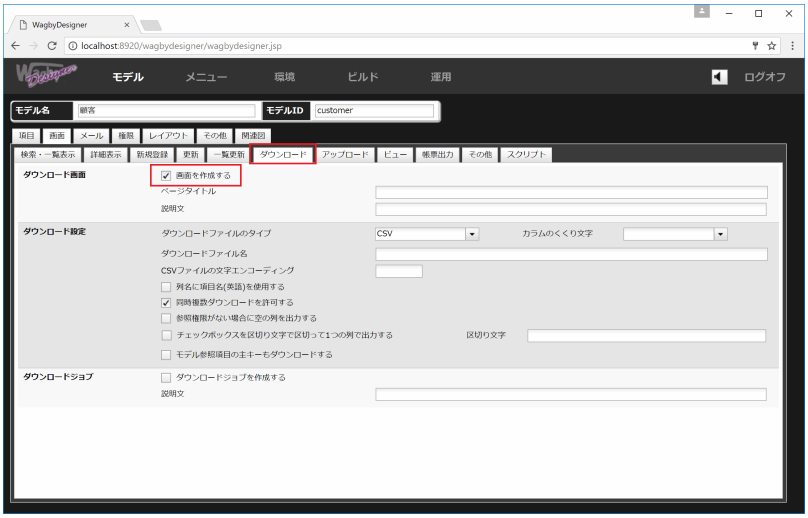


図 ダウンロード画面を作成する

実行

検索結果が存在するとき、ダウンロードボタンが用意されます。

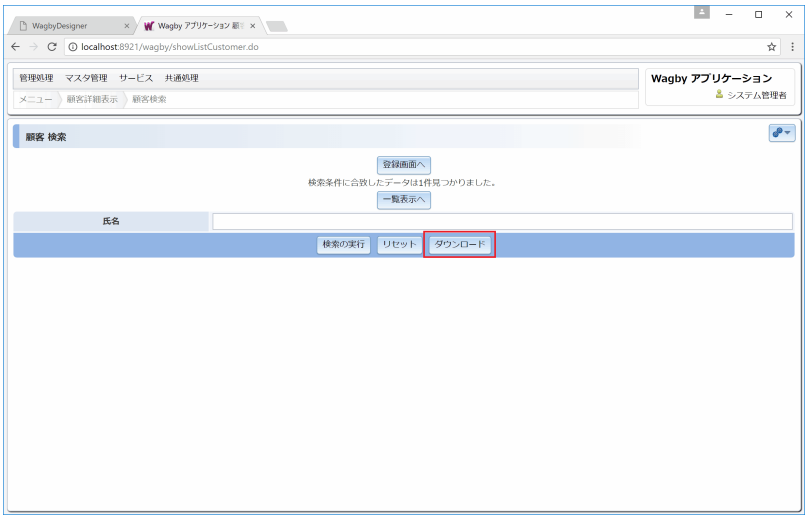


図 ダウンロードボタン

ダウンロード機能の動作は「CSV・Excel入出力 ＞ ダウンロード」に準拠します。

検索結果が存在しないとき（リセットボタンで検索条件をクリアしたときを含む）は、ダウンロードボタンは表示されません。

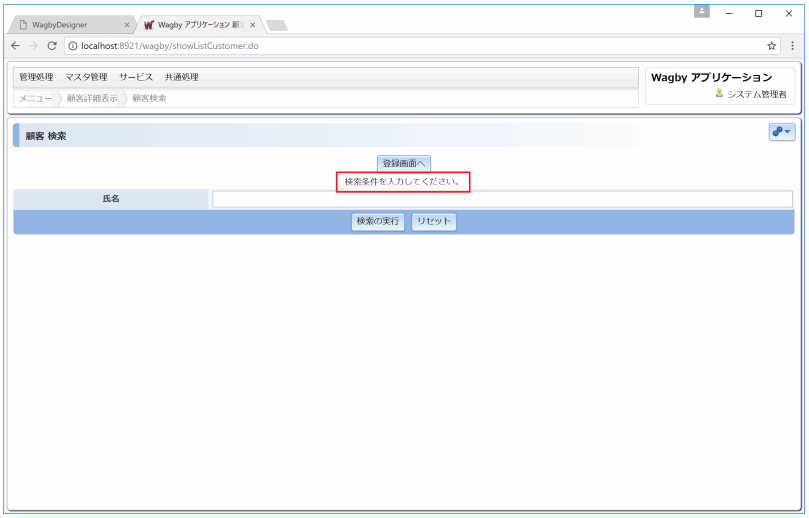


図 検索を実行していない

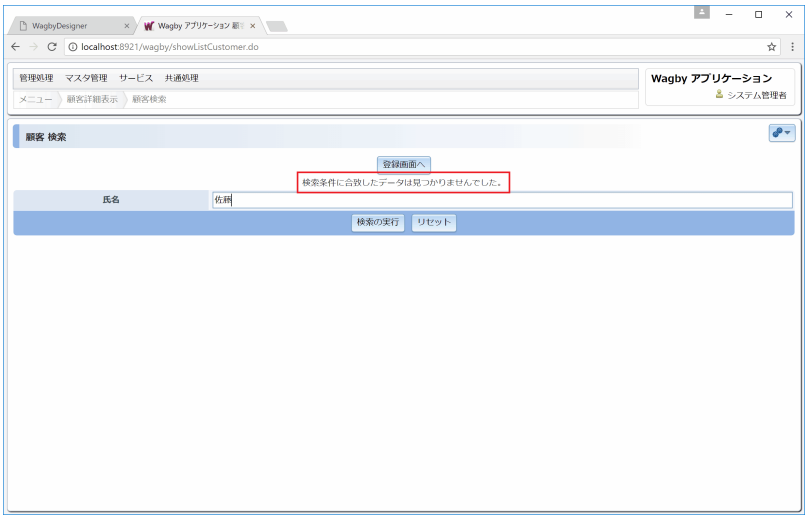


図 検索結果が0件である

ダウンロード件数の上限を指定する

最大ダウンロード件数を指定することができます。固定値になります。（式を記述することはできません。）

検索結果数が上限を超えた場合、ダウンロードボタンを押下することができなくなります。

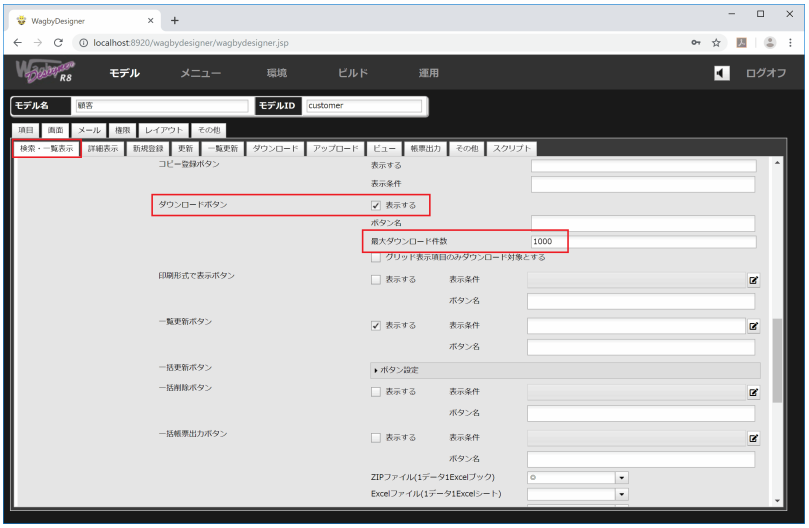


図 最大ダウンロード件数

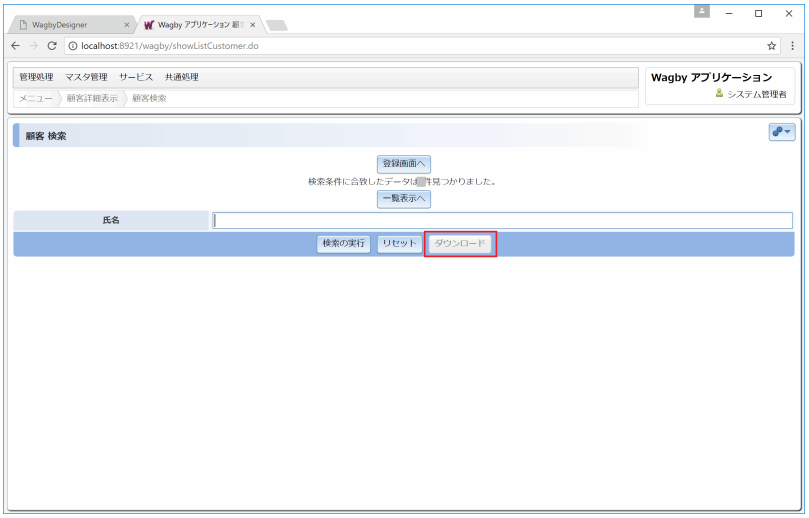


図 検索結果数が上限を超えるとボタンが非活性になる

グリッド表示項目のみダウンロード対象とする

一覧表示をグリッド形式としたとき、かつ、ダウンロードボタンを用意したときに有効です。この設定を行うと、グリッドで指定した列順でCSVファイルをダウンロードします。

(204)

(204) この設定を行わない場合は、モデルの項目順でダウンロードされます。

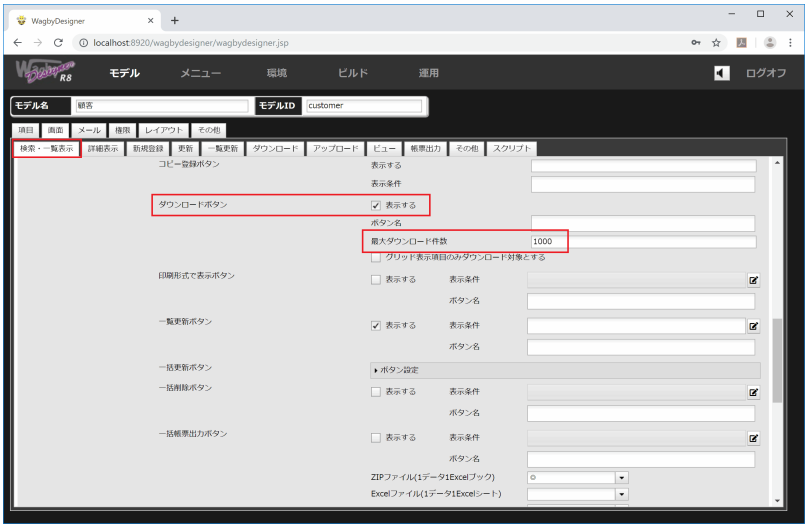


図 グリッド表示項目のみダウンロード対象とする

グラフの出力

一覧表示の値を使ってグラフを出力することができます。詳細は集計・グラフ ＞ 一覧表示画面にグラフを表示するをお読みください。

仕様・制約

絞り込み設定

〝他モデルの参照 ＞ 参照モデル同士の関係性で選択肢を絞り込む ＞ トラブルシューティング ＞ 一覧表示画面で絞込が動作しない〟をお読みください。

行クリックで詳細画面へ

一覧表示画面から詳細画面へ遷移するには二つの方法があります。

- 各行の左に用意された「詳細」ボタンを押下する。
- 対象行にマウスをあわせ、クリックする。（「詳細」ボタン押下と同じ効果）

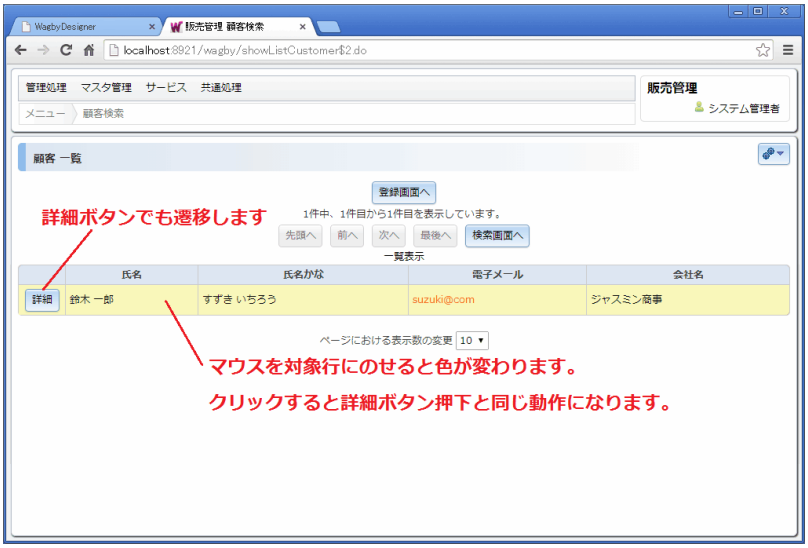


図 詳細画面への遷移方法

一覧表示の行クリックで詳細表示画面に遷移する機能を制御することができます。標準では有効になっています。

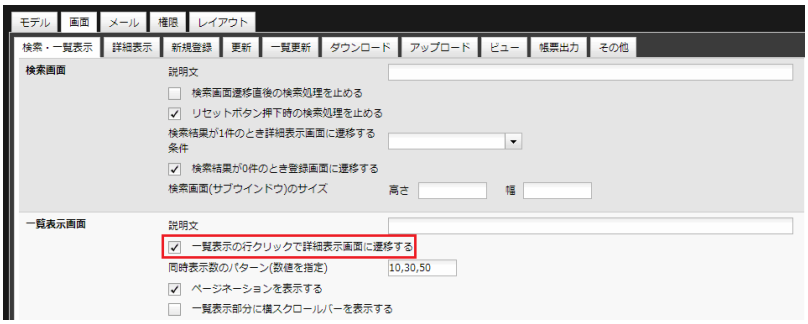


図 一覧表示の行クリックで詳細表示画面に遷移する設定

ご注意ください

- 詳細表示画面を作成しない場合、本機能は無効になります。
- 一覧表示の行クリックで詳細表示画面に遷移する機能は、詳細ボタンが用意されていることが前提です。詳細ボタンを無効にして、行クリックのみを有効にすることはできません。

検索・一覧表示（一覧表示）グリッドを利用する

定義方法

対象モデルの「画面 > 検索・一覧表示」で、「グリッドで表示する」を有効にします。（このとき「検索画面と一覧表示画面を同時に表示する」も自動的に有効になります。）

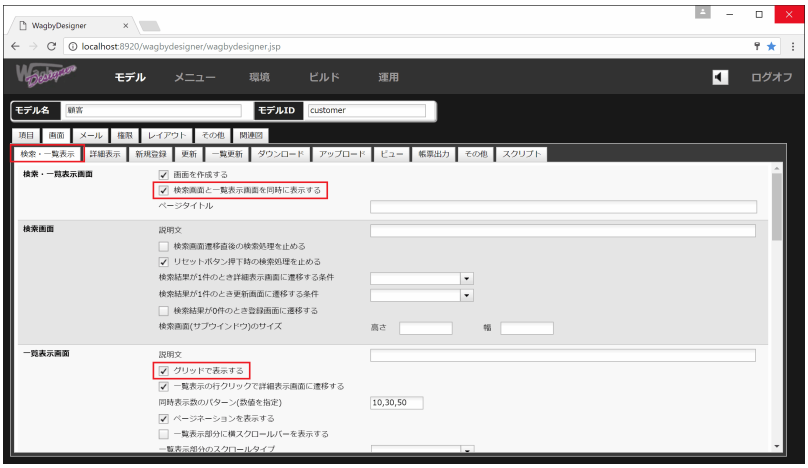


図 グリッドで表示する設定

「グリッドで表示する」を有効にしたとき、このモデルの REST API 機能が自動的に有効になります。（手動で設定する必要はありません。）

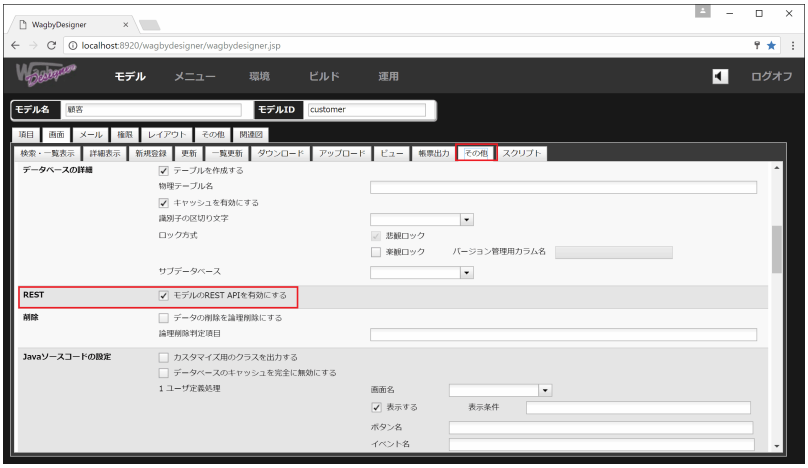


図 RESTAPIが有効になる

実行

グリッドで表示された一覧表示の例を示します。

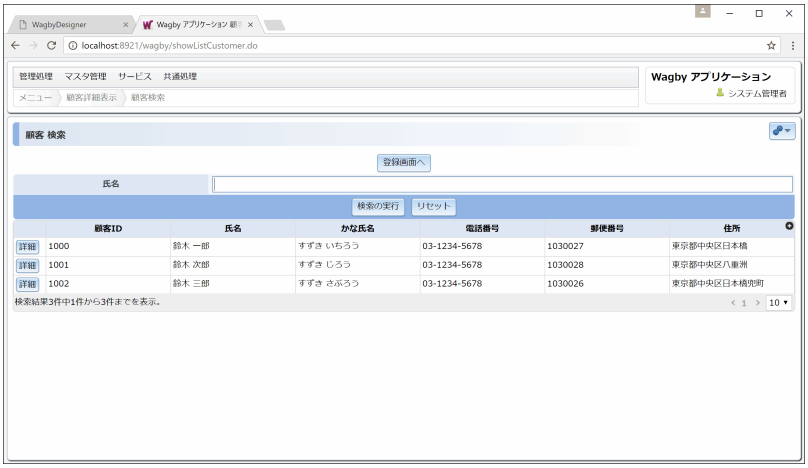


図 グリッドで表示された一覧表示

ソート（並び替え）

列名をクリックすることでソートが行われます。再びクリックすると昇順と降順が入れ替わります。

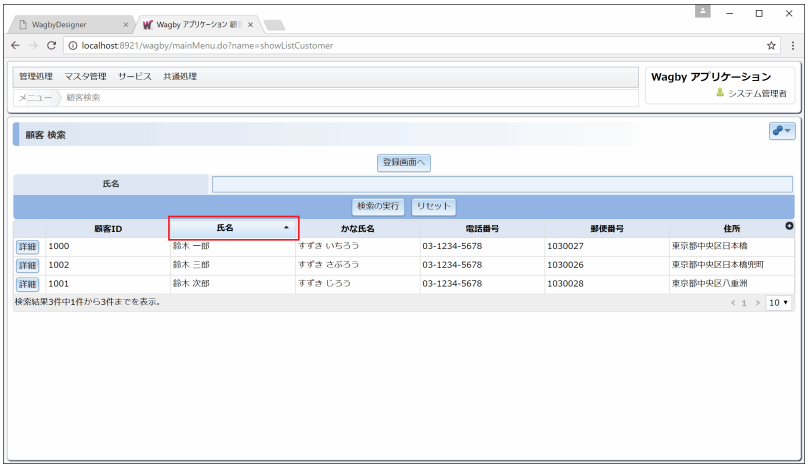


図 項目によってソートされる

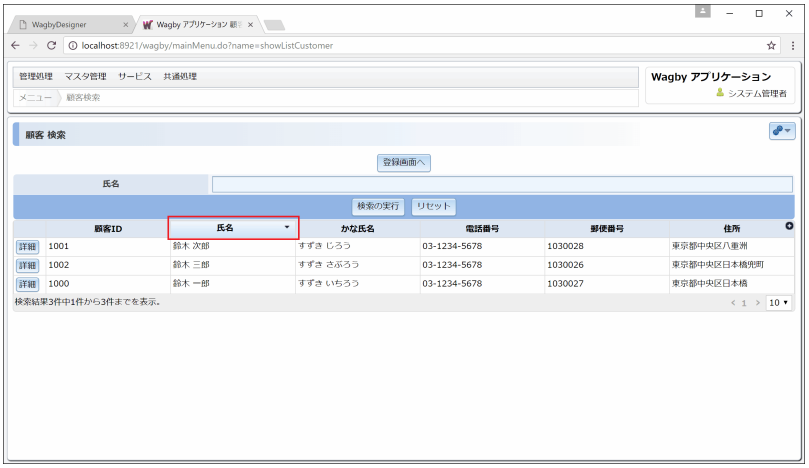


図 クリックすると昇順と降順が入れ替わる

列の移動

列部分をドラッグ&ドロップで移動することができます。

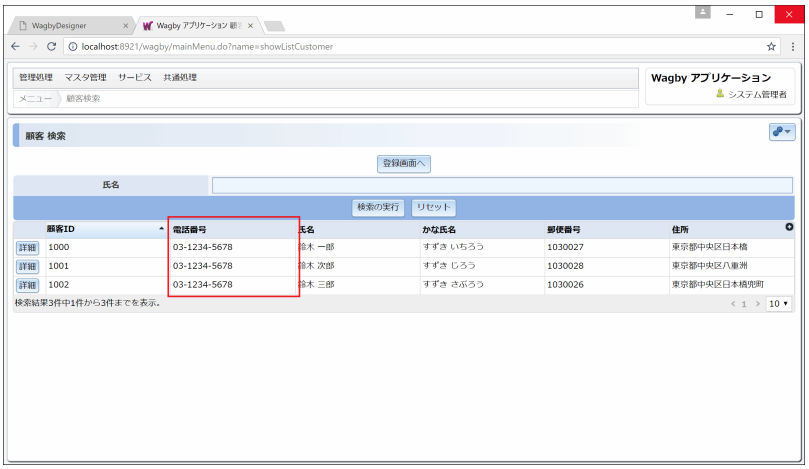


図 「電話番号」列をドラッグ&ドロップで移動した

(205)

列の表示幅の変更

列の表示幅をドラッグ&ドロップで変更することができます。

(205) 列の表示並びは利用者毎に記憶されます。再ログオンした場合でも復元されます。

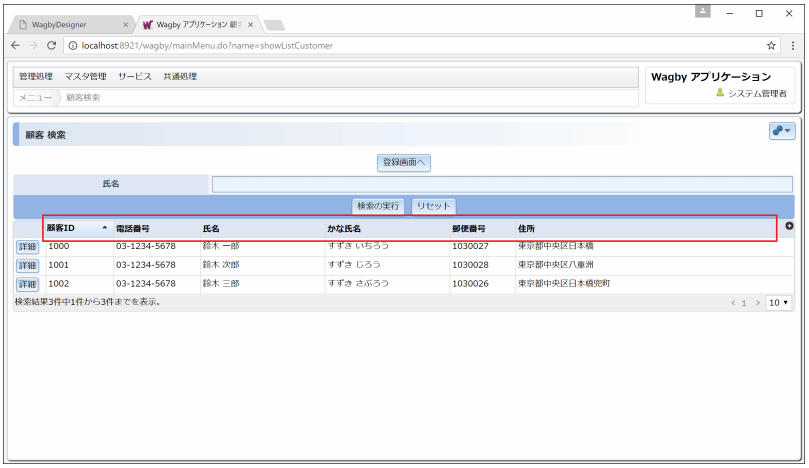


図 各列の表示幅を変更した

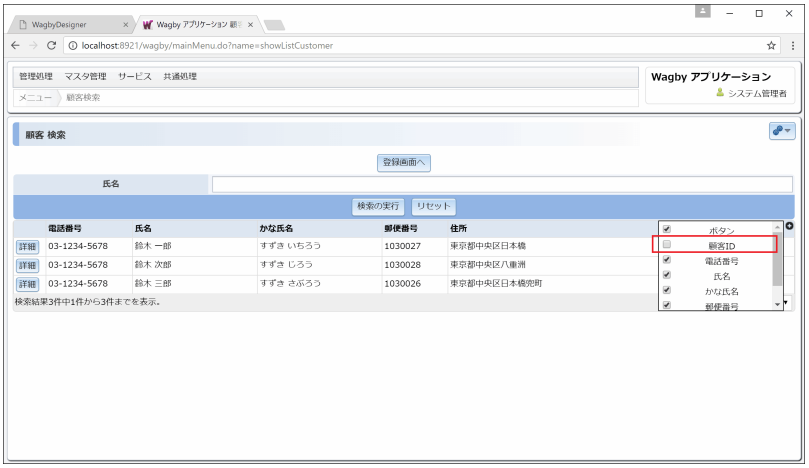
(206)

列を非表示にする

グリッドの右端にある「+」ボタンを押下すると、表示対象列名の一覧が表示されます。

図 表示対象列名の一覧

列を非表示とすることができます。



(206) 列の表示幅は利用者毎に記憶されます。再ログオンした場合でも復元されます。

図 「顧客ID」列を非表示にした

(207)

同時表示数とページネーション

グリッドの右下にある「同時表示数」を変更することができます。変更と同時にページネーションも変わります。

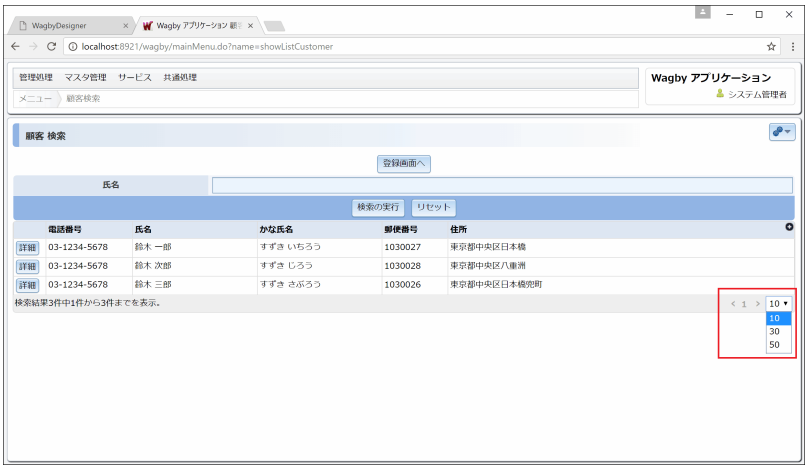


図 同時表示数を変更する

図 ページネーションも変更される

(208)

設定

画面タブ内の検索・一覧表示タブに用意された「同時表示数のパターン」が反映されます。コンマ区切りで複数の数値を指定することができます。

(207) 列の非表示設定は利用者毎に記憶されます。再ログオンした場合でも復元されます。

(208) 選択した値は利用者毎に記憶されます。再ログオンした場合でも復元されます。

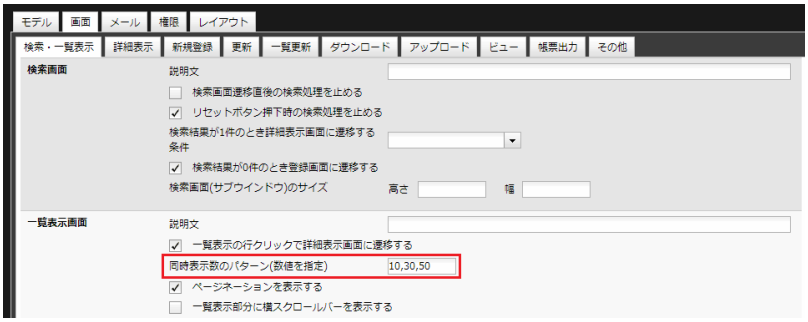


図 同時表示数のパターンを設定する

(209) (210)

縦スクロールバーを表示する

グリッドの縦のサイズを指定することができます。縦スクロールバーが表示されます。

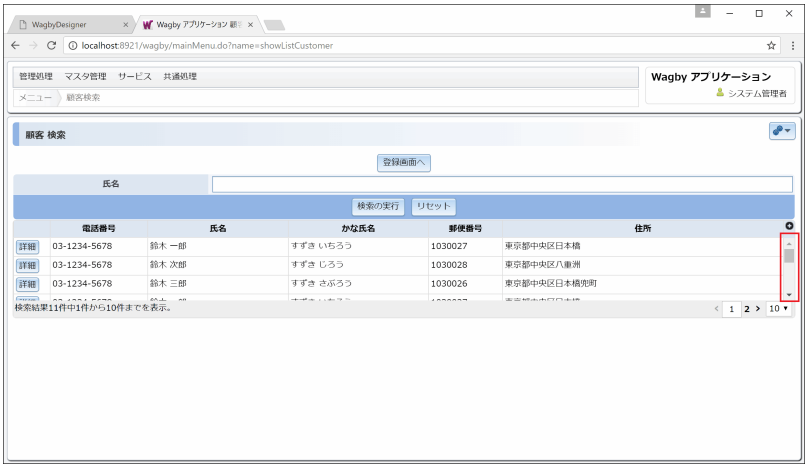


図 縦のサイズを固定した（縦スクロールバーが表示される）

定義方法

一覧表示部分のスクロールタイプを「ヘッダ列を固定」とします。同時に、ヘッダ行固定時のテーブルの高さを指定します。単位はピクセル（px）またはエム（em）を指定してください。

(209) この値を一つ（例 100）とすることで、図のリストボックスは非表示になります。表示数は固定で、変更不可と扱います。

(210) この値を空白にした場合、"10" と扱われます。（単一の値のため、リストボックスも非表示となります。）また、負数を指定することはできません。

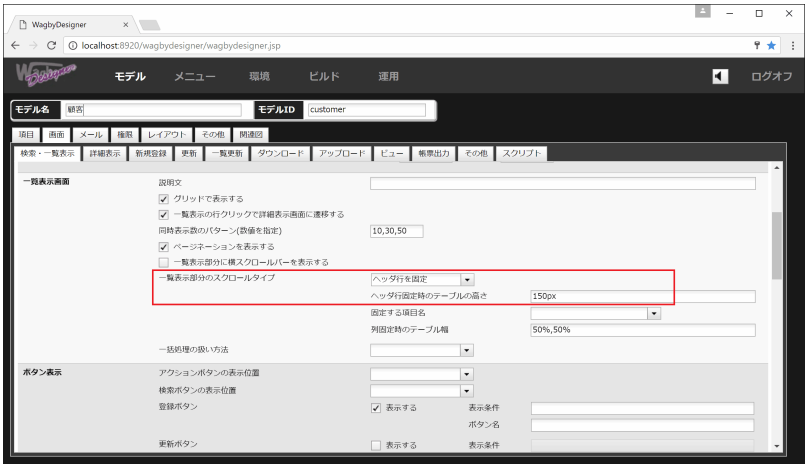


図 ヘッダ列の設定

スクロール時にデータを読み込む(onDemand Grid)

ページネーションを使わず、スクロールバー操作時に必要に応じてデータを取得することができます。(OnDemand Gridと呼びます。)

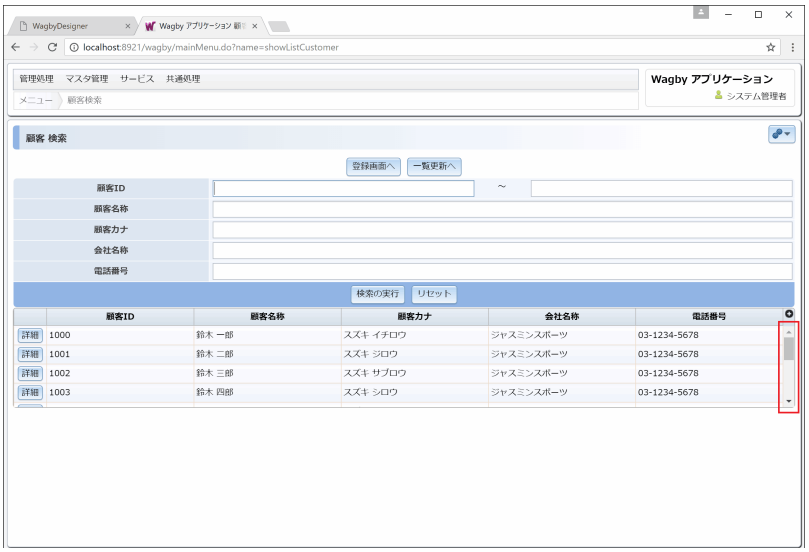


図 スクロールバーに連動して必要なデータが読み込まれる

定義方法

「ページネーションを表示する」の設定を解除します。あわせて、一覧表示部分のスクロールタイプを「ヘッダ列を固定」と、ヘッダ行固定時のテーブルの高さを指定します。単位はピクセル (px) またはエム (em) を指定してください。

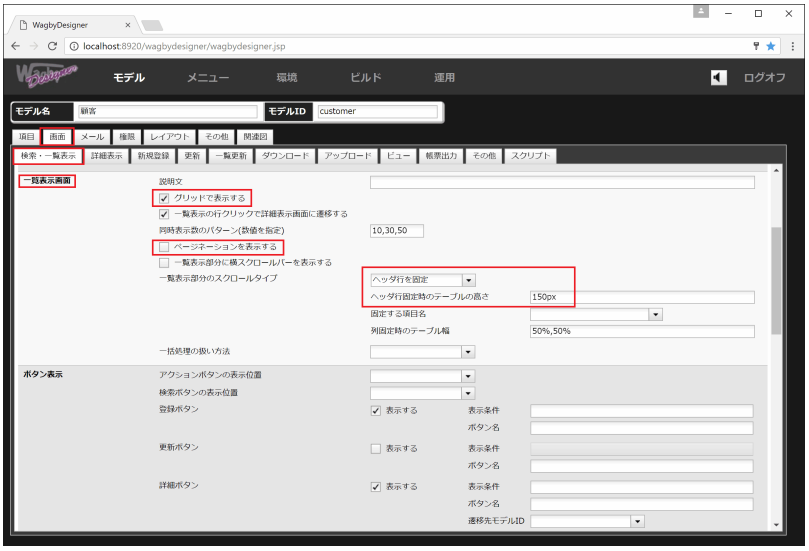


図 ページネーションの表示を解除する

一括処理

”一括処理 > グリッドでの設定”をお読みください。

設定を共通で利用する

グリッドの設定（列の表示・非表示と、並び）を異なる利用者間で共通に利用することができます。

プリファレンス共通設定プリンシパル

グリッドの設定を公開できるアカウントを指定します。これは「プリファレンス共通設定管理」として用意されています。この例ではアカウント「ユーザー1」に付与します。[プリンシパルの詳細...]

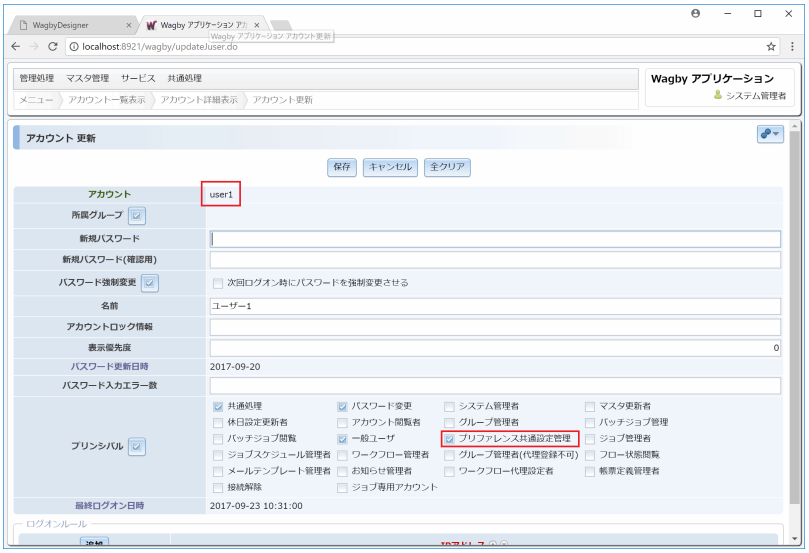


図 プリファレンス共通設定管理プリンシパル

設定を公開する

アカウント「ユーザー1」でログオンします。一覧表示グリッドの下部に、共通設定名を入力できる欄が用意されます。

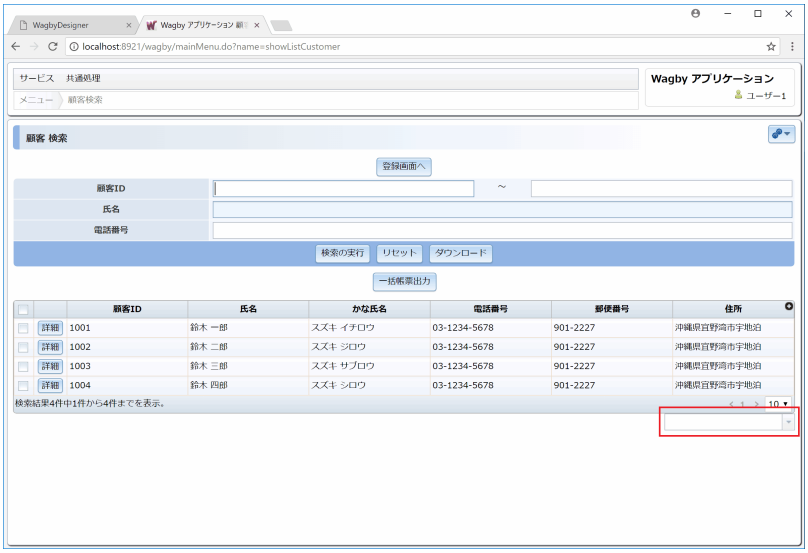


図 共通設定名の入力欄

ここで、標準の列設定を変更してみます。

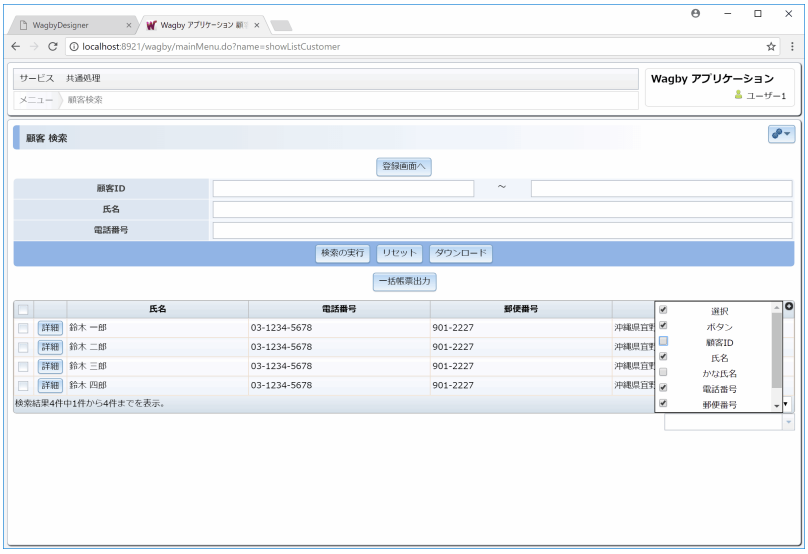


図 列設定の変更

変更後、共通設定名として「顧客一覧ユーザー1設定」と入力します。（名称は任意です。）

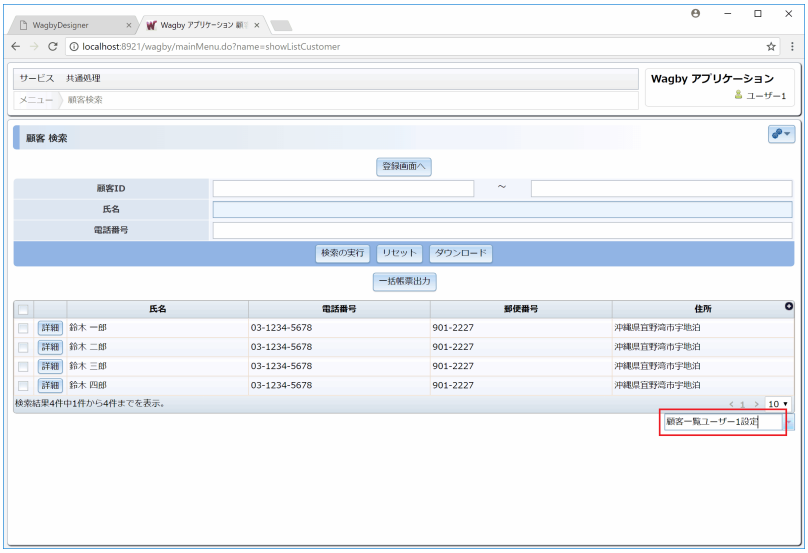


図 共通設定名を入力する

カーソルを移動すると「保存」ボタンが表示されますので、これを押下します。

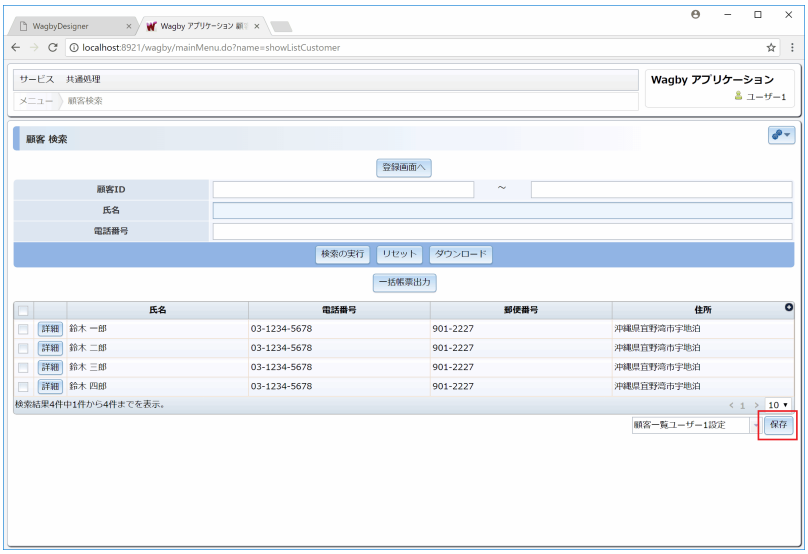


図 共通設定を保存する

共通設定を再利用する

アカウント「ユーザー2」でログオンします。ユーザー2は「プリファレンス共通設定管理」プリンシパルを付与されていません。そのため共通設定の登録は行えませんが、既存の設定を選択することができます。

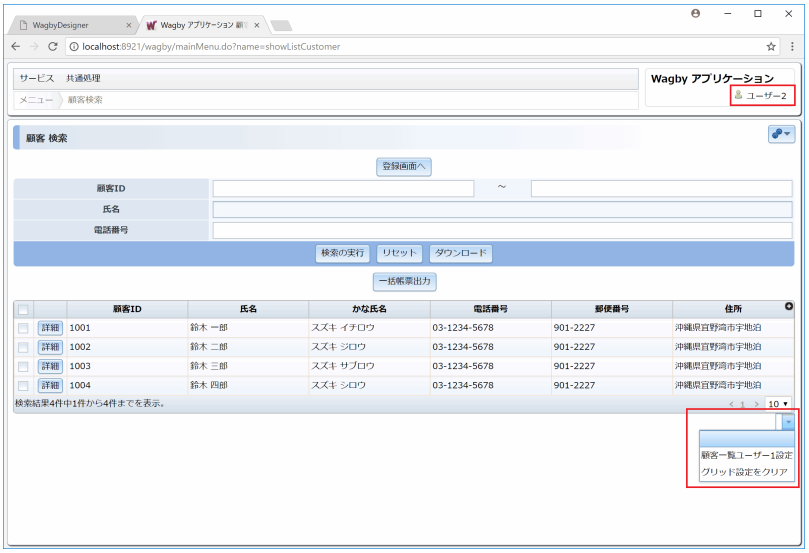


図 保存された共通設定を選択する

共通設定を読み込むと、列の設定が変更されます。

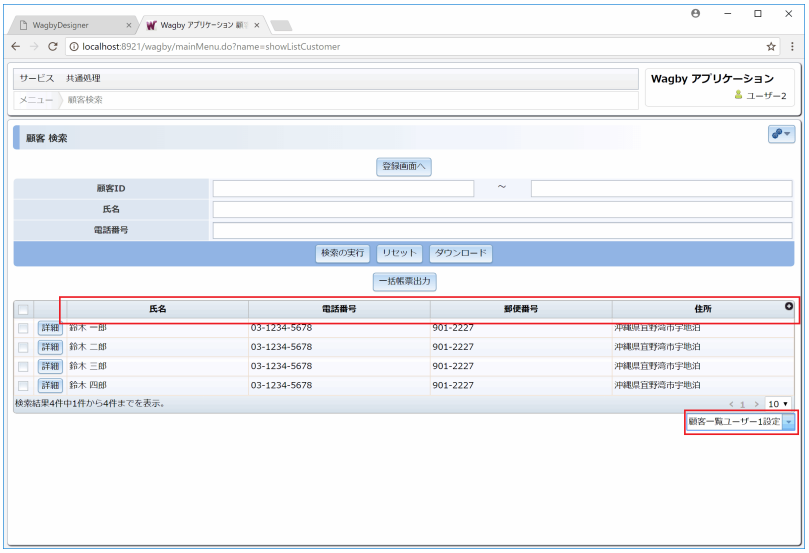


図 共通設定を再利用した

標準の設定に戻す

リポジトリ（設計情報）に指定された標準設定に戻す場合、「グリッド設定をクリア」を選択します。

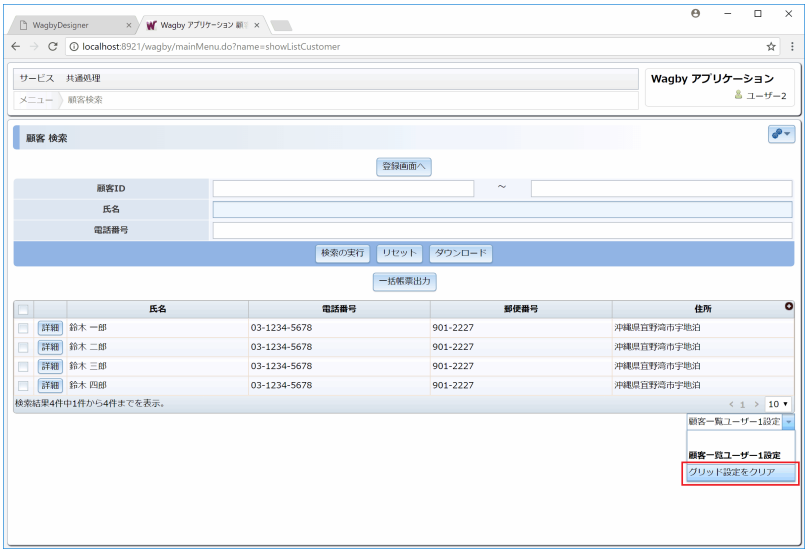


図 グリッド設定をクリア

標準の設定に戻ります。

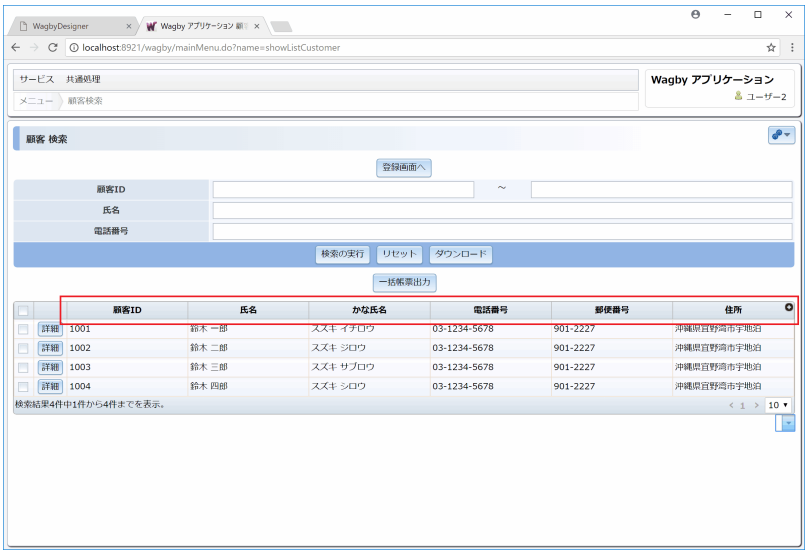


図 標準設定に戻した

グラフの出力

一覧表示の値を使ってグラフを出力することができます。詳細は集計・グラフ 〉一覧表示画面にグラフを表示するをお読みください。

仕様・制約

仕様

- 検索と一覧表示は一つの画面にまとまります。検索画面と一覧表示画面を別にすることはできません。
- 「一覧表示部分に横スクロールバーを表示する」設定は無効です。グリッド内で必要に応じて横スクロールバーが自動的に表示されます。
- グリッドが提供する機能（ソート、列の並び替え、列幅の設定、列の非表示設定）を個別に無効とすることはできません。
- グリッドのソートおよび列並び替えは、グリッドのみで有効です。ソートや列の並び替え設定は他の画面機能（印刷形式表示の画面、CSVダウンロードなど）に反映されるものではありません。

- 全文検索と一覧表示（グリッド）の組み合わせを行うことはできません。
- モデル参照（ラジオ、チェック）の折り返し数の設定には対応していません。

項目の型について

- 複合主キーには未対応です。
- メールアドレス、URL、ファイルはリンク形式ではなく、文字情報として表示されます。
- ファイル型項目のアイコンはグリッドでは表示されません。
- 郵便番号から住所を求める機能は動作しません。 R8.3.1で利用できるようになります。

利用できない設定

- 一覧表示の「列固定」は無効です。（設定しても機能しません。）
- 複数段一行明細レイアウトを行うことはできません。
- 検索画面遷移直後の検索処理を止める設定と併用することはできません。
- 一覧表示項目に条件付き閲覧権限を指定することはできません。
- 一括処理を利用する場合「全データー括処理の対象とする」設定には未対応です。
- グリッド内に用意されるボタン（一覧更新画面へ遷移するボタン、帳票出力ボタンなど）に表示条件を指定することはできません。（指定すると意図しない動作になることがありますので、指定しないようにしてください。）
- 検索結果が1件のとき詳細表示画面に遷移する機能は未対応です。
- 親子モデルの表示で、子モデルをグリッド形式で表示することはできません。
- 数値型項目の出力フォーマット指定は無視されます。

絞り込み設定

他モデル参照項目の「入力制御>モデル参照(絞込)」設定は利用できません。 R8.3.1で利用できるようになります。

なお設定時は、"他モデルの参照 > 参照モデル同士の関係性で選択肢を絞り込む > トラブルシューティング > 一覧表示画面で絞込が動作しない"をお読みください。

パフォーマンス

グリッド表示では実行時に利用者が列の非表示設定を行うことができます。そのためモデル定義ではすべての項目の一覧表示設定を有効にして、利用者に表示列を自由に選択させるという運用を想定されるかも知れません。しかしその場合、表示対象データをいったんサーバで作成してブラウザに送出するためパフォーマンスが劣化します。

設計段階では、一覧表示画面で必要となる項目はできるだけ限定し、さらに利用者で表示選択させるというような方針としてください。

一括処理

一括処理とは

画面例

図に「一括削除」機能を有効にした一覧表示画面を示します。

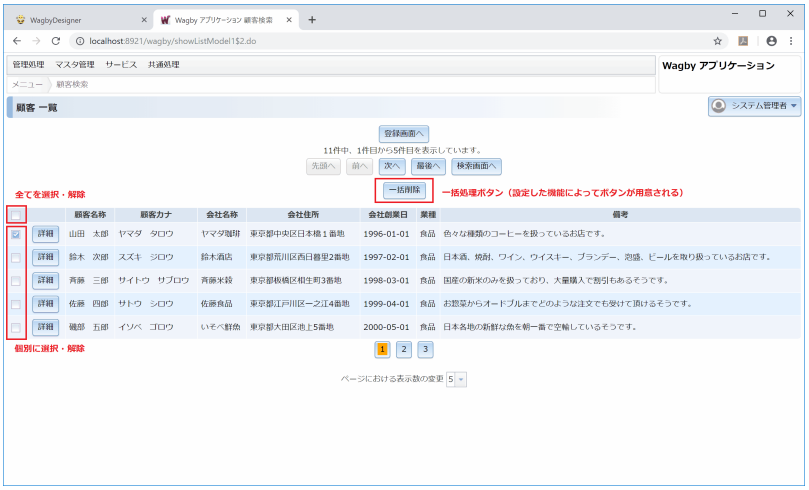


図 一括削除機能を有効にした一覧表示画面

複数のデータをチェックする（有効にする）ことで対象データを選択します。その後「一括削除」ボタンを押下することで、複数データをまとめて削除することができます。

ボタンの表示制御

一括処理系のボタン（一括削除・一括メール送信・一括帳票・一括更新）は、検索結果が存在する場合のみ表示されます。検索結果が存在しない場合、ボタンも非表示となります。

未選択時はボタンを押すことができない（8.1.2）

一括処理では処理対象のデータがチェックボックスで選択されるまで、一括処理のボタンは押下できません。

また押下不可となっている間は、ボタンにマウスオーバーすると ″一括処理対象のデータを選択してください。″ という Tips が表示されます。

ページネーションはボタン形式になる（8.2.2）

一括処理ではページネーションはボタン形式になります。

(211)

一括削除

「画面>検索・一覧表示>ボタン表示」を開きます。「一括削除ボタン」を表示する、と設定します。

(211) すべての選択肢の情報を Web フォームとして送信するためです。

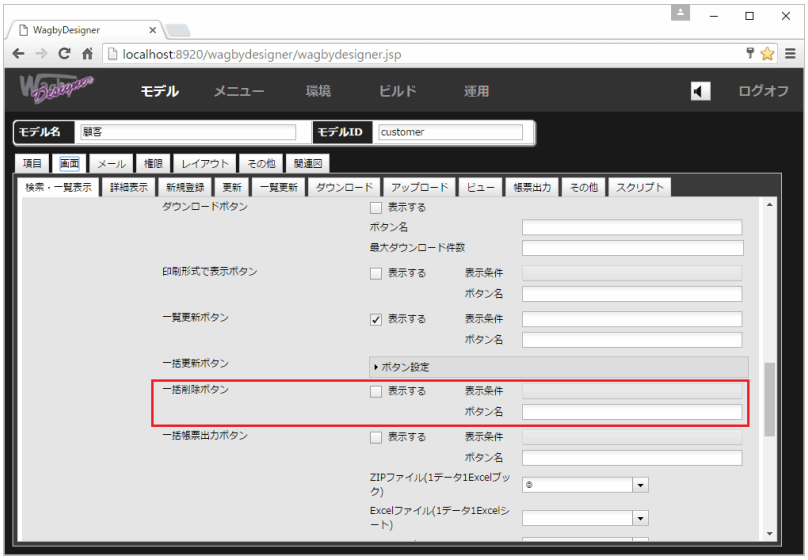


図 一括削除ボタンを表示する

一括更新

ボタンの設定

「画面>検索・一覧表示>ボタン表示」を開きます。「一括更新ボタン > ボタン設定」を押します。

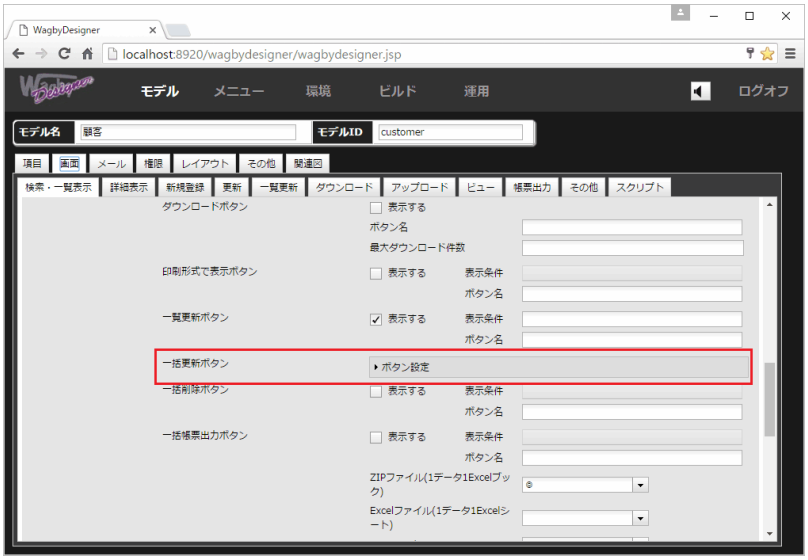


図 一括更新ボタンを設定する(1)

一括更新ボタンは最大で10個まで用意することができます。

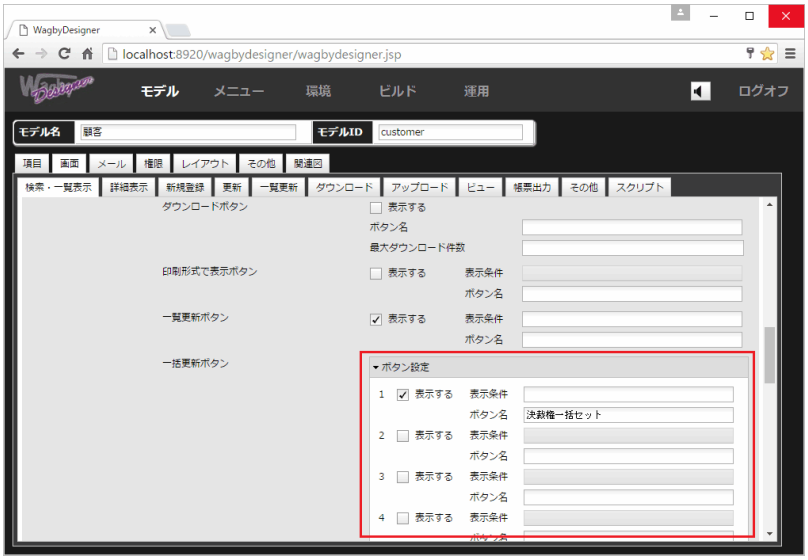


図 一括更新ボタンを設定する(2)

スクリプトの設定

「一括更新」ボタン押下時の処理をスクリプトで記述することができます。

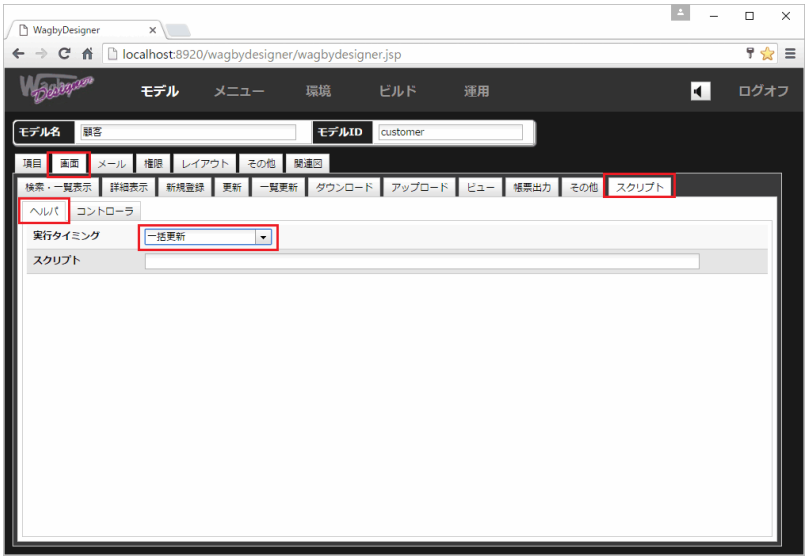


図 スクリプト記述欄

スクリプトの詳細は"一括更新"で説明します。

選択方式

Wagbyは一括処理について二通りの操作方法を提供しています。

「画面>検索・一覧表示>一覧表示画面」に用意された「一括処理の扱い方法」で指定できます。標準はこのリストボックスは空欄となっています。この場合は "画面表示データのみ一括処理の対象とする" と解釈されます。

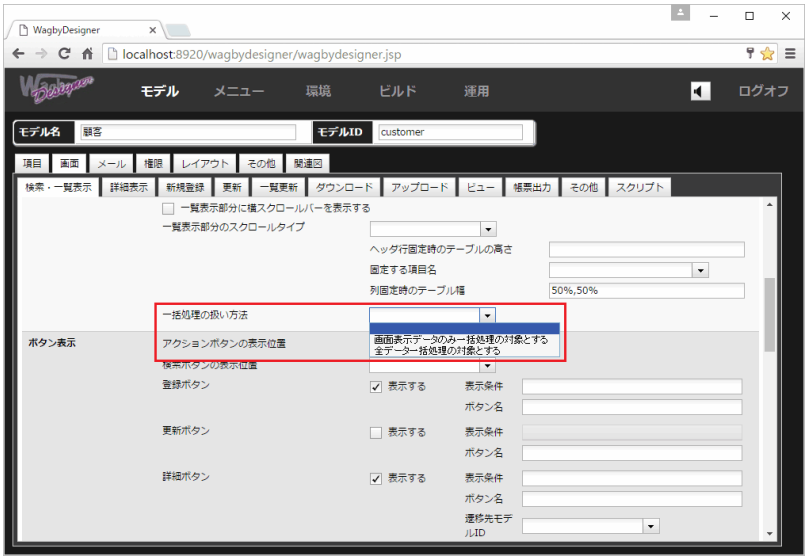


図 一括処理の扱い方法

画面に表示されているデータを一括処理の対象とする

一括処理の標準方式です。初期表示時は、チェックボックスが全て未選択の状態です。 ヘッダー部分に用意されたチェックボックスをクリックすることで、表示されているデータを全て選択することができます。チェックを外すと、チェックされていたデータのチェックも外れ、未選択状態となります。

初期表示時

データは選択されていません（チェックボックスは未選択状態）。 表示しているデータへの全選択／全解除を行うチェックボックスが用意されています。

データが複数のページにまたがっており、前後のページに移動した場合

チェックボックスの選択状態は記憶されません（ページ移動すると初期状態の未選択状態となります）。 これを図で説明します。

最初のページで、全てのデータを選択します。この状態で「次」のページへ移動します。

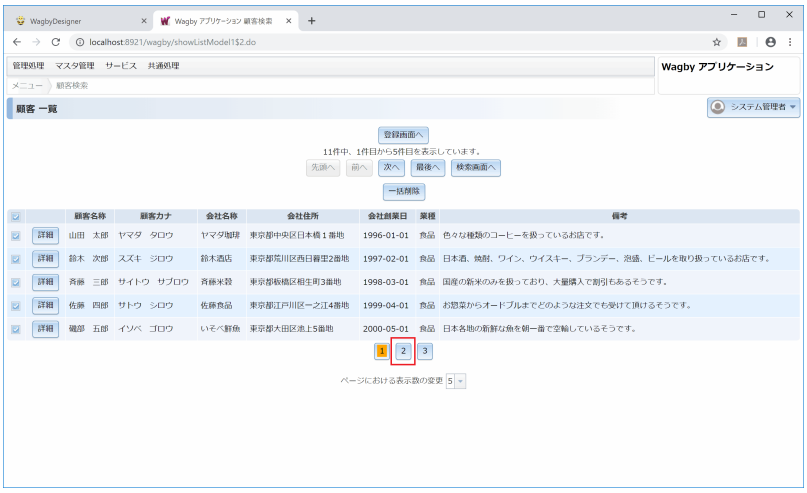


図 選択状態で、次ページへ移動する

ここで、前のページへ戻ります。

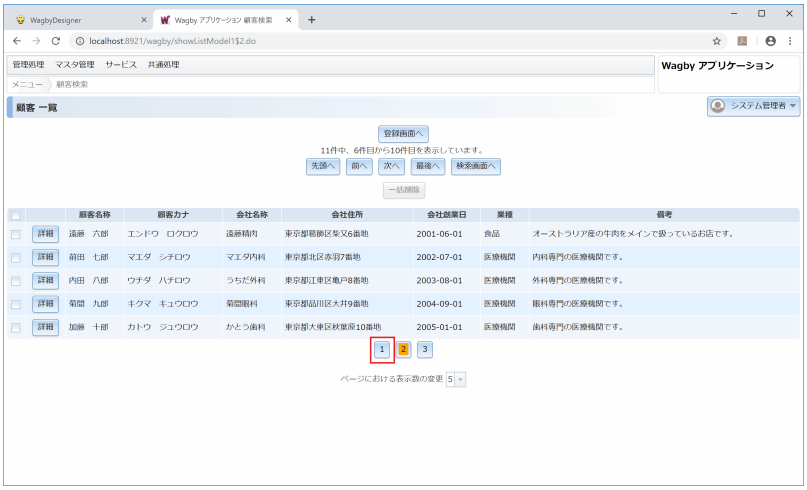


図 前のページへ戻る

図で選択していた状態は解除されます。

図 選択状態は解除される

処理対象データ

表示しているページのチェックボックスで選択されているデータのみが対象となります。 複数ページにまたがるデータがある場合、表示されていないページは対象となりません。（表示されていないページは未選択状態として扱います）。

全データを一括処理の対象とする

画面表示時に、全データがチェックされた状態となります。 データが複数のページにまたがっている場合でも、全てのページのデータがチェックされた状態になっています。 例えばこの状態で「削除」を行うと、画面に表示されていないデータもすべて削除されます。

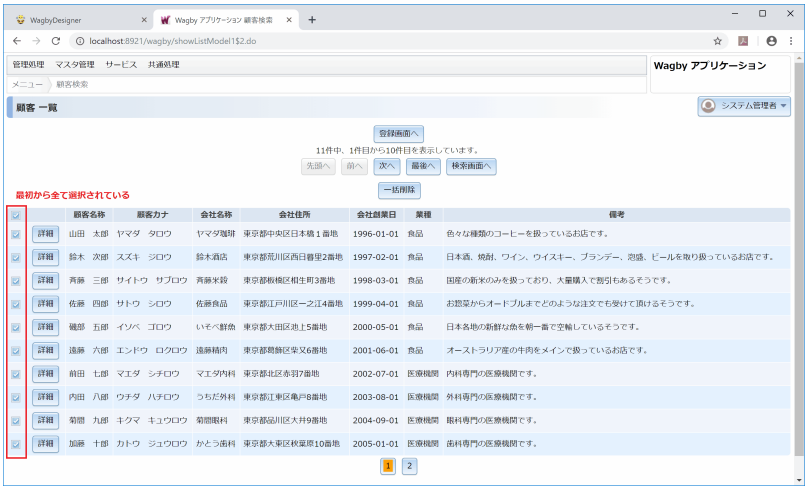


図 画面表示時にすべてが選択されている（すべてのページにまたがって選択されている）

ヘッダー部分に用意されたチェックボックスをクリックすることで、表示されているデータを全て選択することができます。 チェックを外すと、チェックされていたデータのチェックも外れ、未選択状態となります。

初期表示時

全てのデータが選択されています。 複数ページにまたがるデータがある場合、表示されていない他のページも全て選択されています。

データが複数ページあり、前後のページに移動した場合

チェックボックスの選択状態は記憶されます（ページを移動しても前に選択した状態を保持します）。

処理対象データ

チェックボックスで選択されている全てのデータ。 複数ページにまたがるデータがある場合、表示されていないページのデータもチェックボックスで選択されているものは対象となります。

「全データ一括処理の対象とする」設定は、グリッド形式表示では未対応です。

グリッドでの設定

グリッド表示で、データを選択するためのチェックボックス欄を用意することができます。次の図は一括削除を行う例です。

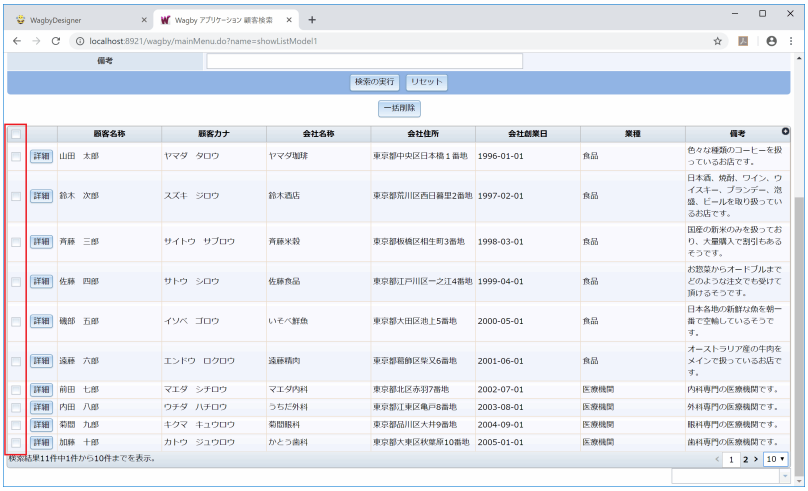


図 チェックボックス欄が用意された

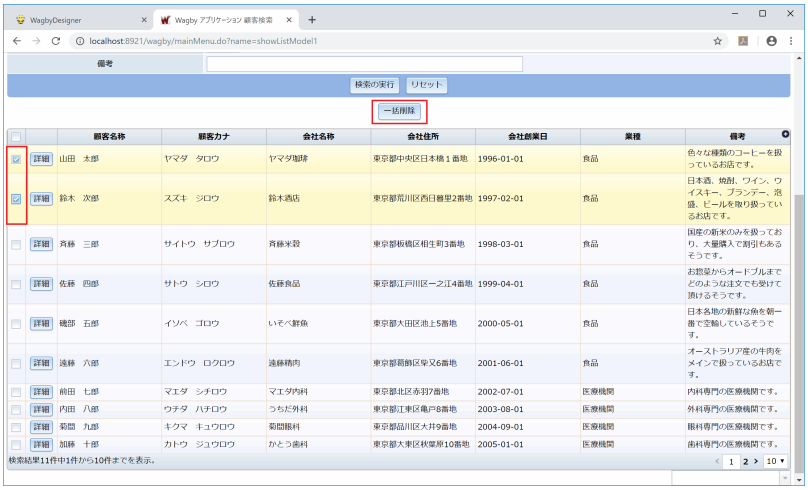


図 複数データを一括削除する例

定義方法

一括選択を利用する時は「一覧表示の行クリックで詳細表示画面に遷移する」設定は無効としてください。

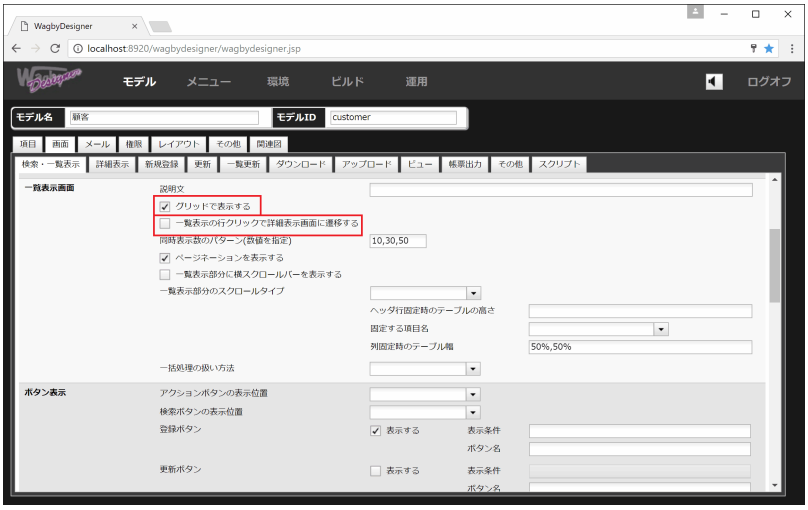


図 「一覧表示の行クリックで詳細表示画面に遷移する」設定を無効とする

ここでは一括削除を有効にしています。

図 一括削除を有効にする

仕様・制約

- 一括選択を利用する時は「一覧表示の行クリックで詳細表示画面に遷移する」設定は無効としてください。
- 「全データー括処理の対象とする」設定は、グリッド形式表示では未対応です。

一覧表示の集計

例

各項目の最大値、最小値、平均値、合計値、個数を一覧表示画面の集計欄に表示することができます。

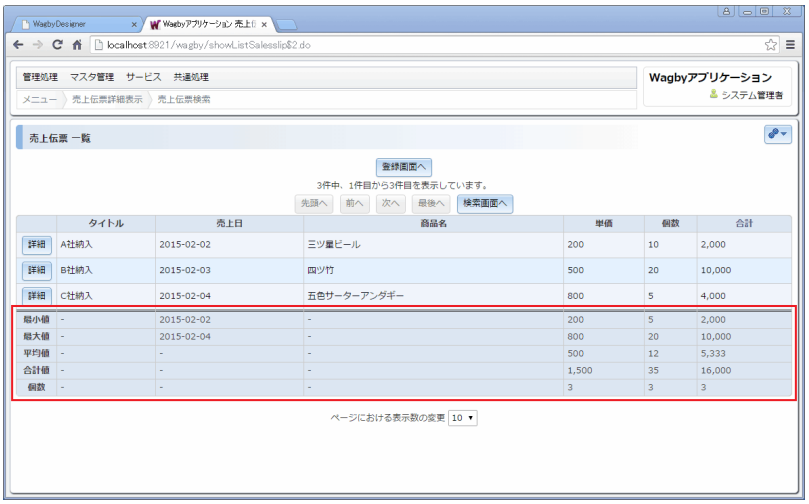


図 集計欄の表示例

定義方法

図で示したサンプルアプリケーションのモデル定義を示します。 集計欄に出力する項目を、一覧表示の対象とします。

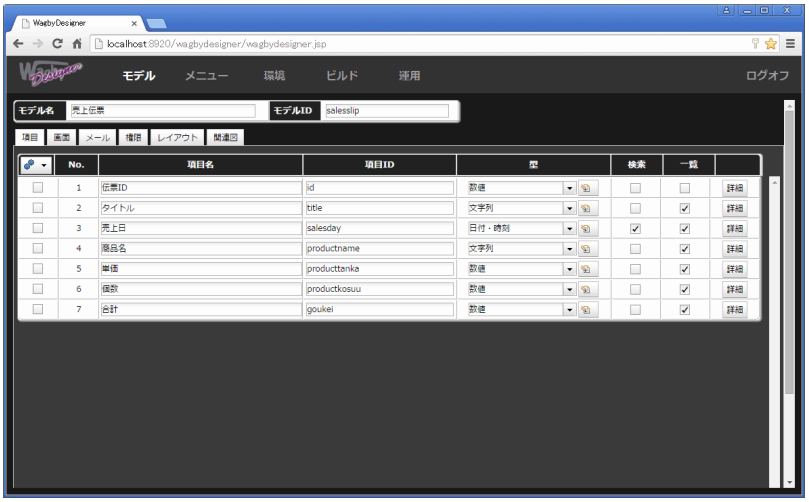


図 サンプルアプリケーションのモデル定義

集計欄の設定

モデル項目詳細定義の「出力制御 ＞ 一覧画面 ＞ 集計欄」を指定します。

「最小値」「最大値」「平均値」「合計値」「個数」を指定できます。 項目の型によって指定できる内容が異なります。（後述） 図は日付型の例です。

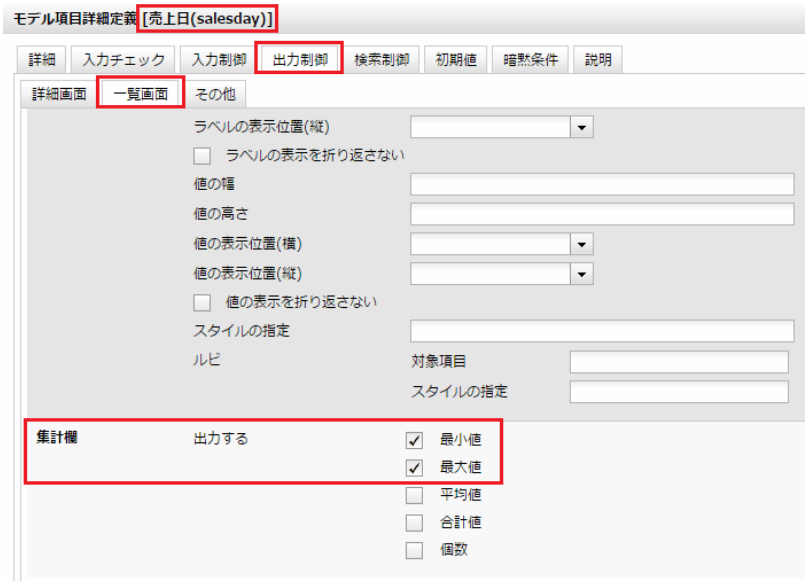


図 日付型項目に「最小値」「最大値」を指定する

図は数値型の例です。

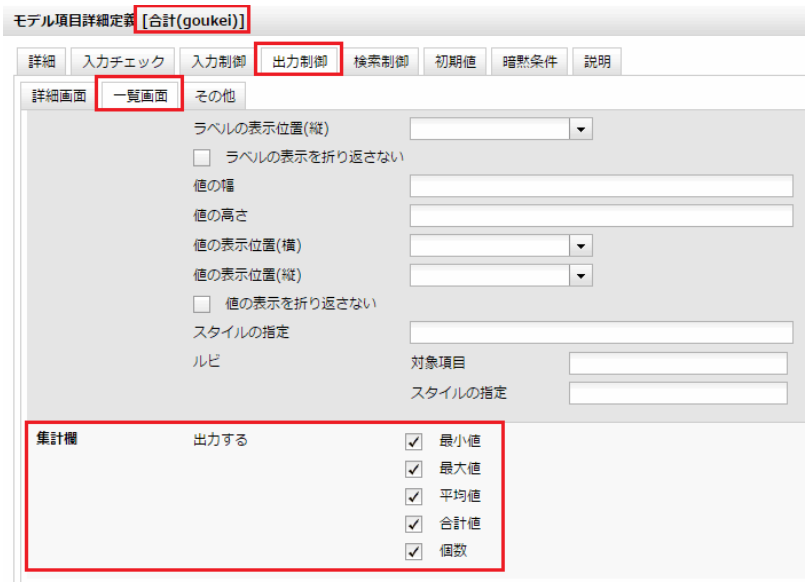


図 数値型項目に「最小値」「最大値」「平均値」「合計値」「個数」を指定する

数値の表示フォーマット

集計の切り口ごとに数値フォーマットを指定することができます。フォーマットの指定方法は"数値 ＞ 出力フォーマット"をお読みください。

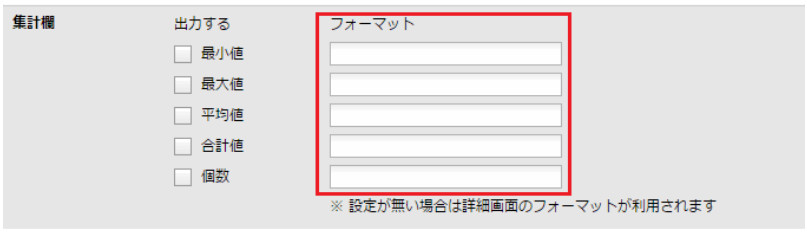


図 数値フォーマットの指定

上のフォーマットが未指定（空白）の場合、集計欄に表示される数値は対象の項目の数値フォーマットが反映されます。

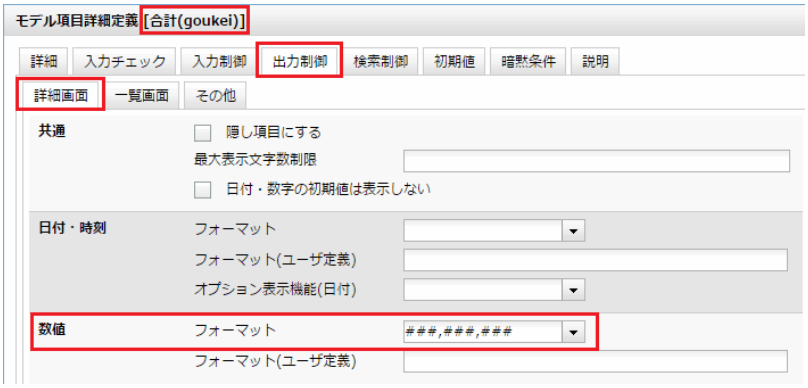


図 その項目自体の数値フォーマットが利用される

対応する関数

関数 AGCOUNT, AGSUM, AGAVG, AGMAX, AGMINを使って、画面に表示された値を取得することができます。

仕様・制約

利用できる型

項目の型によって利用できる演算が定まります。

項目の型	最小値	最大値	平均値	合計値	個数
文字列	○	○	×	×	○
整数・小数	○	○	○	○	○
日付・時刻	○	○	×	×	○
モデル参照	×	×	×	×	×
(繰り返し項目のコンテナ)	×	×	×	×	×
(繰り返し項目コンテナ用のID)	×	×	×	×	×

未対応の項目

本集計機能は、以下の項目には対応していません。（定義を行っても無視されます。）

- 繰り返し項目
- 繰り返しコンテナ内の項目
- 「値をデータベースに保存する」に「○」が設定されていない項目

集計対象が空の場合

集計対象項目の値が「空」（値なし）の場合、最小・最大・平均・合計値は空白で表示されます。個数は"0"と表示されます。

内蔵データベース利用時の制約

Wagby に同梱されている内蔵データベース（HSQLDB）を利用した場合、平均値は整数値のみとなります。

セル結合（行）利用時の制約

"セル結合（行）"と併用する場合、レイアウト上の制約があります。

その他

本機能は一覧表示（グリッド）には対応していません。

詳細表示

ページタイトル

画面例

ページタイトルを"お得意様 最新情報"へ変更した例を示します。

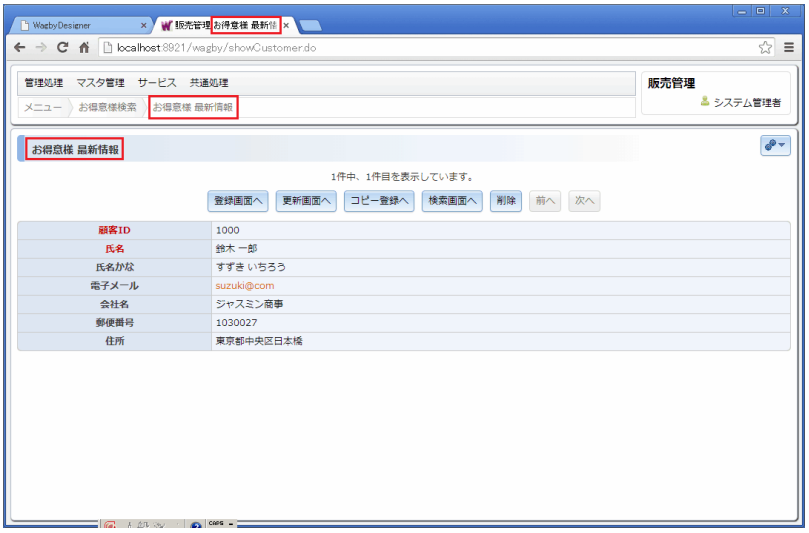


図 ページタイトルを変更

定義方法

画面タブ内の検索・一覧表示タブを開きます。「ページタイトル」の欄に文字を入力します。

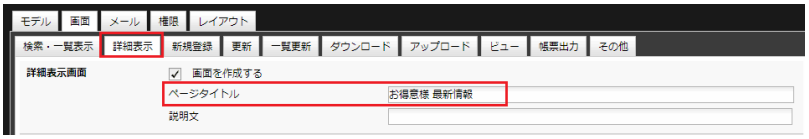


図 ページタイトルの設定

説明文

画面例

説明文を設定した例を示します。

図 説明文を表示する

定義方法

画面タブ内の検索・一覧表示タブを開きます。「説明文」の欄に文字を入力します。

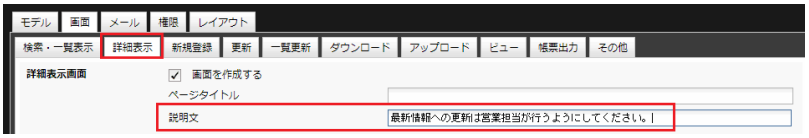


図 説明文の設定

HTMLタグを加える

説明文にHTMLタグを加えることもできます。詳細は「検索・一覧表示>説明文 － HTMLタグを加える」をお読み下さい。

アクションボタンの表示位置

（画面遷移を行うための）アクションボタンの位置を変更することができます。

上部に配置する

図 アクションボタンを上部に配置した例（標準）

下部に配置する

図 アクションボタンを下部に配置した例

上部と下部の両方に配置する

図 アクションボタンを上部と下部の両方に配置した例

定義方法

画面タブ内の検索・一覧表示タブに用意された「ボタン表示>アクションボタンの表示位置」を設定します。空白時は「上」として扱います。

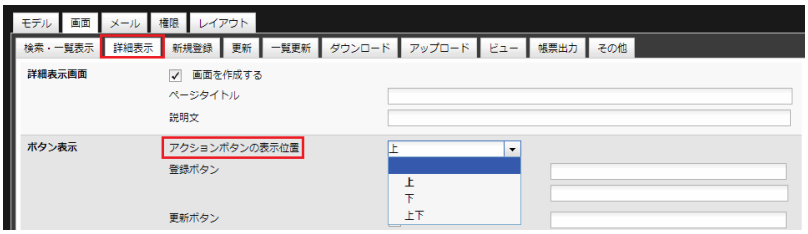


図 アクションボタンの表示位置の設定

用意されているボタン（機能）

詳細表示画面に用意できるボタン（機能）を説明します。

図 詳細表示画面に用意できるボタン

定義方法

画面タブ内の検索・一覧表示タブに用意された「ボタン表示」を有効にします。

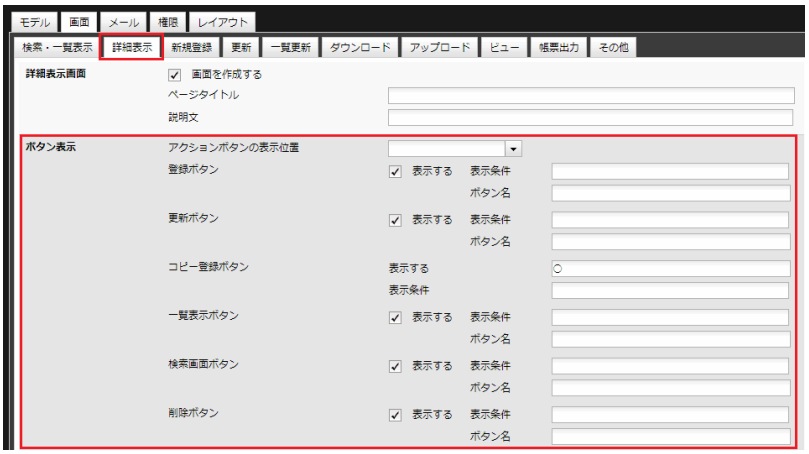


図 ボタン表示の設定

各ボタンの表示を有効にする場合、関連する画面機能が作成されている必要があります。 画面機能が作成されない設定では、ボタンも作成されません。

種別	初期状態	関連する画面
登録	○	登録画面
更新	○	更新画面
コピー登録	- 「表示する」設定欄に「○」と入力するとボタンが一つ用意される。 同欄にコンマ区切りで複数のボタン名を記載することもできる。	新規登録画面
一覧表示	○	検索・一覧表示画面
検索画面	○	検索・一覧表示画面
削除	○	（なし）
印刷形式で表示	-	（なし）
前後のデータにアクセス	○	（なし）

ボタンの表示制御

「ボタン名と表示条件のカスタマイズ」で説明します。

前後のデータにアクセスする

詳細表示画面は標準で前後のデータにアクセスする機能が備わっています。 カード型データベースの感覚で、前へ、次へといった操作が可能です。

ここで「前・後」とは、現在の検索結果に対する操作となります。検索条件や並び順の指定によって、前・後のデータは変動します。

本機能はパフォーマンスに影響があります。詳細は"パフォーマンスチューニング > 前後のデータにアクセスするボタンの動作"をお読みください。

仕様・制約

この前後にアクセスするボタンは、検索処理を行うことによって機能します。（検索結果に対する前後への移動を行います。）

このため、次のような場合はボタン操作が行えません。（ボタンが押せない状態となります。）

・　親子モデルの親モデルに用意された子モデル一覧表示で詳細ボタンを押し、子モデルを表示した場合。（指定した子モデルの主キーのデータを直接表示するため検索が行われていない。）
・　カレンダービューから詳細画面に遷移した場合。上と同じ理由による。

トラブルシューティング

前・次ボタンで意図しないデータが表示される

主キー項目の〝初期値 ＞ 詳細画面表示時〟に式または値を設定している可能性があります。

関連する機能

・　更新 ＞ 前後のデータにアクセスする

新規登録

ページタイトル

画面例

ページタイトルを〝お得意様 ご登録〟へ変更した例を示します。

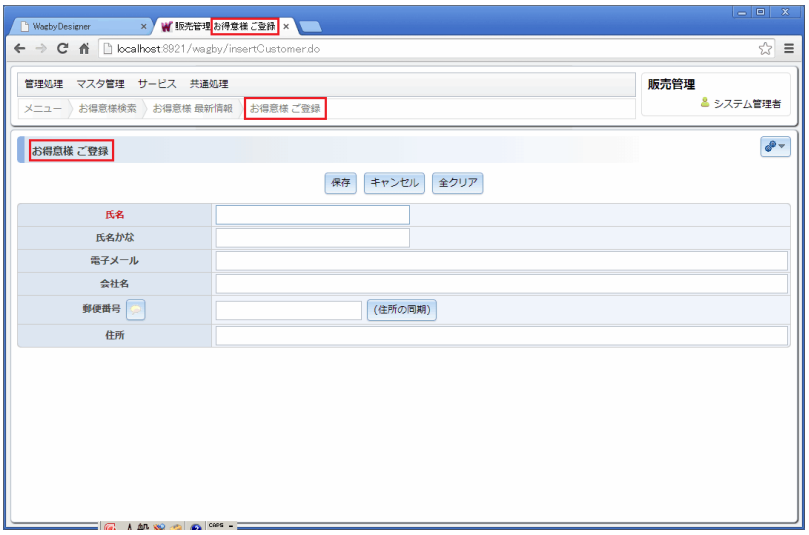


図 ページタイトルを変更

定義方法

画面タブ内の新規登録タブを開きます。「ページタイトル」の欄に文字を入力します。

図 ページタイトルの設定

説明文

画面例

説明文を設定した例を示します。

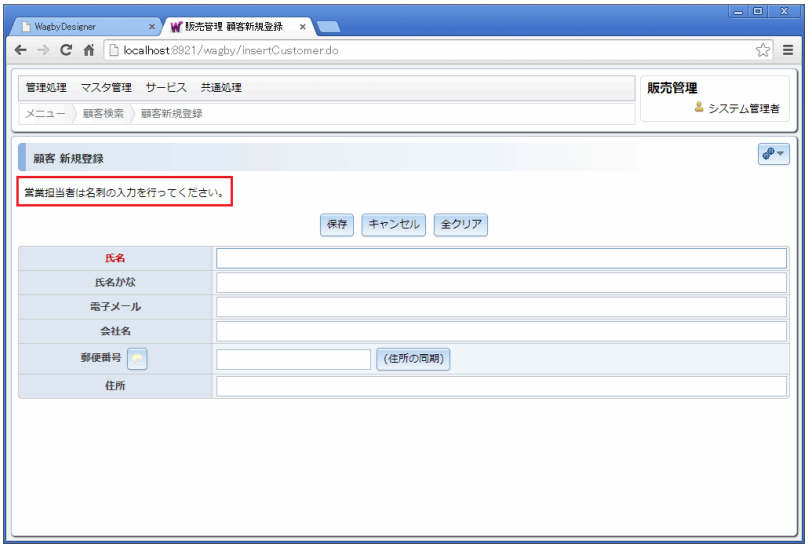


図 説明文を表示する

定義方法

画面タブ内の新規登録タブを開きます。「説明文」の欄に文字を入力します。

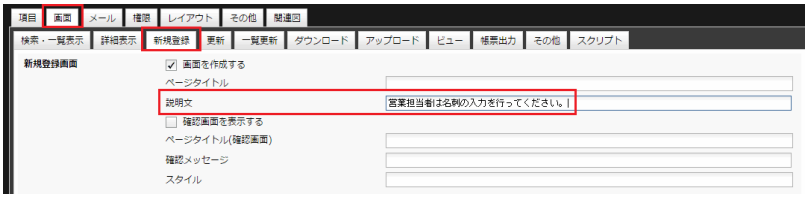


図 説明文の設定

HTMLタグを加える

説明文にHTMLタグを加えることもできます。文章を途中で改行させる場合は br タグを使うということができます。詳細は「検索・一覧表示>説明文 - HTMLタグを加える」をお読み下さい。

確認画面を用意する

例

登録時に確認を行うことができます。「保存」ボタンに代わって「登録確認」ボタンが用意されます。

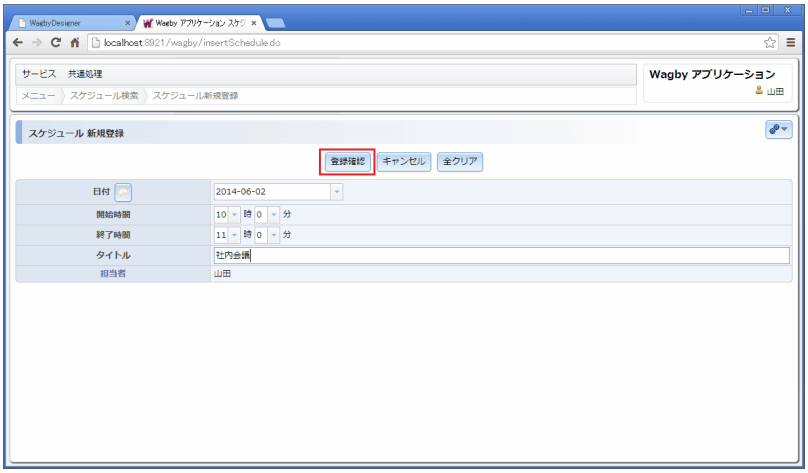


図 登録確認ボタン

登録確認ボタンを押下すると、「このデータを登録します。よろしいですか？」という確認メッセージが表示されます。ここで「登録する」ボタンを押下すると、データが登録されます。「登録画面へ戻る」ボタンを押下すると図の画面へ戻ります。

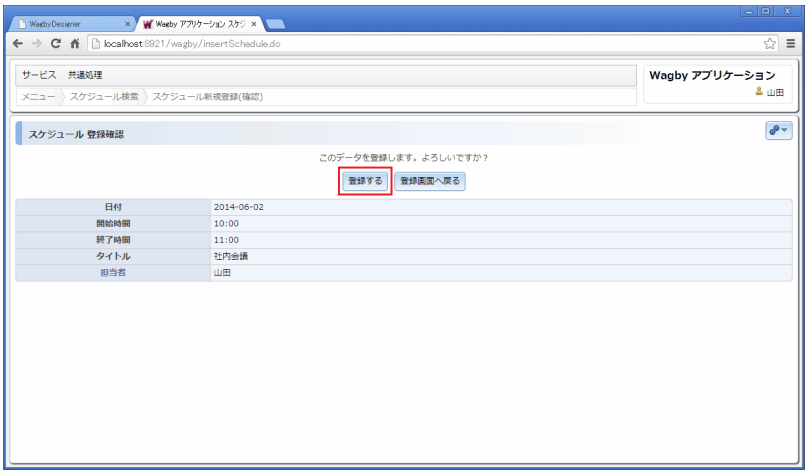


図 確認メッセージ

図 データ登録の完了

定義方法

「確認画面を表示する」設定を有効にします。

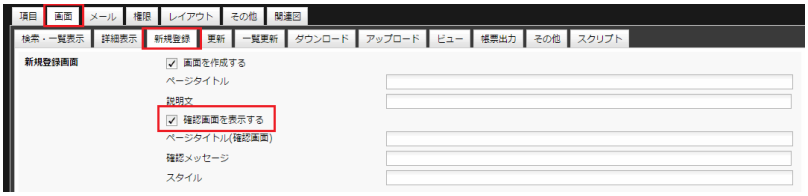


図 確認画面を表示する

確認画面用のページタイトルを指定する

確認画面に用意されるページタイトルを指定することができます

図 ページタイトルの設定（確認画面用）

通常の登録画面ページタイトルを設定し、本欄を空白とした場合は、確認画面用のページタイトルに登録画面ページタイトルの設定が流用されます。

確認メッセージとスタイルを指定する

確認画面に用意されるメッセージとスタイルを変更することができます。

図 確認メッセージとスタイルを指定する

図の設定を反映した画面を図に示します。

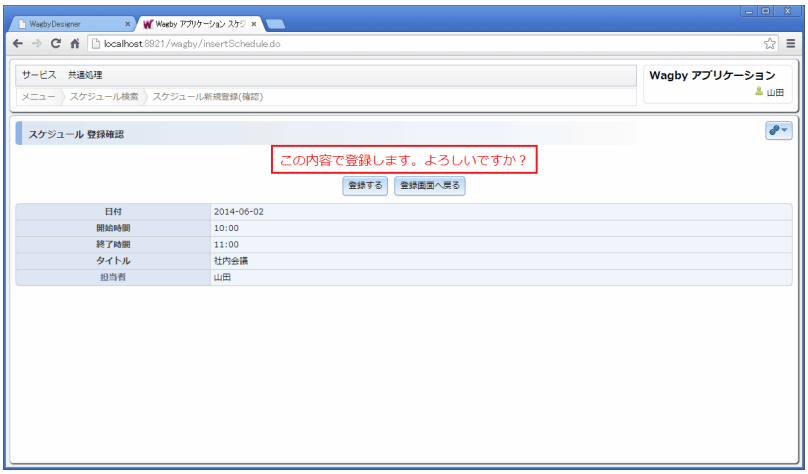


図 オリジナルの確認メッセージ

仕様・制約

- 確認画面は、詳細画面を再利用して実現しています。確認画面を用意する場合は、同時に詳細画面も作成してください。
- 確認画面の挙動は、詳細画面と同じになります。
- 子モデルの同時更新機能を用いる場合、確認画面を設定することはできません。（子モデル同時更新と確認画面は併用できません。）

アクションボタンの表示位置

（画面遷移を行うための）アクションボタンの位置を変更することができます。

上部に配置する

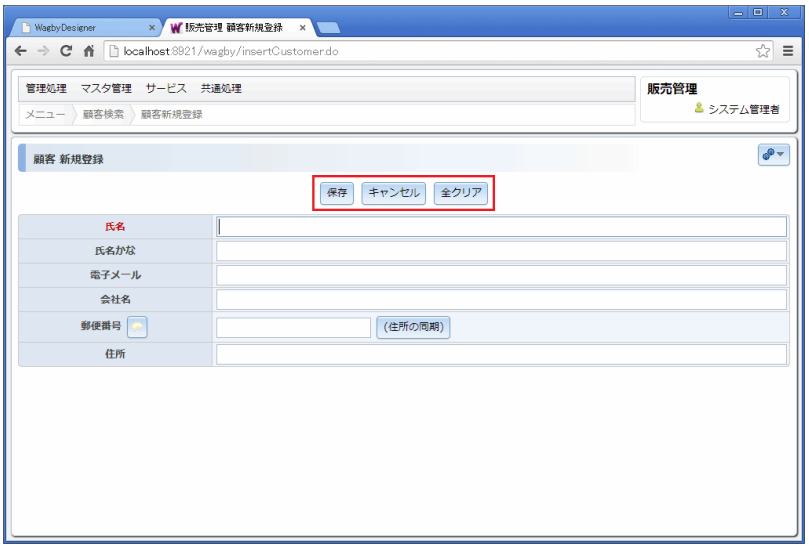


図 アクションボタンを上部に配置した例（標準）

下部に配置する

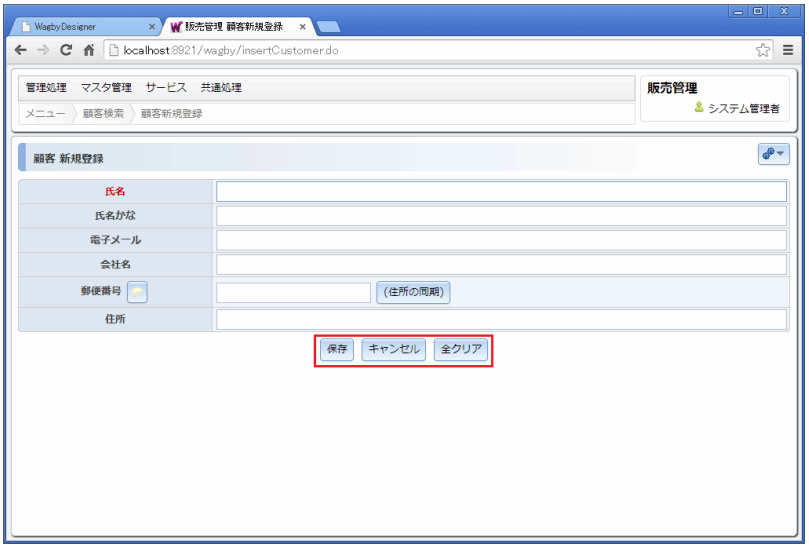


図 アクションボタンを下部に配置した例

上部と下部の両方に配置する

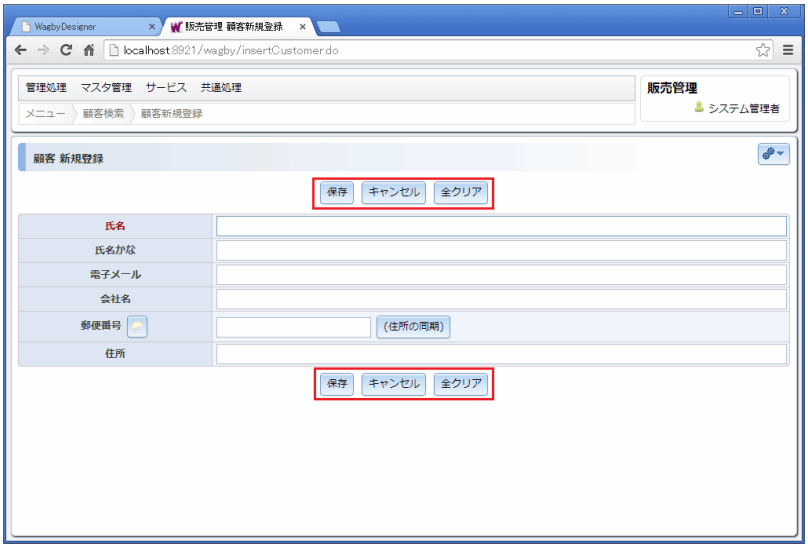


図 アクションボタンを上部と下部の両方に配置した例

定義方法

画面タブ内の検索・一覧表示タブに用意された「ボタン表示>アクションボタンの表示位置」を設定します。空白時は「上」として扱います。

図 アクションボタンの表示位置の設定

用意されているボタン（機能）

新規登録画面に用意できるボタン（機能）を説明します。

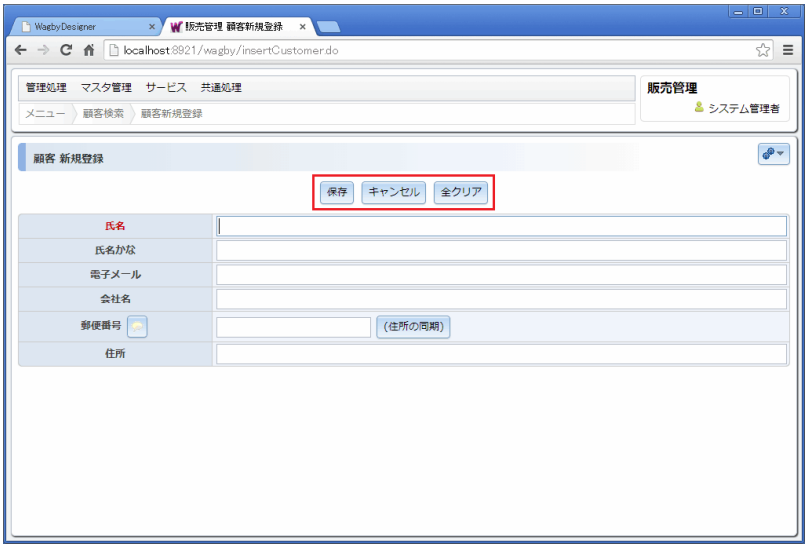


図 新規登録画面に用意できるボタン

定義方法

画面タブ内の新規登録タブに用意された「ボタン表示」を有効にします。

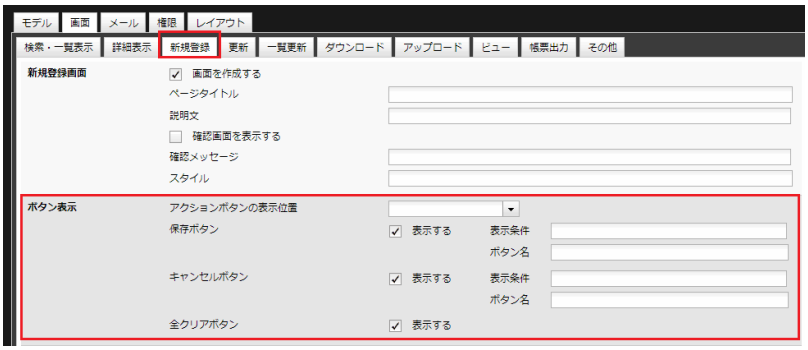


図 ボタン表示の設定

各ボタンの表示を有効にする場合、関連する画面機能が作成されている必要があります。 画面機能が作成されない設定では、ボタンも作成されません。

種別	初期状態	備考
保存	○	
キャンセル	○	
全クリア	○	入力した項目をすべて消去します。（データベースからデータが消去されるわけではありません。）

ボタンの表示制御

〝ボタン名と表示条件のカスタマイズ〟をお読みください。

全クリアボタン押下時の読込専用項目の扱い

- 自動計算によって求められた値は、全クリアボタン押下時に、再計算されます。
- 初期値がセットされた値は、全クリアボタン押下時に、初期値を再セットします。

キャンセル操作時のダイアログ

登録画面に用意された「キャンセル」ボタンを押下すると、図に示す確認ダイアログが開きます。

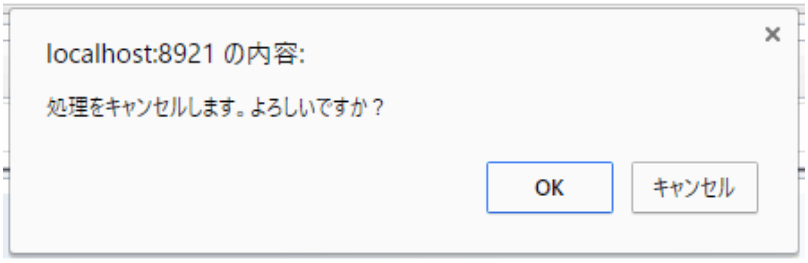


図 キャンセル操作の確認

図で「OK」ボタンを押下すると処理がキャンセルされます。「キャンセル」ボタンを押下すると元の画面へ戻ります。

ここで、いったん確認ダイアログの「キャンセル」ボタンを押下したあと、再び（登録画面の）「キャンセル」ボタンを押下すると、ご利用の Web ブラウザによっては確認ダイアログに「このページでこれ以上ダイアログボックスを生成しない」というメッセージが表示されます。

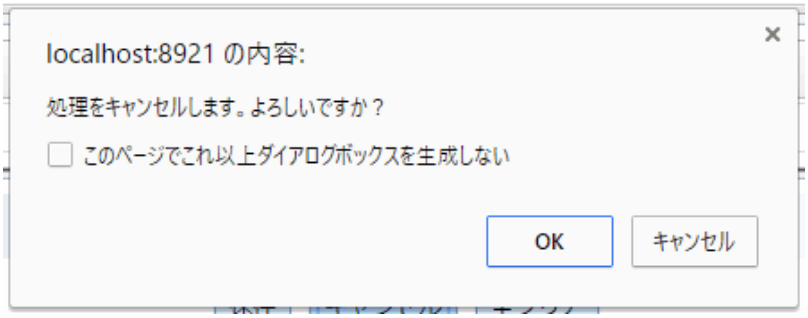


図 二回目のキャンセル操作確認ダイアログ

(212)

ここで、「これ以上ダイアログボックスを生成しない」とした場合、Web ブラウザは以後、ダイアログを表示せずに自動的に「いいえ」を選択した状態で動作するようになります。この動作は望ましくないため、「このページでこれ以上ダイアログボックスを生成しない」はチェックしないよう、ご注意ください。

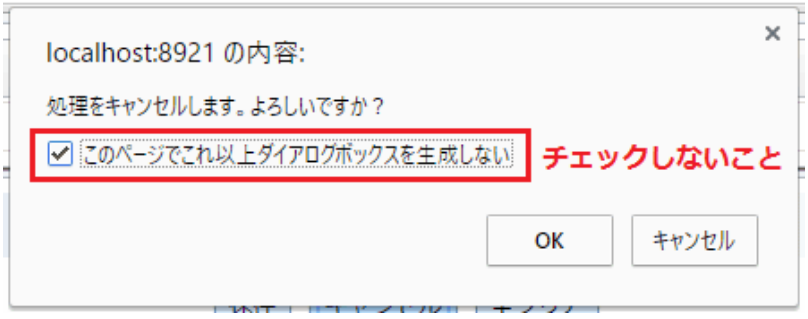


図 「これ以上ダイアログボックスを生成しない」はチェックしないこと

(212) 「このページでこれ以上ダイアログボックスを生成しない」メッセージはブラウザが表示しているもので Wagby で制御することができません。

仕様

- 新規登録（およびコピー登録）画面は、Post/Redirect/Get （PRG） パターンを適用しています。これにより、フォームデータの二重送信（ブラウザの画面リロードによって二重にフォームデータが登録されること）を防止しています。

コピー登録

コピー登録画面を用意する

詳細画面または一覧画面にコピー登録ボタンを用意することができます。

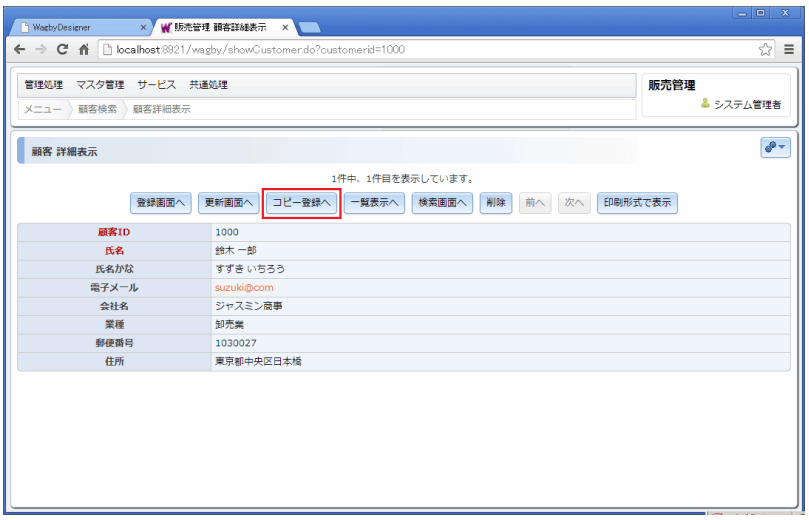


図 詳細画面に用意されたコピー登録ボタン

図 一覧表示画面に用意されたコピー登録ボタン

定義方法 - 一覧表示・詳細表示

画面設定の「検索・一覧表示」「詳細表示」で、それぞれコピー登録ボタンの「表示する」欄に true と記述します。

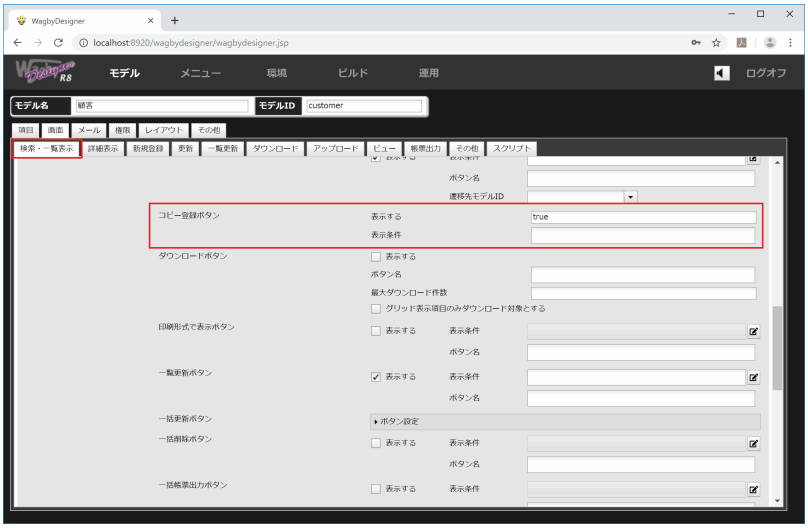


図 コピー登録ボタンを用意する（一覧表示）

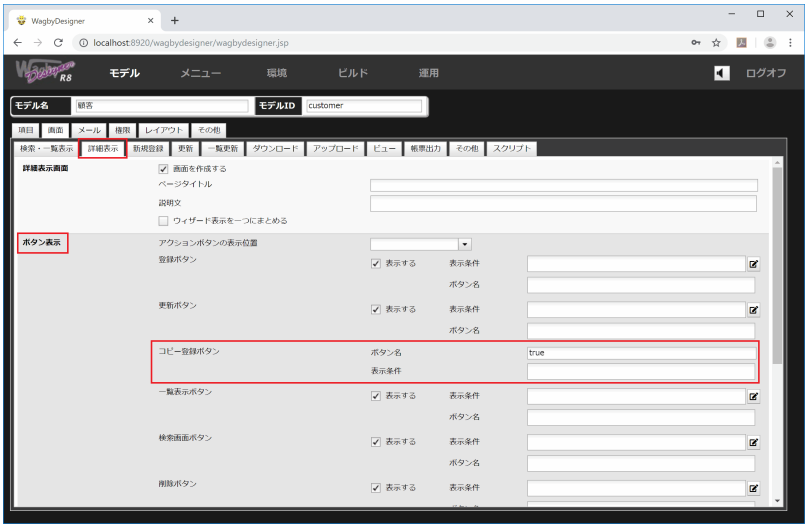


図 コピー登録ボタンを用意する（詳細表示）

定義方法 - 一覧更新

画面設定の「一覧更新」で、コピー登録ボタンを有効にします。（ボタンは1つのみ）

図 コピー登録ボタンを用意する（一覧更新）

なお、本機能を使用する場合は「新規登録画面」を作成する必要があります。

図 新規登録画面を作成する設定

キャンセル操作時のダイアログ

〝新規登録 〉キャンセル操作時のダイアログ〟をお読みください。

コピー対象項目を指定する

コピー登録を有効にしたモデルでは、原則としてすべての項目がコピー登録の対象となります。 これを項目毎に（コピーさせないように）制御することができます。

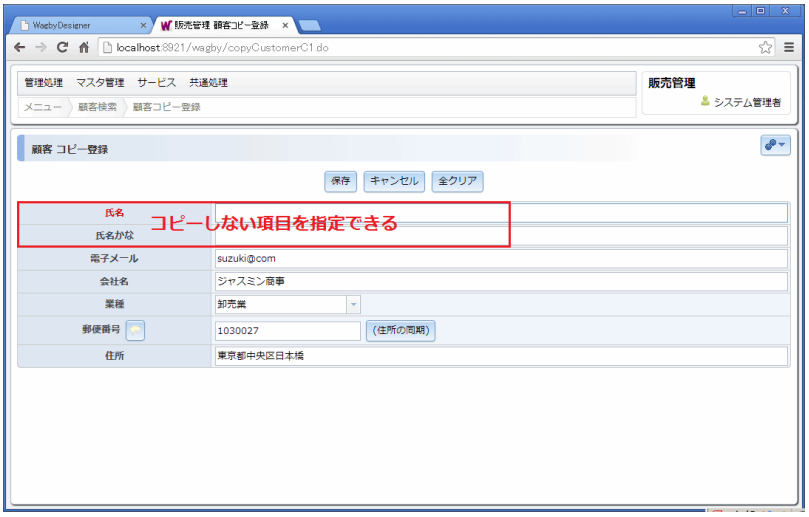


図 「氏名」「氏名かな」をコピー対象外とした例

定義方法

各項目のモデル項目詳細定義ダイアログを開きます。「コピー登録時に値をコピーする」のチェックを解除します。（標準では有効になっています。）

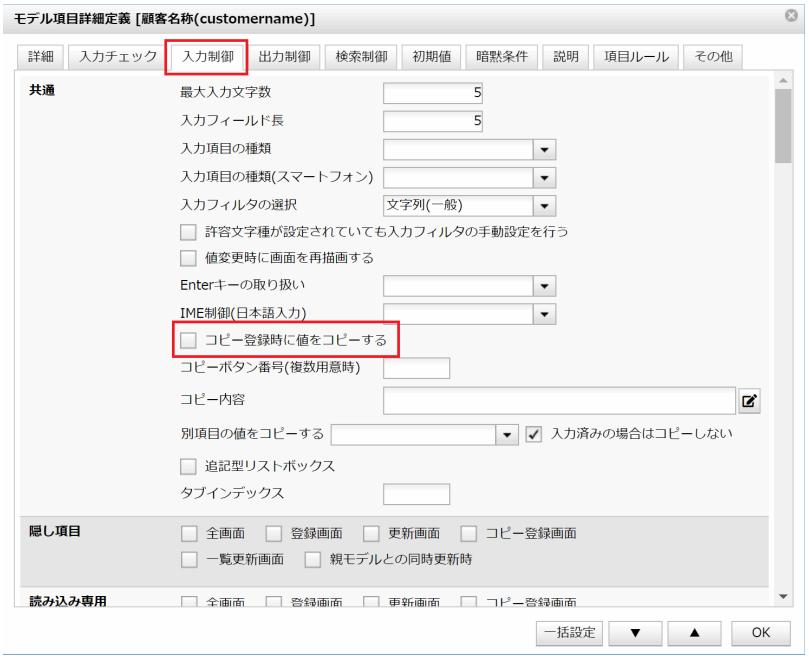


図 コピー登録時に値をコピーする設定（チェックを解除した状態）

複数のコピーボタンを用意する

コピーボタンを複数用意することができます。図のように、ボタン名を ″,″ で区切って複数、定義します。（″,″ は半角文字です。）

図 コピーボタンを複数用意する

複数コピーボタン用意時の項目のコピー設定（8.3.2）

ボタン毎に、項目のコピー設定を行うことができます。 「入力制御>コピー登録時に値をコピーする」に 1 と記述すると、1番目のボタン「氏名コピーあり」を指します。 つまり″氏名コピーあり″ボタンをクリックした際にコピーされるという意味になります。

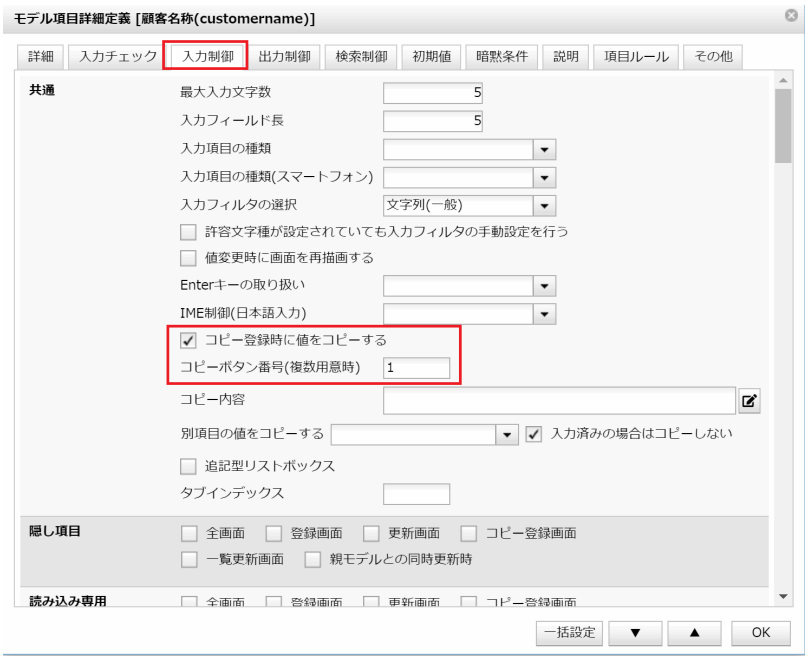


図 コピーボタンに対応した制御

ボタン名ではなく、コピーボタンの番号（並びの順序）を指定してください。複数ある場合はコンマで区切ります。（例 1,2 と記述します）

コピーボタン用意時の表示条件の設定

コピー登録ボタンを用意した場合、それぞれの式を二重引用符で囲み、コンマで区切って記述します。

次の例は、二つのボタンそれぞれに条件式を指定したものです。

```
"EXACT(USERNAME(),"admin")", "EXACT(USERNAME(),"manager")"
```

二番目のボタンは条件なし、という場合は次のように記述します。

```
"EXACT(USERNAME(),"admin")", "true"
```

または条件を記述しない、ということもできます。 次の例は二番目のボタンは条件なし("true") と扱われ、常に表示されます。（ただしボタンが複数あるため、二重引用符で囲む必要があります。）

```
"EXACT(USERNAME(),"admin")"
```

コピーボタンが1つの場合でも、表示条件式の前後は二重引用符で囲うようにしてください。

コピー時に式を適用する

コピー処理に式を適用することもできます。 図は、項目名の前後に括弧を加えてコピーした例です。

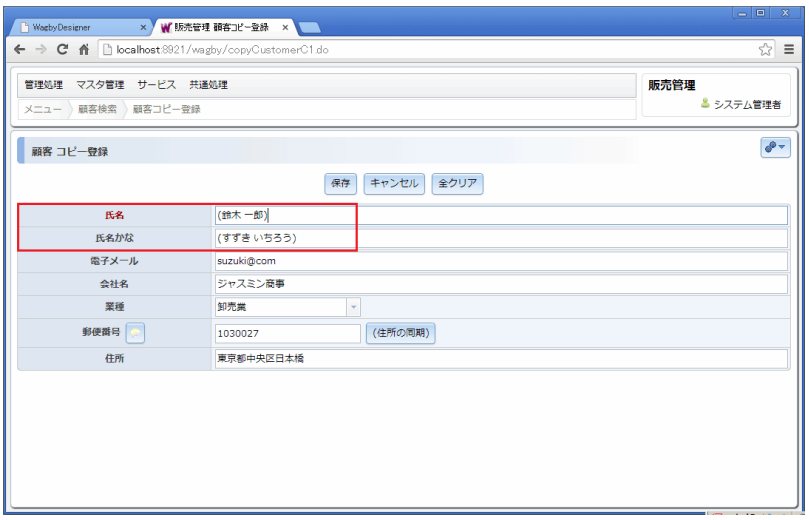


図 コピー処理に式を適用した例

定義方法

「入力制御>コピー登録時に値をコピーする」の「コピー内容」に式を記述します。

```
"(" + ${name} + ")"
```

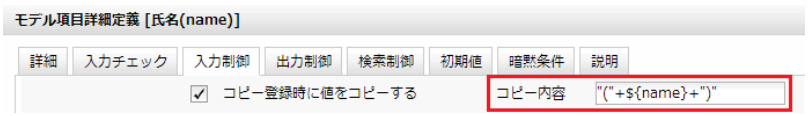


図 式の設定

計算式の記述方法の詳細は「計算式 > 基本的な書き方」をお読み下さい。

(213)

モデル参照項目をコピーする

コピー内容に記述する式に「#id」を付与すると、モデル参照項目をコピーすることができます。item1 項目の例を示します。

```
${item1#id}
```

(213) 複数のコピーボタンがあった場合、「コピー内容」の設定はすべてのコピーボタンで同一動作となります。ボタンによってコピー内容を変更したい場合はスクリプトをご利用ください。

ページタイトル

画面タブ内の新規登録タブを開きます。「ページタイトル(コピー登録画面)」の欄に文字を指定してください。

説明文

画面タブ内の新規登録タブを開きます。「説明文(コピー登録画面)」の欄に文字を指定してください。

アクションボタンの表示位置

新規登録画面のアクションボタン表示位置が、コピー登録画面にも反映されます。 詳細は「新規登録>アクションボタンの表示位置」をお読み下さい。

用意されているボタン（機能）

新規登録画面に準拠します。 詳細は「新規登録>用意されているボタン（機能）」をお読み下さい。

外部キー子モデルを含む場合

子モデルの同時更新 > コピー登録をお読みください。

なお、子モデル一覧更新の初期表示数指定はコピー登録画面には適用されません。 （これは繰り返しコンテナと同じ動きです。）

コピー登録時に枝番を付与する

"複合キーによる枝番"をお読みください。 主キーの手動入力パターンと、順序（自動採番）利用パターンの二通りの方法があります。

更新

悲観ロック

Wagbyの標準は「悲観ロック」です。ここでは悲観ロック時の挙動を説明します。

ある利用者 (A) がデータの更新画面を開いている場合、その更新作業が終わるまで、他の利用者 (B) は同じデータの更新画面を開くことができません。 この場合、利用者 B が（そのデータの）更新画面を開こうとすると図のようなメッセージが表示されます。

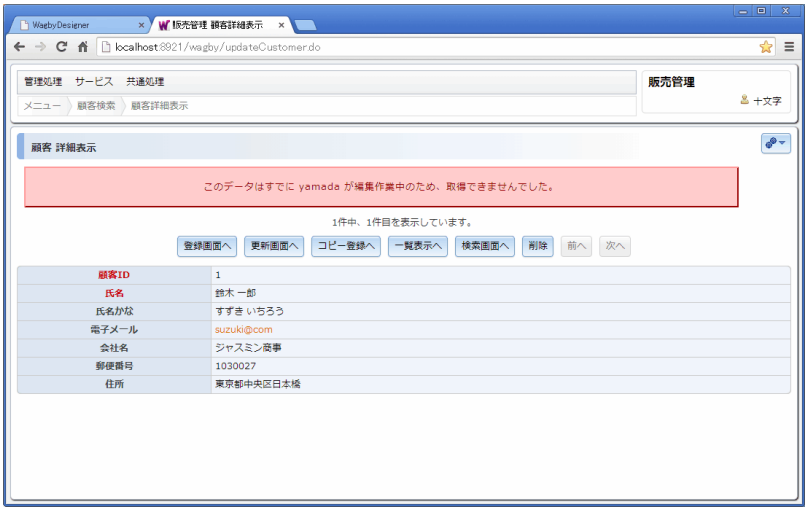


図 ロックエラーのメッセージ

このため、Wagbyでは更新の競合によるデータの破壊という事故は発生しません。最初に更新画面を開いた人が作業を終了するまで、他の人が更新することができないためです。

このロック機構は、更新・削除・一覧更新・アップロード更新といった、Wagbyが管理するすべての更新処理で機能します。

更新画面を開いたまま放置された場合

利用者（A）が更新画面を開いたまま、何らかの理由で放置したとします。この場合、A はある時間が経過すると自動的に「ログオフ」されます。この ″まったく操作しない時間″ は「セッション・タイムアウト」として設定されています。標準は 20 分ですが、この値を変更することもできます。

強制ログオフ

利用者（A）を強制的にログオフさせることもできます。システム管理者は専用の画面操作で現在、ログオンしている利用者をログアウトさせるという機能を提供しています。

ロックの解放についてのまとめ

更新画面で取得したロックは、次のいずれかの場合に解放されます。

- 更新画面で「保存」または「キャンセル」ボタンを押下して、更新画面を正常に終了させる。
- メニュー画面に戻る。（メニュー画面に戻るタイミングで、すべてのロックは解除されます。）
- サブメニューから別の業務（画面）を選択する。
- セッションタイムアウトによる自動ロック解除。
- ログオフ。（システム管理者による強制ログオフを含む。）

[ご注意ください] 上記以外の操作では、取得したロックは解放されません。例えば更新画面から、パンくずを使って別画面に移動した場合、ロックは維持されたままです。

ページタイトル

画面例

ページタイトルを ″お得意様 更新″ へ変更した例を示します。

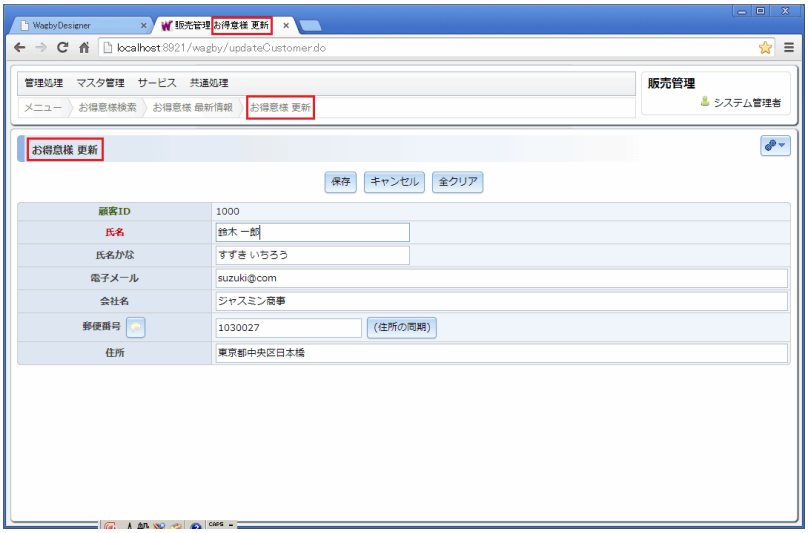


図 ページタイトルを変更

定義方法

画面タブ内の検索・一覧表示タブを開きます。「ページタイトル」の欄に文字を入力します。

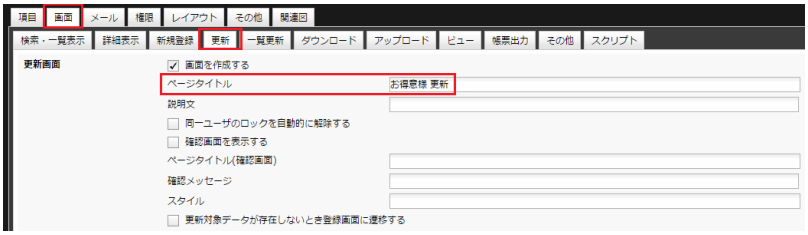


図 ページタイトルの設定

説明文

画面例

説明文を設定した例を示します。

図 説明文を表示する

定義方法

画面タブ内の検索・一覧表示タブを開きます。「説明文」の欄に文字を入力します。

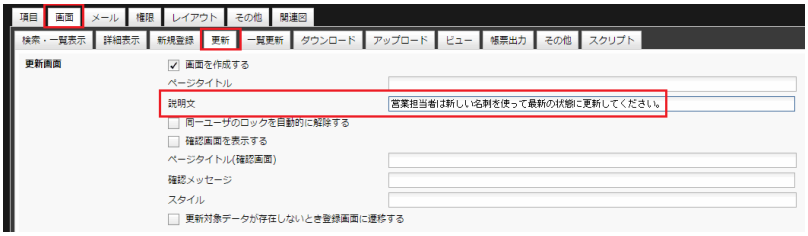


図 説明文の設定

HTMLタグを加える

説明文にHTMLタグを加えることもできます。文章を途中で改行させる場合は br タグを使うということができます。詳細は「検索・一覧表示>説明文 - HTMLタグを加える」をお読み下さい。

同一ユーザのロックを自動的に解除する（悲観ロック利用時）

Wagbyでは、ブラウザを閉じてしまった利用者はセッションタイムアウトを過ぎると自動的にログオフとします。この設定は、セッションタイムアウトの前に再ログオンした場合、当該利用者のロックを解除するというものです。

したがって、利用者が更新画面を開いたまま Web ブラウザを閉じてしまった場合、そのデータについての更新ロックは残ったままとなります。

この設定は同一利用者に限り、再ログオンすると自動的にロックを解除します。標準では無効となっています。

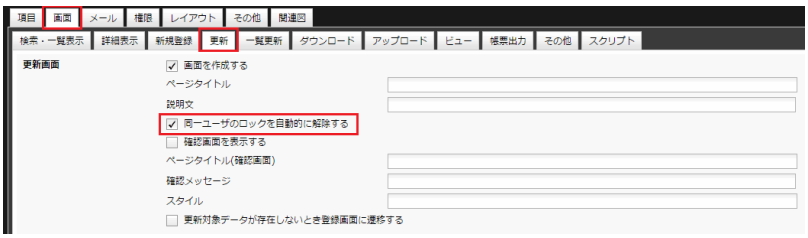


図 ロックの解除

例

例えば、user1アカウントでログオンし、あるデータの編集画面を開きます。この時、別のPC(ブラウザ)で再びuser1アカウントでログオンし、同じデータの編集画面を開くことができるようになります。自動的に最初の更新ロックを解除し、再度ロックを取得します。

なお、他のアカウントでログオンした場合は「ロックされているため更新できない」というエラーになります。

確認画面を用意する

例

更新時に確認を行うことができます。「保存」ボタンに代わって「更新確認」ボタンが用意されます。

図 更新確認ボタン

更新確認ボタンを押下すると、更新を行ってよいかどうかを示すメッセージが表示されます。 ここで「更新画面へ戻る」ボタンを押下すると、図の画面へ戻ることができます。

図 更新確認画面

図で「更新する」ボタンを押下すると、更新されます。

図 更新終了画面

定義方法

更新画面の設定欄にある「確認画面を表示する」を有効にします。

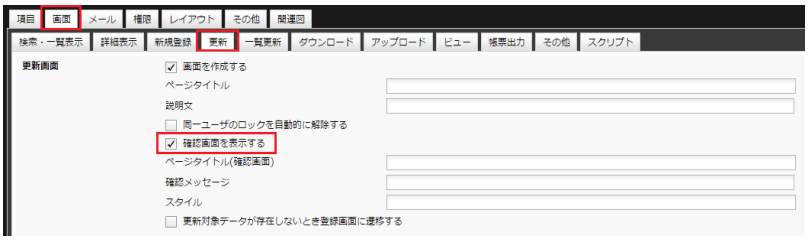


図 確認画面を表示する

確認画面用のページタイトルを指定する

確認画面に用意されるページタイトルを指定することができます

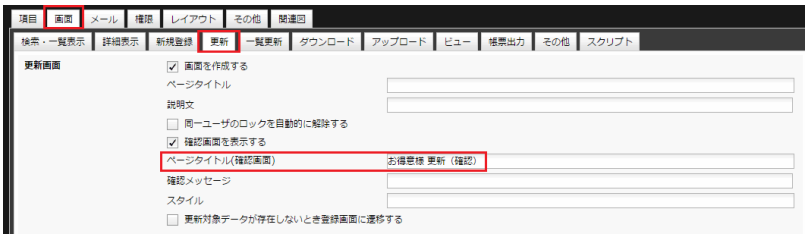


図 ページタイトルの設定（確認画面用）

通常の更新画面ページタイトルを設定し、本欄を空白とした場合は、確認画面用のページタイトルに更新画面ページタイトルの設定が流用されます。

確認メッセージとスタイルを指定する

確認画面に用意されるメッセージとスタイルを変更することができます。

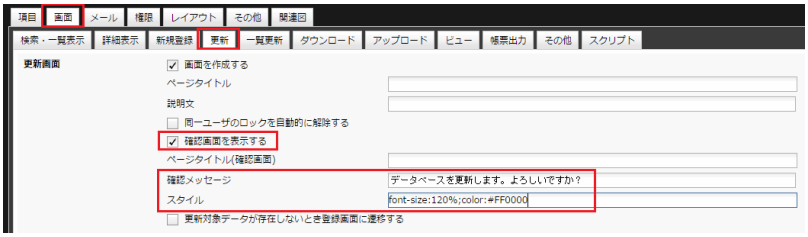


図 メッセージとスタイルの指定

図の設定を反映した画面を図に示します。

図 独自の確認メッセージを用意した例

仕様・制約

- 確認画面は、詳細画面を再利用して実現しています。確認画面を用意する場合は、同時に詳細画面も作成してください。
- 確認画面の挙動は、詳細画面と同じになります。
- 子モデルの同時更新機能を用いる場合、確認画面を設定することはできません。（子モデル同時更新と確認画面は併用できません。）

更新対象データが存在しないとき登録画面に遷移する

「ボタン名と表示条件のカスタマイズ」で説明します。

アクションボタンの表示位置

（画面遷移を行うための）アクションボタンの位置を変更することができます。

上部に配置する

図 アクションボタンを上部に配置した例（標準）

下部に配置する

図 アクションボタンを下部に配置した例

上部と下部の両方に配置する

図 アクションボタンを上部と下部の両方に配置した例

定義方法

画面タブ内の検索・一覧表示タブに用意された「ボタン表示>アクションボタンの表示位置」を設定します。空白時は「上」として扱います。

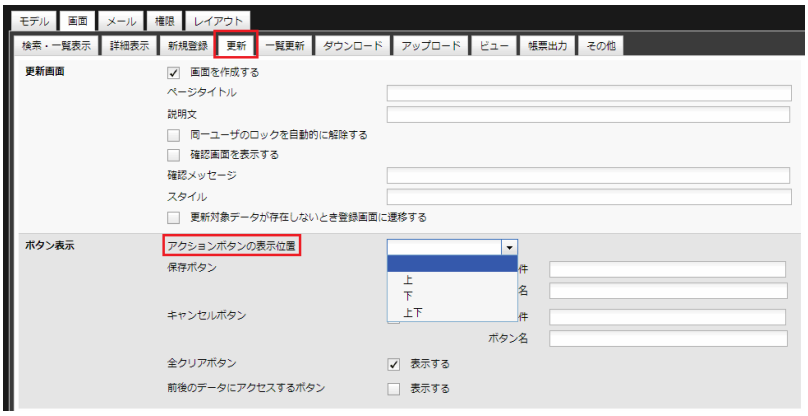


図 アクションボタンの表示位置の設定

用意されているボタン（機能）

更新画面に用意できるボタン（機能）を説明します。

図 更新画面に用意できるボタン

定義方法

画面タブ内の更新タブに用意された「ボタン表示」を有効にします。

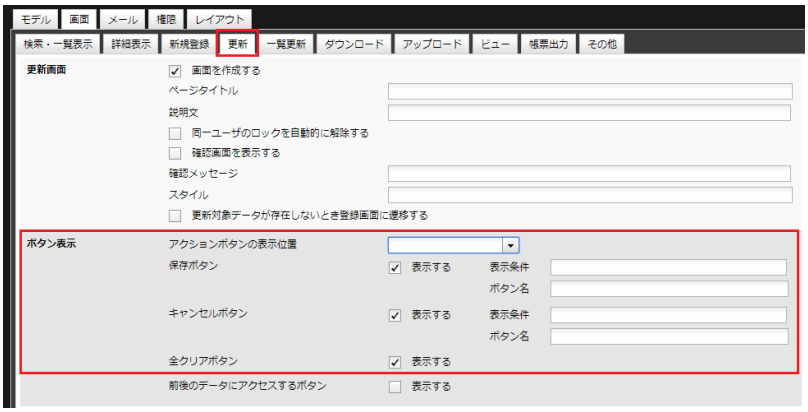


図 ボタン表示の設定

各ボタンの表示を有効にする場合、関連する画面機能が作成されている必要があります。 画面機能が作成されない設定では、ボタンも作成されません。

種別	初期状態	備考
保存	○	
キャンセル	○	
全クリア	○	入力した項目をすべて消去します。（データベースからデータが消去されるわけではありません。）

ボタンの表示制御

〝ボタン名と表示条件のカスタマイズ〟をお読みください。

全クリアボタン押下時の読込専用項目の扱い

- 自動計算によって求められた値は、全クリアボタン押下時に、再計算されます。
- 初期値がセットされた値は、全クリアボタン押下時に、初期値を再セットします。

前後のデータにアクセスする

更新画面でも（詳細画面と同様に）前後のデータにアクセスする機能が備わっています。 カード型データベースの感覚で、前へ、次へといった操作が可能です。

図 更新画面に用意される前後移動ボタン

ここで「前・後」とは、現在の検索結果に対する操作となります。検索条件や並び順の指定によって、前・後のデータは変動します。

定義方法

更新画面>ボタン表示から「前後のデータにアクセスするボタン」の設定を有効にします。

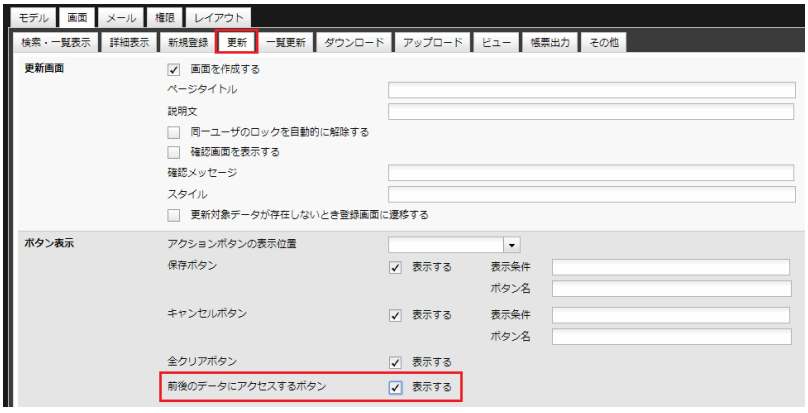


図 前後のデータにアクセスするボタン設定

"画面 > 検索・一覧表示 > ボタン表示 > 更新ボタン" もあわせて有効にすると、一覧表示画面から更新画面へ直接、遷移し、その後は「更新して次へ」ボタンで連続したデータ更新を行うことができます。

キャンセル操作時のダイアログ

"新規登録 > キャンセル操作時のダイアログ"をお読みください。

削除

詳細表示画面

詳細表示画面で削除ボタンを用意することができます。標準では「表示する」設定となっています。

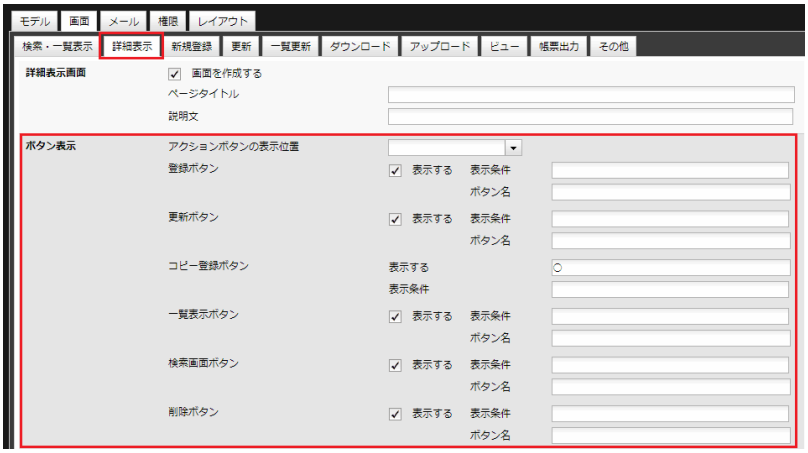


図 削除ボタンの設定（詳細表示）

一覧表示画面（一括削除）

一覧表示に一括削除ボタンを用意することができます。

図 削除ボタンの設定（一覧表示）

例

顧客モデルに一括削除ボタンを用意した例を示します。

一覧表示の先頭行に用意されたチェックボックスは全選択を行います。二行目以降のチェックボックスは個別データの選択を行います。

図 一括削除を有効にした一覧表示

図は最初の三行を削除対象として指定した例です。選択後に「一括削除」ボタンを押下します。確認ダイアログが表示されますので「OK」を押します。

図 選択後に一括削除ボタンを押下する

選択したデータを削除しました。

図 選択データを削除した

全選択チェックボックスの影響範囲

全選択チェックボックスは、表示されているデータのみを対象とします。複数ページにまたがった検索結果があった場合でも、現在表示されているページのみとなります。

よって一括削除ボタンも、表示されているページで選択されたデータが対象になります。

削除後に遷移する画面

- 一覧表示画面の一括削除ボタン押下後は、同じ一覧表示画面に遷移します。
- 詳細画面の削除ボタン押下後は、ぱんくずリストを辿って最初に見つかった詳細画面へ遷移します。詳細画面の遷移を行っていない場合は、同モデルの一覧表示画面に遷移します。

一覧更新

基本動作

検索画面から一覧更新画面へ遷移するボタンが用意されます。

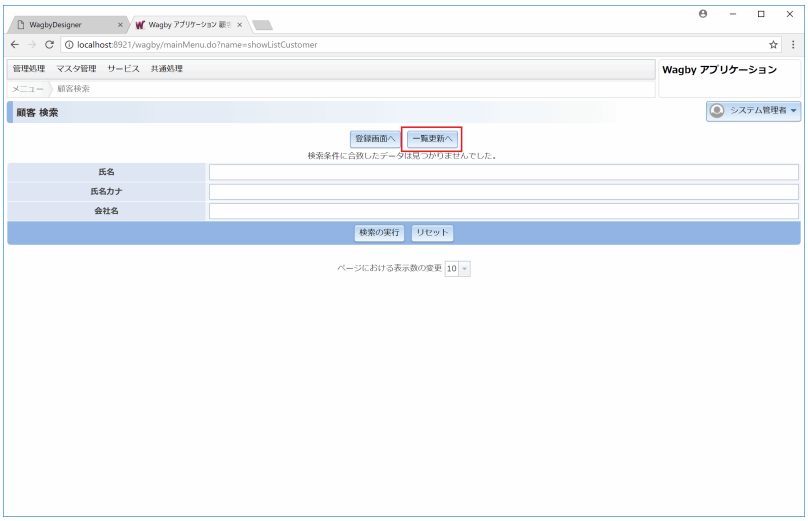


図 「一覧更新へ」ボタン

一覧更新画面では表形式でデータの更新・新規登録・削除処理を行うことができます。

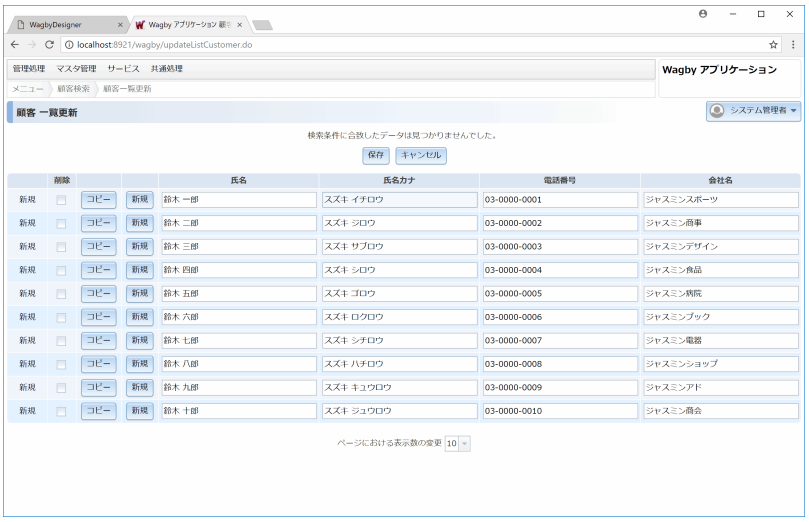


図 一覧更新画面

更新

各行がそれぞれ一つのデータに対応します。行内の入力欄を修正します。修正中のデータは ″修正中″ というステータスで管理され、ロックされます。保存処理では、修正中のデータのみが更新されます。

コピー登録・新規登録

各行の左側に用意された「コピー」または「新規」ボタンを押下します。

削除

左側に用意された「削除」欄で、削除したいデータをチェックします。保存ボタンを押下することで削除されます。

保存

現在の編集状態を保存します。保存後にロックは解除されます。エラー行は無視されます。たとえば10行編集中で1行に入力エラー（例 必須入力漏れ）があった場合、保存ボタンを押下することで9件が保存されます。

キャンセル

すべての編集状態を解除し、元の画面（一覧表示画面）へ戻ります。途中まで編集した値は保存されません。

入力チェック

更新があったデータの入力チェック処理を行います。エラーがあった場合は ″エラー″ と表示されます。

複数ページにまたがる修正

件数が多い場合、複数のページに分割して表示されます。ページを移動した場合でも、編集作業の内容は記憶されています。元のページに戻ると、編集中のデータが復元されます。

テキストエリアの行数・文字数

画面例

一覧更新用のテキストエリアを別に指定することができます。

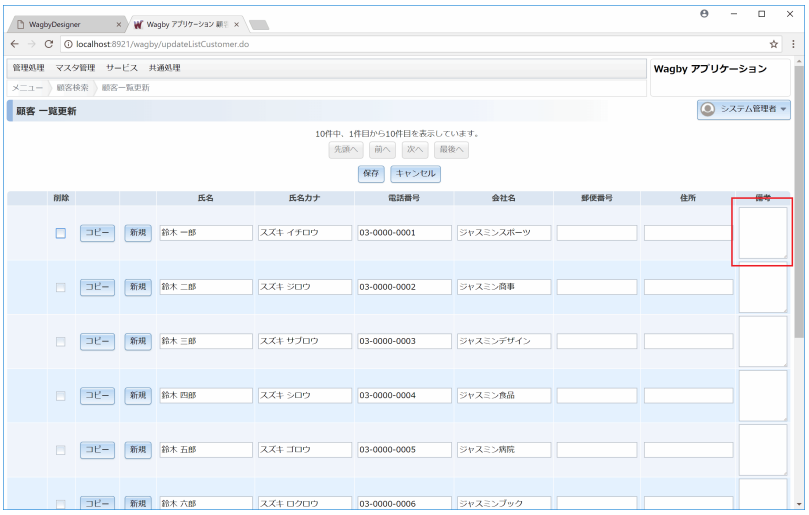


図 テキストエリアの利用

定義方法

対象項目（ここでは「備考」項目）の「入力項目の種類」をテキストエリアとします。

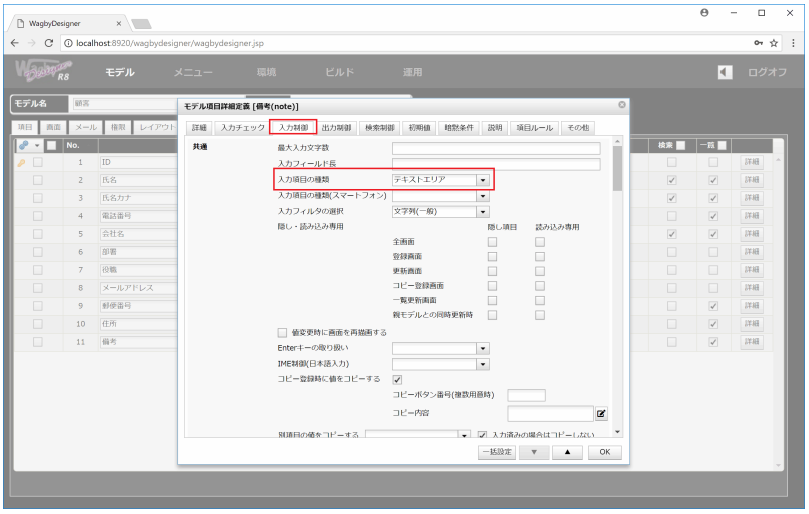


図 入力項目の種類を指定する

入力制御タブ内のテキストエリア設定で、一覧更新画面向けの「行数」と「1行の文字数」を指定します。

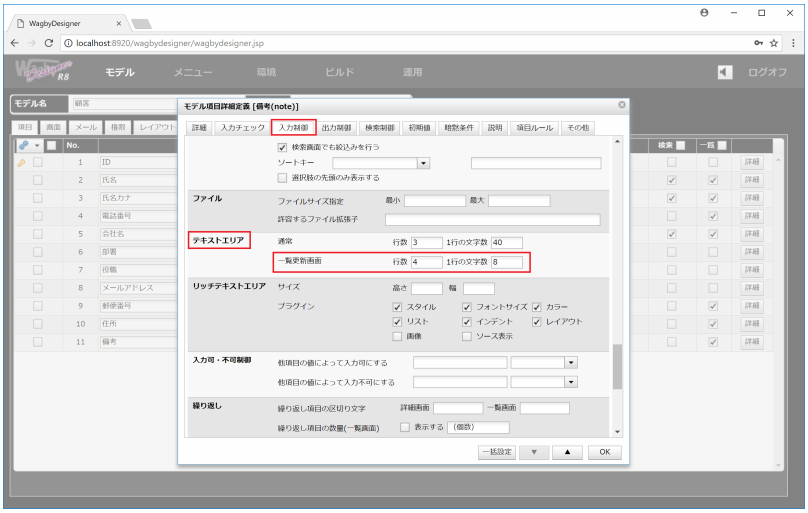


図 一覧更新画面向けの「行数」と「1行の文字数」

一覧更新画面の同時表示数

この設定は一覧表示と同様の動作になります。詳細は"検索・一覧表示>一覧表示画面の同時表示数"をお読みください。

一覧更新画面のページネーション

この設定は一覧表示と同様の動作になります。詳細は"検索・一覧表示>一覧表示画面のページネーション"をお読みください。

一覧更新ボタンの表示制御

検索結果が見つかったときのみ、一覧更新ボタンを表示するという制御を行うことができます。 一覧更新ボタンの表示制御欄に、次の式を記述します。

```
RESULTSIZE()>0
```

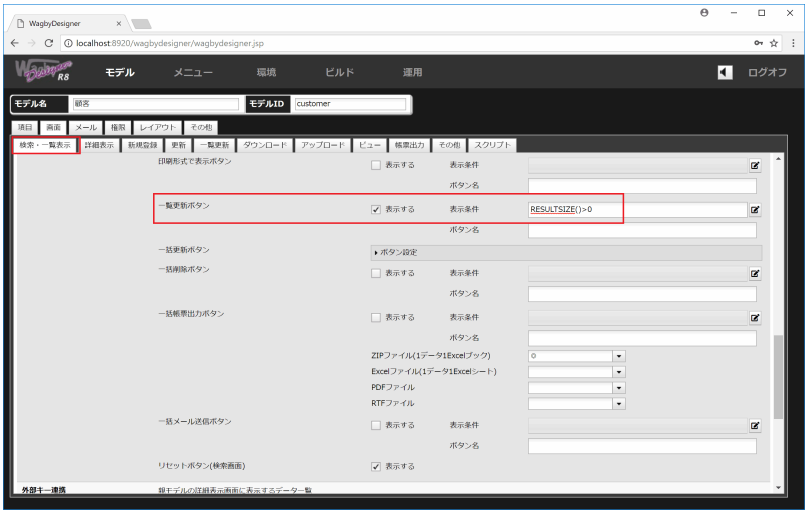


図 一覧更新ボタンの表示制御の例

RESULTSIZE 関数は、検索・一覧表示画面でのみ有効です。

用意されているボタン（機能）

一覧更新画面に用意できるボタン（機能）を説明します。

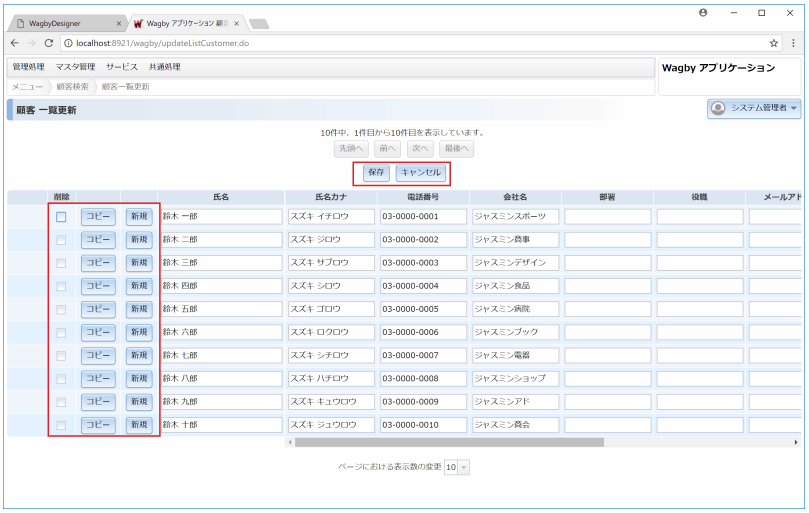


図 一覧更新画面に用意されるボタン

定義方法

画面タブ内の一覧更新タブに用意された「ボタン表示」を有効にします。

図 ボタン表示の設定

種別	初期状態	説明
保存	○	
キャンセル	○	一覧表示画面へ戻ります。
新規登録	○	新規登録画面を用意している必要があります。
コピー登録	○	新規登録画面を用意している必要があります。

ボタンの表示制御

「一括削除・一括メール送信・一括帳票・一括更新・印刷形式で表示」ボタンは、検索結果が存在する場合のみ表示されます。検索結果が存在しない場合、ボタンも非表示となります。

開発者による表示制御の詳細は"ボタン名と表示条件のカスタマイズ"をお読みください。

データベースコミット前に呼び出されるスクリプト

登録、削除、一覧更新画面ではデータベース保存の直前に動作するスクリプトを記述することができます。これはトランザクション境界の内側で動作します。

一覧更新処理の「データベースコミット前」にデータが適切かどうかをチェックし、不適切な場合に例外をスローさせると、この一覧更新処理をロールバックさせることができます。この場合、データベースに保存されず、編集画面に戻ります。

例 画面にエラーメッセージを表示、ログにも記録する

```
var BusinessLogicException =
 Java.type("jp.jasminesoft.jfc.core.exception.BusinessLogicException");
throw new BusinessLogicException("エラーです。");
```

例 画面にエラーメッセージを表示、ログには記録しない

```
var Jfcerror = Java.type("jp.jasminesoft.jfc.error.Jfcerror");
var error = new Jfcerror();
error.content = "エラーが発生しました。";
p.errors.addJfcerror(error);
```

例 画面にエラーメッセージを表示せず、ログにも記録しないが、ロールバックさせる

```
p.request.getAttribute("TransactionStatus").setRollbackOnly();
```

ページタイトル

画面例

ページタイトルを "お得意様 まとめて更新" へ変更した例を示します。

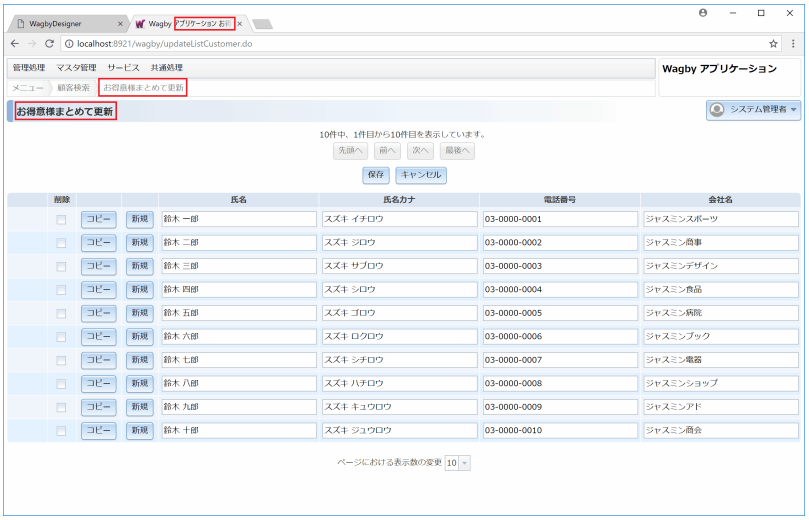


図 ページタイトルを変更

定義方法

画面タブ内の検索・一覧表示タブを開きます。「ページタイトル」の欄に文字を入力します。

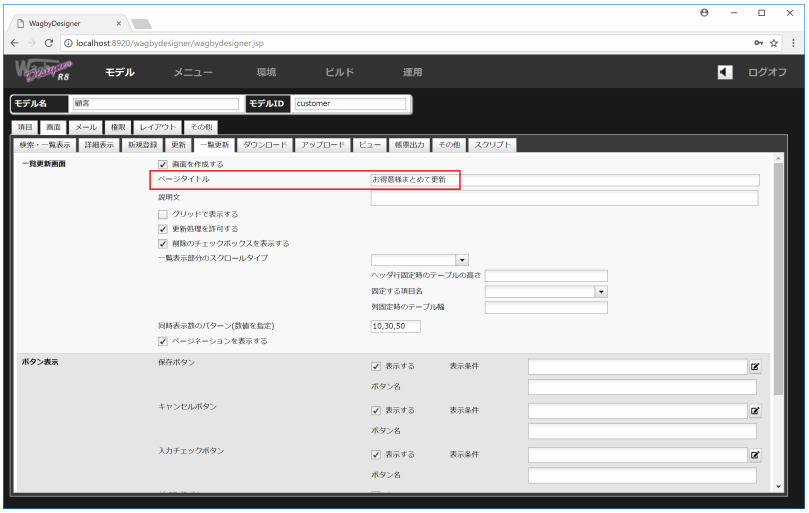


図 ページタイトルの設定

説明文

画面例

説明文を設定した例を示します。

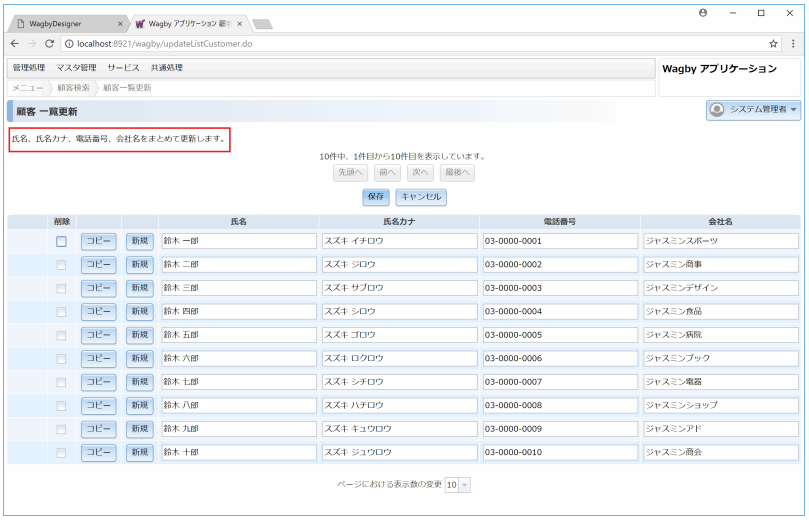


図 説明文を表示する

定義方法

画面タブ内の検索・一覧表示タブを開きます。「説明文」の欄に文字を入力します。

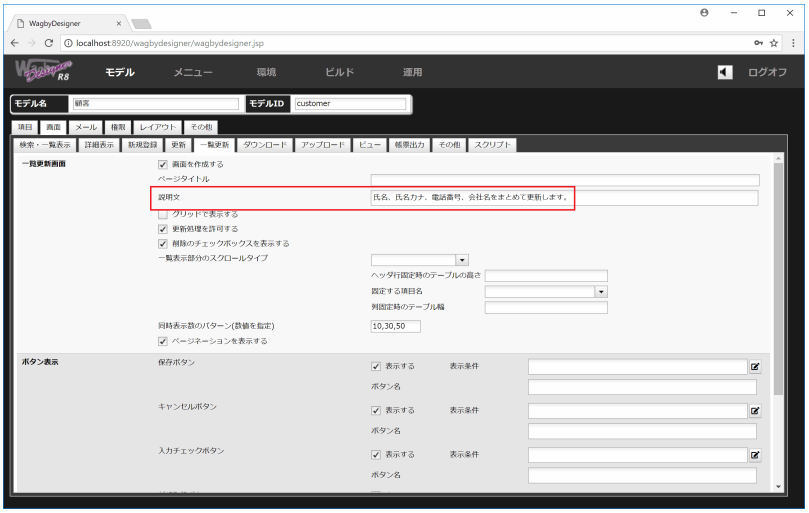


図 説明文の設定

HTMLタグを加える

説明文にHTMLタグを加えることもできます。詳細は"検索・一覧表示>説明文>HTMLタグを加える"をお読み下さい。

更新処理を許可する

画面例

更新処理を許可しない、という設定を行うことができます。次のような動作になります。

- 新規登録は行える。
- 更新・削除処理が行えない。

すなわち一覧更新画面中に「入力できるレコード」と「入力できないレコード」が混在した更新モードになります。

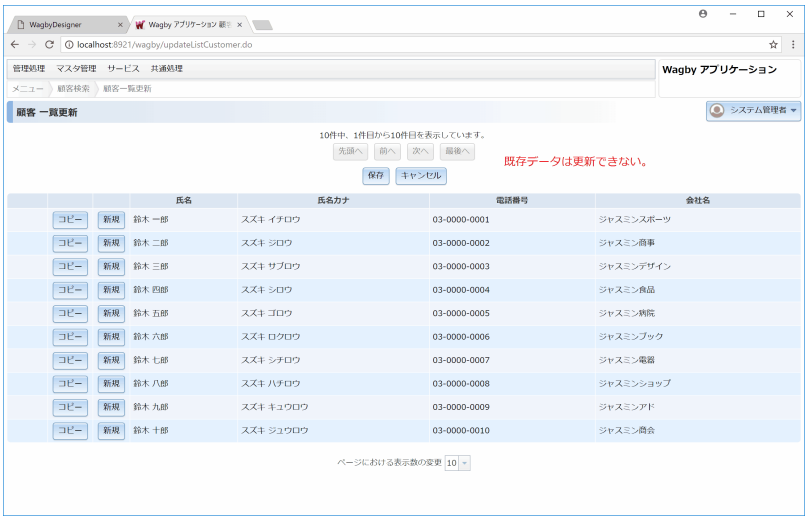


図 更新が許可されていない一覧更新画面

行の「新規」ボタンを押下すると新規入力用のレコードは用意されます。

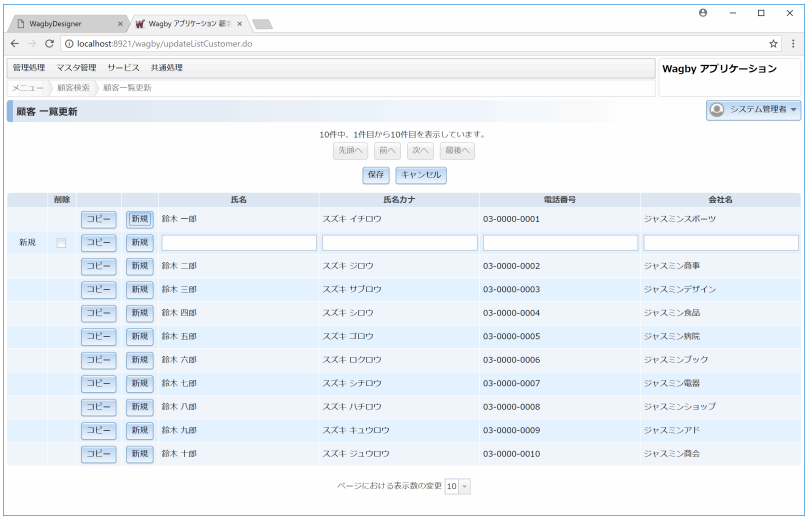


図 新規でレコードを追加することはできる

定義方法

「更新処理を許可する」設定を解除します。（標準で有効になっています。）

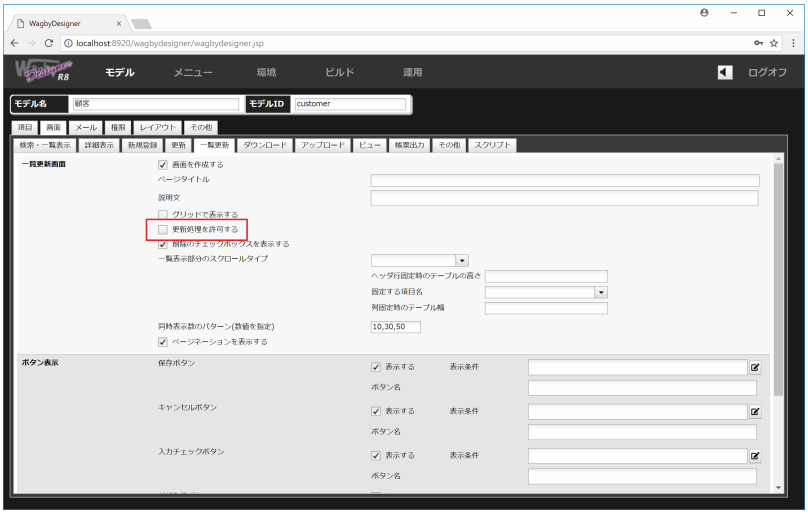


図 更新処理を許可する設定

削除のチェックボックスを表示する

削除用のチェックボックスを非表示とすることもできます。削除機能を無効にします。

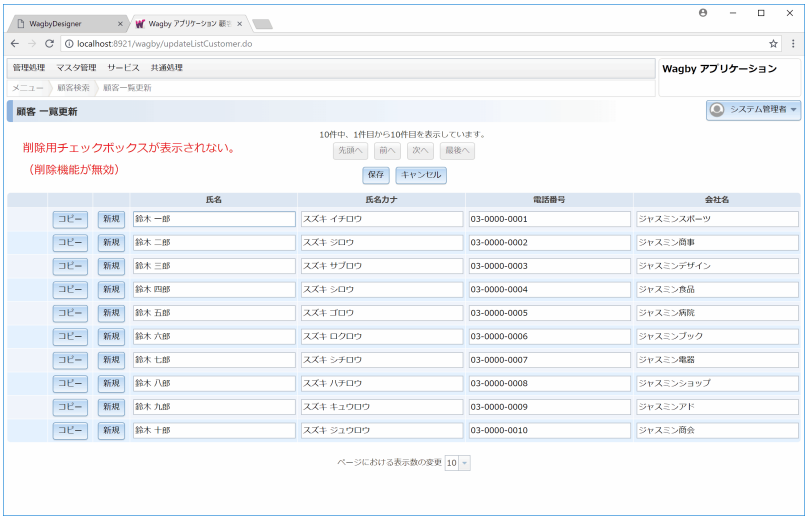


図 削除用チェックボックス非表示の例

定義方法

「削除のチェックボックスを表示する」設定を解除します。（標準で有効になっています。）

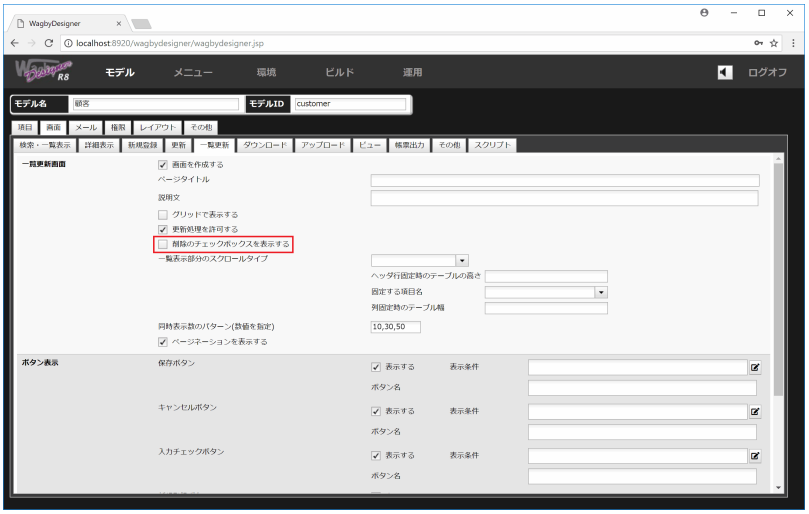


図 削除のチェックボックスを表示する設定

ロック制御

一覧更新画面では、表示されているデータがロックされます。

「次のページ」や「前のページ」に移動した場合、修正されたデータはそのままロックを保持し、修正されていないデータのロックは解除されます。

ページを移動した時にロックの取得に失敗した場合は、そのデータの左端にロック失敗と表示され、編集することはできません。

一覧表示部分のスクロールタイプ（列固定）

画面例

特定の列を軸に、左側と右側に分割することができます。（Excelの「ウィンドウ枠の固定」のイメージです。） 図は「氏名」項目を左側に、それ以降を右側に分割した例です。右側の部分は項目が多い場合でも自動で均等割付けされます。

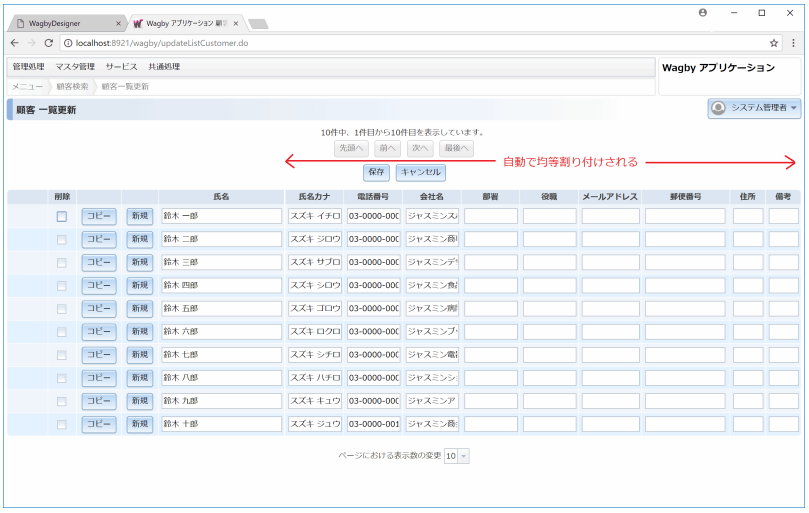


図 列固定を行った画面

定義方法

「画面>一覧更新」タブで、「一覧表示部分のスクロールタイプ」に「列固定」を指定します。 このとき「固定する項目名」と「列固定時のテーブル幅」を指定する必要があります。 前者は項目ID(英語名)を、後者はコンマ区切りで左側と右側部分の比率を指定します。

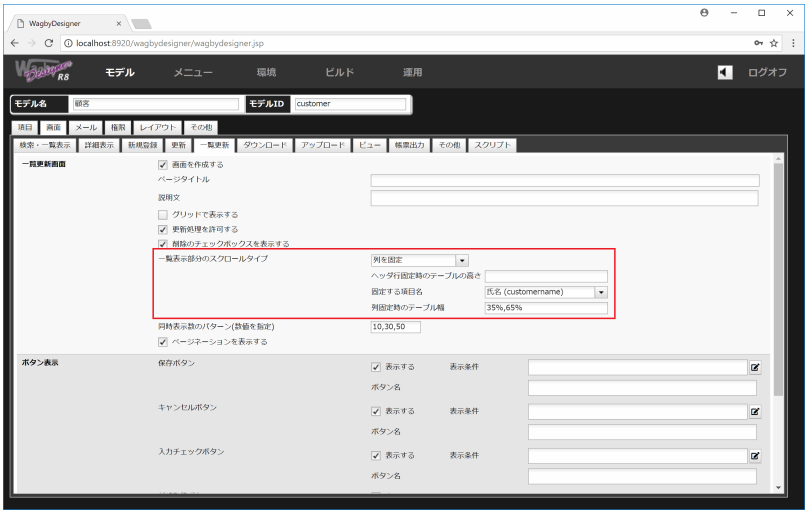


図 列固定の設定

ラベル幅を指定する

図の表示例は、項目数が増えた場合でも均等割付となるため、一つ一つの列幅が小さくなり、入力しづらくなります。 各項目のラベル幅を指定することで、図のようにスクロールバーを表示させることができます。

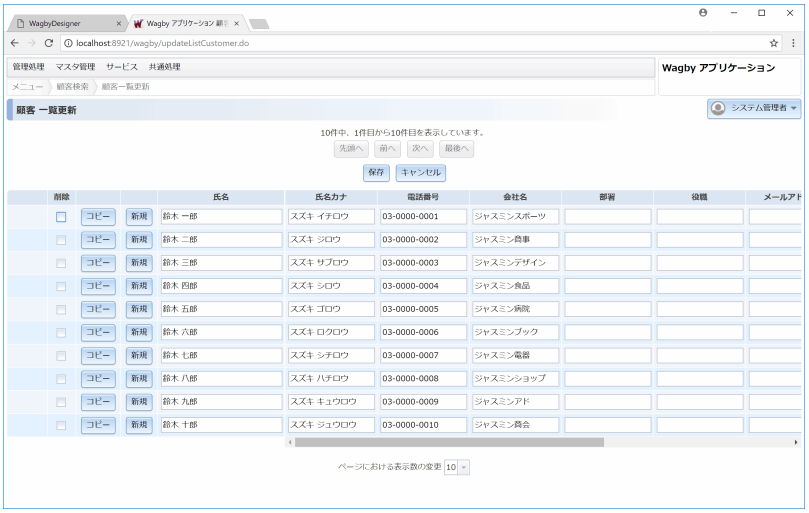


図 スクロールバーを使う

一覧表示に用いる項目について、ラベル幅をピクセル値で指定します。

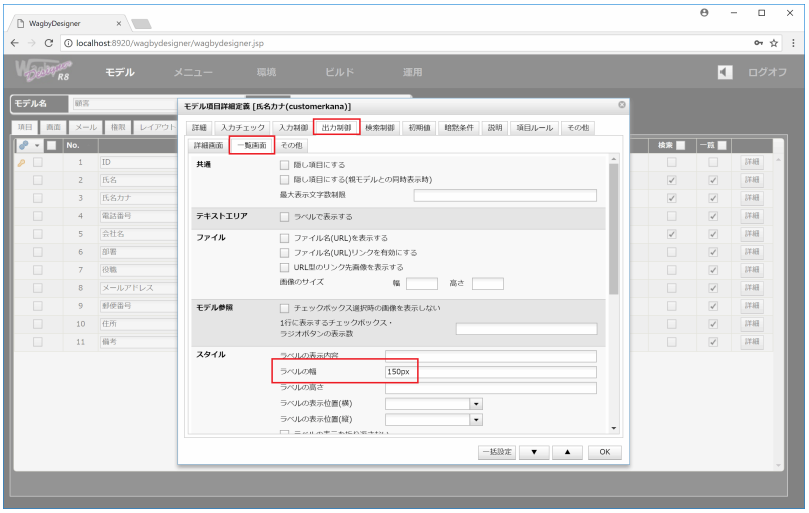


図 一覧表示のラベル幅を指定する

ラベル幅指定を省略すると、Webブラウザが自動的に幅調整を行います。項目が多い場合、必ずしも最適なレイアウトにならないことがあります。この場合はラベル幅の設定を行ってください。

一覧表示部分のスクロールタイプ（ヘッダ行固定）

画面例

列の項目名部分（ヘッダ行）を固定にします。（Excelの「先頭行固定」のイメージです。） 図,図はヘッダ行を固定とし、表示領域の高さを300ピクセルと指定した例です。表示部をスクロールしてもヘッダ行は固定されています。

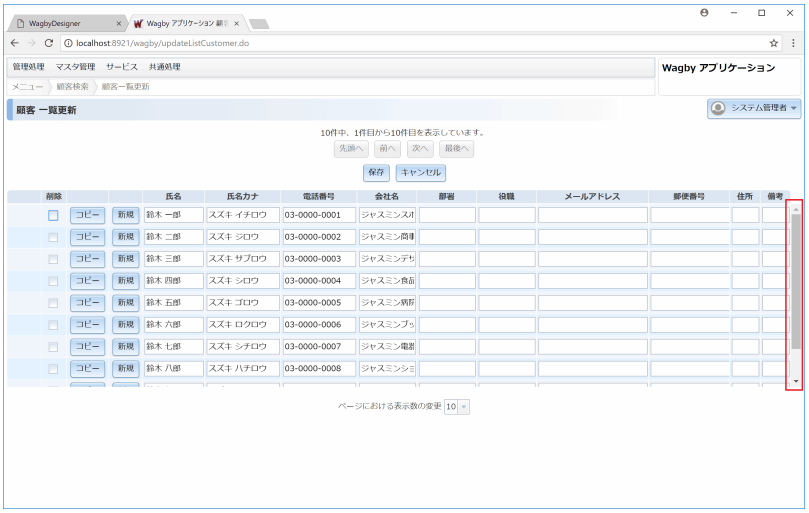


図 ヘッダ行固定(1)

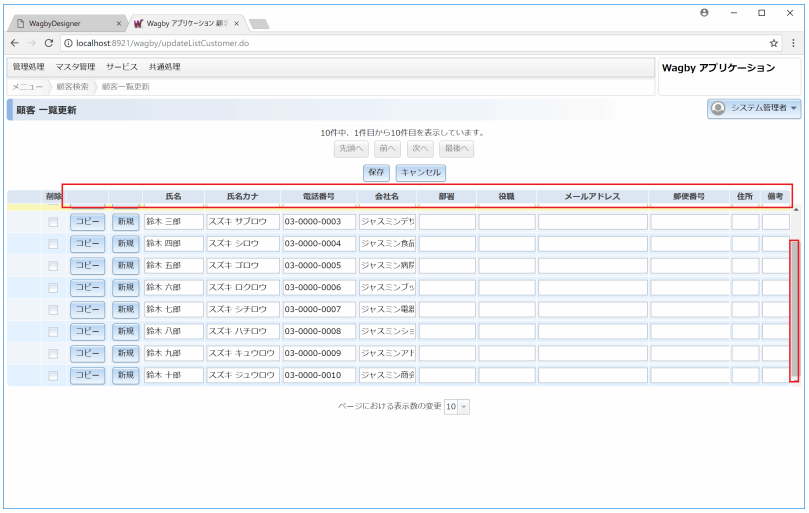


図 ヘッダ行固定(2)

定義方法

「画面>一覧更新」タブで、「一覧表示部分のスクロールタイプ」に「ヘッダ行固定」を指定します。 このとき「ヘッダ行固定時のテーブルの高さ」を指定する必要があります。空白時の初期値は"240px"です。

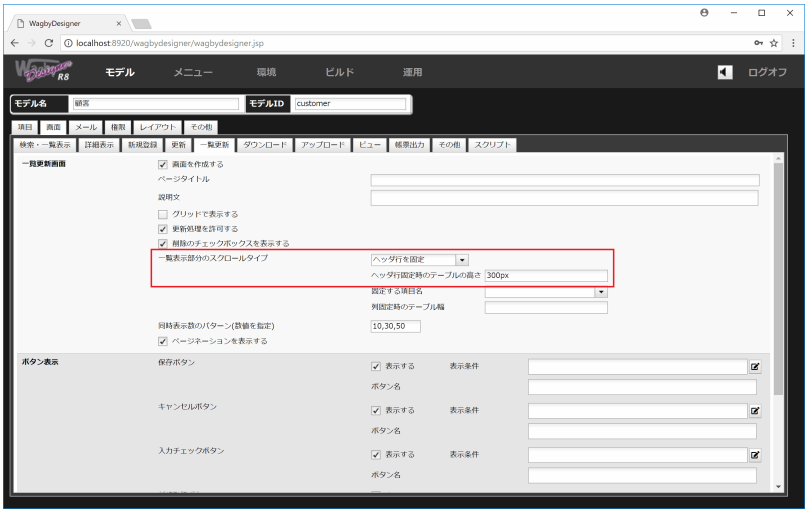


図 ヘッダ行固定の設定

一覧表示に用いる項目について、ラベル幅をピクセル値で指定します。（「列固定」の図と同様）

ラベル幅指定を省略すると、Webブラウザが自動的に幅調整を行います。項目が多い場合、必ずしも最適なレイアウトにならないことがあります。この場合はラベル幅の設定を行ってください。

折り返しが効かない場合

"出力制御（スタイル）> CSSクラス名・スタイルの指定 > 例 一覧表示項目の折り返し幅を指定する"をお読みください。

制約

- ヘッダ行を固定したとき、ブラウザの幅を変更して表示ウィンドウサイズを変えた場合でも、一覧表示の幅はウィンドウサイズにあわせて変動しません。最初に表示された幅に固定されます。

仕様と制約

扱うことができない項目

- 繰り返し項目
- 繰り返しコンテナ（繰り返しコンテナ用のID および 繰り返しコンテナ内の項目を含みます）
- 「追記型リストボックス」を指定している項目
- 参照先のモデルが複合キーの場合のチェックボックス

編集することができない（表示のみの）項目

- 参照連動
- ファイル名
- 外部キーを指定した項目
- 隠し項目
- 読み込み専用項目

検索条件で絞り込んだ結果に対する一覧更新の実行 !-- N20161202_143417 -->

一覧更新画面では、検索条件を設定する欄は用意されません。

図のように、検索・一覧表示画面に一覧更新画面へ移動するボタンを用いる場合は、検索画面で絞り込んだデータのみが一覧更新の対象となります。（すなわち絞り込んだデータに対して、一覧更新を行うことができます。）

メニュー画面より直接、一覧更新画面へ遷移した場合は、すべてのデータが一覧更新の対象となります。

(214)

項目同士の絞り込み設定

"他モデルの参照 > 参照モデル同士の関係性で選択肢を絞り込む > トラブルシューティング > 一覧更新画面で絞込が動作しない"をお読みください。

入力チェック

一覧更新画面では変更を行わなかった行（データ）については入力チェックも行われません。変更のあった行（データ）だけが更新の対象となります。

その他の制約

- 外部キーによる親子モデル関係において親モデルでコピー登録を行ったとき、子モデルもコピー登録されますが、一覧更新でのコピー登録では子モデルはコピーされません。

(214) 対象モデルの検索条件に初期値など、デフォルトの検索条件を設定していた場合は、条件によって絞り込まれたデータが一覧更新の対象となります。 暗黙条件も有効です。

- 外部キーによる親子モデル関係において、親モデルの一覧表示項目に子モデルを集計した値を（計算式で）指定し、かつこの項目はデータベース保存しない場合、この親モデルの一覧更新画面では当該項目の値は不正になります。つまり一覧表示画面における項目で子モデルを参照する計算式は、一覧更新画面では機能しません。このため一覧更新画面に含める項目で集計した値を使う場合は、データベースに保存するようにしてください。
- 一覧表示部分のスクロールタイプ（列固定、ヘッダ行固定）と、複数段一行明細レイアウトを同時に指定することはできません。レイアウト設計はいずれか一方を使うようにしてください。
- 必須指定を行った項目は、一覧表示の設定を行うか、または初期値（「登録時」または「登録画面表示時」）の設定を行う必要があります。（ただし新規登録画面/コピー登録画面（詳細画面、一覧表示画面）を作成しない場合は、この制約に該当しません。）
- 「エラー行を無視して保存する」設定を無効にすることはできません。 R8.0.3 (ENH936301) で入力エラー時はエラーメッセージを表示して編集画面に戻るようになりました。（これまでは入力エラーでも無視して保存していました。）この改善に伴い、入力チェックボタンは廃止されました。
- 郵便番号から住所を求める機能は行われません。 R8.1.0 (ENH1036301) で対応しました。
- 住所正規化処理（オプション）には対応していません。

一覧更新 グリッドを利用する

定義方法

「画面 ＞ 一覧更新 ＞ グリッドで表示する」を有効にします。

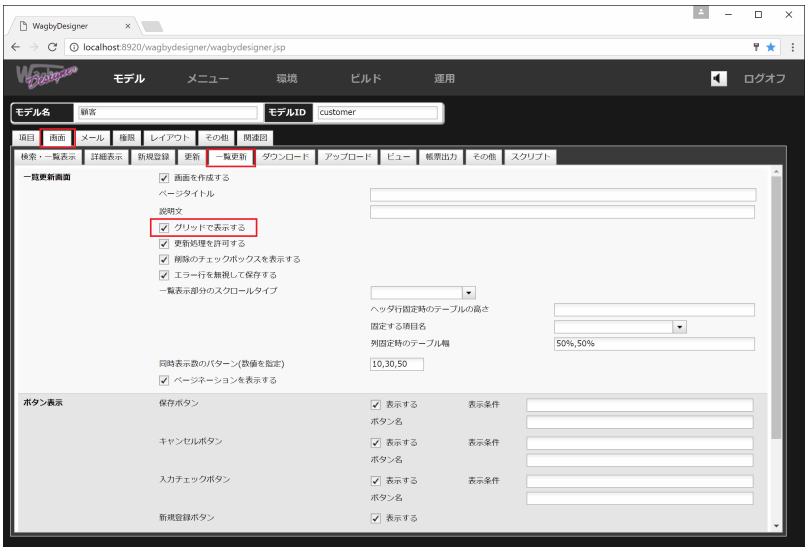


図 グリッドで表示する

基本機能

一覧更新画面ではグリッド形式でデータの更新・新規登録・削除処理を行うことができます。

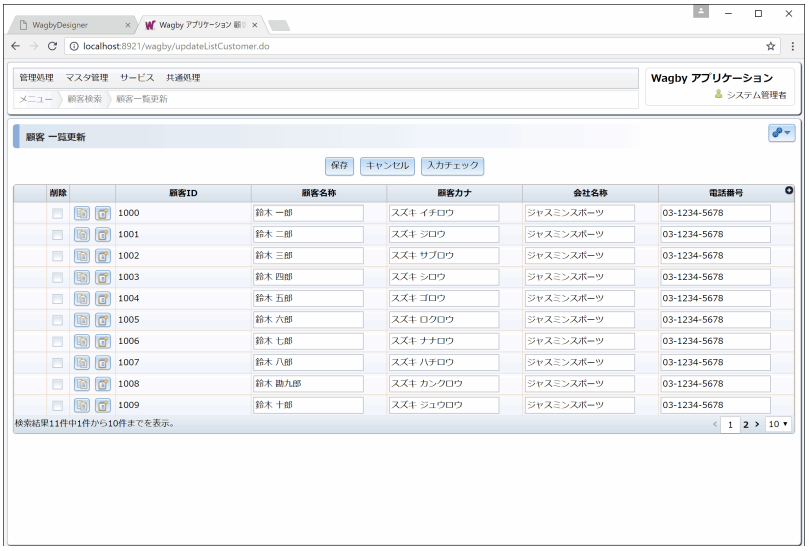


図 一覧更新画面グリッド利用

更新

各行がそれぞれ一つのデータに対応します。行内の入力欄を修正します。修正中のデータはロックされます。保存処理では、修正中のデータのみが更新されます。

コピー登録・新規登録

各行の左側に用意された「コピー」または「新規」ボタンを押下します。

削除

左側に用意された「削除」欄で、削除したいデータをチェックします。保存ボタンを押下することで削除されます。

保存

現在の編集状態を保存します。保存後にロックは解除されます。エラー行は無視されます。たとえば10行編集中で1行に入力エラー（例 必須入力漏れ）があった場合、保存ボタンを押下することで9件が保存されます。

キャンセル

すべての編集状態を解除し、元の画面（一覧表示画面）へ戻ります。途中まで編集した値は保存されません。

入力チェック

現時点のデータの入力チェック処理を行います。エラーがあった場合は ″エラー″ と表示されます。保存ボタンを押下する前にチェックを行う運用ができます。

複数ページにまたがる修正

件数が多い場合、複数のページに分割して表示されます。ページを移動した場合でも、編集作業の内容は記憶されています。元のページに戻ると、編集中のデータが復元されます。

対応している項目の型

グリッドで対応している項目の型は次のとおりです。

・ 文字列（テキストボックス）

・　数値（テキストボックス）
・　リストボックス
・　ラジオボタン
・　チェックボックス
・　日付カレンダ

ページタイトル

″画面機能 > 一覧更新 > ページタイトル″をお読みください。

説明文

″画面機能 > 一覧更新 > 説明文″をお読みください。

更新処理を許可する

画面例

更新処理を許可しない、という設定を行うことができます。次のような動作になります。

・　新規登録は行える。
・　更新・削除処理が行えない。

すなわち一覧更新画面中に「入力できるレコード」と「入力できないレコード」が混在した更新モードになります。

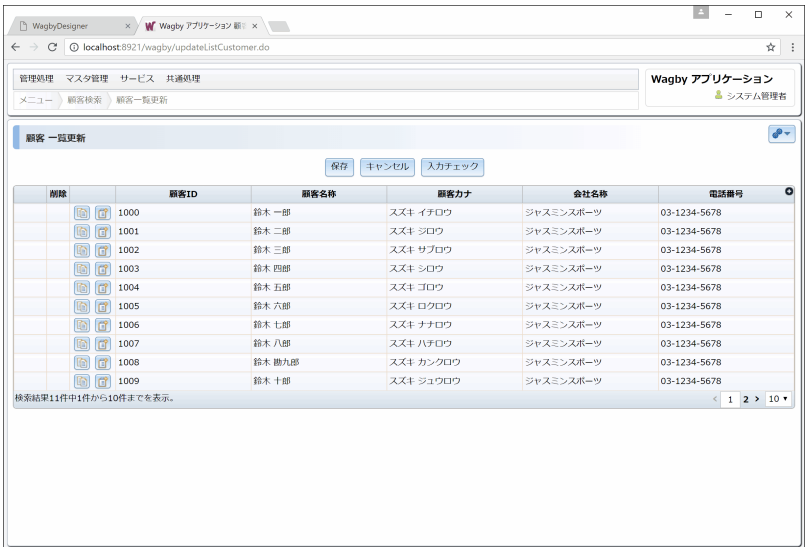


図 更新が許可されていない一覧更新画面

行の「新規」ボタンを押下すると新規入力用のレコードは用意されます。

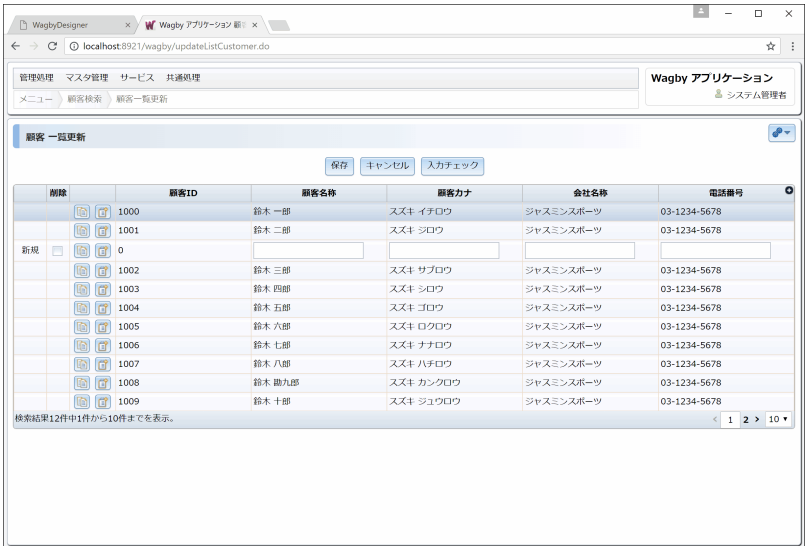


図 新規でレコードを追加することはできる

定義方法

「更新処理を許可する」設定を解除します。（標準で有効になっています。）

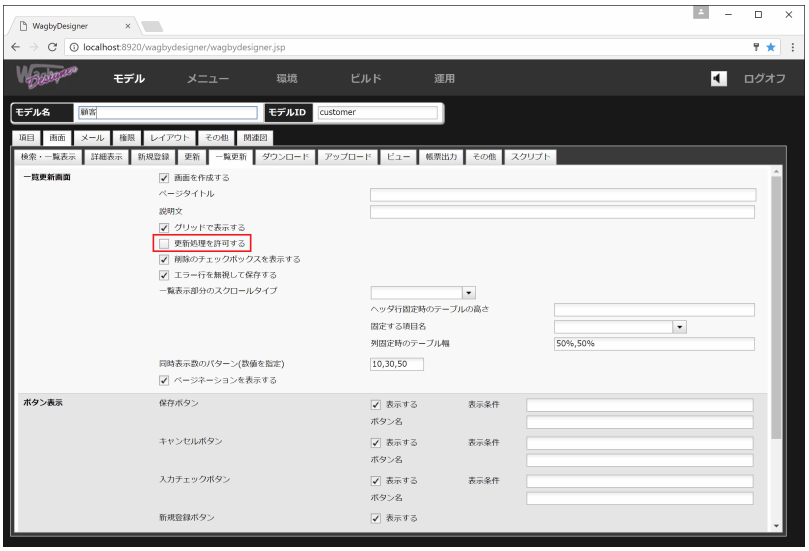


図 更新処理を許可する設定

ロック制御

一覧更新画面では、表示されているデータがロックされます。

「次のページ」や「前のページ」に移動した場合、修正されたデータはそのままロックを保持し、修正されていないデータのロックは解除されます。

ページを移動した時にロックの取得に失敗した場合は、編集することはできません。一覧更新（非グリッド形式）と異なり、そのデータの左端にロック失敗とは表示されません。

テキストエリア

グリッド形式の一覧更新画面では、テキストエリア項目のサイズは固定です。複数行入力時はスクロールバーが付与されます。

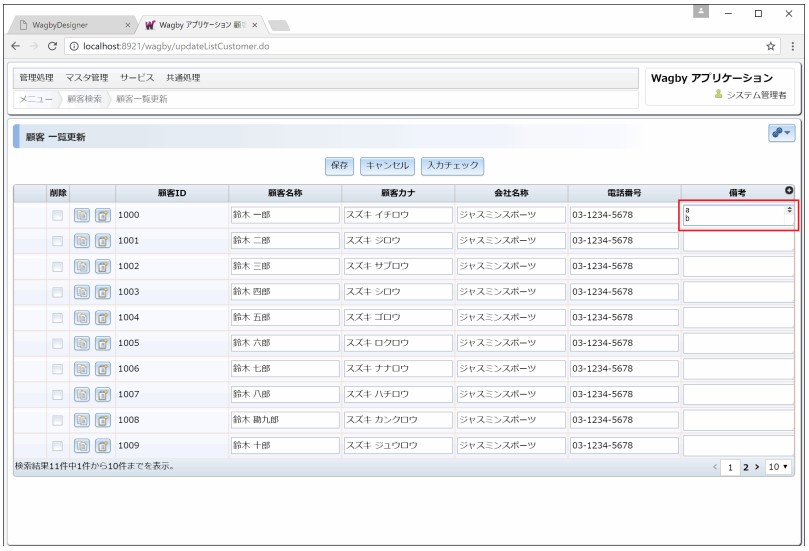


図 テキストエリアの利用

一覧更新画面の同時表示数

この設定は一覧表示と同様の動作になります。詳細は"検索・一覧表示>一覧表示画面の同時表示数"をお読みください。

一覧更新画面のページネーション

この設定は一覧表示と同様の動作になります。詳細は"検索・一覧表示>一覧表示画面のページネーション"をお読みください。

仕様と制約

「画面機能 > 一覧更新 > 仕様と制約」に準じます。 グリッド形式の場合に追加された制約は次のとおりです。

- スクロールタイプ（列固定、ヘッダ行固定、ヘッダ行・列固定）を指定することはできません。
- グリッド内のボタンの表示制御に式を記述することはできません。（式による判定処理は動作しません。）
- 郵便番号と住所の連動機能には対応していません。
- 一覧表示の集計には対応していません。
- 主キーを入力する方式には対応していません。主キーは自動採番のように、入力しない方式のみ動作します。

用意されているボタン（機能）

"一覧更新>用意されているボタン（機能）"をお読みください。グリッド形式利用時は次の制約があります。

- 「保存」ボタンを無効にすることはできません。
- グリッド内のボタンの表示制御は動作しません。すべてのボタンは表示されます。

ウィザード

基本的な使い方

画面例

顧客モデルにウィザードを適用し、二画面に分割した例を図,図に示します。画面移動ボタン（「前画面へ」「次画面へ」）で前後のページに移動します。さらに画面指定ボタン（「画面1」「画面2」）を使って直接、特定の画面を開くこともできます。

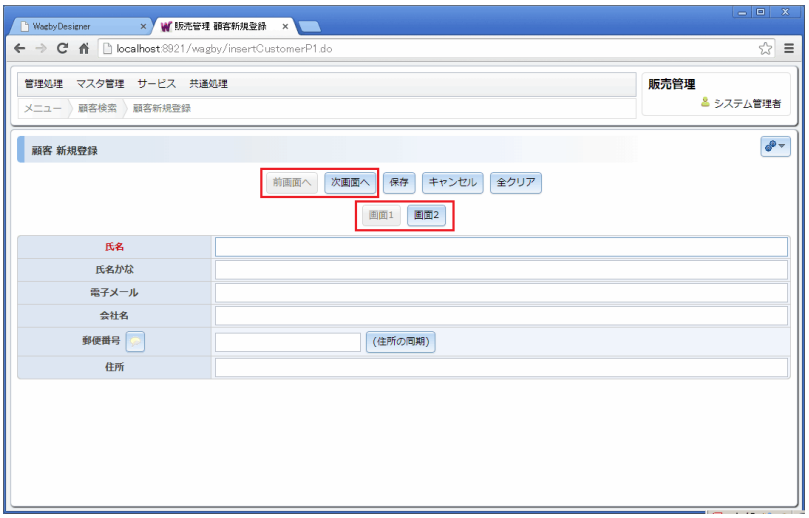


図 最初の画面

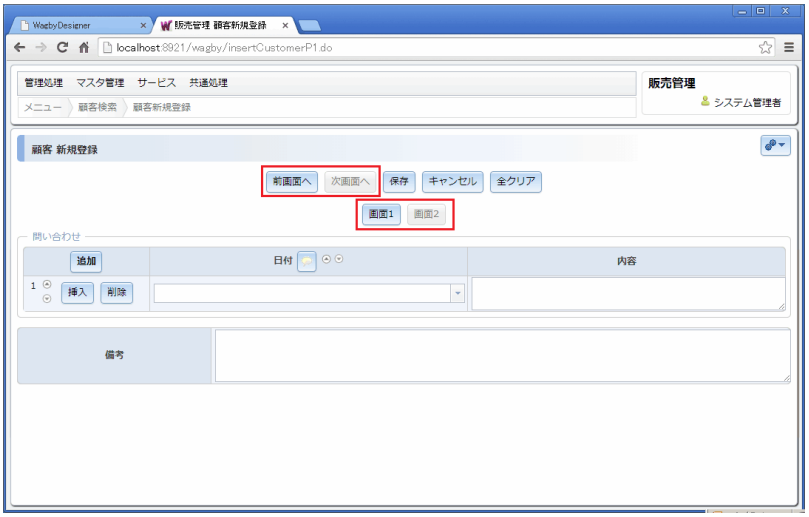


図 二枚目の画面

詳細画面もウィザードで分割されています。

図 詳細画面の例

定義方法

顧客モデルの各項目に「ウィザード画面番号」を設定します。

数値は画面番号を表しています。上記の例の場合、「顧客ID」や「氏名」などの項目は1つ目の画面に表示され、「問合せ」や「備考」などの項目は2つ目の画面に表示されます。

画面番号は上から順に指定していく必要はありません。例えば図の設定で、「電子メール」項目のウィザード画面番号を ″2″ と指定することは可能です。（2つ目の画面に表示されるようになります。）

割り当てしない項目はウィザードの対象外となります。

図 ウィザード画面番号を割り当てる

一括設定を利用する

ウィザード画面番号の設定は「モデル項目の一括設定」を使うと、さらに効率よく行うことができます。

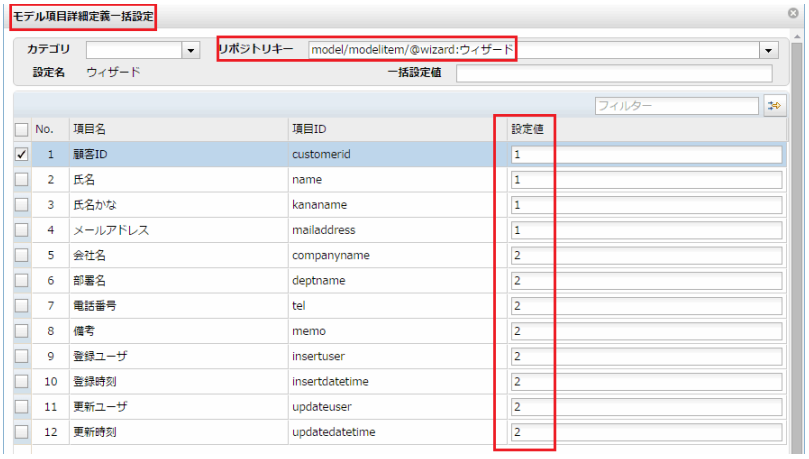


図 一括設定を利用する

タブレイアウトとの違い

類似のユーザーインタフェースに「タブレイアウト」があります。 両方式の相違点は次のとおりです。

ウィザード

順番に従って入力を行なう方式です。順番に、業務のルール（意図）が含まれます。ある画面に必須項目が用意されていた場合、同項目を入力しなければ、次の画面に進むことができません。

タブレイアウト

入力項目をグループ化したものです。入力の順番を意識する必要はありません。

ウィザードとタブレイアウトのどちらが適切かは、開発する業務アプリケーションによって異なります。上記の違いを元に判断してください。

画面指定ボタン

画面例

画面指定ボタンを非表示にした例を図に示します。

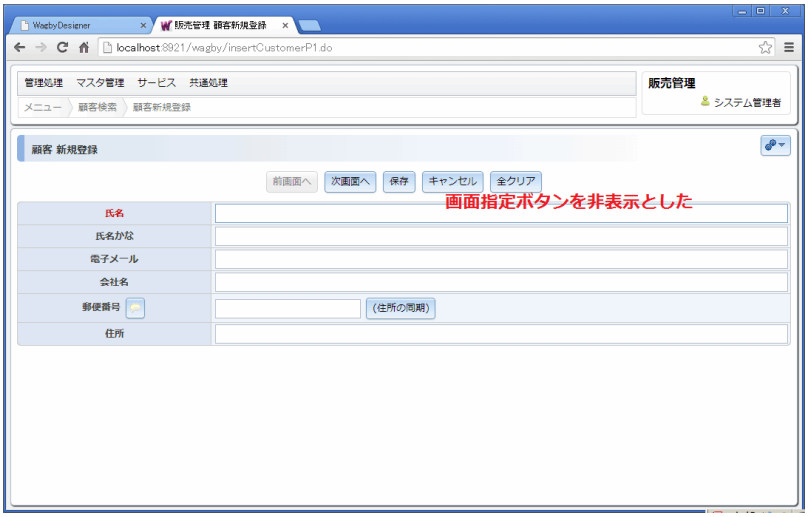


図 画面指定ボタンを非表示にする

定義方法

画面タブ内の「その他」タブを開きます。「ウィザード>画面ボタン」の「表示する」設定を解除します。

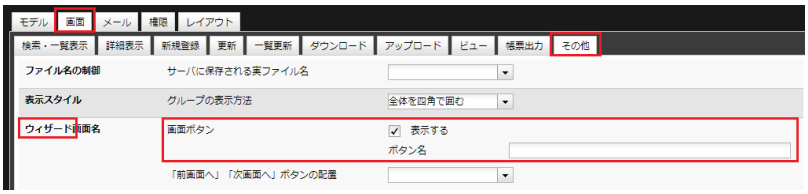


図 画面ボタンの表示制御

画面指定ボタンの名称を指定する

画面指定ボタンの名称を指定することもできます。図はウィザード画面にそれぞれ「基本」「応用」と名付けた例です。

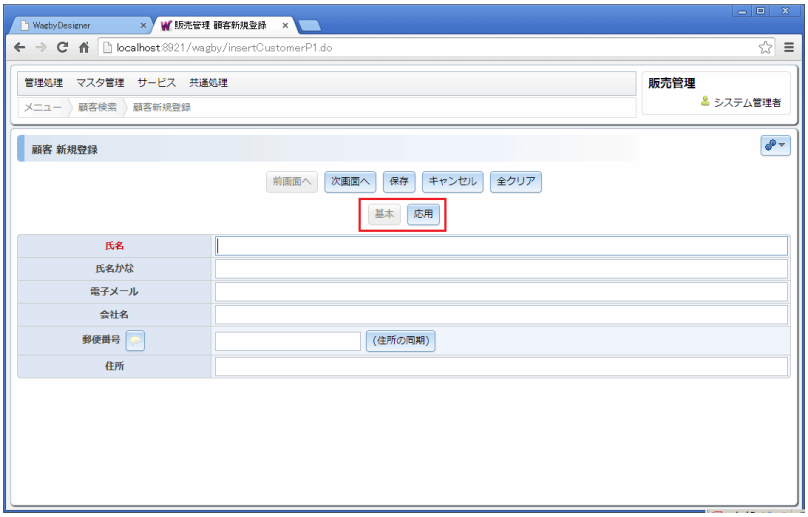


図 画面指定ボタンを命名する

ボタン名にコンマ区切りで画面名を指定します。ウィザードの画面数と合致するようにしてください。

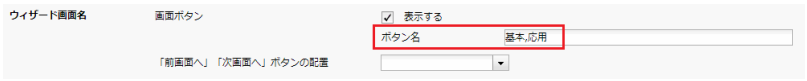


図 画面指定ボタンの設定

画面移動ボタン

画面例

画面移動ボタンの表示位置を指定することができます。標準値は空白で、"上"と同等です。

これを"下"にすると、画面指定ボタンと同じ位置に表示されるようになります。

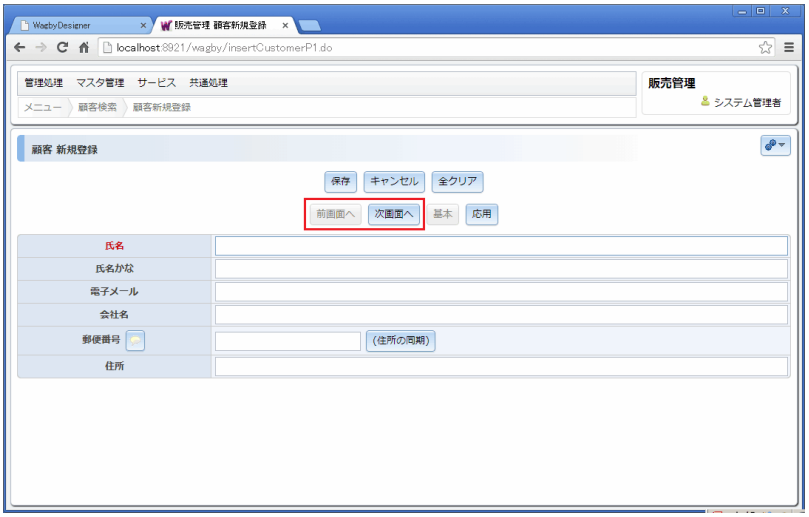


図 画面移動ボタンの表示位置が下の例

さらに、画面移動ボタンを非表示にすることもできます。

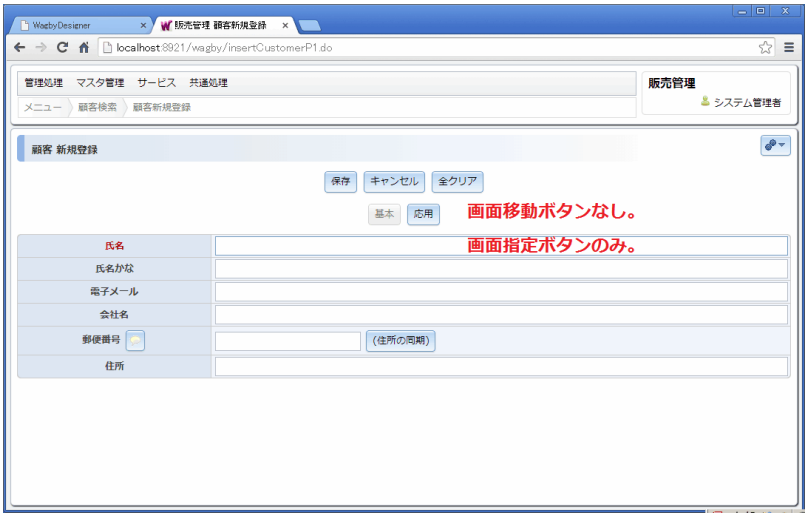


図 画面移動ボタンを非表示とした例

定義方法

「前画面へ」「次画面へ」ボタンの配置を選択肢から指定します。

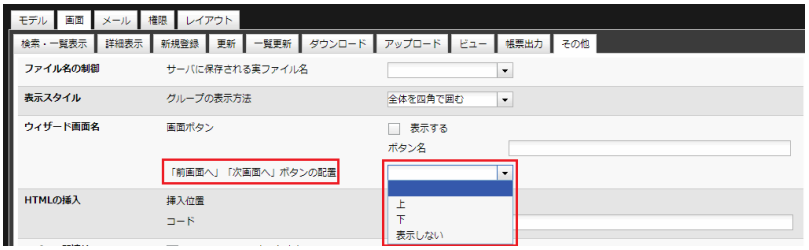


図 画面指定ボタンの設定

全画面共通の項目

図, 14は、氏名項目をすべてのウィザード画面で共通に用いると指定した例です。図で入力した値が、図でも読み取り専用項目として表示されていることがわかります。

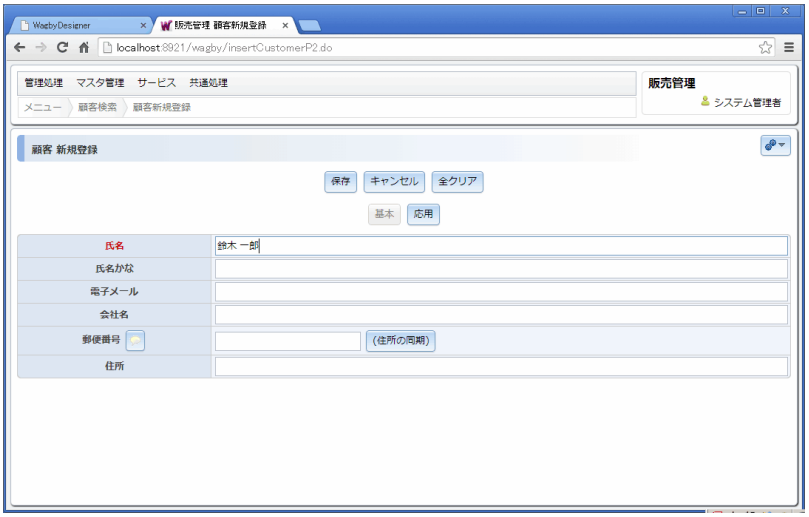


図 氏名項目を共通とした例(1)

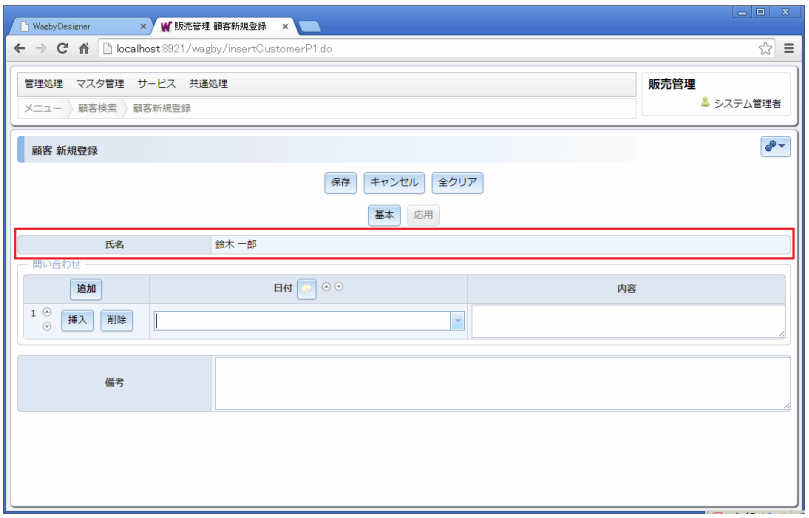


図 氏名項目を共通とした例(2)

この設定は、どのデータか特定できるような項目で、登録・更新時に常に画面に表示させながら別の項目も入力したい、という場合に有効です。

定義方法

全ての画面に表示させたい項目に "all" と記述します。

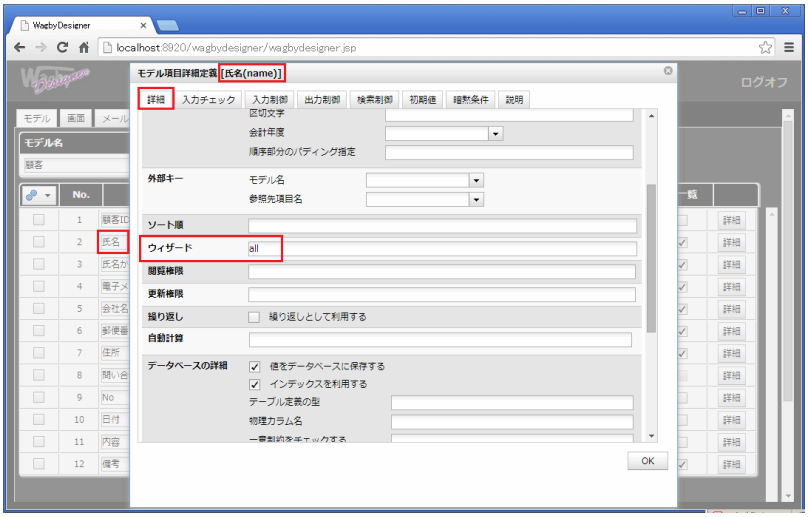


図 共通項目にはallと指定する

"all" で指定した項目の画面での並び順は、リポジトリで定義した順番により異なります。連続して表示させたい場合には、リポジトリ上でも連続させるなど、工夫してください。

確認画面を用意する

ウィザード利用と、確認画面を併用することができます。確認画面の詳細は"画面機能 > 新規登録 > 確認画面を用意する"をお読みください。

確認画面を用意すると、登録ボタンが "登録確認" となります。

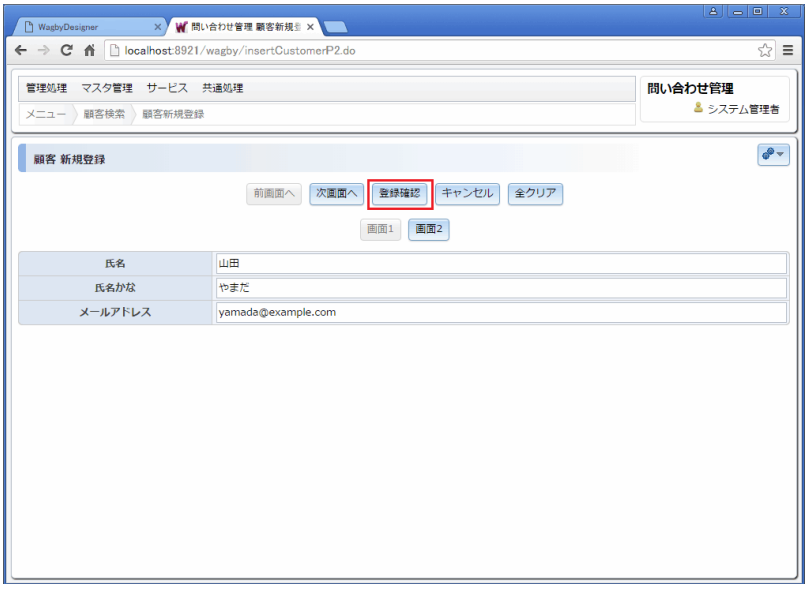


図 確認画面を用意したウィザード登録画面(1)

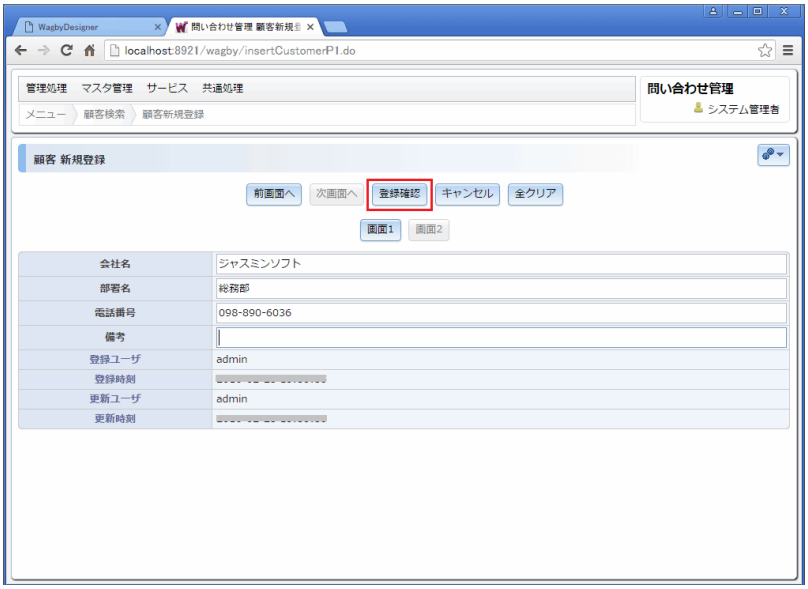


図 確認画面を用意したウィザード登録画面(2)

確認画面では、ウィザードとして分割されていた各画面を統合した一つの画面として内容を確認できます。

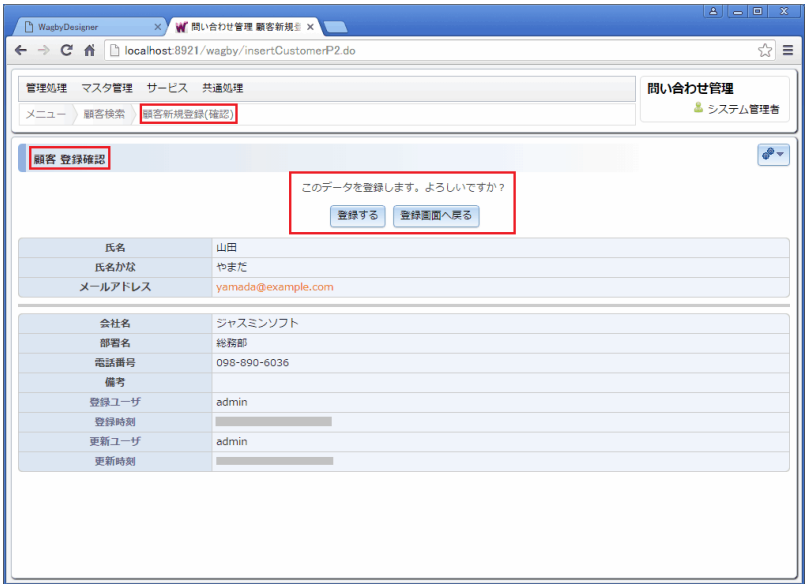


図 確認画面を用意したウィザード登録画面(3)

ウィザード表示を一つにまとめる

詳細画面を(確認画面のように)一画面にまとめて表示することができます。

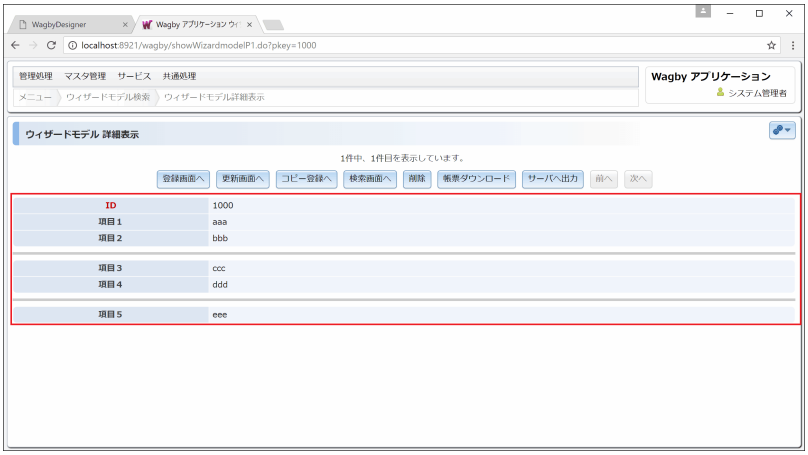


図 詳細画面をまとめて表示した例

定義方法

この例では、入力時は3つのウィザード画面を利用するようにしています。

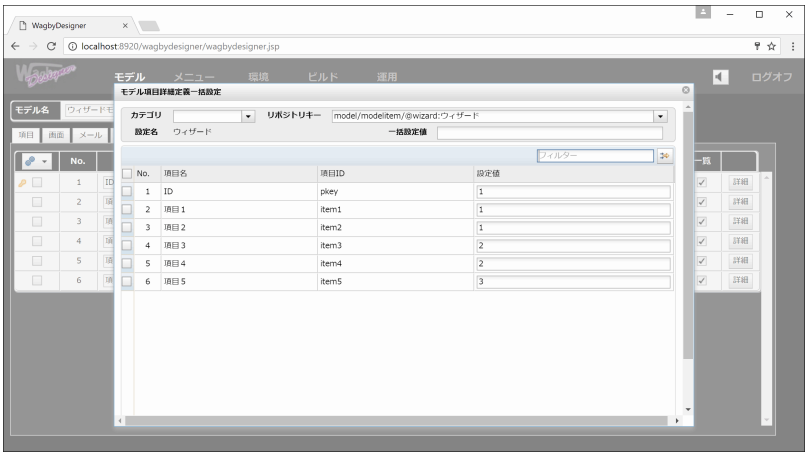


図 ウィザードの設定

「画面 ＞ 詳細表示 ＞ ウィザード表示を一つにまとめる」を有効にします。

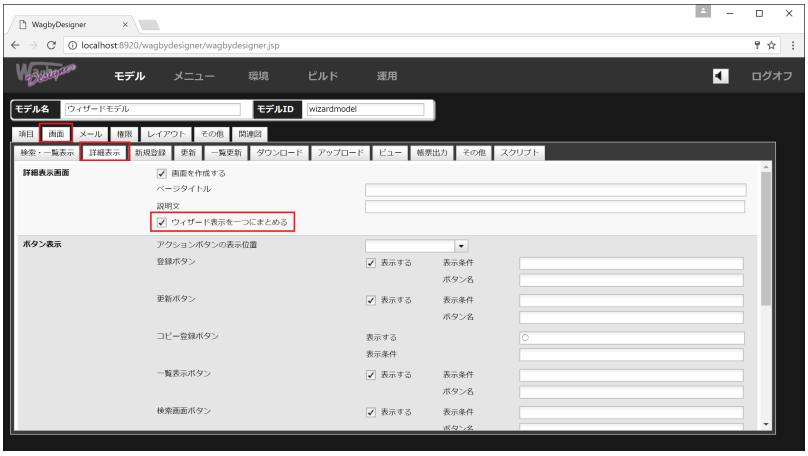


図 ウィザード表示を一つにまとめる

条件によってボタンを制御する

「画面指定ボタン」を条件によって非表示とするような制御を行うことができます。

条件を指定しない場合の動作は下図のようになります。画面指定ボタンのボタン名は「画面1」「画面2」となっており、いずれも押下可能です。

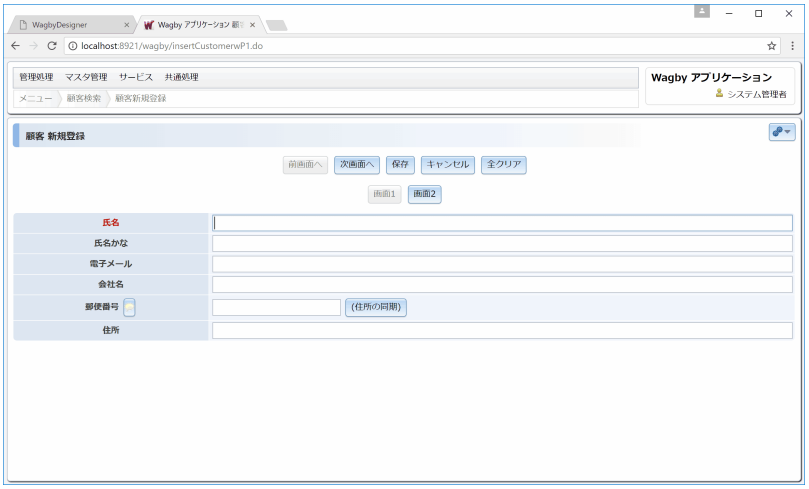


図 標準の画面指定ボタン(1)

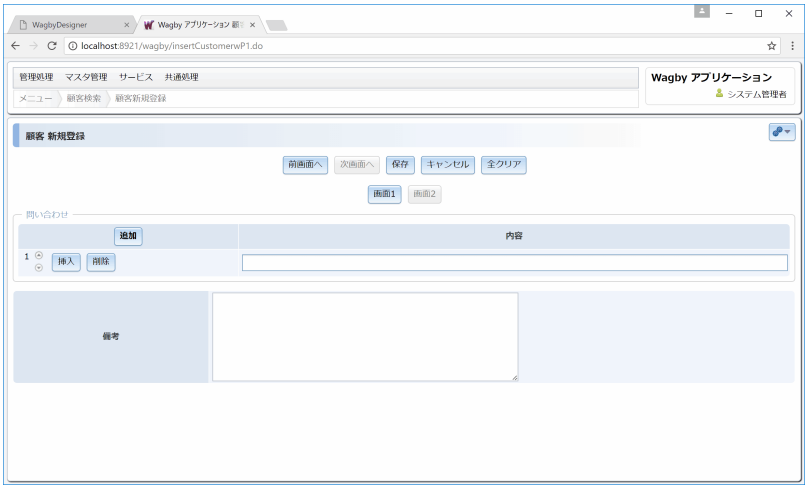


図 標準の画面指定ボタン(2)

定義方法

「画面 ＞ その他」タブから、「ウィザード」設定欄を開きます。ボタン名は空欄になっています。

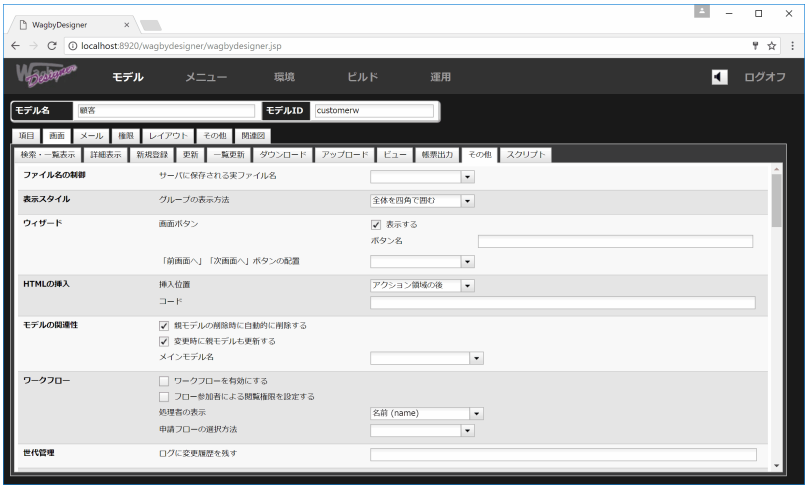


図 ウィザード設定欄

最初に、ボタン名を設定します。ここでは二画面のウィザードですので、コンマ区切りで二つのボタン名を指定しています。

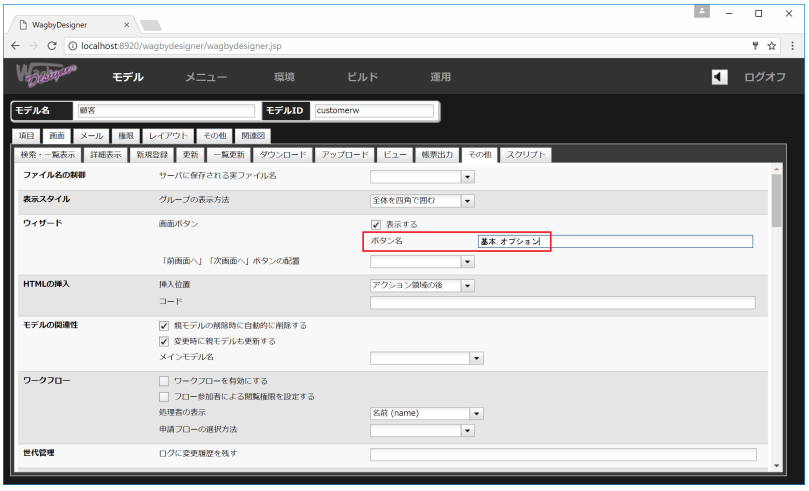


図 ボタン名を設定する

ボタンの数に応じて、「表示条件」の入力欄が用意されます。

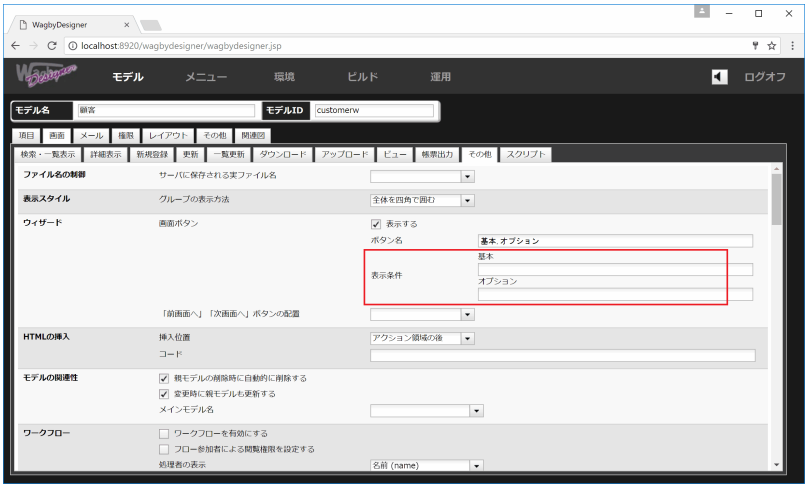


図 表示条件の入力欄が用意される

表示条件（式）を設定します。常に表示する場合は、空白とします。

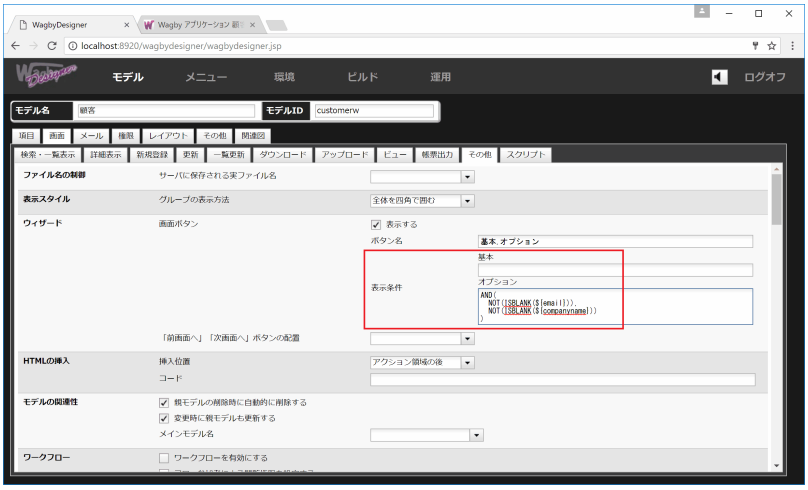


図 表示条件を設定する

今回の例は、次の式を設定しました。メールアドレス項目と、会社名項目の両方が入力されたらボタンが表示されるように、としています。

```
AND(
  NOT(ISBLANK(${email})),
  NOT(ISBLANK(${companyname}))
)
```

ここで指定した項目の値が変更されたら、画面を再描画するようにします。再描画によりサーバの判定処理が有効になります。

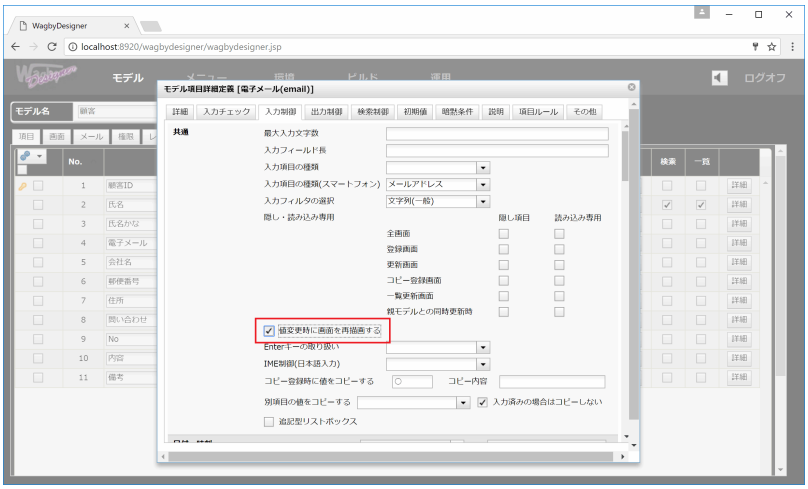


図 項目の値が変わったら画面を再描画する

実行

新規登録画面を開きます。メールアドレス項目と、会社名項目が未入力のため、「オプション」ボタンは表示されていません。「基本」ボタンは表示されていますが、自画面なので押下できないようになっています。（これは正常な動作です。）

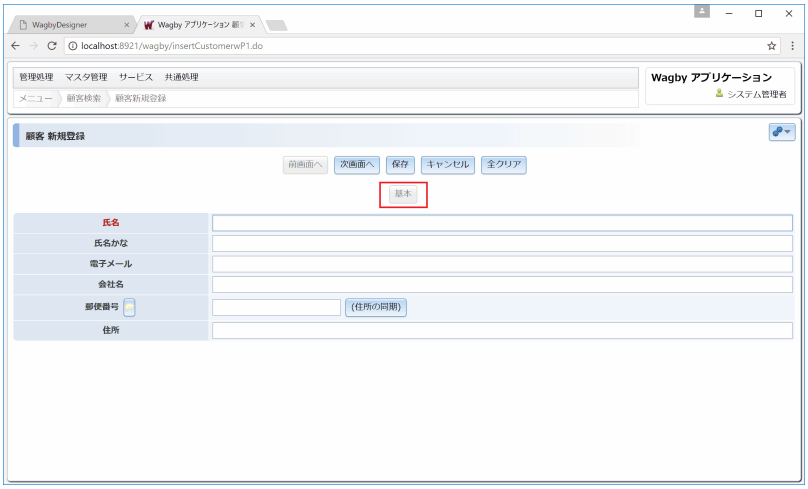


図 新規登録画面を開く

必要な項目を設定すると（画面が再描画され）「オプション」ボタンが表示されました。

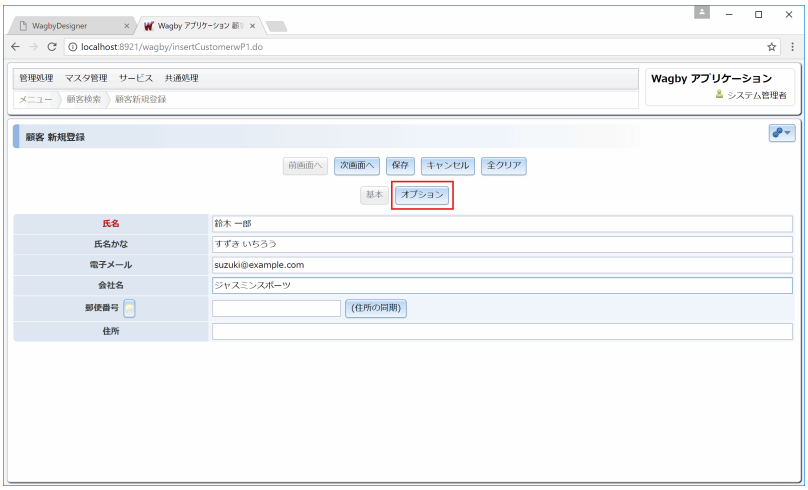


図 オプションボタンが表示される

ウィザードを移動すると、「基本」ボタンを押下することができるようになっていることがわかります。

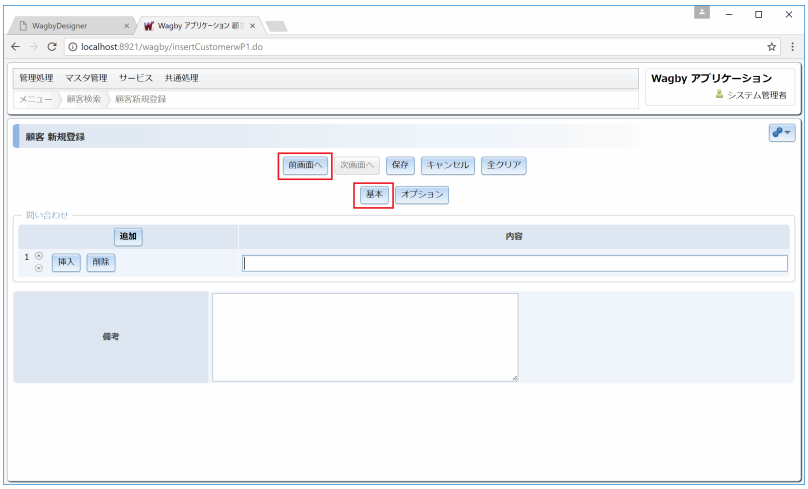


図 ウィザードを移動した場合

画面移動ボタンを非表示にする

上の例で、さらに画面移動ボタンを「表示しない」に設定すると、ウィザード画面の遷移を厳密に制御することができます。

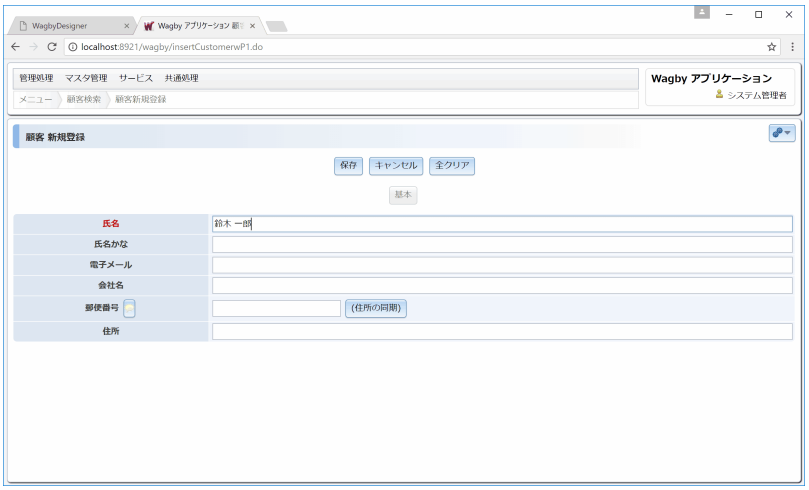


図 画面移動ボタンを表示しない

仕様と制約

繰り返しコンテナを分割しない

繰り返しコンテナ項目を利用する場合、同一コンテナ内の項目には同じページ番号を指定してください。

設定する数値は連続させる

ウィザードに設定する数値は連続である必要があります。欠番は認められません。

マップビュー欄の設定を行った項目は、1つの画面に収まるように設定する

マップビューとウィザードを併用する場合、マップビューの設定を行った項目は、同じページ番号を指定してください。（そのように設定しない場合、マップビューが正常に機能しなくなります。）

登録画面/更新画面は必要

ウィザードを利用するモデルでは、登録画面・更新画面は必ず必要になります。

入力チェックのタイミング

入力チェックは画面毎に行われます。エラーがあった場合は、次の画面へ進むことができません。

カレンダビュー（クラシック）

例

カレンダビュー機能を有効にした「スケジュール」モデルを用意した例を示します。メニューには、カレンダビューに直接、遷移するためのボタンを用意しています。

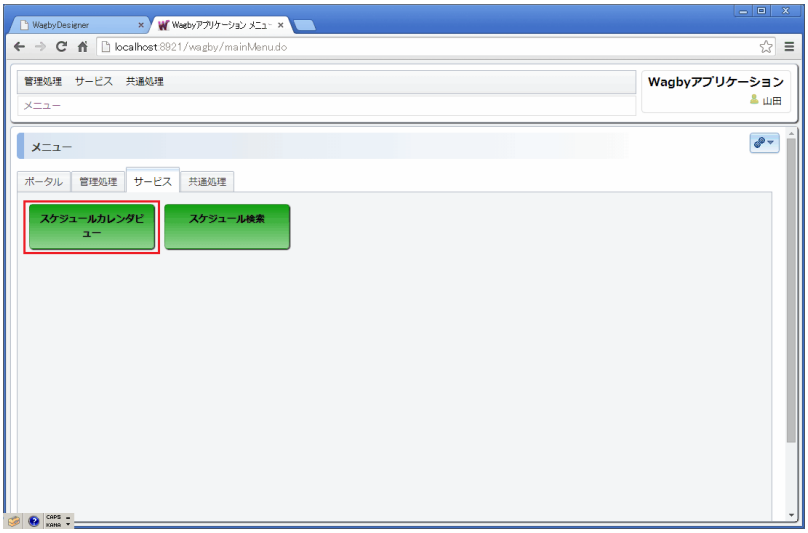


図 カレンダビューを開くためのボタン

図は、カレンダビューを開いた直後の画面です。標準では「月単位」の表示ですが、「日」や「週」で表示させることもできます。

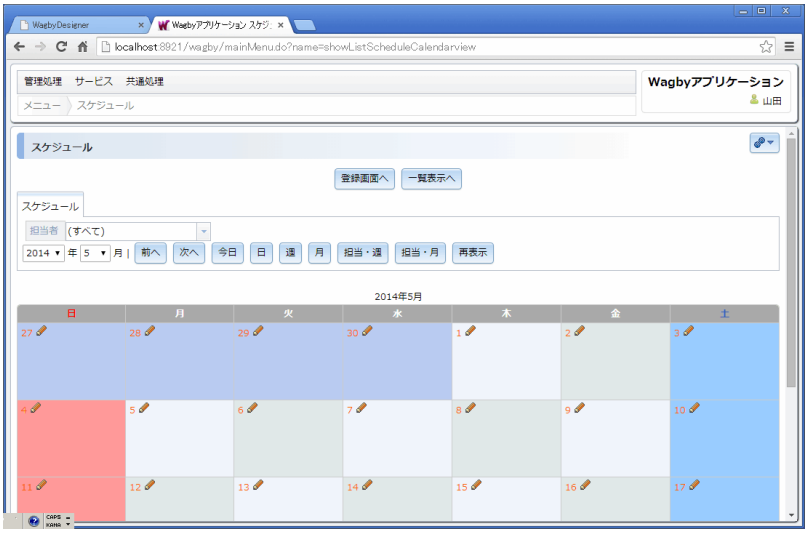


図 カレンダビュー

日付の欄に、鉛筆型アイコンが用意されています。これをクリックすると、当該日のデータを登録できます。

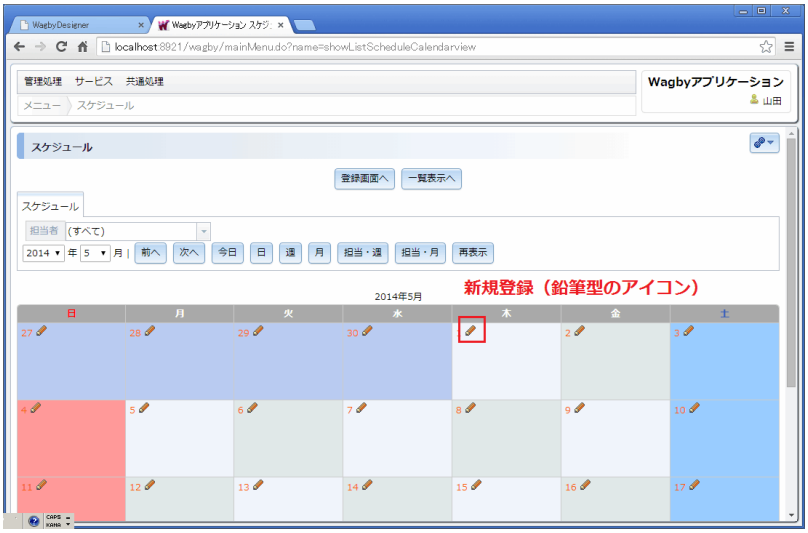


図 鉛筆型アイコンをクリックする

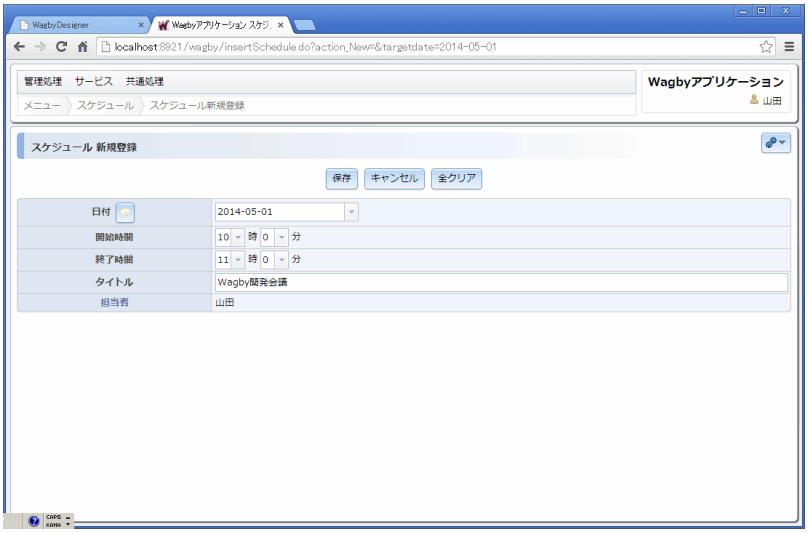


図 スケジュール新規登録画面が開く

スケジュールを登録すると、再びカレンダビューに戻ります。

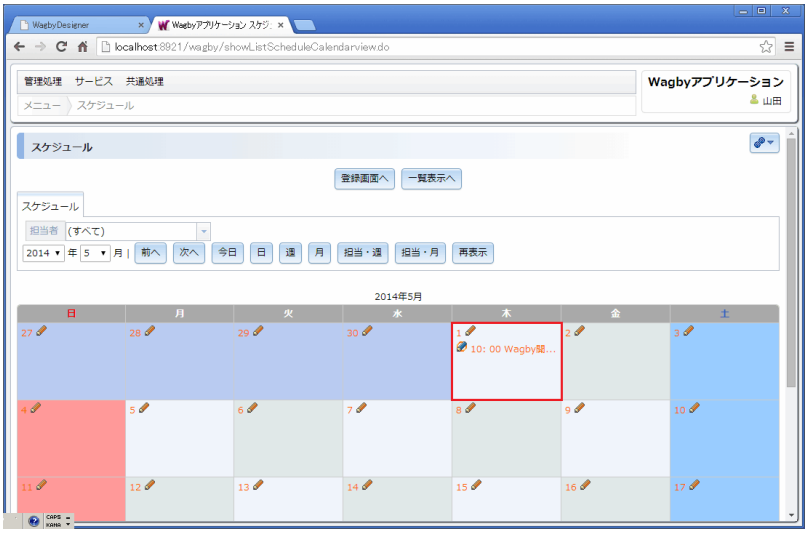


図 スケジュールが登録された

スケジュール毎に用意された鉛筆アイコンをクリックすると、当該スケジュールの更新画面が開きます。

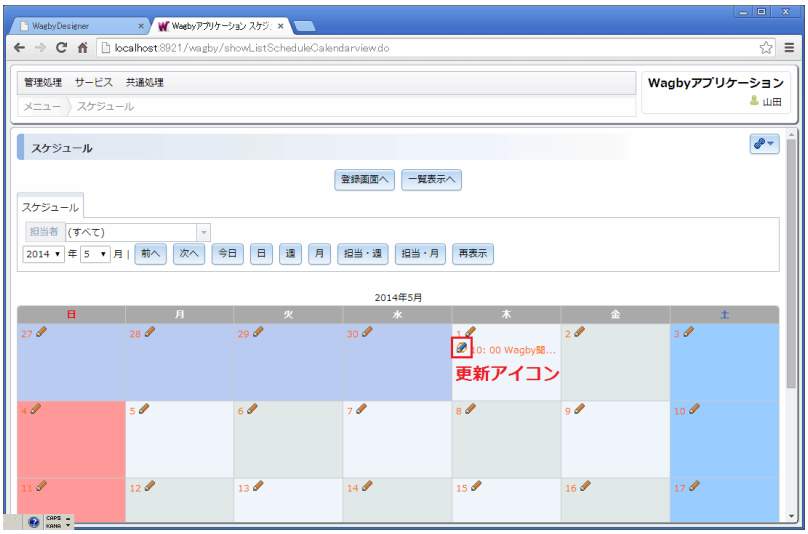


図 更新用の鉛筆型アイコンをクリックする

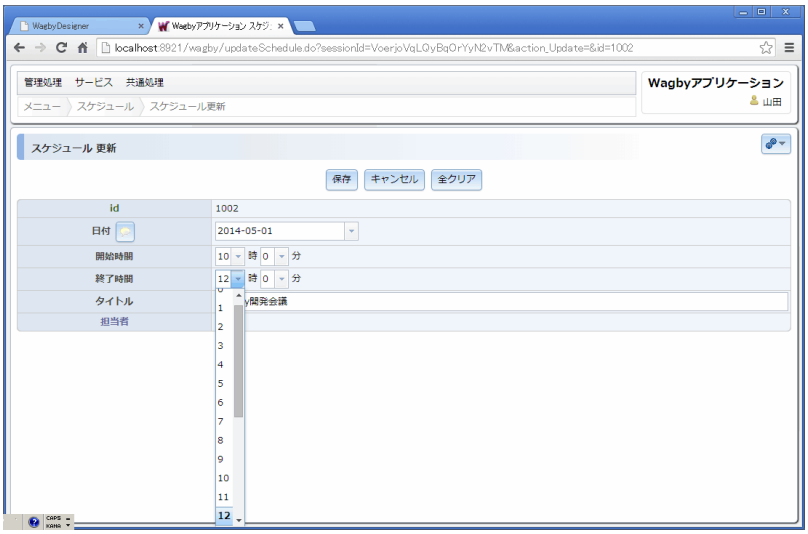


図 スケジュール更新画面が開く

ここでは終了時間を変更しました。タイトルにマウスをあてると、スケジュールの概要が表示されます。終了時間が変更されていることがわかります。

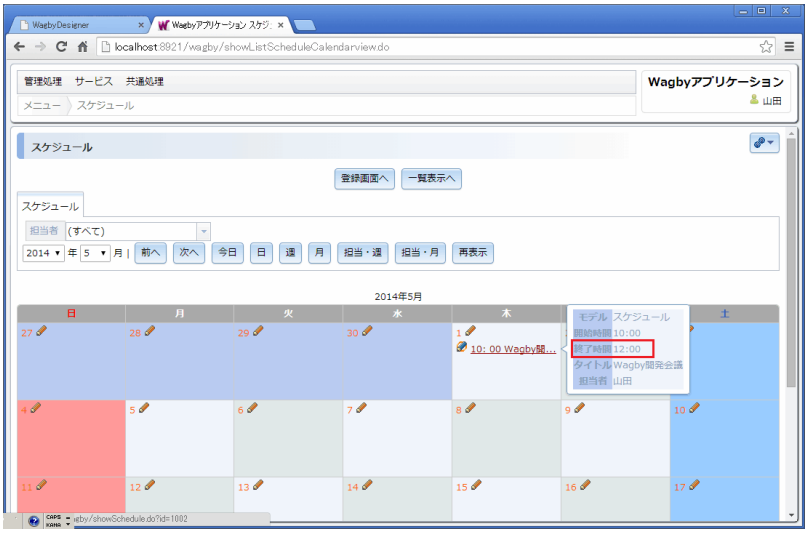


図 スケジュールの概要を表示する

定義方法

モデルの定義

スケジュールモデルの定義を図に示します。

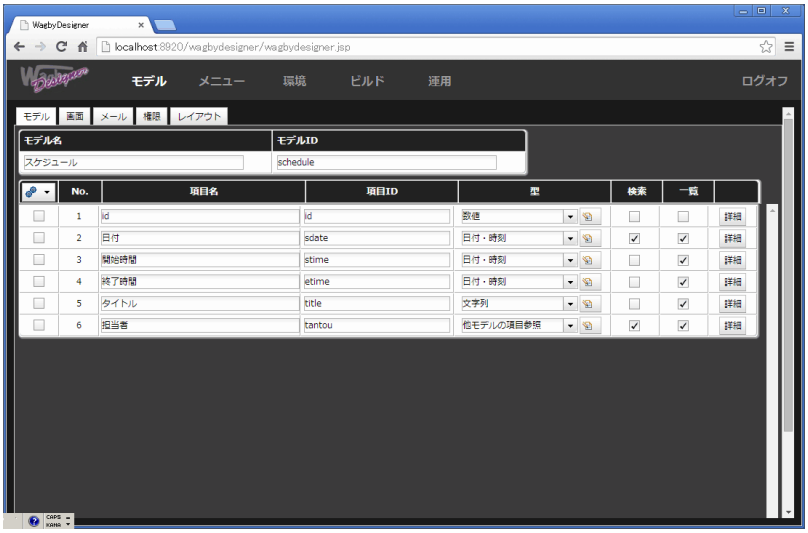


図 スケジュールモデルの定義

開始時間と終了時間は「時刻」とします。検索項目は日付および担当者とします。（時刻部分は検索項目に含めません。）

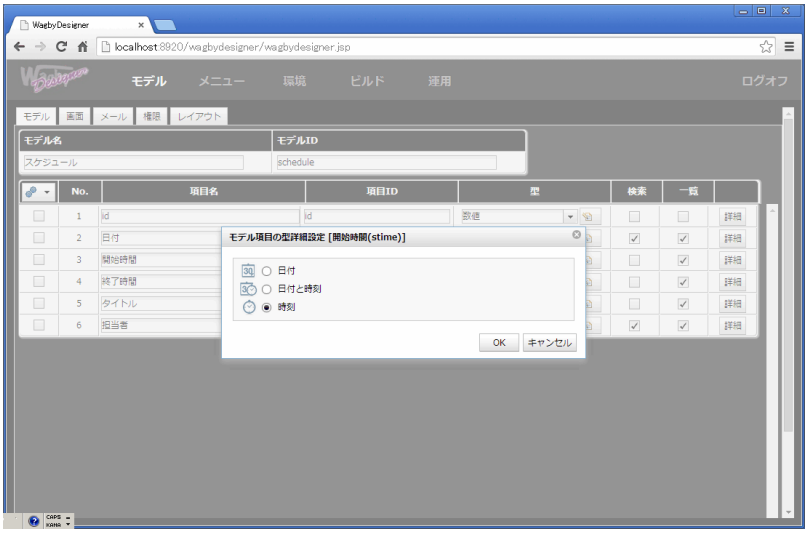


図 日付・時刻型の詳細で、時刻を選択する

担当者の項目は、Wagbyが提供するアカウント情報（juser）の名前を利用します。

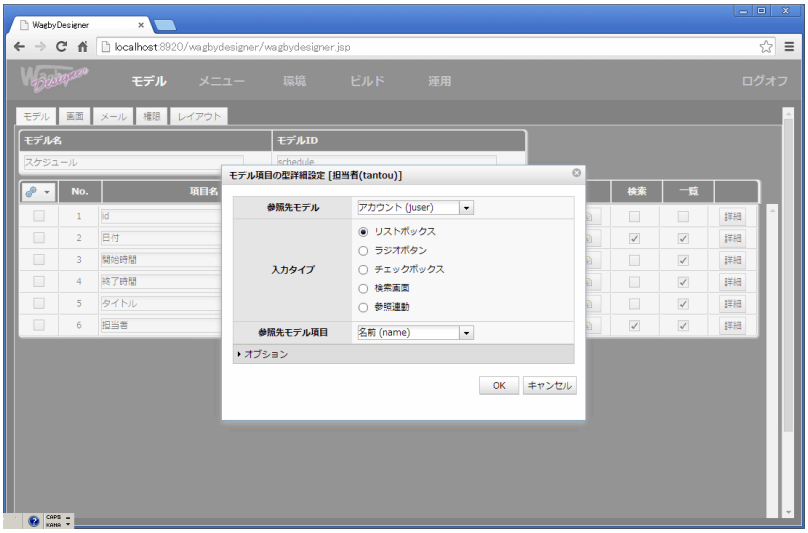


図 アカウントモデルを参照する

開始時間、終了時間の詳細を設定します。いずれもリストボックスで入力できるようにします。

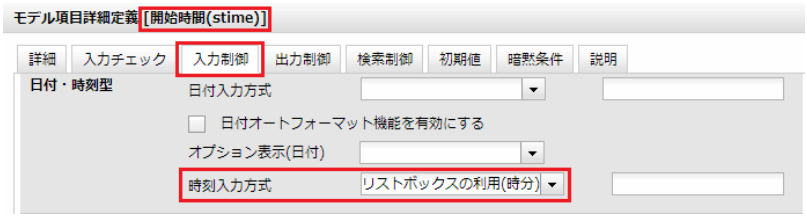


図 開始時間、終了時間はリストボックスで入力させる

担当者項目はログオンアカウントに紐付けます。入力させないようにするため、読み込み専用とします。また、初期値を〝ログオンID〟とします。

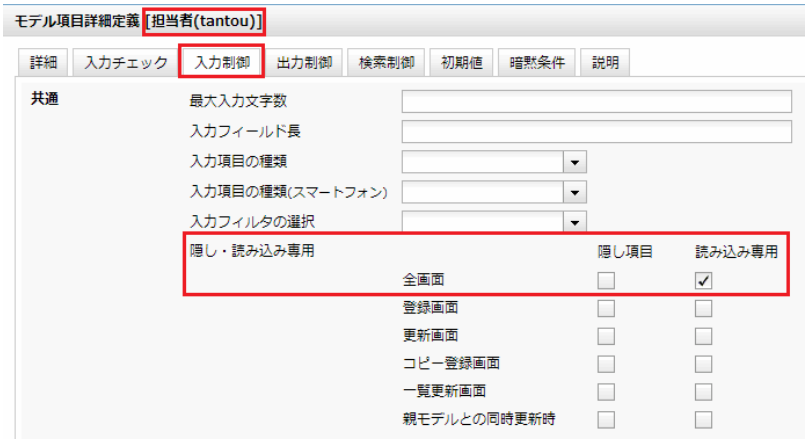


図 担当者は自動設定させるため、読み込み専用とする

図 担当者はログオンアカウントを初期値とする

カレンダビューの設定

「画面>ビュー>カレンダビュー」を開きます。「画面を作成する」をチェックし、ページタイトルを設定します。

「表示方式」を〝クラシック〟にします。

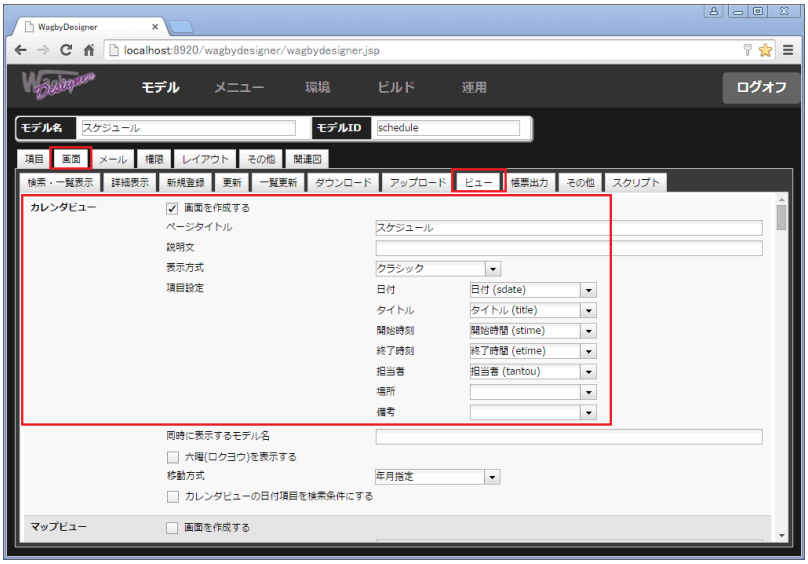


図 カレンダビューの設定

項目設定欄に、モデルの項目を割り当てます。次のルールがあります。

設定値	対応する項目の型	説明	必須
日付	日付型	この日付情報を元にカレンダビューに表示を行います。	○
タイトル	文字列、整数、モデル参照	カレンダビューに表示する項目として定義します。	○
開始時刻	時刻	カレンダビューに表示する際、時刻順で並び替えを行います。	−
終了時刻	時刻	−	−
担当者	文字列、整数、モデル参照	カレンダビューに表示する項目として定義します。	−
場所			−
備考			−

メニューの設定

「スケジュールカレンダビュー」ボタンをメニューに用意します。

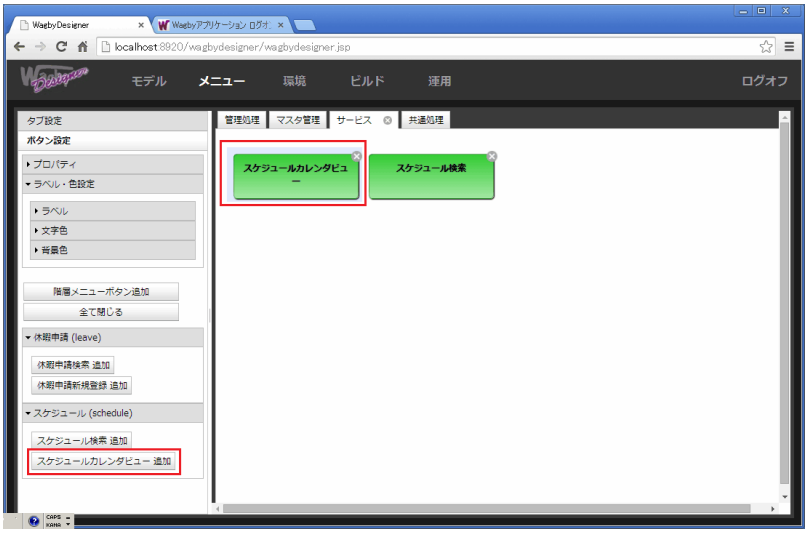


図 メニューの設定

複数のモデルを同時に表示する

1つのカレンダに複数モデルの情報を表示させることができます。図は「イベント」と「問合せ」の2つのモデルを表示させた例です。検索条件部はタブによって切り替えることができます。

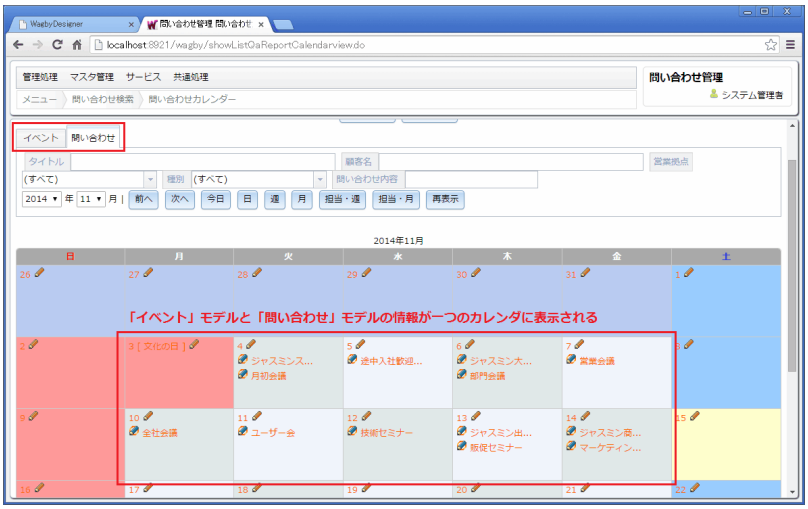


図 複数のモデルを1つのカレンダに表示する

定義方法

「イベント」モデルを用意してみます。日付とタイトルを、カレンダビューに設定します。

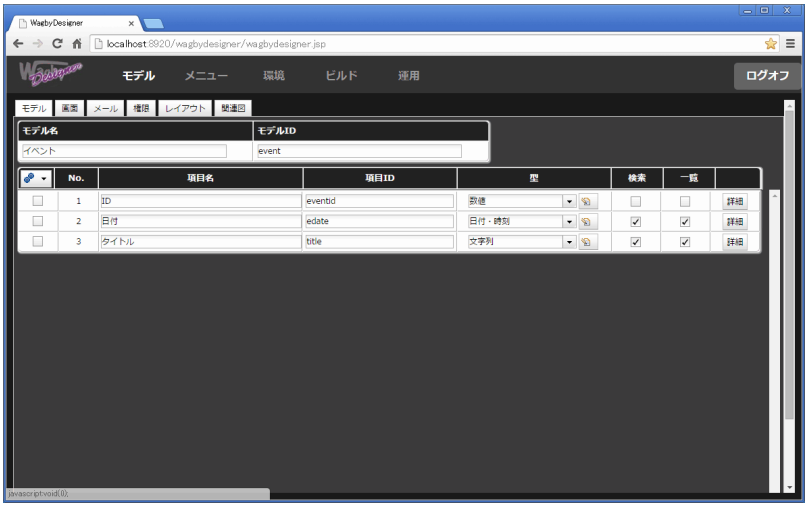


図 イベントモデル

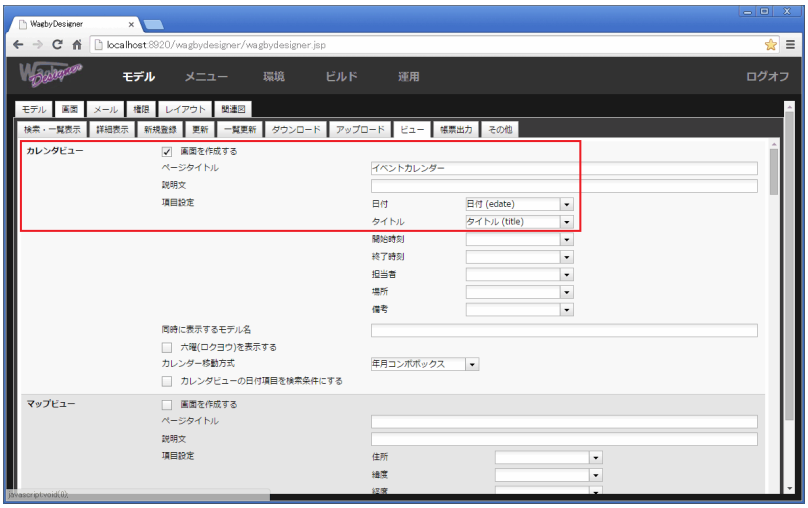


図 イベントモデルにカレンダビューを適用する。

もう一つの「問合せ」モデルを用意します。ここで、問合せモデルのカレンダビューで「同時に表示するモデル」に、イベントモデルを指定します。（モデルID "event" を記載します。）(*1)

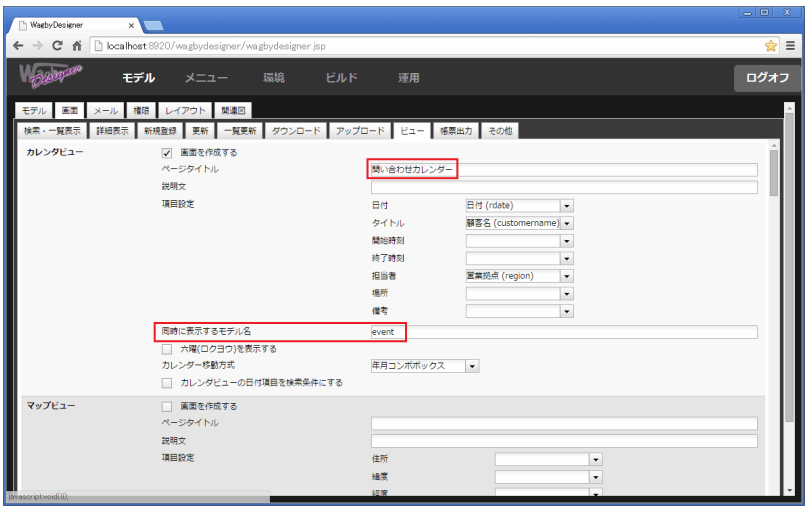


図 問合せモデルのカレンダビュー定義

(215)

複数のモデルを指定することもできます。2つ以上指定する場合は、「,（カンマ）」区切りでモデルIDを記述します。 例：modelA,modelB

六曜を表示する

「大安」などの六曜（ロクヨウ）を表示させることができます。

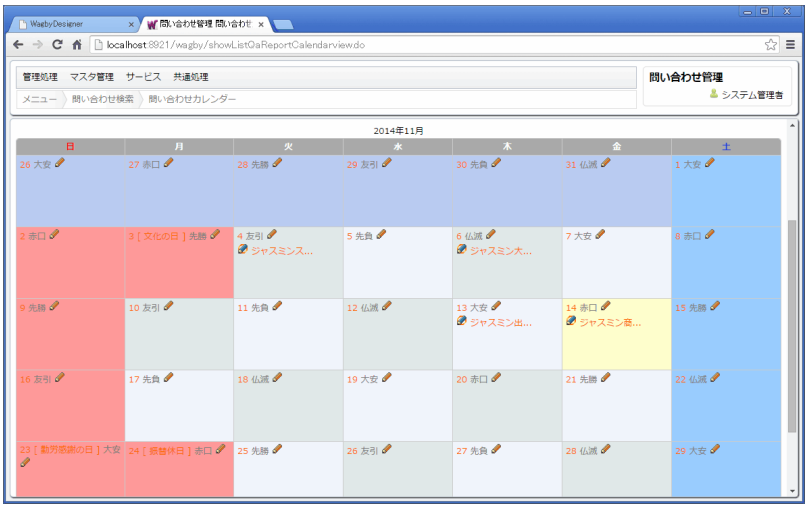


図 六曜を表示した例

定義方法

「六曜（ロクヨウ）を表示する」を有効にします。

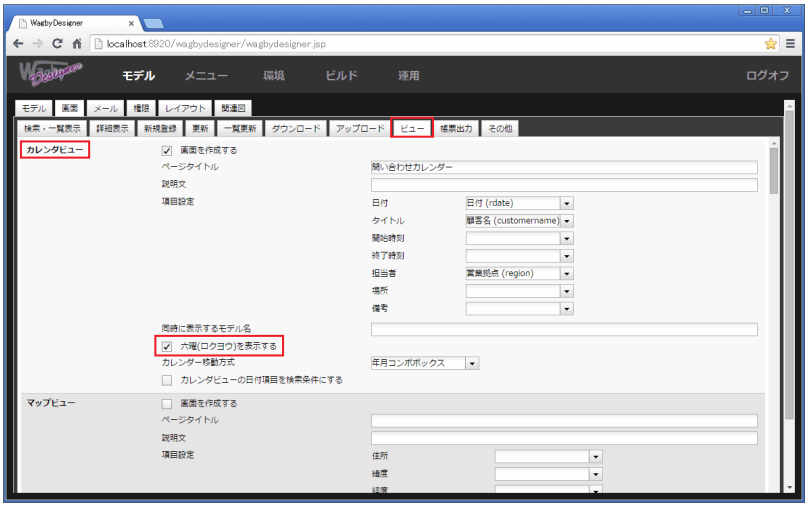


図 六曜の表示を有効にする

カレンダー移動方式

カレンダーの移動方式を「年月」「ステップ」から選択できます。

年月方式（標準）

(215) 1. イベントモデルでは同時に表示するモデルを指定していません。この場合、イベントモデルのカレンダビューは自分自身（イベントモデル）のみが表示されます。

年と月を選択するリストボックスが用意されます。選択肢を変更するとすぐに画面が再描画されます。

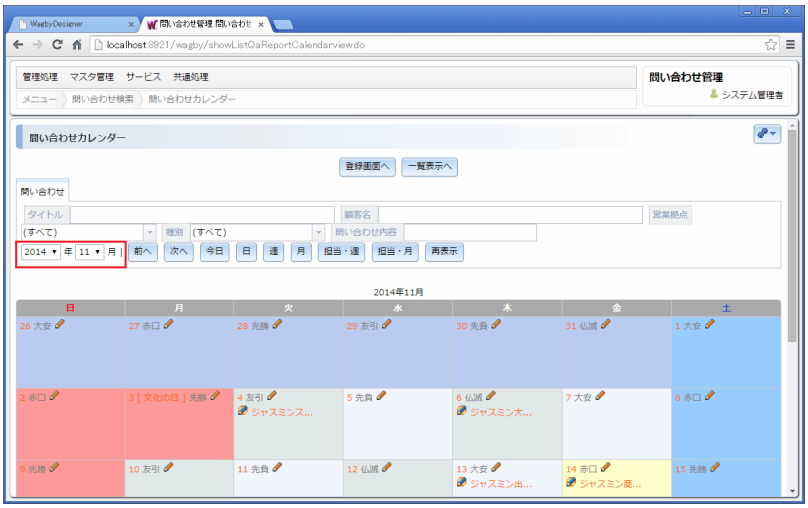


図 年月を指定した移動

ステップ方式

現在からいくつ、前後するか、という数値を指定したあとに「前へ」「次へ」ボタンを押下します。例えば2014年11月の時点で、移動ステップを ″2″ として「次へ」ボタンを押下すると、2015年1月が表示されます。

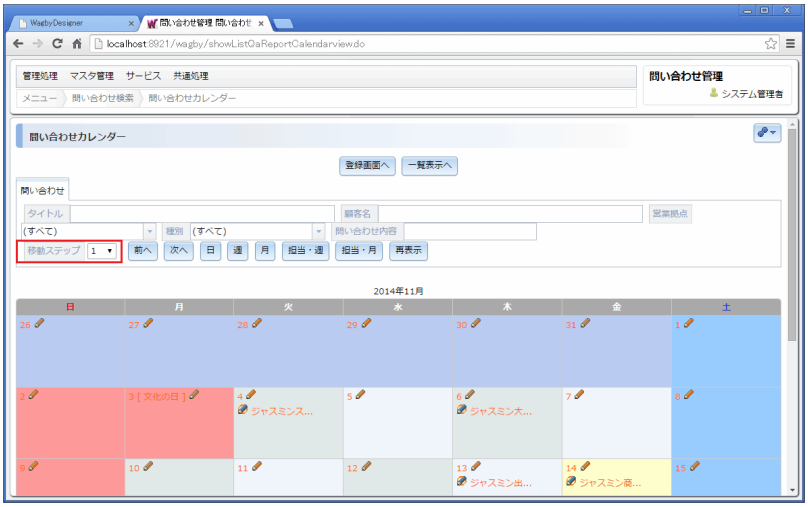


図 ステップを指定した移動

定義方法

「カレンダー移動方式」を選択します。標準は年月方式です。

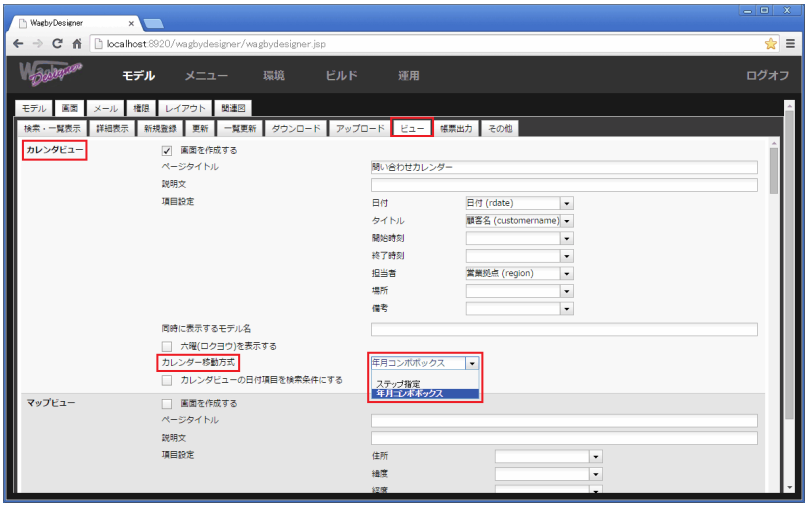


図 カレンダー移動方式の設定

日付を検索条件に加える

カレンダビューでは、軸として指定された日付項目は通常、検索条件部には表示されません。これは日付で表示を切り替える機能が検索と同等と解釈しているためです。

この日付項目を検索条件部に含めることができます。これによって、例えば「11月のカレンダ表示で、さらに1日から10日までのデータを絞込んで表示させる」といった使い方ができるようになります。

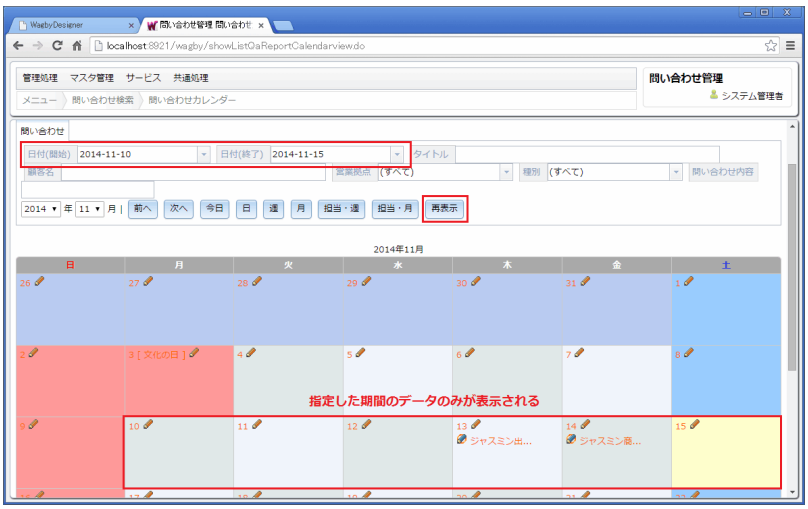


図 日付による表示絞込

定義方法

日付項目を検索条件に加えます。

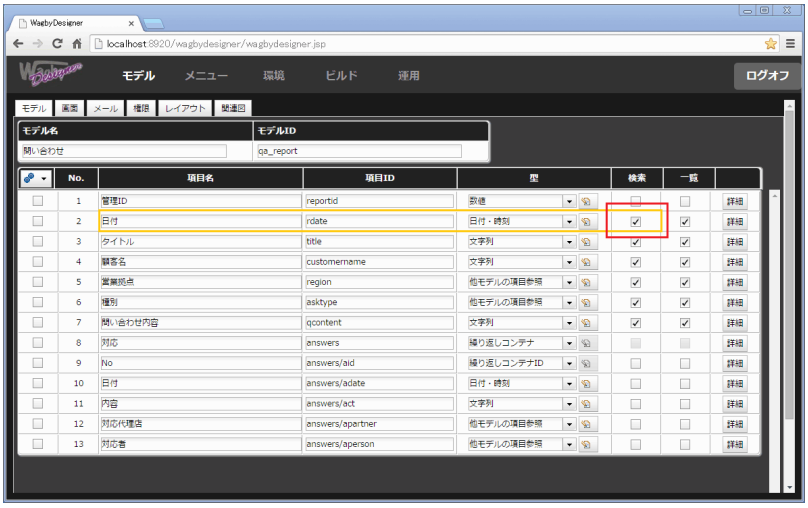


図 日付項目を検索条件とする

「カレンダビューの日付項目を検索条件にする」を有効にします。

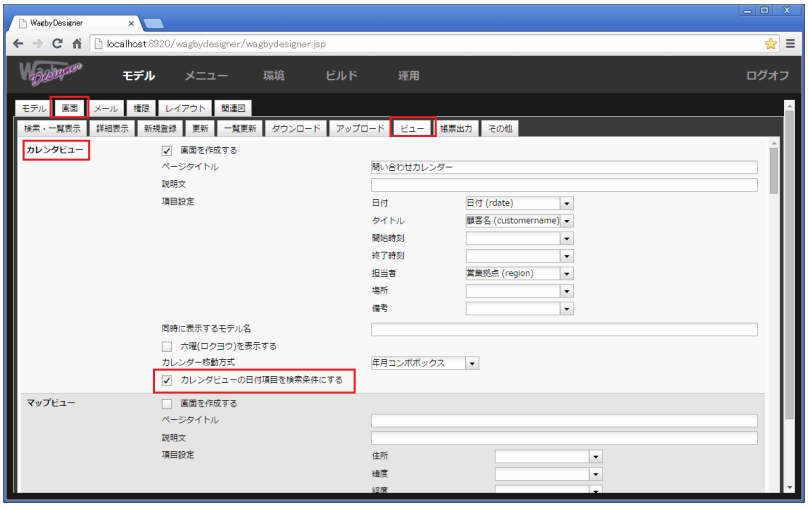


図 カレンダビューの日付項目を検索条件にする

ポータル画面に表示する

ここで用意したスケジュールモデルを、ポータル画面に表示することができます。 詳細は「ポータル>カレンダポートレット」をお読み下さい。

休日の扱い

同梱している休日情報

Thunderbird が提供する日本の公休日情報 "JapanHolidays.ics" を同梱しています。 これは iCalendar 形式で記述されています。

別の ics ファイルを指定する（国際化対応など）

他国の休日情報を適用することもできます。 詳細は"メッセージの変更・国際化 > 休日ファイルの変更"をお読みください。

独自の休日を設定する

会社の創立記念日など、独自の休日情報を設定することもできます。方法を説明します。

休日管理

Wagbyは標準で（システムタブ内に）休日モデル jholiday を用意しています。 メニューの「マスタ管理 > 休日...」を開きます。

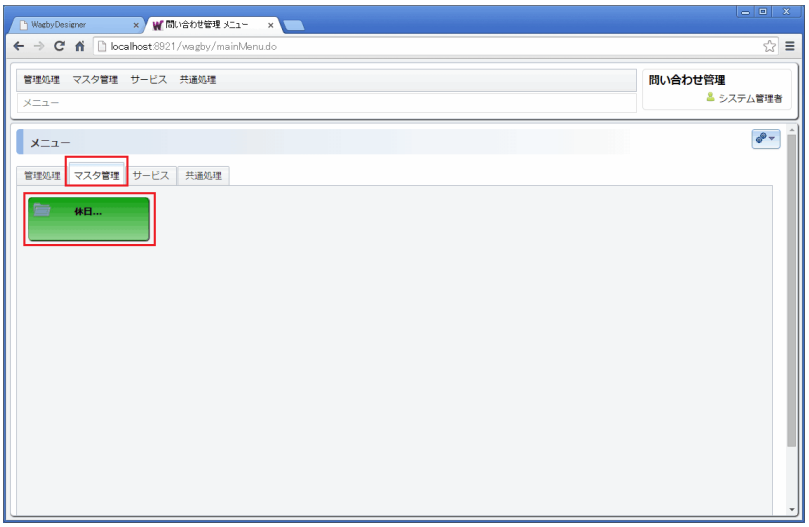


図 マスタ管理に用意された休日アイコン

休日は画面からの設定またはCSVファイルによる一括登録機能を用意しています。ここでは画面からの登録を説明します。

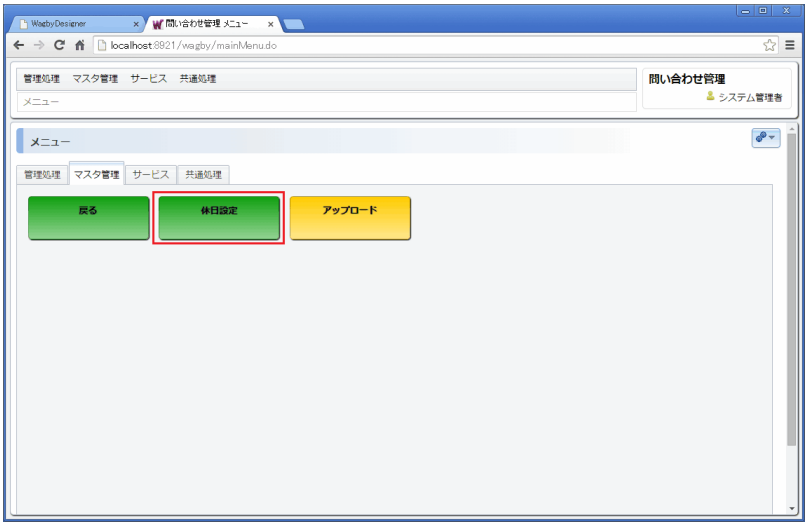


図 休日設定アイコン

カレンダから、休日にしたい日の、鉛筆アイコンを選択します。

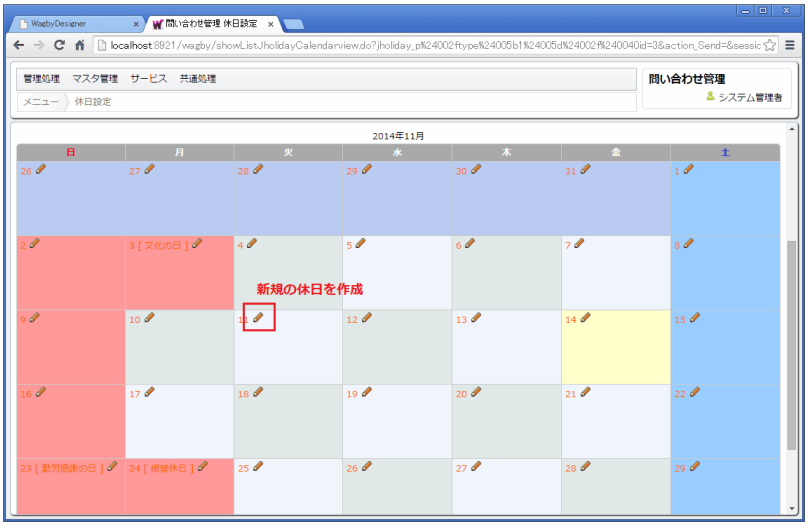


図 休日対象とする日を選択する

休日情報を設定します。

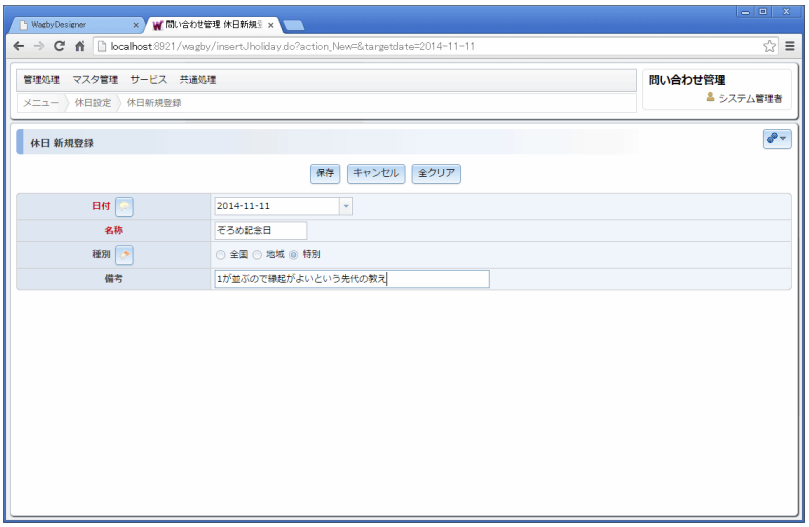


図 新しい休日を登録する

休日が追加されます。なお、この段階では背景は赤にはなりません。（背景が赤いのは、JapanHoliday.ics で用意された標準の休日情報となっており、独自に登録した休日とは区別できるようにしています。）(*2)

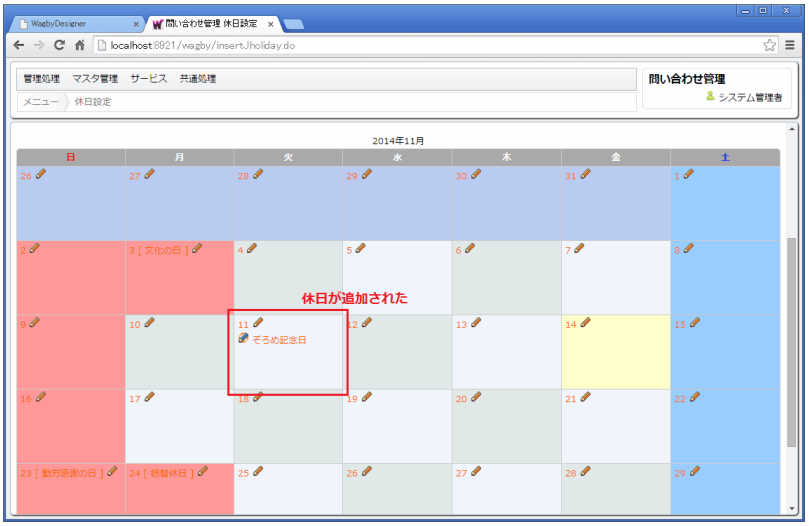


図 新しい休日が登録された

(216)

カレンダビューによる確認

「問合せ」モデルのカレンダビューを表示した例を示します。先に登録した休日は背景が赤くなっています。

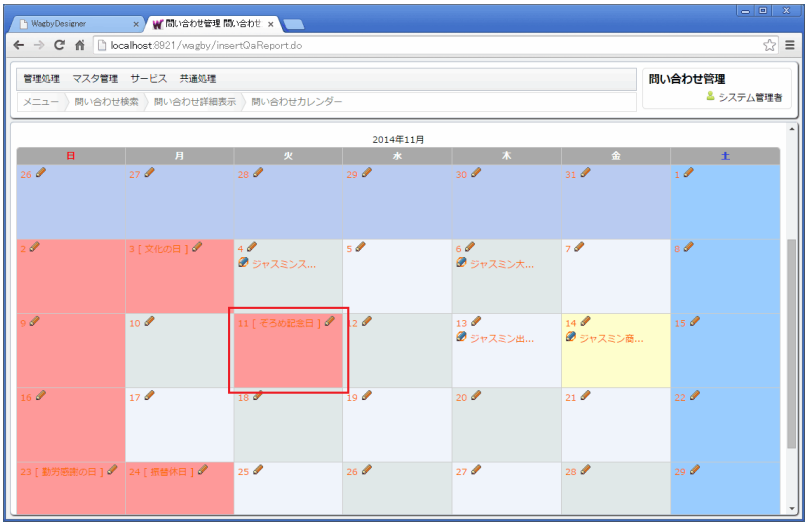


図 カレンダビューで休日を確認する

仕様・制約

- 月単位表示の場合、タイトル文字列は最長で6文字までが表示されます。7文字目以降は"..."と省略されます。
- 月単位表示の場合、担当者文字列は最長で10文字までが表示されます。11文字目以降は"..."と省略されます。
- カレンダビューの「日付」に指定されている項目は検索制御の必須チェックを設定できません。
- カレンダビューの「日付」に指定されている項目は値をデータベースに保存するを指定する必要があります。
- カレンダビューの「日付」に指定されている項目は繰り返しを設定できません。
- カレンダビューの「日付」に指定されている項目は繰り返しコンテナに含めることができません。
- 「日付と時刻」型かつ時刻部分の入力方式がリストボックスの項目を検索項目と指定したとき、時刻部分は手入力方式となります。(*3)

(216) 2. 同梱している休日情報と、ユーザーが登録した休日が重なった場合は、ユーザーが登録した休日情報が優先されます。

・　カレンダビューを「担当・週」形式で表示する場合、縦に並ぶアカウントは辞書順に整列します。（名前がアルファベットで表記されている場合はアルファベット順となりますが、名前が漢字で表示されている場合は文字コードでの辞書順となります。）
・　カレンダビューを「月」形式で表示する場合、同じ時刻の場合の表示順は次の通りです。
　・　R8.3.1までは表示順の規定がない。（並びは不定であった）
　・　R8.3.2からはタイトル順に表示する。

(217)

クラシックカレンダビューからモダンカレンダビューへの移行

クラシックカレンダビューで登録済みのデータを、モダンカレンダビューへ変換することができます。（逆は対応していません。）

具体的な方法は"サポート ＞ インポートとエクスポート機能活用ガイド(R8) ＞ クラシックカレンダビューのデータをモダンカレンダビューへ取り込む"をお読みください。

カレンダビュー（モダン）

例

モダンカレンダビューを用いた「スケジュール」モデルの例を紹介します。標準は「月」表示ですが、「日」「4日間」「週」を切り替えることができます。

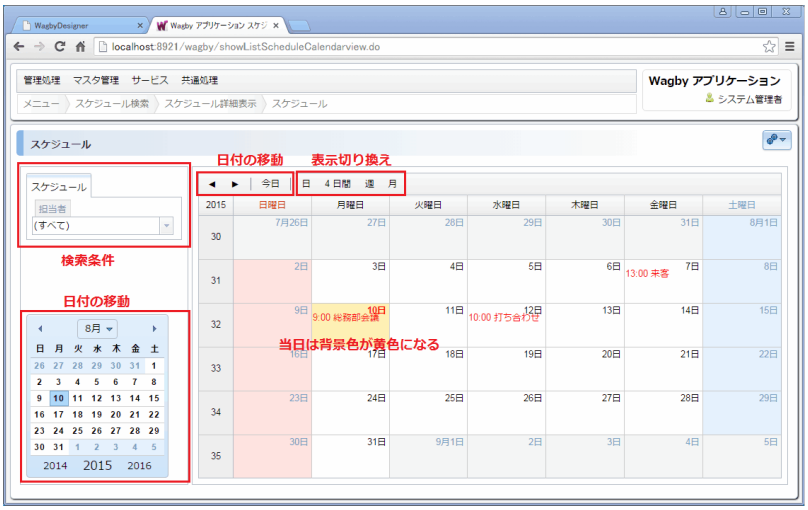


図 モダンカレンダビュー月表示

(217) 3. R8.2.3までは実行時エラーになります。R8.3.0以降およびR8.2.3向け2019.07パッチを適用することでエラーは回避されますが、時刻部分は手入力方式での運用となります。

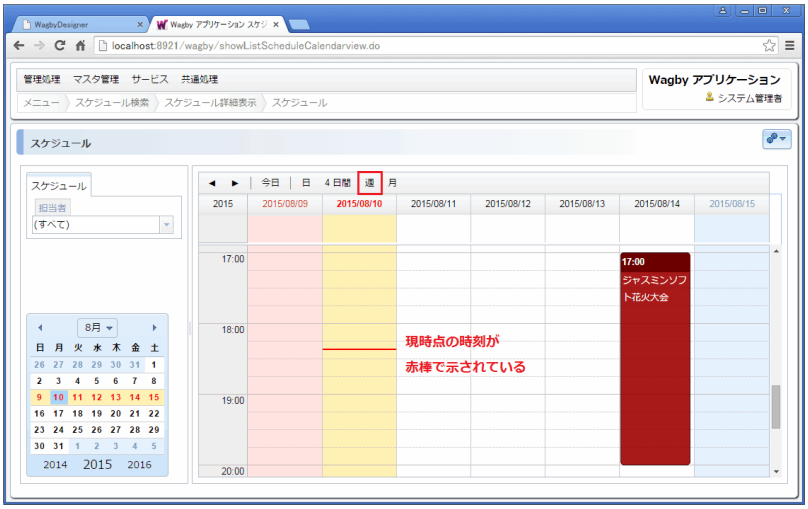


図 モダンカレンダビュー週表示

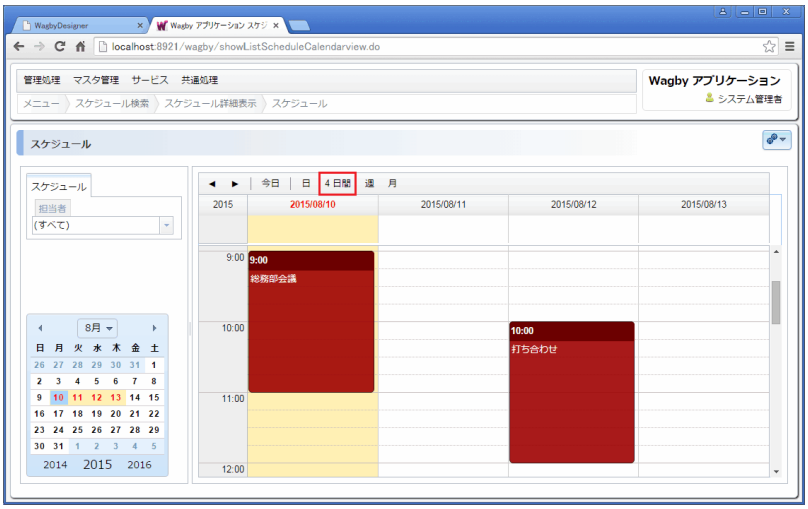


図 モダンカレンダビュー4日間表示

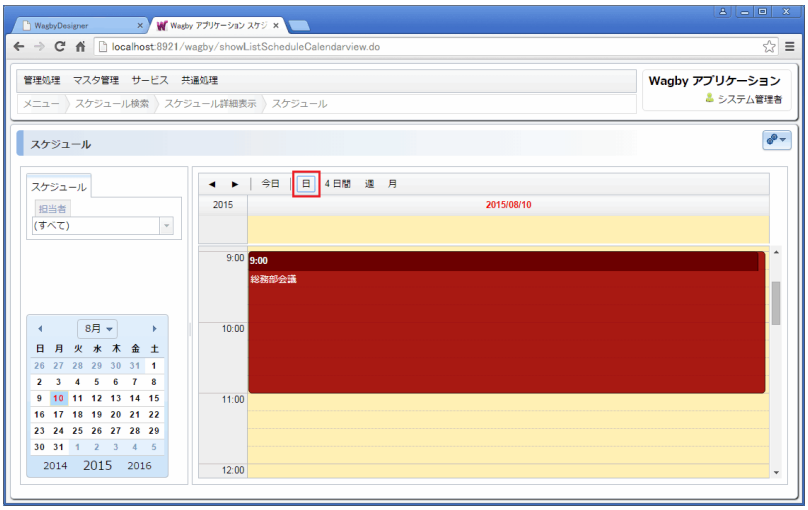


図 モダンカレンダビュー日表示

タイトル部をマウスで押下すると、詳細情報を示すダイアログが開きます。このダイアログから「詳細画面への遷移」「更新画面への遷移」「削除」を行うことができます。

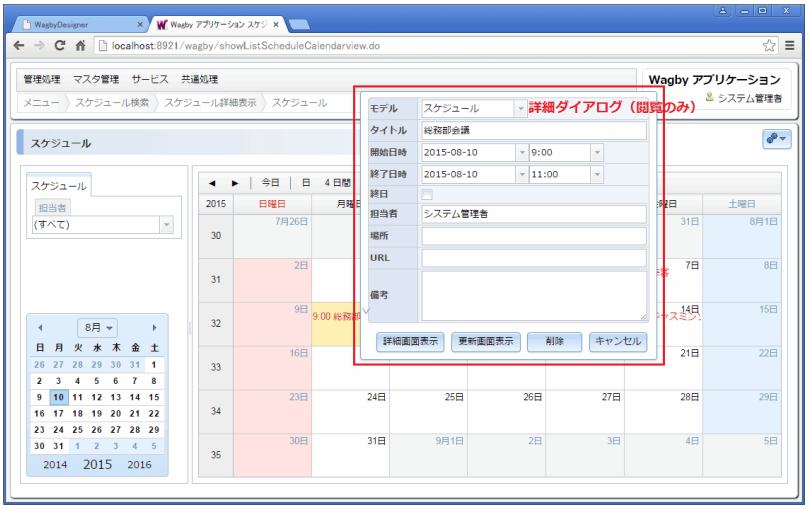


図 詳細ダイアログ

休日の扱い

同梱している休日情報

Thunderbird が提供する日本の公休日情報 "JapanHolidays.ics" を同梱しています。 これは iCalendar 形式で記述されています。

別の ics ファイルを指定する（国際化対応など）

他国の休日情報を適用することもできます。 詳細は"メッセージの変更・国際化 > 休日ファイルの変更"をお読みください。

独自の休日を設定する

会社の創立記念日など、独自の休日情報を設定することもできます。 詳細は"カレンダビュー（クラシック）> 休日の扱い"をお読みください。

モダンカレンダビューの文字サイズを固定する

Wagbyの文字フォントサイズを「大きいフォントサイズ」にすると、モダンカレンダビューの日付とカレンダデータが重なって表示される場合があります。これを防ぐため、モダンカレンダビューに表示される文字（タイトル部をマウスで押下した時に表示されるダイアログを含む）のサイズを固定するために、CSS のカスタマイズを行うことができます。

詳細は "レイアウト > CSSのカスタマイズ > モダンカレンダビューの文字サイズを固定する" をお読みください。

仕様・制約

- カレンダビューの「開始日付時刻」「終了日付時刻」に指定されている項目は、「値をデータベースに保存する」を指定する必要があります。
- カレンダビューの「開始日付時刻」「終了日付時刻」に指定されている項目が自動計算によって導出される場合、ドラッグ&ドロップによる移動操作は無効になります。
- 開始日時と終了日時の時刻がいずれも0時0分0秒の場合、このデータはカレンダビューに表示されません。また、開始日時と終了日時が逆になっているような場合も表示されません。
- カレンダビューを「月」形式で表示する場合、同じ時刻の場合の表示順は次の通りです。
  - R8.3.1までは表示順の規定がない。（並びは不定であった）
  - R8.3.2からは項目のソート順指定に準じる。

クラシックカレンダビューからモダンカレンダビューへの移行

クラシックカレンダビューで登録済みのデータを、モダンカレンダビューへ変換することができます。

具体的な方法は"サポート > インポートとエクスポート機能活用ガイド(R8) > クラシックカレンダビューのデータをモダンカレンダビューへ取り込む"をお読みください。

定義方法

モデルの定義

スケジュールモデルの定義を示します。「開始日付時刻」と「終了日付時刻」項目はそれぞれ「日付・時刻型」とします。

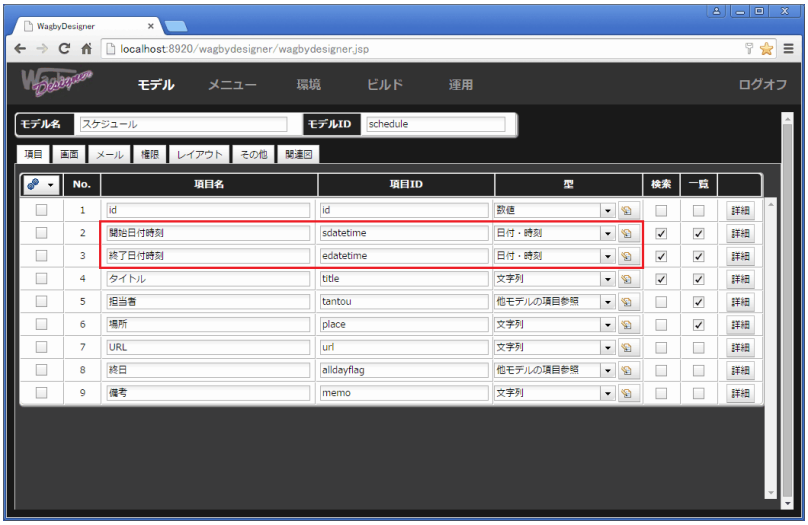


図 スケジュールモデルの定義

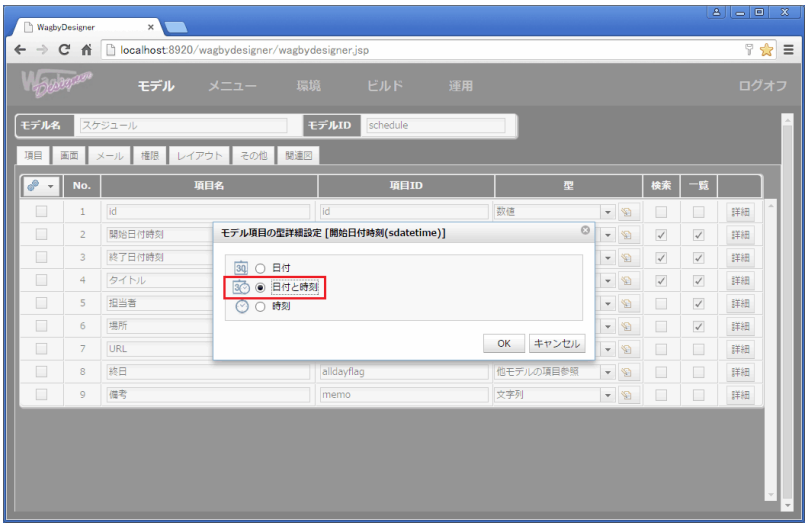


図 日付・時刻型の詳細で、日付時刻を選択する

カレンダビューの設定

「画面>ビュー>カレンダビュー」を開きます。「画面を作成する」をチェックし、ページタイトルを設定します。

「表示方式」を"モダン"にします。

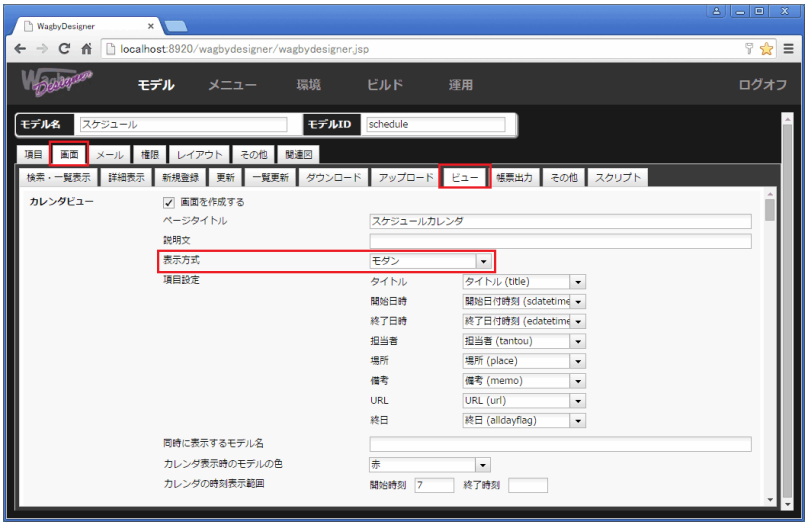


図 モダンカレンダビューの設定

項目設定欄に、モデルの項目を割り当てます。次のルールがあります。

設定値	対応する項目の型	説明	必須
開始日時	日付時刻	この日付情報を元にカレンダビューに表示を行います。	○
終了日時	日付時刻	この日付情報を元にカレンダビューに表示を行います。開始日時と終了日時は複数日にまたがることもできます。（5月1日から5月5日まで、など）	○
タイトル	文字列、整数、モデル参照	カレンダビューに表示する項目として定義します。	○
担当者	文字列、整数、モデル参照	カレンダビューに表示する項目として定義します。	－
場所			－
備考			－
URL			－
終日	カレンダ終日フラグモデル参照	この値をセットすると、終日イベント扱いになります。[後述]	－

開始日付時刻と終了日付時刻は必ず設定してください。（開始と終了の両方の設定が必要です。）

メニューの設定

「スケジュールカレンダビュー」ボタンをメニューに用意します。ここでは背景色も変えています。

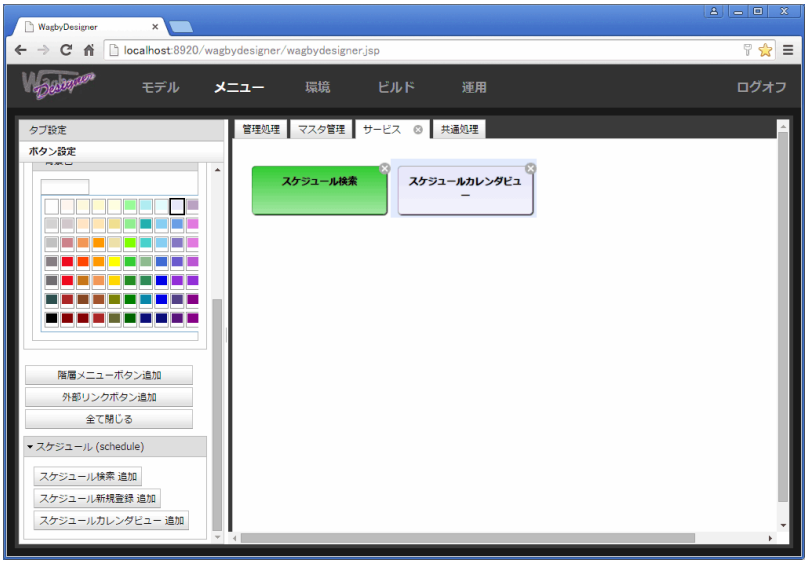


図 メニューの設定

カレンダ表示時のモデルの色

モデルの色を指定することができます。「橙色」に変更した例を示します。

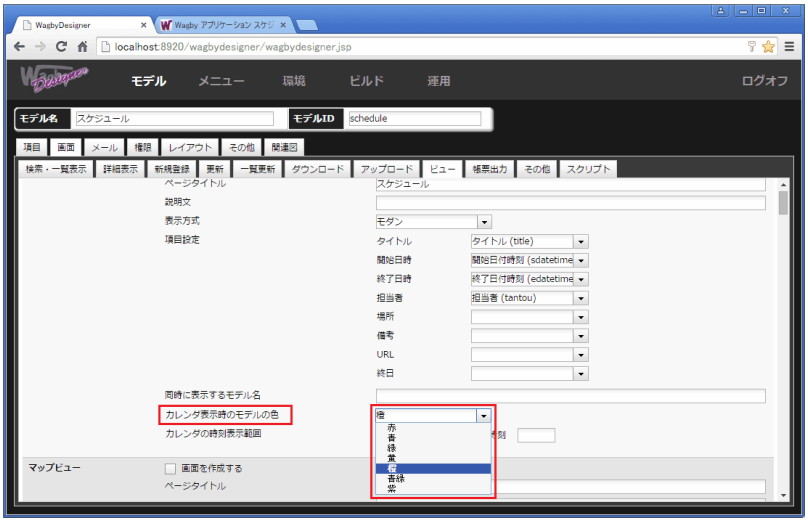


図 カレンダ表示時のモデルの色を変更する

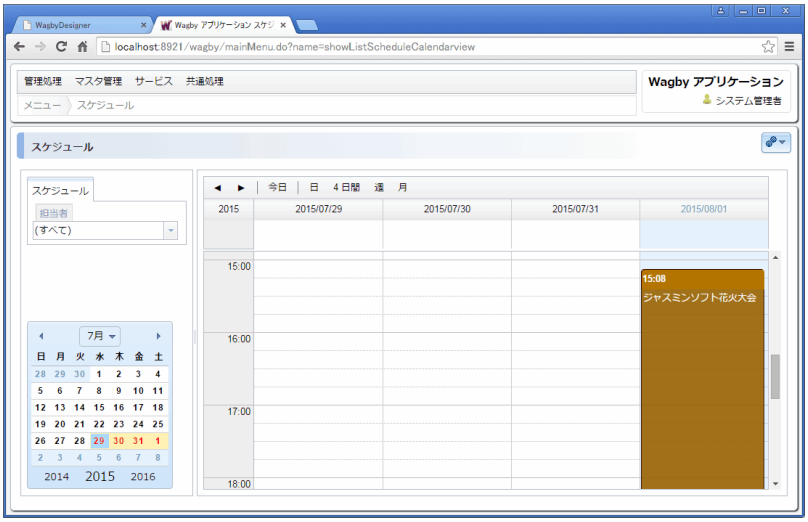


図 色が変更された

複数のモデルを同時に表示する

定義方法

「問い合わせ（ask)」モデルを用意します。カレンダビューを使うようにします。

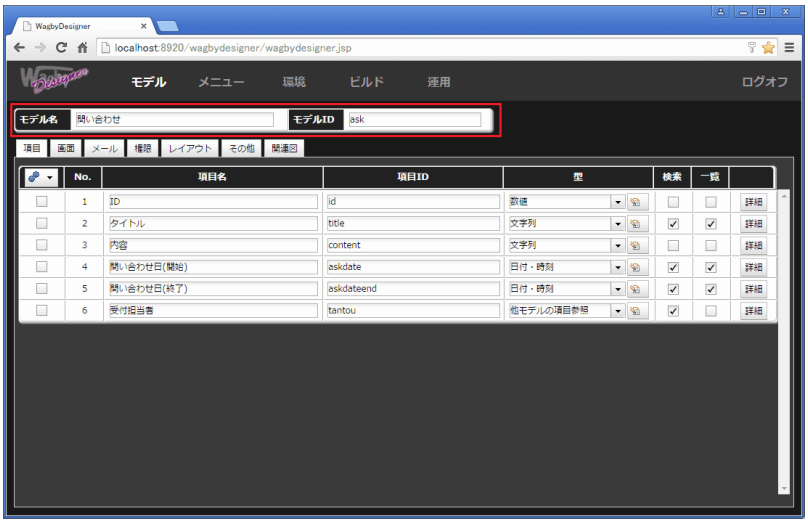


図 問い合わせモデル

先に用意した「スケジュール」モデルのカレンダビューで「同時に表示するモデル」に、問い合わせモデルを指定します。（モデルID ″ask″ を記載します。）

図 問合せモデルを同時に表示する設定

複数のモデルを指定することもできます。2つ以上指定する場合は、「,（カンマ）」区切りでモデルIDを記述します。 例：modelA,modelB

例

スケジュールカレンダビューでは、問い合わせモデルも同時に表示されています。ここではモデル毎に表示色を設定しています。

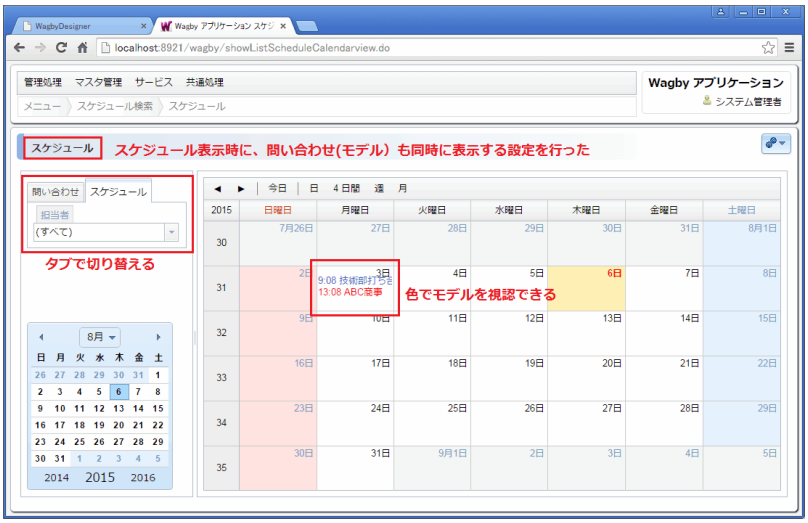


図 複数モデルを同一カレンダビューに表示する

問い合わせモデルでは同時に表示するモデルを指定していません。この場合、イベントモデルのカレンダビューは自分自身（イベントモデル）のみが表示されます。

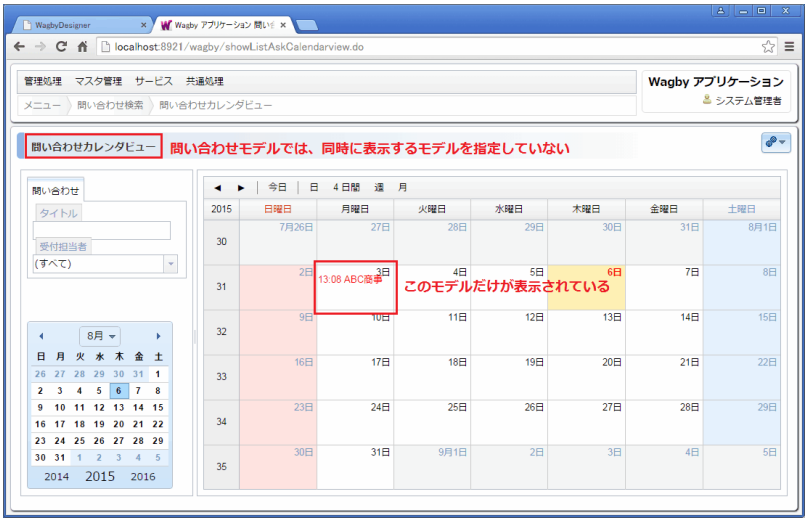


図 単一モデルのみの表示に切り替わる

カレンダの時刻表示範囲

定義方法

カレンダの時刻表示範囲（開始時刻、終了時刻）を指定できます。

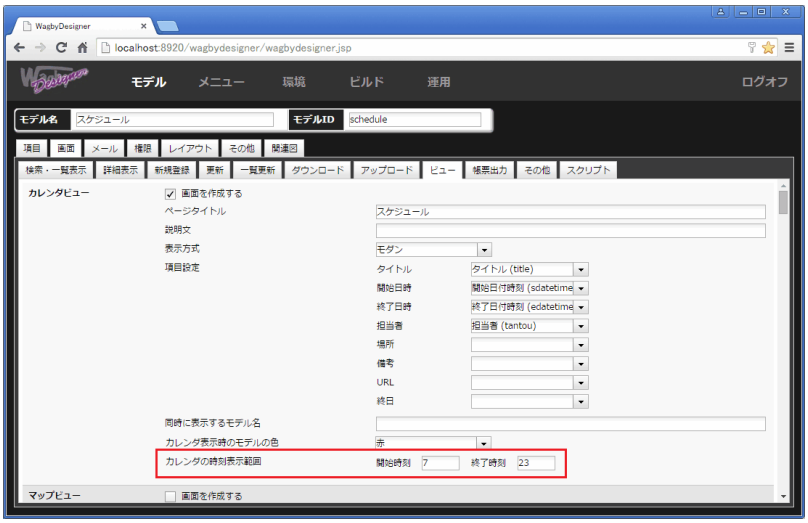


図 時刻表示範囲の設定

例

「日」「4日間」「週」形式において、指定した開始時刻から終了時刻までの範囲で表示されるようになります。

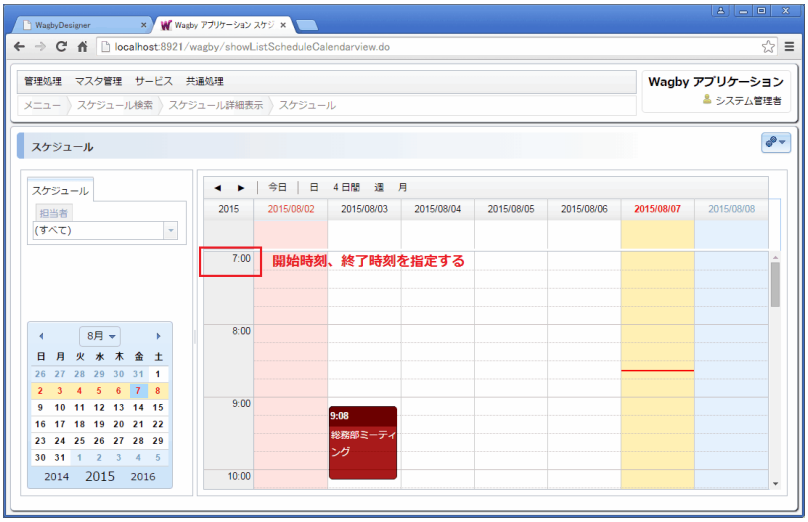


図 時刻表示範囲を指定したカレンダビュー

ドラッグ&ドロップによる変更

カレンダ上で、スケジュールをドラッグ&ドロップ操作することができます。日付・時刻を変更できます。

図 ドラッグ&ドロップによる操作

終日イベントとする

例

そのイベントを「終日」とすることができます。月単位の場合、表示方法が変わります。

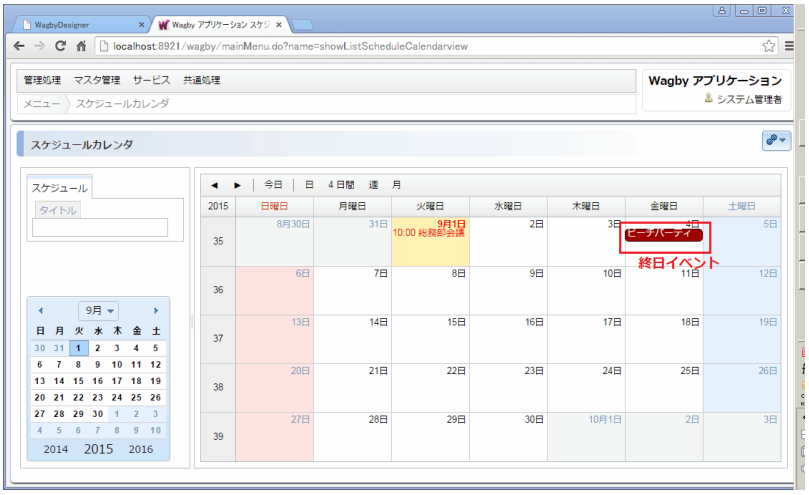


図 終日イベントの表示（月単位）

週単位の場合、開始時間軸の上部に表示されます。

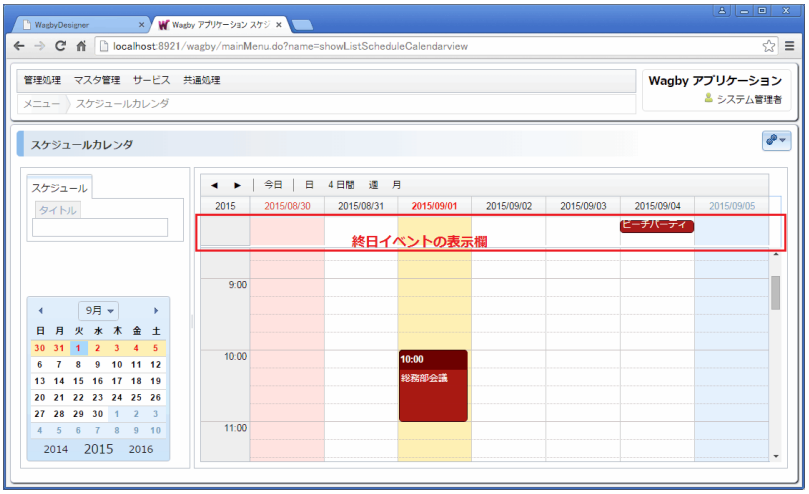


図 終日イベントの表示（週単位）

定義方法

終日フラグを格納する項目を用意します。

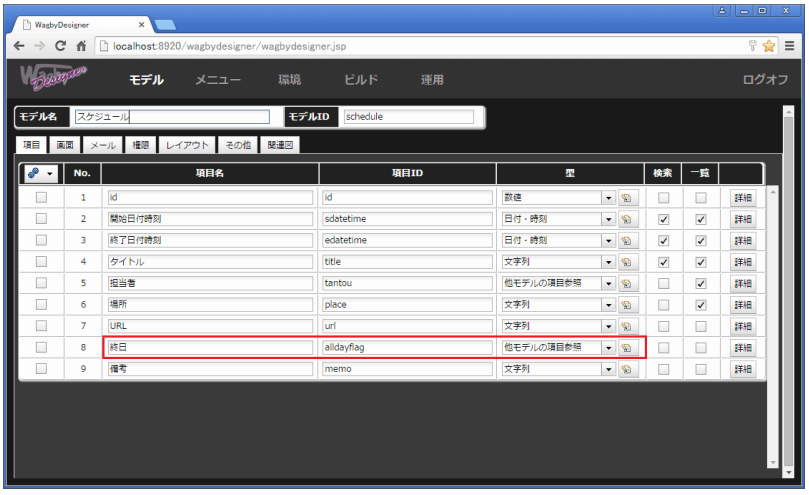


図 終日フラグを格納する項目を用意する

システムが提供する「カレンダ終日フラグ」を参照します。入力タイプは「リストボックス」「ラジオボタン」「チェックボックス」のいずれかとしてください。

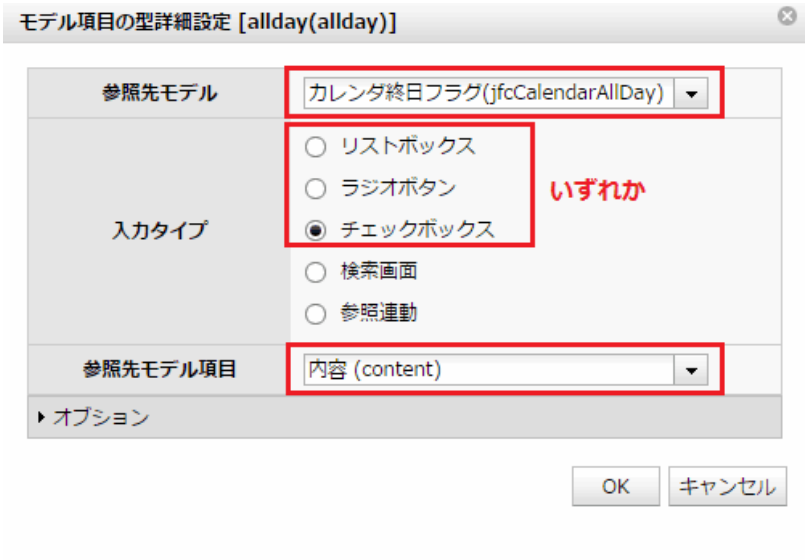


図 終日フラグのモデル参照設定

データの登録

対象イベントを登録する際、開始日付時刻は0時0分を起点とし、終了日付時刻は翌日の0時0分としてください。あわせて、終日フラグをセットしてください。

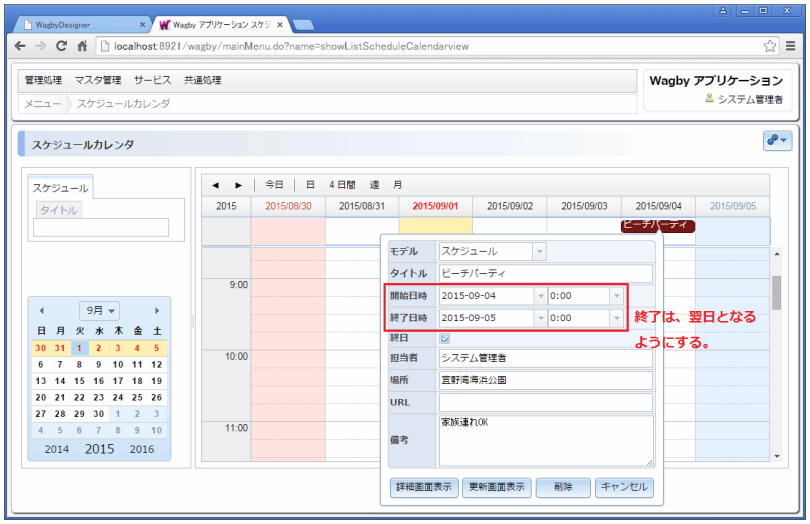


図 終日イベントの開始と終了の設定

ポータル画面に表示する

″ポータル > カレンダポートレット（モダン）″をお読みください。

カレンダビューから新規登録画面へ遷移する

新規登録画面へ遷移する

カレンダの日付をダブルクリックすると、新規登録用のダイアログが開きます。

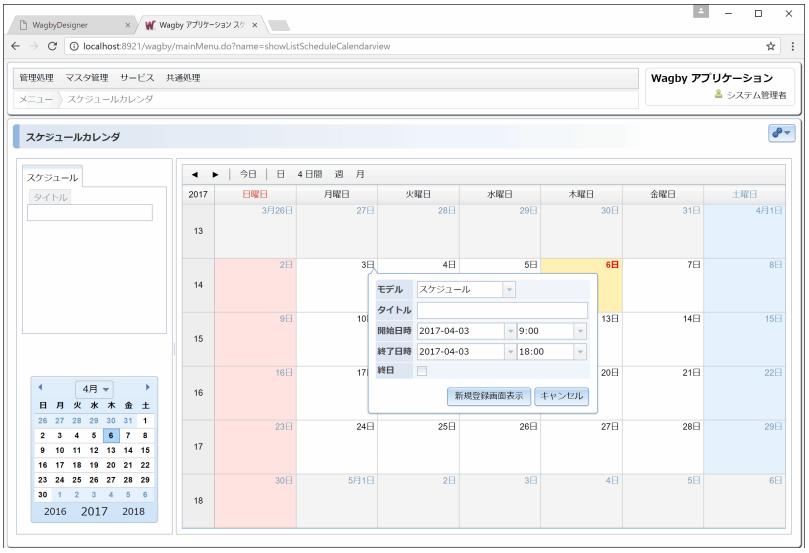


図 日付をダブルクリックすると表示されるダイアログ

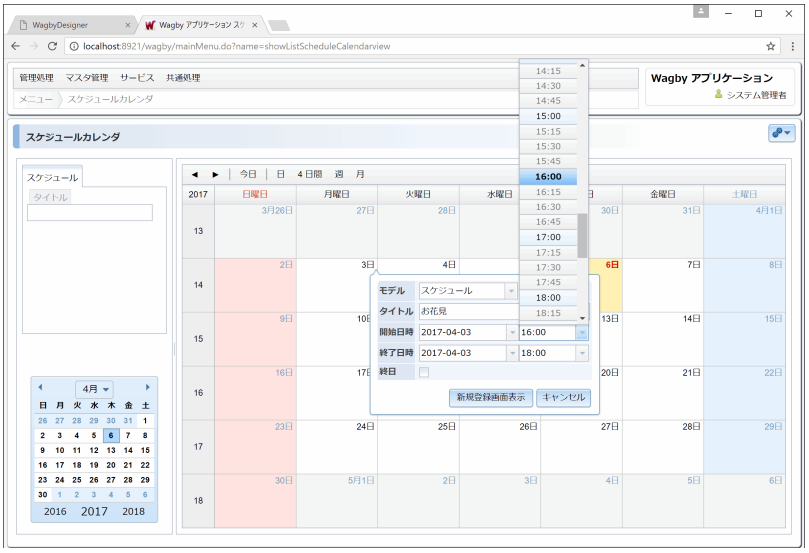


図 モデルを選択し、タイトルと開始・終了日時を指定できる

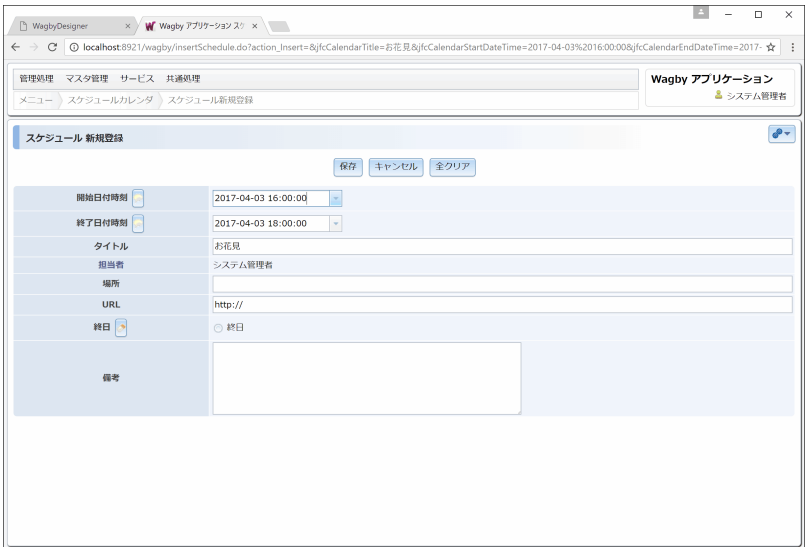


図 新規登録画面が開く

定義方法

「初期値 ＞ 登録画面表示時」のタイミングで、ダイアログから渡されるパラメータを受け取るための式を指定します。対象は「開始日時」「終了日時」「タイトル」に相当する項目です。

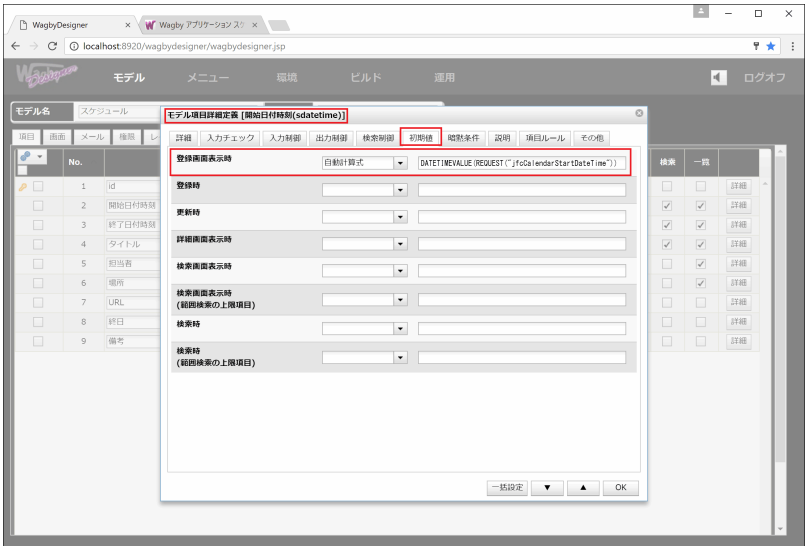


図 「初期値>登録画面表示時」に式を指定する

式は次のとおりです。

項目	式
開始日時	DATETIMEVALUE(REQUEST("jfcCalendarStartDateTime"))
終了日時	DATETIMEVALUE(REQUEST("jfcCalendarEndDateTime"))
タイトル	REQUEST("jfcCalendarTitle")
終日フラグ(チェックボックスとした場合)	IF(NOT(ISBLANK(REQUEST("jfcCalendarAllDay"))), ARRAY_INT(TOINT(REQUEST("jfcCalendarAllDay"))), ARRAY_INT())

(218)

独自の検索初期値を設定する

モダンカレンダビューは内部で REST API を使っています。そのため、ISREST 関数を使って「通常の検索画面と、モダンカレンダビューで異なる検索条件の初期値をセットする」ような設定を行うことができます。

```
IF(ISREST(), モダンカレンダビューでの検索条件初期値, 通常の検索画面での検索条件初期値)
```

(219) (220)

例 モダンカレンダビューでは検索条件を指定しない

数値型項目で、モダンカレンダビューでは検索条件を指定せず、通常の検索画面では初期値 1 を指定する場合は次のようになります。

```
IF(ISREST(),null,1)
```

(218) 新規登録ダイアログで終日フラグにチェックを付けたとき、開始日時は0:00となります。終了日時は開始日時の翌日となります。

(219) この指定は、カレンダポートレット（モダン）にも反映されます。

(220) クラシックカレンダビューは REST API を使っていないので ISREST関数での判定は行えません。代わりにSCREENTYPE 関数を利用するとよいでしょう。

マップビュー（Google Maps）

ご利用の前に

Google Maps はいつでも無料で利用できるサービスではありません。必ず以下の内容をご確認ください。

- Google Maps APIの使用については、Google Map APIの利用規約(Google Maps API Terms of Use)に従って下さい。
- ジャスミンソフトはGoogle Maps APIそのものの機能、サービスについて保障しません。
- Google Maps API は定期にバージョンアップされる予定です。Google Maps APIのバージョンアップが行われた場合は、それ以降にリリースされるWagbyで対応します。
- Google Maps 機能に不具合が発生した場合、ジャスミンソフトは、その原因がジャスミンソフト実装部分かGoogle Maps API部分のどちらにあるかを判断します。原因がジャスミンソフト実装部分にある場合は、サポートの範囲内で対応します。原因がGoogle Maps APIの部分にある場合は、ジャスミンソフトの実装部分で吸収可能かどうか判断した上で、サポートの範囲内で対応します。
- 2018年6月より、APIキーが必須となりました。詳細は同社の「パートナーディレクトリ」で"Regions"に"Japan"を選択して表示されるパートナー企業様へお問い合わせください。

APIキーの設定

Google Maps APIを利用する場合、APIキーが必要になります。

1. API キーの入手

Google Maps APIの入手は、「Google Maps API ＞ ウェブ向け」から行ってください。

Wagbyでは "Maps JavaScript API" と "Geocoding API" の両方を利用します。これらを有効にしてください。

2. API キーの登録

「環境 ＞ アプリケーション」タブに用意された「Google Maps」欄に API キーを設定します。

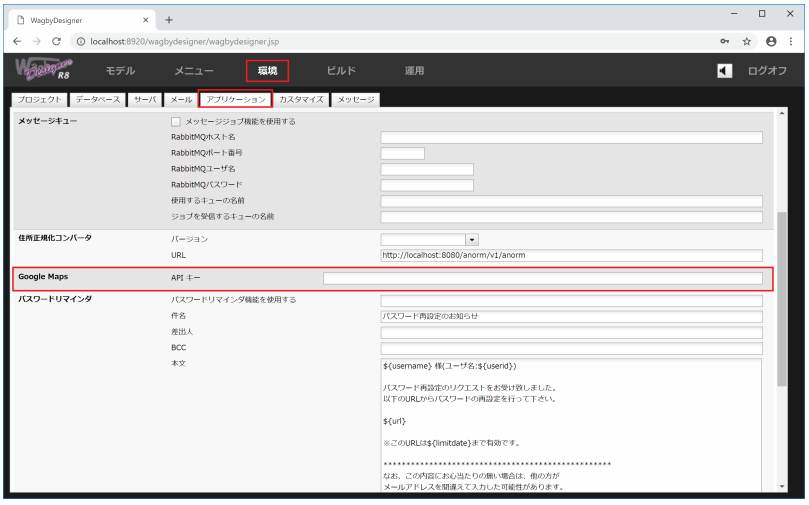


図 APIキーの設定

(221)

(221) R8.1.1より前の Wagby をご利用の場合、ライセンス種別を指定する欄があります。こちらは「標準」を選んでください。(2018年6月より実施された、GoogleMaps利用ルール改訂のため。)

例

マップビュー機能を有効にした「店舗」モデルを用意した例を示します。1つのデータに、1組の緯度、経度情報を保持させることができます。

図 詳細画面では地図上にポイントが表示される

一覧表示画面では、データの数だけポイントが地図上に表示されます。

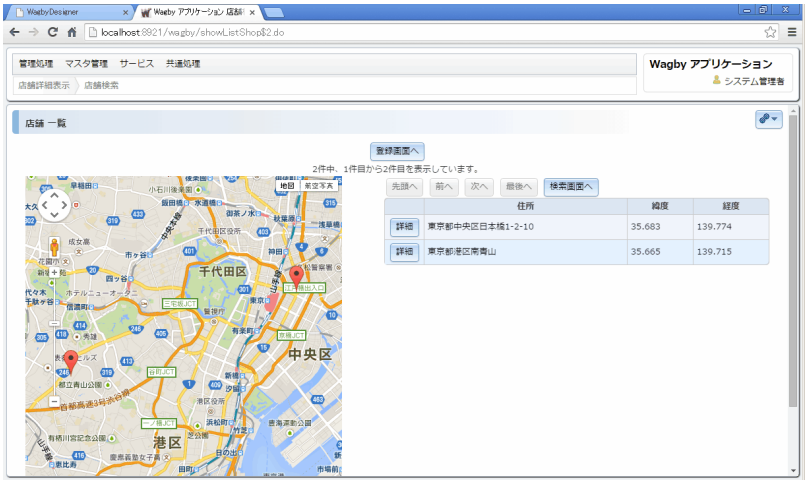


図 一覧画面では地図上に複数のポイントが表示される

定義方法

モデルの定義

店舗モデルの定義を図に示します。ここに用意した「住所」「緯度」「経度」はマップビュー利用時に必須となる項目です。

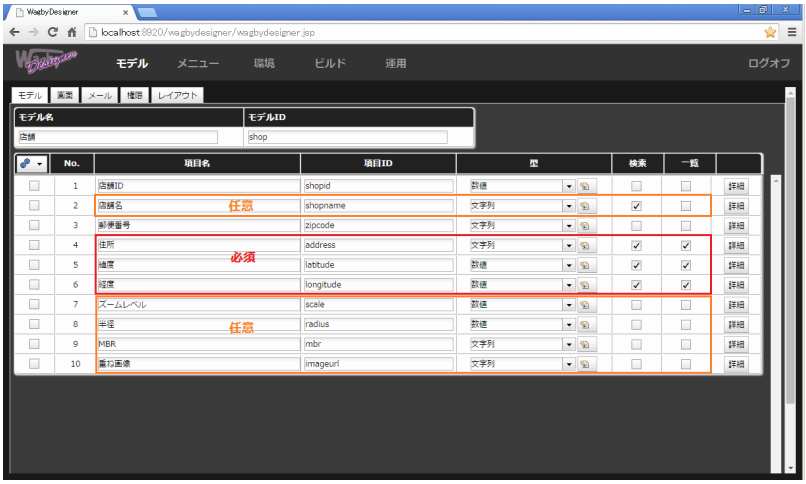


図 店舗モデルの定義

画面定義

「画面 > ビュー > マップビュー」を有効にします。

図 マップビューの定義

項目設定欄に、モデルの項目を割り当てます。次のルールがあります。[222]

設定値	対応する項目の型	説明	必須
名称	文字列	一覧表示画面においてマップビューに表示されたマーカにカーソルを当てると、名称で指定した項目のデータが吹き出しで表示されます。（「名称」の指定がない場合は「住所」に指定した項目のデータが表示されます。）	-
住所	文字列	「座標取得」ボタン押下時に Google Maps API を利用して	○

[222] 以下の表で「必須」とは、必須入力チェックを行うということではありません。必ず用意してほしい項目という意味です。

		住所から緯度経度を取得します。	
緯度	8バイト浮動小数点	この座標情報を元にマップビューにマーカが表示されます。[223]	○
経度	8バイト浮動小数点	この座標情報を元にマップビューにマーカが表示されます。	○
ズームレベル	整数	マップの拡大縮小レベルです。0～20の範囲で指定します。	-
半径	8バイト浮動小数点	マップビューに描画する円の半径です。単位は m です。	-
MBR	文字列	マップビューに配置する画像の座標情報です。	-
重ね画像	URL	マップビューに配置する画像のファイルパス(URL)です。	-

その他の設定は次のとおりです。

ページタイトル

ページタイトルを設定します。指定がない場合は、「モデル名_マップビュー」となります。

説明文

マップビュー用に説明文を用意することができます。 画面の説明文とは別にマップビューの表示位置に合わせて、地図の下、または右に表示されます。

マップの種類

「Google Maps」を指定します。

マップの表示位置

マップの表示位置を指定します。「左」または「下」が選択できます。標準では「左」が指定されています。

マップの表示比率

マップの表示位置が「左」の場合に有効です。 空白時は「50%」となります。

マップのサイズ

マップの縦横をピクセルで指定します。単位は省略できます。空白時は縦：360px、横：360pxとなります。

マップビュー定義のルール

マップビューでは、次のようなルールがあります。

検索項目の指定

[223] アップロード更新機能を用いて住所データ等を登録する場合、緯度と経度の項目も埋めておく必要があります。緯度経度情報が未定の場合は空白ではなく「0(ゼロ)」を設定してください。

「緯度」「経度」「住所」項目は、検索条件を有効にしてください。 「緯度」「経度」は範囲検索を有効にしてください。

一覧表示項目の指定

「緯度」「経度」「住所」項目は、一覧表示画面に加えてください。なお、「緯度」「経度」を隠し項目にすることもできます。

読み込み専用設定を行わない

「緯度」「経度」「住所」項目は読み込み専用の指定を行わないようにしてください。

一覧表示画面の振る舞い

「検索条件入力画面と検索結果一覧表示を同一画面にて取り扱う」を指定することはできません。

ウィザードとの併用

マップビューとウィザードを併用する場合、マップビュー欄の設定を行った項目は、ウィザード欄に同じページ番号を指定してください。

重ね画像を指定する場合

「重ね画像」を指定する場合はMBRの指定が必須となります。

MBRを指定する場合

テキストエリアをMBRに指定することはできません。

一覧表示画面

一覧表示画面では、一覧表示されたデータの緯度経度情報を基にマップビュー上にマーカが表示されます。 マーカにカーソルを合わせると「名称」に指定した項目のデータが吹き出しで表示されます （「名称」の指定がない場合は、「住所」に指定した項目のデータが表示されます。）

マップビュー上のマーカをクリックするとそのデータの詳細表示画面に遷移します。

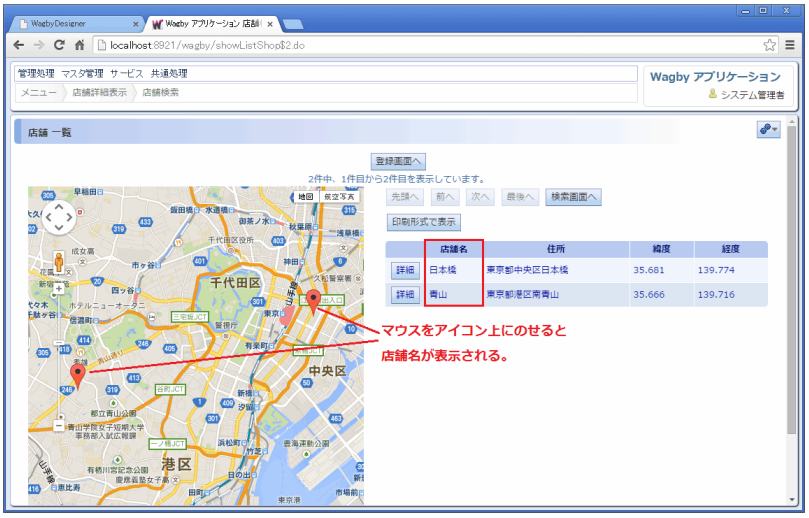


図 一覧表示画面

定義方法

マップビューの「名称」に指定した項目は、一覧表示画面でも表示するとします。

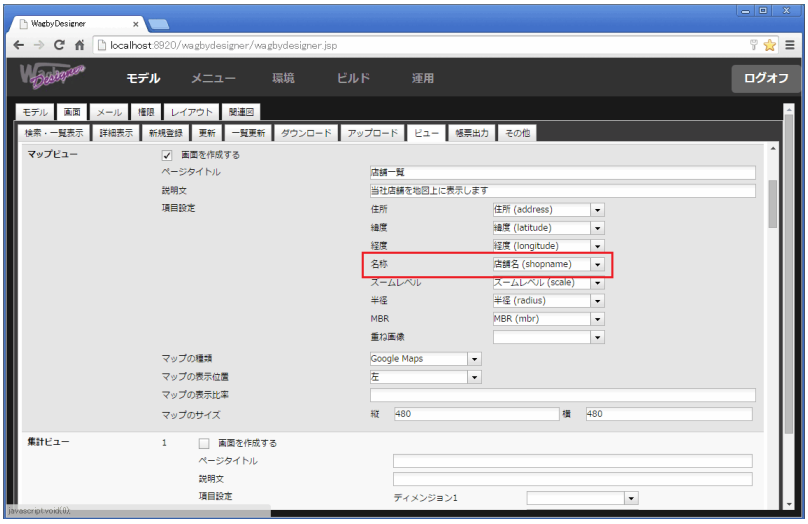


図 マップビューの名称項目を指定する

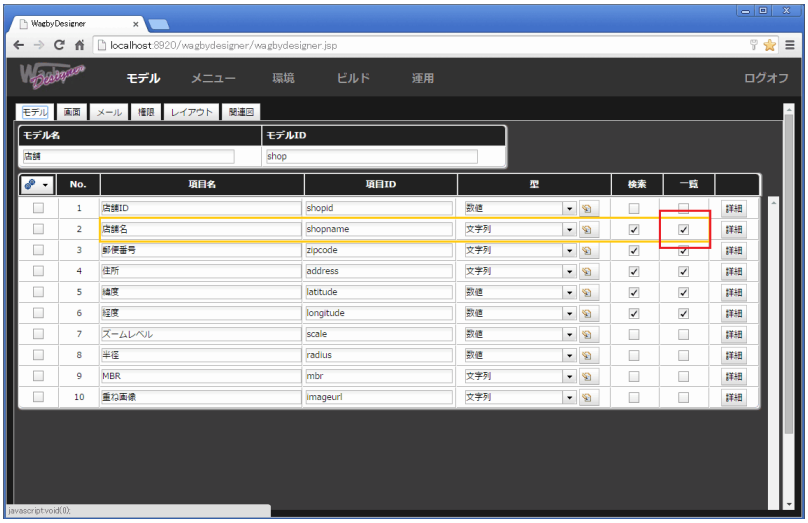


図 マップビューの名称項目を一覧表示画面でも用いる

緯度と経度が登録されていないデータは、地図上にマーカは表示されません。

検索画面を開いたときの地図表示の範囲を指定する

標準では地図表示の範囲は"全世界"です。緯度経度項目の初期値「検索画面表示時」と「検索画面表示時（範囲検索の上限項目）」を設定すると、範囲を限定することができます。ズームレベルもあわせて設定することができます。

半径、MBR、重ね画像

半径

マップビューの「R」ボタンを押下すると、円描画・表示モードになります。　もう一つのマーカをドラッグ&ドロップすることで、円の半径を変更できます。

円の半径は、項目「半径」にセットされます。項目「半径」は手入力可能です。単位はメートルです。

詳細画面で半径を視認する場合は、同じく「R」ボタンを押下してください。

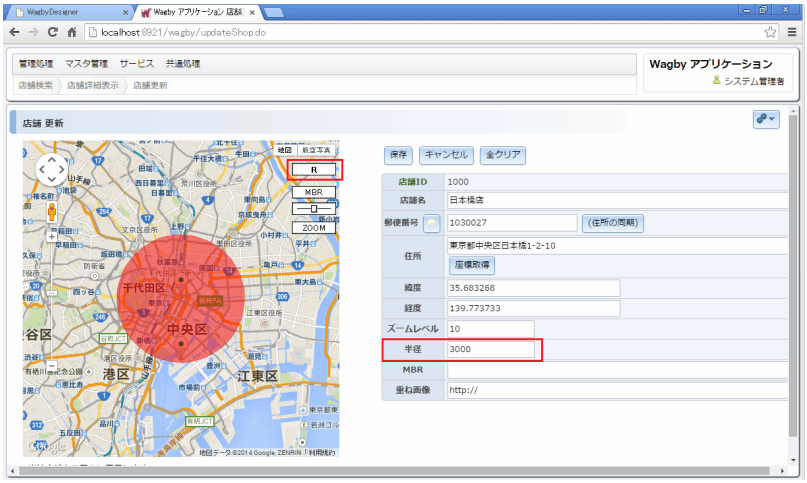


図 半径の入力

MBR（minimum bounding rectangle; 最小外接矩形）

マップビューの「MBR」ボタンを押下すると、MBR描画・表示モードになります。　もう一つのマーカをドラッグ&ドロップすることで、MBR矩形を変更できます。

MBRの値は緯度経度で、"(x1,y1)-(x2,y2)"　という書式になります。手入力も可能です。項目「MBR」にセットされます。

詳細画面でMBRを視認する場合は、同じく「MBR」ボタンを押下してください。

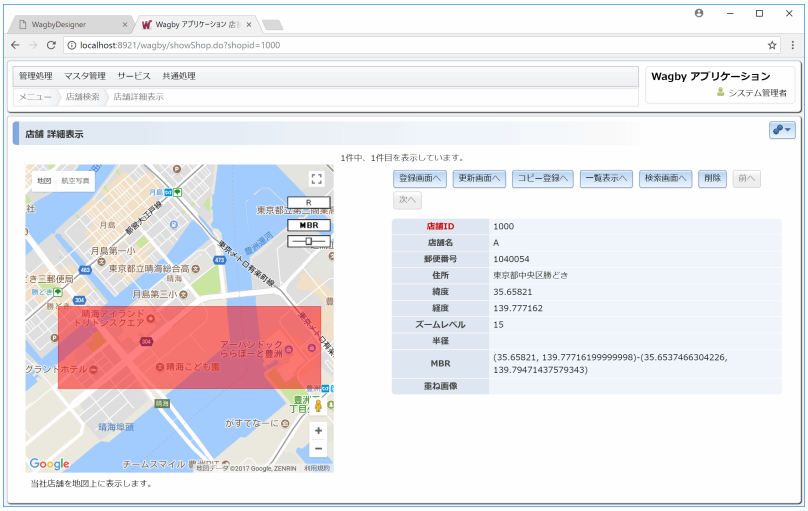


図 MBRの表示

重ね画像

MBR を利用しつつ、さらに項目「重ね画像」で画像ファイルを示す URL が指定されている場合は、矩形内にその画像を表示します。マーカに紐付けて任意の画像を表示させることができます。「MBR」ボタン下のスライダで画像の透過率を変更できます。

詳細画面で重ね画像を視認する場合は、同じく「MBR」ボタンを押下してください。

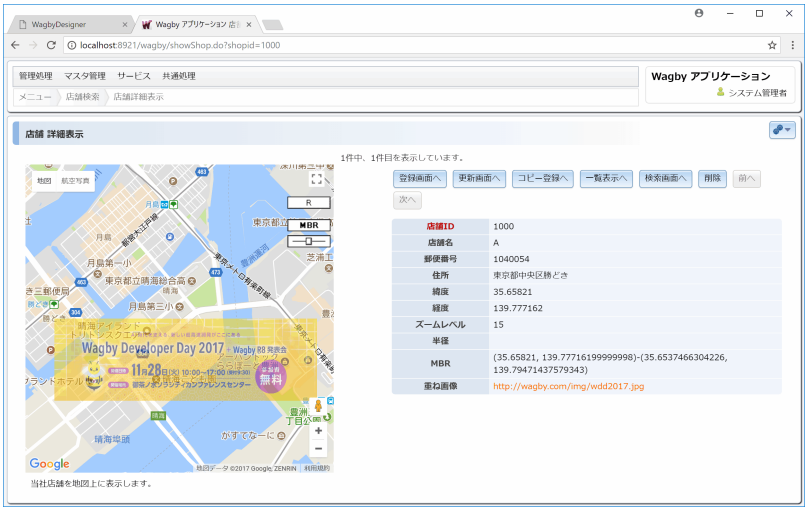


図 重ね画像の表示

マーカーの色と画像を指定する

色の指定

マーカーの色を指定することができます。

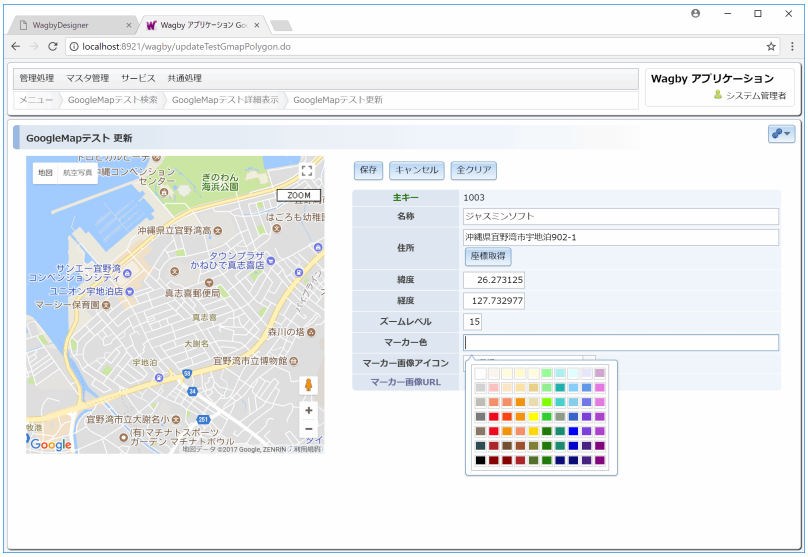


図 マーカーの色を指定する

色は RGB 形式の 16 進数として管理されます。（16進数のため先頭に "#" が付与されます。）

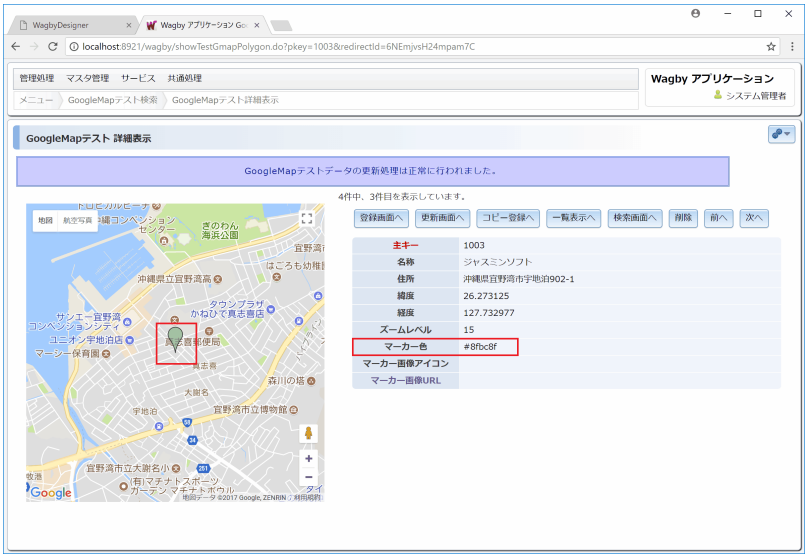


図 マーカーの色を変更した

16進数表記に代わって「black」「brown」「green」「purple」「yellow」「blue」「gray」「orange」「red」「white」という10種類の文字列を指定することもできます。

画像を利用する

画像はあらかじめ用意してください。ここでは「GoogleMapマーカー画像」モデルを作成し、4つのアイコンを事前に用意しています。

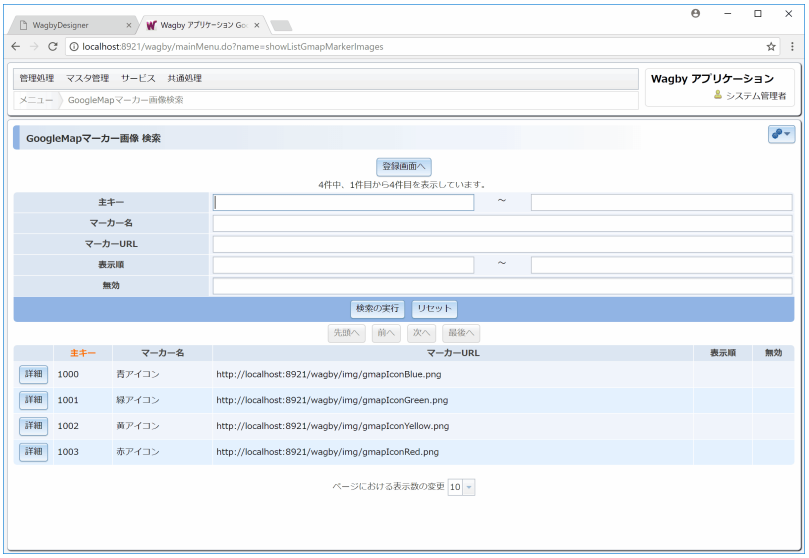


図 GoogleMapマーカー画像モデル

サンプルのアイコンは次のとおりです。

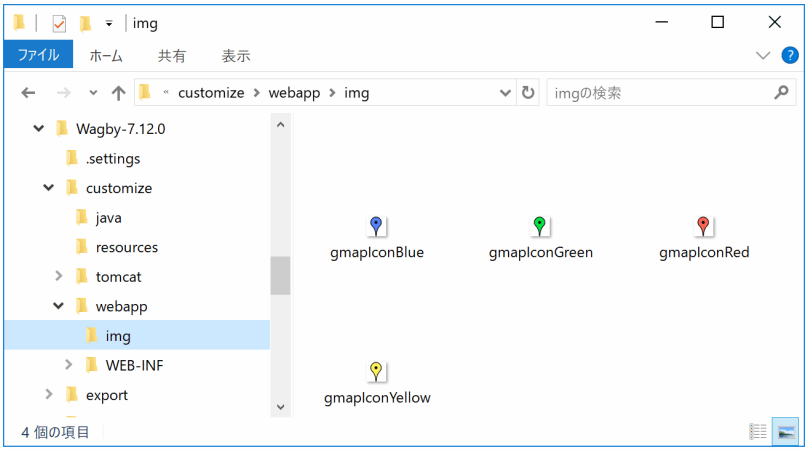


図 サンプルのアイコン

マーカー画像アイコンを指定します。

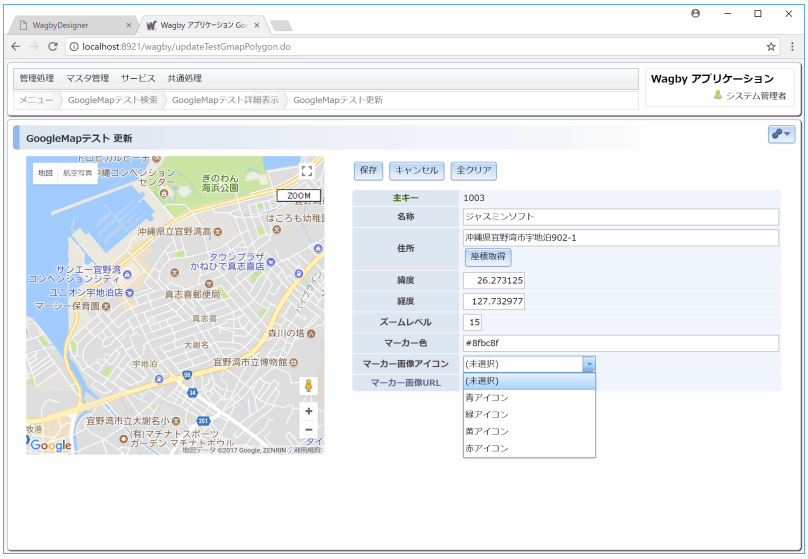


図 マーカー画像アイコンを指定する(1)

図 マーカー画像アイコンを指定する(2)

(224)

定義方法

はじめに、マーカーの画像を管理する「GoogleMapマーカー画像モデル」を用意します。選択肢モデルの定義に類似しています。式や初期値などの設定は行っていません。シンプルなモデルです。

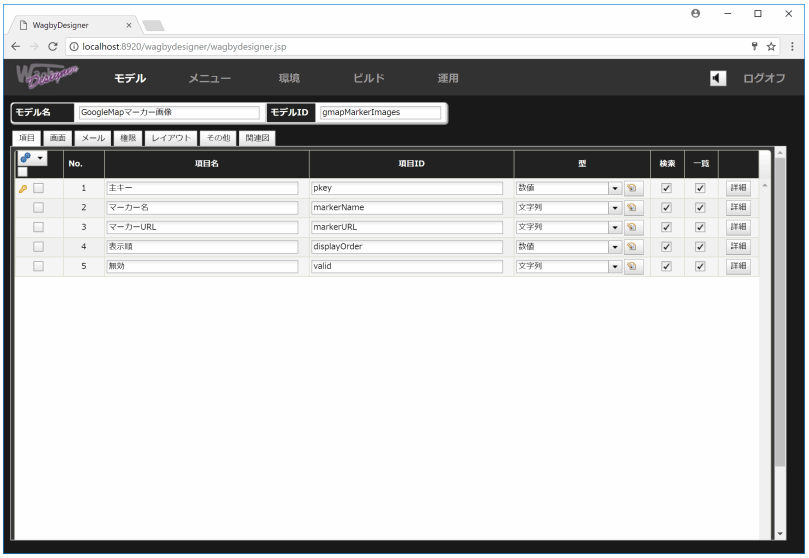


図 GoogleMapマーカー画像モデルの定義

(225)

GoogleMapを利用するモデルでは、「マーカー色」「マーカー画像アイコン」「マーカー画像URL」の3つの項目を追加します。

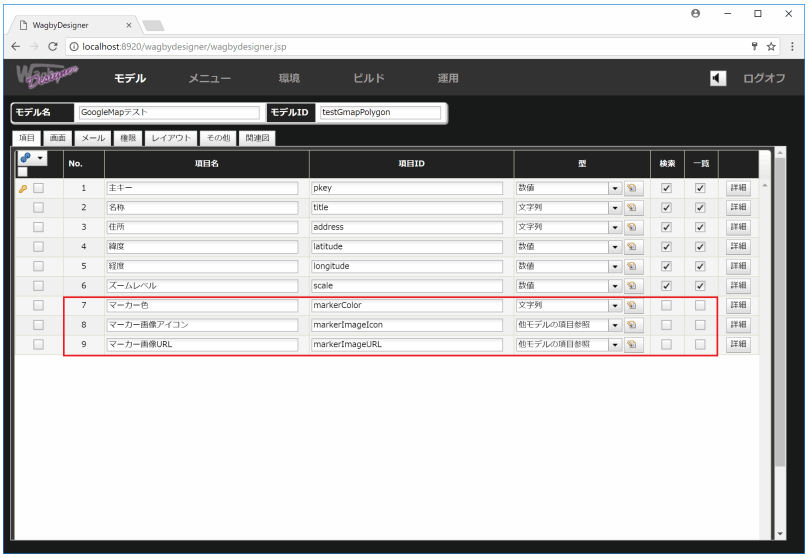


図 色と画像に関する項目を追加する

「マーカー画像アイコン」は、上で用意した「GoogleMapマーカー画像モデル」を参照します。

---

(224) マーカーの色と画像の両方を指定した場合は、画像を優先します。

(225) マーカーの色だけを設定させ、画像を指定しない場合は、このモデルの定義は不要です。

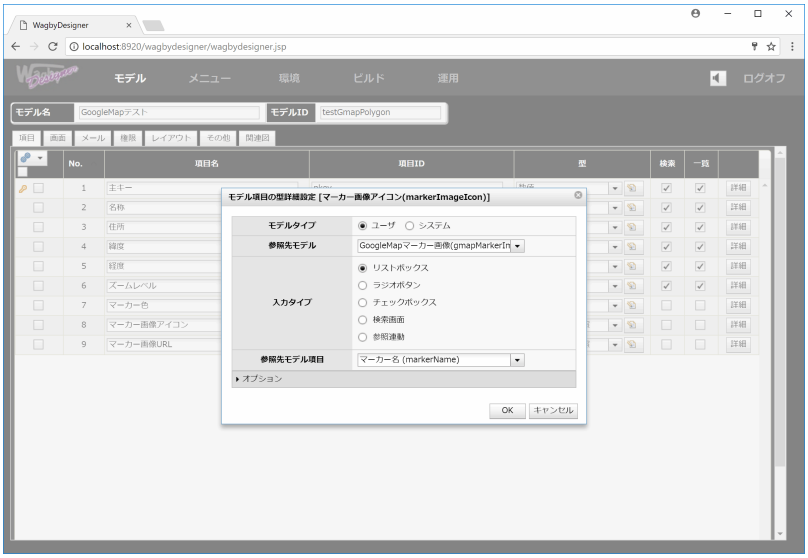


図 マーカー画像アイコン項目

「マーカー画像URL」は、参照連動項目とします。

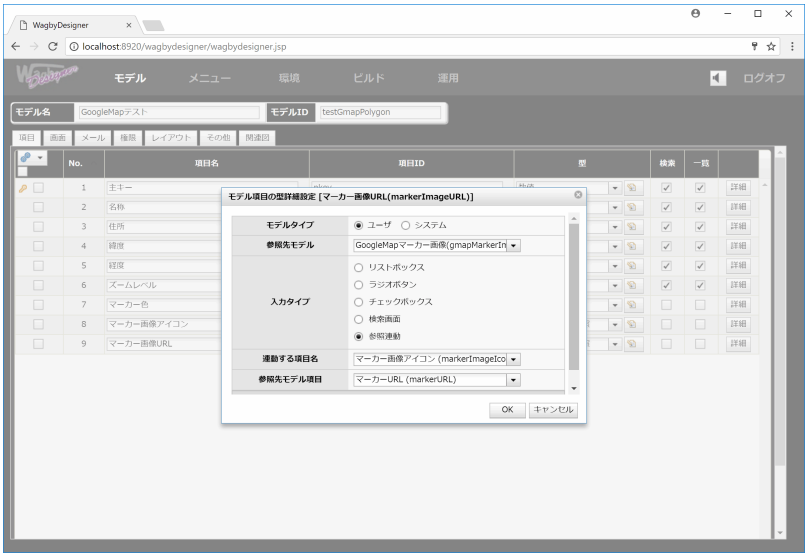


図 マーカー画像URL項目

マップビュー設定で、「マーカー色」「マーカーアイコンURL」に対応した項目をそれぞれ指定します。
「マーカーアイコンURL」は（参照連動項目である）「マーカー画像URL」を指定します。

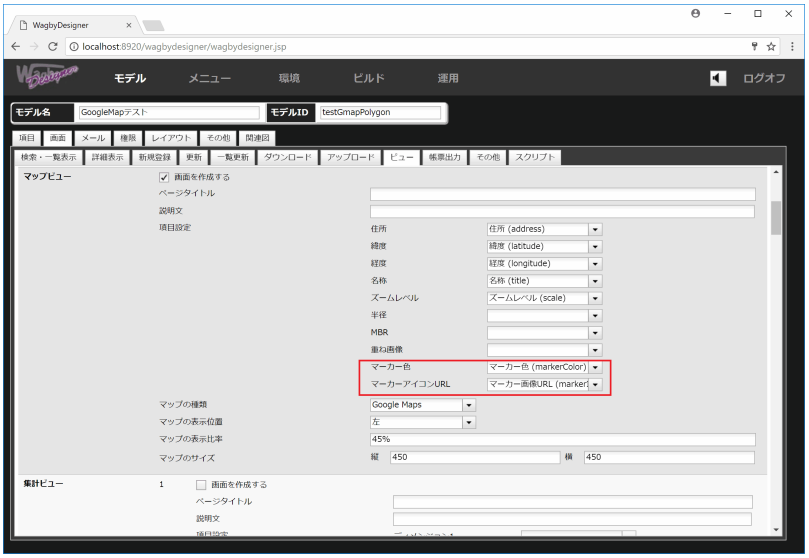


図 マップビューの設定

(226)

一覧表示画面への対応

一覧表示画面にもマーカー色と画像を表示させることができます。

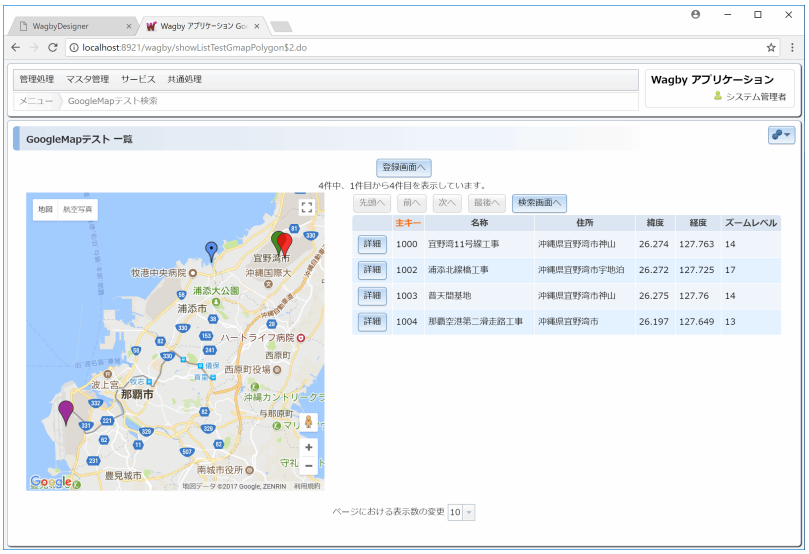


図 一覧表示画面への適用

「マーカー色」「マーカーアイコンURL」を一覧表示に適用するようにします。

(226) 「マーカーアイコンURL」を参照連動項目ではなく、文字列型として定義することもできます。その場合は画像のありかを URL で直接、入力するようにしてください。

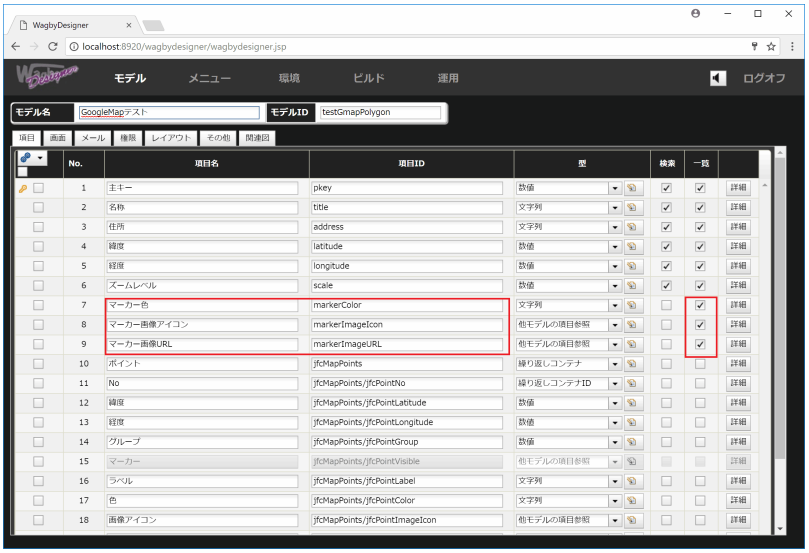


図 一覧表示項目に含める

さらに、これらの項目を「一覧表示画面の隠し項目」とすることができます。これによって一覧表示の内容を最小限に抑えることができます。

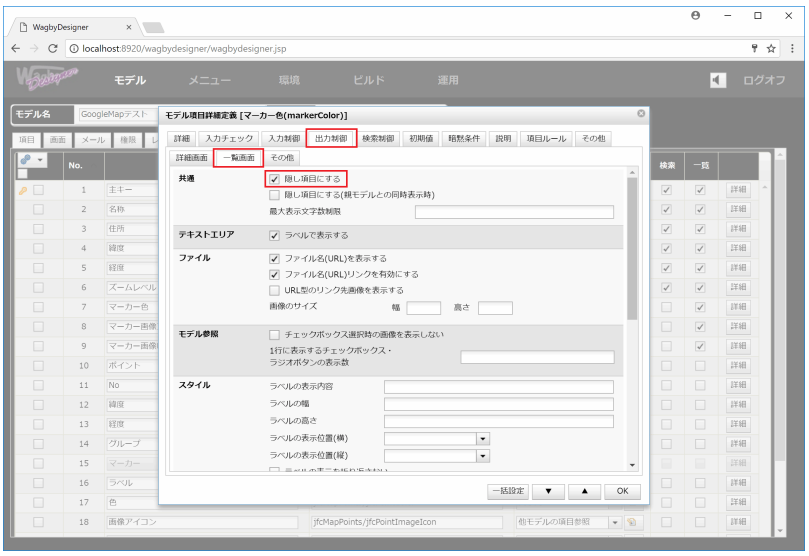


図 隠し項目とする

画像のダウンロード

ここで紹介した4つの画像をダウンロードできます。ダウンロードしたファイルを Wagby をインストールしたフォルダに展開してください。customize/webapp/img フォルダの中に4つの png ファイルがコピーされます。

- gmapIconSample.zip (6,553 bytes)

スタイルを指定する

Google Maps のスタイルを指定することができます。道路などの地物のスタイルを JSON オブジェクトの配列として定義できます。

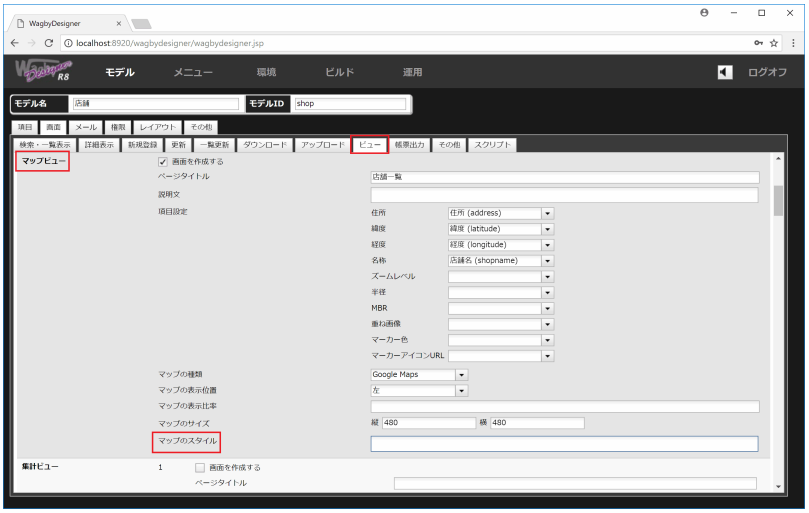


図 スタイルの設定(1)

全ての地物をを灰色とし、幹線道路の形状を青色、風景ラベルを非表示とするスタイルの例は次のとおりです。

```
[
  {
    "featureType": "all",
    "stylers": [
      { "color": "#COCOCO" }
    ]
  },{
    "featureType": "road.arterial",
    "elementType": "geometry",
    "stylers": [
      { "color": "#CCFFFF" }
    ]
  },{
    "featureType": "landscape",
    "elementType": "labels",
    "stylers": [
      { "visibility": "off" }
    ]
  }
]
```

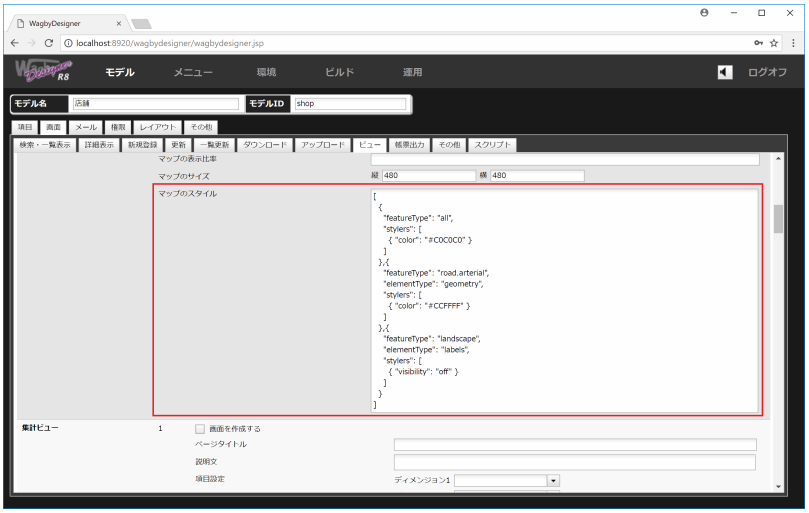


図 スタイルの設定(2)

このスタイル指定を組み込んだときの表示イメージは次のとおりです。

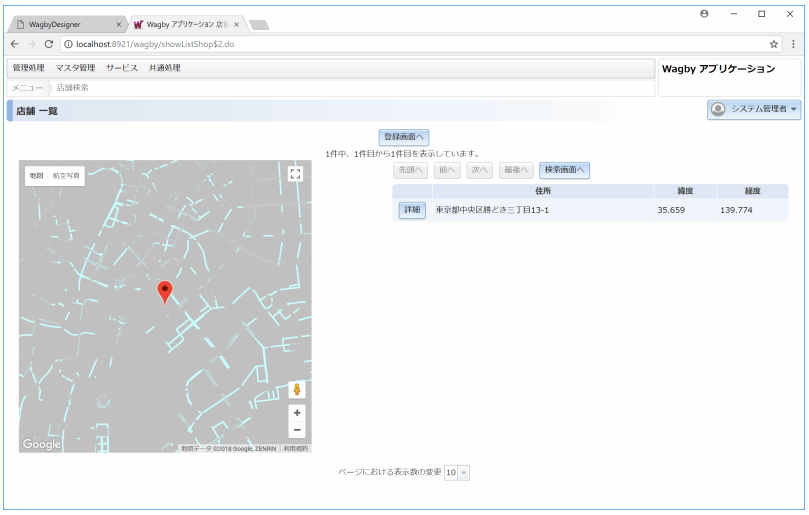


図 スタイルの設定(3)

参考資料

- マップのスタイル設定を開始する
- スタイル リファレンス

マップビューにポリゴンを描画する

例

二点間に線を引く

「ポイント」欄に緯度・経度を指定します。2つのポイントを設定すると、線を引くことができます。

ポイントは、このモデルの登録（または更新）画面で、マウスで地図上から直接、位置を指定するこで設定することができます。

図 二つのポイントを指定する

ポイントの属性

登録（または更新）画面で、各ポイントは次の属性を指定できます。

項目	説明	入力例
緯度	座標値です。	26.27149
経度	座標値です。	127.724522
グループ	複数の点の集合をグループ化できます。[後述]	1,2,3 などの整数値
マーカー	チェックしたポイントにマーカーが表示されます。	（チェック）
ラベル	マーカーに文字を埋め込みます。マーカー表示が有効で、ラベルが指定されていないときは 1,2,.. という数値が表示されます。	"A", "B"
色	マーカーの色を指定します。	red, #ff0000
画像アイコン	マーカーの画像を示すURLを指定します。	http:

マーカーの色と画像の指定については"マップビュー　>　マーカーの色と画像を指定する"をお読みください。

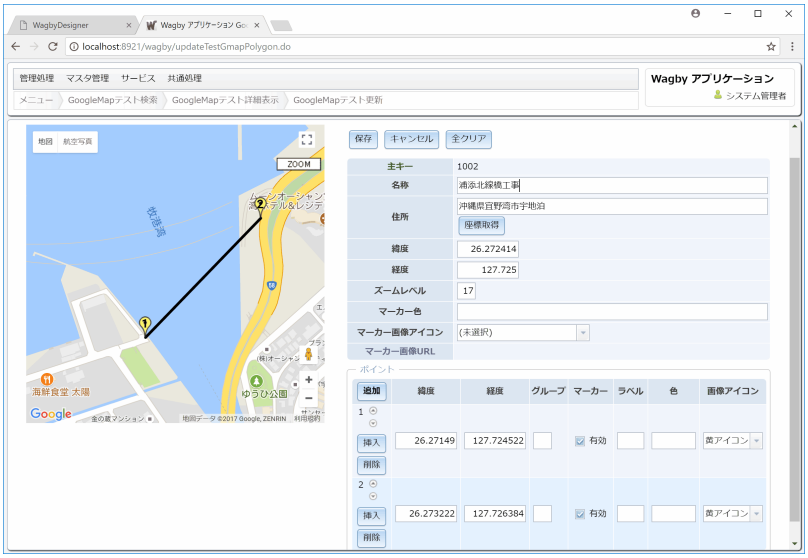


図 ポイントの属性を設定する

ポリゴンを描画する

二つ以上のポイントを設定すると、これを「一筆書き」で描画します。始点と終点を同一値とすることで、ポリゴンを描画することができます。

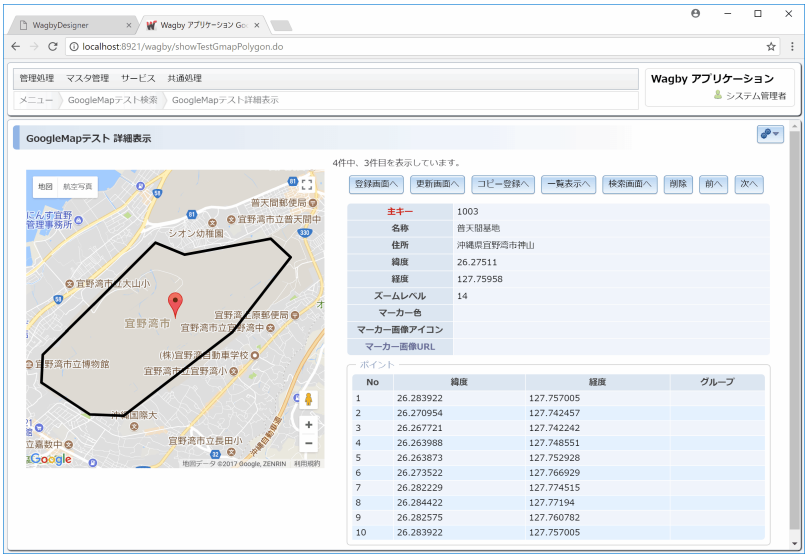


図 ポリゴンを描画する

グループを指定する

複数のポイント群を、グループとしてまとめることができます。グループ単位でライン（3点以上の場合はポリゴン）を描画します。

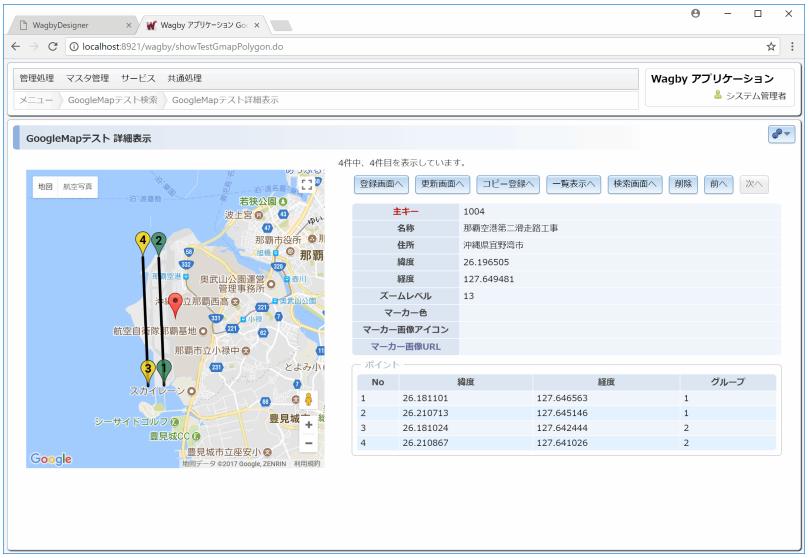


図 グループを指定する

定義方法

マップビューを利用するモデルに、「テンプレート ＞ GoogleMapポイントコンテナ」を加えます。

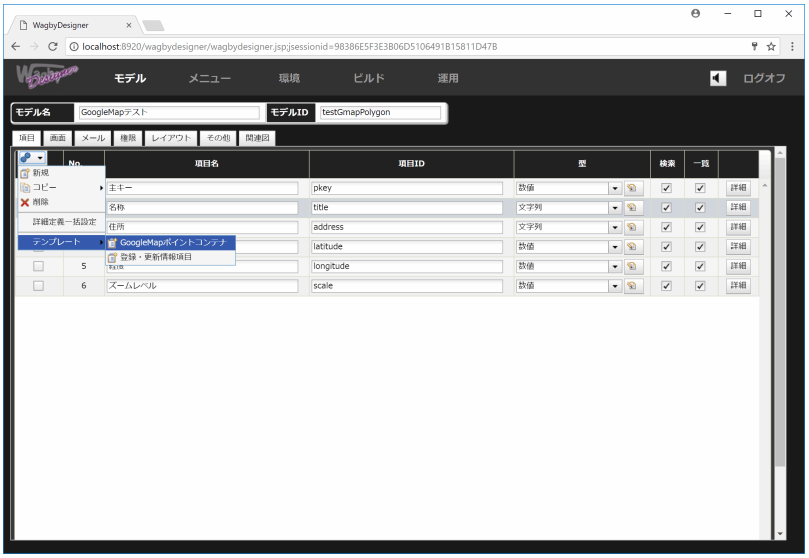


図 GoogleMapポイントコンテナを加える(1)

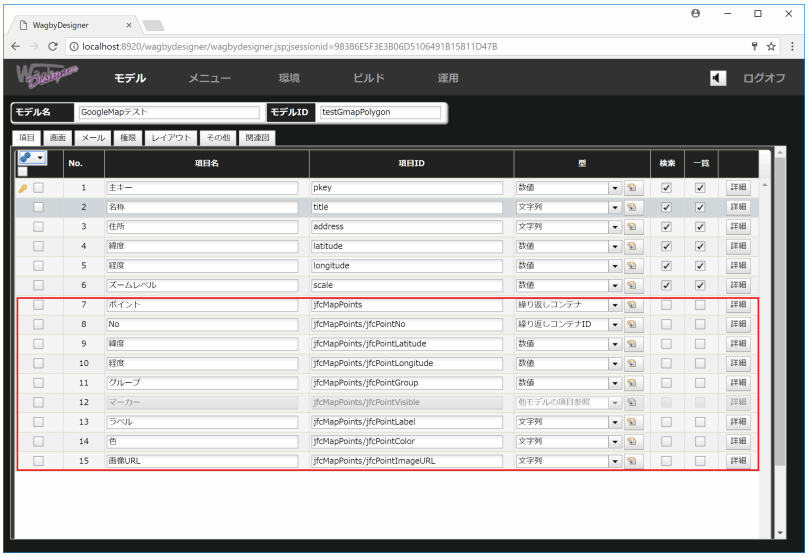


図 GoogleMapポイントコンテナを加える(2)

GoogleMapマーカ画像モデルを用意する

マーカー画像を保存するモデル「GoogleMapマーカ画像（gmapMarkerImages)」を用意します。詳細は"マーカーの色と画像を指定する"をお読みください。

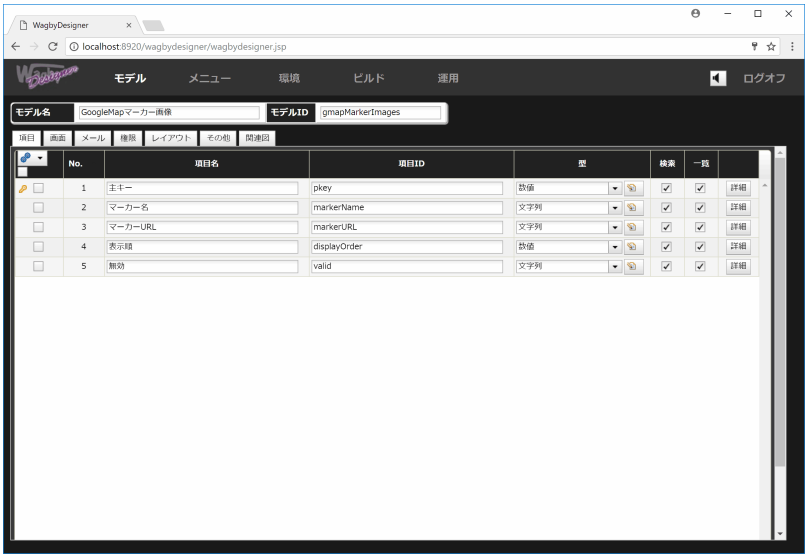


図 GoogleMapマーカ画像モデルを用意する

「テンプレート ＞ GoogleMapポイントコンテナ」で加わった項目の「画像アイコン」「画像URL」を文字列から、上の「GoogleMapマーカ画像」を参照するように型を変更します。

詳細は"マーカーの色と画像を指定する"をお読みください。

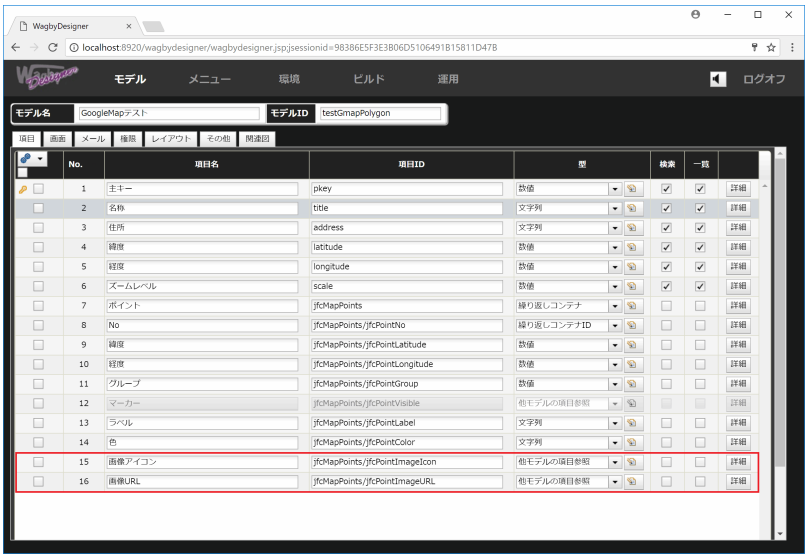


図 型を変更する(1)

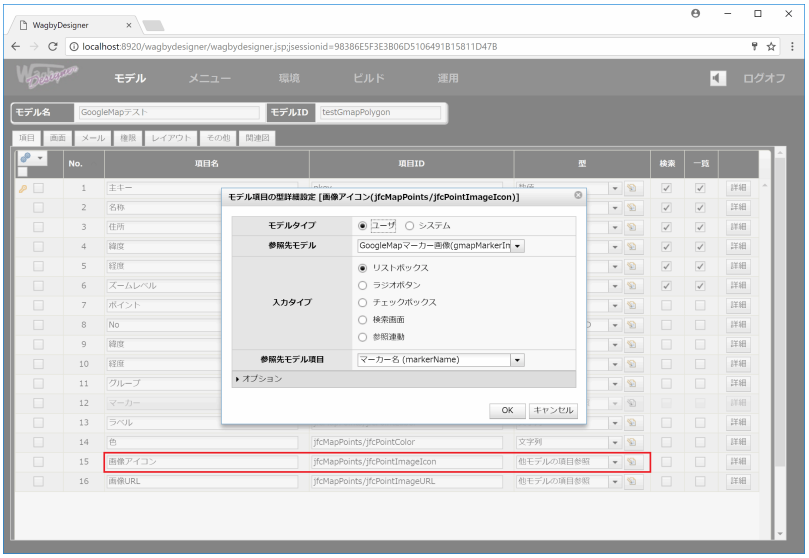


図 型を変更する(2)画像アイコン項目

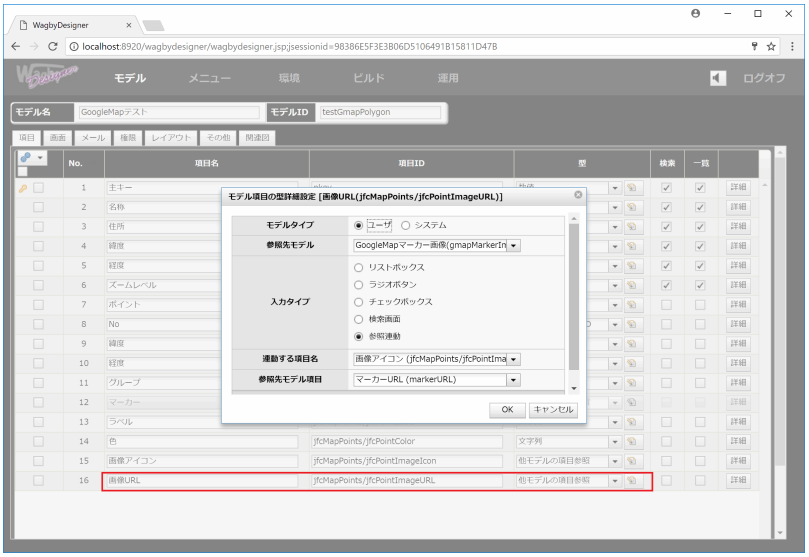


図 型を変更する(3)画像URL項目

ダウンロード

ここで説明したサンプルアプリケーションのリポジトリ、データをダウンロードできます。

- GMPolygonSample.zip (1,825,278 bytes)

メインモデルとサブモデル

概要

1つのモデルで複数の画面を作成する場合は、まずメインとなるモデル(以後「メインモデル」と呼びます)を定義した後に、その定義内容をコピーして新しいモデル(以後「サブモデル」と呼びます)を定義します。 １つのメインモデルに対して複数のサブモデルを用意することもできます。

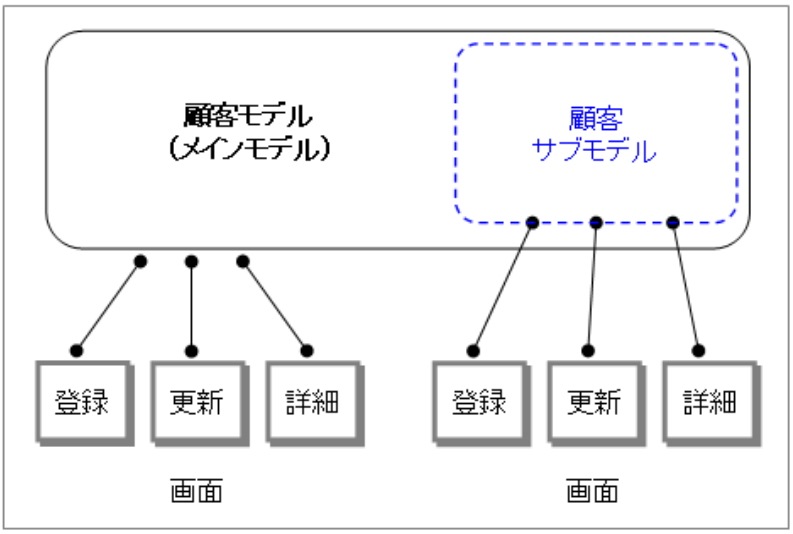


図 メインモデルとサブモデルの構成イメージ

- 画面を複数用意できますが、データをやり取りするモデルそのものは１つです。
- データベース上では、メインモデルとサブモデルは共通のテーブルが使用されます。
- サブモデルはメインモデルのサブセットの関係にあるので、既存項目(主キーを除く)を削除して減らすことはできますが、メインモデルにない新たな項目を追加定義することはできません。[詳細...]

サブモデルの作成方法

メインモデルを一つ選択します。

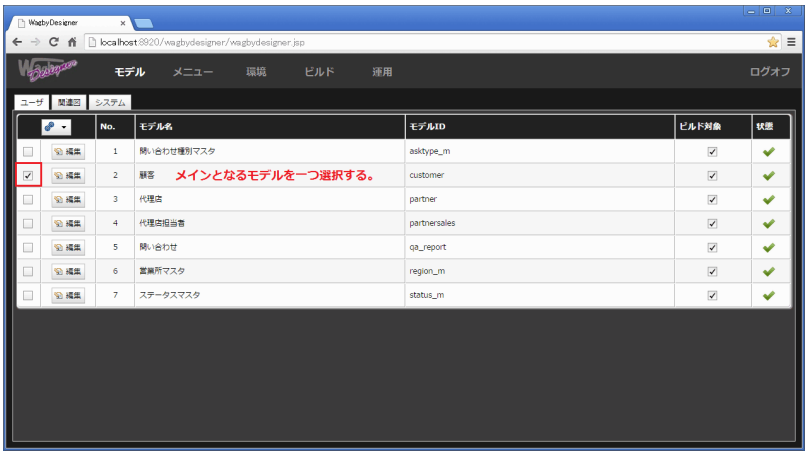


図 メインとなるモデルを一つ選択する

ギアアイコンから「テンプレート 〉 サブモデル」を選択します。

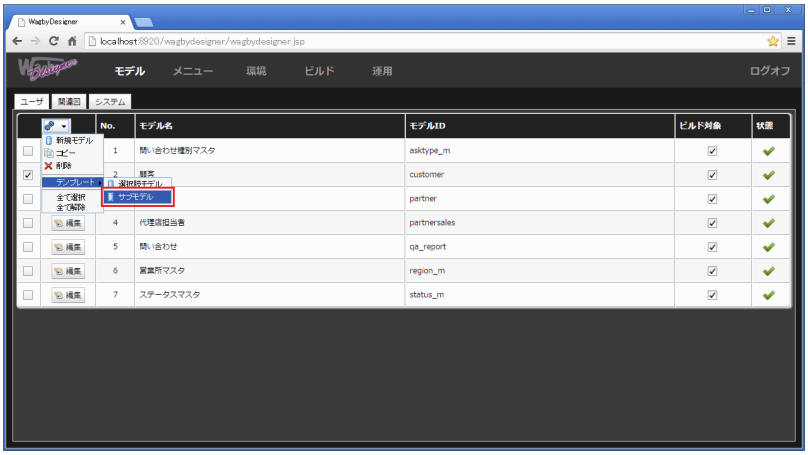


図 サブモデルを作成する

標準では、モデル名に「- サブモデル」と付与されたモデルが新規で作成されます。モデルIDは「1」という数字が付与されます。

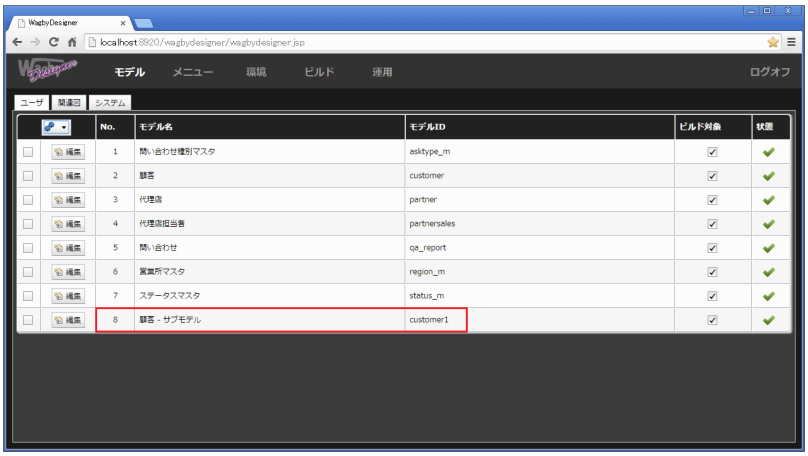


図 サブモデルが作成された

サブモデルの定義を開きます。最初にモデル名、モデルIDを適切に変更してください。

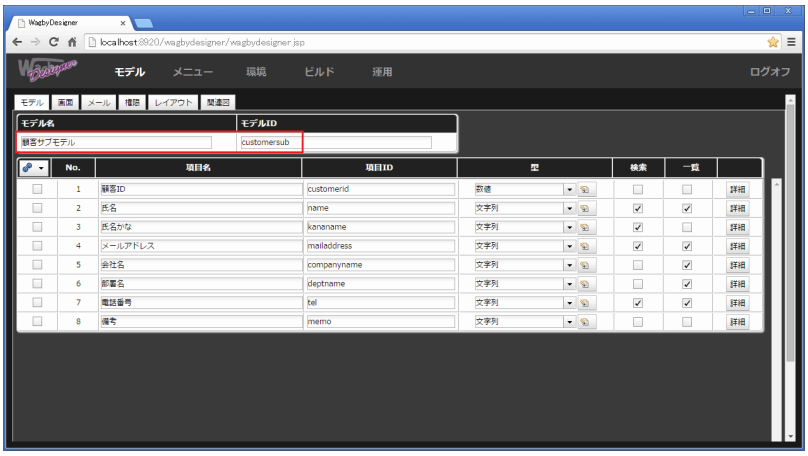


図 サブモデルのモデル名とモデルIDを変更する

ここでは項目名にすべて ″[サブ]″ という名称を加えてみました。（項目名は変更可能です。項目IDは変更できません。） また、備考という項目を削除してみました。（サブモデルでは項目を減らすことはできます。）

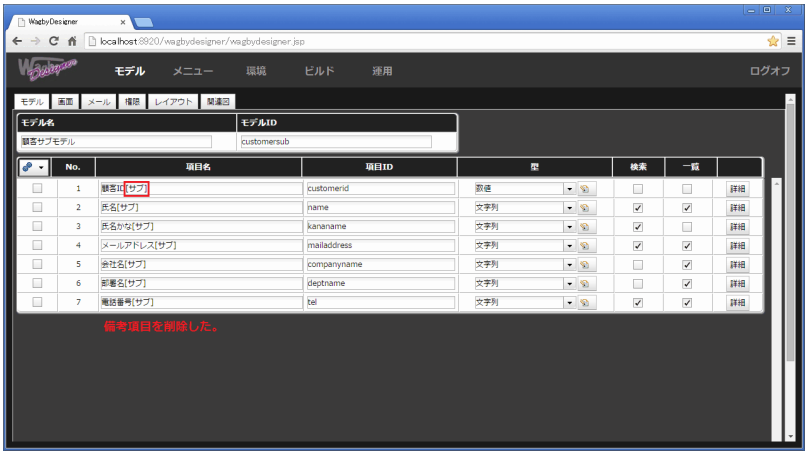


図 テストのためにサブモデルの定義を微修正する

例

はじめに、サブモデルからデータを登録してみます。

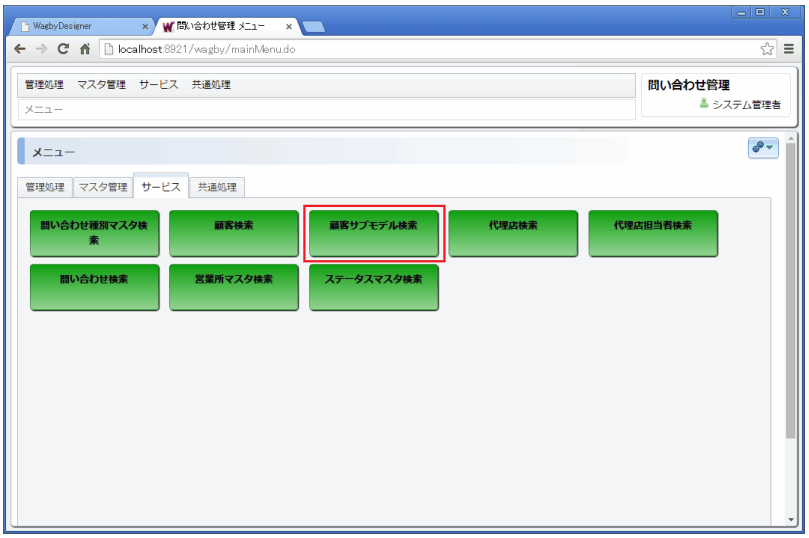


図 サブモデルから操作する

図は、サブモデルからデータを1件、登録した例です。

図 サブモデルのデータを1件、登録した

続いて、メインモデルの検索を行なってみます。

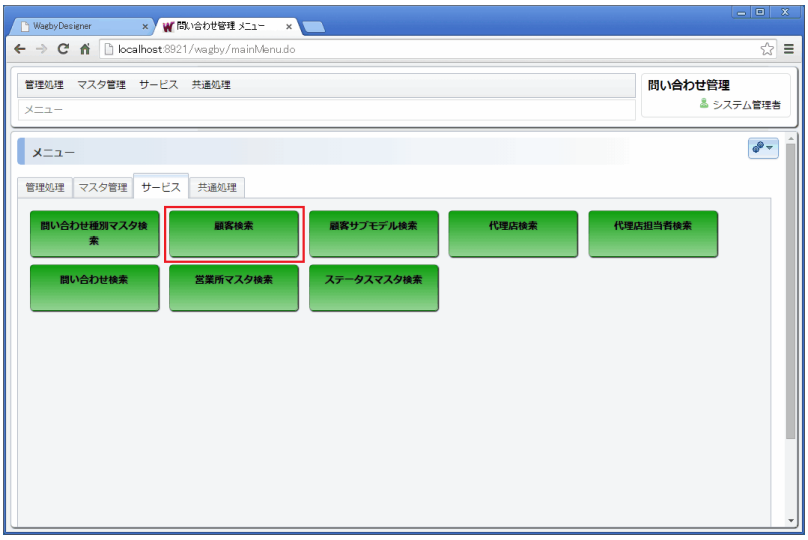


図 メインモデルの検索を行なう

メインモデルでは、サブモデルから登録したデータを共有できていることがわかります。（テーブルは1つなので、共有されています。）

図 メインモデルの検索結果は、サブモデルと同じになっている

具体的な活用シーン

スマートフォン

スマートフォン版の画面は、PC版のサブセット（一部）という場合に適用できます。PC版のモデルをメインモデルとし、スマートフォンで使うモデルをサブモデルとします。サブモデルは、メインモデルの項目の一部、という扱いになります。

その上で、サブモデルに対して「スマートフォンで使う」設定を有効にします。

複数の一覧表示画面

サブモデルを使って、1つのモデルに複数の一覧表示画面を作成する、といったことができます。ユーザごとにアクセスする画面を分けたいという場合にも有効です。

セキュリティ

権限のあるユーザにはメインモデルを見せる、権限のないユーザには項目の少ないサブモデルを見せるという設計に適用できます。

権限設定との組み合わせ

例

サブモデルを使って、ユーザの権限ごとに操作できる画面を分ける例を紹介します。

この例では「山田」「鈴木」という二つのアカウントを用意しています。

アカウント	プリンシパル	モデル
山田	顧客管理	顧客（メインモデル）
鈴木	一般	顧客(一般ユーザ用)（サブモデル）

図は、ユーザ山田による顧客の一覧表示画面の操作例です。

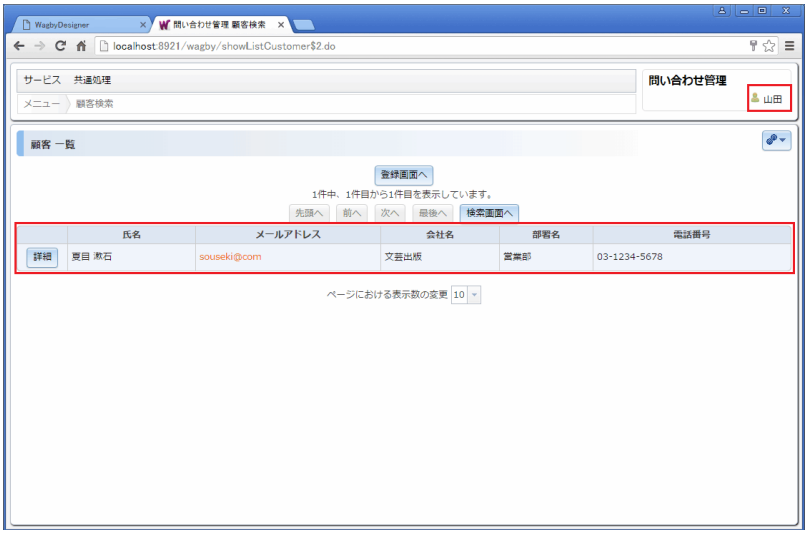


図 顧客一覧表示

図は、ユーザ鈴木による顧客の一覧表示画面の操作例です。このサブモデルは「電話番号」項目が含まれていません。

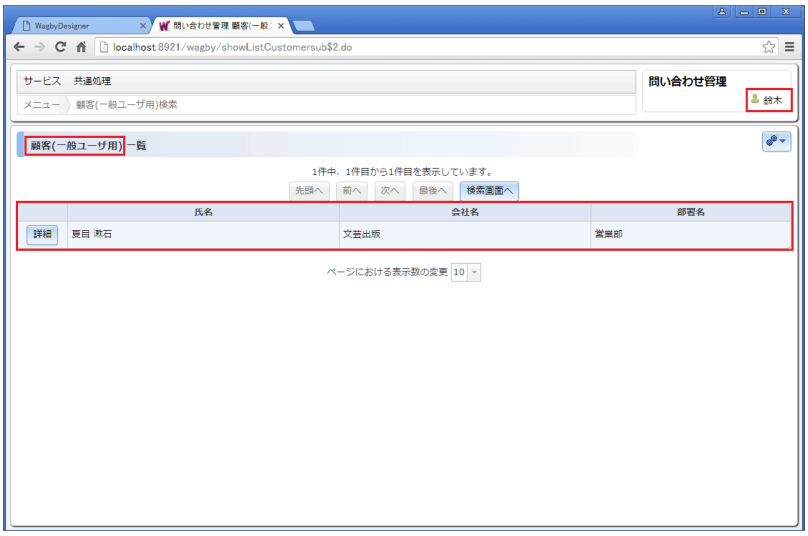


図 顧客(一般ユーザ用)一覧表示

図、図は、それぞれのユーザのプリンシパルを示しています。

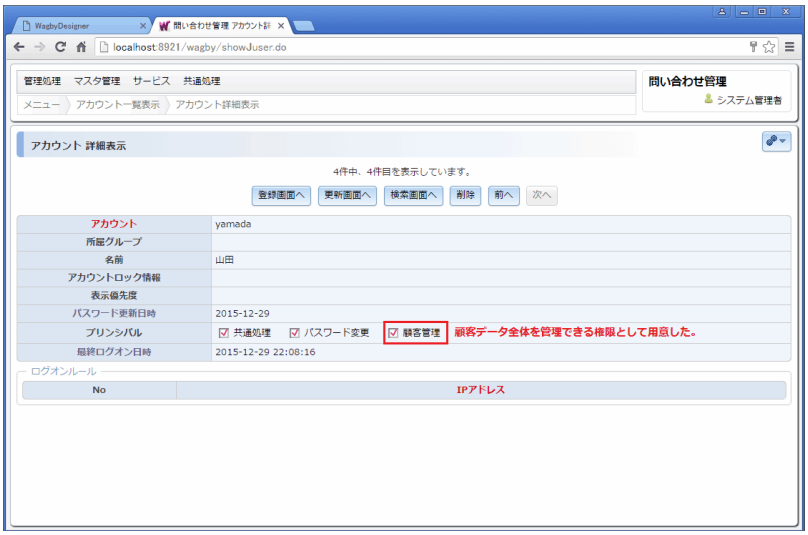


図 ユーザ山田は、プリンシパル「顧客管理」をもつ。

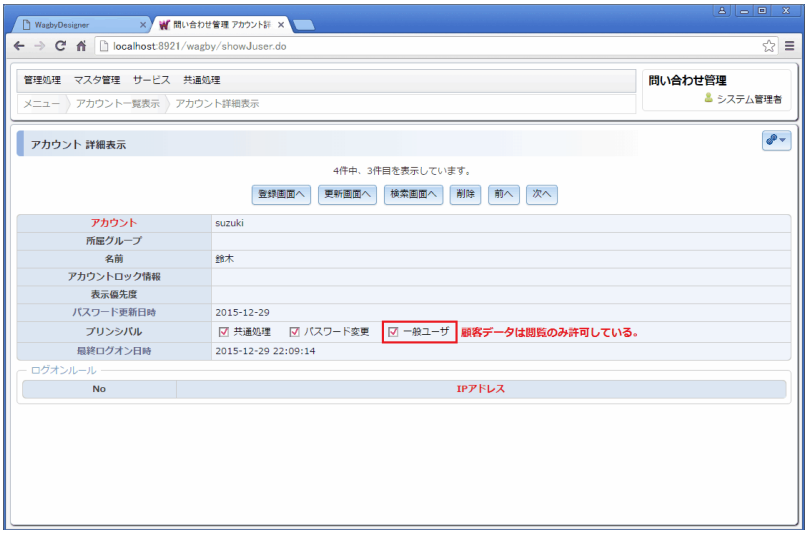


図 ユーザ鈴木は、プリンシパル「一般ユーザ」をもつ。

プリンシパルに応じて、操作できるモデルを変えることができます。ユーザ山田とユーザ鈴木では、ログオン後のメニューに表示されるモデルが異なります。ユーザ山田はメインモデルを、ユーザ鈴木はサブモデルを操作します。

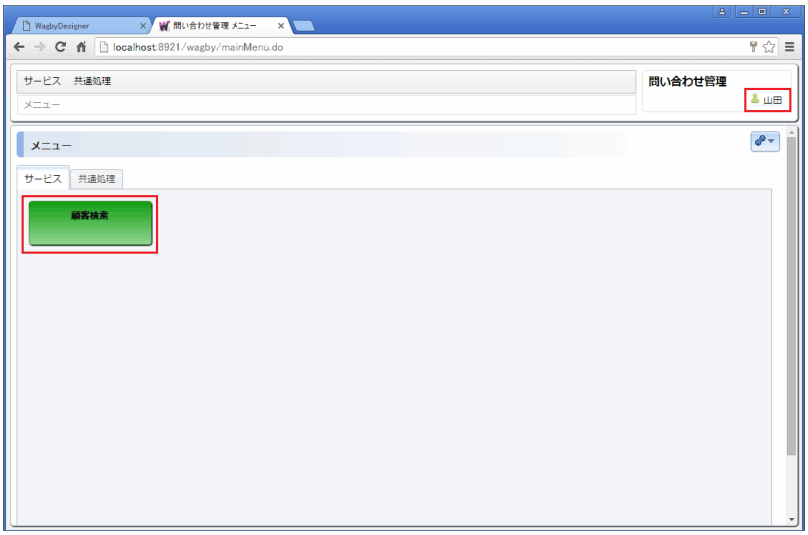


図 ユーザ山田はメインモデル「顧客」を操作できる

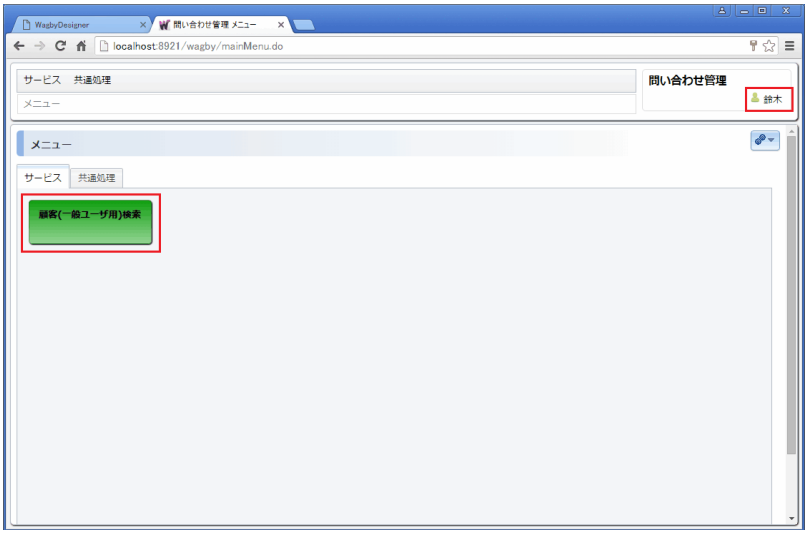


図 ユーザ鈴木はサブモデル「顧客(一般ユーザ用)」を操作できる

なお、システム管理者はすべてのモデルを操作できます。

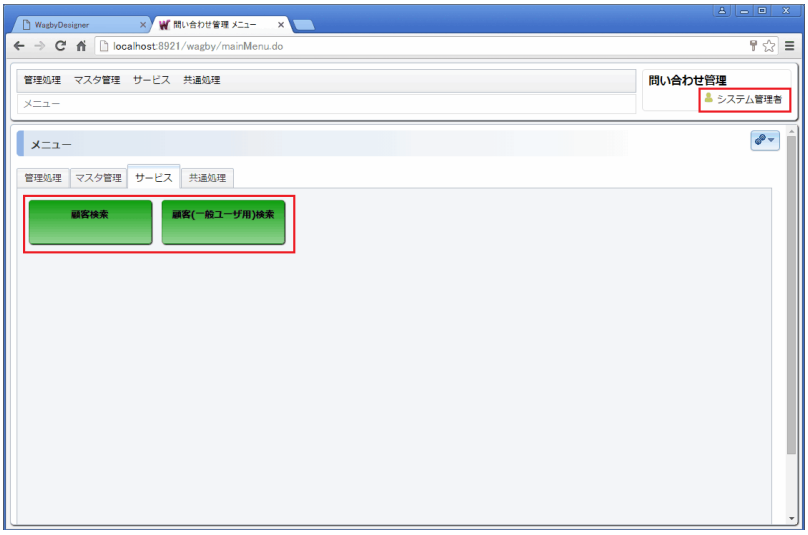


図 システム管理者はメインモデル、サブモデルいずれも操作できる

定義方法

「顧客」モデルに対して、サブモデル「顧客(一般ユーザ用)」を用意しました。（参考：サブモデルの作成方法）

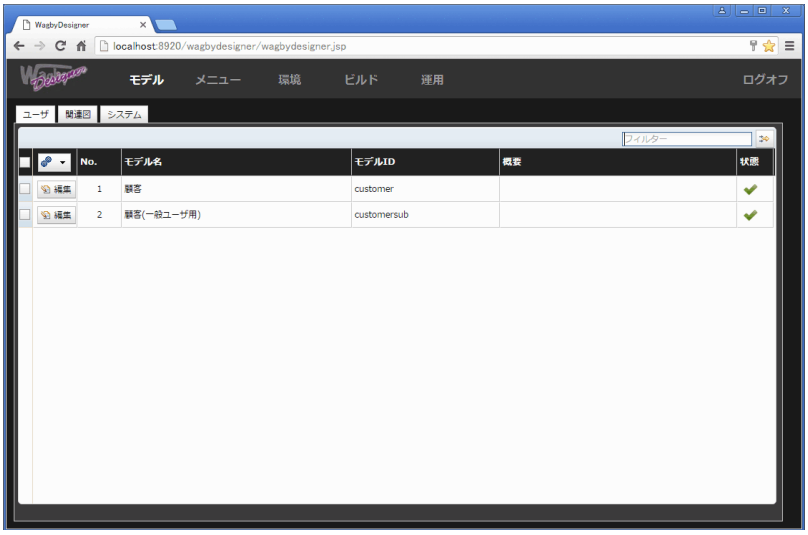


図 「顧客」と「顧客(一般ユーザ用)」モデル

図はメインモデル「顧客」、図はサブモデル「顧客(一般ユーザ用)」です。サブモデルは「メールアドレス」と「電話番号」を削除したものとなっています。

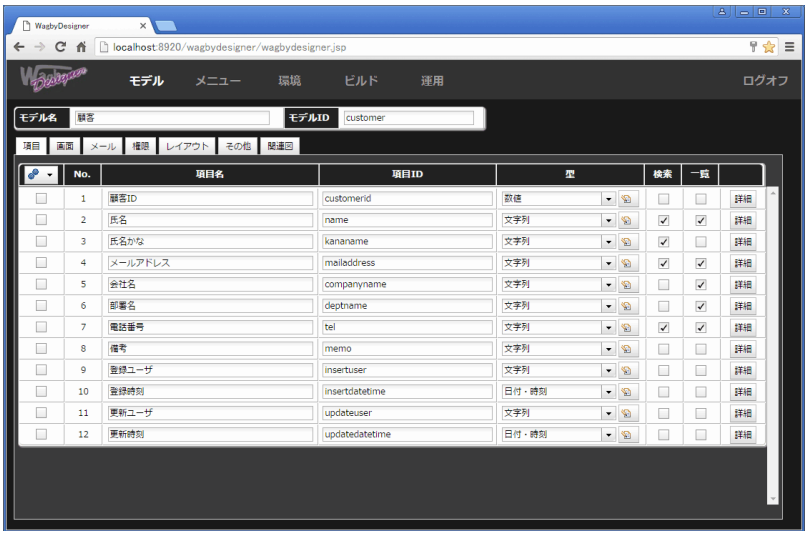


図 メインモデル「顧客」

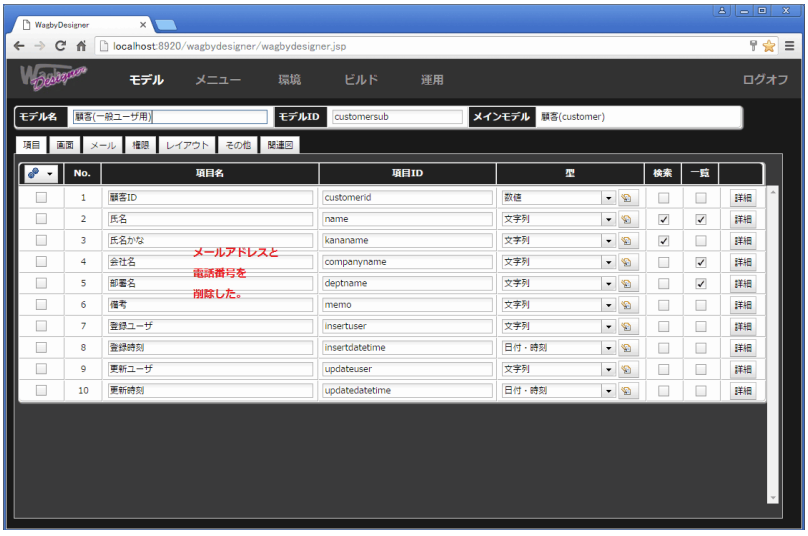


図 サブモデル「顧客(一般ユーザ用)」

権限の設定

メインモデル「顧客」に最初から備わっていたプリンシパル「一般ユーザ」を削除します。

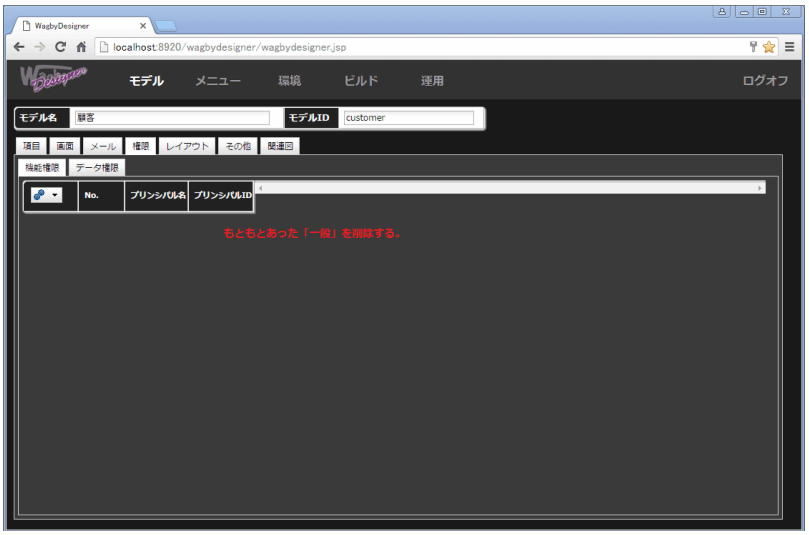


図 プリンシパル「一般ユーザ」を削除する

新しいプリンシパル「顧客管理」を用意します。すべての操作を許可します。

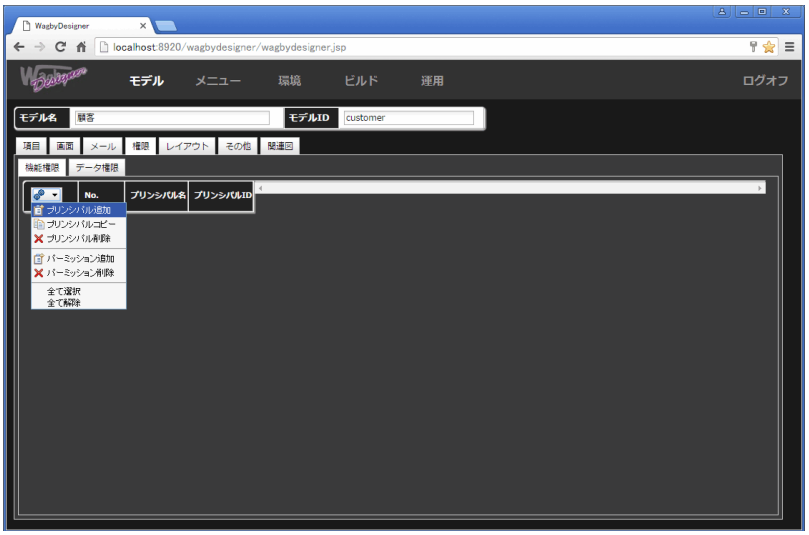


図 プリンシパル「顧客管理」を追加する(1)

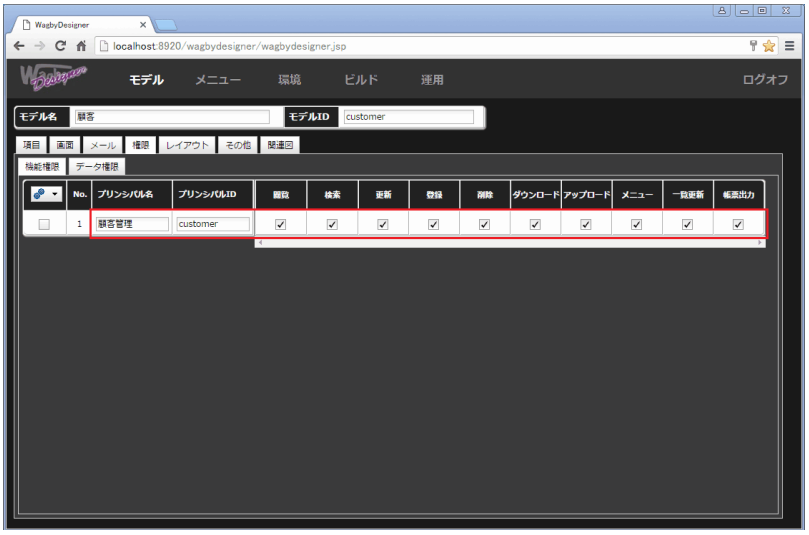


図 プリンシパル「顧客管理」を追加する(2)

サブモデル「顧客(一般ユーザ用)」のプリンシパルは、更新系の権限を無効とします。これによって閲覧のみが行えるようになります。

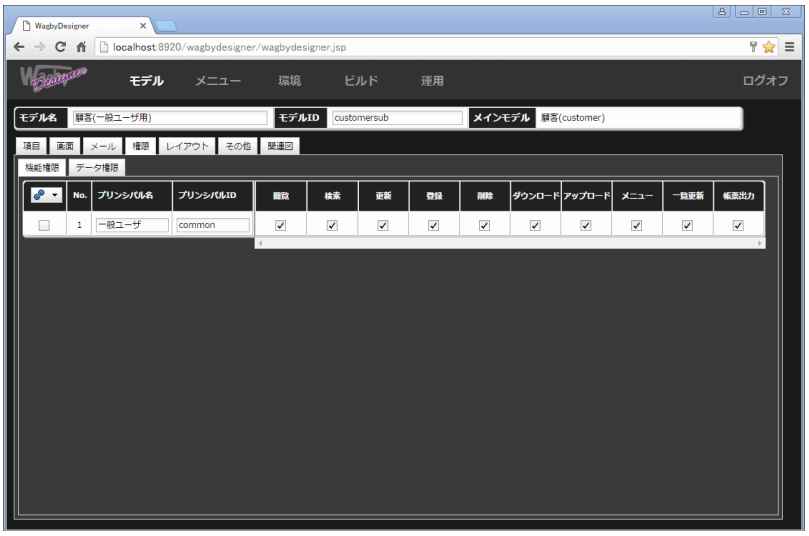


図 サブモデル「顧客(一般ユーザ用)」のプリンシパル(1)

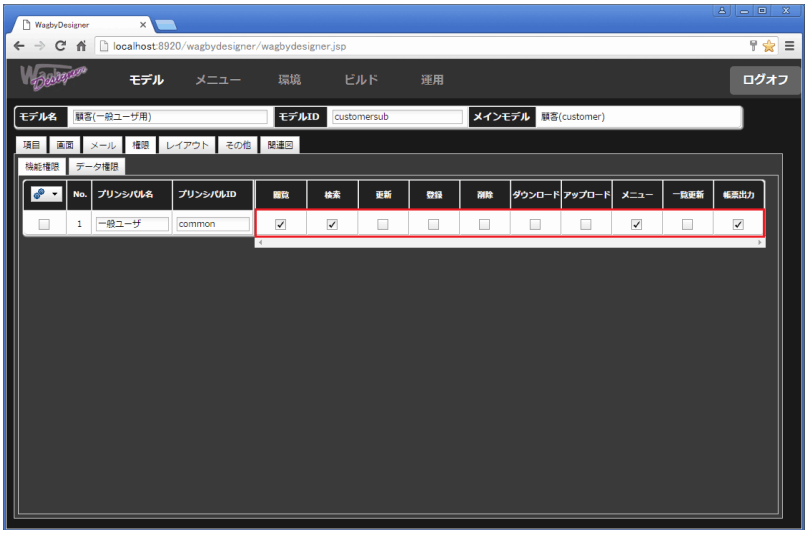


図 サブモデル「顧客(一般ユーザ用)」のプリンシパル(2)

メインモデルを変更する

　サブモデルを作成したあと、（自身の）メインモデルを変更することができます。 「画面 〉 その他 〉 モデルの関連性」から「メインモデル名」を指定します。

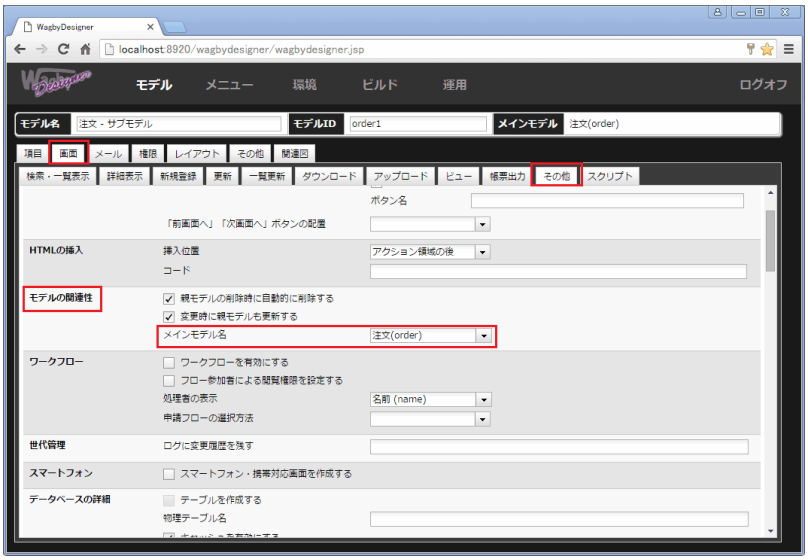


図 メインモデル名を変更する

サブモデル一覧画面からメインモデル詳細画面へ遷移する

例

　顧客（サブモデル）の一覧表示画面を図に示します。ここで「詳細」ボタンを押下すると、顧客（メインモデル）に遷移します。

図 顧客（サブモデル）一覧画面

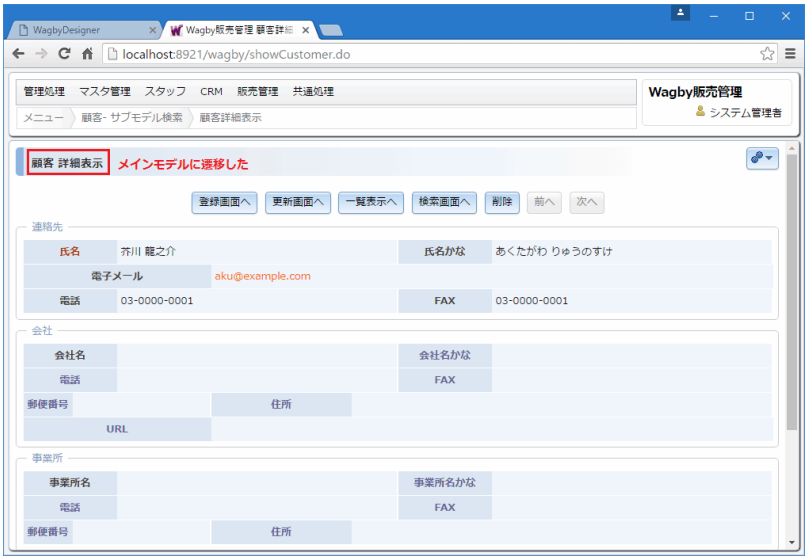


図 顧客（メインモデル）詳細画面

定義方法

サブモデルを用意します。

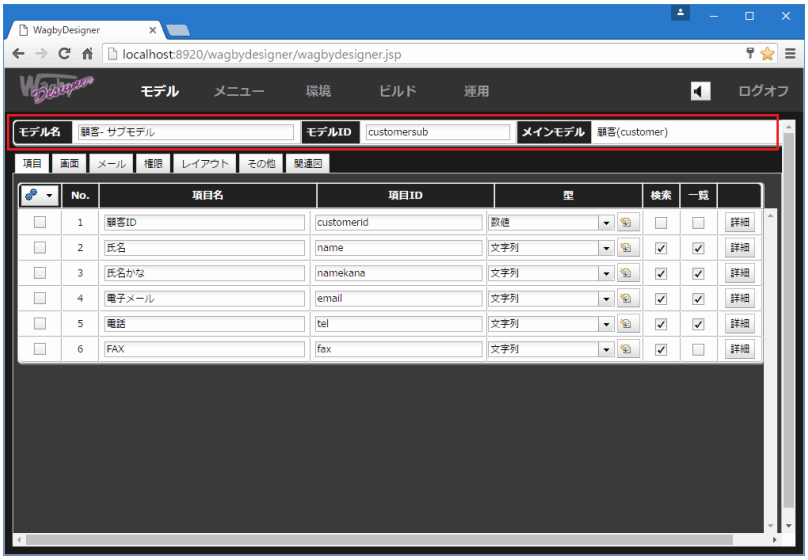


図 サブモデルを用意する

サブモデルの「画面 ＞ 検索・一覧表示 ＞ 詳細ボタン」で、「遷移先モデルID」から自身の親モデルを指定します。

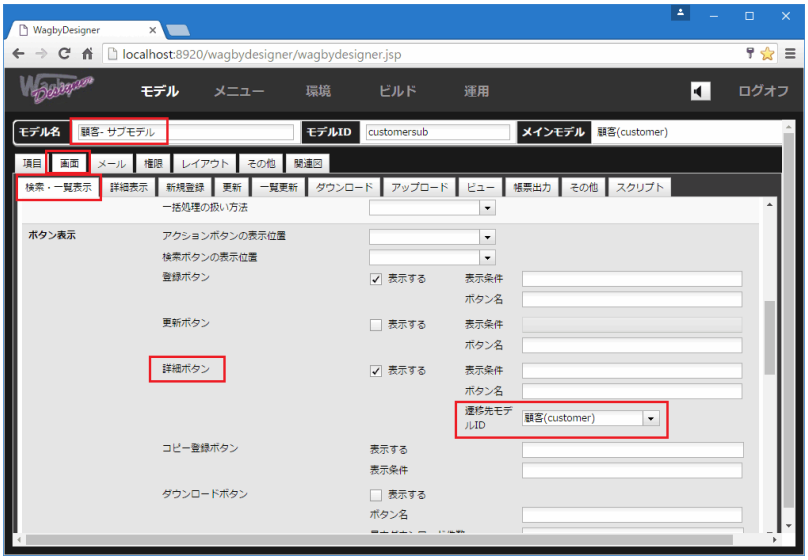


図 遷移先モデルIDを指定する

(227)

外部キー連携における子モデル一覧表示の場合

外部キー連携において、親モデル詳細画面の下部に表示される子モデル一覧表示の「詳細」ボタンと「更新」ボタンの遷移先をそれぞれ指定できます。

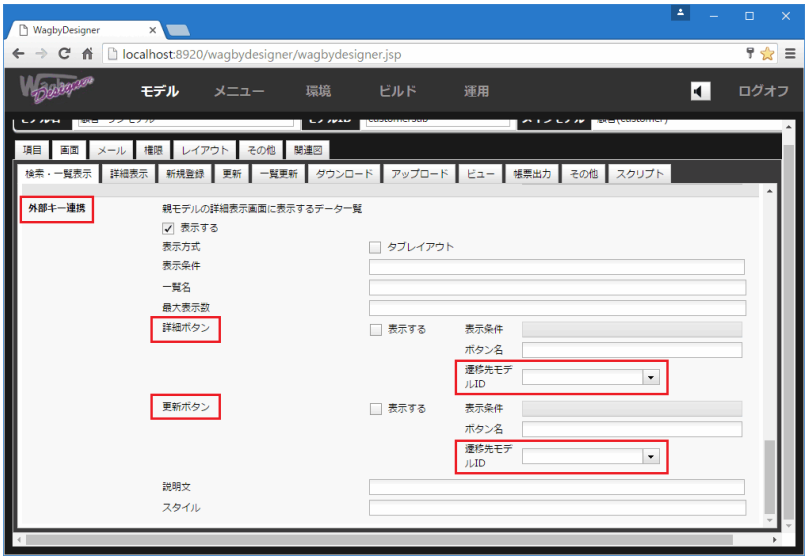


図 外部キー連携の設定

仕様・制約

- サブモデルはメインモデルの部分集合のため、テーブル構造が変わるような変更を行うことはできません。 よって、サブモデルを定義する場合、メインモデルで定義された以下の点は変更してはいけません。 （変更した場合は、ビルド時もしくは実行時にエラーとなります。）
  - 項目ID
  - 項目の型
  - 「データベース保存」対象の項目を追加すること。（データベース非保存の自動計算項目や参照連動項目等はサブモデルへ追加することができます。）

(227) この機能は、遷移先対象は親モデルには限定されません。主キー（名前、型）が同じであれば、異なるモデルへ遷移させることもできます。

    - モデル項目定義シートの「リレーショナルデータベースの詳細」「必須チェック」等の設定について、メインモデルと異なる定義をすること（メインモデル、サブモデルともに同じ設定にする必要があります。）
- サブモデルのサブモデルを定義することはできません。
- サブモデルでデータの削除を行う場合、メインモデルに含まれる「モデル参照（チェックボックス）」「繰り返し項目」「繰り返しコンテナ項目」を必ずサブモデルでも用意してください。（サブモデルで不要な場合は、隠し項目とすると良いでしょう。）この対応を行わない場合、サブモデルからのデータ削除に失敗します。
- メインモデルとサブモデルで、同じ項目に対してモデル参照先の項目を変更する場合、モデル参照先の「型」は同じとなるように配慮してください。
- 親子モデル関係において、親モデルは子モデルの値を自動計算式で利用できますが、親のサブモデルでは（子モデルを）利用できません。例えば親モデル parent と子モデル child があったとき、親モデルのサブモデル parentsub 内の項目で計算式に child_lp.item1 と記述することはできません。このような場合はサブモデルの方でも、同じ子モデルを参照するようにするように設定してください。

よくある質問と回答

サブモデルにメインモデルと違うグループ権限を設定することはできますか。

サブモデルにメインモデルと違うグループ権限を設定することは可能です。

例えば、メインモデルは権限3とし、サブモデル1は権限2、サブモデル2は権限3にするということができます。

項目の必須チェックはメインモデルとサブモデルで同じようにするという仕様ですが、「警告」はメインモデルとサブモデルで変えることはできますか。

必須チェックについては、指定する箇所に「○」「警告」「未指定」の3種類の値が設定可能です。「○」を設定した場合データベースのテーブル定義にて、対応する列にNOT　NULL制約が設定されます。このため、いずれかのモデルに「○」を指定した場合、メインサブ関係にある他のモデルにも「○」を指定する必要があります。これは同じテーブルを使うためです。

警告や未指定は組み合わせることが可能です。例えば下記のようになります。

設定可能な組み合わせ

メインモデル	サブモデル
○	○
警告	警告
（未指定）	警告
警告	（未指定）
（未指定）	（未指定）

設定できない組み合わせ

メインモデル	サブモデル
○	警告
○	（未指定）
警告	○
（未指定）	○

メインモデルとサブモデルという関係ではなく、物理テーブルを共用することはできますか。

可能です。詳細は"モデル ＞ データベースの詳細 ＞ 物理テーブル名"をお読みください。

(228) (229)

(228) ただし Wagby が提供するロックやキャッシュの考慮を開発者自身が行う必要があります。

(229) メインモデルとサブモデルの関係を設定すると、ロックやキャッシュは適切に制御されます。

ボタン名と表示条件

ボタン名の変更例

ボタン名に式を適用した例を図,図に示します。ログオンアカウントによってボタン名が変わります。

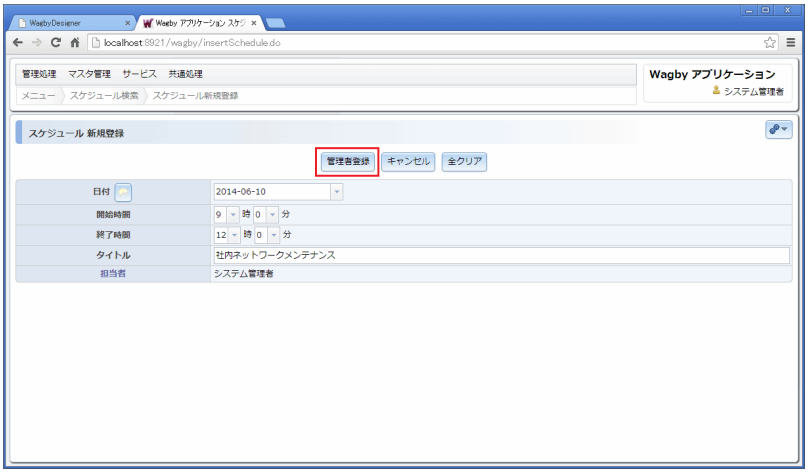


図 システム管理者が利用する場合はボタン名が「管理者登録」になる

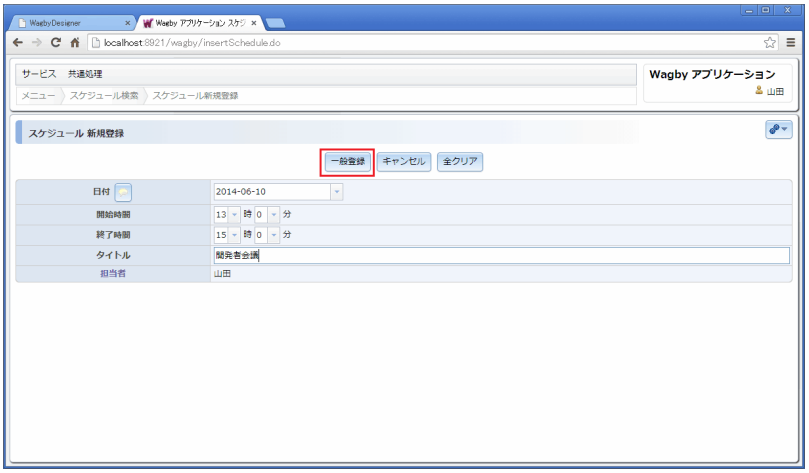


図 一般が利用する場合はボタン名が「一般登録」になる

ボタン名の定義方法

各画面に用意された「ボタン表示」の「ボタン名」は固定の名称だけでなく式が設定できます。

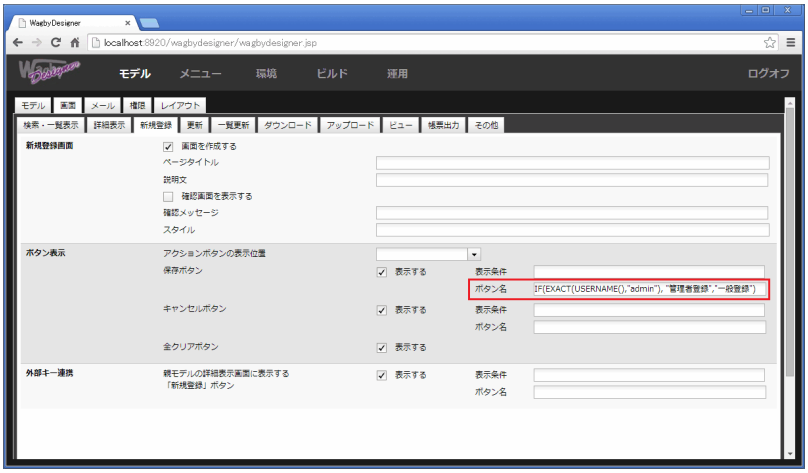


図 ボタン名の設定

固定名称を指定する場合

ボタン名をそのまま指定します。

```
スケジュール登録
```

式を用いる場合

式の計算結果の戻り値の型が文字列型となるようにします。

```
IF(EXACT(USERNAME(),"admin"), "管理者登録","一般登録")
```

式の詳細は「計算式」をお読み下さい。

ボタン表示条件

「ボタン表示条件」を設定すると、条件に合致した場合にのみ、当該ボタンが画面に表示されるようになります。図は "日付が本日以降の場合" という条件を指定した例です。

```
DATE(${sdate})>=DATE(TODAY())
```

式の戻り値は真偽値としてください。trueまたはfalseが戻るようにします。

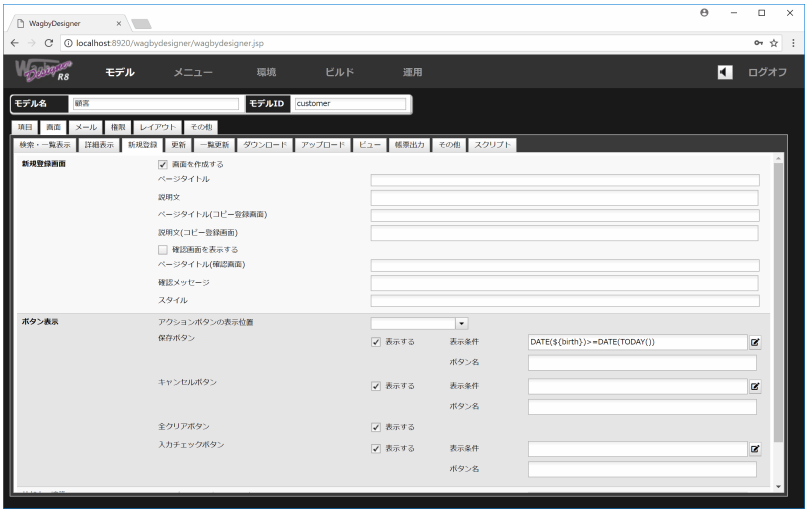


図 日付を条件に適用した

条件となる項目の値に連動してボタンの表示・非表示を制御する

図の例では、新規登録画面を開いた時に条件に合致しなければ、登録ボタンは非表示となります。その後、条件となる日付項目を入力しても、ボタンは非表示のままです。

この動作を修正し、条件となる項目の値に連動してボタンの表示・非表示を制御させる場合は、図に示すように（条件となっている項目について）「値変更時に画面を再描画する」設定を有効にします。[230]

図 値変更時に画面を再描画する

条件式の記述例

項目 item の値が 1 のとき、登録ボタンを表示する

```
${item}==1
```

項目 item の値が "東京" のとき、登録ボタンを表示する

```
EXACT(${item}, "東京")
```

EXACT関数は文字列の比較を行います。

モデル schedule について、権限 "perm_button" を持っている利用者が操作する場合にボタンを表示する

```
HASMODELPERMISSION("perm_button", "schedule")
```

HASMODELPERMISSION関数は、利用者が指定した権限を有しているかどうかを返します。

プリンシパル "publicuser" を持っている利用者が操作する場合にボタンを表示する

```
HASPRINCIPAL("publicuser")
```

管理者のみボタンを表示する

```
HASPRINCIPAL("all")
```

allは管理者権限を意味します。

[231]

---

[230] この設定を行うと、当該項目の値が変更されるたびに、フォームの値を再送信するようになります。ボタン表示判定の制御はサーバ側で行います。

[231] 管理者権限でログオンしたとき、HASPRINCIPAL 関数は常にtrueを返します。そのため、管理者権限のみボタンを表示する、という場合、カッコ内に与える引数（プリンシパル名）は実際にはなんでもよい、ということになります。しかし他のプリンシパルと区別するため、上記のように all と指定することを推奨します。

検索条件の値を用いる（8.1.1）

ボタン表示制御の式に特別表記 ${cond.項目名} を指定することができます。 「検索の実行」ボタンを押すことで検索条件部の値を用いて判断します。

(232)

例えば一覧帳票を出力する際、検索結果により帳票出力ボタンの制御するために、次のような式を記述することができます。

検索項目 item1 の値が "A" の場合:

```
EXACT(${cond.item1},"A")
```

検索項目 status がモデル参照項目で、そのID部の値が 1 の場合:

```
${cond.status#id}==1
```

検索項目 num が整数型項目で範囲検索が有効となっており、その値が 1 の場合:

```
${cond.num1jshparam}==1
```

数値および日付型で範囲検索が有効の場合、開始の項目には接尾語 "1jshparam" を、終了の項目には "2jshparam" を付与してください。

外部キー関係での式の扱い

外部キー関係では、親モデルの画面に複数の子モデルを一覧表示することができます。 この場合のボタン表示制御は次のルールになります。

- 条件式に含めることができる項目は親モデルになります。
- ただし、親からみた子の一覧表示として、"子モデル_lp" という表記が利用できます。

例：「外部キー連携>表示条件」で、親モデルの項目 item1 の値が 100 より大きい場合に子モデルの新規登録ボタンを表示する。

```
${item1} > 100
```

例：この親モデルに紐づく子モデルの数が 3 未満のとき、子モデルの登録ボタンを表示する。

```
COUNT(${support_lp}) < 3
```

COUNT関数は、子レコードの総数を返します。

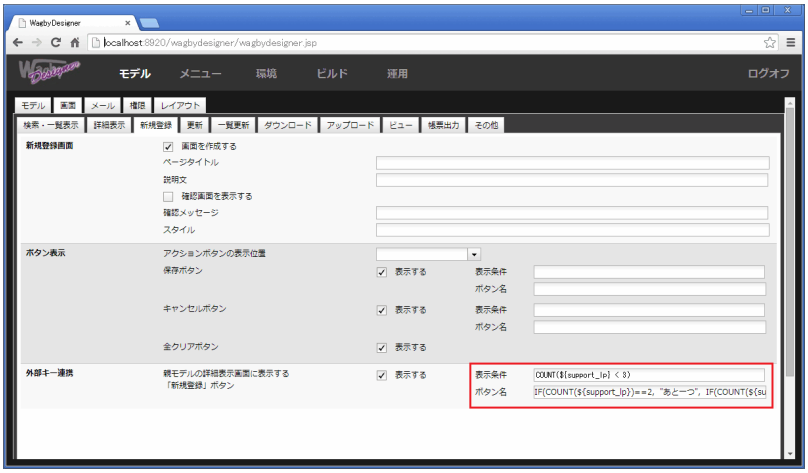


図 外部キー連携でのボタン表示制御

ボタン名を式で制御する例も示します。子モデルの数によってボタン名が変わります。

(232) Wagbyが内部で利用している、検索条件を保持する「コンディションモデル」を用います。

```
IF(COUNT(${support_lp})==2,
    "あと一つ",
    IF(COUNT(${support_lp})==1,
        "あと二つ",
        "あと三つ"
    )
)
```

制約

この指定は子モデル一覧更新画面の「新規」ボタンには反映されません。（一覧更新画面の「新規」ボタンの制御は現在、未対応です。）

Ajaxによる画面描画

Ajaxの概要

Ajax（Asynchronous JavaScript ＋ XML）とは、画面再描画することなしに画面（ページ）の一部を書き換える技術を指します。

一般的な Web アプリケーションでは、クライアントからサーバにデータを要求すると、画面再描画によってページ内容を更新します。しかし Ajax では、登録や更新画面を開いたままの状態で、裏で通信処理を行い、画面の一部を書き換えます。（これを非同期通信と呼びます。）

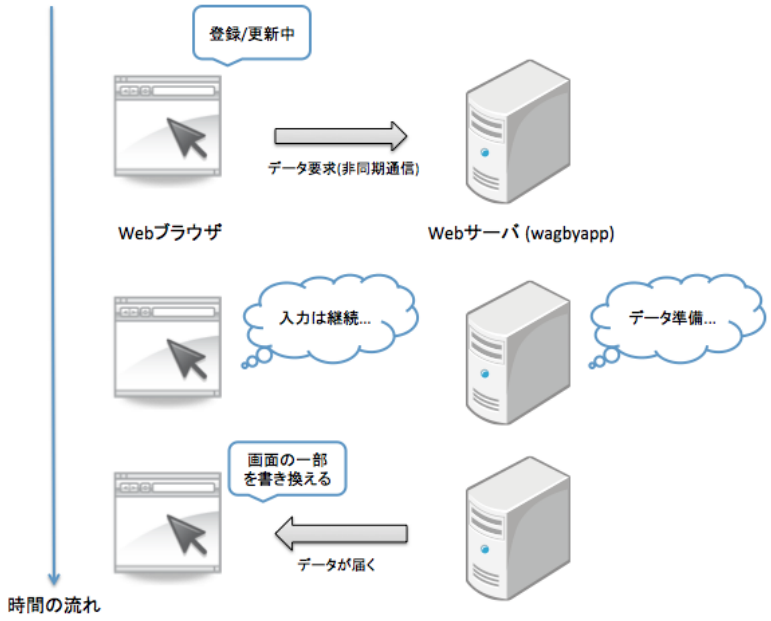


図 Ajax動作のイメージ

Ajaxが動作するパターン

WagbyではAjaxに関する設定箇所はありません。登録や更新画面で、以下のパターンについて自動的にAjaxが動作します。

- 自動計算式が設定された項目。
- 参照連動が設定された（他モデル参照）項目。
- 他モデル参照項目(リストボックス等)の絞り込みに指定された項目。
- 他モデル参照項目(リストボックス等)にて、モデル参照(絞込)の自モデル項目名に指定された項目。
- 他モデル参照項目にて、サジェスト機能が使われている項目。

Ajaxの多重度

1利用者につき、同時に処理される Ajax は1つとなります。（利用者のHTTPセッションごとに1つのAjax処理が逐次、実行されます。）

画面遷移（1）詳細画面から登録画面へ遷移する

詳細画面から登録画面への遷移

顧客モデルと、お得意様モデルの二つのモデルを用意します。　図, 図にそれぞれの定義を示します。両者は（定義上は）関連がなく、独立したモデルです。

（項目名は似せています。お得意様モデルは、顧客モデルの各項目名に接頭語 ″cs″ を付与しています。これは説明をわかりやすくするためであり、必ずしもこのような項目名とすることはありません。）

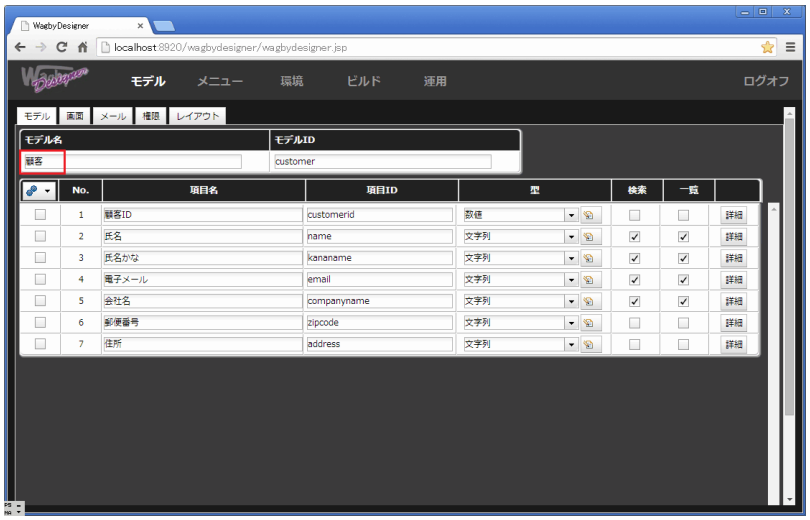


図 顧客モデルの定義

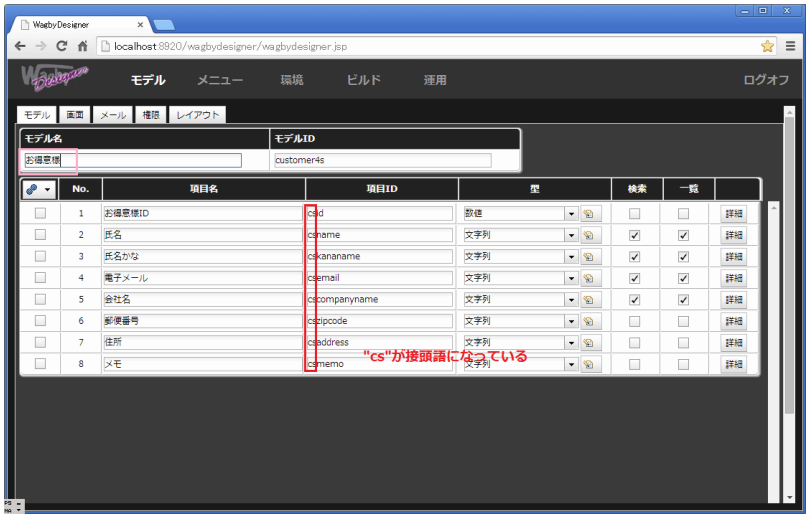


図 お得意様モデルの定義

顧客モデルの詳細画面から、お得意様モデルの登録画面へ遷移するボタンを用意した例を図に示します。同ボタンを押下することで、顧客モデルの各項目の値を（パラメータとして）お得意様モデルに渡すようにしています。

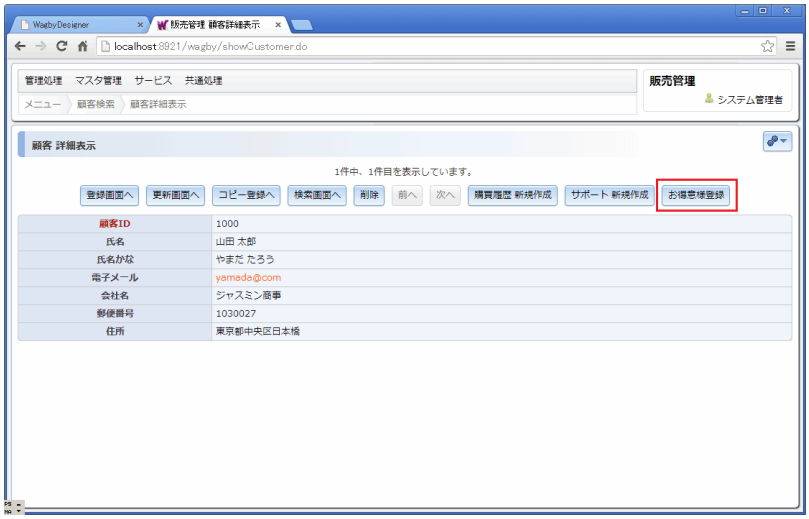


図 お得意様モデルの登録画面へ遷移するボタンを用意した例

お得意様モデルの新規登録画面へ遷移した例を図に示します。値が引き渡されていることがわかります。

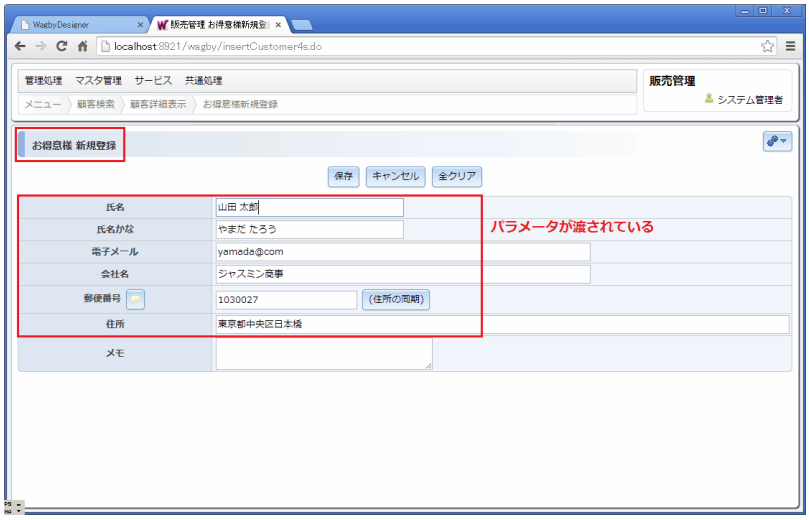


図 お得意様モデルの新規登録画面

この例に示すように、任意のモデルの画面を開くことができます。パラメータを渡すこともできます。

定義方法

お得意様モデルの設定

新規登録画面を開いたときに REQUEST 関数を適用します。 外部から渡されるパラメータを受け取ります。[233]

[233] 外部から渡されるときのキー名が項目名と異なる場合、REQUEST関数に（外部の）キー名を指定します。

図 REQUEST関数の設定

数値型項目の場合

REQUEST関数の戻り値は文字列型です。受け取り側（ここでは「お得意様」モデル）の項目が数値型の場合、TOINT関数などを用いて文字列を数値型に変換してください。

日付型項目の場合

REQUEST関数の戻り値は文字列型です。受け取り側（ここでは「お得意様」モデル）の項目が数値型の場合、DATEVALUE関数などを用いて文字列を日付型に変換してください。

顧客モデルの設定

図で示した「顧客モデル」で、「画面>その他>Javaソースコードの設定」を開きます。 ここで追加ボタンを次のように定義します。

項目	説明	入力例
画面名	どの画面にボタンを追加するかを指定します。ここでは"詳細表示画面"とします。	詳細表示画面
表示する	有効にします。	（チェック）
表示条件	常にボタンを表示する場合、本欄は空白とします。条件については"ボタン表示条件"をお読みください。	
ボタン名	任意の名前を指定します。式で表現することもできます。	お得意様登録
イベント名	遷移した画面に対するイベントを指定します。[後述]	New
アクション名	遷移する画面を指定します。[後述]	insertCustomer4s
追加パラメータ	遷移する画面へパラメータを指定することができます。[後述]	csname=${name}

図 追加ボタンの設定

イベント名

画面毎に、受け取ることのできるイベントが用意されています。新規登録画面で受理できるイベントは次のとおりです。

イベント名	説明
New	新規登録画面を開きます。

イベント名は大文字小文字を区別します。正確に入力してください。

アクション名

モデルと画面の組み合わせから構成される文字列です。アドレスバー(URL)の一部になります。

新規登録画面は次のようになります。

```
"insert" + モデル名(キャメル記法)
```

このルールにより、モデル "customer4s" の新規登録画面に対応するアクション名は "insertCustomer4s" になります。 拡張子は不要です。

(234)

追加パラメータ

登録画面に対して、直接、初期値を渡すことができます。

書式は「キー=値」となっています。キー部分は（遷移先モデルの）項目IDになります。 値部分にプレースホルダ（${ ... }）を用いて、遷移元の項目の値を渡すことができます。

複数のパラメータを渡す場合は&で区切ります。

図の動作を実現するための追加パラメータは次のようになります。 (235)

```
csname=${name}&cskananame=${kananame}&csemail=${email}&cscompanyname=${companyname}&cszipcode=
${zipcode}&csaddress=${address}
```

制約

追加パラメータに記載する項目名に以下を指定することはできません。

- 繰り返し項目
- 繰り返しコンテナ項目
- このモデルに紐づく外部キー子モデル

モデル参照項目の扱い

図,2の例は、顧客モデルとお得意先モデル内の「会社名」は文字列型でした。 続いて「会社モデル」と「顧客モデル」を分割した例を示します。顧客モデルの「会社名」は、会社モデルの主キーを保持する構造に変わります。

(234) キャメル記法 : 先頭の1文字を大文字とします。モデル名に含まれるアンダースコアは取り除き、その次の文字を大文字にします。例えば my_funny_valentine は MyFunnyValentine になります。

(235) この例ではキー部分（csname,cskananame）はお得意様モデルの項目名と一致しています。そのため図でREQUEST関数の引数を省略しています。 今回、お得意様モデルの主キー（csid）は順序を利用して自動採番しています。そのため外部からパラメータとして渡す必要はありません。

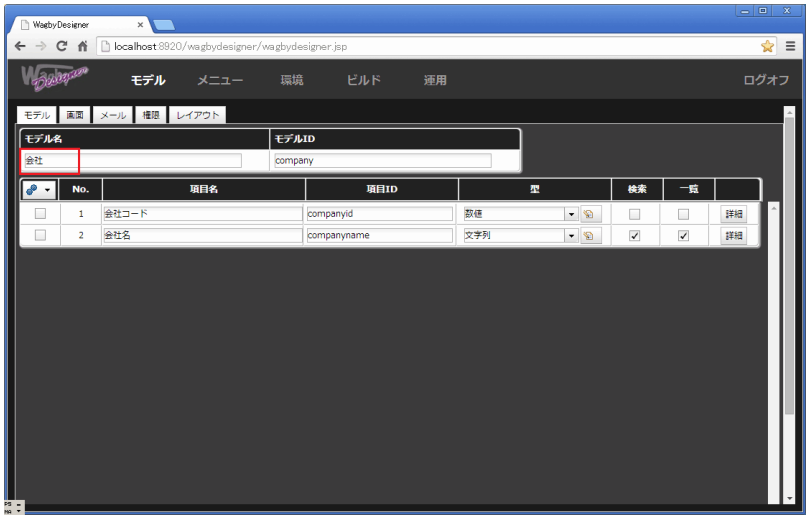


図 会社モデルの定義

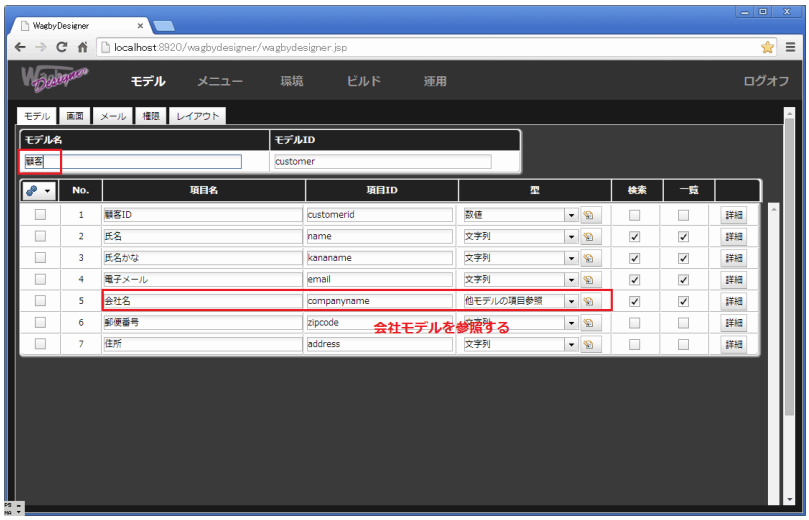


図 修正された顧客モデル

実行

顧客モデルにおける「会社名」は（文字列型から）モデル参照に変わっています。このとき「お得意様登録」ボタンを押下すると、会社名コード（ID部）が渡されるようになります。

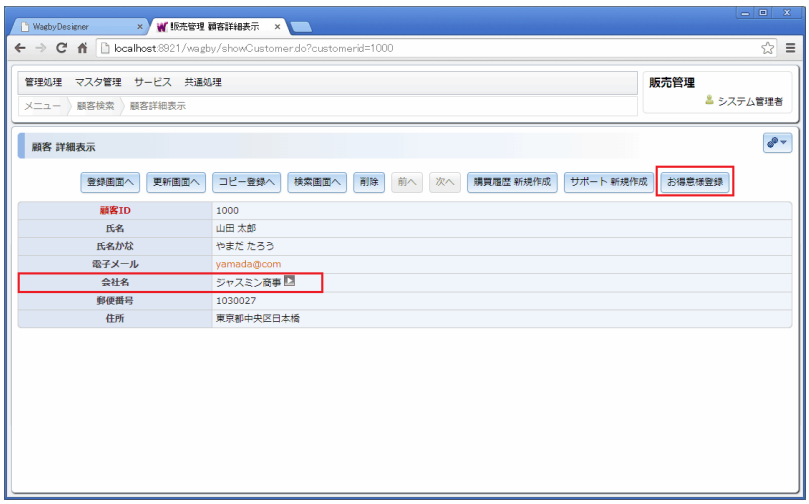


図 修正された顧客モデルの詳細画面

モデル参照についても、パラメータが渡っていることが確認できます。

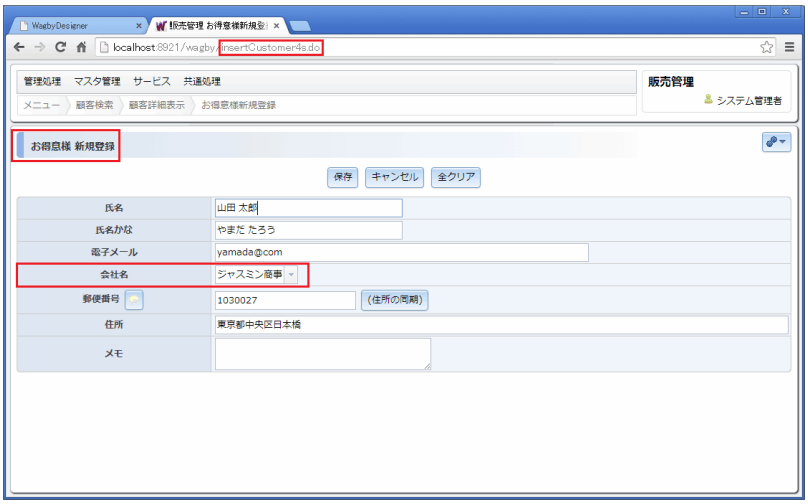


図 ボタン押下後のお得意様登録画面

設定

お得意様モデル

お得意様モデルの会社名項目の初期値は、コード(ID部)を受け取るようにします。 初期値の式を数字型に変換するためのTOINT関数を適用します。

```
TOINT(REQUEST())
```

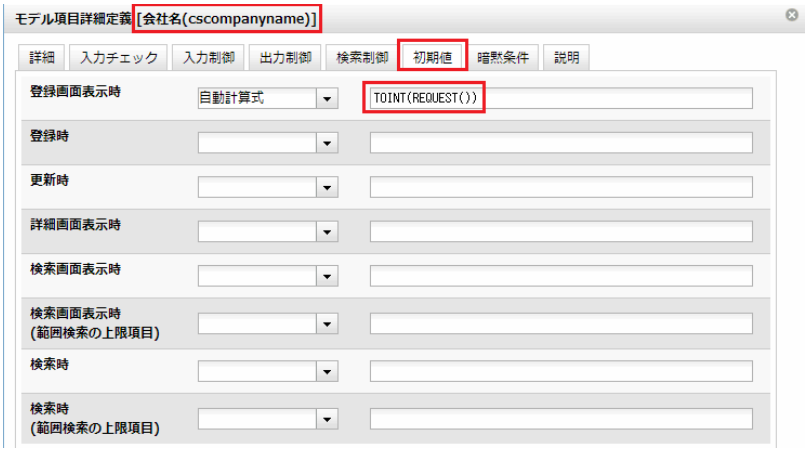


図 初期値の式の変更(TOINT関数の適用)

顧客モデル

変更はありません。パラメータ部 "cscompanyname=${companyname}" も、そのまま利用できます。 (236)

仕様・制約

ボタンの配置

(236) 顧客モデルのcompanynameはモデル参照に変わっています。このとき、上記パラメータの ${companyname} 指定は自動的にコード(ID部)を渡すと解釈されます。このルールの詳細は"仕様・制約"をお読みください。

ボタンの表示順序を制御することはできません。オリジナルボタンは常にボタン表示部の右側に配置されます。

モデル参照項目のルール

"画面遷移パラメータまとめ > モデル参照項目のルール"をお読みください。

起点は詳細画面から

本機能は登録画面・更新画面から遷移させることはできません。詳細画面が起点となります。

説明

新規登録および更新画面にボタンを追加する場合、この画面遷移は必ず同じモデルに戻ることが仕様となってます。つまり登録画面から詳細画面へ遷移したり、更新画面から検索画面へ遷移させることはできません。

その理由ですが、新規登録および更新画面は「Webフォーム」となっており、フォーム内に配置されるボタンはすべてこの登録（更新）画面に POST されるようになっています。つまりフォームの遷移先は常に一つ（自分自身）であり、遷移先の異なるボタンを用意することはできないようになっています。

スクリプトを使って、この制約を回避することができます。

画面遷移（2）検索画面へ遷移する

詳細画面から検索画面への遷移

見積書モデルの詳細画面から、顧客モデルの検索画面へ遷移するボタンを用意した例を図に示します。 ここでは「会社名」の値を検索条件として再利用しています。

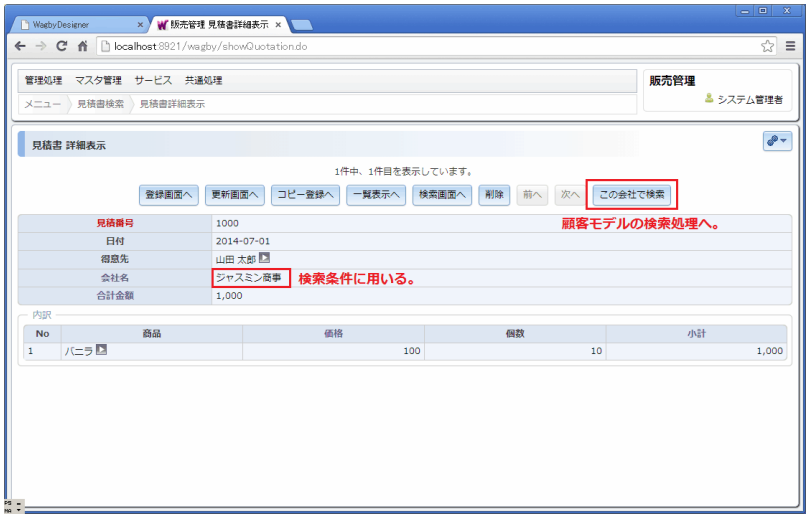


図 顧客モデルの検索画面へ遷移するボタンを用意した例

このボタンを押すと、検索条件に会社名が渡された状態で、顧客モデルの検索処理が行われます。 図では、検索と一覧を同一画面にまとめています。

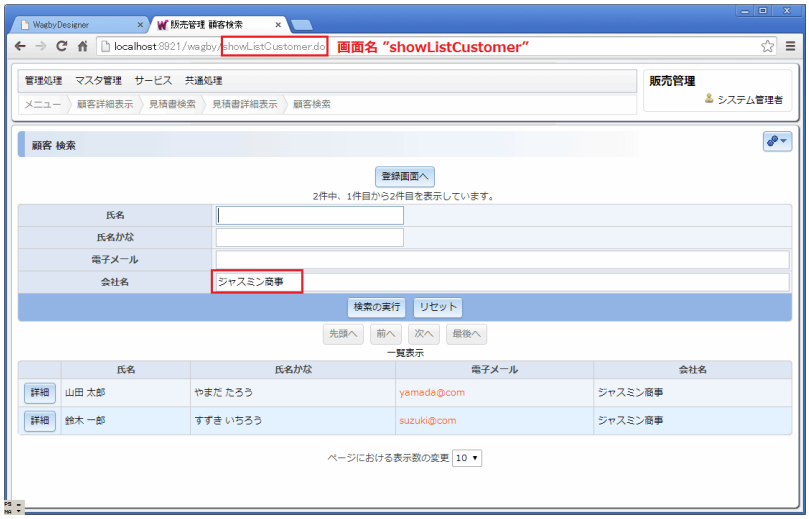


図 顧客モデルの検索＆一覧表示画面

定義方法

図で示した「見積書モデル」で、「画面>その他>Javaソースコードの設定」を開きます。 ここで追加ボタンを次のように定義します。

項目	説明	入力例
画面名	どの画面にボタンを追加するかを指定します。ここでは"詳細表示画面"とします。	詳細表示画面
表示する	有効にします。	（チェック）
表示条件	常にボタンを表示する場合、本欄は空白とします。条件については"ボタン表示条件"をお読みください。	
ボタン名	任意の名前を指定します。式で表現することもできます。	この会社で検索
イベント名	遷移した画面に対するイベントを指定します。[後述]	Reset
アクション名	遷移する画面を指定します。[後述]	showListCustomer
追加パラメータ	遷移する画面へパラメータを指定することができます。[後述]	companyname=${companyname}

図 追加ボタンの設定

イベント名

画面毎に、受け取ることのできるイベントが用意されています。検索画面で受理できるイベントは次のとおりです。

イベント名	説明

Reset	現在の検索条件をリセットし、外部パラメータを用いた検索処理を行います。今回のように外部からパラメータを渡す場合に有効です。
Search	最初にセットされた検索条件を使います。このため二回目以降の外部パラメータはセットされません。これは検索画面と一覧表示画面を行き来するような画面遷移で、検索条件を保持するためのイベントです。
Search_Condition	Searchと同様ですが、パラメータがない場合は空の検索条件が指定されたものとして扱います。

(237)

アクション名

モデルと画面の組み合わせから構成される文字列です。アドレスバー(URL)の一部になります。

検索と一覧表示画面は次のようになります。

```
"showList" + モデル名(キャメル記法)
```

このルールにより、モデル "customer" の検索一覧表示画面に対応するアクション名は "showListCustomer" になります。 拡張子は不要です。

(238)

追加パラメータ

検索画面に対して、直接、検索条件を渡すことができます。

書式は「キー=値」となっています。キー部分は（遷移先モデルの）項目IDになります。 値部分にプレースホルダ（${ ... }）を用いて、遷移元の項目の値を渡すことができます。

複数のパラメータを渡す場合は&で区切ります。

図の動作を実現するための追加パラメータは次のようになります。イコールの左側は顧客モデルの項目で、今回は検索条件になります。イコールの右側は見積書モデルの項目です。

```
companyname=${companyname}
```

受ける側（検索条件）の初期値設定

受側のモデル（この例では顧客モデル）の当該項目について、「初期値 > 検索画面表示時」に REQUEST 関数を設定します。

(237) イベント名は大文字小文字を区別します。正確に入力してください。

(238) キャメル記法 : 先頭の1文字を大文字とします。モデル名に含まれるアンダースコアは取り除き、その次の文字を大文字にします。例えば my_funny_valentine は MyFunnyValentine になります。

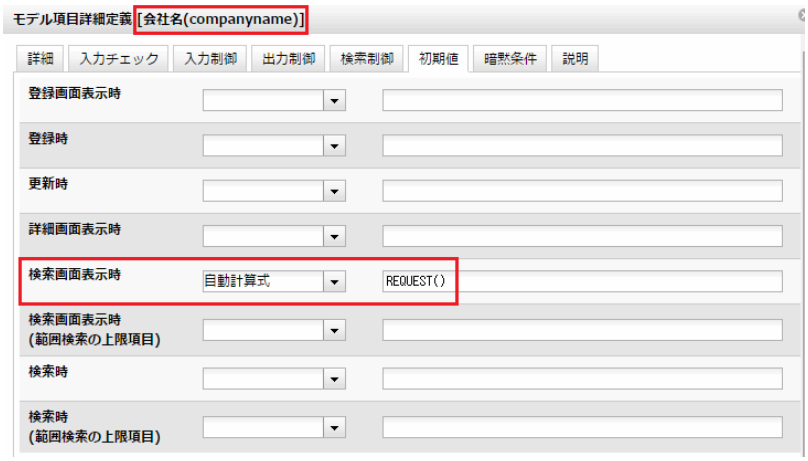


図 検索条件の初期値

制約

追加パラメータに記載する項目名に以下を指定することはできません。

- 繰り返し項目
- 繰り返しコンテナ項目
- このモデルに紐づく外部キー子モデル

モデル参照項目の扱い

図,2の例は、顧客モデル内の「会社名」は文字列型でした。 続いて「会社モデル」と「顧客モデル」を分割した例を示します。顧客モデルの「会社名」は、会社モデルの主キーを保持する構造に変わります。

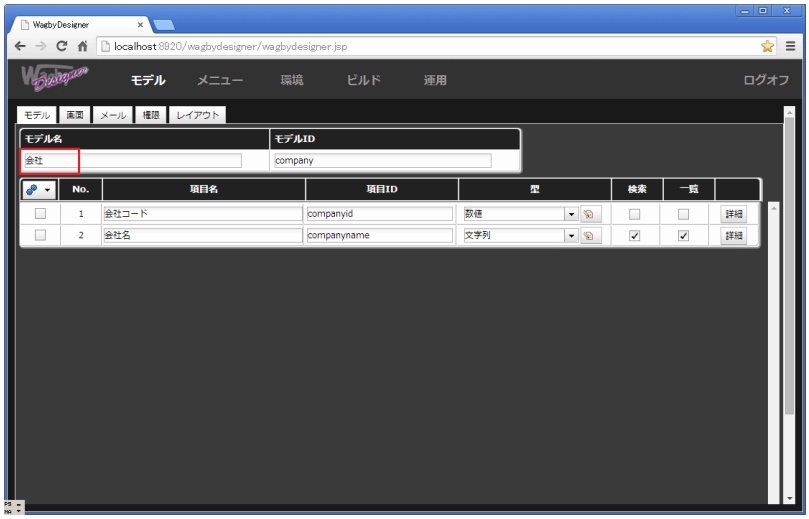


図 会社モデルの定義

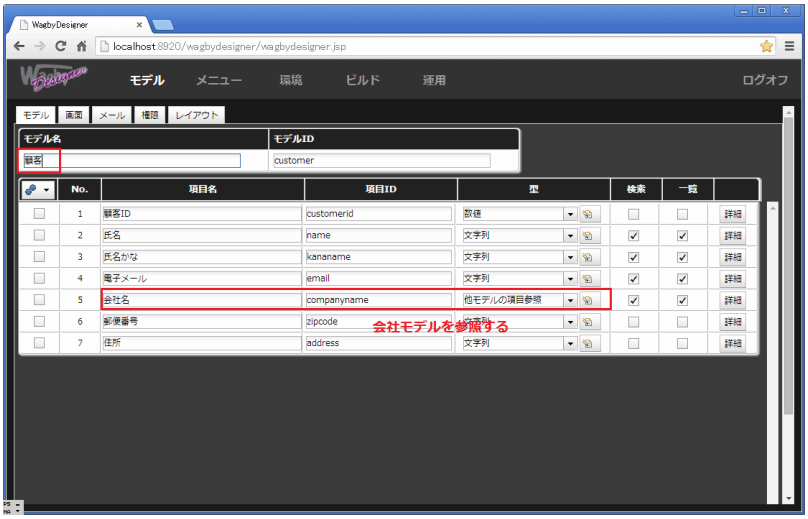


図 修正された顧客モデル

実行

見積書における「会社名」は（文字列型から）モデル参照 － 参照連動 － となっています。このとき「この会社で検索」ボタンを押下すると、会社名コード（ID部）を渡しています。

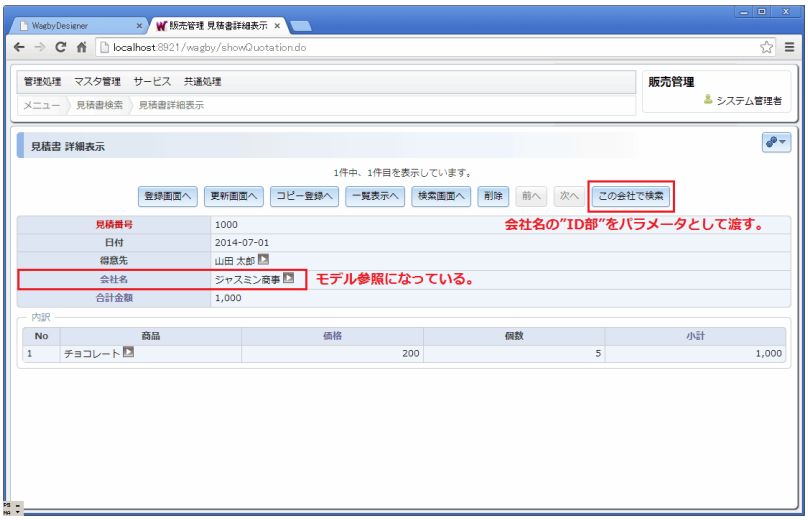


図 修正された見積書モデルの詳細画面

モデル参照についても、パラメータが渡っていることが確認できます。

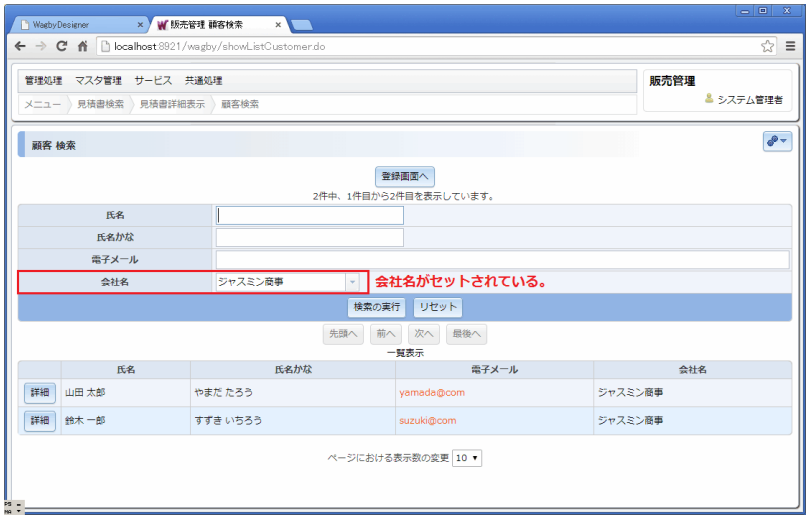


図 ボタン押下後の顧客検索画面

設定

見積書モデル

見積書モデルにおける「会社名」は参照連動です。参照連動の場合は標準で「内容部」を渡すようになりますが、今回は引数に「ID部」を渡すことが求められています。（顧客モデルの「会社名」がモデル参照となっているためです。）

そこで図の見積書モデルの「追加パラメータ」を次のように修正します。"#id" を付与するとIDを渡すことができます。

#id表記の詳細は"計算式>モデル参照"をお読みください。

```
companyname=${companyname#id}
```

図 追加パラメータの修正

顧客モデル

検索条件の初期値にコード(ID部)を設定するため、TOINT関数を適用します。

```
TOINT(REQUEST())
```

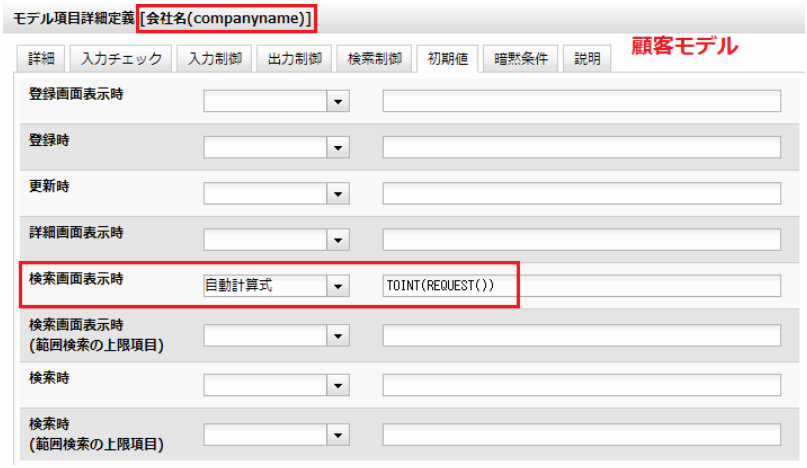


図 初期値の式の変更(TOINT関数の適用)

検索画面から検索画面への遷移

model2 の検索画面から model1 の検索画面へ遷移する例を説明します。ここで item1 は文字列型、item2 は数値型で範囲検索を有効としています。

model2の定義

model2 に用意したボタンの「追加パラメータ」は次のとおりです。

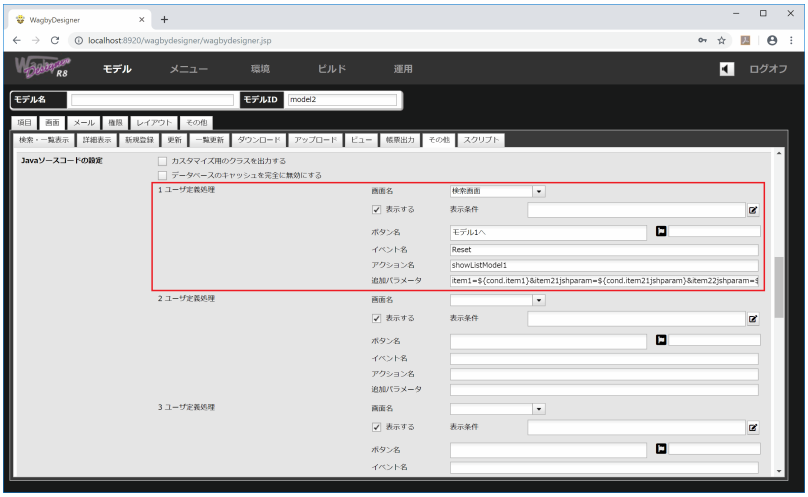


図 model2に用意したボタンの定義

```
item1=${cond.item1}&item21jshparam=${cond.item21jshparam}&item22jshparam=${cond.item22jshparam}
```

- ${ と } で囲んだ（model2の）項目名について、検索条件を示すcond.を付与します。詳細...
- 範囲検索の場合、さらに項目名に "1jshparam" と "2jshparam" を付与します。それぞれ開始と終了を意味します。

model1の定義

item1の初期値

文字列型の項目 item1 の式を示します。上図の11で、同じ名前item1を使っているため、REQUEST関数の引数を省略しています。

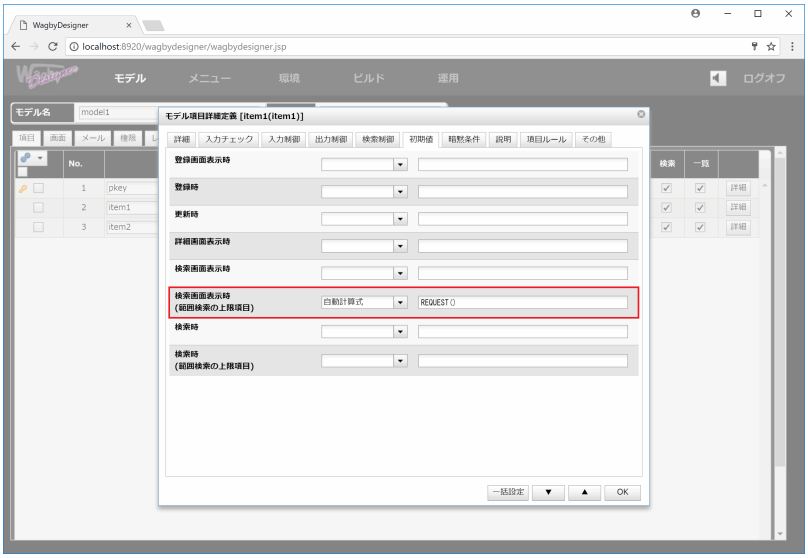


図 受け側となるmodel1のitem1項目（文字列型）

タイミング	式の種類	式
検索画面表示時	自動計算式	REQUEST()

item2の初期値

数値型かつ範囲検索を有効としている項目 item2 の式を示します。数値型のため TOINT で囲んでいます。開始部分は item21jshparam となります。終了部分は item22jshparam となります。

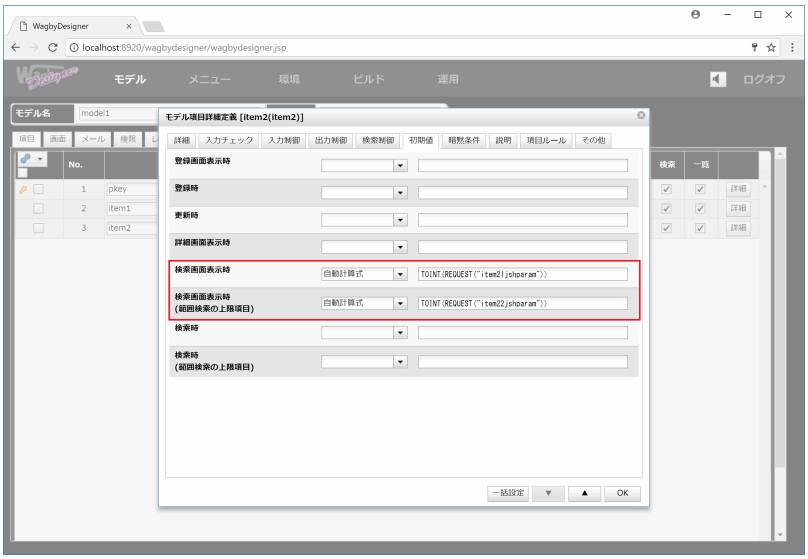


図 受け側となるmodel1のitem2項目（数値型）

タイミング	式の種類	式
検索画面表示時	自動計算式	TOINT(REQUEST("item21jshparam"))
検索画面表示時(範囲検索の上限項目)	自動計算式	TOINT(REQUEST("item22jshparam"))

運用時の注意点

本設定を利用する場合、必ず model2 の検索処理を行なうようにしてください。（一度も検索処理を行わない場合、パラメータを渡すことはできません。）

仕様・制約

〝画面遷移（1）詳細画面から登録画面へ遷移する 〉仕様・制約〟をお読みください。

画面遷移（3）一覧表示画面から別のモデルへ遷移する

一覧表示画面から詳細画面への遷移

顧客モデルの一覧表示画面から、サポートモデルの詳細画面へ遷移するボタンを用意した例を図に示します。

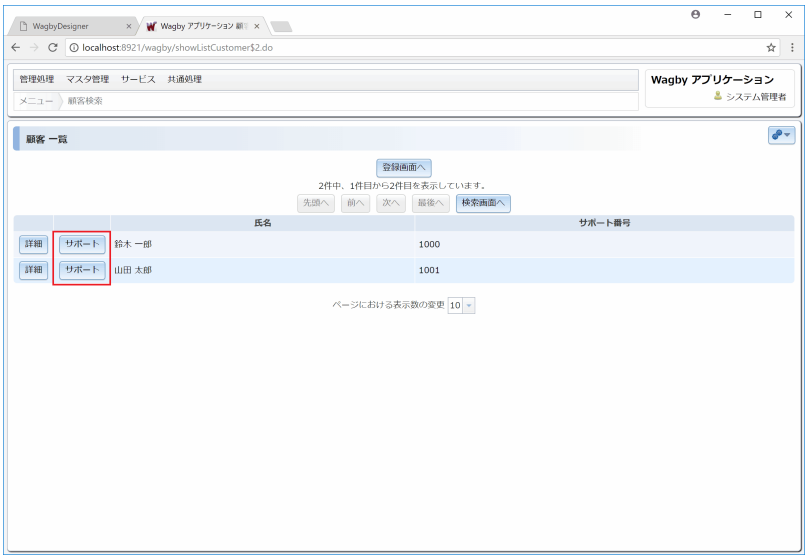


図 サポートモデルの詳細画面へ遷移するボタンを用意した例

このボタンを押すと、（顧客モデルが保持している）サポートモデルへの主キーを用いて、サポートモデルの詳細画面を開きます。

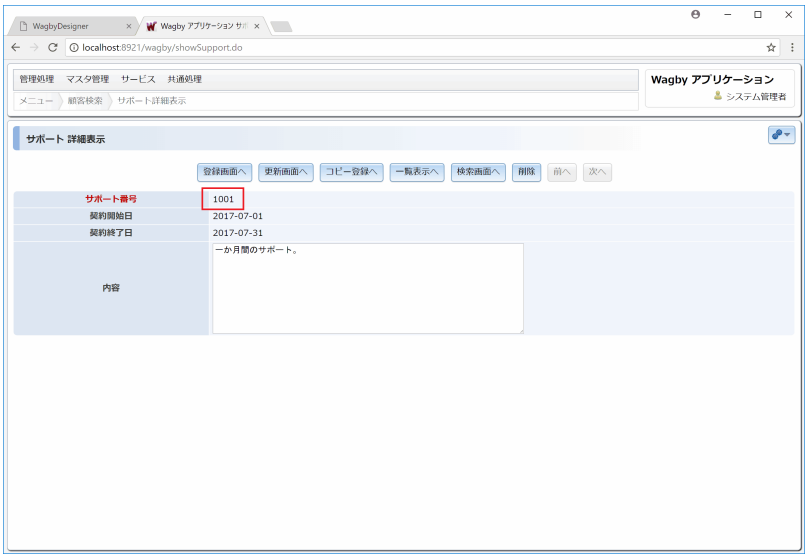


図 サポートモデルの詳細表示

モデル定義

用意したモデルは「顧客」と「サポート」の二つです。

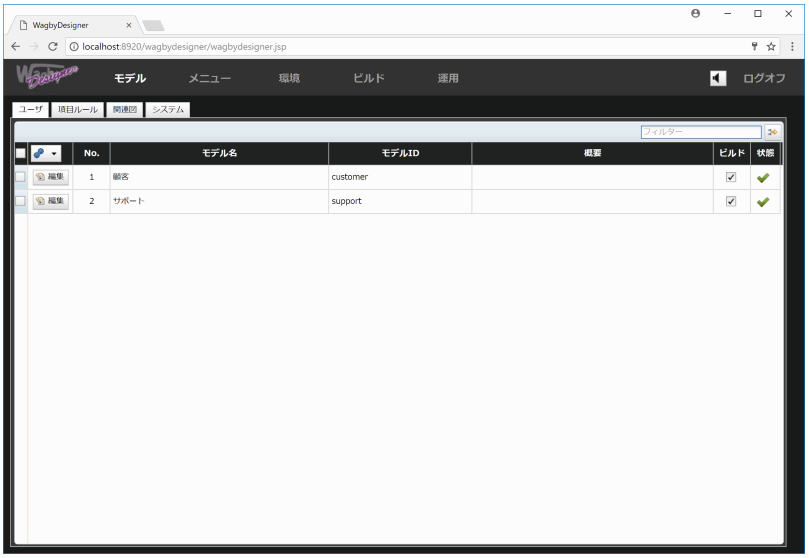


図 顧客モデルとサポートモデル

顧客モデルに、サポートモデルの主キーを保持する項目「サポート番号」を用意します。

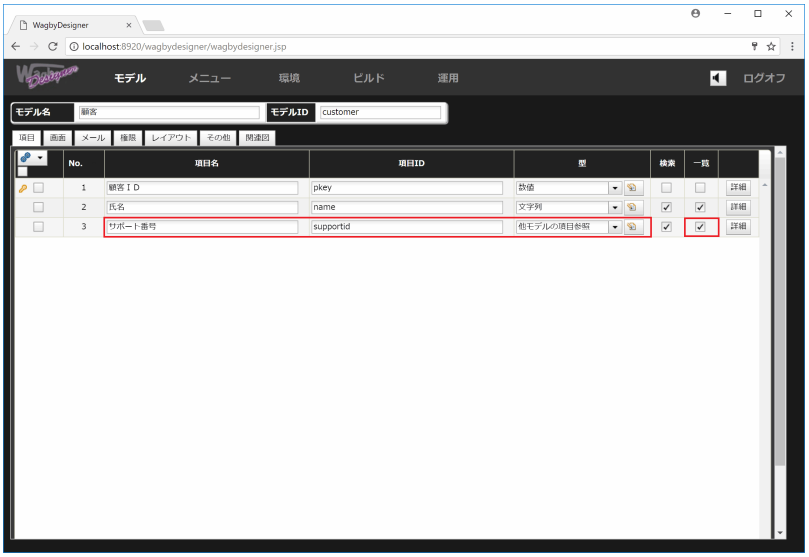


図 顧客モデルに「サポート番号」項目を用意する

ボタンを用意する

図で示した「顧客モデル」で、「画面>その他>Javaソースコードの設定」を開きます。 ここで追加ボタンを次のように定義します。

項目	説明	入力例
画面名	どの画面にボタンを追加するかを指定します。ここでは"一覧表示画面(行毎)"とします。	一覧表示画面(行毎)
表示する	有効にします。	(チェック)
表示条件	常にボタンを表示する場合、本欄は空白とします。条件については"ボタン表示条件"をお読みください。	
ボタン名	任意の名前を指定します。式で表現することもできます。	サポート
イベント名	遷移した画面に対するイベントを指定します。遷移先は詳	Show

	細画面ですので、詳細画面で受理できるイベント Show を指定します。	
アクション名	遷移する画面を指定します。[後述]	showSupport
追加パラメータ	遷移する画面へパラメータを指定することができます。[後述]	supportid=${supportid#id}

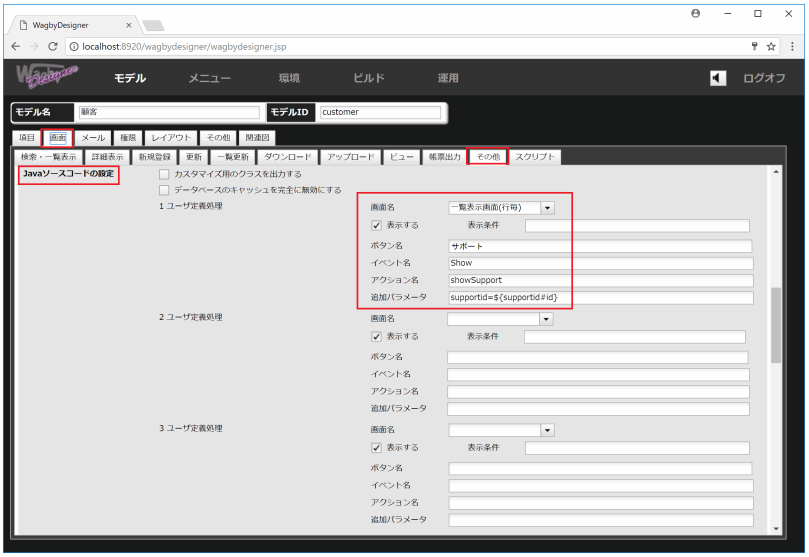


図 追加ボタンの設定

(239)

アクション名

モデルと画面の組み合わせから構成される文字列です。アドレスバー(URL)の一部になります。

詳細画面は次のようになります。

```
"show" + モデル名(キャメル記法)
```

このルールにより、モデル "support" の詳細画面に対応するアクション名は "showSupport" になります。拡張子は不要です。

(240)

追加パラメータ

詳細画面に対して、直接、主キーを渡すことができます。

書式は「キー=値」となっています。キー部分は（遷移先モデルの）項目IDになります。　値部分にプレースホルダ　(${　...　})　を用いて、遷移元の項目の値を渡すことができます。遷移元項目がモデル参照の場合、#id を付与することで ID 部を渡すようになります。

#id表記の詳細は"計算式>モデル参照"をお読みください。

図の動作を実現するための追加パラメータは次のようになります。イコールの左側はサポートモデルの項目で、主キーになります。イコールの右側は顧客モデルの項目です。たまたま同じ名前(supportid)ですが、両者はそれぞれモデルが異なります。

```
supportid=${supportid#id}
```

(239) イベント名は大文字小文字を区別します。正確に入力してください。

(240) キャメル記法 : 先頭の1文字を大文字とします。モデル名に含まれるアンダースコアは取り除き、その次の文字を大文字にします。例えば my_funny_valentine は MyFunnyValentine になります。

受ける側の初期値設定

受側のモデル（この例ではサポートモデル）の主キー項目に初期値を設定する必要はありません。主キー項目は、外部パラメータがあればそれを自動的に利用するようになっています。

制約

追加パラメータに記載する項目名に以下を指定することはできません。

- 繰り返し項目
- 繰り返しコンテナ項目
- このモデルに紐づく外部キー子モデル

一覧表示画面から更新画面への遷移

顧客モデルの一覧表示画面に二つの独自ボタン「サポート表示」と「サポート更新」を用意した例を紹介します。

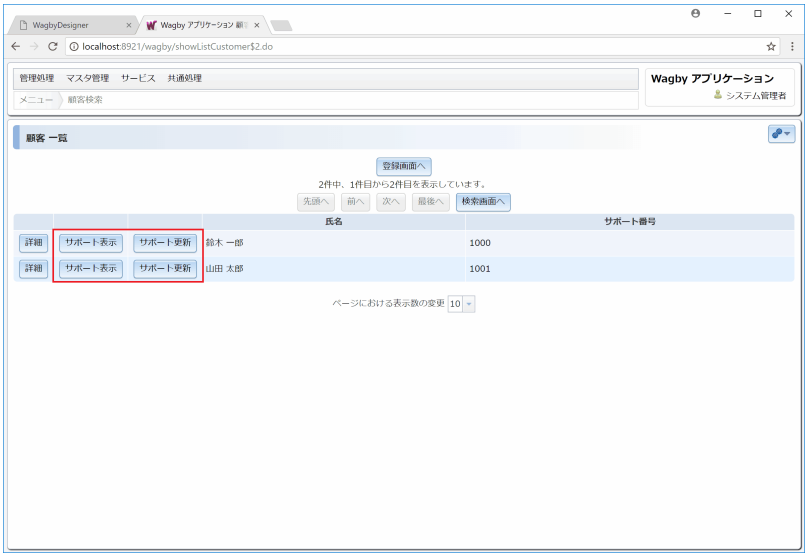


図 二つの独自ボタンを用意する

「サポート更新」ボタン押下により、サポートモデルの更新画面を開くことができます。

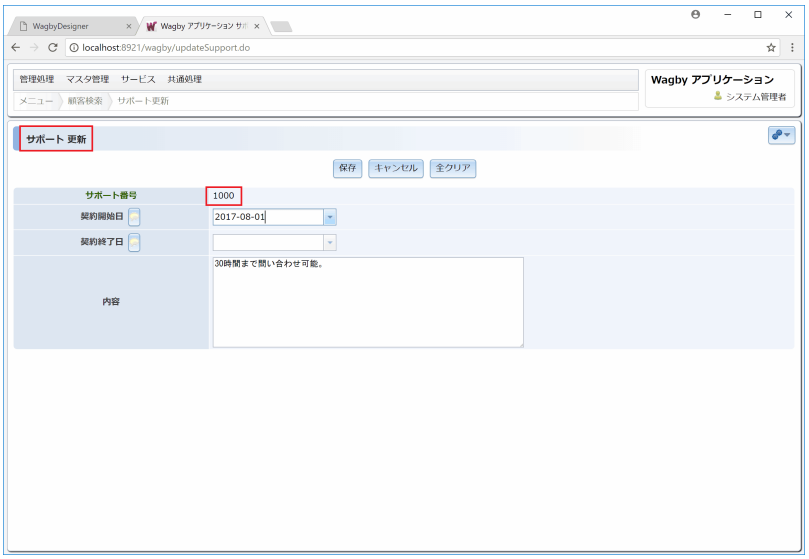


図 サポートモデルの更新画面

ここで更新を終了またはキャンセルしたときは、サポートモデルの詳細画面へ遷移します。遷移元である顧客モデルの一覧表示画面へ戻る場合は、パンくずを利用します。

設定

二つ目の独自ボタンとして、更新用のボタンを設定します。

項目	説明	入力例
画面名	どの画面にボタンを追加するかを指定します。"一覧表示画面(行毎)"とします。	一覧表示画面(行毎)
表示する	有効にします。	(チェック)
表示条件	常にボタンを表示する場合、本欄は空白とします。条件については"ボタン表示条件"をお読みください。	
ボタン名	任意の名前を指定します。式で表現することもできます。	サポート更新
イベント名	遷移した画面に対するイベントを指定します。遷移先は更新画面ですので、更新画面で受理できるイベント Update を指定します。これにより更新画面が開きます。	Update
アクション名	遷移する画面を指定します。	updateSupport
追加パラメータ	遷移する画面へパラメータを指定します。	supportid=${supportid#id}

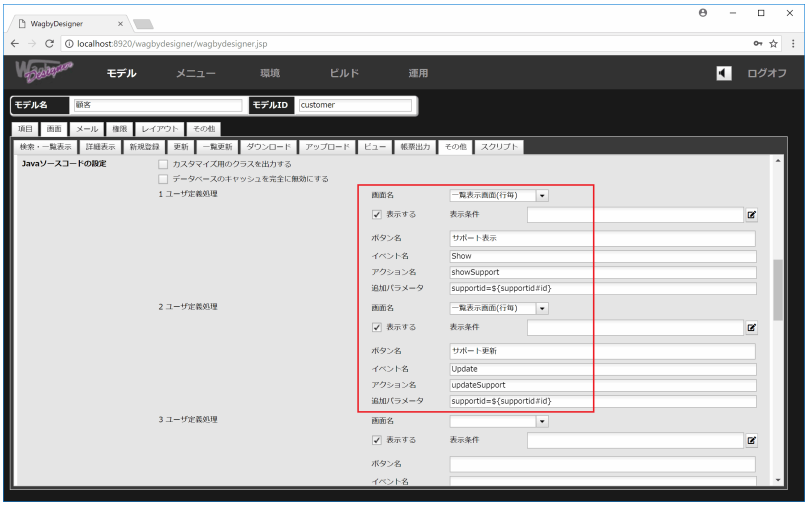


図 更新用のボタン

一覧表示画面から一覧更新画面への遷移

一覧更新画面へ遷移するボタンを設定します。なお、対象モデルは一覧更新機能が有効になっている必要があります。

項目	説明	入力例
画面名	どの画面にボタンを追加するかを指定します。"一覧表示画面(行毎)"とします。	一覧表示画面(行毎)
表示する	有効にします。	(チェック)
表示条件	常にボタンを表示する場合、本欄は空白とします。条件に	

	ついては"ボタン表示条件"をお読みください。	
ボタン名	任意の名前を指定します。式で表現することもできます。	削除
イベント名	空白にします。	
アクション名	一覧更新画面を指定します。	updateListCustomer
追加パラメータ	遷移する画面へパラメータを指定します。検索条件の初期値としてこのパラメータを受け取ることで、この値で絞り込んだ状態の一覧更新画面を開くことができます。	

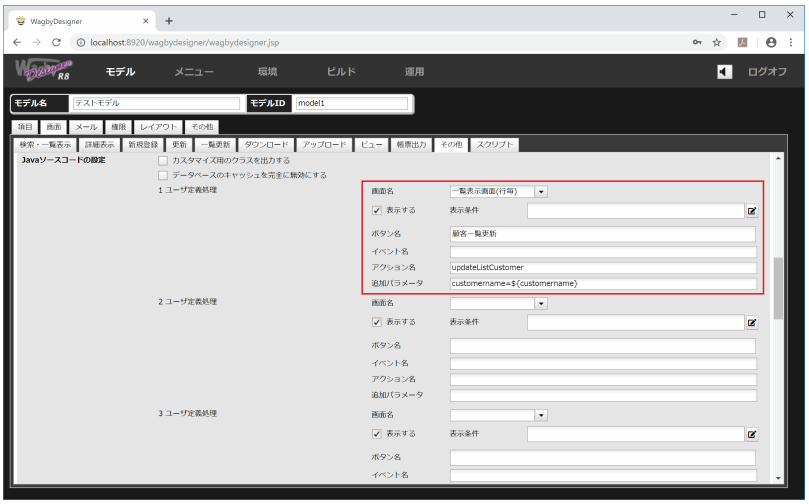


図 一覧更新画面へ遷移するためのボタンを用意する

遷移先のモデルでは、検索条件の初期値として REQUEST 関数を指定します。

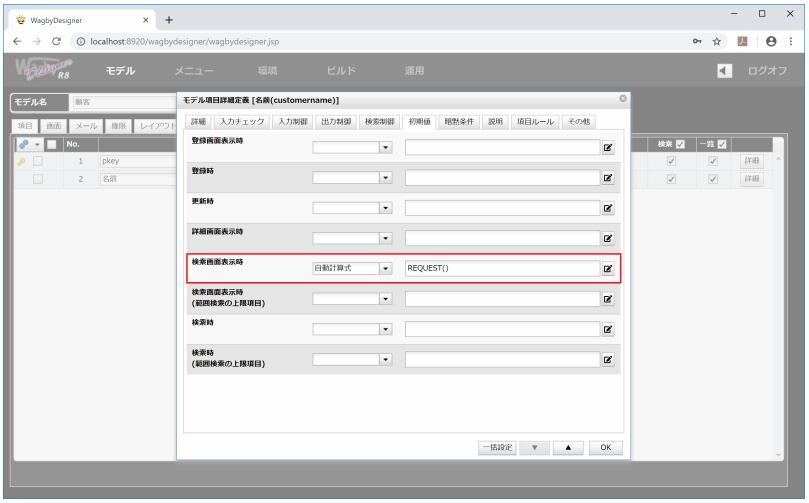


図 遷移先モデルの設定（検索条件の初期値）

削除

削除ボタンを追加した例を示します。

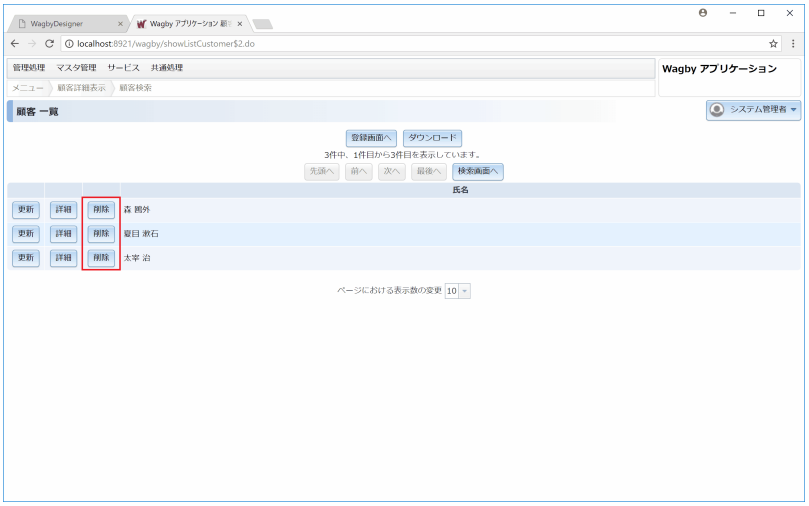


図 削除ボタン(1)

削除ボタンを押下すると、データが削除され、一覧表示画面が再描画されます。

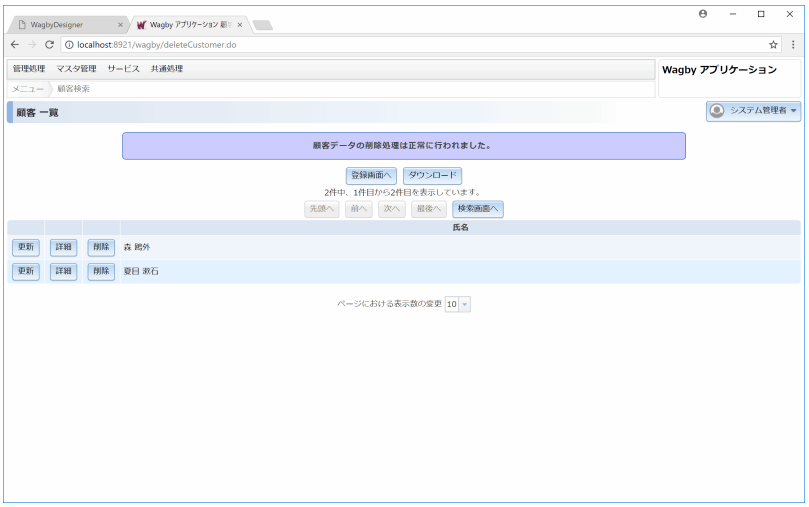


図 削除ボタン(2)

設定

削除用のボタンを設定します。なお、対象モデルは削除機能が有効になっている必要があります。

項目	説明	入力例
画面名	どの画面にボタンを追加するかを指定します。"一覧表示画面(行毎)"とします。	一覧表示画面(行毎)
表示する	有効にします。	(チェック)
表示条件	常にボタンを表示する場合、本欄は空白とします。条件については"ボタン表示条件"をお読みください。	
ボタン名	任意の名前を指定します。式で表現することもできます。	削除
イベント名	遷移先である削除画面が受理できるイベント Send を指定します。これは削除処理を行うものです。	Send
アクション名	削除画面を指定します。(実際にはこの画面は画面をもた	deleteCustomer

	ず、削除機能のみを提供するものです。）	
追加パラメータ	遷移する画面へパラメータを指定します。標準で（このモデルの）主キーが渡されるため、記述は不要です。	

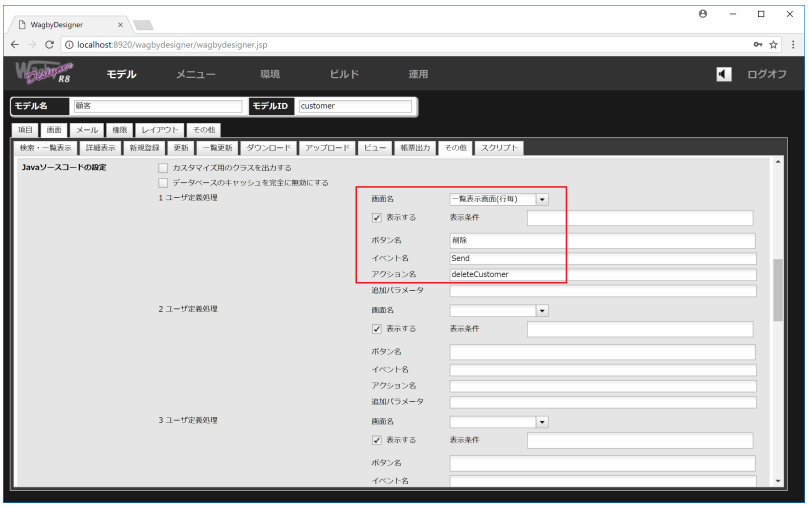


図 削除ボタンを用意する

印刷

印刷ボタンを追加した例を示します。ボタンを押下すると Excel 帳票が実行されます。（ファイルがダウンロードされています。）

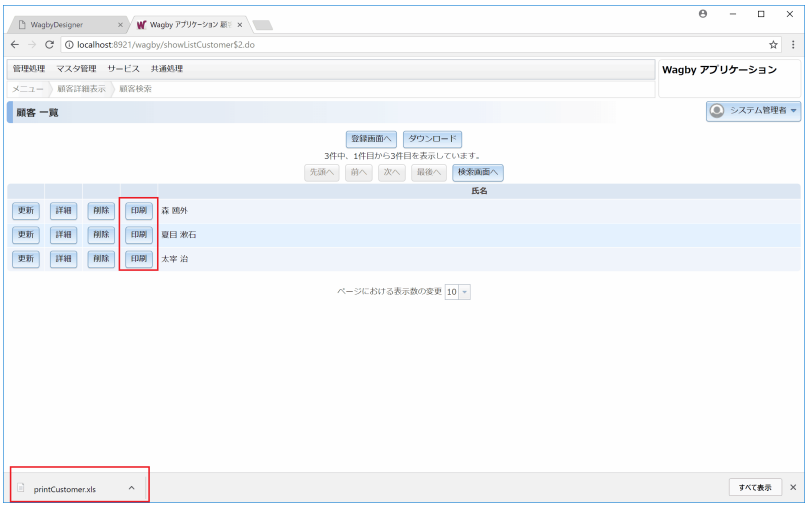


図 印刷ボタン

設定

削除用のボタンを設定します。なお、対象モデルは詳細画面の印刷機能が有効になっているものとします。

項目	説明	入力例
画面名	どの画面にボタンを追加するかを指定します。"一覧表示画面(行毎)"とします。	一覧表示画面(行毎)
表示する	有効にします。	（チェック）

表示条件	常にボタンを表示する場合、本欄は空白とします。条件については"ボタン表示条件"をお読みください。	
ボタン名	任意の名前を指定します。式で表現することもできます。	印刷
イベント名	遷移先である詳細画面が受理できるイベント Print を指定します。これは印刷処理を行うものです。	Print
アクション名	詳細画面を指定します。	showCustomer
追加パラメータ	遷移する画面へパラメータを指定します。標準で（このモデルの）主キーが渡されるため、記述は不要です。	

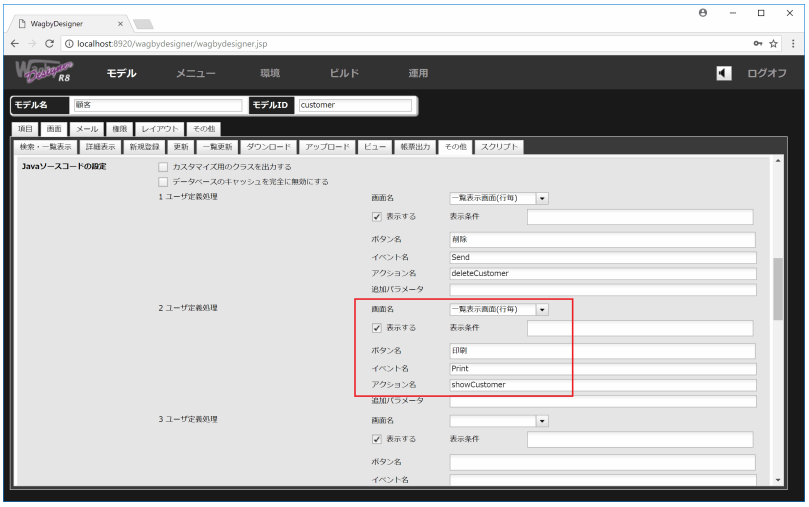


図 帳票ボタンを用意する

印刷ボタンを複数用意する（8.3.1）

一つのモデルの詳細表示用帳票に、複数の帳票パターンを定義する場合があります。

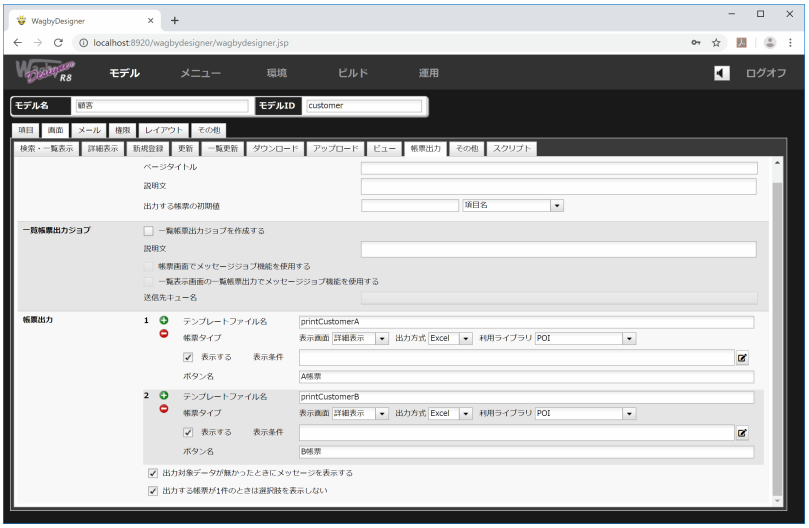


図 複数の帳票パターン

この場合、アクション名は同じ（画面）とし、イベント名を次のように指定してください。テンプレートファイル名部分で、どの帳票パターンかを区分します。

Print<モデルID>_<テンプレートファイル名>

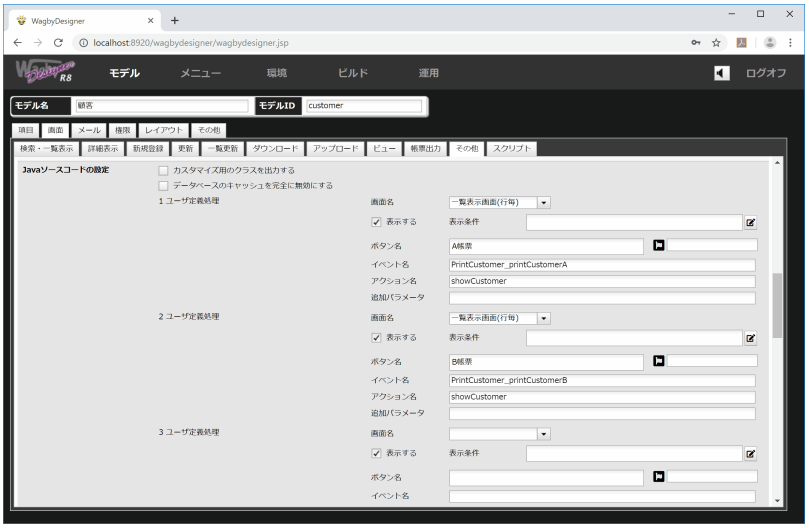


図 複数の帳票用ボタン

パラメータに検索条件の値を用いる（8.2.0）

一覧表示画面に用意する独自ボタンの追加パラメータに、検索条件部の値を渡すことができます。 特別表記 ${cond.項目名} を指定します。例えば次のように表記します。

```
customername=${cond.name}&ourdept=${cond.dept#id}
```

仕様・制約

- 「検索画面と一覧表示画面を同時に表示する」設定を有効にしてください。
- 一度検索を行わないと値はセットされません。

仕様・制約

- この設定は親子モデル同時表示の親画面における子モデル一覧表示には反映されません。
- その他の制約については ″画面遷移（1）詳細画面から登録画面へ遷移する 〉 仕様・制約″をお読みください。

画面遷移（4）詳細画面から更新画面へ遷移する（更新データがない場合、登録画面へ自動的に遷移する）

例

顧客モデルに、独自ボタンを用意した例です。 このボタンは別モデル（サポートモデル）の更新画面を直接、開くようになっています。

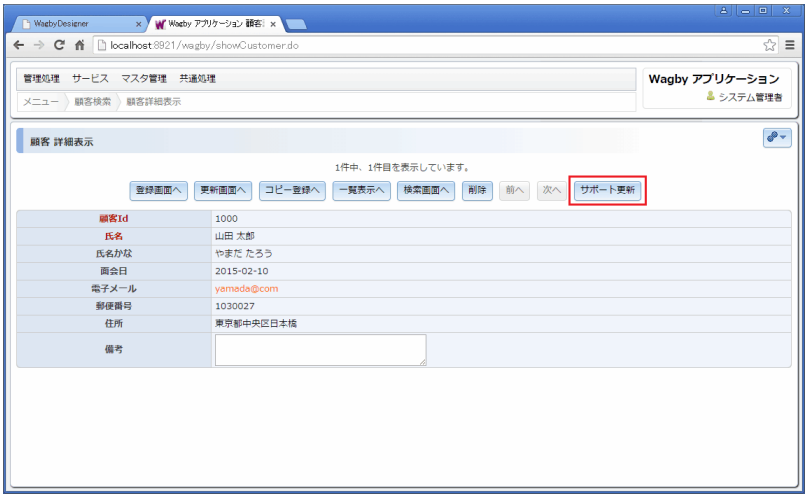


図 顧客モデルからサポートモデルの更新画面を開く

ここで、顧客モデルとサポートモデルには、モデル参照の関係は定義していません。ただし顧客モデルの主キーとサポートモデルの主キーは 1:1 の関係性がある、とします。

最初に更新ボタンを押下したとき、（この顧客に紐づく）サポートは存在しません。このとき、自動的に更新画面から登録画面へ遷移しています。さらに、登録画面には顧客モデルのいくつかの値が引き継がれています。

図 データ未登録の場合は新規画面へ遷移する

サポートデータの新規登録が完了しました。

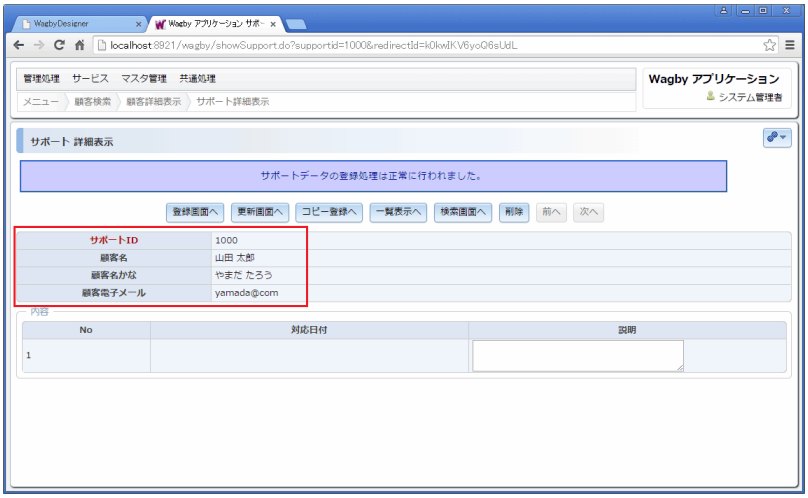


図 サポートデータの新規登録が完了

図に戻ります。再び、同じ顧客データの詳細画面から「サポート更新」ボタンを押下すると、今度は紐づくデータが存在するため、更新画面が開きます。つまりデータが存在する場合は、更新処理を行えるようになっています。

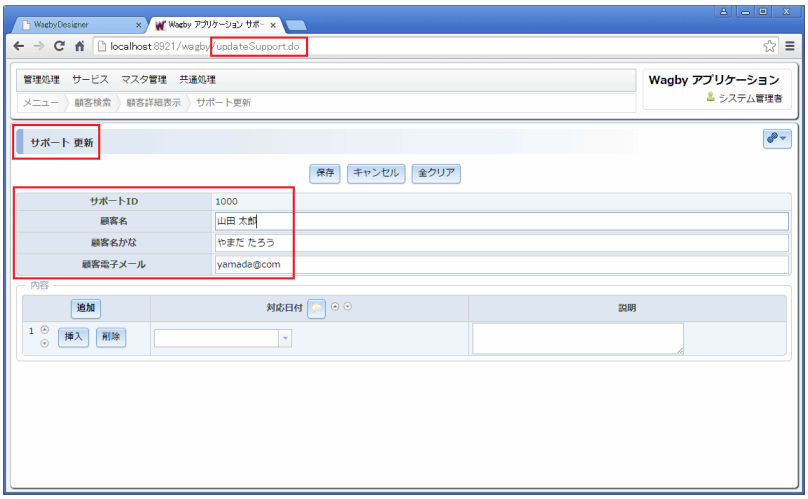


図 データ登録後は更新画面へ遷移する

定義方法

図で示した「顧客モデル」の定義は次のとおりです。

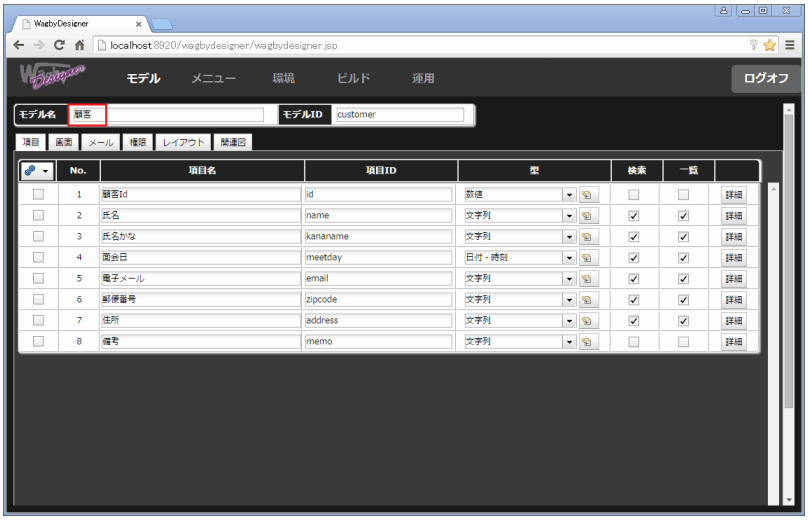


図 顧客モデル

対応する「サポートモデル」の定義は次のとおりです。顧客モデルとの外部キー関係は定義していません。「顧客名」「顧客名かな」「顧客電子メール」は参照ではなく、文字列型としています。

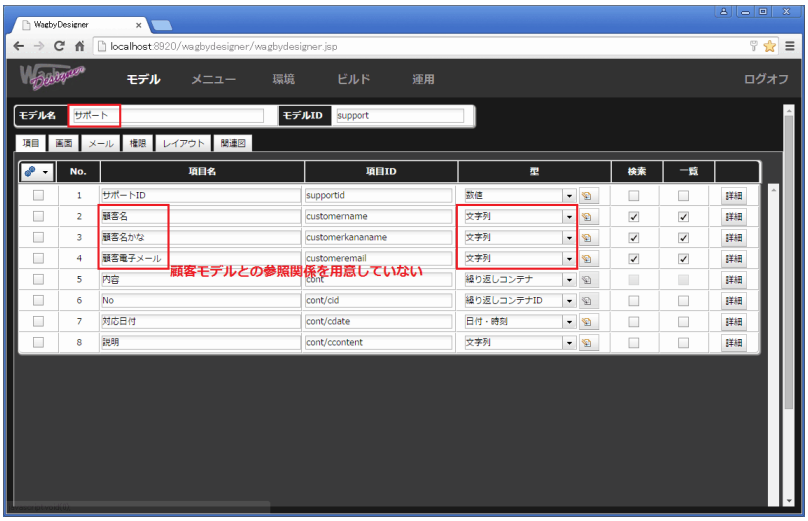


図 サポートモデルの定義

顧客モデルに用意する独自ボタン

顧客モデル側に、サポートモデルの更新画面を開くための独自ボタンを用意します。

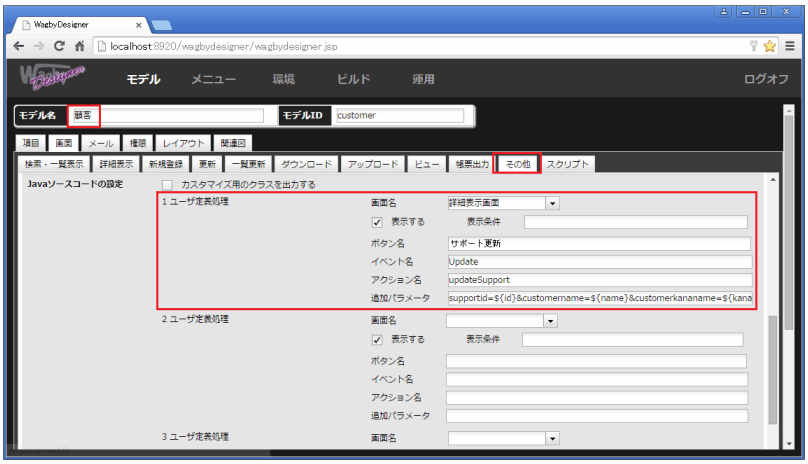


図 独自ボタンの定義

項目	説明	入力例
画面名	どの画面にボタンを追加するかを指定します。ここでは"詳細表示画面"とします。	詳細表示画面
表示する	有効にします。	（チェック）
表示条件	常にボタンを表示する場合、本欄は空白とします。条件については「ボタン表示条件」をお読みください。	
ボタン名	任意の名前を指定します。式で表現することもできます。	サポート更新
イベント名	遷移した画面に対するイベントを指定します。更新イベントは "Update" です。	Update
アクション名	遷移する画面を指定します。更新画面は "update" + モデル名（キャメル記法）です。	updateSupport
追加パラメータ	遷移する画面へパラメータを指定します。ここでは紐づくサポートモデルの主キーと、顧客モデルの名前などのデータを渡します。	（後述）

(241) (242)

追加パラメータ

更新画面（または新規登録画面）に対して、直接、値を渡すことができます。

書式は「キー=値」となっています。キー部分は（遷移先モデルの）項目IDになります。　値部分にプレースホルダ（${ ... }）を用いて、遷移元の項目の値を渡すことができます。

複数のパラメータを渡す場合は&で区切ります。

ここで示した例を実現するための追加パラメータは次のようになります。イコールの左側はサポートモデルの項目です。イコールの右側は顧客モデルの項目です。

```
supportid=${id}&customername=${name}&customerkananame=${kananame}&customeremail=${email}
```

今回の例では、通常用意される更新ボタンではなく、オリジナルボタンを使っています。これはボタンを配置する画面内の値を、更新処理を行うモデル（の登録・更新画面）に引き継ぐためです。

制約

追加パラメータに記載する項目名に以下を指定することはできません。

- 繰り返し項目
- 繰り返しコンテナ項目
- このモデルに紐づく外部キー子モデル

サポートモデルの設定

「画面 > 更新 > 更新画面」にある「更新対象データが存在しないとき登録画面に遷移する」を有効にします。

(241) イベント名は大文字小文字を区別します。正確に入力してください。

(242) キャメル記法 : 先頭の1文字を大文字とします。モデル名に含まれるアンダースコアは取り除き、その次の文字を大文字にします。例えば my_funny_valentine は MyFunnyValentine になります。

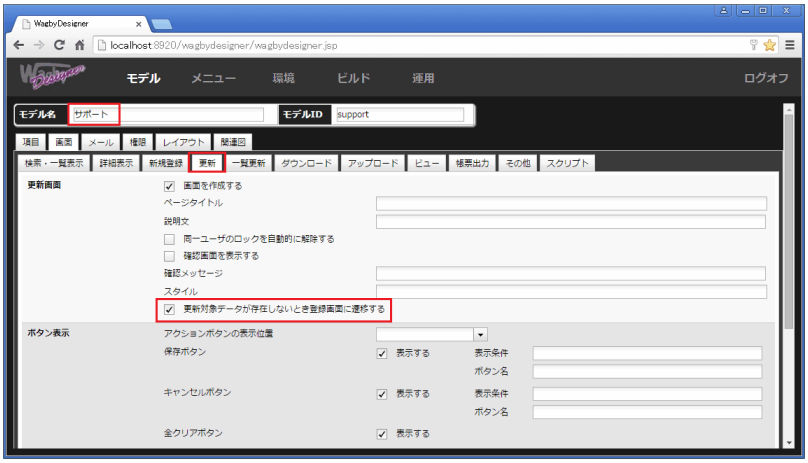


図 更新対象データが存在しないとき登録画面に遷移する

顧客名

顧客モデルから渡されてきた初期値を受け取るためにREQUEST関数を適用します。

```
REQUEST()
```

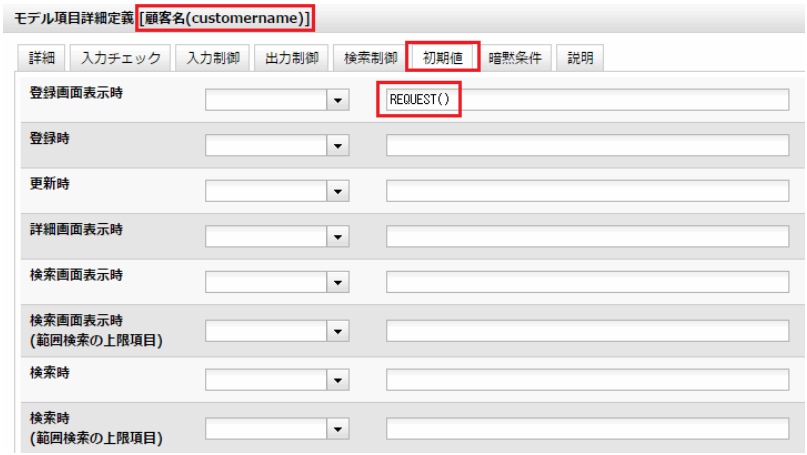


図 顧客名の初期値を外部から取得する

顧客名かな

同じくREQUEST関数を適用します。

図 顧客名かなの初期値を外部から取得する

顧客電子メール

同じくREQUEST関数を適用します。

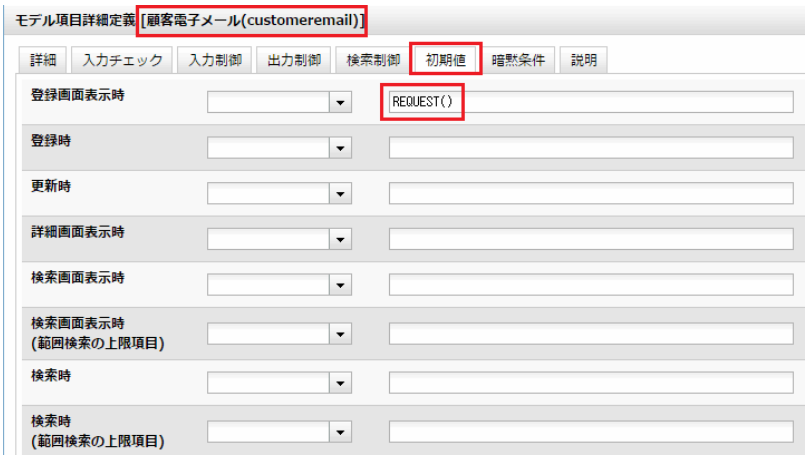


図 顧客電子メールの初期値を外部から取得する

(243)

仕様・制約

「画面遷移（1）詳細画面から登録画面へ遷移する ＞ 仕様・制約」をお読みください。

画面遷移（5）メインモデルの更新画面からサブモデルの更新画面へ遷移する(スクリプト)

例

顧客モデルに、独自ボタンを用意した例です。 このボタンは自身のサブモデル（顧客サブモデル）の更新画面を直接、開くようになっています。

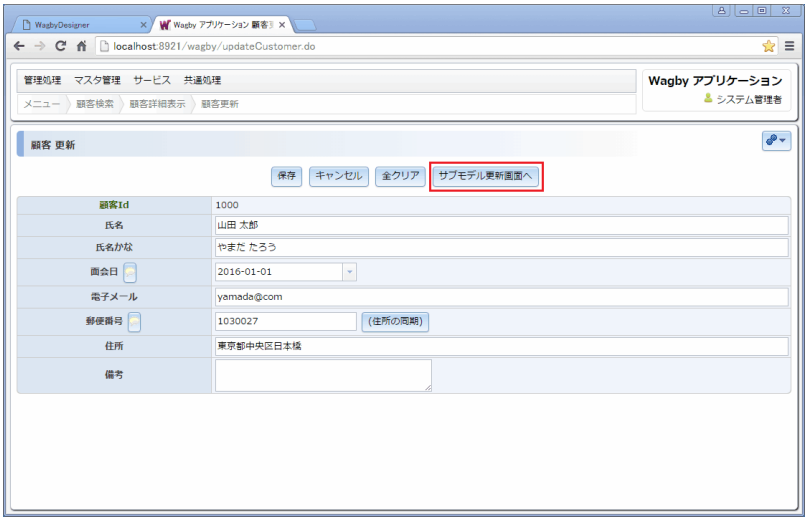


図 顧客モデルからサブモデルの更新画面を開く

(243) 項目名と違うパラメータの値を反映させる場合は REQUEST("パラメータ名") のように記述します。

顧客モデルと顧客サブモデルの主キーは同一です。すなわち 1:1 の関係性があります。

「サブモデル更新画面へ」ボタンを押下すると、顧客サブモデルの更新画面に遷移します。ここではサブモデルを使ったため、データは同じものになっています。

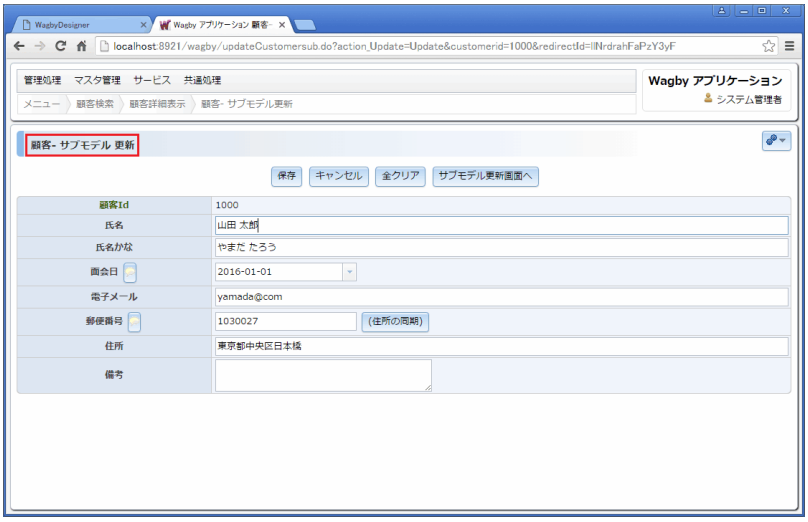


図 メインモデルの更新画面から、サブモデルの更新画面へ遷移した

定義方法

顧客モデルに用意する独自ボタン

顧客モデル側に、顧客サブの更新画面を開くための独自ボタンを用意します。

図 独自ボタンの定義

項目	説明	入力例
画面名	どの画面にボタンを追加するかを指定します。ここでは"更新画面"とします。	更新画面
表示する	有効にします。	（チェック）
表示条件	常にボタンを表示する場合、本欄は空白とします。条件については"ボタン表示条件"をお読みください。	

ボタン名	任意の名前を指定します。式で表現することもできます。	サブモデル更新
イベント名	遷移した画面に対するイベントを指定します。今回は処理内容をスクリプトで記述するため、独自イベントを指定します。	UpdateSubmodel
アクション名	登録、更新画面を起点とした場合、アクション名を記述することはできません。空白としてください。	
追加パラメータ	遷移する画面へパラメータを指定します。今回は省略しています。	(後述)

(244)

スクリプト

登録・更新画面を起点とした場合、遷移先の画面（上の表にある「アクション名」）を記述することができません。そこで、スクリプトを使って直接、画面遷移制御を行います。

ビルドしたアプリケーション wagbyapp/webapps/wagby/WEB-INF/script/モデルID フォルダに、テキストエディタを使って新規ファイルを作成します。ファイル名は次のようにします。

```
UpdateモデルID_イベント名.js
```

モデルIDは、キャメル記法を使います。今回の例では、次のようになります。

```
UpdateCustomer_UpdateSubmodel.js
```

(245)

スクリプトの内容は次の通りです。

```
function process() {
var ExcelFunction = Java.type("jp.jasminesoft.util.ExcelFunction");
var Jfcerror = Java.type("jp.jasminesoft.jfc.error.Jfcerror");
var Jfcwarn = Java.type("jp.jasminesoft.jfc.error.Jfcwarn");
var Jfcinfo = Java.type("jp.jasminesoft.jfc.error.Jfcinfo");

var customerid = p.request.getParameter("customer_p$002fcustomerid");
return "redirect:updateCustomersub.do?action_Update=Update&customerid="+customerid;
}
```

return で画面遷移先を指定します。

(246)

パラメータ

上のスクリプトでは、Wagbyが更新画面用に自動生成したパラメータ "customer_p$002fcustomerid" を再利用しています。

パラメータ名は、自動生成された JSP ファイルで確認し、コピー&ペーストするとよいでしょう。

wagbyapp/webapps/プロジェクト名/モデルID/ フォルダの中に「update + "モデルID" + P.jsp」というファイルがあります。これをテキストエディタで開くと、次のようなコードが見つかります。

```
<span id="content_customer_p$002fcustomerid" name="customer_p$002fcustomerid">
```

この id="..." の部分が、パラメータ名になります。

---

(244) イベント名は大文字小文字を区別します。

(245) キャメル記法 : 先頭の1文字を大文字とします。モデル名に含まれるアンダースコアは取り除き、その次の文字を大文字にします。例えば my_funny_valentine は MyFunnyValentine になります。

(246) このスクリプトは Wagby Designer から直接、記述していません。そのため process 関数の宣言も含めてください。

自動生成されたパラメータではなく「追加パラメータ」を用意することもできます。上の例では customerid=${customerid} と指定すると、スクリプトでは p.request.getParameter("customerid") で値を取得できます。

仕様・制約

メインモデル・サブモデル関係に限定している理由

　このページで説明した例は、メイン-サブモデル間の遷移かつ、同じ主キーのデータと限定しています。そうでない場合、遷移元モデルのロックが（遷移先に移動したときに）解放されないという問題が発生します。

遷移先を記述しない理由

　登録、コピー登録、更新画面を起点とした場合、遷移先（「アクション名」）を直接、記述することができません。本ページで紹介したように、スクリプトと併用してください。

(247)

その他の仕様・制約

　その他の仕様・制約については"画面遷移 (1) 詳細画面から登録画面へ遷移する > 仕様・制約"をお読みください。

画面遷移 (6) スクリプトを使った画面遷移(スクリプト)

スクリプトの作成方法

「画面 > スクリプト > コントローラ」を設定します。

はじめに、どの画面についてのスクリプトを記述するかを指定します。

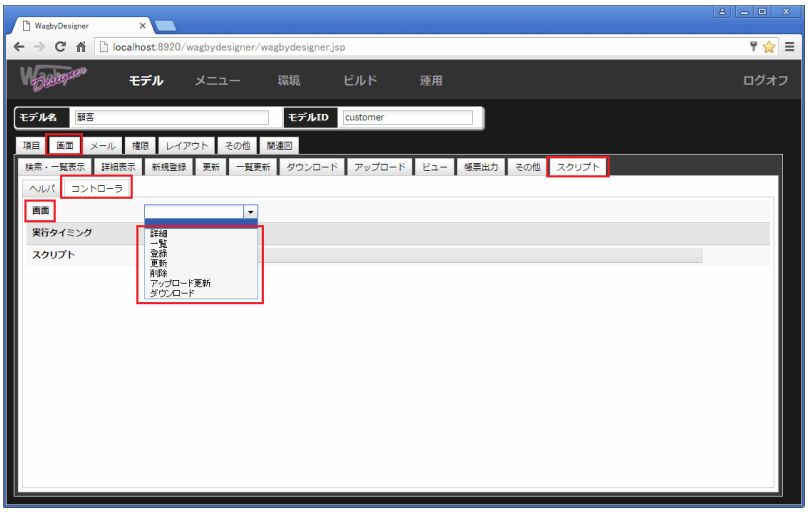


図 スクリプトの入力-画面の指定

　画面ごとに「実行タイミング」が用意されています。スクリプトは、指定したタイミングが終了したあとに呼び出されます。

(247) 遷移先（「アクション名」）を記載すると、ボタンが form 要素で囲まれるようになります。しかし、もともと登録・更新画面が form 要素で囲まれているため、form 要素の中に form 要素が含まれる形となるため、動作しなくなります。

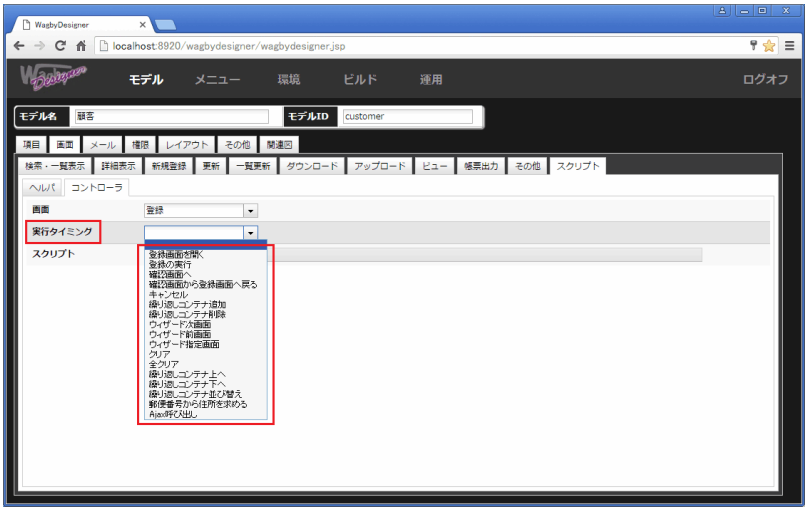


図 スクリプトの入力-実行タイミングの指定

呼び出されるタイミング

ヘルパの「登録時」とコントーラの「登録時」の違い

ヘルパの登録時は XXXHelper#beforeInsert メソッドがスクリプトを呼び出します。これはデータベースに登録する「直前」に呼び出されるものです。

一方、コントローラの登録処理は、データベース登録処理などがすべて完了し、画面表示（JSP）に制御が移る直前に呼び出されます。

この違いから、「登録時にデータを加筆・修正する」といった業務ロジックはヘルパの「登録時」に記述してください。この場合は、修正した値がデータベースに保存されます。

スクリプトの記述ルール

Wagby標準の画面遷移ルールを変更する

コントローラのスクリプトを作成する目的は、Wagby標準の画面遷移ルールを変更することです。 具体的には次のように記述します。

```
return "redirect:" + 遷移先の画面種別 + モデルID + イベント名(アクション文字列形式)
return "forward:" + 遷移先の画面種別 + モデルID + イベント名(アクション文字列形式)
```

接頭語 "redirect:" または "forward:" が必要です。両者の違いは「リダイレクトとフォワード」の節で説明します。標準は "redirect:"を使うようにしてください。画面種別はSCREENTYPE関数の仕様に準じます。モデルIDは先頭を大文字とします。（例:customerモデルの場合、Customerとなる。）アクション文字列は "action_" という接頭語のあとにイベント名を付与します。具体的には次節以降のサンプルで確認してください。

画面遷移はそのままに、追加のコードを記述する

画面遷移に変更がない場合は、return文を省略できます。追加コードのみ記述してください。

[例] 登録画面から、再び登録画面へ遷移する

連続した登録処理を行うため、登録画面終了後、再び登録画面に遷移するようにした例です。

```
return "redirect:insertCustomer.do?action_New";
```

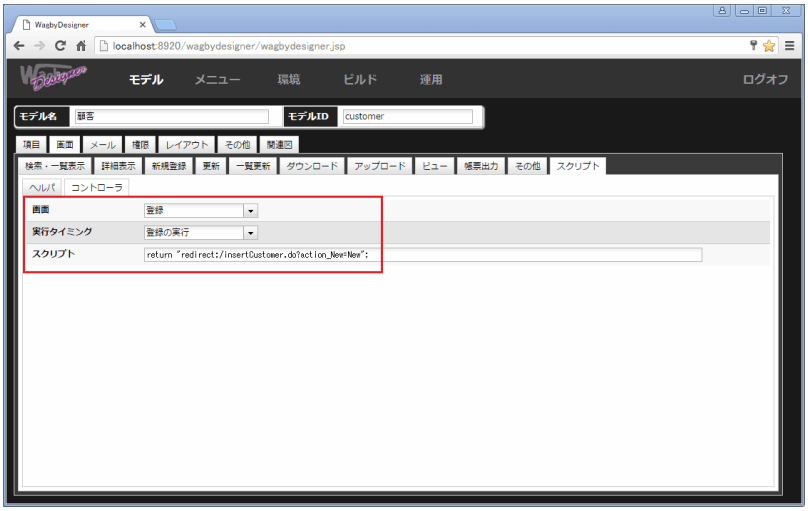


図 顧客モデルの登録時に動作するスクリプトを記述する

パラメータを付与する

再呼出された登録画面に、前回の登録画面で選択した値を渡し、初期値として表示することができます。

例えば、他モデル参照（リストボックス）項目 item1 の値を渡す場合、次のようにします。

```
return "redirect:insertCustomer.do?action_New&item1="+customer.getItem1();
```

受け取る側（item1）は、初期値（登録画面表示時）に次のような式を書きます。

```
TOINT(REQUEST())
```

詳細は「画面遷移 (1) 詳細画面から登録画面へ遷移する ＞ モデル参照項目の扱い」を参考にしてください。

複数のパラメータを付与する

複数のパラメータを付与する場合は、各パラメータを & で区切ります。

```
return "redirect:insertCustomer.do?
action_New&item1="+customer.getItem1()+"&item2="+customer.getItem2();
```

すべての受け取る側の項目には、初期値（登録画面表示時）に REQUEST() 関数を適用してください。

日本語文字列をパラメータとして付与する

日本語文字列（を含む、英語圏以外の文字列）をパラメータとして付与する場合は、文字エンコーディングの配慮が必要となります。

ここではWagbyが提供するユーティリティクラス jp.jasminesoft.util.WebStringEncodeUtilities を使う方法を紹介します。

```
var WebUtils = Java.type("jp.jasminesoft.util.WebStringEncodeUtilities");
return "redirect:insertCustomer.do?
action_New&customername="+WebUtils.encodeString(customer.getCustomername());
```

[例] 更新処理から、メニューへ遷移する

システムが提供するログオンアカウントモデル "juser" と 1:1 で紐づく「スタッフ (staff)」モデルを作成したとします。このとき、次の要件を満たすようにします。

- メニューからは、自分（ログオンID）だけの「スタッフ」モデルの更新処理が行えるようにする。
- 検索画面や一覧画面は不要。

・　更新後、詳細画面ではなく、すぐにメニューに戻るようにする。

スタッフモデルを用意する

1. スタッフは juser と 1:1 の関係にあるため、スタッフの主キーを juser の主キーと同一にします。具体的には、主キー「スタッフID」項目は、juser モデルの主キー項目 userid を参照するようにします。

図 スタッフIDは、juser項目と1:1の関連とする

2. 主キー「スタッフID」項目は、読み込み専用とします。

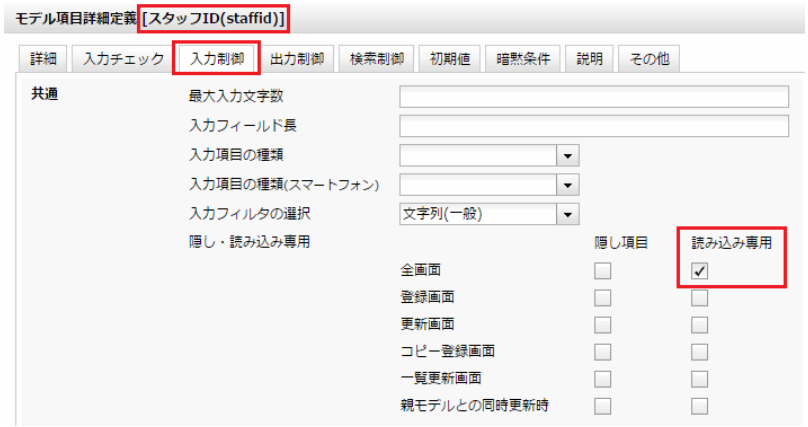


図 読み込み専用設定

3. 主キー「スタッフID」項目は、検索時に隠し項目とします。

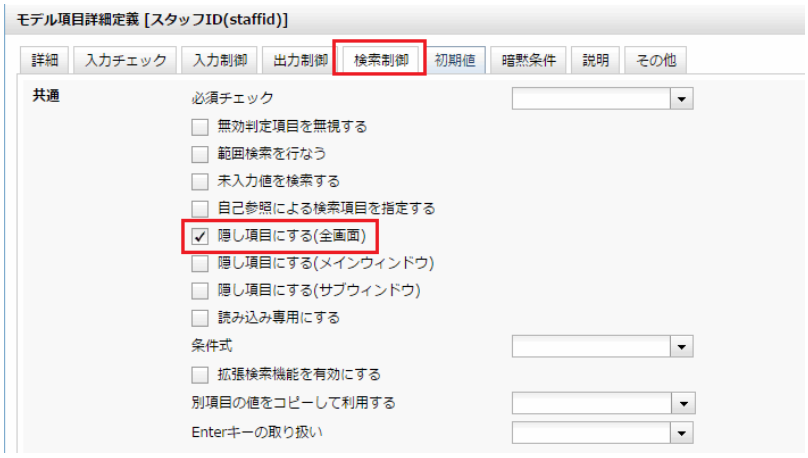


図 検索時に隠し設定（検索条件にする設定と併用）

4. 主キー「スタッフID」項目の初期値を「ログオンID」とします。これは登録画面表示時、登録時、更新、検索時いずれも設定します。

図 初期値の設定

5. 「検索画面と一覧表示画面を同時に表示する」を有効にします。
6. 「検索結果が1件のとき更新画面に遷移する」を〝常に〟と設定します。

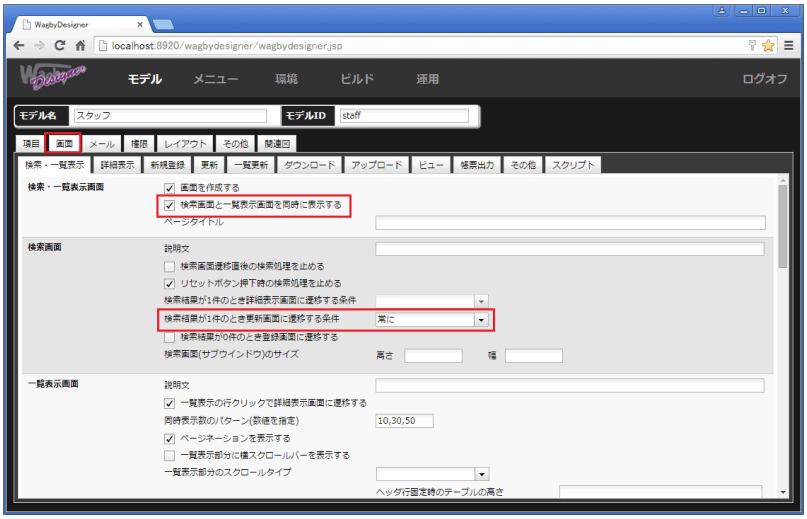


図 画面動作の設定(1)

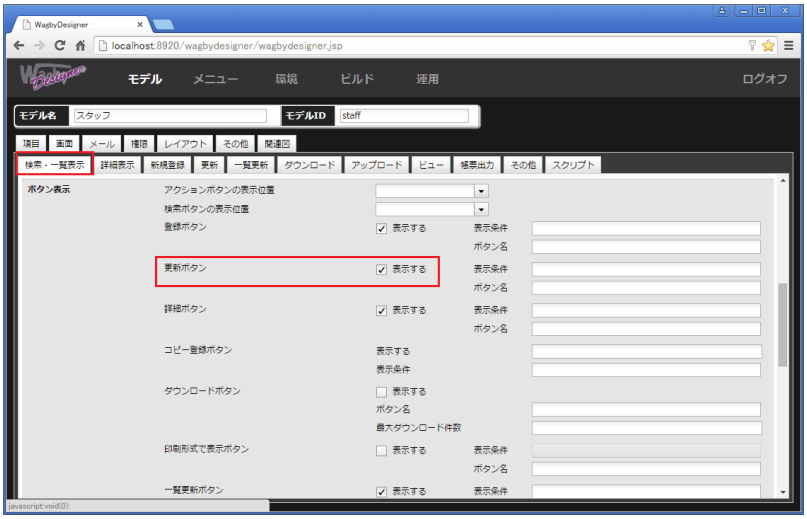


図 画面動作の設定(2)

7. メニューに表示されるラベル名を「スタッフ検索」から「スタッフ更新」へ変更します。

スクリプト

更新画面で保存ボタン押下時に呼び出されるスクリプトを設定します。メニュー画面へ遷移させるようにします。

```
if (p.errors.sizeJfcwarn() == 0 && p.errors.sizeJfcerror() == 0) {
    return "redirect:mainMenu.do";
}
```

p.errors.sizeJfcwarn() は警告メッセージの数を返します。p.errors.sizeJfcerror() はエラーメッセージの数を返します。入力チェックで警告またはエラーとなった場合は、画面遷移ルールを指定しません。

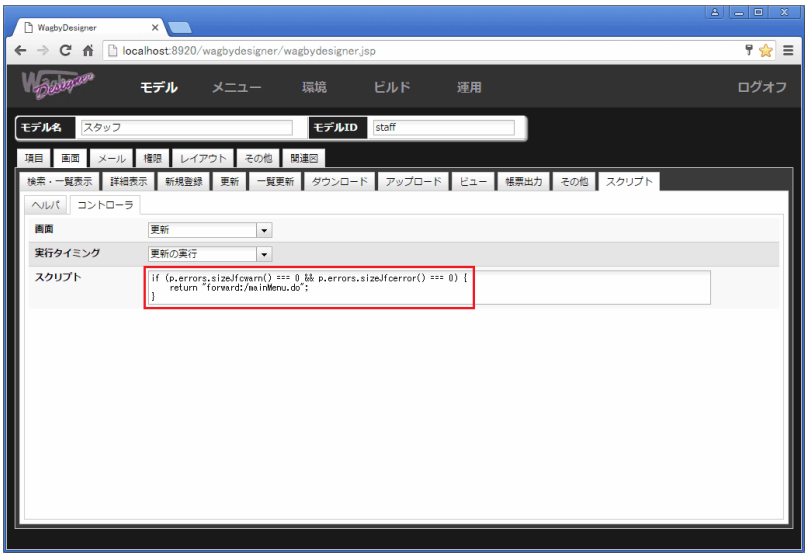


図 更新時のスクリプト

更新画面でキャンセルボタン押下時に呼び出されるスクリプトも用意します。ここは return のみとします。

```
return "redirect:mainMenu.do";
```

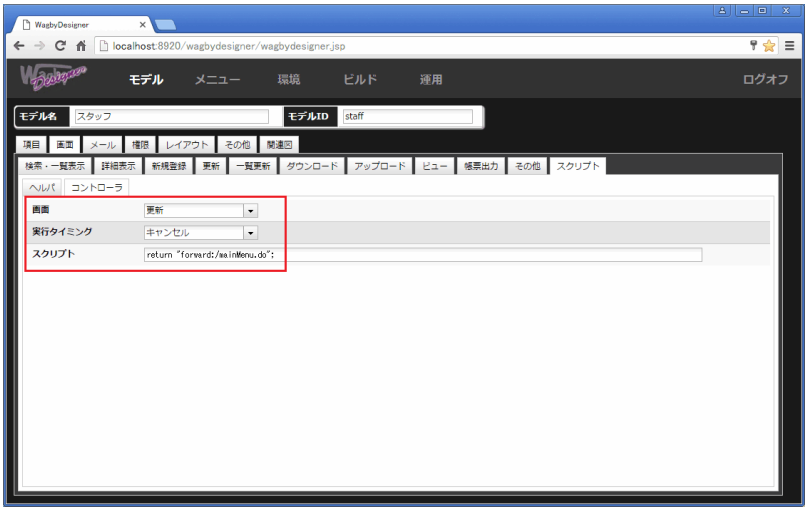


図 キャンセル時のスクリプト

例

メニューから「スタッフ更新」を選択すると、ログオンアカウントに紐付いたスタッフの更新画面が開きます。これは実際には検索一覧画面を開きますが、検索結果が1件のとき更新画面に遷移する設定と併用して実現しています。

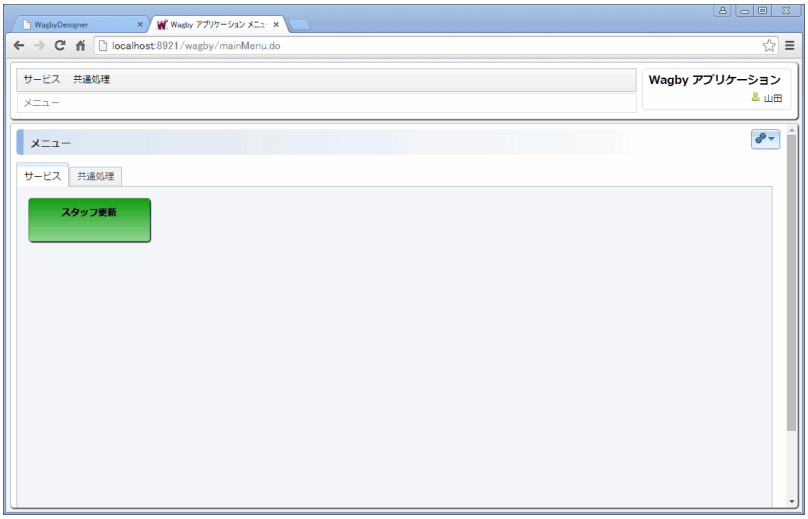


図 メニューに用意された更新ボタン（実体は検索画面）

保存ボタンまたはキャンセルボタンを押下すると、メニュー画面へ戻ります。

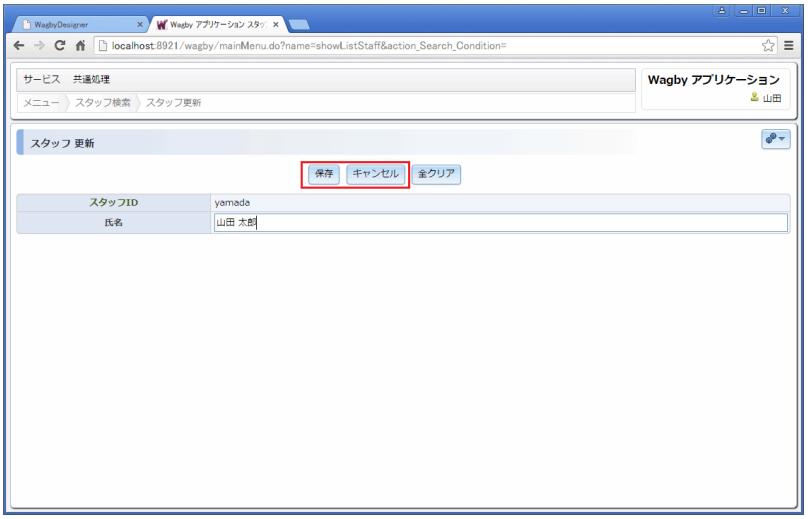


図 保存ボタンを押下した例

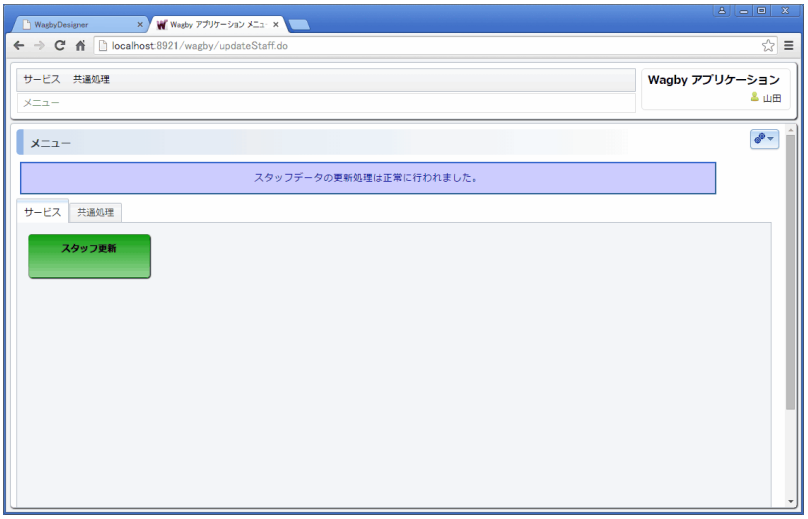


図 メニューへ戻っている（更新は行われている）

[例] ウィザードの任意の画面へ遷移する

コントローラのスクリプト、もしくはオリジナルボタンのスクリプトに次のような記述を行うことができます。

```
return "redirect:showModel1P1.do?pkey=1000&action_WizChange&_wizard_page=2";
```

(248)

- モデルが model1 のとき、showModel1P1.do と、"P1" が必要になります。
- イベント名として、action_WizChange=WizChangeを加えてください。
- _wizard_page=数字を加えてください。_wizard_page で画面番号を指定します。数字は1から開始されます。1未満もしくは数字ではないイレギュラー文字が指定された場合は1と扱います。数字が、準備している画面より大きい場合は、最後の画面となります。

[例] 独自ボタン押下で、マルチセッションを使って任意の画面を開く

マルチセッションとは、1つのブラウザでタブを開いたとき、それぞれのタブで Wagby アプリケーションにログオンする仕組みです。[詳細...]

ここでは独自ボタン押下時、マルチセッションを使ってログオンし、任意の画面に遷移するスクリプトを説明します。

独自ボタンのイベント名を "Original1" とします。WEB-INF/script/<モデルID>/ShowList<モデルIDキャメル記法>_Original1.js をエディタで作成します。

```
function process() {
  var ExcelFunction = Java.type("jp.jasminesoft.util.ExcelFunction");
  var now = ExcelFunction.NOW();
  var s = ExcelFunction.RIGHT(
            ExcelFunction.RPAD(
              ExcelFunction.TEXT(
                ExcelFunction.TIME(
                  ExcelFunction.HOUR(now),
                  ExcelFunction.MINUTE(now),
                  ExcelFunction.SECOND(now)
                ),
                ".#####"
              ),
              6,
              "0"
            ),
            5
          );
  //print(s);
  var _ss = p.request.getParameter("_s");
  if (_ss === null) {
    _ss = "0";
  }
  return "redirect:/"+s+"/showList<遷移先のモデルIDキャメル記法>.do?_ss="+_ss;
}
```

- Wagbyのマルチセッションは、URLの一部に5桁のランダムな数字を含めて管理しています。これがマルチセッションを実現するための識別子となっています。新規ログオンを行うたびに、新しい数字が生成されます。つまり5桁の数値を使って異なるセッションと扱いますが、これは重複しなければ任意の値でよいとなっています。上のスクリプトでは時分秒から求めています。（厳密には重複するかもしれませんが、可能性は低い、としています。）
- 遷移時に _ss パラメータを付与します。初回は "0" を指定します。そのあとは付与されたマルチセッションの値を使います。一度付与されたマルチセッションの値は p.request.getParameter("_s") で取得できる仕様です。

(248) オリジナルボタンのスクリプトはDesignerでは作成できないため、手動でご用意ください。

フォワードとリダイレクト

return で次の画面遷移先を指定するとき、接頭語 forward または redirect のいずれかを指定します。両者の違いは次の通りです。

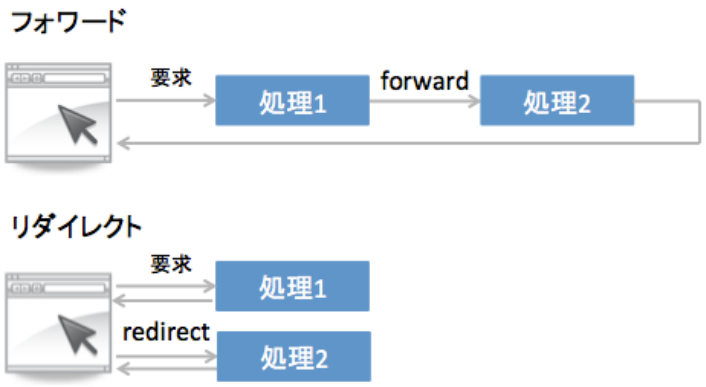


図 フォワードとリダイレクト

	フォワード	リダイレクト
処理の流れ	ブラウザからの要求（リクエスト）を、forward で指定した先に引き継ぐ。	処理がいったんブラウザに返されたあと、ブラウザが続けて redirect 先に新たに要求（リクエスト）を行う。
使い分け	ブラウザからみると、フォワード先の処理も含めて「一つの（一連の）」処理とみなせる場合に有効。同じ画面に戻る場合といった、限定的な用途で使う。	いったん処理を完了し、新しく次の処理に進めたい場合に有効。別画面へ遷移する場合、原則はこちらを利用する。
利用者（ブラウザ）の視点	一回の通信で完結。内部で「処理2」も行われているが、利用者はそれに気づかない。処理2から結果が送信されるが、利用者は「処理1を要求した結果」だと解釈する。	通信は二回、発生する。最初の結果を受け取ってすぐ、次の画面を開く要求が生じる。
p.requestの値	forward で指定した先でも利用できる。遷移先が異なる画面の場合、遷移元の値が引き継がれると誤動作の原因となることに留意すること。	redirect で指定した先では利用できない。

(249)

仕様・制約

生成されるスクリプトファイル

動作するスクリプトは、次のファイル命名規則に従って WEB-INF/script/モデルID フォルダ内に生成されます。

```
操作アクション + モデル名 + "_" + ボタン操作でサーバに送られる文字列.js
```

このスクリプトは所定のアクション処理が終了した後に呼び出されます。ファイル名の具体例を示します。

(249) 画面に表示される"登録が正常に終了しました。"などのメッセージは、p.requestに保持されています。forward を使うと（forward 先の処理を行ったあと）画面にメッセージが表示されます。redirect を使うとメッセージは表示されません。

juserモデルの登録処理のあとに呼び出されるスクリプトファイル

```
WEB-INF/script/juser/InsertJuser_Send.js
```

customerモデルの「印刷形式で表示」ボタン押下のあとに呼び出されるスクリプトファイル

```
WEB-INF/script/customer/ShowCustomer_JFCPrintScreen.js

WEB-INF/script/customer/ShowListCustomer_JFCPrintScreen.js
```

staffモデルの「更新」ボタンを押したあとに呼び出されるスクリプトファイル

```
WEB-INF/script/staff/UpdateStaff_Send.js
```

staffモデルの「キャンセル」ボタンを押したあとに呼び出されるスクリプトファイル

```
WEB-INF/script/staff/UpdateStaff_Cancel.js
```

帳票出力後の画面遷移は指定できません

帳票出力（一括出力を含む）のスクリプトでは、画面遷移を指定することができません。（指定しても無視されます。）

画面遷移（7）独自ボタンから外部コマンドを呼び出す(スクリプト)

例

新規登録画面に「バッチ起動」というボタンを配置した例です。

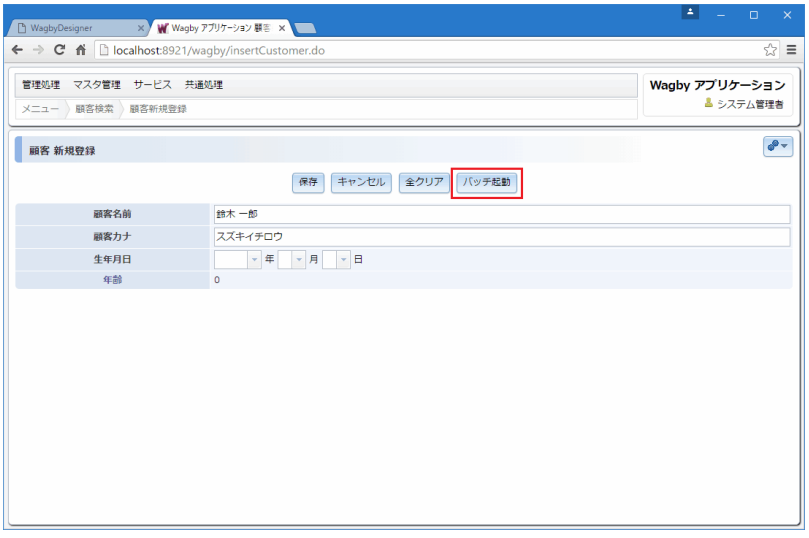


図 「バッチ起動」ボタンを用意する

今回の例では、このボタンを押すと、Tomcatコンソールに「Wagbyをインストールしたフォルダの内容」が出力されるようにしています。

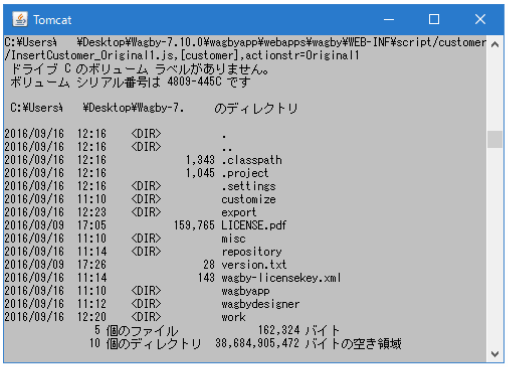


図 ボタン押下の例

定義方法

「画面 > その他 > Javaソースコードの設定」欄で、ボタンを用意します。

ボタン名は任意ですが、イベント名は接頭語 "Original" とします。この例では Original1 と、数字1を加えたイベント名としています。

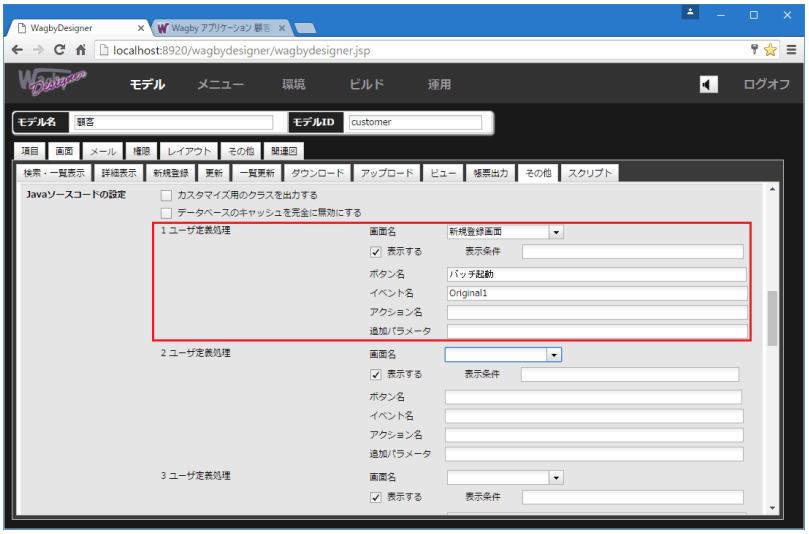


図 オリジナルボタンを用意する

スクリプトを用意する

上の設定によってボタンが用意されますが、このボタンを押したときの処理は Java ソースコードまたはスクリプトによって記述します。ここではスクリプトによる記述方法を説明します。

スクリプト名

スクリプト名は「画面種別 + モデルID + "_" + イベント名」になります。

画面種別は SCREENTYPE 関数の戻り値です。ただし先頭は大文字となります。具体的には Show,Insert,Update などです。

モデルIDの先頭は大文字とします。customer モデルの場合、Customer になります。

イベント名は図で設定した記載を使います。

以上をまとめます。図の場合、スクリプト名は InsertCustomer_Original1.js になります。

「呼び出されるスクリプトファイル名を確認する」方法を使うことで、簡単に知ることができます。

配置場所

ビルド済みのアプリケーションに対して直接、スクリプトファイルを上書き保存することで、すぐに動作を確認できます。（ボタンを押すとスクリプトが実行されます。） この場合は wagbyapp/webapps/プロジェクト識別子/WEB-INF/script/モデルID フォルダに保存します。

動作確認できたスクリプトファイルは、customize/webapp/WEB-INF/script/モデルID フォルダに保存してください。（script/モデルIDフォルダは開発者で作成します。）次回以降のビルド処理で、このファイルが含まれるようになります。

スクリプトの基本形

開発者が作成するスクリプトは「業務ロジック > 生成されるスクリプトファイルの詳細 > 基本テンプレート」の雛形を使うとよいでしょう。

```
function process() {
var ExcelFunction = Java.type("jp.jasminesoft.util.ExcelFunction");
var Jfcerror = Java.type("jp.jasminesoft.jfc.error.Jfcerror");
var Jfcwarn = Java.type("jp.jasminesoft.jfc.error.Jfcwarn");
var Jfcinfo = Java.type("jp.jasminesoft.jfc.error.Jfcinfo");

...ここに開発者独自のコードを記述する...

}
```

外部コマンドを実行するスクリプト

EXEC関数を利用する

Wagbyが提供する関数 ExcelFunction.EXEC を使います。引数などの詳細は"関数 > EXEC"をお読みください。

```
function process() {
  var ExcelFunction = Java.type("jp.jasminesoft.util.ExcelFunction");
  var Jfcerror = Java.type("jp.jasminesoft.jfc.error.Jfcerror");
  var Jfcwarn = Java.type("jp.jasminesoft.jfc.error.Jfcwarn");
  var Jfcinfo = Java.type("jp.jasminesoft.jfc.error.Jfcinfo");

  var result = ExcelFunction.EXEC("../..", "cmd", "/C", "dir");/*Windows OS用*/
  print(result);
}
```

環境変数を指定したい場合

Wagbyが提供するクラス jp.jasminesoft.jfc.job.ProcessExecTask を使います。これは"外部コマンドを実行するジョブ"で用いられているものです。

提供されるメソッド	説明	例
setCmd(String cmd)	実行したい処理を記述します。	cmd /C ¥"dir C:¥"
setDir(String dir)	実行するときのフォルダを指定します。省略時は wagbyapp/bin です。カレントフォルダは "." で表現します。フォルダの区切り文字 "/" で終わらないようにしてください。例えば "../" ではなく "../." と記述します。	C:¥work.../..

setMap(Map<String,String> map)	環境変数をjava.util.Mapに格納して渡すことができます。Mapのキーが環境変数名、値が環境変数の値として解釈されます。	（省略）
execute(ActionParameter p)	setCmd で渡された処理を実行します。当該処理がコンソールに出力した文字が、戻り値となります。	（省略）

このクラスを使って外部コマンドを実行するサンプルスクリプトは次のとおりです。

```
function process() {
  var ExcelFunction = Java.type("jp.jasminesoft.util.ExcelFunction");
  var Jfcerror = Java.type("jp.jasminesoft.jfc.error.Jfcerror");
  var Jfcwarn = Java.type("jp.jasminesoft.jfc.error.Jfcwarn");
  var Jfcinfo = Java.type("jp.jasminesoft.jfc.error.Jfcinfo");

  var ProcessExecTaskClass = Java.type("jp.jasminesoft.jfc.job.ProcessExecTask");
  var processor = new ProcessExecTaskClass();

  processor.setCmd("cmd /C ¥"dir C:¥"");
  processor.setDir("../..");
  var result = processor.execute(p);
  print(result);
}
```

- setCmd メソッドの引数に、実行したい処理（バッチファイル名）を記述します。ここでは Windows OS が標準で提供する、ディレクトリリストを返すコマンドを記述しています。
- setDir メソッドに、実行時のフォルダを指定します。省略時は wagbyapp/bin であるため、上の例のように相対パス指定することで、Wagby インストールフォルダを指定することもできます。
- execute メソッドの引数「p」はスクリプト内で暗黙的に利用できます。そのままご利用ください。

ボタン押下後に再び自画面に戻る

詳細表示画面に独自ボタンを定義し、ボタン押下後に再び自画面（同じデータの詳細表示画面）を開く場合は、スクリプトを次のように記述します。

ここではモデル customer とし、主キー項目を customerid とします。スクリプトファイルを ShowCustomer_Original1.js とします。

```
function process() {
  var ExcelFunction = Java.type("jp.jasminesoft.util.ExcelFunction");
  var Jfcerror = Java.type("jp.jasminesoft.jfc.error.Jfcerror");
  var Jfcwarn = Java.type("jp.jasminesoft.jfc.error.Jfcwarn");
  var Jfcinfo = Java.type("jp.jasminesoft.jfc.error.Jfcinfo");

  ...(開発者が行う処理)...

  var id = p.request.getParameter("customerid");
  return "redirect:showCustomer.do?customerid="+id;
}
```

returnで自分自身の詳細画面に戻す際に、主キーを指定します。

画面遷移（8）登録・更新画面に独自ボタンを用意する(スクリプト)

仕様・制約

新規登録および更新画面にボタンを追加する場合、この画面遷移は必ず同じモデルに戻ることが仕様となってます。つまり登録画面から詳細画面へ遷移したり、更新画面から検索画面へ遷移させることはできません。

このため、別画面へ遷移させる独自ボタン（ボタンのアクション名を指定するという意味です）を、新規登録および更新画面に配置することはできません。

(250)

例 登録画面から別モデルの検索画面へ遷移する

スクリプトを使って異なる画面遷移を実現する方法を説明します。

スクリプト

model1 モデルの登録画面に、独自ボタンを配置します。ボタンのアクション名を空白にします。またイベント名は適当に設定します。ここでは "Original1" とします。

その後、スクリプトを作成します。　オリジナルボタン押下のスクリプトは　Designer　では作成できないため、手動で用意します。今回は　wagbyapp/webapps/wagby/WEB-INF/script/model1　フォルダに InsertModel1_Original1.js を作成します。 内容はリダイレクト処理で別モデル model2 の検索画面へ遷移させるものです。

```
function process() {
  return "redirect:showListModel2.do?action_Reset=";
}
```

(251)

- 応用 : 画面遷移 > メインモデルの更新画面からサブモデルの更新画面へ遷移する

例 ボタン押下時に繰り返しコンテナを追加する

実行例

新規登録画面に「商談履歴10件追加」というボタンを配置した例です。

(250) ボタンのアクション名を指定した独自ボタンがあると、生成されるHTMLフォームの中にフォーム要素が含まれる構造となります。しかしフォーム要素の入れ子は認められません。

(251) イベント名にResetを指定した場合、検索条件や開始件数がクリアされます。遷移した一覧表示画面では1ページ目から表示されます。8.3.4

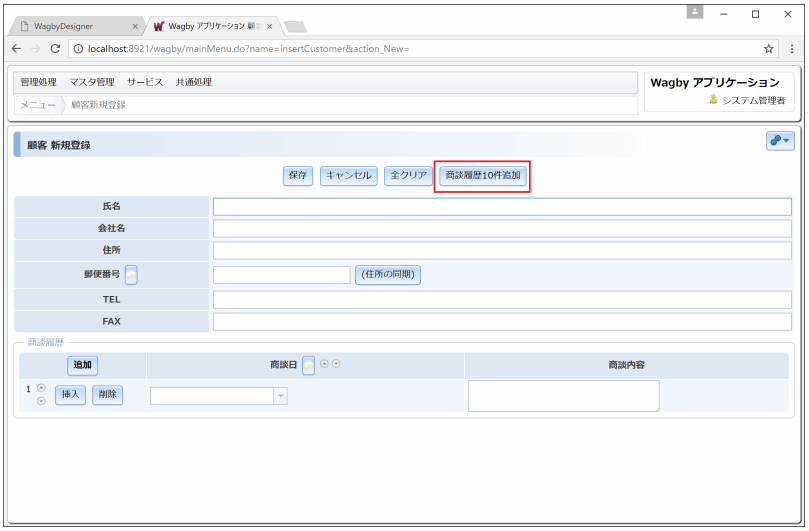


図 「商談履歴10件追加」ボタンを用意する

このボタンを押すと「商談履歴」（繰り返しコンテナ）が10件追加されます。

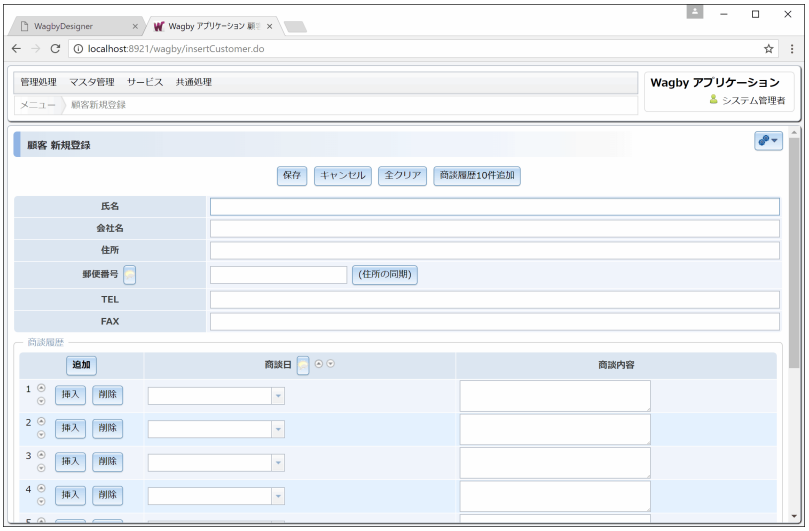


図 ボタン押下の例

定義方法

「画面 ＞ その他 ＞ Javaソースコードの設定」欄で、ボタンを用意します。

ボタン名は任意です。イベント名は "Add10Reports" としました。

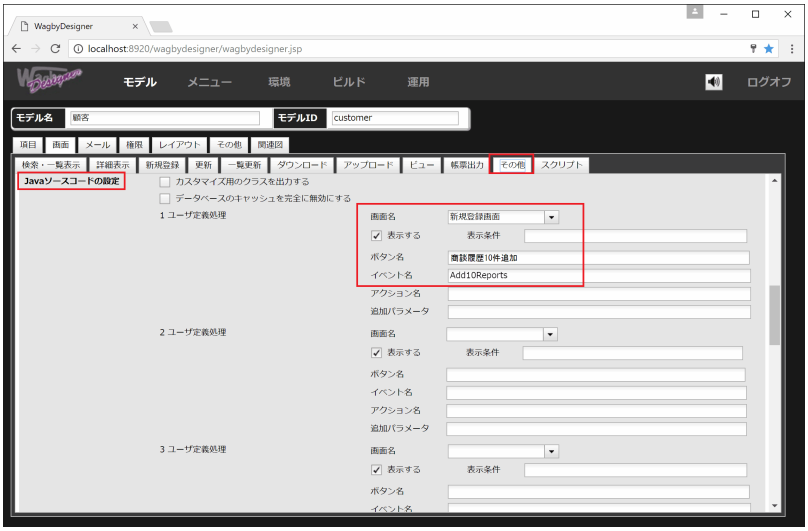


図 オリジナルボタンを用意する

スクリプト

オリジナルボタン押下のスクリプトは Designer では作成できないため、手動で用意します。今回は wagbyapp/webapps/wagby/WEB-INF/script/customer フォルダに InsertCustomer_Add10Report.js を作成します。

スクリプトは次の通りです。

```
function process() {
  var PHelperClass = Java.type("jp.jasminesoft.jfc.IPresentationHelper");
  var helper = p.appctx.getBean("CustomerHelper");
  var phelper = p.appctx.getBean("CustomerPHelper");

  var customer = p.request.getAttribute("customer");
  for (var i = 0; i < 10; i++) {
    /* 商談履歴の最後尾に1レコード追加 */
    helper.addReport(customer, -1, true, p, errorManager);
  }
  var customer_p = phelper.s2p(customer, p, PHelperClass.UPDATE);
  p.request.setAttribute("customer_p", customer_p);
}
```

- p.request から対象のストアモデルを取り出します。p.request.getAttribute メソッドに渡すキー名はモデルIDになります。
- 繰り返しコンテナの追加は、同モデルのヘルパクラスに用意されるメソッド add<繰り返しコンテナ名> を使うことができます。第一引数は対象となるストアモデルです。これはスクリプト内で暗黙オブジェクトとして利用できます。第二引数は挿入位置をさします。第三引数が true の場合は"追加"を意味します。第二引数が -1 で、第三引数が true の場合は末尾に追加します。
- 修正後のストアモデルを、プレゼンテーションモデルに変換します。このためにプレゼンテーションモデルを操作するためのヘルパクラス CustomerPHelper の s2p メソッドを用います。第一引数にストアモデルを、第二引数に p を指定します。第三引数は、jp.jasminesoft.jfc.IPresentationHelper で定義された定数 UPDATE を指定します。これは編集画面用のプレゼンテーションモデルになります。
- 変換後のプレゼンテーションモデルを p.request に格納することで画面に表示されるようになります。p.request.setAttribute メソッドに渡すキー名は "モデルID_p" としてください。

リアルタイム入力チェックを無効にする (8.3.1)

登録・更新画面では、入力チェックエラー発生時はボタンを押下することができません。これを無視して、入力チェックエラー発生時でもオリジナルボタンを押下できるようにする方法を説明します。

各画面のカスタマイズ JavaScript ファイル(insert<モデルID>.jsなお)、またはモデルの My.js ファイルに以下の関数を定義します。

```
function isSubmitIfError(actionName) {
 if (actionName === "action_Original1") {
   return true;
 }
 return false;
}
```

引数の actionName には、ボタン押下時に実行される doSubmitForm 関数の第二引数のアクション文字列が引き渡されます。

(252)

画面遷移（9）画面遷移パラメータまとめ

アクション名

モデルと画面の組み合わせから構成される文字列です。アドレスバー(URL)の一部になります。

遷移先画面	記載方法	例	補足
新規登録画面	"insert" + モデル名(キャメル記法)	insertCustomer	新規登録画面を開くときはイベント"New"が必要です。
更新画面	"update" + モデル名(キャメル記法)	updateCustomer	更新画面を開くときはイベント"Update"が必要です。パラメータに対象となる主キーを指定してください。
検索一覧画面	"showList" + モデル名(キャメル記法)	showListCustomer	イベント"Reset","Search","Search_Condi いずれかを指定してください。
一覧更新画面	"updateList" + モデル名(キャメル記法)	updateListCustomer	一覧更新画面を開くのみとなります。
詳細画面	"show" + モデル名(キャメル記法)	showCustomer	パラメータに対象となる主キーを指定してください。

(253) (254)

仕様・制約

- 「検索画面と一覧表示画面を同時に表示する」設定を有効とした場合、「検索画面」の設定では検索一覧画面上に独自ボタンは表示されません。「一覧表示画面」の設定で検索一覧画面上にボタンが表示されます。

イベント名

適用画面	イベント名	コマンド	説明
登録画面	New	BeginInsert	新規登録画面を開きます。
登録画面	Send	DoInsert	登録処理を行います。（フォームデータが送信され、

(252) この JavaScript はブラウザで実行されます。サーバで動作する「スクリプト」とは区別されます。

(253) キャメル記法 : 先頭の1文字を大文字とします。モデル名に含まれるアンダースコアは取り除き、その次の文字を大文字にします。例えば my_funny_valentine は MyFunnyValentine になります。

(254) アクション名に拡張子（".do"）は記述しません。

			データが登録されます。）
更新画面	Update	BeginUpdate	更新画面を開きます。
更新画面	Send	DoUpdate	更新処理を行います。（フォームデータが送信され、データが更新されます。）
登録・更新画面	Add	AddItem	繰り返し項目または繰り返しコンテナ項目を追加します。
登録・更新画面	Remove	RemoveItem	繰り返し項目または繰り返しコンテナ項目を削除します。
登録・更新画面	MoveUp	MoveUpItem	対象となる繰り返し項目または繰り返しコンテナ項目の位置を一つ上へ移動します。
登録・更新画面	MoveDown	MoveDownItem	対象となる繰り返し項目または繰り返しコンテナ項目の位置を一つ下へ移動します。
登録・更新画面	Sort	DoSort	対象となる繰り返し項目または繰り返しコンテナ項目の並び替えを行います。
登録・更新画面	Zipcode	Zipcode	郵便番号から住所を求めます。
登録・更新画面	FileUpload	FileUpload	ファイルを送信します。
登録・更新画面	Cancel	Cancel	登録・更新処理をキャンセルします。
登録・更新画面	Confirm	Confirm	確認画面へ遷移させます。
登録・更新画面	Clear / AllClear	AsIs	値を消去後、画面再描画を行います。
登録・更新画面	GetXml	GetXml	Ajaxでデータをやりとりします。
更新画面	SUpdateNext	DoUpdateAndBeginUpdate	更新して次へボタン押下時 (*1)8.3.3
更新画面	SUpdatePrev	DoUpdateAndBeginUpdate	更新して前へボタン押下時 (*1)8.3.3
検索画面	Reset	Search	現在の検索条件をリセットし、外部パラメータを用いた検索処理を行います。外部からパラメータを渡す場合に有効です。また検索画面表示時の初期値設定を用いたい場合も、こちらを指定してください。
検索画面	Search	Search	最初にセットされた検索条件を使います。このため二回目以降の外部パラメータはセットされません。これは検索画面

			と一覧表示画面を行き来するような画面遷移で、検索条件を保持するためのイベントです。
検索画面	Search_Condition	Search	Searchと同様ですが、パラメータがない場合は空の検索条件が指定されたものとして扱います。
一覧表示画面	Show	Forward	詳細画面へ遷移します。
一覧表示画面	Update	Forward	更新画面へ遷移します。
一覧表示画面	Copy	Forward	コピー登録画面へ遷移します。
一覧表示画面	SizePerChange	ChangeSizePerPage	表示件数を変更します。
一覧表示画面	Sort	DoSort	一覧表示データの並び替えを行います。
一覧表示画面	Next	Next	一覧表示データの、次の N 件を表示します。
一覧表示画面	Prev	Prev	一覧表示データの、前の N 件を表示します。
一覧表示画面	First	First	一覧表示データの、最初の N 件を表示します。
一覧表示画面	Last	Last	一覧表示データの、最後の N 件を表示します。
一覧表示画面	Jump	Jump	一覧表示データでページネーション処理を行います。
一覧表示画面	DeleteAll	DoDeleteAll	一覧表示で選択されたデータの一括削除処理を行います。
一覧表示画面	UpdateAll	DoUpdateAll	一覧表示で選択されたデータの一括更新処理を行います。
詳細画面	Show	DoShow	詳細画面を開きます。
削除画面	Send	DoDelete	データを削除します。
登録・更新・詳細画面	Print	DoPrint	帳票処理を行います。
一覧表示画面	PrintAll	DoPrintAll	一覧表示データの帳票処理を行います。
一覧更新画面	ULInsert	UpdateListInsert	データを新規に作成します。編集画面のみに反映されます。
一覧更新画面	ULCopy	UpdateListCopy	データのコピー登録を行います。編集画面のみに反映されます。
一覧更新画面	ULExecute	UpdateListExecute	編集画面上のデータを更新します。
一覧更新画面	ULCancel	UpdateListCancel	編集画面のデータを破棄します。

ダウンロード画面	Reset	InputCondition	ダウンロード画面（検索画面）を開く。
アップロード更新画面	Status	ShowStatus	アップロード更新の稼働状況画面を開く。
（すべて）	（開発者が決めるイベント名）	UserDefined	オリジナルボタン押下時に送信します。

イベント名は大文字小文字を区別します。正確に入力してください。

(*1) "更新して次へ/前へ" の動作 (8.3.3)

更新画面に用意される "更新して次へ/前へ" ボタン押下時に呼び出されるスクリプトでは、対象モデルは次または前の画面へ遷移したあとが対象になります。

例えば主キー　1000,1001,1002　のモデル　model1　があり、主キー1000の更新画面で　"更新して次へ" ボタンを押下すると、スクリプト UpdateModel1_SUpdateNext.js が呼び出されます。このスクリプトで print(model1) とすると、主キー 1001 のモデルが対象となります。画面遷移後というルールです。

ここでスクリプトで画面遷移前(つまり主キー1000)の値を参照したい場合は、次のようにします。

```
var smodel = p.request.getAttribute("model1.update");//ストアモデル
var pmodel = p.request.getAttribute("model1_p.update");//プレゼンテーションモデル
```

接尾語 ".update" はこの "更新して次へ/前へ" スクリプトでのみ利用できる特別な指定です。この接尾語を用いると、更新されたオブジェクトを参照できます。

なお、ここで参照するオブジェクトはデータベースへの更新が終わった後の値です。このオブジェクトの値を変更することはできません。（値を変更してもデータベースには反映されません。）

URLに直接、イベントを指定する

スクリプトや、外部リンクの仕組みを使って遷移先画面の URL を直接指定したとき、イベント名を渡すこともできます。パラメータに　"action_"　という接頭語を付与し、そのあとに続けてイベント名を記述します。

例:外部リンクメニューボタン作成で、パラメータとイベントを渡す

モデル model1 の検索画面へ遷移し、さらにパラメータ item1 に値を、そして（検索条件の初期値を使うために）Reset イベントを指定する場合は次のように記述します。

```
/wagby/showListModel1.do?item2=1&action_Reset=
```

(255)

パラメータのキー部分に"action_" + イベント名を指定します。イコールの右側は空白とします。（イベントの指定はキー名のみが参照されるため、値部分は使いません。）

参考:メニュー > 外部サイトへのリンク

モデル参照項目のルール

パラメータの右側に記載する項目（${...}で指定した項目）がモデル参照の場合、次のルールで値を渡します。

- モデル参照の場合はID部を渡します。
- 参照連動の場合は内容部を渡します。

このルールを変更したい場合、項目名の後ろに #id または #content を指定してください。 例えば参照連動項目 companyname について、ID 値を渡したい場合は ${companyname#id} と記述します。

(255) イベント名にResetを指定した場合、検索条件や開始件数がクリアされます。遷移した一覧表示画面では1ページ目から表示されます。8.3.4

遷移先モデルが複合キーの場合

遷移元を model1、遷移先を model2 とします。model2 は複合キーで、それぞれ pkey1 と pkey2 とします。

この場合は model1 が保持する二つの項目（例 item1, item2）をそれぞれ pkey1, pkey2 に渡すようにします。渡す側と、受け取る側の項目の型を合致させてください。

遷移先モデルのモデル参照項目が複合キーの場合

遷移元を model1、遷移先を model2 とします。model2 内の項目 item3 はモデル参照項目で、参照先モデルを model3 とします。model3 は複合キーで、それぞれ pkey1, pkey2 とします。

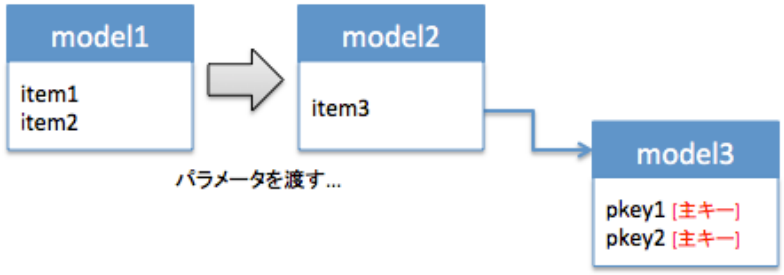


図 サンプルの構成

model1に用意したボタンで記載するパラメータ

参照先モデルの主キーが複合キーの場合、追加パラメータを次のように記述します。

```
項目名=${主キー1}$SEP$${主キー2}
```

主キー項目名を ${...} で記述しますが、さらにキーの連結文字列として、間にSEPを加えています。

```
item3=${item1}$SEP$${item2}
```

図 model2へ渡すパラメータ

model2に用意するスクリプト

受け側となる model2 では、初期値の式ではなく、スクリプトによって値を解析してセットします。タイミングは「登録（初期データ作成）」です。

```
var item3 = p.request.getParameter("item3");
if (item3 != null) {
    var ss = item3.split("$SEP$");
    /*
    for (var i in ss) {
        print(i+"="+ss[i]);
    }
    */
    model2.item3Pkey1 = ss[0];
    model2.item3Pkey2 = ss[1];
```

```
}
```

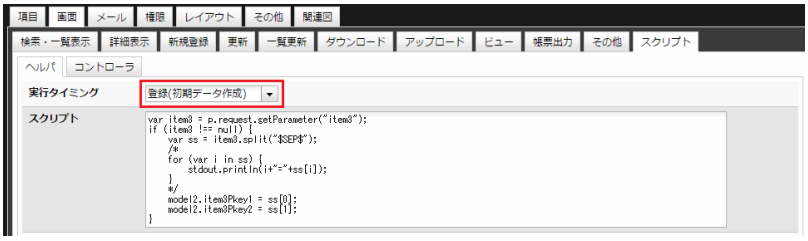


図 model2に用意するスクリプト

p.request は、Java アプリケーションサーバが管理する request オブジェクトです。getParameter メソッドを使うと、図で指定したパラメータを受け取ることができます。

SEP を区切り文字として split 関数を呼び出すと、配列が用意されます。個々の値を model2 の要素にセットします。

ここで、model2 では "item3" という項目を用意していますが、内部では次のように管理されています。

```
項目名_参照先モデルの主キー項目名(1番目)
項目名_参照先モデルの主キー項目名(2番目)
```

すなわち model2 では "item3_pkey1" と "item3_pkey2" という二つの項目が内部で保持されます。サーバサイドJavaScriptではキャメルルールにより、それぞれ "item3Pkey1", "item3Pkey2" と表記されます。(Pkeyの'P'が大文字になっています。) *1

(256)

遷移元モデルのモデル参照項目が複合キーの場合

遷移元を model1、遷移先を model2 とします。model1 内の項目 item1 はモデル参照項目で、参照先モデルを model3 とします。model3 は複合キーで、それぞれ pkey1, pkey2 とします。

参照先モデルの主キーが複合キーの場合、追加パラメータを次のように記述します。

```
項目名=${項目名#id}$SEP$${項目名#id2}
```

項目名を ${...} で記述しますが、さらに項目名の後ろに #id, #id2,... という接尾語を加えています。接尾語の数字に対応して、model3 の主キー pkey1, pkey2 がそれぞれ渡されます。

遷移先モデル model2 の item 項目に渡す場合、次のようになります。

```
item=${item1#id}$SEP$${item1#id2}
```

受け側である model2 では、スクリプトを使って区切り文字 SEP を分割してください。(「遷移先モデルのモデル参照項目が複合キーの場合」で説明済みのため、省略します。)

チェックボックスの場合

チェックボックス型項目の場合、コンマ区切りで値(参照先モデルのID部)が渡されます。遷移先モデルではスクリプトを使って、値を分割してから格納するようにしてください。遷移先モデル model2 の item 項目(これもチェックボックス型を想定)に値を格納するスクリプトの例を示します。

```
var item = p.request.getParameter("checkbox");
if (item !== null) {
  var array = item.split(",");
  var i;
  for (i=0; i<array.length; i++) {
    /*print("i="+i+","+array[i]);*/
    if (array[i] !== '') {
```

(256) 1. pkey1,pkey2という表記は固定ではありません。例えば参照先モデルの主キー項目名が id1, id2 だった場合は、model2.item3Id1 = ss[0]; model2.item3Id2 = ss[1]; となります。

```
        model2.addItem(parseInt(array[i]));/*model2のitemはチェックボックス型*/
      }
    }
}
```

区切り文字を変更する

コンマ以外の文字を区切り文字としたい場合、パラメータを次のように記述します。

```
item=${checkbox;|}
```

セミコロンのあとに区切り文字を指定します。受け側の JavaScript もそれにあわせて改造してください。

制約：この区切り文字指定機能に、次の文字は利用できません。

「;」「(」「)」「[」「]」「&」「=」「>」「<」「$」

画面遷移（10）JavaScript(フロントエンド)との連携(スクリプト)(JavaScript)

画面遷移時に確認ダイアログを表示する

model1 の画面に配置したオリジナルボタンを押下した際に確認ダイアログボックスを表示させるようにします。「OK」選択時にのみmodel2画面へ遷移させるものとします。

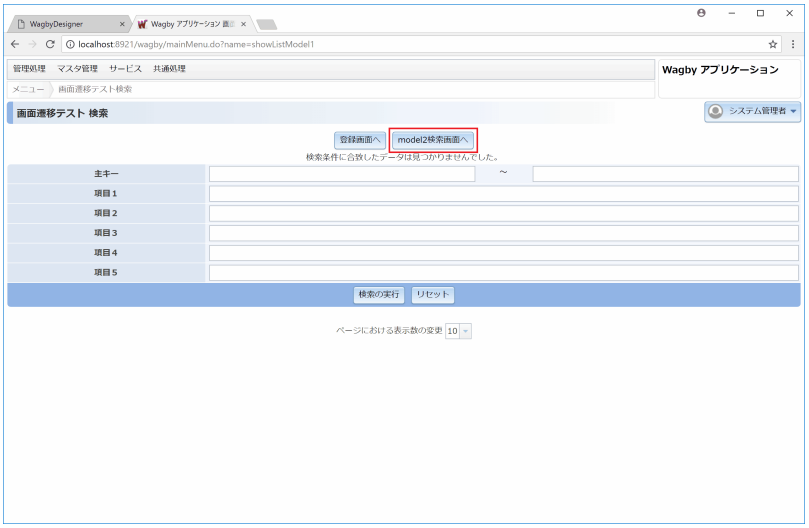


図 model2へ遷移するオリジナルボタン

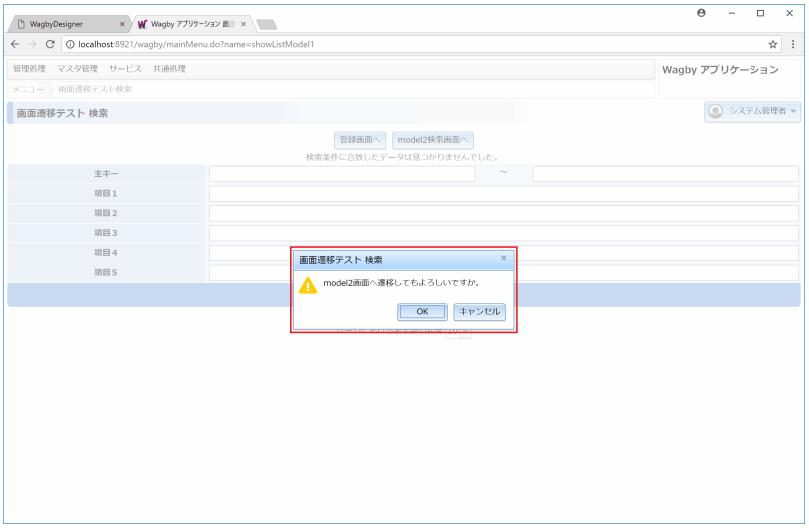


図 ボタン押下時に確認ダイアログが表示される

1. model1にオリジナルボタンを配置する

　ここでは　model1　一覧表示画面に「model2検索画面へ」というボタンを用意してみます。イベント名を ShowListModel2 とします。

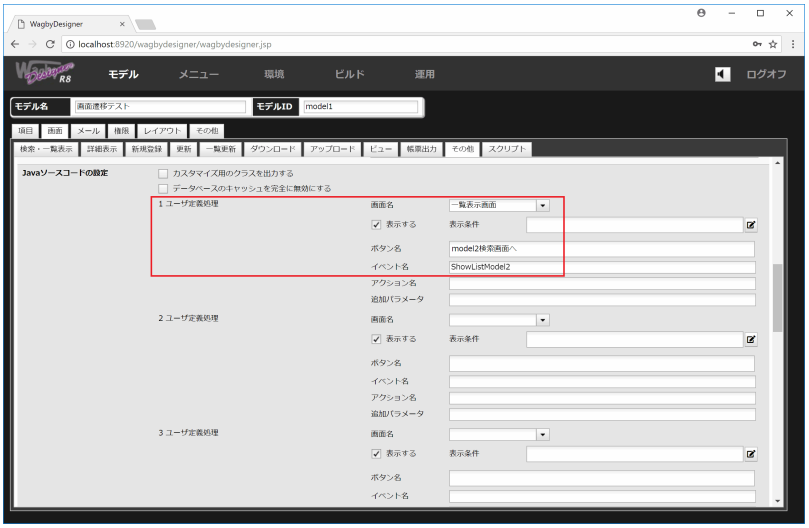


図 オリジナルボタンを用意する

2. オリジナルボタン押下時のスクリプト

　図で用意したオリジナルボタン押下時に呼び出されるスクリプトを作成します。スクリプトファイルは WEB-INF/script/model1/ShowListModel1_ShowListModel2.js となります。

```
function process() {
  var ExcelFunction = Java.type("jp.jasminesoft.util.ExcelFunction");
  var Jfcerror = Java.type("jp.jasminesoft.jfc.error.Jfcerror");
  var Jfcwarn = Java.type("jp.jasminesoft.jfc.error.Jfcwarn");
  var Jfcinfo = Java.type("jp.jasminesoft.jfc.error.Jfcinfo");

  return "redirect:mainMenu.do?name=showListModel2";
}
```

　別画面に遷移させるため redirect を指定しています。

3. ボタンIDの確認

自動生成されたファイル showList<モデルID>_navigationbar.jsp をテキストエディタで開きます。今回の場合は wagby/model1/showListModel1_navigationbar.jsp となります。

図で追加したオリジナルボタンのボタンIDを確認します。今回の場合は btnShowListModel1ShowListModel2 となっていました。

4. ブラウザで動作する JavaScript を用意する

customize フォルダに webapp/model1/MyShowListModel1.js を用意します。

(257) (258)

このファイルの内容は次のとおりです。

```
require(["dijit/registry", "dojo/ready"], function(registry, ready) {
  // 画面表示時に以下の関数が実行される
  ready(function() {
    // オリジナルボタンの id と onClick コードを showListModel1_navigationbar.jsp で確認します
    // オリジナルボタンの id でボタンのオブジェクトを取得します
    var btnShowListModel2 = registry.byId("btnShowListModel1ShowListModel2");
    if (!btnShowListModel2) {
      // 権限等でボタンが表示されていない場合、イベント設定は行なわない
      return;
    }
    // ボタン押下時のデフォルトの処理を書き換えて確認ダイアログを表示するようにする
    btnShowListModel2.onClick = function() {
      showConfirmDialog("model2画面へ遷移してもよろしいですか。", function() {
        // ボタン押下時のデフォルトの処理を実行する
        submitNavigationCommand('navigation', 'showListModel1.do', 'action_ShowListModel2',
'null');
      });
    }
  });
});
```

- フロントエンド部（ブラウザ）で動作する JavaScript ファイルでは、Dojotoolkit という UI ライブラリを利用します。先頭行の require 指定は Dojotoolkit のライブラリを有効にしています。
- showConfirmDialog 関数は Dojotoolkit が提供するライブラリです。

論理削除

一般利用者と業務管理者を用意する

ここでは図,図に示すような2つのアカウントを用意して説明します。

アカウント	プリンシパル	説明
user	一般ユーザ	論理削除のみ
manager	業務管理者	論理削除と物理削除が行える

(257) customize/webapp 以下の model1 フォルダは手動で作成する必要があります。

(258) ビルド後は wagby/model1/MyShowListModel1.js として保存されます。

図 一般ユーザ

図 業務管理者

例

一般ユーザ

一般ユーザでログオンします。顧客データを1件、登録します。

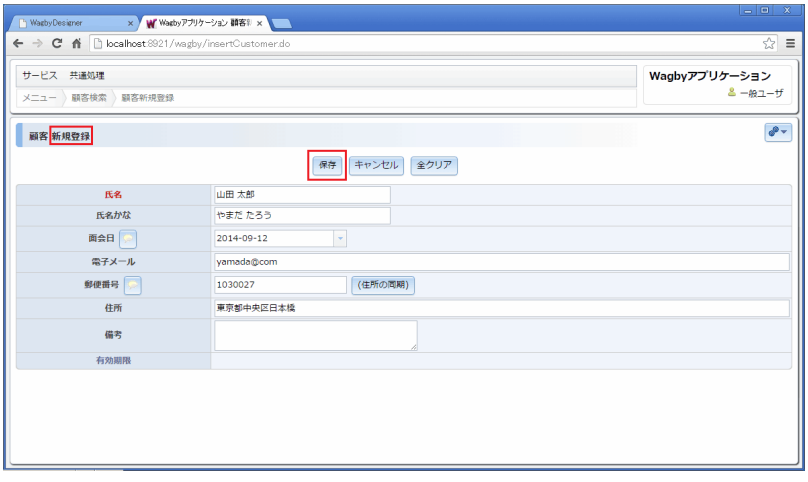


図 顧客データの登録

登録後、「削除」ボタンを押下します。この時点で、削除ボタンは論理削除として動作する仕様となっており、ボタン押下によって「有効期限」項目に日付時刻をセットして更新という動作になります。

図 論理削除機能を実行する

暗黙条件を設定し、「有効期限」項目の値が空白(null)のみ検索するようにしています。そのため論理削除したデータは検索されません。

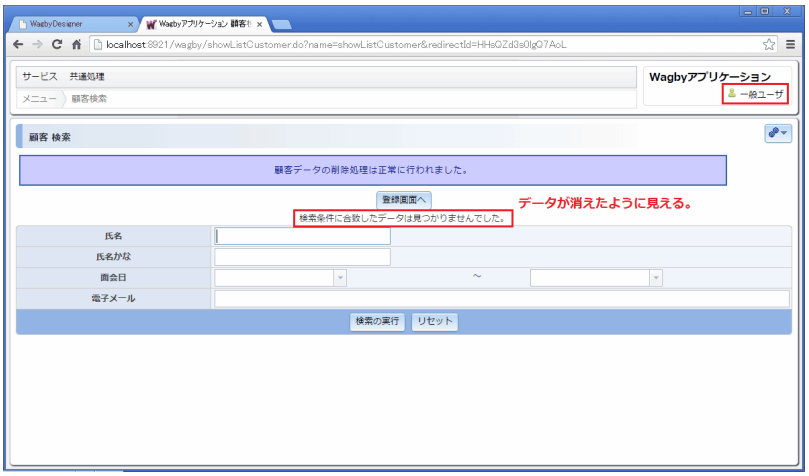


図 論理削除されたデータは不可視となる（暗黙条件の設定）

業務管理者

続いて業務管理者でログオンします。暗黙条件は解除される設定のため、データは検索できます。

図 論理削除されたデータも検索できる

詳細画面を開きます。「有効期限」項目に、論理削除を行った日付時刻がセットされていることがわかります。

図 有効期限項目の確認

有効期限項目の名前は任意ですが、型は「文字列型」としてください。保存する値は日付時刻（を意味する文字列）になります。

物理削除を行う

業務管理者は物理削除を行うことができます。通常の削除ボタンは論理削除となっているため、新たに「物理削除」ボタンを用意します。

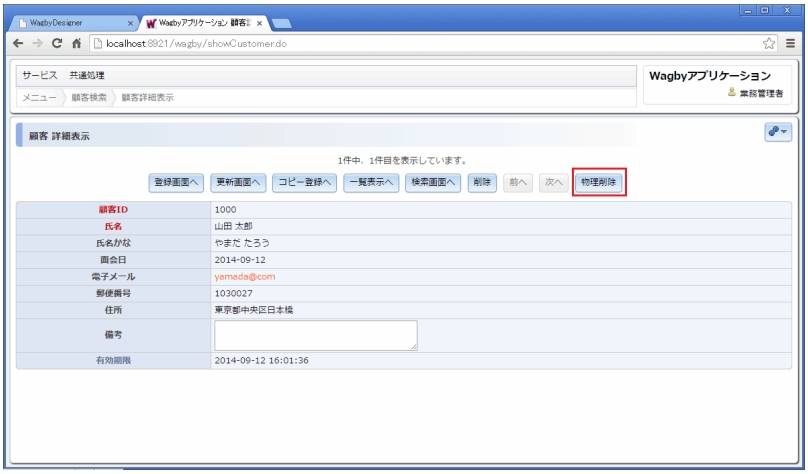


図 物理削除ボタンを用意する

ボタン押下により、データベースからの削除が行われます。

図　物理削除の実行

定義方法

　ここでは「顧客」モデルに論理削除機能を設定します。図に示すように、「有効期限」項目を用意します。　（項目名、項目IDは任意ですが、型は文字列型として用意してください。）

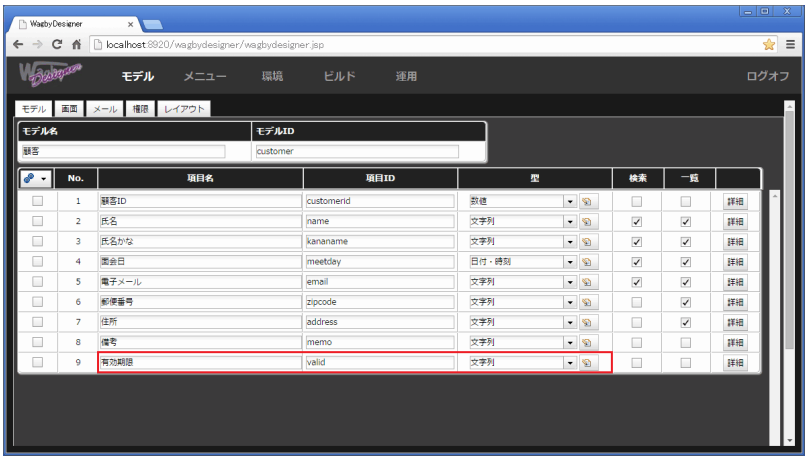


図　顧客モデルに有効期限項目を用意する

モデル参照の「無効判定」項目として利用できます。 今回用意した「有効期限」項目は、モデル参照の無効判定項目としても利用できます。 無効判定の使い方は"選択肢モデル > 選択肢を無効にする"をお読みください。

有効期限項目の設定

　この項目は暗黙条件として用います。そのためデータベースに保存するようにします。

図 データベースに保存する

また、入力できないよう、読込専用とします。（本番運用時には隠し設定としてもかまいません。）

図 読込専用と設定する

暗黙条件を設定します。「値が "null" と等しい」というルールにします。 ただし適用除外権限を同時に設定します。権限名は "pdelete" となります。(後述)[259]

図 暗黙条件の設定

論理削除を有効にする

「画面 ＞ その他 ＞ 削除」を設定します。 「データの削除を論理削除にする」を有効にします。あわせて、論理削除判定用の項目を設定します。これは図で用意した項目になります。

[259] これにより、データ検索時には常に SQL の where 句に "where valid IS NULL" が付加されます。

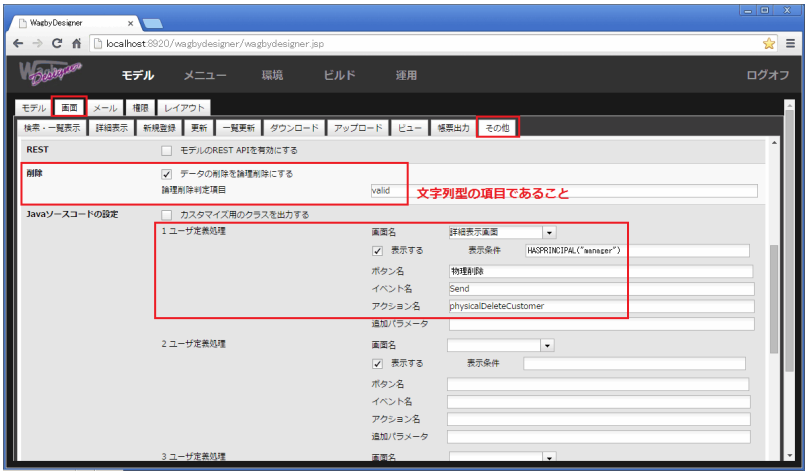


図 論理削除を有効にする

物理削除ボタンを用意する

図の画面で、さらに物理削除ボタンをオリジナルボタンとして用意します。

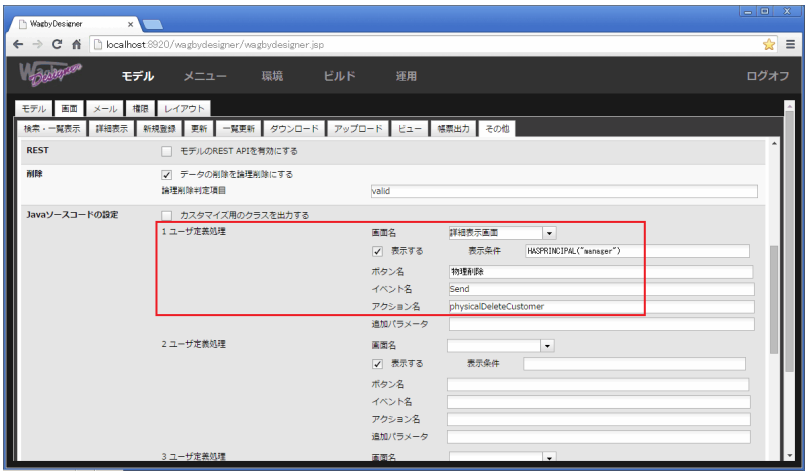


図 物理削除ボタンの設定

項目	説明	入力例
画面名	"詳細表示画面"となります。	詳細表示画面
表示する	有効にします。	表示する
表示条件	業務管理者プリンシパル "manager" を持っている場合に、物理削除ボタンを表示するようにします。	HASPRINCIPAL("manager")
ボタン名	任意の名前を指定します。	物理削除
イベント名	物理削除画面に対してSendイベントを送信する必要があります。	Send
アクション名	物理削除画面は physicalDelete<モデル名> となります。(*1)	physicalDeleteCustomer

(260)

(260) 1 キャメル記法を適用します。例えばモデル名が my_funny_valentine の場合は physicalDeleteMyFunnyValentine になります。

物理削除権限を用意する

新しいパーミッション "pdelete" を用意します。これは固定名です。日本語名は任意ですが、ここでは"物理削除"とします。

このパーミッションを保有するプリンシパルを用意します。一般ユーザと区分し、ここでは業務管理者(manager)を用意しました。

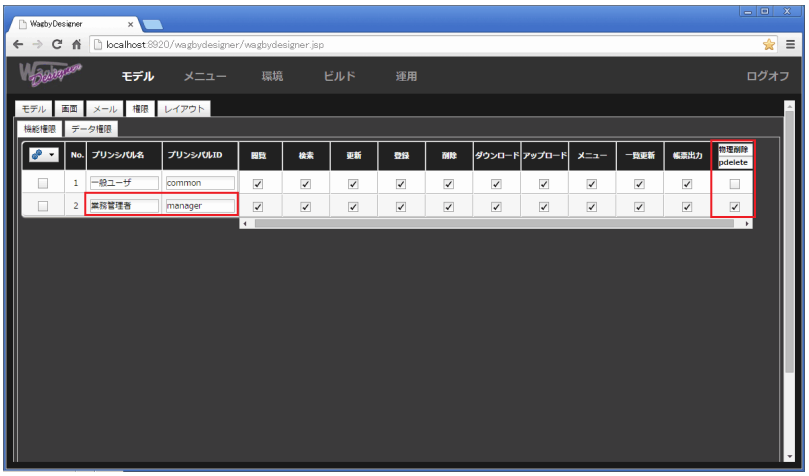


図 権限設定

- 図,図で用意したアカウントは、このプリンシパルを持つように設定されています。
- 図の暗黙条件設定では、適用外の権限（パーミッション）として、"pdelete"を設定してます。
- 図のボタン追加設定では、このプリンシパル"manager"を持っている場合にボタンを表示すると設定しています。

判定項目を独自に設定する

標準では、論理削除の判定項目は（削除時に）日付時刻（を示す文字列）が格納されます。この項目がnull なら削除されていない、とみなします。

このルールをカスタマイズすることができます。例として「0:正常、1:削除」というルールを設定します。

暗黙条件の設定

図の設定を"0に等しい"とします。

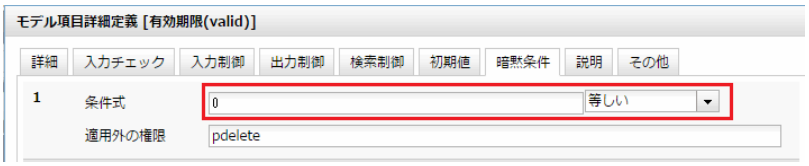


図 暗黙条件の設定

スクリプトの設定

論理削除時に動作するスクリプトを記述します。これは「画面 > スクリプト > ヘルパ」の実行タイミングに"論理削除"を指定します。

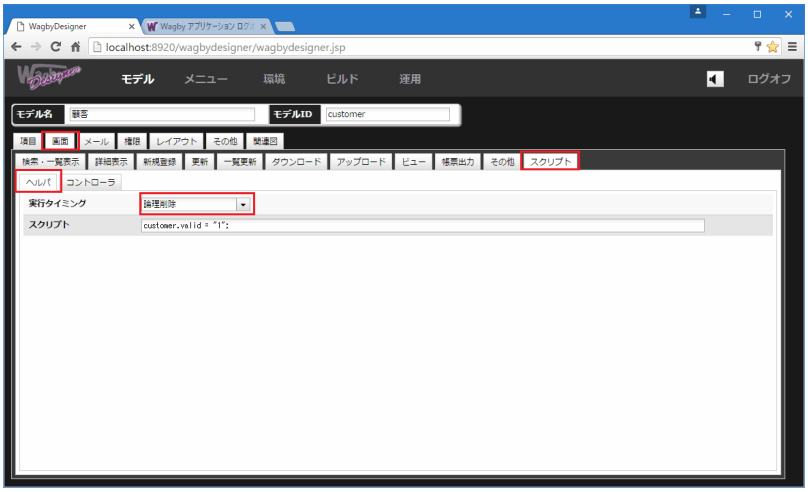


図 論理削除時に動作するスクリプトの設定

今回は次のようなコードになります。

```
customer.valid = "1";
```

仕様・制約

詳細画面が必要

本設定を用いる場合、対象モデルの詳細画面を用意することが必要です。

未対応の機能

- CSV・Excelアップロード更新には未対応です。アップロード更新での削除は「物理削除」になります。
- REST API に対応していません。現在は物理削除のみです。

ボタンの配置

ボタンの表示順序を制御することはできません。オリジナルボタンは常にボタン表示部の右側に配置されます。

スクリプトが優先される

「画面 ＞ スクリプト ＞ ヘルパ」の実行タイミング "論理削除" のスクリプトを用意した場合、こちらが優先されます。（この場合、論理削除項目に日付時刻をセットするJavaコードは生成されません。）

登録データの一時保存

例

ここでは2つのアカウントを用意して説明します。

アカウント	プリンシパル	説明
admin	業務管理者	データの一時保存を行う
user1	一般ユーザ	データの検索を行う

1. adminでログオンします。図の画面にて、一時保存ボタンを用意しています。新規登録時、一時保存ボタンを押してデータを保存します。

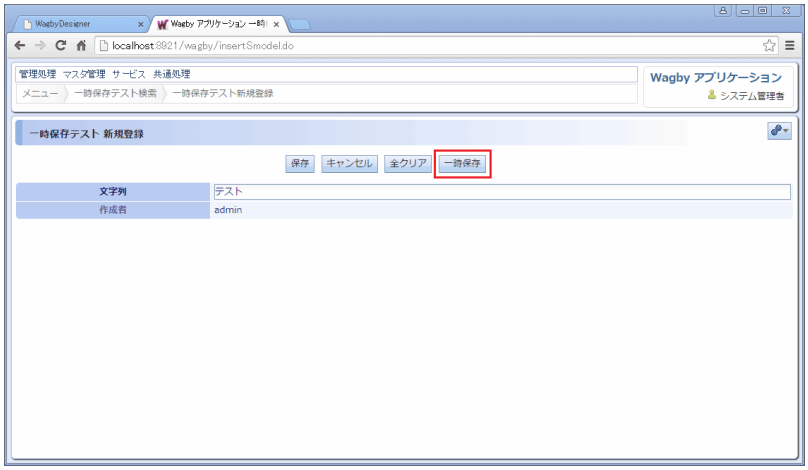


図 「一時保存ボタン」を押す

2. 「保存状態」という項目は自動設定されます。ここでは〝一時保存〟となっています。

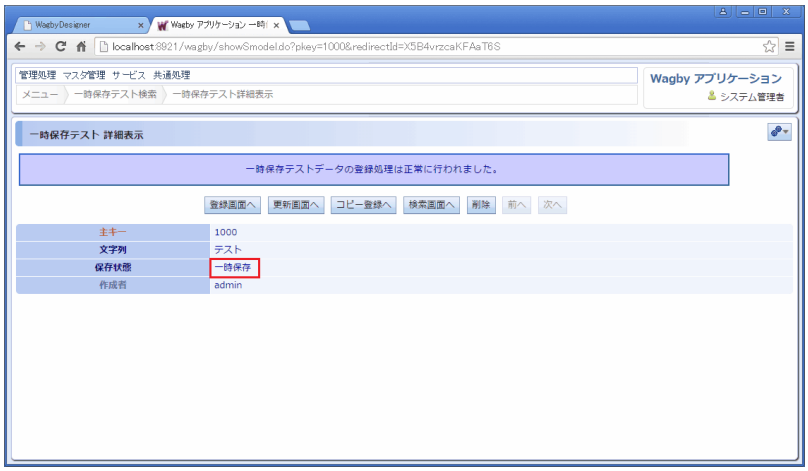


図 一時保存状態

3. 検索条件の「保存状態」は 〝通常保存〟 されたデータを検索するようになっているため、ここではデータが見つかりません。

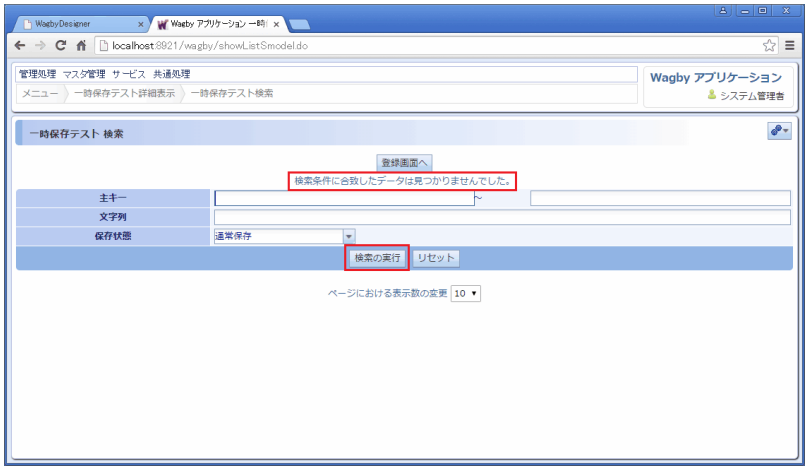


図 通常保存されたデータの検索

4. 検索条件の「保存状態」を 〝一時保存〟 にするとデータが見つかります。 ただし、この場合は自分（ログオンユーザ）が一時保存したデータのみが対象となります。他ユーザーによる一時保存データは検索されません。

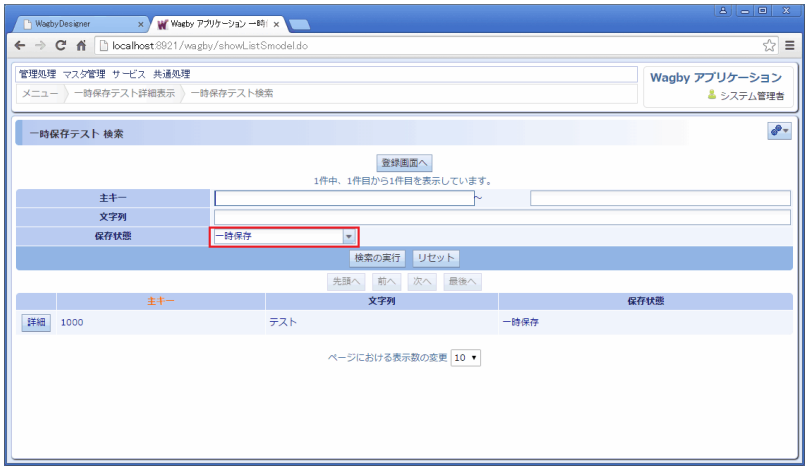


図 一時保存されたデータの検索

ログオンアカウントを user1 に切り替えると、検索できないことがわかります。

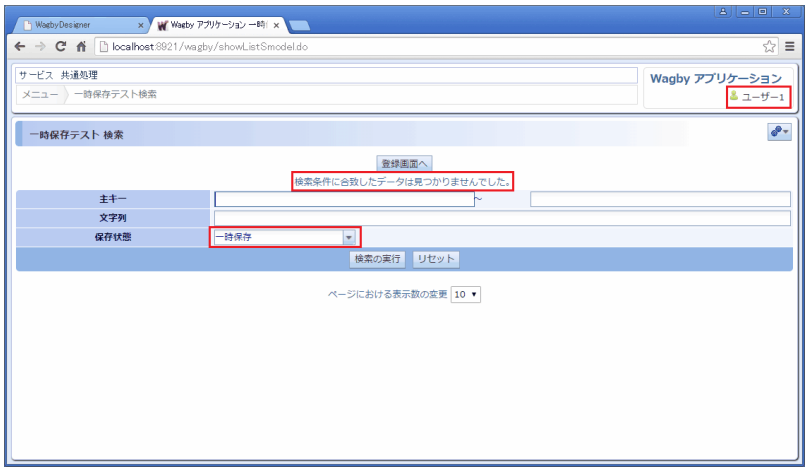


図 別のアカウントでは検索されない

5. 一時保存されたデータを更新します。今度は「保存」ボタンを押下します。

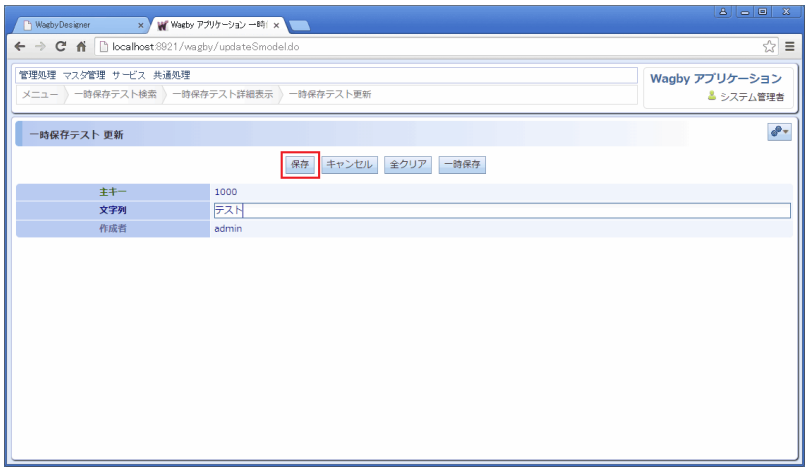


図 「保存」ボタンを押下する

6. データは ″通常保存″ となります。

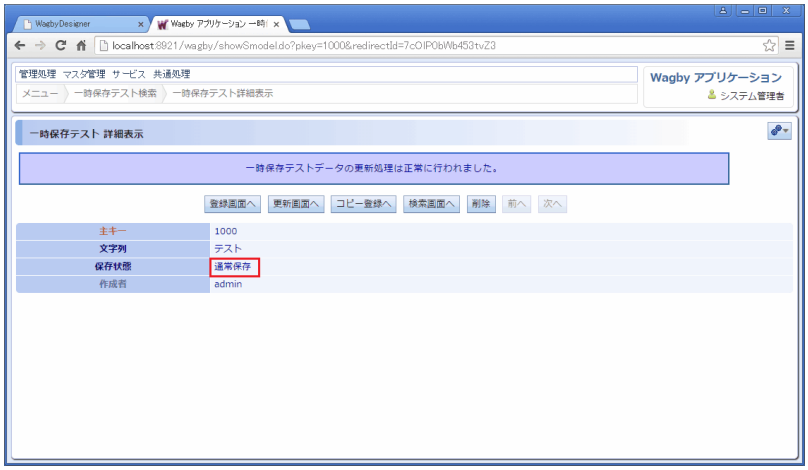


図 通常保存状態

7. 通常の検索処理でデータが見つかるようになります。

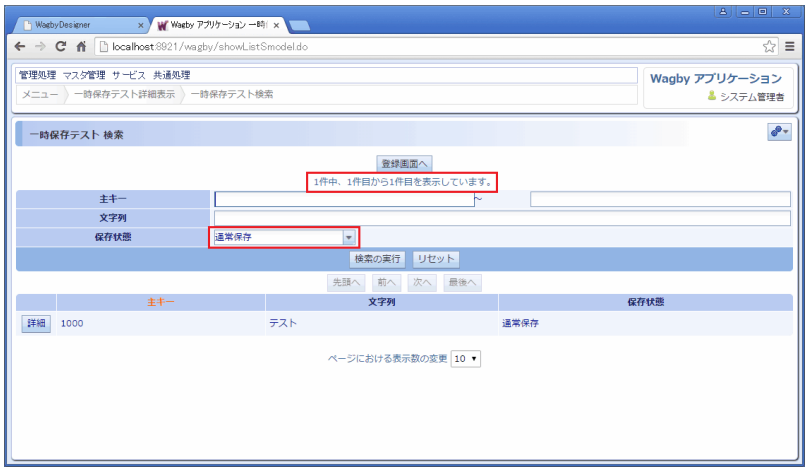


図 通常保存されたデータの検索(2)

別のアカウントからも検索できるようになります。

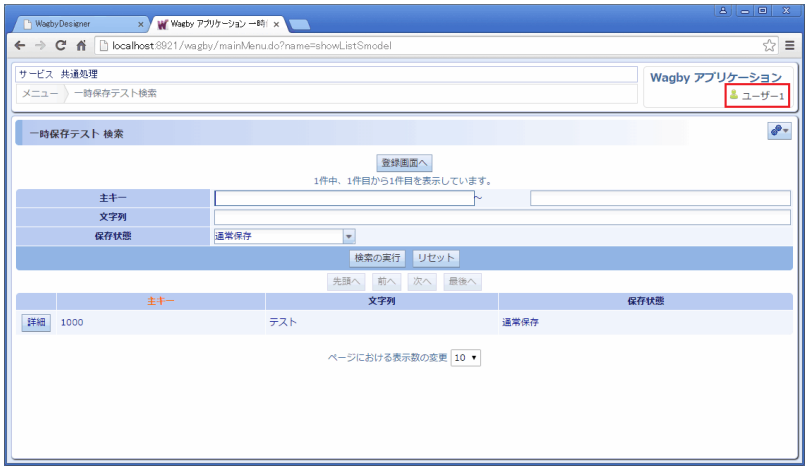


図 アカウントuser1からの検索

定義方法

本機能は複数の設定を組み合わせて実現します。 上で説明したサンプルは次の二つのモデルから実現されています。

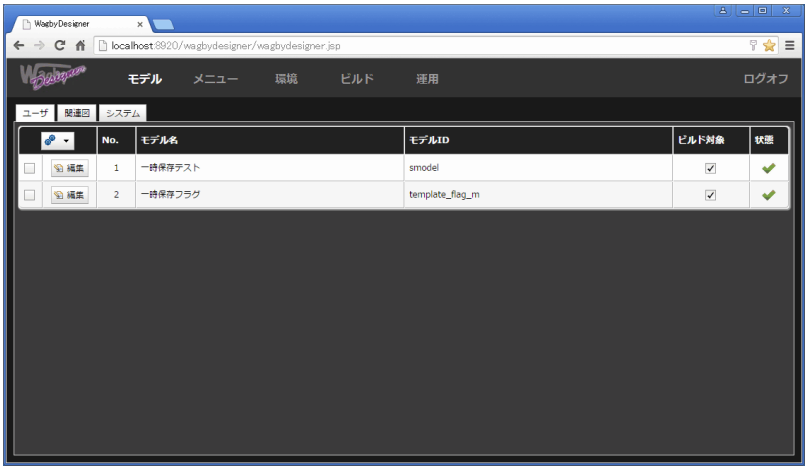


図 サンプルアプリケーションのモデル

一時保存フラグモデル

「一時保存フラグ」モデルは、状態を示す選択肢モデルです。ここでは図のように ″一時保存″ と ″通常保存″ を用意します。

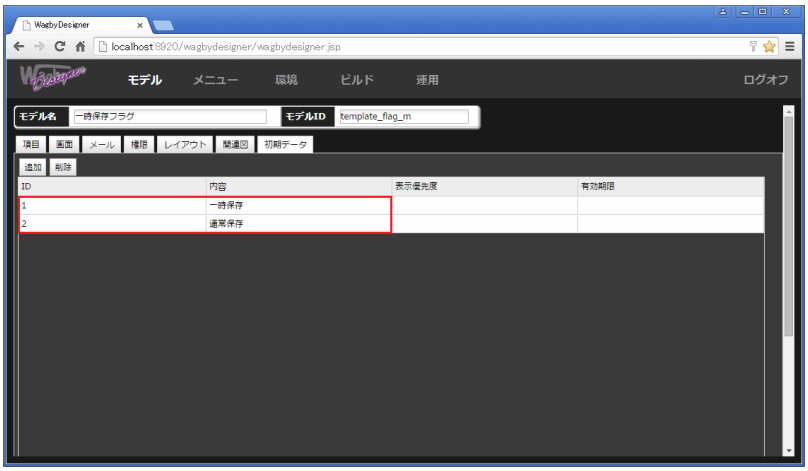


図 保存状態を表すコード

一時保存テストモデル

テストで用いたモデルを図に示します。「保存状態」項目は、図の一時保存フラグモデルを参照します。

図 一時保存テストモデルの定義

「一時保存」ボタンを用意する

(1)「画面 > その他 > Javaソースコードの設定」を開きます。

(2)「カスタマイズ用のクラスを出力する」を有効にします。

(3) 新規登録画面と更新画面にそれぞれ「一時保存」ボタンを用意します。

図 一時保存ボタンを用意する

ポイントは、イベント名 "TemplateSave" の横にある拡張指定 [@method=insert] および [@method=update] です。これによって本ボタンは（従来の）「保存」「更新」と同じ役割を担うようになります。

(261) (262)

「保存状態」項目

「保存状態」項目は、本機能を制御するために用意したものです。はじめに初期値を設定します。

図 「保存状態」項目の初期値

登録時および更新時の初期値は次のとおりです。

(261) ボタン名は任意です。イベント名は "TemplateSave" としてください。 大文字小文字は区別されます。

(262) ボタンの表示位置は固定です。追加したオリジナルボタンは、右側に配置されます。

```
IF(EXACT(EVENT(),"TemplateSave"),1,2)
```

EVENT　関数は、オリジナルボタンのイベントを受け取ることができます。一時保存ボタン押下時は　"1"を、そうでない場合は "2" をセットしています。これは図におけるコード "一時保存" "通常保存" に対応しています。

検索画面表示時の初期値は "通常保存" のコード "2" をセットしています。

```
2
```

また、この項目は入力時に隠し項目としています。

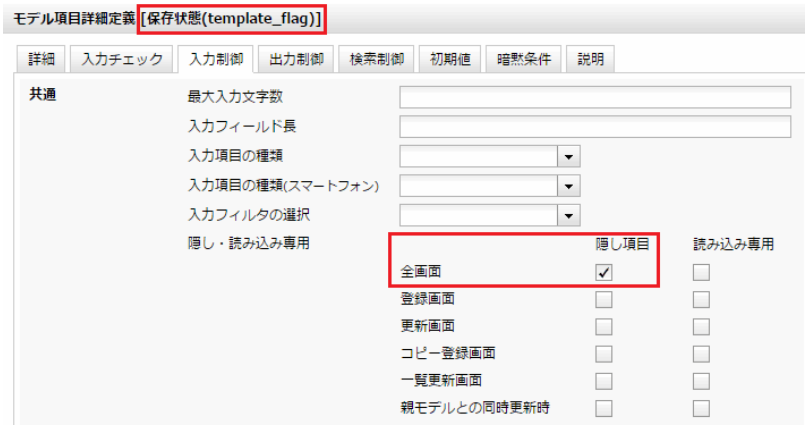


図 隠し項目の設定

(263)

「作成者」項目の設定

「作成者」項目もまた、本機能を制御するために用意したものです。はじめに初期値を設定します。

検索値を設定することで、本項目の検索条件は入力された値に関わらず、ここで指定された値が用いられるようになります。

図 作成者項目の検索初期値の設定

```
IF(${template_flag#id}==1,USERNAME(),null)
```

この式は、検索条件項目である「保存状態」の　ID　が　1、すなわち一時保存の場合にのみ USERNAME 関数の値である「ログオンアカウント」を設定するようになっています。一時保存の場合、自動的に自分のログオンアカウントで検索条件が絞込まれるようになります。

(263) 実際の運用では、表示時も隠し項目とするとよいでしょう。

(264)

この設定により、本項目は利用者による検索条件入力が無視されます。よって、本項目は検索時に非表示とします。

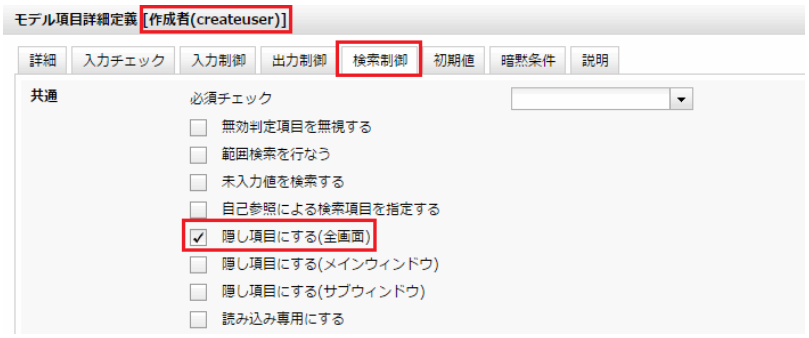


図 「作成者」項目は検索時に隠し項目とする

確認画面を利用する場合

例

登録確認画面を用意した例を紹介します。「一時保存」ボタンを押下します。

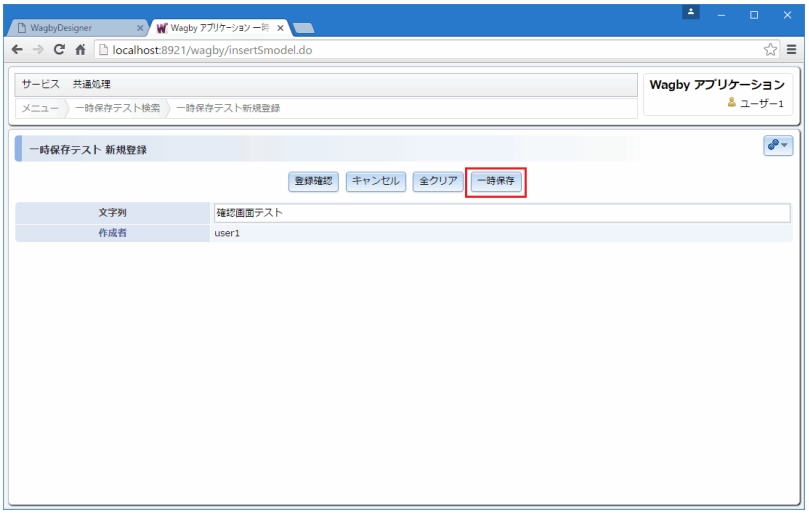


図 一時保存ボタン

確認画面に遷移します。ここで「登録する」ボタンを押下します。

(264) データベースへの SQL 検索時、WHERE 句にログオンアカウント名が付与されます。

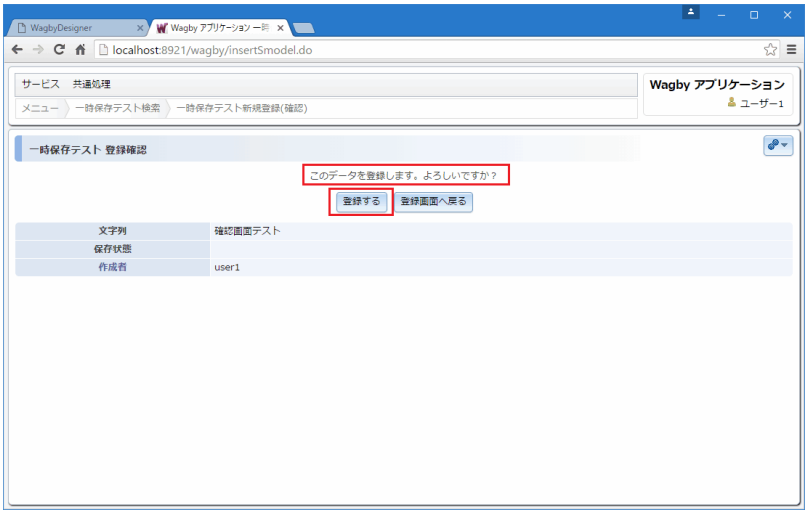


図 確認画面に遷移する

保存状態が〝一時保存〟となっていることがわかります。

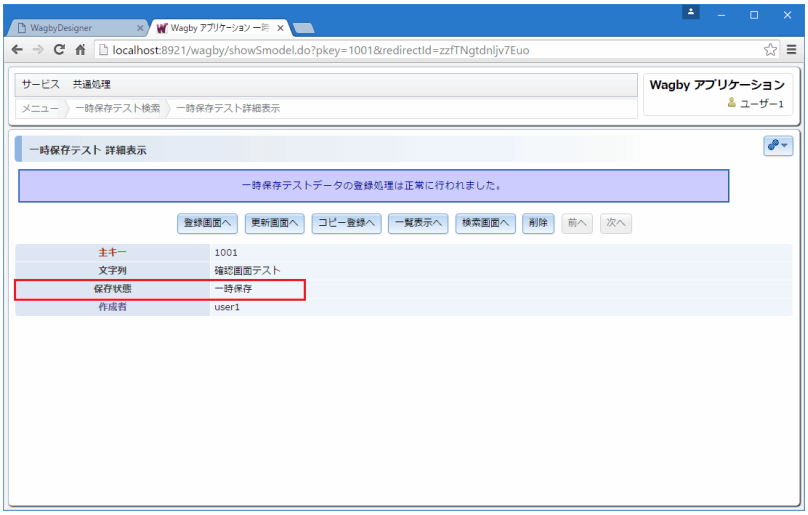


図 一時保存状態となっている。

定義方法

確認画面の設定

「画面 ＞ 新規登録」で「確認画面を表示する」を有効にします。

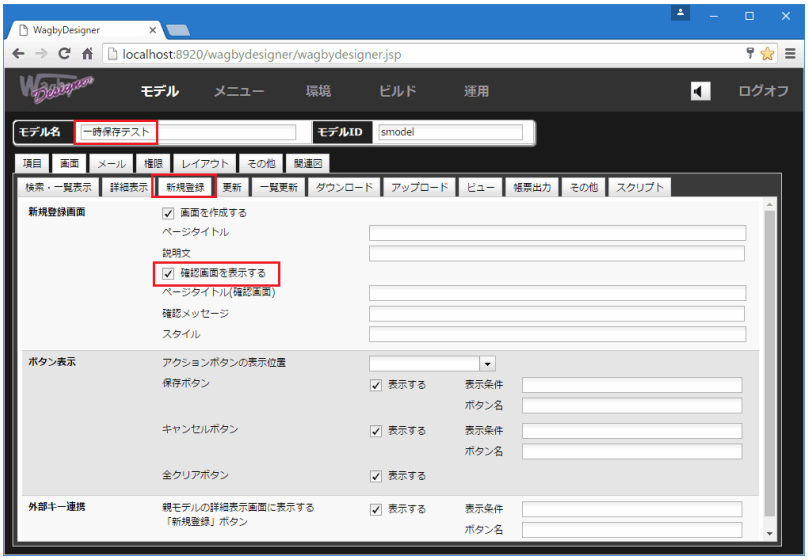


図 「確認画面を表示する」を有効にする

隠し項目「イベント」を用意する

ボタンのイベントを保持する隠し項目「イベント」を用意します。確認画面を経由した場合、その前のボタンで「一時保存」ボタンを押したかどうかを記録することが目的です。

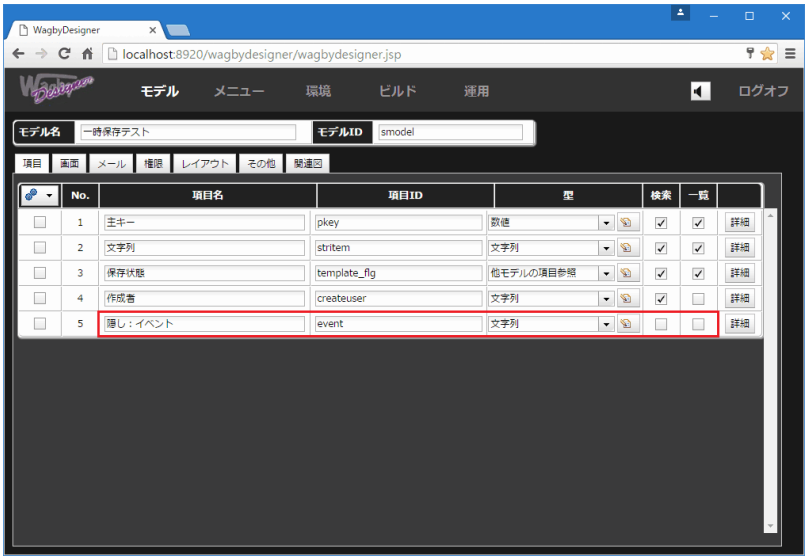


図 隠し項目「イベント」

この項目は入力時、出力時それぞれ隠し項目とします。

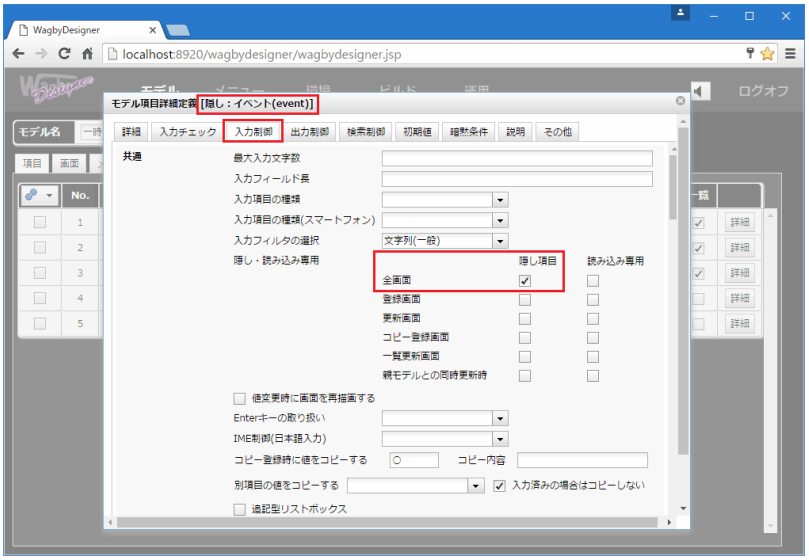


図 入力制御>隠し項目の設定

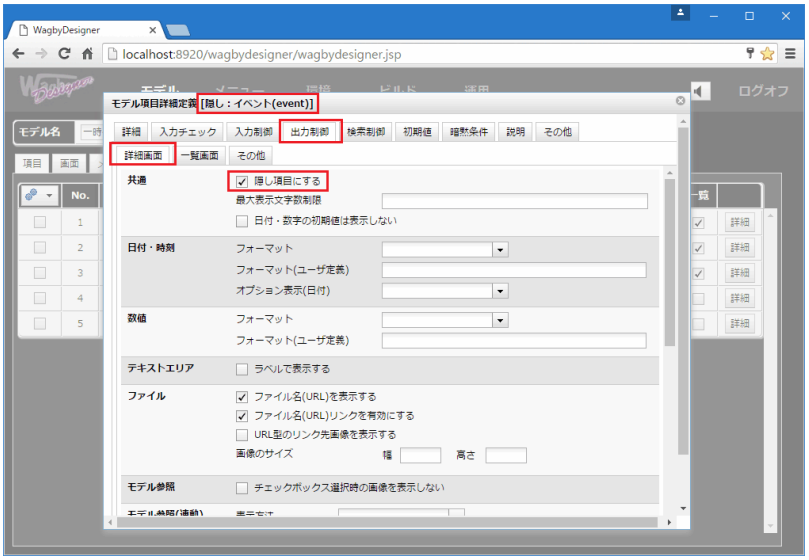


図 出力制御>隠し項目の設定

式の設定

ボタン押下により発生したイベントが「一時保存」("TemplateSave") または「確認」("Confirm") のいずれかであればそれを保持します。それ以外の場合は、現在保存している値を保持するために Me() を使います。

```
IF(
  OR(
    EXACT(EVENT(),"TemplateSave"),
    EXACT(EVENT(),"Confirm")
  ),
  EVENT(),
  Me()
)
```

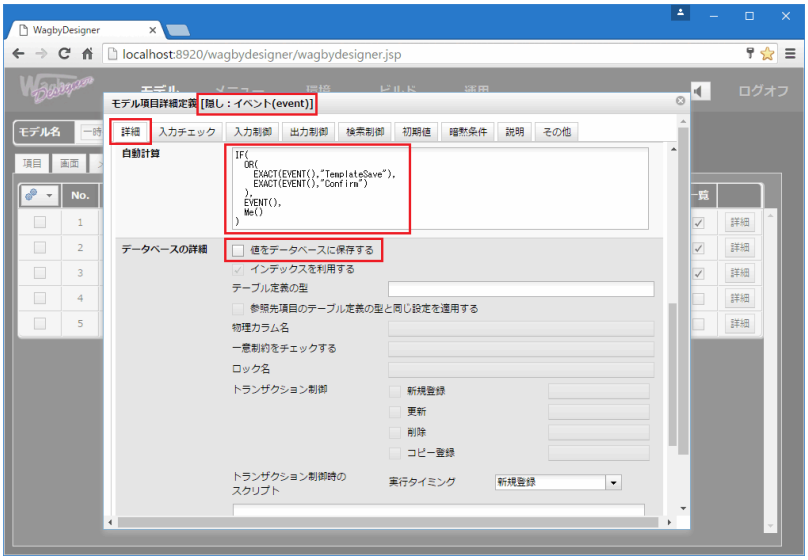


図 隠し項目イベントの式

「保存状態」項目の初期値

図の式を次のように修正します。押下ボタンによって発生したイベントが"TemplateSave"か、または隠し項目 event に保持されていたイベントが"TemplateSave"のいずれかを考慮するようにします。

```
IF(
  OR(
    EXACT(EVENT(),"TemplateSave"),
    EXACT(${event}, "TemplateSave")
  )
  ,1
  ,2
)
```

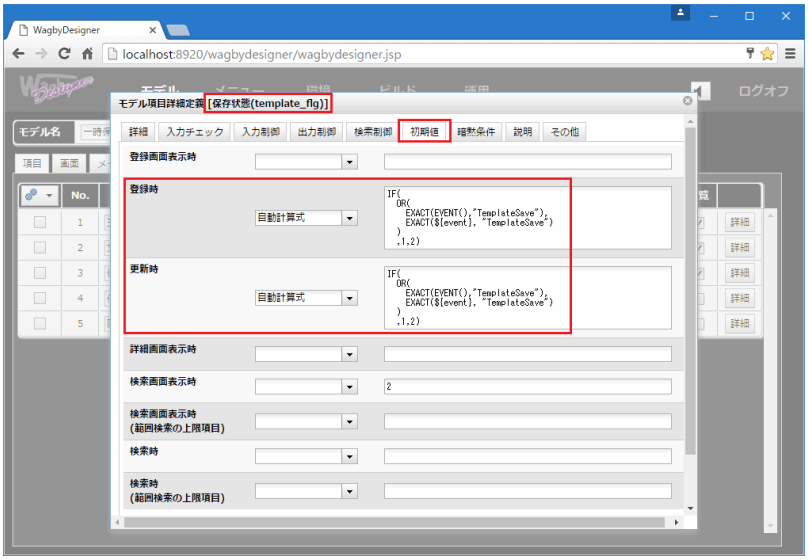


図 「保存状態」項目の初期値を変更する

ワークフローとの関係

ワークフローを設定したモデルに本設定を適用すると、他ユーザーによる一時保存データは検索されなくなります。ワークフローの申請をグループとしていたとき、グループに所属する他ユーザーから検索されない限り、ワークフローの「申請」ボタンを押すことはできません。このため、一時保存したデータはワークフローを開始しないという運用を行うことができます。

(265)

設計上の注意点

この一時保存機能は、データベースへの登録処理を伴います。（他者に閲覧できないという条件を組み込むことで、運用での一時保存を実現しています。）

そのため必須チェックを指定した項目は、一時保存であっても必須となります。必須チエックはデータの論理的整合性を担保するため、データベースのテーブルに NOT NULL 制約を適用するためです。

一覧表示画面から詳細画面（サブウィンドウ）を開く

例

顧客モデル一覧表示画面に「確認」という独自ボタンを用意した例です。

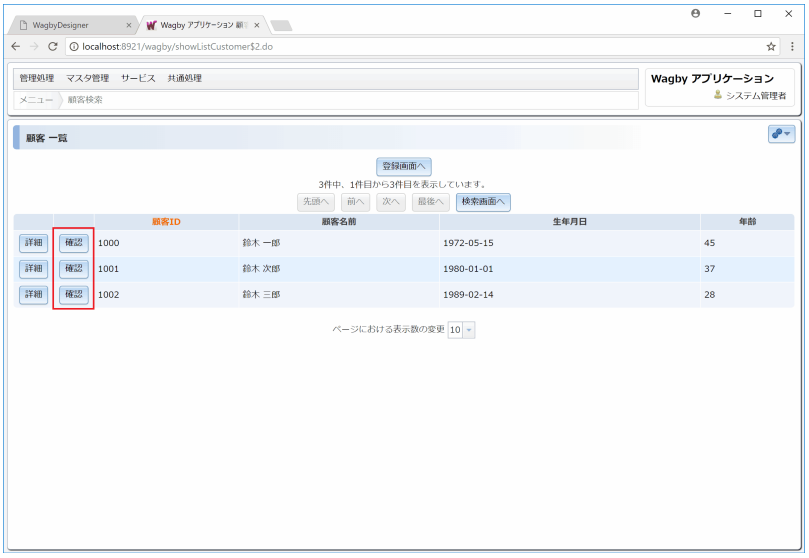


図 「確認」ボタンを用意した例

「確認」ボタンを押下すると、詳細画面の内容がサブウィンドウで表示されます。

(265) 一時保存されたデータも登録者（自ユーザー）は検索できるため、登録者がワークフローの申請を行うことはできます。

図 詳細画面サブウィンドウ

行毎（データ毎）に複数のサブウィンドウを開くことができます。（新しいサブウィンドウが、上に重なる形で表示されます。）

定義方法

顧客モデルの定義例を図に示します。

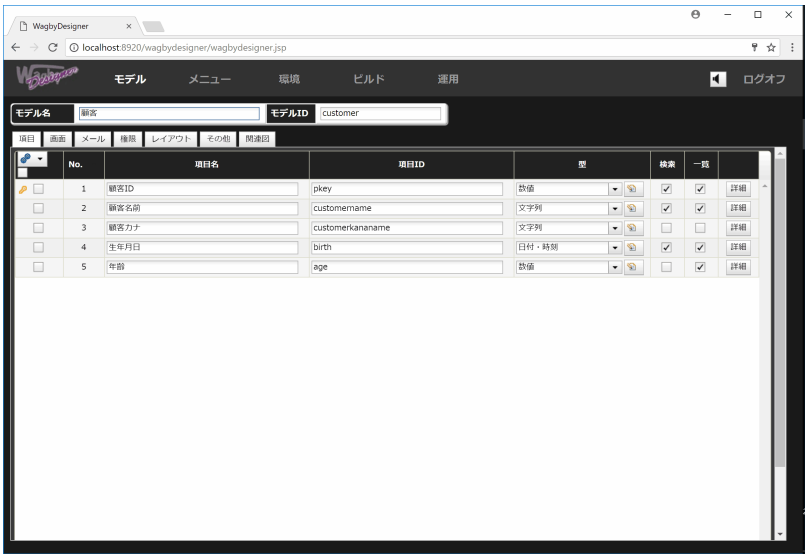


図 例に用いた顧客モデル

サブウィンドウを用意する

「画面 ＞ 一覧表示画面 ＞ 一覧表示の行ボタン押下で詳細画面のサブウィンドウを表示する」を有効にします。

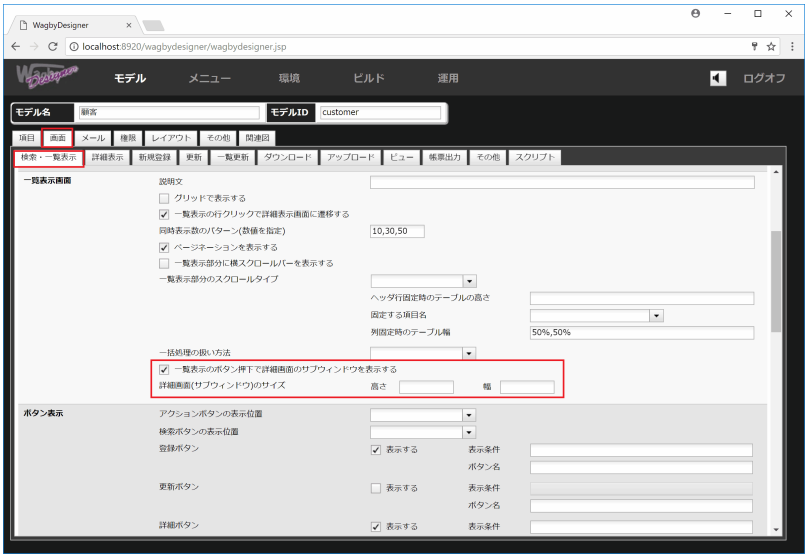


図 一覧表示の行ボタン押下で詳細画面のサブウィンドウを表示する設定

サブウィンドウの高さと幅

サブウィンドウの高さと幅を指定することができます。単位はピクセルです。省略時は「環境 > 詳細 > 検索画面(サブウィンドウ)のサイズ」を用います。

独自ボタンを用意する

「画面 > その他 > Javaソースコードの生成」欄で、独自ボタンを設定します。

項目	説明	入力例
画面名	どの画面にボタンを追加するかを指定します。ここでは"一覧表示画面(行毎)"とします。	一覧表示画面(行毎)
表示する	有効にします。	(チェック)
表示条件	常にボタンを表示する場合、本欄は空白とします。条件については"ボタン表示条件"をお読みください。	
ボタン名	任意の名前を指定します。	確認
イベント名	ここでは ShowSubwindow と記述します。大文字小文字を区別します。	ShowSubwindow
アクション名	遷移する画面を指定します。通常は "show" + モデルID となります。show は小文字です。モデルIDはキャメル記法を使います。	showCustomer
追加パラメータ	空白とします。主キーは自動的に渡されます。	(空白)

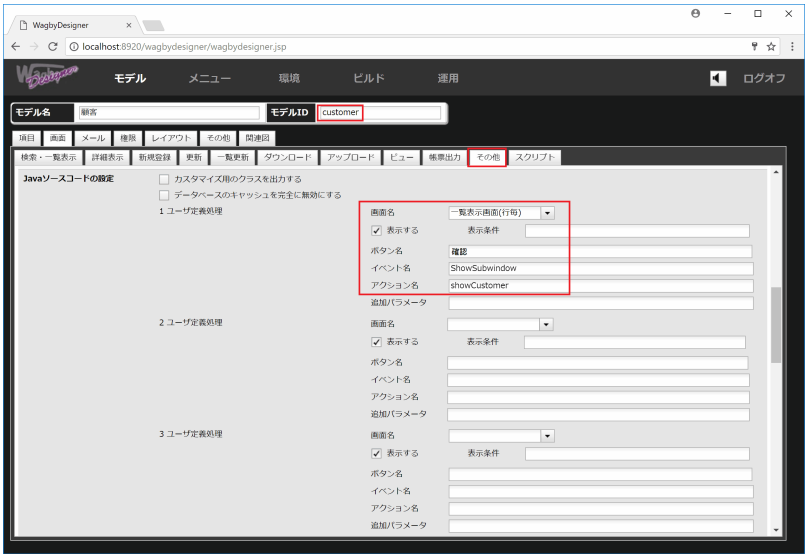


図 独自ボタンを設定する

ウィンドウの高さ、幅、位置、名称をボタン毎に指定する

表示するサブウィンドウの高さと幅は、図の設定欄で指定することができます。この場合、サブウィンドの高さと幅は共通で利用されます。

もう一つの方法として、用意するボタン毎にサブウィンドウの高さ、幅、位置およびサブウィンドウの名前を指定することができます。図の独自ボタン設定で、追加パラメータに次の情報を付与します。

```
__width=幅&__height=高さ&__top=左上隅を原点としたY座標値&__left=左上隅を原点としたX座標値
&__wname=名前の接頭語
```

例えば次のように記述します。

```
__width=300&__height=200&__top=100&__left=100&__wname=s
```

値の単位はピクセルです。固定値のみを記述することができます。この例だと左上隅を原点として、上端から100ピクセル、左端から100ピクセルの位置に、幅300ピクセル、高さ200ピクセルのサブウィンドウを開きます。名前は 's' を接頭語とし、そのあとに主キーが付与されます。（例 主キーが1000のデータの場合、s1000となります。）

(266)

一覧表示の行毎に用意するボタンを複数、配置し、それぞれのボタン毎に違う位置でサブウィンドウを開くことができます。このモデルの「サブモデル」を用意し、アクション名にサブモデルを指定することで、同じデータに関する部分的な情報をそれぞれのサブウィンドウで表示することができます。

仕様・制約

- サブウィンドウは詳細画面表示のみとなります。更新や削除といった機能を設定することはできません。
- この設定を行うためには、モデルの「詳細画面を作成する」を有効にする必要があります。
- ウィザードには対応していません。表示する項目を絞ったサブモデルを用意し、これを表示させるようにするとよいでしょう。（独自ボタンの「アクション名」で、サブモデルへ遷移させるようにします。）
- 繰り返しコンテナや外部キー子モデル同時表示には対応していません。これらの情報が含まれる場合は、あらかじめ繰り返しコンテナや外部キー子モデルを除いたサブモデルを用意しておき、アクション名にサブモデルを指定するようにしてください。(*1)

(266) 項目の値を参照することはできません。例えば ${item1} と書くことはできません。

- ファイル型に対応していません。ファイル型を含めた場合、常に同じファイル（画像）が表示されます。(*2)
- 一覧表示（グリッド）には対応していません。(*3)
- 任意のモデルが、このモデル（本マニュアルにおける「顧客モデル」）を「モデル参照（検索）」で参照する設定を行う必要があります。(*4)
- 行ごとに複数のボタンを配置したとき、異なる行のボタンを押下するとサブウィンドウは別々に開きます。しかし同一行の同一ボタンは複数回押下してもサブウィンドウは複数個、開くことはありません。同じボタンを押した場合は同一のサブウィンドウになります。

(267) (268) (269) (270)

一覧表示からの選択（独自ボタン利用）（スクリプト）

概要

一覧表示画面で選択したデータの主キーを、スクリプトで受け取ることができます。開発者はこれらの（複数の）データに対して何らかの処理を行ってください。

この機能は一括更新に似ていますが、次の点が異なります。

- 一括更新スクリプトはデータ1件ごとに呼び出されます。スクリプトではデータ（正確にはストアモデル）の値を変更する処理のみを記述します。対して本機能は独自ボタン押下時に一回だけ呼び出されます。更新はもとより、それ以外のさまざまな業務処理を行うことができます。（例えば、選択された主キーをパラメータにして非同期処理を呼び出す、など。）
- 一括更新スクリプトは、更新処理自体は Wagby が行います。対して本機能で更新処理を行う場合、開発者がすべてを作成する必要があります。

設定方法(画面に表示されているデータを一括処理の対象とする)

ボタンを用意する

画面 ＞ その他 タブのJavaソースコードの設定欄の画面名の選択肢から ″一覧表示画面(一括)″ を指定します。

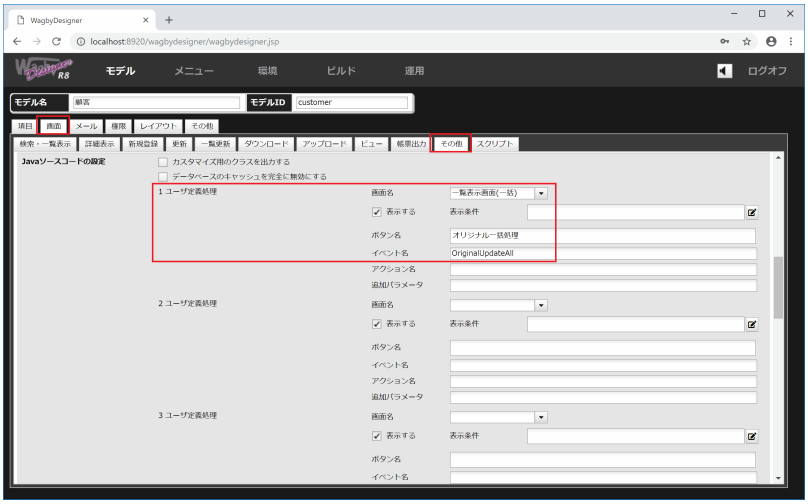


図 独自ボタン

- ボタン名は任意です。

(267) 1 この制約は将来の Wagby のバージョンアップで解決するよう、対応を継続します。
(268) 2 (同上)
(269) 3 (同上)
(270) 4 この制約はR8.4.1(Patch2)およびR8.4.2以降のWagbyで解消されました。

- イベント名は、末尾に "All" を含めてください。後述するスクリプトのファイル名に含まれます。またWagbyが標準で提供するスクリプトファイル名と重複しないようにしてください。上の例にあるように、接頭語 "Original" を付与するといった工夫を行うとよいでしょう。
- アクション名は空白とします。自画面に戻すためです。
- 追加パラメータも通常は指定する必要はありません。

一括指定のためのチェックボックスを有効にする

Wagbyが標準で提供する、何らかの一括処理を有効にします。ここでは「一括削除」を有効にしておきます。

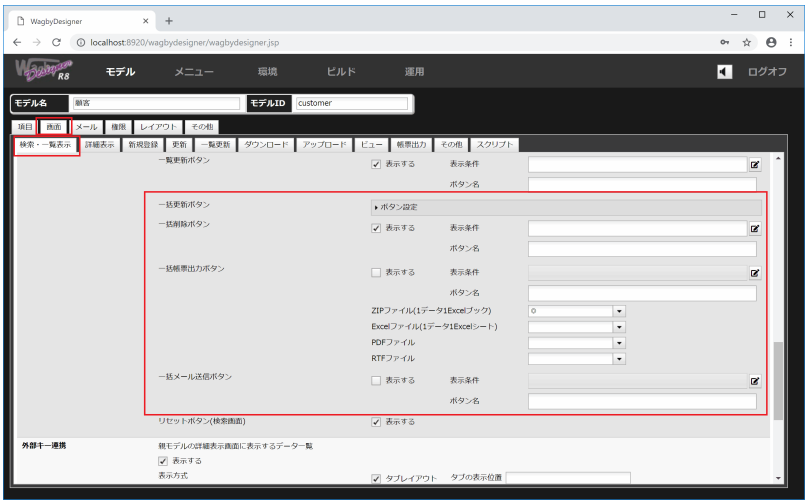


図 いずれかの一括処理を有効にする

(271)

動作例

ビルドしたアプリケーションの画面を確認します。独自ボタン（ここでは「オリジナル一括処理」という名前のボタン）が用意されています。データを選択していない状態では、ボタンを押下することはできません。

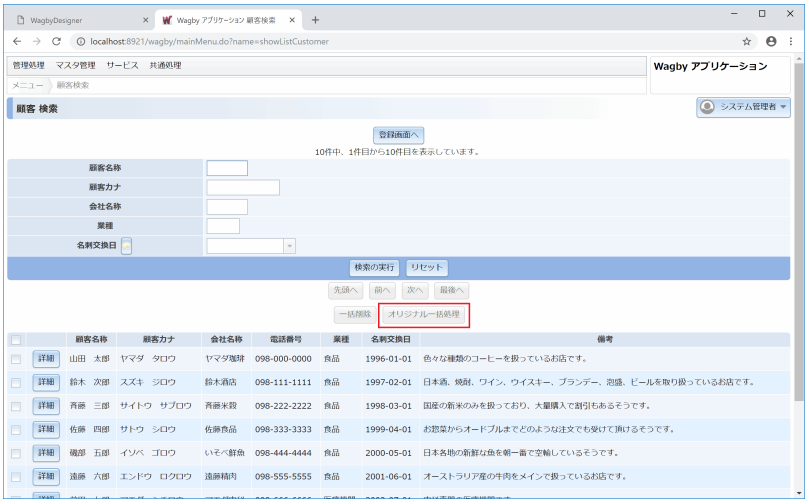


図 「オリジナル一括処理」ボタン

データを選択すると、ボタンを押下できるようになります。

(271) 一つ以上の一括処理を有効にしてください。これによって一覧表示画面の各行の左端に、データを選択するためのチェックボックスが用意されます。

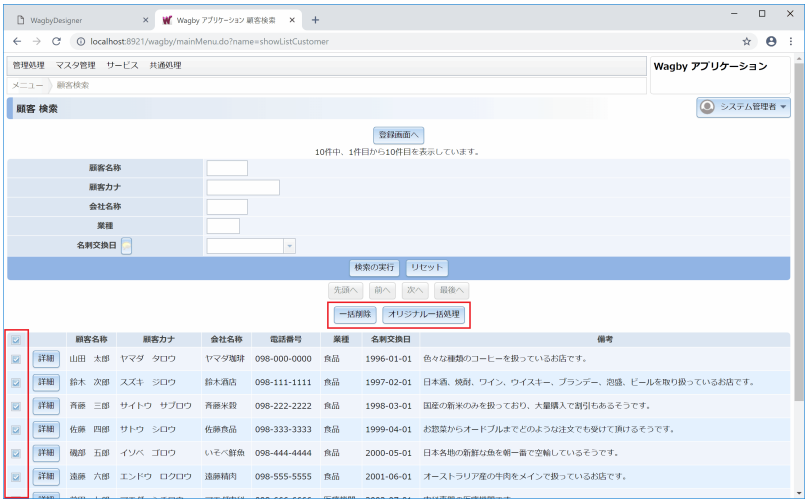


図 ボタン押下が可能になった

スクリプト

「オリジナル一括処理」ボタンのスクリプトを用意することができます。

このスクリプトファイルは Wagby Designer では作成できません。開発者は直接、WEB-INF/script/<モデルID> フォルダ内にスクリプトファイルを用意してください。

例えばモデルIDが customer で、開発者が用意したボタンのイベントが OriginalUpdateAll のとき、スクリプトファイル ShowListCustomer_OriginalUpdateAll.js が呼び出されます。

スクリプトでは、選択されたデータの主キーの集合を取得することができます。

```
function process() {
  var DbShowListSelectableBaseController =
    Java.type("jp.jasminesoft.jfc.controller.DbShowListSelectableBaseController");
  var ids = p.request.getAttribute(DbShowListSelectableBaseController.SelectIdSetRequestName);
  print(ids);
}
```

上の変数 ids の実体は java.util.Set です。1つ以上の値（主キー）を文字列型で保持しています。

開発者は得られた主キーを用いて、EntityService から値を取得したり、更新するといった業務ロジックを記述することができます。 例を示します。

```
function process() {
  var ExcelFunction = Java.type("jp.jasminesoft.util.ExcelFunction");
  var entityService = p.appctx.getBean("CustomerEntityService");

  var DbShowListSelectableBaseController =
    Java.type("jp.jasminesoft.jfc.controller.DbShowListSelectableBaseController");
  var ids = p.request.getAttribute(DbShowListSelectableBaseController.SelectIdSetRequestName);
  var it = ids.iterator(); /* ids は java.util.Set なので java.util.Iterator を利用できる */
  while (it.hasNext()) {
    var id = it.next(); /* it.nextの戻り値は常に文字列型 */
    try {
      var obj = entityService.findById(ExcelFunction.TOINT(id));/* 主キーが整数型の場合の例 */
      print(obj);
    } catch (e) {
      print("error:"+e);
    }
  }
}
```

全データを一括処理の対象とする方式の場合

選択方式を "全データを一括処理の対象とする" とした場合、扱いが変わります。

この場合は"全てを選択・解除チェックボックス" の状態によって選択肢の意味が変わります。この状態を selectMode として内部で管理しています。

チェックボックスが ON のとき、selectMode は "__UNSELECT" になります。これは初期状態ですべてのデータが選択された状態です。スクリプトでは、選択を解除した主キーを取得することができます。

チェックボックスが OFF のとき、selectMode は "__SELECT" になります。これは初期状態ですべてのデータが未選択の状態です。スクリプトでは、選択した主キーを取得することができます。

```
function process() {
  var DbShowListSelectableAllBaseController =
    Java.type("jp.jasminesoft.jfc.controller.DbShowListSelectableAllBaseController");

  // "全てを選択・解除チェックボックス" の状態によって selectMode が変わる。
  // チェックONのとき、selectMode は "__UNSELECT" になる。
  // このときは選択されなかったものだけが ids に格納される。
  // チェックOFFのとき、selectMode は "__SELECT" になる。
  // このときは選択されたものが ids に格納される。
  var selectMode = p.appMap.get("showListModel1_selectMode");

  if (selectMode !== null && selectMode === "__SELECT") {
    var ids = p.appMap.get(DbShowListSelectableAllBaseController.selectIdSetAppmapName);
    ids.forEach(function(pkey){
       print("select:" + pkey);
    });
  } else {
    var ids = p.appMap.get(DbShowListSelectableAllBaseController.unselectIdSetAppmapName);
    ids.forEach(function(pkey){
       print("unselect:" + pkey);
    });
  }
}
```

- DbShowListSelectableBaseControllerクラスではなく、DbShowListSelectableAllBaseControllerクラスの値 selectIdSetAppmapName もしくは unselectIdSetAppmapName をキーにします。
- selectMode は p.appMap (実体は java.util.Map) の get メソッドを使って確認します。キー名は SCREENID 関数の戻り値 ("showList" + 画面名) に "_selectMode" が付与されたものになります。
- 同じく p.appMap の get メソッドを使って、キーの集合を取得します。この場合の集合は selectMode によって意味が変わります。選択肢れた値（主キー）または選択を解除された値（主キー）が文字列型として格納されます。

これを踏まえ、選択された主キーの集合を取得するスクリプトのサンプルを示します。対象モデルを model1 とし、主キー項目名を pkey (型は整数)としています。

```
function process() {
  var DbShowListSelectableAllBaseController =
    Java.type("jp.jasminesoft.jfc.controller.DbShowListSelectableAllBaseController");

  // "全てを選択・解除チェックボックス" の状態によって selectMode が変わる。
  // チェックONのとき、selectMode は "__UNSELECT" になる。
  // このときは選択されなかったものだけが ids に格納される。
  // チェックOFFのとき、selectMode は "__SELECT" になる。
  // このときは選択されたものが ids に格納される。
  var selectMode = p.appMap.get("showListModel1_selectMode");

  if (selectMode !== null && selectMode === "__SELECT") {
    var ids = p.appMap.get(DbShowListSelectableAllBaseController.selectIdSetAppmapName);
    ids.forEach(function(pkey){
       print("select:" + pkey);
    });
  } else {
    var ids = p.appMap.get(DbShowListSelectableAllBaseController.unselectIdSetAppmapName);
```

```
    var Model1C = Java.type("jp.jasminesoft.wagby.model.model1_c.Model1C");
    var FinderContext = Java.type("jp.jasminesoft.jfc.dao.FinderContext");

    var finderContext = new FinderContext();
    finderContext.setPageSize(-1);//無制限
    finderContext.setCondition(new Model1C());//空の検索条件セットによる全件取得
    finderContext.setCriteriaConverter(p.appctx.getBean("Model1CriteriaConverter"));
    var entityService = p.appctx.getBean("Model1EntityService");
    var results = entityService.find(finderContext);
    for (i=0; i<results.size(); i++) {
      var entity = results.get(i);
      var pkey = java.lang.Integer.toString(entity.pkey);//文字列型として扱うこと
      if (ids != null && ids.contains(pkey)) {
        print("unselect:"+pkey);
      } else {
        print("  select:"+pkey);
      }
    }
  }
}
```

selectMode が "__UNSELECT" の場合の補足説明

- モデル model1 の全件を取得するため、新しい FinderContext オブジェクトを用意します。空のコンディションモデル（検索条件なし）と、クライテリアコンバータをセットします。クリイテリアコンバータをセットすることで暗黙条件などが反映されるようになります。
- エンティティサービスの find メソッドを呼び出します。引数に FinderContext オブジェクトを渡してください。全件を取得します。
- 上述した "選択を解除された" 主キーを判定するようにします。

全件取得ではなく、検索条件で絞り込む

上のスクリプトは finderContext にセットした検索条件は空でしたので、全件取得となっていました。これを画面に用意された検索条件の値を使う場合は、次のようにします。

```
function process() {
  var DbShowListSelectableAllBaseController =
    Java.type("jp.jasminesoft.jfc.controller.DbShowListSelectableAllBaseController");

  var selectMode = p.appMap.get("showListModel1_selectMode");

  if (selectMode !== null && selectMode === "__SELECT") {
    var ids = p.appMap.get(DbShowListSelectableAllBaseController.selectIdSetAppmapName);
    ids.forEach(function(pkey){
       print("select:" + pkey);
    });
  } else {
    var ids = p.appMap.get(DbShowListSelectableAllBaseController.unselectIdSetAppmapName);
    var Model1C = Java.type("jp.jasminesoft.wagby.model.model1_c.Model1C");
    var FinderContext = Java.type("jp.jasminesoft.jfc.dao.FinderContext");
    var model1_c = p.appMap.get("model1_c_preserve");

    var finderContext = new FinderContext();
    finderContext.setPageSize(-1);
    finderContext.setCondition(model1_c);
    finderContext.setCriteriaConverter(p.appctx.getBean("Model1CriteriaConverter"));
    var entityService = p.appctx.getBean("Model1EntityService");
    var results = entityService.find(finderContext);
    for (i=0; i<results.size(); i++) {
      var entity = results.get(i);
      var pkey = java.lang.Integer.toString(entity.pkey);
      if (ids != null && ids.contains(pkey)) {
        print("unselect:"+pkey);
      } else {
        print("  select:"+pkey);
      }
```

```
    }
  }
}
```

- p.appMap からキー<モデルID>_c_preserveを指定して、直前に用いた検索条件を取得します。これをfinderContextにセットします。

仕様・制約

- ボタンの配置を変更することはできません。Wagbyが提供する標準の一括処理ボタンの右隣から順に配置されます。
- グリッド形式表示ではいくつかの制約があります。詳細はこちらをお読みください。

スクリプト

ヘルパとコントローラ（スクリプト）

ヘルパのスクリプト

「画面 > スクリプト」では実行タイミングを指定して、スクリプトを記述することができます。

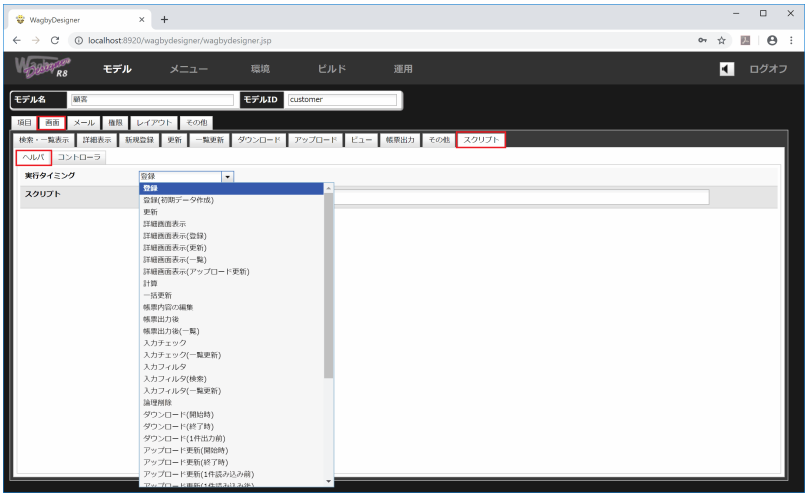


図 スクリプトの入力

Wagby Designer で記述するスクリプトは「呼び出されるタイミング」を指定します。これは「ヘルパ」と「コントローラ」に大別されます。この二つの役割は次のとおりです。

	ヘルパ	コントローラ
タイミング	モデル（1件のデータ）を処理している間。	ユーザによるボタン操作（アクション）
記述する内容	主にモデルの操作。SQLを使わずにモデルの値を直接、操作することができる。	主にユーザから入力された値の取得と、画面遷移。
トランザクションとの関係	トランザクション境界の内部である。（トランザクションは開始されている。）更新系のヘルパでは、モデルへの値の変更は自動的にデータベースに反映される。	トランザクションは開始されていない、または終了後である。
ロールバック	例外 BusinessLogicException をスローすることで、現在	できない。

	のトランザクションをロールバックさせることができる。	
SQLを利用する	データベースとの接続を行うsessionオブジェクト(*1)が利用できる状態にあるため、SQL(*2)を利用できる。(*3)	データベースとの接続を行うsessionオブジェクトをスクリプト内で用意し、使い終わったら必ずクローズ処理を行う必要がある。(*4)(*5)

(272) (273) (274) (275) (276)

ヘルパの詳細

入力したスクリプトから、拡張子 ".js" のファイルが生成されます。実ファイルは wagbyapp/webapp/$(APPNAME)/WEB-INF/script/モデルIDフォルダ の中に用意されます。

(277)

タイミング	説明	生成されるファイル
登録	「初期値 > 登録時」を行ったあとに呼び出されます。登録直前のデータを修正することができます。この処理を行った後のデータが、データベースに保存されます。	<モデルID>Helper_beforeInsert.js
登録(初期データ作成)	「初期値 > 登録画面表示時」を行ったあとに呼び出されます。新規登録用のデータを修正することができます。この処理を行った後のデータが、登録画面に表示されます。（このタイミングではデータベースへは保存されません。）	<モデルID>Helper_initialize.js
更新	「初期値 > 更新時」を行ったあとに呼び出されます。更新直前のデータを修正することができます。この処理を行った後のデータが、データベースに保存されます。	<モデルID>Helper_beforeUpdate.js
詳細画面表示	「初期値 > 詳細画面表示時」を行ったあとに呼び出されます。表示直前のデータを修正することができます。この処理を行った後のデータが、詳細画面に表示されます。（データベースには反映されません。表示の直前で内部のデータを改変する必要がある場合に利用できます。例えば一部のデータを表示時に隠す、などです。）このスクリプトは自動計算やAjax処理など、さまざまな処理で呼び出されることがありますので、	<モデルID>Helper_beforeShow.js

(272) 1. その実体は HibernateSession です。

(273) 2. 正確には Hibernate に最適化された HQL (Hibernate Query Language) です。

(274) 3. ヘルパで更新系のSQLを使う場合、自モデルの値の変更は行わないようにしてください。スクリプトの終了後に、メモリ上のモデル（ストアモデル）の値でデータベースが更新されます。

(275) 4. 詳細はSQLを利用するをお読みください。

(276) 5. コントローラでは自モデル、他モデルを問わず更新系のSQLを使うことができます。ここでトランザクションは開始されていないか、または終了しているため、いったん更新されたデータに対して再度、更新を行うという動作になります。

(277) $(APPNAME) は、プロジェクト識別子です。標準は "wagby" となっています。

	ご利用の際には複数回呼び出されることを想定してください。例えば、ここで値を更新するようなスクリプトを記述しないといった配慮が必要です。	
詳細画面表示(登録)	登録画面を開くときに呼び出されます。	<モデルID>Helper_beforeShowInInsert.js
詳細画面表示(更新)	更新画面を開くときにデータベースの値を読み込みますが、そのあとにこのスクリプトが呼び出されます。	<モデルID>Helper_beforeShowInUpdate.js
詳細画面表示(一覧)	一覧画面を開くときにデータベースの値を読み込みますが、そのあとにこのスクリプトが呼び出されます。一覧表示の直前で内部のデータを改変する必要がある場合に利用できます。	<モデルID>Helper_beforeShowInShowList.js
詳細画面表示(アップロード更新)	アップロード更新でデータを1件処理した直後に呼び出されます。	<モデルID>Helper_beforeShowInUploadUpdate.js
計算	定義済みの計算式を実行したあとに、このスクリプトが呼び出されます。	<モデルID>Helper_calc.js
一括更新	検索・一覧表示画面 > 一括更新ボタンを表示する の処理内容を記述します。	ShowList<モデルID>Controller_do_updateRule.js
帳票内容の編集	Excel/PDF帳票の前に呼び出されます。出力内容を修正することができます。	ReplaceMapHelper_remake.js
帳票出力後	Excel/PDF帳票出力後に呼び出されます。実際にはコントローラのスクリプトとなります。例えば出力後に"帳票出力日"を設定したいという場合、EntityService を使ってモデルを取得し、更新するようにしてください。	Show<モデルID>Controller_print.js
帳票出力後(一覧)	Excel/PDF帳票出力後に呼び出されます。実際にはコントローラのスクリプトとなります。例えば出力後に"帳票出力日"を設定したいという場合、EntityService を使ってモデルを取得し、更新するようにしてください。	ShowList<モデルID>Controller_print.js
入力チェック	標準で用意している必須チェックや文字数チェックのあとに呼び出されます。独自のチェック処理を追加することができます。	<モデルID>PInputCheckHelper_input_check.js
入力チェック(一覧更新)	標準で用意している必須チェックや文字数チェックのあとに呼び出されます。独自のチェック処理を追加することができます。	<モデルID>UlpInputCheckHelper_input_check.js
入力フィルタ	標準で用意している入力フィルタ処理のあとに呼び出されます。独自のフィルタ処理を追加することができます。	<モデルID>PFilterHelper_filter.js

入力フィルタ(検索)	標準で用意している入力フィルタ処理のあとに呼び出されます。独自のフィルタ処理を追加することができます。	<モデルID>CpFilterHelper_filter.js
入力フィルタ(一覧更新)	標準で用意している入力フィルタ処理のあとに呼び出されます。独自のフィルタ処理を追加することができます。	<モデルID>UlpFilterHelper_filter.js
論理削除	判定ルールを独自に決めることができます。	<モデルID>Helper_logicalDelete.js
ダウンロード(開始時)8.3.0	ダウンロード処理の開始時に一回だけ呼び出されます。	Download<モデルID>_processStart.js
ダウンロード(終了時)8.3.0	ダウンロード処理の終了時に一回だけ呼び出されます。	Download<モデルID>_processEnd.js
ダウンロード(1件出力前)8.3.1	ダウンロードするファイルの内容を加工することができます。	Download<モデルID>_outputCsv.js
アップロード更新(開始時)	アップロード更新処理の開始時に一回だけ呼び出されます。	UploadUpdate<モデルID>_processStart.js
アップロード更新(終了時)	アップロード更新処理の終了時に一回だけ呼び出されます。	UploadUpdate<モデルID>_processEnd.js
アップロード更新(1件読み込み前)8.3.1	アップロード更新でデータを1件読み込む前に呼び出されます。ファイルの内容を加工することができます。	UploadUpdate<モデルID>_readRowDataInFileProcessor.js
アップロード更新(1件読み込み後)	アップロード更新でデータを1件読み込んだ直後に呼び出されます。	UploadUpdate<モデルID>_AfterRead.js
アップロード更新(ヘッダ読み込み時)8.3.0	スクリプトでヘッダ行を指定することができます。	UploadUpdate<モデルID>_readHeader.js
検索(初期値)	検索条件の初期値を設定することができます。	<モデルID>CHelper_initialize.js
検索(前処理)	検索処理の直前に呼び出されます。クライテリアのカスタマイズではなく、コンディションモデルの値を書き換えるという場合は、こちらを使ってください。	<モデルID>CHelper_beforeSearch.js
検索	検索条件（クライテリア）をカスタマイズするタイミングとして用意されています。(*6)	<モデルID>CriteriaConverter_convert.js
検索(一覧更新)8.3.2		UpdateList<モデルID>CriteriaConverter_convert.js
検索(サブウィンドウ検索)8.3.2		SearchList<モデルID>CriteriaConverter_convert.js
検索(カレンダビュー)8.3.2		CalendarView<モデルID>CriteriaConverter_convert.js
検索(帳票画面)8.3.2		ReportList<モデルID>CriteriaConverter_convert.js
検索(ダウンロード)8.3.3(*8)		Download<モデルID>CriteriaConverter_convert.js
削除	削除処理の直前に呼び出されます。なおアップロード更新処理からは、この削除スクリプト(beforeDelete)は呼び出されません。	<モデルID>Helper_beforeDelete.js
ストアモデルからプレゼンテーションモデルへの変換	ストアモデルからプレゼンテーションモデルへの変換の前に preprocess() を、変換の後に process() を呼び出し	<モデルID>PHelper_s2p.js

	ます。(*7) 絞り込みで利用した例を紹介しています。	
プレゼンテーションモデルからストアモデルへの変換	プレゼンテーションモデルからストアモデルへの変換の前に preprocess() を、変換の後に process() を呼び出します。(*7)	<モデルID>PHelper_p2s.js
ファイル項目ダウンロード 8.3.0	ファイル項目のダウンロード処理の直前に呼び出されます。モデルやモデル項目に関わらず、すべてのファイル項目で呼び出されます。	DownloadFile_beforeDownload.js
ファイル項目アップロード 8.3.0	ファイル項目のアップロード処理の直後に呼び出されます。モデルやモデル項目に関わらず、すべてのファイル項目で呼び出されます。	UploadFile_afterUpload.js

(278) (279) (280)

スクリプトと画面の関係

スクリプトは、用意する画面と関連しています。そのため、例えば詳細画面を作成しないモデルでは"詳細画面表示"や"削除"に関するスクリプトは生成されません。

コントローラのスクリプト

画面に用意されたボタン押下のタイミングで実行されるスクリプトです。この詳細は「画面遷移」で説明します。

スクリプトと画面の関係

スクリプトは、用意する画面と関連しています。そのため、例えば詳細画面を作成しないモデルでは、画面"詳細"に関するスクリプトは生成されません。

共通スクリプト

「環境タブで設定できる共通スクリプト」をお読みください。

トランザクション

複数の登録、更新、削除処理を一つの塊としてデータベースにコミットまたはロールバックすることをトランザクションといいます。Wagby はトランザクションを実現するために二つの方法を提供しています。

(1) 関連するモデルが定義で定まっている場合

Wagby が標準で提供するモデル参照という仕組みを用いて自身に紐づくモデルが定まっている場合、簡単にトランザクションのスクリプトを作成することができます。自身の変更にあわせて（自身に）紐づくモデルの更新を同時に行うことができます。

このトランザクション処理はモデル項目詳細定義で記述します。詳細は「モデルをまたがる計算（トランザクション）」で説明します。

(278) 6. R8.3.2までは、ダウンロード処理および全文検索と併用したモデルでは呼び出されません。R8.3.3より内部処理を刷新し、ダウンロード処理からも呼び出されるようになりました。

(279) 7. 処理を行う前に呼び出す場合は関数 preprocess を用意することができます。

(280) 8. R8.3.3ではcustomizeフォルダ直下にスクリプトを用意します。R8.3.4でDesignerから設定できます。

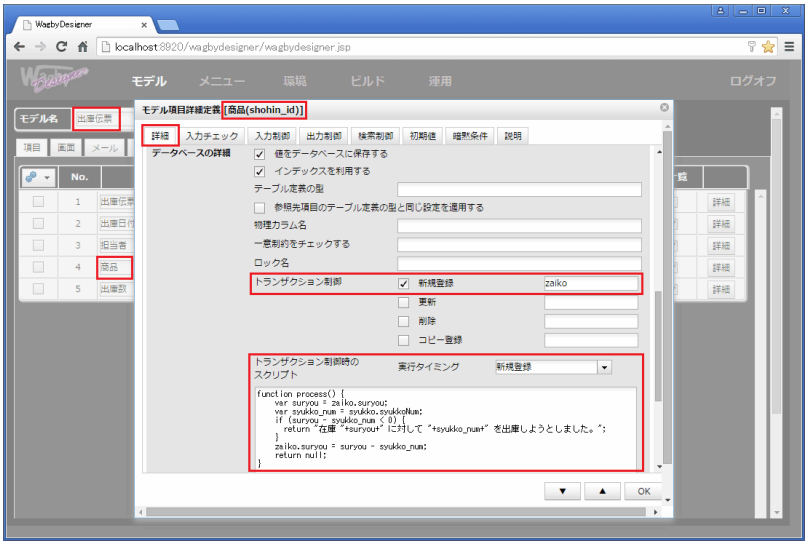


図 トランザクション制御時のスクリプト

(281)

(2) 開発者が自由にトランザクションのコードを記述する場合

開発者は直接、Wagbyが生成する EntityService というクラスを使ってトランザクションのコードを記述することができます。 詳細は「Service/Daoクラス、ヘルパ、SQLを利用する」で説明します。

キャメル記法

Wagby ではスクリプトやパラメータ記述においてキャメル記法というルールが適用されます。

- 項目IDが複合語の場合、先頭を小文字で書き始めます。項目IDが Item123 の場合、スクリプトでは item123 となります。
- 項目IDに "_" （アンダースコア）が含まれている場合、その "_" は除去され、その次の文字が大文字になります。例えば customer_id は customerId になります。kana_name は kanaName になります。
- アンダースコアを取り除いたあとの先頭が小文字で、二文字目が大文字の場合、先頭も大文字となります。例えば p_number は pNumber となったあと、最終的に PNumber になります。

スクリプト内では「項目ID」にキャメル記法が適用されます。またスクリプトファイル名に含まれるモデルIDにもキャメル記法が適用されます。

呼び出されるスクリプトファイル名を確認する

次の方法で、どのようなスクリプトファイルがどのタイミングで呼び出されるかを調べることができます。

1. ビルドされたアプリケーションの webapps¥$(APPNAME)¥WEB-INF¥classes フォルダに含まれる jfcbase.properties をテキストエディタで開きます。
2. 次の行を見つけ、先頭の ";" を除きます。 （修正前）

```
;jp.jasminesoft.jfc.ScriptCodeRunner.isOutputStatusToConsole=true

;jp.jasminesoft.jfc.ScriptCodeRunner4Controller.isOutputStatusToConsole=true
```

（修正後）

```
jp.jasminesoft.jfc.ScriptCodeRunner.isOutputStatusToConsole=true

jp.jasminesoft.jfc.ScriptCodeRunner4Controller.isOutputStatusToConsole=true
```

3. アプリケーションを起動します。

(281) なおトランザクションスクリプトで（自身に紐づくモデル以外の）任意のモデルを操作する方法も提供しています。この場合は Dao クラスを使います。[詳細...]

4. アプリケーションを操作するたびに、Tomcatのコンソールに、どのスクリプトファイルを実行しようとしたかが表示されます。（ファイルが存在しなければ実行されることはありません。）

モデルの基本（1）（スクリプト）

モデルとオブジェクト

Designerに入力した「モデル」の設計情報（リポジトリ）から、データベースのテーブル定義（CREATE DDL 文)と、クラス定義（Javaのクラス）が自動生成されます。この関係を図に示します。

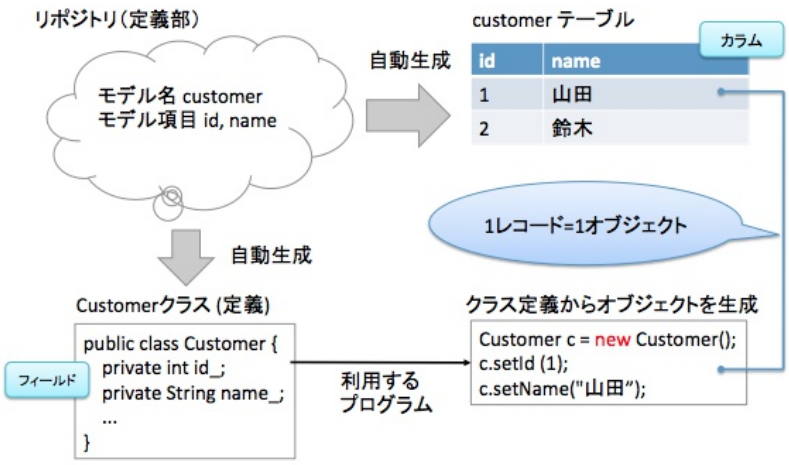


図 モデル定義とテーブル、クラス図の関係

「モデル」は、すべて Java オブジェクトとして操作することができます。リレーショナルデータベースからみた場合、1データ(1レコード)が、1つのモデルの実体となります。

この実体のことを「オブジェクト」と呼びます。

(282)

スクリプトでは、対象のオブジェクトを直接、操作できます。例えばモデルID customer を扱うためには、スクリプト内に直接 "customer" という変数名が使えます。これがオブジェクトです。

「項目ID」は、「オブジェクト名 + "." + 項目ID」で参照することができます。例えば name という項目を用意していた場合、"customer.name" と記述できます。

> 「項目ID」はキャメル記法を使います。例えば customer.kana_name はスクリプト中では customer.kanaName と記述します。

ファイル型

- ファイル型は内部では二つのフィールド（ならびにテーブル上でも二つのカラム）から構成されます。
- Designerで定義した項目名には、ファイル名が入ります。
- 内部で自動的に用意される「項目名_jshfilename」は、ファイルの実パス情報（文字列）が格納されます。
- ファイル情報そのものはBLOBとして管理しません。あくまでファイル名と実パスとして管理しています。そのため物理ファイルが削除されてしまうと、参照エラーになります。

(282) 「インスタンス」と呼ぶこともあります。本マニュアルでは「オブジェクト」という呼び方で統一します。

生成されたモデルのJavaコードを直接確認する

自動生成された Java のソースコードを直接、確認することで、どのようなメソッドが用意されているかを知ることができます。

例えば customer モデルに関するストアモデルは次のファイルになっています。

```
wagbydesigner/webapps/wagbydesigner/WEB-INF/env/work/srcgen/jp/jasminesoft/wagby/model/customer/Customer.java
```

(283)

それぞれの区分毎に、生成されるモデルのソースコードのフォルダが変わります。（フォルダの起点は、wagbydesigner/webapps/wagbydesigner/WEB-INF/env/work/srcgen です。）

モデルIDを customer としたときの例を示します。

ストアモデル	
FQCN	jp.jasminesoft.wagby.model.customer.Customer
ファイル	jp/jasminesoft/wagby/model/customer/Customer.java
プレゼンテーションモデル	
FQCN	jp.jasminesoft.wagby.model.customer_p.CustomerP
ファイル	jp/jasminesoft/wagby/model/customer_p/CustomerP.java
コンディションモデル	
FQCN	jp.jasminesoft.wagby.model.customer_c.CustomerC
ファイル	jp/jasminesoft/wagby/model/customer_c/CustomerC.java
コンディションのプレゼンテーションモデル	
FQCN	jp.jasminesoft.wagby.model.customer_cp.CustomerCp
ファイル	jp/jasminesoft/wagby/model/customer_cp/CustomerCp.java
リストモデル	
FQCN	jp.jasminesoft.wagby.model.customer_l.CustomerL
ファイル	jp/jasminesoft/wagby/model/customer_l/CustomerL.java
リストのプレゼンテーションモデル	
FQCN	jp.jasminesoft.wagby.model.customer_lp.CustomerLp
ファイル	jp/jasminesoft/wagby/model/customer_lp/CustomerLp.java

(284) (285)

モデルの種類

Designerで定義した「モデル」から、図に示す複数のクラスが生成されます。

(283) 上記ファイルでの「パッケージ名に対応するフォルダ（この例では jp/jasminesoft/wagby）」は、パッケージ名を指定したプロジェクトでは変更されます。適切に読み替えてください。

(284) FQCN : Fully Qualified Class Name の略。完全修飾クラス名といい、パッケージ名を付与したクラスとして表現します。

(285) 繰り返しコンテナも独立したクラスとなっているため、上と同様に表現できます。例えば customer モデルの繰り返しコンテナ cont であれば、jp.jasminesoft.wagby.model.customer.Cont となります。ファイルは jp/jasminesoft/wagby/model/customer/Cont.java になります。

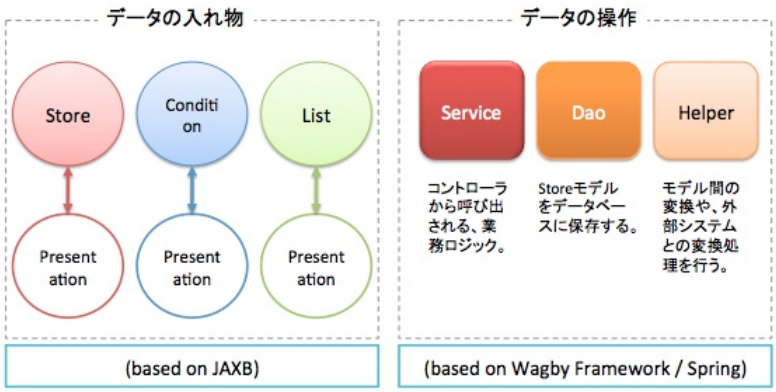


図 モデルの種類

分類	名称	説明
入れ物	ストアモデル	データベースのレコードと 1:1 の関係がある。
	コンディションモデル	検索条件を格納する。
	リストモデル	一覧表示項目を格納する。
	プレゼンテーションモデル	ストア/コンディション/リストそれぞれに対となるプレゼンテーションモデルが用意される。Webフォーム(画面に表示される入力フォーム)に対応したモデル。すべて文字列型の値となる。
操作	サービス	業務処理を行う。
	データ操作(DAO)	ストアモデルとデータベース間の基本的なやりとり(CRUD)を行う。
	ヘルパ	モデルの初期や計算、参照連動の解決といったモデル操作に付随する処理を行う。また各モデルとプレゼンテーションモデルの相互変換も行う。

(286) (287) (288) (289) (290)

スクリプト内では、次の変数名で利用できます。

名前	スクリプト内での表現	例	説明
ストアモデル	モデル名	customer	データベースに保存されるオブジェクト。
プレゼンテーションモデル	モデル名_p	customer_p	Webフォームから入力された値。
コンディションモデル	モデル名_c	customer_c	検索条件が格納されたオブジェクト。
コンディションのプレゼンテーションモデル	モデル名_cp	customer_cp	Webフォームから入力された検索条件の値。

(286) DAO : Data Access Object の略。[Wikipedia...]
(287) CRUD : Create/Read/Update/Delete の略。データの登録、検索、更新、削除といった基本操作を示す。[Wikipedia...]
(288) JAXB : Java Architecture for XML Binding の略。Java と XML の相互変換を行う。[Wikipedia...]
(289) Wagby Framework : Wagby に同梱されているフレームワーク。自動生成される各クラスの基底部分を提供する。
(290) Spring : Spring Framework のこと。Java によるエンタープライズアプリケーション開発の標準的な基盤技術。[Wikipedia...]

リストモデル	モデル名_l	customer_l	一覧表示される値が格納されたオブジェクト。
リストのプレゼンテーションモデル	モデル名_lp	customer_lp	一覧表示オブジェクトの画面出力用。(入力は用意されない)

(291) (292)

ストアモデル

ストアモデルは「テーブル」の「1レコード」の情報を格納するためのオブジェクトです。ほとんどのスクリプト中で利用することができます。

Wagbyによって生成されたコードの一部を抜粋します。

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlRootElement(name = "customer")
@XmlType(propOrder={ "customerid_", "name_", "namekana_", "email_", "tel_", "fax_",
 "companyname_", "companynamekana_", "companytel_", "companyfax_", "companyzipcode_",
 "companyaddress_", "companyurl_", "officename_", "officenamekana_", "officetel_", "officefax_",
 "officezipcode_", "officeaddress_", "keyperson_", "retiredate_", "note_" })
public class Customer extends jp.jasminesoft.jfc.app.ContainerBase<Customer> implements
 java.io.Serializable, Cloneable {
    @XmlElement(name="customerid")
    private int customerid_;
    @XmlElement(name="name")
    private String name_;
    ...
    public final String getName() {
        return (name_);
    }
    public final void setName(String name) {
        this.name_ = name;
    }
    ...
}
```

特徴

- 「項目ID」が（Java クラスの）"フィールド" に対応します。
- すべてのフィールドはprivateで宣言されます。
- フィールドに対応するアクセッサ（setter/getter）メソッドが用意されます。
- "@Xml" ではじまるアノテーションは JAXB が提供するものです。これによりオブジェクトと XML 表現の相互変換を実現しています。（例 ： Wagby のインポートエクスポート機能でやりとりする xml ファイルは、このアノテーションによって実現されています。）
- リレーショナルデータベースのテーブル、列は、それぞれストアモデルのクラス名、フィールド名と対応しています。この対応処理は（Wagbyに同梱されている）Hibernate というミドルウェアが管理しています。[Wikipedia...]

クラス名、フィールド名の命名規則

クラス名

- モデルIDの先頭を大文字にします。元々が大文字の場合は、そのままです。
- アンダースコア（_）は除去し、その次の文字を大文字にします。sales_slip は SalesSlip になります。

フィールド名

(291) ストアモデル以外のオブジェクトは、存在しない場合もあります。例えば検索画面の操作でない場合、コンディションモデルは含まれません。

(292) 状況によってはストアモデルの取得もできない場合があります。例えば削除処理の場合、ストアモデルは用意されていません。

- 項目IDに対応したフィールドが用意されます。
- 項目名の末尾にアンダースコアを含めます。id は id_ になります。
- アンダースコア（_）を取り除き、その次の文字を大文字にします。customer_id は、customerId_ になります。
- データを取得する getter と、値をセットする setter メソッドが用意されます。例えば getCustomerId() や setCustomerId(引数) などです。
- スクリプトから操作する場合、常に getter/setter メソッドが使われます。例えばスクリプト中で customer.customerId と記述したとき、内部では getCustomerId() メソッドが呼び出されています。つまりスクリプトで直接、モデルのフィールドを操作しません。

Javaの型

Designerで指定した型と、実際に生成されるコードで使われる Java の型との関係は次のとおりです。

Designerで指定した型	Javaの型
整数型	Integer（非・必須)int（必須）
1,2,4,8バイト整数	Byte, Short, Integer, Long（非・必須)byte, short, integer, long（必須）
4バイト小数	Float（非・必須)float（必須）
8バイト小数	Double（非・必須)double（必須）
文字列、ファイル、メール、URL、郵便番号	String
日付	java.sql.Date
時刻	java.sql.Time
日付時刻	java.sql.Timestamp

(293) (294) (295)

必須項目の扱い

Designerで「必須」と設定した場合、生成されるコードが変わります。

Designerの型	区分	Javaの型	初期値	未入力の確認方法	テーブル
整数型	非・必須	Integer	null	checkメソッド	NOT NULL が付与される
整数型	必須	int	0	なし	-

整数型項目で非・必須の場合、checkメソッドが提供されます。具体的には check項目ID() というメソッドです。戻り値は boolean 型となり、値はtrueかfalseのいずれかです。これを使って未入力かどうかの確認を行うことができます。

なお getメソッド（get項目ID）を使ったとき、値が null の場合は "-1" が返されます。

Integer と int の違い / null と 0 の違い

Integer と int は異なる扱いです。Integer は null（未入力）という状態がありますが、int は必ず何らかの値が入るもので、null を許容しません。int の初期値は 0 となりますが、入力された値が 0 なのか、それとも未入力で初期値が 0 なのかは判断できません。

テーブル定義が変わります

必須項目とした場合、そのモデルのテーブル定義が変更されます。該当項目に NOT NULL が付与されます。

単一の値と、配列

モデル項目の型によってスクリプトで扱う方法が変わります。

項目の型	扱い

(293) 日付、時刻は java.util パッケージではなく java.sql パッケージです。

(294) Javaのboolean型に相当する(Wagby Designerの)定義はありません。intで代替します。

(295) JavaのBigInteger/BigDecimalクラスに相当する(Wagby Desginerの)定義はありません。Wagbyが提供する関数ADD,SUB,MUL,DIVは内部でBigIntegerを使った演算を行います。その結果をlongやdouble型で保存しています。

文字列型、数値型、日付型、モデル参照（リストボックス、ラジオボタン、検索）	一つの値をもつ
モデル参照（チェックボックス）、繰り返し項目、繰り返しコンテナ項目	配列となる

詳細はこのあと説明します。

モデルの基本操作

customer モデルを用意し、「画面 ＞ スクリプト」で実行タイミングを「登録」とし、次のスクリプトコードを入力します。

```
var customerid = customer.customerid; /* 主キー(整数型) */
var name = customer.name; /* 文字列型 */
var companyname = customer.companyname; /* モデル参照 */
var companynamekana = customer.companynamekana; /* 参照連動 */
var retiredate = customer.retiredate; /* 日付型 */

print("customerid="+customerid);
print("name="+name);
print("companyname="+companyname);
print("compnaynamekana="+companynamekana);
print("retiredate="+retiredate);
```

customer モデルの新規登録が成功すると、Tomcat を実行しているコンソールに次のように表示されます。

```
customerid=1000
name=山田 太郎
companyname=1000
compnaynamekana=じゃすみんそふと
retiredate=2010-03-31
```

数字、文字列、日付はデータベースに格納している値がそのまま出力されます。

モデル参照項目 companyname は、参照先モデルの「主キー」が格納されていることがわかります。また、参照連動項目 companynamekana は、参照先のデータが含まれていることがわかります。なお、参照していない場合は companyname の値は "-1" になります。（参照先モデルの主キー型項目が整数型の場合）

モデル参照項目の詳細は、このあと説明します。

[例] 値をセットする

上の例は、もともと格納されている値を表示させるものでした。今度は値をセットするスクリプトを紹介します。

```
customer.age = 20; /* 整数型 */
customer.name = "鈴木 一郎"; /* 文字列型 */
customer.retiredate = ExcelFunction.DATEVALUE("1972-5-15"); /* 日付型 */
```

整数型の項目は直接、数値を指定することができます。 文字列型の項目はダブルクォーテーションで値を指定します。日本語文字も指定できます。 日付型の項目は Java の日付オブジェクトを指定します。ここでは Wagby が提供している関数 DATEVALUE を用いた例を紹介します。スクリプトで関数を利用する場合は接頭語 ExcelFunction を付与してください。

DAO

1つのストアモデルに対して、1つのDAO (Data Access Object) が用意されます。 DAOが提供する get メソッドを使って、データベースから1つのデータをオブジェクトとして読み込むことができます。スクリプト中での利用例を示します。

```
// 顧客ID（主キー)が1000の顧客データを読み込む。
var customerDao = p.appctx.getBean("CustomerDao");
```

```
var customer = customerDao.get(1000);
```

DAO の実体は Java のオブジェクトです。p.appctx.getBean("モデルIDDao"); で、オブジェクトを取得してください。

スクリプト内では通常、customer オブジェクトを DAO を使って取得したり、あるいは（Java の new 演算子を用いて）生成することはありません。多くの場合、スクリプトが呼び出される前に、Wagby がデータベースから読み込んだ値からオブジェクトを生成し、すぐにスクリプトで使えるように準備済みです。現在扱っているオブジェクトとは別のオブジェクトを利用する必要がある、というときに DAO から値を取得してください。この場合、取得したいデータの主キーが必要です。

繰り返し

繰り返し項目

Designerで「繰り返し」を指定した場合、内部では次のように扱われます。

- 繰り返し項目は「文字列」「数字」「日付、時刻、日付時刻」に指定できます。
- 内部では List というコレクションに複数の値を保持します。

```
@XmlElement(name="tel")
private java.util.List<String> tel_ = new java.util.ArrayList<String>();
```

- データベースには別テーブルが用意されます。例えば customer モデルの tel という項目が繰り返し指定されたとき、別テーブル "customer$tel" が用意されます。
- DAOからオブジェクトを取得したとき、Listに値は入った状態です。（つまりテーブルの join は必ず行われます。）

繰り返しコンテナ

Designerで「繰り返しコンテナ」を指定した場合、内部では次のように扱われます。

- 繰り返しコンテナは、独立した（コンテナ名の）クラスになります。例えば繰り返しコンテナ "cont" という項目を用意したとき、Cont というクラスが生成されます。
- 内部では List というコレクションに複数のコンテナを保持します。

```
@XmlElement(name="cont")
private java.util.List<Cont> cont_ = new java.util.ArrayList<Cont>();
```

- データベースには別テーブル "customer$cont" が用意されます。自動的に外部キー関係になります。
- DAOからオブジェクトを取得したとき、Listに値は入った状態です。（つまりテーブルの join は必ず行われます。）
- 繰り返しコンテナの入れ子はできません。

(296)

スクリプト例 : 参照

スクリプトでは、繰り返しコンテナは「配列」として扱うことができます。 取得した繰り返しコンテナは、モデルと同様にオブジェクトとして操作できます。

次の例は休暇申請 (leave) モデルに含まれる繰り返しコンテナ「理由 (memo)」を取得するスクリプトです。

```
var array = leave.memo; /* memoは繰り返しコンテナ。arrayは配列形式となっている。*/
for (var i=0; i<array.length; i++) {
    print("memo="+array[i]);
    print("memo.memo_id="+array[i].memoId);
    print("memo.memo_reason="+array[i].memoReason);
}
```

スクリプト例 : 和を求める

次の例は売上伝票 (salesslip) モデルに含まれる「明細情報 (precord)」を操作し、単価と数量の和を求めるコードです。

```
var total = 0;
```

(296) この制約のため、深いネスト構造によるパフォーマンスの低下、という問題は起こりません。

```
var array = salesslip.precord; /* 繰り返しコンテナ */
for (var i=0; i<array.length; i++) {
    if (条件が成立) {
        total = total + (array[i].PPrice * array[i].PNumber);
    }
}
salesslip.totalPrice = total; /* 値を代入（セット）*/
```

上記コードにあるように、計算結果をオブジェクト（のモデル項目）に代入することもできます。

(297)

スクリプト例 : 新規追加

次の例は見積伝票 (quotation) モデルの新規登録画面を開いたタイミングで、繰り返しコンテナ "precord" を3件、追加するスクリプトです。

```
var PRecordClass = Java.type("jp.jasminesoft.wagby.model.quotation.Precord");
var array = new Array(3);
array[0] = new PRecordClass();
array[1] = new PRecordClass();
array[2] = new PRecordClass();

(各配列への、具体的な値の設定コードは省略)

quotation.precord = array;
```

(298) (299)

スクリプト例 : 削除

「ヘルパ > 更新」のスクリプトを用いると、更新直前のモデルに何らかの操作を行うことができます。ここでは kanridata モデルに含まれる繰り返しコンテナ zrecord について、(zrecord の項目) suryo が未入力のコンテナ行を削除するという例を紹介します。

```
var array = kanridata.zrecord;

var removeList = new java.util.ArrayList();
for (var i=0; i<array.length; i++) {
  if (!array[i].checkSuryo()) {/*非必須の数値項目はcheckXXメソッドが利用できる*/
    removeList.add(i);/*いったん削除対象の行のインデックスを記憶しておく*/
  }
}

for (var i=removeList.size()-1; i>=0; --i) {/*逆から削除していく*/
  var index = removeList.get(i);
  kanridata.removeZrecord(index);/*removeメソッドを用いる*/
}

/*繰り返しコンテナの内部IDを詰める*/
array = kanridata.zrecord;
for (var i=0; i<array.length; i++) {
  array[i].zrecordjshid = i;
}
```

(300) (301)

---

(297) Wagby が標準で提供する SUM 関数では記述が難しいような複雑な業務ルール（上記にあるような、演算のための条件式）にも対応できます。

(298) Java.type は特別な表記です。Java.type(FQCN形式のクラス名) と記述することで、Javaのクラスを表す変数を用意することができます。この変数を使って、オブジェクトを new 演算子で生成することができるようになります。

(299) FQCN : Fully Qualified Class Name の略。完全修飾クラス名といい、パッケージ名を付与したクラスとして表現します。

(300) オブジェクトへのデータ操作系メソッドは get/set に加え add/remove/check などがあります。詳細は自動生成された Java のモデルに関するソースコードを直接、ご確認いただくこともできます。

(301) 繰り返しコンテナの内部では "繰り返しコンテナ名" + "jshid" という内部ID項目が自動的に用意されています。この数値は連続した番号である必要があります。上記のサンプルコードでは途中のレコードを削除することがあるため、この内部IDを詰めるようにしています。

トランザクションとの関係

繰り返しコンテナは(データベース上では)別テーブルですが、ストアモデルでは一つのデータ内に明細情報が含まれる構造です。ここで、繰り返しコンテナを含むモデルの登録・更新・削除は、二つのテーブルの操作がセットで行われます。すなわち、暗黙的なトランザクション処理になります。

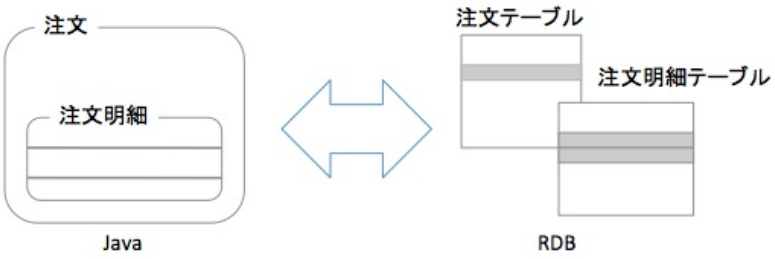


図 Javaクラスとテーブルの関係

削除に関する留意点

上図において、注文モデルを削除すると、注文明細も同時に削除されます。つまり繰り返しコンテナは強い結合関係にあります。

モデル参照（リストボックス、ラジオボタン、検索ウィンドウ）

モデル参照（リストボックス、ラジオボタン、検索ウィンドウ）では、参照しているモデルの主キーを保持します。

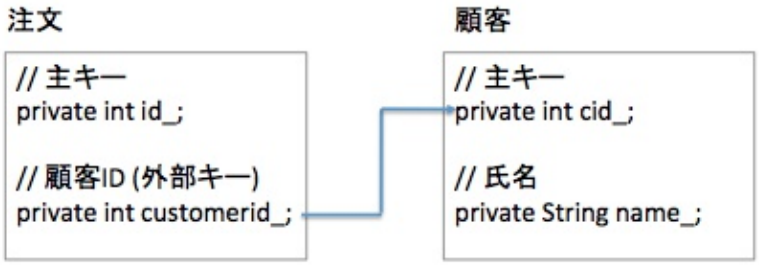


図 モデル参照関係

「注文DAO」からモデルを読み込んだとき、外部キーである「顧客ID」の値（例："1000"）は含まれていますが、「顧客」の情報そのものは含まれません。顧客IDを引数とし、「顧客DAO」を使うことで「顧客」データを取得することができます。

```
var customerDao = p.appctx.getBean("CustomerDao");
var customer = customerDao.get(order.customerid);
```

削除に関する留意点

「注文」モデルを削除しても、キーで紐付いている「顧客」モデルが削除されることはありません。　一方、「顧客」モデルを削除すると、「注文」モデルの参照先が不定となります。

モデル参照（チェックボックス）

モデル参照（チェックボックス）では、参照元モデルの主キーをListに保持します。

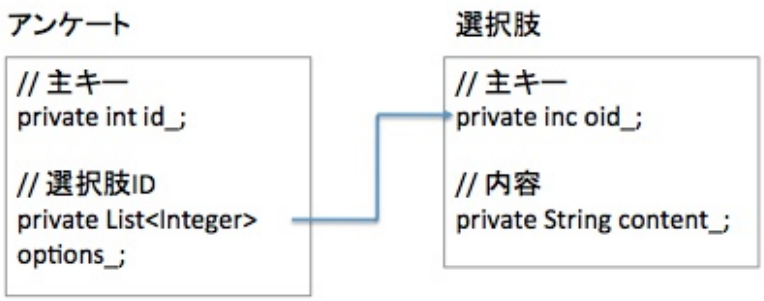


図 モデル参照関係（チェックボックス）

「アンケートDAO」からモデルを読み込んだとき、外部キーである「選択肢ID」の値を含むリストが用意されます。スクリプトでは、これを配列として取得することができます。

値の参照

enq モデルに、チェックボックス項目である options を用意しました。どのような値をもっているかを確認するためのコードは次のようになります。

```
var array = enq.options; /* チェックボックスは配列として取得できる */
...
for (i=0; i<array.length; i++) {
  print("option="+array[i]);
}
```

チェックされたものだけが配列 array に含まれます。例えば、次のように表示されます。

```
...
option=1
...
```

このように配列となっている値であっても、スクリプトでは常に「var 型」として用意します。 配列なので、for ループなどを用いて値を参照することができます。

IDではなく内容部を取得する

上のコードにあるように、ストアモデルには「選択肢」の情報そのものは含まれません。「選択肢DAO」を使って「選択肢」データを取得することができます。

```
var optionDao = p.appctx.getBean("OptionDao");
var array = enq.options;
for (int i=0; i<array.length; i++) {
  var option = optionDao.get(array[i]);
  print(option.content);/* 内容部を表示する */
}
```

チェックボックスを個別に操作する

enq.options に（チェックボックスの）ID を追加したり、削除することができます。addXXX, removeXXX, clearXXX, setXXX, size というメソッドを使います。これらは常に自動生成されるメソッドです。

```
/* test 1:値を一つずつ追加する */
enq.addOptions(1);
enq.addOptions(2);
print("test1 "+enq.options);

/* test 2:値を消去する */
/* 添字は0から始まる。以下は最初にチェックされていたものを解除する */
enq.removeOptions(0);
print("test2 "+enq.options);
```

```
/* test 3:すべての値を消去する */
enq.clearOptions();
print("test3 "+enq.options);

/* test 4:複数の値をまとめてセットする */
var IntArrayType = Java.type("int[]");
var arr = new IntArrayType(3);
arr[0] = 1;
arr[1] = 2;
arr[2] = 3;
enq.setOptions(arr);
print("test4 "+enq.options);

/* いくつチェックされているかを知る */
var size = enq.sizeOptions();
print("size="+size);
```

外部キー

親子関係では、親（1）に対して子（N）という関係が成立します。親には特段の記述はなく、子が親のキーを保持します。

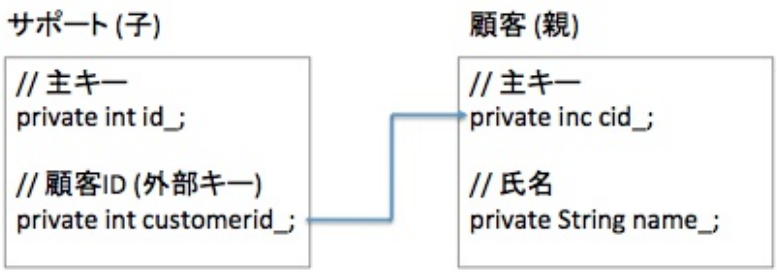


図 外部キー

このことから、Wagbyの「モデル参照」と「外部キー」に、クラス定義上の差異はありません。「外部キー」と指定することで、親の詳細画面に子の一覧が表示されるといった機能が自動的に付与されます。

削除に関する留意点

標準では「親」モデルを削除すると、キーで紐付いている「子」モデルも同時に削除されます。しかし設定によって、子モデルを残すこともできます。この場合の親子関係は弱い参照関係になります。

参照連動

参照連動は、参照元モデルの値を、参照先に転記する機能です。自動生成されたコードではDAOからデータを読み込み、値のセット処理を行います。

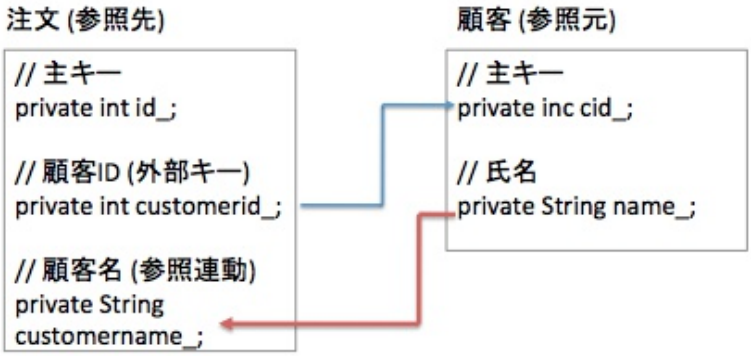


図 参照連動

ストアモデルの役割

ストアモデルは「データベースに保持される情報」だけでなく「画面に表示される情報」も含めることができます。参照連動はそのために用意された仕組みです。

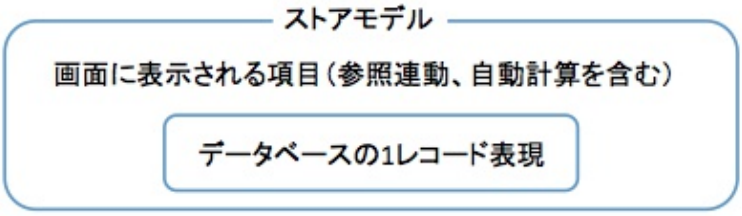


図 ストアモデルの役割

モデルの基本（2）（スクリプト）

プレゼンテーションモデル

ストアモデルにおけるモデル参照項目は、参照先モデルの主キーのみを保持します。これに「内容部」を含めたものがプレゼンテーションモデルです。

(302)

プレゼンテーションモデルには、画面に表示されるすべての情報が含まれます。 プレゼンテーションモデルは「表示モード」と「更新モード」があります。両者はモデル参照の扱いが異なります。

表示モードは、ストアモデルがもつ参照モデルのIDに対応した内容部をもちます。これは1:1の関係です。

更新モードは、選ばれなかった他の選択肢もすべて含まれています。登録、更新画面では、現在選ばれているもの以外（候補）も画面に表示されます。これらの選択肢が含まれています。

(302) それぞれ「ID部」「内容部」と呼んで区別します。

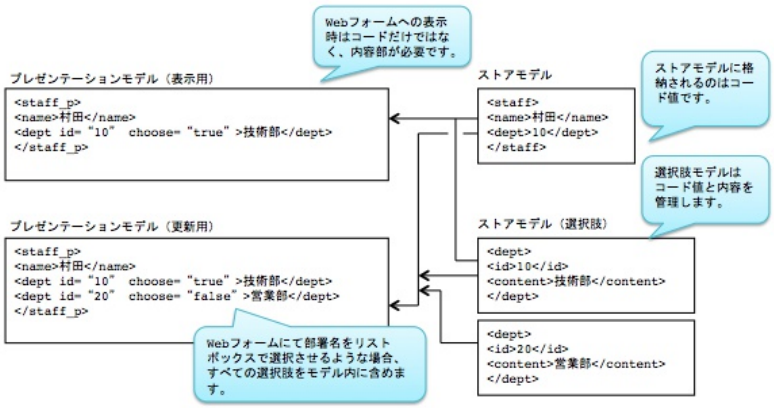


図 プレゼンテーションモデルのイメージ(XML形式で表現)

生成のタイミング

登録画面 - 画面表示

初期値が設定された最初のストアモデルが生成されます。主キーは未定の状態です。これをプレゼンテーションモデル（更新モード）に変換し、Webフォームで用います。

登録画面 - 登録

Webフォームから入力された値を元に、プレゼンテーションモデルが生成されます。このプレゼンテーションモデルをストアモデルに変換し、データベースに格納します。

更新画面 - 画面表示

データベースから読み込んだストアモデルをプレゼンテーションモデル（更新モード）に変換し、Webフォームで用います。

更新画面 - 更新

Webフォームから入力された値を元に、プレゼンテーションモデルが生成されます。このプレゼンテーションモデルをストアモデルに変換し、データベースに格納します。

詳細画面

データベースから読み込まれたストアモデルを、プレゼンテーションモデル（表示モード）に変換します。これを画面に出力します。

型について

プレゼンテーションモデルがもつ値はすべて「文字列型」です。ストアモデルとの相互変換の際、適切な型へ変換されます。

プレゼンテーションモデル固有の情報

モデル項目毎に、入力エラーに関する情報を保持します。また選択肢（リストボックス、ラジオボタン、チェックボックス）の場合、選択状態にあるかどうかという情報を保持します。

入力エラーの扱い

登録・更新画面における入力エラー発生時は、エラーメッセージが埋め込まれたプレゼンテーションモデルが再作成され、入力画面に再送されます。

スクリプトでの操作

プレゼンテーションモデルで扱う項目は、型によって「ID部」と「内容部」を保持します。ID部は参照先モデルの主キーを、内容部は画面に表示させる値をいいます。また、選択状態にあるものは、choose部の値がtrueとなります。

詳細表示画面

型	ID部	内容部	choose部
文字列	-	モデルID.項目.content	-
文字列(メールアドレス)			
文字列(URL)			
郵便番号			
整数			
1バイト整数			
2バイト整数			
4バイト整数			
8バイト整数			
4バイト浮動小数点			
8バイト浮動小数点			
日付			
時刻			
日付と時刻			
文字列(追記型リストボックス)	-	項目[0].content	-
文字列(郵便番号)	-	項目.content	-
モデル参照(リストボックス)	項目[0].id(* "スクリプトの例[1]" を参照)	項目[0].content(* 左に同じ)	項目[0].choose(* 左に同じ)
モデル参照(ラジオボタン)	項目[index].id(* "スクリプトの例[2]" を参照)	項目[index].content(* 左に同じ)	項目[index].choose(* 左に同じ)
モデル参照(チェックボックス)	項目[index].id(* "スクリプトの例[2]" を参照)	項目[index].content(* 左に同じ)	項目[index].choose(* 左に同じ)
モデル参照(検索画面)	項目[0].id	項目[0].content	-
参照連動	-	項目.content	-

一覧表示画面

型	ID部	内容部	choose部
文字列	-	item.項目	-
文字列(メールアドレス)			
文字列(URL)			
郵便番号			
整数			
1バイト整数			
2バイト整数			
4バイト整数			
8バイト整数			
4バイト浮動小数点			
8バイト浮動小数点			
日付			
時刻			
日付と時刻			
文字列(追記型リストボックス)	-	item.項目	-
文字列(郵便番号)	-	item.項目	-
モデル参照(リストボックス)	-	item.項目[0].content	-

モデル参照(ラジオボタン)	item.項目[index].id	item.項目[index].content	item.項目[index].choose
モデル参照(チェックボックス)	item.項目[index].id	item.項目[index].content	item.項目[index].choose
モデル参照(検索画面)	-	item.項目[0].content	-
参照連動	-	item.項目	-

登録・更新画面

型	ID部	内容部	choose部
文字列	-	項目.content	-
文字列(メールアドレス)			
文字列(URL)			
郵便番号			
整数			
1バイト整数			
2バイト整数			
4バイト整数			
8バイト整数			
4バイト浮動小数点			
8バイト浮動小数点			
日付			
時刻			
日付と時刻			
文字列(追記型リストボックス)	項目[index].id(* "スクリプトの例[2]"を参照)	項目[index].content(* 左に同じ)	項目[index].choose(* 左に同じ)
文字列(郵便番号)			
モデル参照(リストボックス)			
モデル参照(ラジオボタン)			
モデル参照(チェックボックス)			
モデル参照(検索画面)			
参照連動	-	項目.content	-

スクリプトの例 [1]

コントローラ>詳細画面表示タイミングのスクリプトで model1_p モデルのリストボックス型項目 s_item1 を操作する例は次のとおりです。

```
var id = model1_p.SItem1[0].id; /* ID部 */
var content = model1_p.SItem1[0].content; /* 内容部 */
var choose = model1_p.SItem1[0].choose; /* 常にtrue */
```

(303)

スクリプトの例 [2]

更新画面を開くタイミングでは、次のように配列を操作してください。

```
var array = model1_p.SItem1;
for (var i=0;i<array.length;i++) {
  var id = array[i].id; /* ID部 */
  var content = array[i].content; /* 内容部 */
  var choose = array[i].choose; /* 選択されたものがtrue、残りはfalse */
  ...(処理)...
}
```

(303) 項目にはキャメル記法が適用されます。

コンディションモデル

コンディションモデルはストアモデルとほぼ同じ形ですが、データベースに保存されません。

数値型と日付型の範囲検索では、二つの入力欄を管理します。Javaクラス内部では「項目名 + "jshparam1"」「項目名 + "jshparam2"」というフィールドが用意されます。

コンディションモデルは「(HTTP)セッション」に保存されます。このため、検索画面毎にコンディションモデルの状態が記憶されます。

リストモデル

検索結果（ストアモデルの集合）から、一覧表示画面に出力させる項目を選択し、「Item」という入れ物に格納しています。

表示側では、リストモデルに格納されているItemをループで処理することで一覧表示を実現します。

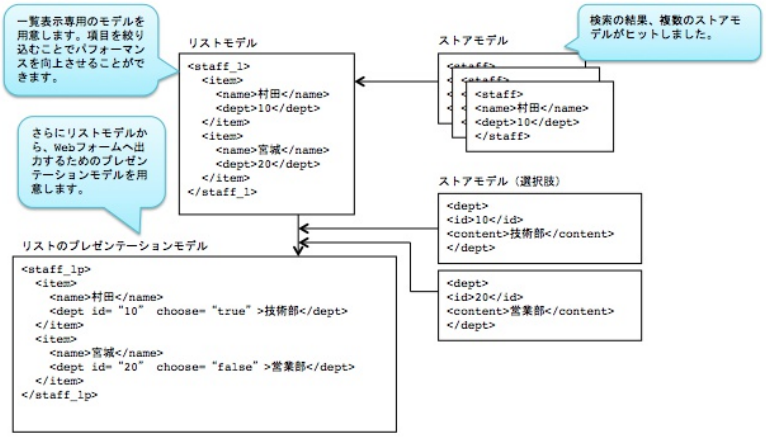


図 リストモデルのイメージ(XML形式で表現)

生成されるスクリプトファイルの詳細（スクリプト）

ジョブパラメータ (8.2.0)

ジョブパラメータをリクエストから取得することができます。

```
var dataMap = p.request.getAttribute("__jfcjob_dataMap");
if (dataMap != null) {
    print("value="+dataMap.get("key"));
} else {
    print("dataMap is null.")
}
```

- ヘルパ ＞ アップロード更新(開始時)のタイミングで上記のようなスクリプトを用いて、ジョブパラメータを取得するといった使い方ができます。
- スクリプトで null チェックは必ず行うようにしてください。例えばジョブではなく画面操作でのアップロード更新では、上記スクリプトの dataMap 変数は null となっています。

基本テンプレート

開発者によって記述されたコードは、次に示すテンプレートコードの内部に埋め込まれます。process という名前の関数の中にコードが含まれます。

(304)

```
function process() {
var ExcelFunction = Java.type("jp.jasminesoft.util.ExcelFunction");
var Jfcerror = Java.type("jp.jasminesoft.jfc.error.Jfcerror");
var Jfcwarn = Java.type("jp.jasminesoft.jfc.error.Jfcwarn");
var Jfcinfo = Java.type("jp.jasminesoft.jfc.error.Jfcinfo");
var JFCUtils = Java.type("jp.jasminesoft.jfc.JFCUtils");
var HolidayManager = Java.type("jp.jasminesoft.util.HolidayManager");

...ここに開発者独自のコードが埋め込まれる...

}
```

(305)

生成されるファイル

Designerで入力したスクリプトは、wagbyapp フォルダに存在するスクリプトファイル wagbyapp/webapp/$(APPNAME)/WEB-INF/script/<モデルID>/スクリプトファイル.js に上書き保存されます。

そのため再ビルドなしで、動作しているアプリケーションの挙動を確認することができます。

本番機のファイルは変更されません。同じ開発機上の wagbyapp フォルダのみ、変更されます。本番機へ反映させる場合は上のファイルを手動でコピーするか、改めてビルドを行ってから wagbyapp フォルダ全体を差し替えてください。

基本テンプレートを使わない

基本テンプレートを使わないようにすることもできます。例えば、独自の関数を用意する場合などです。

この場合、開発するコードは必ず function process() から開始してください。コードの先頭がこの記述で始まる場合、基本テンプレートは使わないと判断します。

例えば、次のように記述します。

```
function process() {
var ExcelFunction = Java.type("jp.jasminesoft.util.ExcelFunction");
var Jfcerror = Java.type("jp.jasminesoft.jfc.error.Jfcerror");
var Jfcwarn = Java.type("jp.jasminesoft.jfc.error.Jfcwarn");
var Jfcinfo = Java.type("jp.jasminesoft.jfc.error.Jfcinfo");
var JFCUtils = Java.type("jp.jasminesoft.jfc.JFCUtils");
var HolidayManager = Java.type("jp.jasminesoft.util.HolidayManager");

...開発者独自のコードを記述する...

}

function foo() {...独自の関数を記述する...}
function bar() {...独自の関数を記述する...}
```

定義した関数を（別のファイルに保存されたスクリプトから）呼び出すことはできません。

ルール

スクリプトに関する詳細なルールは次のとおりです。

- [基本テンプレート] 開発者が作成したスクリプトコードの先頭に "function process() {" が付与されます。また末尾に "}" が付与されます。Wagbyは、常にこの process 関数を呼び出します。

(304) ExcelFunction, Jfcerror, Jfcwarn, Jfcinfo, JFCUtils, HolidayManager は後述する「利用できる変数」で説明します。

(305) Java.type は特別な表記です。Java.type(FQCN形式のクラス名) と記述することで、Javaのクラスを表す変数を用意することができます。この変数を使って、オブジェクトを new 演算子で生成することができるようになります。

- ただし開発者が直接、process 関数の宣言 "function process() {" を明示的に記述することもできます。この場合、いくつかの変数の宣言も省略されるため、開発者自身で再定義してください。
- Designerでスクリプトを変更すると、同じ開発機上の wagbyapp フォルダに存在するスクリプトファイル wagbyapp/webapp/$(APPNAME)/WEB-INF/script/<モデルID>/スクリプトファイル.js が上書きされます。そのため、ビルドすることなく、動作しているアプリケーションの挙動を確認することができます。（注意：本番機のファイルは変更されません。同じ開発機上の wagbyapp フォルダのみ、変更されます。）

customizeフォルダとの関係

もし customize フォルダ（正確には customize/webapp/WEB-INF/script/<モデルID> フォルダ） にもスクリプトを保存していた場合は注意が必要です。Designerに用意された設定欄でスクリプトを編集したあとビルド処理を行うと、customize フォルダにあるスクリプトに上書きされます。すなわち、customizeフォルダにあるファイルが優先されます。

ドキュメンテーションコメント (8.3.0)

スクリプトエディタで、先頭にドキュメンテーションコメントを記述することができます。この内容がスクリプト一覧表で抽出されます。

書式は次の通りです。

```
/**
 * タイトル
 *（空行）
 * 本文
 * 複数行にまたがってよい。
 */
```

例を示します。

```
/**
 * 受注明細自動作成処理
 *
 * 受注登録時に、受注明細を自動作成する。
 */
```

- このドキュメンテーションコメントが未指定の場合、設計書出力ツールで作成されるスクリプト一覧表のタイトルと本文は空白になります。
- タイトルが存在しない場合は「(スクリプト設定ずみ)」というタイトルのみ記載されます。
- タイトルの最大文字列は50文字です。
- タイトルが指定の最大文字列を超えた場合、超えた分は本文に含まれるようになります。
- 本文中の改行はそのまま維持されます。
- ドキュメンテーションコメントの有無でスクリプトの動作は変わりません。（スクリプト自体に影響を与えるものではありません。）

Java 標準クラスを利用する

スクリプト中に、Javaが提供する標準クラスを利用することができます。これらは Java.type で再宣言する必要はありません。パッケージ名も含めたクラス名を使ってください。

```
var list = new java.util.ArrayList();
list.add("Apple");
list.add("Orange");
list.add("Banana");
print(list);
```

利用できる変数

スクリプト内で利用できる変数をまとめます。

名前	スクリプト内での表現	例	説明
ストアモデル	モデル名	customer	データベースに保存されるオブジェクト
プレゼンテーションモデル	モデル名_p	customer_p	Webフォームから入力された値。すべて文字列型。
コンディションモデル	モデル名_c	customer_c	検索条件が格納されたオブジェクト。
リストモデル	モデル名_l	customer_l	一覧表示される値が格納されたオブジェクト
エラーメッセージ	errorManager	errorManager	(エラーメッセージで解説)
SCREENTYPE	if (SCREENTYPE == "copy") {...}		SCREENTYPE関数と同じ。[例..]
UPLOADFILENAME 8.1.0	UPLOADFILENAME	UPLOADFILENAME	アップロード更新の「ファイル名」が格納されている。アップロード更新のタイミングで動作するスクリプト内で利用できる。
UPLOADFILEPATH 8.3.1	UPLOADFILEPATH	UPLOADFILEPATH	アップロード更新の「実ファイルのパス名」が格納されている。アップロード更新のタイミングで動作するスクリプト内で利用できる。
UPLOADFILEENTRY 8.3.1	UPLOADFILEENTRY	UPLOADFILEENTRY	アップロード更新でzipファイルがアップロードされた時、UPLOADFILENAMEはzipファイルのファイル名となる。そこで展開後のzipファイルで、今、処理されている(zipファイル内の)ファイル名を保持するものとして UPLOADFILEENTRY を用意した。アップロード更新のタイミングで、zipファイルがアップロードしているときにスクリプト内で利用できる。
UserElement 8.1.0	$owner	$owner.group().descend	階層グループで解説。
ActionParameter	p	p	p.appctxで(Springの)ApplicationContextを参照できるなどの使い方がある。
ExcelFunctionクラス (*1)	var s = ExcelFunction.FIND(...);		(Wagby関数の利用で解説)
Jfcerrorクラス (*1)	var error = new Jfcerror;		(エラーメッセージで解説)
Jfcwarnクラス (*1)	var warn = new Jfcwarn;		(エラーメッセージで解説)
Jfcinfoクラス (*1)	var info = new Jfcinfo;		(エラーメッセージで解説)

JFCUtilクラス (*1)	var s = JFCUtils.Message(p, "M0001")	(Wagby関数で解説)
HolidayManagerクラス (*1)	var d = HolidayManager.getInstance().getEigyoDayAsSQLD7)	(Wagby関数で解説)
ジョブパラメータ 8.2.0	var dataMap = p.request.getAttribute("__jfcjob_dataMap");	(ジョブパラメータで解説)

(306) (307) (308) (309) (310)

共通して利用できる関数 (8.3.0)

"環境 > スクリプト" では「共通関数」を用意することができます。

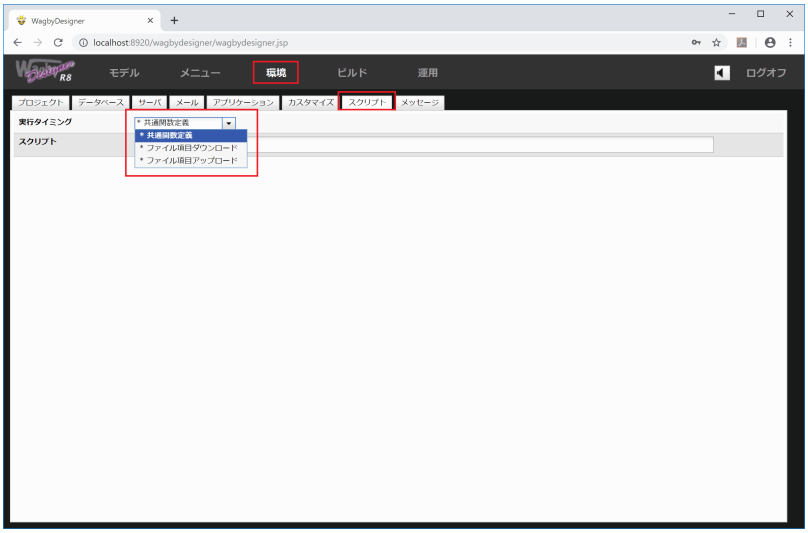


図 環境>スクリプトの定義

「共通関数」の実体は WEB-INF/script/myfunction.js ファイルです。このファイルが存在した時、その内容を各スクリプトに含めます。

(311)

利用例

あるスクリプトで、次のように記載します。

```
print(foo(1,2));
```

開発者は共通関数の定義で、関数 foo を用意することができます。

```
function foo(a1,a2) {
  return a1+a2;
}
```

スクリプトを実行すると、コンソールに 3.0 と表示されます。

R8.3.0 以前のバージョンをご利用の場合

---

(306) 1. ExcelFunction, Jfcerror, Jfcwarn, Jfcinfo, JFCUtils, HolidayManager は基本テンプレートで宣言したものです。

(307) ストアモデル以外のオブジェクトは、存在しない場合もあります。例えば検索画面の操作でない場合、コンディションモデルは含まれません。

(308) 状況によってはストアモデルの取得もできない場合があります。例えば削除処理の場合、ストアモデルは消えています。

(309) トランザクションスクリプトでは、トランザクションの範囲内として指定された複数のストアモデルを扱うことができます。

(310) トランザクションスクリプトでは、トランザクションの対象となる繰り返しコンテナもオブジェクトとして扱うことができます。

(311) 正確には、スクリプトの後にmyfunction.jsのコードがコピーされます。

Designer で設定するのではなく、開発者が作成した myfunction.js を customize/webapp/WEB-INF/script フォルダに保存してください。（インストール直後の状態では、WEB-INF フォルダ内に script フォルダはありませんので、これも作成してください。）次回のビルドから反映されるようになります。

ワークフローのイベントスクリプト

R8.2より、共通関数の定義(myfunction.js) はワークフローのフローイベントスクリプトでも利用できるようになりました。

(312)

独自に拡張したスクリプトファイルを読み込む (8.1.1)

独自に用意したスクリプトファイルを読み込むために load 関数を使うことができます。

```
load("wagby:xxx.js");
```

次のコードはいずれも /WEB-INF/script/sample/test.js を読み込みます。

```
load("wagby:sample/test.js");
load("wagby:/sample/test.js");
load("wagby:¥¥sample¥¥test.js");
load("wagby:../../WEB-INF/script/sample/test.js");
```

(313)

```
load("wagby:../test.js");//IllegalArgumentException が発生。
```

ファイルの区切り文字

"/" の利用を推奨します。Windows/Linux いずれの環境でも動作します。"¥¥" は Windows では動作しますが、Linux では動作しません。

ワークフローのイベントスクリプト

R8.2より、load 関数はワークフローのフローイベントスクリプトでも利用できるようになりました。

(314) (315)

関数 preprocess を用意する

ヘルパの次のタイミングで、関数 preprocess を用意することができます。処理の前に呼び出されます。

実行タイミング	ファイル名
登録	モデルIDHelper_beforeInsert.js
登録(初期データ作成)	モデルIDHelper_initialize.js
更新	モデルIDHelper_beforeUpdate.js
詳細画面表示	モデルIDHelper_beforeShow.js

開発者は WEB-INF/script/<モデルID> フォルダに作成された上記ファイルを直接、テキストエディタで修正し、関数 preprocess を用意してください。

```
function preprocess() {
var ExcelFunction = Java.type("jp.jasminesoft.util.ExcelFunction");
var Jfcerror = Java.type("jp.jasminesoft.jfc.error.Jfcerror");
```

(312) R8.1.2 利用時は、R8.1.2 向け公開パッチ適用でも myfunction.js を利用できます。それ以前の Wagby では利用できません。

(313) セキュリティの理由から、/WEB-INF/script より上位のフォルダにあるファイルを読み込むことはできません。例えば次のコードは実行時に IllegalArgumentException が発生します。

(314) R8.1.2 利用時は、R8.1.2 向け公開パッチ適用でも load 関数を利用できます。

(315) R8.1.2 より前の Wagby で、ワークフローのフローイベントスクリプトで load 関数を利用する場合は、フルパスでファイルの位置を記述するか、wagbyapp/bin を起点とした相対パスで記述してください。

```
var Jfcwarn = Java.type("jp.jasminesoft.jfc.error.Jfcwarn");
var Jfcinfo = Java.type("jp.jasminesoft.jfc.error.Jfcinfo");

...ここに開発者独自のコードを記述する...

}

function process() {
...
}
```
(316)

複数のスクリプト間でデータを受け渡す

スクリプトの内部で用意した変数は、このスクリプトの中だけで有効です。他のスクリプトから参照することはできません。

他のスクリプトにデータを渡す場合は、p.request オブジェクトを使います。p.request オブジェクトはユーザから要求のあった処理を行なっている間中、共通で利用できる「入れ物」です。

例 処理中の値によって画面遷移ルールを変更する

ヘルパのスクリプトで、p.request に何らかの制御フラグをセットします。

```
p.request.setAttribute("適当なキー名", 値);
```

コントローラのスクリプトでは、この値によって遷移先を変更することができます。

```
var flag = p.request.getAttribute("上のキー名");
if (flag === 求める値) {
  return 画面遷移先;
}
```

すでにあるキー名は使ってはいけません。値が上書きされてしまいます。開発者が独自のルールを決め、重複しないように配慮してください。例えばキー名の最初に "__プロジェクト名_" という接頭語を付与する、などは有効です。

Wagby関数を利用する（スクリプト）

利用方法

多くの関数は、接頭語 "ExcelFunction." を付与することで利用できます。

```
customer.autoid = ExcelFunction.NUMTOJTEXT(1234, 1, 0);
```

利用できない関数

次の関数はスクリプト内部では利用できません。

- 集合処理
- モデル参照

同様のことをJavaScriptプログラミングで記述することができます。SUM関数の例をこちらに示します。

(316) preprocess 関数を定義した場合は、修正したスクリプトファイルを customize フォルダ（正確には customize/webapp/WEB-INF/script/〈モデルID〉フォルダ）以下に保存するとよいでしょう。関数 preprocess を作成したあと、WagbyDesigner で編集するとスクリプトファイルは上書きされ、preprocess 関数の定義が消去されてしまうためです。

Javaクラスによって代替する関数

次の関数は「代替表記」を利用してください。　("p"は暗黙オブジェクトとして、スクリプト内で利用できます。)

区分	関数名	代替表記	注記
数値	RAND()	JFCUtils.__rand("項目名",p)	
数値	RANDBETWEEN(開始,終了)	JFCUtils.__randbetween目名",p,開始,終了)	
文字列	RANDSTR(桁数)	JFCUtils.__randstr("項目名",p,桁数)	
日付時刻	TODAY()	p.request.getAttribute	
日付時刻	NOW()	p.request.getAttribute	
日付時刻	WORKDAY(開始日,日数)	HolidayManager.getInst始日,日数)	ate(開
日付時刻	NETWORKDAYS(開始日,終了日)	HolidayManager.getInst始日,終了日)	
権限	HASMODELPERMISSION(引数文字列)	JFCUtils.isValidPermis引数文字列)	
権限	HASPRINCIPAL(引数文字列)	JFCUtils.hasPrincipal(引数文字列)	
アカウント	USERNAME()	((p.user!=null)?p.user.getUsername():"	
アカウント	USERINFO(引数文字列)	JFCUtils.getJuserAttri引数文字列)	
アカウント	JGROUPID()	JFCUtils.getJgroupid(p	
画面制御	BROWSERTYPE()	p.request.getAttribute	
画面制御	EVENT()	JFCUtils.getEvent(p)	
画面制御	RESULTSIZE()	JFCUtils.getResultSize	
画面制御	SCREENID()	p.request.getAttributep.request.getAttribute	(*4)
画面制御	SCREENTYPE()	p.request.getAttribute	
画面制御	WIZARDPAGE()	JFCUtils.getWizardPage	
国際化	LOCALE()	JFCUtils.getLocale(p)	
リソース	MESSAGE()	JFCUtils.Message(p,"M0001")JFCUtils.Messa"M0002", "hello")	
リソース	ERRORMESSAGE()	JFCUtils.ErrorMessage("E0001")JFCUtils.Error"E0002", "hi")	
ワークフロー	FLOWSTATUS("モデル名",主キー)	JfcworkstateServiceImpデル名",主キー)	(*1) "モ
ワークフロー	WF_CURRENT_USERID("モデル名",主キー)	JfcworkstateServiceImpデル名",主キー)	(*1)
ワークフロー	WF_NEXT_USERID("モデル名",主キー)	JfcworkstateServiceImpデル名",主キー)	(*1)
ワークフロー	WF_USERID()	モデル名(キャメル記法)Helper.wf_nextuseriデル名",ストアモデル)	(*2)
ワークフロー	WF_CURRENT_GROUPID("モデル名",主キー)	JfcworkstateServiceImpデル名",主キー)	(*1)
ワークフロー	WF_NEXT_GROUPID("モデル名",主キー)	JfcworkstateServiceImpデル名",主キー)	(*1)
ワークフロー	WF_GROUPID()	モデル名(キャメル記法)Helper.wf_nextgroupデル名",ストアモデル)	(*2)

ワークフロー	WF_IS_ADMITUSER()	モデル名(キャメル記法)Helper.wf_isadmitusデル名",ストアモデル)	(*2)
外部連携	REQUEST()	p.request	
外部連携	SESSION()	p.request.getSession()	(*3)
外部連携	ISREST()	JFCUtils.isRestAPI(p)	
外部連携	HTTPSEND(url, "ユーザID:パスワード")	ExcelFunction.httpsendurl, "ユーザID:パスワード")	
外部連携	HTTPSEND(url, 接続タイムアウト, 読み込みタイムアウト, リトライ回数)	ExcelFunction.httpsendurl, 接続タイムアウト, 読み込みタイムアウト, リトライ回数)	
外部連携	HTTPSEND(url, 接続タイムアウト, 読み込みタイムアウト, リトライ回数, "ユーザID:パスワード")	ExcelFunction.httpsendurl, 接続タイムアウト, 読み込みタイムアウト, リトライ回数, "ユーザID:パスワード")	
その他	CLIENTHOSTNAME()	JFCUtils.getRealIP(p)	
その他	SERVERHOSTNAME()	JFCUtils.getServerIP()	

1. スクリプトの先頭でJavaClass型の変数を宣言してください。var JFCUtils = Java.type("jp.jasminesoft.wagby.app.jfcworkstate.JfcworkstateServiceImple"); と記述します。パッケージ名を変更している場合は、"jp.jasminesoft.wagby"を適切に読み替えてください。
2. スクリプトの先頭でJavaClass型の変数を宣言してください。var JFCUtils = Java.type("jp.jasminesoft.wagby.app.<モデルID>.<モデルID>Helper"); と記述します。パッケージ名を変更している場合は、"jp.jasminesoft.wagby"を適切に読み替えてください。
3. 関数 SESSION() は HttpSession クラスを指します。Hibernate の Session ではありません。
4. Myが接頭語に付与されたカスタマイズクラスが存在したとき、__jfc_screen_id の戻り値は My が付与されます。一方で __jfc_screen_nid を指定すると My が除かれた（正規化された）戻り値に統一されます。

例

呼び出される画面によって、処理をスキップさせる

```
var screen_id = p.request.getAttribute("__jfc_screen_id");
if (screen_id != null && screen_id == "XXX") { /* 画面名を指定する */
    return;
}
... 処理の本体 ...
```

(317)

スクリプトを用いた主キーの拡張（スクリプト）

例

レポートモデル内の「ステータス」項目によって、次のルールが適用されるものとします。

新規顧客

N1404001（1桁目:N, 2～5桁目:年月, 6～8桁目:001...999連番）

既存顧客

E1404001（1桁目:W, 2～5桁目:年月, 6～8桁目:001...999連番）

このように、区分毎に連番を用意します。

(317) 画面名の詳細はSCREENID関数の説明をお読みください。

図に新規顧客のレポート登録を、図に既存顧客のレポート登録の例を示します。いずれも連番は001となっており、独立して管理されます。

図 新規顧客のレポート, 主キーがN1404001となっている

図 既存顧客のレポート, 主キーがE1404001となっている

定義方法

「レポート」モデルの全体構成を図に示します。

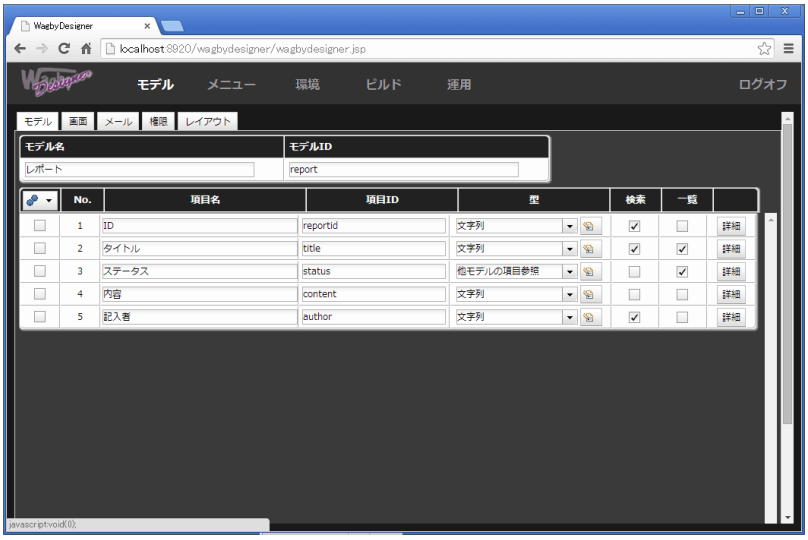


図 レポートモデル

主キー項目「ID」の、主キーに関する詳細は設定しません。これをスクリプトとして実現します。[後述]

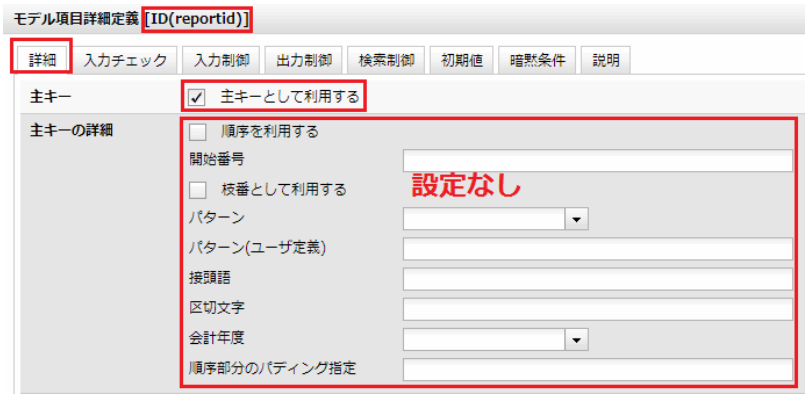


図 主キーの詳細設定を行わない

「ID」項目を入力時は隠し項目・読込専用として扱います。これによって主キーですが入力なし（自動設定）を実現できます。

図 隠しと読込専用の設定

「ステータス」モデルの初期値を図に示します。″新規″ はコード1を、″既存″ はコード2をそれぞれ割り当てます。

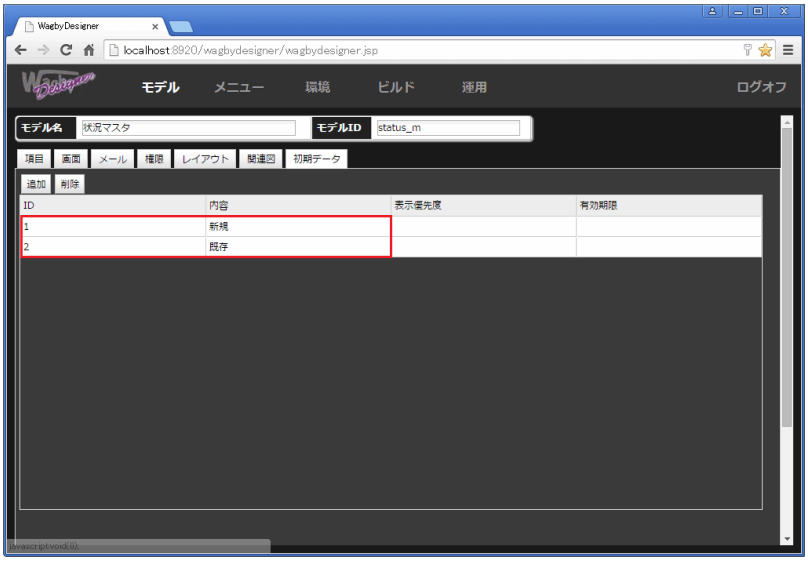


図 ステータスモデルの初期値

順序を定義する

区分毎の連番を用意するため、Wagbyが用意した順序テーブル "seq" に独自の順序 "report_seq1" と "report_seq2" を登録します。それぞれ開始番号を "1" とします。

独自の DDL を発行するために、$(DEVHOME)/customize/webapp/WEB-INF/export/conf/initdb_ex.xml を用意します。(*1)

(318)

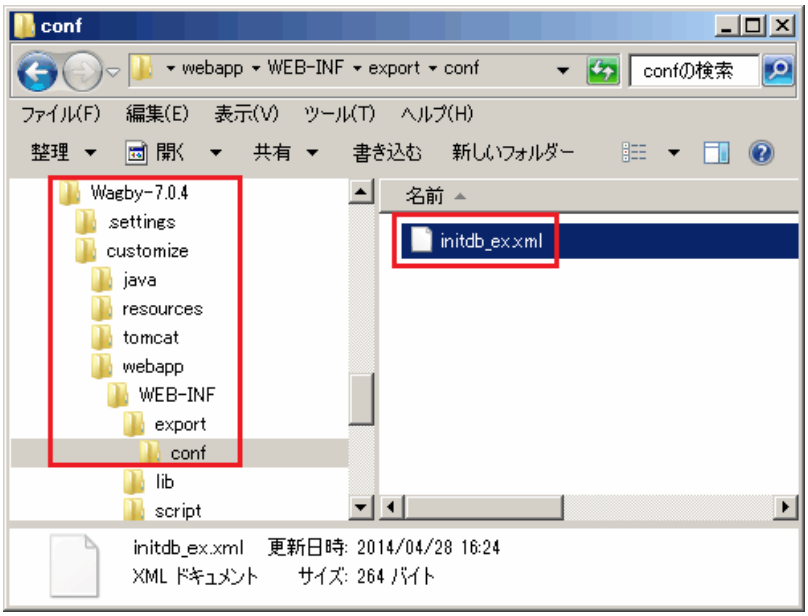


図 initdb_ex.xmlを新規作成する

次の内容とします。

(318) exportフォルダならびに export/conf フォルダは手動で作成してください。

図 initdb_ex.xmlの内容

```
<?xml version="1.0" encoding="UTF-8"?>
  <sql group="data"
        acceptcommand="sql sql-create-after" denycommand="all"
        tablename="seq">
INSERT INTO &quot;seq&quot; VALUES('report_seq1',1);
INSERT INTO &quot;seq&quot; VALUES('report_seq2',1);
  </sql>
```

ビルド後は以下に配置されます。(*1) wagbyapp/webapps/wagby/WEB-INF/export/conf/initdb_ex.xml

(319)

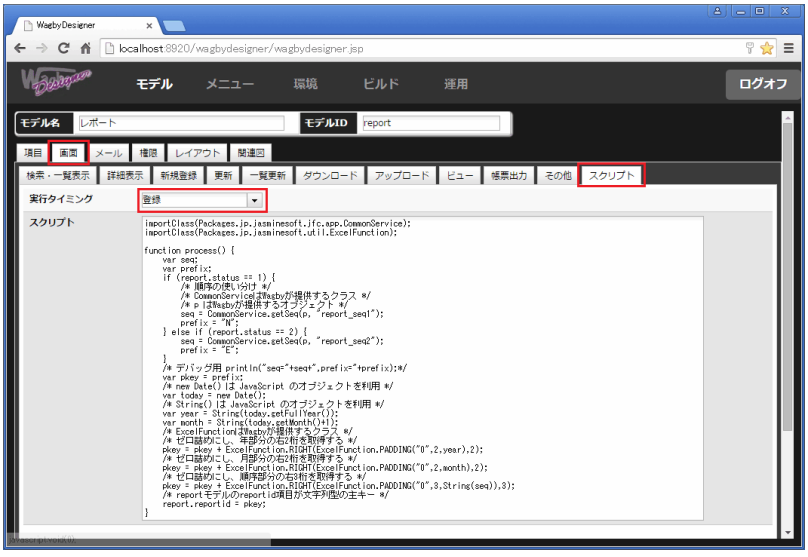
ここでは順序テーブル "seq" を使いましたが、お使いのデータベースが順序の仕組みをもっている場合、CREATE SEQUENCE で順序を作成するとよいでしょう。(Oracle, PostgreSQL, SQL Server 2012 は順序を提供しています。)

このようにinitdb_ex.xmlは、sql要素中に独自のDDLを記述することができます。Wagbyによるデータベース作成時に、順序やビューの作成など、独自のDDLを発行できます。[関連ページ:インポートとエクスポート機能活用ガイド(R8) > 設定ファイル > 拡張ファイル initdb_ex.xml]

テーブル名を囲むダブルクォートの有無などは、ご利用になるデータベースによって異なります。

スクリプトを実装する

「画面 > スクリプト」を開きます。新規登録時のスクリプトコードを作成します。 これは新規登録処理の最後に呼び出されますので、主キーの値をセットするのに都合のよいポイントです。



(319) 1. R7.12.6/R8.0.4 より前のバージョンの Wagby で Java 9 以上の環境を利用する場合、initdb_ex.xml を $(DEVHOME)/customize/tomcat/bin に配置してください。ビルド後は wagbyapp/bin にコピーされます。(R7.12.6/R8.0.4 以降の Wagby では、この手順は不要です。)

図 主キー生成のためのコード

```
var CommonService = Java.type("jp.jasminesoft.jfc.app.CommonService");

var seq;
var prefix;
if (report.status == 1) {
    /* 順序の使い分け */
    /* CommonServiceはWagbyが提供するクラス */
    /* p はWagbyが提供するオブジェクト */
    seq = CommonService.getSeq(p, "report_seq1");
    prefix = "N";
} else if (report.status == 2) {
    seq = CommonService.getSeq(p, "report_seq2");
    prefix = "E";
}
/* デバッグ用 print("seq="+seq+",prefix="+prefix);*/
var pkey = prefix;
/* new Date() は JavaScript のオブジェクトを利用 */
var today = new Date();
/* String() は JavaScript のオブジェクトを利用 */
var year = String(today.getFullYear());
var month = String(today.getMonth()+1);
/* ExcelFunctionはWagbyが提供するクラス */
/* ゼロ詰めにし、年部分の右2桁を取得する */
pkey = pkey + ExcelFunction.RIGHT(ExcelFunction.PADDING("0",2,year),2);
/* ゼロ詰めにし、月部分の右2桁を取得する */
pkey = pkey + ExcelFunction.RIGHT(ExcelFunction.PADDING("0",2,month),2);
/* ゼロ詰めにし、順序部分の右3桁を取得する */
pkey = pkey + ExcelFunction.RIGHT(ExcelFunction.PADDING("0",3,String(seq)),3);
/* reportモデルのreportid項目が文字列型の主キー */
report.reportid = pkey;
```

- RIGHT 関数は、文字列の右からN桁までを取り出します。
- PADDING 関数は、文字列に指定文字を詰めて、切り出します。

(320)

サブデータベースで順序を使う（スクリプト）

準備

スクリプトで主キーの値をセットする場合は、利用者が主キー項目に値を入力しないよう、下記の設定を追加することを推奨します。

- モデル項目詳細定義 > 入力制御 > 隠し・読み込み専用を設定する。
- 全画面 読み込み専用を "有効" にする。
- 登録画面 隠し項目を "有効" にする。
- コピー登録画面 隠し項目を "有効" にする。

例 PostgreSQLからserial値を取得する

顧客モデル (customer) の主キー項目 (customerid) に順序の値をセットします。 「ヘルパ > 登録」のタイミングに次のスクリプトを用意します。

```
// デフォルトで用意されている session はメインデータベース用です。
// サブデータベース1 にアクセスするには sessionFactory2 から session を取得して下さい。
// サブデータベース2 では sessionFactory3 となります。
var session2 = p.appctx.getBean("sessionFactory2").getCurrentSession();
var nextval= session2.createSQLQuery(
```

(320) CREATE SEQUENCE で順序を作成した場合、CommonService.getSeq ではなく CommonService.getSequence を使ってください。

```
    "SELECT nextval('serial')").uniqueResult();
  //print("nextval=" + nextval);
  customer.customerid = nextval;
```

PostgreSQL 以外のデータベース利用時も、上の SQL 文を（利用するデータベースに）合わせることで対応できます。

アップロード更新を有効にしたモデルの場合

アップロード更新では SessionFactory#getCurrentSession() メソッドは利用できません。そのため、アップロード更新を有効にしたモデルの場合は次のようにスクリプトを修正してください。

```
// デフォルトで用意されている session はメインデータベース用です。
// サブデータベース1 にアクセスするには sessionFactory2 から session を取得して下さい。
// サブデータベース2 では sessionFactory3 となります。
var session2;
if (SCREENTYPE == "uploadUpdate") {
  // データの INSERT 処理とは別トランザクションとなる。
  session2 = p.appctx.getBean("sessionFactory2").openSession();
} else {
  session2 = p.appctx.getBean("sessionFactory2").getCurrentSession();
}
try {
  var nextval = session2.createSQLQuery(
   "SELECT nextval('serial2')").uniqueResult();
  print("nextval=" + nextval);
  customer.customerid = nextval;
} finally {
  if (SCREENTYPE == "uploadUpdate") {
    session2.close();
  }
}
```

openSession で開いた Hiberate Session は、必ず close するようにしてください。

例 MySQLのAUTO_INCREMENTを利用する

Wagby で MySQL を利用する場合、AUTO_INCREMENT 機能による採番方式は使いません。代わりに「順序値を保持する管理テーブル」を用意し、このテーブルから順序を取得するようにしています。

しかしサブデータベースで MySQL を利用し、かつ既存テーブルが AUTO_INCREMENT を有効にしている場合、Wagby にこの機能を認識させる必要があります。ここではこの方法を説明します。

MySQL が提供する AUTO_INCREMENT 機能は次のような特徴があります。

- CREATE TABLE 時に主キーカラムに AUTO_INCREMENT の指定を行う。
- 該当カラムが NULL または 0 の時に自動的に数値がセットされる。
- セットされる値は1ずつ増加する連番となる。

問題点

既存テーブルに AUTO_INCREMENT の指定がある場合、Wagby では主キーを何もセットしなければ(0となるので)、自動的に採番されます。 しかし Wagby では、ここで採番された主キーの値を知る方法がないため、データ登録後の詳細画面の遷移に失敗し、「対象データは既に削除されているか、対象データを閲覧する権限がありません。」というエラーメッセージが表示されてしまいます。

対応方法

Wagby が生成する <モデルID>.hbml.xml ファイル (Hibernate 用のマッピングファイル) をカスタマイズします。 customer モデルの customerid 項目のカスタマイズ例は次のようになります。

customer.hbm.xml

```
 ...
 <class name="Customer" table=""customer"">
      <id name="customerid_">
```

```
            <column name=""customerid""/>
            <generator class="identity"/>
        </id>
    ...
```

上にあるように generator 要素を追加してください。修正したファイルを customize/webapp/WEB-INF/classes フォルダに配置することでビルド時にこの（修正）ファイルが使われるようになります。

(321) (322) (323)

スクリプトでの初期化処理（スクリプト）

定義方法

初期化処理をスクリプトコードとして記述することができます。

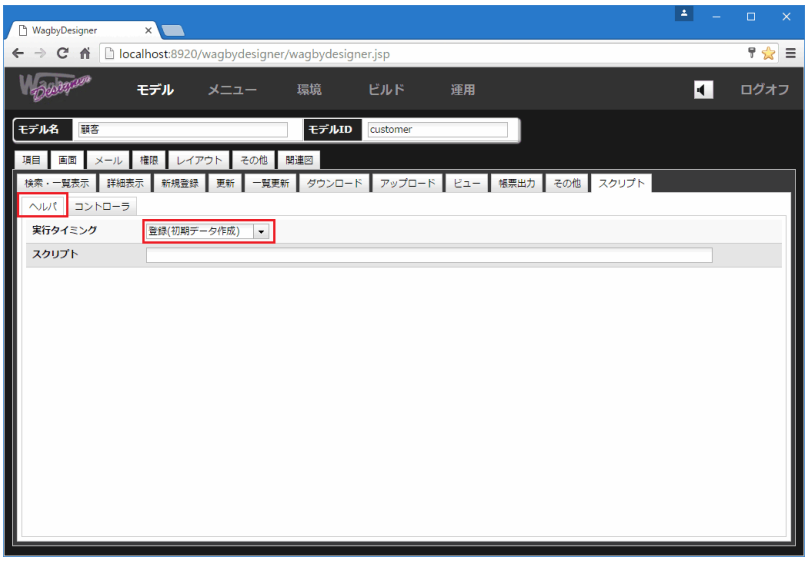


図 スクリプト記述欄

ここに書いた処理は、Designerで設定した初期化処理の後に実行されます。

Designerでスクリプトを修正すると、開発機上のアプリケーションへ即座に反映されます。ビルド処理なしで修正を確認することができます。

スクリプト内では、モデル名に対応した「オブジェクト」を利用できます。

モデルを新規に作成する場合

図のスクリプトを利用する場合、対象オブジェクトの生成は（自動生成されたコードが行っているため）完了しており、値をセットする処理を追加するだけです。

しかしスクリプト中で開発者が自らオブジェクトを(new演算子を用いて)生成したい場合があります。この場合に、Designerで設定した初期値の設定をもれなく行う方法を説明します。

```
var customerClass = Java.type("jp.jasminesoft.wagby.model.customer.Customer");
var customer = new customerClass();
```

(321) Hibernate マッピングファイルをカスタマイズした場合、その後の項目の追加や変更に応じて、開発者が常に同期をとるようにしてください。

(322) LAST_INSERT_ID() 関数を利用して AUTO_INCREMENT の値を取得する方法もありますが、これは接続毎の値の管理となるので、他のシステムからデータの登録が行われた場合の主キーの増加分は LAST_INSERT_ID() に反映されないという問題があります。

(323) MySQL の AUTO_INCREMENT 機能は、ここで説明したように採番された値の取得が行いにくいことから、利用については一長一短があります。Wagby ではこの機能は採用せず、独自の管理テーブルで順序値を保持しています。この方法はスクリプトでこの管理テーブルへアクセスする SQL を実行することで現在の順序値を取得できるというメリットもあります。

```
var helper = p.appctx.getBean("CustomerHelper");
helper.initialize(customer, p);
```

helper.initialize() を実行することで Designer で定義した初期値(登録画面表示時)の値がセットされます。 グループ権限などを利用しているモデルでは、このタイミングで「データ所有者」などの値もセットされます。

[例] 親子モデル同時更新機能で親モデル登録時、子モデル毎に異なる初期値を設定する

外部キーにより連携した子モデルの「外部キーの同時更新機能 > 初期表示数」を設定することで、親モデルの新規作成時に子モデルのレコードが同時に N 個作成されます。

このとき、子モデルの要素ごとに異なる値を設定するスクリプトは次のようになります。（子モデル名を child とします。子モデルには項目 item1 が用意されているものとします。）

```
var count = p.request.getAttribute("_child_ulp_count");
if (count != null) {
    child.item1 = count;
}
```

親子同時更新画面では、対象となる子モデルの要素番号が次の名前で request オブジェクトに含まれています。値は整数型です。

```
_子モデルID_ulp_count
```

これを使って、何番目の子モデルを対象とするかを知ることができます。

入力チェック・整合性チェック（スクリプト）

定義方法

入力チェック処理をスクリプトコードとして記述することができます。

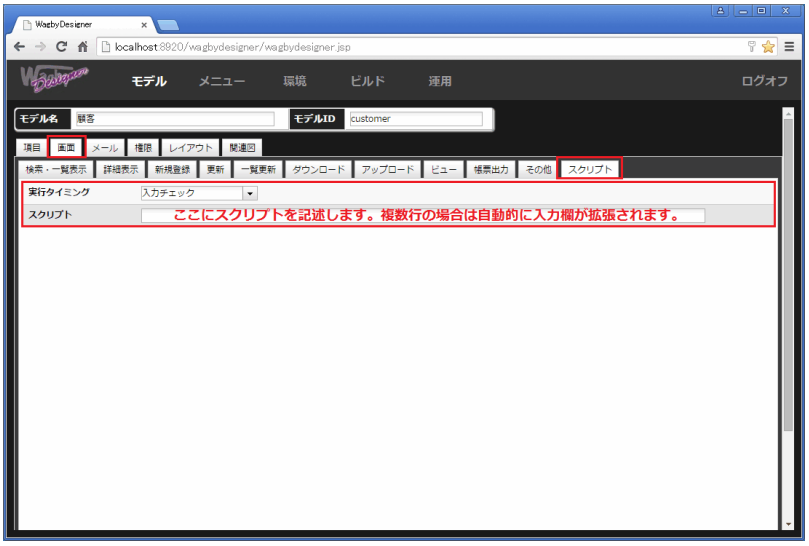


図 スクリプト記述欄

ここに書いた処理は、その他の（必須チェックや、文字数チェックといった設定で記述できる）チェックの後に実行されます。

(324)

Designerでスクリプトを修正すると、開発機上のアプリケーションへ即座に反映されます。ビルド処理なしで修正を確認することができます。

スクリプト内では、モデル名に対応した「オブジェクト」を利用できます。

[例] EntityServiceとCriteriaを用いた存在チェック

商品 (product) モデルを想定します。主キー項目 pkey は順序による自動採番とし、また、主キーではない項目「商品コード (pcode)」を持つとします。ここで、重複する商品コードを事前にチェックし、重複があった場合はエラーメッセージを表示するスクリプトを紹介します。

```
var screentype = p.request.getAttribute("__jfc_screen_type");/* SCREENTYPE 関数のこと */

var service = p.appctx.getBean("ProductEntityService");
var criteriaConverter = p.appctx.getBean("ProductCriteriaConverter");
var criteria = criteriaConverter.defaultCriteria();
var metaClass = Java.type("jp.jasminesoft.wagby.model.product.ProductMeta");
var meta = new metaClass();

criteria.eq(meta.pcode, product.pcode);/* pcode の重複を確認する。*/

var list = service.find(criteria);
if (list !== null && list.size() >= 1) {
    if (screentype === "update" && list.size() === 1) {
        var o = list.get(0);
        if (o.pkey === product.pkey) {
            /* 自分自身なのでエラーとしない */
            return;
        }
    }
    var error = new Jfcerror();
    error.content = "商品コード '"+product.pcode+"' が重複しています。";
    p.errors.addJfcerror(error);
}
```

- クライテリア (criteria) は検索条件を扱います。詳細は"クライテリアを利用する"をお読みください。
- 「一意制約」の設定でも同じことができます。一意制約は、データベースにエラーを判定させる方法です。上のスクリプト方式はアプリケーションで判断するため、複雑な条件設定や、任意のエラーメッセージを用意できるといった拡張性があります。

[例] 一覧更新画面の入力チェック (1) 画面に表示されているデータのみ

ここでは model1 モデルのラジオボタン項目「有効・無効 (enabled)」がセットされたデータが1つだけかどうかをチェックする例を説明します。有効・無効(enabled)項目の値が "1" となっているデータが2件以上あった場合、エラーメッセージを表示して、編集画面に戻します。

(325)

(324) 正確には、入力チェックの最後の処理となります。

(325) なお、このチェック処理は、画面に表示されている分だけを対象としています。そのため、この一覧更新画面にすべてのデータが表示されるようになっていると仮定します。1ページに表示される編集対象データは「同時表示数のパターン」で指定することができます。

図 一覧更新画面の入力チェック

入力チェックスクリプトのため、ここで値を書き換えることはできません。値の参照のみを行うことができます。

データベースコミット前

「コントローラ ＞ 一覧更新 ＞ データベースコミット前」を用意します。

```
function process() {
    var Jfcerror = Java.type("jp.jasminesoft.jfc.error.Jfcerror");
    var Item = Java.type("jp.jasminesoft.wagby.model.model1_ulp.Item");

    // Map key is primary key.
    // 編集中更新データのUlpモデルのItem
    // Map<String, jp.jasminesoft.wagby.model.model1_ulp.Item>
    var updateMap = p.pageMap.get("update_model1_ulp");
    // 編集中新規登録データのUlpモデルのItem
    // keyがnullのデータは既存データに紐づかない新規登録データ
    // keyがnullでないデータは、既存データに紐づくデータ
    // Map<String, List<jp.jasminesoft.wagby.model.model1_ulp.Item item>>
    var insertMap = p.pageMap.get("insert_model1_ulp");
    // 編集中削除データのUlpモデルのItem
    // Map<String, jp.jasminesoft.wagby.model.model1_ulp.Item>
    var deleteMap = p.pageMap.get("delete_model1_ulp");
    // 更新データの編集元のEntity
    // Map<String, Model1>
    var updateSrcMap = p.pageMap.get("update_src_model1_ulp");
    // 新規登録データの編集元のEntity
    // Map<String, List<Model1>>
    var insertSrcMap = p.pageMap.get("insert_src_model1_ulp");
    // 画面に表示されているデータに対応する編集中データのEntity
    // List<Model1>
    var datas = p.pageMap.get("datas_model1_ulp");
    // 画面に表示されているデータに対応する編集元データのEntity
    // List<Model1>
    var datasrc = p.pageMap.get("datasrc_model1_ulp");
    // 画面に表示されているデータのUlp
    // jp.jasminesoft.wagby.model.model1_ulp.Model1Ulp
    var _listp = p.pageMap.get("model1_ulp");

    var errormsg = "エラーが発生しました。有効・無効にて有効とするのは1か所のみとしてください。"

    var items = _listp.item;
    var enableditems = new Array();
    for (i=0 ; i < items.length ; i++) {
        var item = items[i];
        //print("item="+item);
        if (item.jfcdelete) {
            continue;
        }
```

```
        var enabled = getChoosedEnabled(item);
        //print ("item.enabled.id="+enabled.id)
        if (enabled.id === 1) {
            enableditems.push(item);
        }
    }
    if (enableditems.length > 1) {
        var error = new Jfcerror();
        error.content = errormsg;
        p.errors.addJfcerror(error);// 画面上部に表示されるエラーメッセージ。

        // 画面内のデータごとにエラーメッセージを表示するために、insertMap,updateMap
        // 内のItemにエラーメッセージをセットする。
        // do_setDataにて、これらのデータから表示用のデータにエラーメッセージがセットされるた
め。
        for (i=0 ; i < enableditems.length ; i++) {
            var item = enableditems[i];
            var jfcupdatetype = item.jfcupdatetype;
            var item1 = null;
            if (jfcupdatetype === "insert" || jfcupdatetype === "copy") {
                var itemlist = insertMap.get(item.jfcpkey);
                //print("itemlist="+itemlist)
                //print("item.pkeygroupnum="+item.jfcpkeygroupnum)
                if (itemlist !== null) {
                    item1 = itemlist.get(item.jfcpkeygroupnum-1);
                }
            } else if (jfcupdatetype === "update") {
                item1 = updateMap.get(item.jfcpkey);
                if (item1 === null) {// 未編集のデータであったため、新しい Item を用意する。
                    item1 = new Item(item);
                    updateMap.put(item.jfcpkey, item1);
                }
            }
            if (item1 !== null) {
                //print("item1="+item1)
                item1.jfcerror = errormsg;
            }
        }
    }
}

function getChoosedEnabled(item) {
    var enabledary = item.enabled;
    var enabled;
    for (j=0; j < enabledary.length; j++) {
        if (enabledary[j].choose) {
            enabled = enabledary[j];
            break;
        }
    }
    return enabled;
}
```

- スクリプトの前半部分var xxx = p.pageMap.get("...")は、一覧更新処理を行っている UpdateListModel1Controller クラスの動作のために必要な内部の変数値を取得しています。スクリプト内ですべての変数を利用しているわけではありませんが、このような変数が利用できるということを示すため、すべての変数を用意しています。詳細は各行のコメントをお読みください。
- 画面に表示された1ページのデータをvar _listp = p.pageMap.get("model1_ulp")にて取得しています。接尾語に "_ulp" が付与されたプレゼンテーションモデルは一覧更新画面のみで利用できるもので、現在表示されているページのデータが格納されています。
- 保存ボタン押下時に、UpdateListModel1Controllerクラスのdo_updateメソッドが呼び出されます。この処理で、HttpServletRequestに格納された修正データが変数 _listp に反映されるようになっているため、上のスクリプトで取得した _listp で編集データとして取得することができるようになっています。

- 変数 _listp はさらに内部で複数の行データを保持しています。var items = _listp.item;とすることで、この行データを配列として取得できます。この実態は jp.jasminesoft.wagby.model.model1_ulp.Item クラスです。
- スクリプトの中盤では、取得した items を for ループで一件ずつ処理しています。ここで item.jfcdelete が true の場合とは、削除のチェックボックスがチェックされているデータです。今回は対象としないようにしています。
- getChoosedEnabled関数は、本スクリプトの後半部分に定義しています。item から enabled 項目の値を取得します。enabled項目はラジオボタンとなっているため、その戻り値は jp.jasminesoft.wagby.model.model1_ulp.Enabled の配列です。この配列内から enabled.choosed が trueとなっているデータを見つけます。具体的には enabled.id の値が 1 の場合、ラジオボタンで"有効"が選択されているとみなしているため、enableditems に item を格納するようにしています。
- enableditems.length のサイズが 1 より大きい場合、"有効"が選択されたラジオボタンが 2 つ以上あるため、エラーとしています。（今回のエラーチェックの要件となります。）
- スクリプトの後半で、エラーのあったデータに対して、エラーメッセージをセットしています。このエラーメッセージは _listp に格納された item にセットするのではなく、画面表示用のデータを格納するための insertMap および updateMap 変数を用いる必要があります。
- _listp内の item の jfcupdatetype を使うと、このデータが更新、新規登録、コピー登録のいずれかを知ることができます。
- また item の jfcpkey は主キーが格納されています。この値をキーに insertMap や updateMap から表示用の item を取得します。
- insertMap はコピー登録にも対応しています。コピー登録の場合、一つのデータに対して複数のコピー登録が可能であるため、insertMapには List<Item> が格納されます。pkeygroupnum の値を使って、何番目のデータと紐づいているかを知ることができます。
- これらの操作を行なって insertMap および updateMap から対応する Item を取得し、エラーメッセージをセットします。
- updateMap に（主キーに対応する）Item が存在しなかった場合は、未編集のデータであることを意味します。上のスクリプトでは未編集のデータだった場合は新しい Item を生成し、updateMap にセットしています。この場合、画面上は未編集であっても、このスクリプトによって編集中として扱われるようになります。

更新の実行

「コントローラ > 一覧更新 > 更新の実行」を用意します。

上で説明した「データベースコミット前」のスクリプトで、insertMap や updateMap に直接、エラーメッセージをセットしていました。「更新の実行」スクリプトで、これらのエラーメッセージを消去しておきます。（消去しないとエラーメッセージが残ってしまうためです。）

```
function process() {
    var Jfcerror = Java.type("jp.jasminesoft.jfc.error.Jfcerror");

    // Map key is primary key.
    // 編集中更新データのUlpモデルのItem
    // Map<String, jp.jasminesoft.wagby.model.model1_ulp.Item>
    var updateMap = p.pageMap.get("update_model1_ulp");
    // 編集中新規登録データのUlpモデルのItem
    // keyがnullのデータは既存データに紐づかない新規登録データ
    // keyがnullでないデータは、既存データに紐づくデータ
    // Map<String, List<jp.jasminesoft.wagby.model.model1_ulp.Item item>>
    var insertMap = p.pageMap.get("insert_model1_ulp");
    // 編集中削除データのUlpモデルのItem
    // Map<String, jp.jasminesoft.wagby.model.model1_ulp.Item>
    var deleteMap = p.pageMap.get("delete_model1_ulp");
    // 更新データの編集元のEntity
    // Map<String, Model1>
    var updateSrcMap = p.pageMap.get("update_src_model1_ulp");
    // 新規登録データの編集元のEntity
    // Map<String, List<Model1>>
    var insertSrcMap = p.pageMap.get("insert_src_model1_ulp");
    // 画面に表示されているデータに対応する編集中データのEntity
    // List<Model1>
    var datas = p.pageMap.get("datas_model1_ulp");
    // 画面に表示されているデータに対応する編集元データのEntity
    // List<Model1>
    var datasrc = p.pageMap.get("datasrc_model1_ulp");
```

```
    // 画面に表示されているデータのUlp
    // jp.jasminesoft.wagby.model.model1_ulp.Model1Ulp
    var _listp = p.pageMap.get("model1_ulp");

    var errormsg = "エラーが発生しました。有効・無効にて有効とするのは1か所のみとしてください。"

    // insertMap, updateMapからエラーメッセージを削除する。
    // エラーメッセージが残っていると、修正してもエラー状態となり、データベース保存への
    // トランザクションが実行されないため。
    var updateitems = updateMap.values().toArray();
    for (i=0 ; i < updateitems.length ; i++) {
        var item = updateitems[i];
        //print("update item="+item);
        if (item.sizeJfcerror() == 1 && item.getJfcerror(0) === errormsg) {
            item.clearJfcerror();
        }
    }

    var insertitemlists = insertMap.values().toArray();
    for (i=0; i < insertitemlists.length; i++) {
        var itemlist = insertitemlists[i]
        if (itemlist === null) {
            continue;
        }
        for (j=0; j < itemlist.size(); j++) {
            var item = itemlis.get(j);
            //print("insert item="+item);
            if (item.sizeJfcerror() == 1 && item.getJfcerror(0) === errormsg) {
                item.clearJfcerror();
            }
        }
    }
}
```

・　このスクリプトでは insertMap や updateMap から Item を取得し、今回表示したエラーメッセージと同じ文言が設定されている場合に限り、エラーメッセージをクリアする処理を行っています。クリア処理は item に用意された clearJfcerror メソッドを呼び出します。

よりシンプルなスクリプト

これまでの説明は、updateMap や insertMap といった（一覧更新画面の内部で利用している）表示用データを操作していました。次に紹介するスクリプトは、これらの内部変数を使わないパターンです。

updateMap や insertMap を意識しないでよいという要件であれば、こちらの方式を推奨します。

データベースコミット前

```
function process() {
   var Jfcerror = Java.type("jp.jasminesoft.jfc.error.Jfcerror");

   // 画面に表示されているデータのUlp
   // jp.jasminesoft.wagby.model.model1_ulp.Model1Ulp
   var _listp = p.pageMap.get("model1_ulp");

   var errormsg = "エラーが発生しました。有効・無効にて有効とするのは1か所のみとしてください。"

   var items = _listp.item;
   var count = 0;
   for (i=0 ; i < items.length ; i++) {
       var item = items[i];
       //print("item="+item);
       if (item.jfcdelete) {
           continue;
       }
       var enabled = getChoosedEnabled(item);
```

```
        //print("item.enabled.id="+enabled.id);
        if (enabled.id === 1) {
            count++;
        }
    }
    if (count > 1) {
        var error = new Jfcerror();
        error.content = errormsg;
        p.errors.addJfcerror(error);// 画面上部に表示されるエラーメッセージ。
    }
}

function getChoosedEnabled(item) {
    var enabledary = item.enabled;
    var enabled;
    for (j=0; j < enabledary.length; j++) {
        if (enabledary[j].choose) {
            enabled = enabledary[j];
            break;
        }
    }
    return enabled;
}
```

更新の実行

```
function process() {
    // 画面に表示されているデータのUlp
    // jp.jasminesoft.wagby.model.model1_ulp.Model1Ulp
    var _listp = p.request.getAttribute("model1_ulp");

    var errormsg = "エラーが発生しました。有効・無効にて有効とするのは1か所のみとしてください。";

    // p.request.getAttribute("__jfc_jfcerrors")をチェックして「データベースコミット前」のスクリプトで
    // errormsgのエラーメッセージが設定されたかどうかをチェックする。
    var jfcerrors = p.request.getAttribute("__jfc_jfcerrors");
    var isSetErrorMessage = false;
    if (jfcerrors !== null) {
        var existerrmsg = jfcerrors.jfcerror;
        for (var i=0; i<existerrmsg.length; i++) {
            var error = existerrmsg[i];
            if (error.content === errormsg) {
                isSetErrorMessage = true;
                break;
            }
        }
    }

    if (isSetErrorMessage) { // エラーメッセージが設定されていた
        var items = _listp.item;
        var enableditems = new Array();
        for (i=0 ; i < items.length ; i++) {
            var item = items[i];
            //print("item="+item);
            if (item.jfcdelete) {
                continue;
            }
            var enabled = getChoosedEnabled(item);
            //print("item.enabled.id="+enabled.id)
            if (enabled.id === 1) {
                item.jfcerror = errormsg;// 同じエラーメッセージを各データにもセットする。
            }
        }
```

```
    }
}

function getChoosedEnabled(item) {
    var enabledary = item.enabled;
    var enabled;
    for (j=0; j < enabledary.length; j++) {
        if (enabledary[j].choose) {
            enabled = enabledary[j];
            break;
        }
    }
    return enabled;
}
```

[例] 一覧更新画面の入力チェック (2) 画面に表示されていないデータも含める

前節で説明したスクリプトは、画面に表示されているデータのみを対象にしていました。ここでは（画面に表示されていない分も含めた）データの入力チェックを行うスクリプトを紹介します。

図 一覧更新画面の入力チェック

入力チェックスクリプトのため、ここで値を書き換えることはできません。値の参照のみを行うことができます。

データベースコミット前

「コントローラ > 一覧更新 > データベースコミット前」を用意します。

```
function process() {
    var Jfcerror = Java.type("jp.jasminesoft.jfc.error.Jfcerror");
    var finderContext = p.appctx.getBean("UpdateListModel1FinderContext");
    var entityService = p.appctx.getBean("Model1EntityService");
    var results = entityService.find(finderContext);
    var count = 0;
    for (i=0; i<results.size(); i++) {
        var entity = results.get(i);
        //print("results="+entity);
        if (entity.enabled === 1) {
            count++;
        }
    }
    if (count > 1) {
        var error = new Jfcerror();
        error.content = "エラーが発生しました。有効・無効にて有効とするのは1か所のみとしてください。"
        p.errors.addJfcerror(error);
```

```
    }
}
```

- このスクリプトではデータベースを操作して全データをチェックします。画面上のデータに依存しない代わりに、画面上の各行へのエラーメッセージのセットは行なっていません。

プレゼンテーションモデル

WagbyではWebフォームから入力された値は「プレゼンテーションモデル」が管理しています。

プレゼンテーションモデルの操作では、項目名の末尾に ".content" を付与することで、入力文字列を取得することができます。

次の例は、モデル schedule の日付項目 sch_date の入力が "H25" で始まる場合はエラーにする、というスクリプトです。

```
/* 日付を取り出す。*/
var s =
    (schedule_p.schDate !== null) ? schedule_p.schDate.content : null;
/* slice は JavaScript が提供する */
if (s !== null && s.slice(0,3)=="H25") {
    var error = new Jfcerror();
    error.content = "平成25年はまだ入力できません。";
    p.errors.addJfcerror(error);
}
```

エラーメッセージ

画面に表示するメッセージは次の3種類を利用することができます。

INFO

処理が正常に終了した場合のメッセージです。青枠にメッセージが表示されます。

```
var info = new Jfcinfo();
info.content="...";
p.errors.addJfcinfo(info);
```

WARN

警告メッセージです。黄色枠にメッセージが表示されます。

```
var warn = new Jfcwarn();
warn.content="...";
p.errors.addJfcwarn(warn);
```

ERROR

エラーメッセージです。赤枠にメッセージが表示されます。

```
var error = new Jfcerror();
error.content="...";
p.errors.addJfcerror(error);
```

ルール

- 表示するメッセージの内容の先頭文字が a から z のいずれかであった場合、これを大文字にして画面に表示します。

国際化

メッセージの国際化にも対応しています。

エラーメッセージファイル $(DEVHOME)¥customize¥resources¥myerrormsg_ja.properties.UTF8 に「キー=値」形式のメッセージを記述します。詳細は"メッセージの変更・国際化 > リソースファイルの編集"をお読みください。

(326)

次のコードでロケールに対応した適切なメッセージが出力されます。（この例では、キーに "error.original" という名称を使っています。）

```
p.errors.addJfcerror(errorManager.getJfcerror("error.original", p.locale));
```

エラーメッセージファイルにプレースホルダを指定することもできます。具体的には次のようになります。

```
var arrayOfStrings = Java.type("java.lang.String[]");
var obj = new arrayOfStrings(1);
obj[0] = "テストです。";
p.errors.addJfcerror(errorManager.getJfcerror("error.original", obj, p.locale));
```

例外 BusinessLogicException を利用する

Jfcerror などのクラスを使わず、例外 BusinessLogicException にエラーメッセージを含めることもできます。ここで設定したエラーメッセージが画面に表示されます。詳細...

```
throw new Packages.jp.jasminesoft.jfc.core.exception.BusinessLogicException("エラーメッセージ");
```

警告

エラーではなく警告を使う場合は、JFCHelperUtils クラスを使います。次のようにしてください。

```
var JFCHelperUtils = Java.type("jp.jasminesoft.jfc.JFCHelperUtils");

var s = model1.item1;
if (s != null && s == '(エラーとしたい値)') {
    var warn = new Jfcwarn();
    warn.content="..."; /*警告メッセージ*/
    JFCHelperUtils.checkWarnedError(
        p, s, warn, "model1.item1",
        "model1.item1.error.input.generic");
}
```

JFCHelperUtils.checkWarnedErrorメソッドの第一引数は p となります。第二引数は「入力された値」です。第三引数は警告オブジェクト(warn)です。第四引数は「モデルID.項目ID」としてください。第五引数は国際化対応時、エラーメッセージのリソースのキー名となります。通常は上記例のように「モデルID.項目ID.error.input.generic」としてください。

入力欄の背景色の設定

プレゼンテーションモデルの入力項目に errorcode という属性が用意されています。ここに何らかの値を設定すると、当該入力欄の背景を赤色とします。

例

モデルID「AAA」の繰り返しコンテナ「AAAM」に含まれる項目「AAAM/item1」の errorcode をセットするコードは次のとおりです。

```
var AAAMClass = Java.type("jp.jasminesoft.wagby.model.AAA_p.AAAM");
var Item1Class = Java.type("jp.jasminesoft.wagby.model.AAA_p.Item1");
```

(326) リソースファイルは国際化に対応しています。英語版の場合は、myerrormsg_en.properties を用意してください。

```
var cont_ary = AAA_p.AAAM;
if (cont_ary !== null) {
  var values;
  for (var i=0; i<cont_ary.length; i++) {
    if (cont_ary[i].item1 === null) {
      values = new Item1Class();
      cont_ary[i].item1 = values;
    } else {
      values = cont_ary[i].item1;
    }
    values.errorcode = "ERROR";/* errorcode をセットする */
    var error = new Jfcerror();
    error.content = "ERROR";
    p.errors.addJfcerror(error);
  }
}
```

エラーメッセージにHTMLタグを含める

標準ではエラーメッセージに HTML タグを含めることはできません。これはセキュリティ対策のため、標準ではサニタイズ処理が有効となっているためです。

開発者は次のように sanitize フィールドをfalseとすることで、サニタイズ処理を無効にすることができます。

```
var error = new Jfcerror();
error.sanitize = false;
error.content = "<b>ERROR</b>";
p.errors.addJfcerror(error);
```

[例] 整合性チェック

Webフォームからの入力ではなく、自動計算によって求められた値を使った「整合性チェック」を行うこともできます。この場合はストアモデルを用います。ストアモデルは画面から入力された文字列を適切な型（文字、数字、日付）に変換した値を保持します。また自動計算によって求められた値も保持します。

次の例は、親モデル Parent と子モデル Child （外部キーで関連する親子関係）において、親側の Parent に、子である Child モデルの数(COUNT)および、項目の最大値(MAX)が定義されているものとします。

(327)

このとき、countOfChild 項目は2以上、maxOfChildItem1 項目が100以上というルールは次のように記述します。

```
/* Parent はストアモデル */
if (Parent.countOfChild < 2) {
    var error = new Jfcerror();
    error.content = "子モデルの数は2以上です。";
    p.errors.addJfcerror(error);
}
if (Parent.maxOfChildItem1 < 100) {
    var error = new Jfcerror();
    error.content = "子モデルitem1の最大値は100以上です。";
    p.errors.addJfcerror(error);
}
```

(328)

(327) 具体的には Parent モデルの countOfChild 項目には計算式「COUNT(${child_lp})」が、maxOfChildItem1 項目には計算式「MAX(${child_lp.item1})」が、それぞれ定義されているものとします。

(328) 上記コードでもし Parent_p と表記すると、プレゼンテーションモデルを用います。プレゼンテーションモデルは画面から入力された値（文字列）を保持しますが、countOfChild 項目は計算式なので値が入っていません。よって、ここでプレゼンテーションモデルを用いるのは不適切、となります。

[例] 子モデルの存在チェック

親子モデル関係において、子モデルの同時更新を有効にします。

このとき「親モデルに、必ず1件以上の子モデルが必要」という入力チェックは次のように記述できます。子モデルを Child と定義しています。

```
/* 子モデル名を Child と定義している */
var Child_ary = p.request.getAttribute("Child_ary");/*子モデル配列の取得*/
var count = Child_ary.length;
if (count < 1) {
    var error = new Jfcerror();
    error.content = "子モデルChildは1件以上、必要です。";
    p.errors.addJfcerror(error);
}
```

親子モデル同時更新機能では、入力チェック処理のタイミングで、対象となる子モデルの配列が、p.requestオブジェクトに格納されています。キー名は「子モデル名_ary」です。末尾に "_ary" をつけたキー名で、配列を取得できます。

これを取得し、配列の数をチェックすることで、子モデルの数を知ることができます。

なお、ここで取得した Child_ary はストアモデルです。

(329)

[例] 親モデルと子モデルの値を比較する

親子モデル関係において、子モデルの同時更新を有効にします。

このとき「親モデルの出荷日と、各子モデルの納期日を比較する」というスクリプトを次のように記述することができます。

```
/*親モデルの出荷日と、子モデルの納期を比較する*/
var syukkabi = Parent.sdate;/*Parentはストアモデル*/
var Child_ary = p.request.getAttribute("Child_ary");
var count = Child_ary.length;
for (var i=0; i<count; i++) {
    var Child = Child_ary[i];/*Childもストアモデル*/
    var nouki = Child.nouki;/*syukkabiもnoukiもJavaScriptのDate型オブジェクトとして扱える*/
    if (syukkabi.getTime() > nouki.getTime()) {
        var error = new Jfcerror();
        error.content = (i+1)+"番目の納期は、出荷日よりも早い日になっています。";
        p.errors.addJfcerror(error);
    }
}
```

Parent ならびに Child_ary はストアモデルです。ストアモデルは文字列、数字、日付型を区別しているため、項目 sdate や nouki は日付型と認識されます。そのため JavaScript の日付オブジェクトとして操作できます。具体的には getTime() メソッドを利用した比較処理を行うことができます。

(330)

[例] SQLを用いた件数チェック

customer モデルが最大、10件までしか登録できない、というルールを想定します。このチェックのためには、現在データベースに格納されている「件数」を知る必要があります。

---

(329) この例では Child の先頭は大文字となっています。モデル名が child と先頭が小文字の場合は、p.request.getAttribute("child_ary") となります。

(330) この例では Child の先頭は大文字となっています。モデル名が child と先頭が小文字の場合は、p.request.getAttribute("child_ary") となります。

入力チェックのスクリプトでデータベースを操作するためには、開発者の方で「セッション」を取得します。

(331)

スクリプトの例を示します。

```
var HibernateUtil = Java.type("jp.jasminesoft.jfc.app.HibernateUtil");
var session = HibernateUtil.openSession();
try {
    var o = session.createSQLQuery(
           "SELECT COUNT(*) FROM ¥"customer¥"").uniqueResult();
    if (o > 10) {
        var error = new Jfcerror();
        error.content = "10件以上のデータを登録することはできません。";
        p.errors.addJfcerror(error);
    }
} catch (e) {
    e.printStackTrace();
} finally {
    if (session !== null) {
        session.close();/*忘れないこと*/
    }
}
```

- データベース操作のためのセッションオブジェクトは HibernateUtil.openSession() を使って取得します。
- 取得したセッションは必ずクローズしてください。クローズを忘れると、利用できるデータベースセッションが不足し、運用途中で実行時エラーになります。
- 上のスクリプトではテーブル　customer　の前後をダブルクォートで囲っています。お使いのデータベースによって、シングルクォートまたはダブルクォートのどちらかを使います。（内蔵DBであるHSQLDB利用時は、二重引用符で囲みます。）このため、開発機と本番機で異なるデータベースを用いる場合は、スクリプトファイルの変更も行う必要があるかも知れないということにご留意ください。
- SQLは検索系でのみ利用してください。(*1)
- 上記のサンプルコードは　uniqueResult　メソッドを使っているため、戻り値は常に1件である必要があります。複数件のレコードを返す　SQL　を記述した場合は　list　メソッドを使います。(詳細はHibernate プログラミングを学んでください。) 次節で、EntityService を使って複数件を扱う例を説明します。

(332)

入力フィルタ（スクリプト）

定義方法

入力フィルタ処理をスクリプトコードとして記述することができます。

---

(331) 入力チェックのスクリプトは、開発者が自らでセッションを取得する必要があります。

(332) 1. ここでは入力チェックを扱っているため、更新処理は生じないという前提で差し支えありません。Wagbyは更新時に独自のロックマネージャを使います。それを無視して直接、SQLで更新するとデータの整合性を保証できなくなります。更新を伴う場合は SQL ではなく Wagby が提供する EntityService を利用してください。

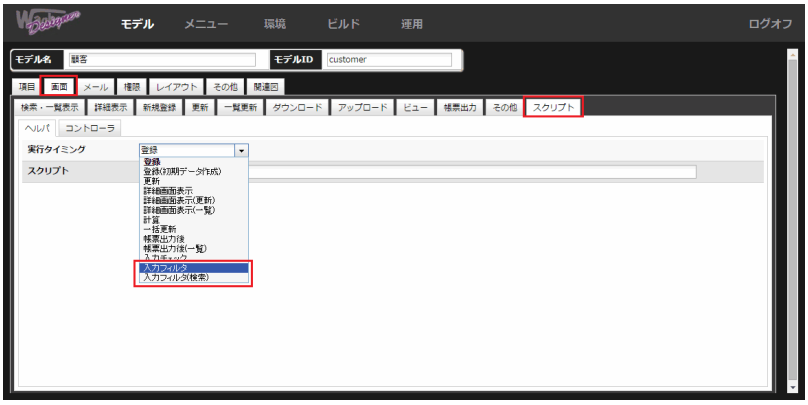


図 スクリプト記述欄

ここに書いた処理は、入力値からWagby内部のプレゼンテーションモデルを作成する前に実行されます。

Designerでスクリプトを修正すると、開発機上のアプリケーションへ即座に反映されます。ビルド処理なしで修正を確認することができます。

[例] 入力値を補完する

WagbyではWebフォームから入力された値は「プレゼンテーションモデル」が管理しています。

スクリプト内では${モデル名_p}という表記で扱います。すべての値は文字列型になっています。

プレゼンテーションモデルの操作では、項目名の末尾に ".content" を付与することで、入力文字列を取得することができます。

次の例は、モデルID customer の電子メール項目 email の入力値に "@" が含まれていない場合、@以下を補完するというスクリプトです。操作はすべてプレゼンテーションモデルである customer_p に対して行います。

```
var email = (customer_p.email != null) ? customer_p.email.content : null;
if (email != null && email != "" && email.indexOf("@") < 0) {
    email = email + "@jasminesoft.co.jp";
    customer_p.email.content = email;
}
```

[例] 文字列拡張検索の標準を OR 検索とする

Wagbyの文字列拡張検索は標準では AND 検索です。これを「空白区切りで検索条件が入力されたら、OR 検索を標準とする」ように変更するスクリプトを紹介します。モデルIDはcustomerとし、対象項目をnameとしています。

このスクリプトは「入力フィルタ(検索)」に記述します。

```
var cname = (customer_cp.name != null) ? customer_cp.name.content : null;
if (cname != null && cname != "" && cname.indexOf(" ") >= 0 && cname.indexOf("OR ") != 0) {
    cname = "OR " + cname;
    customer_cp.name.content = cname;
}
```

検索条件項目 name に空白を含む値が入力されていて、かつ、先頭が"OR "で開始されていない場合、これを付与するようにしています。

Webフォームから入力された検索値は「コンディションのプレゼンテーションモデル」が管理しています。 スクリプト内ではモデル名_cpという表記で扱います。すべての値は文字列型になっています。

一括更新（スクリプト）

定義方法

「一括更新」ボタン押下時の処理をスクリプトコードとして記述することができます。

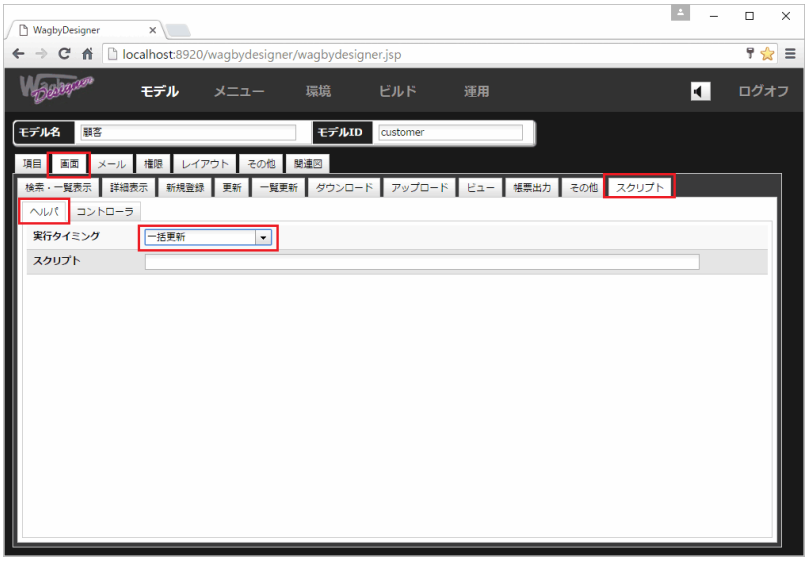


図 スクリプト記述欄

Designerでスクリプトを修正すると、開発機上のアプリケーションへ即座に反映されます。ビルド処理なしで修正を確認することができます。

スクリプト内では、モデル名に対応した「オブジェクト」を利用できます。

[例] 選択されたモデルのフラグをセットする

一覧表示画面でチェックされたすべての customer モデルの radio_flag 項目に "1" をセットします。

```
customer.radioFlag = 1;/*選択された複数のcustomerに適用される。*/
```

[例] チェックボックスの値をトグルスイッチとして扱う

チェックボックス型である keyperson 項目をトグルスイッチとして扱うものです。チェックされていた値は解除され、解除されていた値はチェックされます。チェックボックスであるため、配列として扱います。

```
var flag = customer.keyperson;/*チェックボックスによるON/OFF項目*/
if (flag == null || flag[0] == 0) {
    customer.keyperson = 1; /* OFF から ON へ */
} else if (flag != null && flag[0] == 1) {
    customer.keyperson = 0; /* ON から OFF へ */
}
```

[例] 複数のボタンを扱う

複数の一括更新ボタンを用意した場合、スクリプト中では p.action の値で判定してください。 具体的なコード例を示します。

```
if (p.action == "UpdateAll") {
    /* ボタン1 の処理 */
} else if (p.action == "UpdateAll2") {
```

```
    /* ボタン2 の処理 */
} else ...
```

最初のボタンは UpdateAll で判別します。二個目以降のボタンは UpdateAll2, UpdateAll3,... となります。

更新後に再び計算処理を行う

一括更新スクリプトは、これが呼び出される前に、そのモデルに対する自動計算処理および参照連動の解決処理は行われています。ただし、更新した値を使った再計算や、参照連動は行なわれません。

一括更新スクリプトで修正された値を使って再び計算処理を呼び出す場合は、次のように記述します。

```
return "calc";
```

これは一括更新スクリプトだけのルールです。ここで参照連動項目が含まれる場合、参照連動の解決後に計算が行われます。

計算結果を用いてさらに参照連動処理を行う場合は、次のように記述します。

```
return "calc,resolv";
```

コンマ区切りで calc,resolv と記述すると、自動計算後、参照連動の解決処理を行ってからデータベースに値を保存します。

(333)

一括更新後の画面遷移

一括更新処理後に遷移する画面を指定できます。例えばメインメニューに遷移させるためには「コントローラ 〉 一覧 〉 画面遷移」のタイミングで次のようなスクリプトを記述します。

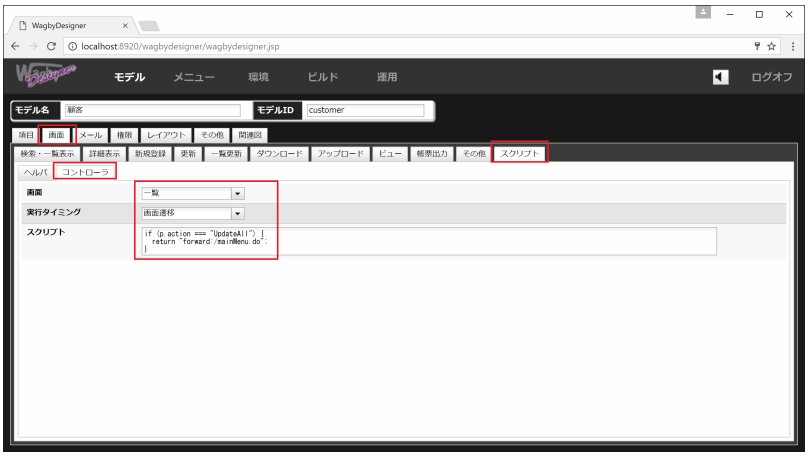


図 画面遷移を制御する

```
if (p.action == "UpdateAll") {
  return "forward:mainMenu.do";
}
```

生成されるスクリプトは ShowList<モデルID>_GetForwardUrl.js になります。

どのボタンが押されたときかを、p.action の値で判別させています。この条件は必ず付与してください。

(333) 自動計算の結果を使って参照連動の解決を行ったあと、さらにその値を使った自動計算を行い、その結果を用いて再び参照連動を行う...という複雑なモデル構成の場合は、次のように記述することもできます。return "calc,resolv,calc,resolv";

［例］更新された数を画面遷移後に表示させる

通常、一括更新処理は終了後に「xx 件の yy データの更新を行いました。」というメッセージが表示されます。これをカスタマイズし、条件によって実際に更新されたデータ件数のみを表示するようにします。ここでは、画面遷移のスクリプトを応用します。

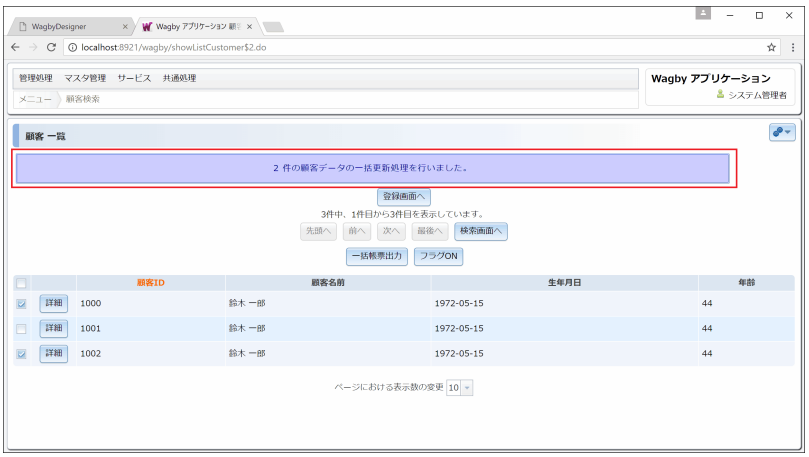


図 標準のメッセージ

カウント処理

「ヘルパ ＞ 一括更新」スクリプトに、処理数をカウントする変数を用意します。

```
...(更新処理)...

var counter = p.request.getAttribute("myapp.counter");
if (counter == null) {
    counter = 1;
    p.request.setAttribute("myapp.counter", counter);
} else {
    counter = counter + 1;
}
```

p.request は、この一括更新処理中に維持したい変数を保持することができる入れ物として利用できる特別なオブジェクトです。この例では名前を myapp.counter としています。任意の名前を利用できますが、Wagbyが利用する他の名前と重複しない、かつ管理しやすい名前を使うようにしてください。（上の例のように、アプリケーションを意味する接頭語を付与するなどの工夫を行なってください。）

上のコードでは、初回は 1 をセットします。2回目以降は counter 変数を増加させています。開発者は、例えば条件によって値を更新しなかった場合は counter を増加しない、といったルールをスクリプトで実現することができます。

メッセージの表示

「コントローラ ＞ 一覧 ＞ 画面遷移」で、メッセージの表示処理を行います。

```
if (p.action == "UpdateAll") {
  var counter = p.request.getAttribute("myapp.counter");
  if (counter != null) {
    var info = new Jfcinfo();
    info.content="更新は"+counter+"件です。";
    p.errors.clearJfcinfo();/* 標準のメッセージを削除する */
    p.errors.addJfcinfo(info);
  }
}
```

p.request は、この一括更新処理が終わるまで myapp.counter の値を保持しています。（一括更新処理が終了すると、この値は自動的に p.request から削除されます。）

Jfcinfo は成功時に表示されるメッセージです。詳細は"エラーメッセージ > INFO"をお読みください。

p.errors.clearJfcinfo()　を呼び出すことで、標準のメッセージを削除することができます。これによってメッセージを強制的に置き換えて表示させています。

仕様・制約

ワークフローとの併用

ワークフローを定義したモデルでは一括更新ボタンが表示されません。

ワークフロー利用時は、フローを保留させている人(処理保留者)のみがデータを更新できるという仕様となっています。 一括更新ボタンの表示には、一覧に表示されているデータのうち保留データが含まれているかの判断が必要となりますが、現在はこの制御を行っておらず、非表示となっています。

代替案として、ワークフローの設定を無効化したサブモデルを作成し、サブモデル側で一括更新処理を行うという方法があります。 ただし、この方法は「処理保留者のみがデータを更新できる」というルールを無視した形でデータの更新処理を行うことにご注意ください。そのため、承認/決裁の判断に利用される重要項目の更新は控えることや、決裁済みのデータは更新しないといった配慮を行うようにしてください。

一覧帳票出力後の更新（スクリプト）

定義方法

「帳票出力後」または「帳票出力後(一覧)」ボタン押下時の処理をスクリプトコードとして記述することができます。

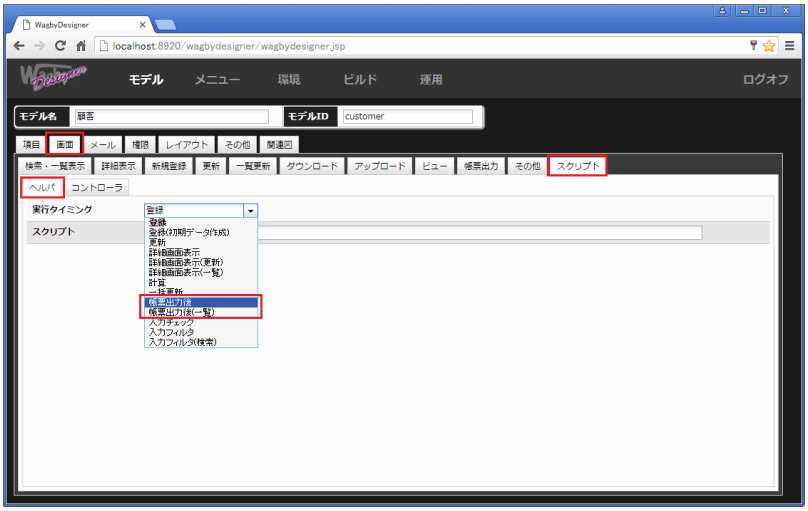


図 スクリプト記述欄

[例] 出力後に「帳票出力日」という項目に値をセットする

帳票出力の対象となったモデルの「帳票出力日」項目に値をセットするスクリプトは次のとおりです。

```
var entityService = p.appctx.getBean("CustomerEntityService");
try {
    var n_customer = entityService.findById(customer.customerid, true);
    n_customer.printdatetime = ExcelFunction.NOW();
    entityService.update(n_customer);
    print(n_customer);
} catch (e) {
    print(e);
}
```

このスクリプトは詳細画面・一覧画面のいずれにも適用できます。一覧画面の場合、帳票出力の対象データすべてに適用されます。

- 更新処理を行うため、CustomerEntityService オブジェクトを利用します。サービスオブジェクトの詳細は"Service/Daoクラス、SQLを利用する"をお読みください。
- サービスオブジェクトの findById メソッドを用いて、データベースから最新の状態を取得します。このとき、第二引数に true を設定することで、ロックをかけます。
- 日付時刻型項目（ここでは customer モデルの printdatetime 項目）に、現在の日付時刻を設定します。ExcelFunction..NOW() は Wagby が提供する関数です。現在の日付時刻を返します。
- 更新に失敗したときのことを想定して、全体を try-catch で囲っています。例えばロックに失敗したときなどは、catch 節が処理されます。

注意点

値の変更後、詳細画面（または一覧画面）に表示されるデータには、printdatetime 項目は更新前の値が表示されています。更新後ではありません。これは帳票出力が成功したタイミングで値が更新されますが、画面表示用のデータはその前に用意されているためです。

画面を再描画すると、更新された値を確認することができます。

[例] 出力した数をカウントアップする

整数型の項目 printcount を用意します。次のスクリプトは、帳票出力の都度、値を 1 つ増やすものです。

```
var entityService = p.appctx.getBean("CustomerEntityService");
try {
    var n_customer = entityService.findById(customer.customerid, true);
    n_customer.printcount = n_customer.printcount + 1;
    entityService.update(n_customer);
} catch (e) {
    print(e);
}
```

このスクリプトは詳細画面・一覧画面のいずれにも適用できます。一覧画面の場合、帳票出力の対象データすべてに適用されます。

- 更新処理を行うため、CustomerEntityService オブジェクトを利用します。サービスオブジェクトの詳細は"Service/Daoクラス、SQLを利用する"をお読みください。
- サービスオブジェクトの findById メソッドを用いて、データベースから最新の状態を取得します。このとき、第二引数に true を設定することで、ロックをかけます。
- 整数型項目（ここでは customer モデルの printcount 項目）の、現在の値に一つ加えた値を設定します。項目 printcount の初期値を 0 としておいてください。（この初期値設定を行わない場合、標準の初期値は '-1' となっています。従って最初の実行結果は 1 ではなく 0 となります。）
- 更新に失敗したときのことを想定して、全体を try-catch で囲っています。例えばロックに失敗したときなどは、catch 節が処理されます。

注意点

値の変更後、詳細画面（または一覧画面）に表示されるデータには、printdatetime 項目は更新前の値が表示されています。更新後ではありません。これは帳票出力が成功したタイミングで値が更新されますが、画面表示用のデータはその前に用意されているためです。

画面を再描画すると、更新された値を確認することができます。

[例] 複数の帳票ボタンによって処理を変える

複数の帳票パターンを用意している場合、どのボタンを押されたかによって処理を分岐させることができます。

ボタン押下によって、p.action に値がセットされています。この値を使って次のような条件分岐を行ってください。

```
if (p.action == "PrintCustomer_showListCustomer") {
...
}
```

開発のポイント

最初はスクリプトに

```
print(p.action);
```

などと記載し、どのボタンが押されたら、どのような値が p.action に格納されるかを調べておくとよいでしょう。

仕様

このスクリプトは一覧帳票出力において、チェックボックスで選択した複数のデータに対しても適用されます。

モデルをまたがる計算（トランザクション）（スクリプト）

在庫管理のイメージ

出庫モデル syukko の登録処理時に、同時に（関連する）在庫モデル zaiko の値を変更する例を説明します。

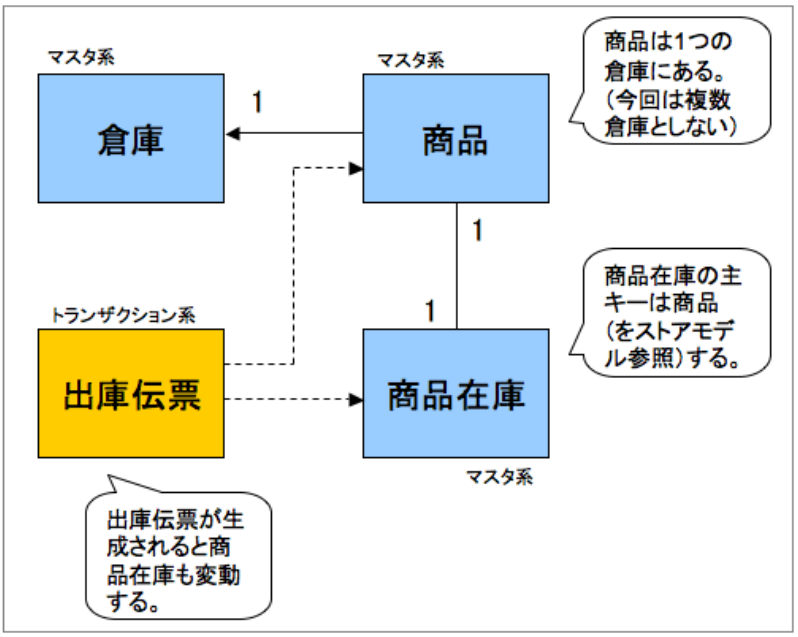


図 在庫管理システムのイメージ

「商品」と「商品在庫」は 1:1 の関係があります。このため、「商品在庫」の主キーは、「商品」を参照するようにしています。

ここではトランザクション系である「出庫伝票」に注目します。このモデルの新規登録処理で、商品在庫を同時に変動させる必要があります。また、商品在庫を超える出庫要求については、エラーを返すようにします。

例

商品在庫マスタモデルに、図に示す初期データを設定しました。それぞれ在庫数を100としています。

図 商品在庫マスタモデルの初期値

出庫伝票を作成します。指定商品を一つ、出庫します。

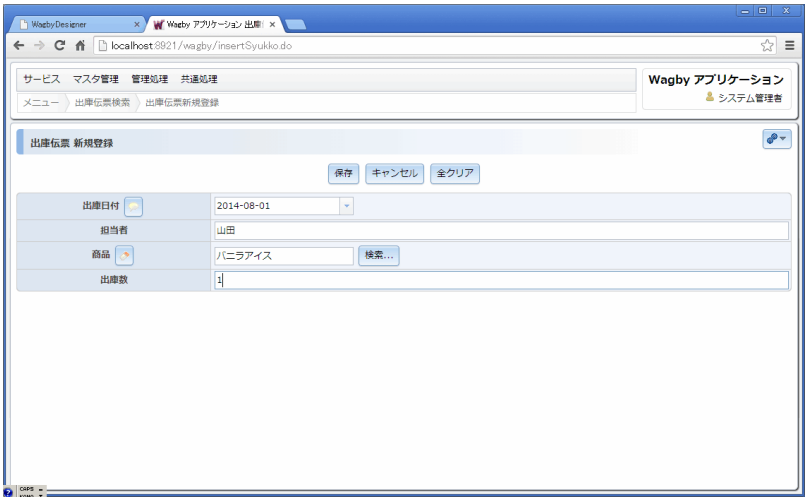


図 出庫伝票の新規作成

出庫伝票が作成されました。このタイミングでトランザクション処理が正常に終了しています。

図 出庫伝票作成時にトランザクション処理も動作している

商品在庫マスタを確認します。在庫が減じられていることがわかります。

図 在庫が減じられている

エラー時の扱い

在庫数を超える数を出庫しようとするとエラーメッセージを返すようにしました。 この場合はデータベースの値は書き変わりません。

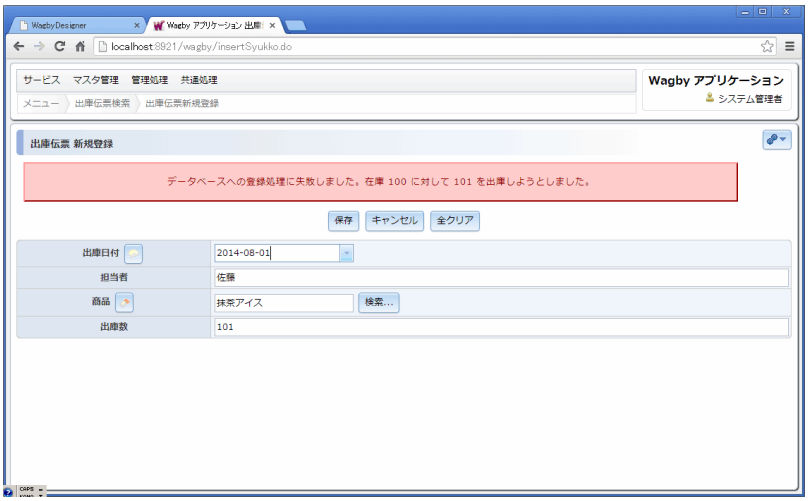


図 トランザクション失敗

定義方法

図に示した4つのモデルを定義します。

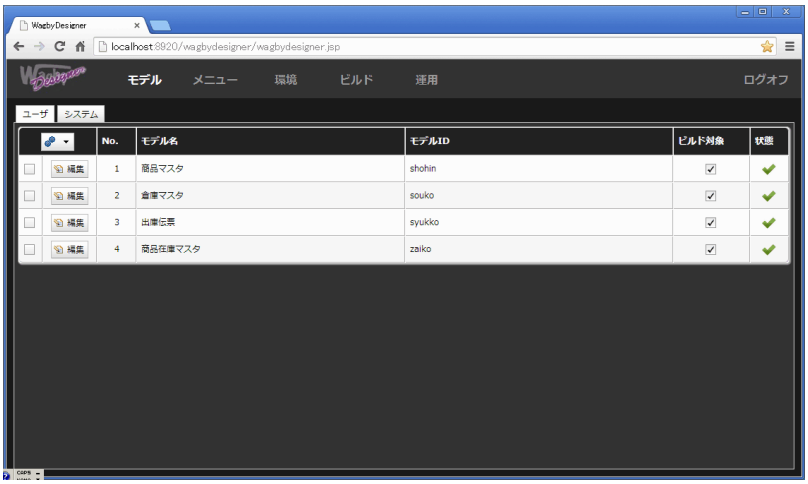


図 定義した4つのモデル

倉庫マスタ

倉庫IDと倉庫名、倉庫の管理者を定義しました。

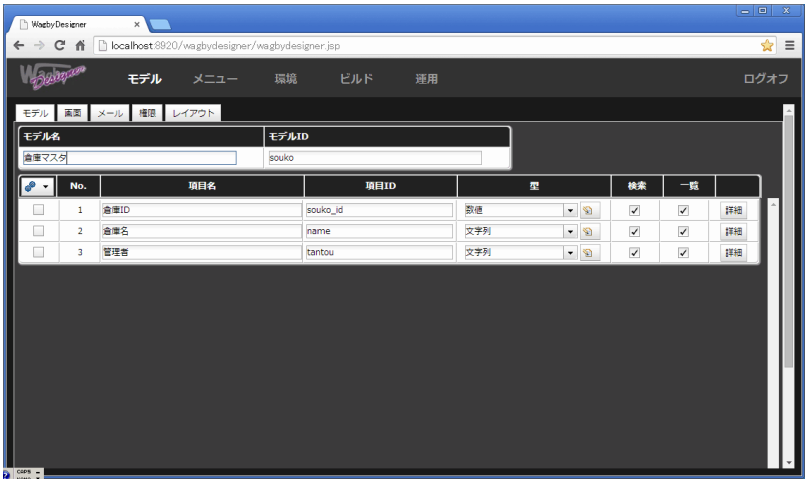


図 倉庫マスタ

商品マスタ

商品ID、商品名、単価を保持します。また、一つの倉庫に紐付けられます。

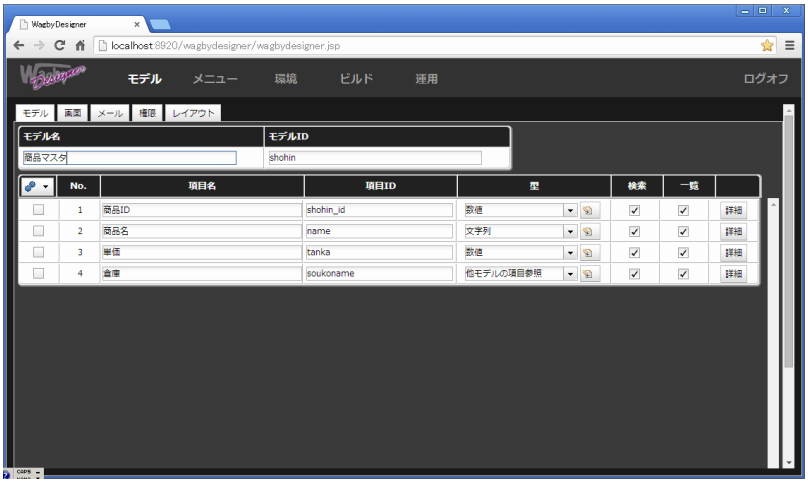


図 商品マスタ

商品在庫マスタ

主キーである「商品」は、商品マスタと 1:1 の関連を持ちます。

主キー「商品」項目は、順序を使わない設定としてください。

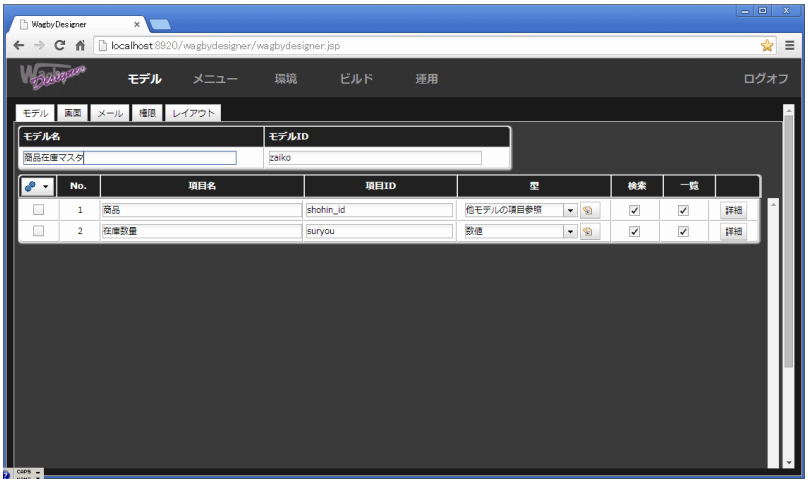


図 商品在庫マスタ

出庫伝票

どの商品をいくつ出庫したかを記録します。

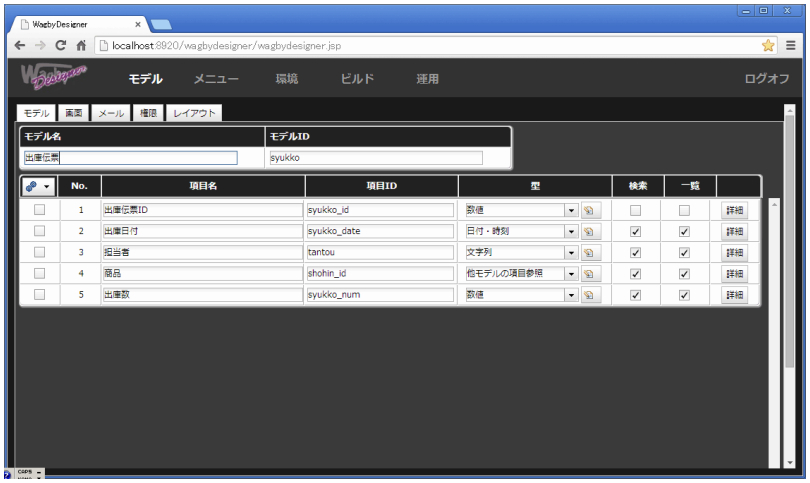


図 出庫伝票

ここで商品の出庫数に合わせて、商品在庫マスタの在庫数量を減らすために、モデル参照項目「商品」にトランザクションの設定を行います。 ここでは、新規登録時に「在庫」モデルと連動する指定を行います。

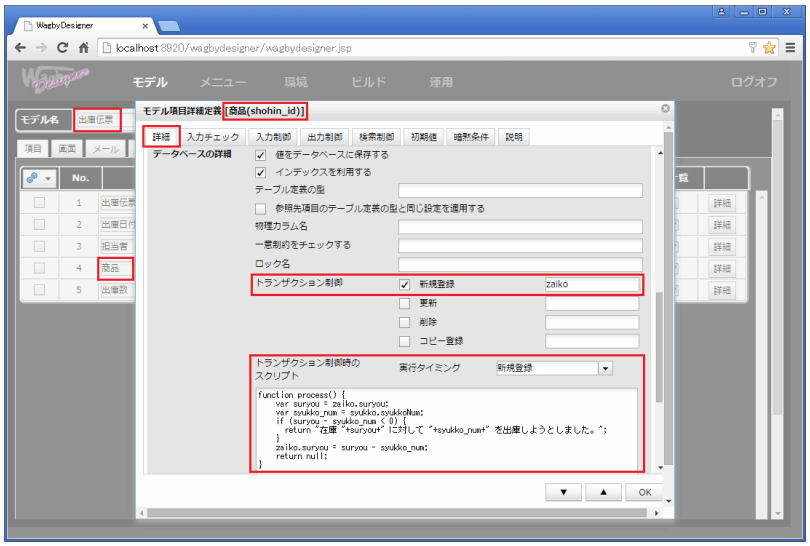


図 トランザクション設定

モデル名を明記しているのは、今回のケースにおいては（参照先のモデルが）「商品」ではなく「商品在庫」となっているためです。「商品」と「商品在庫」は 1:1 の関係にあり、「商品在庫」の主キーは、「商品」の主キーと同じという関係としています。 参照先モデルがそのままトランザクション処理の対象モデルに等しい場合は、モデル名を空白とすることもできます。本欄が空白の場合は、参照モデル名が利用されます。

新しくトランザクションを設定したとき（新規登録・更新・削除・コピー登録のチェックを行ったとき）は、ビルド処理を行ってください。 一度設定したあとは、ビルド処理は不要です。トランザクションスクリプトの変更は即座に反映されます。

図 ビルド処理

文法

実際のトランザクション処理コード（トランザクションスクリプト）は、図における「トランザクション制御時のスクリプト」欄に記述します。このコードはトランザクション境界の中で実行されます。[トランザクション境界について...]

```
var suryou = zaiko.suryou;
var syukko_num = syukko.syukkoNum;
if (suryou - syukko_num < 0) {
  return "在庫 "+suryou+" に対して "+syukko_num+" を出庫しようとしました。";
}
zaiko.suryou = suryou - syukko_num;
return null;
```

- スクリプトでは、return 命令で一つの戻り値を返すようにします。戻り値が null の場合は正常と扱われます。エラーの場合はエラーメッセージ（文字列）を返すようにします。
- 「新規登録」「更新」「削除」のいずれかを指定します。それぞれのタイミングでトランザクションスクリプトを記述できます。
- 登録処理とコピー登録処理の区別はありません。SCREENTYPE 定数を用いて判別することができます。これは画面種別を返すSCREENTYPE関数の値を保持しています。 新規登録時とコピー登録時は同じスクリプトファイルが呼ばれますが、コピー登録時は処理しないような制御を行う例を示します。

```
if (SCREENTYPE == "copy") {
    /* コピー登録時は処理を行わない */
    return null;
}
/* 登録時にのみ実行される */
...
}
```

繰り返しコンテナとの連携

例

繰り返しコンテナ内の項目にトランザクションを指定することができます。 図は、出庫伝票の明細で、複数の商品を同時に指定する例です。

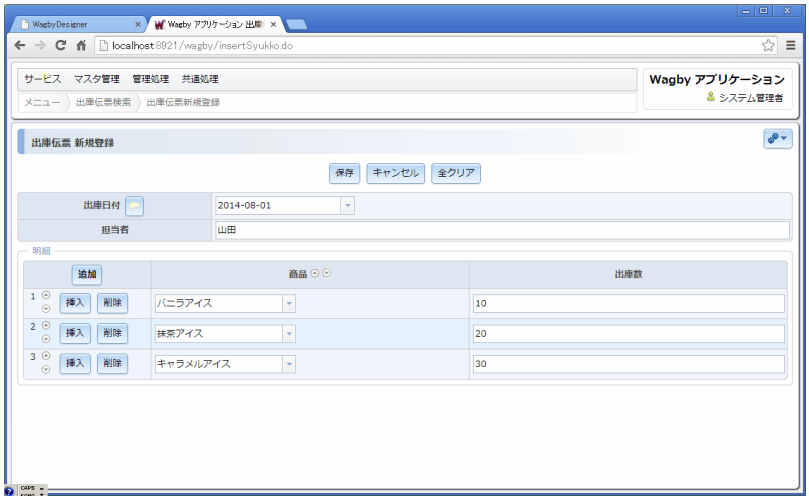


図 出庫伝票に明細を（繰り返しコンテナとして）用意する

同時に複数のデータが書き換えられていることがわかります。

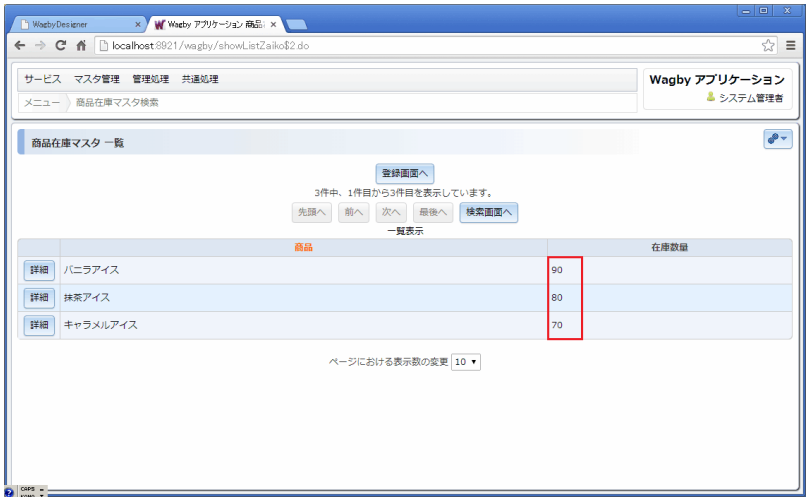


図 トランザクションが成功している

定義方法

図の出庫伝票の定義を変更し、明細を用意しました。

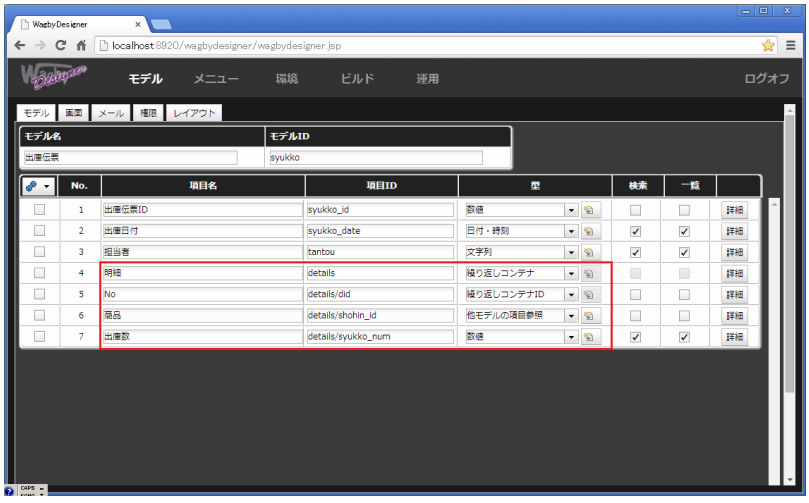


図 明細を用意した出庫伝票の定義

トランザクションスクリプトの変更

ファイル名やファイルの保存位置は変わりません。 スクリプト内では、繰り返しコンテナ "details" を利用できます。このとき、複数の繰り返しコンテナのデータ毎に、このスクリプトが実行されます。

```
/* トランザクションで使うオブジェクト sykko, zaiko は利用可能になっている。
   これは定義ファイル「トランザクション指定」に含めたモデルである。
   繰り返しコンテナも利用できる。details という名前のオブジェクトが用意されている。*/
var suryou = zaiko.suryou;

/* detailsは明細データ毎に呼び出された値がセットされる */
var syukko_num = details.syukkoNum;
if (suryou - syukko_num < 0) {
    return "在庫 "+suryou+" に対して "+syukko_num+" を出庫しようとしました。";
}
zaiko.suryou = suryou - syukko_num;
return null;
```

外部キー子モデルとの連携

Wagbyの外部キー関係では、一つの親データが複数の子データを管理できます。 次の例では親モデル名 parent と、子モデル名 child を用いて説明します。

子モデル child は、親モデルを指し示す外部キー項目 pkey を持つとします。このとき、この pkey 項目に対してトランザクションスクリプトを記載できます。

図 親子モデル関係でのトランザクションスクリプトサンプル

上図の例は、子モデル新規登録のタイミングで、子モデルの値を親モデルの項目へ転記するものです。

```
parent.pname = child.cname;
```

対象データがロックされている場合の動作

Wagby の標準では、モデルの更新制御は悲観ロックを用います。これを楽観ロック方式に切り替えることもできます。トランザクション実行時のロックは、指定した方式に準拠して制御されます。

悲観ロックでは、対象となるデータが更新中（更新画面が開いた状態）の場合、登録・更新処理を行おうとすると画面には "失敗した" というエラーメッセージが表示されます。

楽観ロックでは、対象となるデータが別に更新されていた場合、同様にエラーとなります。

いずれの場合も、そのトランザクションは失敗します。これは正しい動作です。

Service/DaoクラスやSQLを利用する

ここまでの説明は Wagby が標準で提供するモデル参照という仕組みを用いて自身に紐づくモデルが定まっている場合に適用できるものです。そのためトランザクション処理を記述する箇所は限定されています。

一方で、トランザクションスクリプトで（自身に紐づくモデル以外の）任意のモデルを操作する場合は Dao クラスを使うことができます。さらに（トランザクションスクリプトではない）他のスクリプトでも Service クラスを使ってトランザクション処理を記述することもできます。これらの Dao や Service を使うと SQL を利用することもできます。

トランザクション境界

Service/Daoクラス、ヘルパ、SQLを利用する > トランザクション境界 をお読みください。

仕様・制約

- トランザクションを指定できるのは、モデル参照項目または外部キー項目に限定されます。数値・文字列・日付型項目に指定しても無視されます。（動作しません。）
- この方法とは別に Service の内部でさらにスクリプトを用意する方法もあります。一つのトランザクション内で、任意のモデルを操作することができます。[詳細...]

生成されるスクリプトファイル

トランザクションを設定した場合、生成されるファイル名は次の通りです。

モデルIDEntityService_doTransactionBy${トランザクションを設定した項目ID}InsertTransaction.js モデルIDEntityService_doTransactionBy${トランザクションを設定した項目ID}UpdateTransaction.js モデルIDEntityService_doTransactionBy${トランザクションを設定した項目ID}DeleteTransaction.js

Service/Daoクラス、ヘルパ、SQLを利用する（スクリプト）

Serviceを利用する

Wagbyではモデルに対応したServiceクラスとDaoクラスを自動生成します。これは CRUD 処理 (*) に関するメソッドを提供します。

なお、Serviceは内部でDaoを利用しています。開発者はいずれを使うこともできますが、おおむね次の指針で使い分けを行なってください。

- トランザクションスクリプトで利用する場合は Dao を使う。
- それ以外のスクリプトで利用する場合は Service を使う。

(334)

(334) データベースへの登録(Create)、読み込み(Read)、更新(Update)、削除(Delete)といった基本操作の総称です。

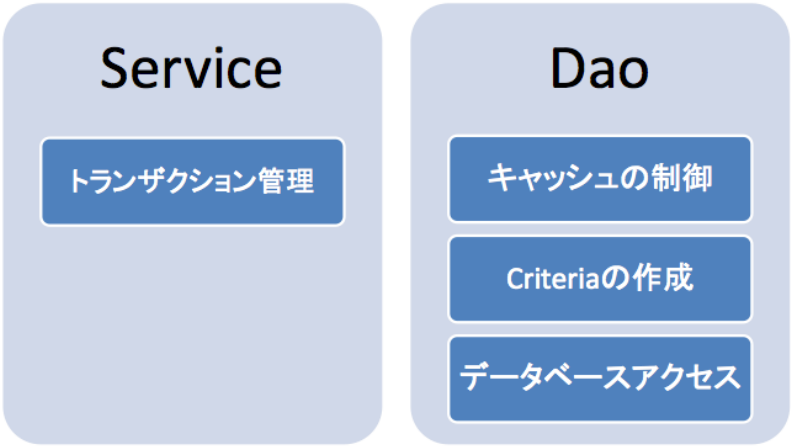


図 ServiceクラスとDaoクラス

ここでは Service の使い方を説明します。開発者はスクリプト内で、次のようにしてServiceオブジェクトを取得することができます。(p.appctxはSpringが提供するDIコンテナです。)

```
var entityService = p.appctx.getBean(Service名);
```

Service名は "<モデルID>EntityService" となります。モデルIDはキャメル記法で表現します。例えばcustomer モデルに対するService名は "CustomerEntityService" になります。

データの取得と更新

具体的なコード例を紹介します。次のコードはデータを1件取得します。

```
var entityService = p.appctx.getBean("JuserEntityService");
var user = entityService.findById("user01"); /* 1件のデータを取得 */
print(user); /* デバッグ用ログ出力 */
```

続いて、取得したデータを更新する例を紹介します。

```
var entityService = p.appctx.getBean("JuserEntityService");
var user = entityService.findById("user01", true); /* 1件のデータを取得 */
print(user); /* デバッグ用ログ出力 */
user.name = "ジャスミン太郎";
entityService.update(user); /* 更新 */
```

- findByIdメソッドの第二引数はロック処理を行うことを示しています。このコードは取得したデータを更新することを前提としているため、findById メソッド実行のタイミングでロックを取得しています。
- Serviceクラスはトランザクション境界となっているため、updateメソッド呼び出しのタイミングでデータベースにコミットされます。update メソッドの実行によって取得したロックは解放されます。

> ロックについて：上記コードでは、findById で取得したロックは、その後の update メソッドの実行によって必ず解除されるため問題ありません。しかしロックを取得したが、update メソッドは実行しない場合、このスクリプトが終了してもロックは保持されたままとなります。そのため更新予定のないデータを findById メソッドで取得する場合、通常は第二引数を省略してください。この詳細と解決方法は後述する"ロックの取得と解除"をお読みください。

新規登録

Customer モデルの新規登録を行うスクリプトを紹介します。new 演算子で生成したストアモデルのオブジェクトをデータベースに登録することができます。

```
var customerClass = Java.type("jp.jasminesoft.wagby.model.customer.Customer");
// 空の状態の Customer オブジェクトを作成
var customer = new customerClass();
// 上記2行のコードを1行で記述することもできる。
```

```
//var customer = new Packages.jp.jasminesoft.wagby.model.customer.Customer();

var helper = p.appctx.getBean("CustomerHelper");
// helper#initialize() を利用することでリポジトリで定義した
// 初期値をセットする。
// 参考: https://wagby.com/wdn8/operation-script-init.html#initalize
helper.initialize(customer, p);

customer.customername = "ジャスミン太郎";

// EntityService を取得
var service = p.appctx.getBean("CustomerEntityService");
// customer データを登録
service.insert(customer);
```

例外 PessimisticLockException

サービス (EntityService) のメソッド呼び出し時に例外 PessimisticLockException が生じることがあります。例えば findById メソッドでデータを取得しようとしてこの例外が生じた場合、このデータは別の利用者がロックを取得していることを意味します。

(335)

開発者は次の点に注意してスクリプトを作成するとよいでしょう。

無用なロックの取得を行わない。

findById メソッド呼び出し時に第二引数にtrueを指定するとロックを取得しようとします。しかし単に値を参照するだけの目的であればロックは不要です。（ロック取得は、その後にデータの更新を行うために行います。）その場合は第二引数にfalseを指定するとよいでしょう。

try-catch 節で処理を囲い、例外を補足する。

サンプルコードを示します。ロック取得失敗時のコードを catch 節に追加することができます。ここではコンソールに出力するのみとしています。

```
var PessimisticLockException = Java.type("javax.persistence.PessimisticLockException");
...
try {
  var entityService = p.appctx.getBean("JuserEntityService");
  var user = entityService.findById("user01"); /* 1件のデータを取得 */
  print(user);
} catch (e if e instanceof PessimisticLockException) { /* ロック取得失敗 */
  print("lock error " + e);
  throw e;
} catch (e) { /* その他の例外 */
  print("error " + e);
  throw e;
} finally {
}
```

トランザクションスクリプトでSQLを併用する

トランザクションスクリプト内で SQL を直接、実行することもできます。

ここでは上の題材を使って、Dao ではなく SQL を用いて明細データの数量を取得するコードを説明します。

```
try {
  if (session != null) {
    /*旧データ読み込み*/
    var sql =
```

(335) 悲観ロックを適用している場合です。

```
      "SELECT ¥"p_number¥" FROM ¥"salesslip$precord¥"" +
      " WHERE ¥"id¥"=" + salesslip.id + " AND" +
      " ¥"precordjshid¥"=" + (precord.PNo - 1);
    /*print("sql="+sql);*/
    var o_PNumber = session.createSQLQuery(sql).uniqueResult();
    /*print("o_PNumber="+o_PNumber);*/
    var suryou = product4s.stock;
    var syukko_num = precord.PNumber - o_PNumber;
    print("now suryou="+suryou+",syukko_num_diff="+syukko_num);
    if (syukko_num != 0) {
      if (suryou - syukko_num < 0) {
        return precord.PName + "の在庫 "+suryou+" に対して "+syukko_num+" を出庫しようとしました。";
      }
      product4s.stock = suryou - syukko_num;
    }
  }
} catch (e) {
  e.printStackTrace();
}
return null;
```

- 繰り返しコンテナは別テーブルとして用意されます。これは「モデル名$繰り返しコンテナ名」というテーブルになります。
- 繰り返しコンテナID項目の物理項目名は常に「繰り返しコンテナ名jshid」となります。
- 繰り返しコンテナテーブルの主キーは、親となるモデルの"主キー"と、"繰り返しコンテナ名jshid"の複合キーになります。
- 「繰り返しコンテナ名jshid」は 0 から開始されます。そのため上のコードでは precord.PNo - 1 と、1を減じた値を使って参照しています。

SQL を利用する説明の詳細は "SQLを利用する" の節もあわせてお読みください。

Service内部で動作するスクリプトを用意する

Service の内部でさらにスクリプトを用意する方法もあります。一つのトランザクション内で、任意のモデルを操作することができます。

例えば model1 の登録、更新、削除時に、同一トランザクションでスクリプトを実行したい場合は次のスクリプトを用意することができま す。

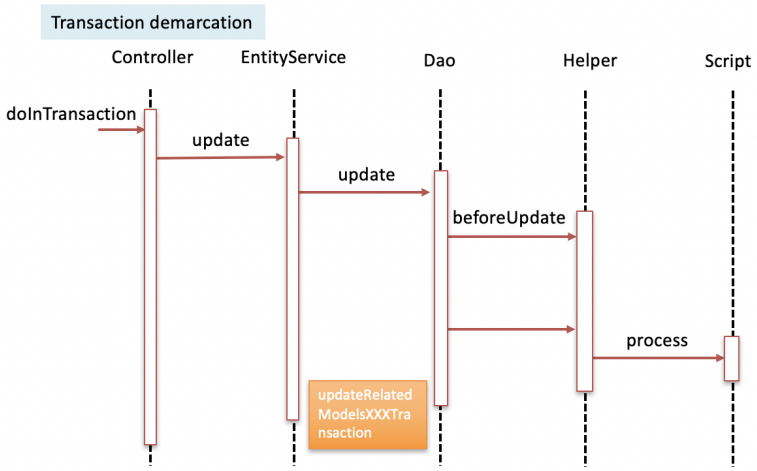


図 <モデルID>EntityService_UpdateRelatedModels<タイミング>Trasaction.js

登録

WEB-INF/script/model1/Model1EntityService_updateRelatedModelsInsertTransaction.js

更新

WEB-INF/script/model1/Model1EntityService_updateRelatedModelsUpdateTransaction.js

削除

WEB-INF/script/model1/Model1EntityService_updateRelatedModelsDeleteTransaction.js

これらのスクリプトは Designer からは設定できません。直接、ファイルを用意してください。

process 関数を作成することで、同一トランザクション内で動作するスクリプトを記述することができます。

```
function process() {

(ここに開発者独自のコードを記載してください。)

}
```

トランザクションスクリプトとの違い

トランザクションスクリプトは対象モデルを明示することで、スクリプトで直接、対象モデル（オブジェクト）を利用することができます。すなわち、対象モデル（オブジェクト）を別途、Dao を使って取得する必要はありません。その代わり、このモデルに紐づいていることが必要です。関連のないモデル（オブジェクト）を利用することはできません。

ここで説明した方法（EntityService の内部から呼び出すスクリプト）は、トランザクションが開始された状態であることがわかっています。そのため、スクリプト内で Dao を使って (1) Aモデルの取得(SELECT) (2) Bモデルの登録(INSERT) (3) Cモデルの更新(UPDATE) を記述すると、これらの処理はすべて同一のトランザクションとして扱われます。

"ヘルパ > 登録/更新/削除" スクリプトとの違い

図からわかるように、ヘルパのスクリプトもまた EntityService のトランザクション内で実行されます。そのため上で説明した方法（EntityService の内部から呼び出すスクリプト）とヘルパのスクリプトは、トランザクション境界という視点では違いはありません。

細かい違いとして、あるモデルの更新時に（同じモデルの）他のデータの更新を行う処理を考えた場合、更新するデータの数だけ、そのモデルに関するヘルパのbeforeUpdateメソッドに紐づくスクリプトが呼び出されます。これが冗長である場合は、本方法が向いているといえます。

本方法が向いている例:

- 有効/無効のフラグを「有効」にして保存した場合、「有効」になっている（同じモデルの）データを「無効」に更新する。つまり、有効データを常に1件のみにする。
- 有効/無効のフラグが「有効」のデータを削除した場合、（同じモデルの）データの中から最新のデータを「無効」から「有効」に更新する。

SQLを利用する

SQLでデータを取得する

Wagbyが内部で利用しているデータベース操作のためのミドルウェアHibernateが提供するsessionオブジェクトを用いて、任意のSQLを実行することができます。トランザクションスクリプト内では、このsessionオブジェクトを利用できます。

次の例は、アカウントモデル(juser)の個数を求めるSQLを実行します。

```
try {
  if (session != null) {
    var o = session.createSQLQuery(
      "SELECT COUNT(*) FROM ¥"juser¥"").uniqueResult();
    print("o="+o);/* デバッグ用 */
  }
} catch (e) {
  e.printStackTrace();
}
```

- createSQLQuery の戻り値の型は、お使いのデータベースによって異なります。Integer, Long, BigInteger などの可能性があります。[詳細...]
- モデル名（英語）に "session" を使うことはできません。この名前は（暗黙オブジェクトとして）予約されています。
- ここで用いているsessionは、Hibernateが提供するクラスです。sessionの利用は、Hibernateの使い方に準じます。
- sessionは、トランザクションスクリプトでは常に有効です。それ以外のスクリプトでは、呼び出す場所によって無効の場合があります。よってトランザクションスクリプト以外の場所で用いる場合、上記例のようにsessionがnullかどうかの判定を加えるようにしてください。
- トランザクションスクリプトで利用できるデータベースコネクション（session）はクローズする必要がありません。内部で自動的にクローズします。
- 上の例では項目名やテーブル名を二重引用符で囲んでいますが、このルールはお使いのデータベースによって変わります。（内蔵DBであるHSQLDB利用時は、二重引用符で囲みます。）このため、開発機と本番機で異なるデータベースを用いる場合は、スクリプトファイルの変更を行う必要があるかも知れませんので、ご注意ください。

トランザクションスクリプト以外の場所で session を利用する

トランザクションスクリプト以外の場所で session を利用する必要がある場合は、まず session が使えるかどうかを確認してください。session が null の場合、スクリプト内で HibernateUtil.openSession() を使って取得します。例を示します。

```
var HibernateUtil = Java.type("jp.jasminesoft.jfc.app.HibernateUtil");
var session = HibernateUtil.openSession();
try {
  /* session を利用した処理を実装する */
  ...
} catch (e) {
  e.printStackTrace();
} finally {
  if (session !== null) {
    session.close();/* 忘れないこと */
  }
}
```

開発者が独自にオープンしたコネクションは、必ずクローズ処理を行ってください。クローズを忘れると、利用できるデータベースセッションが不足し、運用途中で実行時エラーになります。

クローズ漏れの懸念がある場合は"Wagby Developer Network > Javaを用いたカスタマイズ > データベース > データベースのコネクションプーリング数を監視する"の説明にしたがって、クローズを忘れた箇所を特定し、修正してください。

ストアドプロシージャを利用する

トランザクションスクリプトからストアドプロシージャを呼び出す例を紹介します。Hibernate の session オブジェクトが利用できる場合は次のとおりです。

```
try {
  // 戻り値がないパターン（更新なし、データ取得のみ）
  var call1 = "{CALL FOO()}";
  session.createSQLQuery(call1).executeUpdate();

  // 戻り値が1つのパターン（更新なし、データ取得のみ）
  var call2 = "{CALL BAR()}";
  var result = session.createSQLQuery(call2).uniqueResult();
  print(result);

  // 戻り値が複数のパターン（更新なし、データ取得のみ）
  var call3 = "{CALL BAZ()}";
  var list = session.createSQLQuery(call3).list();
  print(list);
```

```
} catch(e) {
  e.printStackTrace();
}
```

- ストアドプロシージャの戻り値が単一の場合は uniqueResult() を、複数の場合は list() を使います。

Hibernate の session オブジェクトを用意する

トランザクションスクリプト外でストアドプロシージャを呼び出す場合は、Hibernate の session オブジェクトを用意してください。

```
var HibernateUtil = Java.type("jp.jasminesoft.jfc.app.HibernateUtil");
var session = HibernateUtil.openSession();

try {
  var tx = session.beginTransaction();
  var call = "{CALL POWER(2, 3)}";
  var result = session.createSQLQuery(call).uniqueResult();
  print(result);
  tx.commit();
} catch(e) {
  e.printStackTrace();
  tx.rollback();
} finally {
  if (session !== null) {
    session.close();
  }
}
```

- トランザクションスクリプト外の場合は Hibernate の session オブジェクトを直接取得し、トランザクションの開始と終了を明示的に行なったあと、session オブジェクトをクローズする必要があります。
- 上で利用した POWER(2, 3) は 2の3乗を計算する関数で、内蔵データベース(HSQLDB)に組み込まれています。実際の利用時は、この部分を適切に変更してください。

IN/OUTパラメータを利用する

ストアドプロシージャの IN/OUT パラメータを利用することもできます。Hibernate が提供する JDBC プログラミング方法を利用します。

```
var JFCUtils = Java.type("jp.jasminesoft.jfc.JFCUtils");
var HibernateUtil = Java.type("jp.jasminesoft.jfc.app.HibernateUtil");
var Work = Java.type("org.hibernate.jdbc.Work");
var DbUtils = Java.type("org.apache.commons.dbutils.DbUtils");
var Types = Java.type("java.sql.Types");

var session = HibernateUtil.openSession();

try {
  /* トランザクションを開始する */
  var tx = session.beginTransaction();
  var out = null;

  var MyWork = Java.extend(Work, {
    execute: function(connection) {
      var st = null;
      try {
        // ストアドプロシージャ呼び出し（更新なし、データ取得のみ）
        st = connection.prepareCall("{CALL XXX(?, ?)}");

        // IN パラメータの指定(1番目のパラメータ)
        st.setInt(1, id);

        // OUTパラメータの指定(2番目のパラメータ)
```

```
        st.registerOutParameter(2, Types.INTEGER);

        st.execute();

        // OUTパラメータの取得
        out = st.getInt(2);
      } finally {
        DbUtils.closeQuietly(st);
      }
    }
  });
  session.doWork(new MyWork());
  print(out);//デバッグ用

  /* コミットする */
  tx.commit();
} catch (e) {
  e.printStackTrace();
  tx.rollback();

} finally {
  if (session !== null) {
      session.close();
  }
}
```

- function(connection) {…} の内部は、一般的な JDBC プログラミングスタイルを利用できます。
- 上の例では整数型となっていますが、実際のご利用では適切な型を指定してご利用ください。
- WorkクラスはHibernateが提供します。WorkクラスのexecuteメソッドではJava.sql.Connectionを利用できます。つまりJDBC APIを用いたLow Level Layerのプログラミングが行なえるため、開発者の自由度は高まります。
- 上の例では Hibernate の session オブジェクトを直接、取得していますが、トランザクション境界内部で（すでに session が使える状態で）利用する場合は openSession と beginTranscation そして close 処理は割愛できます。

時間のかかるストアドプロシージャを呼び出す

Statement クラスが提供する setQueryTimeout を指定することができます。サンプルコードを示します。

```
var JFCUtils = Java.type("jp.jasminesoft.jfc.JFCUtils");
var HibernateUtil = Java.type("jp.jasminesoft.jfc.app.HibernateUtil");
var Work = Java.type("org.hibernate.jdbc.Work");
var DbUtils = Java.type("org.apache.commons.dbutils.DbUtils");
var Types = Java.type("java.sql.Types");

var HibernateUtil = Java.type("jp.jasminesoft.jfc.app.HibernateUtil");
var Work = Java.type("org.hibernate.jdbc.Work");
var DbUtils = Java.type("org.apache.commons.dbutils.DbUtils");
var Types = Java.type("java.sql.Types");

var session = HibernateUtil.openSession();

try {
  var tx = session.beginTransaction();

  var MyWork = Java.extend(Work, {
    execute: function(connection) {
      var st = null;
      try {
        st = connection.prepareCall("EXEC TEST1 ?, ?, ?");
        st.setQueryTimeout(10000); /* タイムアウトの設定 */
        st.registerOutParameter(3, Types.INTEGER);
        st.setInt(1, 4);
        st.setInt(2, 5);
```

```
        st.execute();
        print("result:[" + st.getInt(3) + "]");
      } catch (e) {
        e.printStackTrace();
      } finally {
        DbUtils.closeQuietly(st);
      }
    }
  });
  print("Original1 procedure execute start time: ", new Date());
  session.doWork(new MyWork());
  print("Original1 procedure execute end time: ", new Date());
  print("");
  tx.commit();
} catch (e) {
  e.printStackTrace();
  tx.rollback();
} finally {
  if (session !== null) {
      session.close();
  }
}
```

・　function(connection) {…} の内部は、一般的な JDBC プログラミングスタイルを利用できます。

SQLやストアドプロシージャでデータを更新する

SQLまたはストアドプロシージャで更新・削除を行う場合はロックとキャッシュに関する制御を開発者自身で記述する必要があります。

(336) (337)

ロックの取得

更新したテーブルに対応した Wagby のモデルがあり、かつ悲観ロック方式を適用している場合、そのモデルのロックを取得する必要があります。この手順を踏まない場合、別の利用者がこのモデルの対象データを更新中かどうかを考慮せずにデータを更新してしまうため、データの整合性がとれなくなります。

(338) (339)

model1 モデルの1件のデータのロックを取得するコードは次のとおりです。

```
var LockUtils = Java.type("jp.jasminesoft.jfc.core.util.LockUtils");
var model1DaoHelper = p.appctx.getBean("Model1DaoHelper");
var lockName = model1DaoHelper.getLockName();
...
try {
  LockUtils.lock(lockName, "1000", p);/*ロック取得*/
  ...
  // 更新処理
  ...
} catch (e) {
  throw e;/*ロールバックのため*/
} finally {
  LockUtils.release(lockName, "1000", p);/*ロック解放*/
```

(336) 検索系のSQLを使う限りにおいては、ロックとキャッシュに関する制御は不要です。

(337) 更新を伴う場合は（SQLまたはストアドプロシージャの利用に代わって）Wagby が提供する Service の利用も検討するとよいでしょう。Service はロックの取得と解放、および検索結果のキャッシュ処理が自動的に行われるため、開発者はこれらを意識する必要がありません。

(338) 更新するテーブルが Wagby の管理外（すなわち Wagby のモデルとして定義されていない）場合は、このロック取得は不要です。

(339) Wagby の定義で、このモデルのロック方式を楽観ロックとすることができます。この場合も以下に説明するロック取得は不要です。ただし実行する SQL またはストアドプロシージャで一般的な楽観ロック方式の手順に従った対応を行なってください。（更新処理の前後に「バージョン管理用カラム」の値を取得し、この値が同一かどうかを確認します。同一であれば他者の更新はなかったと判断できます。さらに更新時には、バージョン管理用カラムの値をインクリメントしてください。）

```
}
```

- LockUtils.lockメソッドおよびreleaseメソッドの第一引数はロック名になります。ロック名はXXXDaoHelper#getLockName()で取得してください。
- LockUtils.lockメソッドおよびreleaseメソッドの第二引数は対象データの主キーの文字列表現となります。複合主キーの場合はSEPで連結した文字列としてください。詳細はWagbyが生成したコード<モデルID>Helper.java の getPrimaryKeyAsString メソッドをお読みください。（主キーの文字列表現は）このメソッドの戻り値と同じ結果となるようにしてください。
- 複数のデータを更新する場合、主キーを個別に指定して複数回、lock メソッドと release メソッドを実行してください。
- 他者がすでに当該データの更新画面を開いている場合、lock メソッドは失敗します。この場合は実行時例外 PessimisticLockException が生じます。catch 節で例外を判定することができます。例えば p.errors に更新失敗のメッセージを格納し、処理を中断させるというアプローチがあります。

モデル全体をロックする (8.1.0)

ストアドプロシージャですべてのデータを更新するため、モデル全体をロックしたいという場合は次のコードになります。

```
var LockUtils = Java.type("jp.jasminesoft.jfc.core.util.LockUtils");
var model1DaoHelper = p.appctx.getBean("Model1DaoHelper");
var lockName = model1DaoHelper.getLockName();
...
try {
  LockUtils.modelLock(lockName, p);/*ロック取得*/
  ...
  // 更新処理
  ...
} catch (e) {
  throw e;/*ロールバックのため*/
} finally {
  LockUtils.releaseModelLock(lockName, p);/*ロック解放*/
}
```

- LockUtils.modelLockメソッドおよびreleaseModelLockメソッドの第一引数はロック名になります。ロック名はXXXDaoHelper#getLockName()で取得してください。
- 対象モデルの1データでも他者が更新中の場合、modelLockメソッドは失敗します。
- modelLockメソッドが成功すると、他者は一切このモデルの更新が行えなくなります。更新処理終了後、速やかにロックを解放するようにしてください。

キャッシュのクリア

更新したテーブルに対応した Wagby のモデルがある場合、そのモデルのキャッシュをクリアします。

(340) (341) (342)

model1 モデルのキャッシュクリアを行うコードは次のとおりです。

```
// ... データベースの更新が終了した ...

var CacheManagerClass = Java.type("jp.jasminesoft.wagby.app.CacheManager");
var cman = CacheManagerClass.getInstance(p);
cman.clearModel1();
```

- CacheManagerClass のパッケージ名は適切に読み替えてください。
- モデルのキャッシュをクリアするメソッドclear<モデルID>が、モデルごとに用意されています。このメソッドを実行してください。

ご注意ください - 同じモデルの操作はできません

(340) Wagby は標準で、データベースから読み込んだ値をメモリに保持することでパフォーマンス向上を図っています。Wagby が提供する Service を使う場合、キャッシュのクリアは自動的に行われます。SQLを用いて更新した場合、キャッシュのクリア処理を開発者が行ってください。

(341) 更新したテーブルが Wagby の管理外（すなわち Wagby のモデルとして定義されていない）場合は、このキャッシュクリアは不要です。

(342) Wagby の定義で、このモデルのキャッシュを無効にすることもできます。この場合もキャッシュクリア処理は不要となります。しかしこの方針は実行時のパフォーマンスに影響するため、慎重に判断してください。

対象と同じモデルのデータの操作を　SQLまたはストアドプロシージャを使って操作することはできません。

その理由は、Wagby内部の「ストアモデル」が、最終的に Hibernate によって SQL に変換され、DB 更新されるためです。SQLやストアドプロシージャを使ってデータベースの値を書き換えても、そのあと Hibernate の update メソッドで、もともと Wagby が管理しているストアモデルの情報で上書き保存されてしまいます。

同じモデルの操作を行う要件であればストアドプロシージャではなく、Wagbyが提供するオブジェクトを操作するようにしてください。具体的にはストアモデルが提供している setter/getter メソッドを使って値を変更してください。

SQLやストアドプロシージャを使って別のモデルを操作することは問題ありません。

プログラムによるトランザクション管理

Wagbyは標準で Spring が提供する宣言的トランザクションを利用しています。これとは別に開発者が直接、トランザクション管理を行うことも可能です。（プログラマティックトランザクション）

具体的には Spring が提供する TransactionTemplate クラスを使って、トランザクション境界の中で動作するコードを記述することができます。この処理をスクリプトで実現することができますが、事前知識として Java で書く方法を知っておくとよいでしょう。この詳細は "Javaを用いたカスタマイズ > Service/Daoクラス > プログラムによるトランザクション管理" をお読みください。

例 コントローラクラスの拡張（独自ボタン）

customer モデルの詳細画面に独自ボタン(イベント名:Original1)を配置し、ボタン押下時に実行される処理として作成しています。　このスクリプトファイルは　WEB-INF/script/customer　フォルダに ShowCustomer_Original1.js として保存します。

```
function process() {
  var ExcelFunction = Java.type("jp.jasminesoft.util.ExcelFunction");
  var Jfcerror = Java.type("jp.jasminesoft.jfc.error.Jfcerror");

  // トランザクション処理結果の戻り値が必要な場合は
  // TransactionCallback クラスを利用します。
  // 今回は戻り値が不要なので TransactionCallbackWithoutResult クラス
  // を使っています。
  var TransactionCallbackWithoutResult = Java.type(
      "org.springframework.transaction.support.TransactionCallbackWithoutResult");

  // TransactionCallbackWithoutResult クラスの doInTransactionWithoutResult()
  // メソッドをオーバーライドします。
  var MyTransactionCallbackWithoutResult
      = Java.extend(TransactionCallbackWithoutResult, {
          doInTransactionWithoutResult: function(status) {
            // トランザクションが開始された状態で本メソッドが実行され、
            // メソッド内で Exception が発生すると自動的にロールバック
            // されます。Exception が発生せずに本メソッドが正常終了する
            // と自動的にコミットされます。
            var helper = p.appctx.getBean("JnewsHelper");
            var service = p.appctx.getBean("JnewsEntityService");

            // 1件目のデータ登録
            var jnews = new Packages.jp.jasminesoft.wagby.model.jnews.Jnews();
            helper.initialize(jnews, p);
            jnews.limitdate = ExcelFunction.TODAY();
            jnews.title = "お知らせ1";
            service.insert(jnews);

            // 2件目のデータ登録
            jnews = new Packages.jp.jasminesoft.wagby.model.jnews.Jnews();
            helper.initialize(jnews, p);
            jnews.limitdate = ExcelFunction.TODAY();
```

```
            jnews.title = "お知らせ2";
            service.insert(jnews);
          }
      });

  try {
    // Spring が提供する TransactionTemplate を使って、
    // @Transactional アノテーションを用いずにプログラム実装による
    // トランザクション機能を実現します。
    var transactionTemplate = p.appctx.getBean("RequiredTransactionTemplate");
    transactionTemplate.execute(new MyTransactionCallbackWithoutResult());
  } catch (e) {
    var error = new Jfcerror();
    error.content = "エラーが発生しました。" + e.message;
    p.errors.addJfcerror(error);
  }

  // 元の画面へ遷移する。
  var id = p.request.getParameter("customerid");
  return "redirect:showCustomer.do?customerid="+id;
}
```

このコードの解説はこちらをお読みください。（同じ処理を Java 言語で実装したコードを用意しています。）

TransactionCallback内でHibernateのセッションを取得する

TransactionCallbackWithoutResult や TransactionCallback 内部で Hibernate のセッションオブジェクトを取得することができます。

```
var session = p.appctx.getBean("sessionFactory").getCurrentSession();
```

もっと詳しく

- Wagby Developer Network > Service/Daoクラス（Javaエンジニア向け）

Daoを利用する

トランザクションスクリプトは通常、モデル定義で関連しているモデル間で値の変更に関するコードを記述します。しかし Dao クラスを使うことで、任意のモデルを操作することができます。

コード例

トランザクションスクリプトでは、次のコードを用いてDaoを取得することができます。

```
var dao = p.appctx.getBean(Dao名);
```

Dao名は "<モデルID>Dao" となります。モデルIDはキャメル記法で表現します。例えば customer モデルに対するDao名は "CustomerDao" になります。

具体的なコード例を紹介します。

```
var dao = p.appctx.getBean("JuserDao");
var user = dao.get("admin", true); /* 1件のデータを取得 */
print(user); /* デバッグ用ログ出力 */
user.name = "ジャスミン太郎";
dao.update(user); /* 更新 */
```

dao.get の第二引数はロックを取得することを示します。このあとの dao.update の実行によりロックは解放されます。

Dao クラスはトランザクションが開始されている場合に利用することができます。(Dao 自身はトランザクション境界ではありません。つまり自分自身でトランザクションを開始しません。) 一方、上で説明した Service クラスはトランザクション境界になります。この違いから、Daoを使える場所はトランザクションスクリプトに限られます。トランザクションスクリプトは呼び出しの前後でトランザクションの開始と終了が自動的に実行されるため、Daoによる更新処理は正しく処理されます。

(343)

ヘルパを利用する

スクリプトで (Wagbyが自動生成した) ヘルパクラスを使うことができます。設計情報に記述した初期値や式を再利用できるだけでなく、s2p や p2s メソッドを用いたストアモデルとプレゼンテーションモデルの変換も再利用できます。

例を示します。quotation というストアモデルをプレゼンテーションモデルに変換し、それを po という変数に格納します。

```
var IPresentationHelper = Java.type("jp.jasminesoft.jfc.IPresentationHelper");
var helperClass = p.appctx.getBean("QuotationPHelper");
var po = helperClass.s2p(quotation, p, IPresentationHelper.SHOW);

print(po);
```

- ヘルパクラスも DI コンテナである p.appctx の getBean メソッド経由で取得します。
- jp.jasminesoft.jfc.IPresentationHelperインタフェースは定数 SHOW, UPDATE, CSVDOWNLOAD, CREATEOBJECT という4つのモードを提供します。通常は SHOW または UPDATE を使います。s2p メソッドに SHOW を渡すと表示用のプレゼンテーションモデルへ変換します。UPDATE を渡すと更新画面で利用できるプレゼンテーションモデルへ変換します。これには選択肢の候補が含まれています。[詳細...]

ロックの取得と解除

悲観ロック利用時の問題点

次のコードは条件によって更新処理を行うサンプルです。このコードはロックの処理に問題があります。(なお、ここでは悲観ロックを利用した場合についての説明となっています。)

```
var entityService = p.appctx.getBean("JuserEntityService");
var user = entityService.findById("user01", true);
if (user.kind == 1) {
  user.name = "ジャスミン太郎";
  entityService.update(user);
}
```

具体的には条件に合致しない場合は update メソッドが呼び出されません。このとき、findById の第二引数に true を指定したためロックを取得していますが、このロックが解放されずにスクリプトが終わる可能性があります。

ロック解除が行われない場合、メニュー表示時またはログオフ時までは、このロックは保持されたままとなります。その間、他の利用者はこのデータを更新することができません。

(344)

解決方法

findById メソッドでロックを取得するのではなく、LockUtils クラスが提供する update メソッドの直前でロックを取得します。

---

(343) トランザクションスクリプトではServiceとDaoのいずれも利用できますが、トランザクションが開始されていることがわかっているため、通常はDaoを使います。トランザクションスクリプト以外のスクリプトでは、Serviceを用いてください。(Dao はトランザクション管理機能を持ちません。Dao利用時にトランザクションが開始されていなければエラーが発生します。[詳細...])

(344) Wagbyのロックは、メニュー画面表示またはログオフ時に、自動解除されます。

```
var LockUtils = Java.type("jp.jasminesoft.jfc.core.util.LockUtils");
var userDaoHelper = p.appctx.getBean("JuserDaoHelper");
var entityService = p.appctx.getBean("JuserEntityService");

var user = entityService.findById("user01");
if (user.kind == 1) {
  try {
    LockUtils.lock(user, userDaoHelper);/*ロック取得*/
    user.name = "ジャスミン太郎";
    entityService.update(user);
  } catch (e) {
    throw e;/*ロールバックのため*/
  } finally {
    LockUtils.release(user, userDaoHelper);/*ロック解放*/
  }
}
```

- LockUtils クラスは Wagby が提供します。lock と release というメソッドを用意しています。
- 第一引数には対象オブジェクトを指定します。第二引数には、対象オブジェクトのDaoHelperクラスを指定します。
- catch 節では、この処理をロールバックさせるために、例外をスローするようにしてください。
- LockUtils#lock メソッドでロックの取得ができなかったときは例外 PessimisticLockException が返されます。

例外 PessimisticLockException を補足したい場合

上のコードはすべての例外を throw してロールバックしていますが、PessimisticLockException を補足したい場合は次のように記述することができます。

```
var PessimisticLockException = Java.type("javax.persistence.PessimisticLockException");
...
try {
  LockUtils.lock(user, userDaoHelper);/*ロック取得*/
  user.name = "ジャスミン太郎";
  entityService.update(user);
} catch (e if e instanceof PessimisticLockException) {
  print("lock error " + e);
  throw e;
} catch (e) {
  print("error " + e);
  throw e;
} finally {
  LockUtils.release(user, userDaoHelper);/*ロック解放*/
}
```

クライテリアを利用する

Serviceを使って複数件のデータを取得する場合、クライテリアを使って任意の検索条件を指定できます。モデル設計で、検索条件を有効にしていない項目であってもクライテリアとして（条件を）指定することができます。

(345)

クライテリアを用意する

モデル毎に用意される CriteriaConverter を使います。コード例を示します。

```
var criteriaConverter = p.appctx.getBean("Model1CriteriaConverter");
var criteria = criteriaConverter.defaultCriteria();
```

(345) ただし、モデル定義で検索条件を有効にしていない場合、データベースにインデックスが作成されていません。必要に応じて、手動でインデックスを作成してください。

モデルの定義で暗黙条件を指定していた場合、上の方法で取得した criteria は暗黙条件がセットされた状態となっています。（この回避策をこのあと説明します。）

メタクラスを使う

criteria を用いた検索条件の指定では、対象項目名は文字列ではなく、Wagbyが提供するメタクラスを使うことができます。例えば model1 モデルを定義するとメタクラス Model1Meta が自動生成されます。これは項目名と型の情報（メタデータ）を保持する特別なクラスです。メタクラスを使うことでスペルミスによる実行時エラーといった、デバッグしにくい問題を回避することができます。

メタクラスを使って criteria に検索条件を設定する例を示します。

```
var Model1Meta = Java.type("jp.jasminesoft.wagby.model.model1.Model1Meta");
var model1Meta = new Model1Meta();
criteria.eq(model1Meta.item1, 1000); /* item1 の値が 1000 のデータで絞り込む */
```

eq は等しい、という条件になります。他にもさまざまな条件を指定することができます。

検索する

Service が提供する find メソッドを利用するときに、第一引数にクライテリアを渡します。戻り値はリストとなっており、この中に（検索条件に合致した）複数のストアモデルが格納されています。

```
var list = Model1Service.find(criteria);

if (list != null && list.size() > 0) {
  // list から 1 つずつオブジェクトを取り出す
  for (var i=0; i<list.size(); i++) {
    var m = list.get(i);
    print(m);
  }
}
```

並び替えを指定する [1]

Order クラスを使って、criteria に並び替え条件を追加することができます。例を示します。

```
var OrderClass = Java.type("org.hibernate.criterion.Order");
var order = OrderClass.desc(Model1Meta.start_date.name());
criteria.addOrder(order);/* 並べ替え条件の追加 */

/*criteriaを使って検索する*/
...
```

- Orderクラスは、ascとdescメソッドを利用できます。それぞれ昇順、降順の指定です。
- asc/descメソッドの引数は Meta クラスの項目名を使います。項目名.name() とすることで、適切な項目名が渡されます。Meta クラス利用時は、項目名の部分はキャメル記法を使わず、設計情報に記載した項目IDをそのまま使ってください。（この例では startDate ではなく start_date となっています。）
- 上の例では一つの order を指定していますが、criteria.addOrder に order の配列を渡すことで、複数の並び替え指定を行うことができます。
- 並び順を指定しなかった場合でも、主キーでの並び替えは必ず行われます。addOrder は、主キーによる並べ替えに続くルールとして扱われます。

並び替えを指定する [2]

Wagbyでは、主キーは必ずソートキーとして扱われます。すなわち SQL の ORDER BY 句に主キーが含まれます。

> 主キーを ORDER BY に含めるのはページネーション機能に SQL の LIMIT を利用しているためです。ORDER BY の結果の行が一意とならない場合、改ページしても(ORDER BY のソートルールの曖昧さにより)同じデータが次のページに出現してしまうことがあります。これを避けるため、ORDER BY 句に主キーを含めています。このルールを無効にすることはできません。

(346)

この制約を回避するために、CriteriaConverter クラスの defaultCriteria(Order[]) メソッドを使うことができます。開発者が引数に指定した並べ替え条件を優先し、主キーによる並び替えは ORDER BY 句の最後に行うようにすることができます。（つまり優先度を下げることで影響を抑えます。）

```
var OrderClass = Java.type("org.hibernate.criterion.Order");
var CustomerMetaClass = Java.type("jp.jasminesoft.wagby.model.customer.CustomerMeta");
var customerMeta = new CustomerMetaClass();

var orders = [
    OrderClass.asc(customerMeta.companyname.name()),
    OrderClass.asc(customerMeta.deptname.name())
];

var criteriaConverter = p.appctx.getBean("CustomerCriteriaConverter");
/* 開発者が並び順を指定する */
var criteria = criteriaConverter.defaultCriteria(orders);
```

暗黙条件を適用させないクライテリアを用意する

暗黙条件を適用させたくない場合は criteriaConverter を使わず、次のようにしてください。

```
var DetachedCriteria = Java.type("jp.jasminesoft.jfc.hibernate.DetachedCriteria");
var CustomerClass = Java.type("jp.jasminesoft.wagby.model.customer.Customer");
var criteria = DetachedCriteria.forClass(CustomerClass.class);
```

スクリプトでクライテリアをカスタマイズする

スクリプトでクライテリアをカスタマイズすることもできます。Wagbyの標準機能による検索では、複数の検索項目に検索条件を指定した場合は常にAND検索となります。例えば、ここでOR検索を行わせたい場合はCriteriaをカスタマイズすることで対応できます。

複合キーの場合

複合キークラスをストアモデルのインナークラスとして用意しています。 zaiko モデルが複合キーの場合は次のような記述となります。

```
var Zaiko = Java.type("jp.jasminesoft.wagby.model.zaiko.Zaiko");
var key = new Zaiko.Key();
key.pkey1 = xxx;/* xxx には、主キーの値が入ります */
key.pkey2 = yyy;/* yyy には、主キーの値が入ります */
var zaiko = zaikoEntityService.findById(key,true);/* 第二引数がtrueのときロックをかける */
```

例外発生によるロールバック

トランザクションスクリプトでは return 文で任意の文字列を返すことで、処理をロールバックさせることができます。

ヘルパ系のスクリプトでは、例外 BusinessLogicException をスローすることでロールバックさせることができます。この方法を説明します。

BusinessLogicException

スクリプト内で BusinessLogicException をスローすることで、このトランザクションをロールバックさせることができます。

(346) この対応により、"LIMIT を使用するときは、結果の行を一意な順序に制約する ORDER BY 句を使用する" というデータベースのルールに準拠しています。

```
throw new Packages.jp.jasminesoft.jfc.core.exception.BusinessLogicException("エラーメッセージ");
```

ここで設定したエラーメッセージが画面に表示されます。

対象となるスクリプト

この方法が利用できるスクリプトは、ヘルパ系で、かつ session (Hibernateのsession) が null でない場合に有効です。(*)

例えばヘルパの登録処理または更新処理のスクリプトで、ある条件に合致した場合に BusinessLogicException をスローさせることで、このトランザクションをロールバックさせるといった動作を実現します。

(347)

トランザクション境界

Wagby のトランザクション境界とは、外部データベースに対して "begin transaction" 命令を発行するタイミングをいいます。この命令が発行されてから、再びデータベースに対して "commit" または "rollback" 命令が発行されるまでは「1つのトランザクション」として扱われます。この間データベースに対して行われた変更処理はすべてコミットされるか、またはすべてロールバックされるかのどちらかになります。

新規登録・コピー登録画面、更新画面、一覧更新画面

コントローラクラス内の doInTransaction メソッドがトランザクション境界となります。

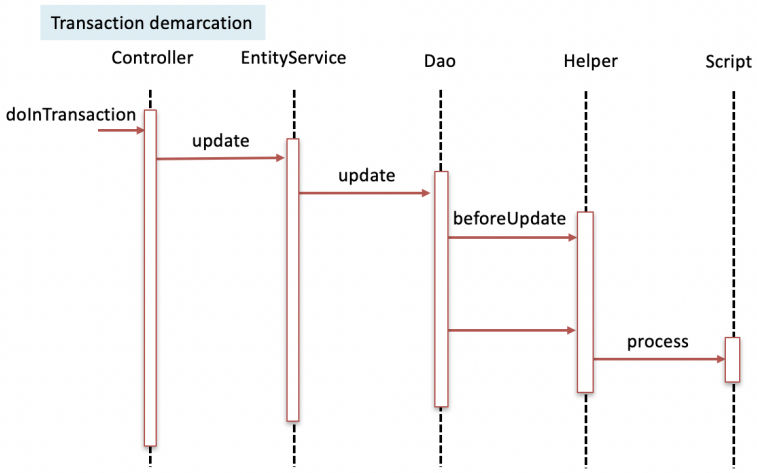


図 トランザクション境界(更新)

削除、アップロード更新画面

コントローラクラスから呼び出される EntityService がトランザクション境界となります。

(347) 適用する前に、if (session != null) {...} で session が有効かどうかを確認してください。

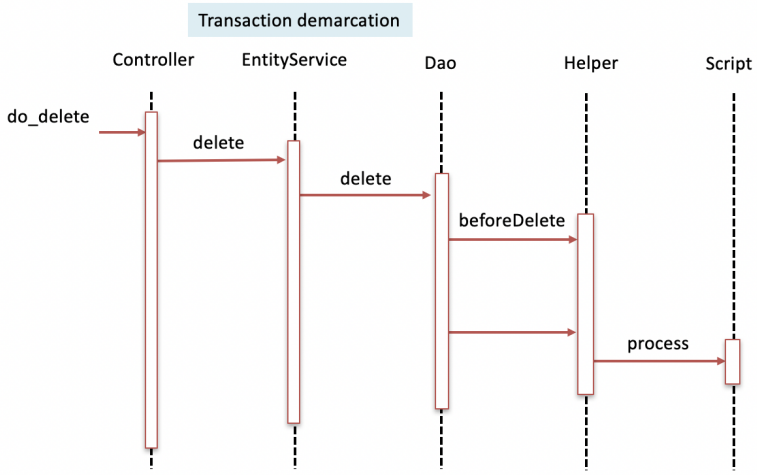


図 トランザクション境界(削除)

doInTransaction メソッドと EntityService の関係

EntityService が提供するメソッドは、まだトランザクションが開始されていない場合は新しくトランザクションを開始するが、すでにトランザクションが開始されている場合は（新しいトランザクションを開始せず）現在のトランザクションをそのまま利用するようになっています。

EntityService と Dao の関係

EntityService が提供するメソッドは、内部で Dao を呼び出します。EntitySerivce は上述したようにメソッド開始時に自動的にトランザクションが開始されます。メソッドが正常終了すれば自動的にコミットされ、例外(Exception)が発生すれば自動的にロールバックされます。一方 Dao はトランザクション管理機能を持ちません。Dao利用時にトランザクションが開始されていなければエラーが発生します。

一覧更新画面や親子同時更新画面での応用

一覧更新画面や親子同時更新画面は doInTransaction メソッドがトランザクション境界となっているため、保存時に（一意制約等のデータベースエラーとなった場合）処理がロールバックされ、編集画面に戻るようになっています。

また、コントローラの一覧更新のスクリプト「データベースコミット前」で例外を発生させると更新処理全体がロールバックされ、編集画面に戻るようになっています。すなわち修正したデータをすべてコミットさせるか、あるいはすべてロールバックさせるかをスクリプトでも制御できるようになっています。[詳細...]

トランザクションスクリプトでもServiceを利用するケース

通常、トランザクションスクリプトで他のモデルの値を操作する場合は Dao を使います。しかしここで説明するように、トランザクションスクリプトであっても Service を使う必要のある場合があります。次のような例を想定します。

- 「売上伝票 (salesslip)」モデルと、「販売商品 (product4s)」モデルがある。
- 売上伝票は、明細として、どの商品をいくつ販売したかを保持する。
- 売上伝票を新規登録するタイミングで、販売商品の在庫数を減らしたい。（トランザクション）

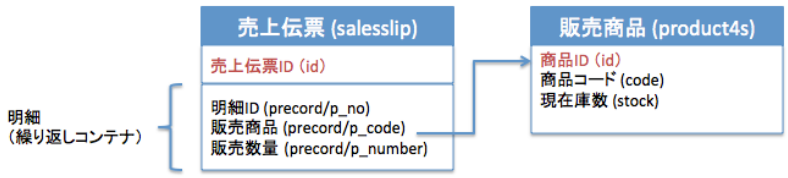


図 売上伝票モデルと販売商品モデル

売上伝票 新規登録時のトランザクションスクリプト

売上伝票モデルの明細（繰り返しコンテナ）に含まれる「商品コード」は、販売商品モデルへの参照項目となっています。この項目にトランザクションクリプトを記述します。

実行したい内容は、売上伝票登録時に、明細に含まれる（いくつ購入したか、という）数量を、商品在庫から減じることです。

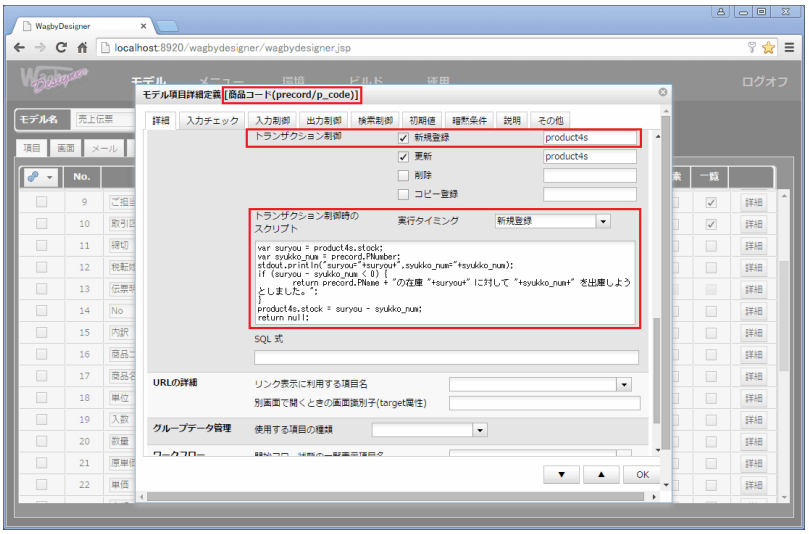


図 新規登録時のトランザクションスクリプト

このときのトランザクションスクリプトは次のようになります。（この時点では Dao も Service も使う必要はありません。）

```
var suryou = product4s.stock;
var syukko_num = precord.PNumber;
/*print("suryou="+suryou+",syukko_num="+syukko_num);*/
if (suryou - syukko_num < 0) {
    return precord.PName + "の在庫 "+suryou+" に対して "+syukko_num+" を出庫しようとしました。";
}
product4s.stock = suryou - syukko_num;
return null;
```

- product4s.stockは、販売商品モデルの在庫数項目を指します。
- precordは明細（繰り返しコンテナ）の1行を指します。明細行が3レコードあった場合、上のスクリプトは3回、実行されます。
- 繰り返しコンテナprecordの、販売数量p_numberを、対応する（販売商品モデルの）在庫から減じています。
- 項目名にはキャメル記法を適用します。p_numebrは、スクリプト中ではPNumberと表記します。
- 処理が正常に終了した場合、nullを返します。

売上伝票 更新時のトランザクションスクリプト

次に更新処理を考えてみます。更新のスクリプトは、登録と同じではありません。

更新の場合は「一度、登録した値」と「更新画面で変更した値」の差をとって、在庫数を再計算する必要があります。　ここで、一度登録した値というのは、データベースに保存された値を指します。　すなわちデータベース上の値と、画面から再入力された値の差を求めるという処理になります。

更新対象のデータとの比較のために更新前のデータをデータベースから取得する場合は別トランザクションで取得する必要があります。

この対応のために、別トランザクションで get 処理を行う仕組み newTransactionEntityService を用いてください。コード例を示します。

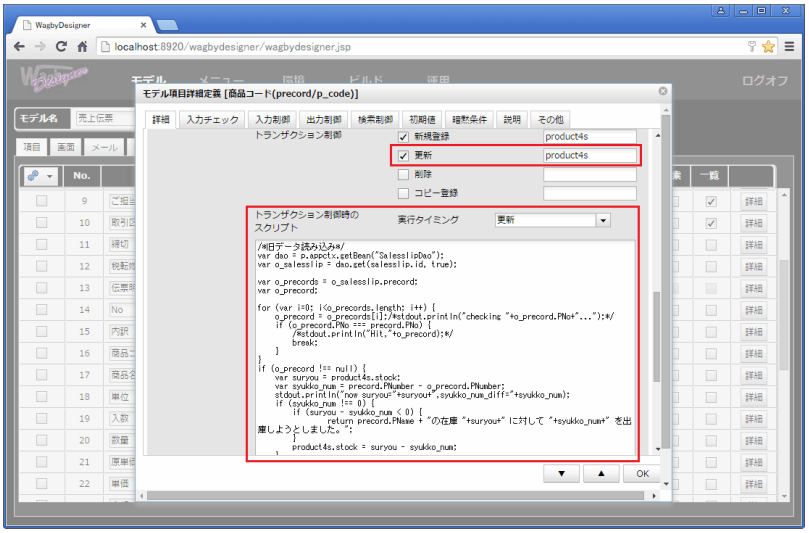


図 更新時のトランザクションスクリプト

```
var entityService = p.appctx.getBean("SalesslipEntityService");
var newTransactionEntityService = entityService.newTransactionEntityService();/*別トランザクションになる*/
var o_salesslip = newTransactionEntityService.findById(salesslip.id, false);/* salesslip.idは主キー */

var o_precords = o_salesslip.precord;
var o_precord;

for (var i=0; i<o_precords.length; i++) {
    o_precord = o_precords[i];
    if (o_precord.PNo == precord.PNo) {
        break;
    }
}
if (o_precord != null) {
    var suryou = product4s.stock;
    var syukko_num = precord.PNumber - o_precord.PNumber;
    /*print("now suryou="+suryou+",syukko_num_diff="+syukko_num);*/
    if (syukko_num != 0) {
        if (suryou - syukko_num < 0) {
         return precord.PName + "の在庫 "+suryou+" に対して "+syukko_num+" を出庫しようとしました。";
        }
        product4s.stock = suryou - syukko_num;
    }
}
return null;
```

- サービスオブジェクトから、データベースの値を取得します。これを o_salesslip という変数に格納します。find は主キーを渡して1件のデータを取得します。
- newTransactionEntityService メソッド経由で取得できるサービスはデータの取得専用です。登録/更新/削除処理などは行えません。(*)
- forループを使って、現在、対象としている明細行とマッチするものを探します。
- マッチした行について、画面からの入力値との差分を求めます。この差を改めて在庫数から減じています。

この例では、一度登録した売上伝票の更新を認めていますが、実際の業務では更新を認めないというシナリオもあります。また更新ではなく、変更のための伝票を新規で登録させるというシナリオもあります。今回のコードは一つの例としてお読みください。

上の例では、データベースの値を取得するためにサービスオブジェクトを用いました。別のアプローチとして、SQL式を使ってデータベースの値を取得することもできます。

トランザクションスクリプトで参照連動の解決を抑制する

上で説明した更新時のトランザクションスクリプトの説明を続けます。このスクリプトでは入力された値とは別に、現在データベースに保持されている値を Dao を使って取得しています。Dao を使うと、参照連動項目もすべて解決します。そのため、データベースに発行するSQLは参照連動の数に比例して増大します。

しかし今回の目的で利用する o_precord.PNo や o_precord.PNumber は参照連動項目ではありません。そこで Dao 利用時に、不要な参照連動項目の解決をスキップさせることでパフォーマンス向上を図ることができます。

DataBindingContext

DataBindingContextもWagbyが提供するクラスです。このクラスに、参照連動の解決を行う項目名を明示することができます。使い方の例を示します。

```
var DataBindingContext = Java.type("jp.jasminesoft.jfc.dao.DataBindingContext");

var dao = p.appctx.getBean("SalesslipDao");
var dataBindingContext = new DataBindingContext();
var targetItemSet = new java.util.HashSet();
targetItemSet.add("customerName");/* 解決したい参照連動項目 */
dataBindingContext.setTargetItemSet(targetItemSet);
var o_salesslip = dao.get(salesslip.id, true, dataBindingContext);/* 第3引数に指定 */
```

このように、dao の get メソッドの第3引数に、dataBindingContext を渡します。この引数が未指定の場合、すべての参照連動項目を解決します。

解決したい参照連動項目が繰り返しコンテナ内の項目の場合、コンテナ名は含めないでください。（コンテナ名の後ろの）項目名のみを指定します。

参照連動処理をスキップする

すべての参照連動処理をスキップする場合、空の dataBindingContext を渡します。

```
var DataBindingContext = Java.type("jp.jasminesoft.jfc.dao.DataBindingContext");

var dao = p.appctx.getBean("SalesslipDao");
var dataBindingContext = new DataBindingContext();
var targetItemSet = new java.util.HashSet();
dataBindingContext.setTargetItemSet(targetItemSet);/* 項目を明示しない */
var o_salesslip = dao.get(salesslip.id, true, dataBindingContext);/* 第3引数に指定 */
```

例

図の更新用トランザクションスクリプトに、参照連動項目の解決をスキップするコードを加えた例を示します。

```
var DataBindingContext = Java.type("jp.jasminesoft.jfc.dao.DataBindingContext");

var dao = p.appctx.getBean("SalesslipDao");
var dataBindingContext = new DataBindingContext();
dataBindingContext.setTargetItemSet(new java.util.HashSet());/* 項目を明示しない */
var o_salesslip = dao.get(salesslip.id, true, dataBindingContext);/* 第3引数に指定 */

print(o_salesslip);/* コンソールに出力。確認用 */

var o_precords = o_salesslip.precord;
var o_precord;
```

(348) Spring の @Transactional(propagation = Propagation.REQUIRES_NEW) を利用しています。

```
for (var i=0; i<o_precords.length; i++) {
    o_precord = o_precords[i];
    if (o_precord.PNo == precord.PNo) {
        break;
    }
}
if (o_precord != null) {
    var suryou = product4s.stock;
    var syukko_num = precord.PNumber - o_precord.PNumber;
    /*print("now suryou="+suryou+",syukko_num_diff="+syukko_num);*/
    if (syukko_num != 0) {
        if (suryou - syukko_num < 0) {
         return precord.PName + "の在庫 "+suryou+" に対して "+syukko_num+" を出庫しようとしました。";
        }
        product4s.stock = suryou - syukko_num;
    }
}
return null;
```

環境タブで設定できる共通スクリプト（スクリプト）

共通関数

"環境 > スクリプト" では「共通関数」「ファイル項目ダウンロード」「ファイル項目アップロード」の実行タイミングで呼び出されるスクリプトを用意することができます。

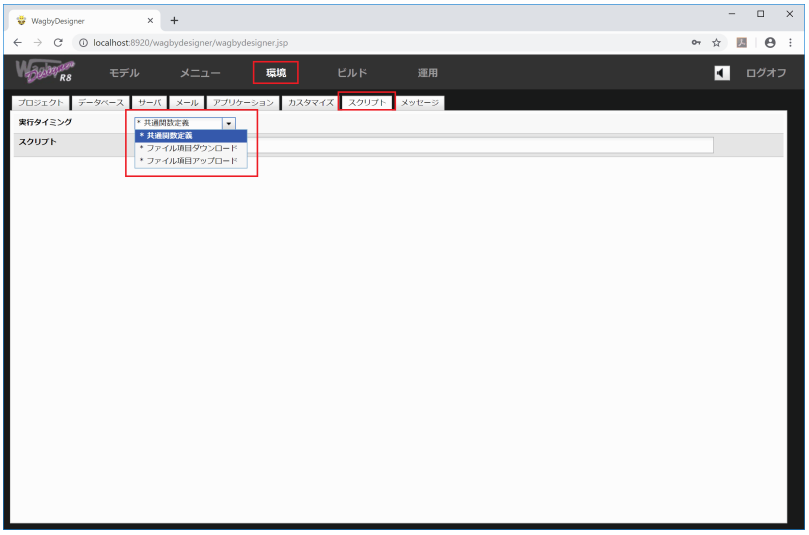


図 環境>スクリプトの定義

共通関数定義

"生成されるスクリプトファイルの詳細 > 共通して利用できる関数" をお読みください。

ファイル項目ダウンロード

（後述）

ファイル項目アップロード

（後述）

ファイル項目ダウンロード

リポジトリで定義したモデル内に存在するすべてのファイル型項目のダウンロード時に共通で呼び出されるスクリプトです。この実体は WEB-INF/script/DownloadFile_beforeDownload.js です。

利用できる暗黙変数

スクリプトでは次の暗黙変数を利用できます。各変数は参照のみの利用となります。

変数名	説明
filename	ダウンロードするファイルの名前です。
path	ダウンロードするファイルの実体を示すパスです。"ファイル型項目名_jshfilename" の値になります。
modelId	対象モデルのIDです。
pkey	対象モデルの主キーです。
modelitemName	対象モデルの項目名です。
p.user.getUsername()	ログオンアカウントの名前です。

次のスクリプトは、すべての暗黙条件の値を表示するものです。

```
print("DownloadFile_beforeDownload variables.");
print("filename="+filename);
print("path="+path);
print("modelId="+modelId);
print("pkey="+pkey);
print("modelitemName="+modelitemName);
print("userid="+p.user.getUsername());
```

ダウンロード処理を中止する

スクリプトで何らかの文字列を return で返した場合、このダウンロード処理は中止されます。return で返された文字列はエラーメッセージとして画面に表示されます。

ファイル項目アップロード

リポジトリで定義したモデル内に存在するすべてのファイル型項目のアップロード時に共通で呼び出されるスクリプトです。この実体は WEB-INF/script/UpdateFile_afterUpload.js です。

利用できる暗黙変数

スクリプトでは次の暗黙変数を利用できます。各変数は参照のみの利用となります。

変数名	説明
filename	ダウンロードするファイルの名前です。
path	ダウンロードするファイルの実体を示すパスです。"ファイル型項目名_jshfilename" の値になります。
modelId	対象モデルのIDです。

次のスクリプトは、すべての暗黙条件の値を表示するものです。

```
print("UpdateFile_afterUpload")
print("filename="+filename);
print("path="+path);
print("modelId="+modelId);
```

アップロード処理を中止する

スクリプトで何らかの文字列を return で返した場合、このアップロード処理は中止されます。return で返された文字列はエラーメッセージとして画面に表示されます。

応用 ウィルスチェック

すべてのファイルのダウンロード時およびアップロード時に、ウィルスチェックを行うという要件を想定します。ウィルスチェックサービスはネットワーク上に REST API のサービスとして提供されているものとします。同サービスが不適切と判断した場合、エラーメッセージを表示してダウンロード（およびアップロード）を中止します。

ダウンロードスクリプト

次のようにします。

```
return checkFile(filename, path);
```

アップロードスクリプト

次のようにします。

```
return checkFile(filename, path);
```

共通関数

ダウンロード処理およびアップロード処理から呼び出される関数 checkFile を定義します。

```
function checkFile(filename, filepath) {
  if (filename === null || filepath === null
      || filename.length === 0 || filepath.length === 0) {
      return null;
  }

  var url = "http://localhost:18080/serv/check.php";//ダミーのサービスとして。
  var file = new java.io.File(filepath);
  if (! file.canRead()) {
    print("Can not read file "+file);
    return "ファイルの読み込みができません";
  }
  print("file size "+file.length());

  var client = new org.apache.commons.httpclient.HttpClient();
  var post = new org.apache.commons.httpclient.methods.PostMethod(url);
  var parts = [ new org.apache.commons.httpclient.methods.multipart.FilePart("file", filename,
 file) ];
  post.setRequestEntity(new
 org.apache.commons.httpclient.methods.multipart.MultipartRequestEntity(parts,
 post.getParams()));

  try {
    print("send file");
    var code = client.executeMethod(post);
    print("response code "+code);
    if (code >= 400) {
      return "不正なファイルです。";
    }
  } catch (e) {
    print("failed, file "+e);
    return "ファイルの送信に失敗しました。";
  }
  return null;
}
```

・　このスクリプトでは、ダミーのサービス(URL)に Apache Commons HttpClient を使ってファイルを（マルチパートリクエストで）送信しています。そのレスポンスコードが400以上なら不正なファイルとしてエラーとしています。

メールを送信する（スクリプト）

MailMessageマネージャを利用する

MailMessageManager とは、メール送信を行うユーティリティクラスです。これをスクリプトから使うことができます。

```
var replaceMap = new java.util.HashMap();
replaceMap.put("${message}", "Hello Wagby");
var mailman = p.appctx.getBean("MailMessageManager");
mailman.send("model1_update", replaceMap, errorManager, "Execute Script ", p);
```

・　1行目の replaceMap は Java 標準クラス java.util.HashMap のインスタンスです。メールテンプレートに用意されたプレースホルダを置換するためのルールを設定します。キー部分がプレースホルダ名で、値部分が（プレースホルダを置換する）値になります。
・　2行目でプレースホルダの値を設定しています。
・　3行目で MailMessageManager オブジェクトを取得しています。
・　4行目でメール送信を行っています。send メソッドの引数を以下に説明します。
・　上の例では用いていませんが、mailman.send の戻り値は boolean 型です。メール送信に成功した場合はtrueを返します。失敗した場合はfalesを返します。

MailMessageManager の send メソッドの引数

引数	説明
1	メールテンプレート識別子を指定します。
2	スクリプト中に用意したプレースホルダの置換マップを指定します。
3	エラーメッセージを扱うオブジェクトです。スクリプト中で暗黙に利用できます。このとおりに指定してください。
4	エラー発生時にログファイルに情報が出力されますが、そのときのログファイルのメッセージの先頭に埋め込まれる文字列（ログヘッダ）です。ログファイルを検索するときに用いることができます。任意の文字列を指定してください。ログファイルを見やすくするため、末尾に空白を一つ加えるとよいです。
5	pはスクリプト中で暗黙に利用できる特別なオブジェクトです。このとおりに指定してください。

MailMessageManager の戻り値

メール送信が成功した場合はtrueが返ります。失敗した場合はfalseが返ります。

ログ出力（スクリプト）

ログ出力

Wagby は標準で log4j というログライブラリを利用しています。次のようなスクリプトを作成することができます。

```
var Logger = Java.type("org.apache.log4j.Logger");
var SCRIPTCODERUNNER = "ServerSide Script";
```

```
var _logger = Logger.getLogger(SCRIPTCODERUNNER);
_logger.info("log output test");
```

- 1行目の Logger は log4j の Logger クラスを表します。
- 2行目の SCRRIPTCODERUNNER は定数としています。名前や内容は任意です。3行目の getLogger メソッドの引数とします。
- 4行目でログ出力を行なっています。info,warn,error,fatal,debug という5つのレベルに対応したメソッドを利用することができます。

ログは system.log ファイルに出力されます。

```
yyyy-MM-dd hh:mm:ss [INFO ServerSide Script process] log output test
```

出力元を特定させる

どのクラスによるログ出力かを特定させるために、クラス名を使うこともできます。例を示します。

```
var UpdateCustomerController =
  Java.type("jp.jasminesoft.wagby.controller.customer.UpdateCustomerController");
var Logger = Java.type("org.apache.log4j.Logger");
var _logger = Logger.getLogger(UpdateCustomerController.class);
_logger.info("log output test");
```

サンプルコード（スクリプト）

繰り返しコンテナ項目の日付を使って、新規データを生成する

要件

モデル model1 に、繰り返しコンテナ cont があります。

model1 には「予定日 (yoteibi)」「タイトル (title)」項目があります。 また、model1 の繰り返しコンテナ cont には「次回予定日 (jikaiyoteibi)」と「タイトル (ctitle)」があります。

項目名	項目ID	型
予定日	yoteibi	日付
タイトル	title	文字列
議事録	cont	繰り返しコンテナ
No	cont/cid	繰り返しコンテナID
次回予定日	cont/jikaiyoteibi	日付
次回タイトル	cont/ctitle	文字列

(349)

繰り返しコンテナ項目内の「次回予定日」に日付が入っていた場合に、予定日およびタイトルに繰り返しコンテナ内の値を設定した *新規オブジェクト* を登録します。

（つまり、model1 データを1件登録したときに、繰り返しコンテナが1件あれば、同時にもう1件のデータも登録されるというものです。）

スクリプトの記述箇所

model1 の新規登録処理が成功したあと、とします。具体的には「コントローラ > 登録(画面) > 登録の実行(タイミング)」に記述します。

```
var Model1Class = Java.type("jp.jasminesoft.wagby.model.model1.Model1");
var service = p.appctx.getBean("Model1EntityService");
var helper = p.appctx.getBean("Model1Helper");
var cont = model1.cont;
```

(349) 「タイトル」と「次回タイトル」は型が同じ、とします。上では文字列としていますが、モデル参照方式にすることもできます。（タイトルマスタというモデルがあり、選択式とする、など。）

```
if (cont != null) {
    for (var i=0; i<cont.length; i++) {
        if (cont[i].jikaiyoteibi != null) {
            var n_model1 = new Model1Class();/*オブジェクト作成*/
            helper.initialize(n_model1, p);/*オブジェクト初期化*/
            n_model1.yoteibi = cont[i].jikaiyoteibi;
            n_model1.title = cont[i].ctitle;
            /*print(n_model1);*/
            service.insert(n_model1);
        }
    }
}
```

- 1,2,3行目はそれぞれストアモデル Model1、(Model1の)サービスクラス、(Model1の)ヘルパクラスを利用するためのコードです。
- 7行目で、繰り返しコンテナ項目内の「次回予定日」に日付が入っているかどうかの判断を行っています。
- 8行目で新規オブジェクトを生成しています。
- 9行目でオブジェクトの初期化処理を行っています。詳細...
- 10,11行目で繰り返しコンテナの値をセットしています。
- 13行目で新規オブジェクトをデータベースに登録しています。この例では主キーは順序を利用しています。（自動採番となるため、service.insert メソッド内で主キーが割り当てられます。）

繰り返しコンテナで、ある値が空のコンテナ行を削除する

要件

アドオンギャラリーで用意した「日報」モデル JSHDAILYREPORT に、繰り返しコンテナ DRCOMMENT があります。[詳細...]

この DRCOMMENT の CMEMO 項目が空の場合、当該行を削除します。

ヘルパの登録ならびに更新処理にスクリプトを設定します。

```
var DRCOMMENT = JSHDAILYREPORT.DRCOMMENT;
if (DRCOMMENT !== null) {
    var removeList = new java.util.ArrayList();
    for (var i=0; i<DRCOMMENT.length; i++) {
        if (ExcelFunction.ISBLANK(DRCOMMENT[i].CMEMO)) {
            removeList.add(i);//CMEMOが空の行の位置を記録する。
        }
    }
    if (removeList.size() > 0) {
        for (var i=removeList.size(); i>0; --i) {
            var idx = removeList.get(i-1);
            JSHDAILYREPORT.removeDRCOMMENT(idx);// うしろから削除していく。
        }
        DRCOMMENT = JSHDAILYREPORT.DRCOMMENT;
        for (var i=0;i<DRCOMMENT.length; i++) {
            // 繰り返しコンテナID (CID) および内部インデックス (DRCOMMENTjshid) を
            // 改めて採番する。これらは飛び番号が認められていないため。
            DRCOMMENT[i].CID = i;
            DRCOMMENT[i].DRCOMMENTjshid = i;
        }
    }
}
```

参照連動処理を呼び出す

Wagbyの参照連動は、連動元の項目が計算式のとき、計算を行った結果の値を使って参照連動を行います。しかし開発者が直接、計算タイミングのスクリプトで値を算出した場合は、参照連動処理とは連動しません。この場合、開発者は手動で参照連動処理の呼び出しを行うようにしてください。

例を示します。スクリプトの「ヘルパ > 計算」のタイミングで model1 モデルの item1 項目に値をセットしたとします。この item1 が参照連動の連動元項目の場合、次のコードで参照連動処理を呼び出すことができます。

```
//(item1に値をセットする)

p.appctx.getBean("Model1Helper").resolvRelatedItem(model1, p);
```

外部コマンドを実行する

ExcelFunction.EXEC を使って外部コマンドを実行することができます。

```
var ret = ExcelFunction.EXEC("../..", "cmd", "/C", "dir");
print(ret);
```

オリジナルボタンから外部コマンドを実行する例を用意しています。

EXEC関数の詳細は"関数 > その他 > EXEC"をお読みください。

スクリプトの再読み込み処理を制御する

起動時のパラメータオプション

Wagby のスクリプトは実行時に毎回、スクリプトファイル (WEB-INF/script 以下のファイル) の日付をチェックし、更新があった場合は再読み込みを行うようになっています。これによって Wagby アプリケーションを再起動することなく、スクリプトファイルを書き換えることができます。

この運用ルールを変更し、Wagby アプリケーション起動時に一度だけスクリプトファイルを読み込むようにすることができます。ファイルの日付チェックをスキップすることで実行パフォーマンスが向上します。運用開始後、スクリプトファイルの変更を行わないという場合に有効です。

起動時のパラメータ

Windows OS の場合

wagbyapp/bin/setenv.batに下記を指定します。

```
set CATALINA_OPTS=%CATALINA_OPTS% -DScriptCodeRunner.reloadable=false
```

Unix 系 OS の場合

wagbyapp/bin/setenv.shに下記を指定します。

```
CATALINA_OPTS=$CATALINA_OPTS¥ -DScriptCodeRunner.reloadable=false
```

仕様・制約

暗黙オブジェクトrequestやsessionへ値を保存する場合の注意点

暗黙オブジェクト request や session には Wagby 本体がさまざまな値をセットしています。例えば主キーの値や、ストアモデル、プレゼンテーションモデル、各種制御変数があります。

開発者は request.getAttribute(キー名) や session.getAttribute(キー名) で値を参照することができます。参照することは問題ありません。

ただし request.setAttribute(キー名) や session.setAttribute(キー名) で値を保存する場合、キー名がすでに Wagby によって使われていないかどうかを考慮する必要があります。誤って同じキー名で値を上書きしてしまうと、予想外の動作を引き起こします。

この対策として、開発者が内部で利用するキー名に prefix や suffix を付与するルールを設けることを推奨します。例えば "local.customer" や "pkey.myapp" など、"local." や ".myapp" といった特定の文字列を付与するようにします。このようにすることで Wagby が内部で利用する値と重複しないため安全になります。

ポータル

ポータル機能を使う

ポータル機能を有効にする

システム管理者でWagbyアプリケーションにログオンします。管理処理タブから「ポータル管理...」を選択します。

図 ポータル管理を行う

はじめに「ポータル設定」を開きます。

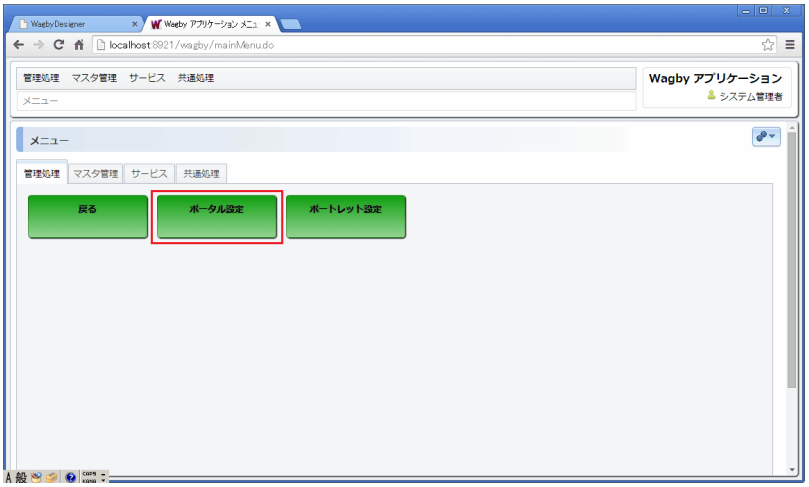


図 ポータルの設定

更新ボタンを押下し、設定画面を開きます。

図 ポータル設定画面へ

「ポータル利用」を有効にします。その他の設定は標準値をそのまま使います。

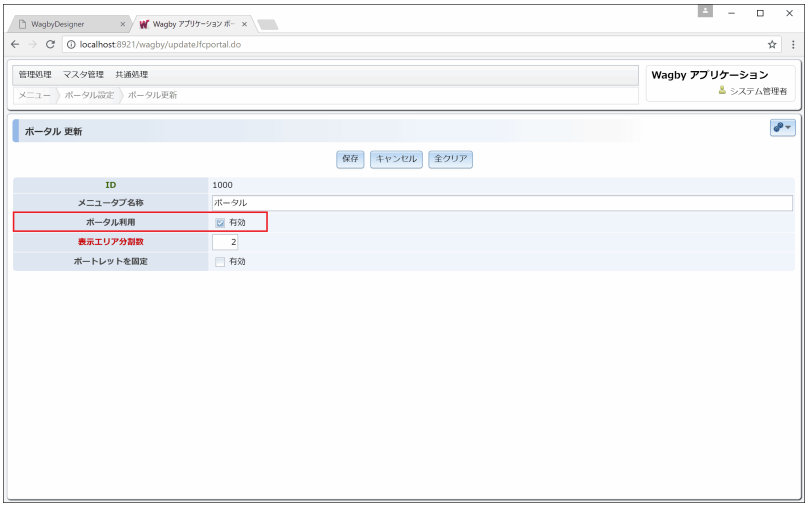


図 ポータル機能を有効にする

利用するポートレットを選択する

ポータル機能を有効にするとポータルタブが用意されます。続いて、ポータルタブ内に表示する「ポートレット」（ポータル画面に表示される部品）を選択します。

メニューから「ポートレット設定」を選びます。

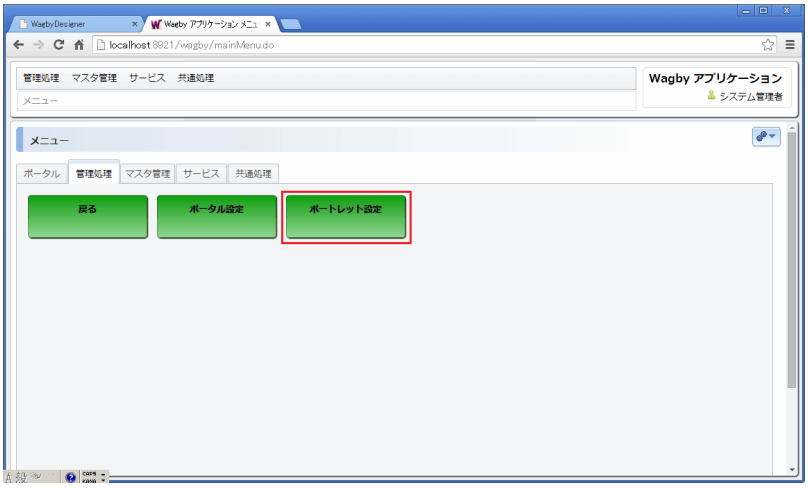


図 ポートレット設定

利用できるポートレットが表示されます。　今回は「カレンダ」「ログオンユーザー覧」「お知らせ」の3つのポートレットを有効にします。

図の画面から「一覧更新」ボタンを押下します。

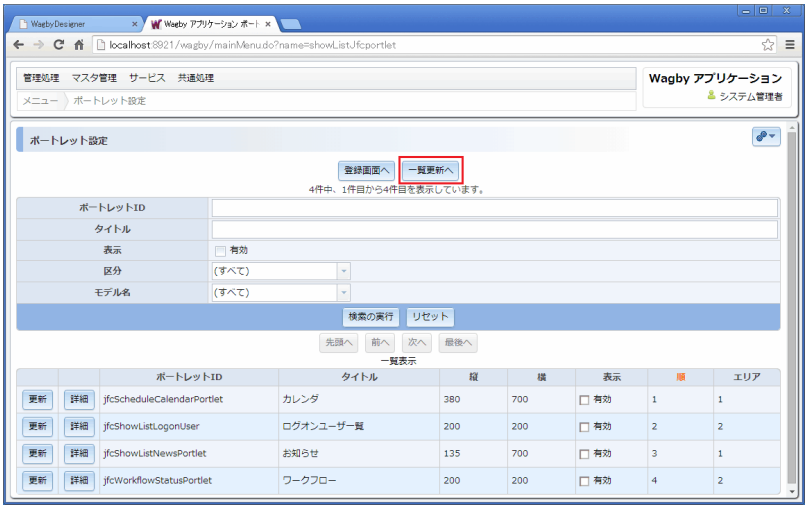


図 ポートレット一覧

一覧更新画面で、それぞれのポートレットを有効にします。（その他の設定欄は標準値を使うとします。）

図 ポートレットを有効にする

3つのポートレットをそれぞれ有効にした例を図に示します。

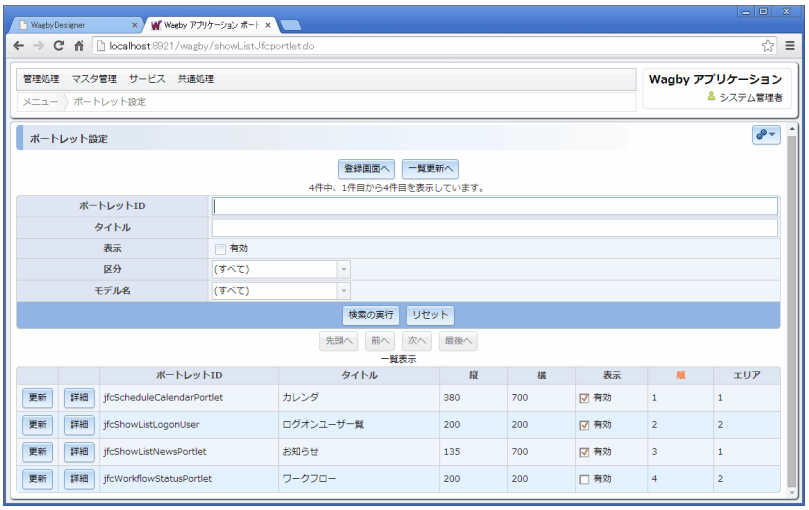


図 3つのポートレットを有効にした

ここまでの設定により、メニューに「ポータル」タブが追加されます。また、3つのポートレットが表示されるようになります。

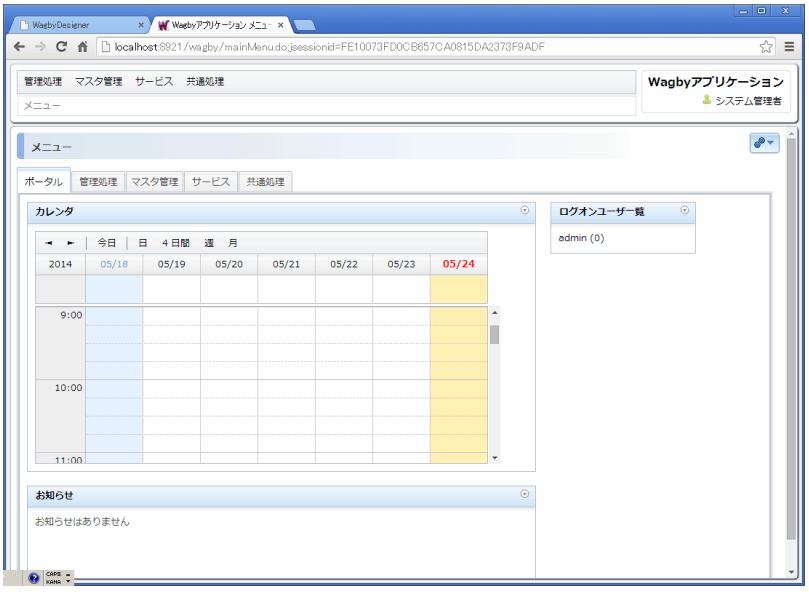


図 ポートレットが表示された

表示エリアと表示順

ポートレットには「表示エリア」という設定があります。表示エリアはポータル設定画面の「表示エリア分割数」で ″2″ または ″3″ を指定できます。表示エリア分割数が ″2″ のとき、図のようになります。各ポートレットの表示エリア指定に基づき、最初の位置が確定します。

さらにポートレットには「表示順」という設定があります。同じ表示エリアでは、表示順の若い方が上に配置されます。

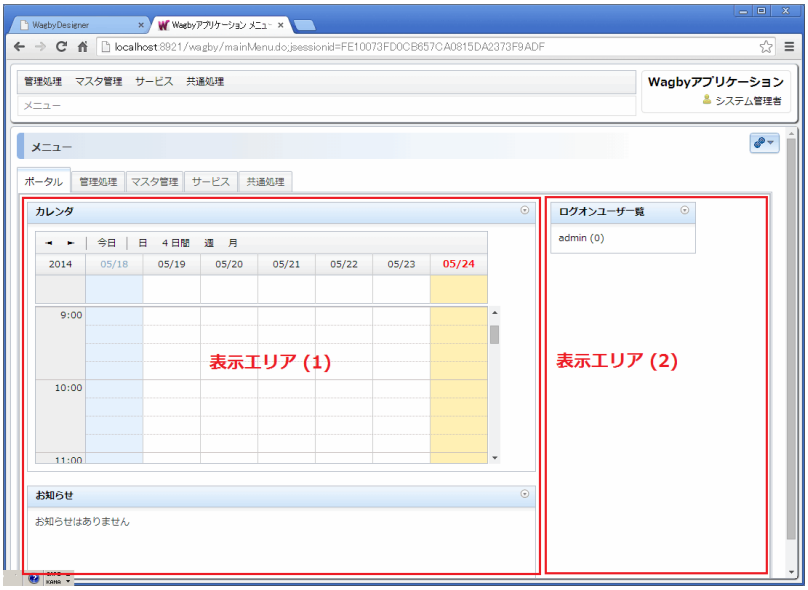


図 ポートレットの表示エリア

ドラッグ&ドロップでポートレットの配置を変える

ポートレットの配置位置をドラッグ&ドロップで変更することができます。

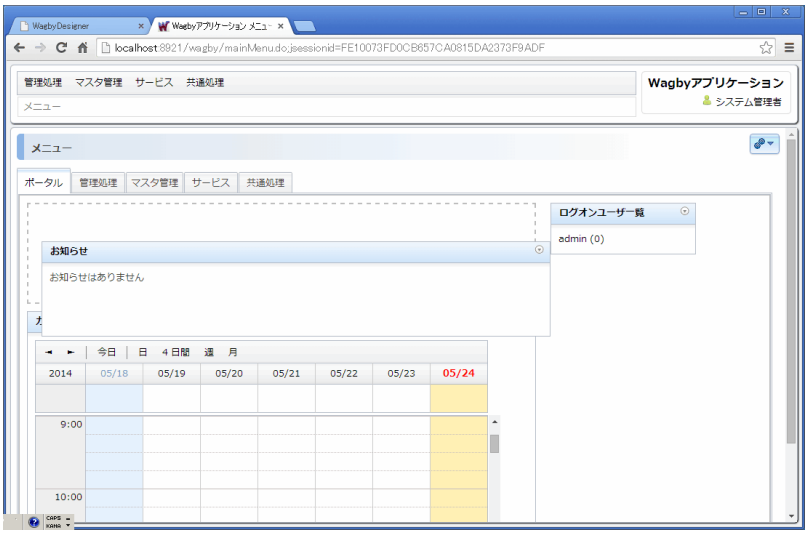


図 ポートレットの配置位置の変更（ドラッグ）

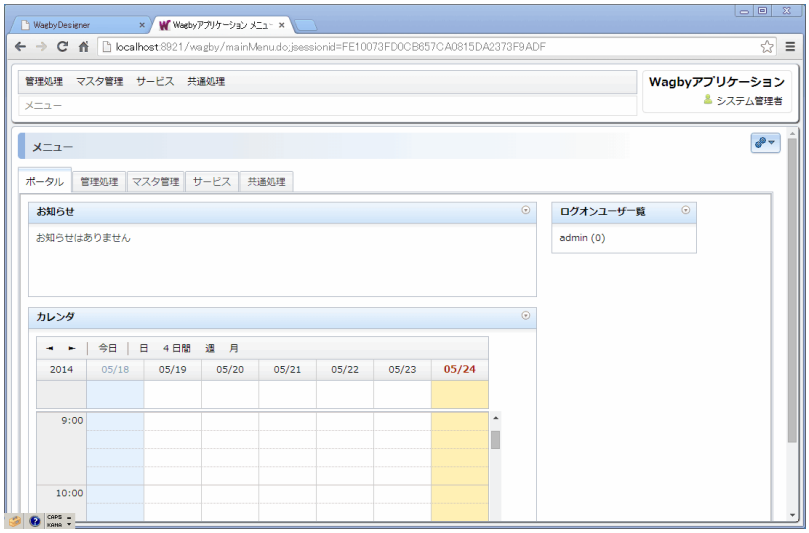


図 ポートレットの配置位置の変更（ドロップ）

配置変更後に再度、ログオンすると、最後に変更された配置が有効になります。

ポートレットにはサイズ（縦、横）が規定されています。ドラッグ&ドロップしても、このサイズは維持されます。

ポートレットの位置を固定にする

アプリケーションの利用者が、ポートレットの移動を行えないように設定することもできます。「ポートレットを固定」の設定を有効にしてください。ドラッグ&ドロップでポートレットの配置を変えることができなくなります。

(350)

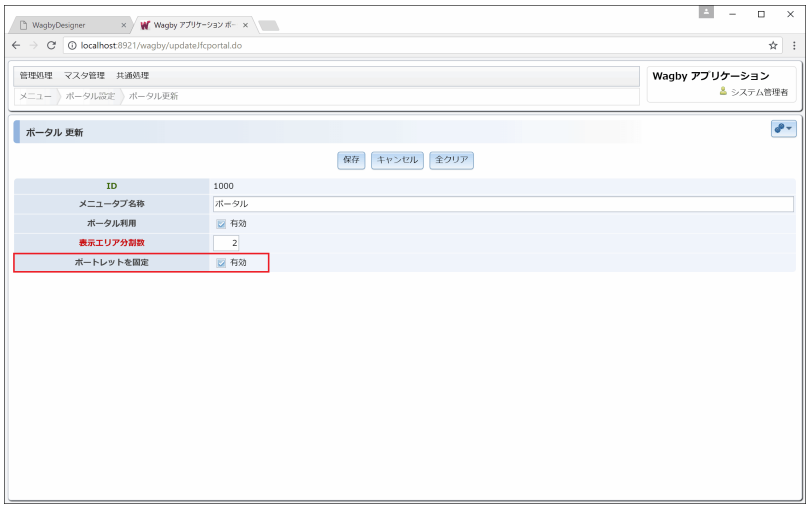


図 ポートレットを固定にする

システム管理者以外のアカウントでログオンする

システム管理者がポータルの設定を行うことで、一般ユーザもポータルを利用できるようになります。

アカウントyamadaは別のブラウザを使っています。ポータルの設定は一部、ブラウザのクッキーに記憶させているため、同一のPCでアカウントを切り替えて試験する場合はブラウザを変えると良いです。

(350) この設定はすべての利用者で有効になります。

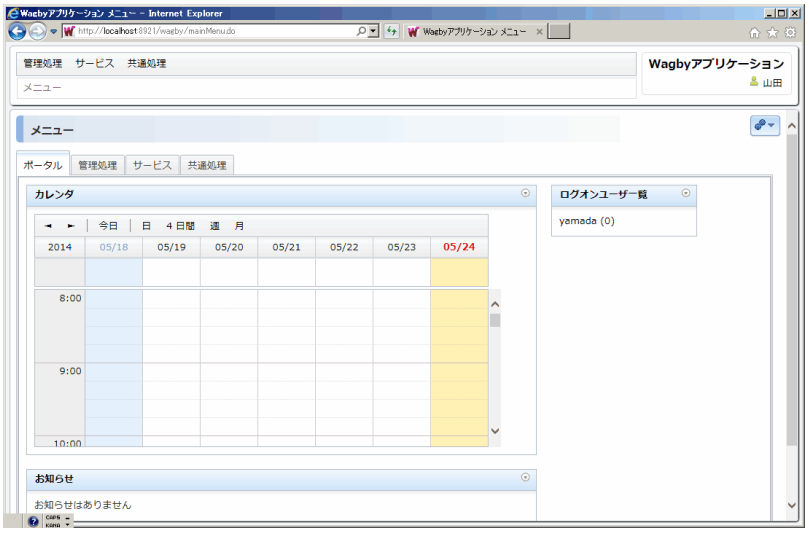


図 アカウントyamadaでログオンした例

メニュータブとの関係

Wagbyでは、ログオン後はメニューの一番左側のタブが選択される（表示される）という仕様になっています。

ポータルのタブは一番左に配置されるよう固定されています。そのためポータルが有効の場合は、ログオン後、必ずポータルが表示されるようになります。

ポータルの表示エリアの高さを変更する

ポータルはメニューのタブに含まれているため、メニューのタブの高さを変更すると、ポータルの表示エリアの高さを変更できます。詳細は"レイアウト > JavaScriptのカスタマイズ > メニュー表示エリアの高さを変更する" をお読みください。

ポータルのタブ名を変更する

図のポータル設定画面で「メニュータブ名称」を変更することができます。

この名称は国際化対応のため、次の書式で管理されています。

```
{"en":"Portal","ja":"ポータル"}
```

文字列 "ポータル" の部分を変更してください。この書式の詳細はメッセージの変更・国際化 > 選択肢モデルの特別表記をお読みください。

トラブルシューティング

ポートレット一覧が表示されない

図に示す、ポートレット一覧が表示されない場合、ポートレットに関する初期データが空となっている可能性があります。

以下の手順でポートレットの初期データのみを再インポートし、ポートレットのデータが表示されるかを確認ください。

1. システム管理者でログオン後、インポートとエクスポート画面を開きます。
2. インポートフォルダ名で「data_init」を選択します。
3. システムによって提供されるモデル一覧から「ポータル > ポートレット > ポートレットタイプマスタ」を選択して、「インポート」ボタンを押下します。

ポートレットのタイトルが空白になっている

ご利用のブラウザの言語情報が適切に設定されるかどうかをご確認ください。日本語環境の場合、日本語が選択されている必要があります。

お知らせポートレット

お知らせの登録

お知らせ情報の登録は「システム管理者」または「お知らせ管理者」のプリンシパルをもったアカウントで行うことができます。

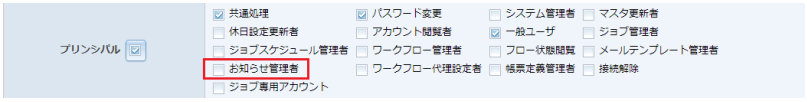


図 お知らせ管理者プリンシパル

図 お知らせ検索メニュー

新しいお知らせ情報を一件、登録します。

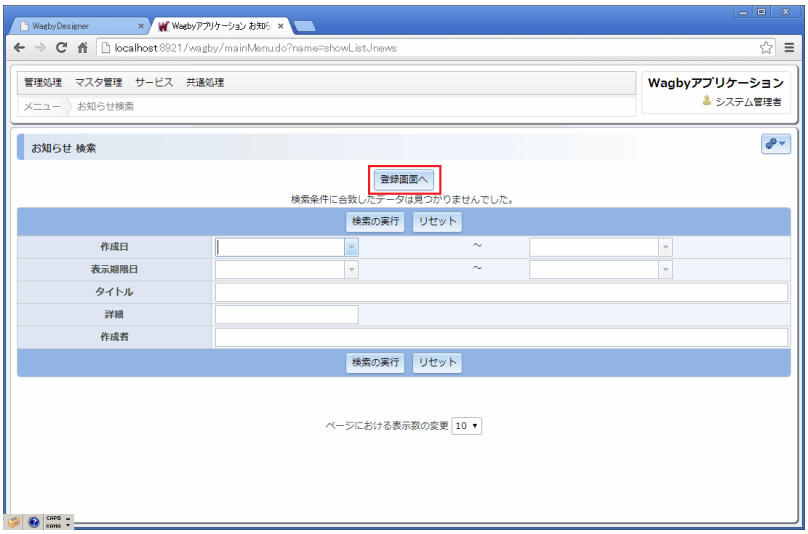


図 お知らせ情報の登録画面を開く

お知らせ情報は表示期限を設定する必要があります。この日付になるまで、お知らせが表示されます。

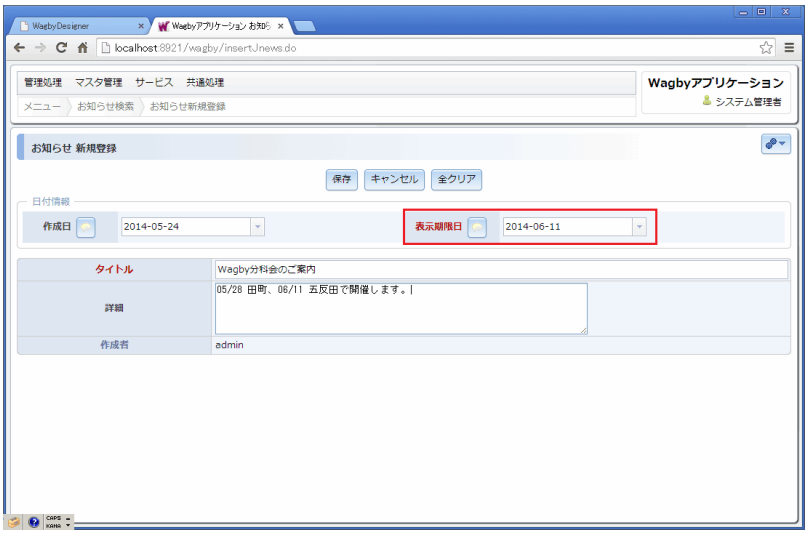


図 お知らせ情報を登録する

お知らせポートレットの表示

「お知らせ」のポートレットに、登録した情報が表示されます。

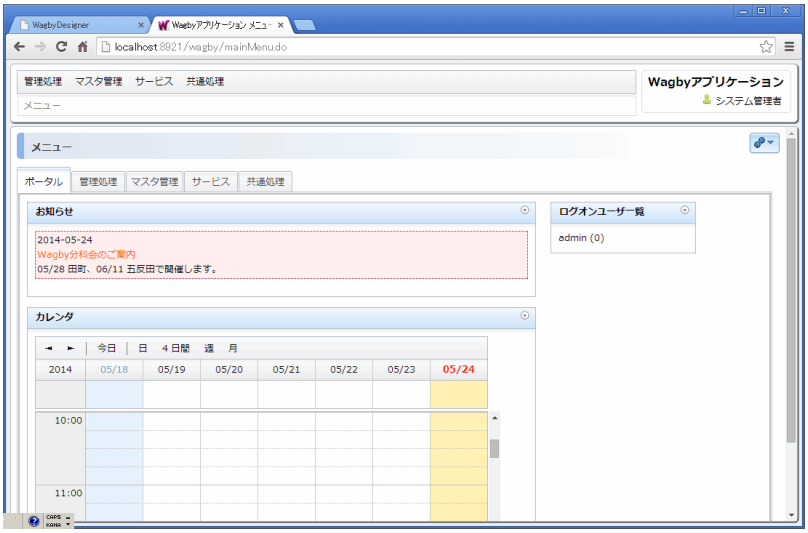


図 お知らせ情報がポートレットに表示されている

お知らせポートレット内のタイトル部をクリックすると、お知らせの詳細画面へ遷移します。

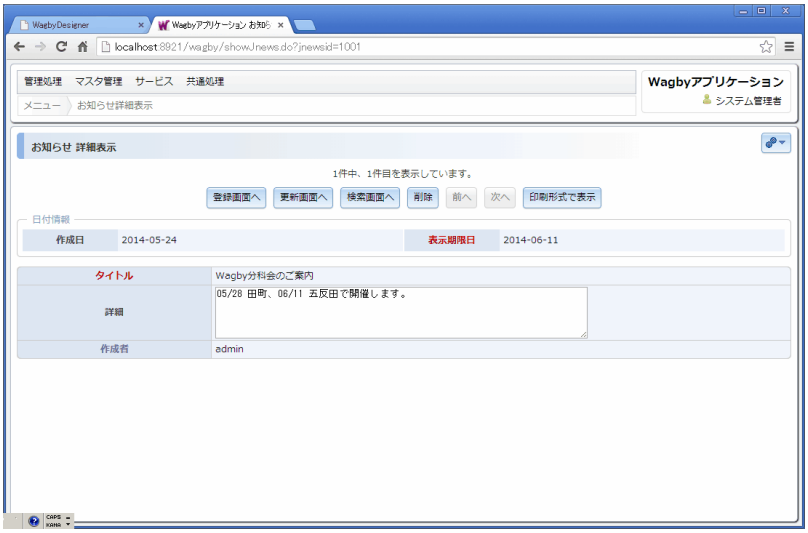


図 お知らせ情報の詳細表示

「表示期限日」を過ぎた〝お知らせ〟は表示されません。

高さの調整

ポートレットは自身の表示幅と高さの情報をもっています。ここでは高さを変更して、2件のお知らせを表示できるように調整します。

例

2件のお知らせ情報が掲載されるようになった例を示します。

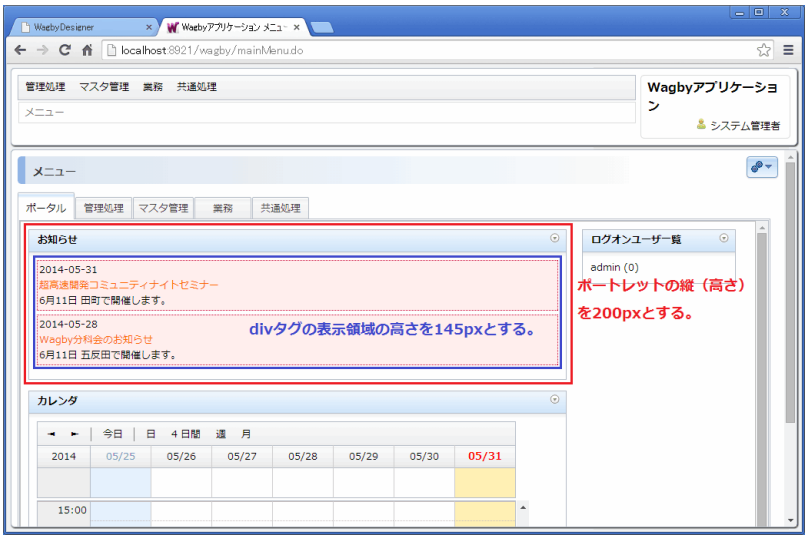


図 お知らせポートレットの縦をひろげた例

設定

お知らせポートレットの更新画面を開きます。ポートレット本体の縦サイズ（高さ）を200ピクセルへ変更します。

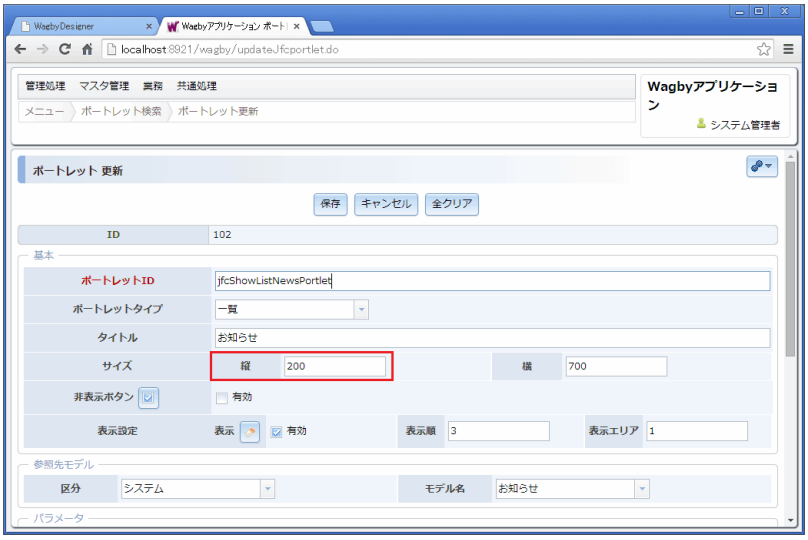


図 お知らせポートレットの縦サイズ（高さ）を変更する

お知らせポートレットは、ポートレット内にも表示領域としての div 要素を持っています。その高さ設定も145ピクセルへ変更します。

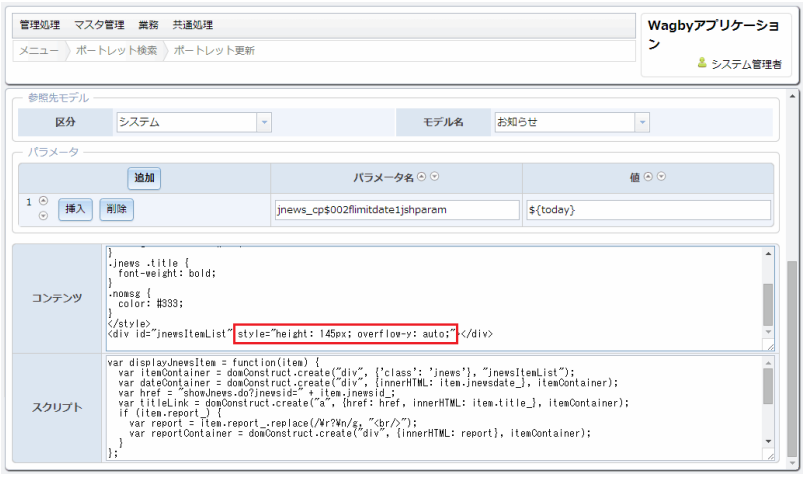


図 お知らせポートレット内のdiv要素の高さを変更する

検索パラメータ

検索条件パラメータとして、次の設定を行うことができます。

パラメータ名	値	説明
jnews_cp $002flimitdate1jshparam	${today}	お知らせモデルの limitdate 項目は、表示期限日が設定されるようになっており、かつ検索項目として定義されています。この設定により、本日以降の表示期限となっているお知らせだけを表示できるようになります。
jnews_cp$002fcreateuser	${userid}	お知らせモデルの createuser 項目は、標準でログオンIDが設定されるようになっており、かつ検索項目として定義されています。この設定により、お知らせを作成したアカ

		ウントのみに表示されるようになります。

(351)

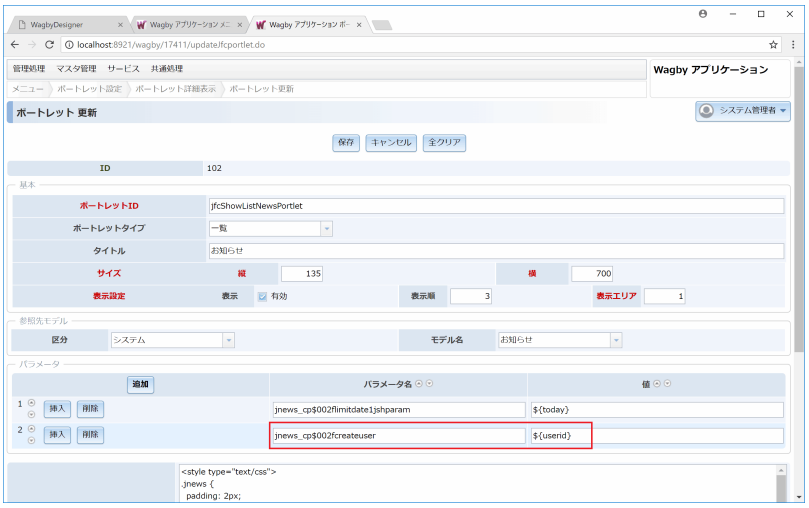


図 ログオンIDで絞り込む設定を追加した

見栄えの変更

お知らせポートレットは「コンテンツ」と「スクリプト」から構成されています。コンテンツには CSS のスタイル定義と HTML を記述します。「スクリプト」には JavaScript を記述します。

お知らせポートレットは JavaScript で、（REST API 経由で取得した）お知らせ情報を HTML に変換し、これをコンテンツ部に用意した div 要素（id属性が "jnewsItemList"）へ埋め込むという動作になっています。

ここではコンテンツ部のCSSと、スクリプトを一部変更し、図のように表示されるようにしてみます。

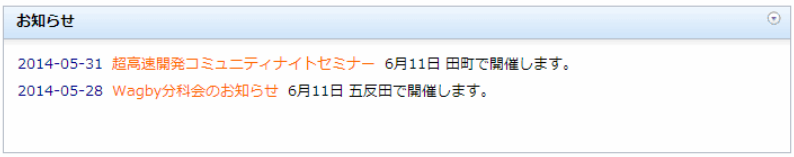


図 お知らせポートレットの表示スタイルの変更

コンテンツ

jnewsクラスから枠線の指定を除去しました。新たにdateinfoクラスを定義し、青色で右に余白を備えるようにしました。

```
<style type="text/css">
.jnews {
  padding: 2px;
}
.jnews .title {
  font-weight: bold;/* borderの設定を除去 */
}
.dateinfo {/* dateinfoを追加 */
  color: #008;
  padding-right:8px;
```

(351) jnews_cp はお知らせモデルの検索条件であることを示します。limitdate の接尾語 "1jshparam" は検索範囲の開始を示します。詳細は「コンディションモデル」の説明をお読みください。

```
}
.nomsg {
  color: #333;
}
</style>
<div id="jnewsItemList" style="height: 80px; overflow-y: auto;"></div>
```

スクリプト

div要素からspan要素へ変更し、新たに追加したCSSクラスを指定しました。（またはstyle属性を直接、指定しています。）

```
var displayJnewsItem = function(item) {
  var itemContainer = domConstruct.create("div", {'class': 'jnews'}, "jnewsItemList");
  var dateContainer = domConstruct.create("span", {'class':'dateinfo', innerHTML:
 item.jnewsdate_}, itemContainer);
  var href = "showJnews.do?jnewsid=" + item.jnewsid_;
  var titleLink = domConstruct.create("a", {href: href, innerHTML: item.title_}, itemContainer);
  if (item.report_) {
    var report = item.report_.replace(/¥r?¥n/g, "<br/>");
    var reportContainer = domConstruct.create("span", {'style':'padding-left:8px;', innerHTML:
 report}, itemContainer);
  }
};

if (data.entity.length > 0) {
  array.forEach(data.entity, function(item) {
    displayJnewsItem(item);
  });
} else {
  domConstruct.create("div", {'class': 'nomsg', innerHTML: 'お知らせはありません'},
 "jnewsItemList");
}
```

上のコードにある domConstruct は、Dojotoolkit が提供する DOM 操作のためのモジュール dojo/dom-construct を使っています。ポートレットタイプが「一覧」の場合、すぐに利用することができるようにしています。dojo/dom-construct の使い方の詳細は、Dojotoolkit のマニュアルをお読みください。

カレンダポートレット（クラシック）

カレンダポートレットの設定

ポートレット一覧から、カレンダポートレットの設定画面を開きます。"jfcScheduleCalendarPortlet" を選択します。

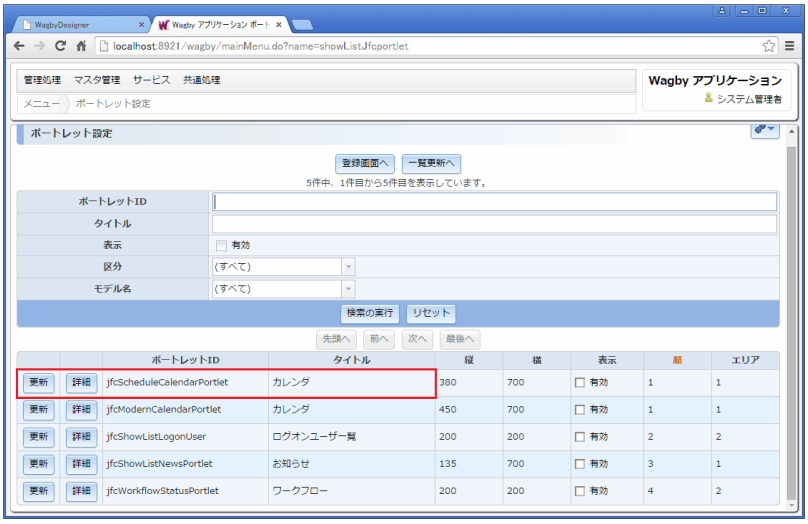


図 カレンダポートレットの設定を行う

タイトルを変更します。また参照先モデルに、カレンダビュー機能を設定済みのモデル（ここでは「スケジュール」モデル）を指定します。

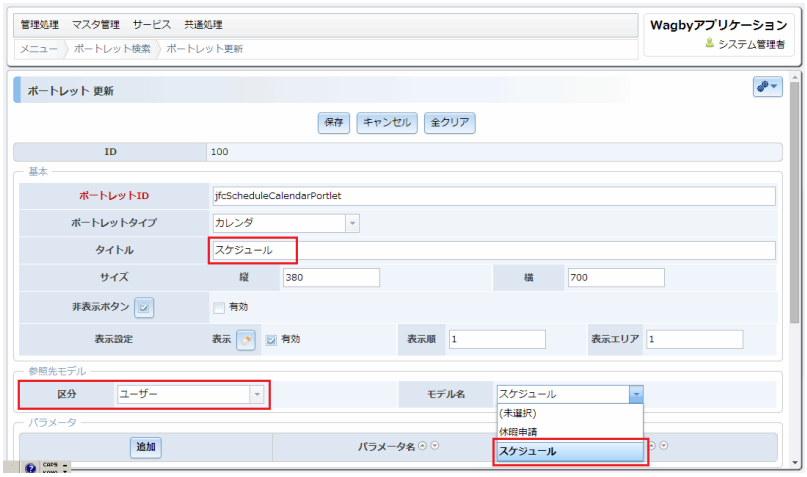


図 対象モデルを指定する

カレンダポートレットの表示

上で設定した「カレンダ」モデルの情報が、ポートレットに表示されます。図は週単位の表示の例です。

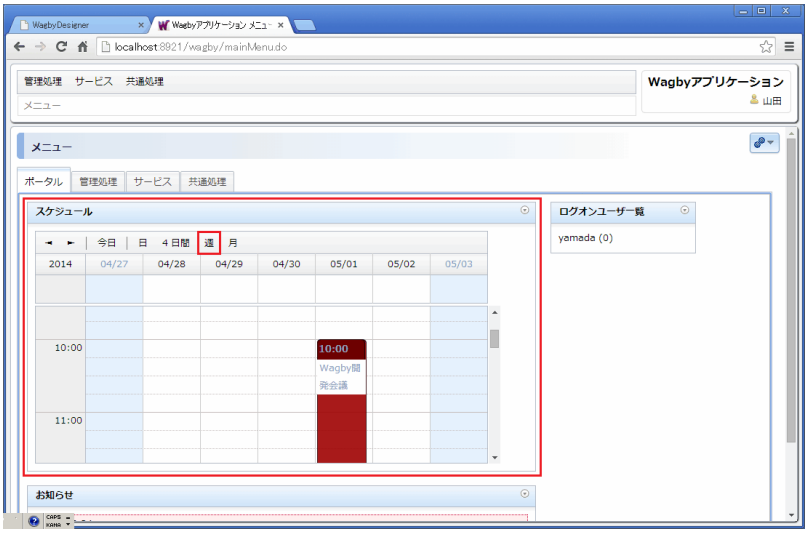


図 カレンダポートレットの表示（週単位）

「日」「4日間」「月」をそれぞれ選択すると、表示方法を変更することができます。

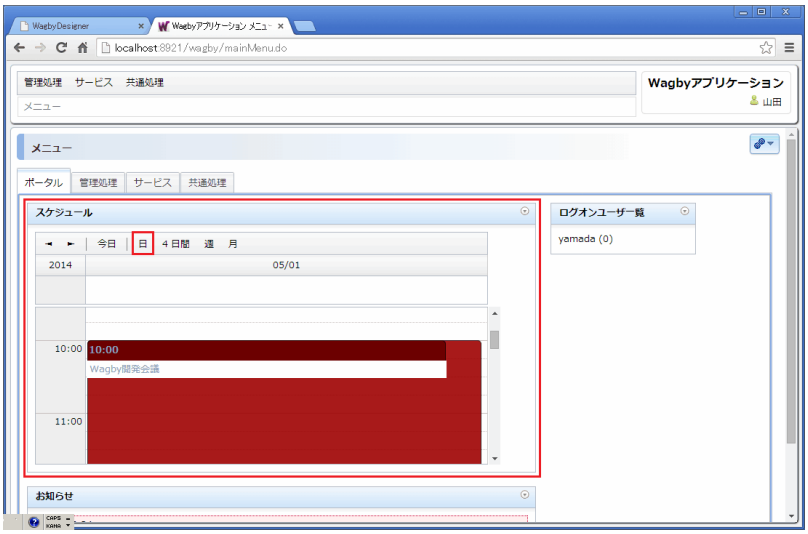


図 日単位の表示

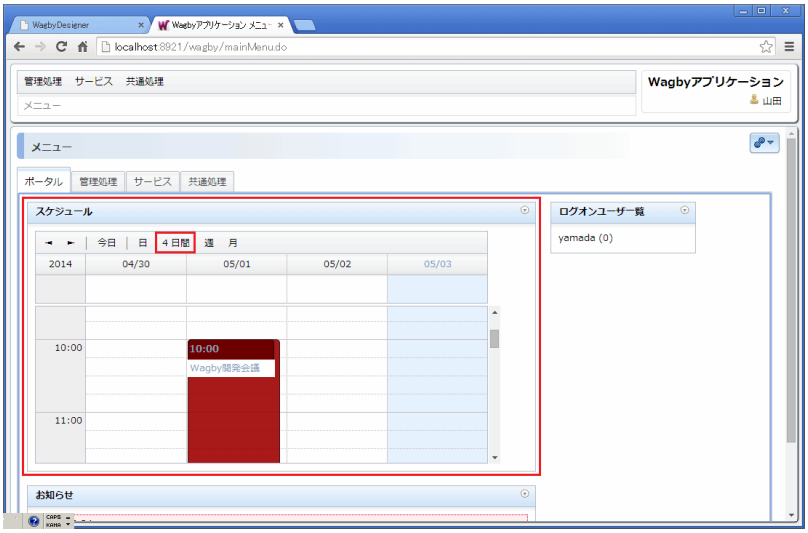


図 4日単位の表示

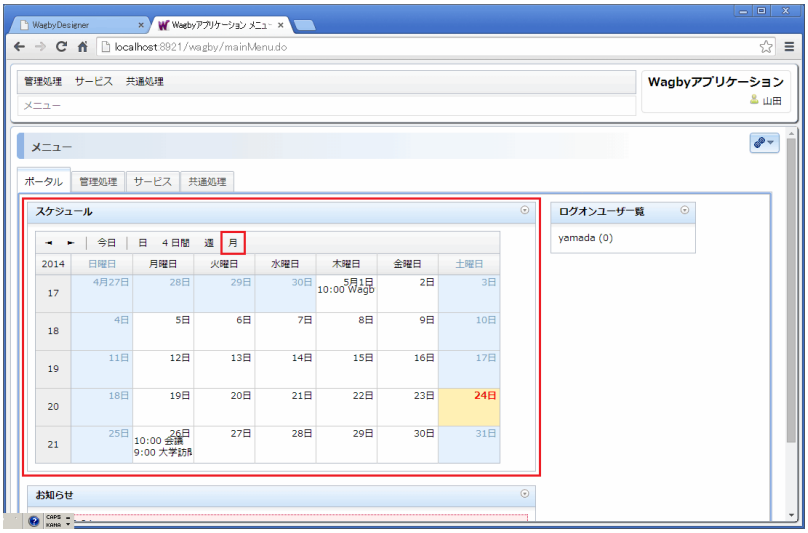


図 月単位の表示

カスタマイズ設定

カレンダポートレットの「スクリプト」に、初期値設定のためのコードが用意されています。これを編集することでカスタマイズできます。

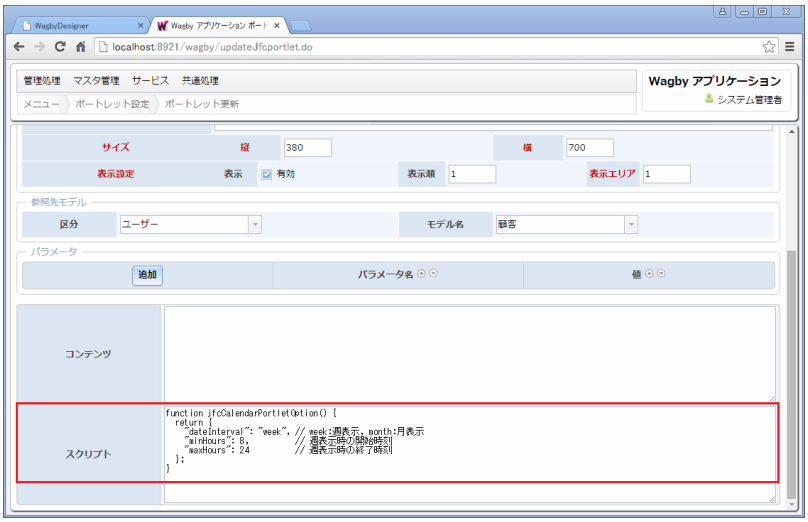


図 カレンダポートレットのスクリプト欄

提供されるスクリプトは次のとおりです。初期パラメータをJSON形式で（カレンダポートレットへ）提供します。

```
function jfcCalendarPortletOption() {
  return {
    "dateInterval": "week", // week:週表示, month:月表示
    "minHours": 8,          // 週表示時の開始時刻
    "maxHours": 24          // 週表示時の終了時刻
  };
}
```

例えば次のように修正することで、初期表示を「月」とすることができます。また週表示に切り替えたときの開始時刻を1時から23時までとすることができます。

```
function jfcCalendarPortletOption() {
  return {
    "dateInterval": "month", // week:週表示, month:月表示
```

```
    "minHours": 1,          // 週表示時の開始時刻
    "maxHours": 23          // 週表示時の終了時刻
  };
}
```

ビルドは不要です。修正後、すぐにポータル画面に反映されます。

仕様・制約

・　カレンダポートレットでは、表示されているデータを変更することはできません。閲覧のみとなります。

カレンダポートレット（モダン）

カレンダポートレットの設定

ポートレット一覧から、カレンダポートレットの設定画面を開きます。"jfcModernCalendarPortlet" を選択します。

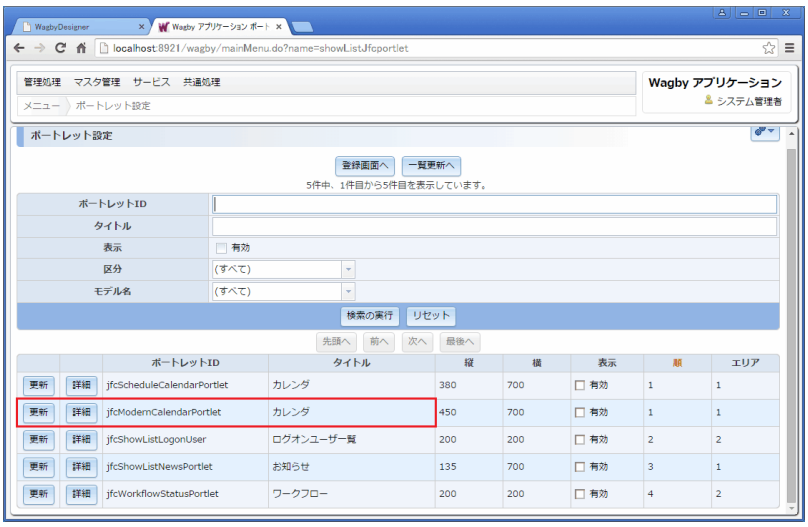


図 カレンダポートレットの設定を行う

タイトルを変更します。また参照先モデルに、カレンダビュー機能を設定済みのモデル（ここでは「スケジュール」モデル）を指定します。

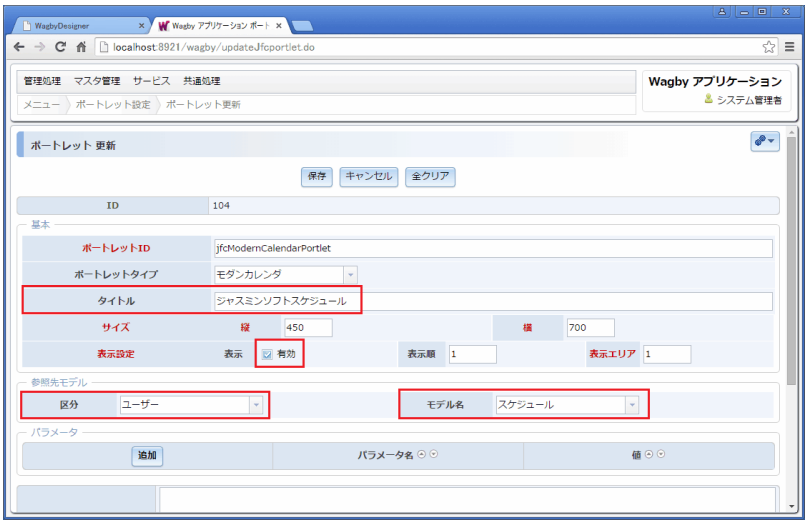


図 対象モデルを指定する

カレンダポートレットの表示

上で設定した「カレンダ」モデルの情報が、ポートレットに表示されます。図は週単位の表示の例です。

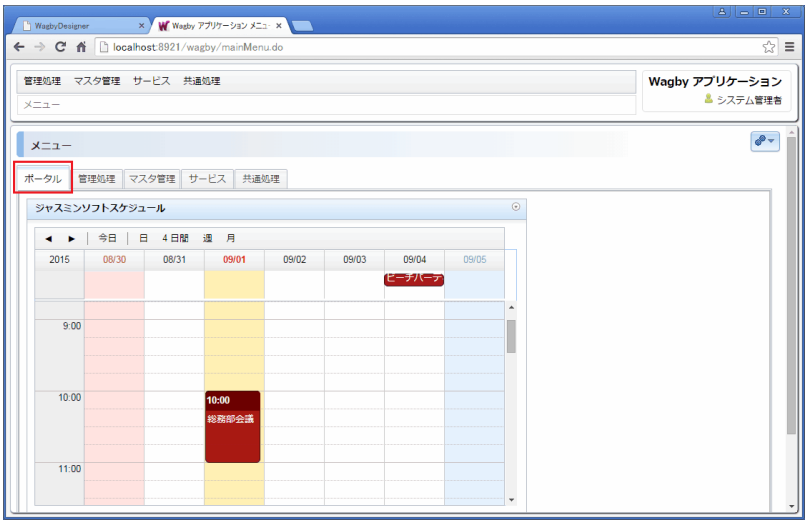


図 モダンカレンダポートレットの表示（週単位）

イベントをダブルクリックすると、ダイアログが表示されます。

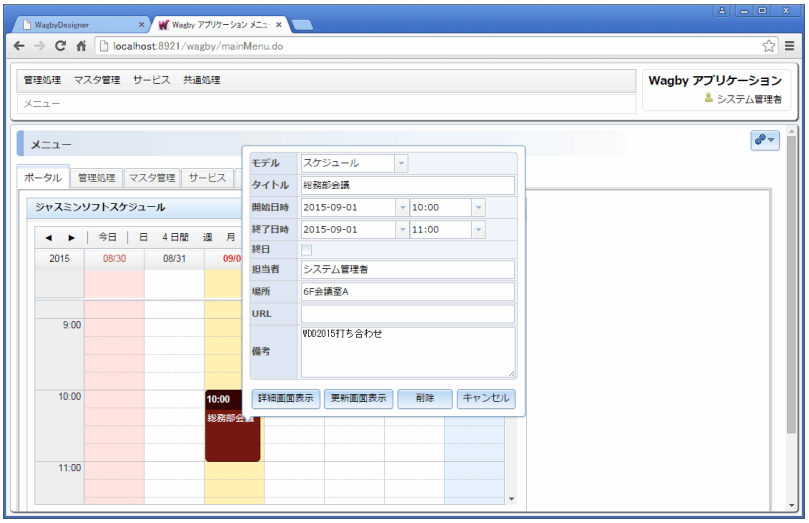


図 詳細ダイアログ

カスタマイズ設定

カレンダポートレットの「スクリプト」に、初期値設定のためのコードが用意されています。これを編集することでカスタマイズできます。

詳細は"カレンダポートレット（クラシック）> カスタマイズ設定"をお読みください。

ビルドは不要です。修正後、すぐにポータル画面に反映されます。

仕様・制約

- カレンダポートレットでは、表示されているデータを変更することはできません。閲覧のみとなります。

ログオンユーザー覧ポートレット

ログオンユーザー覧ポートレットの設定

ポートレット一覧から、ログオンユーザー覧ポートレットの設定画面を開きます。　表示を「有効」として、保存します。

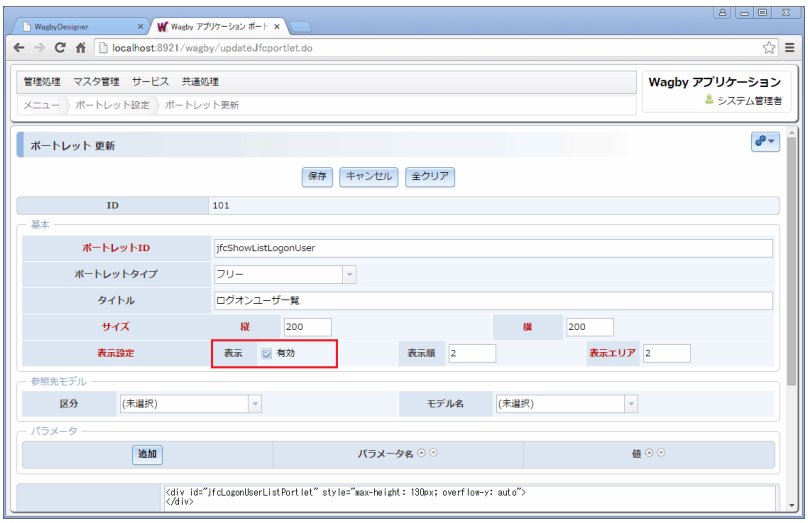


図　ログオンユーザー覧ポートレットを有効にする

ログオンユーザー覧ポートレットの表示

ログオンユーザー覧情報が、ポートレットに表示されます。

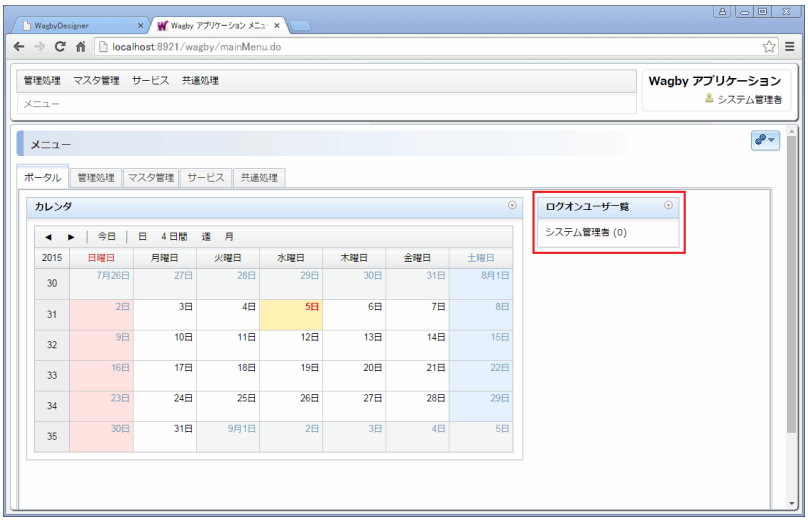


図　ログオンユーザー覧ポートレットの表示

- 標準ではアカウント(juser)モデルの「名前(name)」項目が表示されます。名前が空白の場合は「アカウント(userid)」が表示されます。
- 名前の横のかっこ内の数字は、最後にシステムを操作してからの経過時間（分）です。この利用者のアイドル時間（何も使用していない時間）を知ることができます。アイドル時間が規定のセッションタイムアウト時間を過ぎると、この利用者は自動的にログアウトされます。
- 並び順は「このシステムにアクセスした順（言い換えると、もっともアイドル時間が短い順）」となります。

カスタマイズ設定

ログオンユーザー覧ポートレットの「スクリプト」を編集することでカスタマイズできます。

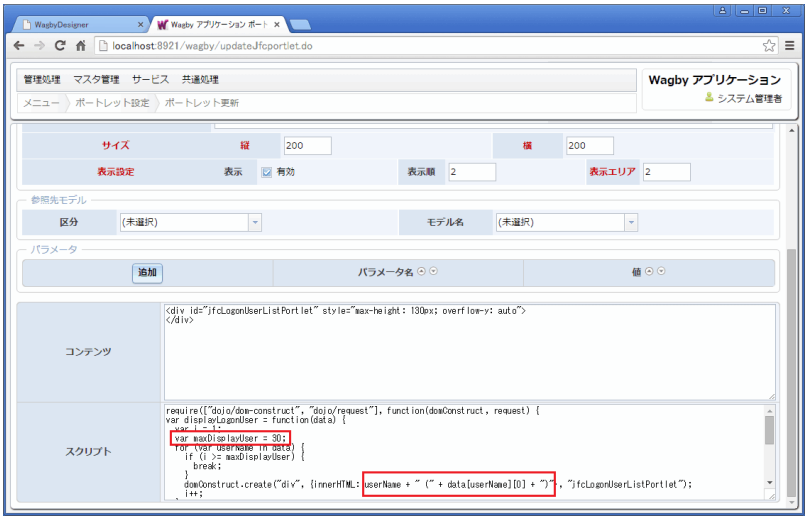


図 ログオンユーザー覧ポートレットのスクリプト欄

上限数を変更する

標準では、最大表示数は30です。図で、赤枠で囲っています。

```
var maxDisplayUser = 30;
```

この値を変更することで、表示上限数を調整できます。（この場合、ポートレット自身の縦サイズも調整するとよいでしょう。）

表示内容を変更する

標準では、「名前（アイドル時間)」です。図で、赤枠で囲っています。

```
userName + " (" + data[userName][0] + ")"
```

例えば次のように修正することで、アイドル時間を表示しないようにすることができます。

```
userName
```

ビルドは不要です。修正後、すぐにポータル画面に反映されます。

グラフポートレット

REST API を有効にする

ここでは「部門別売上集計グラフ」の例をポートレットに表示させる方法を説明します。

このモデルの「REST API を利用する」設定を有効にして、ビルドを行ってください。

グラフポートレットの設定

新しいポートレットを作成します。次のように入力してください。

設定項目	説明	入力例
ポートレットID	任意の文字列とします。英数字の組み合わせとします。	salestotalByDept
ポートレットタイプ	グラフを選択します。	グラフ
タイトル	ポートレット画面に表示されるタイトルです。任意の文字列を設定します。	部門別売上集計
サイズ	ポートレットの縦、横のサイズ（ピクセル値）を指定します。横幅は、同じ表示エリアに属する他のポートレットと合わせるとよいでしょう。	200 x 250
表示設定	有効とします。	有効
表示順	同じ表示エリアの中での初期表示順を指定します。整数値を指定してください。	1
表示エリア	どの表示エリアにポートレットを表示させるかを指定します。整数値を指定してください。	2
グラフ - 種類	利用するグラフの種類を指定します。	集合縦棒グラフ
グラフ - 色	利用するグラフの色を指定します。	青
グラフ - 集計ビュー番号	指定するモデルは複数の集計ビューを定義できます。どの集計ビュー（ディメンジョンとメジャーの組み合わせ）を用いるかを指定します。整数値を指定してください。	1
グラフ - 参照先モデル	区分をユーザーとし、モデルを指定します。ここで指定するモデルは、集計ビューが有効になっている必要があります。	売上集計

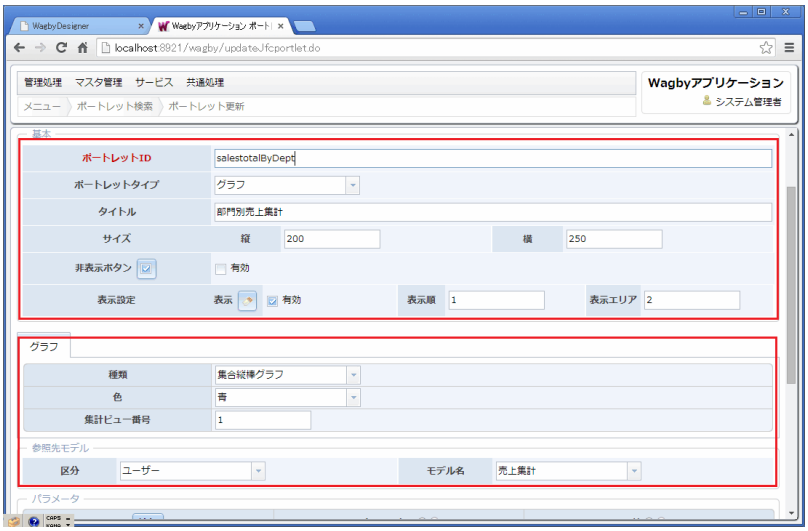


図 グラフポートレットの設定を行う

グラフポートレットを作成した一覧表示を図に示します。

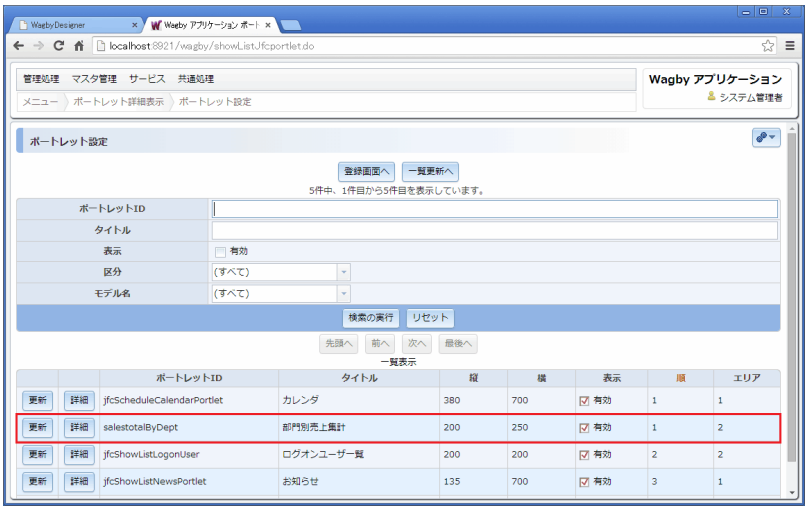


図 ポートレット一覧

グラフポートレットの表示

上で設定したモデルのグラフが、ポートレットに表示されます。

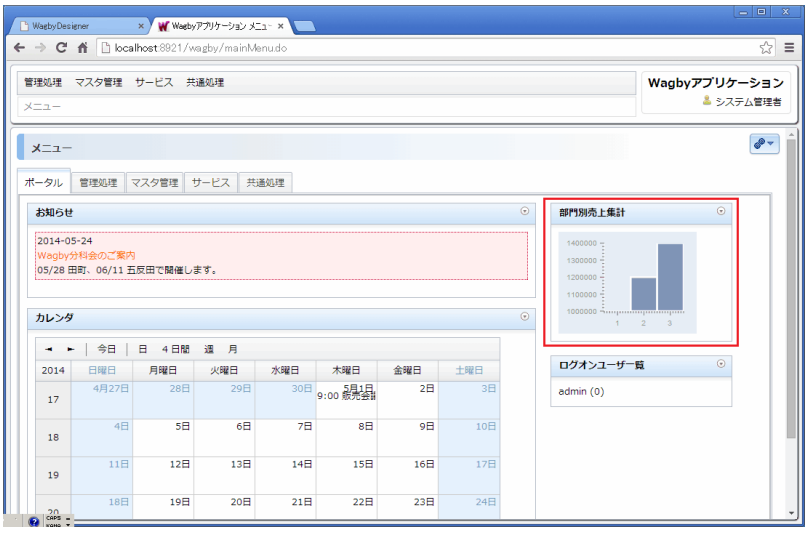


図 グラフポートレットの表示

複数のグラフポートレットを表示する

複数のグラフポートレットを表示させることもできます。図は、二つのグラフを表示させた例です。

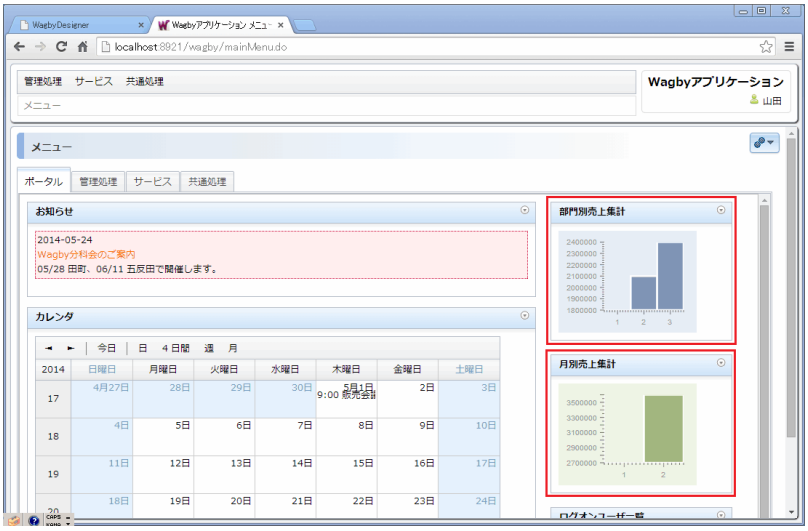


図 部門別売上集計と月別売上集計グラフ

定義方法

二つのグラフポートレットを作成します。

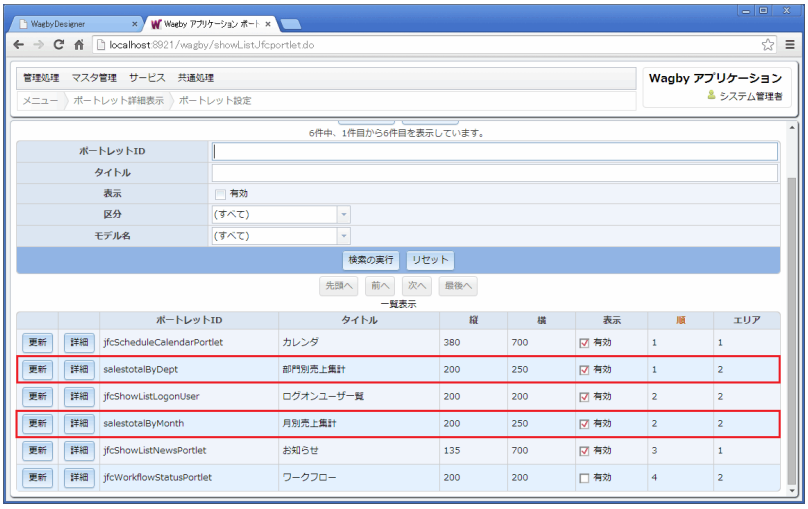


図 部門別売上集計と月別売上集計のためのポートレットを用意する

各ポートレットは、どの「集計ビュー」に対応するかを指定します。（今回は、月別集計は ″2″ 番の集計ビューと対応します。）

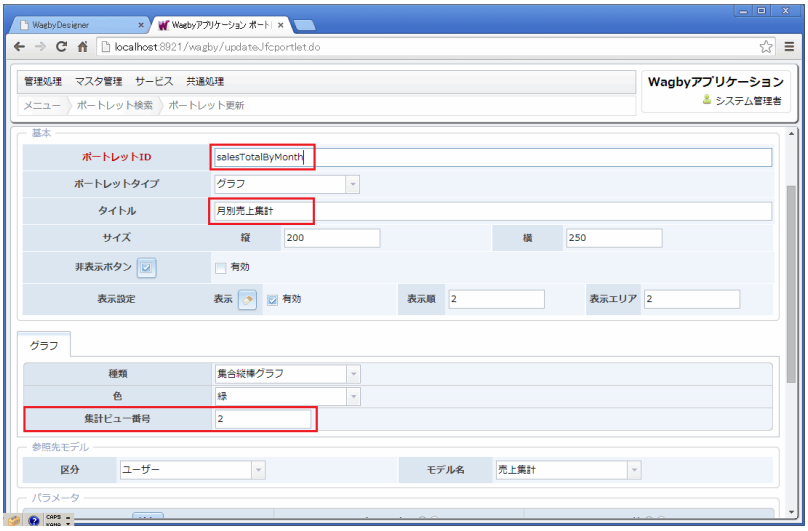


図 月別売上集計ポートレットの設定

利用上のヒント

ポータルのX軸の表示並び

X軸を「月」として1月,2月,...と並べて表示したい場合は、月を文字列型とし、かつ "01","02",..."12" とするとよいでしょう。

ワークフローポートレット

ワークフローポートレットの設定

ポートレット一覧から、ワークフローポートレットの設定画面を開きます。

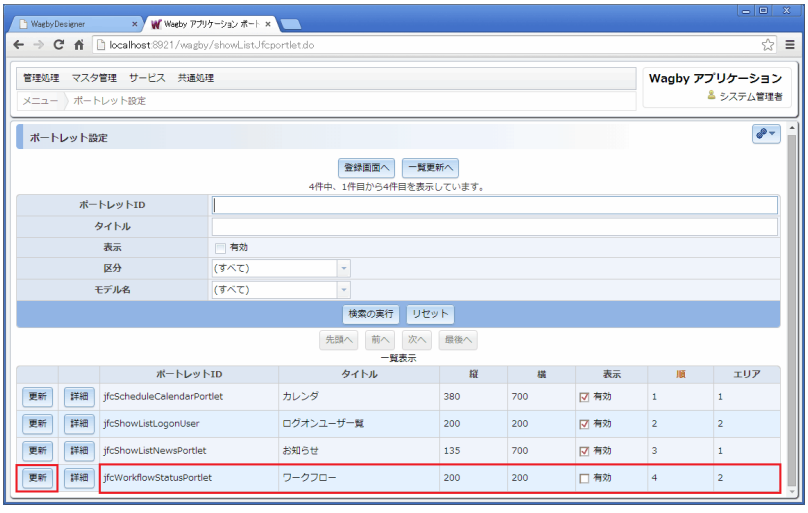


図 ワークフローポートレットを選択する

設定項目は次のとおりです。

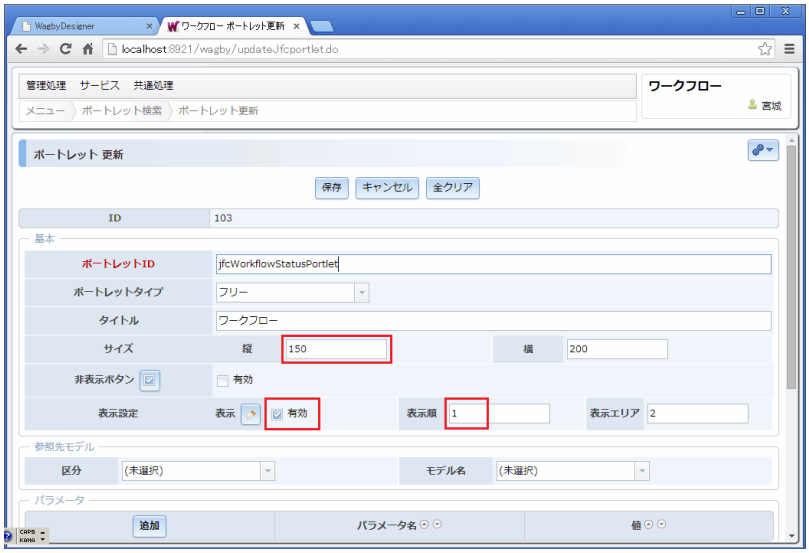


図 ワークフローポートレットの設定

設定項目	説明	入力例
サイズ	ポートレットの縦、横のサイズ（ピクセル値）を指定します。横幅は、同じ表示エリアに属する他のポートレットと合わせるとよいでしょう。縦サイズは開始・保留しているワークフロー件数が多い場合、拡げると見やすくなります。図で設定した縦150は、1件のワークフローを表示する最小限の高さです。	150 x 200
表示設定	有効とします。	有効
表示順	同じ表示エリアの中での初期表示順を指定します。整数値を指定してください。	1
表示エリア	どの表示エリアにポートレットを表示させるかを指定します。整数値を指定してください。	2

ワークフローポートレットを有効にした一覧表示を図に示します。

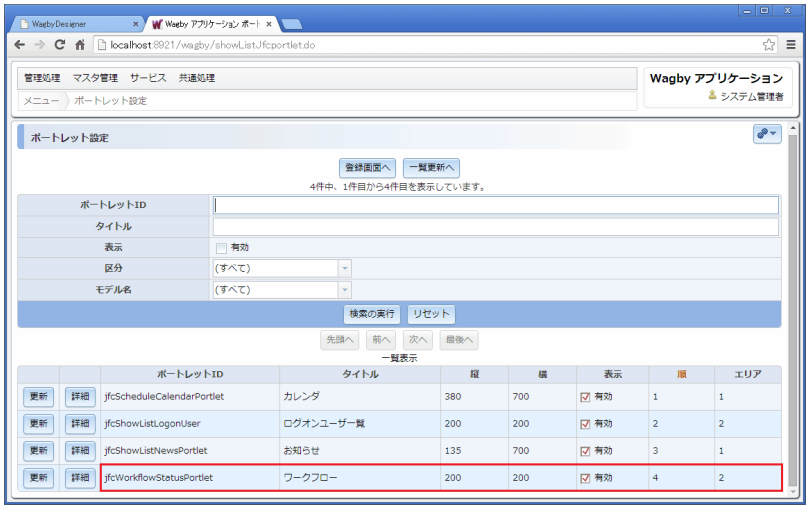


図 ワークフローポートレットが有効になった状態

ワークフローポートレットの表示

決裁権をもったアカウントでポータル画面を開いてみます。図のように、保留フロー（自分の決裁待ちであるワークフロー）が1件あることがわかるようになっています。

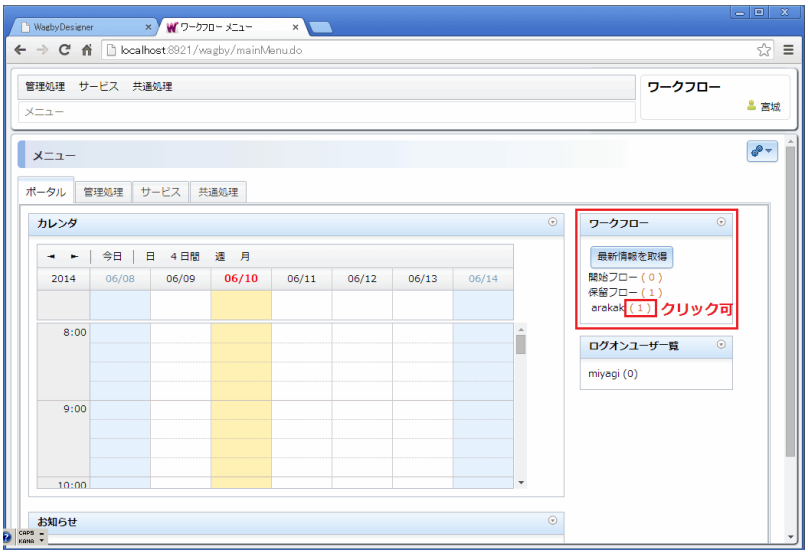


図 保留フローが1件あることが示される

このポートレットに用意された「最新情報を取得」ボタンを押すことで、最新の状態に書き換えることができます。（自動更新ではありません。）

決裁処理を行う

決裁

図で、ワークフロー申請者（ログオンアカウント名）をクリックすると、そのモデルについての「保留フロー状態検索画面」が表示されます。

図 保留フロー状態検索画面

図の画面で、承認するワークフローを選択し、承認することができます。（複数件を選択した一括承認も行えます。）

図 保留フローへの決裁

図 決裁の完了

確認

図はワークフローを申請したアカウントでログオンした画面です。ワークフローが決裁されたことがわかります。

図 決裁されたワークフロー

外部リンクポートレット（JavaScript）

例

図に示すような、外部ページへのリンクを集めたポートレットを用意してみます。

図 外部リンクを用意する

定義方法

新しいポートレットを作成します。次のように入力してください。

設定項目	説明	入力例
ポートレットID	任意の文字列とします。英数字の組み合わせとします。	linkPortlet
ポートレットタイプ	フリーを選択します。	フリー
タイトル	ポートレット画面に表示されるタイトルです。任意の文字列を設定します。	ITイベント・勉強会

サイズ	ポートレットの縦、横のサイズ（ピクセル値）を指定します。横幅は、同じ表示エリアに属する他のポートレットと合わせるとよいでしょう。	180 x 200
表示設定	有効とします。	有効
表示順	同じ表示エリアの中での初期表示順を指定します。整数値を指定してください。	1
表示エリア	どの表示エリアにポートレットを表示させるかを指定します。整数値を指定してください。	2

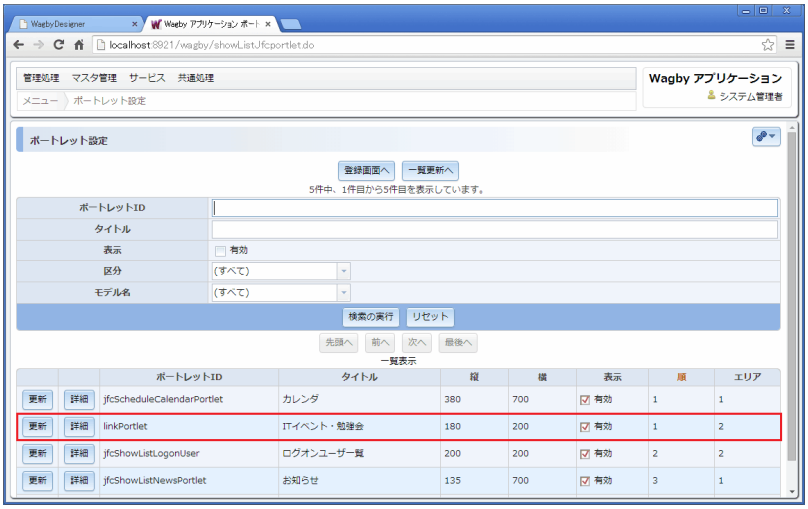


図 ポートレット一覧

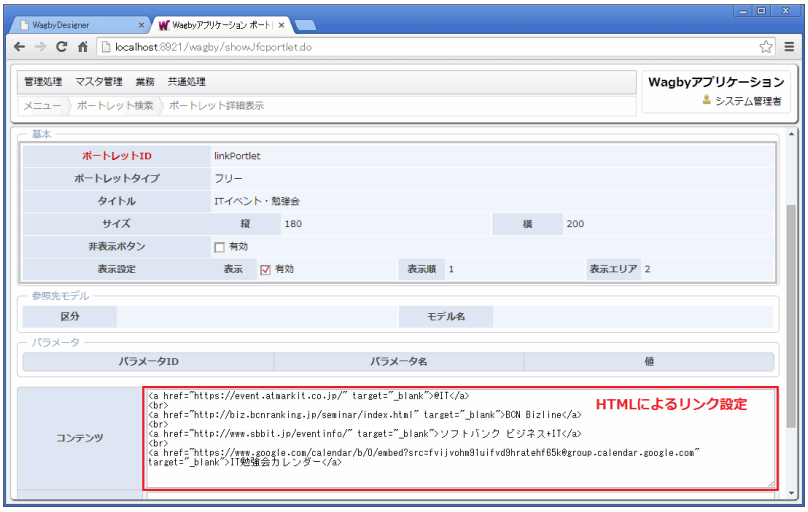


図 リンクポートレットの設定画面例

コンテンツ

図のコンテンツ部は次のように設定しています。anchor要素にtarget属性を指定することで、リンクを押下すると別ウィンドウ（または別タブ）に表示されるようにしています。

```
<a href="https://event.atmarkit.co.jp/" target="_blank" rel="noopener noreferrer">@IT</a>
<br>
<a href="http://biz.bcnranking.jp/seminar/index.html" target="_blank" rel="noopener
 noreferrer">BCN Bizline</a>
```

```
<br>
<a href="http://www.sbbit.jp/eventinfo/" target="_blank" rel="noopener noreferrer">ソフトバンク
 ビジネス+IT</a>
<br>
<a href="https://www.google.com/calendar/b/0/embed?
src=fvijvohm91uifvd9hratehf65k@group.calendar.google.com" target="_blank" rel="noopener
 noreferrer">IT勉強会カレンダー</a>
```

スクリプト

スクリプトはありません。

ブログパーツを利用する（JavaScript）

ブログパーツ

ポートレット内に、インターネット上で公開されている「ブログパーツ」を埋め込むことができます。ここでは、その方法を紹介します。

一部のブログパーツは利用ライセンスに制限が設けられているものがあります。（例えば社内イントラネット環境での利用禁止など。）詳細は、各ブログパーツの説明ページをお読みください。ライセンス条項に適合しない使い方を行うことはできませんのでご注意ください。

アナログ時計

例

Biz-Boxというサイトが公開している、アナログ時計を表示した例を図に示します。

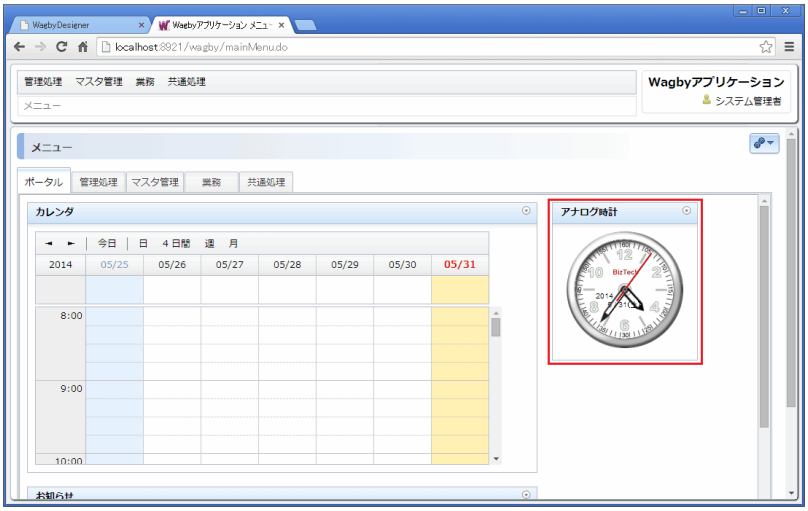


図 アナログ時計

定義方法

新しいポートレットを作成します。次のように入力してください。

設定項目	説明	入力例
ポートレットID	任意の文字列とします。英数字の組み合わせとします。	analogClock
ポートレットタイプ	フリーを選択します。	フリー

タイトル	ポートレット画面に表示されるタイトルです。任意の文字列を設定します。	アナログ時計
サイズ	ポートレットの縦、横のサイズ（ピクセル値）を指定します。横幅は、同じ表示エリアに属する他のポートレットと合わせるとよいでしょう。	220 x 200
表示設定	有効とします。	有効
表示順	同じ表示エリアの中での初期表示順を指定します。整数値を指定してください。	1
表示エリア	どの表示エリアにポートレットを表示させるかを指定します。整数値を指定してください。	2

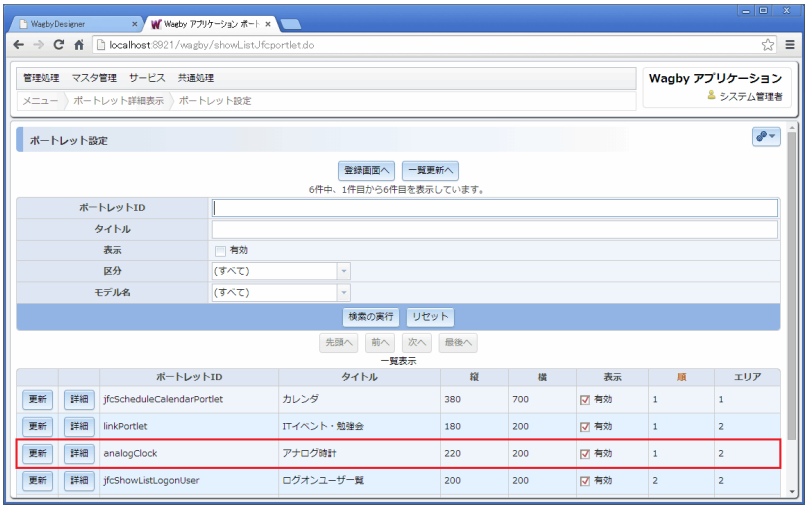


図 ポートレット一覧

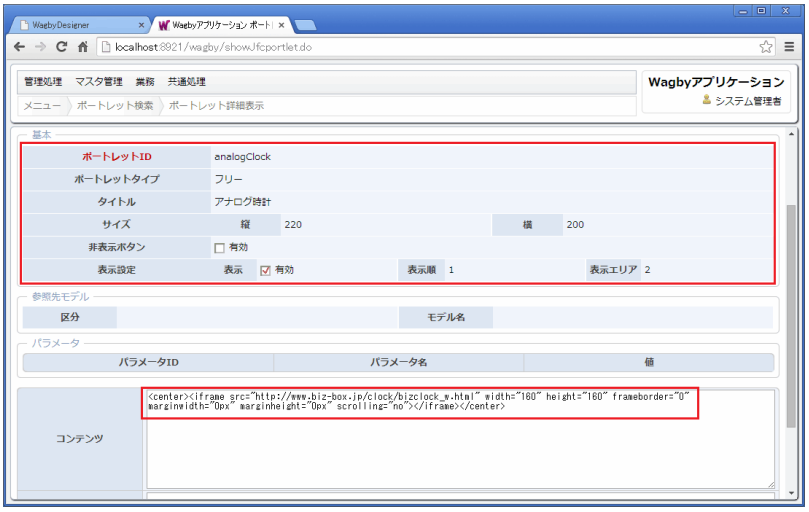


図 アナログ時計を埋め込んだポートレットの設定画面例

コンテンツ

図のコンテンツ部は次のように設定しています。

```
<center><iframe src="http://www.biz-box.jp/clock/bizclock_w.html" width="160" height="160"
 frameborder="0" marginwidth="0px" marginheight="0px" scrolling="no"></iframe></center>
```

任意のモデルの値を表示する（JavaScript）

REST API と JSON

WagbyはREST APIを使って任意のモデルの値を取得することができます。その情報は JSON という形式で得られます。

この仕組みを利用し、ポートレットからREST API経由でJSONデータを取得します。JavaScriptを使ってJSONのデータを解析し、必要な情報を取得してポートレット画面に埋め込むことができます。

「お知らせ」ポートレットは、このような作り方になっています。同ポートレットのコンテンツとスクリプトを教材として使うことができます。

「日報モデル」を用いたサンプル実装

例

「日報」モデルを用いてサンプル実装を行ないます。このモデルは、毎日の営業状況（問合せ数、販売数、クレーム数）を管理することができます。

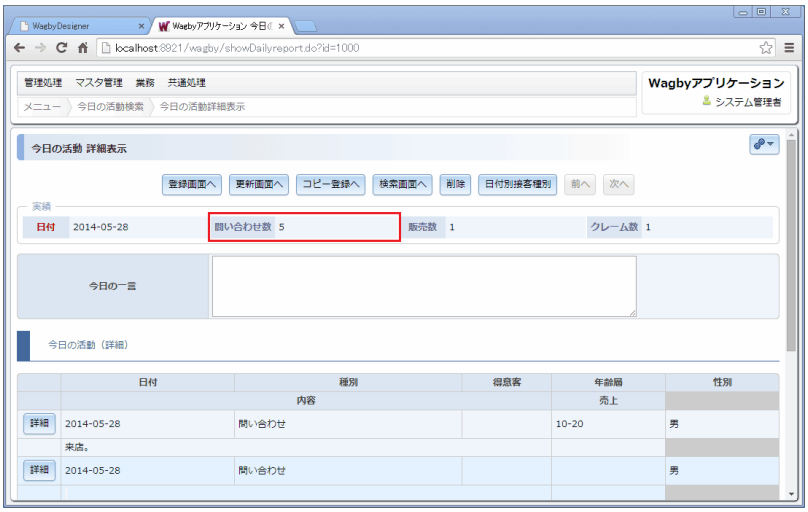


図 日報モデルの表示画面

一覧表示時も、この営業状況を確認できます。

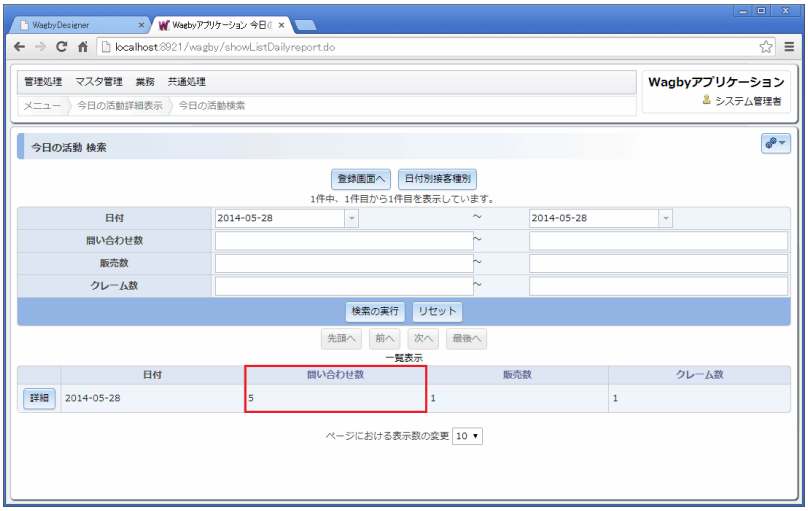


図 日報モデルの一覧表示

図における一覧表示の値を取得し、ポートレットに図のように表示するようにします。

図 ポートレットで最新の営業状況を見える化する

JSONデータの確認

REST API を使って日報モデルの一覧表示を取得します。取得したJSONの構造を確認します。

```
http://localhost:8921/wagby/rest/dailyreport/list
```

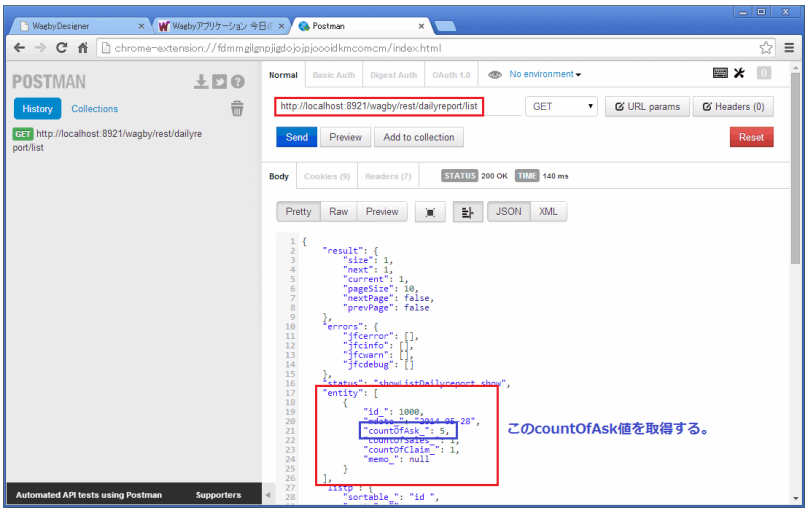


図 JSONの構造を確認する

ここではentity要素の最初の配列内に含まれるcountOfAsk_という値を取得するようにします。

ポートレットを用意する

"displayCountOfAsk" というポートレットを用意します。

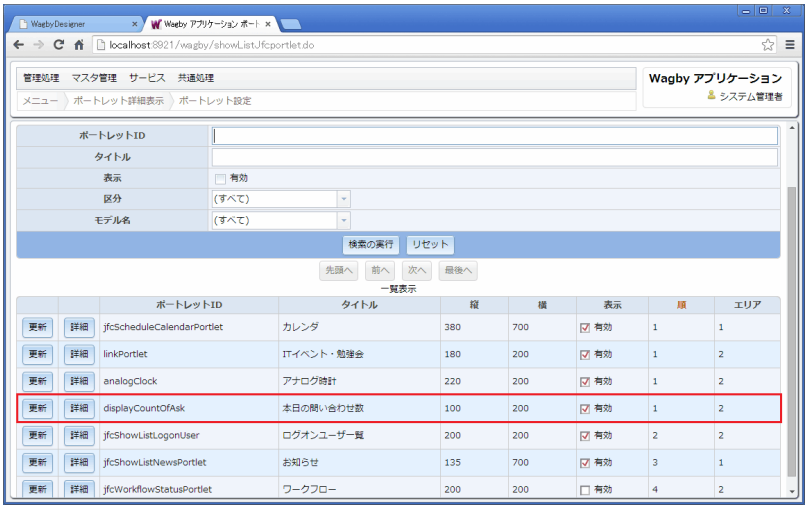


図 値を表示するためのdisplayCountOfAskポートレットを用意した

このポートレットの設定を図,図に示します。

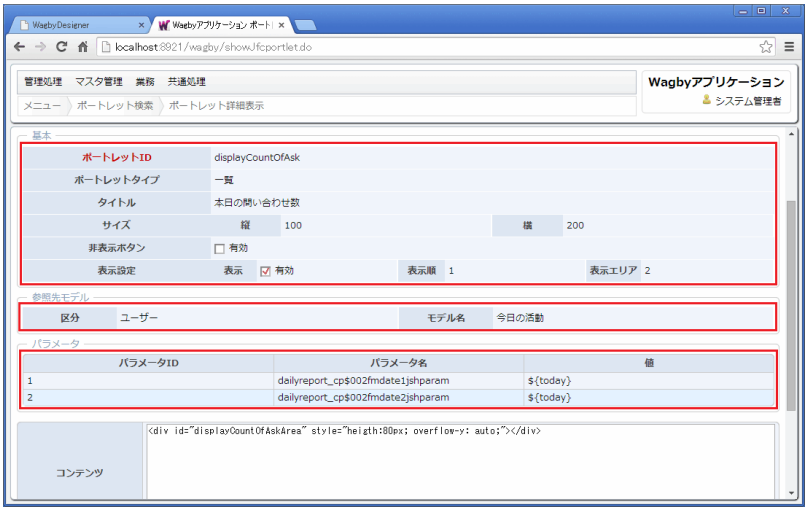


図 ポートレットの設定(1)

設定項目	説明	入力例
ポートレットID	任意の文字列とします。英数字の組み合わせとします。	displayCountOfAsk
ポートレットタイプ	一覧を選択します。対象モデルに対して "list" でデータを取得し、その結果を "data" という入れ物に格納します。	一覧
タイトル	ポートレット画面に表示されるタイトルです。任意の文字列を設定します。	本日の問合せ数
サイズ	ポートレットの縦、横のサイズ（ピクセル値）を指定します。横幅は、同じ表示エリアに属する他のポートレットと合わせるとよいでしょう。	100 x 200
表示設定	有効とします。	有効
表示順	同じ表示エリアの中での初期表示順を指定します。整数値を指定してください。	1
表示エリア	どの表示エリアにポートレットを表示させるかを指定しま	2

	す。整数値を指定してください。	
参照先モデル	ユーザーが作成した日報モデル「今日の活動」を指定します。	今日の活動
パラメータ	"list" の REST API 呼び出し時のパラメータを付与することができます。ここでは「本日の日付」を範囲検索の条件として渡しています。なお ${today} は Wagby が提供する特別な表記です。 パラメータ名は、検索条件画面のWebフォームに埋め込まれたinput要素のid値を用います。（自動生成されたJSPファイルから、該当する項目のid値を探してください。）	dailyreport_cp $002fmdate1jshparamdailyreport_cp $002fmdate2jshparam

コンテンツ

図のコンテンツ部は次のように設定しています。

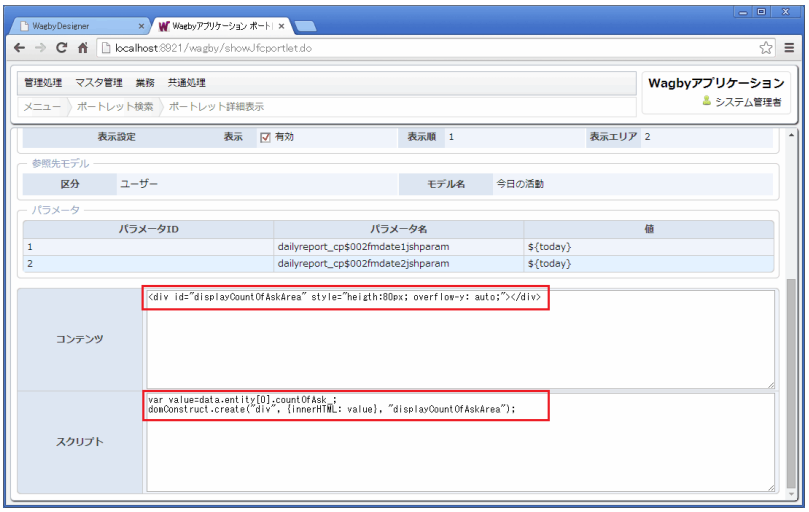


図 ポートレットの設定(2)

```
<div id="displayCountOfAskArea" style="heigth:80px; overflow-y: auto;"></div>
```

この空のdiv要素に対して、スクリプトから値が設定されます。

スクリプト

図のスクリプト部は次のように設定しています。

```
var value=data.entity[0].countOfAsk_;
domConstruct.create("div", {innerHTML: value}, "displayCountOfAskArea");
```

dataは、ポートレットタイプが「一覧」のときに利用できます。検索結果が格納されています。　JSONのデータは "." をつかって辿ります。一行目で、目的の値（countOfAsk_）を取得し、これをvalueという変数に格納しています。

2行目で、コンテンツ部に用意した空のdiv要素に、valueを格納します。全体はdiv要素で囲まれます。

dataの並びは一覧表示での並びと同様となります。（検索時に指定できるパラメータにソートキーを指定した場合は、指定に準じた並びとなります。）

上のコードにある domConstruct は、Dojotoolkit が提供する DOM 操作のためのモジュール dojo/dom-construct を使っています。ポートレットタイプが「一覧」の場合、すぐに利用することができるようにしています。dojo/dom-construct の使い方の詳細は、Dojotoolkit のマニュアルをお読みください。

見栄えの修正

値を中央揃えにし、文字色を青としました。またフォントサイズも大きくしています。

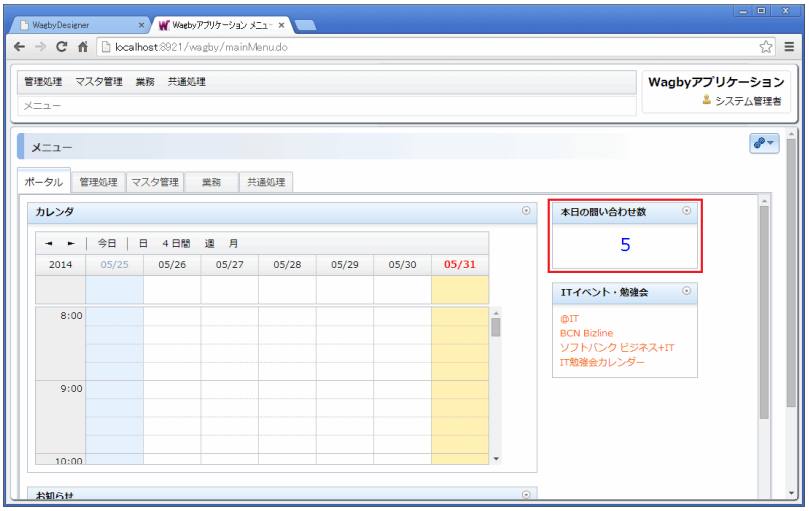


図 表示を修飾した例

設定

コンテンツ部にCSSのクラスを定義しています。 またスクリプトでは、このCSSを使うように指定しています。

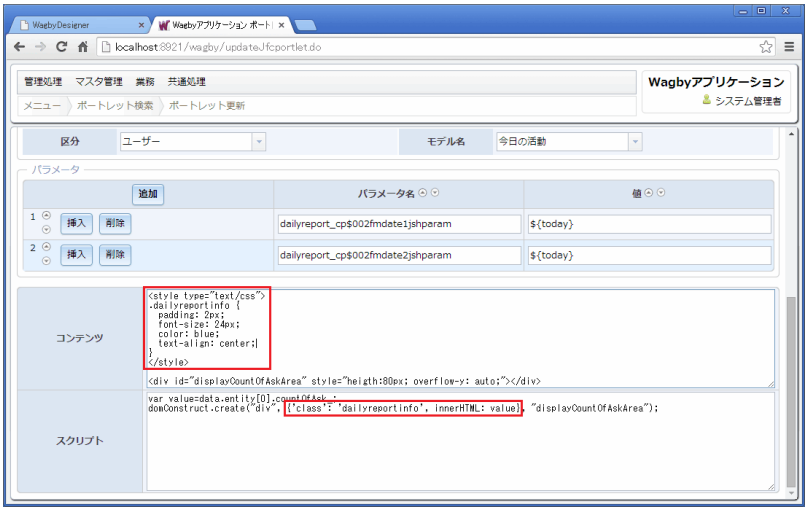


図 修正したコンテンツとスクリプト

```
<style type="text/css">
.dailyreportinfo {
  padding: 2px;
  font-size: 24px;
  color: blue;
  text-align: center;
}
</style>
<div id="displayCountOfAskArea" style="heigth:80px; overflow-y: auto;"></div>
```

```
var value=data.entity[0].countOfAsk_;
domConstruct.create("div", {'class': 'dailyreportinfo', innerHTML: value},
 "displayCountOfAskArea");
```

任意のモデルの値を表示する（2）（JavaScript）

準備

ここでは 1 件の「お知らせ」情報を取得し、ポートレットに出力する方法を説明します。対象モデルの説明は"お知らせポートレット"をお読みください。

プレゼンテーションモデル

Wagbyは1つのデータを表現するために「ストアモデル」と「プレゼンテーションモデル」という形式を扱います。ストアモデルはデータベースに永続化される内容であり、プレゼンテーションモデルは画面表示用です。

今回は「プレゼンテーションモデル」を扱います。はじめに、この内容を確認しておきます。

図に示す情報を扱うものとします。

図 お知らせのサンプルデータ

(352)

POSTMANを使った確認

REST API を使って、このデータを JSON 形式で確認してみます。Google Chrome のプラグインである POSTMAN を使って説明します。

はじめにPOSTMANからWagbyへ、ログオン認証を行っておきます。URLは "http://localhost:8921/wagby/rest/session" となります。詳細は"Wagby Developer Network > REST API 活用ガイド > ログオン"をお読みください。

(352) 上の「お知らせ」モデルは「作成日」「表示期限日」の日付フォーマットを和暦に変更しています。このサンプルの説明そのものには影響ありませんので、以降の説明は、Wagby出荷時に含まれている「お知らせ」モデルをそのまま使ってもかまいません。

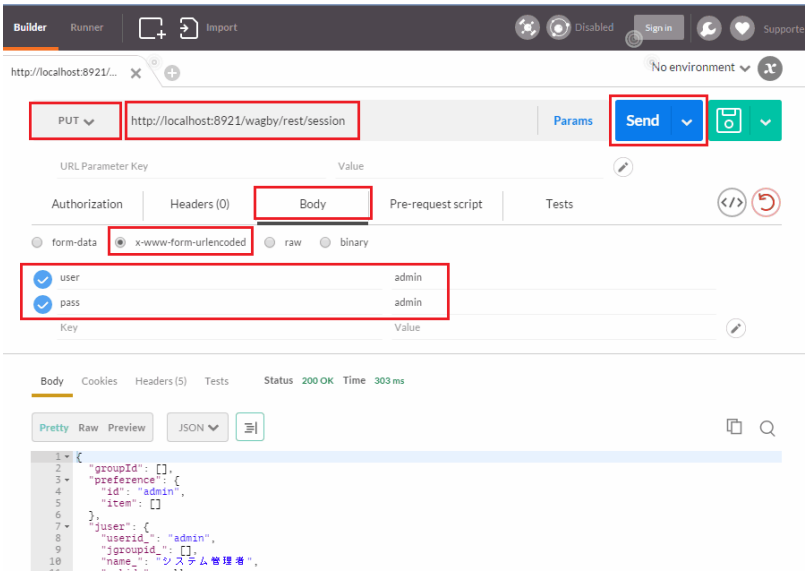


図 POSTMANでログオン認証を行う

図のように、アカウントとパスワードはx-www-form-urlencodedのパラメータとし、PUTで送信します。

ログオン認証に成功したら、主キー1000のお知らせ情報を確認してみます。URLはhttp://localhost:8921/wagby/rest/jnews/entry/1000とします。

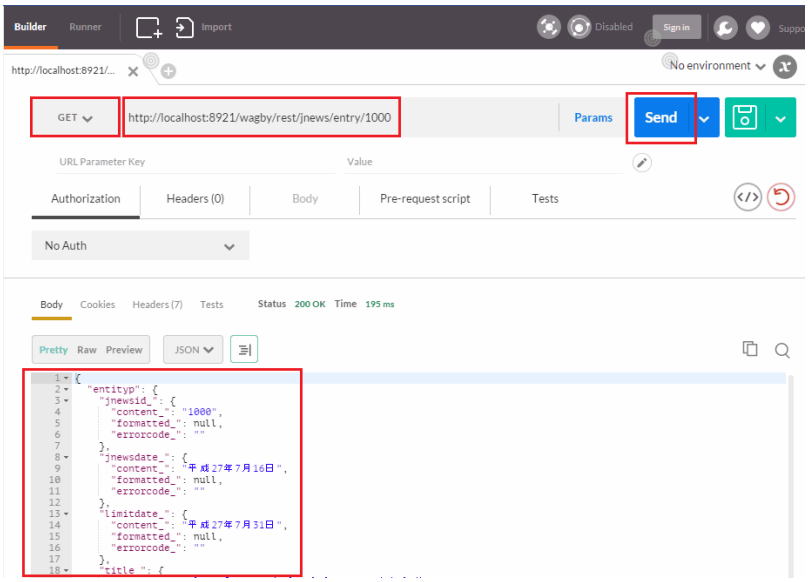


図 1件のお知らせデータを取得する

得られたサンプルデータは次のようになっています。

```
{
  "entityp": {
    "jnewsid_": {
      "content_": "1000",
      "formatted_": null,
      "errorcode_": ""
    },
    "jnewsdate_": {
      "content_": "平成27年7月16日",
      "formatted_": null,
      "errorcode_": ""
    },
    "limitdate_": {
```

```
      "content_": "平成27年7月31日",
      "formatted_": null,
      "errorcode_": ""
    },
    "title_": {
      "content_": "ジャスミンソフトビーチパーティのお知らせ",
      "formatted_": null,
      "errorcode_": ""
    },
    "report_": {
      "content_": "7月31日夕方より恒例のビーチパーティを開催します。¥¥¥n当日はラフな格好で出社してください。",
      "formatted_": null,
      "errorcode_": ""
    },
    "createuser_": {
      "content_": "admin",
      "formatted_": null,
      "errorcode_": ""
    }
  },
  "pkey": null,
  "errors": {
    "jfcinfo": [],
    "jfcdebug": [],
    "jfcwarn": [],
    "jfcerror": []
  },
  "entity": {
    "jnewsid_": 1000,
    "jnewsdate_": "2015-07-16",
    "limitdate_": "2015-07-31",
    "title_": "ジャスミンソフトビーチパーティのお知らせ",
    "report_": "7月31日夕方より恒例のビーチパーティを開催します。¥¥¥n当日はラフな格好で出社してください。",
    "createuser_": "admin"
  },
  "status": "showJnews.input"
}
```


Wagbyのプレゼンテーションモデルは、REST API を使うと "entityp" という名前で扱うことができます。例えば（上の例で赤字とした）タイトル情報は JavaScript を使って、次のように扱うことができます。

```
entityp.title_.content_
```

dojo/requestとdojo/dom-constructを用いたサンプル

それでは、新規のポートレットを作成してみましょう。ポートレットタイプは「フリー」とします。

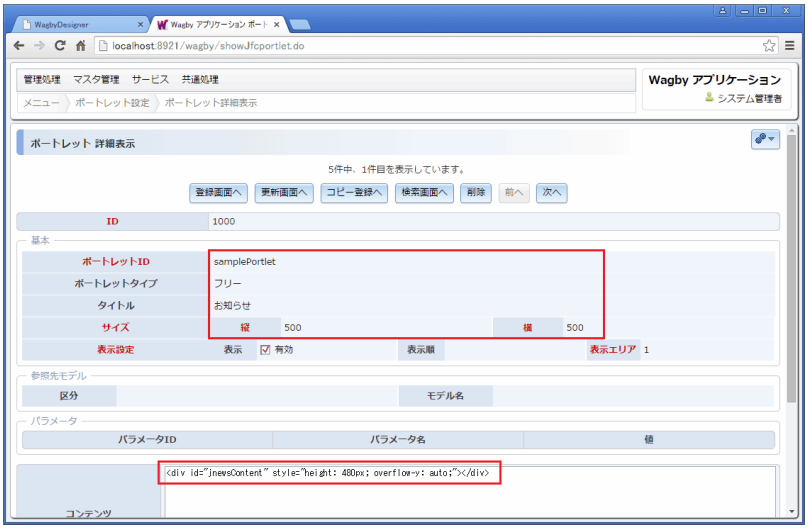


図 サンプルポートレット(1)

図 サンプルポートレット(2)

コンテンツ部は、空の div 要素を用意しておきます。後述するスクリプト（JavaScript）によって、ここに値をセットします。

```
<div id="jnewsContent" style="height: 480px; overflow-y: auto;"></div>
```

スクリプトコードを次に示します。

```
try {
  require(["dojo/request","dojo/dom-construct"],
    function(request,domConstruct) {
      request("http://localhost:8921/wagby/rest/jnews/entry/1000").then(
        function(response) {
          /*console.log(response);*/
          domConstruct.create("div", { innerHTML: "<p>"+response+"</p>" }, "jnewsContent");
        }
      );
    }
  );
} catch (e) {
  console.error(e);
}
```

- 全体を try ... catch で囲っているのはデバッグ用です。ポートレット表示時にエラーが発生した場合、ブラウザのJavaScriptコンソールにエラーメッセージが記載されます。（開発時はJavaScriptコンソールを表示するようにするとよいでしょう。）デバッグが終わったら、この try ... catch は除いてかまいません。
- 今回の例では dojo/request モジュールと dojo/dom-construct モジュールを活用しています。Wagby のポートレット開発では Dojotoolkit をすぐに利用することができます。require のあとに、使いたいモジュールを指定します。
- requestとdomConstructの使い方の詳細は、Dojotoolkitのマニュアルをお読みください。requestはHTTP通信を行います。domConstructはページの書き換えを行います。

・　コメントとしているconsole.logを有効にすることで、JavaScriptコンソールにも値を表示させることができます。デバッグ時にお使いください。

このポートレットを有効にすると、図のように表示されます。

REST API の結果を文字列としてそのまま表示させていることがわかります。

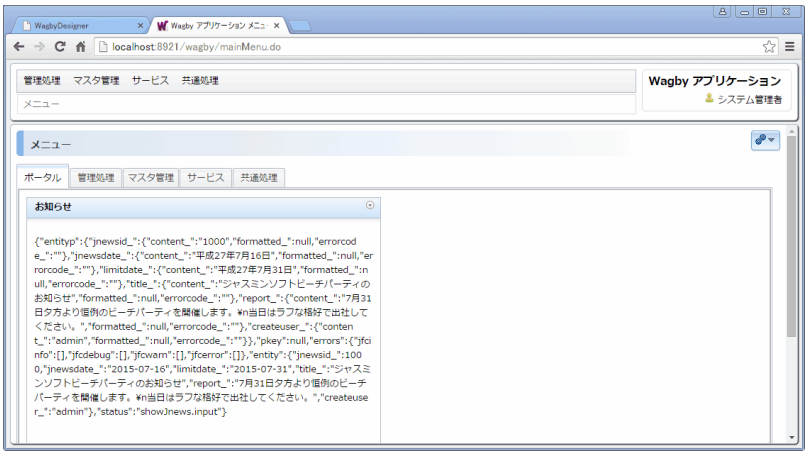


図 ポートレット表示例

JSONオブジェクトへの変換

次に、REST API で返された文字列を JSON オブジェクトとして扱えるようにします。request 利用時に handleAs 指定を付与します。

```
require(["dojo/request","dojo/dom-construct"],
  function(request,domConstruct) {
    request("http://localhost:8921/wagby/rest/jnews/entry/1000", { handleAs: "json" }).then(
      function(response) {
        console.log(response);
        domConstruct.create("div", { innerHTML: "<p>"+response+"</p>" }, "jnewsContent");
      }
    );
  }
);
```

この結果、JavaScriptコンソールには次のように表示されることがわかります。

```
Object {entityp: Object, pkey: null, errors: Object, entity: Object, status: "showJnews.input"}
```

JSONオブジェクトへの変換ができましたので、JavaScriptを使ってプレゼンテーションモデルを走査します。

JSONオブジェクトから値を取得する

スクリプトコードを次のように修正します。

```
require(["dojo/request","dojo/dom-construct"],
  function(request,domConstruct) {
    request("http://localhost:8921/wagby/rest/jnews/entry/1000", { handleAs: "json" }).then(
      function(response) {
        var content = response.entityp.title_.content_;
        /*console.log(content);*/
        domConstruct.create("div", { innerHTML: "<p>"+content+"</p>" }, "jnewsContent");
      }
    );
  }
```

```
);
```

これによって画面は次のようになります。プレゼンテーションモデルから値を取得できていることがわかります。

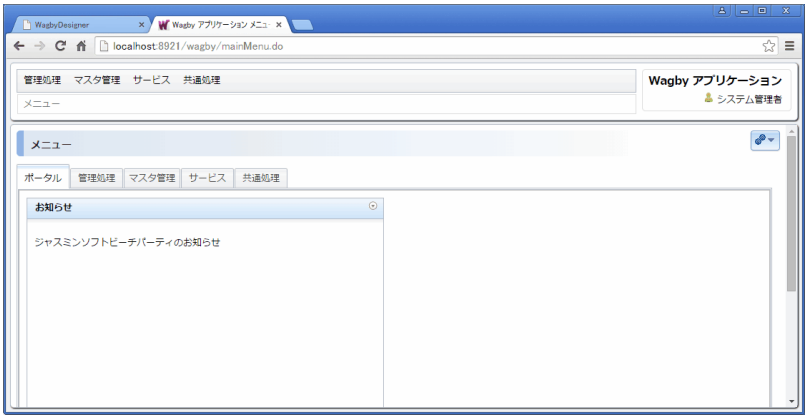


図 ポートレット表示

ゲージ部品を利用する(JavaScript)

準備

このページを読む前に、REST API経由でJSONデータを取得する「任意のモデルの値を表示する」ポートレットを用意してください。

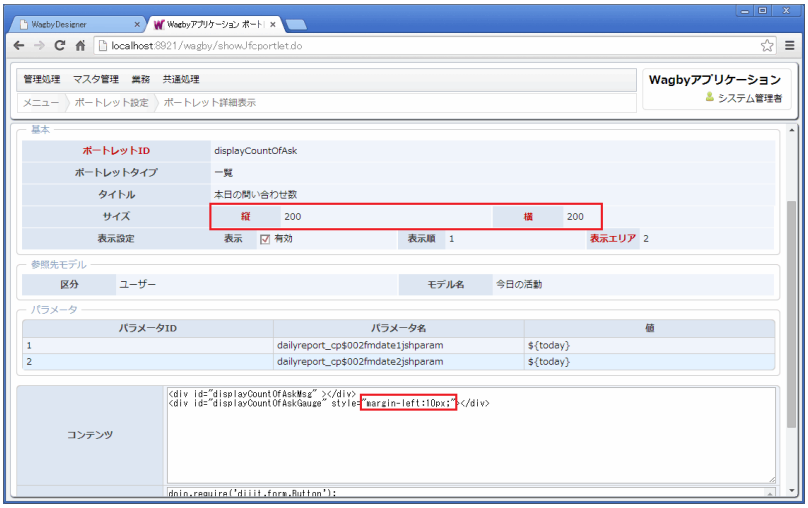
そのときに用意した "displayCountOfAsk" を修正し、3つのゲージ部品でそれぞれ表示させるサンプルコードを紹介します。 ゲージは（Wagbyに同梱されている）Dojotoolkit が提供する部品を使っています。

GlossyCircularGaugeを使う

例

サイズ

縦200, 横200としています。



コンテンツ

```
<div id="displayCountOfAskMsg" ></div>
<div id="displayCountOfAskGauge" style="margin-left:10px;"></div>
```

スクリプト

```
dojo.require('dijit.form.Button');
dojo.require('dojox.gauges.GlossyCircularGauge');
var displayCountOfAskGaugesFunc = function() {
    var value = data.entity[0].countOfAsk_;
    var glossyCircular1 = new dojox.gauges.GlossyCircularGauge({
        background: [255, 255, 255, 0],
        title: 'value',
        id: "displayCountOfAskGaugeParts",
        width: 160,
        height: 140,
        value: value,
        noChange: true
    }, dojo.byId("displayCountOfAskGauge"));
    glossyCircular1.startup();
};
if (data.listp.item_.length > 0) {
  dojo.ready(displayCountOfAskGaugesFunc);
} else {
  domConstruct.create("div", {'class': 'nomsg', innerHTML: '問い合わせデータはありません'},
 "displayCountOfAskMsg");
}
```

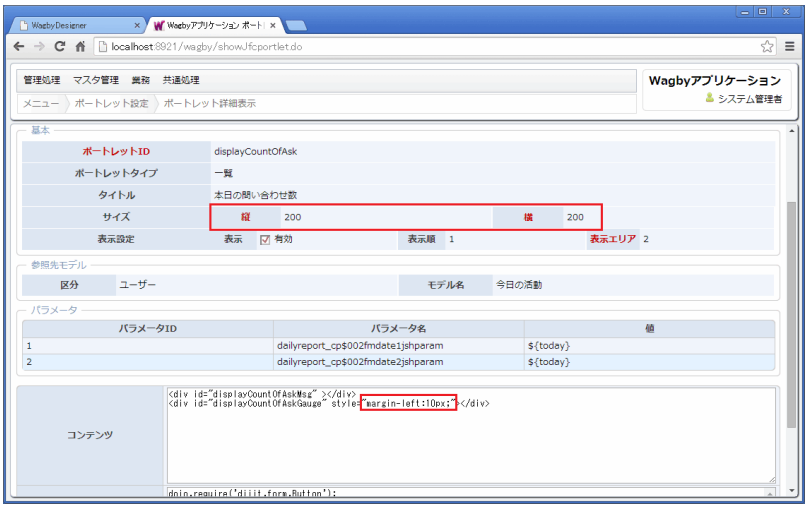
上のコードにある domConstruct は、Dojotoolkit が提供する DOM 操作のためのモジュール dojo/dom-construct を使っています。ポートレットタイプが「一覧」の場合、すぐに利用することができるようにしています。

GlossySemiCircularGaugeを使う

例

サイズ

縦200，横200としています。



コンテンツ

```
<div id="displayCountOfAskMsg" ></div>
<div id="displayCountOfAskGauge" style="margin-left:10px;"></div>
```

スクリプト

```
dojo.require('dijit.form.Button');
dojo.require('dojox.gauges.GlossySemiCircularGauge');
var displayCountOfAskGaugesFunc = function() {
    var value = data.entity[0].countOfAsk_;
    var glossyCircular1 = new dojox.gauges.GlossySemiCircularGauge({
        background: [255, 255, 255, 0],
        title: 'value',
        id: "displayCountOfAskGaugeParts",
        width: 160,
        height: 140,
        value: value,
        noChange: true
    }, dojo.byId("displayCountOfAskGauge"));
```

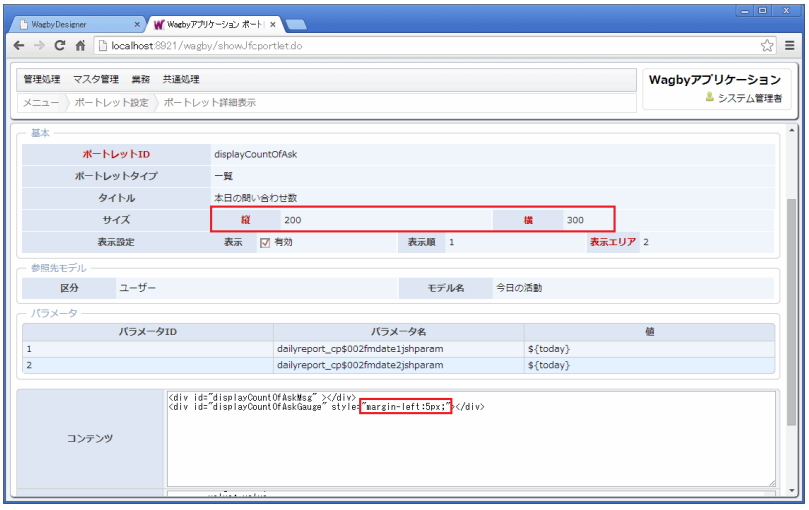
```
    glossyCircular1.startup();
};
if (data.listp.item_.length > 0) {
  dojo.ready(displayCountOfAskGaugesFunc);
} else {
  domConstruct.create("div", {'class': 'nomsg', innerHTML: '問い合わせデータはありません'},
 "displayCountOfAskMsg");
}
```

GlossyHorizontalGaugeを使う

例

サイズ

縦200，横300としています。



コンテンツ

```
<div id="displayCountOfAskMsg" ></div>
<div id="displayCountOfAskGauge" style="margin-left:5px;"></div>
```

スクリプト

```
dojo.require('dijit.form.Button');
dojo.require('dojox.gauges.GlossyHorizontalGauge');
var displayCountOfAskGaugesFunc = function() {
    var value = data.entity[0].countOfAsk_;
    var glossyCircular1 = new dojox.gauges.GlossyHorizontalGauge({
        background: [255, 255, 255, 0],
        title: 'value',
        id: "displayCountOfAskGaugeParts",
        width: 270,
        height: 140,
        value: value,
        noChange: true
    }, dojo.byId("displayCountOfAskGauge"));
    glossyCircular1.startup();
};
if (data.listp.item_.length > 0) {
  dojo.ready(displayCountOfAskGaugesFunc);
} else {
  domConstruct.create("div", {'class': 'nomsg', innerHTML: '問い合わせデータはありません'},
 "displayCountOfAskMsg");
}
```

百分率として利用する場合

上で紹介したゲージに渡すパラメータは "問合せ回数" というものでした。しかしゲージは本来、百分率として使うことが良いでしょう。この場合、表示する値を計算で求めます。以下の例は、一覧表示項目 "earings"（売上）の値が、目標値の何パーセントに達しているかを計算されたものです。

値 item.earnings_ は「文字列」となっているため、JavaScript が提供する関数 parseFloat で数字に変換してから演算します。

```
var displaySalestotalGauges = function() {
    var item = data.listp.item_[0];
    var value = (parseFloat(item.earnings_) / 1000000 * 100);
    var glossyCircular1 = new dojox.gauges.GlossyCircularGauge({
        background: [255, 255, 255, 0],
        title: 'Value',
        id: "salestotalGaugeParts",
        width: 200,
        height: 200,
        value: value,
        noChange: true
    }, dojo.byId("salestotalGauge"));
    glossyCircular1.startup();
};
```

CSV・Excel入出力

基本的な使い方

定義方法

ダウンロード、アップロード更新画面を用意する

ここでは「顧客」モデルに（データの）CSVダウンロード機能およびCSVアップロード更新機能を加えます。 「画面」設定へ遷移します。

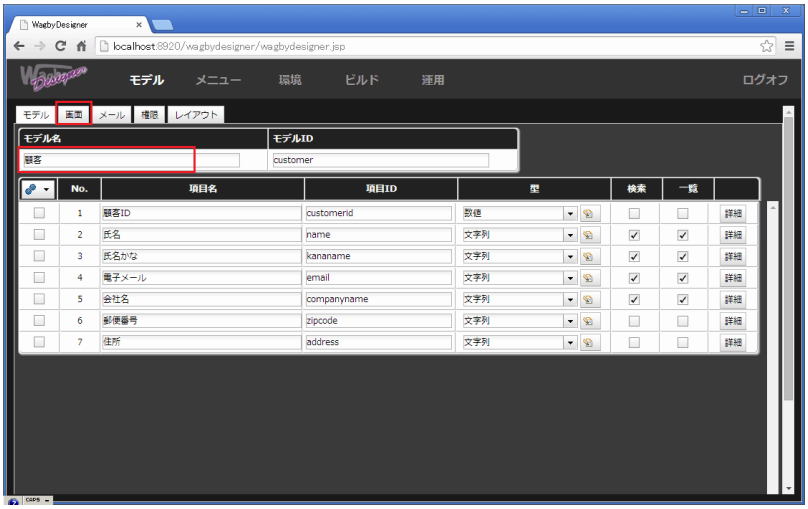


図 顧客モデルの定義

「ダウンロード画面」を有効にします。ダウンロード設定は初期値のままとします。

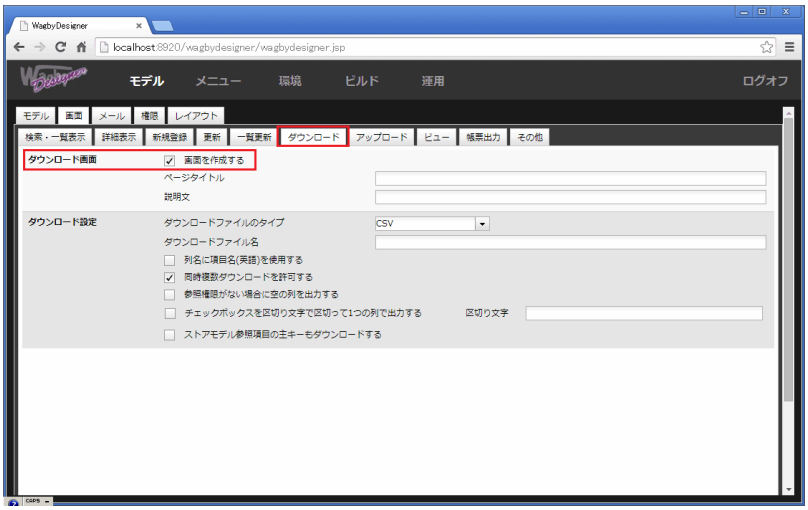


図 ダウンロード画面を有効にする

「アップロード更新画面」を有効にします。アップロード更新設定は初期値のままとします。

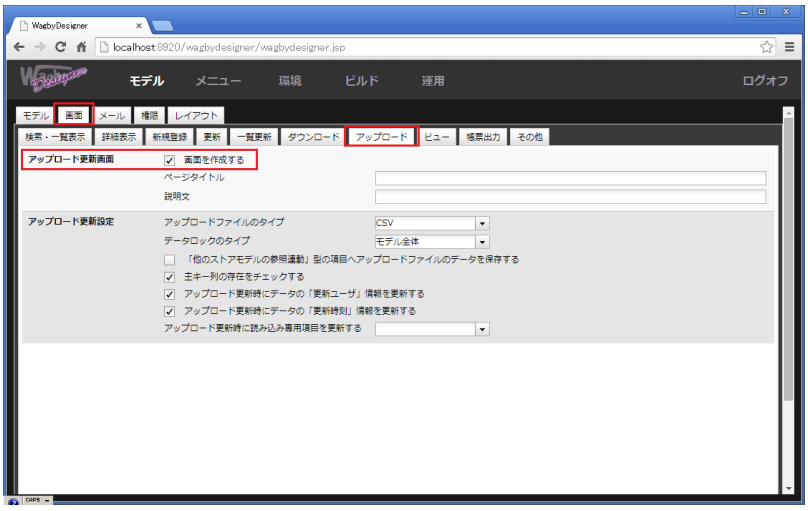


図 アップロード更新画面を有効にする

メニューへの追加

ダウンロードならびにアップロード更新のボタンはメニューに追加されています。

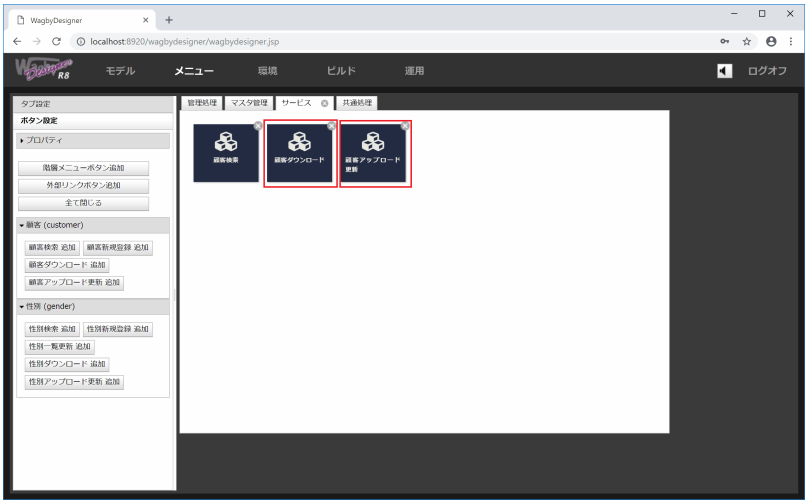


図 メニューが追加された

画面上で識別しやすいよう、背景色を変更するなどの工夫を行うとよいでしょう。詳細はメニューの説明をお読みください。

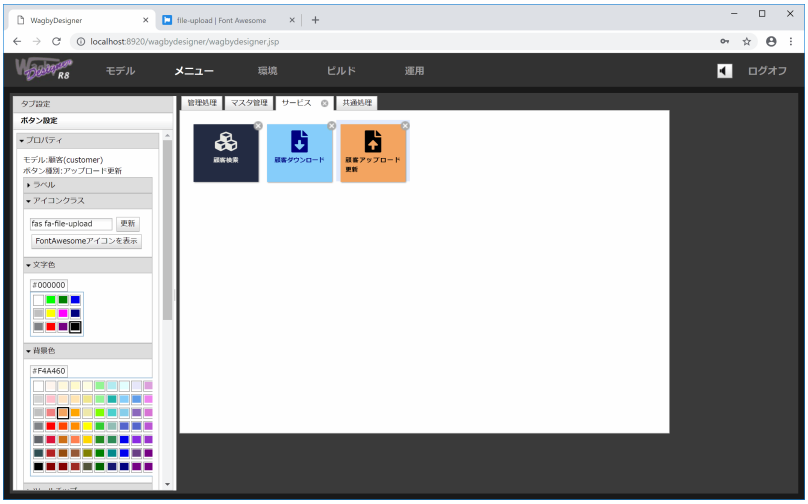


図 背景色を変更した

ダウンロード機能を試す

ビルドするとメニューが追加されています。はじめに「顧客ダウンロード」を試します。

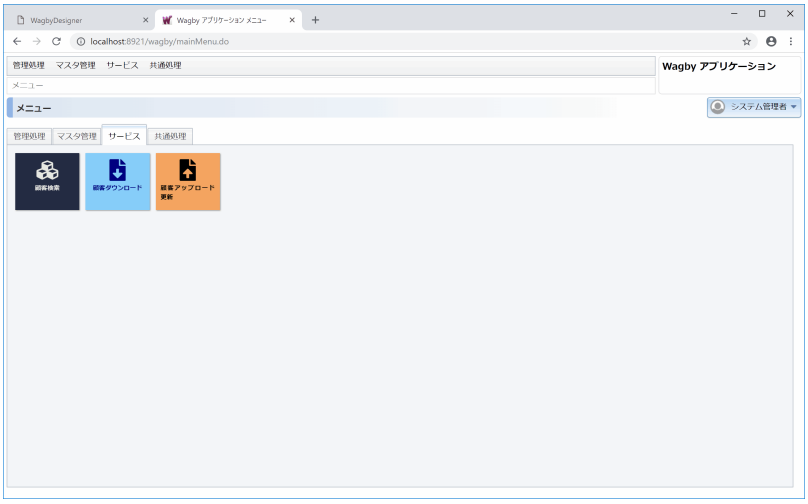


図 顧客ダウンロードと顧客アップロード更新ボタン

図に示すように、検索条件で絞込むこともできます。ダウンロードボタンを押下すると、現在の検索条件で絞り込まれたデータがダウンロードされます。リセットボタンを押下すると、検索条件をクリアします。

ここではデータが1件も登録されていない状態で、ダウンロードボタンを押下してみます。

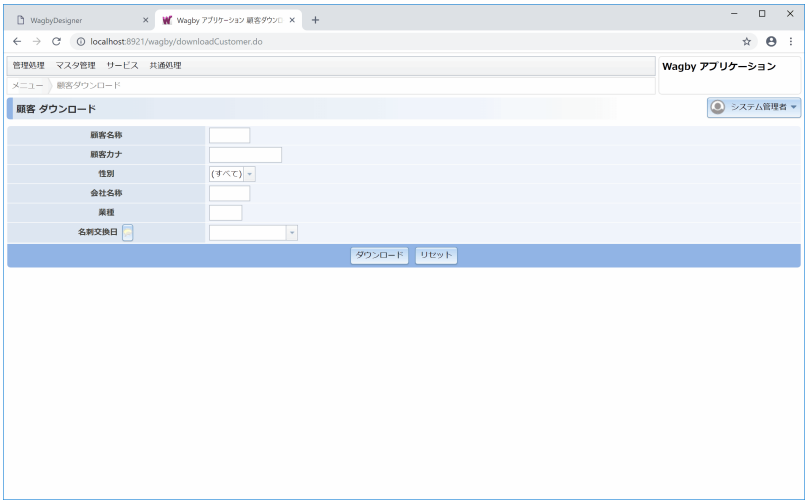


図 検索条件入力とダウンロードボタン

次のルールでダウンロードファイル名が決まります。

ログオンアカウント_data_All_日付時刻.拡張子

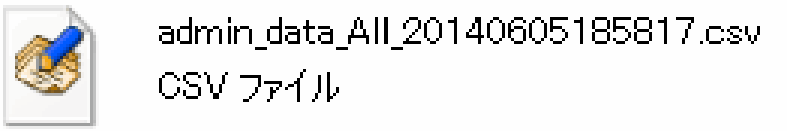


図 ダウンロードされたファイル

テキストエディタで開いてみます。データが1件もない場合でも、ヘッダと呼ばれる行が含まれていることがわかります。項目名がダブルクォートで囲まれています。

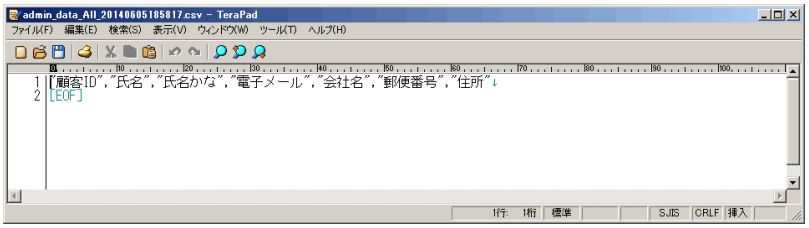


図 ヘッダ行のみが含まれたファイル

ここで2件のデータを事前に登録したとします。すると図のような内容となります。2行目以降がデータで、これも各値がダブルクォートで囲まれていることがわかります。

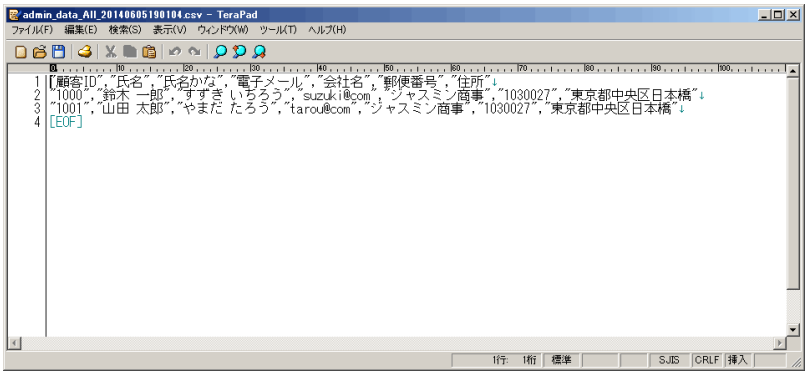


図 データが含まれていた場合のファイル

アップロード更新機能を試す

図のファイルを修正してみます。このファイルをWagbyへ送信して、更新する例を示します。

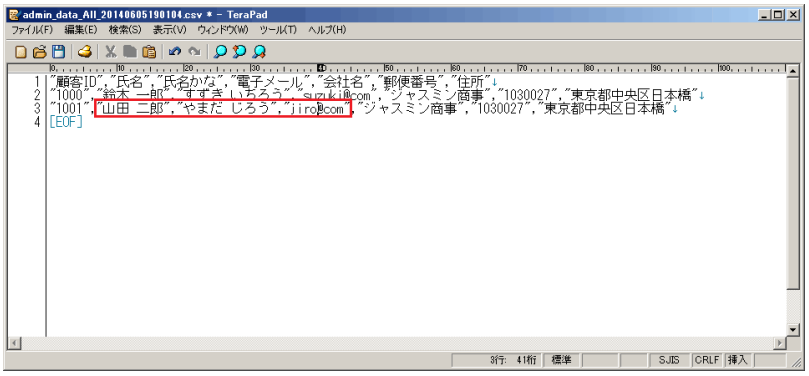


図 ファイルの内容を変更する

メニューから「顧客アップロード更新」を選択します。「ファイルを選択」ボタンを押して、図のファイルを指定します。すると図のようにファイルが確定されます。

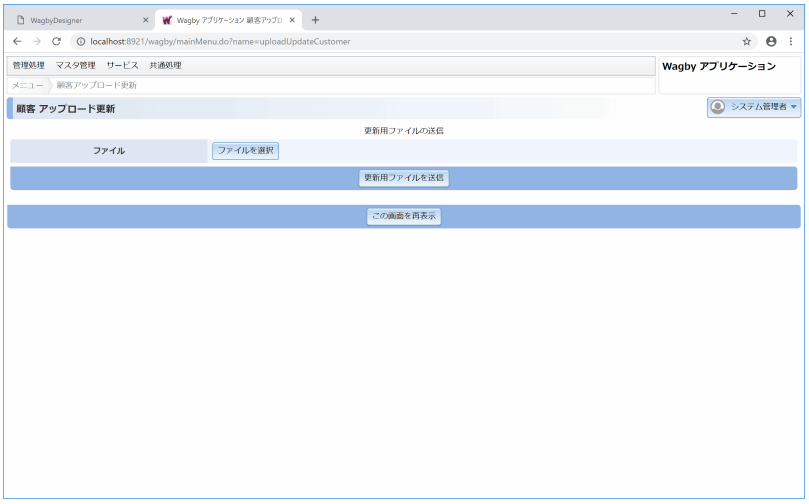


図 ファイルを選択ボタン

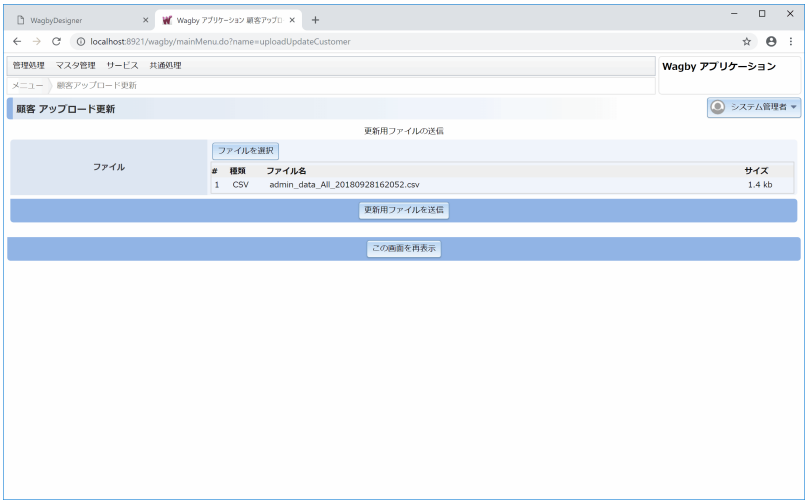


図 ファイルを選択した状態

Google Chrome などのブラウザは、「ファイルを選択」ボタンにファイルをドラッグ&ドロップすることもできます。

「更新用ファイルを送信」ボタンを押下します。すると画面が図のようになります。

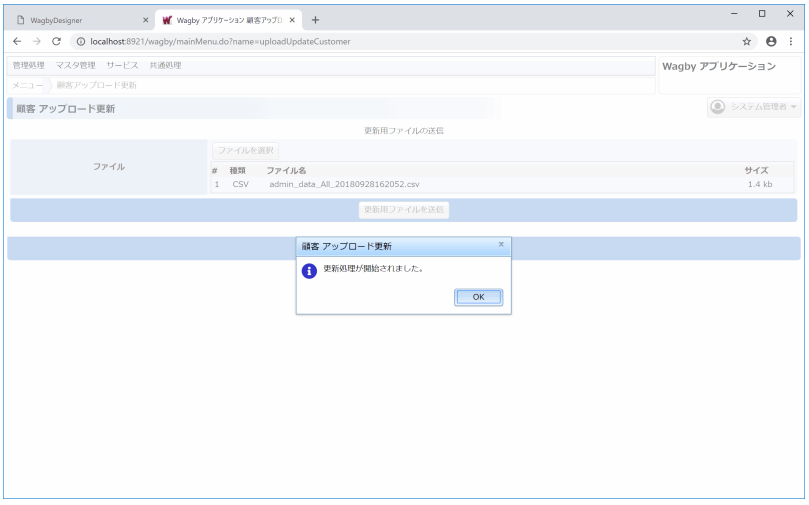


図 更新処理中

処理が終了すると図の画面へ遷移します。

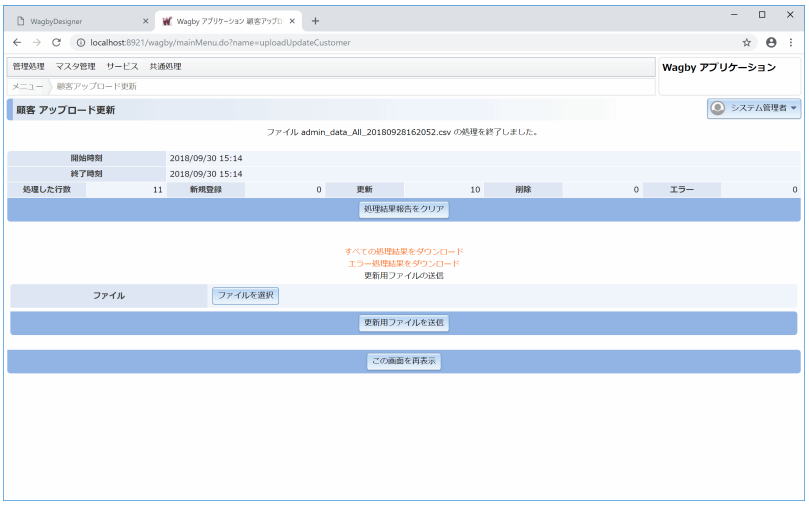


図 アップロード更新処理が終了した

「処理結果報告をクリア」ボタンを押さない限りは、直前のアップロード更新の処理結果が表示され続けます。処理結果の読み方は次のとおりです。

表示項目	説明
処理した行数	ファイルに含まれる行です。ヘッダ行も含めてカウントされます。
新規登録	新規に追加された行（データ）の件数を表示します。
更新	更新された行（データ）の件数を表示します。
削除	削除された行（データ）の件数を表示します。
エラー	処理がエラーになった行（データ）の件数を表示します。「エラー処理結果をダウンロード」というリンクをクリックすると詳細を確認できます。

データを確認してみます。ファイル内容に従って更新されていることがわかります。

図 データが更新された

CSVファイルの仕様

Wagby が扱うCSVファイルは次の仕様に準拠しています。

- Comma-Separated Values, Wikipedia, http://ja.wikipedia.org/wiki/Comma-Separated_Values
- RFC4130, http://www.ietf.org/rfc/rfc4130.txt

ダウンロード

項目の型に応じたルール

繰り返し項目

データの個数分項目名の後に番号が付与されます。 具体的には項目名[1]、項目名[2]・・・となります。

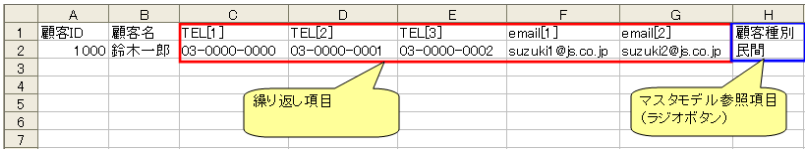


図 ダウンロードされたCSVファイルの例(1)

繰り返しコンテナ

データの個数分項目名の後に番号が付与されます。 コンテナ名[1]/項目名、コンテナ名[2]/項目名・・・となります。

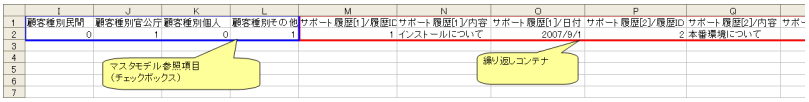


図 ダウンロードされたCSVファイルの例(2)

文字列型 !-- N20180315_163313 -->

改行コードが含まれている場合、改行コードは維持されて CSV/Excel ファイルに出力されます。[トラブルシューティング...]

数値型 !-- N20171207_194337 -->

数値フォーマットは適用されません。

日付・時刻・日付時刻型 !-- N20171207_194337 -->

CSV形式の場合、日付・時刻・日付時刻型のフォーマットは適用されます。

Excel形式の場合、日付・時刻・日付時刻型のフォーマットは "JDBCタイムスタンプ・エスケープ形式" となります。それぞれ "yyyy-mm-dd", "h:mm:ss", "yyyy-mm-dd h:mm:ss" です。

モデル参照（リストボックス、ラジオボタン、検索画面）

項目の内容部が表示されます。（上図の青枠部分） 登録がない場合は、データ部は空白で出力されます。

モデル参照（チェックボックス）

定義した項目名の後に参照内容が出力されます。二行目以降は、選択した内容の箇所に "1" が出力されます。 選択していない場合は "0" が出力されます。（上図の青枠部分）

同じ項目名が定義されていた場合

区別するため、日本語の項目名の後に ″(″ + 項目ID + ″)″ が出力されます。

H	I	J	K
顧客種別(customertype)	顧客種別(customertype2)民間	顧客種別(customertype2)官公庁	顧客種別(customertype2)個人
民間	0	1	0

図 ダウンロードされたCSVファイルの例(3)

項目名が重ならないような設定を行うことで、この動作を抑制できます。

モデル参照（チェックボックス）で、項目名と先頭部分が一致する（他の）項目が定義されていた場合

区別するため、項目名の後に ″(″ + 項目ID + ″)″ が出力されます。

H	I	J	K	L
顧客種別(customertype)民間	顧客種別(customertype)官公庁	顧客種別(customertype)個人	顧客種別(customertype)その他	顧客種別(その他)(customertype.other)
0	0	0	1	非常勤取締役をされている。

図 ダウンロードされたCSVファイルの例(4)

グループ権限機能を指定したモデル

グループ権限を設定したモデルには「データ所有者」を示すための項目 ″jgroupidjshparam″ が含まれます。この項目は Wagby 内部で必要になります。

iPadでダウンロードする場合

文字エンコーディングの指定

iPad上のブラウザでWagbyアプリケーションを利用する場合、CSVファイルの文字エンコーディングを″UTF-8″に変更してください。

iPadでは、ダウンロードしたCSVファイルをEvernoteやNumbersなどの(CSVファイルを扱える)他のアプリケーションと連携することにより、保存や編集を行うことが可能となります。

この設定を行うと、アプリケーション内のすべてのCSVダウンロードの文字エンコーディングがUTF-8となります。通常、ExcelはUTF-8形式のCSVファイルに対応していないため、iPadでは表示できるようになりますが、Excelで開くときには一工夫、必要になります。（検索エンジンにて ″excel utf-8 csv″ で探すと、多くのアドバイスを見つけることができます。）

ブラウザの戻るボタンの利用

iPadからCSVダウンロード機能を利用する場合、ダウンロード画面からCSVファイルを表示した画面に遷移するためにブラウザの「戻るボタン」を押す必要があります。[353]

多言語対応

CSV ファイルの文字エンコーディングの標準は Windows-31J となっています。これは日本語環境のみに対応しています。

多言語のデータを含む場合、このモデルの文字エンコードを utf-8 に変更してください。

ダウンロードしたファイルを Excel で開くときには注意が必要です。Excel は utf-8 形式の CSV ファイルに対応していないため、工夫が必要です。検索エンジンにて ″excel utf-8 csv″ で探すと、多くのアドバイスを見つけることができます。

[353] Wagbyではブラウザの「戻るボタン」の利用は非推奨ですが、この場合だけ例外扱いとなります。

トラブルシューティング

テキストエリア項目をExcelに出力したとき、改行コードが消えているようにみえる。

Excel (XLSやXLSX形式)では、列の高さがデフォルトの1文字分の高さとなっているため、セルに改行がないようにみえます。セルにカーソルを合わせると、改行が含まれていることが分かります。

このテキストエリア項目に対応したセルに、セルの書式設定 - 配置 - 文字の制御にある「折り返して全体を表示する」を設定すると改行を含めてセルが表示されるようになります。

IE11利用時、ダウンロード時の保存、ファイルを開くボタンを素早く押すと"ファイルが移動または削除された可能性があります"というダイアログが表示される。

ファイルのダウンロードが終了する前にエクセルファイルが立ち上がっている可能性があります。「ファイルを開く、保存、キャンセル」のダイアログは IE が出力するもので Wagby では制御することができません。このようなメッセージが出力されたあと、数秒ほど待って実行すると問題ないようです。

参考:Excelが提供する「データ取り込み」機能の使い方

1. Excel を開きます。メニューバーの「データ」-「外部データの取り込み」-「データの取り込み」をクリックします。

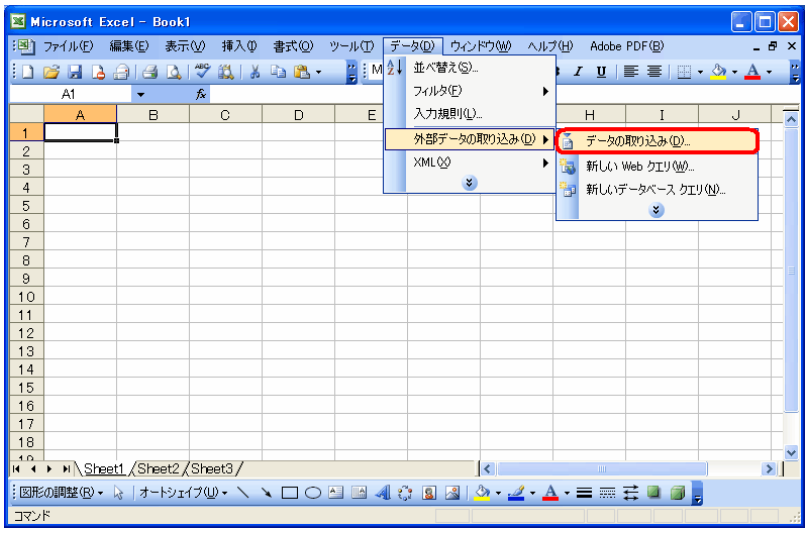


図 Excelを開きデータを取り込む

2. 「データファイルの選択」で任意の CSV（TSV）ファイルを選択します。

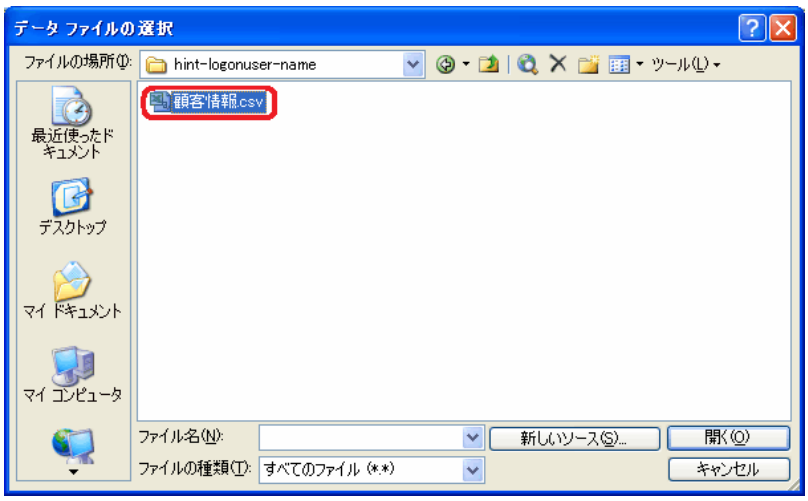


図 Excelを開きデータを取り込む

3. 「テキストファイルウィザード」でデータを確認します。

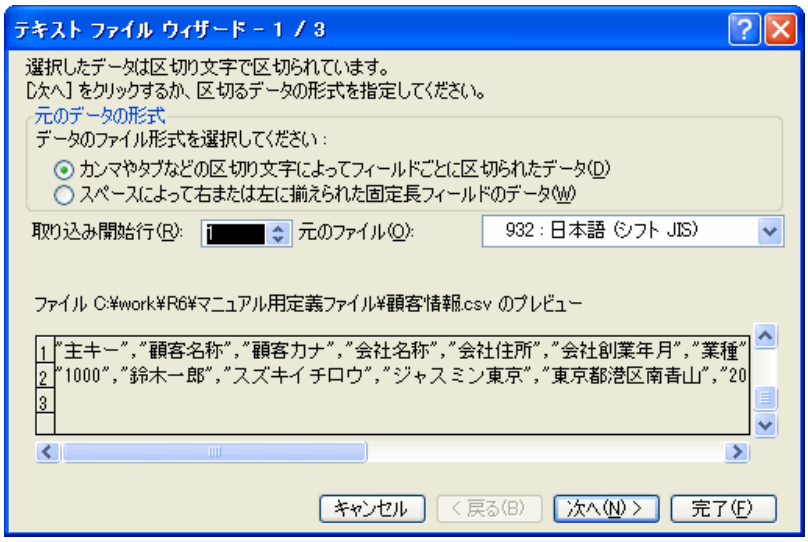


図 テキストファイルウィザード(1)

4. 「区切り文字」で「カンマ」を指定します。（TSV　ファイルの場合は「タブ」を指定。標準で指定されています。）

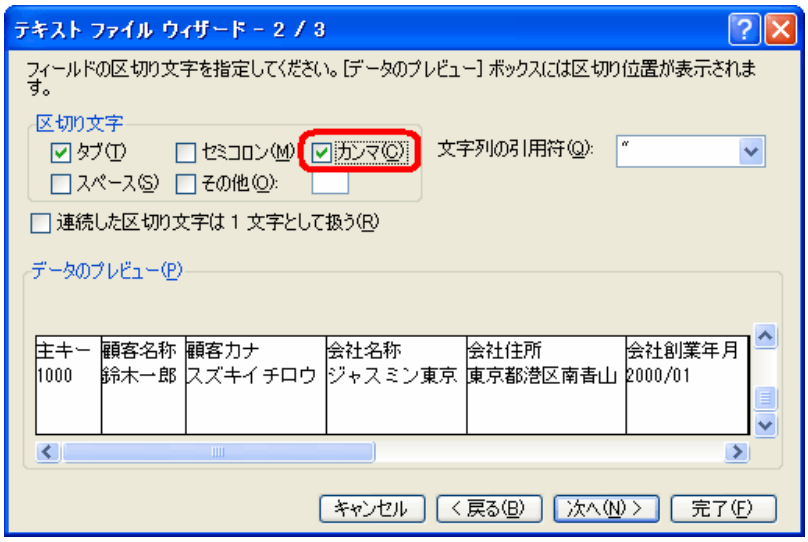


図 テキストファイルウィザード(2)

5. 「データのプレビュー」で指定する列を選択し、「列のデータ形式」を指定します。（ここでは表示形式をそのまま表示するよう「文字列」と指定しました。）

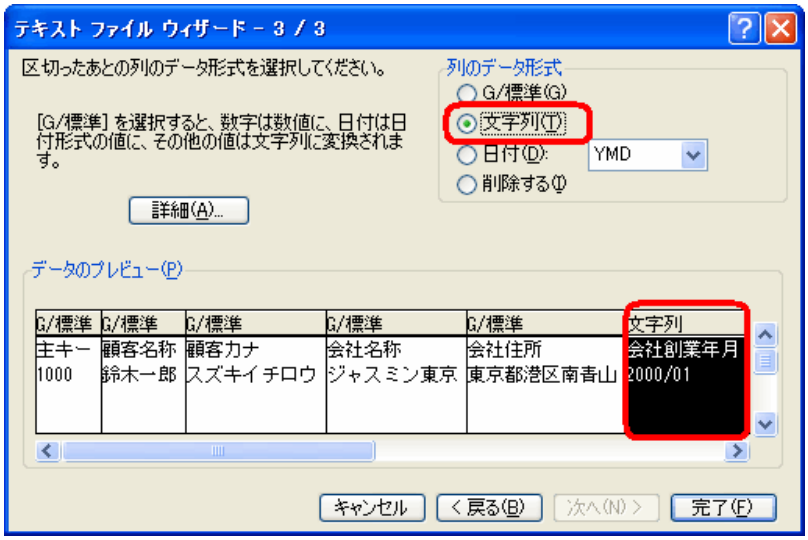
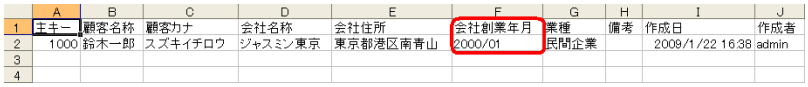


図 テキストファイルウィザード(3)

6. データ取り込み完了です。指定した表示形式で表示されていることが分かります。



	A	B	C	D	E	F	G	H	I	J
1	主キー	顧客名称	顧客カナ	会社名称	会社住所	会社創業年月	業種	備考	作成日	作成者
2	1000	鈴木一郎	スズキイチロウ	ジャスミン東京	東京都港区南青山	2000/01	民間企業		2009/1/22 16:38	admin
3										
4										

図 データ取り込み完了

仕様・制約

対応していない型

参照連動型項目の参照先が繰り返し、およびチェックボックスの場合には未対応です。（出力されません。）

出力フォーマットは適用されない

数値や日付の出力フォーマットは適用されません。CSVの場合、Wagbyからの出力は素の状態となっています。別途、ETLツールなどを使って整形するようにしてください。Excelの場合は、Excelのセルのフォーマットが適用されます。

(354)

子モデル操作の制約

データベース非保存項目で、自モデル（親）に紐づく子モデルの値を取得する計算式は処理されません。例えば親モデルのDB非保存項目item1の式がCOUNT(${child_lp})であった場合、この項目は常に0になります。この理由は、親モデルの1データのダウンロード処理で、紐づく子モデルをすべて読み込むとパフォーマンス低下につながるためです。

代替案として次の方法を行なってください。

- データベース保存項目とする。
- SQL式を使って実現できるかどうかを検討する。
- 自動計算式の代わりにスクリプトを使う。スクリプト内で SQL もしくはサービスクラスを使ってデータを処理し、その値をセットするようにする。

(354) CSVで出力フォーマットを指定する場合の代替案として、計算式で TEXT(${item1},"##0.###") などと指定する方法があります。元の項目はCSV出力しないとします。

ダウンロード画面の設定

定義方法

ページタイトルと説明文を指定することができます。

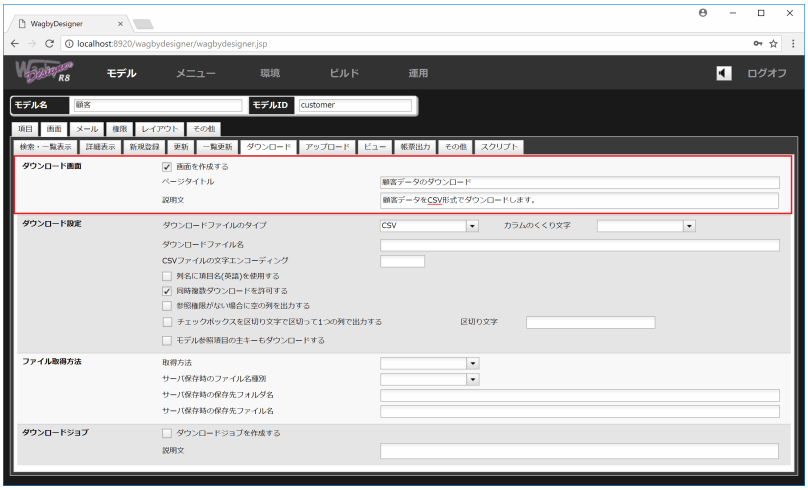


図 ダウンロード画面に関する設定

例

図の設定で生成された画面を図に示します。

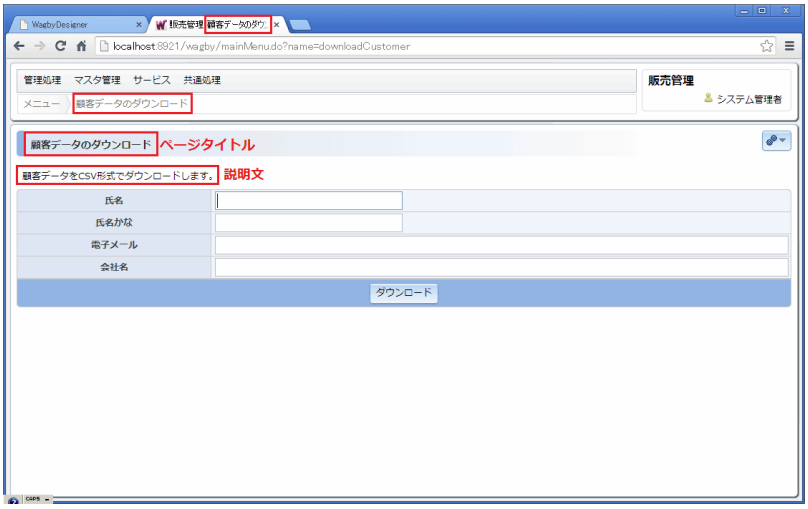


図 ページタイトルと説明文を設定した例

ダウンロード機能の設定

ダウンロード設定の詳細を説明します。

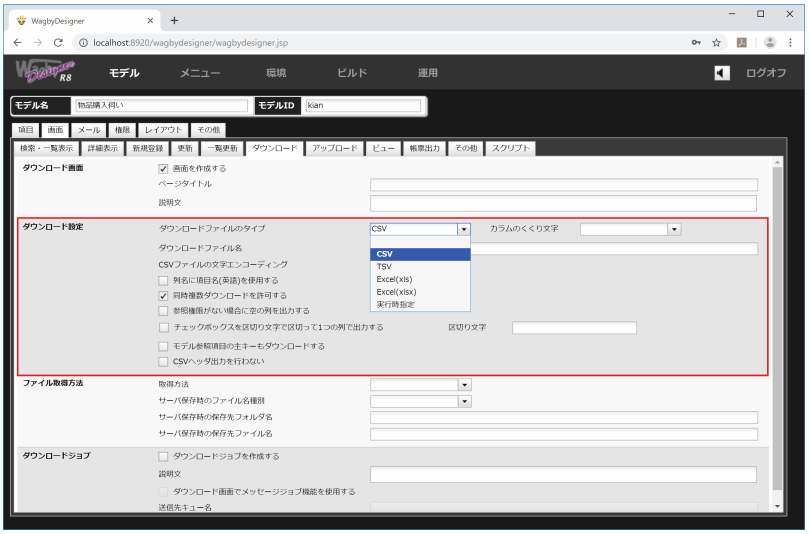


図 ダウンロード機能に関する設定

ダウンロードファイルのタイプ

"CSV（カンマ区切り）", "TSV（タブ区切り）", "Excel(xls)", "Excel(xlsx)", "実行時指定" が選択できます。

ダウンロードファイル名

ダウンロードするファイルの名前を任意に指定することができます。　詳細は"ダウンロードファイル名の設計"をお読みください。

CSVファイルの文字エンコーディング

モデル毎にダウンロードファイルの文字エンコードを指定することができます。この値が空白のときは「環境 ＞ アプリケーション ＞ 初期パラメータ ＞ CSVファイルの文字エンコーディング」の値が用いられます。

例えば UTF-8 で出力したい場合はこの欄に utf-8 と記述します。

列名に項目名(英語)を使用する

チェックすると、ヘッダ部の項目名が項目IDに変わります。

同時複数ダウンロードを許可する

同時に複数の利用者がダウンロードを行うことを許可します。本設定は標準で有効になっています。

ご注意ください：ダウンロード処理は一時的にサーバのメモリを消費します。

参照権限がない場合に空の列を出力する

参照権限がない項目は、項目名およびデータは表示されません。しかし本設定を有効にすると権限がない場合でも項目名は出力されるようになります。データは空白で出力されます。

チェックボックスを区切り文字で区切って1つの列で出力する

チェックボックス型ではデータ項目に対して "1" または "0" が出力されます。 本設定を有効にすると、データを一つにまとめて出力します。データ同士の区切り文字は "," を使います。例えば "官公庁, その他" のように出力します。

チェックボックスを区切り文字で区切って1つの列で出力する - 区切り文字

上述した「チェックボックスを区切り文字で区切って1つの列で出力する」の区切り文字を指定することもできます。空白時は ″,″（カンマ）が用いられます。

モデル参照項目の主キーもダウンロードする

モデル参照の項目は内容部が出力されます。本設定を有効にすると参照先モデルの主キーも同時に出力することができます。 主キーの項目は ″項目名#id″ となります。

> アップロード更新でも、#id 列を用いてモデル参照項目の値を変更することができます。

カラムのくくり文字

標準ではダウンロードファイル中、カラムの前後にダブルクォーテーションが付与されます。本設定欄に「(なし)」を指定すると、ダブルクォーテーションを出力しません。本設定の未指定時（標準）はダブルクォーテーションです。

> カラムのくくり文字を「なし」とした場合、データ中にダブルクォーテーションやカンマが含まれないようにしてください。含まれていた場合、CSVファイルを受け取った側で解釈エラーが生じます。

> カラムのくくり文字がないCSVファイルを、アップロード更新することもできます。この場合もデータ中にダブルクォーテーションやカンマが含まれないようにしてください。

CSVヘッダ出力を行わない (8.3.0)

出力するCSVファイルの先頭に含まれるヘッダ行を出力しません。先頭行からデータとなります。

ダウンロードファイル名の設計

空白時

ダウンロードファイル名を空白としたとき、ファイル名は次のルールに従います。

```
ログオンアカウント_data_日付時刻.拡張子
```

固定ファイル

固定ファイル名を指定することができます。拡張子は不要です。ファイル名は次のようになります。

```
固定ファイル名.拡張子
```

プレースホルダ

検索条件をプレースホルダとして含めることができます。″モデル名_cp.項目名″ となります。拡張子は不要です。 例を示します。

```
${report_cp.title}
```

プレースホルダと他の文字を混在することもできます。

```
${report_cp.title}_${report_cp.type}_ファイル
```

式

ファイル名の設定欄に式（関数）を含めることはできません。そこで次のようにします。

- 検索条件に、隠し項目を用意する。
- 同項目の初期値（検索時）に式を用いてファイル名を決定する。
- この項目をプレースホルダとして用いる。

サンプル

サンプルのモデル定義を図に示します。項目ID "csvfilename" は、ダウンロードするファイル名が格納される項目とします。

図 ダウンロードファイル名を保持するサンプルモデル

この項目はデータベースに保存しません。

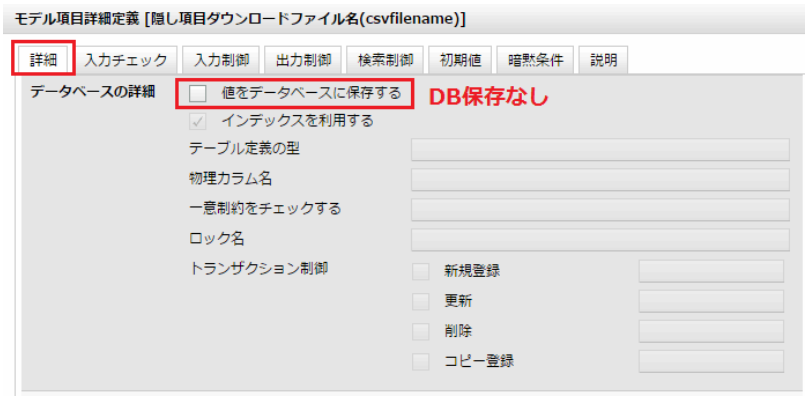


図 データベース非保存とする

この項目は入力時に隠し項目とします。

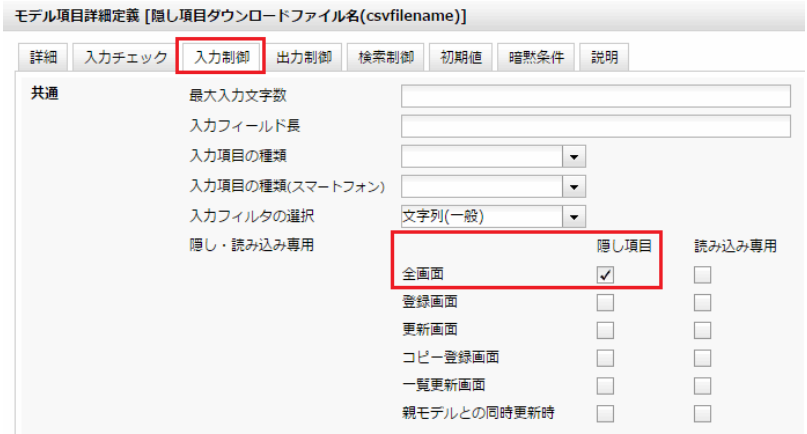


図 入力時に隠し項目とする

この項目は出力時にも隠し項目とします。

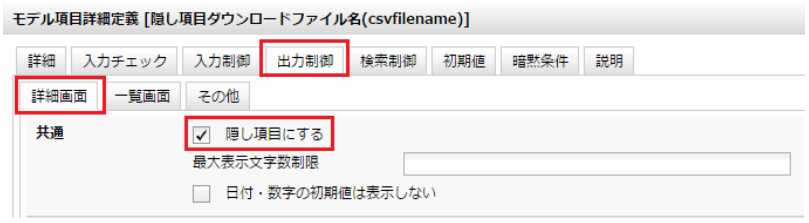


図 出力時に隠し項目とする

この項目は検索時にも隠し項目とします。

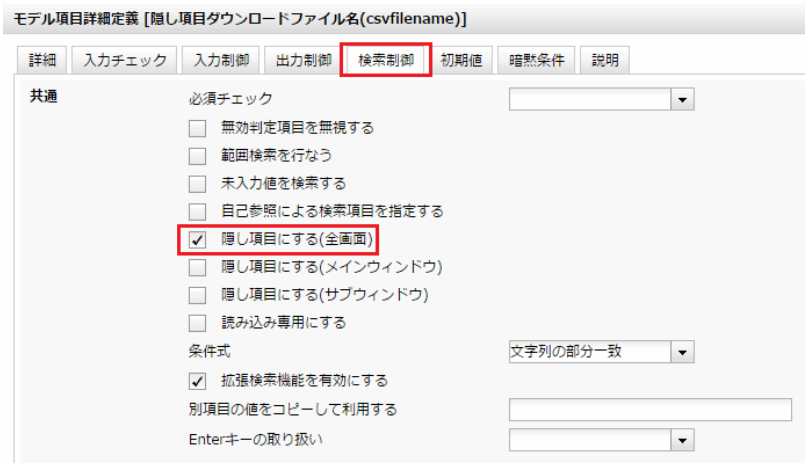


図 検索時に隠し項目とする

検索時の初期値に式を指定します。これがファイル名になります。

次の例は、検索条件時のname項目が空であればログオンアカウント名を使います。そうでなければ検索条件欄に入力されたname項目の値を使います。その後ろに日付を "yyyy-MM-dd" 書式で加えるものです。

```
IF(ISBLANK(${name}),USERNAME(),${name})+TEXT(TODAY(),"yyyy-MM-dd")
```

図 式の設定

「画面 > ダウンロード > ダウンロード設定 > ダウンロードファイル名」には、この隠し項目を指定します。

```
${customer_cp.csvfilename}
```

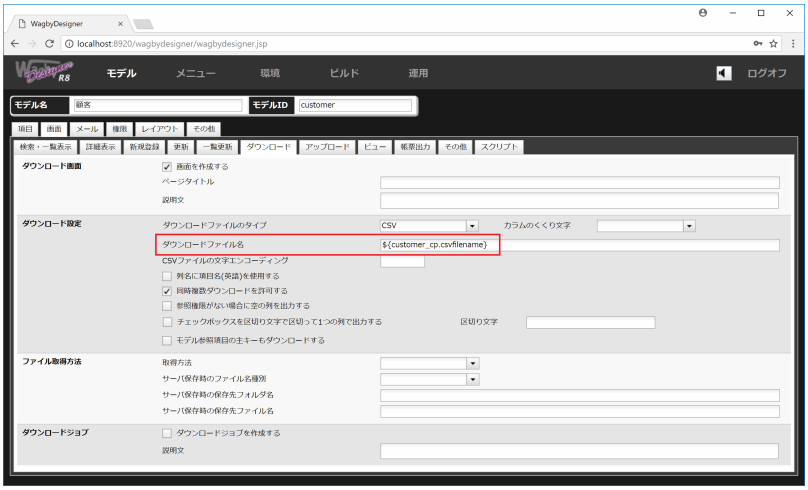


図 ダウンロードファイル名の設定

項目の設定

ダウンロードの対象に含める

各項目をダウンロードの対象に含めるかどうかを個別に指定できます。

「モデル項目詳細定義>出力制御>その他」の「CSVに出力する」を制御します。標準では有効となっています。

図 項目の出力設定

並び順

出力されるデータの並び順は、次のようになります。

	R8.3.1以降	R8.3.1より前
初期状態	リポジトリで指定したソート順に従う。	
利用者がブラウザでソート順を指定したとき	指定が反映される。	反映されない。

繰り返し項目・繰り返しコンテナ項目の扱いに関するルール

モデルに繰り返し項目および繰り返しコンテナが含まれる場合、これらのデータを独立してダウンロードすることができます。

図は、繰り返し項目「電子メール」ならびに繰り返しコンテナ項目「商談履歴」を含むモデルです。 検索条件下部の「ダウンロード形式」で出力する内容を選択することができます。

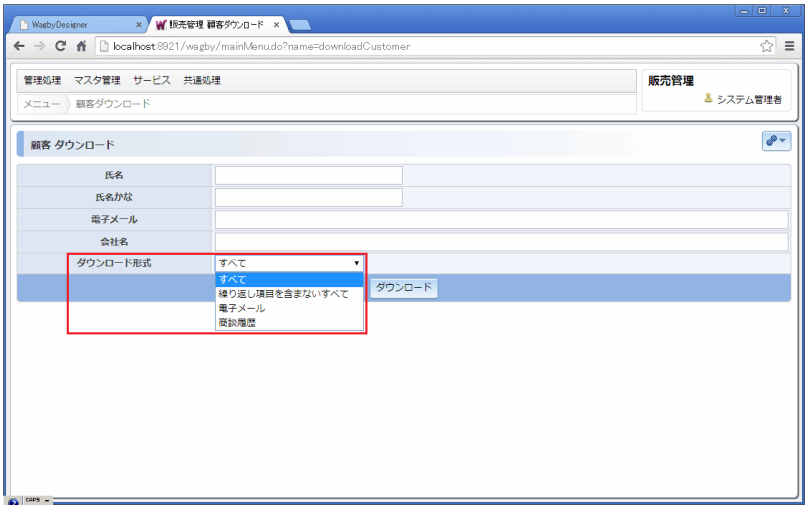
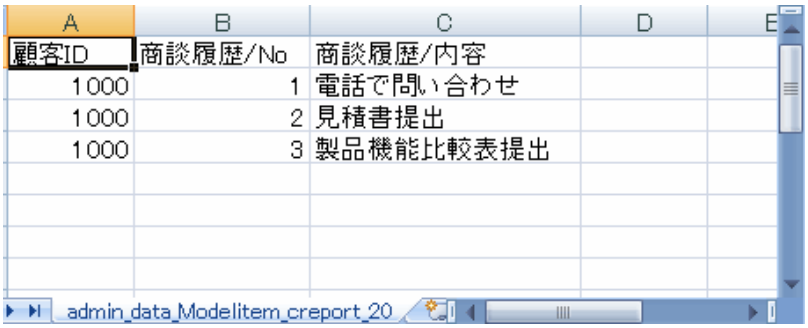


図 繰り返し項目を含むモデルのダウンロード画面

繰り返しコンテナ「商談履歴」をダウンロードし、Excelで開いた例を図に示します。



	A	B	C	D	E
1	顧客ID	商談履歴/No	商談履歴/内容		
2	1000	1	電話で問い合わせ		
3	1000	2	見積書提出		
4	1000	3	製品機能比較表提出		

admin_data_Modelitem_creport_20

図 繰り返しコンテナ「商談履歴」をダウンロードした例

"すべて" をダウンロードした場合、繰り返し項目・繰り返しコンテナ項目がすべて一行にまとめて出力されます。 繰り返し項目や繰り返しコンテナ項目を独立してダウンロードすると、行毎に展開して出力されます。

ファイル名の命名規則は次のようになります。

```
ログオンアカウント_data_Modelitem_繰り返し項目ID_日付時刻.拡張子
ログオンアカウント_data_Modelitem_繰り返しコンテナ項目ID_日付時刻.拡張子
```

ファイル名の命名規則を変更することはできません。

「ダウンロード形式」を表示しない

モデルに繰り返し項目または繰り返しコンテナがあり、かつ、これらがダウンロード対象となっていた場合、「ダウンロード形式」は必ず用意されます。これを非表示とする設定はありません。

そのため、この項目を非表示とするためには次のような方法で回避してください。

- 繰り返し項目または繰り返しコンテナ項目をダウンロード対象にしない。
- これらを除いたサブモデルを作成する。

ダウンロードファイルの取り扱いに関するルール

CSV, TSV ファイルをExcelで開く場合

CSV（カンマ区切り）,TSV（タブ区切り）ファイルをダブルクリックするとExcelが起動する環境では、次の問題が生じることがあります。

- 先頭0詰めの数字で0が抜け落ちる。
- 任意の文字列としての項目の内容がたまたま日付の形式に近いと、自動的に日付型に自動変換されてしまう。

この問題を回避する方法は次のとおりです。

- ファイルのアイコン上で右クリックし、「プログラムから開く」から「Microsoft Office Excel」を指定する。
- 前もってExcelを起動しておき、開きたいファイルをドラッグ&ドロップする。
- Excelが提供する「データの取り込み」機能を使う。[後述]

Excelファイル

Excel形式でダウンロードしたファイルは、型の情報が反映されます。

- 文字列型、数値型、日付型は Excel のセルの型に反映されます。
- 数値型、日付型のフォーマット指定は、Excel に反映されません。（Wagbyの指定方法とExcelの指定方法が同一ではないため）
- モデル参照型の項目は、すべて文字列型として扱われます。
- 参照連動型の項目は、参照先の型がExcel のセルの型に反映されます。

> Excel形式でダウンロードする場合、一時的に大量のメモリを消費します。また、データの作成に非常に時間がかかることがあります。そのため、データ量によってはダウンロード処理が失敗する場合があります。大量データを扱う場合はダウンロードするデータを（検索によって）絞込むようにしてください。または Excel 形式ではなく、CSV 形式に代えて運用してください。

.xls と xlsx の違い

拡張子 xls 形式を選択した場合、256列までのデータ項目を扱うことができます。（256列を超える部分は無視されます。） 取り扱うことができるデータ数の上限は 65,536 行です。

拡張子 xlsx 形式を選択した場合、16,384列までのデータ項目を扱うことができます。 取り扱うことができるデータ数の上限は 1,048,576 行です。

権限管理機能との組み合わせ

項目に権限設定を行った場合は次のように動作します。

- 項目名の出力は、権限のみが判定材料となります。条件付き権限を設定されていた場合も条件部は無視されます。
- 値の出力は、条件部に応じて出力可否が判断されます。条件部がなく固定で権限が設定されていた場合は、指定された権限が適用されます。

アップロード更新と併用する場合の注意点

条件付き閲覧権限を設定したモデルのデータをCSV形式ファイルでダウンロードした場合、これを編集してアップロード更新する運用には配慮が必要です。

以下のモデル定義を例に説明します。

項目	説明
id	主キー
a	
b	条件付き閲覧権限を指定。項目aが"0"の場合にのみ閲覧可

次のようなデータを想定します。

```
データ1:id:1000 a:0 b:aaa
データ2:id:1001 a:1 b:bbb
```

権限を有したアカウントでログオンし、このモデルのデータをダウンロードすると、下記のようなCSVとなります。

```
(id),(a),(b)
1000,0,aaa
1001,1,
```

条件付き閲覧権限によって、"id=1001" のデータの b 列は、空になっています。（条件を満たさないため閲覧不可）

この状態で、"id=1001" のデータの a 列を 1 から 0 に修正し、CSVアップロード更新します。 このとき、列 b の値が（本来、データベースには保存されている "bbb" から）"から文字" に更新されるということを配慮する必要があります。

この問題への対応は次のように行ってください。

- 条件付き権限を含むモデルのアップロード更新を行う場合には、常に管理者アカウントで実行する。
- 条件付き権限を含む項目は、CSVダウンロードの対象としない。

一覧表示画面で直接、ダウンロードする

検索・一覧表示画面で、検索結果をすぐにダウンロードすることができます。 詳細は"画面機能 > 検索・一覧表示（一覧表示）> ダウンロード"をお読みください。

サーバにファイルを保存する|8.0.2

生成したCSVファイルを（Wagbyが稼働する）サーバの所定フォルダに保存することができます。「画面 > ダウンロード > ファイル取得方法」で設定します。

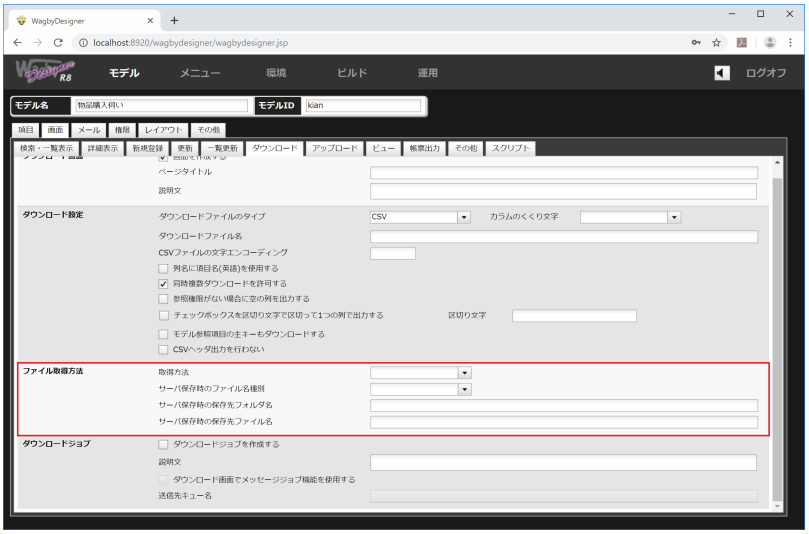


図 ファイル取得方法に関する設定

設定欄	説明	記述例
取得方法	"download" と "local" を選択できる。標準は "download" だが、"local" にするとサーバに保存される。	-
保存先ファイル名	取得方法が "local" の場合に有効。保存するファイル名。保存先は本番運用サーバ。 指定がない場合は wagbyapp¥tempフォルダに、download_xxx.csvというファイル名で出力される。（xxxはランダムな文字列。）	C:¥Downloads¥YOURFILE.csv/ home/wagby/YOURFILE.csv
保存先フォルダ名	取得方法が "local" の場合に有効。保存先ファイル名	C:¥Downloads/home/wagby

	が指定されていない場合に用いる。指定されたフォルダ名に保存する。ファイル名は「ファイル名種別」の指定に準じる。	
ファイル名種別	ファイル名種別。未指定、temp、filename、extを指定できる。 未指定、tempの場合は一時ファイル名として、download_[ランダムな文字列].csvとなる。必ず一意になる。現状のダウンロードジョブと同様。tsvやxlsなどcsvでない形式を選択しても、拡張子は.csvとなる。filenameの場合はDesignerで設定されたファイル名で保存する。拡張子も指定した形式に対応して設定される。 extの場合は拡張子はfilenameと同様に指定した形式に対応して設定されるが、拡張子を除くファイル名はtempと同様に必ず一意になるファイル名が設定される。	tempfilenameext

実行時にダウンロード項目を指定する

例

この機能は「一覧表示（グリッド利用）」で動作します。同画面に「ダウンロード」ボタンを用意することができます。

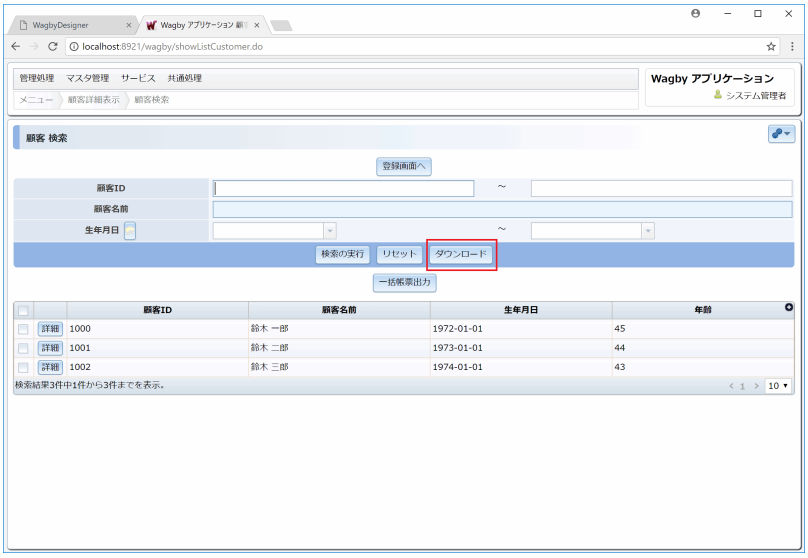


図 一覧表示（グリッド利用）画面にダウンロードボタンを用意する

グリッドの機能として、表示されている列を、さらに絞り込むことができます。

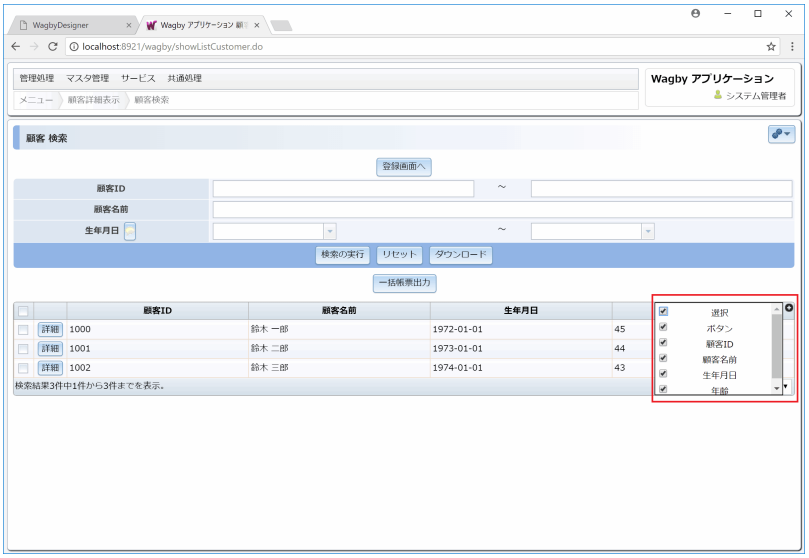


図 表示されている列を絞り込む

同じくグリッドの機能で、列の並びを（ドラッグ&ドロップで）変えることができます。

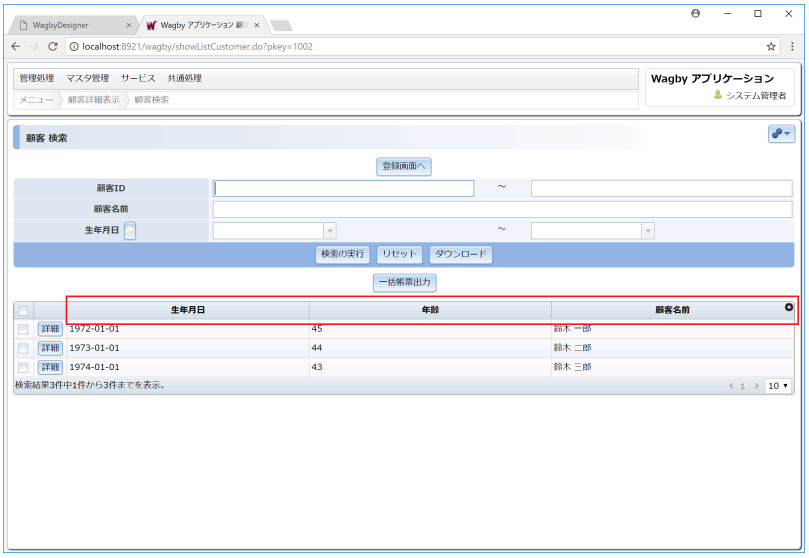


図 列の並びを変える

ここで「ダウンロード」ボタンを押下すると、画面の見た目（表示されている列、およびその並び）にあわせて、データがダウンロードされます。

定義方法

対象モデルの「画面 > 検索・一覧表示」で、「グリッドで表示する」を有効にします。（このとき「検索画面と一覧表示画面を同時に表示する」も自動的に有効になります。）

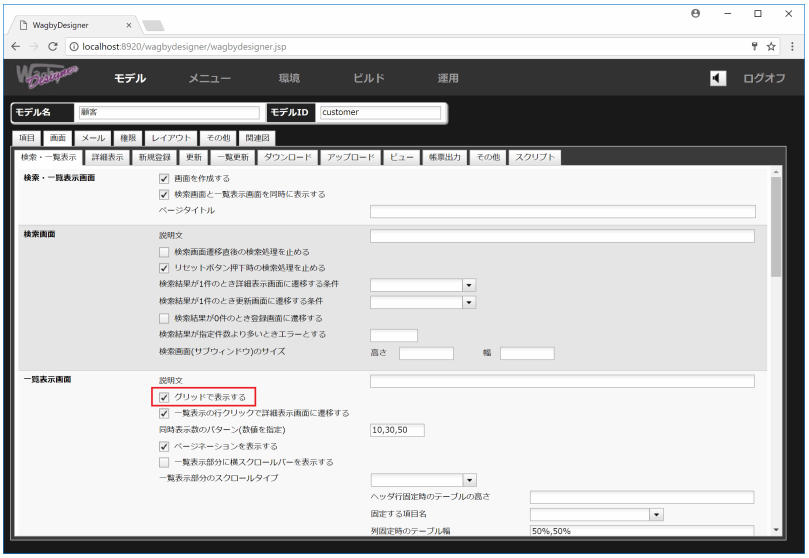


図 グリッドで表示する設定を有効にする

同じ「画面 > 検索・一覧表示」内にある、「ダウンロード」ボタンを有効にします。あわせて「グリッド表示項目のみダウンロード対象とする」を有効にします。

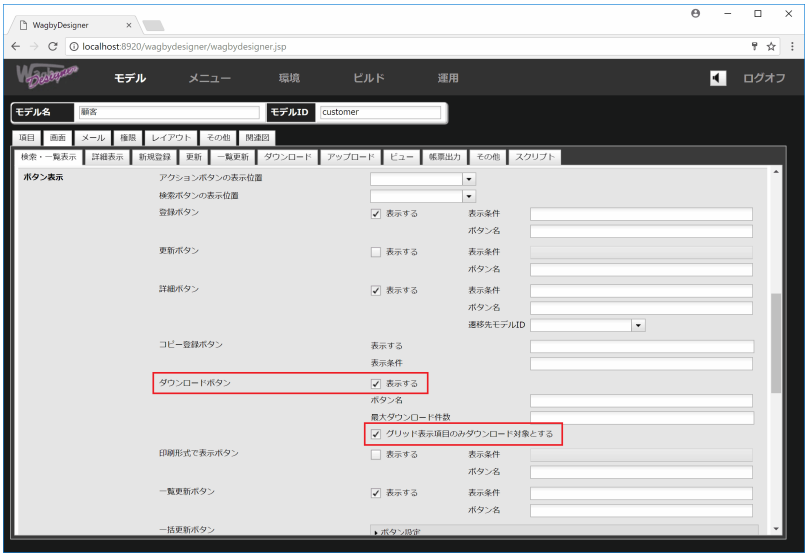


図 ダウンロードボタンを有効にする

「画面 > ダウンロード」タブに移ります。「画面を作成する」を有効にします。

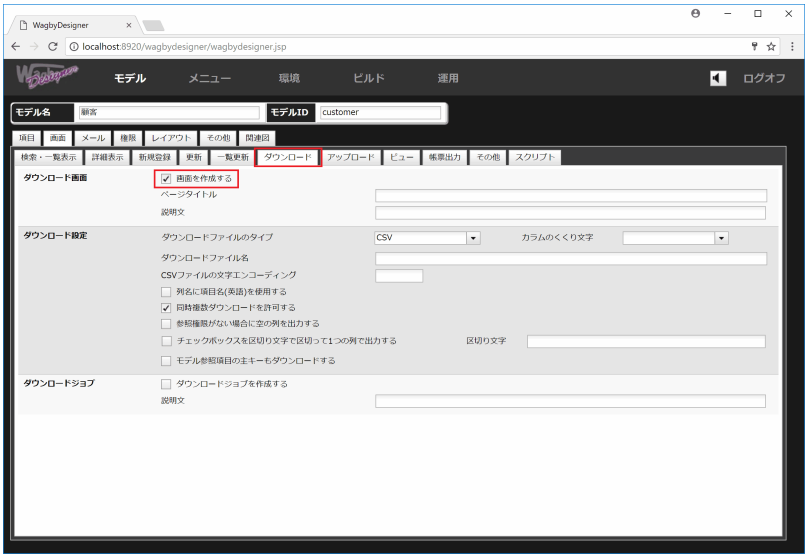


図 ダウンロード画面を作成する

ダウンロード画面を作成すると、自動的にメニューに追加されます。これが不要の場合は、メニューから削除してください。

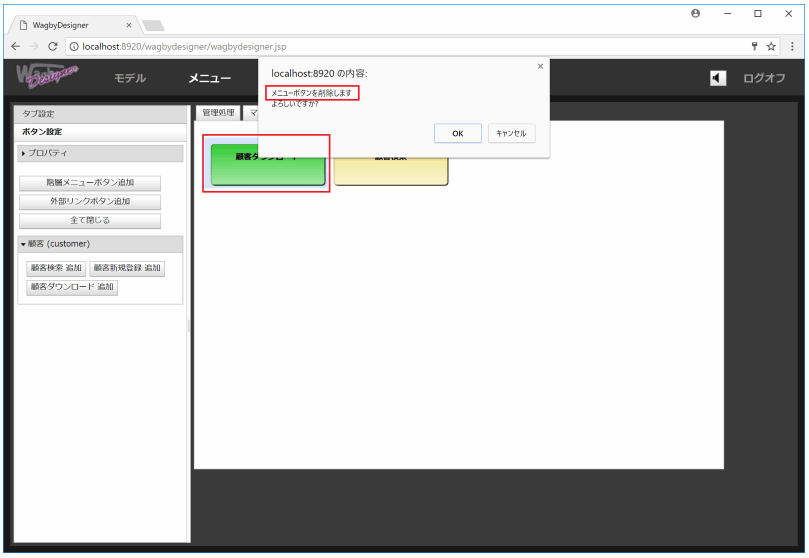


図 メニューから「ダウンロード」ボタンを削除する

設定を共通で利用する

グリッドの設定（列の表示・非表示と、並び）を異なる利用者間で共通に利用することができます。この機能を用いて、ダウンロードパターンをあらかじめ複数、用意しておくことができます。

仕様・制約

ダウンロードファイル形式

ダウンロードファイル形式は「画面 ＞ ダウンロード」の指定に準拠します。標準は「CSV」です。これを「実行時指定」にすると、選択用のラジオボタンが用意されます。

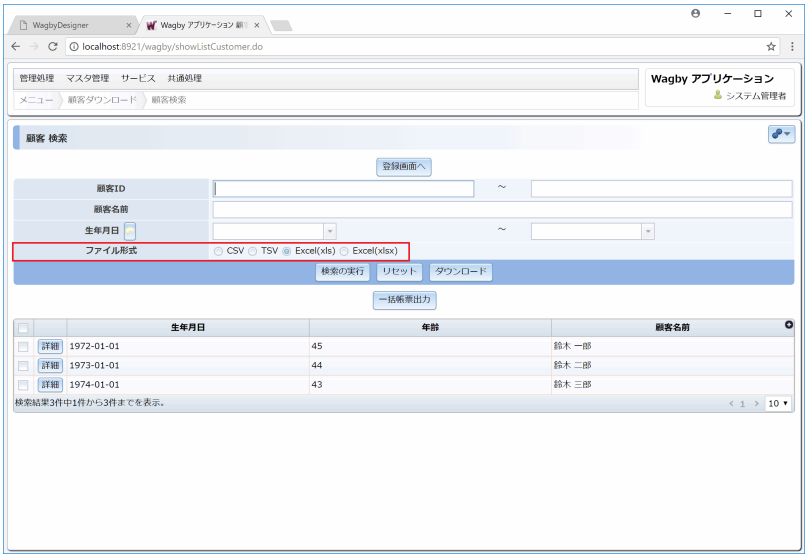


図 実行時にファイル形式を指定する

繰り返しコンテナ内の項目を一覧表示に含めた場合の動作

繰り返しコンテナ内の項目を一覧表示に含めると、本ダウンロード機能の対象となります。

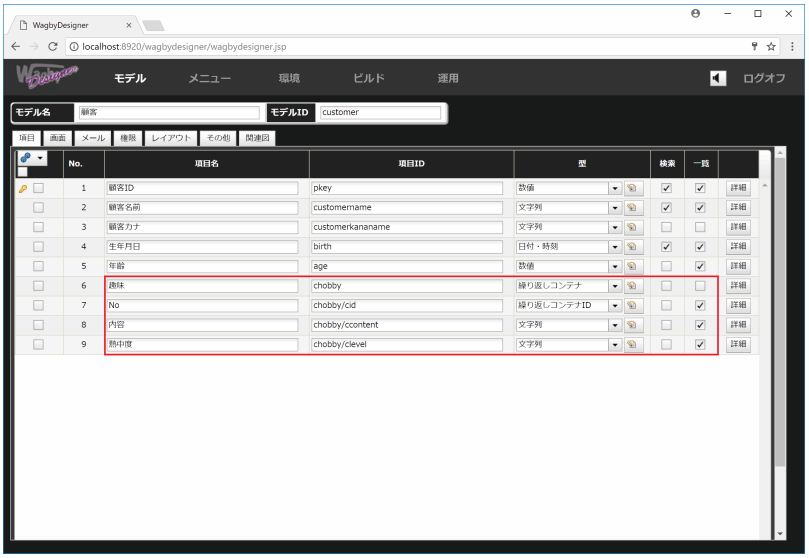


図 繰り返しコンテナ「趣味」を用意する

データを入力した例です。繰り返しコンテナに複数行のデータがあった場合、一覧表示画面ではコンマで区切って表示されます。

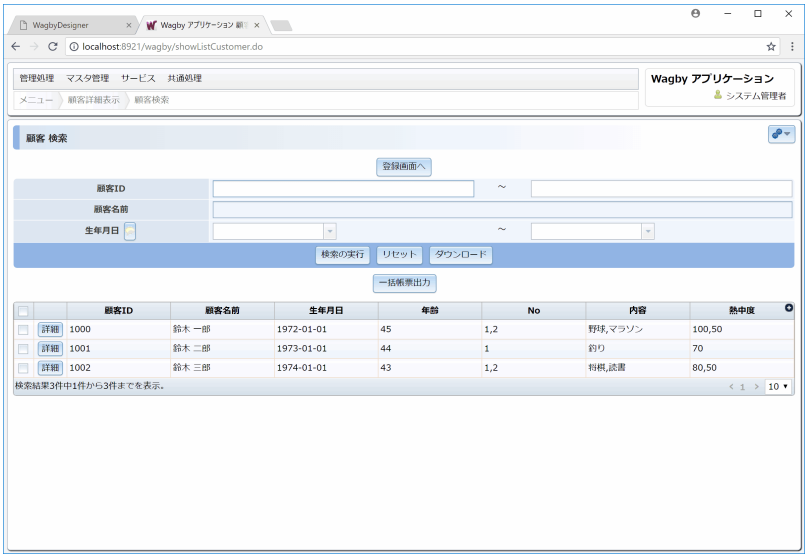


図 一覧表示の例

ダウンロードすると図のような列の並びになります。コンテナ内の項目単位で、複数行が展開されます。

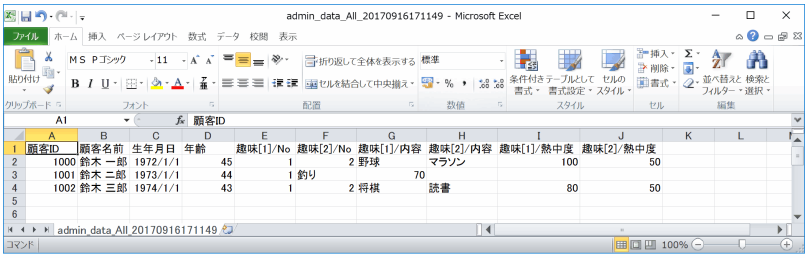


図 図のデータをCSV形式でダウンロードしたとき

繰り返しコンテナの列を移動させた例を示します。

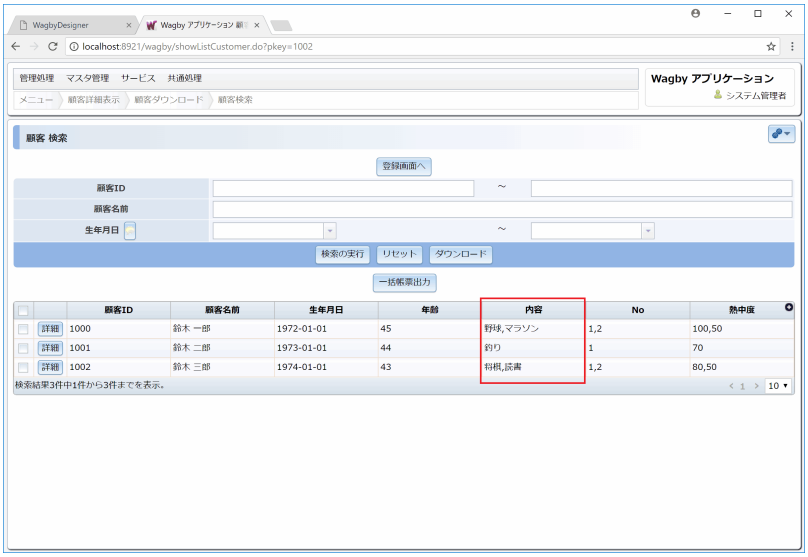


図 繰り返しコンテナの列「内容」を移動させた

ダウンロードすると図のような列の並びになります。

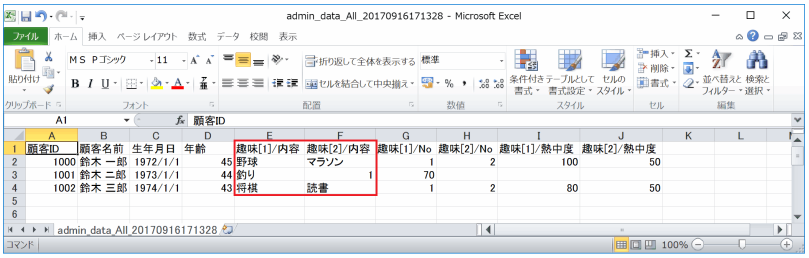


図 図のデータをCSV形式でダウンロードしたとき

未対応の機能

- 「繰り返しコンテナのみダウンロード」機能には対応していません。

ソート

- 一覧表示グリッド形式に用意したダウンロード機能で、ソート状態を反映させたい場合は、「グリッド表示項目のみダウンロード対象とする」を有効にしてください。

アップロード更新

処理の詳細

アップロード更新機能は、1行目にヘッダ（項目名）が含まれていることが前提となります。ヘッダはダウンロード機能で出力されます。

更新

- 列の並びを変更することができます。ヘッダ部を含めて、並びを変更してください。
- 特定の列を削除すると、その列を更新の対象から除外することができます。（データベースの値がそのまま維持されます。データベースの値が null の場合も、null のままとなります。）
- アップロードするデータに含まれる主キーが一致するデータが存在していれば更新となります。存在していなければ新規登録となります。
- アップロードするデータが空白の場合、新規登録画面や更新画面で空欄として入力することと同様の扱いとなります。必須項目の場合はエラーとなります。非・必須項目の場合は null で上書きされます。
- 新規登録時に、対象モデルの主キーに順序を適用していた場合、主キーを自動設定することができます。詳細は以下の説明をお読みください。

新規登録

対象データの主キーが順序を利用する設定の場合、データの主キー項目を "-1" にすることで順序が適用されます。（自動採番処理）

	A	B	C	D	E	F	G	H	I
1	主キー	顧客名称	顧客カナ	会社名称	会社住所	会社創業日	業種	備考	作成日
2	1000	鈴木一郎	スズキイチロウ	ジャスミン開発	沖縄県うるま市州崎		民間企業		2008/8/15 14:00
3	1001	山田花子	ヤマダハナコ	沖縄県	沖縄県那覇市		地方公共団体		2008/8/15 14:01
4	1002	宮城勝也	ミヤギカツヤ	那覇経済大学	沖縄県那覇市		民間企業		2008/8/15 14:02
5	-1	田中一郎	タナカイチロウ	ABC商事	沖縄県浦添市	2000/5/15	民間企業	名刺交	2008/8/15 14:02
6	-1	佐藤次郎	サトウジロウ	東京第一出版	東京都港区	1995/4/1	その他		2008/8/15 14:02
7	-1	渡辺三郎	ワタナベサブロウ	大阪グリーンワー	大阪府北区	2005/10/1	NPO	支所隹	2008/8/15 14:02
8									

図 主キー項目に「-1」を指定

(355)

(355) 主キー項目を用意することができない場合、「主キー列の存在をチェックする」機能を無効にすることで対応できます。この場合、アップロードデータはすべて新規登録扱いになります。

削除

データの削除を行うには、アップロードするファイルに「<Status>」という列を追加します。 図では、主キーが 1007 のデータを削除するように指定しています。[356]

	A	B	C	D	E	F	G	H	I	J
1	<Status>	主キー	顧客名称	顧客カナ	会社名称	会社住所	会社創業日	業種	備考	作成日
2	n	1000	鈴木一郎	スズキイチロウ	ジャスミン開発	沖縄県うるま市州崎		民間企業		2008/8/15 14:0
3	n	1001	山田花子	ヤマダハナコ	沖縄県	沖縄県那覇市		地方公共団体		2008/8/15 14:0
4	u	1002	宮城勝也	ミヤギカツヤ	那覇経済大学	沖縄県那覇市		民間企業		2008/8/15 14:0
5		1003	田中一郎	タナカイチロウ	XYZ商事	沖縄県浦添市	2000/5/15	民間企業	名刺交換	2008/9/3 19:0
6	n	1004	佐藤次郎	サトウジロウ	東京第2出版	東京都港区	1995/4/1	その他		2008/9/3 19:0
7	d	1007	渡辺三郎	ワタナベサブロウ	大阪グリーンワーク	大阪府北区	2005/10/1	NPO	支所管理	2008/9/3 19:0
8	i	-1	山本太郎	ヤマモトタロウ	山本物産	福岡県	1990/4/1	民間企業	新規登録	
9										

図 <Status>列を追加する

<Status>列は自動では付与されません。必要に応じて手動で付与してください。

<Status>列にはデータを新規登録・更新・削除のいずれかを識別するための文字（命令）を設定することができます。

識別文字	処理	内容
i	新規登録	新規にデータを登録します。
u	更新	データを更新します。なお、主キーに対応するデータがない場合はエラーとなります。
d	削除	データの削除を行います。
n	なし	何も処理を行いません。
空文字	通常と同じ処理	主キーに対応するデータがある場合は更新、ない場合は新規登録を行います。

ロックとコミット単位

(Wagby標準の)悲観ロックを利用している場合、更新処理と削除処理で1件ずつ、ロックの取得と解放が行われます。 またコミットは1行(1データ)単位です。

[357]

エラーの扱い

アップロード更新の途中にエラーが発生した場合、エラーが発生した行はスキップされます。全体の処理は継続されるため、それ以降の行はアップロード更新されていきます。

[358]

読み取り専用項目の扱い

読み取り専用項目はアップロード更新の対象から除外されます。（設定によって変更することができます。）

入力チェック

アップロード更新では、リポジトリで指定した入力チェック（「警告」を含む）が適用されます。

データの形式エラー(必須チェックや、数値を入力すべきところに非数字を設定した等)が発見された場合、画面にエラーデータの行番号とエラーメッセージが表示されます。データは更新（登録）されません。エラーのないデータのみが更新（登録）されます。

---

(356) 識別文字は大文字、小文字のいずれも利用できます。

(357) このため、あるデータで何らかのエラーが発生した場合、このデータは更新されませんが、その他で正常更新されたデータはすべてコミットされます。全体をロールバックする機能はありません。

(358) 100件のアップロード更新で、50件目がエラーであり、それ以外の行はすべて成功となっている場合、99件はコミットされ、50件目となる1件だけがエラーとなります。

「警告」を指定した場合も本チェックは適用されます。つまり該当データはアップロード更新処理で失敗と扱われます。

親子モデル関係における、親モデルからみた子モデルの扱い

対象モデルに子モデルが紐づいており、自動計算式や入力チェックで子モデルの値を参照するような場合、アップロード更新では「子モデルなし」とみなして処理されます。

これはデータ1件を処理する毎にデータベースから関連する子モデル情報をすべて呼び出すことはパフォーマンスに影響を及ぼすための措置です。

親子モデル関係における、子モデルの単独アップロード !-- N20200110_155339 -->

子モデルを新規登録する場合、親モデルの存在チェックは行いません。そのため、親が存在しない子のデータを作成することができます。

(359)

モデル参照項目の扱い

ダウンロード機能の設定で「モデル参照項目の主キーもダウンロードする」を有効にすると、参照先モデルの主キーで更新することができるようになります。(360)

id　列と項目列の両方が設定された場合は、項目列（内容部）が優先されます。id　列の値で更新する場合、項目列を（手動で）削除してください。

モデル参照項目の参照先モデルが複合キーの場合

1. 上で説明した「モデル参照項目の主キーもダウンロードする」を有効にします。
2. id 列（「項目名#id」）と項目列の両方を含むファイルをダウンロードします。
3. 項目列を削除します。id列で更新するためです。
4. id 列の値に、複合キーを "SEP" で連結して設定します。例えば項目 pkey1 と pkey2 で複合キーを構成するモデルで pkey1 の値が 1000 、pkey2 の値が ABC だった場合、アップロードするファイルの id 列の値を "10000SEPABC" と記述します。

モデル参照項目の内容部に適用されるフィルタ

アップロード更新でも（画面入力と同様）フィルタが適用されます。　モデル参照項目の内容部文字列には、参照項目先のフィルタ設定が用いられます。(361)

ワークフロー動作中のデータ (8.4.0)

ワークフロー動作中のデータは、アップロード更新での更新と削除が行えません。（更新しようとした場合、"ワークフローの状態により更新(削除)できません。" というエラーとなります。）

制約 : 対応していない型

参照連動型項目の参照先が繰り返し、およびチェックボックスの場合には未対応です。（更新されません。）

ログオンセッションの数え方

CSV/Excelアップロード更新は処理の最初に内部でログオン処理を行なっています。これにより、ライセンス制限上のログオン数を超えた場合は注意が必要です。

(359) 親の存在チェックを行いたい場合は、スクリプトによるチェックを追加してください。（実行タイミングは "ヘルパ > 入力チェック" になります。アップロード更新だけでなく、登録時や更新時も同じスクリプトが実行されます。）

(360) モデル参照による更新はいくつかの注意事項があります。詳細はトラブルシューティングの節で説明します。

(361) 例えば「顧客名」が顧客モデルを参照している場合、顧客モデルの「顧客名」に設定されたフィルタが用いられます。

例えば1つのログオンアカウント(ユーザー)で同時に5モデルのアップロード更新を行った場合、ログオン数は6とカウントされます。

このとき、1アカウントのログオン数上限を2と指定していた場合でも、上記のような5モデル同時アップロード更新の運用は行うことができます。すなわち、ライセンスによるログオン数の上限いっぱいまで、1アカウントで（アップロード更新処理で）消費してしまうといったことが可能です。

(362)

トラブルシューティング

特定のデータが処理できない

次の点をご確認ください。

- データに不正な文字（特殊文字、外字など）が含まれていませんか。
- データが「一意制約チェック」に違反していませんか。
- CSV形式でダウンロードしたファイルをExcelで開き、その後、保存しましたか？ Excel の仕様で、前詰めに "0" を含むデータは、その "0" が消えてしまいます。例えば、実際のデータが "000123" だった場合、このCSVファイルをExcelで開き、保存すると "123" となります。この値が主キーだった場合、該当データが見つからないという問題が生じます。

これを防ぐために、Excel では「CSV取り込み」という機能があります。CSVファイルをダブルクリックするのではなく、ExcelのメニューからCSVファイルを読み込むようにすることでこの問題を回避できます。

特定の列が処理できない

特定の列だけ正しく処理されない場合、アップロードするファイルの先頭行の「列名」と、設計情報に記載した「列名」が正しく一致しているかどうかを確認してください。列名が完全に一致しない場合、その列の処理は行われません。

[事例] 新規登録が正しく処理されない

主キーに順序を利用したモデルについて、次のようなファイルをアップロードしました。なお、主キーは1000番から開始されるものとし、処理時にデータは1件も登録されていなかったとします。

```
pkey, name
1000, aaa
1002, bbb
1001, ccc
```

このとき、次のような動作となります。

1. データは1件も登録されていない状態ですので、1行目は新規登録となります。ここで "1000" という値は使われず、順序から自動採番されます。たまたま、同じ "1000" が発番され、1行目の登録は成功します。
2. 2行目も新規登録となります。ここで "1002" という値は使われず、順序から "1001" が発番され、登録されます。この時点でのデータベースの内容は次のとおりです。

```
1000,aaa
1001,bbb
```

3. 3行目はすでに "1001" というデータが登録済みのため、更新扱いとなります。よってデータベースの内容は次のように書き換えられます。

```
1000,aaa
1001,ccc
```

対策(1) : このような誤動作を防ぐため、順序を用いたモデルへの登録は主キー部分を "-1" と設定するとよいです。

(362) 無制限ライセンス利用時は（理論上の上限がないため）問題はありません。プロジェクトライセンス利用時は、上限を超えないような運用を行うようにしてください。

対策(2) ： アップロードするデータはあらかじめ主キーでソートしておくようにします。（例えばダウンロード機能でデータを取得する場合、主キーでソートさせたデータを使うといったルールを定めることで未然に回避できます。）

[事例] モデル参照項目の更新で、参照先の名称が異なっていた

例えば「会社」モデルを参照しているデータで、会社名に「山田商店」と記載していたが、実際の会社モデルにこのデータが存在しなかった場合、この項目そのものの更新が行われません。

[事例] モデル参照項目の更新で、無効なデータに更新しようとした

「選択肢モデル>選択肢を無効にする」設定で、あるデータを無効化（論理削除）していたとします。ここで、無効なデータに更新しようとした場合、更新エラーになります。

[事例] モデル参照(チェックボックス)型項目に対して必須入力チェックを指定している

すべてチェックしない状態でアップロード更新するとエラーとなります。一つ以上チェックしてアップロード更新するとエラーになりません。

[事例] 異なるコードに同じ内容（データ）を設定した

このようなデータが存在した場合は、無視またはエラー扱いとなります。例えば「性別」モデルに次のようなデータを設定したとします。

コード	値
1	男
2	女
3	男 !重複した内容

このとき、″女″ から ″男″ へのアップロード更新はエラーになります。コード1と3のどちらを使えばいいか判断できないためです。

画面上には、次のようなエラーメッセージが表示されます。

項目 XXX に指定された値 YYY で問題が発生しました。同じコードに複数の値が定義されています。そのためどのコードを更新するか特定できません。

[事例] データをアップロードしたとき、関係のないモデルも同時に更新される

アップロード対象が「子モデル」の場合、「画面 > その他 > モデルの関連性 > 変更時に親モデルも更新する」が有効の場合、親モデルも更新されることがあります。

さらに、その親モデルが他のモデルと関係性がある場合（例えば他の子モデルがあり、お互いを参照している）、関連するモデルも必要に応じて更新されることがあります。

アップロード更新画面の設定

定義方法

ページタイトルと説明文を指定することができます。

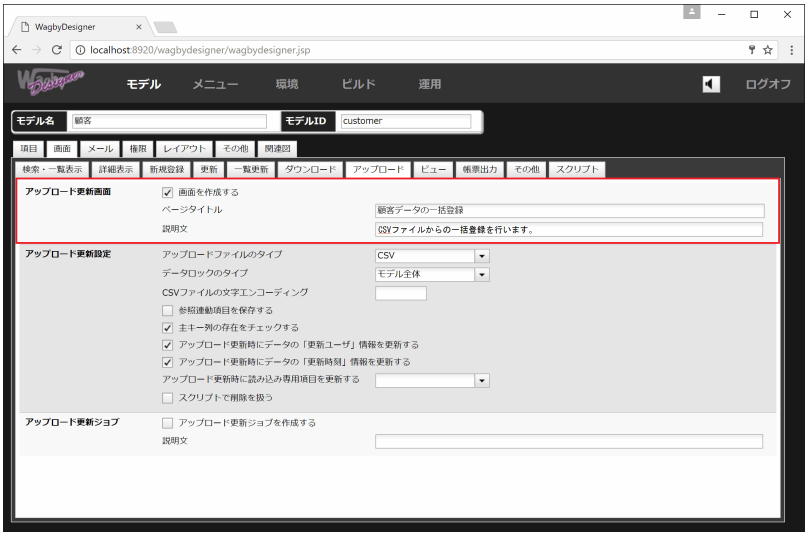


図 アップロード更新画面に関する設定

例

図の設定で生成された画面を図に示します。

図 ページタイトルと説明文を設定した例

アップロード更新機能の設定

アップロード更新設定の詳細を説明します。

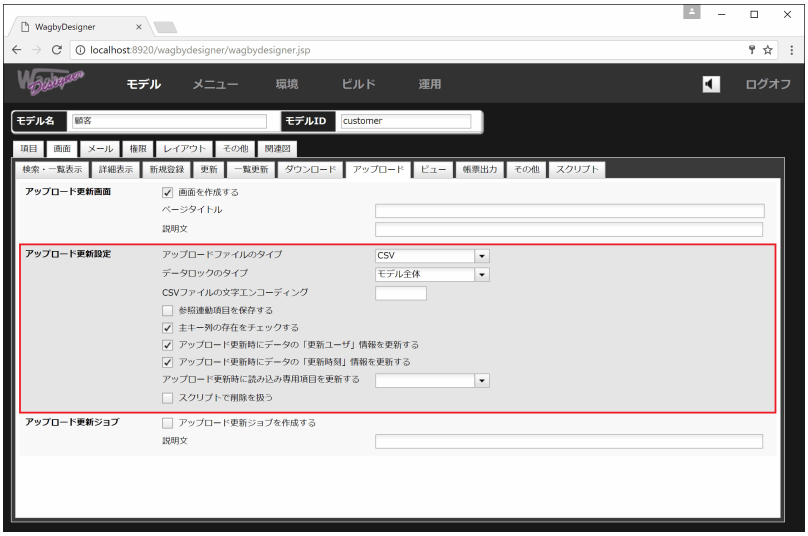


図 アップロード更新機能に関する設定

アップロードファイルのタイプ

アップロードファイルは次のルールによって自動判定されます。

ファイルの拡張子	説明
csv	CSV(カンマ区切り)として処理する。
tsv	TSV(タブ区切り)として処理する。
xls または xlsx	Excel 形式のファイルとして処理する。
上記以外	「アップロードファイルのタイプ」で指定された方式と解釈して処理する。

すなわち本設定欄は、拡張子で判断できない場合に csv または tsv のいずれかを決め打ちで解釈させるものです。

データロックのタイプ

アップロード更新のロックは次の二通りから選択できます。

方式	説明
モデル全体	本処理を実行している間、対象モデルのデータはすべてロックされます。 ・ データを更新しているユーザがいるときには、本機能を実行できません。 ・ すべてのユーザは、この処理が終了した後に対象モデルの更新、削除処理を行うことができます。
レコード毎	データを更新しているユーザがいるときでも、本機能を同時に行うことができます。 ・ アップロード更新中に、アップロード更新の対象ではないデータを更新・削除することができます。 ・ 他のユーザが更新中のデータをアップロード更新で更新しようとした時はロックエラーが発生します。そのデータのアップロード更新の処理は失敗します。エラー処理結果にてどのデータが更新に失敗したかを確認することができます。

いずれの方式でも、検索や一覧・詳細表示などの閲覧系ならびに新規登録は（本処理の実行中でも）行うことができます。

CSVファイルの文字エンコーディング

モデル毎にアップロードファイルの文字エンコードを指定することができます。この値が空白のときは「環境 〉 アプリケーション 〉 初期パラメータ 〉 CSVファイルの文字エンコーディング」の値が用いられます。

参照連動項目も保存する

モデルに「参照連動項目を自モデルに保存する」設定を行っている項目が含まれていた場合、本設定を有効にするとアップロード更新でも自モデル保存機能を有効にすることができます。

主キー列の存在をチェックする

本機能はアップロードファイルに新規登録データと更新データを混在させることができます。そのため主キー列の存在をチェックしています。具体的には次のように動作します。

- 主キー値は設定されていないが、それ以外のいずれかの列に値が設定されている場合。エラーとします。"必須項目となっています" というメッセージが画面に表示されます。
- 主キー列そのものが存在しない場合。エラーとします。"CSVファイルのヘッダ行が不正です" というメッセージが表示されます。
- すべての列の値が未設定（空行）の場合。エラーではありません。この行の処理をスキップします。

アップロードデータが全て新規登録データで更新は行わないという運用を行う場合、この主キー存在チェックを行わないようにすることができます。

ご注意ください：本設定を無効（主キー存在チェックを行わない）とした場合、主キー項目は必ず人工キー（順序）を適用するように設定してください。

アップロード更新時にデータの「更新ユーザ」情報を更新する

本設定が有効の場合、「固定値 － 更新ユーザ」は（アップロード更新を処理したユーザに）更新されます。これを更新させない運用を行う場合、本設定を解除してください。

ただし「アップロード更新時に読み込み専用項目を更新する」を許可し、かつ、アップロードファイルに「更新ユーザ」が含まれている場合は、アップロードファイル中のデータを "正" として、その値に更新します。

アップロード更新時にデータの「更新時刻」情報を更新する

本設定が有効の場合、「固定値 - 更新時刻」は（アップロード更新が処理された日付時刻に）更新されます。これを更新させない運用を行う場合、本設定を解除してください。

ただし「アップロード更新時に読み込み専用項目を更新する」を許可し、かつ、アップロードファイルに「更新時刻」が含まれている場合は、アップロードファイル中のデータを "正" として、その値に更新します。

アップロード更新時に読み込み専用項目を更新する

この設定を有効にすると、登録画面または更新画面のいずれかが読み込み専用であった場合でも更新されるようになります。選択肢は次のとおりです。

設定欄	説明
（空白）	プロジェクト定義の設定に従う（標準は「×（許可しない）」）
○	更新を許可する
×	更新を許可しない

(363)

スクリプトで削除を扱う

(363) 標準は「×」です。アップロード更新で修正する項目に読み込み専用項目は含まれている場合のみ「○」としてください。

″画面 > アップロード > アップロード更新設定 > スクリプトで削除を扱う″ を有効にしてください。

項目の設定

アップロード更新の対象に含める

各項目をアップロード更新の対象に含めるかどうかを個別に指定できます。

「モデル項目詳細定義>入力制御>アップロード更新」の「処理対象とする」を制御します。標準では空白となっています。

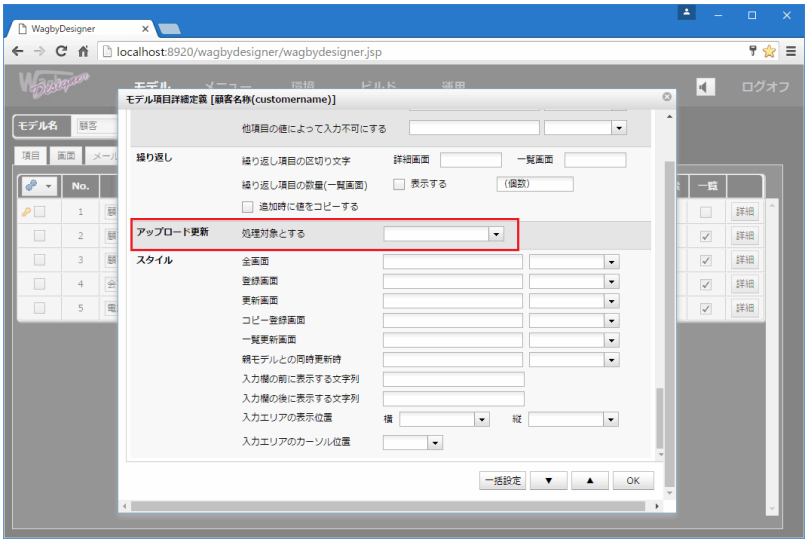


図 アップロード更新の処理対象の指定

処理対象の指定	説明
（空白）※標準	「出力制御 - その他 - CSV - CSVに出力する」の値を使います。[後述]
○	アップロード更新の処理対象項目となります。上記「CSVに出力する」の値は考慮しません。
×	アップロード更新の処理対象項目となりません。上記「CSVに出力する」の値は考慮しません。

CSV出力の設定

上記設定欄が「空白」の場合、「モデル項目詳細定義>出力制御>その他」の「CSVに出力する」によって制御されます。これは標準では有効となっています。

図 項目の出力設定

つまり標準では ″処理対象とする″ が空白で、″CSVに出力する″ が有効のため、アップロード更新対象として解釈されます。さらに、ダウンロードとアップロード更新の有効・無効をそれぞれ指定することができます。

処理件数の扱い（8.3.0）

処理件数は「登録」「更新」「削除」「エラー」の行数の合計になります。

入力チェックエラーにおけるエラーメッセージ

例えばアップロード更新でエラーが発生し、そのメッセージが次の内容とします。

"2-4 行目 : 1 の項目に 1 を入力することはできません。"

これはヘッダ行込みの行数表示です。エラー処理結果ファイルをダウンロードした場合も同様に行数表示となっています。

この理由は、アップロードしたCSV/Excelファイルの実際の行番号と、エラー行を一致させるためです。（どの行がエラーか、を把握しやすくするため、です。）

(364)

エラー時の動作

特定行のみエラーの場合

特定の行にエラーが含まれていた場合、当該行のみ処理がスキップされます。次の行へと処理を継続します。

致命的なエラーの場合

致命的なエラー（ファイルの破損、ダブルクォーテーションの整合性がとれていない）に遭遇した場合、その時点で処理は停止されます。このとき、エラー行の前までは更新されます。エラー行以降は処理がスキップされます。

更新結果レポートの確認

「すべての処理結果をダウンロード」と「エラー処理結果をダウンロード」のリンクをクリックすることで、それぞれの結果をファイルでダウンロードすることができます。

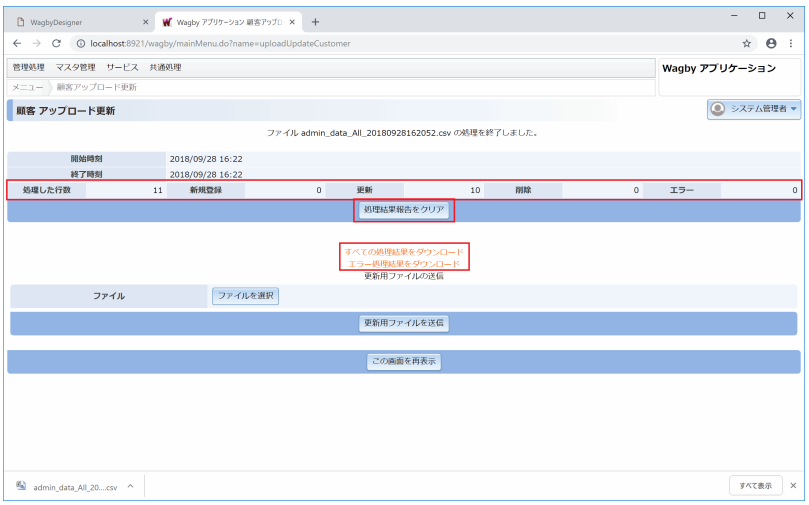
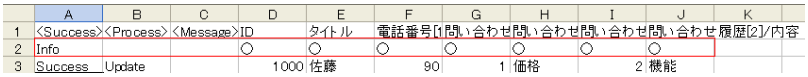


図 処理結果をダウンロードできる

処理結果レポートの確認

(364) R8.2.3 までは、処理件数は読み込んだファイルの行数となっています。これはヘッダ行や、無視される行も含まれた数です。Excelファイルをアップロード更新する場合、罫線が含まれている行はデータと判定されます。実際にはデータは含まれていない場合、これは「無視される行」としてカウントされます。つまり、罫線のみの行も最終的な処理件数に含まれます。

「すべての処理結果をダウンロード」のリンクをクリックするとファイルをダウンロードできます。 ファイル名は "process_モデル名" となっており、中身はテキストファイルです。Excelを使って開いた例を図に示します。



	A	B	C	D	E	F	G	H	I	J	K
1	<Success>	<Process>	<Message>	ID		タイトル	電話番号[1]	問い合わせ	問い合わせ	問い合わせ	履歴[2]/内容
2	Info			○		○	○	○	○	○	○
3	Success	Update		1000		佐藤	90	1	価格	2	機能

図 処理結果のレポートをExcelで開いた例

アップロードしたデータ全件について「Success」「Process」「Message」という列で結果を確認できます。それぞれ次のような意味です。

列名	説明
Success	処理の成功、失敗を表します。成功の場合は "Success"、失敗の場合は "Error" と出力されます。
Process	新規登録は "Insert"、更新は "Update"、削除は "Delete"、何も行われていない場合は "None" と出力されます。
Message	エラー内容が出力されます。複数のエラーがあった場合は改行して出力されます。

なお、図の二行目（赤枠部）は「どの列が処理されたか」を示す情報です。処理対象となった列に "○" が表示されています。(※注)

エラー処理レポートの確認

「エラー処理結果をダウンロード」のリンクをクリックするとファイルをダウンロードできます。 ファイル名は "error_モデル名" となっており、中身はテキストファイルです。Excelを使って開いた例を図に示します。 ここではエラーがないため2行目以降は空行となっています。

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	<Success>	<Process>	<Message>	<Status>	主キー	顧客名称	顧客カナ	会社名称	会社住所	会社創業日	業種	備考	作成日
2													
3													
4													
5													
6													
7													

図 エラー結果のレポートをExcelで開いた例（エラーなし）

エラーがあった場合は「Success」の列に "Error" が設定されます。「Message」にはエラー内容が出力されます。[(365)]

処理結果レポートのクリア (8.1.1)

「処理結果報告をクリア」ボタンを押下することで、前回の処理結果レポートを消去することができます。

独自に用意したファイルをアップロードする

Wagby のダウンロード機能で取得したものではない、オリジナルのCSVファイルもアップロード更新に用いることができます。次のルールを適用するようにしてください。

- 先頭（1 行目）は日本語の項目名を記載します。項目名はダウンロード機能で取得する形式と同じものとします。異なる項目名を使用した場合は、正しく更新されません。
- データ中に 「"」, 「'」, 「(改行)」のいずれかが含まれる場合、そのデータ全体を引用符 「"」で囲むようにしてください。これはCSVファイルの仕様です。
- データ中に含まれる「"」は「""」へと置換してください。これはCSVファイルの仕様です。

(365) 絞り込み（チェックボックス）を設定した項目は、絞り込み項目の値を考慮しません。モデル参照のデータすべてを見て判断します。条件付き権限を指定した場合、条件の部分は判断に使用しません。権限の部分のみを考慮します。

Excel が提供する「CSV形式で保存」機能を使うことで、上記で示したルールに基づく（引用符「"」を正しく考慮した）CSV ファイルを用意することができます。

リポジトリの設定によってはヘッダ行の項目名とデータベースのカラム名が同一とならない場合があります。 このため独自にアップロード更新用ファイルを用意する場合は、一度ダウンロード機能を実行し、アップロード更新対象となる項目名が完全に合致するかどうかを確認するようにしてください。

複数のファイルをzip圧縮して送信する

複数の csv（または tsv）形式のファイルを 1 つの zip ファイルに圧縮し、これをアップロード更新することができます。次のルールが適用されます。

- 同じモデルに関するデータを複数の CSV ファイルに分割して登録できます。
- ファイル名のアルファベット順に処理されます。
- zip ファイル内に処理可能なファイルが一つも含まれていない場合は、エラーメッセージを表示して終了します。
- エラーがあった場合、エラーメッセージの先頭に処理対象ファイル名を記載します。これによって、どのファイルでエラーが生じたかがわかるようになっています。

zip ファイル内に Excel ファイルを含めることはできません。（Excel ファイルは無視されます。）

zip ファイル内に csv 形式および tsv 形式ファイルを混在させることもできます。ファイルの拡張子で判別します。

Excel利用時に数値フォーマットを指定する

Excelファイルを使ったアップロード更新時に、数値項目の解析に使う数値フォーマットを指定できます。 未指定の場合は "##0.###" という固定フォーマットが用いられます。これは小数点3桁までの値を取り込むようになっています。

定義方法

モデル項目の「入力制御 > アップロード更新 > 数値フォーマット(Excel)」を指定します。次の例は小数点以下17桁まで入力するように設定しています。

```
##0.################
```

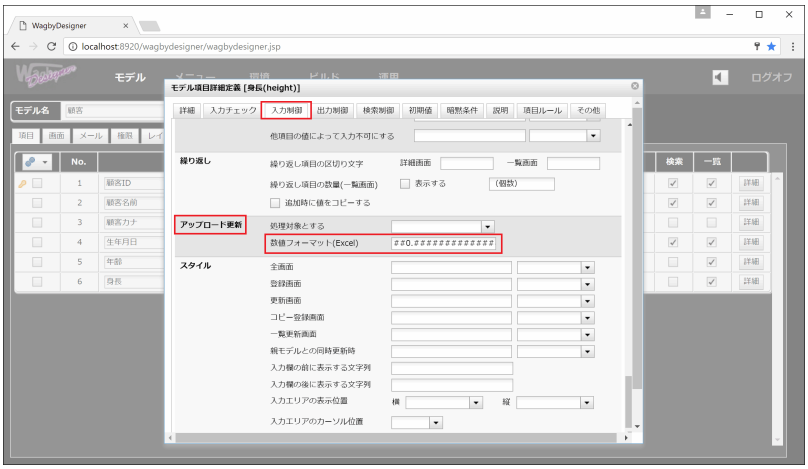


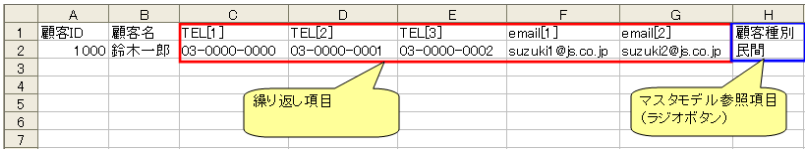
図 数値フォーマット(Excel)を指定する

この設定は、CSVファイルを用いたアップロード更新には影響しません。（CSVファイルを用いたアップロード更新では、入力値はそのまま用いられます。）

アップロード更新（繰り返し項目と繰り返しコンテナ）

繰り返し項目（モデル全体）

ダウンロード機能によって取得したファイル内における「繰り返し項目」のデータを変更し、アップロード更新を行うことができます。図のようなファイルになります。



	A	B	C	D	E	F	G	H
1	顧客ID	顧客名	TEL[1]	TEL[2]	TEL[3]	email[1]	email[2]	顧客種別
2	1000	鈴木一郎	03-0000-0000	03-0000-0001	03-0000-0002	suzuki1@js.co.jp	suzuki2@js.co.jp	民間
3								
4								
5								
6								
7								

図 繰り返し項目（一つのファイルに含まれる場合）

追加

添え字（インデックス）部分を増やしたカラム（列）を追加します。 添え字にマイナス値を指定することで、″前″ を表現することができます。（例： ″電話番号[-1]″）

削除

削除したいカラム（列）の値を空白とし、アップロード更新を行います。同データは削除されます。

繰り返し項目（独立したファイル）

主キーに対応する値を記述した行を追加し、<Count>列の数値を指定します。図の例を使って説明します。

	A	B	C	D
1	顧客ID	<Count>	電話番号	
2	1000	1	03-0000-0000	
3	1000	2	03-0000-0001	
4	1001	1	06-1111-1111	
5				
6				

図 繰り返し項目（独立したファイル）

先頭への挿入

繰り返し項目の先頭に挿入する場合、<Count>列に-1を指定します。

顧客ID	<Count>	電話番号
1000	-1	03-0000-0009

最後尾への追加

繰り返し項目の最後尾に追加する場合、<Count>列に3を指定することで、3番目の繰り返し項目が追加されます。

顧客ID	<Count>	電話番号
1000	3	03-0000-0010

削除

主キーに対応する値を空白とします。 この例では、電話番号列に空白を指定しています。

顧客ID	<Count>	電話番号
1000	1	

複数行を先頭に挿入する

<Count>列に "-1" を指定したデータを複数個、用意します。記載順に登録されます。

複数行を後尾に追加する

<Count>列に空文字を指定したデータを複数個、用意します。記載順に登録されます。

Count列を除外したファイルの扱い

<Count>列を含まないファイルをアップロードした場合、繰り返し項目の後尾にデータを追加します。

繰り返しコンテナ（モデル全体）

ダウンロード機能によって取得したファイル内における「繰り返しコンテナ項目」のデータを変更し、アップロード更新を行うことができます。図のようなファイルになります。

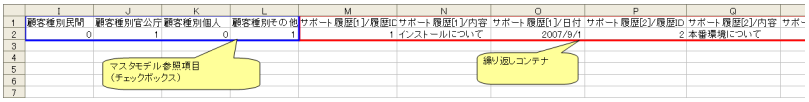


図 繰り返しコンテナ項目（一つのファイルに含まれる場合）

以下、ダウンロード機能によって下表のファイルを得られたところから説明します。 (366)

表. ダウンロードされた CSV ファイル

id
1000

新規登録

列を追加したファイルをアップロードすることで、繰り返しコンテナを新規に登録することができます。

表. 2つの繰り返しコンテナを新規に登録する

id	履歴[1]/履歴ID	履歴[1]/内容	履歴[2]/履歴ID	履歴[2]/内容
1000	1	aaa	2	bbb

繰り返しコンテナの並びは列の添え字（上記例の "[1]","[2]"）で指定します。

追加

列を追加したファイルをアップロードすることで、3番目の繰り返しコンテナが追加されます。

表. 3番目の繰り返しコンテナを追加する

id	履歴[1]/履歴ID	履歴[1]/内容	履歴[2]/履歴ID	履歴[2]/内容	履歴[3]/履歴ID	履歴[3]/内容
1000	1	aaa	2	bbb	2 (*)	ccc

(366) "id" 列は主キー項目とします。

追加した繰り返しコンテナのコンテナIDの値（上記例では（*）の ″2″）は、アップロード更新後に（順序よく）再設定されます。すなわち、この値は（整数であれば）どのような値でもかまいません。

繰り返しコンテナの並びは列の添え字（上記例の ″[1]″,″[2]″,″[3]″）で指定します。

コンテナIDが空白の場合

コンテナIDが空白の場合は、追加処理は行われません。

表. 誤った指定方法（繰り返しコンテナは追加されない）

id	履歴[1]/履歴ID	履歴[1]/内容	履歴[2]/履歴ID	履歴[2]/内容	履歴[3]/履歴ID	履歴[3]/内容
1000	1	aaa	2	bbb		ccc

挿入

添字に ″-1″ を指定した列を追加することで、1番目の繰り返しコンテナが挿入されます。 既存の繰り返しコンテナのデータは、1つずつ、ずれることになります。[367]

表. 1番目の繰り返しコンテナを挿入する

id	履歴[1]/履歴ID	履歴[1]/内容	履歴[2]/履歴ID	履歴[2]/内容	履歴[-1]/履歴ID	履歴[-1]/内容
1000	1	aaa	2	bbb	2	ccc

削除

コンテナID の値を空白にすることで、繰り返しコンテナの削除を行うことができます。 下表に示すデータをアップロードすることで、1番目の繰り返しコンテナが削除されます。

表. 1番目の繰り返しコンテナを削除する

id	履歴[1]/履歴ID	履歴[1]/内容	履歴[2]/履歴ID	履歴[2]/内容
1000		aaa	2	bbb

更新

コンテナIDを除いたファイルをアップロードすることで、指定された値で更新を行います。

表. 既存データを更新する

id	履歴[1]/内容	履歴[2]/内容	履歴[3]/内容 !誤記
1000	ddd	eee	fff

この例では 1 番目の繰り返しコンテナ 内容 の値が ddd へ、2 番目の繰り返しコンテナ 内容 の値が eee へと更新されます。

3 番目の繰り返しコンテナは追加されず、無視されます。これはコンテナIDが指定されていないためです。

繰り返しコンテナ（独立したファイル）

繰り返しコンテナを独立したファイルとして扱うことができます。図のようなファイルとなります。

[367] ファイルでは右側の列に位置していますが、アップロード更新後のデータベースでは、添字 ″-1″ のコンテナは「先頭に挿入」されます。

	A	B	C	D	E
1	顧客ID	商談履歴/履歴ID	商談履歴/商談日	商談履歴/商談内容	
2	1000	1	2009/2/2	担当者挨拶。	
3	1000	2	2009/2/16	WagbyPro Plusの紹介。	
4	1001	1	2008/12/3	名刺交換。次期開発案件について。	
5					
6					

図 繰り返しコンテナ項目（独立したファイル）

挿入

主キーおよびコンテナIDを含む行を追加することで対応します。コンテナIDを適切に指定することで、データの挿入も行えます。

繰り返しコンテナの先頭に挿入する場合、商談履歴/履歴ID(コンテナID)列に ″-1″ を指定します。

顧客ID	商談履歴/履歴ID	商談履歴/商談日	商談履歴/商談内容
1000	-1	2014/06/01	機能拡張について。

コンテナIDが ″-1″ のデータを複数個、指定することもできます。記載順に登録されます。

顧客ID	商談履歴/履歴ID	商談履歴/商談日	商談履歴/商談内容
1000	-1	2014/05/01	機能拡張について。
1000	-1	2014/05/10	機能拡張詳細について。

追加

繰り返しコンテナの最後尾に追加する場合、商談履歴/履歴ID(コンテナID)列に3を指定することで、3番目の繰り返しコンテナが追加されます。

顧客ID	商談履歴/履歴ID	商談履歴/商談日	商談履歴/商談内容
1000	3	2014/06/01	機能拡張詳細について。

複数行を先頭に挿入する

コンテナID列に ″-1″ を指定したデータを複数個、用意します。記載順に登録されます。

複数行を後尾に追加する

コンテナID列に空文字を指定したデータを複数個、用意します。記載順に登録されます。

コンテナID列を含まないファイル

コンテナID列を含まないファイルをアップロードした場合、コンテナの後尾にデータを追加します。

繰り返しコンテナの削除

本方式では、繰り返しコンテナの削除を行うことはできません。

制約

項目の型

参照連動型項目の参照先が繰り返し、およびチェックボックスの場合には本機能は動作しません。

詳細情報の出力について

繰り返し項目、繰り返しコンテナの独立したファイルを扱う場合、「処理結果ファイル」「エラー結果ファイル」にはその主キーが最後に出現した行に処理情報が出力されます。これ以前の行では処理状態がNONEとなり、エラーメッセージは出力されません。

この詳細を説明します。 下記の例で、1番目のデータの商談履歴/商談日項目に日付として認識できない文字を指定します。これはエラーになります。

顧客ID	商談履歴/履歴ID	商談履歴/商談日	商談履歴/商談内容
1000	1	あああ	担当者あいさつ
1000	2	2014/06/01	機能拡張詳細について。

このときの処理結果ファイルは次のとおりです。 最後に1000番の主キーが指定されている行にエラーメッセージが表示されます。

<Success>	<Process>	<Message>	顧客ID	商談履歴/履歴ID	商談履歴/商談日	商談履歴/商談内容
Success	None		1000	1	あああ	担当者あいさつ
Error	Update	商談日の項目に日付として認識できない文字が含まれています。	1000	2	2014/06/01	機能拡張詳細について。

Status列の扱い

繰り返し項目、繰り返しコンテナの独立したファイルを扱う場合、<Status>列を指定した場合、その主キーが最後に出現した行の指定が有効となります。

この詳細を説明します。 下記の例では、1000番の主キーに対して二行、用意されています。このとき、2行目の "u" (主キー1000番のデータの更新)が優先されます。1行目の "d" (主キー1000番のデータの削除)は無視されます。

<Status>	顧客ID	商談履歴/履歴ID	商談履歴/商談日	商談履歴/商談内容
d !無視される	1000	1	2014/04/01	担当者あいさつ
u 有効	1000	2	2014/06/01	機能拡張詳細について。

アップロード更新（ファイル型項目）

用途

次のような場合にお使いいただけます。

- 社員マスタについて、社員の顔写真をまとめて登録する。
- 商品マスタの写真を一括で登録する。

アップロード方法

ここでは社員リスト staff.csv のアップロード更新を例に説明します。

1. 次のようなデータを用意します。ここで "写真" 項目はファイル型です。

```
"アカウント","名前","写真","写真_jfcattachment"
"s110001","日本 太郎","写真_s110001.png","photo_s110001.png"
"s110002","東京 二郎","写真_s110002.png","photo_s110002.png"
"s110003","大阪 花子","写真_s110003.png","photo_s110003.png"
```

```
"s110004","沖縄 四郎","写真_s110004.png","photo_s110004.png"
```

※ 項目"写真"が「ファイル名」型項目です。 ※ "写真_jfcattachment"列に添付する実ファイル名を記述します。 ※ "写真"列は従来通り画面表示時やダウンロードファイル名に利用されます。この項目は必ずしも実ファイル名と同一である必要はありません。

2. この staff.csv と、実際の画像ファイルを以下のようにZIPファイルにまとめます。

```
data.zip
 ├─staff.csv
 ├─photo_s110001.png
 ├─photo_s110002.png
 ├─photo_s110003.png
 └─photo_s110004.png
          :
```

3. 任意のZIPファイル名で保存し、CSVアップロード更新を行います。

繰り返しコンテナへの対応

繰り返しコンテナへの登録も可能です。繰り返しコンテナID項目と、対象項目をコンマ区切りで列挙するとともに、コンテナ番号を[1],[2]と（必要な数だけ）指定してください。

```
"ID","タイトル","コンテナ[1]/CID","コンテナ[1]/ファイル","コンテナ[2]/CID","コンテナ[2]/ファイル"
"1000","uploadtest","1","photo.gif","2","sample.pdf"
```

アップロードする zip ファイルの構成も同じです。

```
data.zip
 ├─uploadtest.csv
 ├─photo.gif
 └─sample.pdf
```

添付ファイルをサブフォルダで管理する

添付ファイルをサブフォルダ以下にまとめて配置することができます。フォルダ名は「files」（すべて英小文字）となります。

```
teacherlist.zip
├─teacherlist.csv
└─files
      ├─miyamoto.png
      ├─sasaki.png
      ├─README.doc
      └─initdata.csv
```

アップロードするCSVファイルの指定は以下のようになります。

```
"アカウント","ファイル名","ファイル名_jfcattachment"
"teacher1","miyamoto.png","./files/miyamoto.png"
"teacher2","sasaki.png","files/sasaki.png"
"teacher3","README.doc",".¥files¥README.doc"
"teacher4","initdata.csv","files¥initdata.csv"
```

上記 CSV 例に示したように、次のルールがあります。

- サブフォルダ名は「files」となる。
- 相対パス表記は、フォルダの区切り文字に「/」と「¥」のいずれも利用できる。
- 「./files」と「files」は同じ意味である。
- 画像ファイルだけでなく、さまざまなファイル形式を送信できる。
- csv ファイルを送信した場合、これは添付ファイルとして解釈される。

サブフォルダ名についての詳細

サブフォルダ名が files の場合、ここに含まれるファイルは（CSV形式ファイルであっても）すべて添付ファイルとして扱われます。 それ以外のフォルダ名を用いた場合、拡張子が csv のファイルは「アップロード対象データ」として扱われます。

例を示します。次のようにフォルダ teacherlist の中に複数の csv ファイルを含めたとします。

```
teacherlist.zip
└─teacherlist
    ├─teacherlist.csv
    └─files
      ├─miyamoto.png
      ├─sasaki.png
      ├─README.doc
      └─initdata.csv
```

この場合、最初の teacherlist フォルダは（files という名称ではないため）その中に含まれる csv ファイル（teacherlist.csv）を「アップロード対象データ」と解釈します。 さらにその中にある files フォルダ内の initdata.csv は、添付ファイルとして扱います。

ファイルが正常にアップロードされない場合

ファイルが正常にアップロードされない場合、次の点をご確認ください。

- Excelファイルを使っていないか。本機能はCSVファイルのみで利用できます。
- 「項目名_jfcattachment」列が、対象となるCSVファイルに含まれているか。
- 指定したファイル名が、実ファイル名と完全に一致しているか。

ダウンロード機能によって取得したファイルには、「項目名_jfcattachment」列は含まれていません。
（この項目はアップロードするファイルという意味ですので、CSVダウンロード時には取得できません。）
そのためダウンロードによって取得したファイルをそのままアップロードしても、ファイルは正常にアップロードされません。

ダウンロードジョブ

定義方法

ダウンロード処理を画面上から手動で行うこととは別に、時間指定ジョブとして実行することができます。

ダウンロードするファイルは「本番運用サーバ」からアクセスできるパスに用意してください。

「画面 ＞ ダウンロード ＞ ダウンロードジョブ」にて、「ダウンロードジョブを作成する」を有効にします。あわせて「説明文」にこのジョブの説明を記述します。

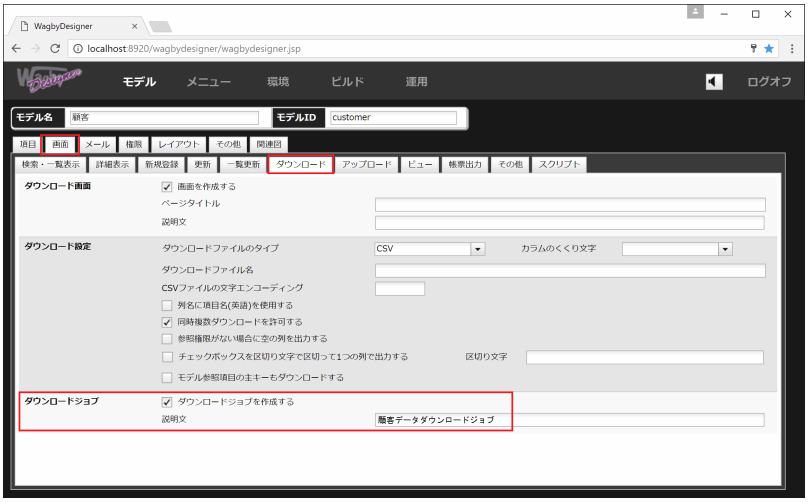


図 ジョブを用意する

ダウンロードファイルの形式は「ダウンロード設定 > ダウンロードファイルのタイプ」で指定した方法になります。図では "CSV" となっているため、CSV ファイル形式で出力されます。ここで "Excel(xls)" を選択すると、ダウンロードジョブも Excel ファイル形式で出力されます。

ジョブの登録

jobadminアカウントを用意する

「サポート > 管理者ガイド(R8) > 準備：ジョブ実行アカウントを利用可能にする」の手順に従って、jobadmin アカウントを用意してください。

jobadmin アカウントは、対象モデルを操作できる権限（プリンシパル）を与えるようにしてください。権限がない場合、アップロード更新ジョブは失敗します。

ダウンロードジョブを設定する

Wagbyのジョブを初めて扱う場合は"サポート > 管理者ガイド(R8) > ジョブ管理"をお読みください。ここでは、ジョブの具体的な使い方を説明します。

1. 管理者権限をもったアカウントでログオンします。
2. メニューから「ジョブ管理」「ジョブスケジュール検索」画面を開きます。
3. 「登録画面へ」ボタンを押下し、ジョブスケジュールの新規登録処理を行います。
4. ジョブ名から「Download + モデルID + Job」を選択します。その下の説明欄に、Designerで入力した説明文が表示されます。
5. ジョブパラメータに、実行したいコマンドを記述します。[後述]
6. スケジュールを指定します。
7. 「有効」フラグをチェックします。
8. このジョブを実行する管理者のアカウントおよびパスワードを入力します。先ほど用意した jobadmin を使うとよいです。
9. ジョブスケジュールを登録します。指定した時間になると、コマンドが実行されます。

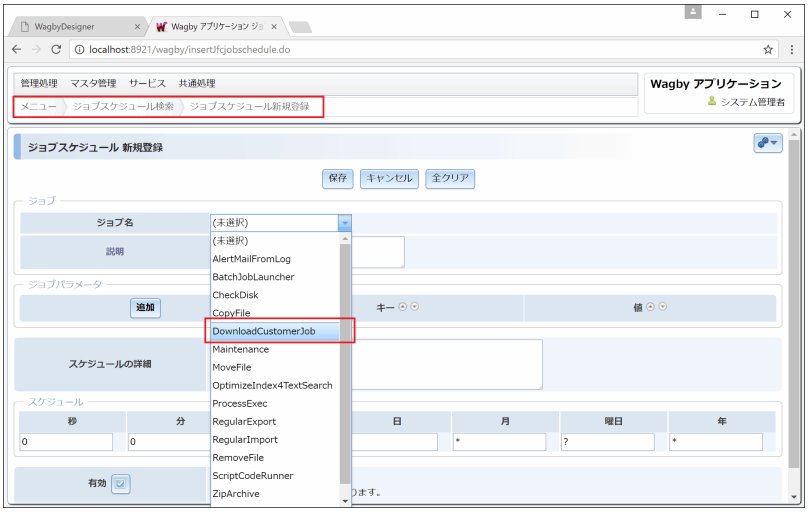


図 ダウンロードジョブを作成する

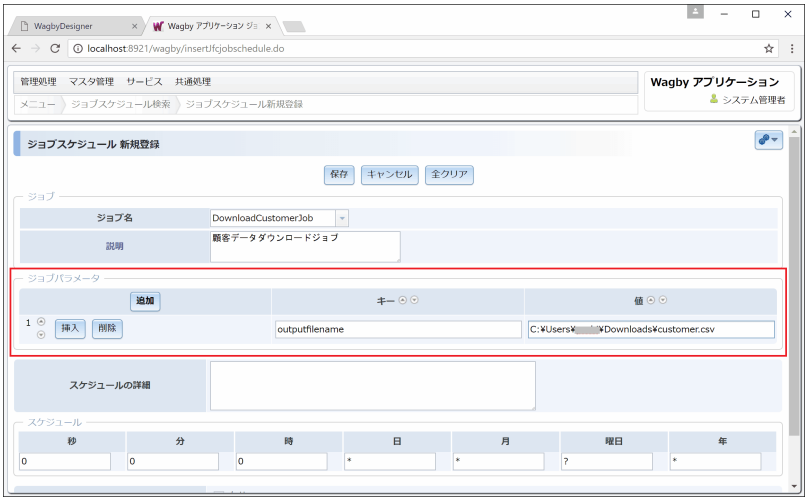


図 ダウンロードジョブパラメータを設定する

図で対象モデルのジョブの選択肢が表示されない場合は、ビルド後に jfcjob モデルのデータインポートを行なっていない可能性があります。初期リポジトリ data_init 内にある jfcjob フォルダに、追加したジョブに関する xml ファイルが追加されていますので、ご確認ください。

指定できるジョブパラメータ

このジョブに関する制御はすべてジョブパラメータとして指定します。

パラメータ名	説明	記述例
outputfilename	保存するファイル名。保存先は本番運用サーバ。指定がない場合は wagbyapp¥tempフォルダに、download_xxx.csv というファイル名で出力される。（xxxはランダムな文字列。）	C:¥Downloads¥YOURFILE.csv/ home/wagby/YOURFILE.csv
outputdir	保存先フォルダ名。outputfilenameが指定されていない場合に用いる。指定されたフォルダ名に保存する。ファイル名はoutputfilenametypeの指定に準じる。	C:¥Downloads/home/wagby
csv_encoding	csv,tsv形式の場合の文字エンコーディングを指定する。未指定の場合は標準の動作となる。	utf-8
outputmode	モデルに繰り返し項目や繰り返しコンテナがある場合、これを含めるかどうかを指定できる。［参考: 繰り返し項目・繰り返しコンテナ項目の扱いに関するルール］	All - すべて Main - 繰り返しコンテナを含まないすべて Modelitem_[項目名] - 繰り返し項目または、繰り返しコンテナのみ
fileext	ダウンロードファイルのタイプに"実行時指定"を設定した場合に指定する。	csvtsvxlsxlsx
outputfilenametype	ファイル名種別。未指定、temp、filename、extを指定できる。 未指定、tempの場合は一時ファイル名として、download_[ランダムな	tempfilenameext

	文字列].csvとなる。必ず一意になる。現状のダウンロードジョブと同様。tsvやxlsなどcsvでない形式を選択しても、拡張子は.csvとなる。filenameの場合はDesignerで設定されたファイル名で保存する。拡張子も指定した形式に対応して設定される。 extの場合は拡張子はfilenameと同様に指定した形式に対応して設定されるが、拡張子を除くファイル名はtempと同様に必ず一意になるファイル名が設定される。	
sort_key 8.3.1	項目名を指定する。指定順に昇順で並べる。":DESC" を付与すると降順となる。コンマ(,)区切りで複数指定できる。	pkey:DESC
targetitems 8.3.1	項目名を指定する。指定した項目を、指定した順番で出力する。コンマ(,)区切りで複数指定できる。	pkey,item1,item5,item3
locale 8.3.1	ロケールを指定する。出力されるヘッダ行はロケールによって適切な値となる。未指定の場合は標準ロケール(日本語環境の場合は"ja")となる。	en

ジョブの実行を停止する

ジョブを一時的に無効にすることができます。

1. 管理者権限をもったアカウントでログオンします。
2. メニューから「ジョブ」「ジョブスケジュール検索」画面を開きます。
3. 該当する「ジョブスケジュール」を検索し、更新画面を開きます。
4. 「有効」フラグについているチェックをはずし、保存します。

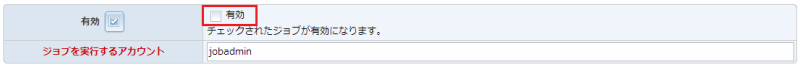


図 ジョブを無効にする

実行結果をメールで送信する

ジョブの実行結果をメールで送信することができます。

多重実行の抑制

ジョブの多重実行を行うことはできません。

あるジョブを 1 分毎に起動させるようにした場合、1 分後にこのジョブの実行が終了していなければ、新しいジョブは実行されません。

すぐに実行する

時間指定ではなく、このジョブを今すぐ実行することもできます。 "ジョブの即時実行"をお読みください。

仕様・制約

- outputfilenameに指定するファイル名に対応したフォルダが存在しない場合、エラーとなります。
- outputfilenameに指定するファイル名に日時や項目値を含めるというようなことはできません。固定ファイル名となります。
- outputfilenameに指定したファイルがすでに存在する場合は、上書きされます。
- ダウンロード時の検索条件を指定することはできません。全件ダウンロードされます。

アップロード更新ジョブ

定義方法

アップロード更新処理を画面上から手動で行うこととは別に、時間指定ジョブとして実行することができます。

アップロードするファイルは「本番運用サーバ」からアクセスできるパスに用意してください。

「画面 ＞ アップロード ＞ アップロード更新ジョブ」にて、「アップロード更新ジョブを作成する」を有効にします。あわせて「説明文」にこのジョブの説明を記述します。

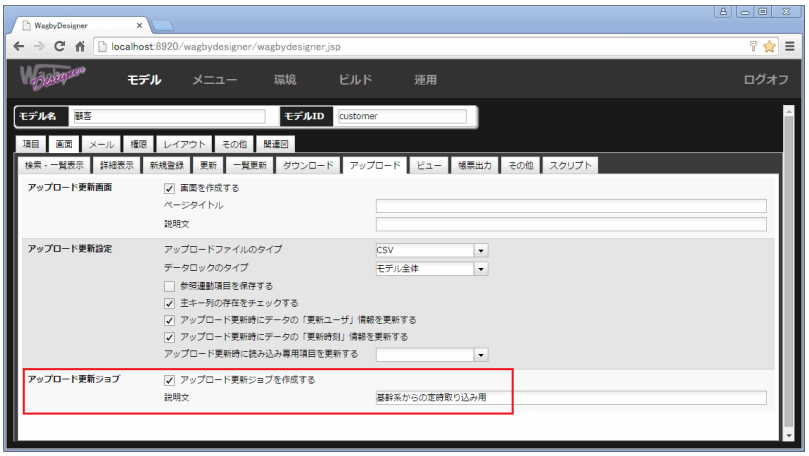


図 ジョブを用意する

ジョブの登録

jobadminアカウントを用意する

「サポート ＞ 管理者ガイド(R8) ＞ 準備：ジョブ実行アカウントを利用可能にする」の手順に従って、jobadmin アカウントを用意してください。

> jobadmin アカウントは、対象モデルを操作できる権限（プリンシパル）を与えるようにしてください。権限がない場合、アップロード更新ジョブは失敗します。

アップロード更新ジョブを設定する

Wagbyのジョブを初めて扱う場合は"サポート ＞ 管理者ガイド(R8) ＞ ジョブ管理"をお読みください。ここでは、ジョブの具体的な使い方を説明します。

1. 管理者権限をもったアカウントでログオンします。
2. メニューから「ジョブ管理」「ジョブスケジュール検索」画面を開きます。
3. 「登録画面へ」ボタンを押下し、ジョブスケジュールの新規登録処理を行います。
4. ジョブ名から「UploadUpdate + モデルID + Job」を選択します。その下の説明欄に、Designerで入力した説明文が表示されます。
5. ジョブパラメータに、実行したいコマンドを記述します。[後述]

6. スケジュールを指定します。
7. 「有効」フラグをチェックします。
8. このジョブを実行する管理者のアカウントおよびパスワードを入力します。先ほど用意した jobadmin を使うとよいです。
9. ジョブスケジュールを登録します。指定した時間になると、コマンドが実行されます。

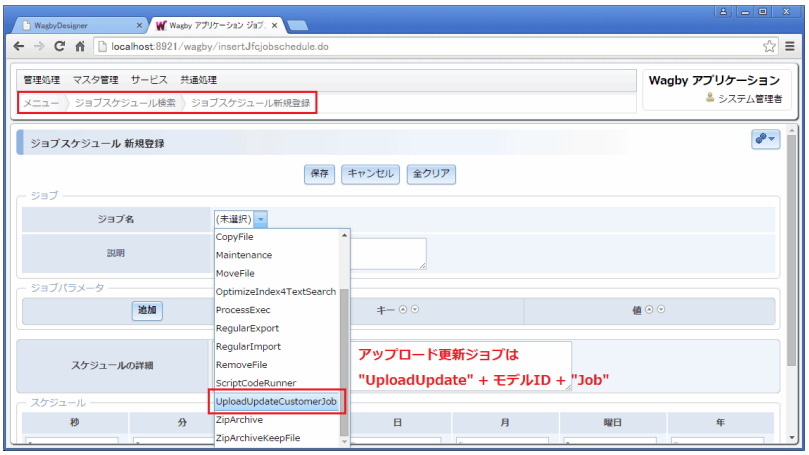


図 アップロード更新ジョブを作成する

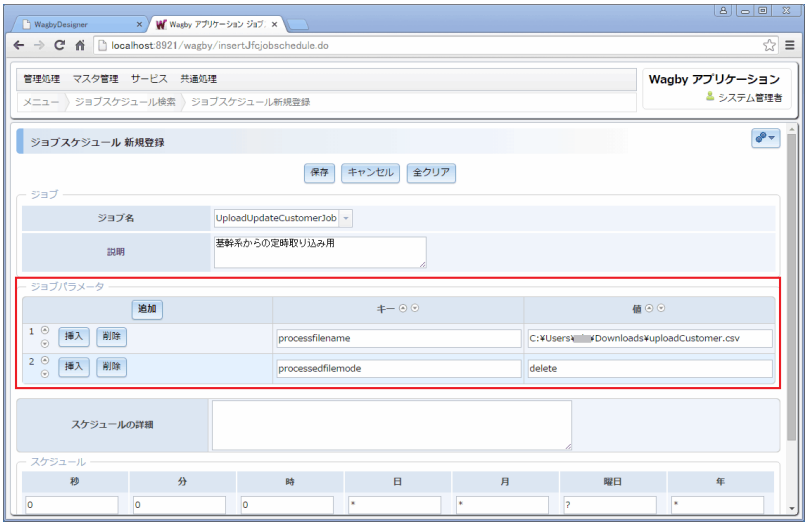


図 アップロード更新ジョブパラメータを設定する

図で対象モデルのジョブの選択肢が表示されない場合は、ビルド後に jfcjob モデルのデータインポートを行なっていない可能性があります。初期リポジトリ data_init 内にある jfcjob フォルダに、追加したジョブに関する xml ファイルが追加されていますので、ご確認ください。

指定できるジョブパラメータ

このジョブに関する制御はすべてジョブパラメータとして指定します。

パラメータ名	説明	記述例
processfilename	処理対象のファイル名。本番運用サーバに存在すること。このパラメータはprocessdirnameより優先されます。processfilename又はprocessdirnameのいずれかを指定してください。両方が未指定の場合、エラーとなります。	C:¥Downloads¥YOURFILE.csv/home/wagby/YOURFILE.csv

processdirname	処理対象ファイルが格納されているフォルダ名。本番運用サーバに存在すること。processfilename又はprocessdirnameのいずれかを指定してください。両方が未指定の場合、エラーとなります。	C:¥Downloads/home/wagby/downloads
processdirfilter	処理対象ファイルのファイル名の正規表現式を指定します。省略時は ".*¥.(csv|tsv|xls|xlsx|zip)"です。つまり拡張子が csv, tsv, xls, xlsx, zip のいずれかのみのファイルを対象とします。拡張子アルファベットの大文字小文字を区別しません。拡張子が csv または CSV のいずれも対象となります。	.*¥.csv
isLockModel	モデル全体ロックを行う場合は "true" を設定します。処理行ごとにモデル1件づつロックする場合は "false" を設定します。省略時は "true" です。	false
processedfilemode	処理対象ファイルの処理後の扱いを指定します。"delete" の場合は削除します。"rename" の場合は指定先に移動します。省略時は "delete" となります。なお "rename" を指定する場合にはrenamedirrenamefilenamerenamefいずれかを設定することができます。これらがすべて省略されていた場合、renamefilenamefooterに ".processed" が設定されたものとして動作します。	rename
renamedir	processedfilemodeが"rename"の場合に有効です。 処理完了後の処理対象ファイルの移動先フォルダを指定します。指定されているフォルダが存在しない場合は作成します。 この指定を行うと「処理結果ファイル」「エラーファイル」も指定フォルダに移動します。 相対パス表記も可能です。処理対象ファイルが置かれているディレクトリからの相対パスとして扱います。 指定されていない場合は、処理対象ファイルが置かれているディレクトリへの移動となります。{0,date,yyyyMMdd-HHmmss}と指定することで、全体の処理開始日時を指定することができます。 {1,date,yyyyMMdd-HHmmss}と指定することで、ファイルの処理開始日時を指定することができます。 {2}	customer{0,date,yyyyMMdd-HHmmss}customer20150801-100000などとなります。

	と指定することで、処理ファイル件数を指定できます。 {3}と指定することで、処理ファイル総件数を指定できます。	
renamefilenameheader	processedfilemodeが"rename"の場合に有効です。 移動先のファイル名の先頭部分を指定します。 例えば "Processed_" と指定すると、a.csv というファイルは処理後に、Processed_a.csv に名前が変更されます。 {0} {1},... といった表記を使うこともできます。 （renamedirの説明文を参照のこと）	{1,date,yyyyMMdd-HHmmss}_20150801-100000_a.csv などとなります。
renamefilenamefooter	processedfilemodeが"rename"の場合に有効です。 移動先のファイル名の末尾部分を指定します。例えば "_Processed" と指定すると、a.csv というファイルは処理後に、a.csv_Processed に名前が変更されます。 {0} {1},... といった表記を使うこともできます。 （renamedirの説明文を参照のこと）	_{1,date,yyyyMMdd-HHmmss}a.csv_20150801-100000 などとなります。
renamefilename	processedfilemodeが"rename"の場合に有効です。 移動先のファイル名を指定します。 {0} {1},... といった表記を使うこともできます。 （renamedirの説明文を参照のこと） さらに次の表記も有効です。 {4} - ファイル名(例:a.csv) {5} - 拡張子を除いたファイル名(例:a) {6} - 拡張子(例:.csv)	{5}_backup{6}a_backup.csvとなります。
processingdir	処理開始時の対象ファイルの移動先を指します。 指定すると、対象ファイルをこのフォルダに移動後に処理を行うようになります。 未指定の場合はprocessingfilenamefooterに ".processing" が設定されたとみなします。	C:¥work/tmp
processingfilenameheader	処理を開始した対象ファイルの先頭に指定文字を付与します。作業中であることを示します。 指定すると、対象ファイルの名前を変更したあとに、処理を行うようになります。 未指定の場合はprocessingfilenamefooterに ".processing" が設定されたとみなします。 Excelファイルをアップロードする場合、この値を設定してください。 [仕様・制約...]	processing_
processingfilenamefooter	処理を開始した対象ファイルの末尾に指定文字を付与しま	_processing

	す。作業中であることを示します。 指定すると、対象ファイルの名前を変更したあとに、処理を行うようになります。	
processingfilename	処理を開始した対象ファイルを、指定した名前に変更します。作業中であることを示します。	Working.csv
errordir	エラー発生時の対象ファイルの移動先を指します。 指定すると、エラー発生時に対象ファイルをこのフォルダに移動後に処理を行うようになります。 未指定の場合はerrorfilenamefooterに".error"が設定されたとみなします。	C:¥errorfile/home/wagby/errorfile
errorfilenameheader	エラー発生時に対象ファイルの先頭に指定文字を付与します。 指定すると、エラー発生時に対象ファイルの名前を変更します。 未指定の場合はerrorfilenamefooterに".error"が設定されたとみなします。	error_
errorfilenamefooter	エラー発生時に対象ファイルの末尾に指定文字を付与します。	_error
errorfilename	エラー発生時に指定した名前に変更します。	Error.csv
isAttachmentProcessfilesInMail	実行結果メール送信で、処理結果(全て)のファイルをメールに添付します。	true
isAttachmentErrorfilesInMail8.	実行結果メール送信で、処理結果(エラー)のファイルをメールに添付します。	true
uploadfilename8.3.1	processfilename でファイル名を指定したとき、スクリプトで利用できるUPLOADFILENAME変数の値は、本番運用サーバのディスクに保存されたファイル名が反映されます。しかし、ここでuploadfilenameをさらに指定すると、UPLOADFILENAME変数の値は（実際のファイル名ではなく）uploadfilenameで指定した値が反映されるようになります。ただしprocessdirname で指定した場合は（このuploadfilenameの値は）無視され、実際にディスクに保存されたファイル名が使われます。	（任意のファイル名）

[補足] 更新結果レポートを指定のフォルダへ移動する

アップロード更新の完了時に作成される "すべての処理結果" と "エラー処理結果" を指定のフォルダへ移動させることができます。 ジョブパラメータを次のように設定してください。

- processedfilemode に "rename" と指定する。
- renamedir に移動先フォルダを指定する。

システム管理者はこのサーバのフォルダを経由して、これらの結果レポートを直接、参照することができます。

ジョブの実行を停止する

ジョブを一時的に無効にすることができます。

1. 管理者権限をもったアカウントでログオンします。
2. メニューから「ジョブ」「ジョブスケジュール検索」画面を開きます。
3. 該当する「ジョブスケジュール」を検索し、更新画面を開きます。
4. 「有効」フラグについているチェックをはずし、保存します。

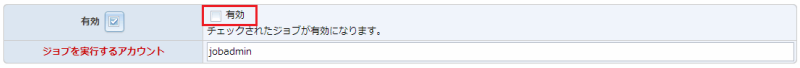


図 ジョブを無効にする

実行結果をメールで送信する

ジョブの実行結果をメールで送信することができます。

送信されるメールの例を示します。

```
処理対象ファイル .¥a.csv を処理します。

開始時刻:2015-07-09 11:56:16 / 終了時刻:2015-07-09 11:56:17
End 処理した行数:3/新規登録:1/更新:1/削除:0/エラー:0

処理対象ファイル .¥a.csv.processing を .¥20150709-115616¥a.csv に移動しました。
処理結果ファイル C:¥wagbyapp¥temp¥process_model13555561895887649907.csv を .¥20150709-115616 に移動しました。
エラー結果ファイル C:¥wagbyapp¥temp¥error_model1387678353843031222.csv を .¥20150709-115616 に移動しました。
```

なおジョブパラメータ processfilename, processdirname で指定されたファイルが存在しない場合、メール本文は空としています。これは、ジョブパラメータ mail:doNotSendEmptyMessage の設定でスキップしてもらうことを想定しているためです。

多重実行の抑制

ジョブの多重実行を行うことはできません。

あるジョブを 1 分毎に起動させるようにした場合、1 分後にこのジョブの実行が終了していなければ、新しいジョブは実行されません。

すぐに実行する

時間指定ではなく、このジョブを今すぐ実行することもできます。 "ジョブの即時実行"をお読みください。

仕様・制約

- 複数ファイル処理の途中でエラーが発生した場合、そのファイルはスキップして次のファイル処理を行います。またその際、エラーとなったファイルは名前が変更されます。（エラーとなったことがわかるようになります。）
- 処理開始前のエラー発生時には、エラーメッセージがログ(system.log)とメールにそれぞれ記録されます。

・　Excelファイルを扱う場合、processingfilenameheaderを設定してください。上の例にあるように"processing_" とするとよいでしょう。この制約は R8.3.6/R8.4.1 以降の版で解消しました。

ダウンロード時にスクリプトを呼び出す(スクリプト)

ヘルパ

「ヘルパ」で設定できるスクリプトは次の通りです。

タイミング	説明	生成されるファイル
ダウンロード(開始時)	ダウンロード処理の開始時に一回だけ呼び出されます。	Download<モデルID>_processStart.js
ダウンロード(終了時)	ダウンロード処理の終了時に一回だけ呼び出されます。	Download<モデルID>_processEnd.js
ファイル項目ダウンロード	ファイル項目をダウンロードしたときに呼び出されます。	DownloadFile<モデルID>_beforeDownload.js

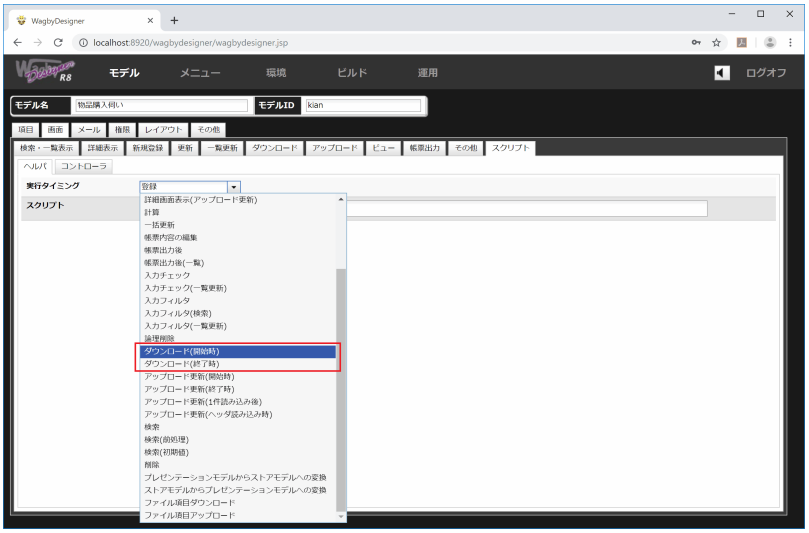


図 ヘルパのスクリプト

例 ダウンロードしたデータに出力済み文字列をセットする

ダウンロード(開始時)のスクリプトで主キーの集合を用意します。ダウンロード(終了時)に、この集合を使ってデータを更新するようにします。

ダウンロード(開始時)

```
p.request.setAttribute("__jfc_ADD_PRIMARYKEY_IN_DOWNLOAD", new java.util.ArrayList());
```

・　p.requestにセットするときのキー名__jfc_ADD_PRIMARYKEY_IN_DOWNLOADは特別な名前です。この名前から取得できるオブジェクトはリスト（java.util.ArrayList）としてください。ここに主キーの集合がセットされます。

ダウンロード(終了時)

対象モデルを model1 とした例です。このモデルの item1 に、"downloaded" という文字列をセットすると、ダウンロード済みという意味付けとしています。（モデルIDを適切に読み替えてください。）

```
var LockUtils = Java.type("jp.jasminesoft.jfc.core.util.LockUtils");

var pkeylist = p.request.getAttribute("__jfc_ADD_PRIMARYKEY_IN_DOWNLOAD");
if (pkeylist !== null) {
    var entityService = p.appctx.getBean("Model1EntityService");
    var daoHelper = p.appctx.getBean("Model1DaoHelper");
    //print("pkeylist size="+pkeylist.size());
```

```
    for (var i=0 ; i < pkeylist.size() ; i++) {
        var pkey = pkeylist.get(i);
        //print("pkey=" + pkey);
        var model1 = entityService.findById(pkey); // 1件のデータを取得
        //print(model1);
        try {
            LockUtils.lock(model1, daoHelper);// ロック取得
            model1.item2 = "downloaded";// ダウンロード済みマークをセット
            entityService.update(model1); // 更新
        } catch (e) {
            throw e;
        } finally {
            LockUtils.release(model1, daoHelper);// ロック解放
        }
    }
}
```

条件によってダウンロード処理をキャンセルする

ダウンロード(開始時)のスクリプトで例外 BusinessLogicException が発生した場合、このダウンロード処理はキャンセルされます。ダウンロードは行われません。

ヘルパのスクリプトで利用できるパラメータ

ダウンロード開始時や終了時のスクリプトで、次の暗黙変数を参照することができます。

(368)

名前	説明
outputmode	ダウンロード形式に指定した値。次の値を返す。 すべてを選択した場合は "All" 繰り返し項目を含まないすべてを選択した場合は "Main" 繰り返し項目や繰り返しコンテナを選択した場合は "Modelitem_<項目名>"
csv_encoding	CSVファイルの文字エンコーディング
condition	検索条件（コンディションモデル）

スクリプトの例を示します。

```
print("outputmode", outputmode)
print("csv_encoding", csv_encoding)
print("condition", condition)
```

コントローラ

「コントローラ > ダウンロード」で設定できるスクリプトは次の通りです。

タイミング	説明	生成されるファイル
ダウンロード実行	ダウンロードの開始ボタンを押下したときに呼び出されます。	Download<モデルID>_Download.js

(368) 参照のみです。書き換えることはできません。

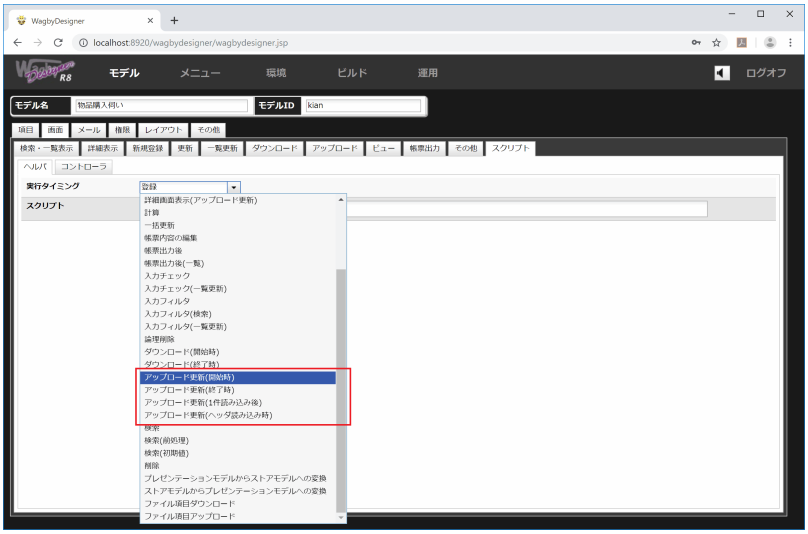


図 スクリプトを記述する

アップロード更新時にスクリプトを呼び出す(スクリプト)

ヘルパ

「ヘルパ」で設定できるスクリプトは次の通りです。

タイミング	説明	生成されるファイル
アップロード更新(開始時)	アップロード更新処理の開始時に一回だけ呼び出されます。	UploadUpdate<モデルID>_processStart.js
アップロード更新(終了時)	アップロード更新処理の終了時に一回だけ呼び出されます。	UploadUpdate<モデルID>_processEnd.js
アップロード更新(1件読み込み前)8.3.1	アップロード更新でデータを1件読み込む前に呼び出されます。対象データの数だけ呼び出されることになります。新規登録、更新いずれも対象です。[利用例...]	UploadUpdate<モデルID>_readRowDataInFileProcessor.js
アップロード更新(1件読み込み後)8.1.2	アップロード更新でデータを1件読み込んだ直後に呼び出されます。対象データの数だけ呼び出されることになります。ファイルから読み込んだ値を書き換える場合は、このスクリプトを使ってください。なお"データを読み込む"タイミングであるため、新規登録は対象外です。また削除時も呼び出されるようにする場合は「削除時のスクリプトを記述できるようにする」の設定を行ってください。	UploadUpdate<モデルID>_AfterRead.js
アップロード更新(ヘッダ読み込み時)8.3.0	アップロード更新でヘッダ部を読み込む直前に一回だけ呼び出されます。ヘッダ部を自作する場合に利用することができます。	UploadUpdate<モデルID>_readHeader.js

詳細画面表示(アップロード更新)	アップロード更新でデータを1件処理するときに呼び出されます。対象データの数だけ呼び出されることになります。さらに1回の処理で複数回呼び出されることがあるため、このスクリプトで値を変更するような処理を記述しないようにしてください。想定している使い方はイレギュラーデータを発見したときにエラーメッセージをセットすることです。(エラーがセットされた場合、そのデータは処理されません。)	<モデルID>Helper_beforeShowInUploadUpdate.js

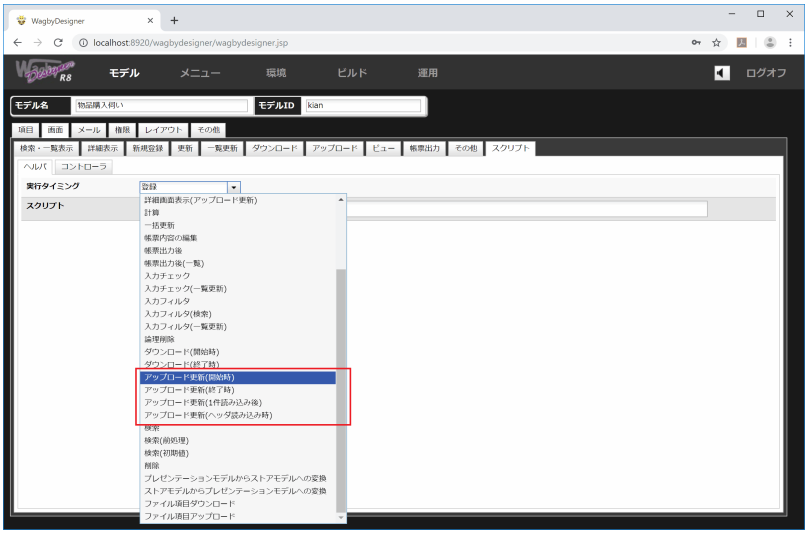


図 ヘルパのスクリプト

アップロード更新(開始時)の詳細

アップロードされたときのファイル名は次のコードで取得できます。

```
p.request.getAttribute("__jfc_uploadupdate_filename")
```

ディスク上に保存されているファイル名は次のコードで取得できます。

```
p.request.getAttribute("__jfc_uploadupdate_filepath")
```

アップロード更新(1件読み込み後)の例

このスクリプトは更新と削除のいずれでも呼び出されるため、どちらかの処理に限定したい場合は UPLOADACTION の値で判断させるようにしてください。

```
if (UPLOADACTION === "update") {
  // 値を変更する処理
}
```

(369)

アップロード更新(ヘッダ読み込み時)の例 (8.3.0)

(369) 削除処理で呼び出されるようにする場合は "画面 > アップロード > アップロード更新設定 > スクリプトで削除を扱う" を有効にしてください。

このスクリプトはヘッダ部を自作したい場合に利用することができます。　具体的には、ヘッダ行のないCSVファイルをアップロードする場合に、スクリプト内で代わりとなるヘッダ行を指定することで実現できます。

(370)

例を示します。

```
var StrArray = Java.type("java.lang.String[]");
var headers = new StrArray(6);
headers[0] = "主キー"
headers[1] = "内容"
headers[2] = "ファイル1"
headers[3] = "URL"
headers[4] = "コンテナ[1]/ID"
headers[5] = "コンテナ[1]/ファイル3コンテナ"
p.request.setAttribute("DbUploadUpdateRunnableMonitor_header", headers);
```

- p.requestにセットするキーDbUploadUpdateRunnableMonitor_headerは予約された特別な名前です。ここに文字列の配列が格納されていた場合、この内容をCSVファイルのヘッダとみなします。
- スクリプトが return で何らかの文字列を返した場合はエラー扱いとなり、以降のアップロード更新処理は行われません。
- このスクリプトがない場合は、標準の動作としてアップロードされたファイルの先頭行をヘッダとみなします。

詳細画面表示(アップロード更新)の例

model1 モデルの item1 項目の値が空でない場合、その行（データ）の処理を失敗させる例を示します。

```
var s = model1.item1;
if (s !== null) {
  var error = new Jfcerror();
  error.content = "値がセット済みです。";
  p.errors.addJfcerror(error);
}
```

エラー処理 (8.3.3)

スクリプトでエラーが発生した時、そのデータの更新および削除処理は失敗します。

ここで "エラー発生" は次のいずれかの方法で実装することができます。

- スクリプト中で BusinessLogicException 例外を発生させる。推奨するコーディング方法。
- スクリプトで return 文字列; とする。文字列は何でもよい。nullでない文字列であればエラー発生とみなす。
- p.errors に一つ以上のエラーメッセージが格納されている。この場合もエラー発生とみなす。

コントローラ

「コントローラ > アップロード更新」で設定できるスクリプトは次の通りです。

タイミング	説明	生成されるファイル
ステータス表示	ステータスを表示するときに呼び出されます。	UploadUpdate<モデルID>_ShowStatus.js
アップロード実行	アップロードの開始ボタンを押下したときに呼び出されます。	UploadUpdate<モデルID>_Update.js
処理の停止	アップロードの停止ボタンを押下したときに呼び出されます。	UploadUpdate<モデルID>_Stop.js
アップロード更新後(1件毎)	アップロード更新で1件のデータの登録または更新が成功し	UploadUpdate<モデルID>_AfterUpdate.js

(370) Wagbyでは、CSVファイルのアップロード時にはヘッダ行が必要になるためです。

	た後に呼び出されます。削除時も呼び出されるようにする場合は「削除時のスクリプトを記述できるようにする」の設定を行ってください。	

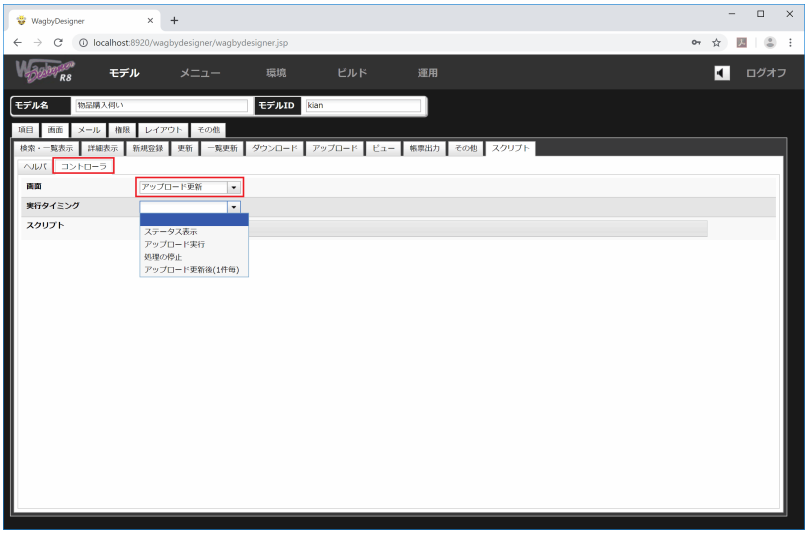


図 コントローラのスクリプト

アップロード更新後(1件毎)の詳細

削除時のスクリプトを記述できるようにする

「コントローラ 〉 アップロード更新 〉 アップロード更新後(1件毎)」に記述したスクリプトは登録または更新時に呼び出されます。 これを削除時にも呼び出されるようにするためには、「画面 〉 アップロード 〉 アップロード更新設定 〉 スクリプトで削除を扱う」を有効にしてください。

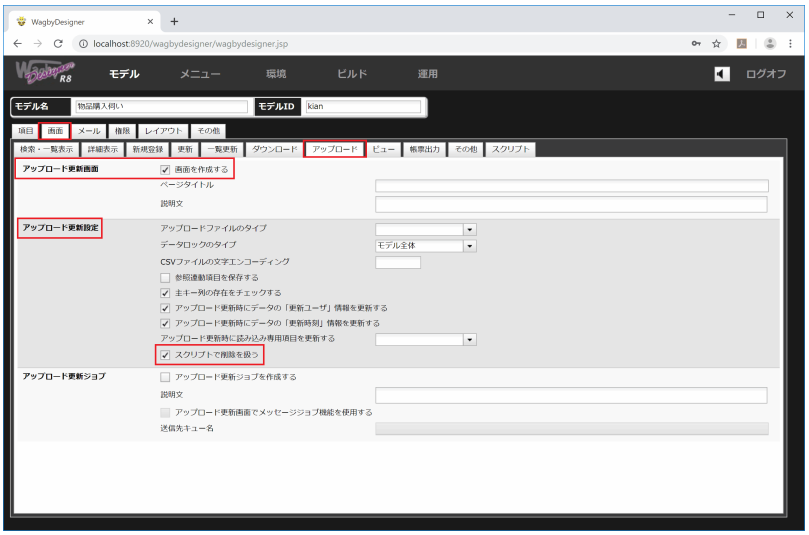


図 削除時のスクリプトを記述できるようにする

(371)

スクリプトで利用できるオブジェクト

このスクリプト内では UPLOADACTION というオブジェクトが利用できます。 現在の処理が登録、更新、削除のいずれかを示します。

(371) この設定を有効にすると、削除前にいったんデータベースから値を読み込むようになります。読み込んだ値をスクリプトに渡します。そのため、削除処理のパフォーマンスに影響があります。（データベース読み込みという手順が削除処理の都度、発生します。）

値	説明
insert	そのデータが新規に登録されたことを示します。
update	そのデータが更新されたことを示します。
delete	そのデータが削除されたことを示します。

- 登録、更新、削除いずれも同じスクリプトが呼び出されます。UPLOADACTION の値をみて、いずれの動作によるものかを知ることができます。
- 更新と削除が混在したアップロード更新では UPLOADACTION を利用することはできません。主キーが指定されており、削除ではない (Status列に "D" を指定していない) 場合、主キーに対応するデータの有無を判定するためにデータベースアクセスが発生します。この時点では新規登録となるか更新となるか不明のため、UPLOADACTION を設定していません。この場合は p.request.getAttribute("UPLOADACTION") の値を使うことができます。更新時は null、削除時は "delete" を取得できます。

例 アップロード更新後にメールを送信する

アップロード対象データにメールアドレスが含まれていると仮定します。アップロード更新成功時に、そのメールアドレスに対してメール送信を行う例を紹介します。

```
var replaceMap = new java.util.HashMap();
replaceMap.put("${customer_p.mailaddress}", customer_p.mailaddress.content);
var mailman = p.appctx.getBean("MailMessageManager");
mailman.send("customer_upload", replaceMap, errorManager, "SendmailAtUploadCustomer ", p);
```

2行目の replaceMap の値部分はプレゼンテーションモデルの項目を使うため、".content" を付与して値部分を取得するようにしてください。

- replaceMapは "メールを送信する" のページでも説明しています。詳細...

仕様

- 処理が失敗した場合は、このスクリプトは呼び出されません。例えばユニーク制約によりデータベース例外が発生し、更新が失敗した、などがあります。
- スクリプト中に、モデルの値を変更することはできません。（変更しても無視されます。）スクリプトは登録、更新処理の「後」に呼び出されるためです。
- アップロード更新の削除処理を有効にした場合でも、削除のスクリプト <モデルID>Helper_beforeDelete.js は呼び出されません。

固定長ファイルのダウンロードとアップロード更新(スクリプト)

ダウンロード

ダウンロードするファイルからコンマを取り除き、固定長レコードに書き換えます。 "スクリプト>ヘルパ" タブの実行タイミング "ダウンロード(1件出力前)" にスクリプトを用意します。

このスクリプトは、ダウンロードファイルのタイプがCSV, TSVの場合にのみ実行されます。Excel(xls),Excel(xlsx)の場合は実行されません。

```
function process() {// 注意:宣言部が必要

  // data は、もともと、出力しようとしていたデータが格納された変数。
  // 以下のコードはデバッグ用。
  /*
  for (var i=0; i<data.length; i++) {
    print(data[i]);
  }
  */

  // 固定長のバイトサイズ。開発者が適切な値を指定すること。
  var ColumnSize = [
    6, 8, 8, 8, 8, 8
  ];
```

```
  if (data.length != ColumnSize.length) {// 個数があわない場合はエラーとする。
    //print("Illegal size " + data.length);
    return;
  }

  var recordsize = 0;
  for (var i=0; i<ColumnSize.length; i++) {
    recordsize += ColumnSize[i];
  }
  recordsize = recordsize + 2; // リターンコード(CRLF)のため2バイト多く確保する。

  var ByteArray = Java.type("byte[]");
  var bdata = new ByteArray(recordsize);

  var cnt = 0;
  for (var i=0; i<ColumnSize.length; i++) {
    var bdata1 =
      Packages.jp.jasminesoft.util.ExcelFunction.STR2BYTEA(
        data[i], "MS932", ColumnSize[i], 0x20); // 0x20 はスペース文字。
    java.lang.System.arraycopy(bdata1, 0, bdata, cnt, ColumnSize[i]);
    cnt += ColumnSize[i];
  }
  bdata[cnt++] = 0xd; // CR
  bdata[cnt++] = 0xa; // LF
  ostream.write(bdata);
}

function getWriteType() {
  return "OutputStream";
}
```

- getWriterType 関数を上のように記述してください。すると process 関数で暗黙変数 ostream が利用できます。固定長レコードをバイトの配列として出力させるための write メソッドを利用してください。
- getWriterType 関数を定義するため、関数 process の宣言が必要です。そのためスクリプトの先頭は function process から始まっています。[詳細：基本テンプレートを使わない...]
- 文字列から指定した大きさのバイト配列を作成するためのメソッドExcelFunction.STR2BYTEAを用意しています。第二引数は文字コード、第三引数は長さ（バイト）、第四引数は固定長を埋める文字で、通常は空白文字を指定します。

利用できる暗黙変数

このスクリプトで利用できる暗黙変数は次の通りです。

変数名	説明
ostream	バイト配列を出力するために用意されている。この実体はjava.io.BufferedOutputStreamクラスのインスタンス。
csvencoding	CSVファイルの文字エンコーディング。
sepChar	列の区切り文字。デフォルトはコンマ(,)やタブ(¥t)
repChar	列をくくる文字。デフォルトはダブルクォーテーション(")。Designerの設定「カラムのくくり文字」にて「(なし)」に変更できる。この場合は'¥¥u0000'となる。

アップロード更新

アップロードされるファイルの各行を固定長レコードとみなし、レコードをコンマ区切りの列に分解したあとで、Wagbyのアップロード更新処理に渡すようにします。 "スクリプト>ヘルパ" タブの実行タイミング"アップロード更新(1件読み込み前)" にスクリプトを用意します。

このスクリプトは、アップロードファイルがCSV,TSVファイル（もしくはZIPファイルの場合、展開後に含まれるCSV,TSVファイル）の処理中に実行されます。Excel(xls),Excel(xlsx)の場合は実行されません。

```
// 固定長のバイトサイズ。開発者が適切な値を指定すること。
var ColumnSize = [
  6, 8, 8, 8, 8, 8
];
var recordsize = 0;
for (var i=0; i<ColumnSize.length; i++) {
  recordsize += ColumnSize[i];
}
recordsize = recordsize + 2; // リターンコード(CRLF)のため2バイト多く確保する。

var ByteArray = Java.type("byte[]");
var bdata = new ByteArray(recordsize);
var readsize = istream.read(bdata);
if (readsize == -1) {// End Of File の判定。
  return "EOF";
}
if (readsize <= 0) {// 読み込みエラーの可能性。
  return;
}
if (readsize != recordsize) {// 想定していない固定長レコードは無視する。
  return "Illegal record size " + readsize;
}

var StrArray = Java.type("java.lang.String[]");
var data = new StrArray(ColumnSize.length);
var cnt = 0;
for (var i=0; i<ColumnSize.length; i++) {
  data[i] = new java.lang.String(bdata, cnt, ColumnSize[i], "MS932").trim();
  print(data[i]);//デバッグ用
  cnt += ColumnSize[i];
}

p.request.setAttribute("DbUploadUpdateRunnableMonitorFileProcessor_rowData", data);
```

- この関数は process のみであるため（ダウンロードのスクリプトとは異なり）宣言部は省略できます。
- ファイルを読み込むための暗黙変数 istream が利用できます。read メソッドを使って読み込んでください。
- 読み込んだ値を data という配列に格納し、DbUploadUpdateRunnableMonitorFileProcessor_rowDataというキー名で p.request にセットしてください。
- このスクリプトが （return文で） 何らかの文字列を返したとき、その行の処理はエラーとなり、スキップされます。（アップロード更新処理自体は継続され、次の行の処理を行います。）

利用できる暗黙変数

このスクリプトで利用できる暗黙変数は次の通りです。

変数名	説明
istream	1レコードを読み込むために用意されている。この実体はjava.io.BufferedInputStreamクラスのインスタンス。

帳票

HTML帳票

一覧表示

一覧表示画面に用意された「印刷形式で表示」ボタンを押下します。

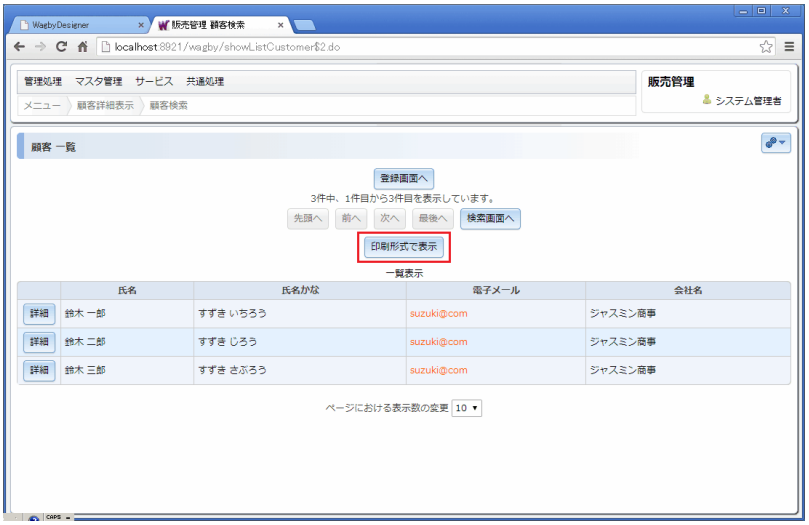


図 一覧表示画面の「印刷形式で表示」ボタン

別タブが開きます。メニューが取り払われた、データのみの表示が行われます。

図 印刷形式（一覧表示）

この画面をWebブラウザの印刷機能を用いて出力してください。

定義方法

「画面機能>検索・一覧表示>ボタン表示」内にある「印刷形式で表示」ボタンを有効にします。

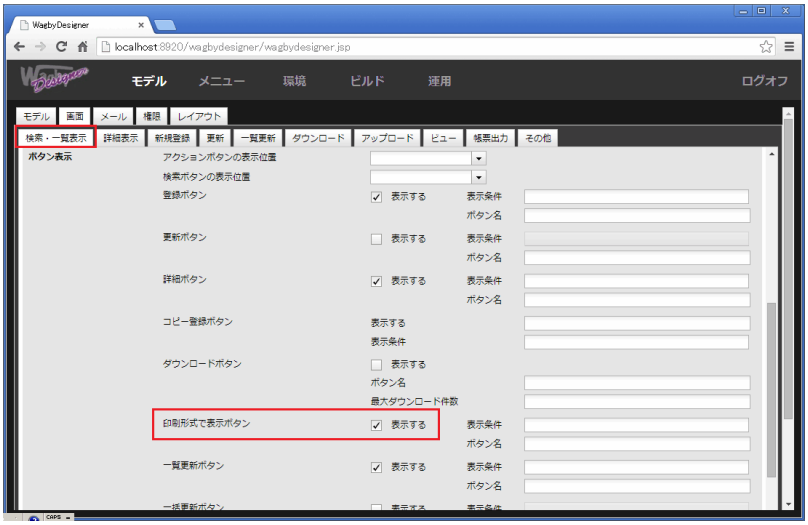


図 印刷形式で表示ボタンを有効にする

「印刷形式で表示」ボタンは、検索結果が存在する場合のみ表示されます。検索結果が存在しない場合、ボタンも非表示となります。

詳細表示

詳細表示画面に用意された「印刷形式で表示」ボタンを押下します。

図 詳細表示画面の「印刷形式で表示」ボタン

別タブが開きます。メニューが取り払われた、データのみの表示が行われます。

図 印刷形式（詳細表示）

この画面をWebブラウザの印刷機能を用いて出力してください。

定義方法

「画面機能>詳細表示>ボタン表示」内にある「印刷形式で表示」ボタンを有効にします。

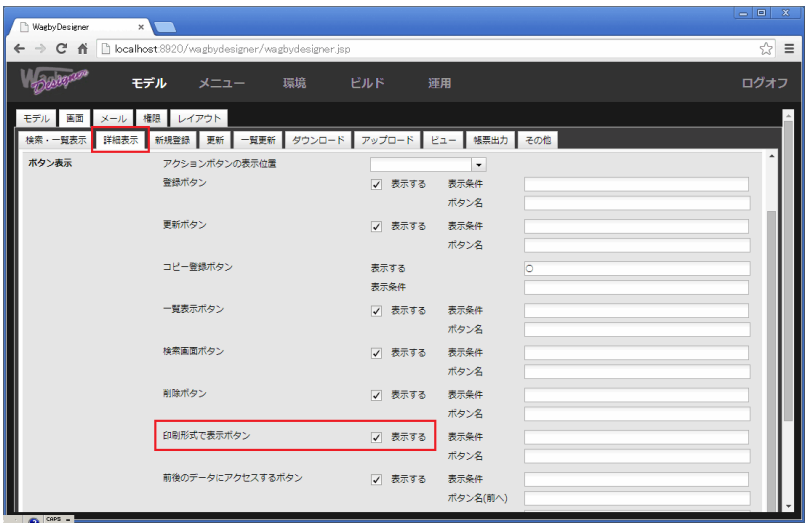


図 印刷形式で表示ボタンを有効にする

Excel帳票 － 詳細表示

例

Excel帳票機能を設定した顧客モデルの詳細表示画面を開きます。図に示すように「帳票」ボタンが用意されています。

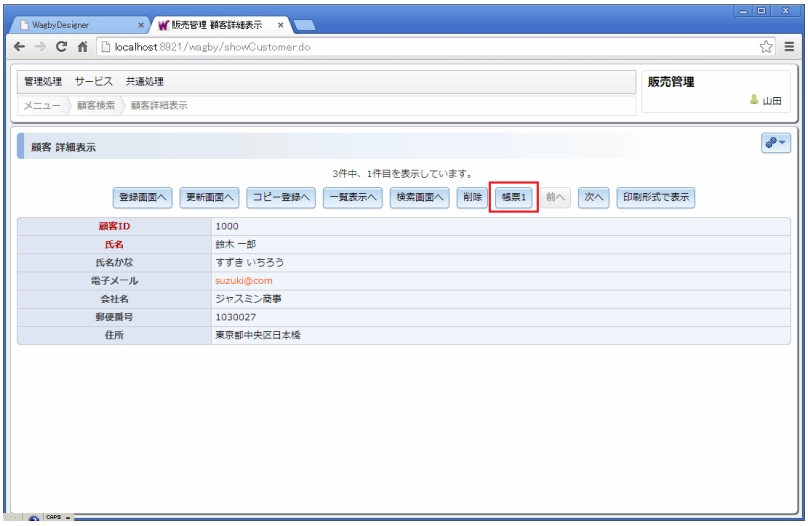


図 詳細画面に用意された帳票ボタン

このボタンを押下すると、Excel形式のファイルがダウンロードされます。

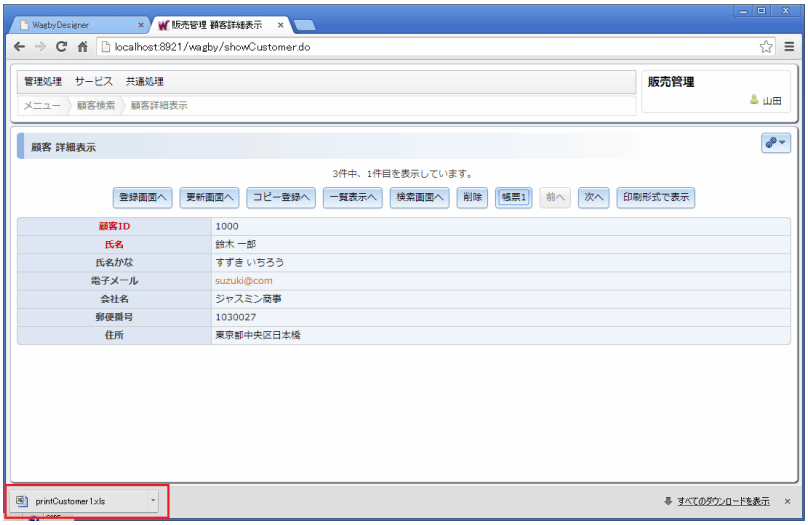


図 Excelファイルがダウンロードされる

Excelでファイルを開きます。この顧客の詳細情報がレイアウトされていることがわかります。これをプリンタで印刷することで帳票出力を行います。

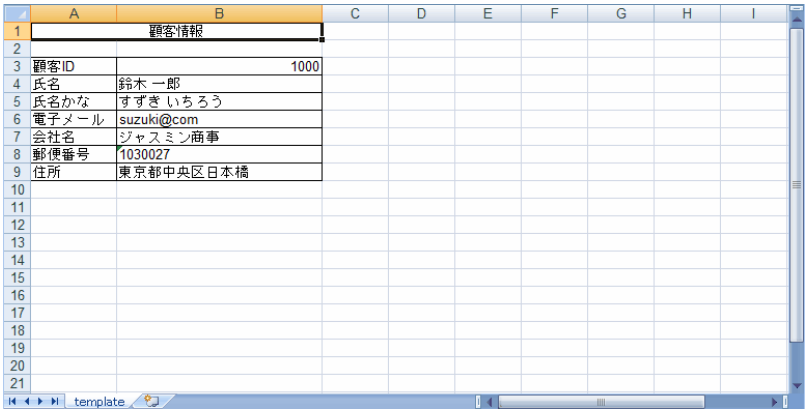


図 Excelに埋め込まれた詳細情報

ライブラリの選択

Excel帳票出力機能は、オープンソースのExcel生成ライブラリである Apache POI を使って実現しています。

仕様・制約

テンプレートファイル

- テンプレートファイルとして用いるExcelファイルの拡張子は ".xls" または ".xlsx" となります。
- テンプレートファイルのファイル名は半角英数字のみとしてください。日本語文字は指定できません。
- テンプレートの Excel ファイルでは、以下の機能を利用することはできません。
  - ピボットテーブル
  - ドロップダウンリスト
  - パスワード保護
- テンプレートの Excel ファイルにマクロを設定した場合、正しく出力されないことがあります。この場合はマクロの内容を変えるなどして調整してください。
- 帳票(Excel)側の　"表示形式"　を、Wagby定義ファイルで指定することはできません。帳票へ出力する"型"は定義ファイルの型の設定に準じます。帳票データの表示形式は、帳票テンプレートファイル(Excel)の"セルの書式設定"で行ってください。
- 書式設定を指定したにも関わらず、有効にならない場合があります。その場合は、データ出力用のセルまたはシートと、帳票テンプレートの部分と分け、テンプレートでは出力されたデータを参照するよう設定を行ってみてください。詳細は"テンプレートファイル作成のポイント"をお読みください。データ出力用シートと、帳票テンプレートのシートを分ける方法を紹介しています。
- うまく印刷されない場合、帳票テンプレートの最初のシートに印刷領域が設定されているかどうかを確認ください。
- 帳票テンプレートの「印刷範囲」にマクロなどの式を設定することはできません。（設定しても Wagby では式の実行ができないため、無視されます。）
- プレースホルダから1つのセルに値をセットする場合、上限が32,767文字となります。

テキストエリア型の扱い

テキストエリア型の項目を Excel のセルに割り当てる場合、次の設定を行なってください。

- 当該セルの “セルを折り返して全体を表示する” を有効にする。
- セルの高さを事前に調整する。

文字列に改行コードが含まれていた場合は、改行コードもそのまま出力されます。

テンプレートファイルに .xlsx を利用した場合

拡張子 ".xlsx" 利用時には次の制約があります。

- テンプレートファイルにセルコメントを含めることができません。

実行環境

- Wagby アプリケーションを稼働させるサーバ機に Excel をインストールする必要はありません。サーバ機のメモリ上で直接 Excel ファイルを生成します。そのため出力処理中は一時的に多くのメモリを消費します。同時接続数が多い環境で、帳票出力が集中するような運用環境では、メモリを多めに搭載するようにしてください。
- Excel ファイルを開くために、各利用者の PC（端末）には Excel または OpenOffice が必要です。

トラブルシューティング

IE11利用時、ダウンロード時の保存、ファイルを開くボタンを素早く押すと"ファイルが移動または削除された可能性があります"というダイアログが表示される。

"CSV・Excel入出力 > ダウンロード > トラブルシューティング"をお読みください。

定義方法

Excel帳票の定義は、次の手順で行います。

1. Designerによるボタン設定とビルド。
2. ビルドによって得られたExcelテンプレートファイルのレイアウト設計
3. （ビルドした）Wagbyアプリケーションで行う「帳票テンプレート」の設定

ここでは、それぞれの手順の詳細を説明します。

Designerの設定

「画面>帳票出力」タブを開きます。帳票出力設定欄で、次の設定を行います。

設定項目	説明	入力例
テンプレートファイル名	後述する「帳票テンプレート」の識別子になります。重複しない名前を英数字の組み合わせで決めてください。	printCustomer1
表示画面	詳細表示を選択します。	詳細表示
出力方式	Excelを選択します。	Excel
利用ライブラリ	出力方式に Excel を指定した場合、POIを指定します。Wagbyに同梱されているPOIのバージョン...	POI
表示する	この欄をチェックすると、帳票出力が有効になります。	-
表示条件	ボタンの表示条件式を記述することができます。「画面機能>ボタン名と表示条件のカスタマイズ」で説明しています。	（中略）
ボタン名	ボタン名を指定します。省略時は「"帳票" + 連番」となります。式を使ってボタン名を決定することもできます。	-

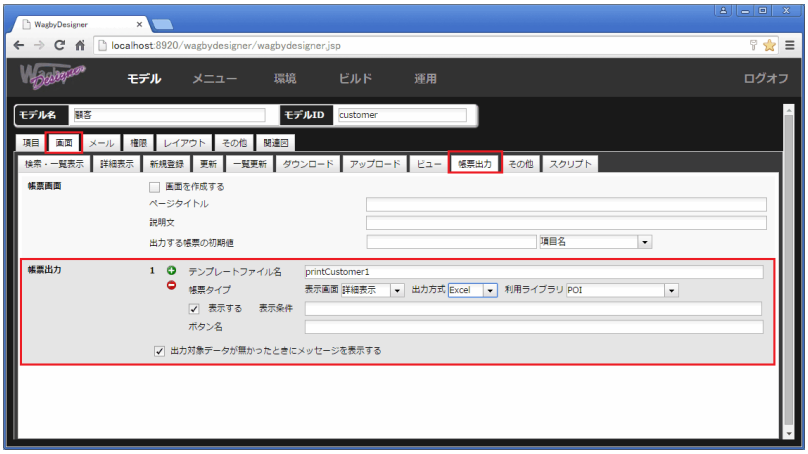


図 帳票出力の設定

帳票設定数の上限はありません。「+」ボタンを押下すると、設定欄が追加されます。

Excelテンプレートファイルのレイアウト設計

図の設定後にビルドすると、Wagbyインストールフォルダ内のwork/reportフォルダ内に、テンプレートファイル名で指定したExcelファイルが自動生成されます。

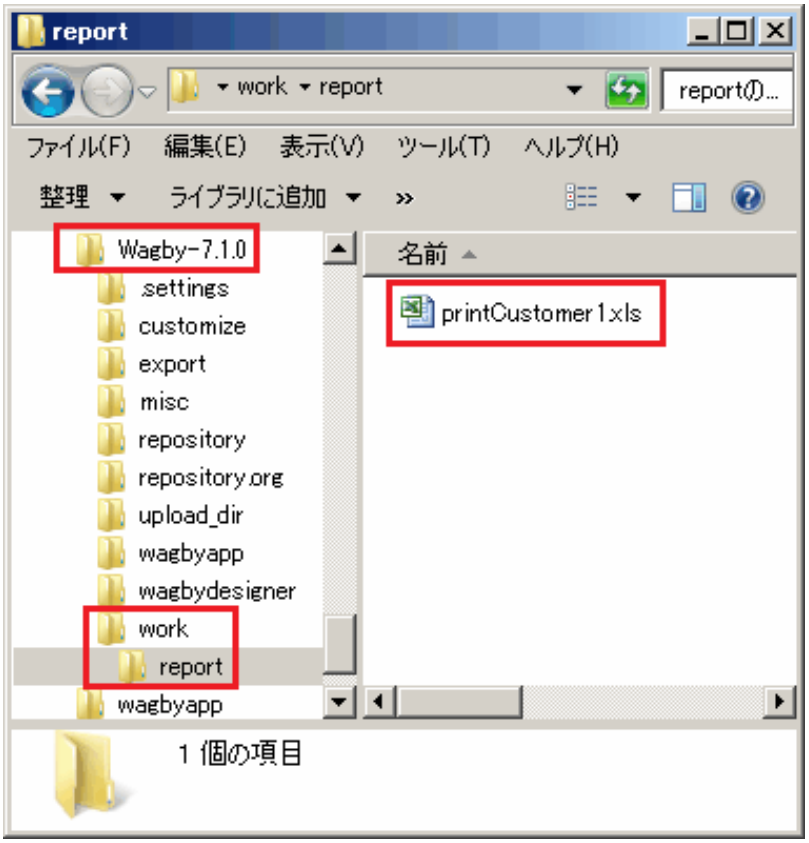


図 自動生成されたテンプレートファイル

テンプレートファイルの形式 (8.3.0)

自動生成されるテンプレートファイルは次のルールに従います。

	R8.2.3まで	R8.3.0以降
テンプレートファイルの拡張子を指定しない	.xlsファイル	.xlsxファイル
テンプレートファイルの拡張子を.xlsxとする	.xlsファイル	.xlsxファイル
テンプレートファイルの拡張子を.xlsとする	.xlsファイル	.xlsファイル

このテンプレートファイルをExcelで開くと、図のようになっています。レイアウト設計なしで、項目名とプレースホルダが列挙されています。

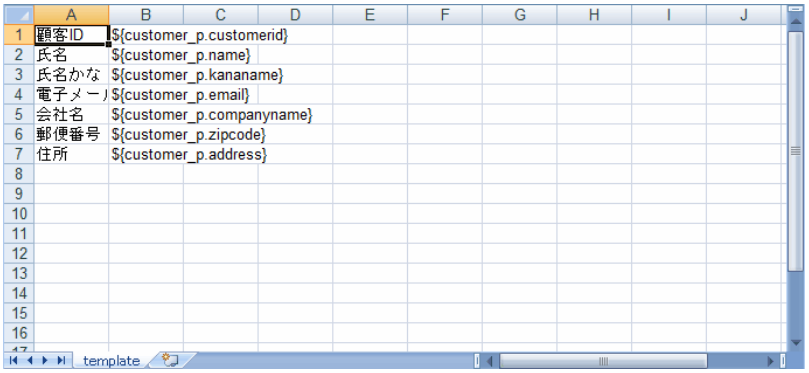


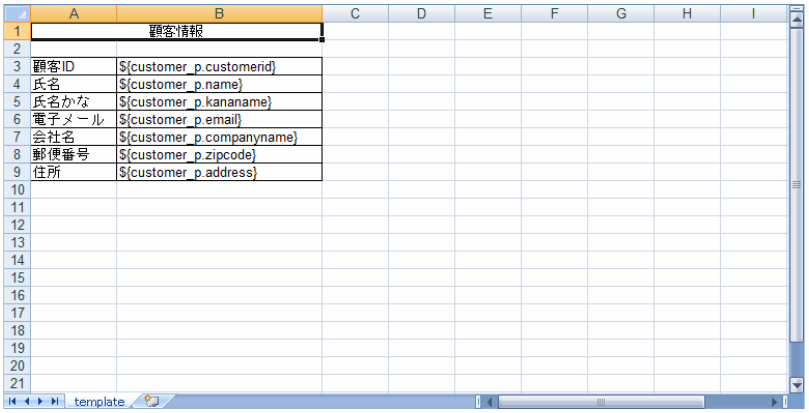
図 項目名とプレースホルダが埋め込まれたExcelファイル

ここで "${" ではじまり、"}" で終わる部分をプレースホルダと呼びます。プレースホルダの構造は次のとおりです。

${モデル名_p.項目名}

プレースホルダには、実際の値が設定されます。プレースホルダの記述ルールの詳細は"プレースホルダ"をお読みください。

このプレースホルダ表記を残しつつ、帳票レイアウト設計を行います。図にレイアウトの例を示します。



	A	B
1	顧客情報	
2		
3	顧客ID	${customer_p.customerid}
4	氏名	${customer_p.name}
5	氏名かな	${customer_p.kananame}
6	電子メール	${customer_p.email}
7	会社名	${customer_p.companyname}
8	郵便番号	${customer_p.zipcode}
9	住所	${customer_p.address}

template

図 帳票レイアウトを行った例

帳票テンプレートの設定

ビルドしたアプリケーションにシステム管理者でログオンします。モデルと、図の帳票ファイルを紐付けるために「帳票テンプレート」の設定を行います。

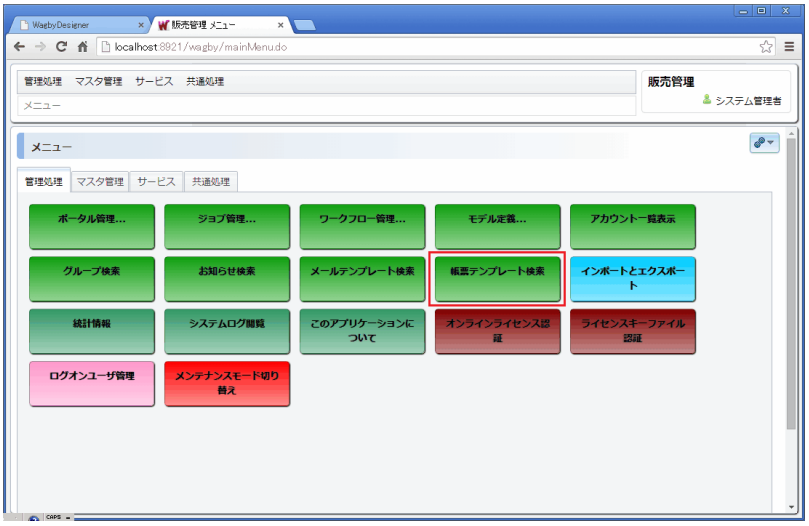


図 帳票テンプレートメニュー

新しい帳票テンプレートを作成します。

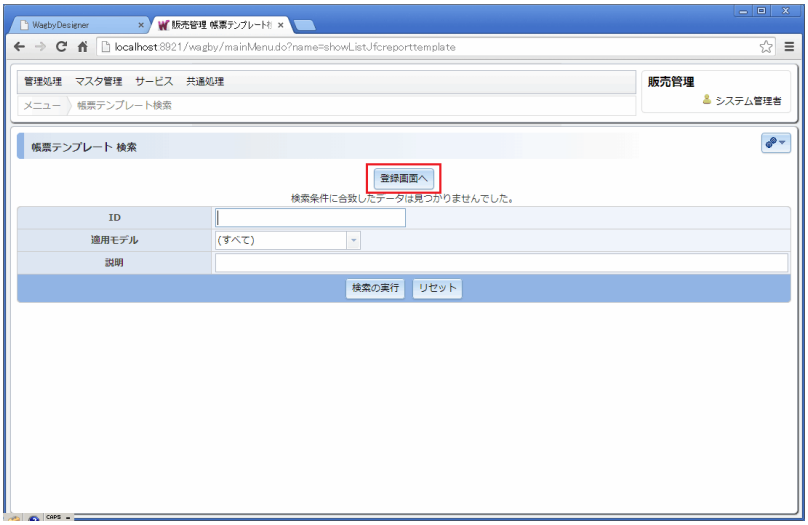


図 帳票テンプレートの新規登録を行う

ID項目は、Designerに設定した帳票テンプレートIDを指定します。拡張子 （.xls や .xlsx） は含めません。

ここでは適用モデルを”顧客”とします。説明欄は自由に設定してください。

テンプレートファイルとして、図を保存したExcelファイルを指定します。

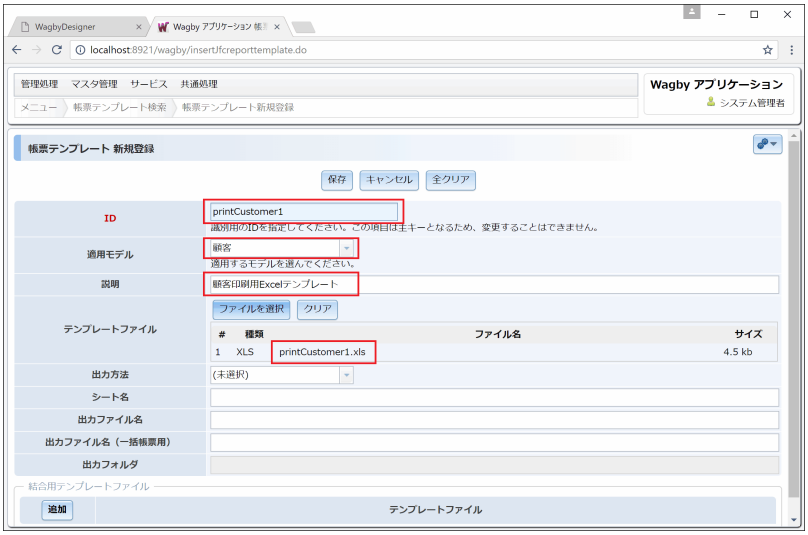


図 帳票テンプレートの設定

図 帳票テンプレートを登録した

ここで登録したテンプレートファイルは、upload_dir フォルダ以下に保存されます。詳細は"文字列 > ファイル > 仕様 > 内部の構造"をお読みください。

ボタン名を変更する

例

詳細表示画面に用意される帳票用ボタン名を "Excel印刷" に変更した例を図に示します。

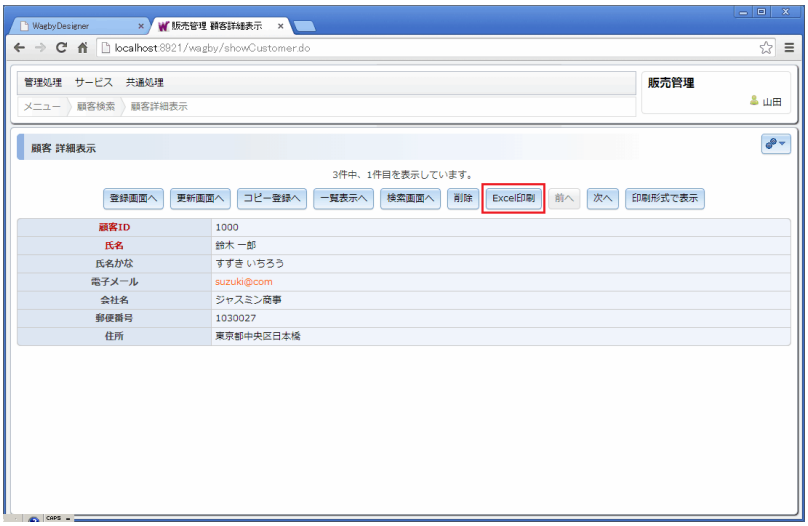


図 帳票ボタン名の変更

定義方法

Designer側でボタン名を指定します。

図 Designerでボタン名を設定する

出力方法を変更する

帳票テンプレートで、出力方法を指定することもできます。空白時は″ダウンロード″となります。

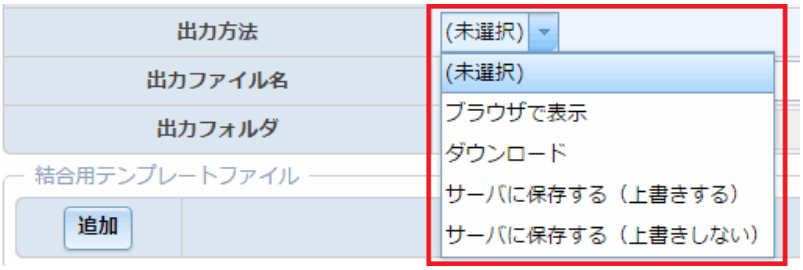


図 出力方法の指定

ダウンロード

ファイルをダウンロードします。

(372)

ブラウザで表示

ファイルをブラウザで表示します。

サーバに保存

帳票ファイルを（Wagbyアプリケーションが稼動している）サーバ上の指定フォルダに書き込みます。この設定を行ったとき、「出力フォルダ」をあわせて指定してください。サーバ上のフォルダを指定します。

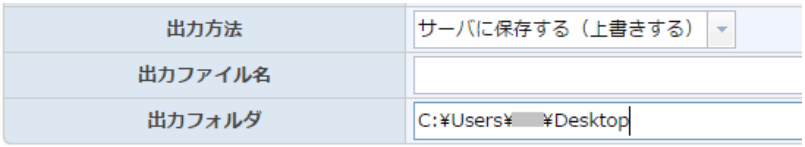


図 出力フォルダの指定

″サーバに保存（上書きする）″を選択した場合、すでに同名の帳票が出力済みの場合、ファイルを上書きします。

″サーバに保存（上書きしない）″を選択した場合、すでに同名の帳票が出力済みの場合、ファイル名に″_1″″_2″...を付与します。(*)

ファイルを保存したあと、画面にメッセージを表示します。保存に失敗すると、画面にエラーメッセージを表示します。

(373)

ファイル名に用いるプレースホルダ

詳細帳票テンプレートのファイル名に次の書式を用いることができます。

プレースホルダ	説明	例
${username}	ログオンアカウント	${username}.xls

(372) Internet Explorer 利用時、出力方法に″ダウンロード″を指定した場合にダイアログが二重に表示されることがありますが、動作に支障はありません。

(373) 同名のファイルを見つけると、ファイル名に″_1″を付与します。もし″_1″も存在すれば、数字を増加させます。この仕組みのため、飛び番号にはなりません。最大の数字が付与されたファイルが最新のファイルです。

${createdatestr}	出力日付時刻を表す文字列。yyyyMMddHHmmss形式。	${createdatestr}.xls

プレースホルダは組み合わせることができます。例えば ${username}${createdatestr}.xls と指定することができます。

出力ファイルを変更する

帳票テンプレート設定で、出力ファイル名を指定することができます。固定値を設定します。

シート名

出力される Excel のシート名になります。

- 詳細帳票、一括帳票いずれにも反映されます。
- プレースホルダを利用できます。
- プレースホルダとして特殊表記 ${sheet} を利用できます。シート番号を示します。
- 拡張子は含めないようにしてください。
- この設定は、PDF出力では無効です。(無視されます。)
- Excelのシート名に指定できない文字（コロン、円記号、疑問符、角カッコ、スラッシュ、アスタリスク）を含めることはできません。これらが含まれていた場合、自動的に除去されます。
- 上のURLに記載の内容に加え、"(","）" （小括弧） を含めることはできません。R8.3.3以降のWagbyでは、この文字が含まれていた場合、自動的に消去するようになっています。

出力ファイル名

出力される Excel のファイル名になります。また、一括帳票を展開したときの各Excelファイルのファイル名になります。

- ファイル名にはプレースホルダ利用できます。
- ファイル名には拡張子を含めてください。(.xlsまたは.xlsx)

(374)

出力ファイル名（一括帳票用）

複数データの帳票を一度に出力するときのファイル名になります。

- 一括でダウンロードされるファイル名は固定です。プレースホルダは適用されません。
- 拡張子は含めないようにしてください。
- 漢字も利用できます。
- "ZIPファイル(1データ1Excelブック)"形式の場合、解凍後の各ファイル名のルールは上記「出力ファイル名」に準拠します。
- "Excelファイル(1データ1Excelシート)"形式の場合、シート名のルールは上記「シート名」に準拠します。

(375)

定義方法

プレースホルダを指定することができます。ファイル名は、拡張子も含めて指定してください。例を示します。

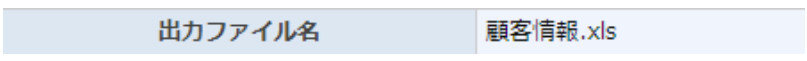


図 出力ファイル名の指定(1)

(374) この設定は、HTTPレスポンスヘッダの"Content-Disposition"フィールドの"filename"属性に反映されます。

(375) この設定は、HTTPレスポンスヘッダの"Content-Disposition"フィールドの"filename"属性に反映されます。

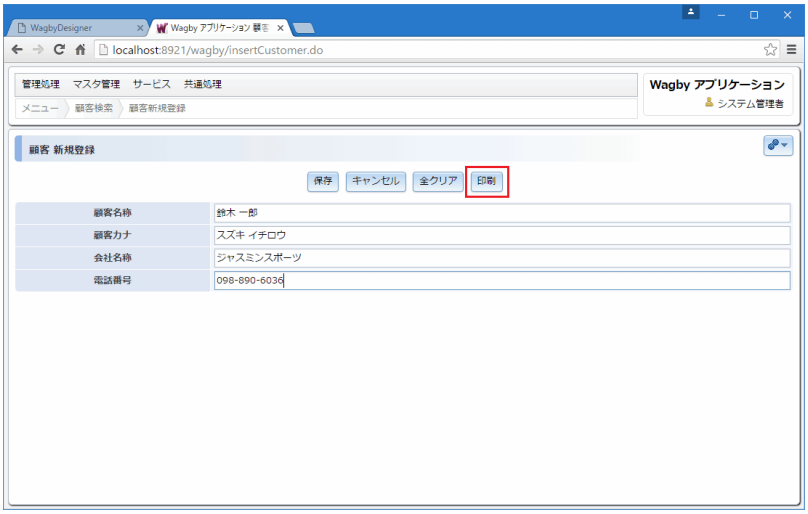


図 出力ファイル名の指定(2)

顧客情報.xls

顧客情報${customer_p.customerid}.xls

出力ファイル名が空欄の場合、テンプレートファイル名が使われます。

出力ファイル名に利用するプレースホルダは、モデル名も含めるようにしてください。また、式の記述はできません。（"+"などの文字列演算を行うことはできません。） （誤）${name}+"_"+${currenttime}+"_"+${userid}+".xls" （正）${customer_p.name}_${customer_p.currenttime}_${customer_p.userid}.xls

出力ファイル名を計算式によって決めたい場合、出力ファイル名という項目をモデル定義内に用意するとよいでしょう。その項目名を、本設定欄に記載します。例 ${customer_p.generatedfilename}

テンプレートファイルが .xls 形式ファイルだが、出力ファイル名を .xlsx とする、といったことはできません。出力ファイル名を指定するときは、テンプレートファイルの拡張子を使ってください。

繰り返し項目の固定長指定

詳細帳票テンプレートにおいて、繰り返し項目（コンテナを含む）や外部キー子モデルの項目は、テンプレートに記述した分だけ出力されます。

例を示します。下図のテンプレートにおいて、商談履歴の項目は繰り返しコンテナ項目となっています。

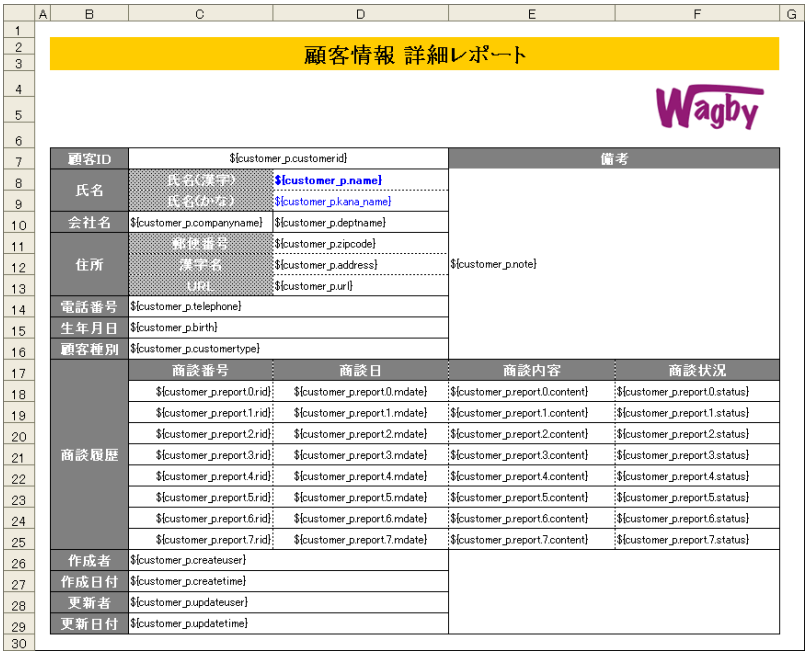


図 顧客情報の詳細帳票用テンプレート

この繰り返しコンテナ項目には ${customer_p.report.0.rid} の行から ${customer_p.report.7.rid} の行までがプレースホルダとして指定されています。

商談番号	商談日	商談内容	商談状況
${customer_p.report.0.rid}	${customer_p.report.0.mdate}	${customer_p.report.0.content}	${customer_p.report.0.status}
${customer_p.report.1.rid}	${customer_p.report.1.mdate}	${customer_p.report.1.content}	${customer_p.report.1.status}
${customer_p.report.2.rid}	${customer_p.report.2.mdate}	${customer_p.report.2.content}	${customer_p.report.2.status}
${customer_p.report.3.rid}	${customer_p.report.3.mdate}	${customer_p.report.3.content}	${customer_p.report.3.status}

図 テンプレートの商談履歴部分-固定長指定

DB上にレコードが1個しかない場合は、${customer_p.report.0.rid} の行はレコードの値が挿入され、${customer_p.report.1.rid} の行以降は空白に置換されます。

一方、DB上にレコードが9個以上ある場合は、最初のレコードから8個目までは帳票に出力されますが、9個目以降は出力されません。この場合は、あらかじめテンプレートの行数を増やし、プレースホルダも同じく追加することで対応してください。

繰り返し項目の可変長指定

繰り返し項目（コンテナ含む）のデータ数によって表示レイアウトを伸縮させる方法を説明します。

プレースホルダに次のような表記を行います。

```
${モデル名_p.コンテナ名.#.項目名}
```

例えば ${customer_p.report.#.rid} や ${customer_p.report.#.mdate} と記述します。これを一行にまとめて記述します。（複数行にまたがって記述することはできません。）

商談番号	商談日	商談内容	商談状況
${customer_p.report.#.id}	${customer_p.report.#.mdate}	${customer_p.report.#.content}	${customer_p.report.#.status}

図 テンプレートの商談履歴部分-可変長指定

一つのテンプレートに複数の繰り返しコンテナを指定することもできます。出力する数に応じて帳票イメージが伸縮します。

セル結合の制約

繰り返しコンテナが一つも登録されていないデータの帳票出力でもセル結合がずれないように、コンテナ部に空のデータ行が1件、表示されます。（これによってセル結合が動作します。）

本方式を用いた場合、繰り返しコンテナのある値の合計を求めるために、セル中に（Excelの）=SUM(D4:D5)などと記載することができません。SUM関数のセル範囲は固定で指定されているためです。（この部分を伸縮させることには対応できていません。） このような場合は別途、Wagby側で合計を求めておき、それをExcel帳票に転記するようにしてください。 もしくは繰り返しコンテナを固定長として利用してください。（SUM関数の対象となるセル範囲と合わせることで、正しく動作します。）

外部キー連携モデルの扱い

詳細帳票テンプレートに「外部参照されているデータ」（外部キーを利用したデータ）を表示させる場合も、繰り返し項目と同様に指定した数だけ表示されます。

外部参照データの場合、プレースホルダは外部参照のモデル名となります。 また、詳細画面における外部参照モデルは一覧表示用のモデルを用います。モデル名は "xxx_lp" となります。

例えば顧客モデルにサポートモデルを外部キーで関連付けている場合、次の表記となります。末尾の数字は固定指定による方式となります。

```
${support_lp.xxx.添字}
```

(376)

顧客ID	${customer_p.customerid}	
氏名	氏名(漢字)	${customer_p.name}
	氏名(かな)	${customer_p.kana_name}
会社名	${customer_p.companyname}	${customer_p.deptname}
住所	郵便番号	${customer_p.zipcode}
	漢字名	${customer_p.address}
	URL	${customer_p.url}
電話番号	${customer_p.telephone}	
生年月日	${customer_p.birth}	
職種	${customer_p.job}	

サポート	購入年月日	保証開始日	サポート内容
	${support_lp.purdate.0}	${support_lp.assudate.0}	${support_lp.content.0}
	${support_lp.purdate.1}	${support_lp.assudate.1}	${support_lp.content.1}
	${support_lp.purdate.2}	${support_lp.assudate.2}	${support_lp.content.2}
	${support_lp.purdate.3}	${support_lp.assudate.3}	${support_lp.content.3}
	${support_lp.purdate.4}	${support_lp.assudate.4}	${support_lp.content.4}
	${support_lp.purdate.5}	${support_lp.assudate.5}	${support_lp.content.5}
	${support_lp.purdate.6}	${support_lp.assudate.6}	${support_lp.content.6}
	${support_lp.purdate.7}	${support_lp.assudate.7}	${support_lp.content.7}

図 顧客の詳細帳票内に「サポート」という外部キー連携のモデルの内容を表示させる

自動生成されるテンプレートファイル

詳細帳票(親側のモデル)に含まれる、外部キー子モデルのテンプレートファイルは showList<子>LpIn<親>.xls という命名規則で生成されます。図に示す work/report フォルダに用意されます。

このファイルのテンプレート部分をコピーし、親側のモデルのテンプレートに貼り付けることで、図のようなレイアウトを作成することができます。

(376) 添字は 0 から開始してください。

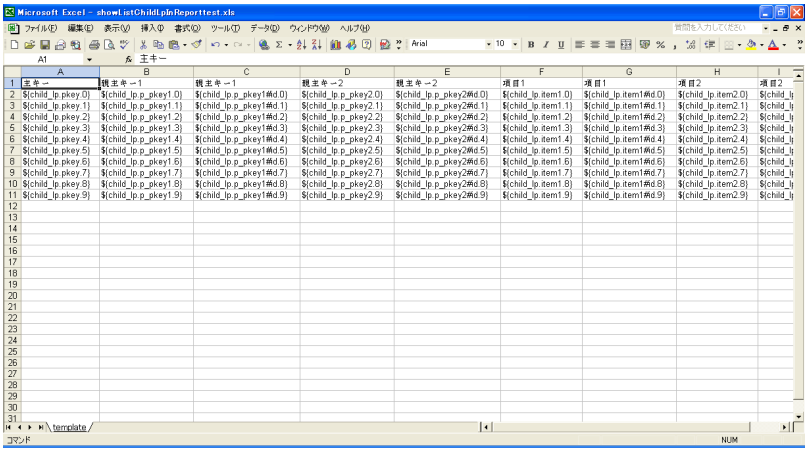


図 詳細帳票に含めるための、外部キー子モデルのテンプレートファイル

ワークフロー（フローイベント）の扱い

モデルの詳細画面の下部に表示される「申請状況」の実体は「フローイベント」というものです。これは管理処理メニューから「ワークフロー 〉 フローイベント検索」を開くことで確認できます。

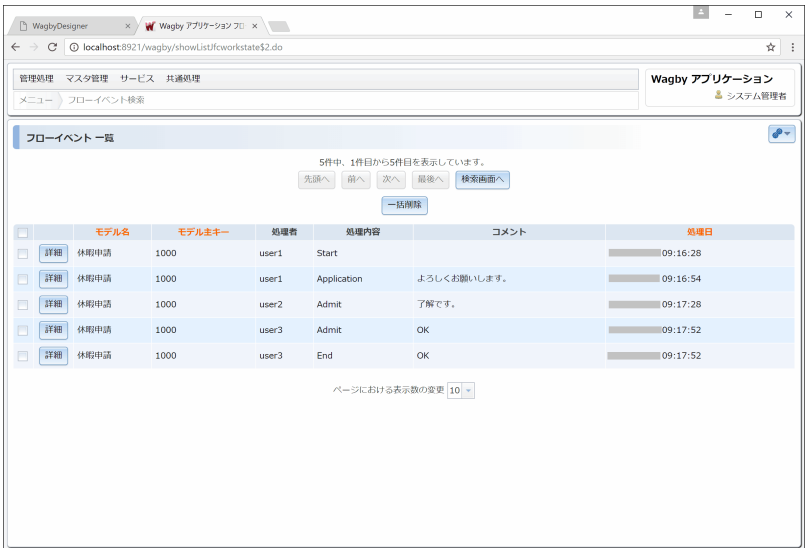


図 フローイベント一覧

詳細帳票に、このフローイベントを含むことができます。プレースホルダは次の通りです。

プレースホルダ	説明	例
${jfcworkstate_lp.id.添字}	フローイベントの主キー	1000
${jfcworkstate_lp.modelname.添字}	モデル名（日本語）	年休申請
${jfcworkstate_lp.modelpkey.添字}	対象モデルの主キー。	1000
${jfcworkstate_lp.username.添字}	処理者のアカウント	user1
${jfcworkstate_lp.processed_la添字}	フローの階層	0,1,2,..
${jfcworkstate_lp.event.添字}	イベント	Start,Application,...
${jfcworkstate_lp.comment.添字}	ユーザが入力したコメント	”よろしくお願いします。”

${jfcworkstate_lp.insert_date.添字}	処理日（日付時刻）	2017-09-01 10:00:00

(377)

登録・更新画面に帳票ボタンを用意する

「帳票出力 > 表示画面」の選択肢で"新規登録", "コピー登録", "更新"を指定することができます。登録や更新画面に帳票出力ボタンが用意されます。

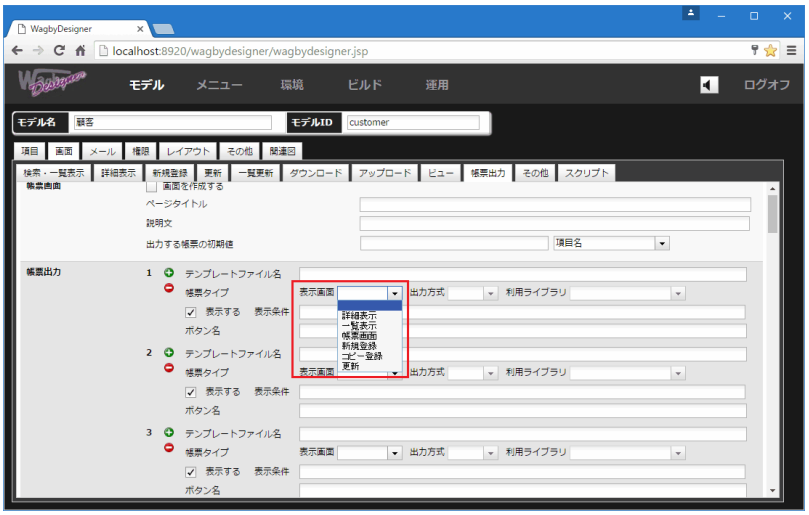


図 帳票出力>表示画面の設定(1)

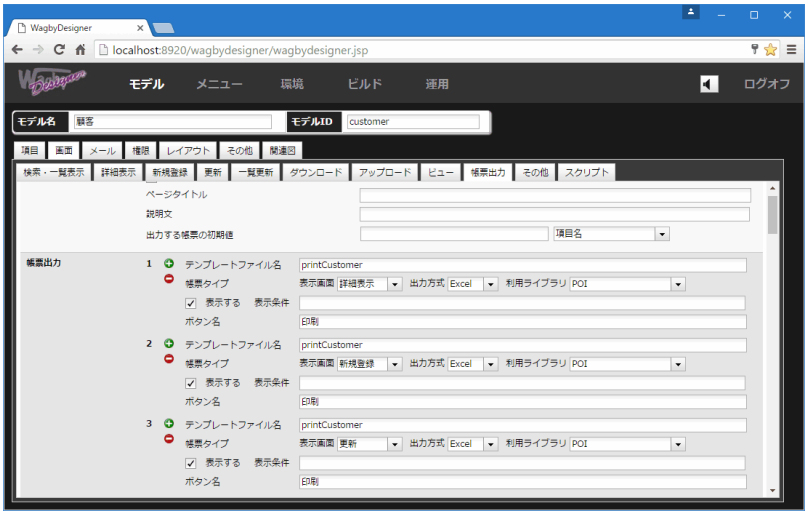


図 帳票出力>表示画面の設定(2)

例

登録画面や更新画面に用意された帳票出力ボタンは次の動作となります。

- データベースに保存されている値ではなく、画面上の（変更途中の）値が帳票出力されます。
- ボタン押下後は、自画面（登録画面や更新画面）に戻ります。

(377) 添字は 0 から開始してください。可変表記には未対応のため、予想される最大行まで出力欄を用意してください。（例 .0 から .9 までの 10行を用意する）これは「外部キー連携モデルの扱い」と同じルールです。

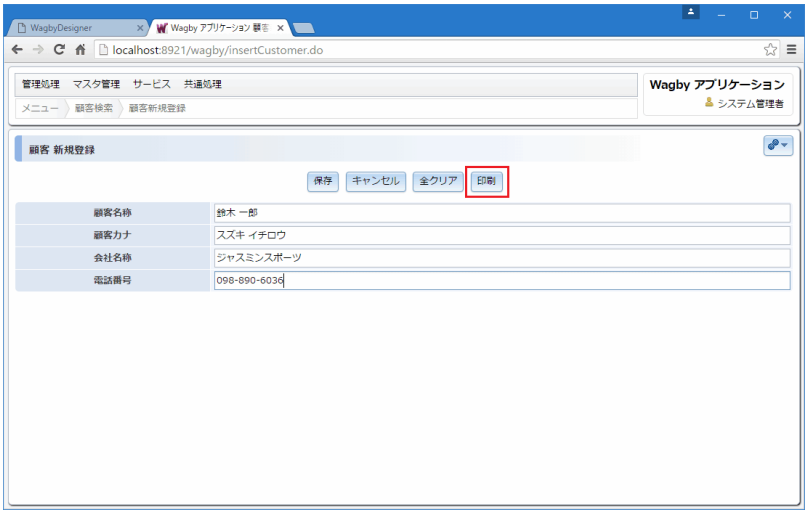


図 登録画面に帳票出力ボタンを用意する

Excel帳票 - 一覧表示

例

Excel帳票機能を設定した顧客モデルの一覧表示画面を開きます。図に示すように「帳票(全件)」ボタンが用意されています。

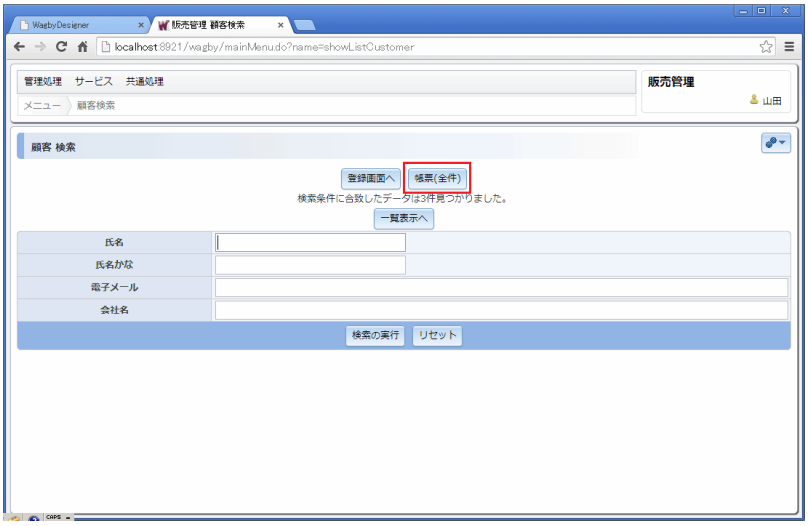


図 一覧画面に用意された帳票ボタン

このボタンを押下すると、Excel形式のファイルがダウンロードされます。

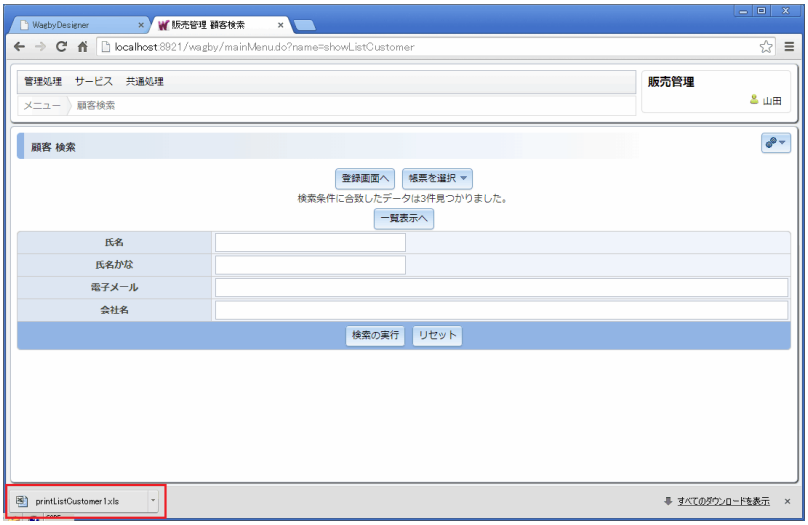
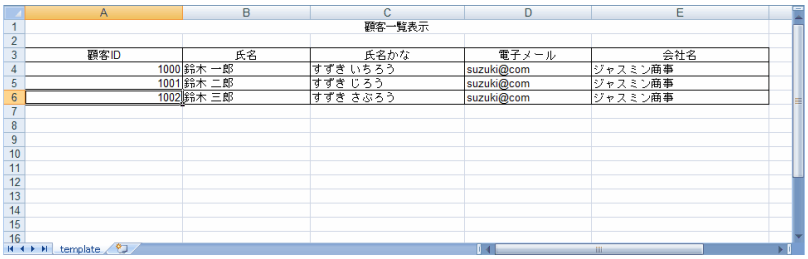


図 Excelファイルがダウンロードされる

Excelでファイルを開きます。この顧客の一覧情報がレイアウトされていることがわかります。これをプリンタで印刷することで帳票出力を行います。



顧客一覧表示

顧客ID	氏名	氏名かな	電子メール	会社名
1000	鈴木 一郎	すずき いちろう	suzuki@com	ジャスミン商事
1001	鈴木 二郎	すずき じろう	suzuki@com	ジャスミン商事
1002	鈴木 三郎	すずき さぶろう	suzuki@com	ジャスミン商事

template

図 Excelに埋め込まれた一覧情報

一覧表示の帳票出力では、現在画面に表示されているデータではなく、検索条件で適合したすべてのデータが対象になります。（例 100件のデータを検索し、画面には10件表示されている場合、帳票出力は100件が対象となります。）

出力データが存在しなかった場合のエラーメッセージ

エラーメッセージを出力する（標準）

データが一件も存在しない場合に印刷ボタンを押すと、図のようなエラーメッセージが表示されます。これが標準の動作です。

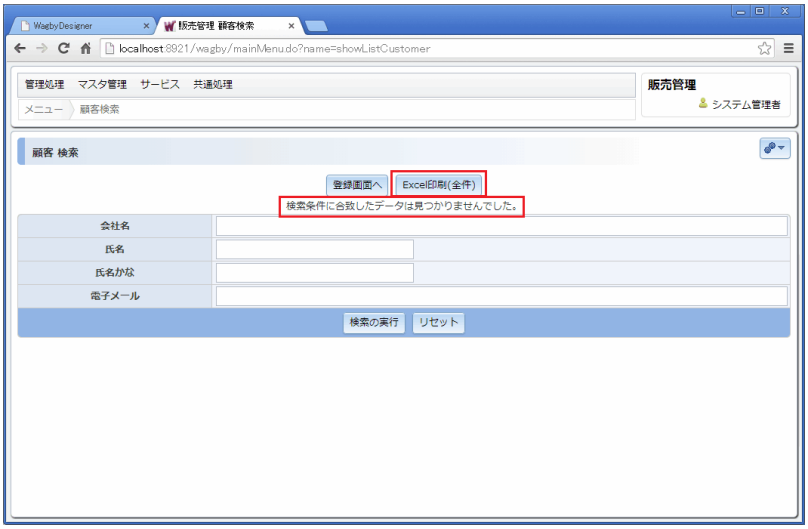


図 データが存在しない場合に印刷ボタンを押下する

図 別タブにエラーメッセージが表示される

エラーメッセージを出力しない設定

　　エラーメッセージを出力しない設定とすると、図に示すようにファイルが生成されます。しかしデータ部は空となっています。

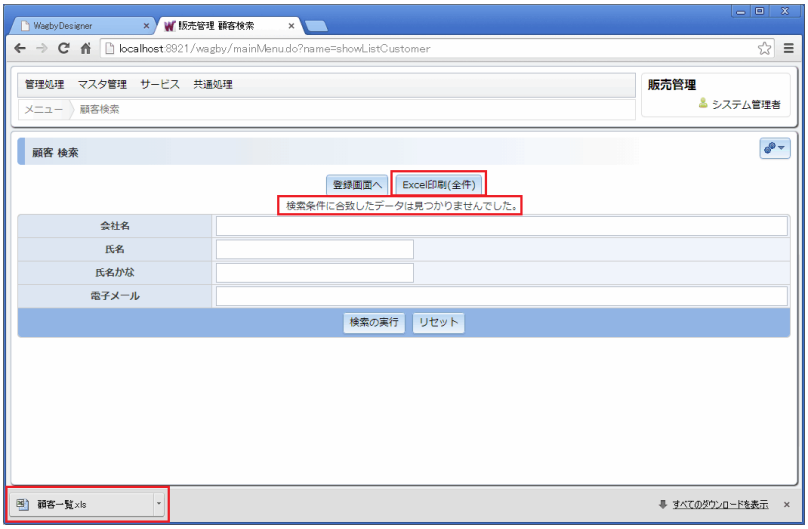


図 空のファイルが出力される(1)

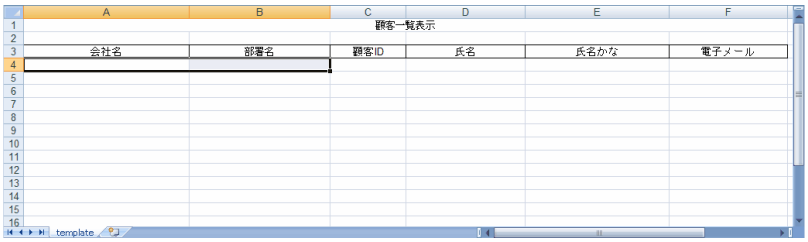


図 空のファイルが出力される(2)

定義方法

「画面>帳票出力>出力対象データが無かったときにメッセージを表示する」を有効にします。（標準では有効になっています。）

図 出力対象データが無かったときにメッセージを表示する設定

ライブラリの選択

Excel帳票出力機能は、オープンソースのExcel生成ライブラリである Apache POI を使って実現しています。

仕様・制約

テンプレートファイル

- テンプレートファイルとして用いるExcelファイルの拡張子は ″.xls″ または ″.xlsx″ となります。[注意点...]
- テンプレートファイルのファイル名は半角英数字のみとしてください。日本語文字は指定できません。
- テンプレートファイル内のシート名に ″(″,″)″（小括弧）を含めることはできません。
- テンプレートの Excel ファイルでは、以下の機能を利用することはできません。
  - ピボットテーブル
  - ドロップダウンリスト
  - パスワード保護
- テンプレートの Excel ファイルにマクロを設定した場合、正しく出力されないことがあります。この場合はマクロの内容を変えるなどして調整してください。
- うまく印刷されない場合、帳票テンプレートの最初のシートに印刷領域が設定されているかどうかを確認ください。
- プレースホルダから1つのセルに値をセットする場合、上限が32,767文字となります。

テキストエリア型の扱い

テキストエリア型の項目を Excel のセルに割り当てる場合、次の設定を行なってください。

- 当該セルの “セルを折り返して全体を表示する” を有効にする。
- セルの高さを事前に調整する。

文字列に改行コードが含まれていた場合は、改行コードもそのまま出力されます。

テンプレートファイルに .xlsx を利用した場合

拡張子 ″.xlsx″ 利用時には次の制約があります。

- テンプレートファイルにセルコメントを含めることができません。

実行環境

- Wagby アプリケーションを稼働させるサーバ機に Excel をインストールする必要はありません。サーバ機のメモリ上で直接 Excel ファイルを生成します。そのため出力処理中は一時的に多くのメモリを消費します。同時接続数が多い環境で、帳票出力が集中するような運用環境では、メモリを多めに搭載するようにしてください。
- Excel ファイルを開くために、各利用者の PC（端末）には Excel または OpenOffice が必要です。

トラブルシューティング

IE11利用時、ダウンロード時の保存、ファイルを開くボタンを素早く押すと″ファイルが移動または削除された可能性があります″というダイアログが表示される。

″CSV・Excel入出力 > ダウンロード > トラブルシューティング″をお読みください。

定義方法

Excel帳票の定義は、次の手順で行います。

1. Designerによるボタン設定とビルド。
2. ビルドによって得られたExcelテンプレートファイルのレイアウト設計
3. （ビルドした）Wagbyアプリケーションで行う「帳票テンプレート」の設定

ここでは、それぞれの手順の一覧を説明します。

Designerの設定

「画面>帳票出力」タブを開きます。帳票出力設定欄で、次の設定を行います。

設定項目	説明	入力例
テンプレートファイル名	後述する「帳票テンプレート」の識別子になります。重複しない名前を英数字の組み合わせで決めてください。拡張子（.xls や .xlsx）は含めません。	printListCustomer1
表示画面	一覧表示を選択します。	一覧表示
出力方式	Excelを選択します。	Excel
利用ライブラリ	出力方式に Excel を指定した場合、POIを指定します。Wagbyに同梱されているPOIのバージョン...	POI
表示する	この欄をチェックすると、帳票出力が有効になります。	-
表示条件	ボタンの表示条件式を記述することができます。「画面機能>ボタン名と表示条件のカスタマイズ」で説明しています。（ただし式中にモデル項目名は利用できません。一覧表示の場合、モデルの主キーを特定して値を読み出す方法は提供されていません。TODAY()など、モデル項目名に拠らない関数などは利用できます。）	（中略）
ボタン名	ボタン名を指定します。省略時は「"帳票" + 連番」となります。式を使ってボタン名を決定することもできます。（ただし式中にモデル項目名は利用できません。一覧表示の場合、モデルの主キーを特定して値を読み出す方法は提供されていません。TODAY()など、モデル項目名に拠らない関数などは利用できます。）	-

図 帳票出力の設定

Excelテンプレートファイルのレイアウト設計

図の設定後にビルドすると、Wagbyインストールフォルダ内のwork/reportフォルダ内に、テンプレートファイル名で指定したExcelファイルが自動生成されます。

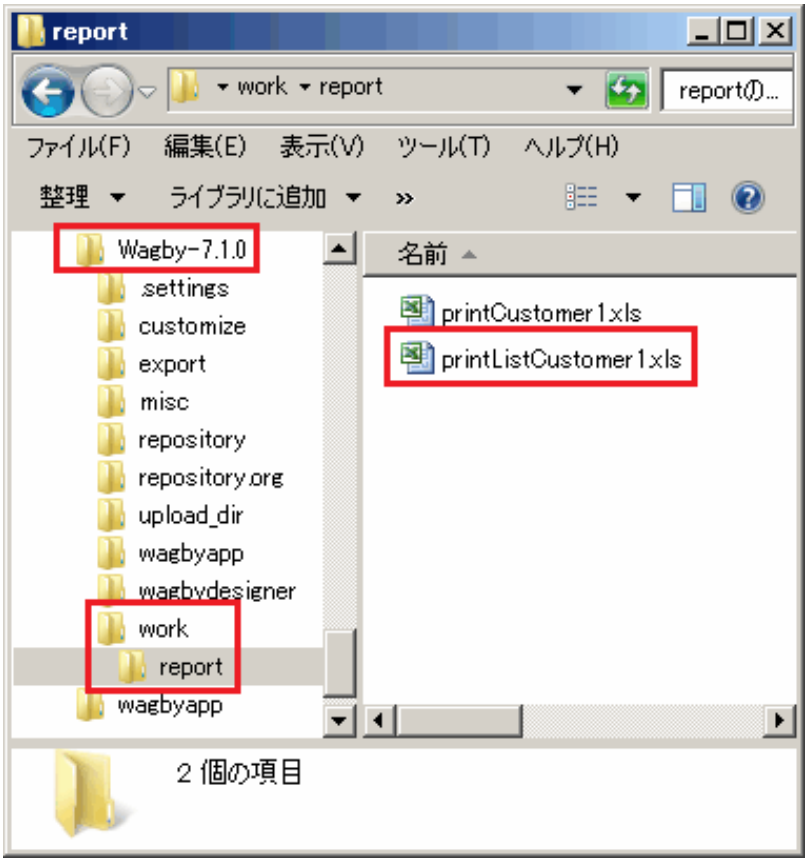


図 自動生成されたテンプレートファイル

このテンプレートファイルをExcelで開くと、図のようになっています。レイアウト設計なしで、項目名とプレースホルダが列挙されています。

モデル定義で、一覧表示として指定した項目がプレースホルダとして使われます。

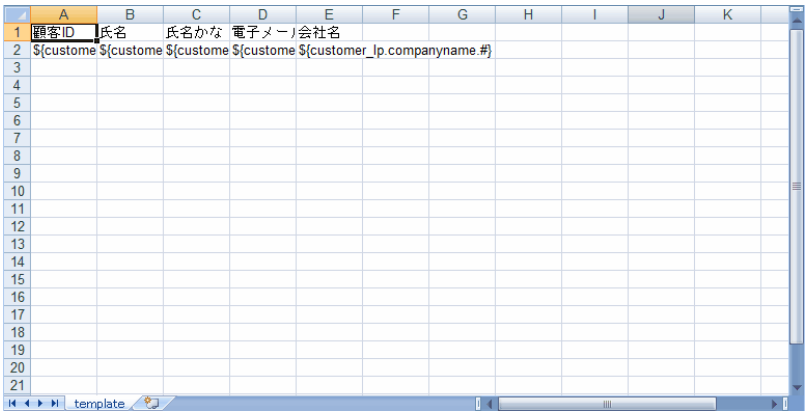


図 項目名とプレースホルダが埋め込まれたExcelファイル

ここで "${" ではじまり、"}" で終わる部分をプレースホルダと呼びます。プレースホルダの構造は次のとおりです。最後の "#" は、行が増えることを示します。

```
${モデル名_lp.項目名.#}
```

プレースホルダには、実際の値が設定されます。プレースホルダの記述ルールの詳細は"プレースホルダ"をお読みください。

このプレースホルダ表記を残しつつ、帳票レイアウト設計を行います。図にレイアウトの例を示します。

図 帳票レイアウトを行った例

一覧帳票用のテンプレートでは詳細帳票とは異なり、レコード数が動的に変更されます。そのためプレースホルダは1行にのみ記述します。（帳票処理時に、出力レコード数だけ自動的に追加されます。）

帳票テンプレートの設定

ビルドしたアプリケーションにシステム管理者でログオンします。モデルと、図の帳票ファイルを紐付けるために「帳票テンプレート」の設定を行います。

図 帳票テンプレートメニュー

新しい帳票テンプレートを作成します。

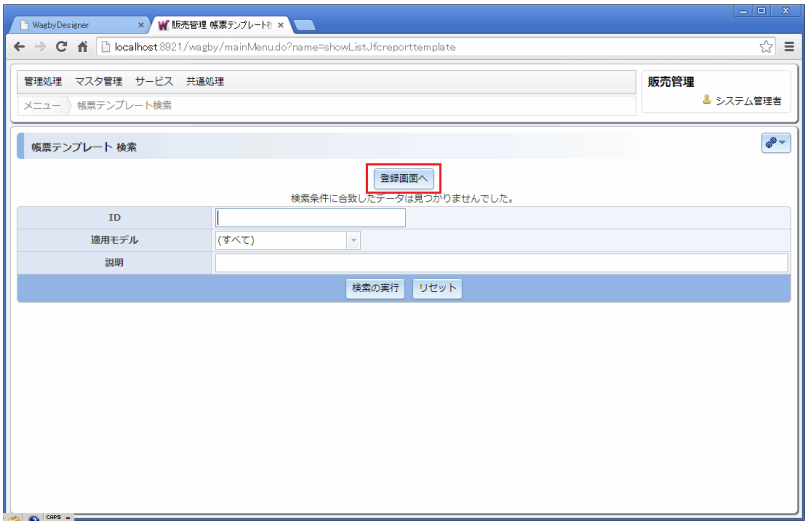


図 帳票テンプレートの新規登録を行う

ID項目は、Designerに設定した帳票テンプレートIDを指定します。拡張子（.xls や .xlsx）は含めません。

ここでは適用モデルを"顧客"とします。説明欄は自由に設定してください。

テンプレートファイルとして、図を保存したExcelファイルを指定します。

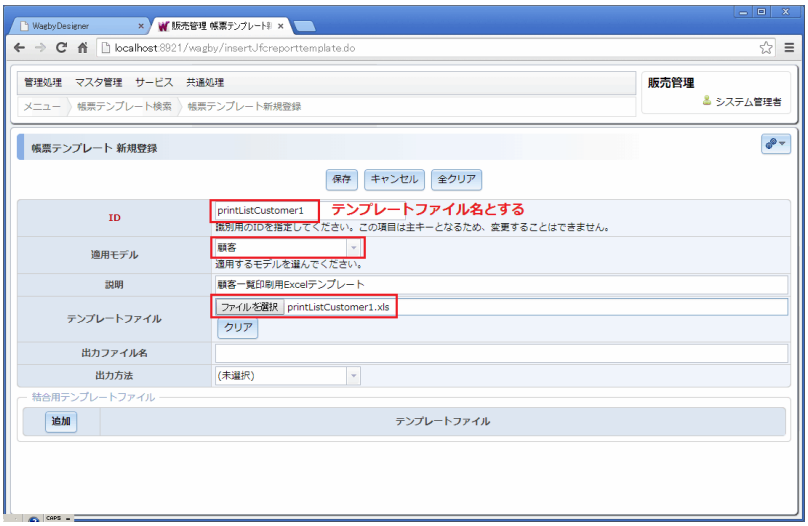


図 帳票テンプレートの設定

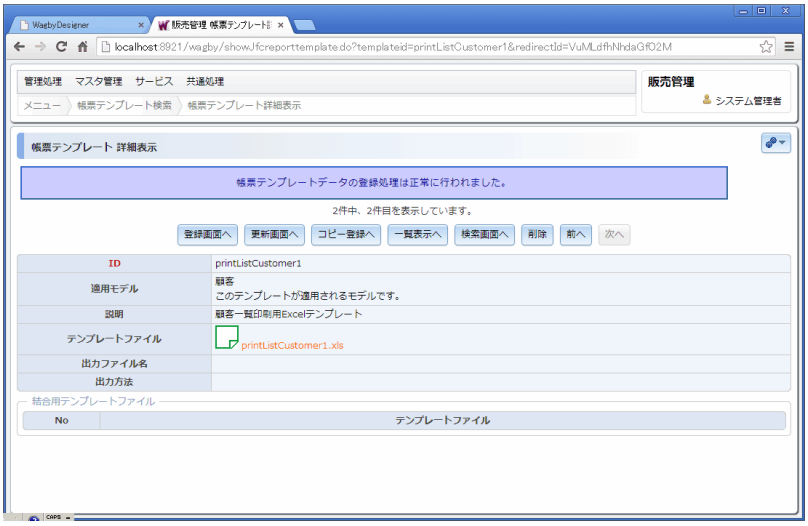


図 帳票テンプレートを登録した

ここで登録したテンプレートファイルは、upload_dir フォルダ以下に保存されます。詳細は"文字列 ファイル > 仕様 > 内部の構造"をお読みください。

ボタン名を変更する

例

一覧帳票用ボタンを二種類用意することができます。一つは、現在表示されているデータのみを出力するボタン。もう一つの"(全件)"という文字が付与されたボタンは、すべてのデータを出力するボタンです。

(378)

(378) ※"現在表示されているデータのみ"とは、検索条件で抽出されたデータのうち現在のページに表示しているデータのみを帳票へ出力するというものです。例えば検索条件で100件抽出されても現在のページに表示されているのが10件であれば10件のみが帳票出力されます。

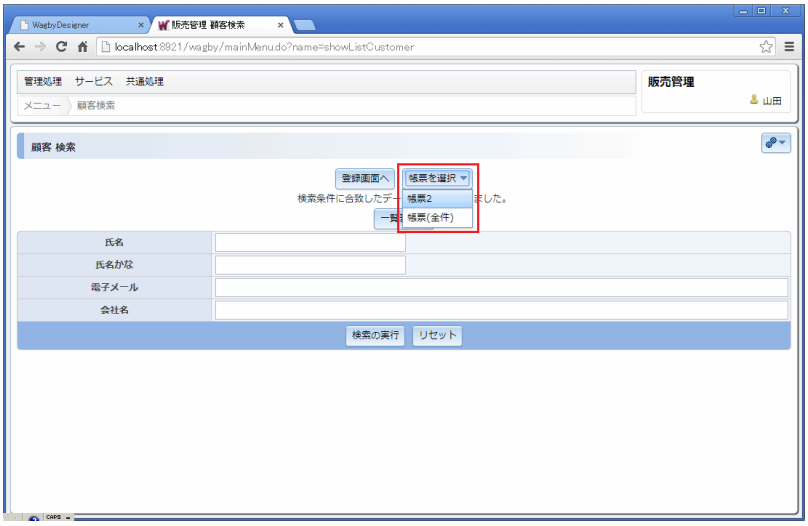


図 二つの一覧帳票ボタン

定義方法

Designer側で指定する表示画面のタイプを「一覧表示」とします。

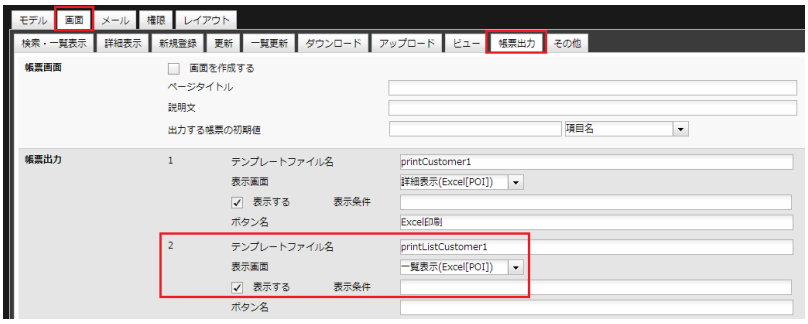


図 Designerで「一覧表示」を指定する

検索処理後にボタンを表示させる

検索処理を実行するまで、帳票ボタンを表示させない設定を行うことができます。必ず検索処理を実行させるようにすることで、大量データの全件出力を行わせないようにします。

検索画面遷移直後の検索処理を止める

「画面 > 検索・一覧表示画面 > 検索画面 > 検索画面遷移直後の検索処理を止める」を有効にします。

図 検索画面遷移直後の検索処理を止める

一覧表示のボタン表示条件

「画面 > 帳票出力」で、帳票タイプが一覧表示のボタン表示条件として次の式を設定します。

```
RESULTSIZE() > 0
```

RESULTSIZE 関数は、検索結果数を返します。

出力ファイルを変更する

帳票テンプレート設定で、出力ファイル名を指定することができます。固定値を設定します。

シート名

出力される Excel のシート名になります。

- プレースホルダは利用できません。
- 拡張子は含めないようにしてください。
- Excelのシート名に指定できない文字（コロン、円記号、疑問符、角カッコ、スラッシュ、アスタリスク）を含めることはできません。

出力ファイル名

出力される Excel のファイル名になります。

- プレースホルダは利用できません。
- ファイル名には拡張子を含めてください。(.xlsまたは.xlsx)

(379)

出力ファイル名（一括帳票用）

一覧帳票では無効です。（設定は無視されます。）

定義方法

プレースホルダを指定することができます。ファイル名は、拡張子も含めて指定してください。例を示します。

出力ファイル名	顧客情報一覧.xls

図 出力ファイル名の指定

```
顧客情報一覧.xls

顧客情報${customer_lp.customerid.0}.xls
```

プレースホルダでは、モデル名の後ろに接尾後 "_lp" を付与します。最後の ".0" は先頭行のデータを意味します。

出力ファイル名が空欄の場合、テンプレートファイル名が使われます。

プレースホルダに指定した項目は「一覧表示項目」として指定されている必要があります。一覧表示項目として指定し、表示時には隠し項目とすることもできます。

テンプレートファイルが .xls 形式ファイルだが、出力ファイル名を .xlsx とする、といったことはできません。出力ファイル名を指定するときは、テンプレートファイルの拡張子を使ってください。

(379) この設定は、HTTPレスポンスヘッダの"Content-Disposition"フィールドの"filename"属性に反映されます。

出力方法を変更する

帳票テンプレートで、出力方法を指定することもできます。空白時は〝ダウンロード〟となります。

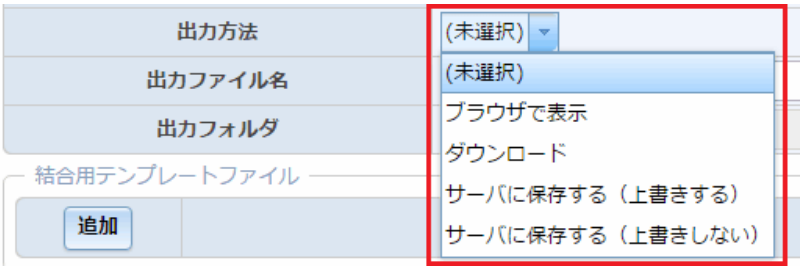


図 出力方法の指定

詳細は Excel帳票 > 詳細表示 > 出力方法 をお読みください。

一覧表示画面には表示させないが、一覧帳票には出力させる

一覧表示画面には表示させないが、一覧帳票には出力させるような定義を行うこともできます。 図では、会社名項目は一覧表示として指定していません。

モデル項目詳細定義の「出力制御>その他>帳票>一覧帳票に出力する」をチェックします。

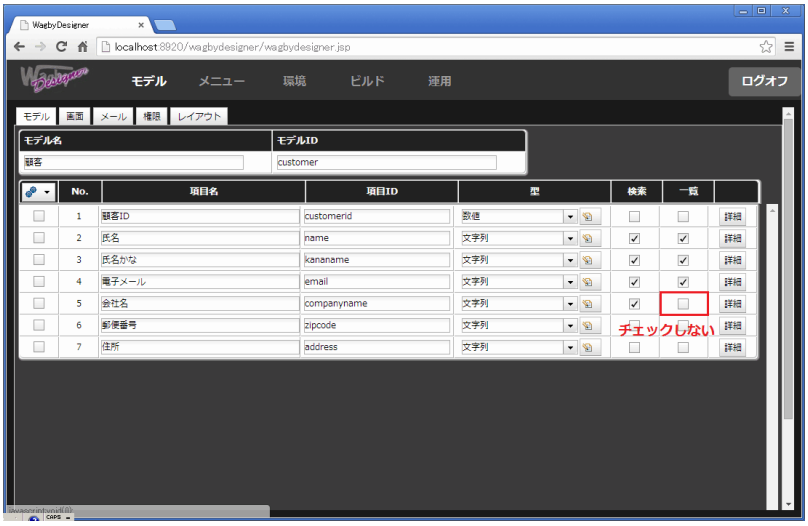


図 モデル定義で、会社名項目の一覧表示を無効とした

図 一覧帳票に出力する設定

一覧表示画面には表示させるが、一覧帳票には出力させない

一覧帳票用のテンプレートから、出力しない項目のプレースホルダを記述しないようにします。 自動生成された一覧帳票テンプレートを用いている場合、当該項目のプレースホルダを除去してください。

一覧表示テンプレートのルール

一覧帳票用のテンプレートでは詳細帳票とは異なり、レコード数が動的に変更されます。そのためプレースホルダは1行にのみ記述します。生成されたWebアプリケーション側で必要なレコード数を増やす処理が行われます。このように、システム側で行の追加を行うことから、テンプレートファイルのレイアウトデザインにはいくつかの制限があります。

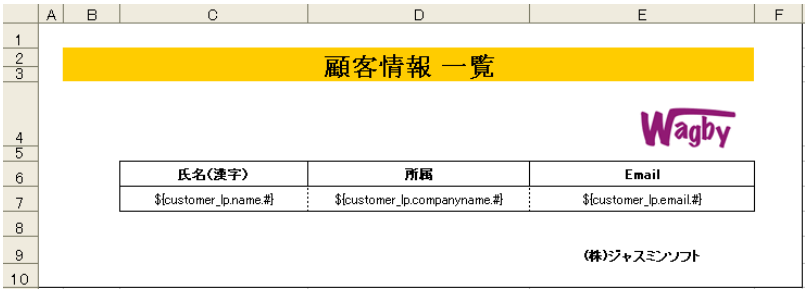


図 顧客情報の一覧帳票用テンプレートの例

(1) プレースホルダの添字(数字部分)は#で記述する

挿入されるレコードは0から連番で増えていきますが、詳細帳票とは異なり、ここでは連番部分を "#" という「ワイルドカード」で表現し、これを1行のみ記述します（例：${customer_lp.name.#}）。

(2) プレースホルダを複数行にわたって記述しない

1行にプレースホルダを記述してください。複数行にわたって記述することはできません。プレースホルダが記述された行より下部分は「フッター」と判断され、レコード出力の最後に記述されます。

(3) フッター部分の制約

プレースホルダの指定されている行より下の部分は、一覧帳票のテンプレートでは「フッター」と判断され、レコードを追加した最後にコピーされます。ただし以下のような制限があります。

- フッター部分でセルの結合はできません。セルの結合情報が残ってしまうため、追加されるレコードのセルに影響します。
- 式を記述できません。式の情報は消去されます。あるいは式のエラーが発生します。
- 画像を貼ることができません。画像が元のセル位置に残ってしまうため、結果的に画像の位置がフッターとは異なる場所に表示されてしまいます。

これらの制約は、プレースホルダ行より上部（ヘッダ部分）には適用されません。

(4) 繰り返し項目の扱い

繰り返し項目は「,（コンマ）」区切りで出力されます。

(5) 繰り返しコンテナ項目の扱い

繰り返しコンテナ内の項目は出力できません。（JOIN関数などを用いて繰り返しコンテナを「1つの」文字列に整形した出力用項目を内部で用意し、それを帳票で用いることはできます。

(6) モデル参照の扱い

モデル参照は「値」部分が出力されます。「コード」部分（モデルの主キー値）を指定することはできません。

(7) 全文検索結果の扱い

全文検索の結果を指定することはできません。

同一値によるグルーピング

単一項目のグルーピング

列の値が同じであった場合にその列のセルを「（縦方向に）グルーピング」できます。このような処理を行わせたい場合は、プレースホルダに「$J」を付与します。

```
$J${customer_lp.companyname.#}
```

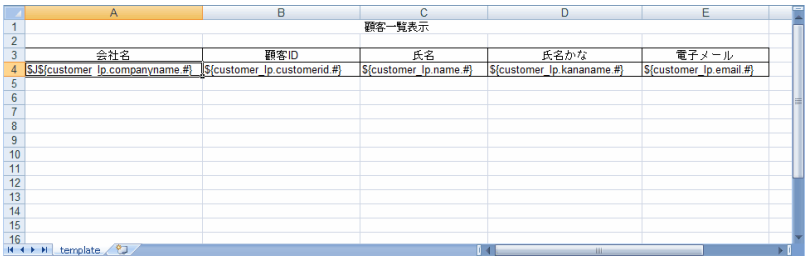


図 グルーピング指定

出力イメージは図のようになります。

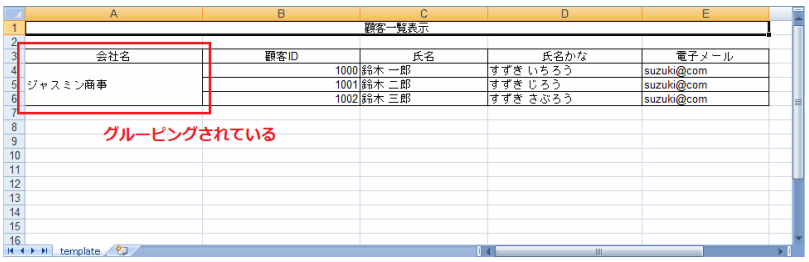


図 グルーピングによる出力イメージ

複数項目のグルーピング

複数項目をグルーピングすることもできます。 $G〇（数字）を指定することで、同じ数字を指定した項目すべてがグルーピングされます。

```
$G1${customer_lp.companyname.#}

$G1${customer_lp.deptname.#}
```

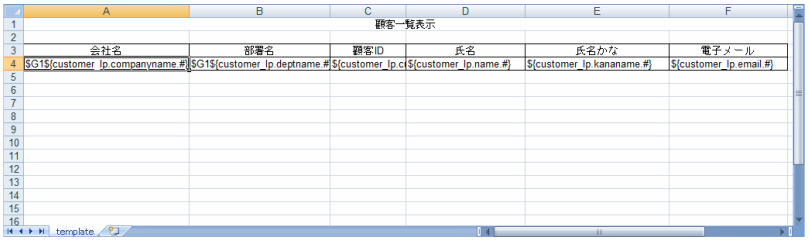
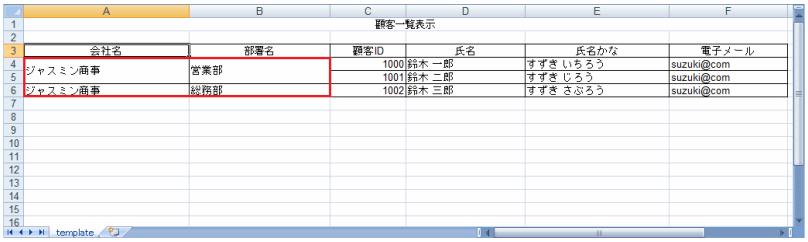


図 グルーピング指定

出力イメージは図のようになります。$G1を含むセルに重複する値が存在した場合に自動的にセルの結合を行います。



	A	B	C	D	E	F
1	顧客一覧表示					
2						
3	会社名	部署名	顧客ID	氏名	氏名かな	電子メール
4	ジャスミン商事	営業部	1000	鈴木 一郎	すずき いちろう	suzuki@com
5			1001	鈴木 二郎	すずき じろう	suzuki@com
6	ジャスミン商事	総務部	1002	鈴木 三郎	すずき さぶろう	suzuki@com
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						

template

図 グルーピングによる出力イメージ

仕様

- グルーピング処理は行(横)方向への結合はできません。
- 詳細帳票には対応していません。

プレースホルダ

プレースホルダの書式

テンプレートファイル内に「プレースホルダ」を用意することができます。その項目に、実際のデータが格納されます。

プレースホルダの書式は次のとおりです。 (380)

```
${モデル名_p.英語項目名}
```

プレースホルダの具体例

ここでは「顧客（customer）」というモデルの「氏名（customername）」という項目を例にとります。

プレースホルダ表記は、まずモデル名の末尾に「_p」を付加します。これに「.」を付与し、英語の項目名をその後につけます。最後に、この文字列の先頭に”${”を、末尾に”}”を付加します。

この例では「${customer_p.customername}」となります。 テンプレートファイル中にこの表記を埋め込んだ部分が、データベースに含まれる顧客氏名データに置き換わります。

プレースホルダのルール

プレースホルダに関するいくつかのルールについて説明します。

(1) プレースホルダ名は半角英数字で入力する。

全角文字でプレースホルダを指定しても正しく置換されません。

(2) 該当するデータが存在しない場合は「空白」文字に置換される。

プレースホルダに該当する項目の値がデータに格納されていない場合、プレースホルダは空白文字に置換されます。

(3) 権限設定を行った項目は、帳票でも（同設定が）有効となる。

項目毎に権限設定を行った場合、帳票でも有効となります。例えばある項目に隠し権限を設定した場合、帳票でも出力されません。（プレースホルダは空白文字に置換されます。）

(380) 詳細表示用テンプレートファイルでは、モデル名に「_p」を付記して区別します。一覧表示用テンプレートファイルでは、モデル名に「_lp」を付記します。

(4) 一つの項目についてのプレースホルダを複数、異なる場所に記述することもできる。

例えば ${customer_p.customername} というプレースホルダを複数個記述した場合、すべてのプレースホルダが置換されます。

(5) プレースホルダは（Excelファイルの）1 シート目のみに適用される。

プレースホルダが適用されるのは 1 シート目のみです。2 シート目以降はプレースホルダが適用されません。 このルールを使って、複雑なレイアウトを作成することもできます。

(6) プレースホルダとそれ以外の文字は混在できる。

プレースホルダとそれ以外の通常の文字を混在させることができます。プレースホルダのみが置換され、他の文字列はそのまま維持されます。例: 「顧客名は${customer_p.customername}です。」

表記一覧（詳細表示）

詳細表示用モデル（末尾が "_p" で終わるもの）について、各データ型やデータの種類におけるプレースホルダの表記方法を示します。

文字列型／数値型／日付型

モデル中の項目の値が転記されます。

表記	${モデル名_p.項目名}
例	${customer_p.name}

数値型（出力フォーマット）

数値型項目は標準で、出力フォーマットは適用されません。 指定した出力フォーマットを適用した値を出力したい場合、接頭語 fmt: を付与します。

表記	${fmt:モデル名_p.項目名}
例	${fmt:sale_p.uriage}

日付型（シリアル値）

日付型項目は標準で、出力フォーマットが適用されます。 この出力フォーマットを無効にし、モデル中の項目の値をExcel のシリアル値として出力するオプションを用意しています。接頭語 e: を付与します。

表記	${e:モデル名_p.項目名}
例	${e:customer_p.date}

シリアル値を使うと、Excel帳票のテンプレートファイルで、(Excelが提供する)任意の日付書式を設定できます。

モデル参照

参照先モデルの「内容部」が転記されます。

表記	${モデル名_p.項目名}
例	${customer_p.ctype}

モデル参照（ID部）

項目名の後ろに "#id" を付与すると、参照先モデルの「ID部」が転記されます。（一覧表示には未対応です。）

表記	${モデル名_p.項目名#id}
例	${customer_p.ctype#id}

繰り返し項目

添字に対応した繰り返し項目の値が転記されます。添字は「0」から開始されます。

表記	${モデル名_p.項目名.添字}
例	${customer_p.email.0}

繰り返しコンテナ（固定長）

添字に対応したコンテナの項目の値が転記されます。添字は「0」から開始されます。詳細は"繰り返し項目の固定長指定"をお読みください。

表記	${モデル名_p.コンテナ名.添字.項目名}
例	${customer_p.report.0.rid}

繰り返しコンテナ（可変長）

出力する数に応じて帳票イメージが伸縮します。詳細は"繰り返し項目の可変長指定"をお読みください。[381]

表記	${モデル名_p.コンテナ名.#.項目名}
例	${customer_p.report.#.rid}

参照先モデルのデータ値

参照先モデルのデータ値が転記されます。 直接項目を参照していない項目についても、次の方法で参照先モデルの任意の項目のデータを転記することができます。

表記	${自モデル名_p.項目名.参照先モデル名_p.参照する項目名}
例	${management_p.name.customer_p.companyname}

参照先モデルが外部キー連携している場合の子モデルのデータ値

参照先モデルが外部キー連携している場合の子モデルのデータ値が転記されます。

表記	${自モデル名_p.項目名.参照先モデル名_p->参照先モデルの子モデル名.添字.参照先モデルの子モデル名_p.参照する項目名}
例	${management_p.name.customer_p->support.0.support_p.supportdate}

外部キー連携している場合の子モデルの繰り返しコンテナのデータ値

外部キー連携している場合の子モデル内の繰り返しコンテナのデータ値が転記されます。

表記	${自モデル名_p->参照先モデルの子モデル名.添字.参照先モデルの子モデル名_p.コンテナ名.添字.項目名}
例	${customer_p->support.0.support_p.report.0.supportdate}

参照先モデルが外部キー連携している場合の子モデルの繰り返しコンテナのデータ値

参照先モデルが外部キー連携している場合の子モデル内の繰り返しコンテナのデータ値が転記されます。

[381] 一行にまとめて記述します。（複数行にまたがって記述することはできません。）

表記	${自モデル名_p.項目名.参照先モデル名_p->参照先モデルの子モデル名.添字.参照先モデルの子モデル名_p.コンテナ名.添字.項目名}
例	${management_p.name.customer_p->support.0.support_p.report.0.supportdate}

画像（ファイル型）

ファイル型項目は次のように扱うことができます。

ファイル名

表記	${モデル名_p.項目名}
例	${customer_p.photo} ${customer_p.report.0.photo} 繰り返しコンテナ（固定長） ${customer_p.report.#.photo} 繰り返しコンテナ（可変長）

ファイルのパス

upload_dirフォルダからの相対パスとなります。

表記	${モデル名_p.項目名_jshfilename}
例	${customer_p.photo_jshfilename} ${customer_p.report.0.photo_jshfilename} 繰り返しコンテナ（固定長） ${customer_p.report.#.photo_jshfilename} 繰り返しコンテナ（可変長）

画像

セルに画像を埋め込みます。対応しているフォーマットはjpeg,pngです。

表記	${モデル名_p.項目名_jshfilename:image}
例	${customer_p.photo_jshfilename:image} ${customer_p.report.0.photo_jshfilename:image} 繰り返しコンテナ（固定長） ${customer_p.report.#.photo_jshfilename:image} 繰り返しコンテナ（可変長）

画像の埋め込み範囲

画像を埋め込むセルの範囲を指定することができます。省略時は col と row がそれぞれ 5 となります。

表記	${モデル名_p.項目名_jshfilename:image:col=(縦方向):row=(横方向):sx=(X軸スケール):sy=(Y軸スケール)}
例	${customer_p.photo_jshfilename:image:col=1:row=1:sx=1.0:

sx, sy は省略できます。省略時は登録画像のサイズが維持されます。1.0 を指定すると、指定したセルのサイズに収まるように調整されます。0.5 を指定すると、指定したセルの半分に収まるように調整されます。なお、sx, sy を指定するときは必ず両方を指定してください。（sxのみ、またはsyのみの指定は無視されます。）

プレースホルダ jshfilename:image 表記ではなく、スクリプトで表示範囲を指定することもできます。（スクリプトの設定が優先されます。）

バーコード（文字列型） (8.3.0)

使い方は「出力制御（スタイル）>バーコード出力」をお読み下さい。

表記	${モデル名_p.項目名_jshBarcode}
例	${customer_p.codebar_jshBarcode}

表記一覧（一覧表示）

一覧表示用モデル（末尾が "_lp" で終わるもの）について、各データ型やデータの種類におけるプレースホルダの表記方法を示します。

添字は「0」から開始されます。この添字は数字ではなく「#」というワイルドカードを使用します。このプレースホルダのある行を(0から)繰り返し生成します。

文字列型／数値型／日付型

モデル中の項目の値が転記されます。

表記	${モデル名_lp.項目名.添字}
例	${customer_lp.name.#}

日付型項目の "e:" 指定、ならびに数字型項目の "fmt:" 指定も利用できます。

画像（ファイル型）

文字列型と同様に、末尾に「#」を指定します。例 ${customer_lp.photo_jshfilename:image.#}

画像の埋め込み範囲

画像を埋め込むセルの範囲を指定することができます。省略時は col と row がそれぞれ 5 となります。

表記	${モデル名_lp.項目名_jshfilename:image:col=(縦方向):row=(横方向):sx=(X軸スケール):sy=(Y軸スケール).#}
例	${customer_lp.photo_jshfilename:image:col=1:row=1:sx=1.0

sx, sy は省略できます。省略時は登録画像のサイズが維持されます。1.0 を指定すると、指定したセルのサイズに収まるように調整されます。0.5 を指定すると、指定したセルの半分に収まるように調整されます。なお、sx, sy を指定するときは必ず両方を指定してください。（sxのみ、またはsyのみの指定は無視されます。）

繰り返し項目

繰り返し項目は「,（コンマ）」区切りで出力されます。

繰り返しコンテナ

繰り返しコンテナ内の項目は出力できません。（JOIN関数などを用いて繰り返しコンテナを「1つの」文字列に整形した出力用項目を内部で用意し、それを帳票で用いることはできます。

検索結果数

一覧表示の場合は検索結果数を、ダウンロードの場合はダウンロード処理を行なったあとでカウントされた行数が転記されます。

表記	${resultsize}
例	（表記に同じ）

検索条件値

一覧表示している検索条件の値が転記されます。検索条件はモデル名に "_cp" という文字を付与することで参照できます。

表記	${モデル名_cp.項目名}
例	${customer_cp.ctype} ※1 ${customer_cp.refitem} ※2 ${customer_cp.refitem#id} ※3

※1は通常の項目です。 ※2はモデル参照項目の内容部です。 ※3はモデル参照項目のID部です。

検索条件値（繰り返しコンテナ）

検索条件に繰り返しコンテナを含む場合、コンテナ名は除きます。

表記	${モデル名_cp.コンテナを除く項目名}
例	繰り返しコンテナ cont 内の項目 field が検索条件として用いられており、これを一覧帳票に含める場合。 ${customer_cp.field}

検索条件値（範囲検索項目）

一覧表示している範囲検索条件の値が転記されます。項目名の末尾に "1jshparam", "2jshparam" を付与します。

表記	開始値：${モデル名_cp.項目名1jshparam} 終了値：${モデル名_cp.項目名2jshparam}
例	${customer_cp.cdate1jshparam}${customer_cp.cdate2jshpara

検索条件値は、検索処理を行ったタイミングで値がセットされます。画面に検索条件を入力しても、検索処理を行わずに一覧表示の出力を行った場合、検索条件のプレースホルダは空となります。

表記一覧（共通）

詳細表示、一覧表示のいずれでも利用できます。

ユーザアカウント

ログオンしているユーザアカウント名が転記されます。

表記	${username}
例	（表記に同じ）

作成日時

帳票を作成（出力）した日時が転記されます。

表記	${createdate}
例	（表記に同じ）

モデルID (8.3.0)

対象モデルのIDが転記されます。

表記	${modelid}
例	（表記に同じ）

モデル名 (8.3.0)

対象モデルの名前が転記されます。

表記	${modelname}
例	（表記に同じ）

制約

・ 繰り返しコンテナの場合、出力可能な上限数は65,536行です。

複雑なレイアウトの作成

1シート目にプレースホルダを集約させる

出力される帳票の Excel ファイルには、データは1シート目にのみ出力されます。 複数テンプレートを用意したい場合は、プレースホルダを指定するシートと帳票のシートを分けます。同じファイル内で複数の帳票を指定する方法を紹介します。 同じデータを異なるテンプレートを用いて帳票出力させることにも利用できます。

1シート目の設定

1シート目にデータ出力のためプレースホルダを用意します。 ここでは、A列に項目名（日本語）、B列にプレースホルダ、C列に連結したい文字列を記述しています。 （項目名と連結文字は任意で指定。プレースホルダは必ず指定します。） また、ここでは区別しやすいように項目名のセルには色を指定しています。

	A	B	C	D	E
1					
2	日付	${createdate}	#VALUE!		
3	会社名(見積)	${order_pecompany_name}	御中		
4	会社名(代理店)	${order_psalesagencyname}	御中		
5	住所(代理店)	${order_pagencyaddress}			
6	件名	${order_ptitle}			
7	支払条件	${order_pepay_condition}			
8	納入期限	${order_pdue_limit}	#VALUE!		
9	納入場所	${order_pdue_place}			
10	製品1				
11	製品	${orderdetails_lp.detail1_product.0}			
12	契約プラン	${orderdetails_lp.detail1_contractname.0}			
13	製品単価(税抜き)	${orderdetails_lp.unit_price1.0}			
14	受託	${orderdetails_lp.detail2.0}			
15	受託単価(税抜き)	${orderdetails_lp.unit_price2.0}			
16	仕切率	${orderdetails_lp.oragencysrate.0}			
17	数量	${orderdetails_lp.num.0}			
18	単位	${orderdetails_lp.unit.0}			
19	消費税	${orderdetails_lp.tax.0}			
20	金額(税抜き)	${orderdetails_lp.amount.0}			
21	特別値引き額	${orderdetails_lp.discountprice.0}	▲		
22	製品2				
23	製品	${orderdetails_lp.detail1_product.1}			
24	契約プラン	${orderdetails_lp.detail1_contractname.1}			
25	製品単価(税抜き)	${orderdetails_lp.unit_price1.1}			
26	受託	${orderdetails_lp.detail2.1}			
27	受託単価(税抜き)	${orderdetails_lp.unit_price2.1}			
28	仕切率	${orderdetails_lp.oragencysrate.1}			
29	数量	${orderdetails_lp.num.1}			
30	単位	${orderdetails_lp.unit.1}			
31	消費税	${orderdetails_lp.tax.1}			

Sheet1 / 注文書 / 注文請書 /

図 データ出力用シート（1シート目）

2シート目以降の設定

2シート目以降に出力したい帳票のテンプレートを用意します。 この例では「注文書」、「注文請書」を用意しています。

図 データ出力用シート（2シート目）

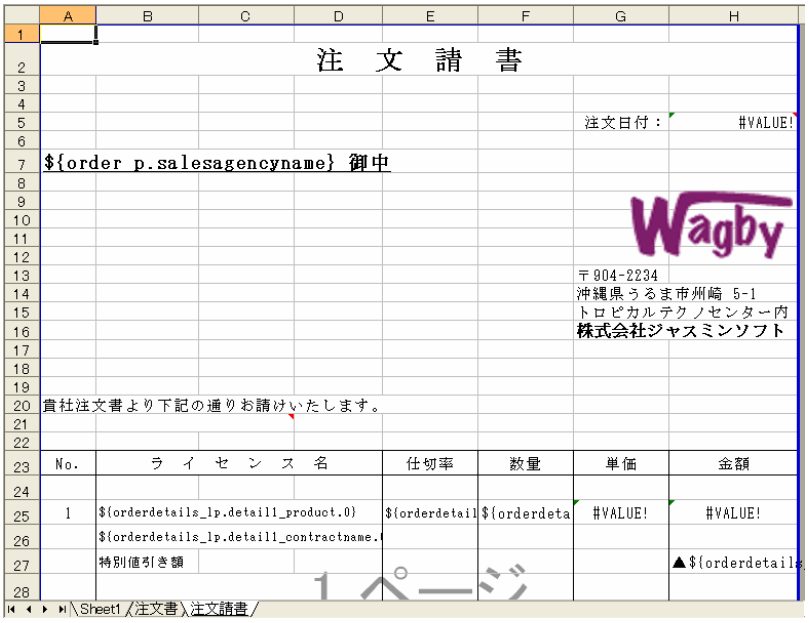


図 データ出力用シート（3シート目）

データ転記を行いたいセルに、1シート目のデータを参照するような式を記述します。　ここでは、Sheet1のB4 の値がある場合、C4の文字と連結して表示するように指定しています。

```
=IF(TRIM(Sheet1!B4)<>"",(Sheet1!B4)&(Sheet1!C4),"")
```

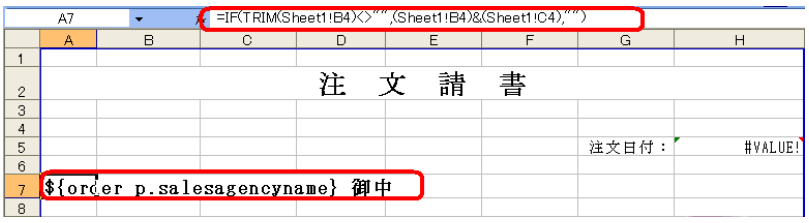


図 セル参照の式を記述する

テンプレートファイル保存時には2シート目以降のシートを開いた状態で保存してください。帳票ファイルを開くと、そのシートが表示されます。

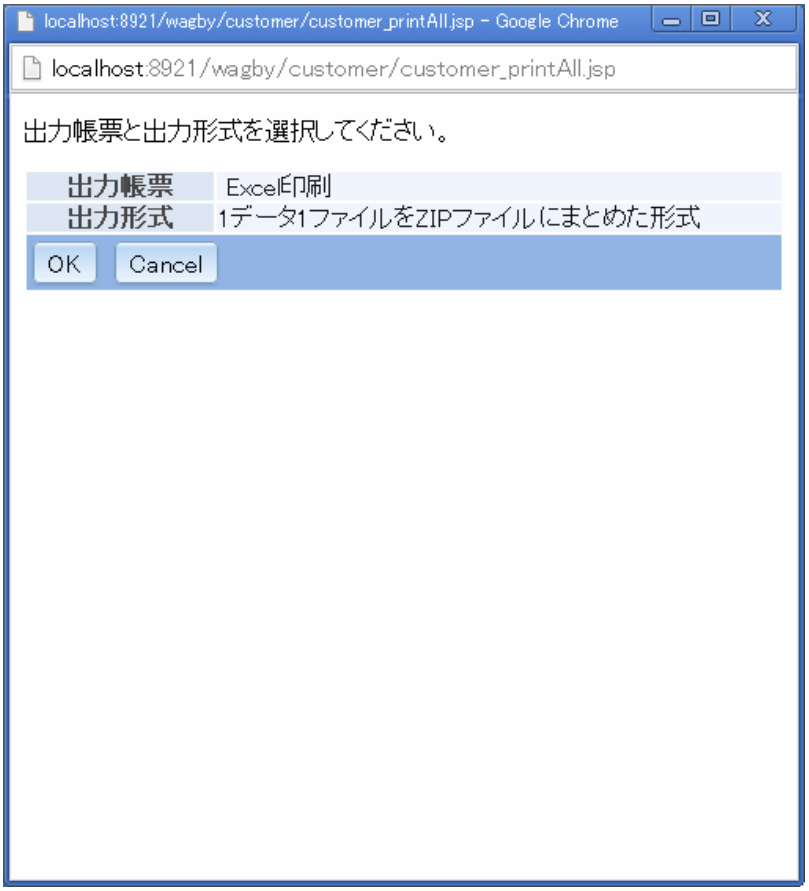
複数データの帳票を一度に出力する(一括帳票出力)

例

一覧表示に表示されているデータを選択し、一度に帳票を出力することができます。 図において、チェックボックスで対象データをチェックし「一括帳票出力」ボタンを押下することで複数の詳細帳票を一括で出力します。

図 一括帳票出力機能を設定した画面

図のようなダイアログが表示されます。「OK」ボタンを押下します。



複数のExcel帳票が1つのzipファイルに圧縮されてダウンロードされます。

図 複数の帳票をzip形式ファイルでまとめた

ダウンロードしたzipファイルを解凍すると（チェックボックスで選択した）複数のファイルが含まれていることがわかります。

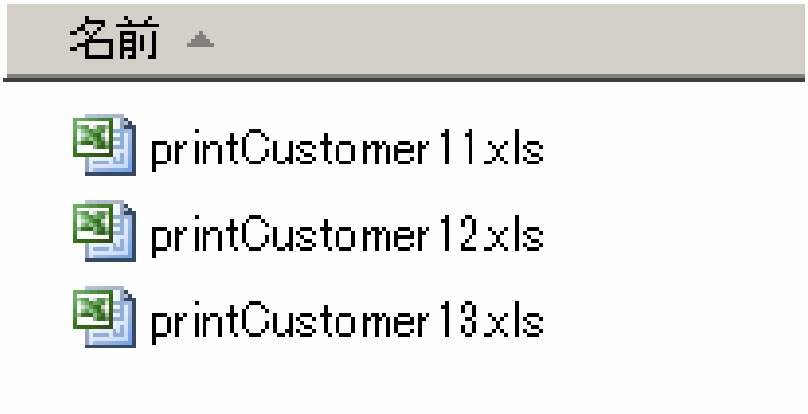


図 zip形式ファイルに含まれているExcel帳票

このように、各データの詳細帳票を一度に入手することができます。

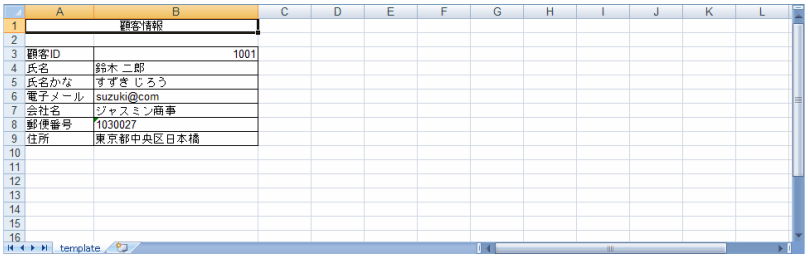


図 各帳票ファイル

定義方法

「画面>検索・一覧表示>ボタン表示>一括帳票出力ボタン」を有効にします。

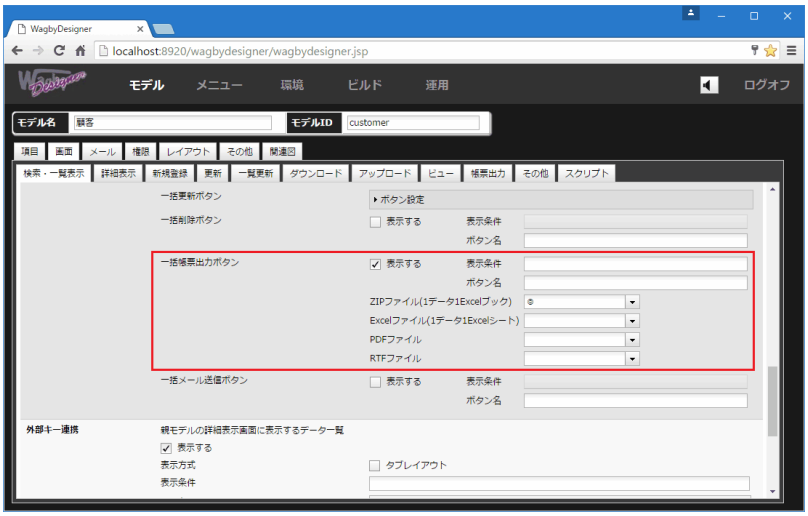


図 一括帳票出力ボタンを表示する設定

未選択は「○」と同じ扱いになります。

このモデルでは別途、詳細表示の帳票出力を設定する必要があります。（帳票出力設定が行われていない場合、この設定は無視されます。）

Excel出力の詳細

一括帳票の出力形式について、次のいずれか（または両方）を選択できます。

ZIPファイル(1データ1Excelブック)

1データ1Excelブックの複数ファイルをZIPファイルにまとめて出力します。

制約：帳票テンプレートは拡張子 xls を利用してください。（拡張子 xlsx に未対応です。）

Excelファイル(1データ1Excelシート)

1つのExcelブック内に、1データ1シートの形式で出力します。

図で、Excel形式を選択したときの出力イメージを図に示します。　Excel形式では、複数の帳票データがシート毎に分かれて出力されます。

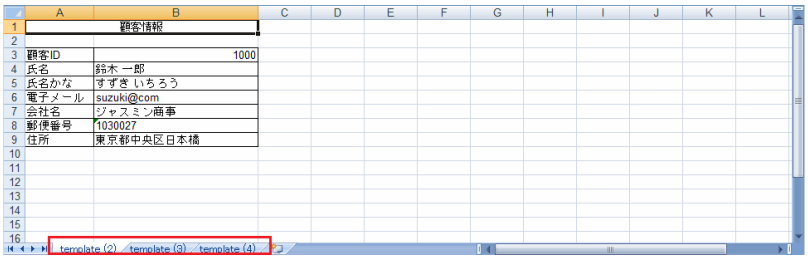


図 シート毎にデータが出力される

「1データ1Excelシート」形式の仕様

- 1 つの帳票のみを選択できます。
- 本機能は 1 番目のシートをコピーして、ブック内に展開します。コピーされるシート名は、先頭のシート名に (2), (3) という接尾辞が付与されます。
- 「ファイルのダウンロード」ダイアログによる確認時に「開く」ボタンを押下すると、操作中の Web ブラウザの親ウインドウに直接、Excel帳票が表示されます。この場合、以前の画面に戻ることができません。「保存」ボタンを選択し、いったん Excel ファイルをフォルダに保存後に開くようにしてください。
- シート内に図ならびに画像を含めないようにしてください。出力されたファイルを開くことができなくなることがあります。
- 拡張子 xls の場合、印刷範囲はシート毎に設定されます。（印刷範囲のシート毎設定機能は拡張子 xlsx には未対応です。）

「1データ1Excelシート」形式のシート名の設定

帳票テンプレートの「出力ファイル名」に記載された名前をシート名に適用します。次のルールが適用されます。

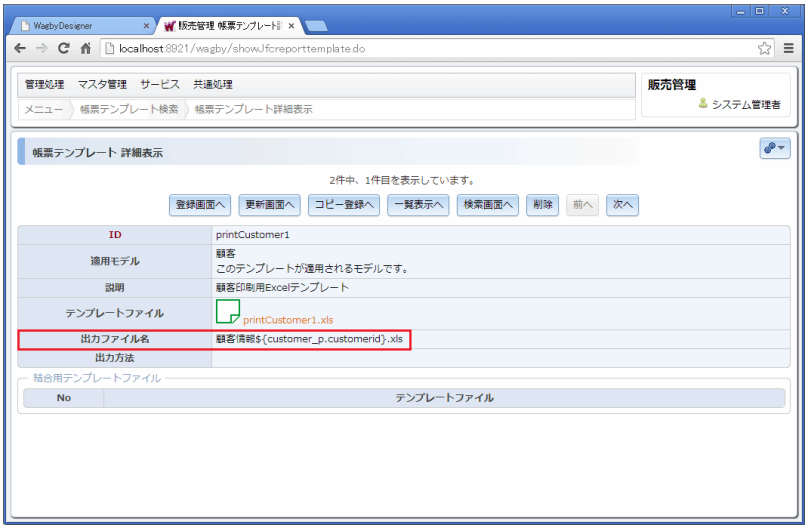


図 帳票テンプレートの設定

- プレースホルダを利用できます。
- 拡張子（.xls, .xlsx）は除去されます。シート名には含まれません。
- シート名の設定時のみ、特別なプレースホルダ ${sheet} を利用できます。1から開始される番号です。例: 顧客情報${customer_p.customerid}_${sheet}
- 出力ファイル名をシート名に適用するため、出力ファイル名にExcelのシート名に指定できない文字（コロン、円記号、疑問符、角カッコ、スラッシュ、アスタリスク）を含めることはできません。

「○」と「◎」の違い

ZIPファイル形式とExcel形式の両方を指定したとき、「◎」と設定された方がデフォルトとなります。 図に、ZIP形式に「○」を、Excel形式に「◎」を設定した例を示します。

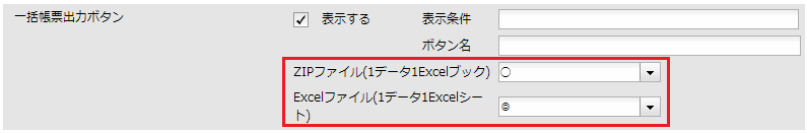


図 ZIPファイル形式とExcel形式の指定方法

図の設定により、図のようなダイアログが開きます。 二つの選択肢を選べますが、デフォルトは「◎」としたもの（ここではExcelファイル形式）になっていることがわかります。

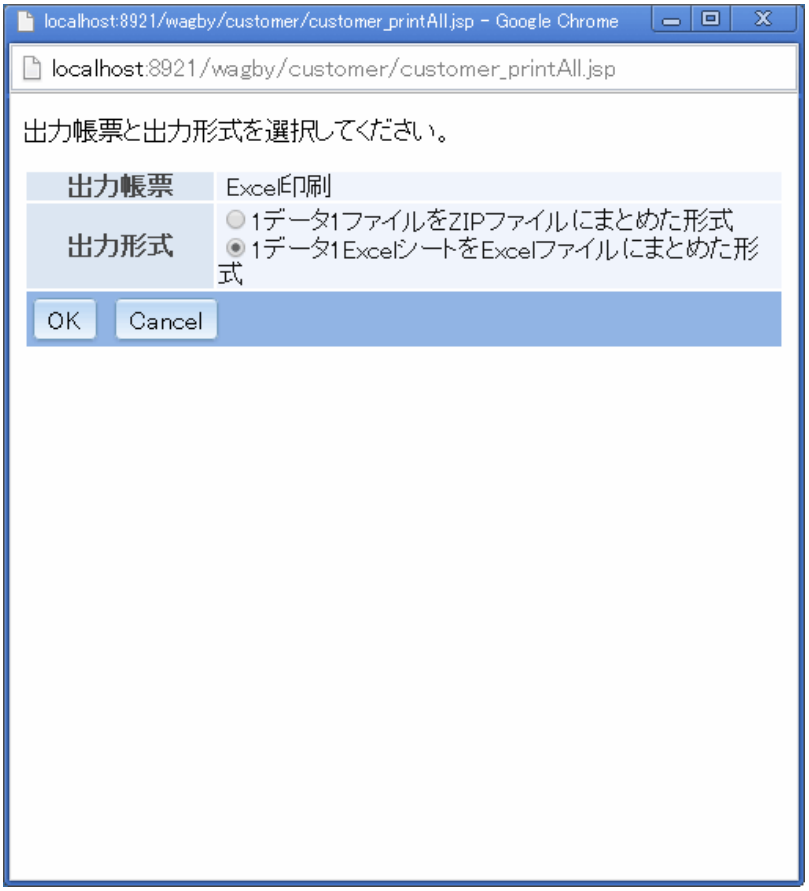


図 ダイアログの表示

PDF出力の詳細

PDFをzip形式でまとめる

図の「一括帳票出力ボタンを表示する設定」で、PDFファイルの欄を「〇」（または空白）とした場合の動作です。

図で複数データをチェックし、一括帳票出力ボタンを押下すると、複数のPDFが一つのzipファイルに圧縮されたファイルをダウンロードできます。

複数データを1PDFにまとめる

図の「一括帳票出力ボタンを表示する設定」で、PDFファイルの欄を「◎」とした場合の動作です。

このとき、複数のデータを1つのPDFファイルにまとめます。

詳細帳票のファイル名にモデル項目の値を指定する

例

詳細帳票ファイル名にモデル内の項目を含めると、一括帳票出力の際にファイル名で判別できるため便利です。

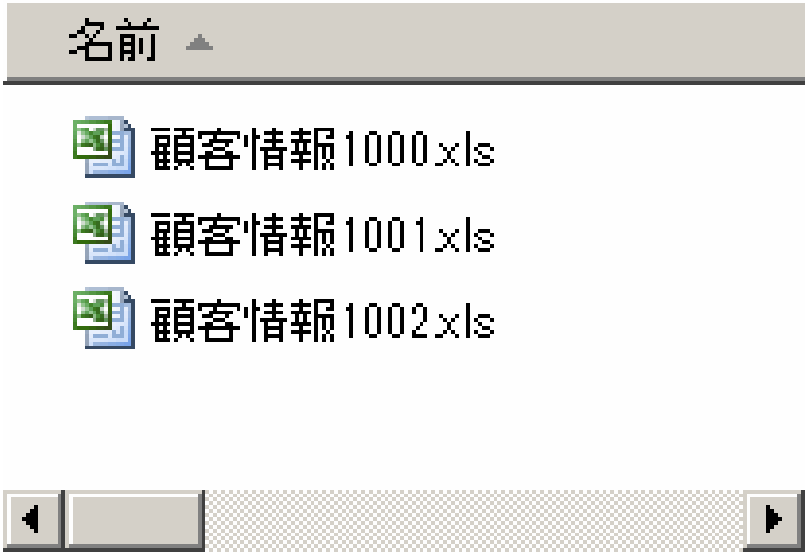


図 zip形式ファイルを展開後の一括出力ファイル（顧客IDが含まれている）

定義方法

帳票テンプレート設定で、出力ファイル名を指定します。 詳細は"Excel帳票 > 詳細表示 > 出力ファイルを変更する"をお読みください。

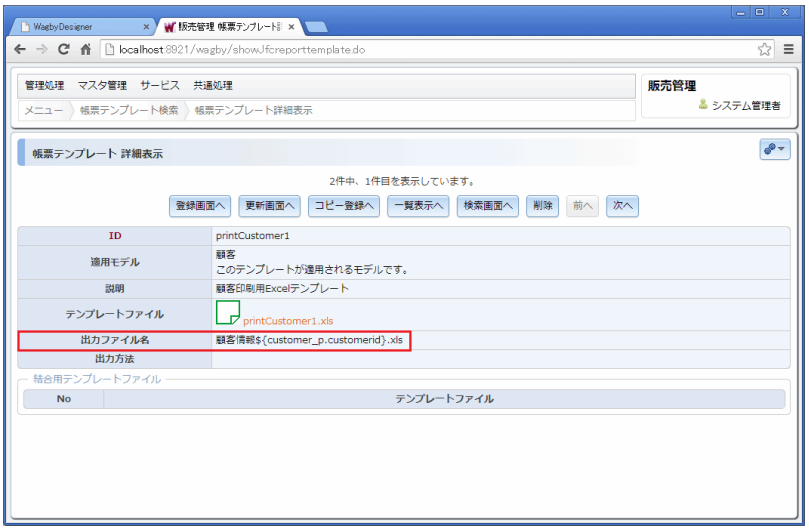


図 帳票テンプレートの設定

仕様・制約

- 一括帳票出力では、帳票テンプレートに指定した出力形式は反映されません。「ダウンロード」の動作で固定となります。
- 詳細表示の帳票出力を設定する必要があります。帳票出力設定が行われていない場合、この設定は無視されます。

一覧表示を非同期で出力する

例

出力処理

一覧表示画面の「帳票（全件）」を非同期処理と設定した例です。

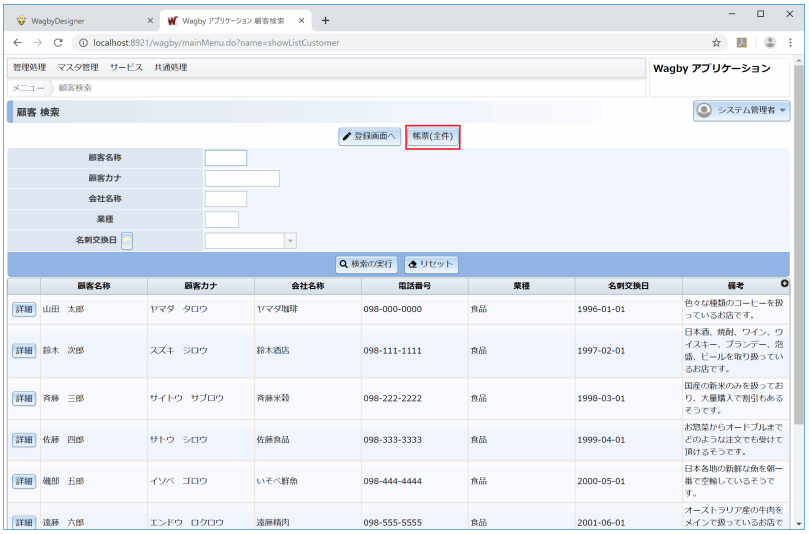


図 一覧表示画面の「帳票（全件）」ボタン

帳票ボタン押下後、″一覧帳票出力ジョブを登録しました。″　というメッセージが表示されます。このとき、実際の帳票出力処理は別の処理として行われます。

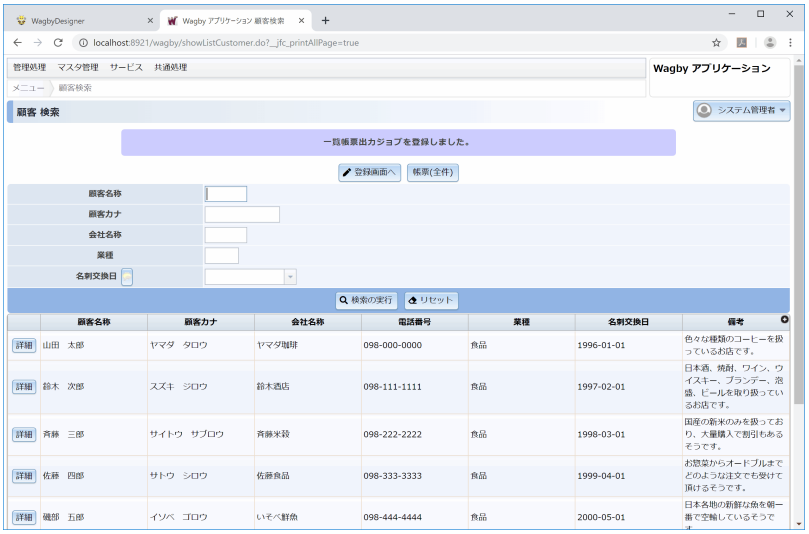


図 一覧帳票出力ジョブが登録される

出力結果を取得する

帳票画面を開きます。状態が″終了″となっていれば、その帳票出力はダウンロード可能です。

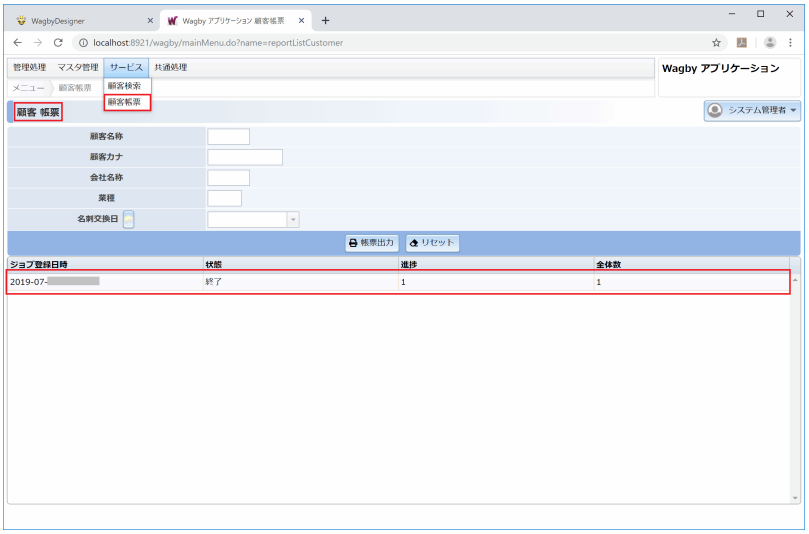


図 帳票画面

ダウンロードしたいデータをクリックすると、ダウンロードを確認するダイアログが開きます。"OK" を押下することでダウンロードすることができます。

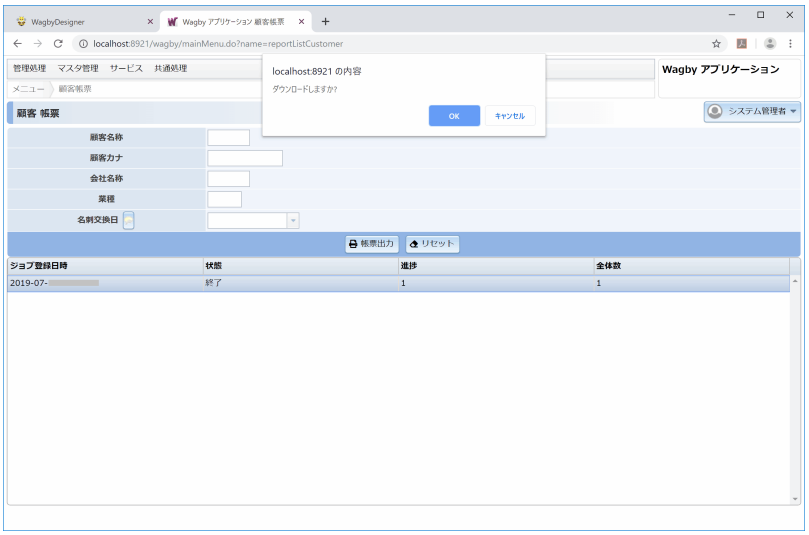


図 データをダウンロードする

定義方法

一覧帳票を非同期処理とする場合の設定方法は次の通りです。

メッセージキューの設定

"ActiveMQ" を有効にします。Wagbyにはインメモリで動作するActiveMQモジュールが同梱されているため、ActiveMQを別途インストールする必要はありません。

(382)

(382) ActiveMQをサーバとして別途用意するか、あるいはすでに用意されているActiveMQサービスを利用することもできます。ここでは説明を割愛します。

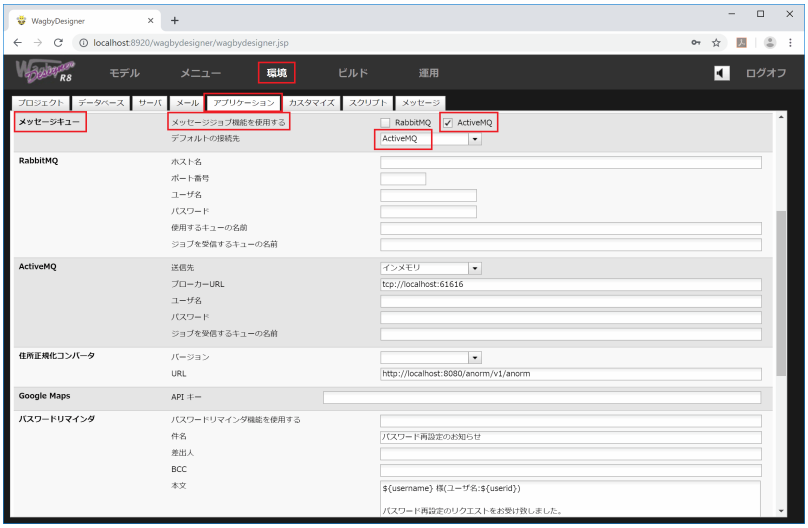


図 メッセージキューの設定

モデルごとの帳票出力の設定

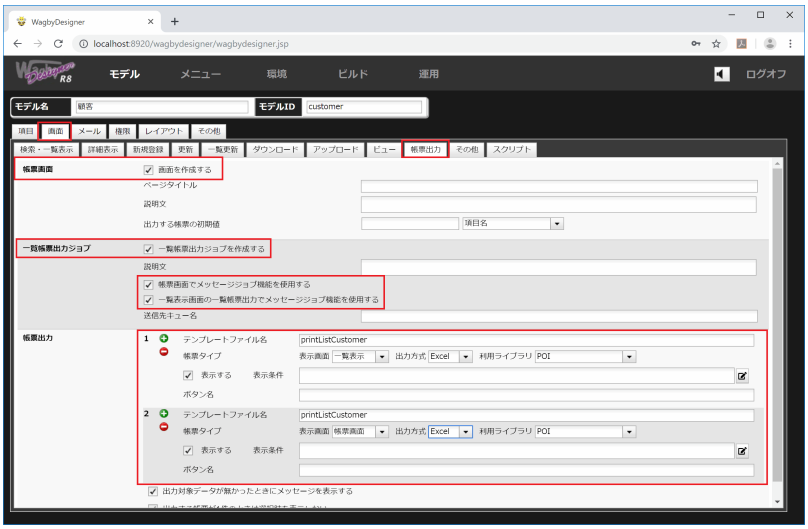


図 帳票出力の設定

帳票画面

″画面を作成する″を有効にします。非同期処理の場合は「帳票画面」から帳票ファイルを取得します。

一覧帳票出力ジョブ

設定項目	説明	入力例
一覧帳票出力ジョブを作成する	（後述する）ジョブクラスを自動生成します。	（チェック）
帳票画面でメッセージジョブ機能を使用する	帳票画面で帳票出力をおこなったとき、メッセージジョブを登録する動作に変わります。	（チェック）
一覧表示画面の一覧帳票出力でメッセージジョブ機能を使用する	一覧表示画面の帳票ボタン押下時にメッセージジョブを登録する動作に変わります。	（チェック）
送信先キュー名	通常は空白でかまいません。空白の場合は″wagbyJobQueue″になります。	wagbyPrintJobQueue

帳票出力

同一のテンプレートファイル名で、表示画面 "一覧表示" と "帳票画面" の二つを用意します。

"一覧表示" の設定により、一覧表示画面に「帳票（全件）」ボタンが用意されます。

"帳票画面" の設定により、帳票画面でデータをダウンロードできるようになります

一覧帳票出力ジョブの確認

"一覧帳票出力ジョブを作成する" を有効にすると、内部で "ReportList<モデルID>Job" というジョブが生成されます。これは管理処理メニュー内のジョブ状態で確認することができます。

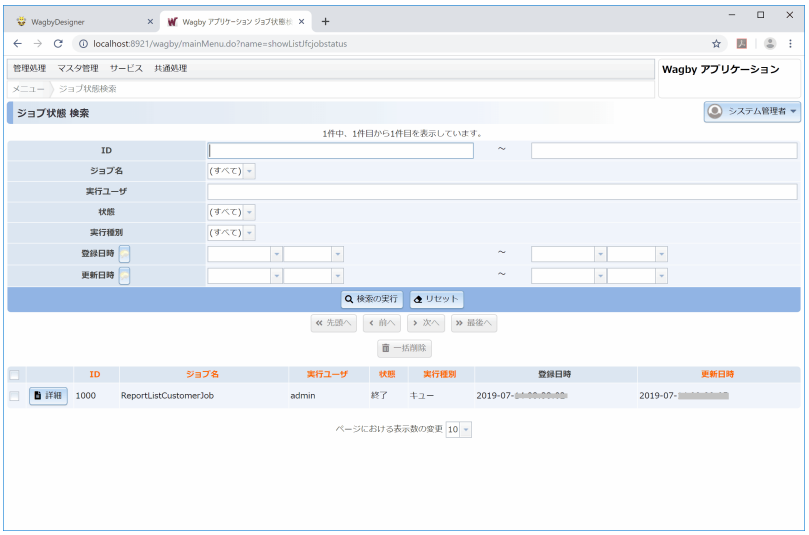


図 ジョブ状態の確認

仕様・制約

- Excel,PDF 帳票いずれにも対応しています。
- 詳細表示の印刷には未対応です。
- 複数データの帳票を一度に出力する方法にも未対応です。

一覧帳票出力用の専用画面を用意する

例

本設定を行うと、メニューに「（モデル）帳票」というボタンが用意されます。

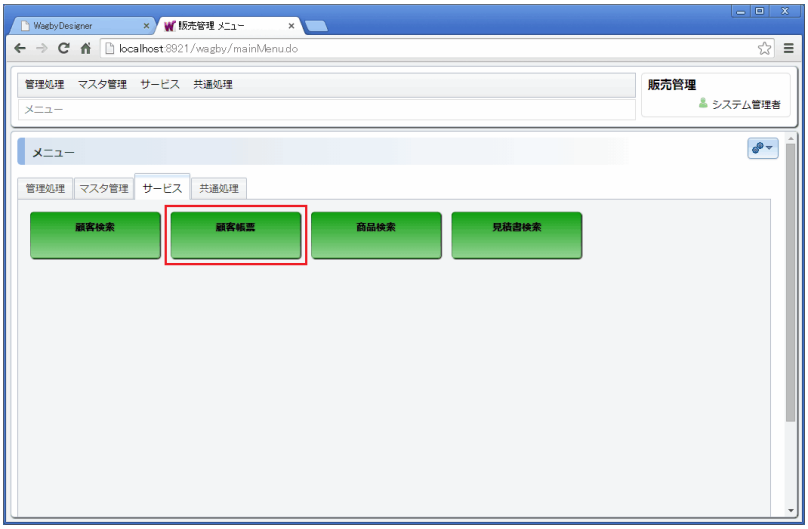


図 メニューに用意された帳票出力ボタン

帳票出力の画面で出力したい一覧帳票の種類を選びます。検索条件を指定後、「帳票出力」ボタンを押下します。 出力条件の項目は、検索条件に指定した項目と同じ項目です。（検索条件画面で表示される内容と同一です。）

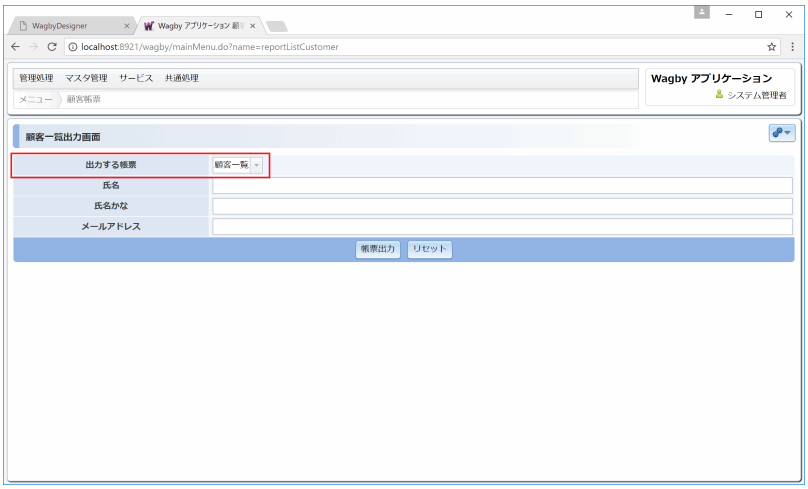


図 帳票出力画面

定義方法

(1)「画面>帳票出力>帳票画面>画面を作成する」を有効にします。

図 帳票画面を作成する

(2) テンプレートを設定します。表示画面は「帳票画面」とします。図の選択肢で判別できるよう、ボタン名も合わせて設定します。

図 テンプレート設定

テンプレートの設定を行わない場合、ビルド時に「モデル名 XXX の帳票出力画面に対応する帳票が見つかりません。」というエラーになります。

(3) メニューに「帳票画面」を用意します。

出力する帳票の初期値

図の設定画面で「出力する帳票の初期値」を指定することができます。

項目名

文字列型の項目、もしくはモデル参照項目を指定します。その項目の"値"が、「テンプレートファイル名」と一致しているものを選択されている状態とします。 モデル参照項目の場合は、参照先モデルの"主キーの値"が「テンプレートファイル名」と一致しているかどうかを探します。（本設定は参照先モデルが複合主キーには対応していません。）

値

指定した値が、「テンプレートファイル名」と一致しているものを選択されている状態とします。

選択肢が1件の場合に出力する帳票を非表示とする

帳票出力画面の種類が一つの場合、「出力する帳票」項目を非表示とすることができます。

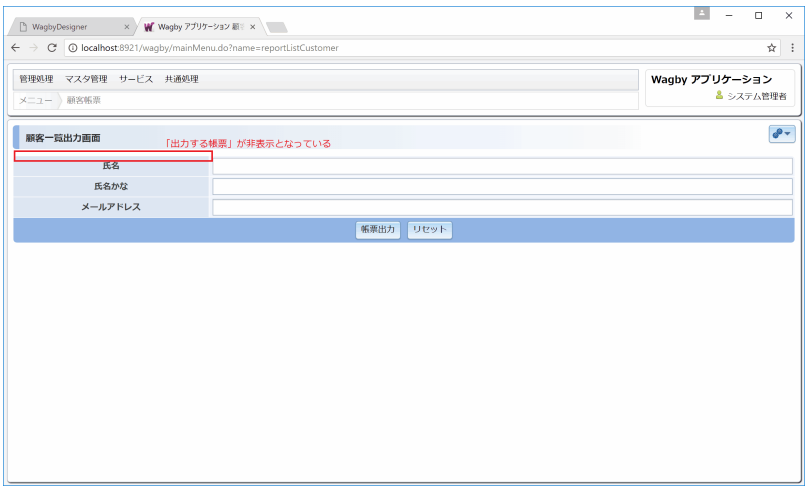


図 「出力する帳票」項目が非表示となっている

定義方法

「出力する帳票が1件のときは選択肢を表示しない」を有効にします。（標準では有効になっています。）

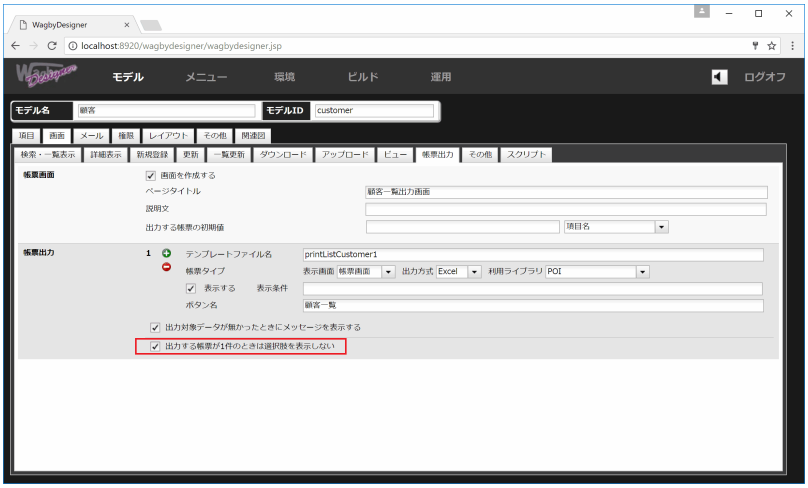


図 出力する帳票が1件のときは選択肢を表示しない

(383)

独自ボタンを用意する（8.2.0）

帳票画面にオリジナルボタンを用意することができます。

項目	説明	入力例
画面名	どの画面にボタンを追加するかを指定します。"帳票画面"とします。	帳票画面
表示する	有効にします。	（チェック）
表示条件	常にボタンを表示する場合、本欄は空白とします。条件については"ボタン表示条件"をお読みください。	
ボタン名	任意の名前を指定します。式で表現することもできます。	（任意の文字列）
イベント名	Wagbyが標準で提供しているイベントと重複しない独自の名前を指定します。 "Original"を接頭語にするとよいでしょう。	Original1
アクション名	指定できません。空白にします。	
追加パラメータ	遷移する画面へパラメータを指定することができます。特にない場合は空白にします。	

(383) この設定は、上のように出力パターンが一つの場合にのみ有効です。帳票画面で選択できる出力パターンが二つ以上ある場合は、選択肢が用意されます。

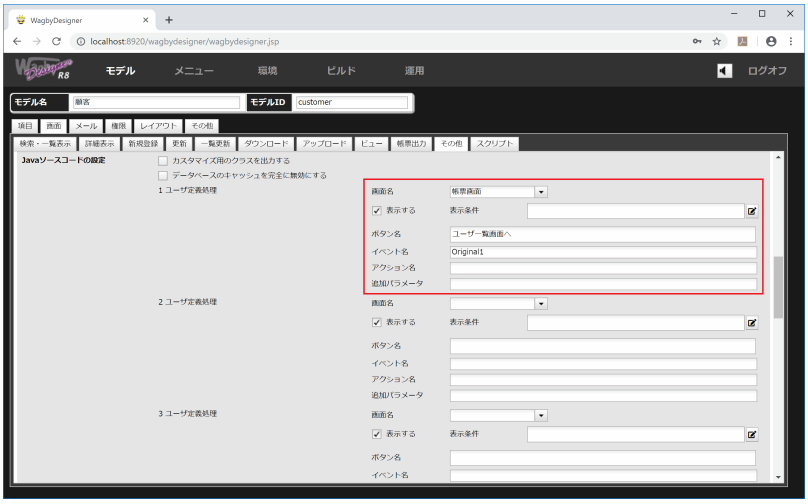


図 帳票画面に用意するボタン

スクリプトを用意する

独自ボタン押下時に呼び出されるスクリプトを用意することができます。WEB-INF/script/<モデルID>フォルダに次の命名規則でスクリプトファイルを用意してください。

ReportList<モデルID>_<イベント名>.js

(384)

例えば customer モデルに、イベント名 Original1 としたときのファイル名は ReportListCustomer_Original1.js となります。

例:ボタン押下でユーザー一覧画面(juser)に遷移する

```
function process() {
  return "redirect:showListJuser.do?action_Reset=Reset";
}
```

(385)

仕様・制約

- 帳票画面では一覧帳票を出力します。詳細帳票の出力は行えません。
- 本設定時は、work/report フォルダにテンプレートファイルは生成（出力）されません。
- この画面では「ボタン表示条件」は無視されます。

出力内容をカスタマイズする(スクリプト)

定義方法

「画面 > スクリプト > ヘルパ」から "帳票内容の編集" を選びます。

(384) モデルIDはキャメル記法ルールが適用されます。[詳細...]

(385) イベント名にResetを指定した場合、検索条件や開始件数がクリアされます。遷移した一覧表示画面では1ページ目から表示されます。8.3.4

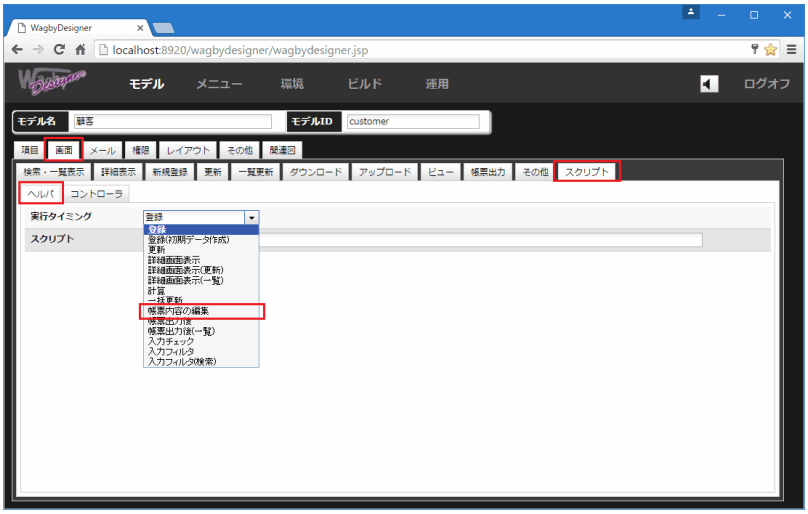


図 スクリプトを記述する

スクリプトが呼び出されるタイミング

作成したスクリプトは、帳票ボタン押下後に標準の帳票データが用意されたあと、印刷処理にそのデータが渡される直前に呼び出されます。そのため、標準の帳票データを差し替えることができます。

仕様・制約

スクリプトで利用できるオブジェクト

スクリプト内では次のオブジェクトが利用できます。

REPLACEMAP

キーおよび値に文字列を格納したマップ（java.util.Map）です。キー部分には次の命名規則があります。

キー表記	説明
${モデルID_p.項目ID}	詳細表示の内容
${モデルID_lp.行番号.項目ID}	一覧表示の内容
${モデルID_cp.項目ID}	一覧表示の検索条件の内容。範囲検索の場合、項目IDの末尾に "1jshparam" と "2jshparam" が付与される。

REPLACEMAPTYPE

項目の型情報を含むマップ（java.util.Map）です。キー部分のルールは REPLACEMAP と同じです。値部分は型情報になります。

例1 REPLACEMAPの内容をコンソールに表示する

REPLACEMAP の実体は java.util.Map です。そのため java.util.Iterator を使ったマップの操作を行うことができます。

```
var it = REPLACEMAP.keySet().iterator();
while (it.hasNext()) {
  var key = it.next();
  var value = REPLACEMAP.get(key);
  print(key+"="+value);
}
```

例2 REPLACEMAPの内容を改変する

次の例は、すべての値に "*" 文字を付与するものです。

```
var it = REPLACEMAP.keySet().iterator();
while (it.hasNext()) {
  var key = it.next();
  var value = REPLACEMAP.get(key);
  value = value + "*";
  REPLACEMAP.put(key,value);
}
```

エラーの扱い

スクリプト中に return 文で任意の文字列を返すと、エラーとして扱います。このとき、スクリプト実行中に BusinessLogicException が生じます。これはログ (system.log) にエラーとして記録されます。

さらに、スクリプト中で画面に表示されるエラーメッセージを用意することもできます。

```
var error = new Jfcerror();
error.content="エラーです";
p.errors.addJfcerror(error);

return "ERROR!";
```

この仕組みを応用して、帳票出力内容がある条件に合致した場合、その出力を行わずにエラーとし、画面にエラーメッセージを表示させることができます。

画像の扱い

プレースホルダ jshfilename:image 表記ではなく、スクリプトで表示範囲を指定することもできます。スクリプトの設定が優先されます。

```
REPLACEMAP.put("${customer_p.phhoto_jshfilename:image:col}","10");
REPLACEMAP.put("${customer_p.phhoto_jshfilename:image:row}","10");
REPLACEMAP.put("${customer_p.phhoto_jshfilename:image:sx}","0.5");
REPLACEMAP.put("${customer_p.phhoto_jshfilename:image:sy}","0.5");
```

値は文字列なので、ダブルクォートで囲む必要があります。

(386)

JasperReportsを利用したPDF出力 (1) 詳細表示

JasperReportsとは

JasperReportsとは、Java で動作するオープンソースの帳票出力ライブラリです。

Wagby は JasperReports を使って PDF,RTF 形式の帳票を出力することができます。 本ドキュメントでは Wagby から JasperReports へデータを渡す方法を説明します。

JasperReportsの参考資料は次のとおりです。

- 日本JasperServerユーザ会 - iReportで作ろう
- 日本JasperServerユーザ会 - 日本語を出力する
- Jaspersoft Community 情報

JasperReports の使い方や帳票の設計方法は Wagby サポートの対象外となります。JasperReports技術者を有しているWagby販売パートナーへご相談ください。いくつかのパートナーで、トレーニングなどの講習会を実施しています。

(386) 帳票テンプレートのプレースホルダに jshfilename:image 指定がなければ、上記の設定は無視されます。

準備するもの

JasperReports Library

Wagbyに同梱されています。jasperreports.jarは6.2.0です。また、iText.jarとiTextAsian.jarも含まれています。お使いの Jaspersoft Studio に含まれる jar ファイルに置換することもできます。[詳細...]

Jaspersoft Studio

開発者自身でダウンロードし、インストールしてください。本ドキュメント執筆時に用いたバージョンは5.6.0.finalです。

XML形式のサンプルデータ

Wagbyは帳票の元となるデータをXML形式のファイルにしてJasperReportsに渡します。ここでは、JasperReportsに渡すXMLファイルを予め入手し、帳票設計に使うものとします。次の手順で入手します。

1. ビルドしたWagbyアプリケーションを起動します。
2. wagbyapp¥webapps¥wagby¥$(モデル名)¥show$(モデル名)_body.jsp をテキストエディタで開きます。Windows付属のメモ帳は使えません。TeraPadなどのテキストエディタを使ってください。
3. 同ファイルに <c:out value="${(モデル名)_p}"/> という一行を加えます。customerモデルの例を示します。

```
<%@ page pageEncoding="UTF-8" %>
<fmt:setBundle basename="jfcapp"/>
<c:out value="${customer_p}"/>
<table width="100%">
...
```

4. アプリケーションにログオンし、このモデルの詳細表示画面を開きます。画面上部に XML 形式のデータが表示されます。これがJaspersoft Studioで帳票レイアウト設計に用いるテストデータになります。
5. 表示された XML データをマウスを使って選択し、コピーします。同データをテキストファイルとして保存します。（ファイル名は半角英数字の組み合わせとし、拡張子は ".xml" とします。）文字コードは utf-8 とします。
6. さらに、このテキストファイルの先頭部分に次の二行を加えます。

```
<?xml version="1.0" encoding="utf-8"?>
<root>
```

7. 最終行に、次の一行を加えます。

```
</root>
```

以降の説明では、詳細帳票用テストデータを "c:¥customer_show.xml" として保存したとします。

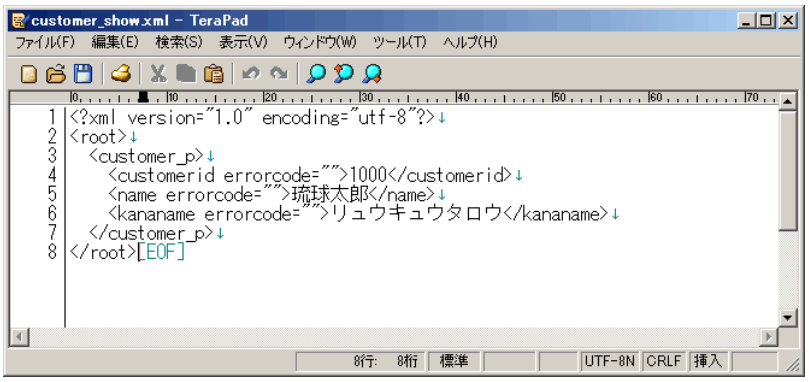


図 用意したcustomer_show.xml

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  <customer_p>
    <customerid errorcode="">1000</customerid>
```

```
    <name errorcode="">琉球太郎</name>
    <kananame errorcode="">リュウキュウタロウ</kananame>
  </customer_p>
</root>
```

Jaspersoft Studioを使ったレイアウト設計

Jaspersoft Studioを使って帳票レイアウトを設計します。具体的には"Text Field"を画面上に配置していき、各TextFieldが(XMLデータの)どの要素にあたるかを設定していきます。本ドキュメントではJaspersoft Studioの詳細な使い方は割愛します。

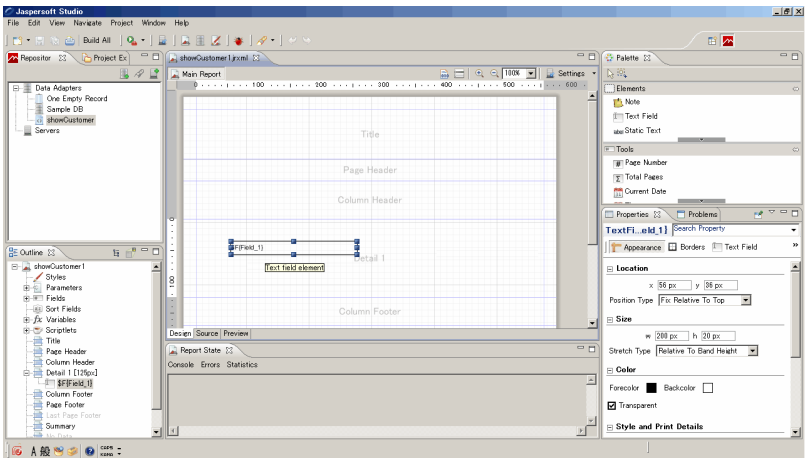


図 JaspersoftStudio

設定のポイントは次のとおりです。

Data Adapter Wizard

データソース「XML document」を用意します。例では名前を「showCustomer」とし、customer_show.xmlと紐づけます。

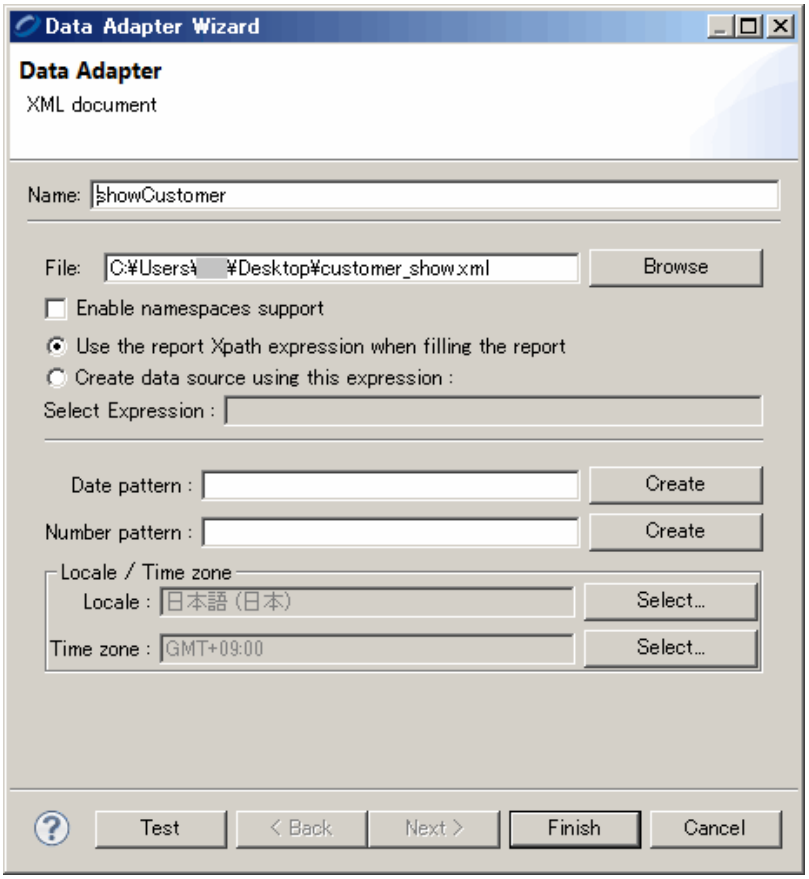


図 DataAdapterWizard設定画面

Dataset and Query

Dataset and Query Dialogを開きます。

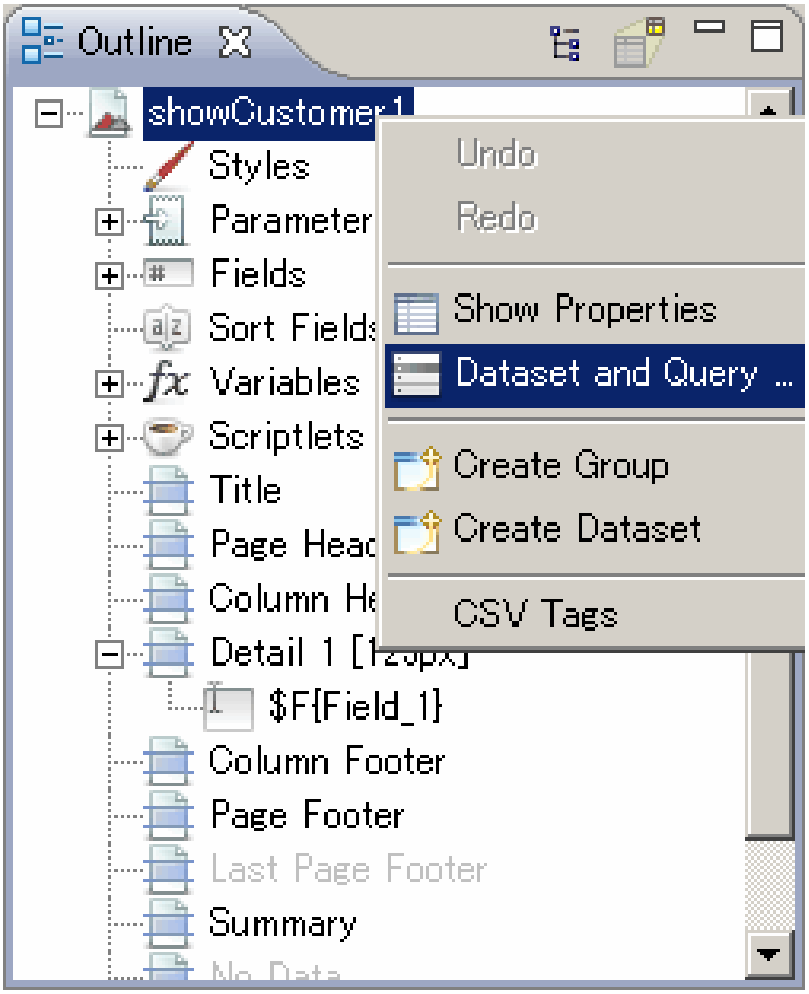


図 DatasetandQueryDialogを開く

データソースに showCustomer を選択します。Language は "XPath" とします。先頭の "root" 要素には "/" という文字を設定します。画面に表示するテキストフィールドと、データのどのノードかを紐づけていきます。

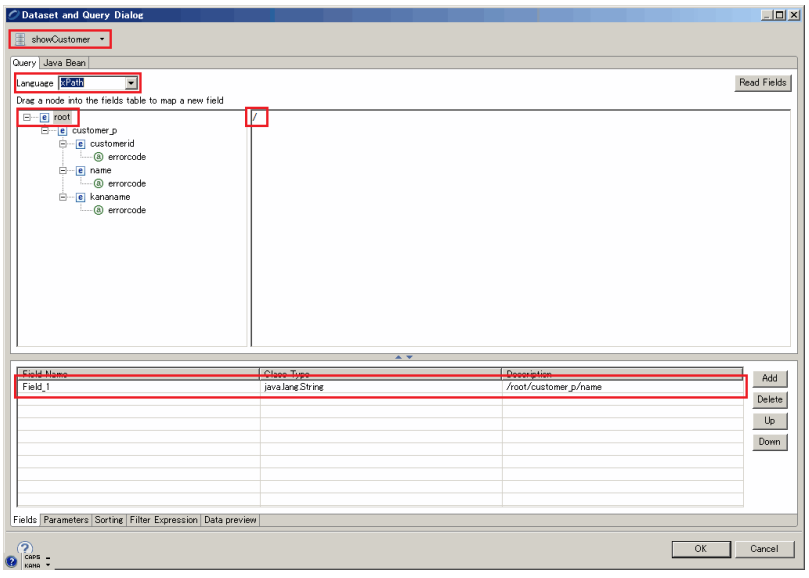


図 XML要素とFieldの紐づけ

Expression Editor

帳票レイアウトに用意するフィールドの詳細な設定を行います。

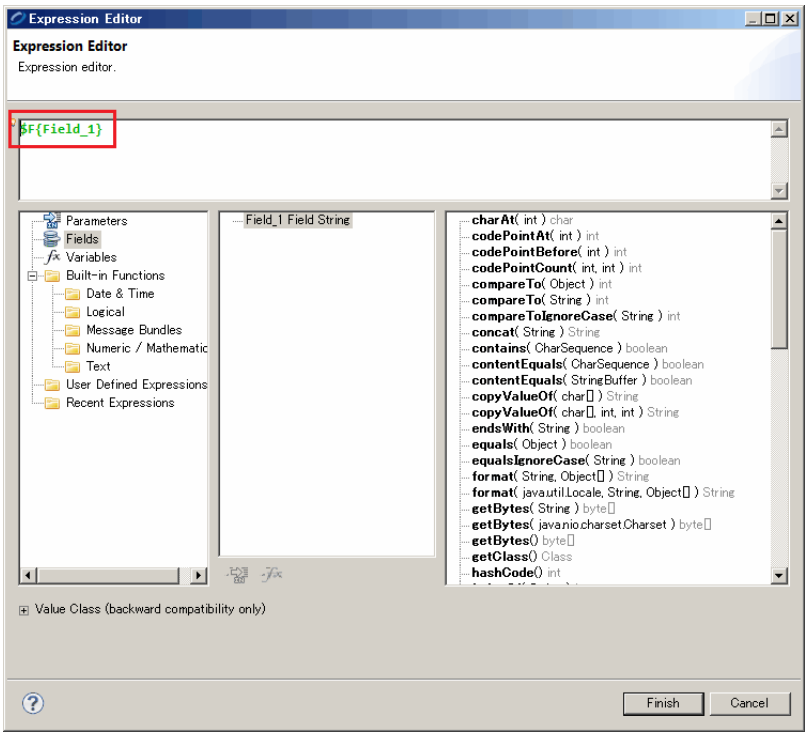


図 ExpressionEditor

PDF Encoding, PDF Font Name

テキストフィールドのプロパティで、PDF Encoding と PDF Font Name を指定します。ここでは Encoding を "UniJIS-UCS2-H" とし、Font Name を "HeiseiKakuGo-W5" とします。（この設定を行わなかった場合は、アプリケーションから PDF を出力する際に日本語が表示されません。）

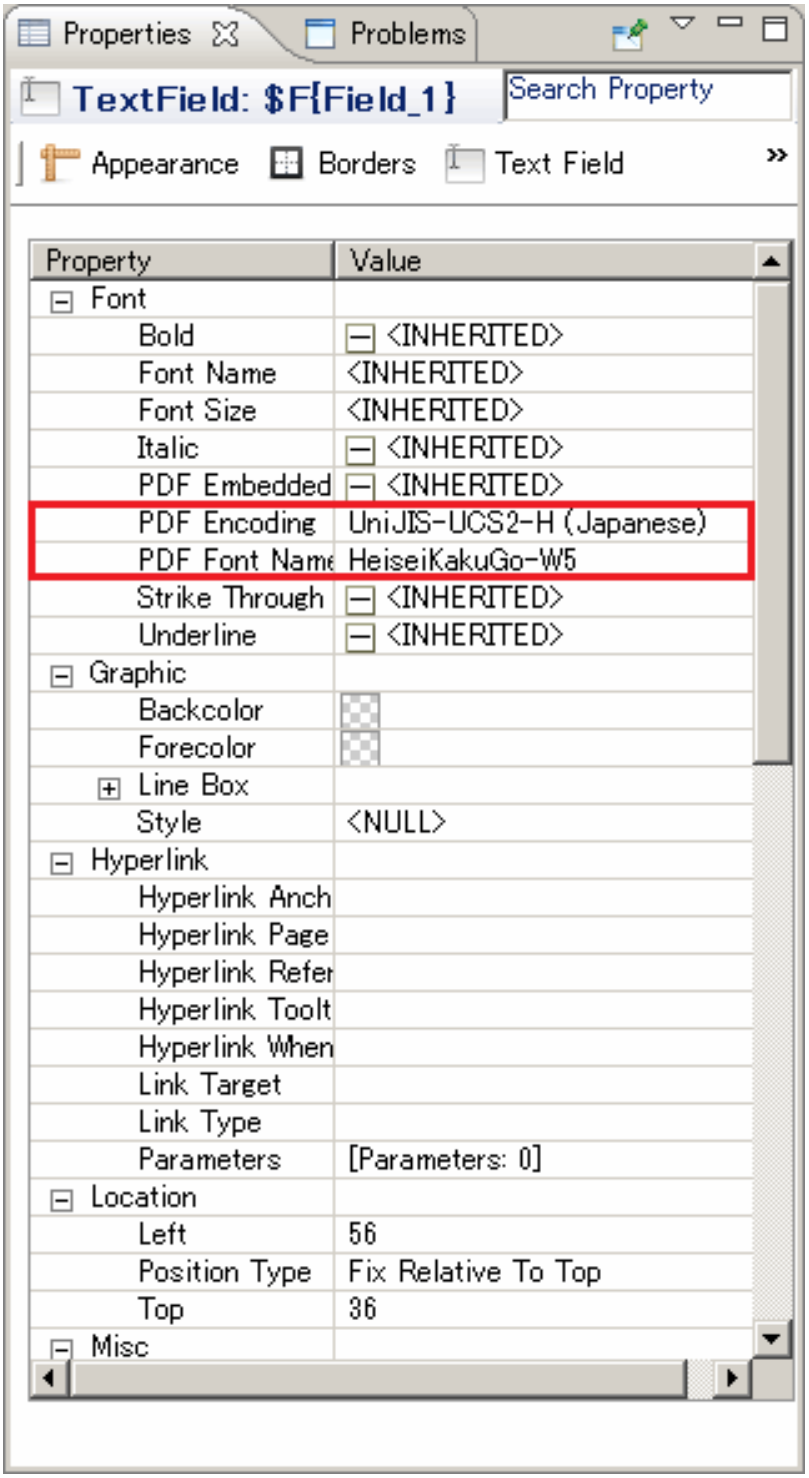


図 Property

Preview

Previewタブを選択すると、帳票イメージが表示されます。

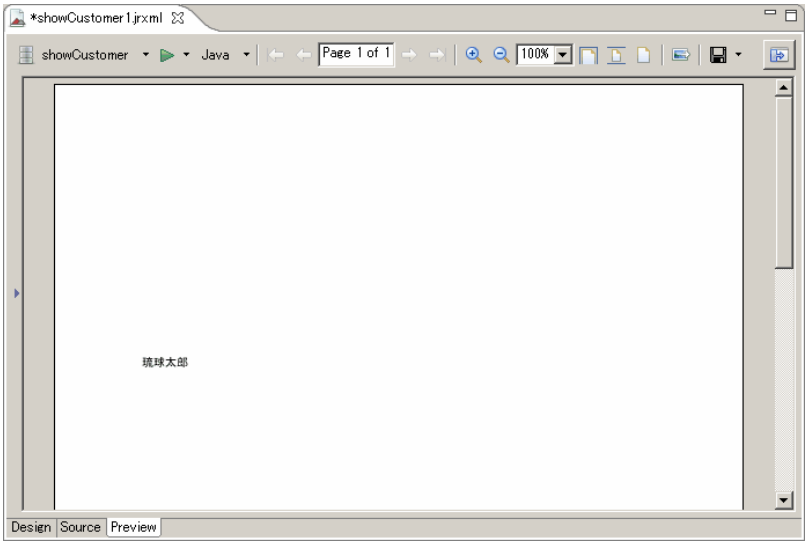


図 Preview

Source

すべての設定はSourceタブで確認できます。詳細帳票のサンプルを示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Created with Jaspersoft Studio version last-->
<jasperReport xmlns="http://jasperreports.sourceforge.net/jasperreports" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://jasperreports.sourceforge.net/
jasperreports http://jasperreports.sourceforge.net/xsd/jasperreport.xsd" name="showCustomer1"
 pageWidth="595" pageHeight="842" columnWidth="555" leftMargin="20" rightMargin="20"
 topMargin="20" bottomMargin="20" uuid="ee5f7461-8b85-4830-9157-4365647f44c4">
  <property name="com.jaspersoft.studio.data.defaultdataadapter" value="showCustomer"/>
  <queryString language="xPath">
    <![CDATA[/]]>
  </queryString>
  <field name="Field_1" class="java.lang.String">
    <fieldDescription><![CDATA[/root/customer_p/name]]></fieldDescription>
  </field>
  <background>
    <band splitType="Stretch"/>
  </background>
  <title>
    <band height="79" splitType="Stretch"/>
  </title>
  <pageHeader>
    <band height="35" splitType="Stretch"/>
  </pageHeader>
  <columnHeader>
    <band height="61" splitType="Stretch"/>
  </columnHeader>
  <detail>
    <band height="125" splitType="Stretch">
      <textField>
        <reportElement stretchType="RelativeToBandHeight" x="56" y="36" width="200" height="20"
 uuid="d9d975f0-1a46-4f50-be3f-73f83f9dd1b4">
          <property name="local_mesure_unitwidth" value="pixel"/>
          <property name="com.jaspersoft.studio.unit.width" value="px"/>
        </reportElement>
        <textElement>
          <font pdfFontName="HeiseiKakuGo-W5" pdfEncoding="UniJIS-UCS2-H"/>
        </textElement>
        <textFieldExpression><![CDATA[$F{Field_1}]]></textFieldExpression>
```

```
      </textField>
    </band>
  </detail>
  <columnFooter>
    <band height="45" splitType="Stretch"/>
  </columnFooter>
  <pageFooter>
    <band height="54" splitType="Stretch"/>
  </pageFooter>
  <summary>
    <band height="42" splitType="Stretch"/>
  </summary>
</jasperReport>
```

XPath指定の詳細

図における「Description」は XPath 表記となります。

項目の型	表記	説明	指定例
文字列,数値,日付	/root/モデル名_p/項目名	項目の値が転記されます。	/root/customer_p/customer_name
モデル参照（チェックボックスを除く）	/root/モデル名_p/項目名/@content	内容部が転記されます。	/root/customer_p/companyname/@content
	/root/モデル名_p/項目名/@id	モデルのID部が転記されます。	/root/customer_p/companyname/@id
モデル参照（チェックボックス）	/root/モデル名_p/項目名[添字]/@content	添字に対応したモデルの内容部が転記されます。添字は「1」から開始されます。	/root/customer_p/customertype[@choose=true()][2]/@content
	/root/モデル名_p/項目名[添字]/@id	添字に対応したモデルのID部が転記されます。添字は「1」から開始されます。	/root/customer_p/customertype[@choose=true()][2]/@id
繰り返し項目	/root/モデル名_p/項目名[添字]	添字に対応した繰り返し項目の値が転記されます。添字は「1」から開始されます。	/root/customer_p/email[2]/@content
繰り返しコンテナ内の項目	/root/モデル名_p/コンテナの項目名[添字]/項目名	添字に対応した繰り返しコンテナ内の項目の値が転記されます。添字は「1」から開始されます。	/root/customer_p/report[2]/rdate

モデル参照(チェックボックス)、繰り返し項目、繰り返しコンテナ内の項目の場合、[添字]を省略すると[1]が指定されたものとして動作します。

可変長データの扱い

上記で紹介したモデル参照(チェックボックス)、繰り返し項目、繰り返しコンテナ内の項目等で「添字」を使う方式では定義されたレポートフィールドの分のみ帳票に表示することができます。たとえば繰り返し項目 email にデータが 10 個あったとして、レポートフィールドを 2 個分(email[1], email[2])しか定義していなかった場合は 2 個以上のデータを出力することはできません。

入力データのデータ数によって表示件数を動的に変化させたい場合は JaspreReports の「サブレポート機能」を使うことができます。ただしサブレポートの定義方法は複雑なため、本ドキュメントでは説明を割愛します。詳細はJasperReports技術者を有しているWagby販売パートナーにお問い合わせください。

ビルドと実行

定義

「画面>帳票出力」で帳票テンプレートを用意します。テンプレートファイル名を決め、詳細表示-PDFで利用ライブラリにJasperReportsを選択します。

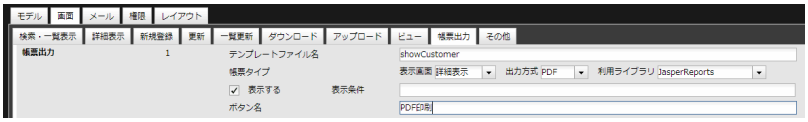


図 帳票設定

帳票テンプレート設定

ビルドしたアプリケーションにログオンし、帳票テンプレートを登録します。

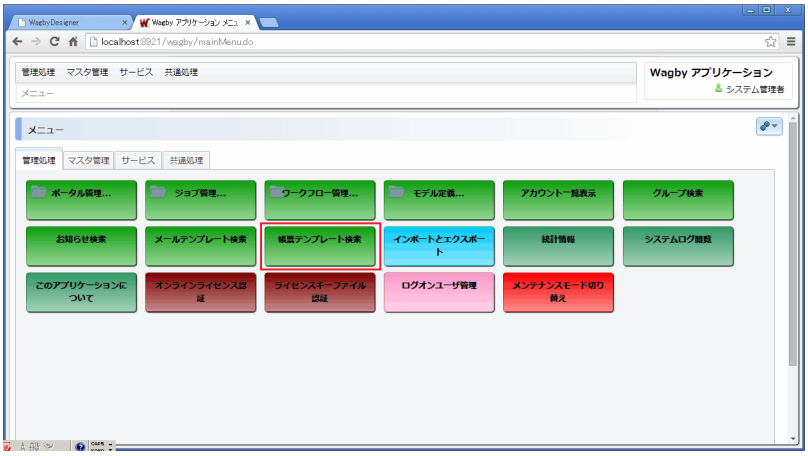


図 メニューから帳票テンプレートを選択する

IDは、図で設定したテンプレートファイル名を指定します。テンプレートファイルは、Jaspersoft Studioで作成したjrxmlファイルを指定します。

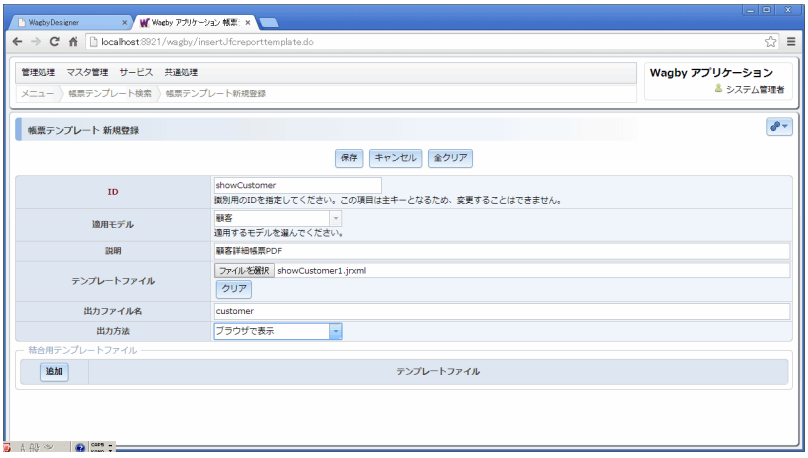


図 帳票テンプレート新規登録画面

jrxmlファイルはテンプレートとして登録します。

コンパイルされたjasperファイルを登録する（8.3.0）

帳票テンプレートファイルに　jrxml　ではなく（これを事前にコンパイルした）jasper　ファイル(*1) を登録することもできます。実行時に毎回、コンパイルする処理が省略されるため、パフォーマンスが向上します。

実行

顧客モデルを新規に登録します。詳細表示画面に「PDF印刷」ボタンが用意されています。（ボタン名は図で設定したものです。）

図 顧客の詳細画面

PDF印刷ボタンを押下すると新しいタブが用意され、PDFが出力されます。図で出力方法に「ダウンロード」を指定すると、PDFファイルのダウンロードが行われるようになります。

図 PDF印刷の例

JDBCによるデータベース接続

JasperReportsにはJDBCによるデータベース接続機能があります。SQLを用いることで、Wagbyが生成したテーブルを直接操作してデータを収集し、レポートを出力させることもできます。

Designerの帳票設定では、"JasperReports-JDBC" を選択してください。

(387) 1. jasperファイルの実体は、Javaのシリアライズオブジェクトです。jrxmlファイルをコンパイルするときに用いた Java のバージョンは、Wagby を実行する Java のバージョンと合わせるようにしてください。

図 利用ライブラリの選択(JDBC)

Japsersoft Studio での JDBC の設定は割愛します。設定ファイル jrxml の queryString 要素には次のようにSQLを指定することができます。

```
<queryString><![CDATA[select * from "juser" where "userid" = 'admin']]></queryString>
```

テーブル名や項目名はダブルクォートで囲みます。固定の値はシングルクォートで囲みます。

JDBC 設定を利用した場合、データセットに XML を用いることはできません。（指定しても無視されます。） 具体的には /root/モデル名_p/項目名 などと記述しても、実行時にログに "No XML source was provided." と出力され、PDFには何も表示されません。

利用するライブラリのバージョン

ライブラリ（jarファイル）は Wagby に同梱されているものではなく、お使いの Jaspersoft Studio に同梱されているライブラリをご利用いただくこともできます。具体的な手順を示します。

1. Jaspersoft Studio に同梱されている jasperreports と iText の jar ファイルを探します。5.6.2.final では以下のフォルダに格納されていました。
   - configuration/org.eclipse.osgi/bundles/38/1/.cp/lib/jasperreports-5.6.1.jar
   - configuration/org.eclipse.osgi/bundles/6/1/.cp/lib/iText-2.1.7.js2.jar

   必要なファイルは上記2ファイルです。バージョン番号は上記例とは異なっている場合があります。
2. それぞれのファイル名を変更します。jasperreports-5.6.1.jar は jasperreports.jar へ、iText-2.1.7.js2.jar は iText.jar とします。
3. 名前を変更した2つのファイルを wagbyapp/webapps/wagby/WEB-INF/lib フォルダへコピー（上書き）します。
4. Wagby アプリケーションを再起動し、PDF出力を行います。

この手順で出力が確認できましたら、2. で用意した jar ファイルを customize/webapp/WEB-INF/lib フォルダへコピーします。このあとのビルド処理では、これらのファイルが使われるようになります。

よくある質問と回答

Q. プレースホルダが空白で出力されてしまいます。 !-- N20180213_232202 -->

A. JasperReportのフィールド名(field/@name)を日本語にすると正しく出力されない場合があります。フィールド名には英字を利用してください。

Q. サブレポートの配置場所を教えてください。

A.「帳票テンプレート」の登録機能はサブレポートに対応していません。サブレポートの配置場所はメインの帳票テンプレートに相対パスを埋め込む形で行ってください。

標準の起点フォルダは wagbyapp/bin になります。

Q. 画像を埋め込む場合、画像ファイルの配置場所を教えてください。

A. 帳票内で固定で利用する画像は、WEB-INF以下に保存の上、JasperReportの設定で（画像ファイルへの）パスを指定してお使いください。

wagbyappの実行時のパスは、wagbyapp/bin になります。(startup.batがこのフォルダで起動します。）そのため、ここを起点とした相対パス表記または、WEB-INF以下のフォルダをJasperReportが認識するのであれば、そこを起点とした相対パス表記で画像ファイルを読み込ませるようにするとよいでしょう。

wagbyapp/bin へ画像ファイルを配置する場合は、customize/tomcat/bin に当該ファイルを配置してください。 以降ビルドのタイミングで wagbyapp/bin へコピーされるようになります。 WEB-INF以下であれば、customize/webapp/WEB-INF 以下に配置することで同様にビルド時にコピーされます。

トラブルシューティング

Chromeブラウザで PDFが表示されないことがある

Chromeの標準のPDF Viewerに代わって、拡張機能のPDF Viewerを追加インストールすることで改善される場合があります。

JasperReportsを利用したPDF出力（2）一覧表示

はじめに

本ドキュメントは「JasperReportsを利用したPDF出力（1）詳細表示」の知識を前提に、一覧表示データの扱いで異なる点を説明します。

> JasperReports の使い方や帳票の設計方法は Wagby サポートの対象外となります。JasperReports技術者を有しているWagby販売代理店へご相談ください。いくつかの代理店で、トレーニングなどの講習会を実施しています。

準備するもの

XML形式のサンプルデータ

1. ビルドしたWagbyアプリケーションを起動します。
2. wagbyapp¥webapps¥wagby¥$(モデル名)¥showList$(モデル名)_body.jsp をテキストエディタで開きます。Windows付属のメモ帳は使えません。TeraPadなどのテキストエディタを使ってください。
3. <c:out value="${(モデル名)_lp}"/> と <c:out value="${(モデル名)_cp}"/> という二行を加えます。前者は一覧表示データで、後者は検索条件データです。customerモデルの例を示します。

```
<%@ page pageEncoding="UTF-8" %>
<fmt:setBundle basename="jfcapp"/>
<c:out value="${customer_lp}"/>
<c:out value="${customer_cp}"/>
```

4. アプリケーションにログオンし、このモデルの一覧表示画面を開きます。画面上部に XML 形式のデータが表示されます。これがJaspersoft Studioで帳票レイアウト設計に用いるテストデータになります。
5. 表示された XML データをマウスを使って選択し、コピーします。同データをテキストファイルとして保存します。（ファイル名は半角英数字の組み合わせとし、拡張子は ".xml" とします。）文字コードは utf-8 とします。
6. さらに、このテキストファイルの先頭部分に次の二行を加えます。

```
<?xml version="1.0" encoding="utf-8"?>
<root>
```

7. 最終行に、次の一行を加えます。

```
</root>
```

以降の説明では、一覧帳票用テストデータを "c:¥customer_showlist.xml" として保存したとします。

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  <customer_lp sortable="customerid " sort="" order=""><!-- 一覧表示用のデータ -->
    <item>
      <customerid>1000</customerid>
      <name>琉球太郎</name>
    </item>
    <item>
```

```
      <customerid>1001</customerid>
      <name>琉球次郎</name>
    </item>
  </customer_lp>
  <customer_cp><!-- 検索条件として入力した値 -->
    <customerid1jshparam errorcode=""></customerid1jshparam>
    <customerid2jshparam errorcode=""></customerid2jshparam>
    <name errorcode=""></name>
  </customer_cp>
</root>
```

一覧表示ではデータ1件毎にitem要素が用意されます。表示データが10件あれば10個のitem要素が存在します。

Jaspersoft Studioを使ったレイアウト設計

詳細表示設定と異なる点を説明します。

Data Adapter Wizard

データソース「XML document」を用意します。例では名前を「showListCustomer」とし、customer_showlist.xmlと紐づけます。

詳細表示設定と異なるのは、Select Expression 欄に /root/customer_lp/item まで記述することです。

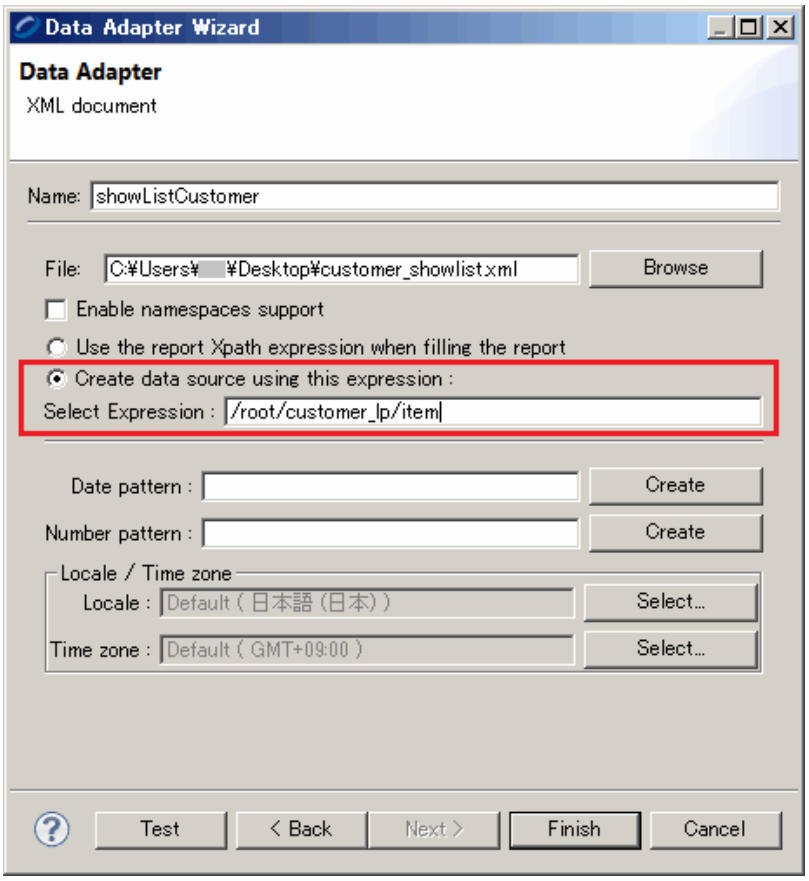


図 DataAdapterWizard設定画面

Dataset and Query

図で Select Expression を指定済みのため、各フィールドの Description は、その後の部分だけを指定するだけで良いです。

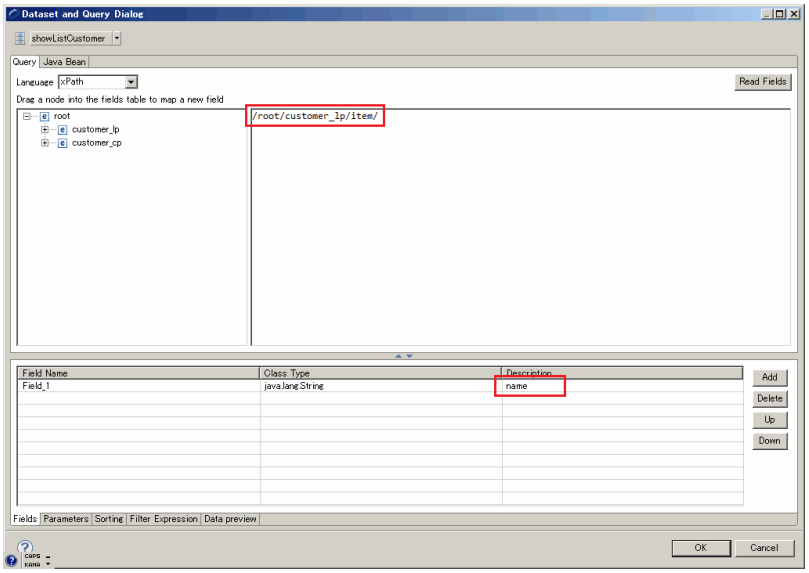


図 DatasetandQueryDialog

Source

すべての設定はSourceタブで確認できます。一覧帳票のサンプルを示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Created with Jaspersoft Studio version last-->
<jasperReport xmlns="http://jasperreports.sourceforge.net/jasperreports"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
jasperreports.sourceforge.net/jasperreports http://jasperreports.sourceforge.net/
xsd/jasperreport.xsd" name="showListCustomer1" pageWidth="595" pageHeight="842"
 columnWidth="555" leftMargin="20" rightMargin="20" topMargin="20" bottomMargin="20"
 uuid="b7ae9a9e-9321-4dc5-9b76-e4fa9877d839">
  <property name="com.jaspersoft.studio.data.defaultdataadapter" value="showListCustomer"/>
  <property name="com.jaspersoft.studio.layout"
 value="com.jaspersoft.studio.editor.layout.FreeLayout"/>
  <queryString language="xPath">
    <![CDATA[/root/customer_lp/item]]>
  </queryString>
  <field name="Field_1" class="java.lang.String">
    <fieldDescription><![CDATA[name]]></fieldDescription>
  </field>
  <background>
    <band splitType="Stretch"/>
  </background>
  <title>
    <band height="79" splitType="Stretch"/>
  </title>
  <pageHeader>
    <band height="35" splitType="Stretch"/>
  </pageHeader>
  <columnHeader>
    <band height="61" splitType="Stretch"/>
  </columnHeader>
  <detail>
    <band height="125" splitType="Stretch">
      <textField>
        <reportElement x="180" y="30" width="100" height="20"
 uuid="84eea6ac-1884-4e4c-87b5-45f1064327d9"/>
        <textElement>
          <font pdfFontName="HeiseiKakuGo-W5" pdfEncoding="UniJIS-UCS2-H"/>
        </textElement>
```

```
        <textFieldExpression><![CDATA[$F{Field_1}]]></textFieldExpression>
      </textField>
    </band>
  </detail>
  <columnFooter>
    <band height="45" splitType="Stretch"/>
  </columnFooter>
  <pageFooter>
    <band height="54" splitType="Stretch"/>
  </pageFooter>
  <summary>
    <band height="42" splitType="Stretch"/>
  </summary>
</jasperReport>
```

XPath指定の詳細

一覧帳票では基準となるXPath(/root/customer_lp/item)を既に指定しているので「Description」は項目名のみとなります。

項目の型	表記	説明	指定例
文字列,数値,日付	項目名	項目の値が転記されます。	customer_name
モデル参照（チェックボックスを除く）	項目名/@content	内容部が転記されます。	companyname/@content
	項目名/@id	モデルのID部が転記されます。	companyname/@id
モデル参照（チェックボックス）	項目名[添字]/@content	添字に対応したモデルの内容部が転記されます。添字は「1」から開始されます。	customertype[2]/@content
	項目名[添字]/@id	添字に対応したモデルのID部が転記されます。添字は「1」から開始されます。	customertype[2]/@id
繰り返し項目	項目名[添字]	添字に対応した繰り返し項目の値が転記されます。添字は「1」から開始されます。	email[2]/@content
繰り返しコンテナ内の項目	コンテナの項目名[添字]/項目名	添字に対応した繰り返しコンテナ内の項目の値が転記されます。添字は「1」から開始されます。	report[2]/rdate

JDBCによるデータベース接続

JasperReportsにはJDBCによるデータベース接続機能があります。SQLを用いることで、Wagbyが生成したテーブルを直接操作してデータを収集し、レポートを出力させることもできます。

Japsersoft Studio での JDBC の設定は割愛します。設定ファイル jrxml の queryString 要素には次のようにSQLを指定することができます。

```
<parameter name="juser_cp.userid" isForPrompting="true" class="java.lang.String">
  <defaultValueExpression ><![CDATA["admin"]]></defaultValueExpression>
</parameter>

<queryString><![CDATA[select * from "juser" where "userid" = $P{juser_cp.userid}]]></
queryString>
```

テーブル名や項目名はダブルクォートで囲みます。$P{ ... } という表記でパラメータを指定することもできます。 "モデル名_cp" は、検索条件が格納されたモデルを指します。

利用するライブラリのバージョン

"帳票 > JasperReportsを利用したPDF出力 (1) 詳細表示 > 利用するライブラリのバージョン"をお読みください。

トラブルシューティング

空の帳票が出力される

設定ファイル jrxml の queryString の記述に誤りがあります。スペルミスや、正しいモデル名かどうかを確認ください。

例: 次の設定は customer モデルを対象にした一覧帳票ですが、root の次の表記に誤りがあります。

```
<queryString language="xPath">
    <![CDATA[/root/customer/item]]>
</queryString>
```

正しくは次のとおりです。

```
<queryString language="xPath">
    <![CDATA[/root/customer_lp/item]]>
</queryString>
```

JasperReportsを利用したPDF出力 (3) ファイルの結合

動作

請求書の出力業務で、最初の数ページは約款でそのあとに実際の請求書が含まれるようなケースを想定します。

JasperReportでは1テンプレートファイルで1パターンを作成することが基本になります。 そこで下記のように複数のテンプレートファイルを用意することになります。

```
約款の1ページ目の帳票テンプレートファイル.jrxml
約款の2ページ目の帳票テンプレートファイル.jrxml
...
請求書の帳票テンプレートファイル.jrxml
```

これらをまとめて、1つのPDFとして出力するようにします。

定義方法

帳票テンプレートに指定します。

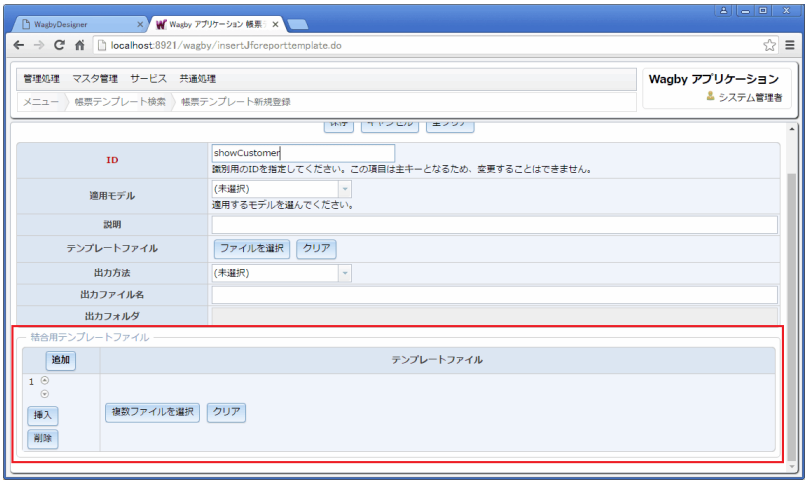


図 結合用テンプレートファイルの設定(JapserReport用)

次のルールが適用されます。

- 二つ目以降のテンプレートファイルは「結合用テンプレートファイル」（繰り返しコンテナ）に登録する。登録順に結合される。
- ただし通常の項目（「テンプレートファイル」）にもJaspterReportのテンプレートファイルが指定された場合は、それを1ページ目とする。
- 帳票テンプレートモデルで出力ファイル名が未指定のときは、通常項目のテンプレートファイル名を用いる。ただし通常項目にテンプレートファイル名が指定されていないときは、繰り返しコンテナの1番目のテンプレートファイル名を用いる。

メール送受信

メール送信（メールテンプレートを固定する）

定義方法

メール送信の設定は、次の手順で行います。

1. Designerによるボタン設定とビルド。
2. （ビルドした）Wagbyアプリケーションで行う「メールテンプレート」の設定

ここでは、それぞれの手順の詳細を説明します。

Designerの設定

「メール>送信」タブを開きます。画面機能毎のタイミングでメール送信の設定を行うことができます。ここでは顧客データの「登録時」に営業部全員へ自動メール配信を行う例を説明します。

設定項目	説明	入力例
メールを送信する	このチェックを有効にするとメール送信機能が動作します。	（チェック）
送信のタイミング	″処理完了時″または″メール送信ボタン″を選択できます。処理完了時は自動メール配信です。メール送信ボタンを用意すると、ボタンを押したときにメールを送信します。一般的には登録・更新・削除といった動作を伴う場合は″処理完了時″を、詳細画	処理完了時

	面では″メール送信ボタン″を選択します。	
メール送信ボタン	送信のタイミングが″メール送信ボタン″のとき、ボタン名に設定した名前でメール送信用のボタンが配置されます。	(チェック)
メールテンプレート項目	後述する「メールテンプレート」の識別子(ID)になります。重複しない名前を英数字の組み合わせで決めてください。	announceForSalesTeam
メール送信後にメッセージを表示する	メール送信が成功または失敗したというメッセージを画面に表示します。	
送信条件	メール送信の条件を指定することができます。条件に合致しなければ、メールは送信されません。[詳細...]	${status}==1
成功時のアクション	メール送信が成功した場合に実行する式を指定できます。[詳細...]	${count}=${count}+1
失敗時のアクション	メール送信が失敗した場合に実行する式を指定できます。[詳細...]	${status}=-1

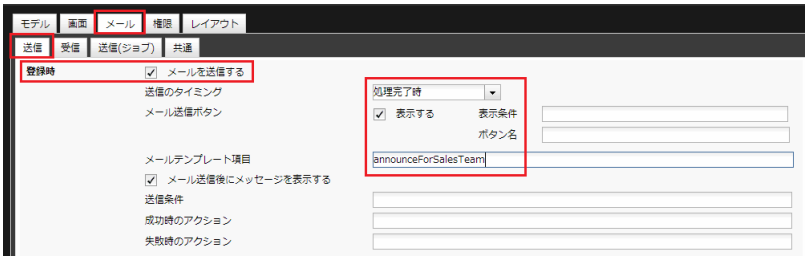


図 登録時の自動メール送信の設定

成功時、失敗時の処理

メール送信が成功した場合、または失敗した場合に実行する式を指定できます。 成功時、失敗時の式は代入形式となります。代入のためのイコールが必要です。

メール送信時に「送信数」という項目をインクリメントする場合は次のように記述します。

```
${mailsend_cnt}=${mailsend_cnt}+1
```

式はセミコロンで区切って複数、記述することができます。

```
${mailsend_cnt}=${mailsend_cnt}+1;${status}=IF(${mailsend_cnt}==0,"Zero","Non-Zero")
```

記述の制約

- 式に関数を用いることはできますが、SUMやCOUNTといった集合関数は利用できません。
- モデル参照項目に対して #content 指定を行うことができません。#id のみ利用できます。
- ダウンロードと一覧表示機能でのメール送信は、モデルの項目を条件に使うことはできません。メールのプレースホルダは「件数」のみが利用できます。モデル項目を使わない式として、例えばHASPRINCIPAL("プリンシパル名") といった権限による制御などは行えます。

モデル定義の制約 !-- N20160719_143046, BR9349 -->

変更履歴を残す設定を行ったモデルには、成功時/失敗時の処理を記述できません。（この制約は将来のWagbyで改善する予定です。）

送信条件

メール送信の条件を指定することができます。条件に合致しなければ、メールは送信されません。

モデル項目「ステータス」が "1" のときメールを送信するという場合は次のように記載します。条件判定式のため、イコールは二つ必要です。

```
${status}==1
```

「成功時のアクション」と組み合わせることで、メール送信成功時にフラグを立て、そのフラグが立っている場合は二度目以降の送信は行えないといった制御を行うことができます。

記述の制約

「${content_10_name}」と同じです。

リトライ回数と間隔を指定する

標準では、メール送信に失敗したときのリトライは行われません。次の手順で、リトライの回数と間隔を指定することができます。

1. customize/resources フォルダに myMimeMailUtilities.properties.UTF8 ファイルを新しく用意します。
2. 次のエントリを追加します。

```
jfc.retry.interval=10000
jfc.max.retry.count=10
```

パラメータ jfc.retry.interval はリトライの間隔を示します。単位はミリ秒です。 パラメータ jfc.max.retry.count はリトライを含めた送信回数を示します。この例では10回となっており、初回のメール送信処理に失敗すると最大9回のリトライ処理を行います。（メールの送信に成功した時点でリトライ処理は終了します。）

3. このファイルを保存し、ビルド処理を行います。

my が接頭語に付与されたリソーフファイルはビルド時に結合されます。詳細は"Wagby Developer Network(R8) > Javaを用いたカスタマイズ > リソースファイル"をお読みください。

この設定はすべてのメール送信に適用されます。

ログにCcとBccアドレスを含める

標準では、メール送信のログ（成功および失敗）に記録される送信先は To で指定したアドレスのみとなっています。 次の手順で、Cc および Bcc アドレスもログに記録させることができます。

1. customize/resources フォルダに myMimeMailUtilities.properties.UTF8 ファイルを新しく用意します。
2. 次のエントリを追加します。

```
jfc.log.cc=true
jfc.log.bcc=true
```

パラメータ jfc.log.cc および jfc.log.bcc はそれぞれ Cc、Bcc のアドレスをログに記録するかどうかを指定します。いずれか、または両方を同時に指定できます。

3. このファイルを保存し、ビルド処理を行います。

my が接頭語に付与されたリソーフファイルはビルド時に結合されます。詳細は"Wagby Developer Network(R8) > Javaを用いたカスタマイズ > リソースファイル"をお読みください。

この設定はすべてのメール送信に適用されます。

トラブルシューティング

com.sun.mail.smtp.SMTPSendFailedException : Invalid sender address エラー

メールテンプレートで指定した「差出人」が、サーバー送信時のアカウントと同じになっているかどうか、確認してください。メールサーバ側で、不一致の場合はエラーとするという設定になっている可能性があります。

仕様・制約

- 指定したアドレスに誤りがありメールが届かなかった場合、ログファイルに送信失敗が記録されます。

メールテンプレート

ビルドしたアプリケーションにシステム管理者でログオンします。図に設定したメールテンプレートの詳細な設定を行います。メールテンプレートの新規登録を行います。

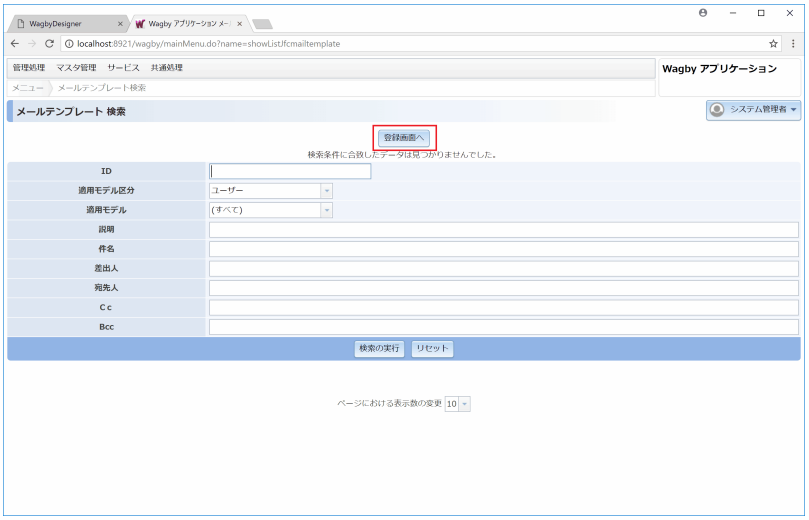


図 メールテンプレートの新規登録

メールテンプレートの設定画面です。

「適用モデル」で「顧客」を選択します。 「メールサーバ」「件名」「差出人」などを設定します。「説明」は必ず入力してください。各項目にはプレースホルダを指定することもできます。

基本設定

メール送信に関する基本設定です。

設定項目	必須	説明	入力例
ID	○	図に設定したメールテンプレート項目の値を指定します。この値がキーとなり、図の値と紐付けられます。	
適用モデル区分	○	通常は〝ユーザー〟を選択します。	ユーザー
適用モデル		このメール送信を行うモデルを指定します。	顧客

説明	○	このメールテンプレートの目的や性質（送信タイミング）、宛先等の情報を明記します。	（自由文章）
件名	○	送信するメールの題名を記述します。文字列内にプレースホルダを含めることもできます。	${customer_p.name}様へのご案内
差出人	○	送信元のメールアドレスを指定します。文字列内にプレースホルダを含めることもできます。	info ${customer_p.createusermail}
宛先人	○	送信先のメールアドレスを指定します。文字列内にプレースホルダを含めることもできます。	sales ${customer_p.email}
宛先人に個別送信		後述する「宛先人を複数指定した場合」をお読みください。	（未設定時は個別送信しない）
Cc		カーボンコピーのメールアドレスを指定します。文字列内にプレースホルダを含めることもできます。	-
Bcc		ブラインドカーボンコピーのメールアドレスを指定します。文字列内にプレースホルダを含めることもできます。	-
添付ファイル		プレースホルダを用いて添付ファイルに該当するモデル項目名を指定します。この項目に登録されたファイルを添付ファイルとして扱います。	${customer_p.filename}
最大添付ファイル数		繰り返しコンテナ内の項目を添付ファイルとして利用すると複数の添付ファイルを指定できますが、この値を設定することで最大添付ファイル数の制限をかけることができます。	10
本文形式		テキストメール,HTMLメールといった指定を行うことができます。この指定方法に応じて、具体的に入力できる本文が選択されます。メール本文は、本指定を行うと下部のテキストエリアに適切な入力欄が有効になりますの	テキストのみ

		で、そこに記載してください。（本文中にプレースホルダを含めることができます。）	

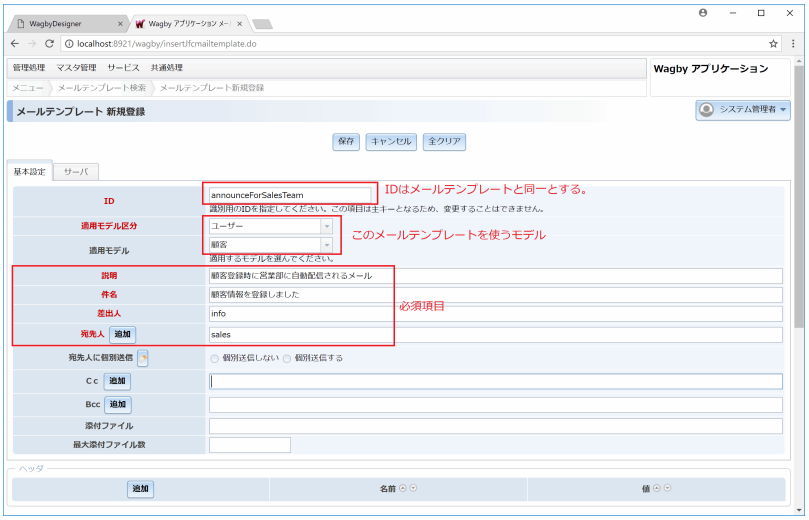


図 メールテンプレートの設定画面(1)

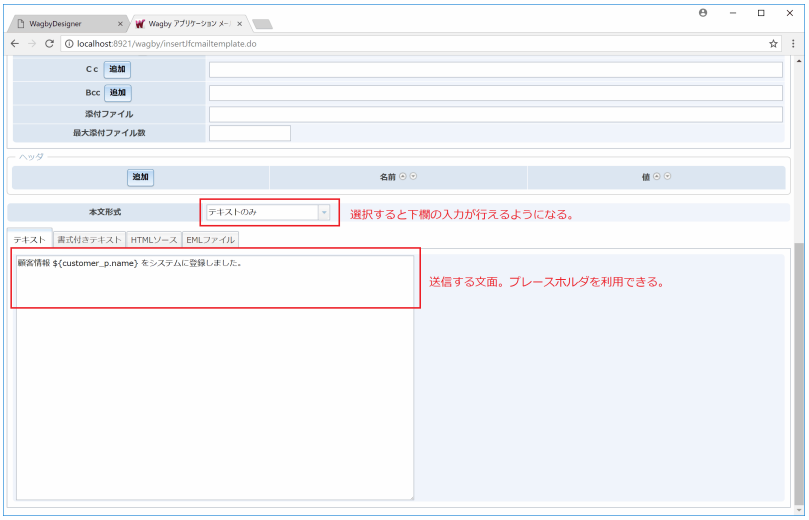


図 メールテンプレートの設定画面(2)

サーバ > SMTP

送信を行う（SMTP）サーバに関する設定です。GmailやOffice365などのサンプルは「メールサーバのサンプル」をお読みください。

設定項目	説明	入力例
メールサーバ	メールサーバを指定します。省略時は「環境>メール>メール設定(共通)>メールホスト名」の値を用います。メールサーバ名の後に「:(コロン)」を付加し、任意のポート番号を指定することもできます。	mailserver.YOURCOMPANYNAME.co.jpmailserv
プロトコル	メールサーバが対応するプロトコルを選択します。SMTP, SMTP over SSL, SMTP STARTTLS が用意されて	SMTP

	います。どのプロトコルを利用するかは貴社メールサーバ管理者へお問い合わせください。	
SMTP認証利用	メールサーバがSMTP認証を利用するかどうかを選択します。未選択時は「利用しない」となります。詳細は貴社メールサーバ管理者へお問い合わせください。	（未選択）
認証アカウント	SMTP認証を利用する場合に入力します。	
認証パスワード	SMTP認証を利用する場合に入力します。	

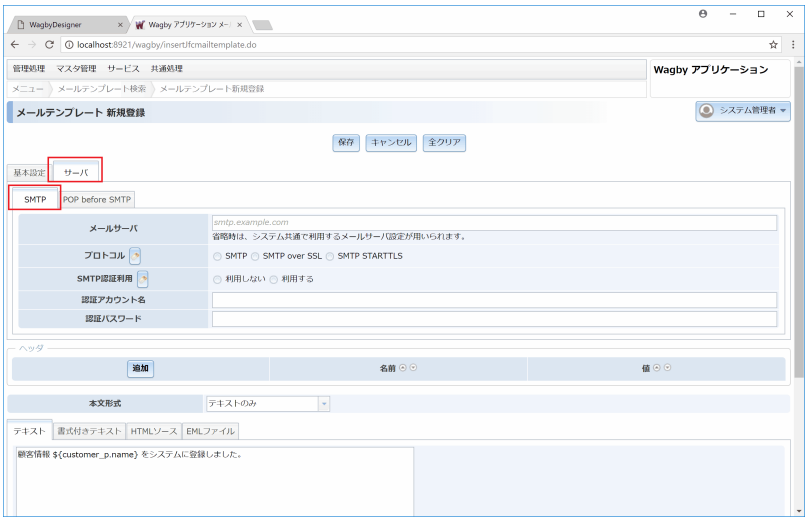


図 メールテンプレートの設定画面(3)

サーバ > POP before SMTP

POP before SMTP 認証を行う場合に設定します。（この認証方式を使わない場合、本設定欄は空白としてください。）

設定項目	説明	入力例
認証サーバ	POP before SMTP 認証に用いる POP3 サーバのホスト名を指定します。ここにホスト名が記載された場合、POP before SMTP 認証を使うようになります。サーバ名のあとにコロン区切りでポート番号を指定することもできます。	pop.YOURCOMPANYNAME.co.jp
プロトコル	"POP3" または "POP3 over SSL" のいずれかを指定します。省略時は "POP3" です。	（未選択）
APOP認証	APOPを利用するかどうかを指定します。省略時は "利用しない" です。	（未選択）
認証アカウント	POP3サーバの認証を行うアカウントを指定します。	
認証パスワード	POP3サーバの認証を行うアカウントのパスワードを指定します。	

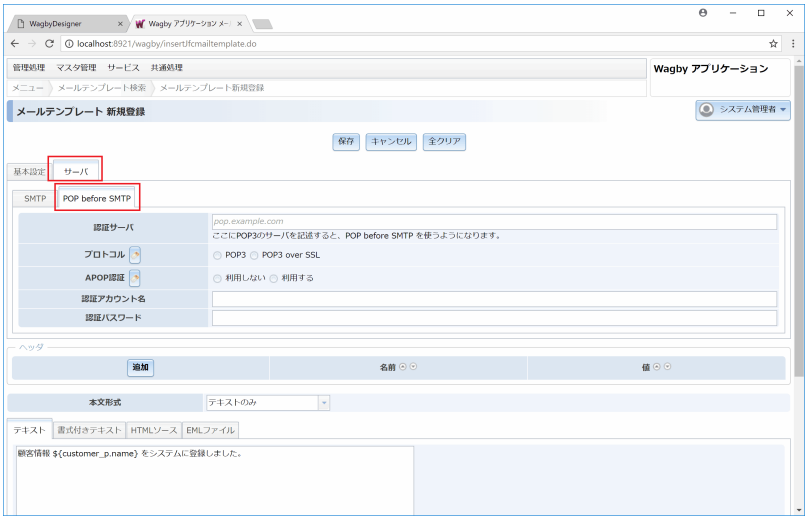


図 メールテンプレートの設定画面(4)

プレースホルダ

メールテンプレートでは、メールサーバ、件名、差出人、宛先人、本文へ特別な置き換え文字表記（プレースホルダ）を指定することで、対象となるモデルの値を埋め込むことができるようになっています。

プレースホルダの書式は「帳票>プレースホルダ」に準じます。

権限設定との関係

項目に隠し権限を設定している場合、権限がなければプレースホルダも空文字を返します。

繰り返し項目を用いる

繰り返し項目を「宛先人」「Cc」「Bcc」に指定することができます。複数人にメールを送信することができます。書式は次のとおりです。

```
${モデル名_p.項目名.#}
```

繰り返しコンテナの項目を用いる

「宛先人」「Cc」「Bcc」への適用

繰り返しコンテナの項目を「宛先人」「Cc」「Bcc」に指定することができます。複数人にメールを送信することができます。書式は次のとおりです。

```
${モデル名_p.繰り返しコンテナ名.#.メールアドレス項目名}
```

繰り返し項目、繰り返しコンテナ項目のサンプルを示します。

例	出力結果（宛先）
${customer_p.receiver.#}	taro@jasminesoft.co.jp , jiro@jasminesoft2.co.jp , hanako@jasminesoft3.co.jp
${customer_p.report.#.email}	taro@jasminesoft.co.jp, jiro@jasminesoft2.co.jp, hanako@jasminesoft3.co.jp
${customer_p.report.#.name} < ${customer_p.report.#.email}>	ジャスミン太郎 <taro@jasminesoft.co.jp>, ジャスミン次郎 <jiro@jasminesoft2.co.jp>, ジャスミン花子 <hanako@jasminesoft3.co.jp>

本文への適用

繰り返しコンテナの項目を「本文」に指定することができます。書式は次のとおりです。

```
${モデル名_p.繰り返しコンテナ名.#.項目名}
```

繰り返し項目、繰り返しコンテナ項目のサンプルを示します。繰り返される部分を1行で(改行なしで)記述してください。

```
変更履歴
No 日付時間 ユーザID 内容
${customer_p.changelog.#.cid} ${customer_p.changelog.#.cdate} ${customer_p.changelog.#.cuserid}
 ${customer_p.changelog.#.ccontent}
```

メールテンプレートの本文をHTML形式とすると、HTMLのテーブルとして出力することができます。上記と同様に、繰り返す部分は1行で記述してください。

```
<p>変更履歴</p>
<table border="1" width="100%">
<tr><th>No</th><th>日付時間</th><th>ユーザID</th><th>内容</th></tr>
<tr><td>${customer_p.changelog.#.cid}</td><td>${customer_p.changelog.#.cdate}</
td><td>${customer_p.changelog.#.cuserid}</td><td><textarea rows="5" cols="80">
${customer_p.changelog.#.ccontent}</textarea></td></tr>
</table>
```

プレースホルダが利用できない場合

一覧表示、ダウンロード画面では宛先や本文にプレースホルダを指定することはできません。

例（1）処理完了時の自動送信

顧客の新規登録を行います。

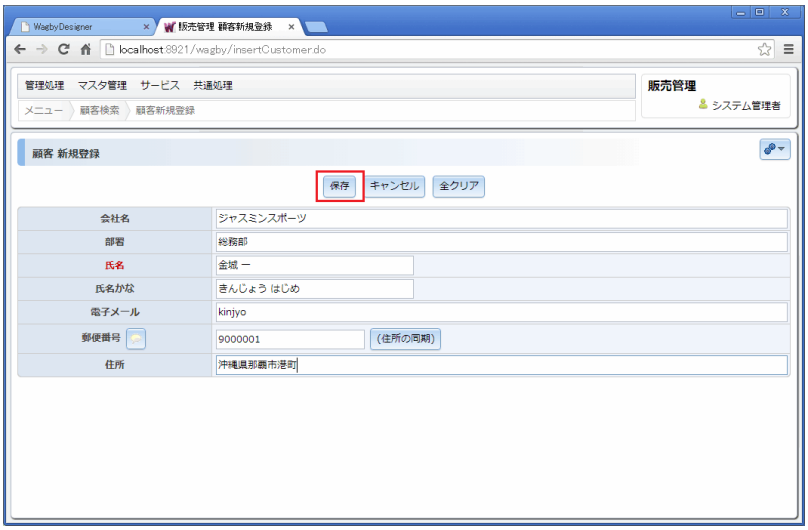


図 顧客の新規登録画面

新規登録が正常に行われたタイミングで、メールが自動送信されます。

図 新規登録完了とメールの自動送信

例（2）メール送信ボタンによる手動送信

メール送信設定で、送信のタイミングを「メール送信ボタン」とすることができます。主に詳細表示画面に用意します。

図 メール送信ボタン押下のメール送信設定

詳細画面では図のようにメール送信ボタンが用意されます。

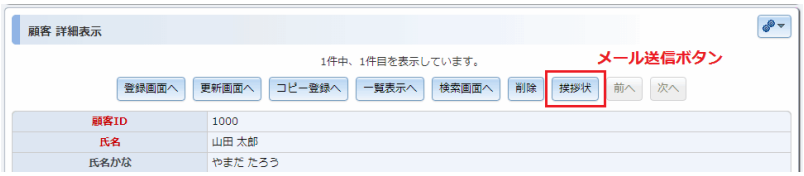


図 メール送信ボタン

図で「メール送信後にメッセージを表示する」を有効にすると、メール送信後に図のようにメッセージが表示されます。

図 メール送信成功のメッセージ

宛先人の詳細

宛先人を複数指定した場合

宛先人を複数指定した場合、受信したメールでは「manager01@xxx.yyy.zzz,manager02@xxx.yyy.zzz,manager03@xxx.yyy.zzz」などと、すべての宛先が見えてしまいます。他の宛先を見えないように送信する（それぞれの宛先に個別に送信する）場合には「宛先人に個別送信」の設定を有効にします。

「宛先人に個別送信」を指定し、かつ宛先人にプレースホルダを適用した場合、データベースから宛先情報を読み込みます。 ここで宛先がゼロの場合、メール送信は行われません。必ず送信できるアドレスを一つ加えることで、この動作を抑制することができます。

宛先人に日本語を含める

「差出人」「宛先人」「CC」「Bcc」には日本語を含むこともできます。　（例）「Wagby応援隊 supporter@wagby.com」など

プレースホルダで指定することもできます。

```
${staff_p.lastupdateuser} <${staff_p.lastupdateusermail}>
```

送信ルール

宛先指定時の送信ルールは以下の通りです。

- To は必須です。Cc、Bcc は任意です。
- To、Cc、Bcc いずれかに不正なメールアドレス記述があれば、送信されません。
- To、Cc、Bcc に重複があった場合、重複分は除かれます。同じメールアドレスに複数のメールが送信されることはありません。

ファイルを添付する

ファイル名型の項目を添付することができます。通常のファイル名型の項目だけでなく、繰り返しコンテナ内のファイル名型にも対応しています。

具体的にはプレースホルダ形式で指定します。書式は次のとおりです。

```
${モデル名_p.項目名}
${モデル名_p.コンテナ名.#.項目名}
```

例を示します。

```
${report_p.attachfile}
${report_p.container.#.attachfile}
```

最大添付ファイル数

「最大添付ファイル数」を指定することができます。添付ファイル数がこの数値を超えるとエラーとなり、メールは送信されません。

（「最大添付ファイル数」が空欄の場合、添付ファイル数のチェックは行われません。）

メールヘッダの設定

任意のメールヘッダを設定することができます。

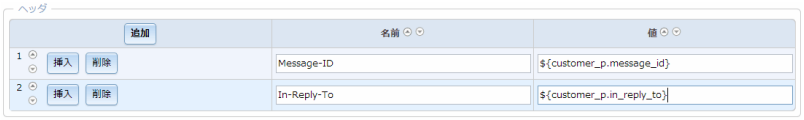


図 メールヘッダの設定

メールヘッダを使って、メールソフトのスレッド表示を実現する

"メールヘッダを使ってメールのスレッド表示を実現する"をお読みください。

メールサーバとプロトコルの詳細

メールサーバ

Wagby 本体はメールサーバの機能をもっていません。Wagby のメール送信は、ブラウザが動作しているクライアントではなく、Wagby アプリケーションが動作しているサーバから、設定時に指定したメールサーバを使って送信されます。メールが送られないという場合、Wagby アプリケーションを稼働しているサーバから指定したメールサーバへ正しく通信ができるかを確認してください。

プロトコル

Wagby が対応しているメール送信のプロトコルは「SMTP」および「SMTPS (SMTP over SSL)」です。SMTP 認証にも対応しています。

メールサーバに Gmail を用いる場合、プロトコルにSMTPSを指定し、SMTP認証の設定を行う必要があります。

SMTP認証は、"LOGIN" と "PLAIN" 方式に対応しています。("CRAM-MD5", "DIGEST-MD5", "NTLM" といった方式には対応していません。)

ボタン名と表示条件

メール送信ボタンの「ボタン名」ならびに「表示条件」を指定できます。

設定の詳細は「画面機能>ボタン名と表示条件のカスタマイズ」を参考にしてください。

図 メール送信ボタンと表示条件

メール送信（実行時にメールテンプレートを指定する）

定義方法

このページを読む前に、「メール送信（メールテンプレートを固定する）」に目を通しておいてください。

モデルの変更

メールテンプレートを動的に変更するために、モデル側に「メールテンプレート」を保持するための項目を追加します。

図に示すように、「メールテンプレート」項目を追加します。参照先モデルはWagbyが標準で提供するメールテンプレートモデル（jfcmailtemplate）です。参照先モデル項目は「説明」とします。

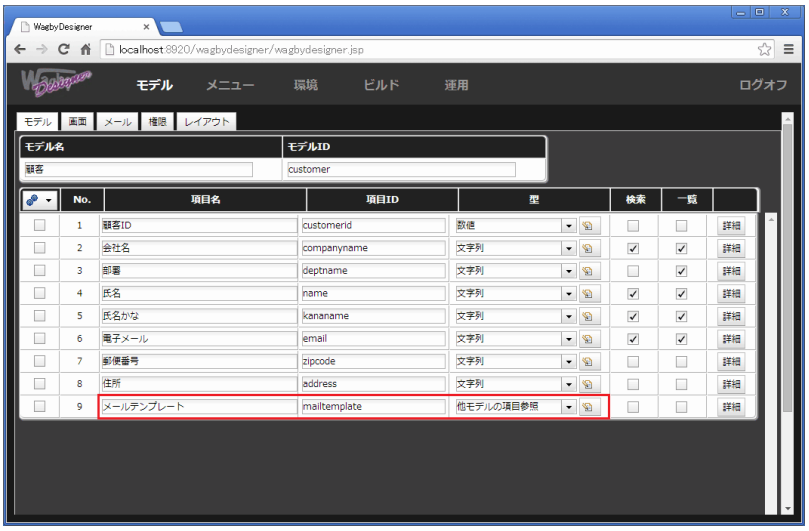


図 メールテンプレートを参照する項目を追加する

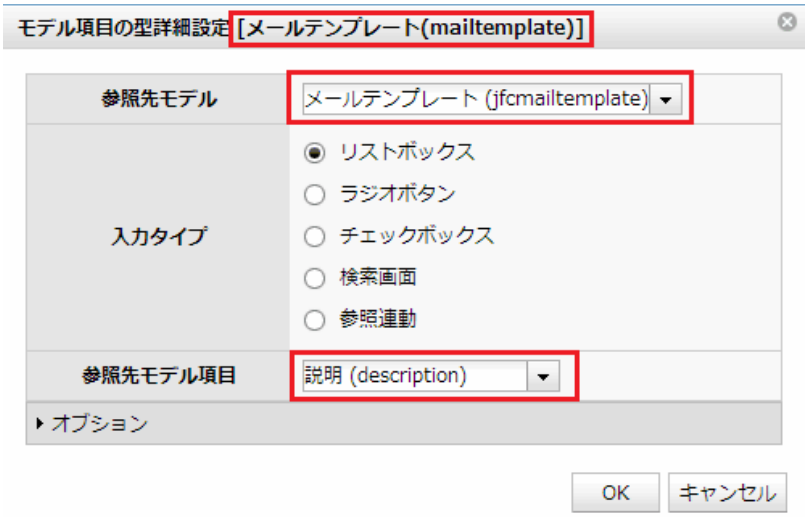


図 メールテンプレート項目の参照設定

メールの設定

ここでは、詳細表示画面に「メール送信」ボタンを用意してみます。送信のタイミングを「メール送信ボタン」とします。ボタン名を「顧客向けメール」とします。

ポイントは、メールテンプレート項目欄に（固定値ではなく、図で追加した）項目名を指定することです。このとき、"${" と "}" で項目名を囲みます。

```
${mailtemplate}
```

図 詳細表示時のメール設定

メールテンプレートの設定

ビルドしたアプリケーションにシステム管理者でログオンします。図に設定したメールテンプレートの詳細な設定を行います。メールテンプレートの新規登録を行います。

「ID」は他のメールテンプレートと重複しない値を設定してください。「説明」はメールテンプレートの選択肢として使います。わかりやすい内容を設定します。

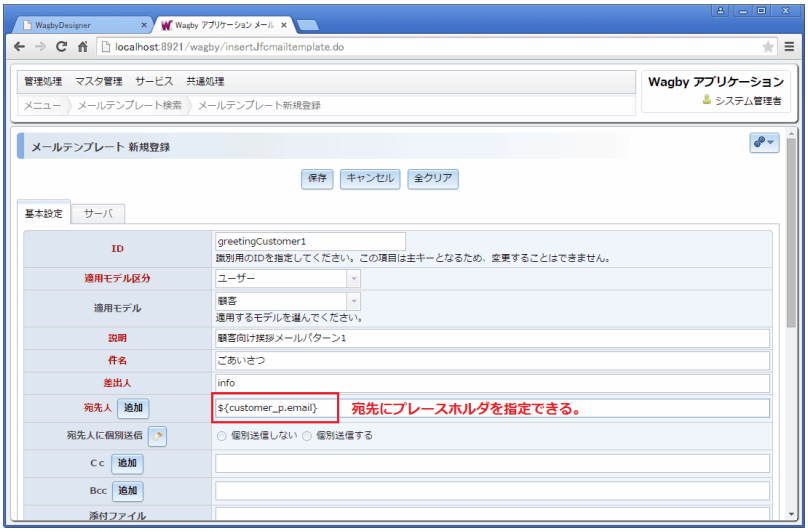


図 メールテンプレートの設定画面(1)

件名、差出人、宛先人、メール本文（テキストエリアの欄）には、プレースホルダを設定できます。

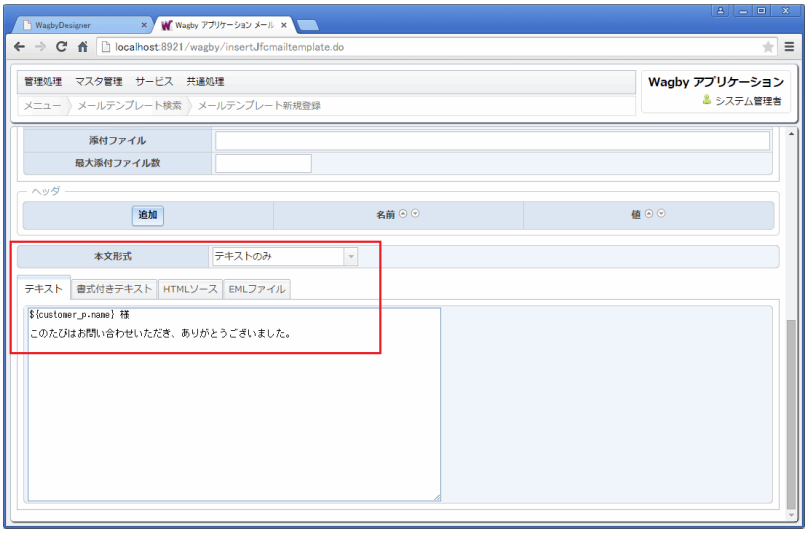


図 メールテンプレートの設定画面(2)

例

メールを送信する前に、どのメールテンプレートを用いるかを設定します。 選択肢は、図で設定した「説明」項目を用いています。

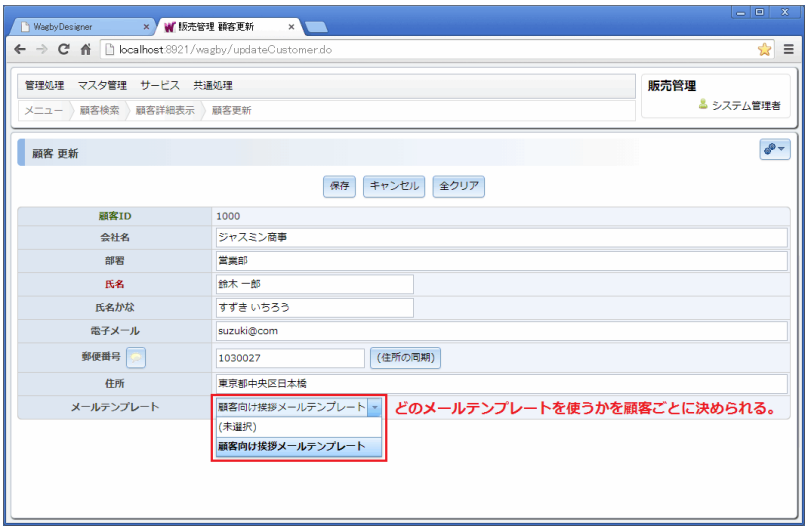


図 メールテンプレートを選択する

詳細表示画面には「メール送信」ボタンが用意されています。 ボタン押下により、メールが送信されます。

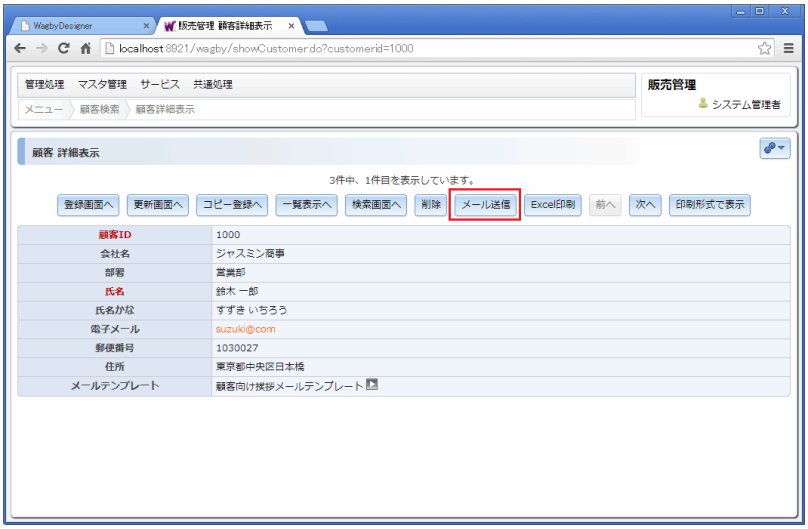


図 詳細表示画面に用意された「メール送信」ボタン

メール送信（ジョブによる時間指定）

ジョブの設定

このページを読む前に、「メール送信（メールテンプレートを固定する）」に目を通しておいてください。

モデルの「メール>送信（ジョブ）」タブを開きます。メール送信に関するジョブを複数、設定することができます。 1つのジョブの設定は、1つの「メールテンプレート」と紐づけます。

図 メール送信ジョブの定義

項目	説明	入力例
ジョブ名	このアプリケーションシステム全体で、重複しないジョブ名（英語）を指定します。メール送信の場合は "SendMail" +（モデル名）+ "Job" とするとよいでしょう。	SendMailLeaveJob
説明	自由に記述してください。	-
起動条件	メール送信を行う条件を指定します。この欄が空白の場合は、全データにメールを送信します。対象モデルに関する項目は "${" +（項目名）+ "}" で表現し	${item1} == 0※数値項目の場合 EXACT(${item1},"MOJI")※文字列型項目の場合

		ます。関数を使うこともできます。	
成功時のアクション		メール送信が成功したあとに行う処理を指定します。対象モデルに関する項目は"${" +（項目名）+ "}" で表現します。関数を使うこともできます。「=」を使った代入文を指定することもできます。	
失敗時のアクション		メール送信が失敗したあとに行う条件を指定します。対象モデルに関する項目は"${" +（項目名）+ "}" で表現します。関数を使うこともできます。「=」を使った代入文を指定することもできます。	
メールテンプレート項目		後述する「メールテンプレート」の識別子(ID)になります。重複しない名前を英数字の組み合わせで決めてください。この値が空白の場合、「ジョブ名」がメールテンプレートIDとして利用されます。	SendMailLeaveJobTemplate
メール送信制御	最大同時送信数	メールサーバの負荷を減らすために一度に送るメールの数を制限します。 送信回数が指定した数を超えた場合、「待ち時間」で指定した時間だけ処理を中断した後に、送信処理を再開します。 入力がない場合は無制限となります。 一度に大量のメールを送信すると、メールサーバがスパムメール（迷惑メール）と判断することがあります。適切な値を調整してください。	10
	待ち時間（秒）	送信回数が指定した最大同時送信数を超えた場合、指定した時間だけ処理を中断した後に送信処理を再開します。 入力がない場合は 0秒となります。 メール送信を連続して行うと、メールサーバがスパムメール（迷惑メール）と判断すること	30

		があります。適切な待ち時間を設定してください。	

メールテンプレートの設定

ビルドしたアプリケーションにシステム管理者でログオンします。図に設定したメールテンプレートの詳細な設定を行います。メールテンプレートの新規登録を行います。

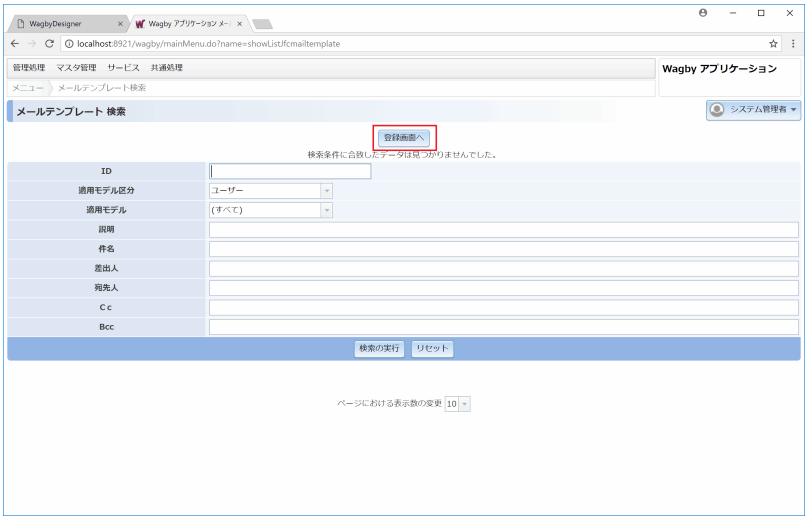


図 メールテンプレートの新規登録画面へ

図 メールテンプレートの新規登録

メールテンプレートの詳細な定義方法は「メール送信>定義方法>メールテンプレートの設定」をお読み下さい。

「ID」は、図の「メールテンプレート項目」に記載したものと同一としてください。

ジョブスケジュールの設定

ジョブ専用アカウントjobadminの設定

「サポート>管理者ガイド>準備：ジョブ実行アカウントを利用可能にする」の手順に従って jobadmin アカウントを有効にします。

メール送信の対象モデルの操作権限（プリンシパル）をjobadminアカウントに与える必要があります。図の例では「一般ユーザ」プリンシパルを付与した例です。

図 jobadminアカウントの設定

スケジュールの設定

ジョブスケジュールの新規登録を行います。

1. 管理者権限をもったアカウントでログオンします。
2. メニューから「ジョブ管理」「ジョブスケジュール検索」画面を開きます。
3. 「登録画面へ」ボタンを押下し、ジョブスケジュールの新規登録処理を行います。
4. ジョブ名から図で設定したジョブ名を選択します。※ ジョブ名が選択できない場合、図下欄の注意書きをお読みください。
5. スケジュールを指定します。
6. 「有効」フラグをチェックします。
7. このジョブを実行するアカウント "jobadmin" を入力します。
8. ジョブスケジュールを登録します。指定した時間になると、コマンドが実行されます。

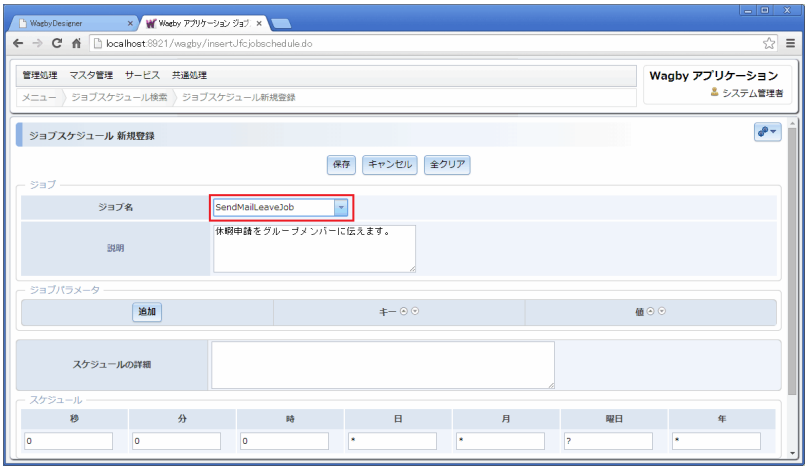


図 スケジュールの新規登録

ジョブ名が選択できない場合

上の手順4でジョブ名が選択できない場合、ビルド後に旧データをすべてインポートした可能性があります。追加したジョブは「ジョブマスタ」モデルとして用意されており、ビルド後の "data_init" フォルダにエクスポートデータが記載されていますが、旧データには存在しません。そのため、旧データを取り込んだタイミングで、ジョブ情報が古いデータとなっています。

この対応として、旧データインポート後、あらためてインポートフォルダ名に "data_init" フォルダを指定し、「システムによって提供されるモデル一覧 > ジョブマスタ」をインポートしてください。これによって（追加した）ジョブを認識できるようになります。

ジョブの実行を停止する

ジョブを一時的に無効にすることができます。

1. 管理者権限をもったアカウントでログオンします。
2. メニューから「ジョブ」「ジョブスケジュール検索」画面を開きます。
3. 該当する「ジョブスケジュール」を検索し、更新画面を開きます。
4. 「有効」フラグについているチェックをはずし、保存します。

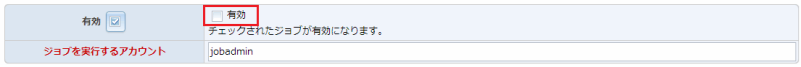


図 ジョブを無効にする

実行結果をメールで送信する

ジョブの実行結果をメールで送信することができます。（すでにメール送信機能で「メールテンプレート」を利用しているため、ここではメールテンプレートを利用せず、直接ジョブパラメータを加えてください。）

多重実行の抑制

ジョブの多重実行を行うことはできません。

あるジョブを 1 分毎に起動させるようにした場合、1 分後にこのジョブの実行が終了していなければ、新しいジョブは実行されません。

メール受信（ジョブによる時間指定）

設定

例 注文モデルを用意する

ここでは、受信メールを保存するモデル「注文」を用意します。図で赤枠の部分は、受信したメールを取り込むために用意したものです。

「差出人」「宛先人」「Cc登録先」は「繰り返しとして利用する」設定を行います。

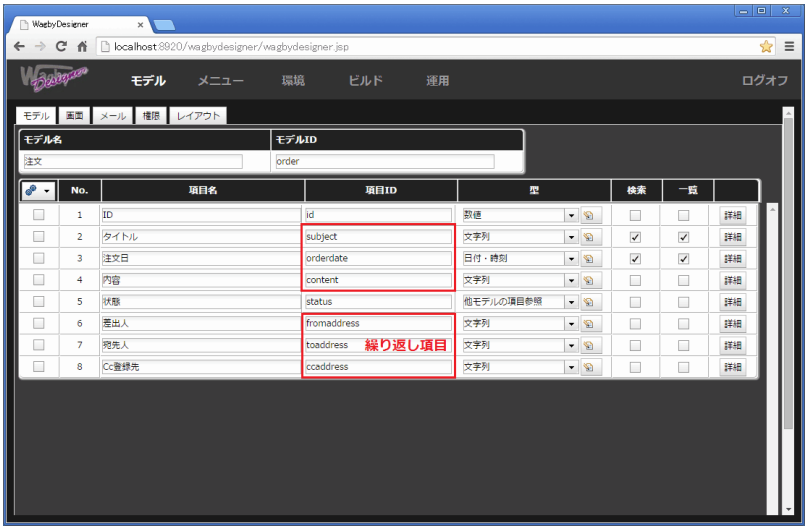


図 注文モデル

メール受信の設定

メニューから「メール ＞ 受信」タブを開きます。設定の詳細は次のとおりです。

図 メール受信設定欄

ジョブ

項目	説明	入力例
ジョブ名	このアプリケーションシステム全体で、重複しないジョブ名（英語）を指定します。	InsertOrderFromMailJob
説明	自由に記述してください。	-
起動条件	取り込んだメールをモデルに変換後、取り込むための条件を指定することができます。標準は空白（条件なし）です。対象モデルに関する項目は"${"＋（項目名）＋"}"で表現します。関数を使うこともできます。 （ただしこの条件に合致しても、入力チェックでエラーになった場合は、やはり取り込まれません。すなわち、この起動条件および入力チェックともに正の場合に、データベースに新規登録されます。）	メールのサブジェクトをsubject項目にセットした場合、サブジェクトが"Wagby"で始まるものだけを登録する：FIND("Wagby", ${subject})==1
成功時のアクション	メール受信が成功したあとに行う処理を指定します。対象モデルに関する項目は"${"＋（項目名）＋"}"で表現します。関数を使うこともできます。また、「=」を使った代入文を指定することもできます。 この指定を行った場合、指定された処理を実行したのちにデータベースへの書き込みを行います。そのため、何らかの理由でデータベースへの書き込みが失敗した場合、取り込まれたメールおよび成功時のアクションはいずれも無効となります。	

許容する差出人アドレス	差出人アドレスを指定します。完全一致で指定します。正規表現式によるパターンマッチを指定することができます。	.*@jasminesoft.co.jpと記載することで、ドメイン名がjasminesoft.co.jp のメールアドレスだけを取り込むことができるようになります。
添付ファイルの登録先	受信メールに添付されているファイルを登録する（このモデルの）項目名を指定します。この項目はファイル名型としてください。添付ファイルが複数あった場合は繰り返しコンテナとして定義します。	例えば繰り返しコンテナattachments 内にファイル型の項目 filename を用意していた場合、この欄にはattachments/filenameと記述します。

メールサーバ

項目	説明	入力例(Gmail利用の場合)
メールサーバ	メールサーバ名を指定します。メールサーバ名の後に「:(コロン)」を付加し、任意のポート番号を指定することもできます。	pop.gmail.com:995
アカウント名	受信用のメールアカウントを指定します。	YOURNAME@gmail.com
パスワード	受信用のパスワードを指定します。	
プロトコル	POP3, IMAP, POP3 over SSL, IMAP over SSL のいずれかを選択できます。未選択時はPOP3 となります。	POP3 over SSL
APOP認証	APOP認証を利用する場合はチェック欄を有効にします。	(Gmailではチェックしない)

受信先の詳細

項目	説明	補足
件名	受信メールの件名（Subject）の値を転記する（このモデルの）項目名を指定します。この値が空白の場合、件名の情報は無視されます。	文字列型としてください。
差出人	受信メールの差出人（From）の値を転記する（このモデルの）項目名を指定します。この値が空白の場合、差出人の情報は無視されます。	文字列型（メールアドレス属性）で、かつ繰り返し指定を行ってください。[388]
宛先人	受信メールの宛先人（To）の値を転記する（このモデルの）項目名を指定します。この値が空白の場合、宛先人の情報は無視されます。	文字列型（メールアドレス属性）で、かつ繰り返し指定を行ってください。
CC	受信メールのカーボンコピー(CC) の値を転記する（このモデルの）項目名を指定します。この値が空白の場合、CCの情報は無視されます。	文字列型（メールアドレス属性）で、かつ繰り返し指定を行ってください。
送信日付	受信メールの送信日付の値を転記する（このモデルの）項目名を指定します。この値が	日付型/日付時刻型/時刻型としてください。

[388] 差出人は1人の場合が多いですが、インターネットメールの仕様では複数人を扱えるようになっています。Wagbyもこの仕様に準拠しています。

	空白の場合、送信日付の情報は無視されます。	
本文	受信メールの本文（BODY）の値を転記する（このモデルの）項目名を指定します。この値が空白の場合、本文の情報は無視されます。	文字列型としてください。大量の文字列を記録する場合は、テキストエリアの利用を推奨します。

本文の自動解析機能

項目	説明	初期値
有効にする	この欄をチェックした場合、受信メールの本文を自動解析し、モデルの各項目に値を設定します。（後述）	
値部分の開始区切り文字	本文の自動解析機能を有効にしたとき、値部分の開始区切り文字を指定できます。	未指定時は「半角スペース」となります。
値部分の終了区切り文字	本文の自動解析機能を有効にしたとき、値部分の終了区切り文字を指定できます。	未指定時は「改行」となります。
対象項目	本文の自動解析機能を有効にしたとき、解析対象項目を指定します。	未指定時は全項目が対象となります。
区切り文字の扱い	本文の自動解析機能を有効にしたとき、開始区切り文字と終了区切り文字の全角・半角文字区分を無視するかどうかを指定します。	未指定時は全角文字と半角文字を区分します。

仕様・制約

- 本機能はジョブが起動している間、指定した間隔でメールを受信し続けます。それらのメールは「受信済み（既読）」となります。これを未受信の状態に戻すことはできません。また、条件によって受信しない、ということもできません。
- 項目の値を参照した、条件付き権限管理を設定することはできません。権限管理の式はストアモデルの値を使いますが、メール受信時はストアモデルが空のためです。画面からの処理で条件付き権限管理を用いる必要がある場合、メール受信用のサブモデルを定義して、併用してください。

トラブルシューティング

メール受信機能が動作しないという場合、以下の点をご確認ください。

ジョブが起動されたかどうかを調べる

管理処理メニューのシステムログ閲覧画面で、次の検索を行うことで確認できます。

```
[検索条件]
タイプ：INFO
キーワード：JFC-00902
```

N 分間隔でメール受信を行うようにしているが起動しない

ジョブスケジュール設定で「時」の部分に数値（例：0など）が含まれていませんか。この場合、0時になった場合のみ起動するという意味になります。「時」の部分を「*」とすることで解決します。

メール受信が失敗することがある

(1)　メール受信の間隔が短すぎませんか。前のジョブが終了していないにも関わらず次のジョブが起動してしまうと、受信に失敗します。または、頻繁なメール受信はメールサーバ側でセキュリティ上の攻撃と誤認識される可能性もあります。

(2)　　複数のメール受信ジョブが登録されていませんか。同じメールボックスを同時に種類の違う複数のメール受信ジョブがアクセスした場合に、 後から起動したジョブが受信に失敗します。

データベース書き込みエラーが発生する

一般に、メール本文は長大なテキスト文字列となります。本文を取り込む場合、データベースの標準的な「文字列型」では、文字列の格納領域の制限により、保存エラーとなることがあります。これは運用時のログファイル（logs/system.log）にエラーとして記録されます。

この場合、ご利用となるデータベースが提供する「長大なテキスト文字列を保存できる型」を別途、指定するとよいでしょう。例えばデータベースに PostgreSQL をご利用の場合、"text" と指定することで 1GB までの文字列を保存できるようになります。

または Wagby の文字列処理関数を用いて、指定した文字列長でカットしたものを保存するというアイデアもあります。

毎回、同じメール読み込みでエラーが発生する

何らかの理由でエラーになったメールは、受信ジョブの実行の都度、毎回受信エラーになってしまいます。　受信エラーとなるメールが発生した場合は、別のメールソフト等で該当するメールを受信してメールボックスから削除してください。

ただし毎回受信エラーとなるメールの原因は、モデルの設計にあります。ほとんどの場合、定義を見直して頂くことで解決できます。例えば次のようなものです。

- 取り込むデータと項目の型の不一致
- ユニーク制約がついている項目に重複する値をセット
- 必須項目にデータがセットされていない

添付ファイルの処理が失敗することがある

添付ファイルのファイル名が日本語文字を含む場合に、このような現象が発生することがあります。

例えばメールサーバが日本語を含むファイル名に対して何らかの処理を行い、ファイル名に　"?"　など、Windows OS が扱うことができない文字を埋め込んでしまうといったケースです。Wagbyは、そのファイル名を使ってファイルを保存しようとしますが、Windows OS がこれをエラーとして扱うため、メール受信処理に失敗します。

このような場合は（メールに添付された）ファイル名を変更するか、または「文字列-ファイル　>　保存ファイル名の扱い」で、"オリジナルファイル名をそのまま使う" を選択しないようにします。

詳細なログの取得

ジョブパラメータ JavaMailDebug を true とすることで、受信状況の詳細なログを取得できます。ファイル名は次のとおりです。

```
javamail-Insert<モデルID>FromMailProcessor-yyyyMMdd_HHmmss.log
```

例えばモデル　Model1　のメール受信ジョブ実行時に　"javamail-InsertModel1FromMailProcessor-20150624_194742.log"というログファイルが用意されます。ログの内容は次のようになります。

```
DEBUG: getProvider() returning
 javax.mail.Provider[STORE,pop3s,com.sun.mail.pop3.POP3SSLStore,Sun Microsystems, Inc]
DEBUG POP3: connecting to host "YOURMAILSERVER", port 995, isSSL true
S: +OK Dovecot ready. <73b1.e.5583d560.2eAI44Sem3e4iVvBMeQCaQ==@XXX.jasminesoft.co.jp>
C: USER [※ユーザ名]
S: +OK
C: PASS [※パスワード]
S: +OK Logged in.
C: STAT
S: +OK 2836 266374220
```

```
C: NOOP
S: +OK
C: TOP 1 0
S: +OK
Return-Path: <yamada@jasminesoft.co.jp>
X-Original-To: who@jasminesoft.co.jp
Delivered-To: who@jasminesoft.co.jp
Received: from XXX.jasminesoft.co.jp (localhost [127.0.0.1])
....
Sender: suzuki@jasminesoft.co.jp
Errors-To: adm@jasminesoft.co.jp


.
C: QUIT
S: +OK Logging out.
```

[※ユーザ名]や[※パスワード]の部分にPOP認証の際のユーザ名、パスワードが入ります。また、すべての受信メールの内容が出力されます。このように個人情報が多く含まれますので、取扱いには注意してください。開発途中のデバッグ用途以外では、このログの出力を行わないようにしてください。

ジョブスケジュールの設定

ジョブ専用アカウントjobadminの設定

「サポート>管理者ガイド>準備：ジョブ実行アカウントを利用可能にする」の手順に従って jobadmin アカウントを有効にします。

メール送信の対象モデルの操作権限（プリンシパル）をjobadminアカウントに与える必要があります。図の例では「一般ユーザ」プリンシパルを付与した例です。

図 jobadminアカウントの設定

スケジュールの設定

ジョブスケジュールの新規登録を行います。

1. 管理者権限をもったアカウントでログオンします。
2. メニューから「ジョブ管理」「ジョブスケジュール検索」画面を開きます。
3. 「登録画面へ」ボタンを押下し、ジョブスケジュールの新規登録処理を行います。
4. ジョブ名から図で設定したジョブ名を選択します。※ ジョブ名が選択できない場合、図下欄の注意書きをお読みください。
5. スケジュールを指定します。
6. 「有効」フラグをチェックします。
7. このジョブを実行するアカウント ″jobadmin″ を入力します。
8. ジョブスケジュールを登録します。指定した時間になると、コマンドが実行されます。

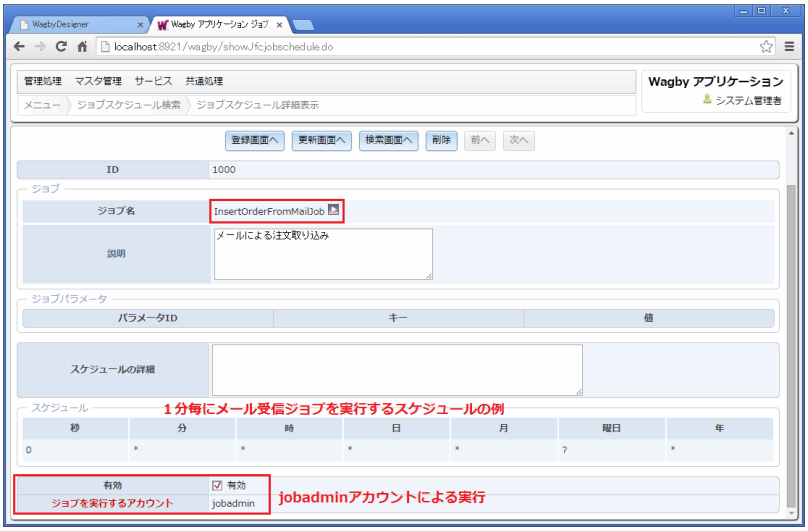


図 スケジュールの新規登録

ジョブ名が選択できない場合

上の手順4でジョブ名が選択できない場合、ビルド後に旧データをすべてインポートした可能性があります。追加したジョブは「ジョブマスタ」モデルとして用意されており、ビルド後の "data_init" フォルダにエクスポートデータが記載されていますが、旧データには存在しません。そのため、旧データを取り込んだタイミングで、ジョブ情報が古いデータとなっています。

この対応として、旧データインポート後、あらためてインポートフォルダ名に "data_init" フォルダを指定し、「システムによって提供されるモデル一覧 > ジョブマスタ」をインポートしてください。これによって（追加した）ジョブを認識できるようになります。

メール受信の間隔

メール受信機能は、一定間隔を空けて実行するようにしてください。短い時間に連続して受信機能を実行すると、メールサーバおよびWagbyが稼動するアプリケーションサーバの負荷が増大します。メールサーバによっては、セキュリティ上の攻撃と解釈し、このアカウントからの受信を停止するといった可能性も生じます。

例

ジョブによってメールが取り込まれた例を図に示します。

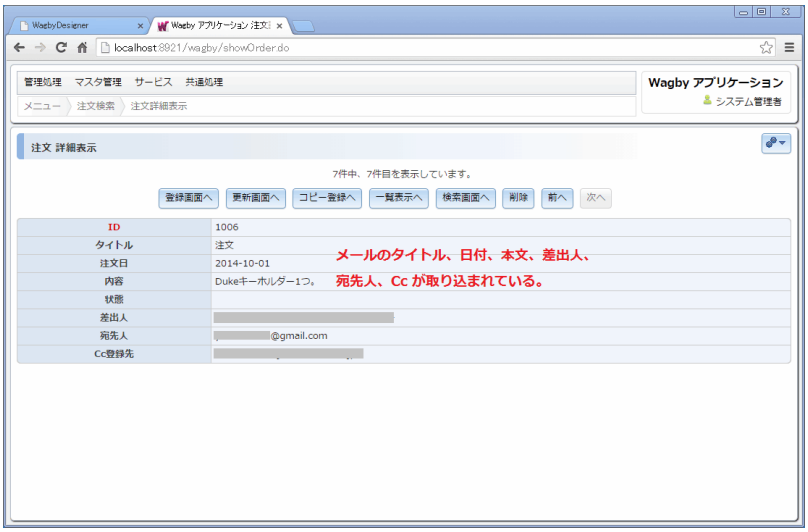


図 メール受信

ログファイル（logs/system.log）には次のような記録が残ります。

```
2014-10-01 14:33:00 [INFO jp.jasminesoft.jfc.JFCJobTriggerListener triggerFired]
 (LOGONUSER@localhost) JFC-00902:ジョブ InsertOrderFromMailJob を開始しました。
2014-10-01 14:33:00 [INFO jp.jasminesoft.jfc.LogonStatusManager addUserName]
 (LOGONUSER@localhost) JFC-90000:アカウント jobadmin がセッションに追加されました。現在の同時接
続数は 1 で、合計セッション数は 2 です。
2014-10-01 14:33:01 [INFO jp.jasminesoft.wagby.app.order.InsertOrderFromMailProcessor process]
 Folder opened
2014-10-01 14:33:01 [INFO jp.jasminesoft.wagby.app.order.InsertOrderFromMailProcessor process]
 Total mail count: 1
2014-10-01 14:33:01 [INFO jp.jasminesoft.wagby.app.order.InsertOrderFromMailProcessor process]
 _mcount: 1
2014-10-01 14:33:02 [INFO jp.jasminesoft.wagby.app.order.InsertOrderFromMailProcessor process]
 Message-ID: <75F5BB63-D2CC-47F1-A004-6E0982C7E865@YOURDOMAIN.co.jp>
2014-10-01 14:33:03 [INFO jp.jasminesoft.wagby.app.order.InsertOrderFromMailProcessor process]
 Folder closed
2014-10-01 14:33:03 [INFO jp.jasminesoft.jfc.job.JFCBaseJobBean executeInternal]
 (jobadmin@localhost)1 mail is processed.
2014-10-01 14:33:03 [INFO jp.jasminesoft.jfc.User logoff] (LOGONUSER@localhost) JFC-00028:アカウ
ント jobadmin のログオフを行います。
2014-10-01 14:33:03 [INFO jp.jasminesoft.jfc.LogonStatusManager removeUserName]
 (LOGONUSER@localhost) JFC-90003:アカウント jobadmin がセッションから除かれました。現在の同時接
続数は 0 で、合計セッション数は 0 です。
2014-10-01 14:33:03 [INFO jp.jasminesoft.jfc.JFCJobTriggerListener triggerComplete]
 (LOGONUSER@localhost) JFC-00900:ジョブ InsertOrderFromMailJob は正常に終了しました。
```

メール本文の自動解析

メール本文を所定のフォーマットにしておくと、アプリケーションがメール本文の内容を自動的に解析して、各項目に値をセットします。

メール本文を次のフォーマットで記述してください。

- 項目名と値は、1行に記述します。
- 項目名には「項目名(日本語)」を使用します。
- 項目名と値の間に、区切り文字として半角空白文字を置きます。(各行2つ目以降の半角空白文字は値の一部として扱われます。)

例えば「氏名」「住所」「電話」(これらはすべて「項目名」)を項目に持つモデルに対しては、メール本文を次のように記述します。

```
氏名 山田 太郎
住所 沖縄県
電話 098-000-0001
```

値の開始区切り文字と終了区切り文字を指定することもできます。例えば、開始区切り文字に「<」を、終了区切り文字に「>」を指定した場合、次のようなメール本文を解析することができます。

```
氏名<山田 太郎>
住所<沖縄県>
電話<098-000-0001>
備考<改行つきの
文章も
格納できます。>
```

本機能の詳細なルールは次のとおりです。

- 開始区切り文字および終了区切り文字は「1文字」を指定できます。
- 開始区切り文字に半角スペース以外の文字を指定すると、項目名に半角スペースが含まれていても解析できるようになります。
- 終了区切り文字に改行以外の文字を指定すると、メール本文中に空白および改行が含まれていても解析できるようになります。
- 項目名の前に余分な空白が含まれているような場合でも正しく解析処理されます。

- モデル参照項目にも対応しています。この場合はコードではなく「値」（内容部）を記述してください。
- 本文解析処理後に、自動計算および参照連動の解決処理も行われます。
- 「メール受信設定」定義で「～の登録先」が指定された場合、項目の値はメール本文解析結果より「～の登録先」指定が優先されます。例えば「件名の登録先」として "subject" 項目を指定していた場合、メール本文解析で subject 項目に値が設定された場合であっても、「件名の登録先」で指定した項目の値に上書きされます。
- 繰り返し項目および繰り返しコンテナ項目は対象外となります。

メール本文をリッチテキストエリアに格納する

HTMLメール受信時、本文を格納する項目がテキストエリア(リッチ)の場合は、改行文字を
に置き換えます。

詳細

HTMLメールはテキスト部分とHTML部分から構成されます。Wagbyのメール受信は、テキスト部分を使用します。テキスト部分の改行文字を
に置換します。

このテキスト部分は、HTML部分の改行文字と必ずしも一致しません。お使いのメールソフトによって扱いが異なります。そのため、リッチテキストエリアの改行は、HTML部分の改行と異なる位置に出現することがあります。疑問が生じた場合は、取り込んだメールの「テキスト部分」の改行文字をご確認ください。

ジョブに指定できるパラメータ

次に示すパラメータを設定することができます。

Wagby Desinger で指定した値より、ジョブパラメータの方が優先されます。運用中にメール受信者などを変更することができます。

パラメータ名	説明	記述例
mailhost	メールホスト	mail.yourdomain.com
username	メール受信を行うアカウント	yamada
password	メール受信を行うアカウントのパスワード	
protocol	プロトコル	pop3pop3s
APOP	APOP認証の使用有無	true
analyzeBody	本文の自動解析機能の有効無効	true
dataStartChar	値部分の開始区切り文字	[
dataEndChar	値部分の終了区切り文字	]
debug	デバッグモードを有効にします。 system.log に mailhost, username, password の情報が表示されます。設定した情報でメールサーバにアクセスされているかどうかを確認することができます。	true
removeErrorMail	処理途中でエラーが発生したメールについてもメールサーバから削除します。	true
JavaMailDebug	ジョブを実行するたびに、詳細なログを出力します。[詳細...]	true

ジョブの実行を停止する

ジョブを一時的に無効にすることができます。

1. 管理者権限をもったアカウントでログオンします。
2. メニューから「ジョブ」「ジョブスケジュール検索」画面を開きます。
3. 該当する「ジョブスケジュール」を検索し、更新画面を開きます。
4. 「有効」フラグについているチェックをはずし、保存します。

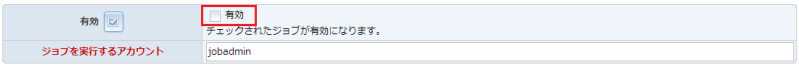


図 ジョブを無効にする

実行結果をメールで送信する

ジョブの実行結果をメールで送信することができます。

多重実行の抑制

ジョブの多重実行を行うことはできません。

あるジョブを 1 分毎に起動させるようにした場合、1 分後にこのジョブの実行が終了していなければ、新しいジョブは実行されません。

juserモデルのメール受信

juserモデルに対するメール受信で、パスワードの登録も行うことができます。　メール本文に「新規パスワード」という項目を加えてください。

メールサーバのサンプル

Gmail

送信

設定項目	値	備考
メールサーバ	smtp.gmail.com	
プロトコル	SMTP over SSL	標準のポート番号は 465。
SMTP認証利用	利用する	
SMTP認証アカウント名	XXX@gmail.com	ご利用のGmailメールアカウント
SMTP認証パスワード	***	ご利用のGmailメールアカウントのパスワード

受信 (POP)

設定項目	値	備考
メールサーバ	pop.gmail.com:995	
アカウント名	XXX@gmail.com	ご利用のGmailメールアカウント
パスワード	***	ご利用のGmailメールアカウントのパスワード

認証に関する問題と対応策

Wagbyのメール送信機能は、SMTP接続時の認証方式としてOAuth　　2.0に対応していません。そのためGmail 側で安全性の低いアプリとして扱われ、接続を拒否されることがあります。POPによるメール受信機能についても、同様に接続できないことがあります。

この場合、下記ページを参考に安全性の低いアプリのアクセスを有効にしてください。

「安全性の低いアプリがアカウントにアクセスするのを許可する」 https://support.google.com/accounts/answer/6010255?hl=ja

もしくは、Google アカウントの2段階認証を有効にして、アプリパスワードを設定する方法もあります。Wagbyからのアクセスには、設定したアプリパスワードをパスワードとして使用してください。

> 将来の Wagby で SMTP 認証方法に OAuth 2.0 に対応する予定です。(POP 認証は Gmail が OAuth 2.0 未対応のため、上述のように Gmail 側の設定変更が必要です。)

トラブルシューティング

- 「POP/IMAP によるメール送信の問題」https://support.google.com/mail/answer/78775?hl=ja

Office365

送信

設定項目	値	備考
メールサーバ	smtp.office365.com:587	
プロトコル	SMTP STARTTLS	
SMTP認証利用	利用する	
SMTP認証アカウント名	-	ご利用のOffice365メールアカウント
SMTP認証パスワード	***	ご利用のOffice365メールアカウントのパスワード

トラブルシューティング

Client does not have permissions to send as this sender

次のエラーです。

```
com.sun.mail.smtp.SMTPSendFailedException: 550 5.7.60 SMTP; Client does not have permissions to send as this sender
```

こちらは、SMTP 認証に利用したメールアカウントと異なる差出人(From)を指定して、メールを送信した場合に発生することがあります。メール差出人とSMTP認証メールアカウントを同じものに指定した上でメールの送信を行ってください。

メールヘッダを使ってメールのスレッド表示を実現する

例

> このページを読む前に、「メール送信（メールテンプレートを固定する）」に目を通しておいてください。

ここではメッセージIDを「<JasmineSoft主キー番号.送信回数@jasminesoft.co.jp>」とし、返信先として最初のメッセージID（が示すメールアドレス）とする例を紹介します。

図は、新規登録直後に最初のメールが自動送信された画面です。「メッセージID」「返信先」はそれぞれメールヘッダのMessage-IDとIn-Reply-Toに用いられます。

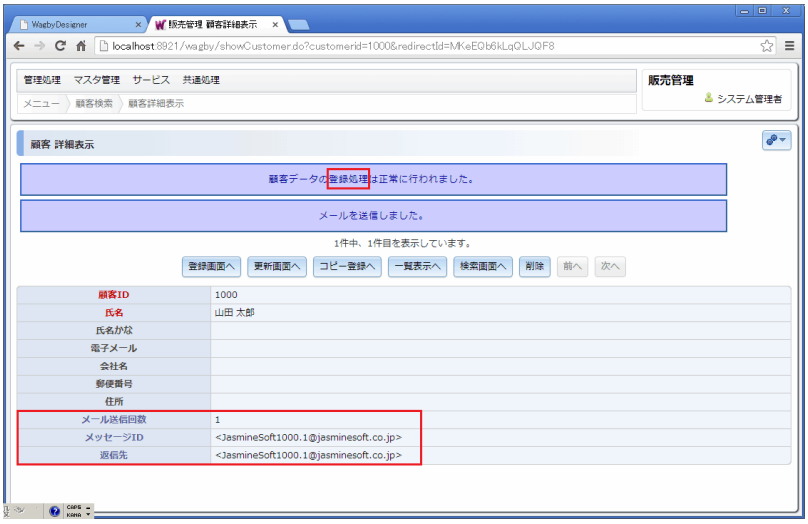


図 新規登録直後の自動メール送信で使われたメッセージIDと返信先

続いて一回目の更新を行った直後の自動メール送信を図に示します。メール送信回数が"2"となり、メッセージIDも".2"となっています。返信先は変わりません。

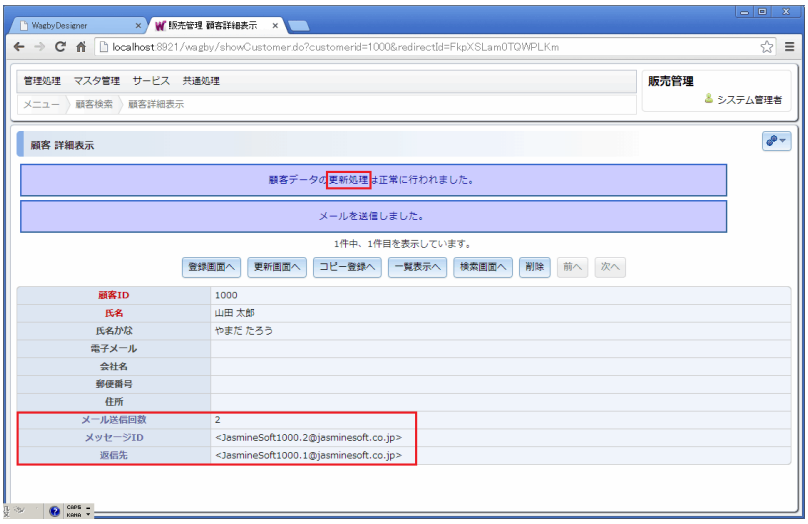


図 最初の更新時の自動メール送信で使われたメッセージIDと返信先

さらに二回目の更新を行った直後の自動メール送信を図に示します。メール送信回数が"3"となり、メッセージIDも".3"となっています。返信先は変わりません。

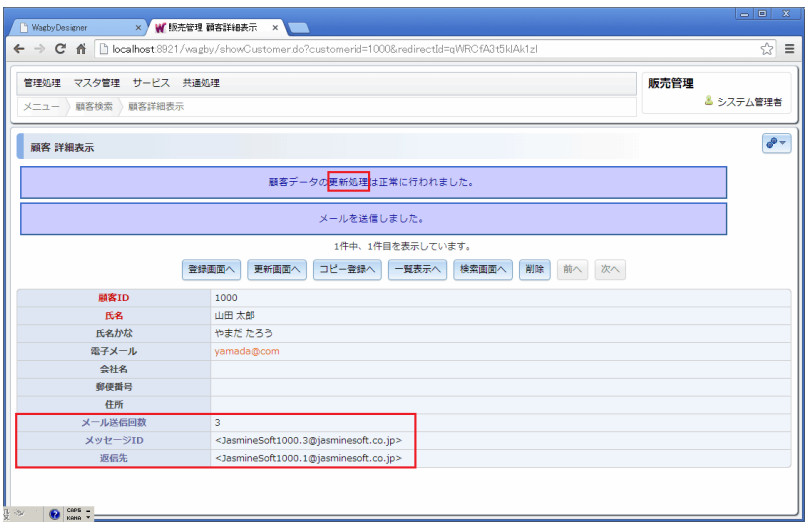


図 次の更新時の自動メール送信で使われたメッセージIDと返信先

図から図で自動送信されたメールです。メールのスレッド表示に対応していることがわかります。

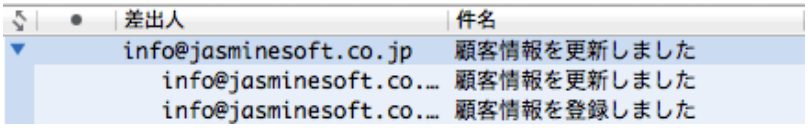


図 メールのスレッド表示に対応している

定義方法

モデルの定義

メール送信機能を設定した顧客モデルの定義を図に示します。「メール送信回数」「メッセージID」「返信先」という3つの項目がポイントです。

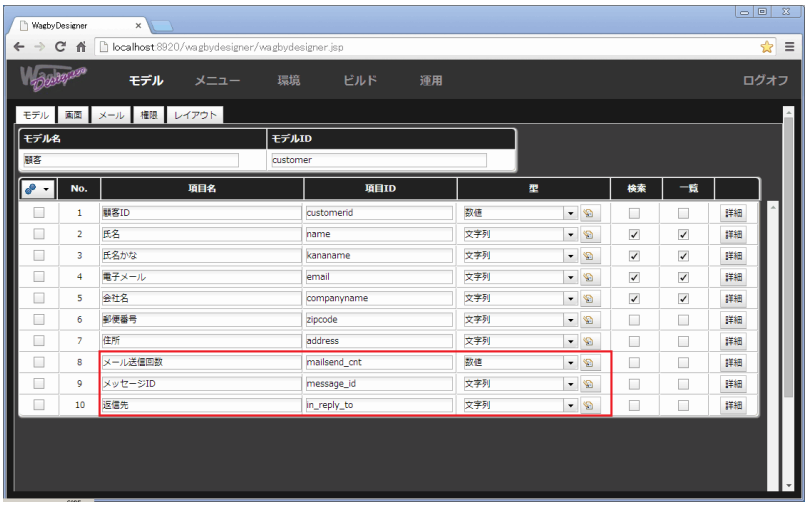


図 顧客モデルの定義

メール送信の設定

今回はデータの登録時と更新時に自動メール送信を行うようにしています。 メールテンプレートは登録用と更新用に分けて用意するとします。

送信成功時に「メール送信回数」項目をインクリメント（1つ増加）するため、次の式を設定しています。

```
${mailsend_cnt}=${mailsend_cnt}+1
```

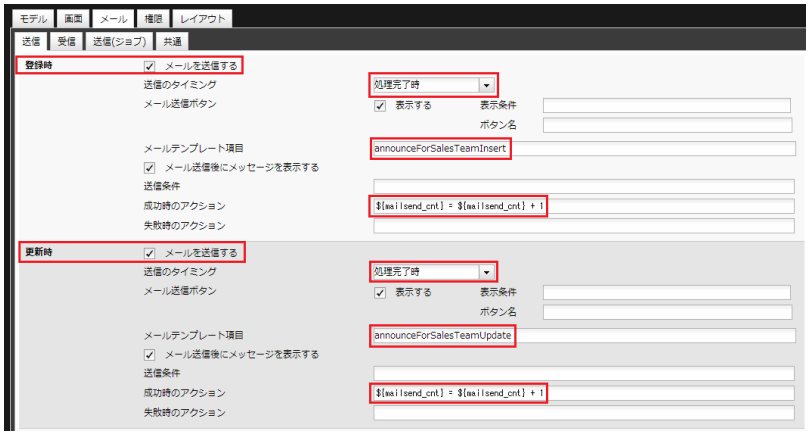


図 メール送信設定

メッセージID項目

データの登録時と更新時の自動メール送信にあわせて、次の式を設定します。

```
CONCATENATE("<JasmineSoft", TOSTR(${customerid}), ".", TOSTR(${mailsend_cnt}),
 "@jasminesoft.co.jp>")
```

図 メッセージIDの式（登録時と更新時に実行される）

CONCATENATE関数は、引数の文字列を結合します。 TOSTR関数は、引数の数値を文字列に変換します。例ではcustomerid項目が数値型のため、TOSTR関数を適用しています。 "JasmineSoft" や "@jasminesoft.co.jp" の文字列は適切に置換してお使いください。

この項目は入力できないよう、読み込み専用とします。

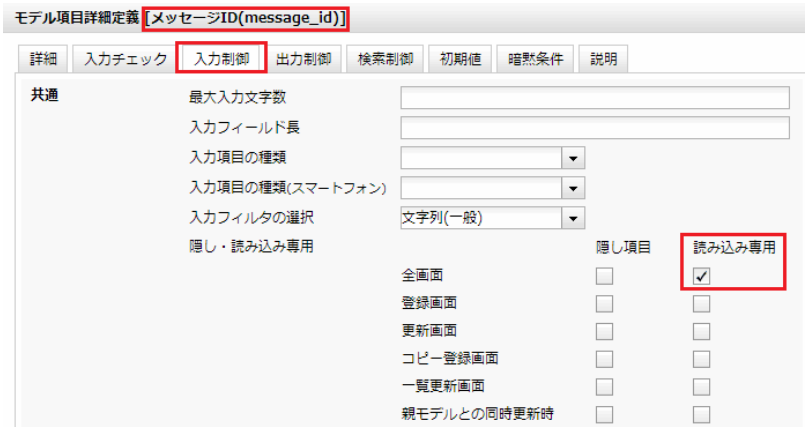


図 読み込み専用の設定

返信先項目

登録時は空とします。更新の都度、最初のメッセージID （送信回数が 1 の状態）を格納するようにします。

```
CONCATENATE("<JasmineSoft", TOSTR(${customerid}), ".1@jasminesoft.co.jp>")
```

図 返信先の式（登録時は空、更新時のみ実行される）

この項目も入力できないよう、読み込み専用とします。

メールテンプレート

メールテンプレートでは、図のようにメールヘッダを指定します。

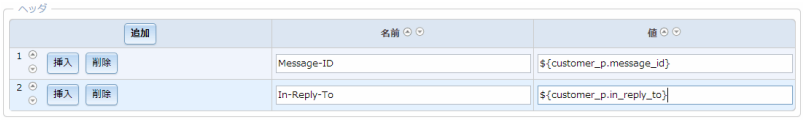


図 メールヘッダの設定

注意点

Message-ID, In-Reply-To の形式は RFC で定められており、不正な形式でメールを送信した場合、メールサーバーやメールクライアントで問題が発生する可能性があります。詳細は下記URLをご参照ください。

```
RFC 2822 "Internet Message Format" http://www.ietf.org/rfc/rfc2822.txt
```

集計・グラフ

数値の集計

例

日付別かつ対応者別の、製品サポート時間の集計例を図に示します。メジャーと指定しているのは整数型項目（ここでは時間）です。

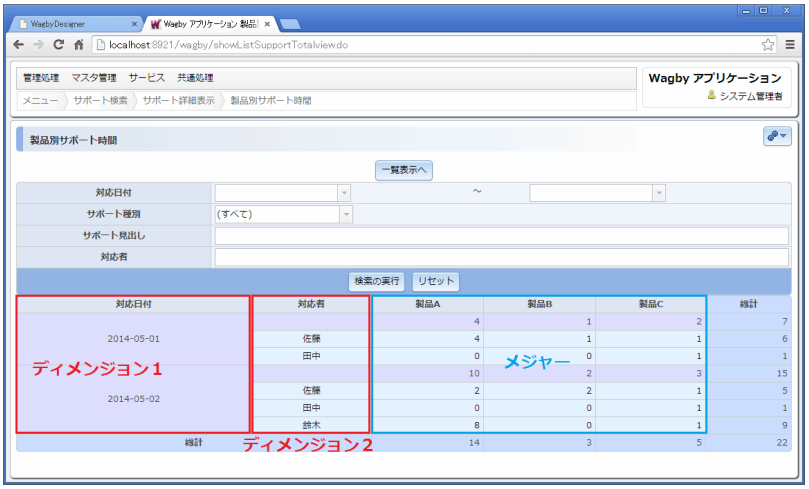


図 日付別対応者別製品サポート時間集計

対応者で絞込むと、集計ビューの結果も変わります。

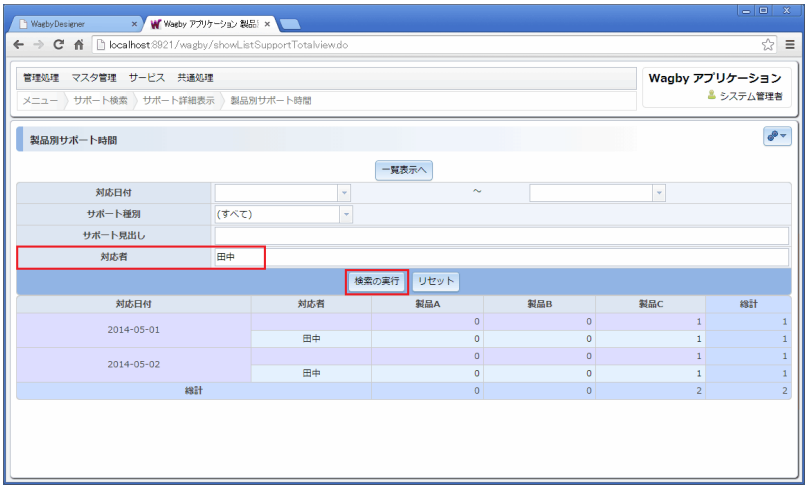


図 対応者での絞り込み

定義方法

集計ビューを用いる「サポート」モデルの定義を図に示します。

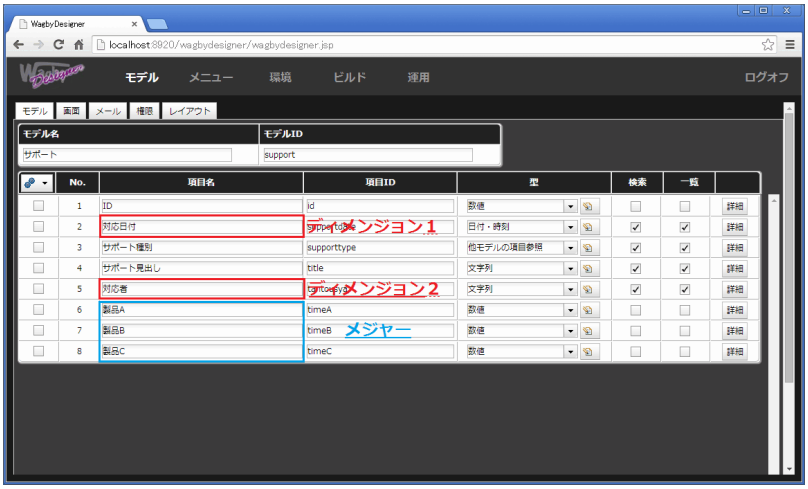


図 サポートモデル

「画面>ビュー>集計ビュー」を開きます。「画面を作成する」を有効にします。

「ページタイトル」は、集計ビューへ遷移するためのボタン名としても利用されます。空白時はボタン名が「集計ビュー1」などとなるため、なるべくページタイトルを設定してください。

「ディメンジョン」の欄に、項目を設定します。今回は二つのディメンジョンを用意します。"対応日付別、対応者別" としています。

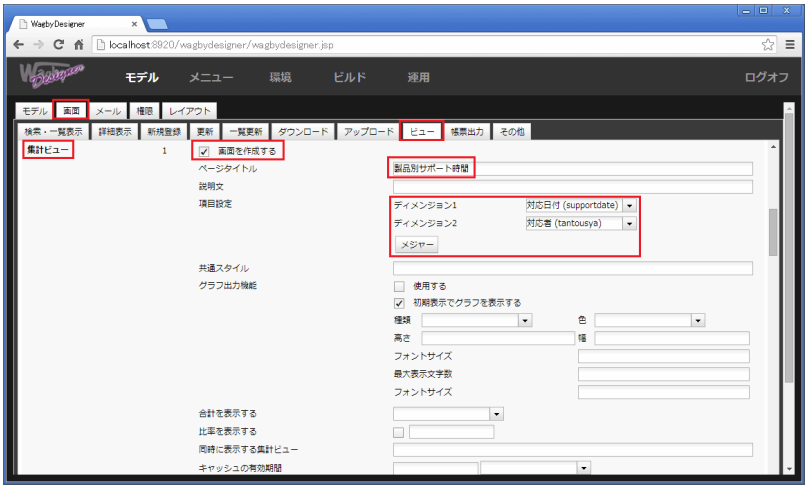


図 ページタイトルとディメンジョンの設定

図で「メジャー」のボタンを押下し、どの項目をメジャーとして使うかを指定します。 ここでは数値型項目である製品A,B,C項目の "合計" を算出するとしています。

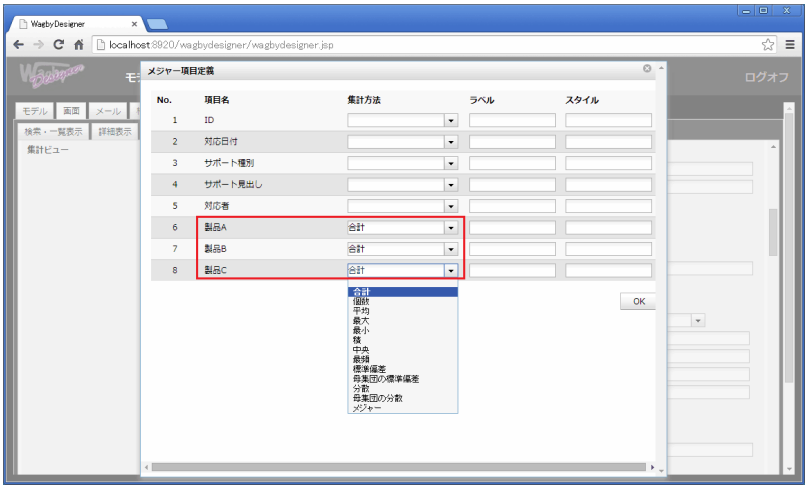


図 メジャーとなる項目の設定

数値フォーマットの扱い

集計ビュー用に別途、数値フォーマットを指定する必要はありません。

集計ビューで用いられる数値フォーマットは、メジャー項目でそれぞれ数値フォーマットが指定されている場合、最も複雑なもの（記述フォーマット文字列が長いもの）を採用します。

図 数値フォーマット指定は集計ビューにも自動的に反映される

集計結果が不定の場合

データによっては集計結果が得られない場合があります。

例えば最大・最小を求める場合で、すべてのデータが未入力（ゼロではない）という場合です。 このようなとき、集計結果は NaN（Not a Number）になります。

一つでもNaNを含む集計結果は、グラフに表示することもできません。

選択肢の集計

例

メジャーにモデル参照項目を指定した場合、すべての値が展開され、各値が選択された回数の集計が行われるようになります。

ここでは、項目の型が「モデル参照（ラジオボタン）」となっている「地域区分」項目をメジャーとして設定した例を示します。

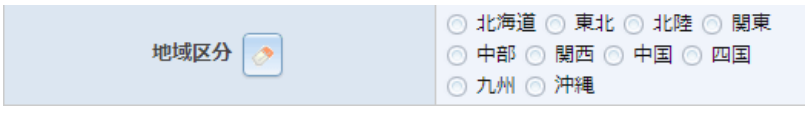


図 地域区分（入力画面）

地域区分に用意された選択肢(北海道、東北、北陸、関東、中部、関西、中国、四国、九州、沖縄)がメジャーに展開され、それぞれが選択された回数を集計結果として扱います。

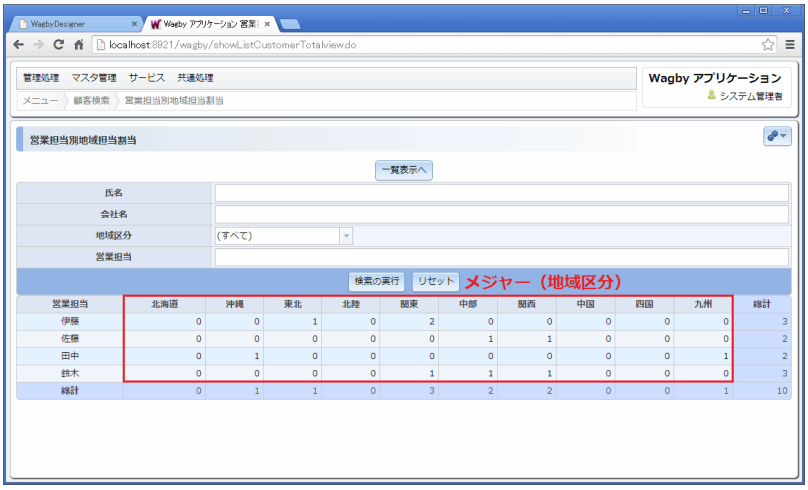


図 地域区分の集計

地域区分を絞込むと、集計ビューの結果も変わります。

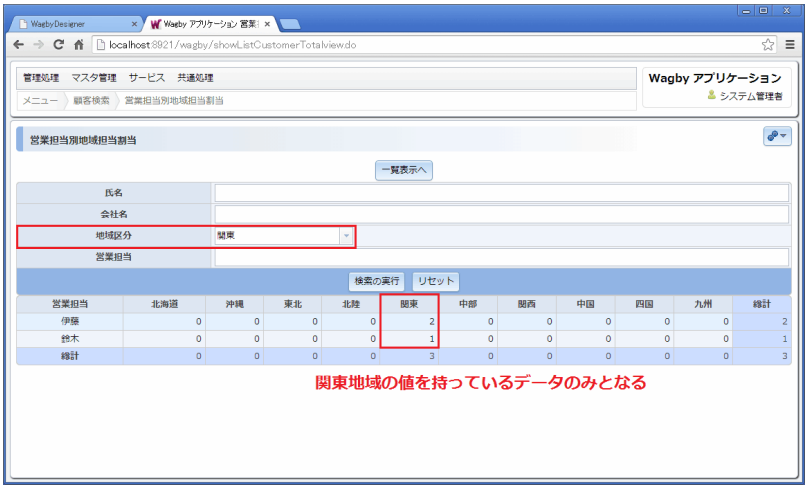


図 地域区分の集計

定義方法

地域区分は選択肢モデルとして用意しました。

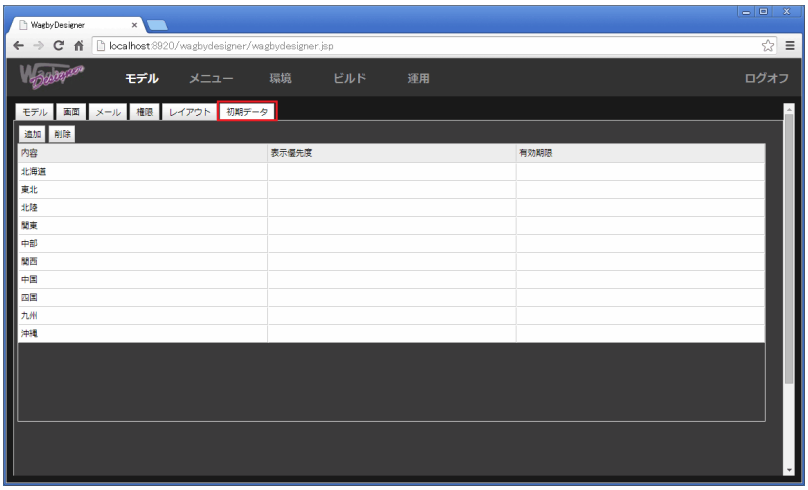


図 地域区分の初期データ

集計ビューを用いる「顧客」モデルの定義を図に示します。

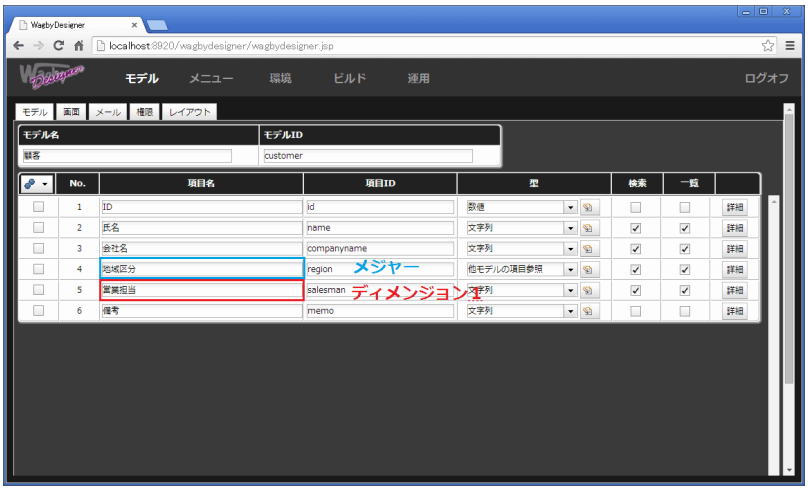


図 顧客モデル

「画面>ビュー>集計ビュー」を開きます。「画面を作成する」を有効にします。

「ページタイトル」は、集計ビューへ遷移するためのボタン名としても利用されます。空白時はボタン名が「集計ビュー1」などとなるため、なるべくページタイトルを設定してください。

「ディメンジョン」の欄に、項目を設定します。

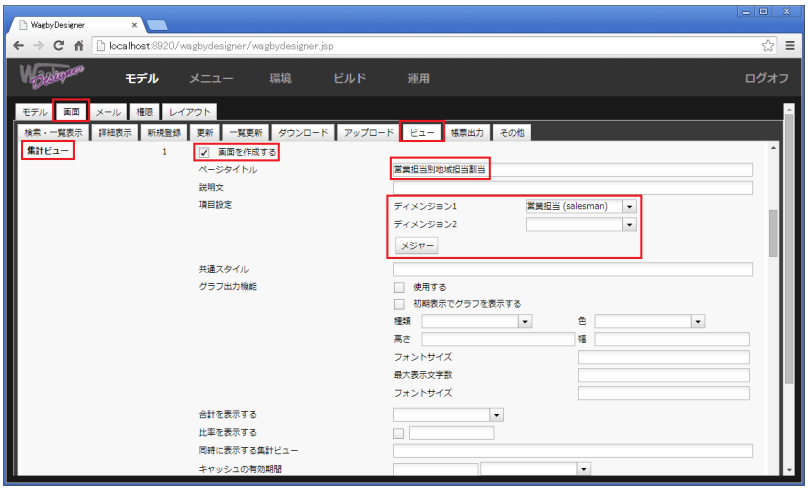


図 ページタイトルとディメンジョンの設定

図で「メジャー」のボタンを押下し、どの項目をメジャーとして使うかを指定します。 選択肢モデルの場合、選択肢の最下部にある〝メジャー〟を指定します。

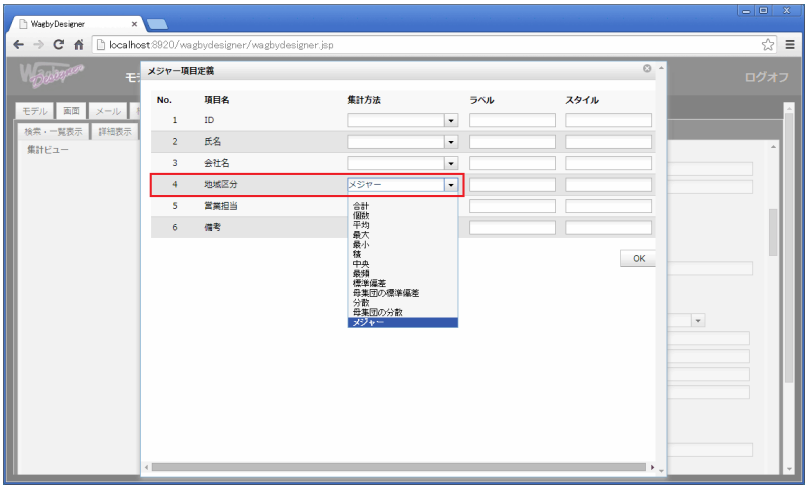


図 メジャーとなる項目の設定

制約

テーブルを作成しない選択肢モデルをメジャーに指定することはできません。

文字列の集計

例

メジャーに文字列型を指定した場合、記述した文字列が展開されます。 各文字列が記述された回数の集計が行われるようになります。 （文字列が一致していた場合は同じ値とみなし、集計結果に加算されます。）

アンケートモデルを例に見てみます。入力された回答は図のようになっています。

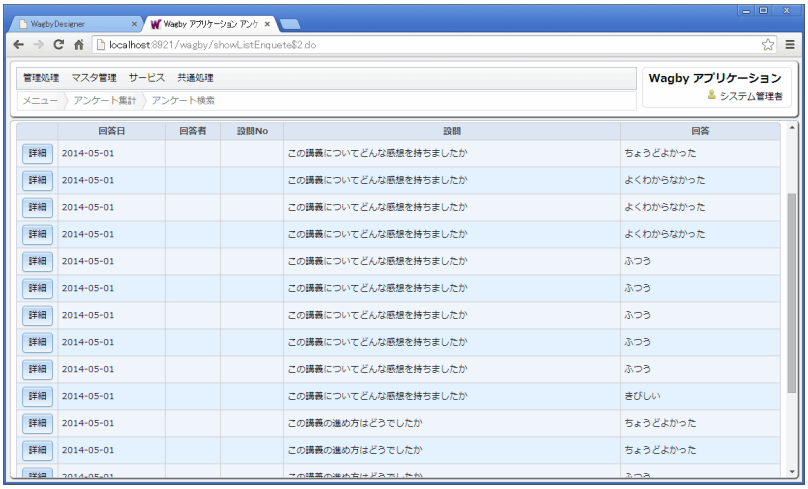


図 アンケート一覧表示

集計ビューでは、項目の型が「文字列」型の「回答」項目をメジャーに指定しています。 回答で記述した内容がメジャーに展開され、それぞれの内容の記述された回数が集計結果として扱われます。

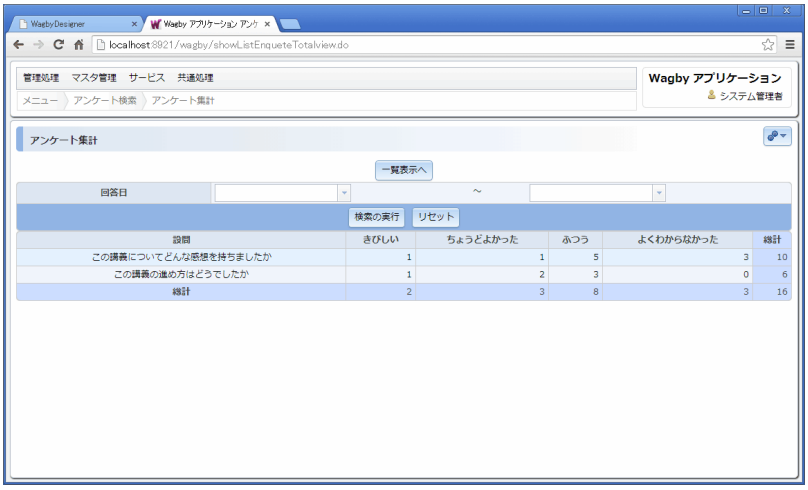


図 アンケート結果の集計

定義方法

集計ビューを用いる「アンケート」モデルの定義を図に示します。

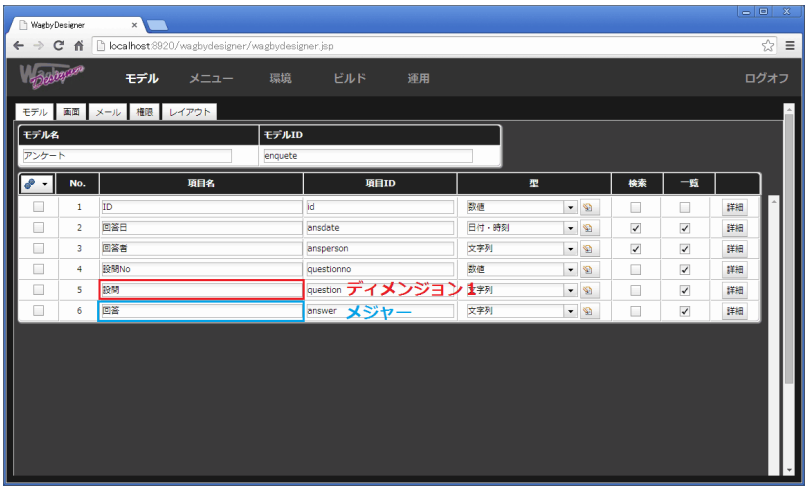


図 アンケートモデル

「画面>ビュー>集計ビュー」を開きます。「画面を作成する」を有効にします。

「ページタイトル」は、集計ビューへ遷移するためのボタン名としても利用されます。空白時はボタン名が「集計ビュー1」などとなるため、なるべくページタイトルを設定してください。

「ディメンジョン」の欄に、項目を設定します。今回は〝設問〟をディメンジョンとします。

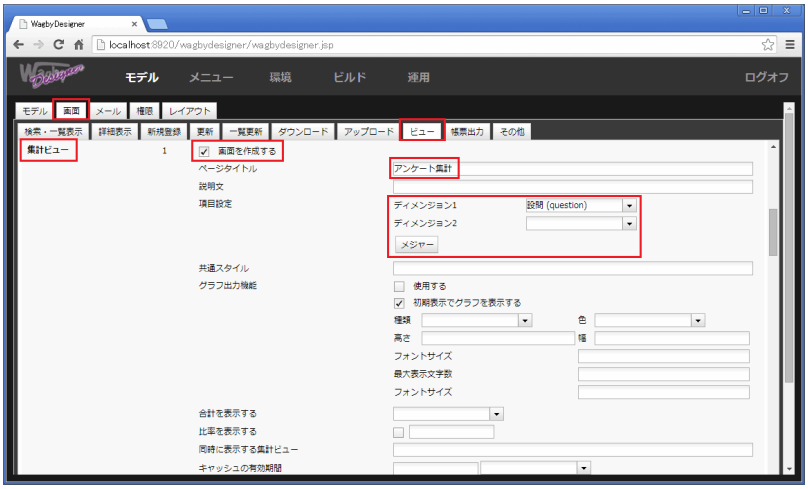


図 ページタイトルとディメンジョンの設定

図で「メジャー」のボタンを押下し、どの項目をメジャーとして使うかを指定します。 ここでは文字列型項目である回答項目を指定します。数値型項目以外では、集計方法は「メジャー」を指定してください。

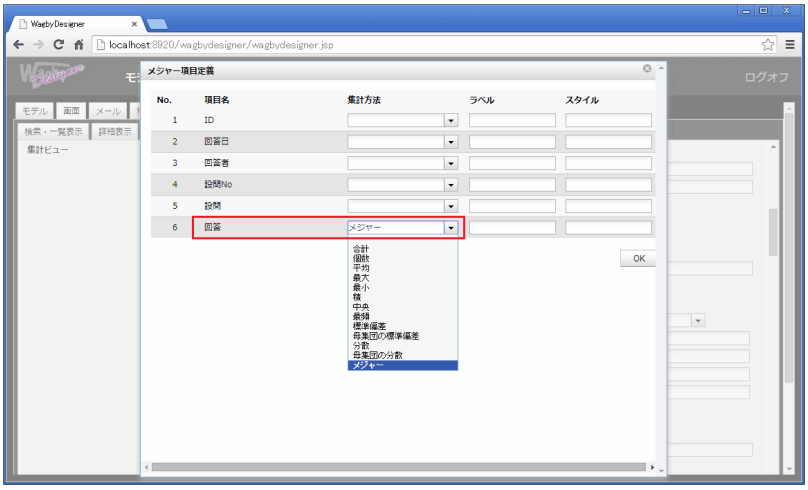


図 メジャーとなる項目の設定

メジャー毎に合計・個数・平均・最大・最小・積・中央・最頻・標準偏差・分散を指定する

例

メジャー毎にさまざまな計算方式を指定することができます。図は、合計と平均を交互に並べた集計の例です。

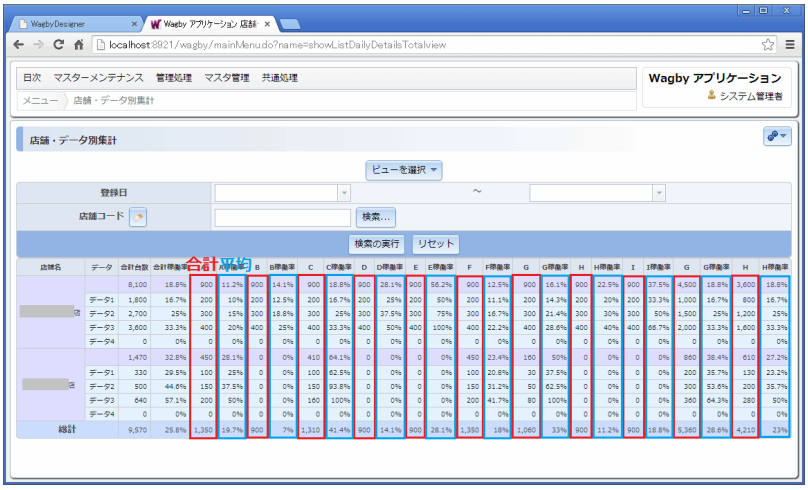


図 合計と平均を交互に並べた集計

定義方法

メジャーとして指定する項目を定義します。（今回はすべて数値型です。）

図 集計対象モデル

「画面>ビュー>集計ビュー」を開きます。「画面を作成する」を有効にします。

「ページタイトル」は、集計ビューへ遷移するためのボタン名としても利用されます。空白時はボタン名が「集計ビュー1」などとなるため、なるべくページタイトルを設定してください。

「ディメンジョン」の欄に、項目を設定します。今回は〝店舗名〟と〝データ〟の二つのディメンジョンを指定します。

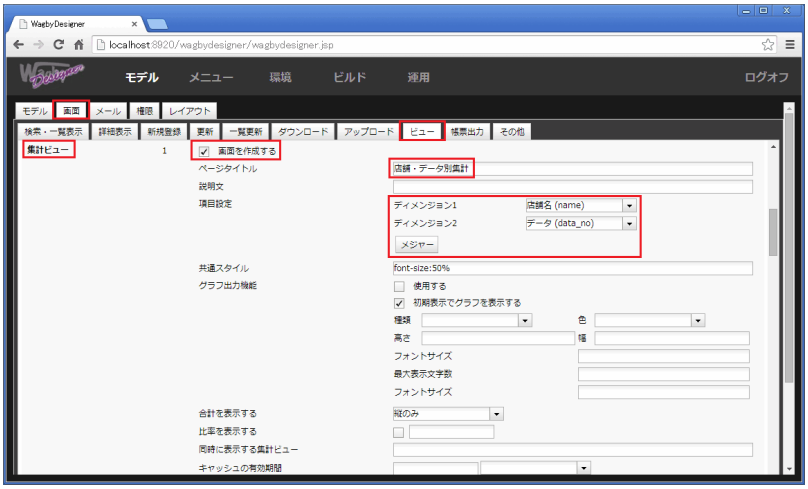


図 ページタイトルとディメンジョンの設定

図で「メジャー」のボタンを押下し、どの項目をメジャーとして使うかを指定します。 ここでは各項目毎に、合計または平均を指定しています。

図 メジャーとなる項目の設定

数値型項目では、集計方法は次のいずれかを指定できます。

- 合計
- 個数
- 平均
- 最大
- 最小
- 積
- 中央 (389)
- 最頻
- 標準偏差
- 母集団の標準偏差
- 分散
- 母集団の分散
- メジャー (390)

(389) 中央、最頻、標準偏差、分散の詳細は、リンク先の関数の説明をお読み下さい。標準偏差や分散は2つ以上の値が必要など、計算上の制約があります。

(390) その項目の型による自動判定が行われます。数値は合計に、モデル参照や文字列型は合計または個数になります。

１つのモデルに複数の集計ビューを用意する

例

一覧表示画面に用意された「ビューを選択」ボタンから、各集計ビュー画面に遷移することができます。ここでは売上伝票モデルに4つの集計ビューを用意した例を紹介します。

図 ビューを選択ボタン

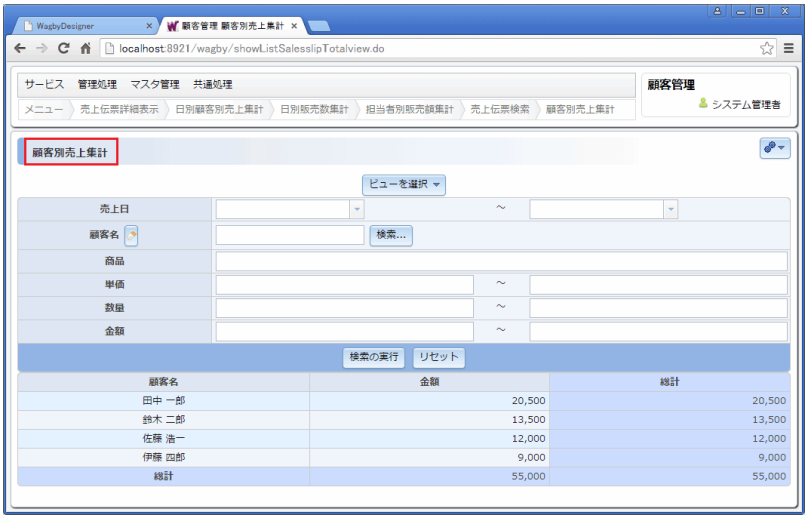


図 顧客別売上集計ビュー

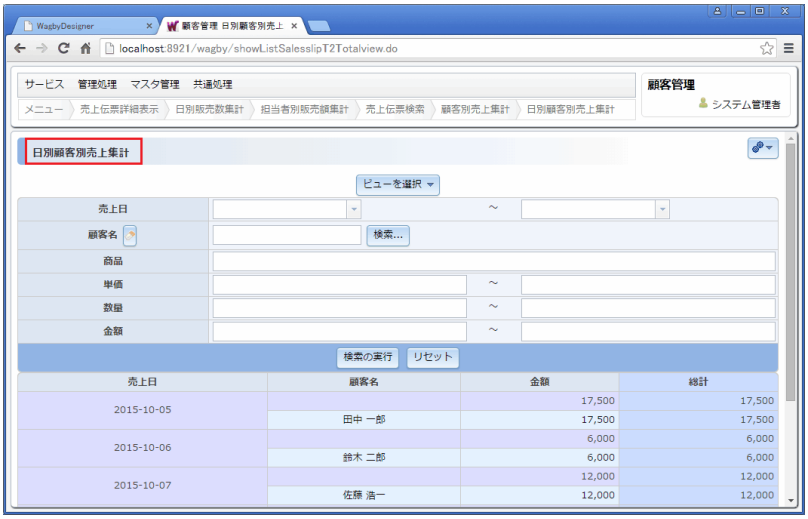


図 日別顧客別売上集計ビュー

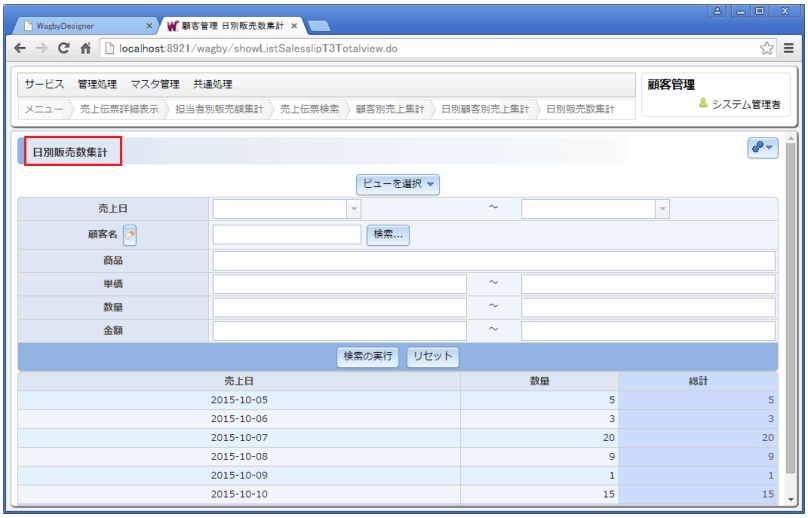


図 日別販売数集計ビュー

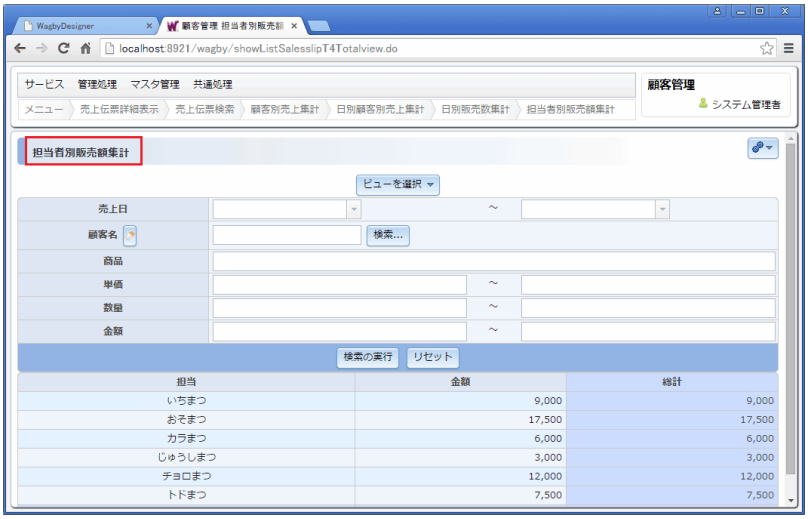


図 担当者別販売額集計ビュー

定義方法

今回用いた売上伝票モデルを図に示します。

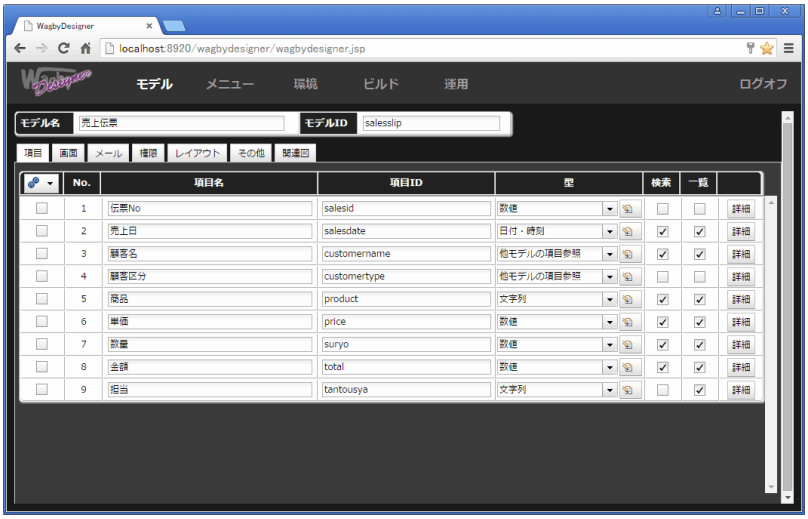


図 売上伝票モデル

集計ビューを複数表示させるには「画面 ＞ ビュー ＞ 集計ビュー」を用意する分について「画面を作成する」を有効にします。ここでは 1 から 4 までの設定を行っています。

最大で10個の集計ビューを定義することができます。

図 最初の設定

図 二番目の設定（以下、略）

グラフも同時に用意する

　グラフを同時に表示させることもできます。ビューを切り替えると、集計表とグラフがそれぞれ表示されます。

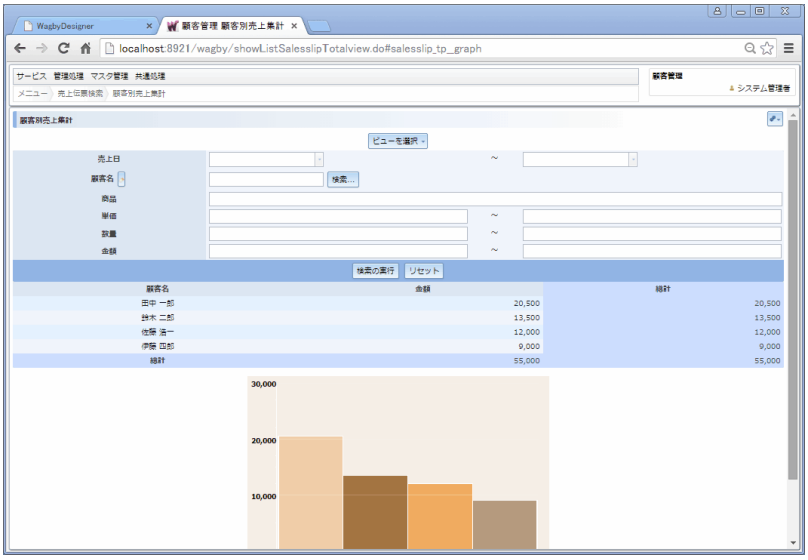


図 顧客別売上集計ビューとグラフ

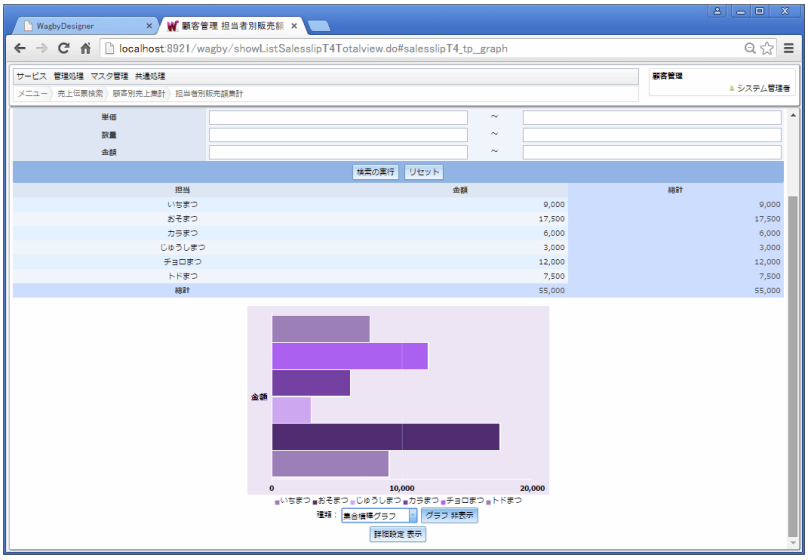


図 担当者別販売額集計ビューとグラフ

1つの画面に複数の集計ビューを表示する

1画面に複数の集計ビューを表示することができます。　次の例は「顧客別売上」を表示する画面に「顧客日付別売上」「顧客区分売上数」「担当者別販売額」を同時に表示しています。

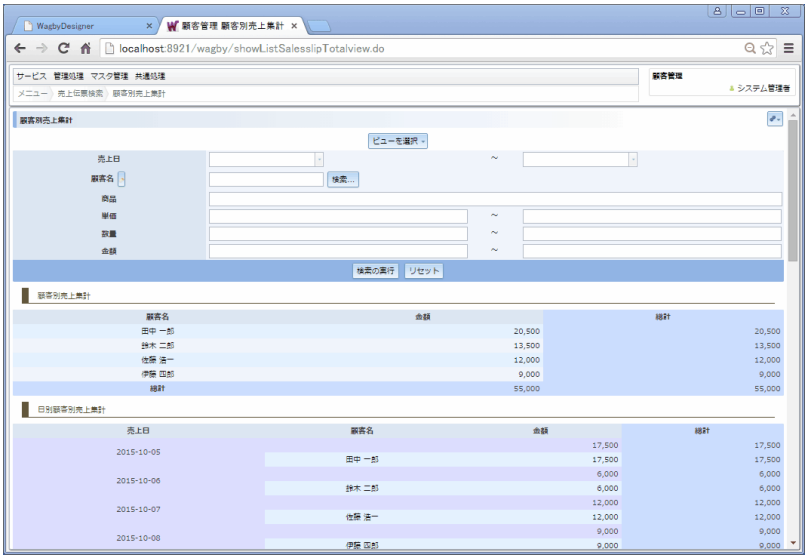


図 顧客別売上集計ビュー(1)

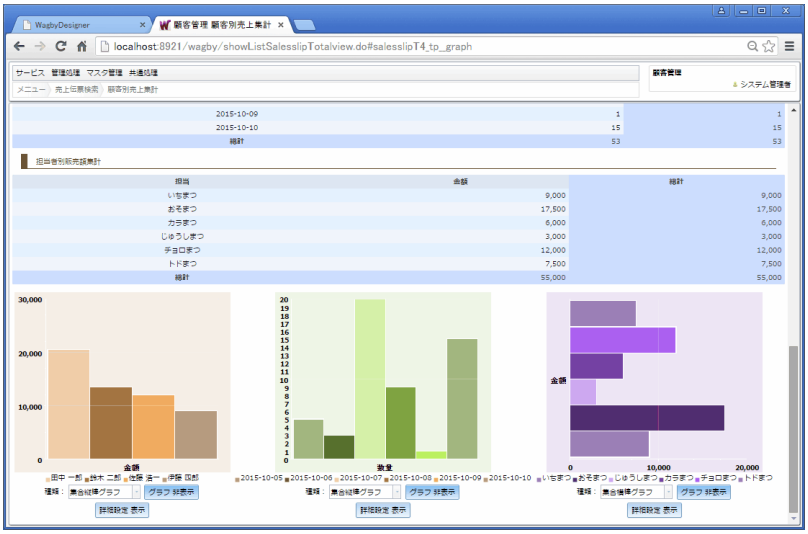


図 顧客別売上集計ビュー(2)

各集計ビューにグラフ描画指定があった場合、すべてのグラフが同一画面に表示されます。表示方法は、最初に各集計表が表示されたあと、グラフが表示されるようになります。

各グラフのサイズを小さめに設定すると、上図のように、可能な限りグラフを横並びで表示するようになります。

定義方法

「同時に表示する集計ビュー」で同時に表示したい集計ビューの画面番号を指定します。 複数指定する場合は、「,（カンマ）」区切りで指定してください。数字、カンマは半角で入力してください。

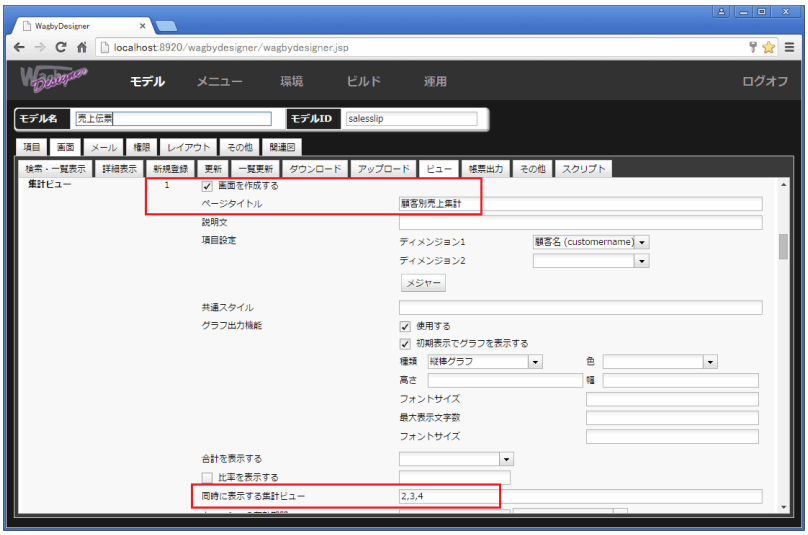


図 同時に表示する集計ビューの設定

指定された集計ビューには「同時に表示する集計ビュー」は指定できません。 この例では、「集計ビュー2」と「集計ビュー3」では他の集計ビュー画面を指定できません。

集計結果をキャッシュする

本機能の概要と効果

集計ビューでは、画面を開くタイミングで対象モデルの全データがいったん読み込まれます。このため大量データの集計には時間がかかります。　これを改善するために、一度集計した結果を指定時間だけキャッシュしておく機能を用意しています。

この機能は、集計結果をメモリ上に保存します。キャッシュ有効期間内に他の利用者が集計ビューを開いた場合、計算された結果を用いるため、体感上の処理速度を向上させることができます。

定義方法

「キャッシュの有効期間」に、数値を入力します。単位は「秒」「分」「時」「日」「月」「年」から選択することができます。

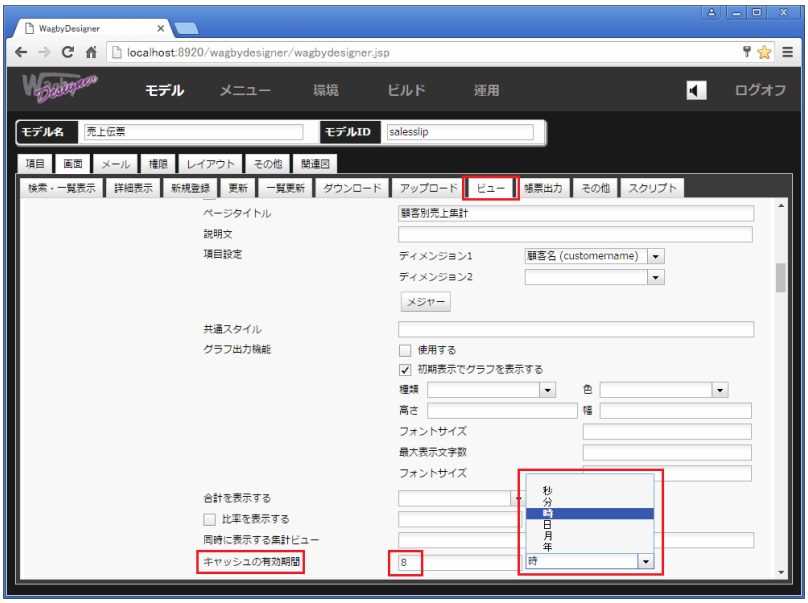


図 キャッシュの設定

随時データの登録や更新が行われているモデルの場合、集計ビューで表示される結果は最新ではない、という状況が起こります。キャッシュの有効危険を調整することで、集計のための演算回数と、データの最新性とのバランスをとることができます。

指定したキャッシュ有効期間内であっても、アプリケーションを再起動した場合は、集計結果はメモリから消去されます。

キャッシュのキー

キャッシュは、検索条件毎に用意されます。従って、検索条件が同じ場合はキャッシュが有効ですが、検索条件が異なる場合はキャッシュされません。

グラフ

例

集計ビューを用意すると、検索画面に集計ビューへ遷移するためのボタンが用意されます。

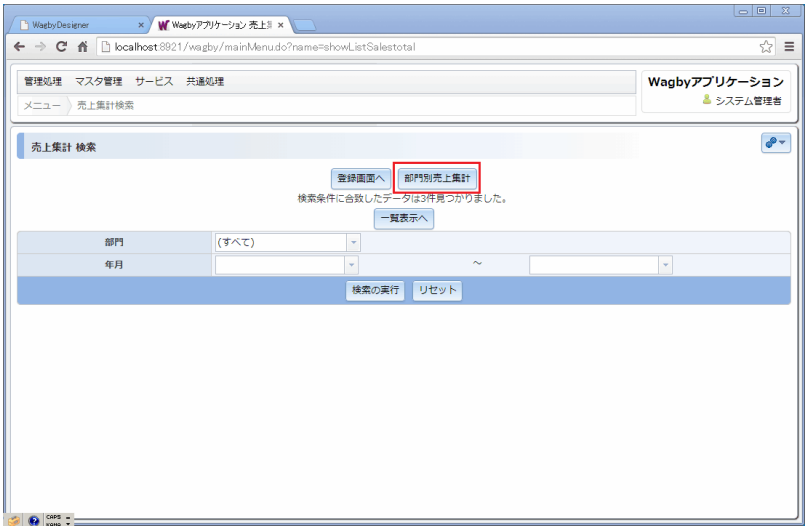


図 部門別売上集計ビューへ遷移するボタン

図は、部門別の売上集計とグラフの表示例です。グラフは集計ビューの下部に用意されます。

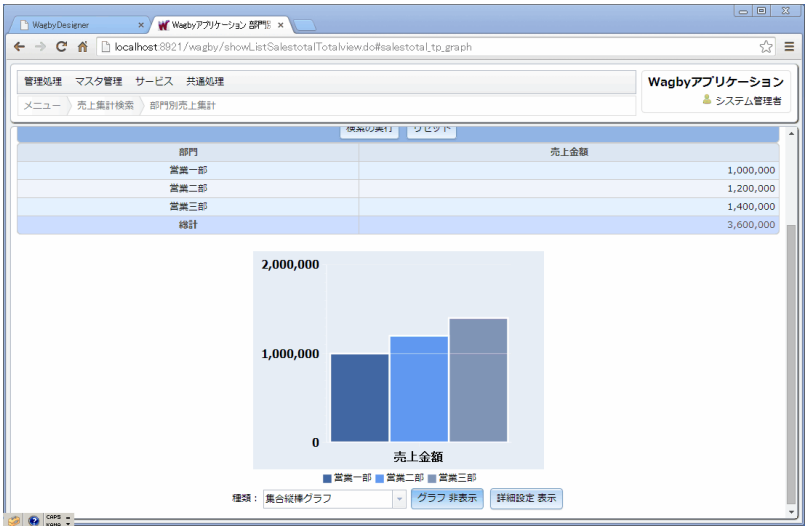


図 部門別売上集計グラフ

定義方法

二つのモデル「部門」「売上集計」を用意します。

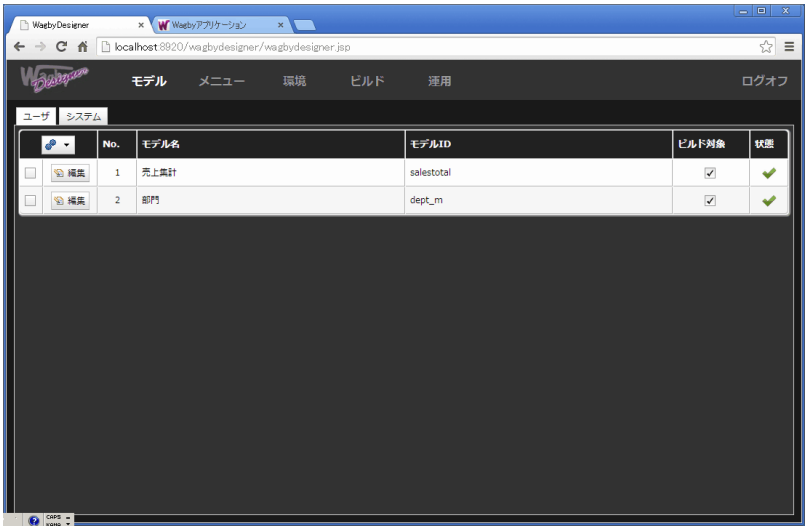


図 二つのモデルを定義する

部門モデルは選択肢モデルとしています。「営業一部」「営業二部」「営業三部」の 3 種類を用意します。

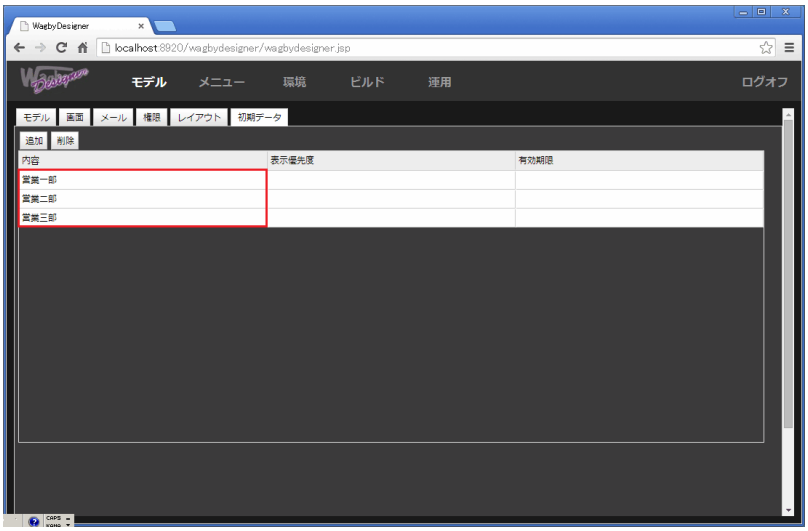


図 年齢区分マスタを用意する

売上集計モデルの定義を図に示します。ディメンジョンは「部門」項目です。メジャーは「売上金額」項目です。(391)

(391) 「売上集計」項目の値は本来、入力ではなく、個々の売上伝票の総計として計算させるのが適切です。ここではグラフ機能の説明のため、単純化しています。

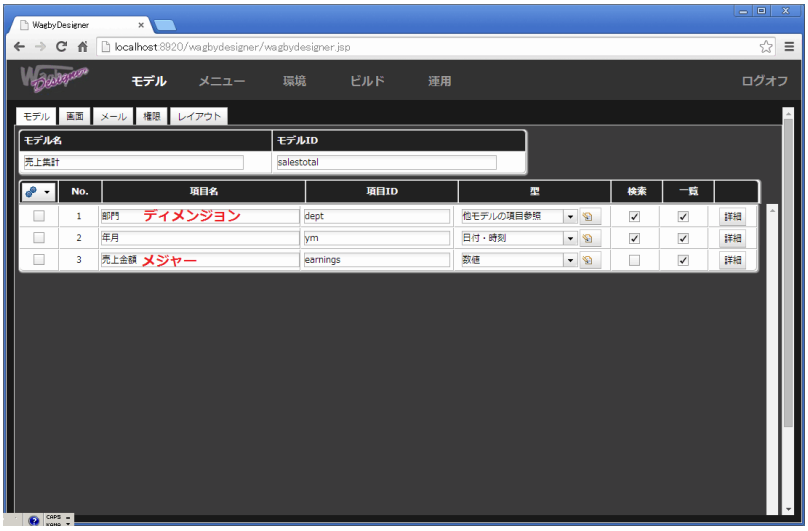


図 売上集計モデルの定義

「部門」と「年月」項目の二つで複合キーを構成します。部門項目は、図に示した部門モデルを参照します。ソートルールをIDの昇順とします。

図 部門モデルを参照する

図 部門項目を主キー（複合キーの一部）とする

「年月」項目も主キーと設定します。

図 年月項目を主キー（複合キーの一部）とする

「年月」項目の入力方式をリストボックス（年月）とします。また初期値を ″本日の日付″ （TODAY関数）と設定しておきます。

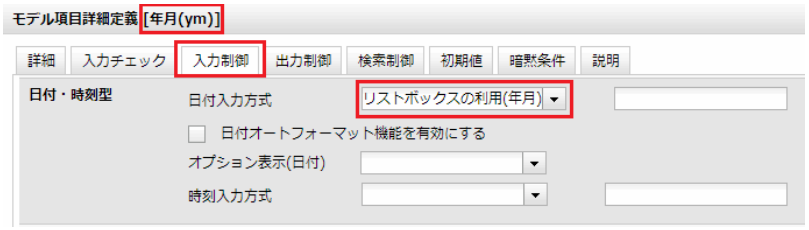


図 日付の入力方式

図 日付の初期値

「売上金額」項目の出力フォーマットを設定します。

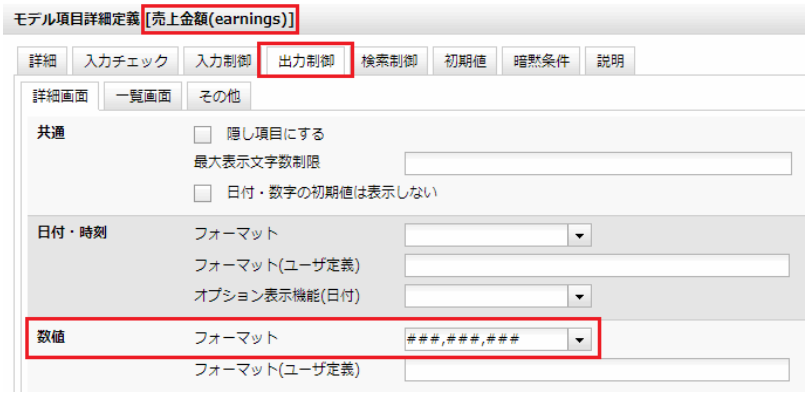


図 売上金額項目の出力フォーマットの設定

「画面>ビュー>集計ビュー」を開きます。集計ビューは複数、用意することができます。ここでは1番の集計ビューを定義します。

「画面を作成する」をチェックし、ページタイトルを設定します。ディメンジョン1は部門項目を指定します。

さらに、グラフ出力を指定します。ここでは種類を「集合縦棒グラフ」とし、色、高さ、幅をそれぞれ指定します。また、合計は縦軸のみとします。

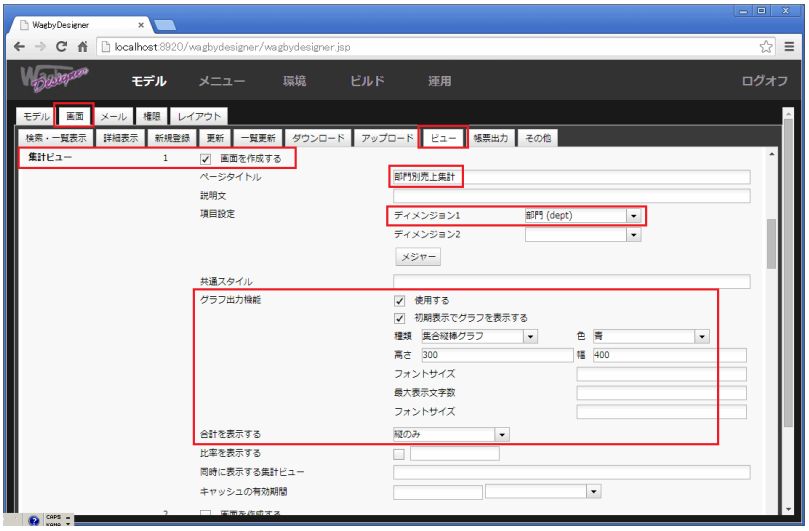


図 集計ビューとグラフの設定

「メジャー」のボタンを押下し、売上集計項目をメジャーと指定します。

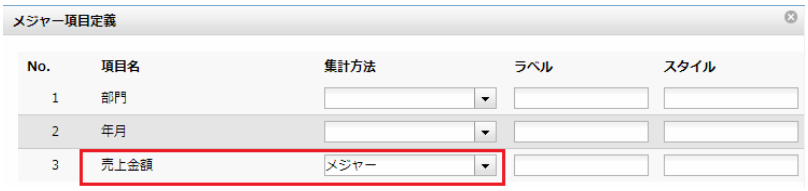


図 メジャーの設定

検索条件（コンディション）の自動設定

例

検索条件に含まれた「年月」について、自動的に今年度（4月はじめ～翌年3月末まで）の集計を行うように検索条件を設定する例を示します。

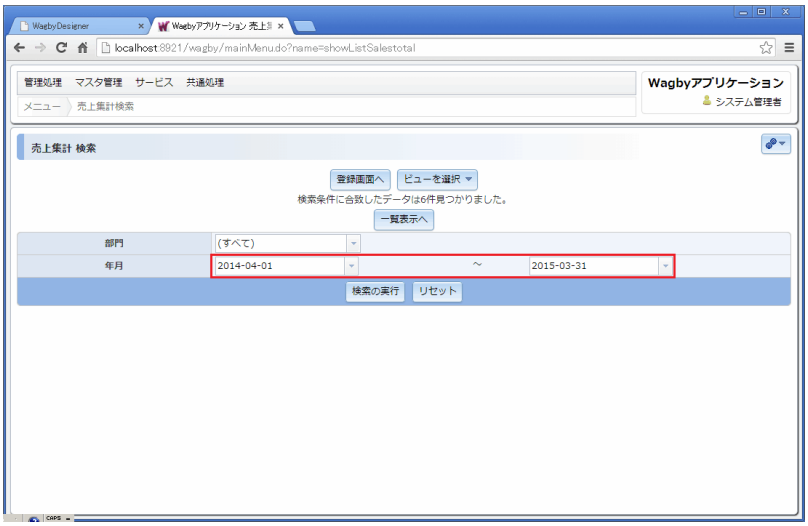


図 検索条件の自動設定

定義方法

数値型と日付型の場合、標準では範囲検索が有効になります。そこで検索条件に二つの式を設定します。それぞれが開始(from)と終了(to)になります。

```
DATEVALUE(YEAR(TODAY())+"-04-01")

DATEVALUE((YEAR(TODAY())+1)+"-03-31")
```

図 検索条件へ式を設定する

比率を表示する

例

合計欄の部分に比率を表示することができます。

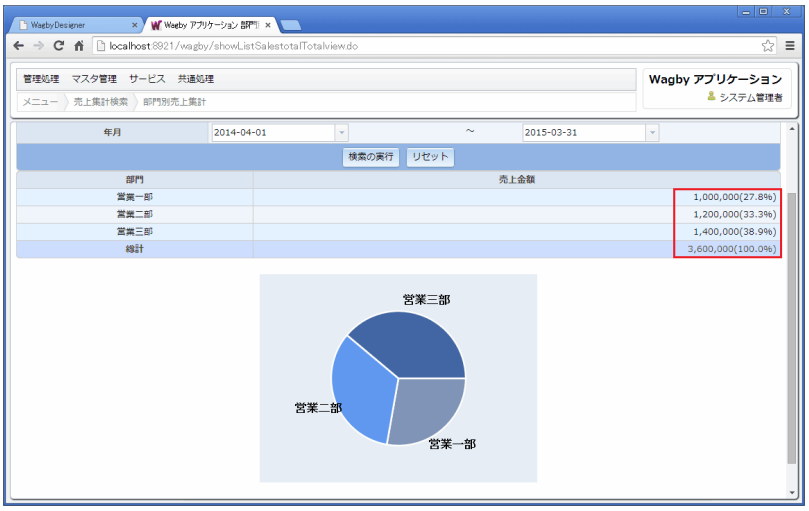


図 比率を表示した例

定義方法

集計ビューの定義に用意された「比率を表示する」を有効にします。

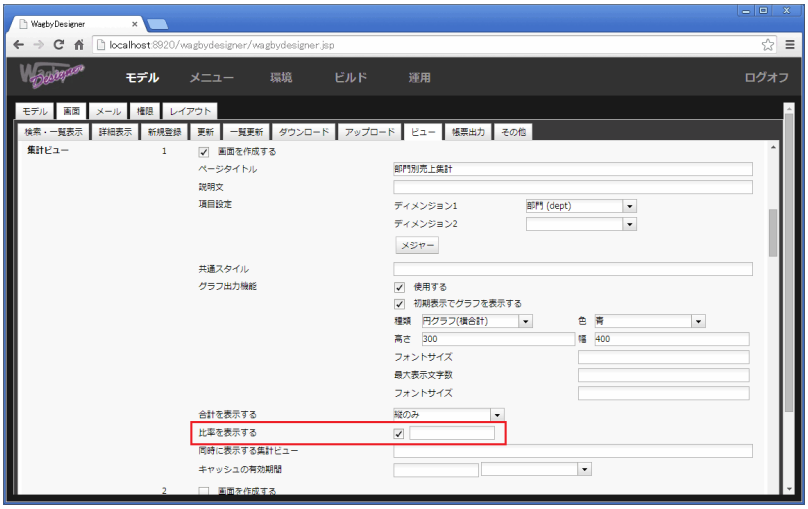


図 比率の設定

比率には数値フォーマットを設定できます。未指定の場合は数値フォーマットで'('##0.0%')'と同じ扱いとなります。 次の例は小数点2桁まで有効とするものです。

```
'('##0.00%')'
```

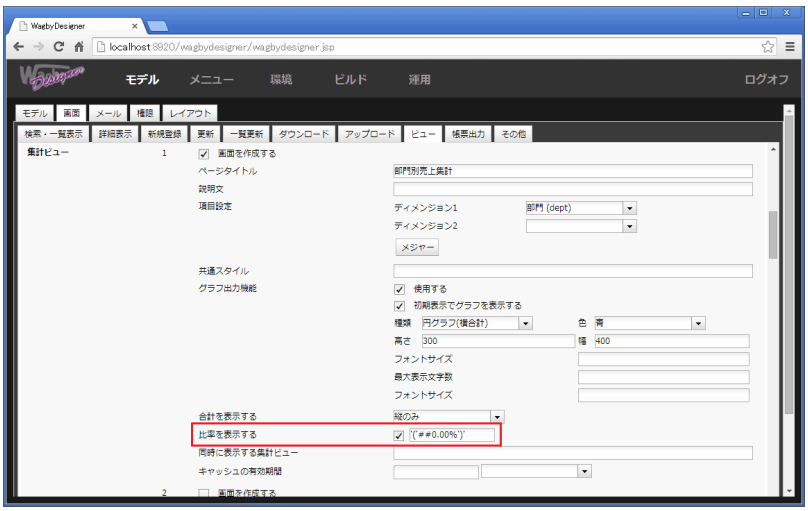


図 比率の設定-数値フォーマットを同時に指定する

仕様・制約

- グラフ出力を行う場合、ディメンジョンは 1 つだけ定義してください。2つのディメンジョンには対応していません。
- 横軸の軸タイトルはメジャー項目名となります。
- 縦軸の軸タイトルのグラフ表示機能は現在、未対応です。
- ラベルの表示エリアが十分確保できない場合には、適当に間引いた形で表示されます。グラフの幅や表示フォントを調整してください。
- 一つでもNaNを含む集計結果は、グラフに表示することができません。

グラフ - 実行時の指定

気象データサンプル

定義方法

ここでは年・月毎の「最高気温」「最低気温」「平均気温」「降水量」を管理する気象データをサンプルとして用います。

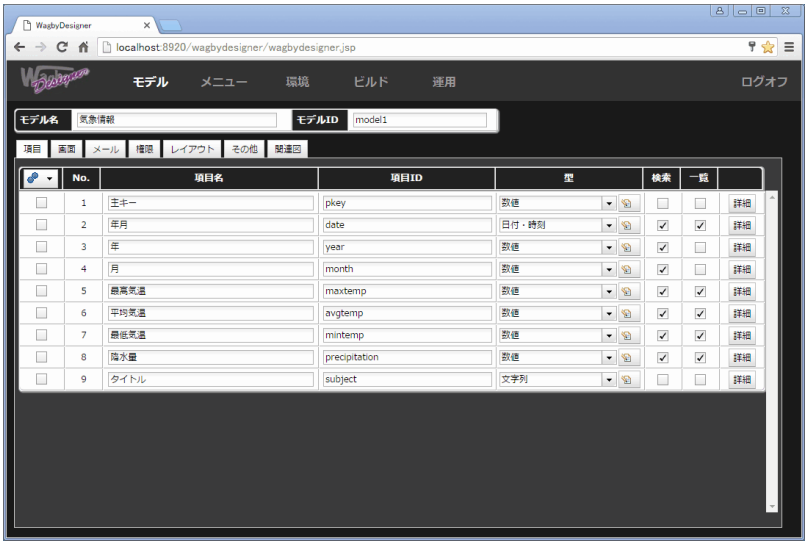


図 気象情報を格納するモデル

初期状態は月別グラフ（集合縦棒グラフ）とします。

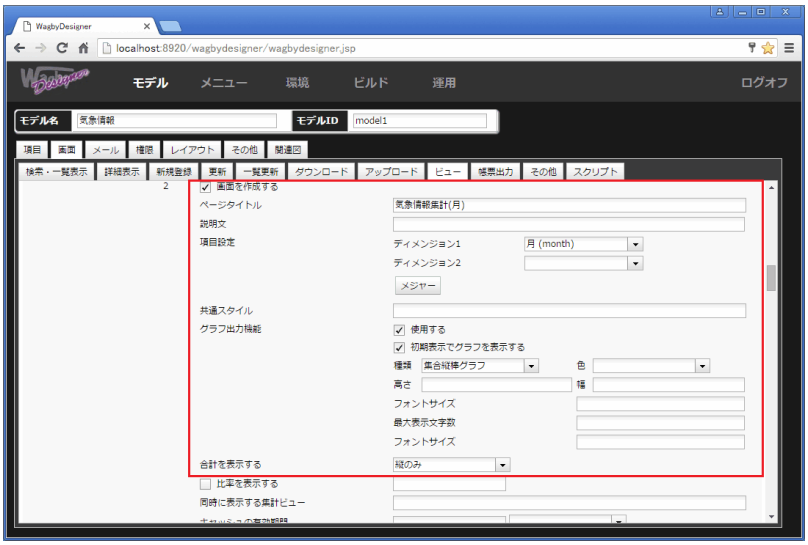


図 グラフの設定(1)

メジャーは「最高気温」「最低気温」「平均気温」「降水量」の平均値とします。

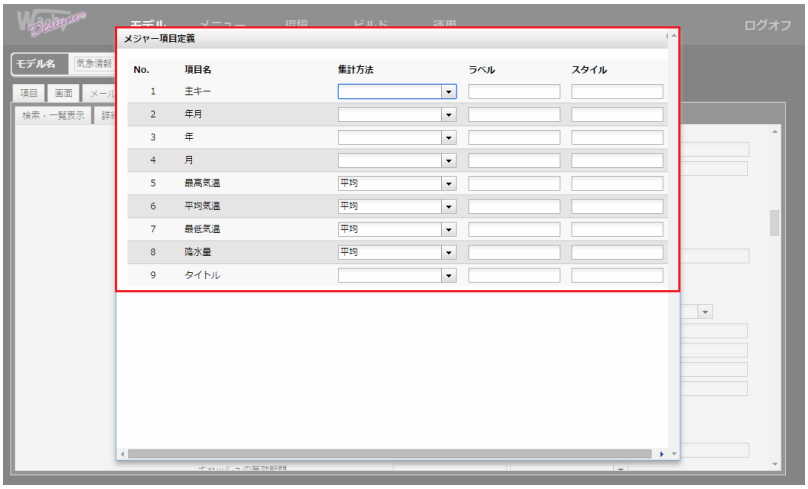


図 グラフの設定(2)

最初の例

ビルド直後の表示は次のとおりです。データはダミーデータを登録済みとします。

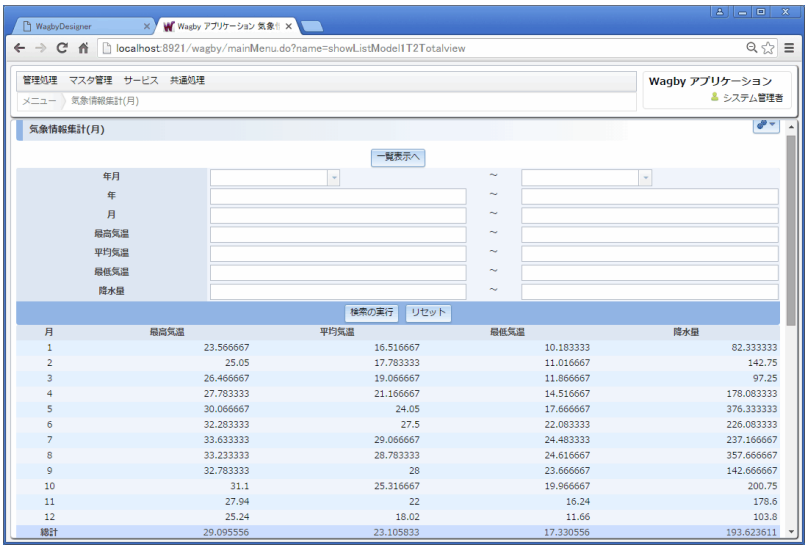


図 ダミーデータ

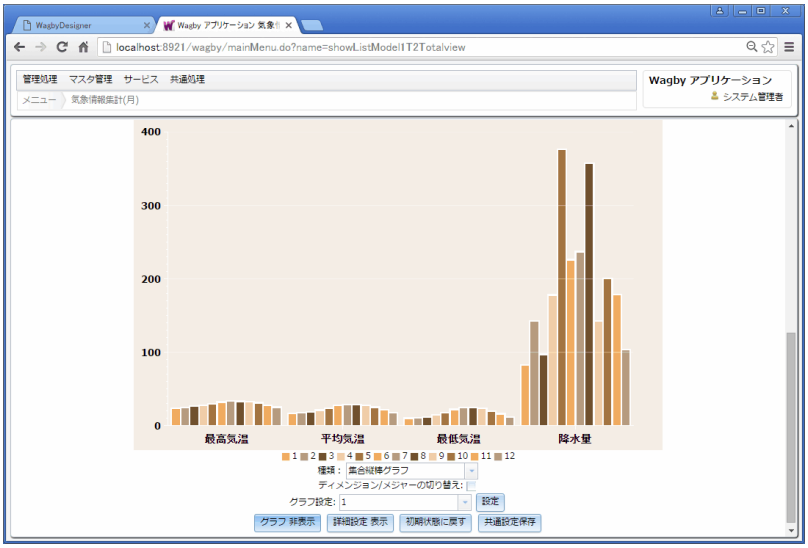


図 グラフの初期表示

ディメンジョンとメジャーの切り替え

はじめに「ディメンジョンとメジャーの切り替え」を指定します。X軸が月に切り替わりました。

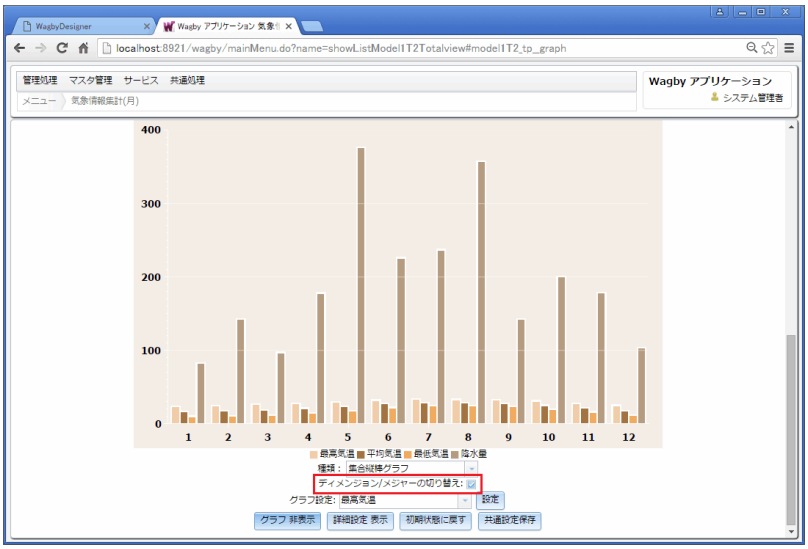


図 ディメンジョンとメジャーを切り替える

グラフの種類（全体）

グラフの種類を指定することができます。この変更は、グラフ全体に及びます。

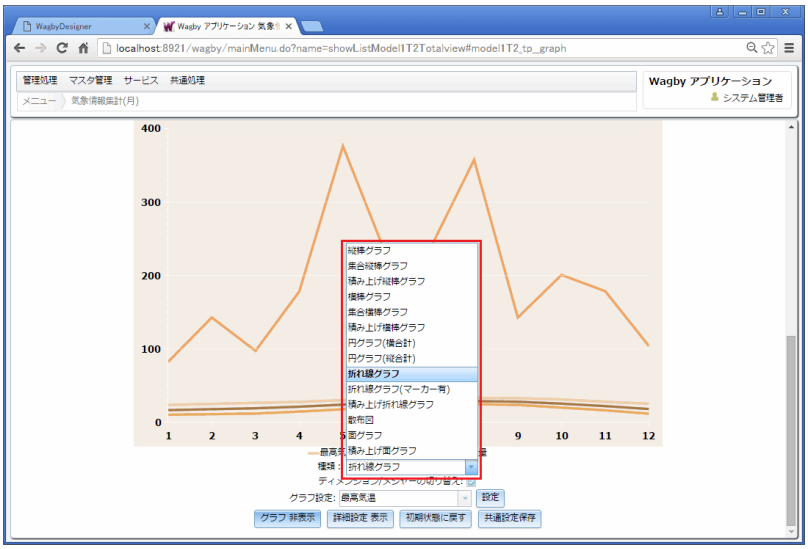


図 グラフの種類の指定（全体）

グラフの設定（個別）

ここでは「降水量」だけを個別に設定する方法を紹介します。「グラフ設定」から降水量を選択します。

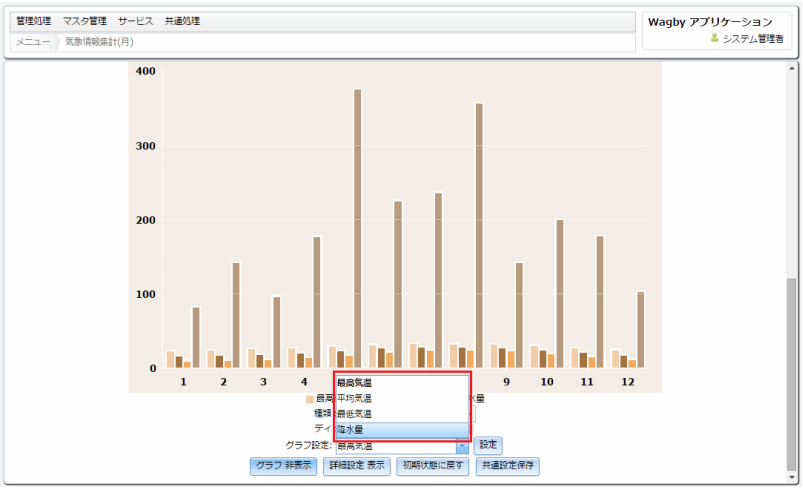


図 降水量を個別に設定する

グラフ設定詳細ダイアログが開きます。

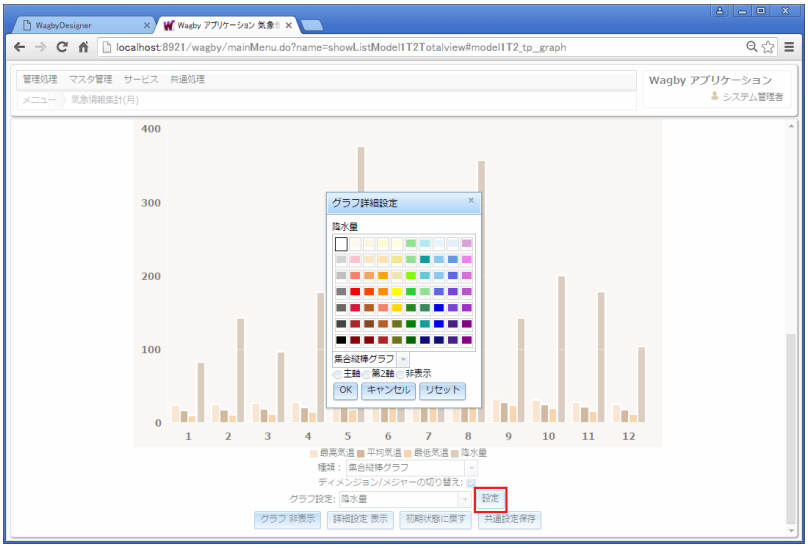


図 グラフ設定詳細ダイアログ

色、グラフの種類、そして軸を指定できます。降水量は温度と別目盛りとするため「第二軸」を指定します。

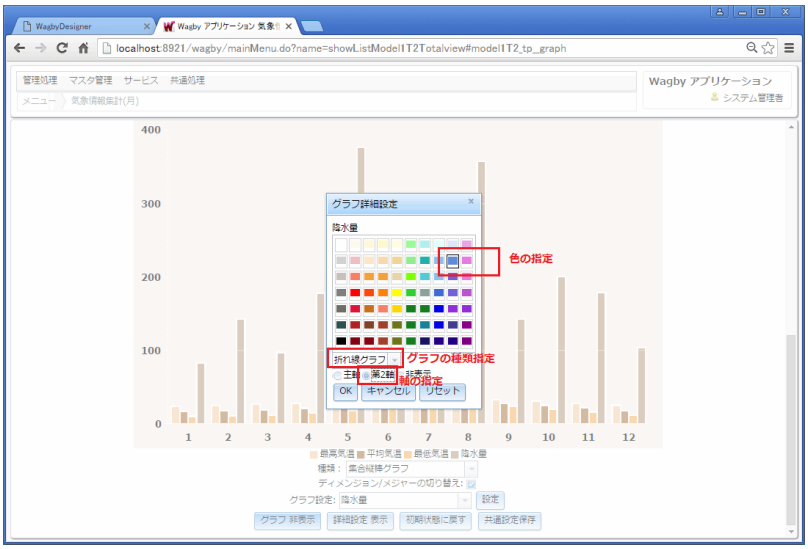


図 グラフ詳細設定を行う

降水量だけが折れ線グラフとなりました。色も変更され、降水量のための目盛りも用意されました。

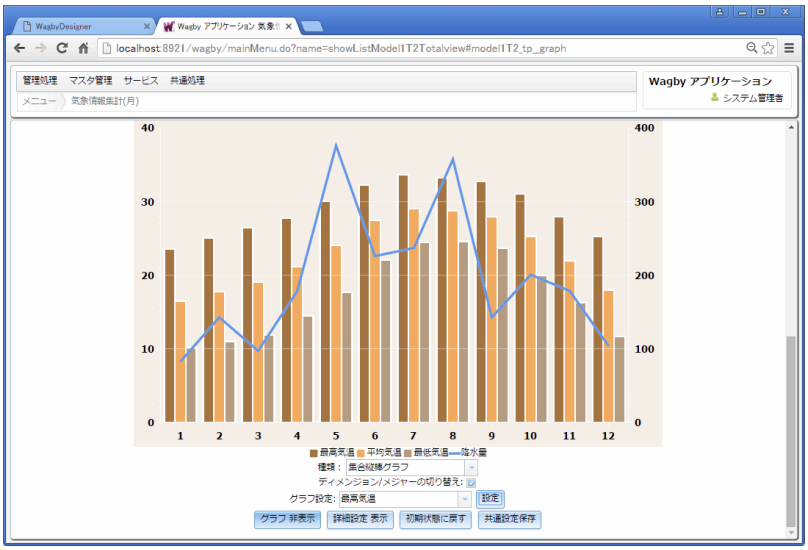


図 降水量だけが折れ線グラフとなった

詳細設定

「詳細設定 表示」ボタンを押すと、グラフの色、大きさ、フォントサイズなどを指定する欄が用意されます。

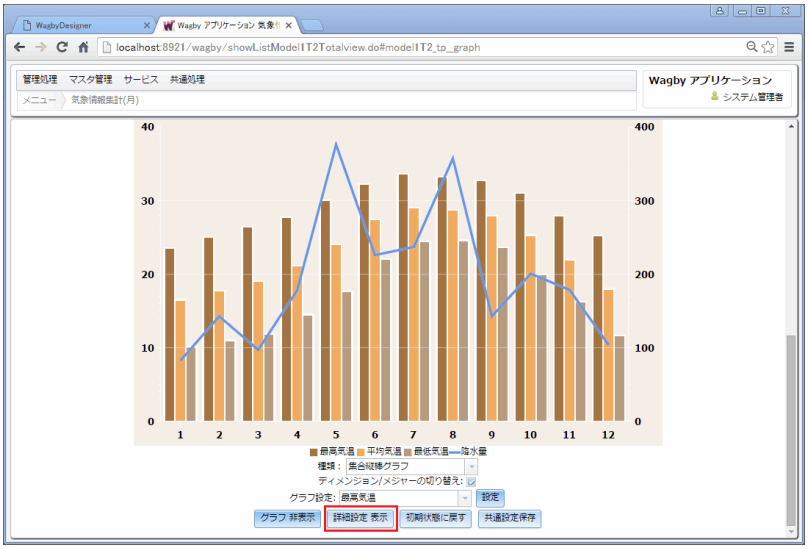


図 詳細設定表示ボタン

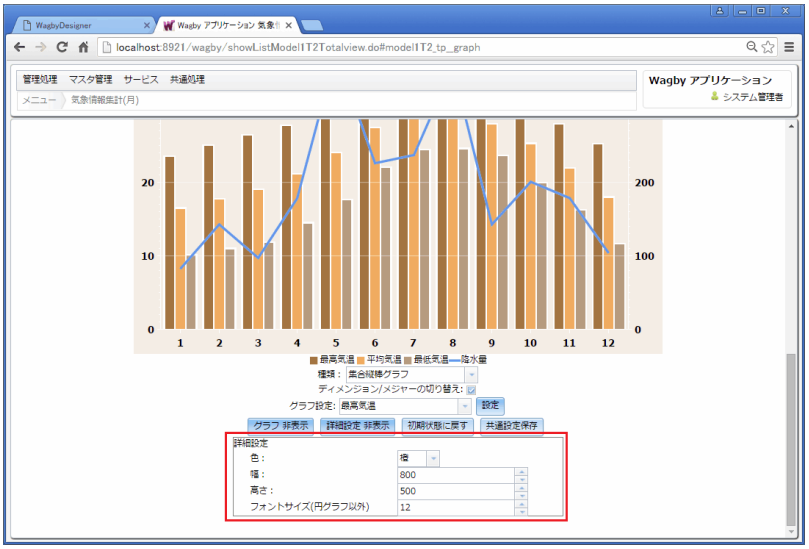


図 詳細設定欄が表示される

各設定について説明します。

色

グラフ表示色を指定します。青、橙、青緑、赤、紫、緑から指定することができます。

幅と高さ

それぞれピクセル値で指定します。単位は不要です。数値のみを入力します。

フォントサイズ

円グラフ用と、それ以外の二つの設定欄を用意しています。

最大表示文字数

円グラフで有効です。"3" と指定すると3文字まで表示され、4文字目以降は "..." となります。図に例を示します。

色、幅、高さを変更した例を示します。

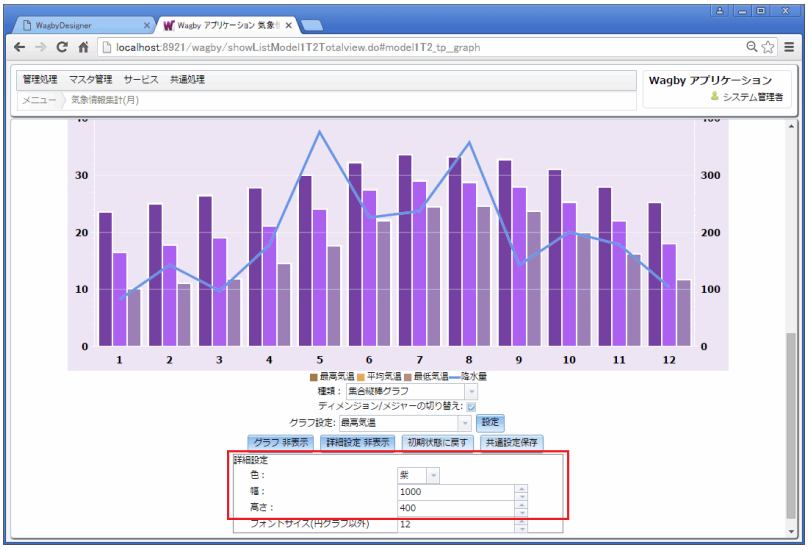


図 色、幅、高さを変更した例

グラフを非表示にする

　グラフの表示と非表示を切り替えることができます。「グラフ非表示」ボタンを押下すると、グラフは隠されます。

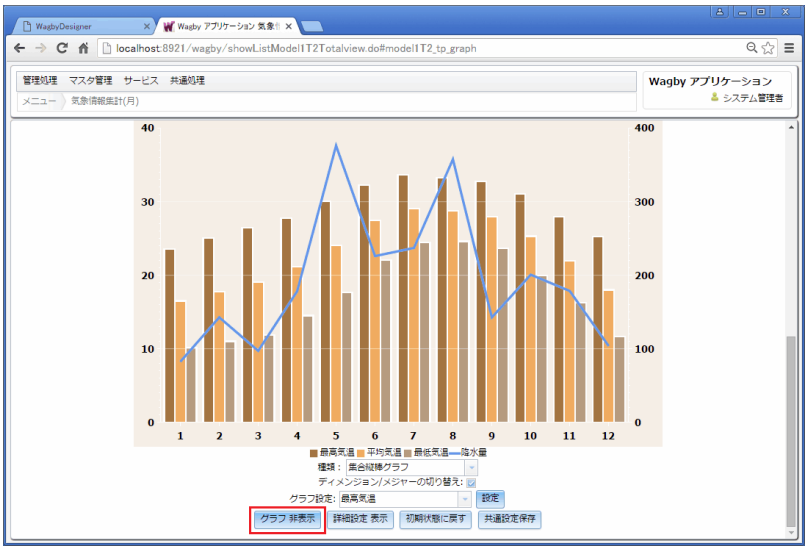


図 グラフ非表示ボタン

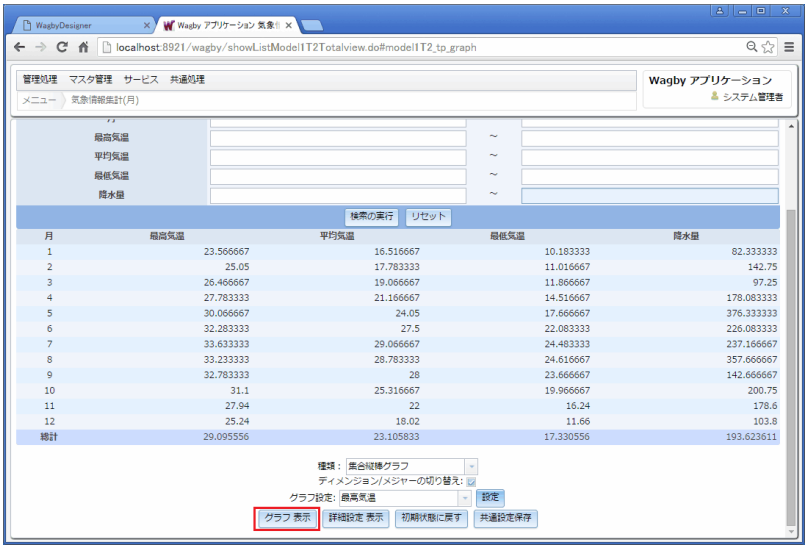


図 グラフが隠された

設定を保存する

共通設定

　利用者は、グラフの設定を保存することができます。特に管理者は、保存した設定を一般利用者の初期値として共通利用させることができます。

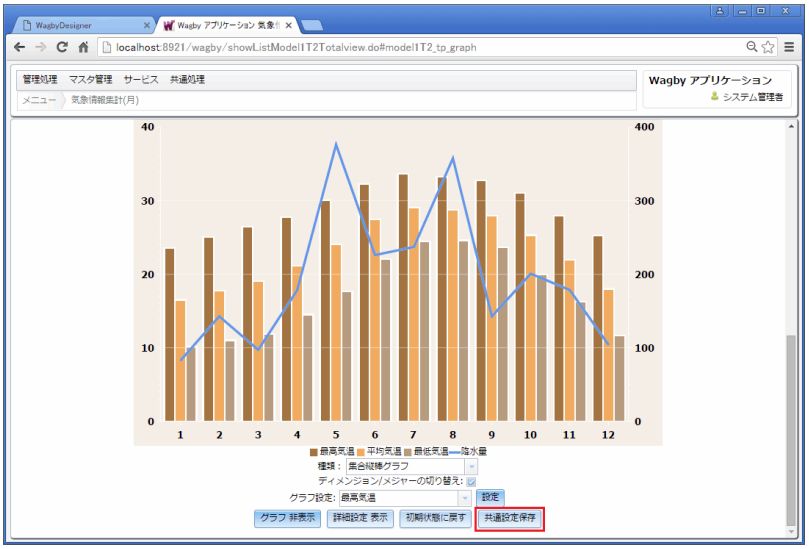


図 共通保存設定ボタン

管理者がグラフを設定し、「共通設定」として保存した例です。

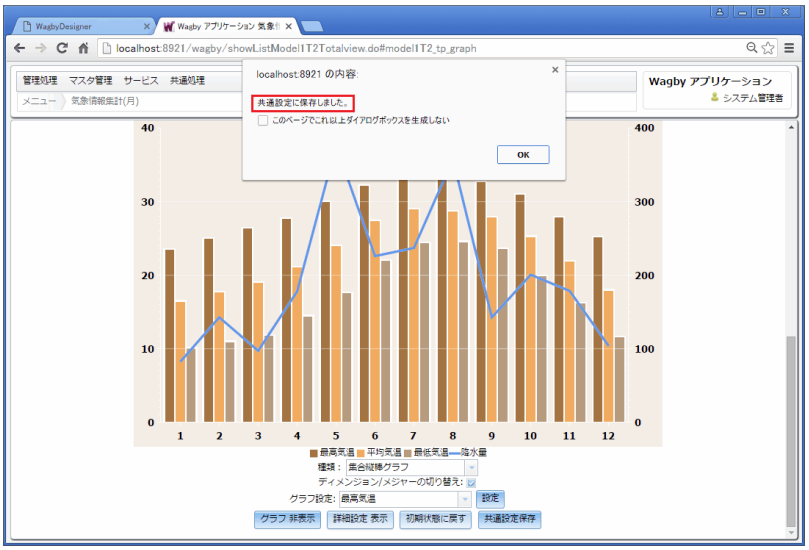


図 共通設定として保存した

グラフ設定

一般利用者および管理者が保存したグラフ設定は「グラフ設定」モデル内に保存されています。このモデルは通常、一般利用者が閲覧・更新することはありません。システムが自動的に管理するように設計されています。管理者のみ、このモデルの内容を確認できます。

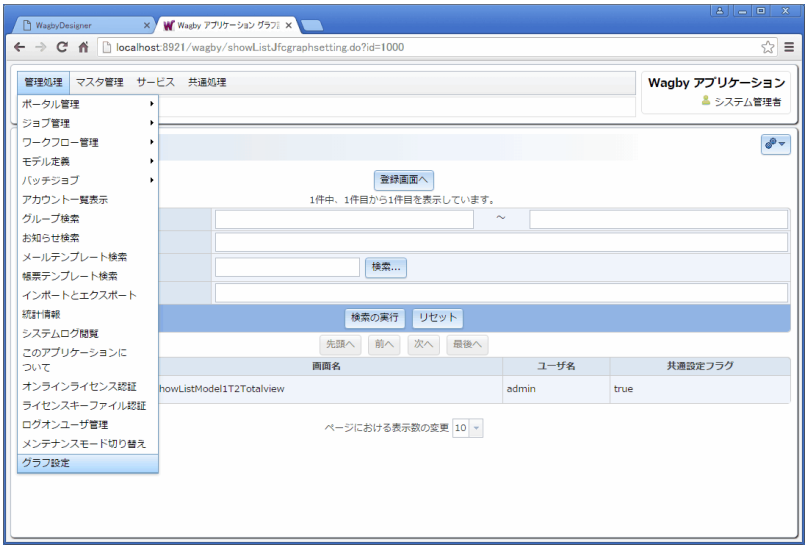


図 グラフ設定モデル

設定したグラフの情報が保存されています。

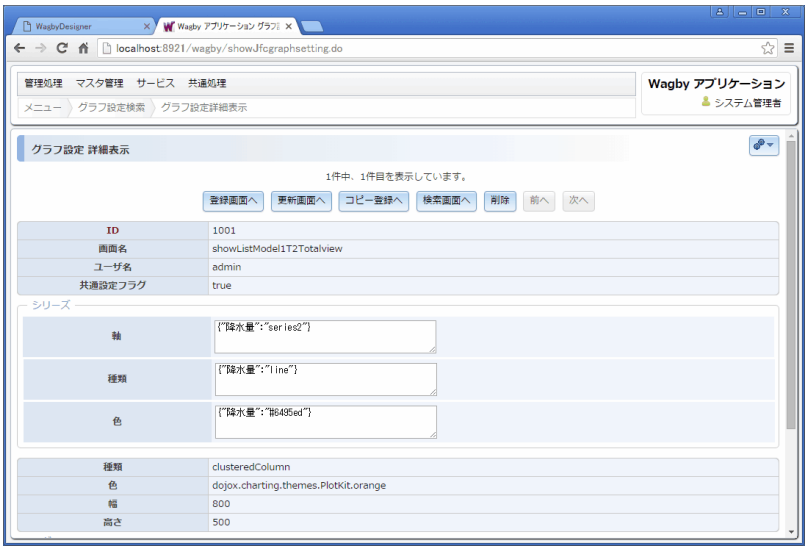


図 設定したグラフ情報

共通設定の利用

一般利用者でログオンすると、管理者が共通設定として保存したグラフ設定を初期値として再利用することができるようになっています。

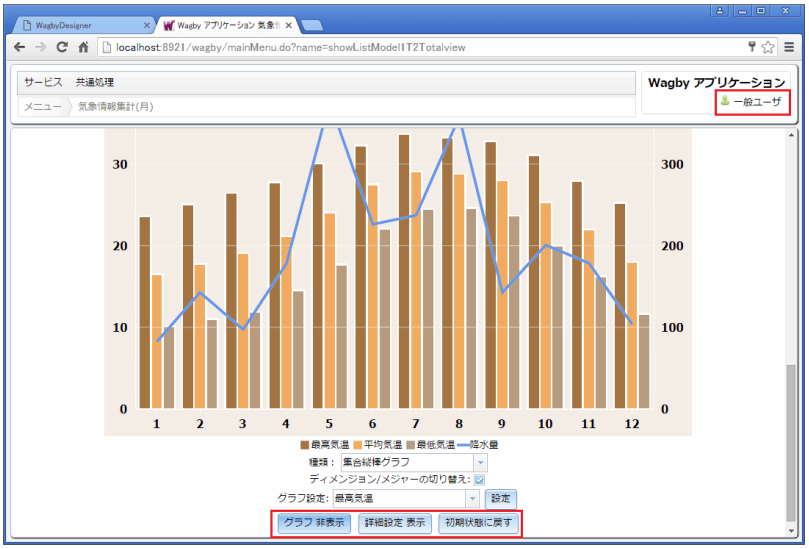


図　共通設定を利用した

一般利用者はさらに、この設定を変更して、個別の設定を行うこともできます。（変更した設定は「グラフ設定」モデルに保存されますが、その利用者専用となります。）

初期状態に戻す

個別設定から共通設定に戻す場合は「初期状態に戻す」ボタンを押下します。

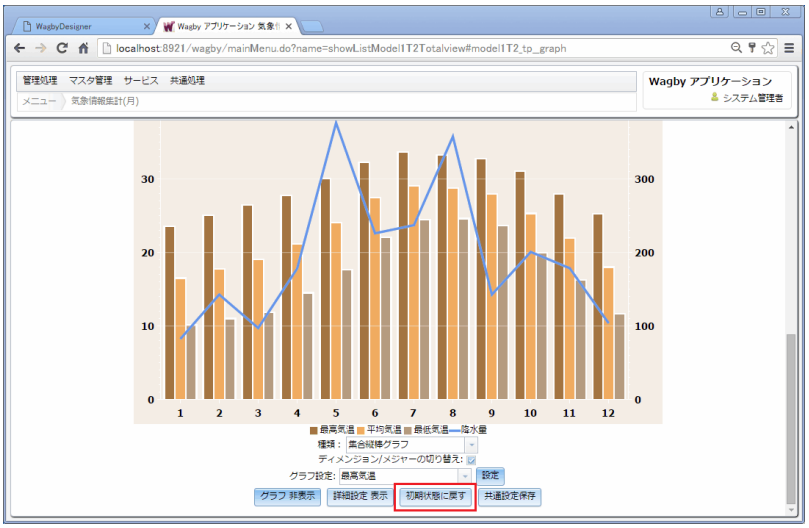


図　初期状態に戻す

ディメンジョンの並び替え

モデル参照項目をディメンジョンに指定した場合、データの並び順は参照モデルのソートルールで指定した並び順になります。このとき、グラフのX軸の並びを指定することができます。

- 「ディメンジョンの並び替えを行わない」が無効の場合[標準] － グラフのX軸を数字として並べ替えます。
- 「ディメンジョンの並び替えを行わない」を設定する － 集計表の並びとグラフのX軸の並びが合致します。

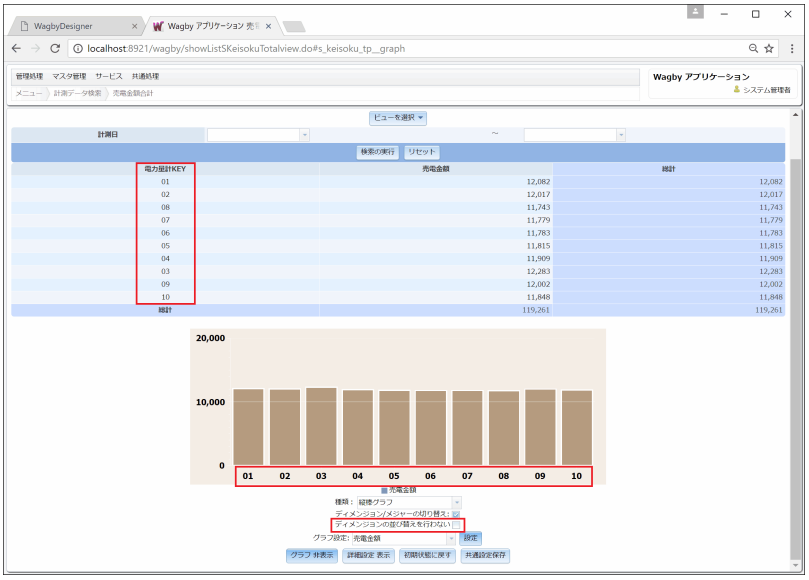


図 ディメンジョンの並び替えを行なう

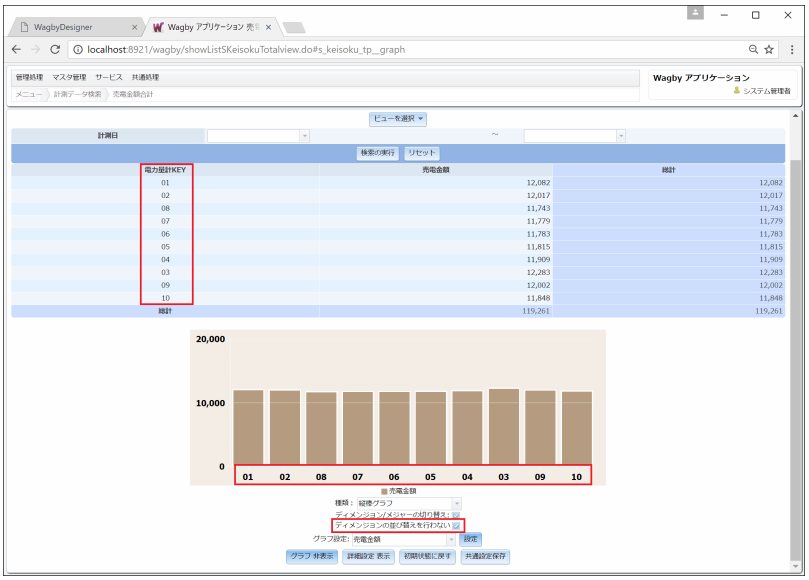


図 ディメンジョンの並び替えを行わない

ディメンジョンを分類する

例

ここでは、ある自治体業務の例を示します。年齢区分毎に給付金を支給する業務において、支給総額をグラフで表示します。

元のデータは年齢(0～100歳)毎に値を持ちます。しかしこれを集計すると、ディメンジョンが0～100と細かくなります。これを3つのカテゴリ「18歳未満」「19歳から65歳未満」「65歳以上」に分類しています。

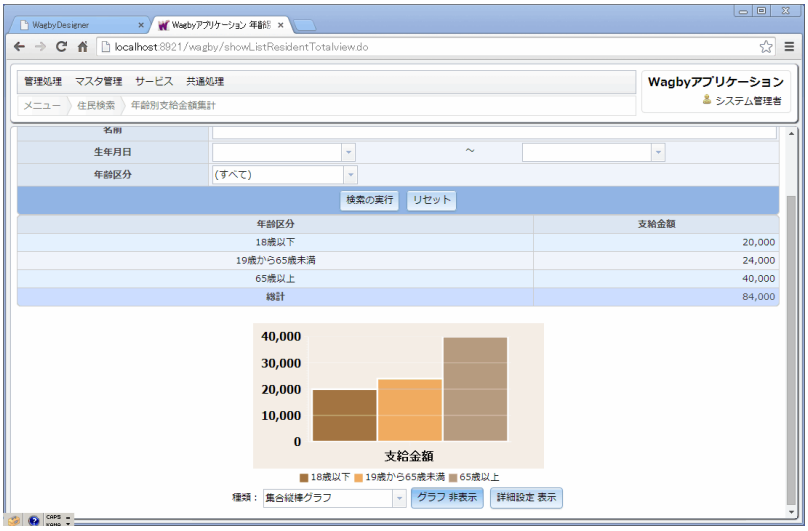


図 年齢区分毎支給総額グラフ（カテゴリの利用）

定義方法

二つのモデル「年齢区分マスタ」「住民」を用意します。

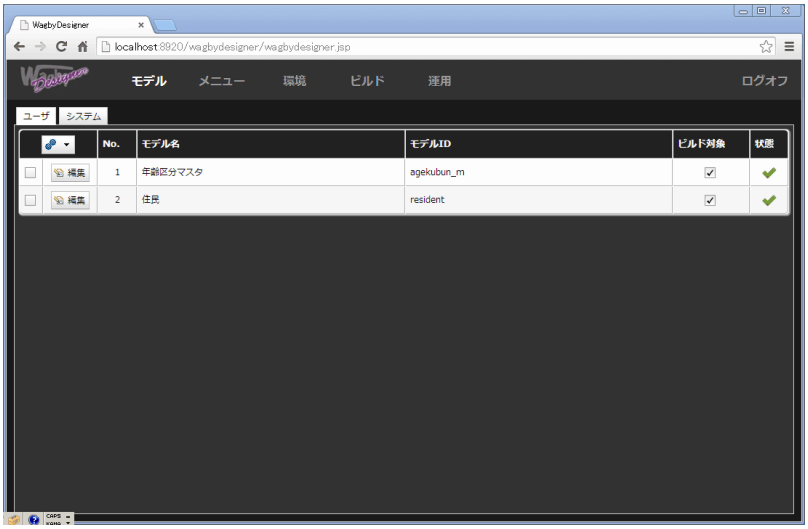


図 二つのモデルを定義する

年齢区分マスタは「18歳未満」「19歳から65歳未満」「65歳以上」の 3 種類を用意します。

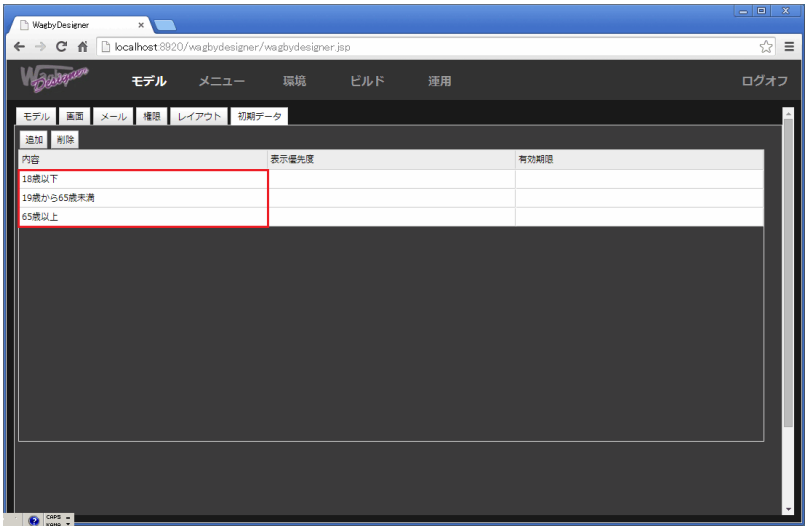


図 年齢区分マスタを用意する

住民モデルの定義を図に示します。ディメンジョンは（「年齢」ではなく）「年齢区分」項目とします。メジャーは「支給金額」項目です。

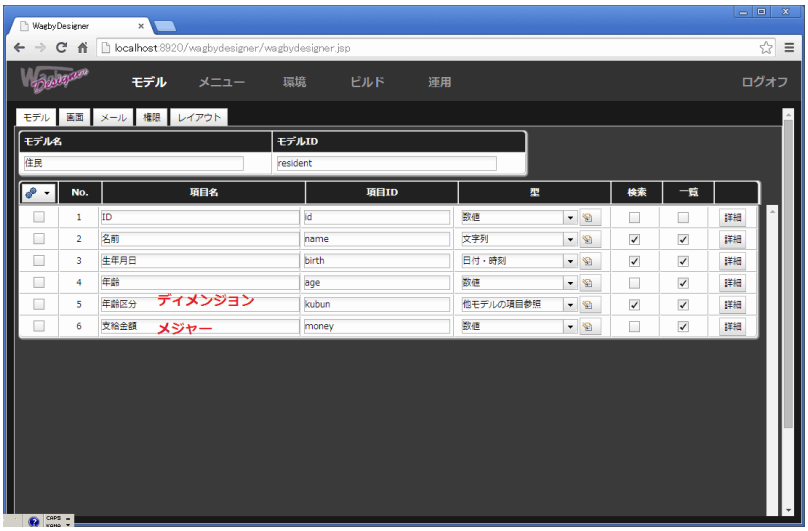


図 住民モデルの定義

年齢は自動計算で求めます。データベースには保存しないとしています。

```
DATEDIF(${birth}, TODAY() ,"Y")
```

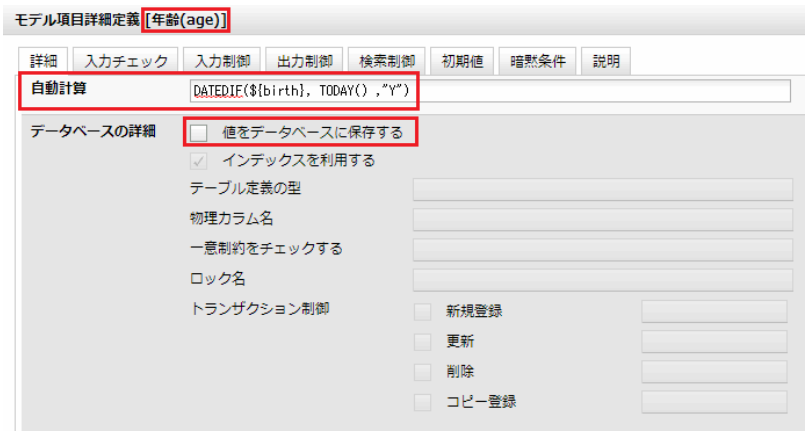


図 年齢の計算式を設定する

年齢区分項目は、図に示した年齢区分マスタを参照します。ソートルールをIDの昇順とします。

図 年齢区分項目のモデル参照設定

年齢区分項目の値は、年齢によって自動的に求めます。これがカテゴライズ操作になります。

```
IF(${age}<=18, 1, IF(${age}<65, 2, 3))
```

この式は「18歳以下なら "1" を、65歳未満なら "2" を、それ以外は "3" を設定する」というものです。求められる値 "1", "2", "3" はそれぞれ、年齢区分マスタモデルの ID に相当します。これもデータベースには保存しないとします。

図 年齢区分項目の計算式を設定する

年齢区分項目は "未選択" という選択肢を用意しません。

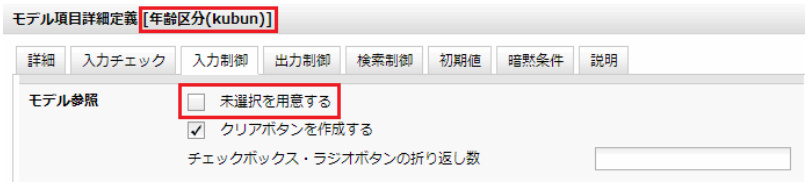


図 未選択を用意しない

「画面>ビュー>集計ビュー」を開きます。集計ビューは複数、用意することができます。ここでは1番の集計ビューを定義します。

「画面を作成する」をチェックし、ページタイトルを設定します。ディメンジョン1は年齢区分項目を指定します。

さらに、グラフ出力を指定します。ここでは種類を「集合縦棒グラフ」とし、高さと幅をピクセル値で指定します。また、合計は縦軸のみとします。

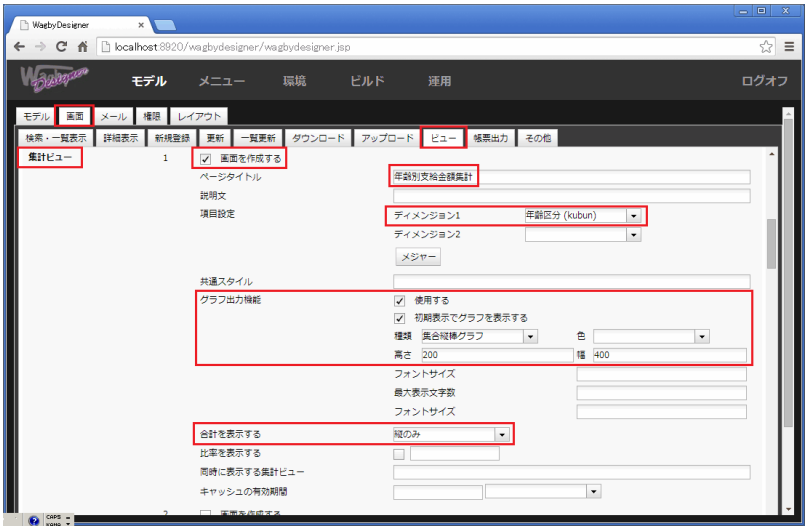


図 集計ビューとグラフの設定

「メジャー」のボタンを押下し、支給金額項目をメジャーと指定します。

図 メジャーの設定

見栄えの調整

共通スタイル

例

集計ビューの表示項目数が多い場合、全体のフォントサイズを指定して調整することができます。 図の例では、文字の大きさ（font-size）を50%と指定しています。

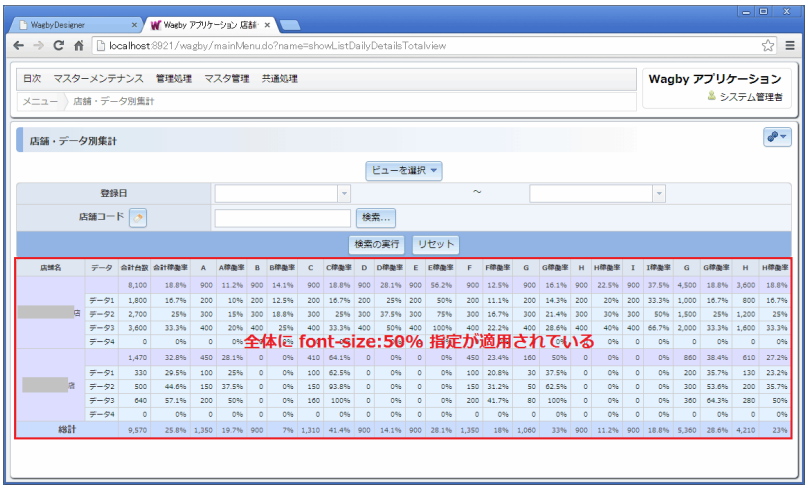


図 全体のフォントサイズを指定した例

定義方法

「画面>ビュー>集計ビュー」を開きます。「共通スタイル」にスタイル（CSS）を指定します。

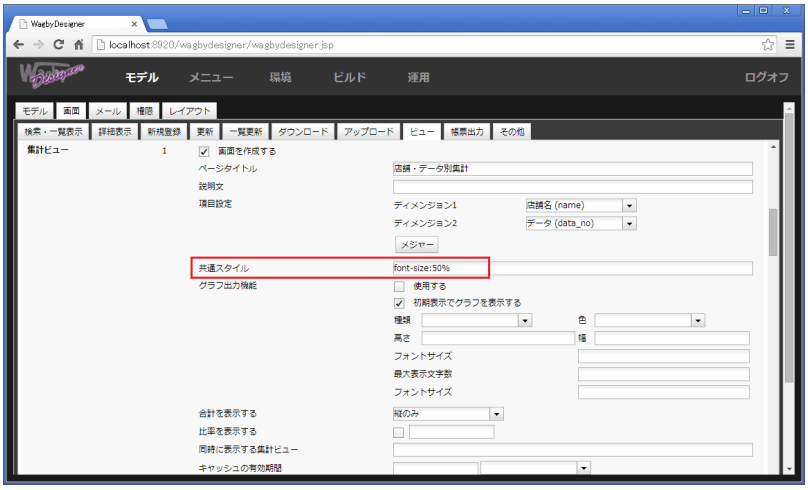


図 共通スタイルの指定

メジャー項目毎のラベル、スタイル設定

例

平均の項目について、見出しの改行と、色を指定した例を図に示します。

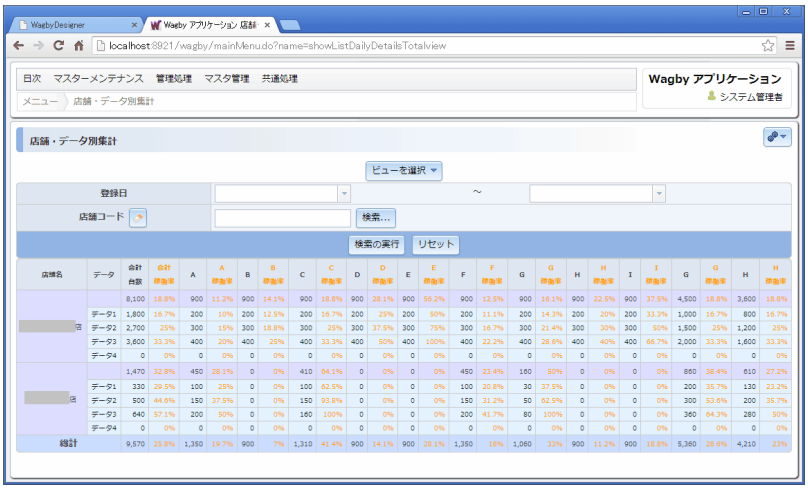


図 見出しの改行と色を指定した例

定義方法

「画面>ビュー>集計ビュー」を開きます。「メジャー」のボタンを押下し、「ラベル」と「スタイル」をそれぞれ指定します。ラベル部分にはHTMLタグを直接、記述することができます。

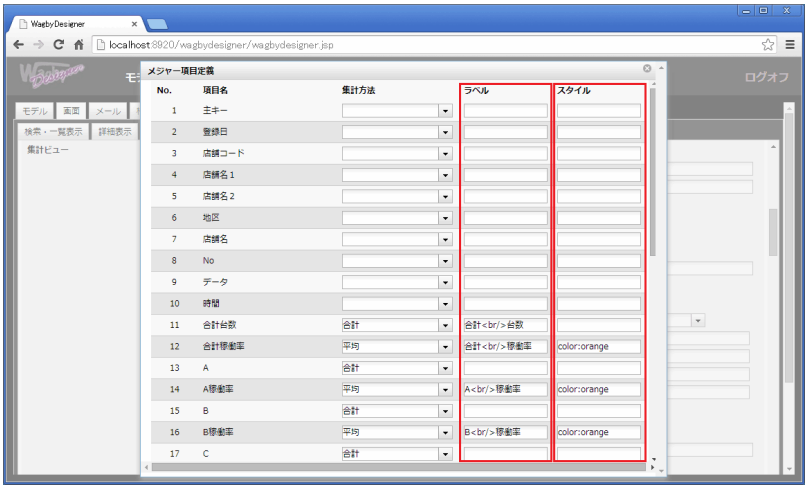


図 メジャー項目毎のラベルとスタイルの指定

一覧表示画面にグラフを表示する

折れ線グラフ

一覧表示項目に含まれる「月度」と「気温」項目の値を使って、折れ線グラフを表示させてみます。グラフに出力させるため、気温項目は数値型となっています。

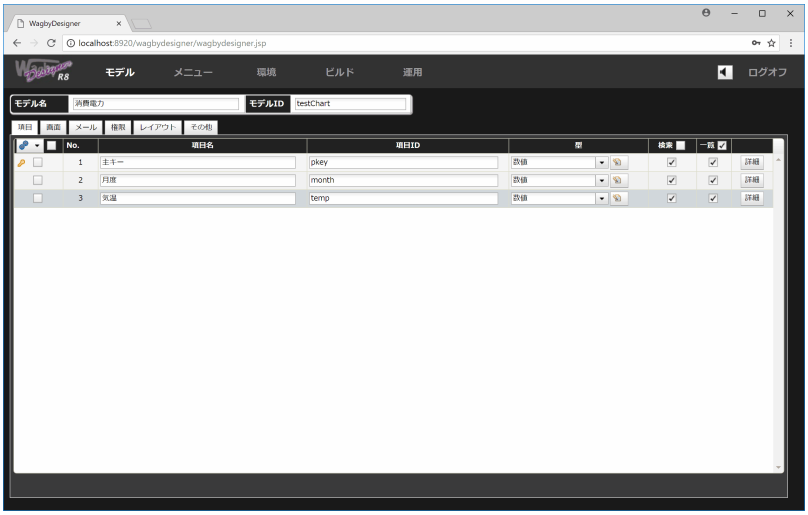


図 サンプルモデル

「気温」項目のモデル項目詳細定義ダイアログを開きます。出力制御 〉 一覧画面 〉 チャートの設定を行います。

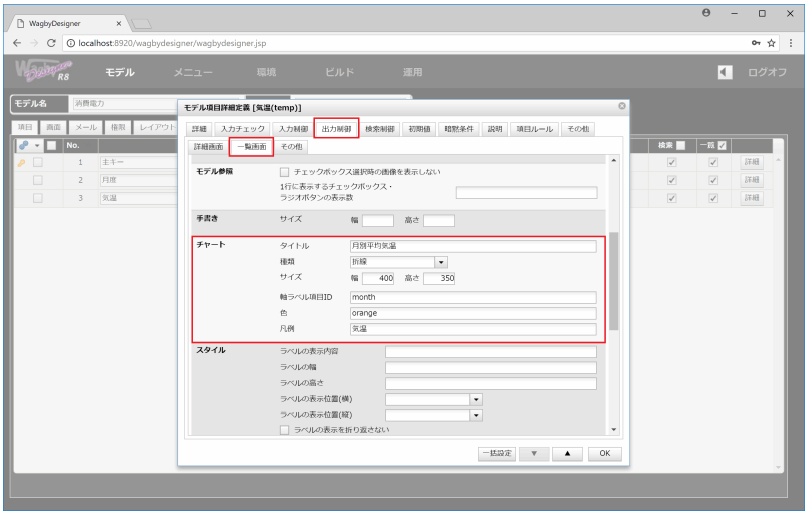


図 チャートの設定

設定項目	説明	設定例
タイトル	グラフのタイトルを指定します。	月別平均気温
種類	グラフの種類を指定します。縦棒、横棒、折線、円、ドーナツ、散布図、レーダーを指定できます。（散布図はカスタマイズが必要です。後述します。）	折線
サイズ	表示するグラフの幅と高さをピクセル値で指定します。	400,350
軸ラベル項目ID	軸となる項目IDを指定します。一覧表示に含まれる項目を指定してください。	month
色	HTMLで利用できる色を指定します。色名またはシャープ文字ではじめる16進数表記が利用できます。	orange
凡例	グラフ内に凡例の情報を含めることができます。	気温

検索画面と一覧表示画面を同時に表示する設定とします。この設定により、検索条件で絞り込んだ値に連動してグラフの表示も変わります。

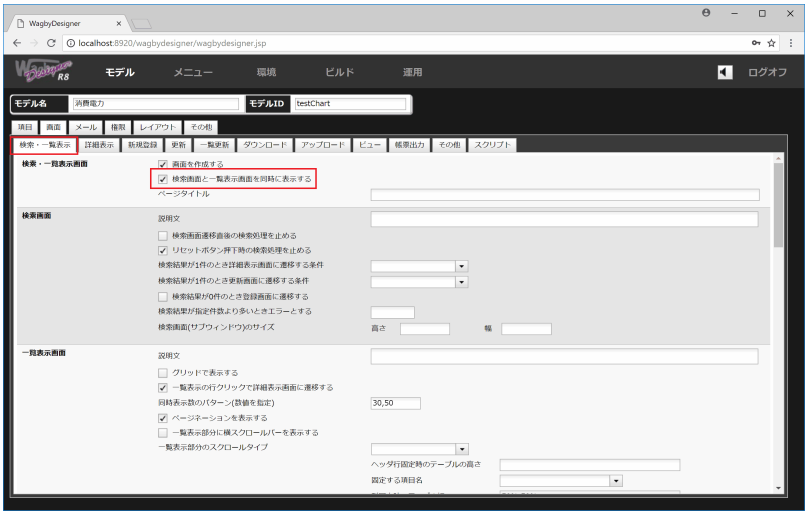


図 検索画面と一覧表示画面を同時に表示する

グラフを表示させる場合、データ取得のために REST API を利用します。そのため、このモデルの REST API を有効にしてください。

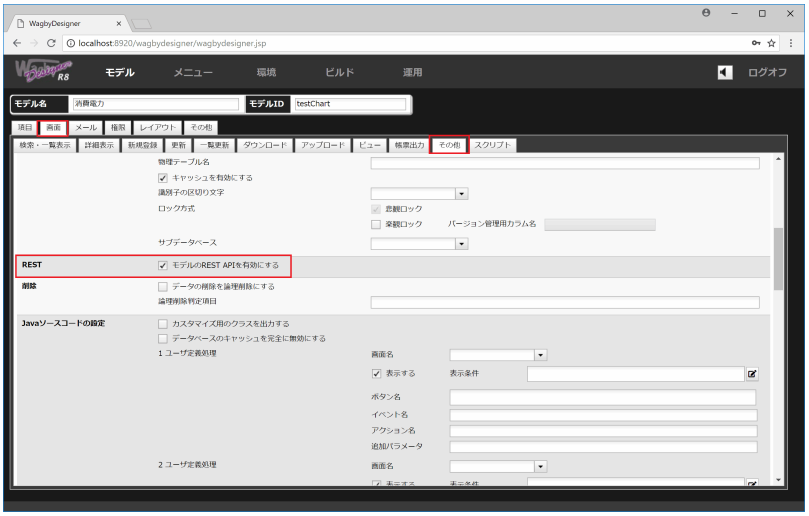


図 RESTAPIを有効にする

実行例

一覧表示部の下部にグラフが表示されます。

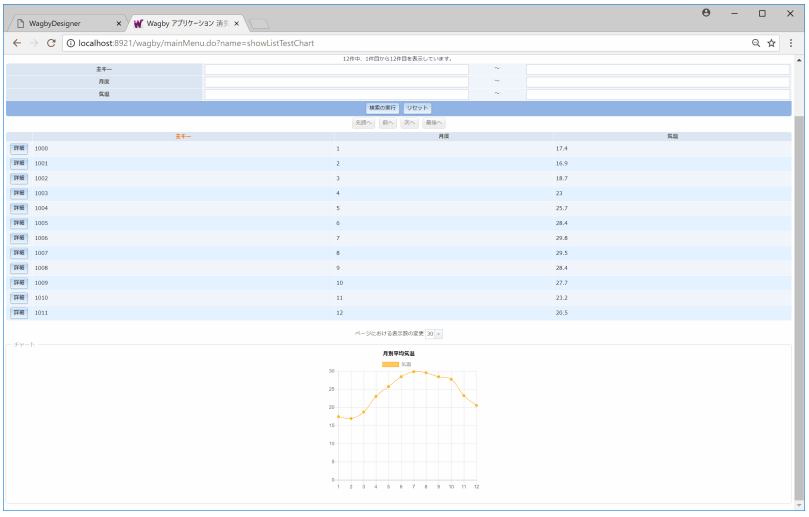


図 折れ線グラフ(全体)

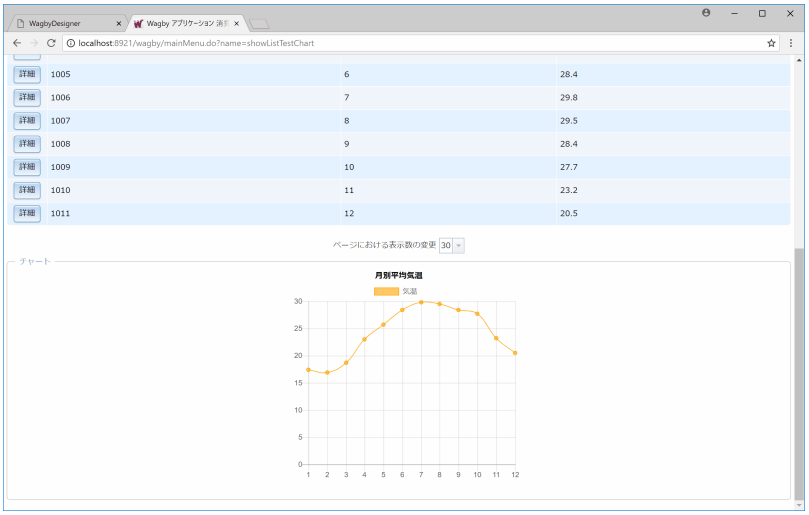


図 折れ線グラフ(拡大)

検索条件を指定すると、絞り込まれた結果でグラフが表示されます。

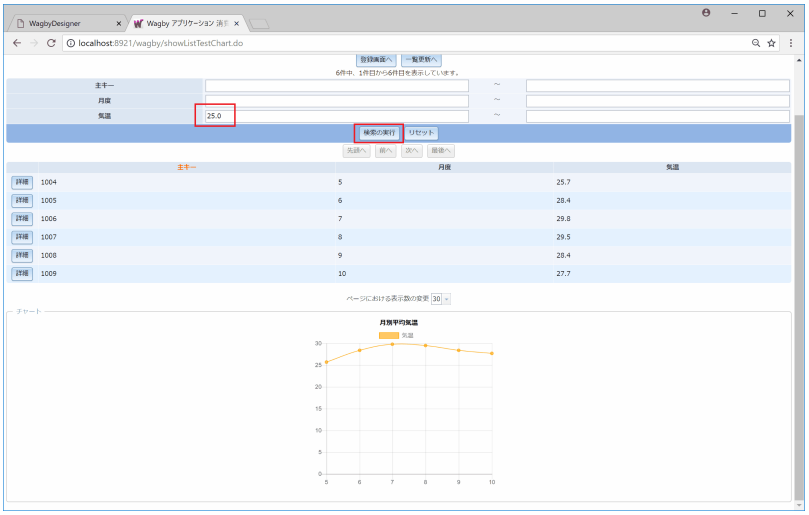


図 検索結果に連動してグラフ表示が変わる

棒グラフ

「消費電力」項目を加えてみます。

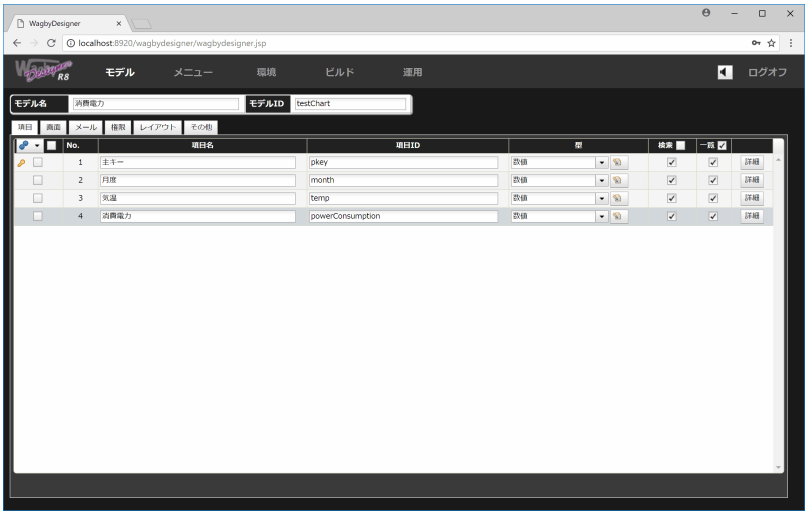


図 消費電力項目を加える

この項目は縦棒グラフとして出力する設定を行います。

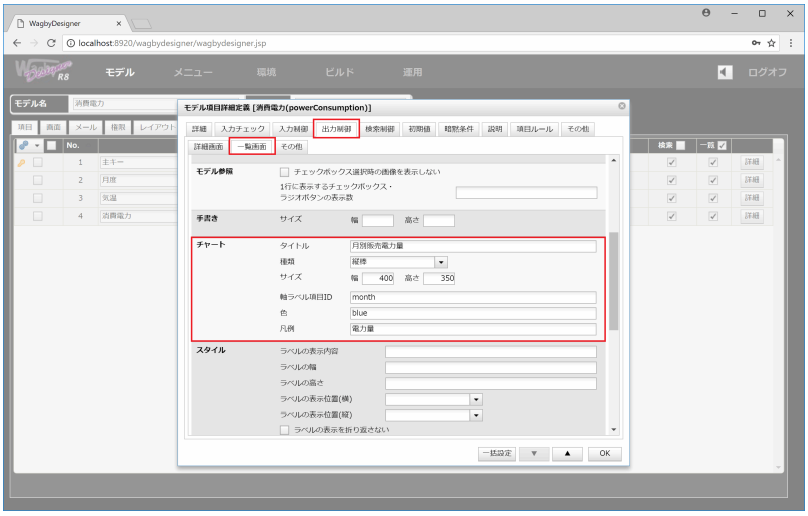


図 縦棒の設定

実行例

グラフは一つのテーマごとに並べて表示されます。（並び順は項目の定義順となります。）

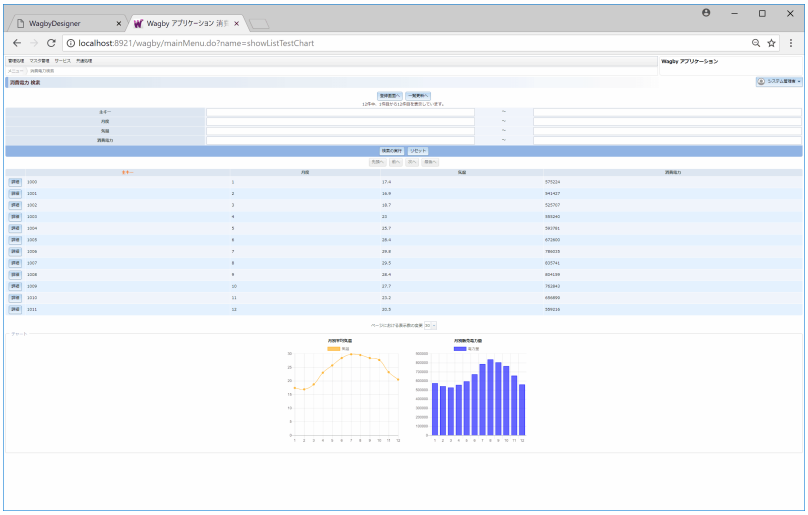


図 折れ線グラフと縦棒グラフの表示(全体)

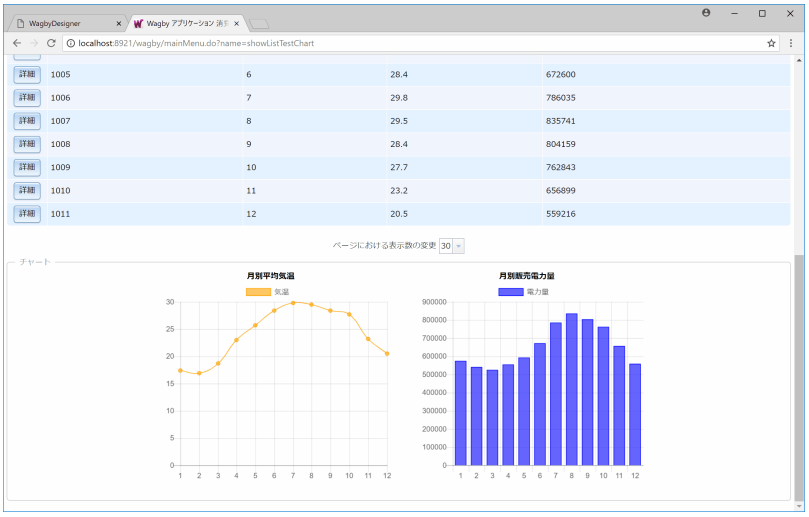


図 折れ線グラフと縦棒グラフの表示(拡大)

[応用] JavaScript を使ってグラフをカスタマイズする

このグラフは Web ブラウザ上で動作する JavaScript によって描画されています。開発者は JavaScript を追加することでカスタマイズを行うことができます。詳細は "一覧表示部のグラフのカスタマイズ" をお読みください。

[応用] 一覧表示グリッドでグラフを表示する

一覧表示グリッド形式でのグラフ表示は JavaScript によるカスタマイズが必要です。詳細は "一覧表示部のグラフのカスタマイズ" をお読みください。

[応用] 散布図

散布図の表示は JavaScript によるカスタマイズが必要です。詳細は "一覧表示部のグラフのカスタマイズ" をお読みください。

仕様・制約

- 検索画面と一覧表示画面を同時に表示する設定としてください。この設定により、検索条件で絞り込んだ値に連動してグラフの表示も変わります。
- グラフ表示のためのデータ取得で REST API を使います。グラフを表示するモデルでは、REST API を有効にしてください。
- グラフを帳票（Excel帳票、PDF帳票）に含めることはできません。Webブラウザでの表示用となります。
- 一覧表示の対象項目を「隠し項目」とすることもできます。一覧表示には含まれませんが、グラフには利用できるようになります。

一覧表示画面にグラフを表示する (2) (JavaScript)

準備

このページは リポジトリ ＞ 集計・グラフ ＞ 一覧表示画面にグラフを表示する の説明の続きという構成になっています。はじめにこのページをお読みください。

カスタマイズするファイル

グラフ出力のカスタマイズは、MyShowList<モデルID>.js に行います。カスタマイズしたファイルをcustomize/webapp/<モデルID> フォルダに保存することで、ビルド時に反映されます。（ファイル名に含めるモデルIDは、キャメル記法を適用してください。）

表示位置の変更、ラベルの作成

次のカスタマイズを行ってみます。

- 検索条件部と、一覧表示部の間にグラフが表示されるように位置を移動する。
- 凡例を非表示とする。
- X軸とY軸にラベルを用意する。

このカスタマイズにより、表示が次のように変わります。

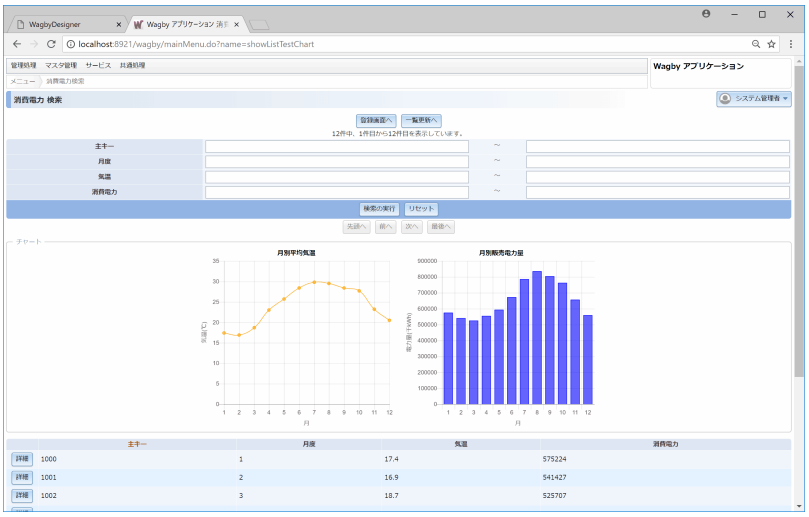


図 月別平均気温（折線グラフ）と月別販売電力量（棒グラフ）全体

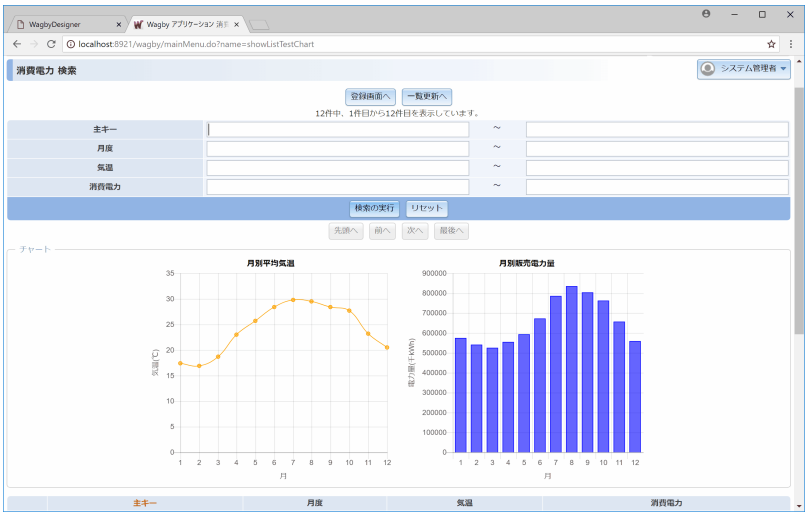


図 月別平均気温（折線グラフ）と月別販売電力量（棒グラフ）拡大

カスタマイズコードは次のようになります。モデルIDを testChart とし、ファイルをMyShowListTestChart.js とします。

```
/* 表示位置の設定 */
function beforeDrawChart(dom, domAttr, domClass, domConstruct, domStyle, query, registry,
 request) {
  var chartContainer = query("div.display_chart")[0];
```

```
  var listContainer = query("div[class^='display_table_showlistdiv'")[0];
  if (!chartContainer || !listContainer) {
    return;
  }
  domConstruct.place(chartContainer, listContainer, "before");
}
/* 気温グラフ(折線)の設定 */
function getTempChartOptions(defaultChartOption) {
  var option = defaultChartOption;
  option.legend = {
    display: false,
  }
  option.scales.xAxes[0].scaleLabel.labelString = "月";
  option.scales.yAxes[0].scaleLabel.labelString = "気温(℃)";
  option.scales.yAxes[0].ticks = {
    min: 0,
    max: 35,
  }
  return option;
}
/* 電力量グラフ(縦棒)の設定 */
function getPowerConsumptionChartOptions(defaultChartOption) {
  var option = defaultChartOption;
  option.legend = {
    display: false,
  }
  option.scales.xAxes[0].scaleLabel.labelString = "月";
  option.scales.yAxes[0].scaleLabel.labelString = "電力量(千kWh)";
  return option;
}
```

- グラフ表示はオープンソースの Chart.js ライブラリを使っています。（集計ビューで使っている Dojotoolkit のグラフライブラリとは別になっています。）
- グラフ描画前に beforeDrawChart 関数が呼び出されます。上の例では一覧表示のテーブルを class 名 display_table_showlistdiv を使って取得し、グラフ表示部をテーブルの前に挿入することで動的に表示位置を変更しています。引数で渡される domConstruct オブジェクトは Dojotoolkit が提供します。
- get<グラフ表示対象項目ID>CharOptions関数の引数 defaultChartOption オブジェクトを操作することができます。操作後のオブジェクトを return で返します。上の例では凡例表示を無効とし、X軸とY軸のラベル文字をそれぞれ設定しています。また ticks を変更することで最小値と最大値を指定することもできます。
- 他にどのような設定が行えるかは Chart.js のドキュメントを参照してください。

取得するデータサイズを変更する

取得するデータサイズは、一覧表示画面の表示数と同じとなっています。 これを調整する場合は、カスタマイズする MyShowList<モデルID>.js ファイルに次のコードを加えてください。

```
function setPagesize(conditions) {
  conditions["pagesizejshparam"] = 100;//100件とする場合
}
```

一覧表示グリッドでグラフを表示する

一覧表示グリッドにグラフを出力するサンプルコードを説明します。

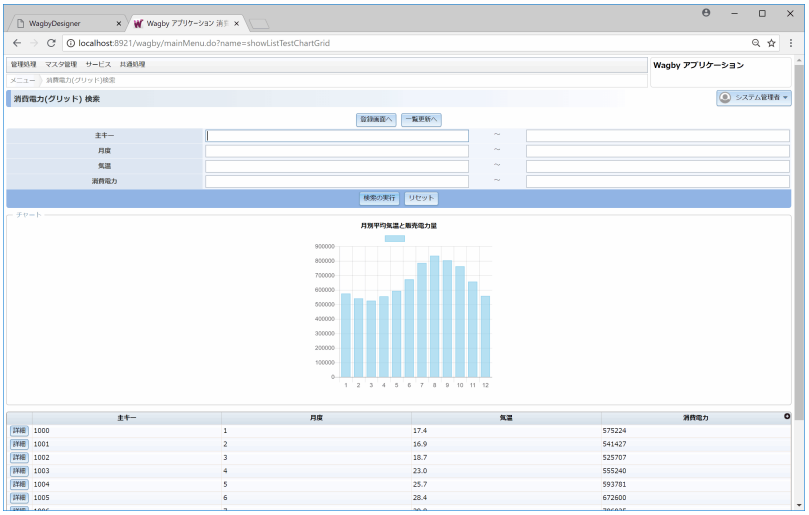


図 一覧表示グリッド形式画面にグラフを出力する

定義方法

モデルIDを testChartGrid とし、グリッド用のモデルを定義します。

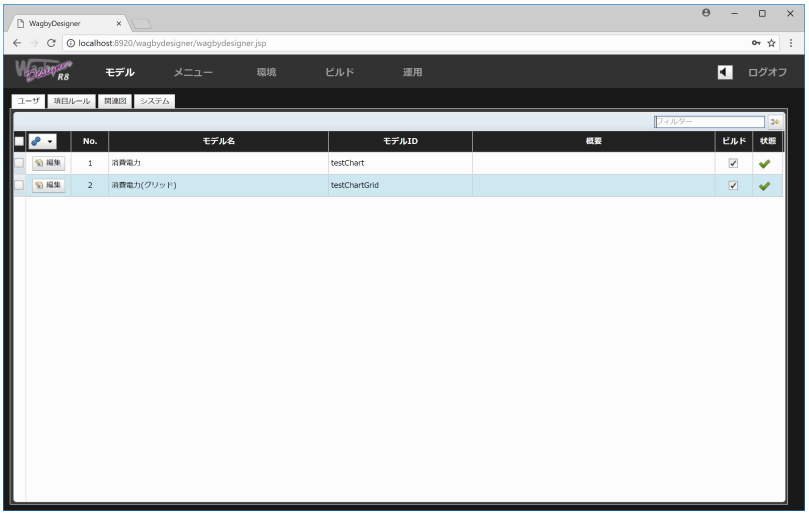


図 グリッド用モデルを用意する(1)

これはサブモデルとし、（グリッドではない、これまでテストで使ったモデル testChart と）同じ項目とします。

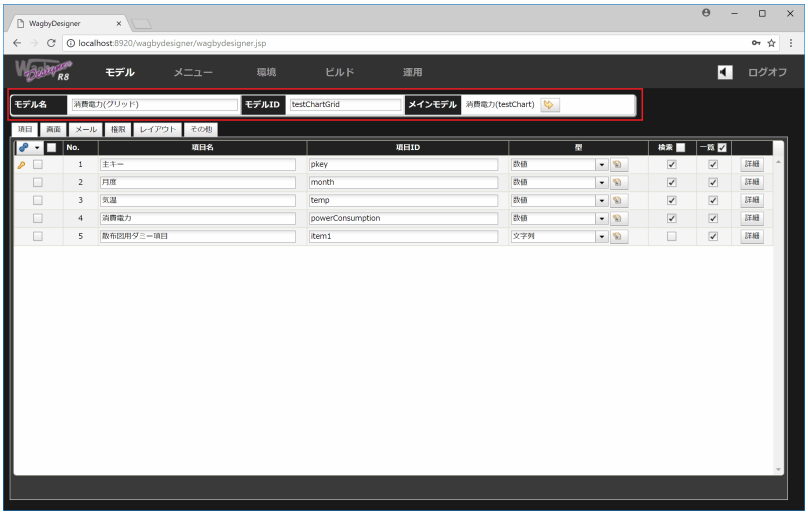


図 グリッド用モデルを用意する(2)

グラフの設定を行います。

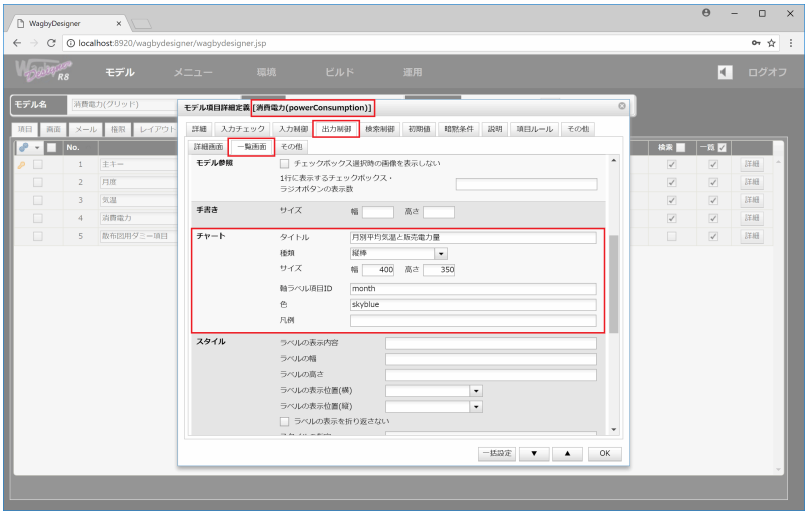


図 グリッド用モデルを用意する(3)

カスタマイズするファイルは MyShowListTestChartGrid.js になります。カスタマイズコードは次のようになります。

```
function beforeDrawChart(dom, domAttr, domClass, domConstruct, domStyle, query, registry,
 request) {
  var chartContainer = query("div.display_chart")[0];
  var form = dom.byId("conditionTestChartGridCp");
  if (!chartContainer || !form) {
    return;
  }
  domConstruct.place(chartContainer, form, "after");
}

function drawChart() {
  require(["dojo/dom", "dojo/dom-attr", "dojo/dom-class", "dojo/dom-construct", "dojo/dom-
style", "dojo/query", "dijit/form/Button", "dijit/registry", "dojo/request"],
  function(dom, domAttr, domClass, domConstruct, domStyle, query, Button, registry, request) {
    beforeDrawChart(dom, domAttr, domClass, domConstruct, domStyle, query, Button, registry,
 request);

    var chartContainer = query("div.display_chart")[0];
    hideChart();
```

```
    var charts = [];

    function showChart(grid) {
      var total = grid.get("total");
      var display = "none";
      if (total > 0) {
        display = "block";
      }
      domStyle.set(chartContainer, "display", display);
      if (display === "block") {
        for (var i=0,l=charts.length; i>l; i++) {
          charts[i].destroy();
        }
        charts = [];
        getChartData(function(data) {
          charts.push(createPowerConsumptionChart(data, dom, domAttr, domClass, domConstruct,
 domStyle, query, Button, registry, request, afterCreatePowerConsumptionChart));
          afterDrawAllChart(charts, dom, domAttr, domClass, domConstruct, domStyle, query,
 Button, registry, request);
        });
      }
    }

    function hideChart() {
      domStyle.set(chartContainer, "display", "none");
    }

    setTimeout(function() {
      var grid = window.getTestChartGridShowListGrid();
      if (grid) {
        grid.grid.on('dgrid-refresh-complete', function(e) {
          showChart(e.grid);
        });
      }
    }, 10);

  });
}
```

一つのグラフにまとめる

上で用意した一覧表示グリッドのグラフをさらにカスタマイズし、一つのグラフに二つの値（気温、販売電力量）をまとめた例です。

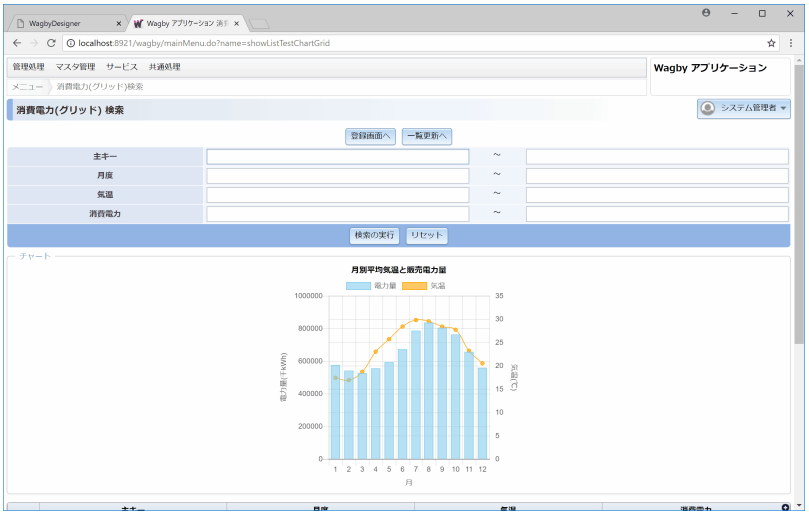


図 一つのグラフにまとめた例

```
function beforeDrawChart(dom, domAttr, domClass, domConstruct, domStyle, query, registry,
 request) {
  var chartContainer = query("div.display_chart")[0];
  var form = dom.byId("conditionTestChartGridCp");
  if (!chartContainer || !form) {
    return;
  }
  domConstruct.place(chartContainer, form, "after");
}

function drawChart() {
  require(["dojo/dom", "dojo/dom-attr", "dojo/dom-class", "dojo/dom-construct", "dojo/dom-
style", "dojo/query", "dijit/form/Button", "dijit/registry", "dojo/request"],
  function(dom, domAttr, domClass, domConstruct, domStyle, query, Button, registry, request) {
    beforeDrawChart(dom, domAttr, domClass, domConstruct, domStyle, query, Button, registry,
 request);

    var chartContainer = query("div.display_chart")[0];
    hideChart();
    var charts = [];

    function showChart(grid) {
      var total = grid.get("total");
      var display = "none";
      if (total > 0) {
        display = "block";
      }
      domStyle.set(chartContainer, "display", display);
      if (display === "block") {
        for (var i=0,l=charts.length; i<l; i++) {
          charts[i].destroy();
        }
        charts = [];
        getChartData(function(data) {
          charts.push(createPowerConsumptionChart(data, dom, domAttr, domClass, domConstruct,
 domStyle, query, Button, registry, request, afterCreatePowerConsumptionChart));
          afterDrawAllChart(charts, dom, domAttr, domClass, domConstruct, domStyle, query,
 Button, registry, request);
        });
      }
    }

    function hideChart() {
      domStyle.set(chartContainer, "display", "none");
    }

    setTimeout(function() {
      var grid = window.getTestChartGridShowListGrid();
      if (grid) {
        grid.grid.on('dgrid-refresh-complete', function(e) {
          showChart(e.grid);
        });
      }
    }, 10);

  });
}

function getPowerConsumptionPowerConsumptionChartDataset(data) {
  return {
    yAxisID: "yAxis-left",
    type: "bar",
```

```
    label: getPowerConsumptionPowerConsumptionChartDatasetLabel(),
    data: getEntityListData(data.entity, 'powerConsumption'),
    backgroundColor: getPowerConsumptionPowerConsumptionChartDatasetBgColor(),
    borderColor: getPowerConsumptionPowerConsumptionChartDatasetBorderColor(),
    borderWidth: getChartBorderWidth(),
  };
}

function getPowerConsumptionPowerConsumptionChartDatasetLabel() {
  return "電力量";
}

function getPowerConsumptionTempChartDataset(data) {
  return {
    yAxisID: "yAxis-right",
    type: "line",
    label: getPowerConsumptionTempChartDatasetLabel(),
    data: getEntityListData(data.entity, 'temp'),
    backgroundColor: getPowerConsumptionTempChartDatasetBgColor(),
    borderColor: getPowerConsumptionTempChartDatasetBorderColor(),
    borderWidth: getChartBorderWidth(),
    fill: false,
  };
}

function getPowerConsumptionTempChartDatasetLabel() {
  return "気温";
}

function getPowerConsumptionTempChartDatasetColor() {
  return "orange";
}

function getPowerConsumptionTempChartDatasetBgColor() {
  return
 Chart.helpers.color(getPowerConsumptionTempChartDatasetColor()).alpha(getChartColorAlpha()).rgbString();
}

function getPowerConsumptionTempChartDatasetBorderColor() {
  return getPowerConsumptionTempChartDatasetColor();
}

function getPowerConsumptionChartDatasets(data) {
  var datasets = [];
  datasets.push(getPowerConsumptionPowerConsumptionChartDataset(data));
  datasets.push(getPowerConsumptionTempChartDataset(data));
  return datasets;
}

function getPowerConsumptionChartOptions(defaultChartOption) {
  var option = defaultChartOption;
  option.scales.xAxes[0].scaleLabel.labelString = "月";
  option.scales.yAxes[0].id = "yAxis-left";
  option.scales.yAxes[0].position = "left";
  option.scales.yAxes[0].scaleLabel.labelString = "電力量(千kWh)";
  option.scales.yAxes[0].ticks = {
    beginAtZero: true,
    maxTicksLimit: 6,
  };
  option.scales.yAxes[1] = {
    id: "yAxis-right",
    position: "right",
    scaleLabel: {
      display: true,
```

```
      labelString: "気温(℃)",
    },
    ticks: {
      beginAtZero: true,
      max: 35,
      min: 0,
    },
  };
  return option;
}
```

- getPowerConsumptionChartDatasets(data)関数で、二種類のデータをひとつのdatasetにまとめています。
- その他、軸のラベル指定などをおこなっています。

散布図

散布図は Designer の設定に加え、JavaScript による追加コードの記述が必要です。散布図を含んだ表示例を示します。

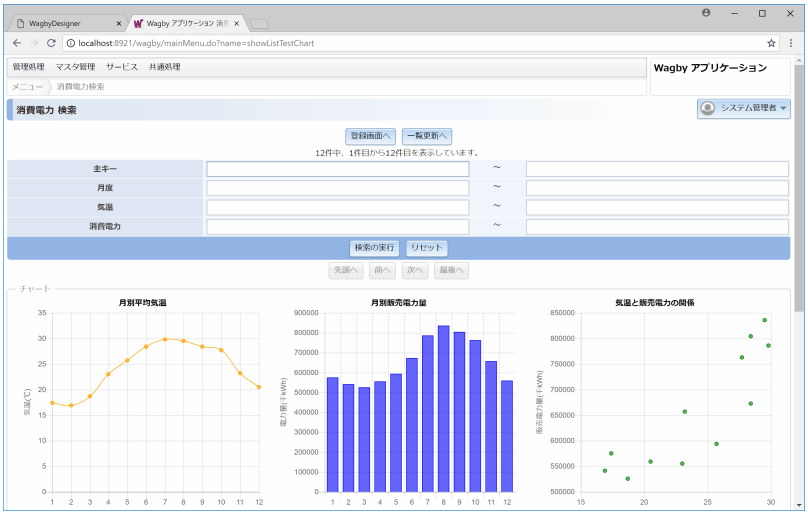


図 折れ線グラフ、棒グラフ、散布図を表示した例

Designerの設定

Designer の設定方法を説明します。ここでは散布図のためにダミー項目 item1 を用意しました。

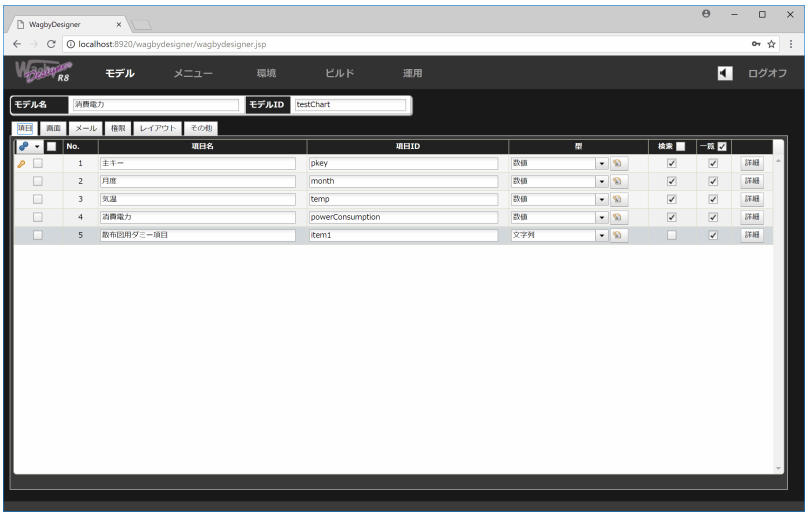


図 散布図用の項目item1を用意する

item1項目に対して設定を行います。ここで「軸ラベル項目ID」は空白とします。散布図の場合、この部分は後述する JavaScript で指定します。

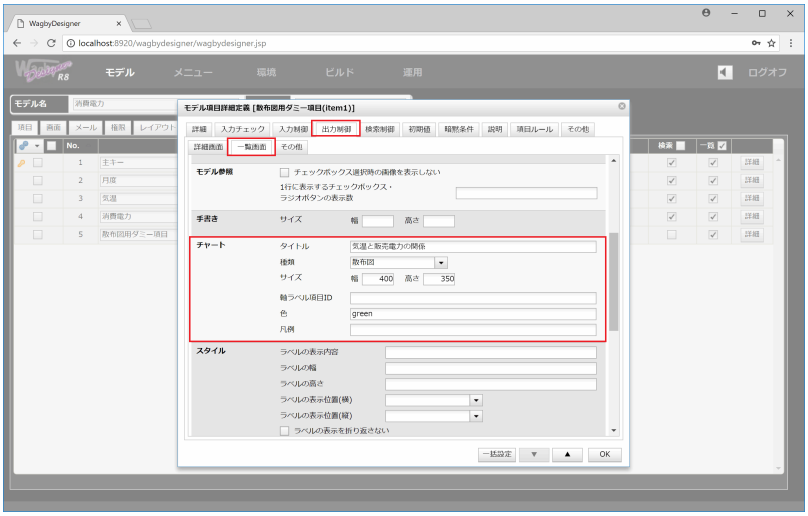


図 チャートの設定

JavaScript

散布図のために次のコードを追加します。

```
/* 散布図のための表示データ設定 */
function getItem1Item1ChartDataset(data) {
  return {
    label: getItem1Item1ChartDatasetLabel(),
    data: getScatterData(data.entity, "temp", "powerConsumption"),
    backgroundColor: getItem1Item1ChartDatasetBgColor(),
    borderColor: getItem1Item1ChartDatasetBorderColor(),
    borderWidth: getChartBorderWidth(),
  };
}
/* グラフ出力カスタマイズ */
function getItem1ChartOptions(defaultChartOption) {
  var option = defaultChartOption;
//console.log("option: ", option);
  option.legend = {
    display: false,
  }
  option.scales.xAxes[0].scaleLabel.labelString = "月別平均気温(℃)";
  option.scales.yAxes[0].scaleLabel.labelString = "販売電力量(千kWh)";
  return option;
}
```

- 散布図以外のグラフデータは、数値の配列を data 属性としてセットします。一方、散布図では次のようにx軸とy軸の値をオブジェクトとした配列をセットする必要があります。

```
[{x: 1, y: 3}, {x: 3, y: 7}, ... ]
```

- この対応を行うため、散布図を利用する場合は get<項目ID><項目ID>ChartDataset 関数を定義します。上のコードを参考にしてください。引数の data に対して、散布図の対象となる項目ID（ここでは "temp" と "powerConsumption"）を指定します。項目IDの大文字と小文字は区別されますので、正確に指定してください。
- これらの関数は、自動生成されたファイル showList<モデルID>.js の drawChart 関数を起点として、順に呼び出されています。カスタマイズ開発者はこの（自動生成された）ファイルも参考にするとよいでしょう。

ダウンロード

ここで紹介したサンプルをダウンロードしてお試しいただけます。R8.1.0 以上の Wagby でご利用ください。

- リポジトリ
- サンプルデータ

詳細画面にグラフを表示する（JavaScript）

実行例

5教科のスコアに関するグラフを用意した例です。このように複数のグラフを表示することができます。

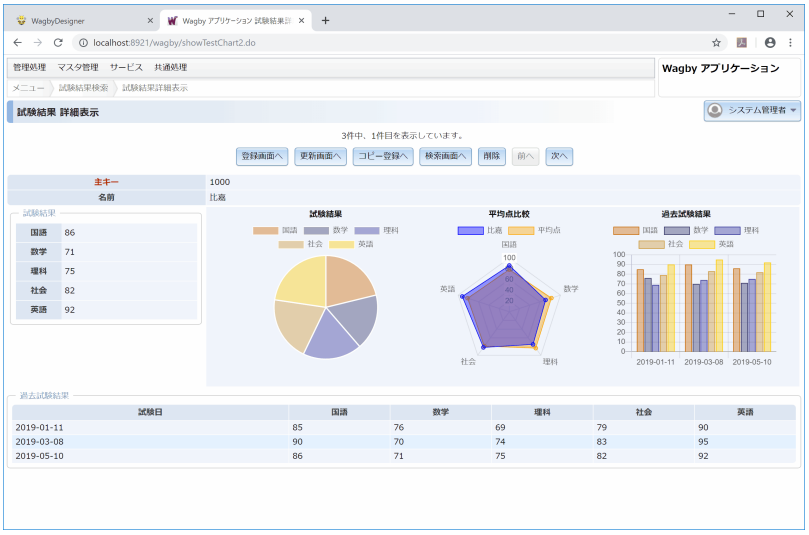


図 円グラフ、レーダーチャート、縦棒グラフの表示

Designerでの設定

個人毎に直近の5教科のスコアと過去テストのスコアを記録するための項目を用意しました。また、平均点を保持する項目も用意しています。

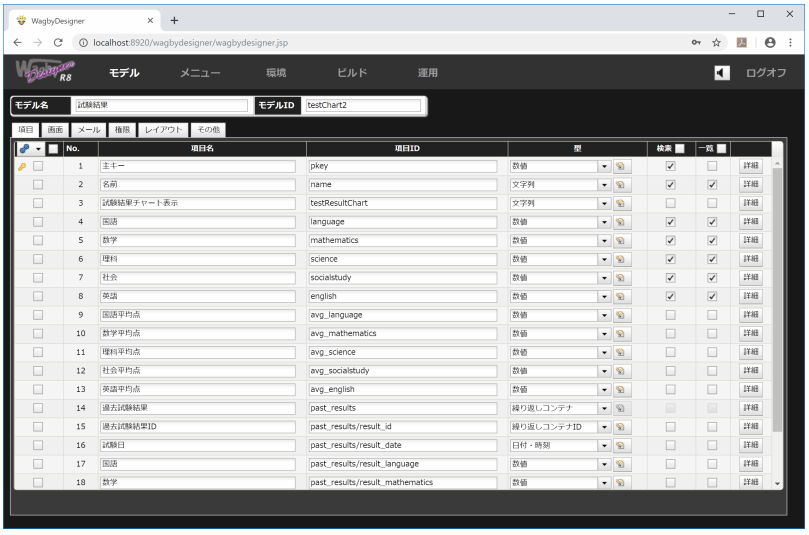


図 スコアを管理するモデル

グラフのタイトルと種類の確認

はじめに、どの項目にグラフの設定を行なっているのかを確認しておきます。先頭の項目（ここでは「主キー」）のモデル項目詳細定義ダイアログから〝出力制御 > 詳細画面〟を開きます。〝チャート > タイトル〟部を選択し、「一括設定」画面を開きます。

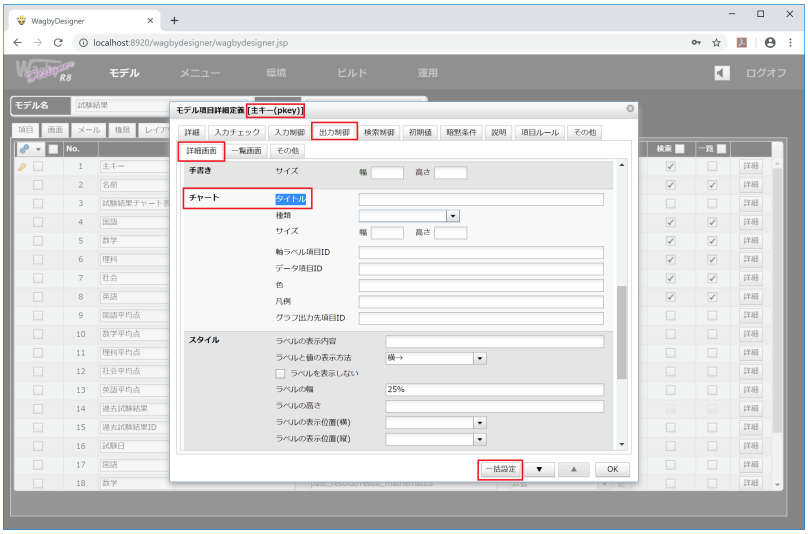


図 チャート>タイトルを確認する

リポジトリキーは 〝presentation/displayitem/displaychart/@titleForShow:タイトル〟 です。図にあるように「名前」「国語」「国語平均点」「試験日」項目にグラフの設定を行なっていることがわかります。

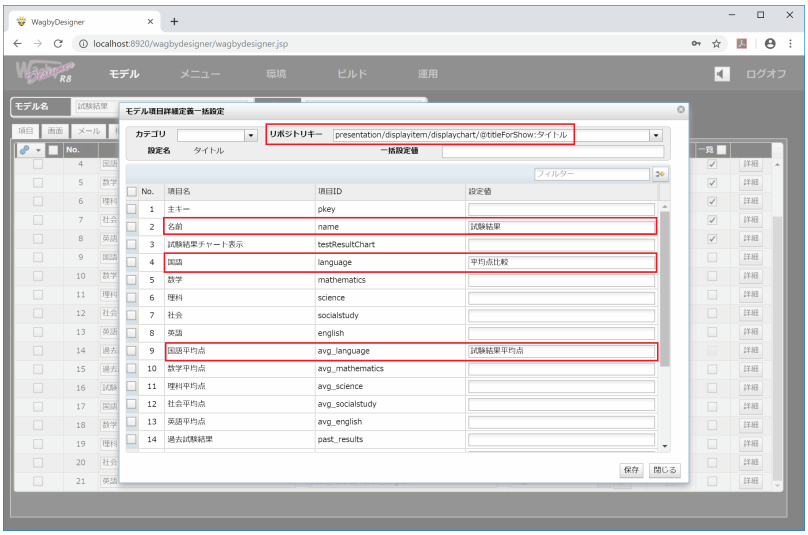


図 グラフのタイトルを設定している項目の確認(1)

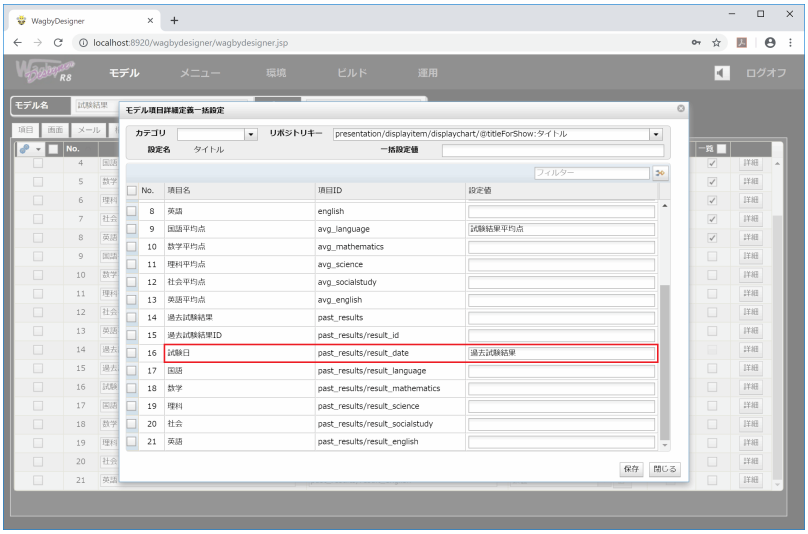


図 グラフのタイトルを設定している項目の確認(2)

同じように、グラフの種類の設定も確認しておきます。"チャート ＞ 種類" 部を選択し、「一括設定」画面を開きます。

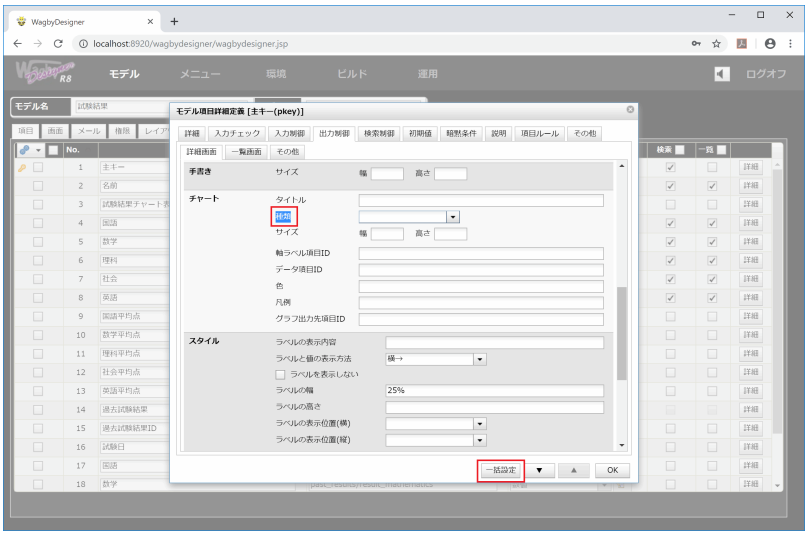


図 チャート>種類を確認する

リポジトリキーは "presentation/displayitem/displaychart/@typeForShow:種類" です。図にあるように "円","レーダー","縦棒" が設定されていることがわかります。

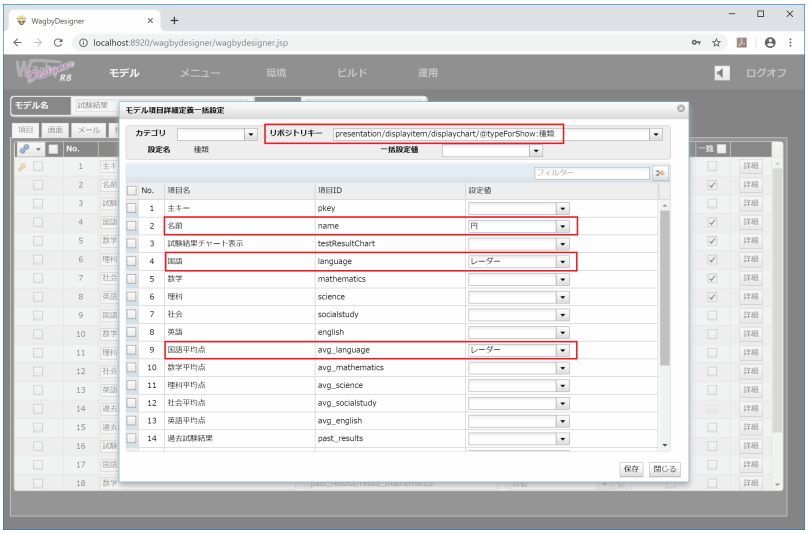


図 グラフの種類を設定している項目の確認(1)

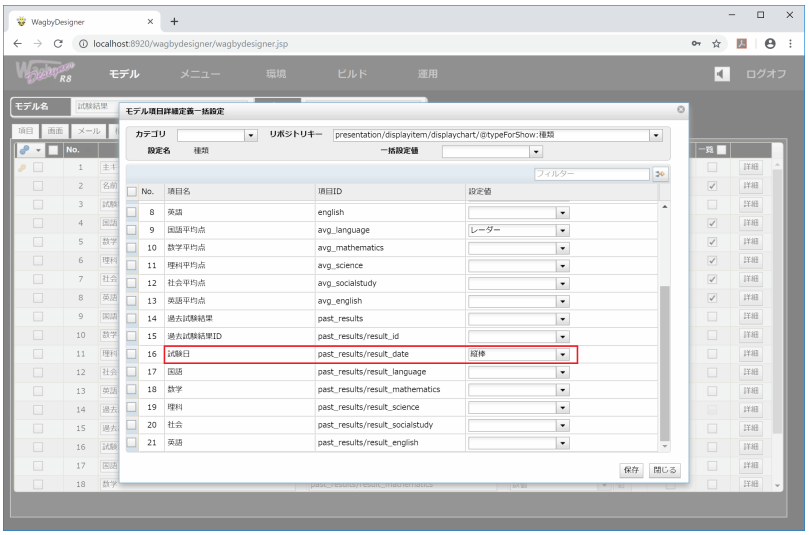


図 グラフの種類を設定している項目の確認(2)

ここであらためて図の実行例の画面と設定を比較します。図で表示されているグラフは、タイトルが ″試験結果″, ″平均点比較″, ″過去試験結果″ の3種類で、それぞれ ″円″,″レーダー″,″縦棒″ が使われています。しかし設計情報では、グラフの設定は4つの項目に対して行われています。

図でレーダーチャートを指定している項目は「国語」と「国語平均点」の二つあることがわかります。実はレーダーチャートは二つのグラフを用意しますが、これを一つのグラフにまとめて（重ね合わせて）表示しています。この処理は JavaScript を使っています。詳細は後述します。

ここでは、レーダーチャートの対象となる設定は二箇所に行なっているが、表示時は一つにまとめている、と解釈してください。

試験結果（円グラフ）の設定

続いて各グラフの設定を確認します。最初は「試験結果」です。これは円グラフとしています。

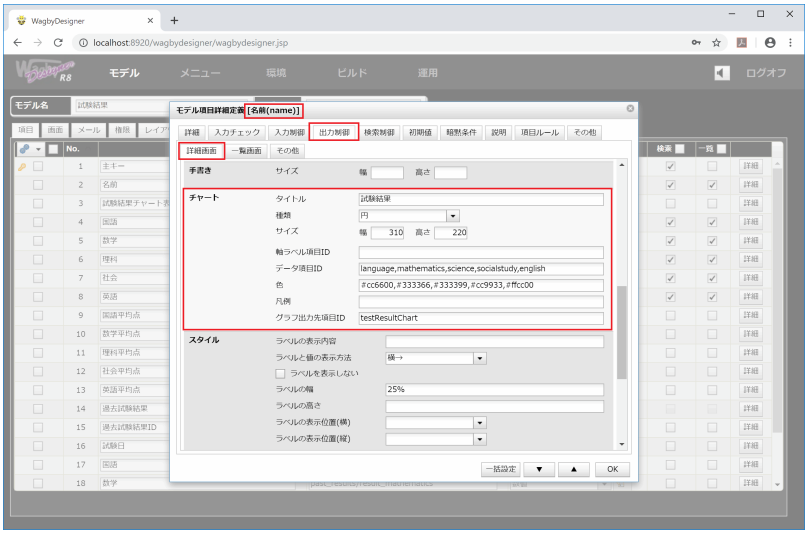


図 試験結果グラフの設定

試験結果は5つの項目（「国語」「数学」「理科」「社会」「英語」）の値を使っています。グラフの設定は、これらの対象項目とは無関係に、任意の項目に対して行うことができます。ここでは「名前」項目に対して設定しています。

設定項目	説明	必須区分	設定例
タイトル	グラフのタイトルを指定します。	必須	試験結果
種類	グラフの種類を指定します。縦棒、横棒、折線、円、ドーナツ、散布図、レーダーを指定できます。	必須	円
サイズ	表示するグラフの幅と高さをピクセル値で指定します。	必須	310, 220
軸ラベル項目ID	軸となる項目IDを指定します。詳細表示に含まれる項目を指定してください。	（縦棒, 横棒, 折線で必須）	
データ項目ID	グラフに表示するデータが格納されている項目IDを指定します。この項目は数値型です。詳細表示に含まれる（複数の）項目をコンマ区切りで指定してください。	必須	
色	HTMLで利用できる色を指定します。色名またはシャープ文字ではじめる16進数表記が利用できます。縦棒、横棒、折線、円についてはデータ項目IDに指定した数だけ（コンマ区切りで）色を指定してください。	-	

凡例	グラフ内に凡例の情報を含めることができます。	-	
グラフ出力先項目ID	詳細画面に用意された項目の位置に、グラフを表示します。レイアウト指定と組み合わせることで、任意の場所にグラフを埋め込むことができます。(後述) 未設定の場合は詳細画面の下部にグラフが表示されます。		

この円グラフは JavaScript によるカスタマイズなしで実現しています。

平均点比較（レーダー）

今回の例では、二つのレーダーチャートを用意し、重ね合わせて表示させるようにしています。一つ目は「今回の5教科のスコア」に関するレーダーチャートです。データ項目IDは5教科の項目を指定しています。レーダーチャートの場合、色は一色のみを指定します。グラフ出力先項目IDは共通で利用しています。

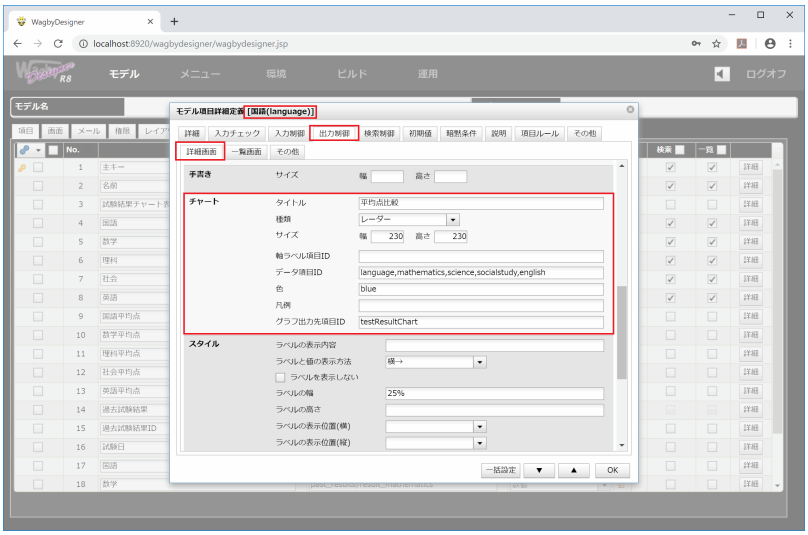


図 平均点比較レーダーチャートの設定(1)

もう一つは「平均点」に関するレーダーチャートです。これを重ね合わせることで図のように「自分のスコア」と「平均スコア」の違いを視覚的に把握することができます。

データ項目IDは、平均点が格納されている5つの項目（「国語平均点」「数学平均点」「理科平均点」「社会平均点」「英語平均点」）の項目IDをコンマ区切りで列挙しています。また、凡例として　"平均点"という文字を指定しています。

グラフ出力先項目IDが空白となっていることに注意してください。この場合、グラフは出力されません。後述しますが、この平均点についてのレーダーチャートは　JavaScript　を用いて「平均点比較」レーダーチャートに重ね合わせるようにしています。

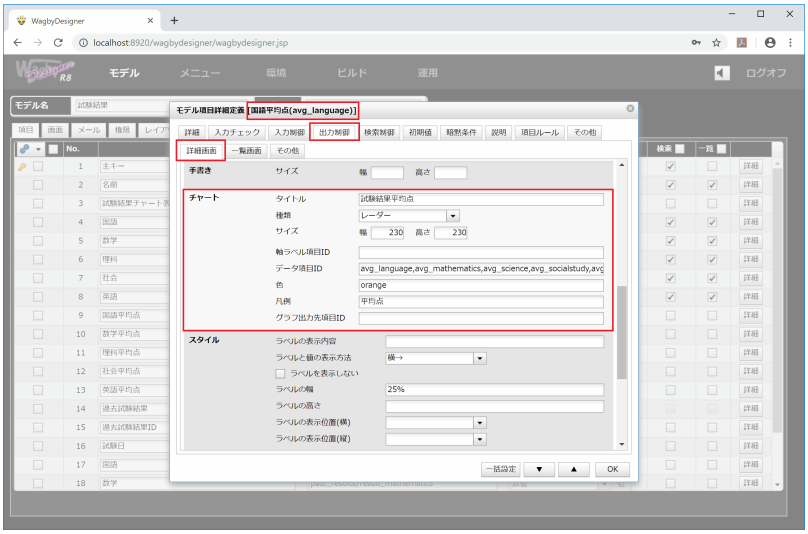


図 平均点比較レーダーチャートの設定(2)

平均点を求める

今回は（複数の学生の）教科ごとの平均点を求めるものとします。Wagbyが提供する一覧表示の集計機能を使います。

具体的には各教科について〝出力制御 ＞ 一覧表示 ＞ 集計値〟に用意された〝平均値〟を算出するようにします。

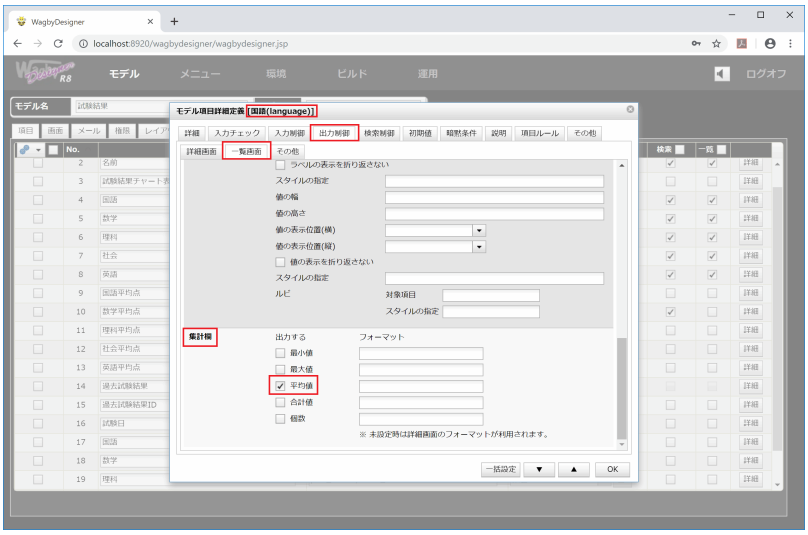


図 集計値>平均値を算出する

どの項目に設定したかを確認しておきます。〝平均値〟をマウスで選択し、「一括設定」ボタンを押下します。

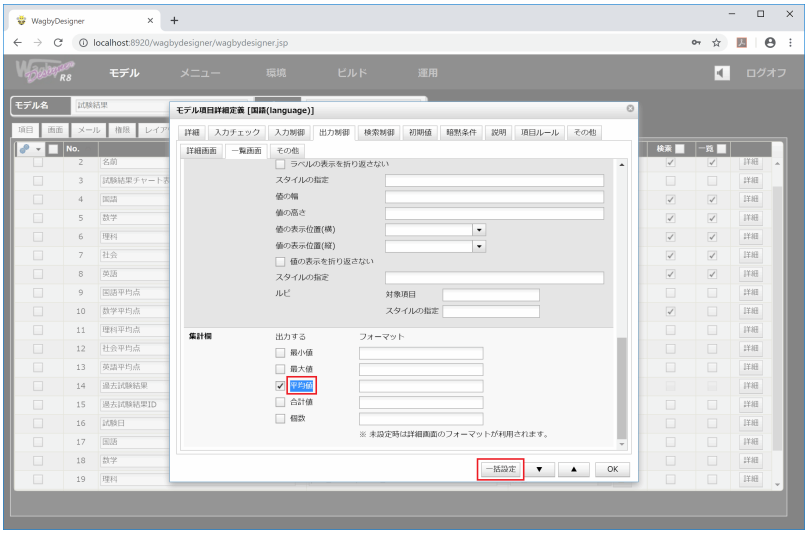


図 平均値に関する一括設定(1)

5教科すべてで、平均値を算出する設定が行われています。

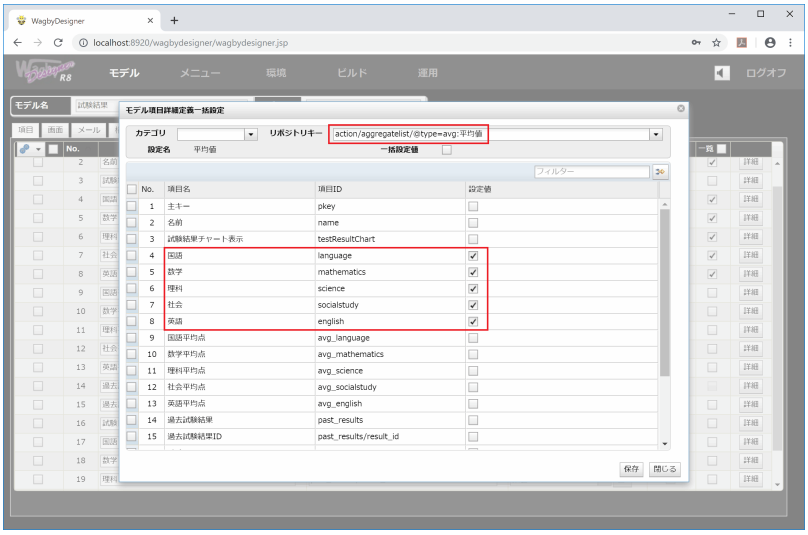


図 平均値に関する一括設定(2)

一覧表示画面では、次のように平均値が出力されるようになっています。

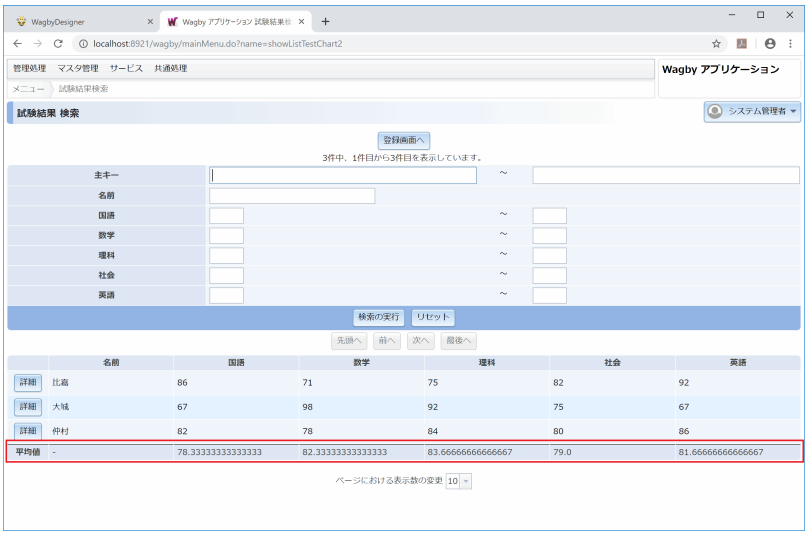


図 一覧表示の集計（平均値）

この平均値を保持する項目として、それぞれ「国語平均点」「数学平均点」「理科平均点」「社会平均点」「英語平均点」を用意しています。 計算式に関数 AGAVG を適用しました。一覧表示画面の集計機能で求めた「平均値」の値を取得できます。かつ、データベースには保存しない項目としています。

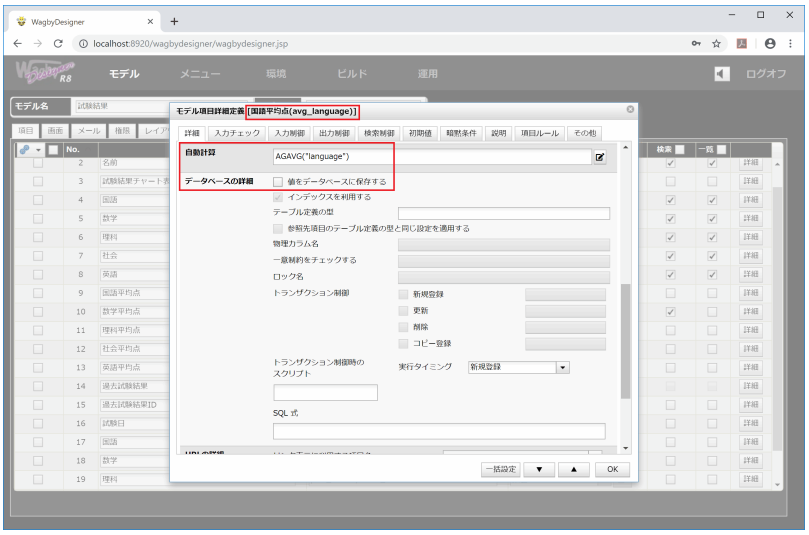


図 AGAVG関数を利用する

これまでと同じように、他の項目でも設定されていることを確認します。「自動計算」をマウスで選択し、「一括設定」ボタンを押下します。

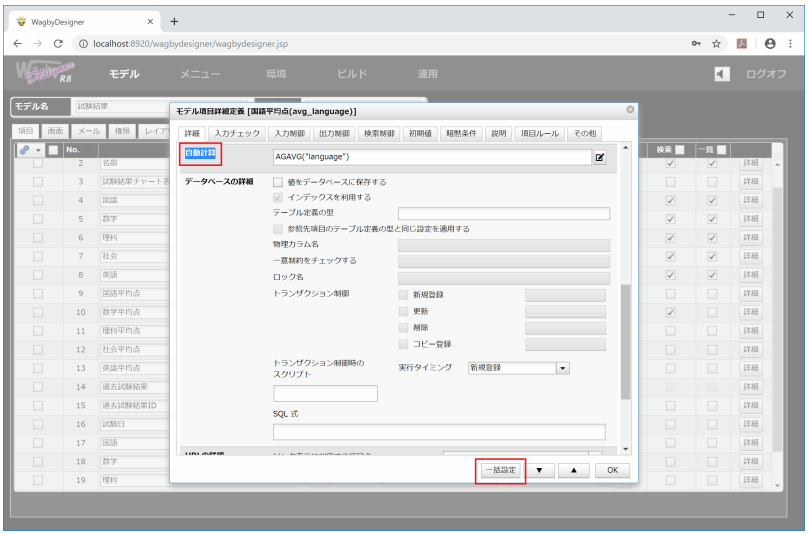


図 自動計算に関する一括設定(1)

平均点を保持する項目では、すべて関数AGAVGが設定されていることがわかります。（同様に、これらの項目はすべてデータベースに保存しない設定となっています。）

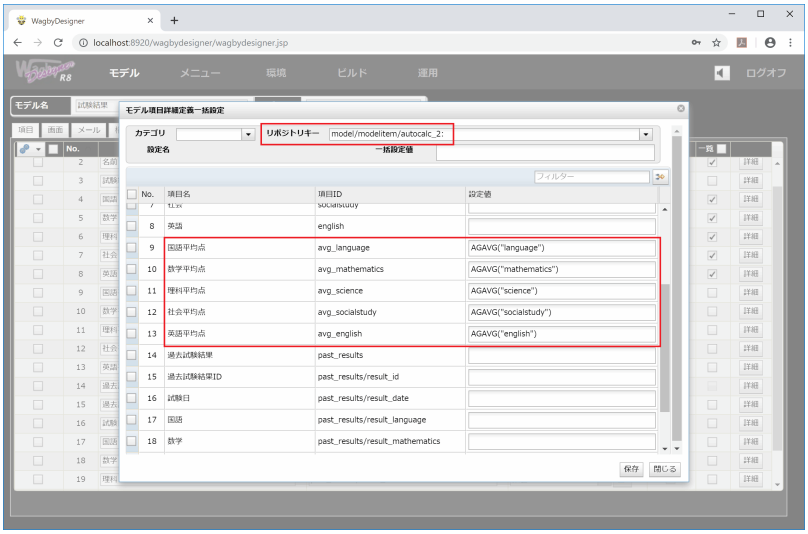


図 自動計算に関する一括設定(2)

過去試験結果（棒グラフ）

最後のグラフは過去試験結果です。設定する項目は任意です。ここでは「試験日」項目に設定しています。

棒グラフの場合、「軸ラベル項目ID」の指定が必要です。ここでは繰り返しコンテナ内の「試験日」項目の項目IDを指定します。

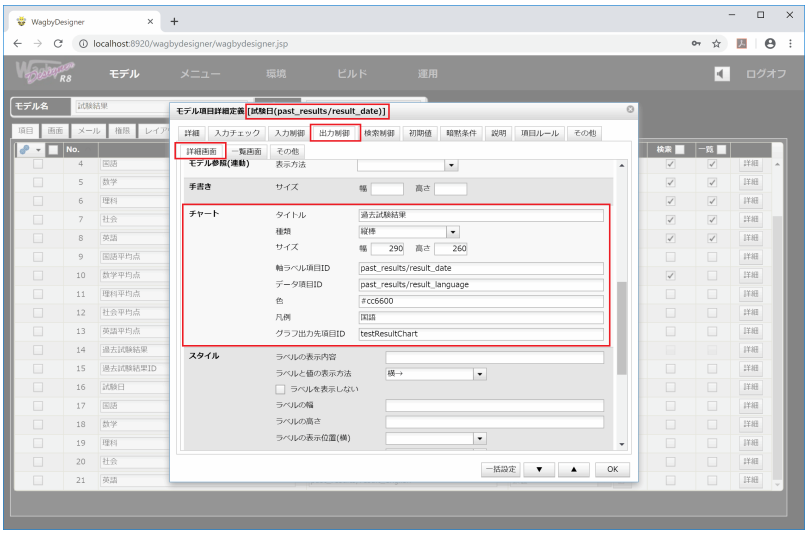


図 過去試験結果グラフの設定

棒グラフのデータ項目は最初の一つである、繰り返しコンテナ内の項目「国語」だけが設定できます。棒グラフに出力するデータが一つ（ここでは「国語」だけ）であればこのまま利用できます。しかし今回は、一つの棒グラフに他の4教科のスコア（繰り返しコンテナ内の項目「数学」「理科」「社会」「英語」）も同時に表示させてみます。これは追加設定となるため、JavaScript を利用して拡張します。[後述]

グラフの出力位置の設定

「試験結果チャート表示」項目をダミー項目としています。この部分にグラフが出力されるよう、各グラフの設定では「グラフ出力先項目ID」に、この項目の項目ID "testResultChart" を指定しています。

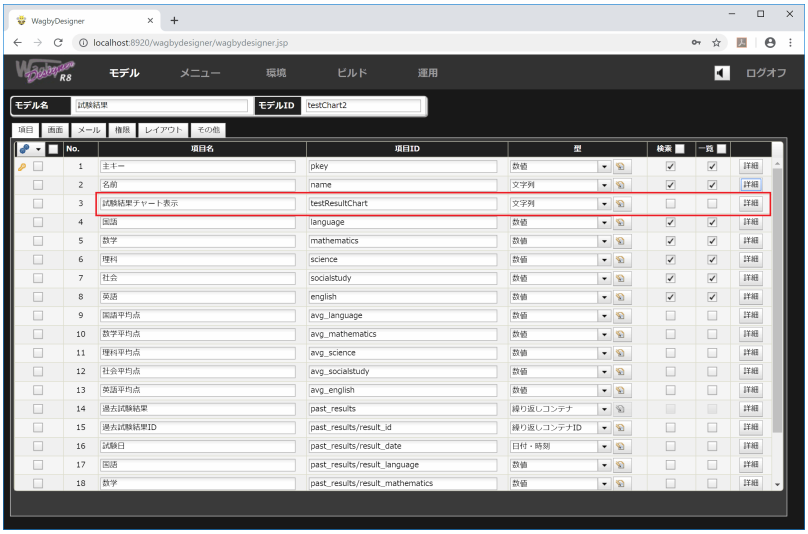


図 「試験結果チャート表示」項目を用意する

「試験結果チャート表示」項目はダミー項目であるため、データベースに保存する必要はありません。

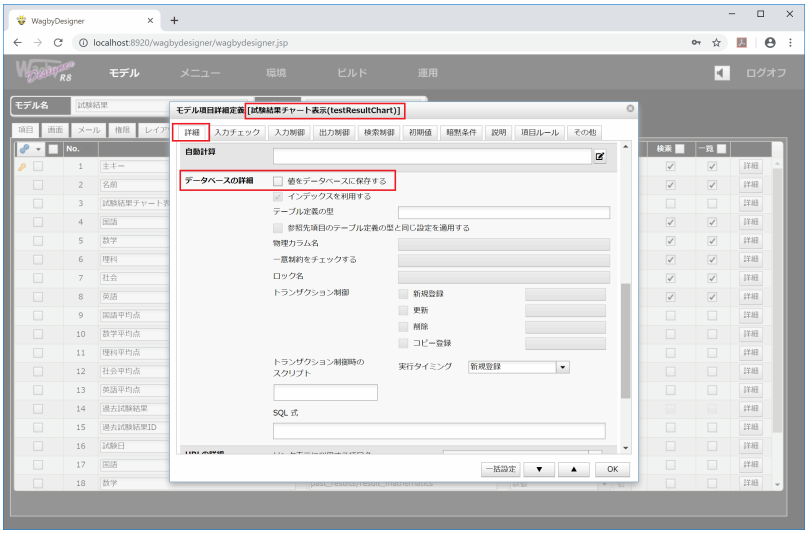


図 ダミー項目はデータベースに保存しない

「試験結果チャート表示」項目はダミー項目であるため、入力時は隠し項目とします。

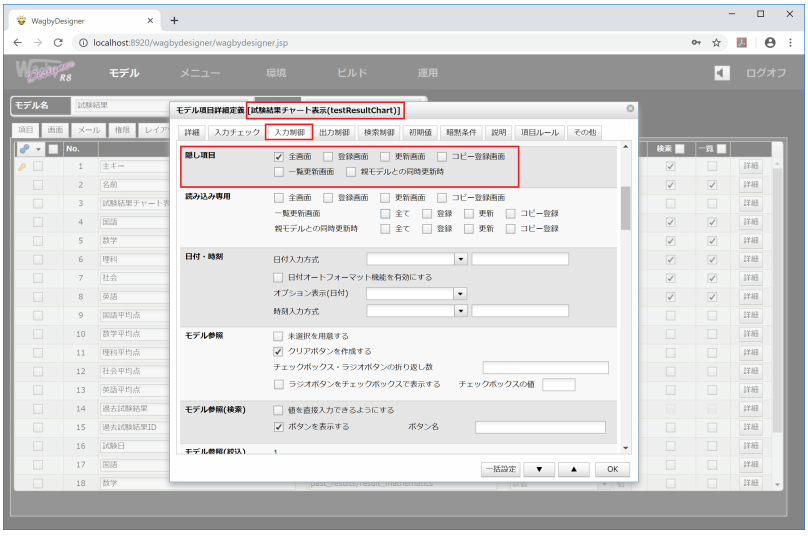


図 ダミー項目は隠し項目にする

スクリーンエディタで、このダミー項目の位置を調整します。

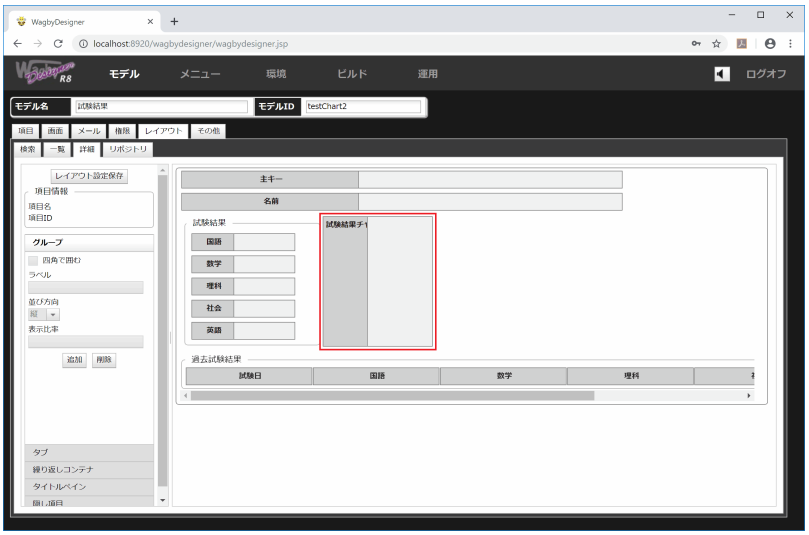


図 レイアウトの設定

REST API の設定

グラフ表示のためのデータ取得で REST API を使います。そのためグラフを表示するモデルでは REST API を有効にします。

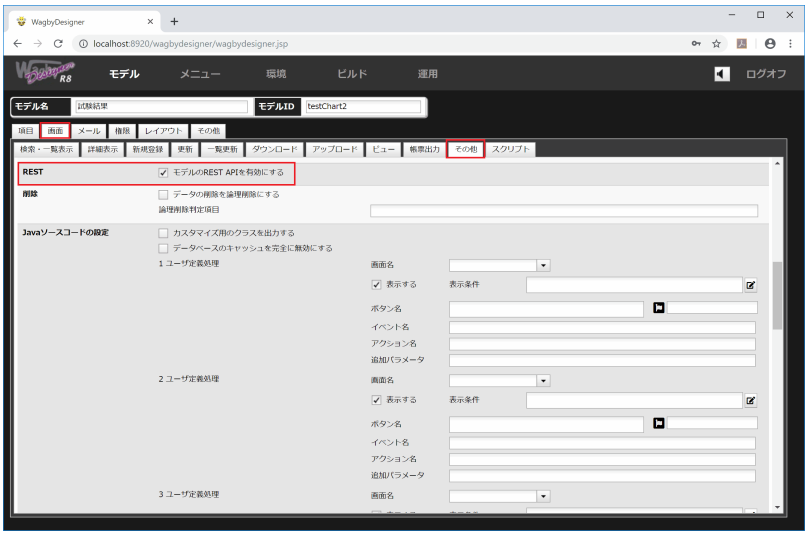


図 RESTAPIを有効にする

JavaScriptでの追加設定

今回はさらに JavaScript を追加しています。開発者は customize/webapp/<モデルID> フォルダを用意し、MyShow<モデルID>.js ファイルを作成します。（ファイル名に含めるモデルIDは、キャメル記法を適用してください。）

```
/* 色の透明度の設定。レーダーチャートで色を重ね合わせるため、見やすくする配慮。*/
function getChartColorAlpha() {
  return 0.4;
}
```

```
/* 図「試験結果平均点」はグラフ出力先項目IDが未設定のため、画面下部に出力される。この出力を抑制する。*/
function beforeDrawChart(dom, domAttr, domClass, domConstruct, domStyle, query, Button,
 registry, request) {
  query("div.display_chart").forEach(function(node) {
    domStyle.set(node, "display", "none");
  });
}

/* 「国語(language)」項目に設定した「平均点比較」レーダーチャートの「凡例」に、受講者氏名(name)を設定する。*/
function getLanguageLanguageChartDatasetLabel(data) {
  return getEntityData(data.entity, ["name"]);
}

/* 「国語(language)」項目に設定した「平均点比較」レーダーチャートへのデータ重ね合わせ。*/
function getLanguageChartDatasets(data) {
  var datasets = [];
  datasets.push(getLanguageLanguageChartDataset(data));/*「国語(language)」項目に設定した「平均点比較」レーダーチャートのデータ */
  datasets.push(getAvgLanguageAvgLanguageChartDataset(data)); /*「国語平均点(avg_language)」項目に設定した「試験結果平均点」レーダーチャートのデータ */
  return datasets;
}

/* 「試験日(result_date)」項目に設定した棒グラフの縦棒の高さの最大値を100とする。*/
function getResultDateChartOptions(defaultChartOption) {
  var option = defaultChartOption;
  option.scales.yAxes[0].ticks.suggestedMax = 100;
  return option;
}

/* 繰り返しコンテナ内項目「数学 (parts_results/result_mathematics)」の情報を棒グラフに追加 */
function getResultMathematicsResultDateChartDataset(data) {
  var color = "#333366";
  return {
    label: "数学",
    data: getEntityData(data.entity, ["past_results/result_mathematics"]),
    backgroundColor: Chart.helpers.color(color).alpha(getChartColorAlpha()).rgbString(),
    borderColor: color,
    borderWidth: getChartBorderWidth(),
  };
}

/* 繰り返しコンテナ内項目「理科 (parts_results/result_science)」の情報を棒グラフに追加 */
function getResultScienceResultDateChartDataset(data) {
  var color = "#333399";
  return {
    label: "理科",
    data: getEntityData(data.entity, ["past_results/result_science"]),
    backgroundColor: Chart.helpers.color(color).alpha(getChartColorAlpha()).rgbString(),
    borderColor: color,
    borderWidth: getChartBorderWidth(),
  };
}

/* 繰り返しコンテナ内項目「社会 (parts_results/result_socialstudy)」の情報を棒グラフに追加 */
function getResultSocialstudyDateChartDataset(data) {
  var color = "#cc9933";
  return {
    label: "社会",
    data: getEntityData(data.entity, ["past_results/result_socialstudy"]),
    backgroundColor: Chart.helpers.color(color).alpha(getChartColorAlpha()).rgbString(),
    borderColor: color,
```

```
    borderWidth: getChartBorderWidth(),
  };
}

/* 繰り返しコンテナ内項目「英語 (parts_results/result_english)」の情報を棒グラフに追加 */
function getResultEnglishDateChartDataset(data) {
  var color = "#ffcc00";
  return {
    label: "英語",
    data: getEntityData(data.entity, ["past_results/result_english"]),
    backgroundColor: Chart.helpers.color(color).alpha(getChartColorAlpha()).rgbString(),
    borderColor: color,
    borderWidth: getChartBorderWidth(),
  };
}

/* 棒グラフへ出力するデータをまとめる */
function getResultDateChartDatasets(data) {
  var datasets = [];
  datasets.push(getResultDateResultDateChartDataset(data));
  datasets.push(getResultMathematicsResultDateChartDataset(data));
  datasets.push(getResultScienceResultDateChartDataset(data));
  datasets.push(getResultSocialstudyDateChartDataset(data));
  datasets.push(getResultEnglishDateChartDataset(data));
  return datasets;
}
```

- グラフ表示はオープンソースの Chart.js ライブラリを使っています。（集計ビューで使っている Dojotoolkit のグラフライブラリとは別になっています。）
- レーダーチャートおよび棒グラフに複数のデータを重ね合わせて表示するために、上のようなコードを追加する必要があります。
- 他にどのような設定が行えるかは Chart.js のドキュメントを参照してください。

仕様・制約

- グラフ表示のためのデータ取得で REST API を使います。グラフを表示するモデルでは、REST API を有効にしてください。
- グラフを帳票（Excel帳票、PDF帳票）に含めることはできません。Webブラウザでの表示用となります。

認証・認可

権限管理の概要

ロール

ロールベースの権限管理では「利用者」と「データ」を直接、結びつけません。その間に「権限」という概念を導入します。

Wagbyはロールベースの権限管理機能を提供します。

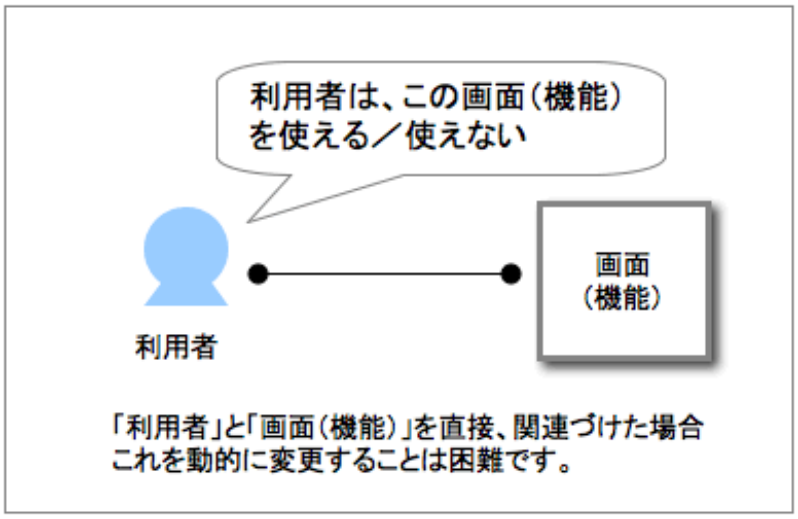


図 ロールを用いない場合

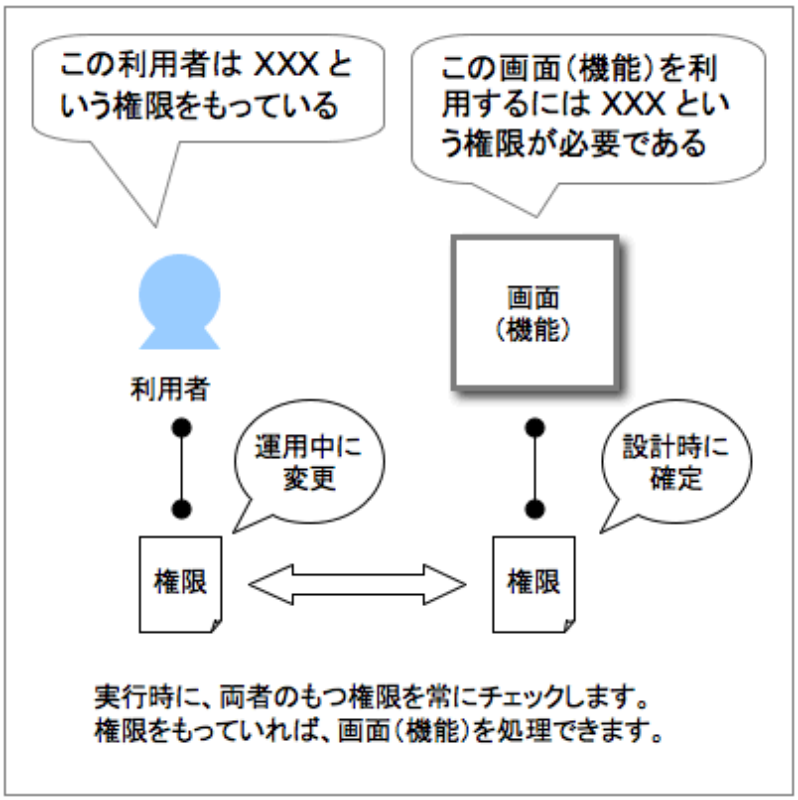


図 ロールを用いる場合(Wagby)

パーミッションとプリンシパル

権限の基本単位を「パーミッション」と呼びます。 たとえば「顧客モデルの表示パーミッション」や「社員モデルの更新パーミッション」などがあります。

各画面には「自分を処理するためには、このモデルについてのXXX権限が必要である」というチェック機能が備わっています。 「XXX」の部分はそれぞれ、「登録」「更新」「削除」「表示」「検索」といった基本的な権限が入ります。

複数のパーミッションをグループ化したものを「プリンシパル」と呼びます。たとえば、「登録」「更新」パーミッションをグループ化して「更新系プリンシパル」を用意することもできます。Wagbyの標準は「複数パーミッション = 1プリンシパル」となっています。

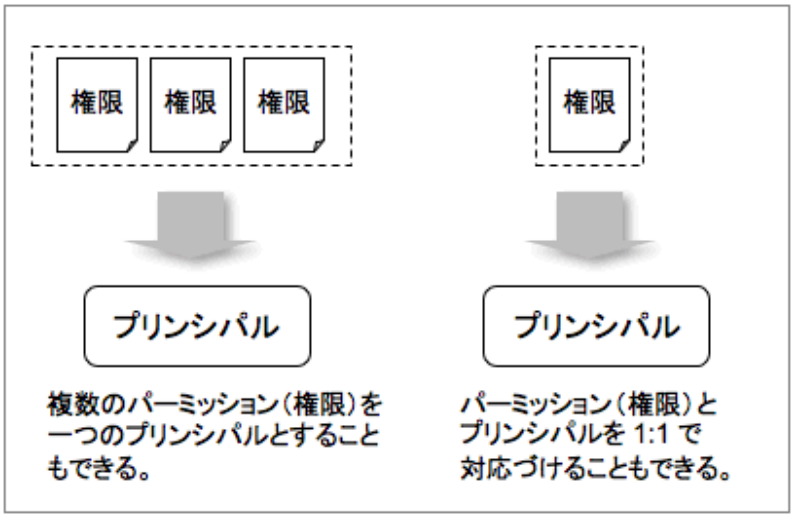


図 パーミッションとプリンシパルの関係

管理者は、利用者（アカウント）に対して「プリンシパル」を割り当てることができます。 この操作は運用中に行えるため、ある利用者に対して一時的に操作権限を付与したり、削除したりといった柔軟な対応を実現できます。

標準のパーミッション

開発者がモデルを定義したとき、モデルごとに次のパーミッションが自動的に付与されます。標準では、これらすべてのパーミッションを有効とした「一般ユーザ」というプリンシパルもあわせて用意されます。

パーミッション	説明	メニューとの関係	内部の表記
閲覧	詳細表示画面へのアクセスを制御します。	なし。詳細表示画面のメニューは用意されません。	select
検索	検索・一覧表示ならびにサブウィンドウ検索のアクセスを制御します。	別途 "メニュー" パーミッションで設定します。(後述)	showlist
更新	更新画面へのアクセスを制御します。	なし。更新画面のメニューは用意されません。	update
登録	登録画面へのアクセスを制御します。	あり。このパーミッションをもたないアカウントは、登録画面のメニューは用意されません。	insert
削除	削除機能へのアクセスを制御します。	なし。削除機能のメニューは用意されません。	delete
ダウンロード	CSV・Excelダウンロード画面へのアクセスを制御します。	あり。このパーミッションをもたないアカウントは、ダウンロード画面のメニューは用意されません。	download
アップロード	CSV・Excelアップロード更新画面へのアクセスを制御します。	あり。このパーミッションをもたないアカウントは、アップロード更新画面のメニューは用意されません。	upload

メニュー	検索・一覧表示画面のメニューへの表示を制御します。	あり。このパーミッションをもたないアカウントは、検索・一覧表示画面のメニューは用意されません。	menu
一覧更新	一覧更新画面へのアクセスを制御します。	あり。このパーミッションをもたないアカウントは、一覧更新画面のメニューは用意されません。	uploadupdate
帳票出力	帳票出力画面へのアクセスを制御します。	あり。このパーミッションをもたないアカウントは、帳票出力画面のメニューは用意されません。	reportlist

標準のプリンシパル

環境タブ内の「権限」でプリンシパルを管理しています。標準で次のプリンシパルを用意しています。

一般ユーザ

閲覧・検索・更新・登録・削除・ダウンロード・アップロード・メニュー・一覧更新・帳票出力という標準パーミッションを含んでいます。通常、ログオンアカウント(juser)には「一般ユーザ」プリンシパルを与えます。これによってすべてのモデルへの操作が可能になります。

アカウントごと、かつ、モデルごとに利用制限をかけたい場合は、一般ユーザプリンシパルを与えずに別途用意した（制限付きの）プリンシパルを与えるとよいでしょう。この方法は次ページ以降で説明します。

それ以外の特別なプリンシパル

「一般ユーザ」プリンシパルは（開発者が定義した）モデルの操作のために用意されたものですが、それ以外に、システム管理のためのプリンシパルが用意されています。これらはDesignerの〝環境 > 権限 > プリンシパル〟で確認できます。

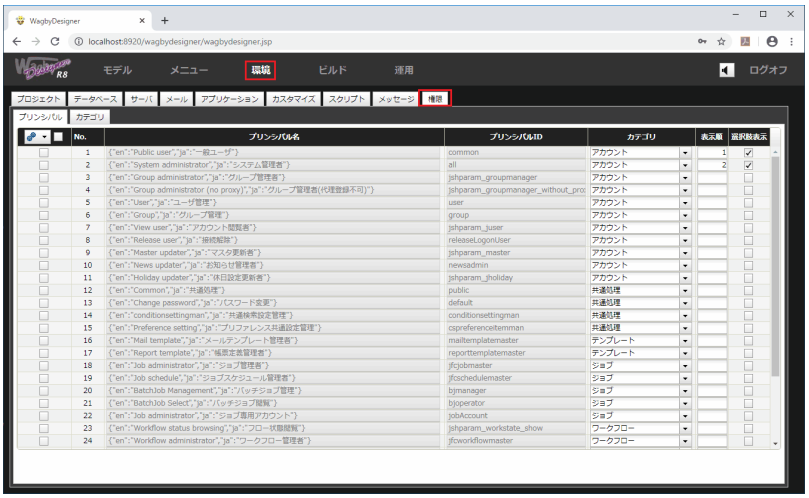


図 定義されているプリンシパル一覧

カテゴリ	種類	内容	備考
アカウント	一般ユーザ	(説明済みのため割愛します。)	
	システム管理者	すべての権限をもちます。標準で用意されているアカウント	

		"admin" に割り当てられています。	
	グループ管理者	各グループには、グループ内のデータに対して特別な権限を持つ「グループ管理者」を用意することができます。	
	グループ管理者（代理登録不可）	グループ管理者の権限でデータを操作できます。代理登録を行うことはできません。	
	ユーザ管理	ユーザアカウント(juser)の管理を行います。	R8.4.0
	グループ管理	グループ(jgroup)の管理を行います。	R8.4.0
	アカウント閲覧者	自分以外のアカウント情報を閲覧することができます。グループ権限設定での代理登録機能。ワークフローの代理承認。	
	接続解除	ログオン済みの利用者を強制的に解除することだけが行える権限です。	
	マスタ更新者	選択肢モデルの操作を行う権限です。	
	お知らせ管理者	「お知らせ」の登録、更新操作を行うことができます。	
	休日設定更新者	システムが管理する「休日」を追加することができます。	
共通処理	共通処理	プレファレンス（配色や、各パーツの表示制御）を指定することができます。	
	パスワード変更	パスワードの変更を行うことができます。	
	共通検索設定管理	検索設定を共通に管理できる権限です。	R8.3.2
	プリファレンス共通設定管理	検索設定を共通に管理できる権限です。	
テンプレート	メールテンプレート管理者	メールテンプレートの登録、更新操作を行うことができます。	
	帳票定義管理者	帳票テンプレートの登録、更新操作を行うことができます。	
ジョブ	ジョブ管理者	Wagbyが提供する「ジョブマスタ」の登録、更新操作を行うことができます。	
	ジョブスケジュール管理者	Wagbyが提供する「ジョブ」の実行ルールを定義でき	

		ます。指定した時間にジョブが起動します。	
	バッチジョブ管理	Spring Batch を用いたバッチ処理で、各バッチジョブの状態を操作するための権限です。初期設定では、更新機能を無効としているため、本権限を設定しても利用できません。（バッチジョブは原則として閲覧のみが許可されるようになっています。）	
	バッチジョブ閲覧	Spring Batch を用いたバッチ処理で、各バッチジョブの状態（インスタンス、実行結果、ステップ）を閲覧できます。	
	ジョブ専用アカウント	ジョブを実行することができる特別な権限です。このプリンシパルはジョブ実行専用のため、これを有効にすると、ログオン画面からのログオンが行えなくなります。	
ワークフロー	フロー状態閲覧	承認すべきワークフローを確認できます。申請したワークフローの状態を確認できます。	
	ワークフロー管理者	フローパターン。 フロー参加者設定。 ワークフロー設定。	
	ワークフロー代理者設定	ワークフローの代理承認で用います。	
システム	システムログ閲覧	システムログ閲覧機能を利用できます。	R8.4.0
	統計情報閲覧	統計情報機能を利用できます。	R8.4.0
	システム情報閲覧	このアプリケーションについての詳細情報を閲覧できます。	R8.4.0
	インポートエクスポート	インポート・エクスポート機能を利用できます。	R8.4.0
	メンテナンスモード切替	メンテナンスモード切り替え機能を利用できます。	R8.4.0

カテゴリ (8.4.0)

すべてのプリンシパルは、次に示すいずれかのカテゴリに属します。開発者は自身でカテゴリを追加することもできます。

ID	カテゴリ名

1	アカウント
2	共通処理
3	テンプレート
4	ジョブ
5	ワークフロー
6	システム
7	アプリケーション

開発者が新規に作成するプリンシパルのカテゴリは、標準では ″アプリケーション″ を指定するとよいでしょう。プリンシパルが増えた場合は新しいカテゴリを作成し、カテゴリごとに管理することで見通しをよくすることができます。

ログオンアカウントにプリンシパルを割り当てる

Wagbyのログオンアカウントは juser モデルとして定義されています。 ログオンアカウントの設定時に、どのプリンシパルを割り当てるかを指定することができます。

通常は次のプリンシパルを付与するとよいでしょう。

- アカウント ＞ 一般ユーザ（モデルの操作のため）
- 共通処理 ＞ 共通処理（プリファレンス変更のため）
- 共通処理 ＞ パスワード変更

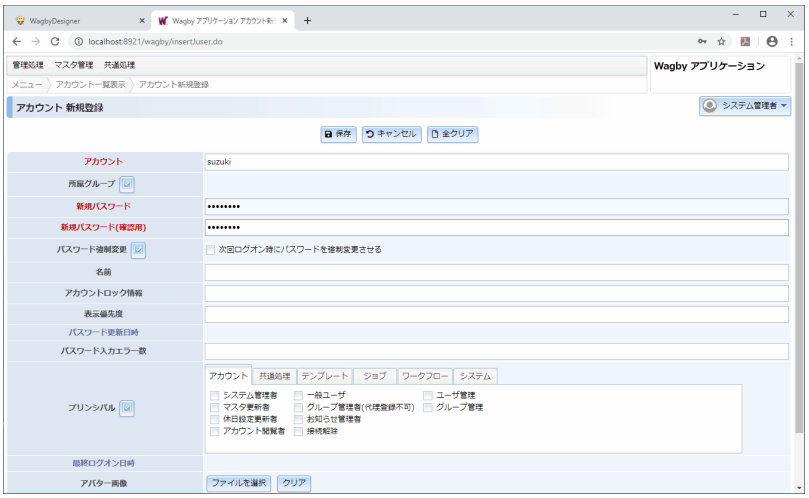


図 ログオンアカウント登録画面におけるプリンシパル設定欄(1)

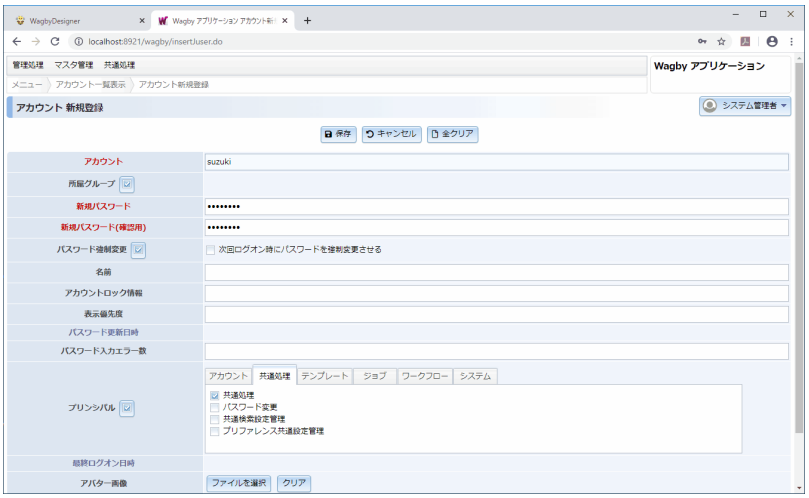


図 ログオンアカウント登録画面におけるプリンシパル設定欄(2)

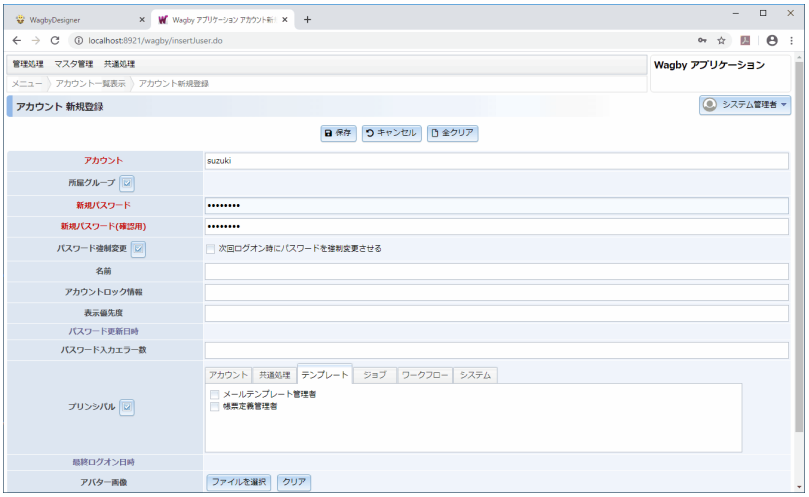


図 ログオンアカウント登録画面におけるプリンシパル設定欄(3)

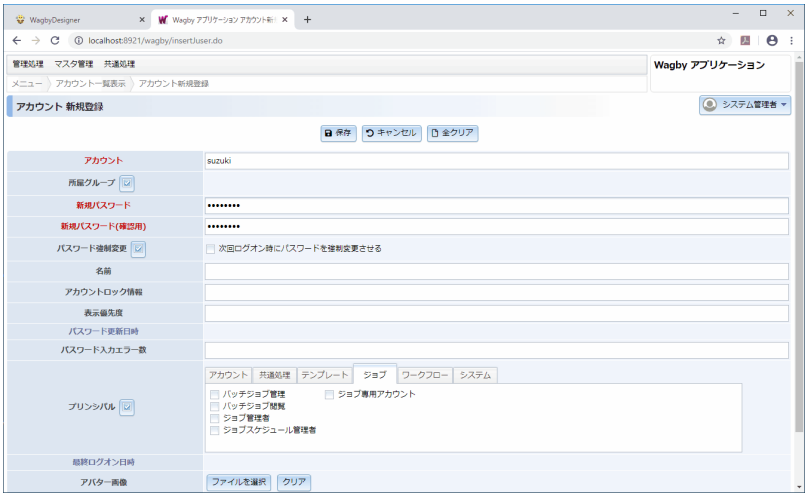


図 ログオンアカウント登録画面におけるプリンシパル設定欄(4)

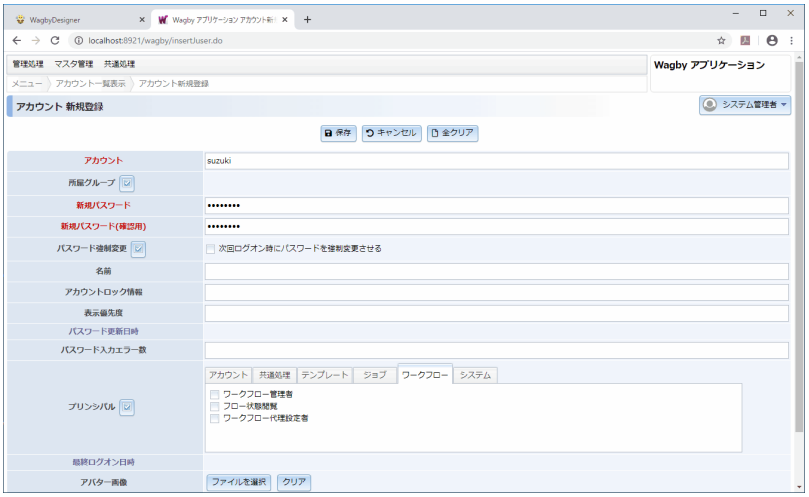


図 ログオンアカウント登録画面におけるプリンシパル設定欄(5)

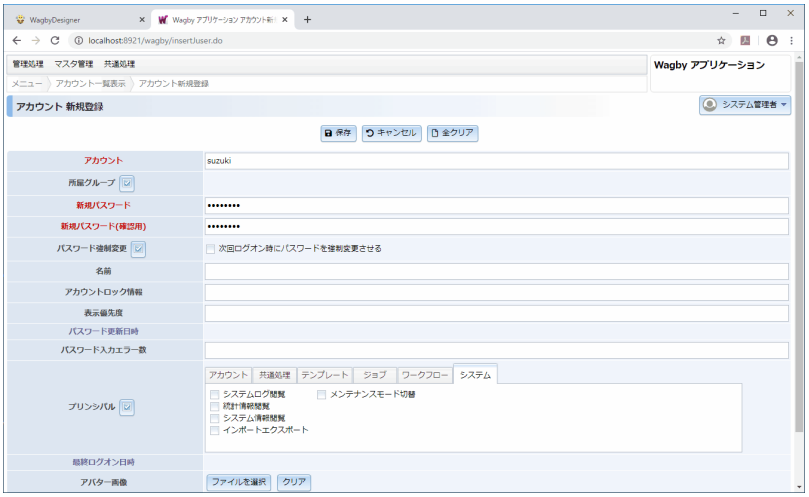


図 ログオンアカウント登録画面におけるプリンシパル設定欄(6)

モデル項目単位で権限を設定する

画面単位ではなく、モデル項目単位に「パーミッション」を設定することができます。 これにより、次のような複雑な設定も行うことができます。

参照権限 - 権限を有していないユーザに対して、項目を隠す

「営業日報」モデルに「上司のコメント」項目を用意した。この項目は管理者が閲覧・更新することができるが、営業スタッフはこの項目は見えない。すなわち、この項目が存在することさえも知らない。

更新権限 - 権限を有していないユーザに対して、項目を読み込み専用にする

「販売」モデルに「注文数量」項目を用意した。この項目は「注文数量更新権限」をもっている利用者は閲覧・更新することができるが、同権限をもっていない場合、閲覧はできるが更新を行うことができない。

この詳細は、次ページ以降で説明します。

権限管理の対象範囲

機能

権限設定は「画面」「帳票」「CSV・Excelダウンロード/アップロード更新」に適用されます。

アカウント

一般利用者が対象範囲となります。システム管理者（admin）は、すべての権限が有効になります。

権限管理の動作テストでは、システム管理者を用いないようにしてください。

システム管理者の扱い

「システム管理者」プリンシパルは、次のように動作します。

- 他のプリンシパルが持つ権限を自動的に継承します。*1
- すべてのメニューを操作できます。
- 権限管理設定の対象外となります。（すべての権限が有効になります。）
- 暗黙条件設定の対象外となります。（システム管理者には暗黙条件が適用されません。）
- すべてのデータの閲覧と更新を行うことができます。
- 代理登録を行うことができます。
- アカウントロックの対象になりません。
- 一般利用者の新しいパスワードを再設定することができます。

(392)

技術情報

インポートエクスポート画面で、システムモデルに「プリンシパル」は表示されません。これは特別なモデルで、アカウント(juser)のインポートエクスポートにあわせて処理されます。

この内部の動作の仕組みをはじめとする、プリンシパルモデルの詳細はこちらをお読みください。

モデル単位の制御

新しいプリンシパルの追加

ここでは「商品」モデルを対象に、設定を行います。

″環境 〉 権限″ タブ内の ″プリンシパル″ を開きます。ギアアイコンから ″新規″ を選択し、プリンシパルを追加してみます。

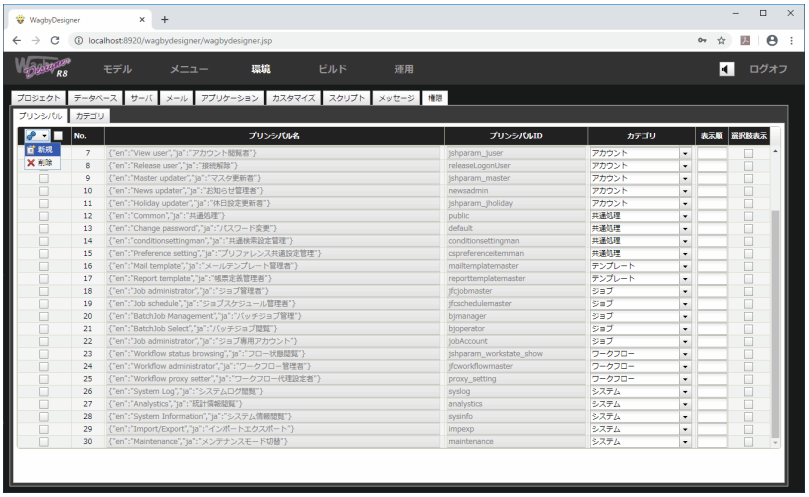


図 機能権限

新しいプリンシパルが用意されました。名前は仮で設定されていますので、適切な名前（英語/日本語）をセットします。

(393)

(392) 1. これは、あるユーザーが表示できる画面をシステム管理者が表示できないという動作は許可しない、ことを意味します。一方で、システム管理者を含む、どのようなプリンシパルも与えられていない権限定義は可能です。この場合は誰も操作できない機能、となります。

(393) 英語名は国際化対応のためと、システム内部での管理用IDとして用いられます。

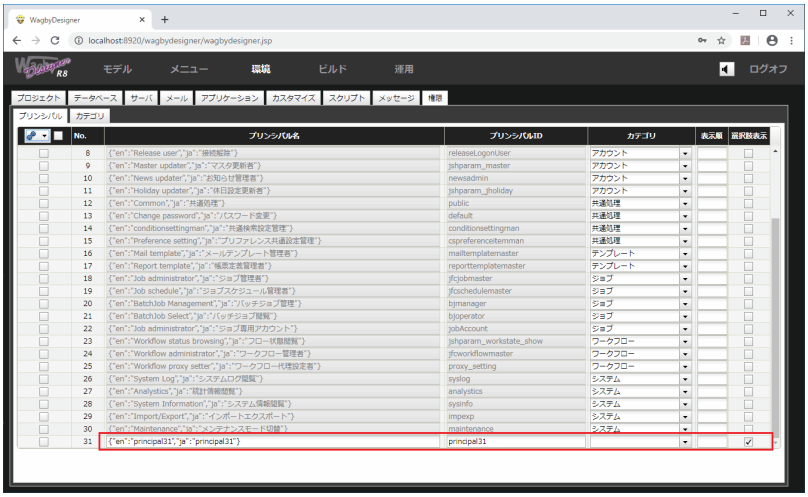


図 プリンシパルの追加(1)

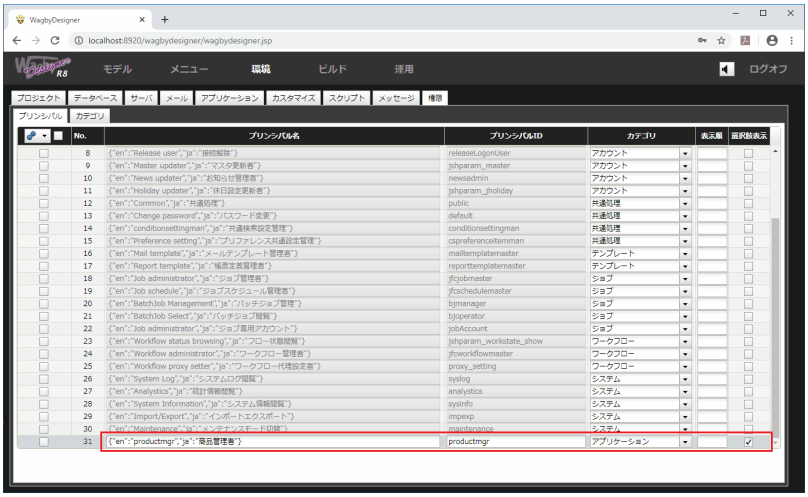


図 プリンシパルの追加(2)

プリンシパル名は国際化に対応した特別表記ルールに基づいて設定してください。[詳細...] ここでは次のように設定しています。

```
{"en":"productmgr","ja":"商品管理者"}
```

また、プリンシパルIDは、プリンシパル名の英語表記と同じとしてください。

プリンシパルにパーミッションを設定する

開発者が「商品」モデルを用意した、とします。このモデルを「商品管理者」プリンシパルで制御できるようにします。

"権限 > 機能権限" タブを開きます。標準では「一般ユーザ」プリンシパルが用意されています。

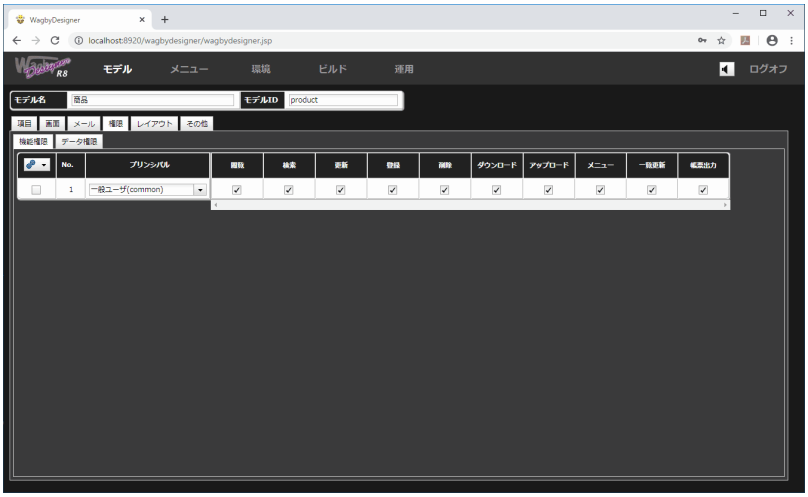


図 モデルに設定されたプリンシパル

ギアアイコンから、″プリンシパル追加″を選びます。

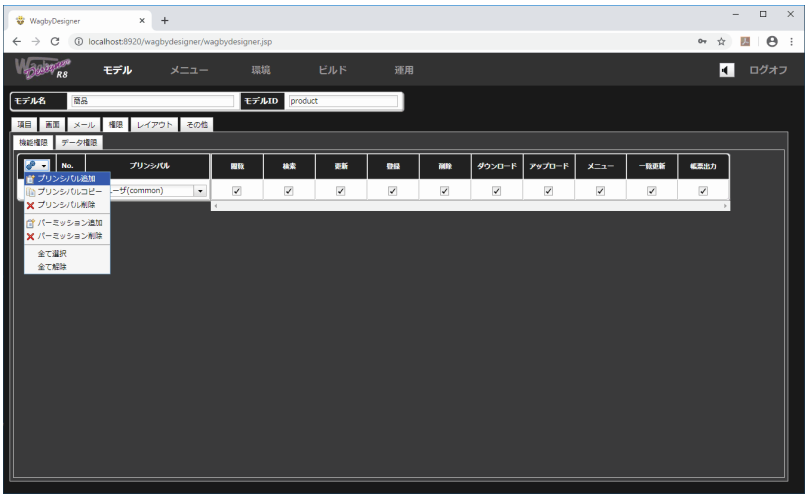


図 プリンシパルの追加(1)

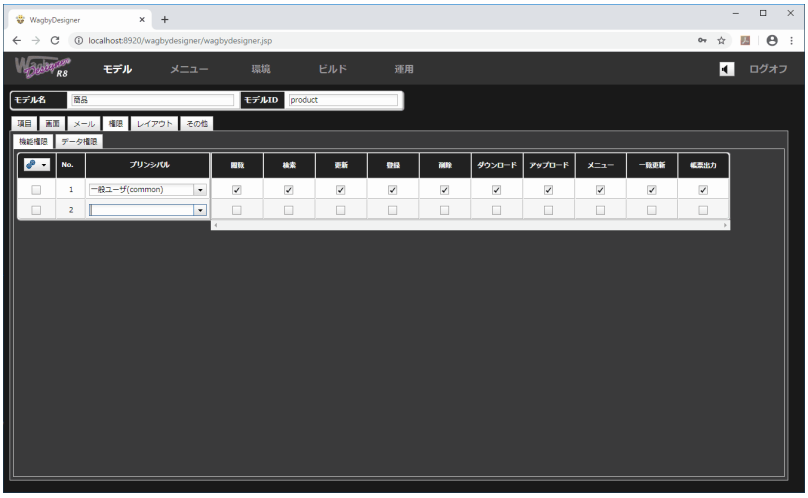


図 プリンシパルの追加(2)

追加するプリンシパルを選択します。環境のプリンシパル一覧で、″選択肢表示″が有効となっているプリンシパルが選択肢に表示されます。

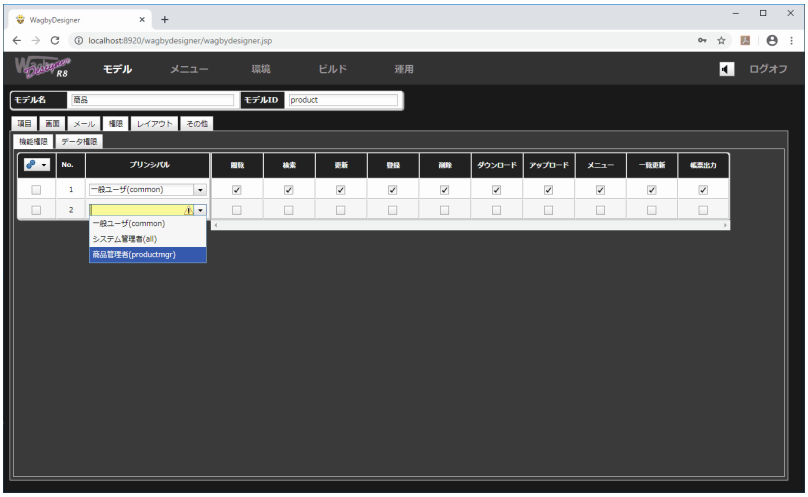


図 プリンシパルの選択

一般ユーザと、商品管理者プリンシパルにパーミッションを割り当てます。ここでは一般ユーザは閲覧のみとし、商品管理者プリンシパルは更新系のパーミッションを割り当てた例を示します。

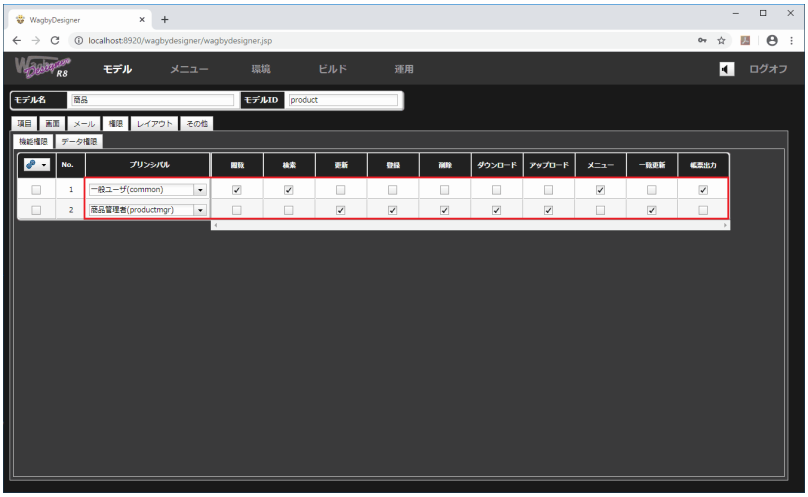


図 プリンシパルにパーミッションを割り当てる

プリンシパルには、一つ以上有効なパーミッションが必要です。パーミッションが未設定（一つも選択されていない）プリンシパルは作成されません。

ビルド後の手順

設定後、ビルドを行い、アプリケーションを起動します。

外部データベースご利用時は、「プリンシパルを追加したときに行うこと」の記載に従いデータのインポートを行ってください。

テストアカウントの作成

以下のテストはトライアルキットではお試しいただけません。アカウントの追加処理を行うためです。開発キットをご利用ください。

はじめに、システム管理者でログオンします。管理処理メニューから「アカウント一覧表示」を選択します。

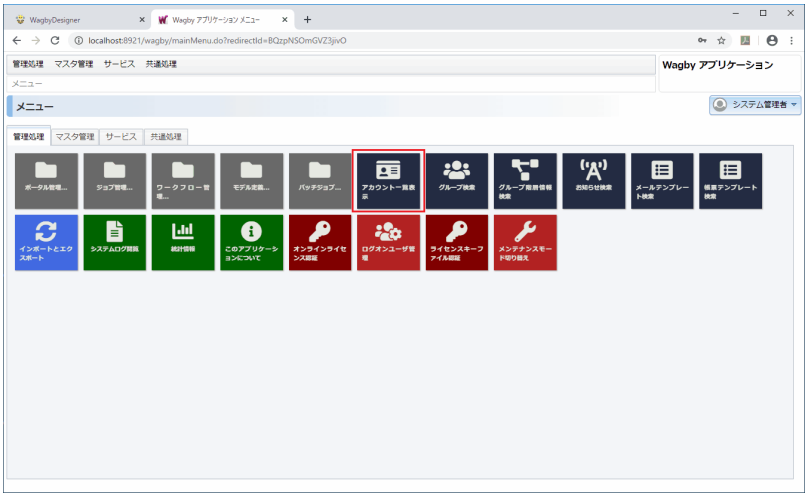


図 アカウント一覧表示へ

アカウントの新規登録を行います。

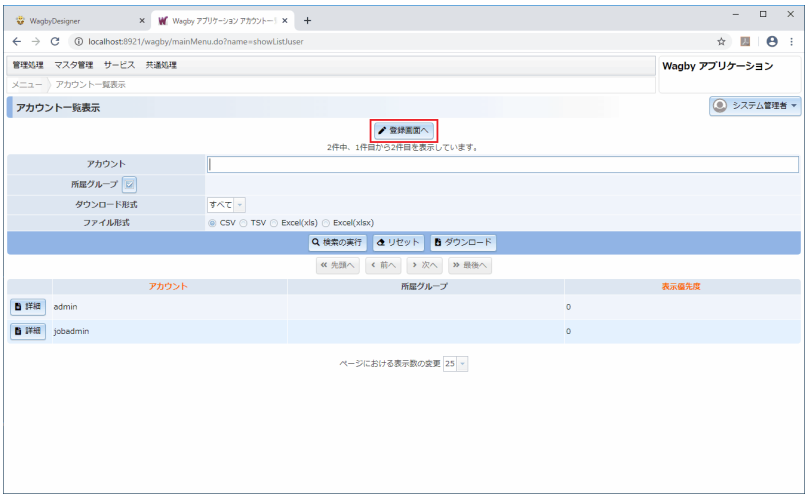


図 アカウント新規登録画面へ

アカウント「user」（一般利用者）を作成します。 プリンシパルに「共通処理」「パスワード変更」「一般ユーザ」を選択します。

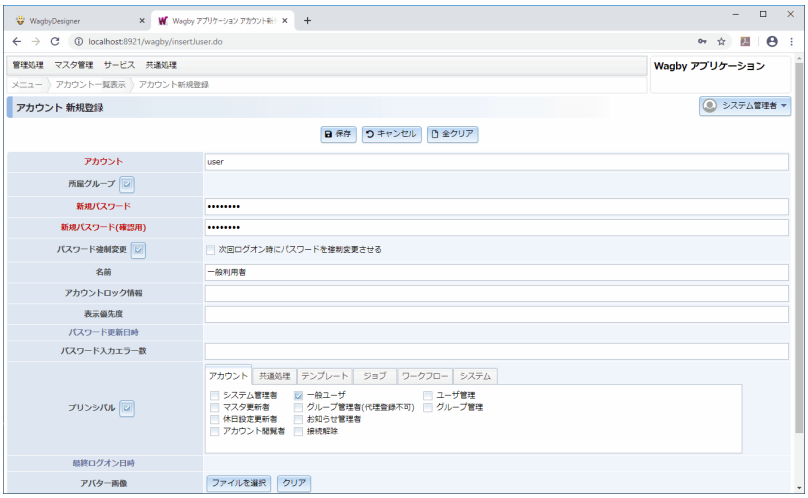


図 アカウントuserの作成(1)

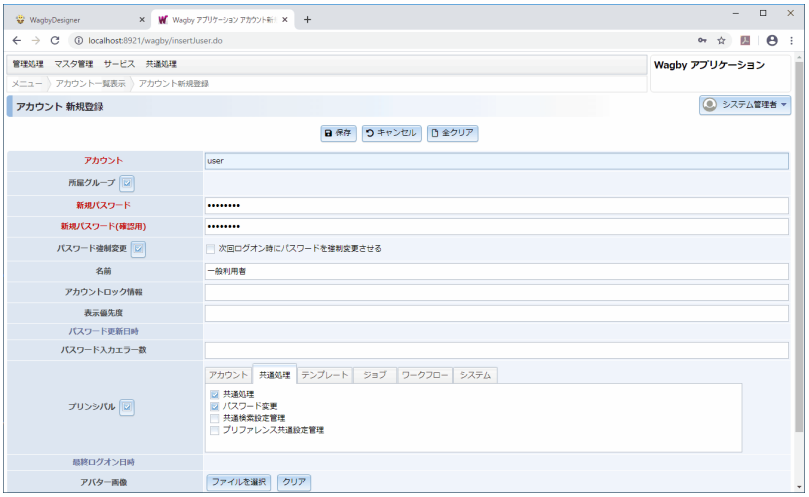


図 アカウントuserの作成(2)

もう一つのアカウント「manager」（マスタデータ管理者）を作成します。 プリンシパルに「共通処理」「パスワード変更」「一般ユーザ」「商品管理者」を選択します。

複数のプリンシパルを選択することができます。このとき「一般ユーザ」かつ「商品管理者」のロールを持ちます。

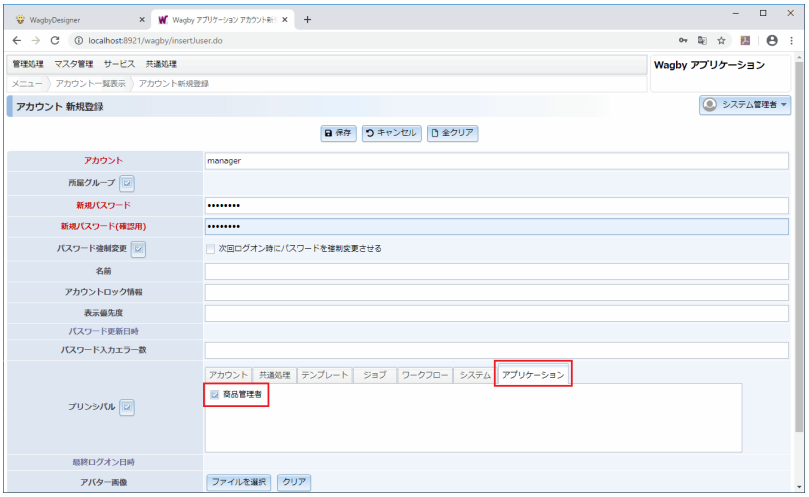


図 アカウントmanagerの作成

複数のプリンシパル（例「一般ユーザ」と「商品管理者」）を有効にしたとき、このアカウントが持つパーミッションは、各プリンシパルに紐づくパーミッションすべて（和集合）となります。

このニつのテスト用アカウントを作成後、システム管理者をログオフします。

実行例

マスタデータ管理者でログオンします。 メニューから「商品検索」を選択します。登録ボタンが表示されています。

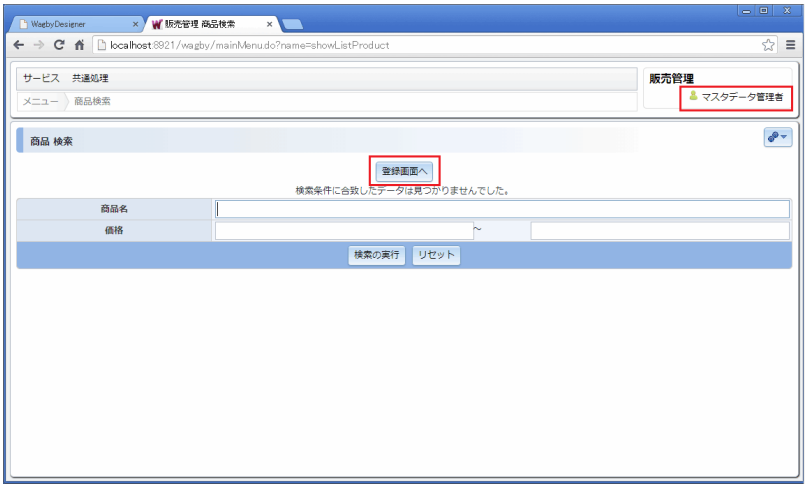


図 商品の登録画面へ

商品データを登録後の詳細画面では、登録・更新・削除といった操作が行えるようになっていることがわかります。

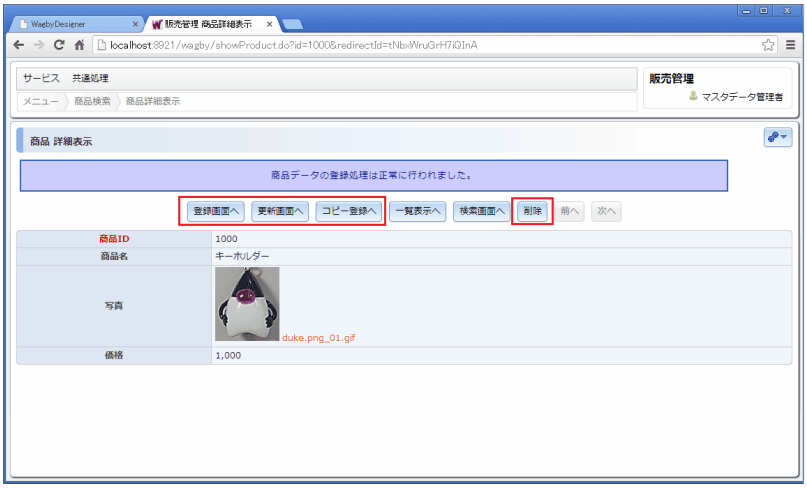


図 更新に関する処理ボタンが表示される

ログオフし、次に一般利用者でログオンします。 メニューから「商品検索」を選択します。登録ボタンは表示されていません。

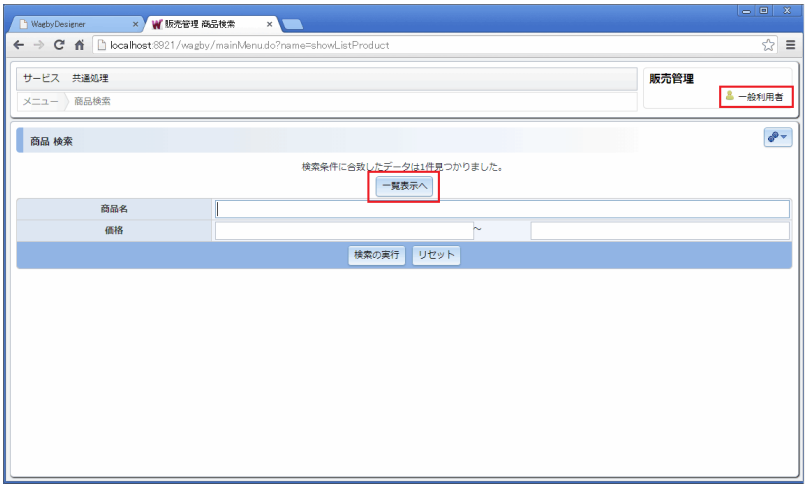


図 商品検索画面

このアカウントは検索と表示のみ行えるようになっています。

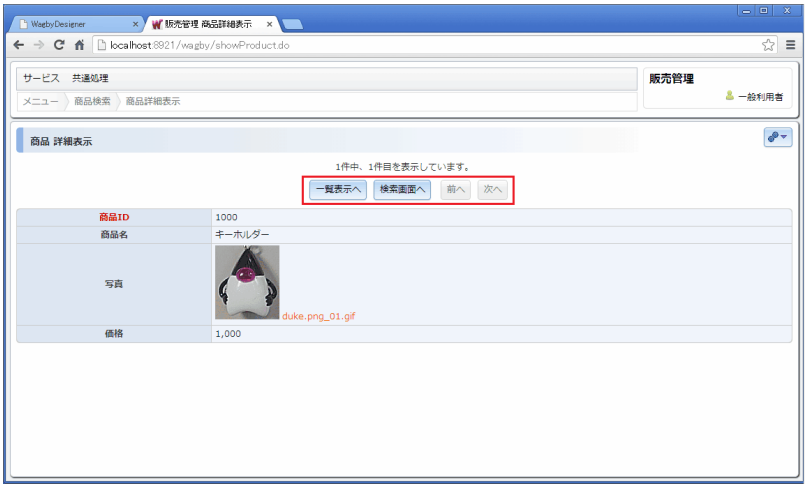


図 更新に関する処理ボタンは表示されない

すべての画面遷移のボタンは、権限管理によって表示制御されます。例えば更新権限をもたない利用者では、「更新」ボタンは表示されない、という動作になります。

権限管理の動作テストでは、システム管理者を用いないようにしてください。（システム管理者はすべての権限を持つという設定になっています。）

メニューの扱い

システム管理者でログオンすると、すべてのメニューが表示されます。それ以外のユーザでログオンすると、許可された権限に対応したメニューだけが表示されます。

権限外の機能を操作しようとした場合

権限を持たない利用者が、許可されていないモデルの操作を行おうとアドレスバーにURLを直接入力するとエラー画面が表示されます。

図 権限エラー

また、ログファイル（wagbyapp/logs/system.log）には次のような記録が残ります。

```
20xx-xx-xx 00:00:00 [INFO jp.jasminesoft.jfc.controller.BaseController checkPermission] (user1) JFC-00108:セキュリティエラーです。権限は [JFCAppPermission [JFCBasePermission Name:juser Actionlist:insert ] ] となっています。
```

ログレベルは ERROR ではなく INFO です。セキュリティエラーの検出は正常な処理であるためです。この検出を行う場合は、ログコード "JFC-00108" で検索してください。

応用

ここでは更新権限の制御方法を紹介しましたが、用途に合わせてパーミッションを組み合わせることができます。例えば「削除」を許可しないプリンシパルや、「CSVダウンロード・アップロード更新」処理専用のプリンシパルなどを作成することができます。

項目単位の閲覧権限（閲覧または不可視）

定義方法

ここでは、顧客モデルの「電子メール」「郵便番号」「住所」の3つの項目に閲覧権限を設定する例を示します。

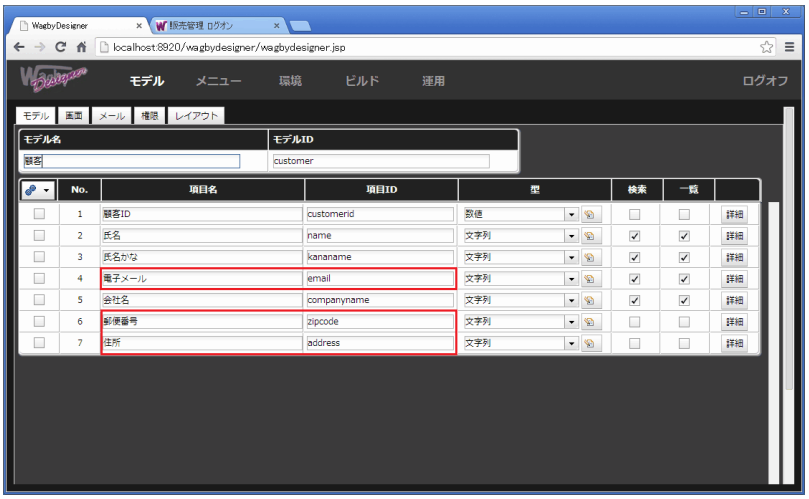


図 モデル定義

項目	閲覧権限
顧客ID	–
氏名	–
氏名かな	–
電子メール	p_mail
会社名	–
郵便番号	p_adr
住所	p_adr

"環境 > プリンシパル" で「個人情報」プリンシパルを作成した、とします。

モデルの "権限 > 機能権限" でプリンシパルを追加します。

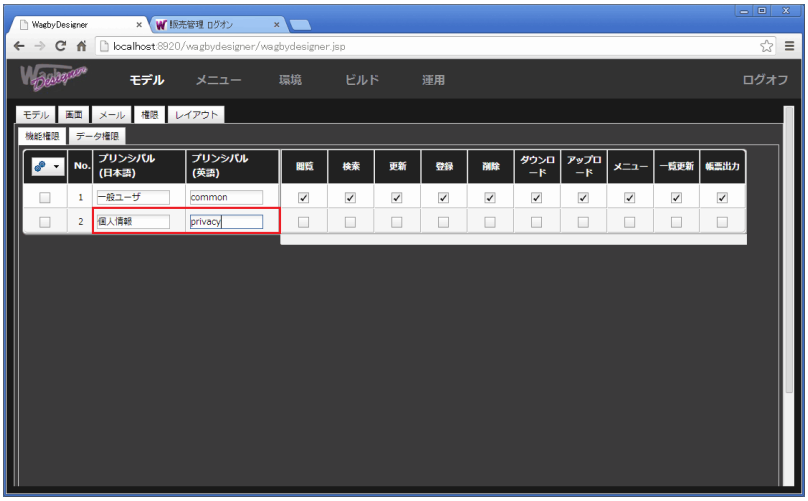


図 プリンシパル「個人情報」を作成する

「個人情報」プリンシパルに加える、新しいパーミッションを追加します。

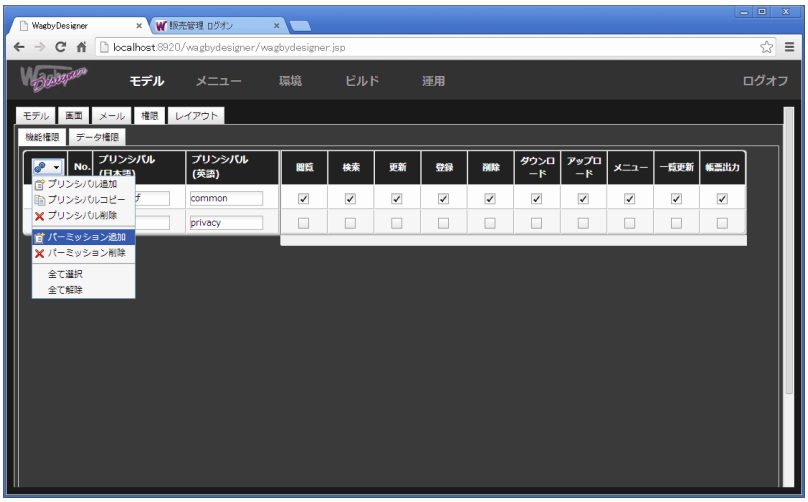


図 プリンシパル「個人情報」を作成する

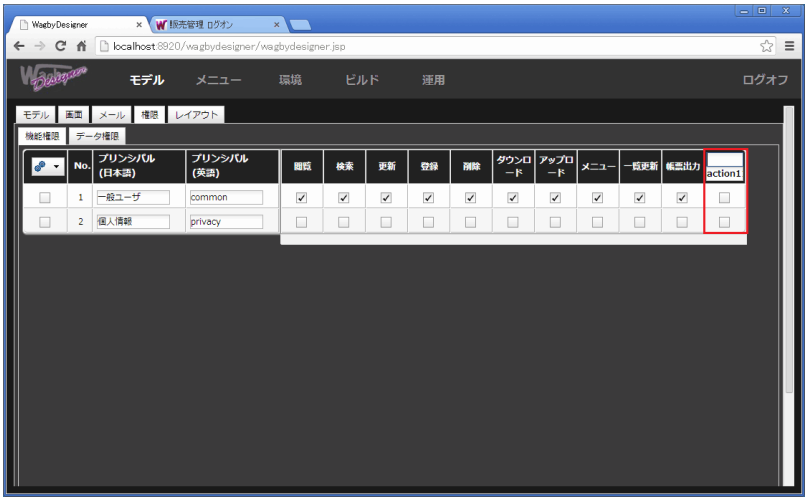


図 パーミッション追加

電子メール項目の閲覧制御を行うためのパーミッション「メール」（英語名　　p_mail）を用意します。「個人情報」プリンシパルで管理します。

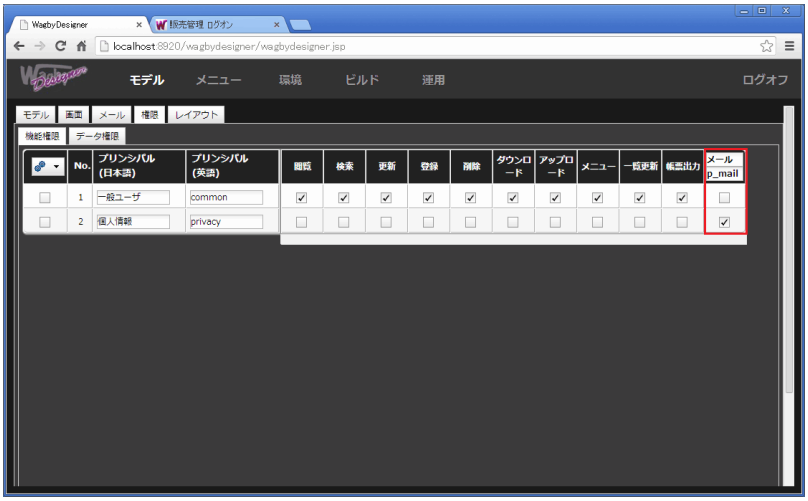


図 パーミッション「メール」を作成する

郵便番号と住所項目の閲覧制御を行うためのパーミッション「住所」（英語名　　p_adr）を用意します。「個人情報」プリンシパルで管理します。

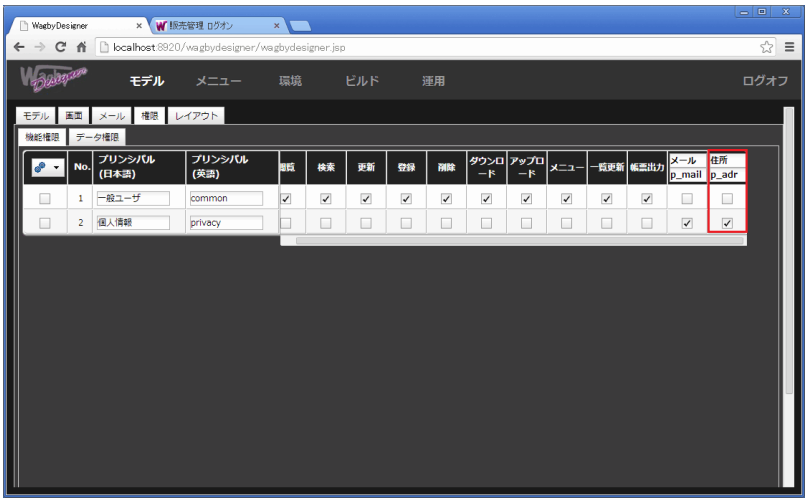


図 パーミッション「住所」を作成する

電子メール項目の詳細定義を開きます。詳細タブ内にある「閲覧権限」に "p_mail" と設定します。 これによって、この項目を閲覧するためには p_mail パーミッションをもったプリンシパルが必要となります。

図 閲覧権限p_mailの設定

同じく、郵便番号および住所項目の詳細定義を開きます。詳細タブ内にある「閲覧権限」に "p_adr" と設定します。 これによって、この項目を閲覧するためには p_adr パーミッションをもったプリンシパルが必要となります。

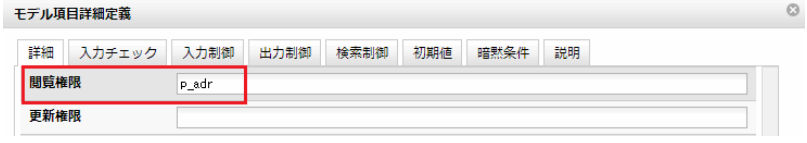


図 閲覧権限p_adrの設定

設定後、ビルドを行い、アプリケーションを起動します。

パーミッションを削除する

追加したパーミッションを削除する場合、次の手順を行ってください。

1. 削除したいパーミッションの枠部分をクリックし、選択状態とします。

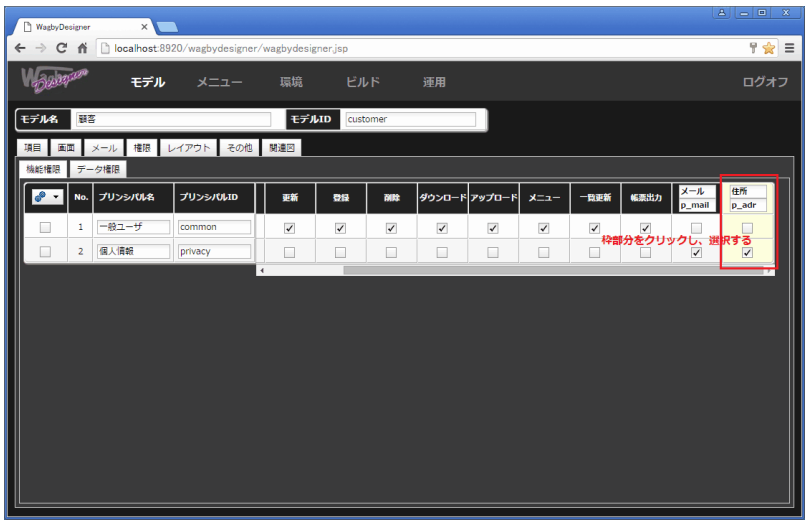


図 削除したいパーミッションを選択する

2. ギアアイコンから、パーミッション削除を行います。

図 パーミッションを削除する

3. パーミッションが削除されました。

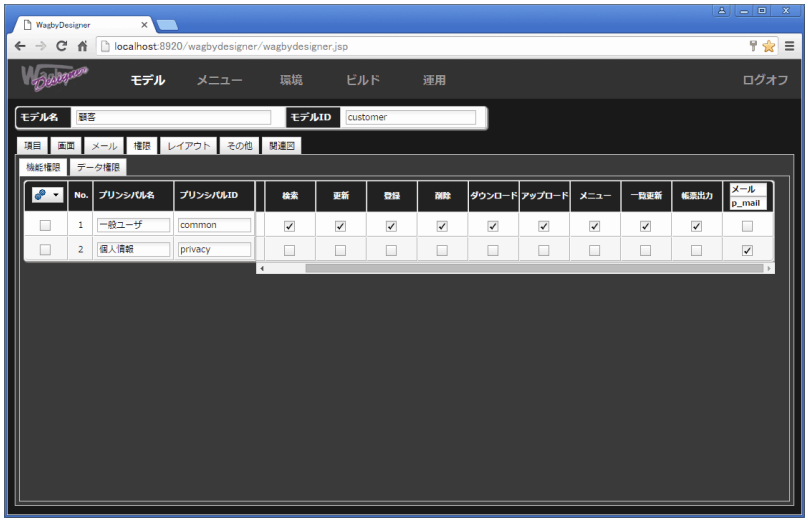


図 パーミッションが削除された

テストアカウントの作成

以下のテストはトライアルキットではお試しいただけません。アカウントの追加処理を行うためです。開発キットをご利用ください。

はじめに、システム管理者でログオンします。管理処理メニューから「アカウント一覧表示」を選択します。

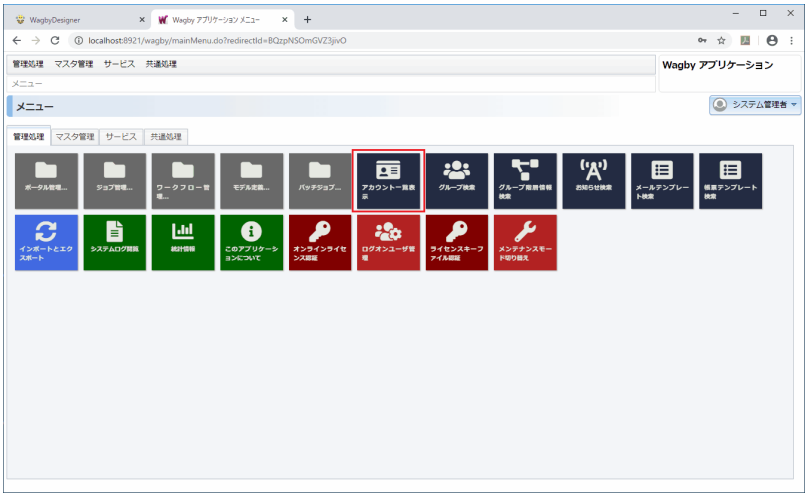


図 アカウント一覧表示へ

アカウントの新規登録を行います。

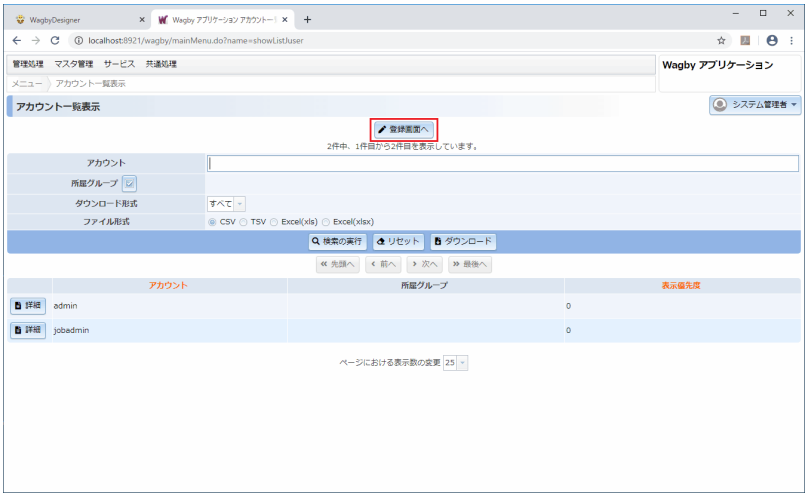


図 アカウント新規登録画面へ

アカウント「user（一般利用者）」と「manager（顧客データ管理者）」を作成します。

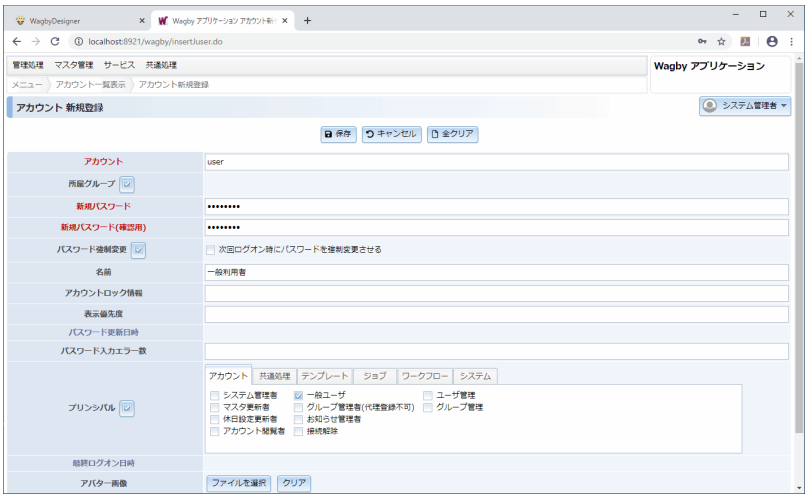


図 アカウントの作成

ユーザ	与えるプリンシパル
user	共通処理、パスワード変更、一般ユーザ
manager	共通処理、パスワード変更、一般ユーザ、個人情報

複数のプリンシパルを選択することができます。このとき「一般ユーザ」かつ「個人情報」のロールを持ちます。

このニつのテスト用アカウントを作成後、システム管理者をログオフします。

例

顧客データ管理者でログオンします。 顧客の新規登録処理を行います。「電子メール」「郵便番号」「住所」の項目は入力可能（かつ閲覧可能）です。

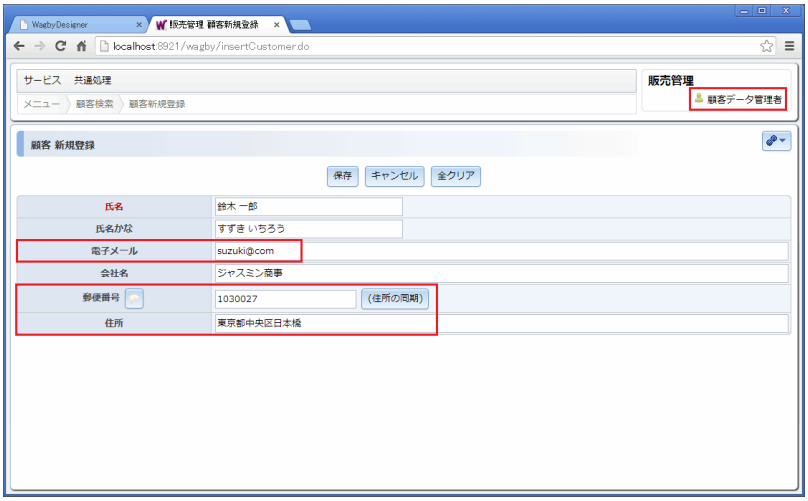


図 顧客データ管理者での操作

次に一般利用者でログオンします。 「電子メール」「郵便番号」「住所」の項目は表示されません。よって入力も閲覧も不可能です。

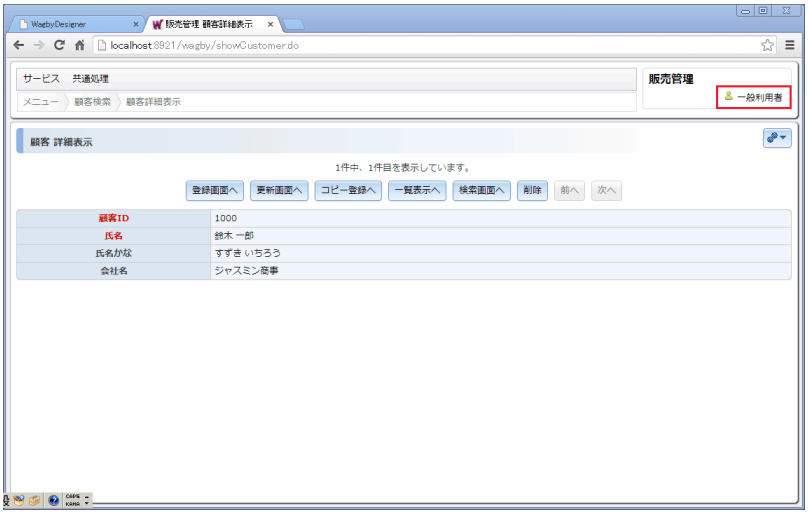


図 一般利用者での操作（表示）

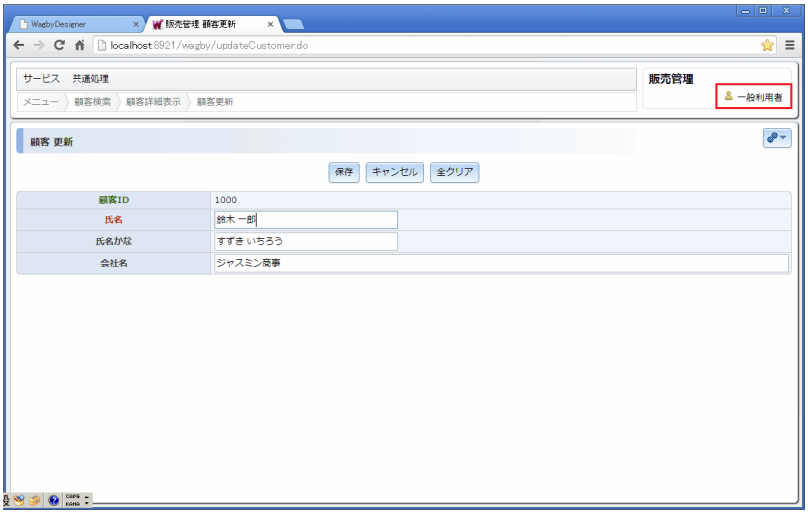


図 一般利用者での操作（更新）

権限管理の動作テストでは、システム管理者を用いないようにしてください。（システム管理者はすべての権限を持つという設定になっています。）

本設定により影響を受ける機能

画面

一覧表示、詳細表示、登録、更新といった、すべての画面に反映されます。権限のない利用者でログオンすると、当該項目は表示されません。（存在することもわかりません。）

検索画面

モデル項目詳細定義ダイアログの「詳細 > 閲覧権限(検索画面)」の式が適用されます。式中で別の項目の値を参照する場合、その項目も検索条件に含まれている必要があります。（隠し項目にして、画面では非表示とすることもできます。）

制約

「検索画面遷移直後の検索処理を止める」設定を有効にすると、最初に表示された画面では式が処理されないため閲覧権限は適用されません。検索を実行すると、閲覧権限の式の結果が適用されます。

帳票出力

閲覧権限を設定した項目は、権限がなければ（プレースホルダを指定していても）データが出力されません。空白で出力されます。

メール送信

権限がなければメール本文にデータが反映されません。

CSV・Excelダウンロード

権限がない場合、項目名も出力されません。

仕様・制約

予約されているパーミッション名

次に示すパーミッション名は Wagby が内部で利用しています。これらの名前を使うことはできません。

select	insert	update	delete	showlis	update	downloa	uploadu	report	menu	default

(394)

項目単位の更新権限（更新可または閲覧のみ）

定義方法

ここでは、顧客モデルの「電子メール」「郵便番号」「住所」の3つの項目に更新権限を設定する例を示します。

(394) 予約されているパーミッション名を使った場合、ビルドはできますが、実行時は制御されません。

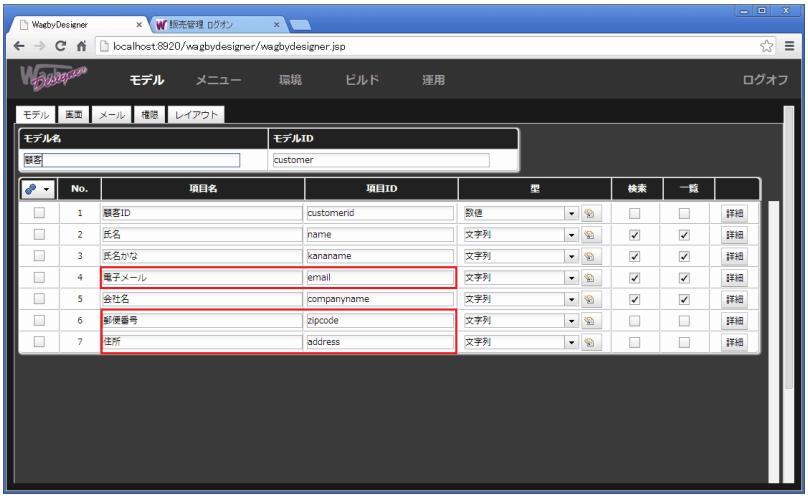


図 モデル定義

項目	更新権限
顧客ID	-
氏名	-
氏名かな	-
電子メール	p_mail
会社名	-
郵便番号	p_adr
住所	p_adr

″環境 > プリンシパル″で「個人情報」プリンシパルを作成した、とします。

モデルの″権限 > 機能権限″でプリンシパルを追加します。

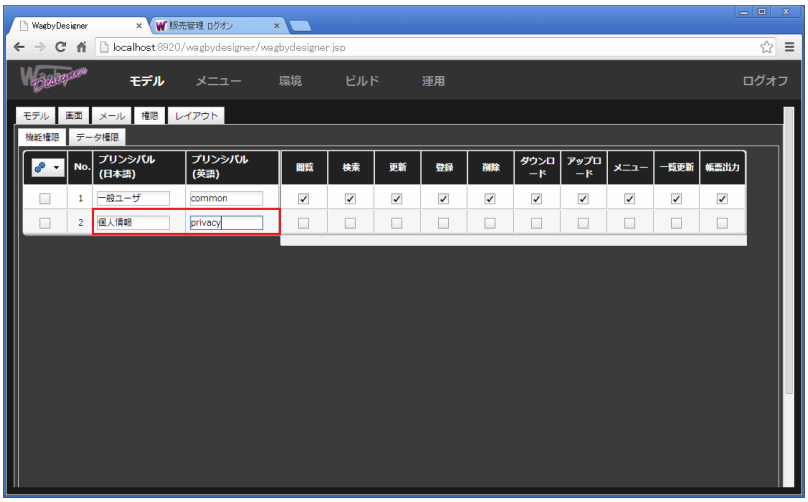


図 プリンシパル「個人情報」を作成する

「個人情報」プリンシパルに加える、新しいパーミッションを追加します。

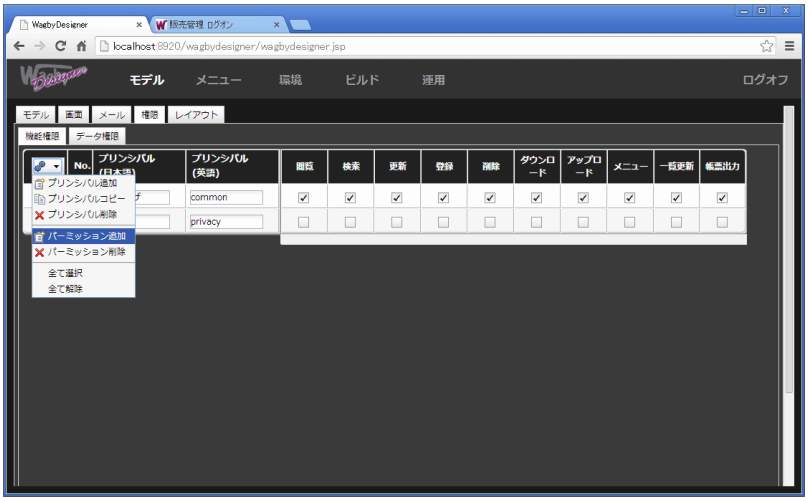


図 プリンシパル「個人情報」を作成する

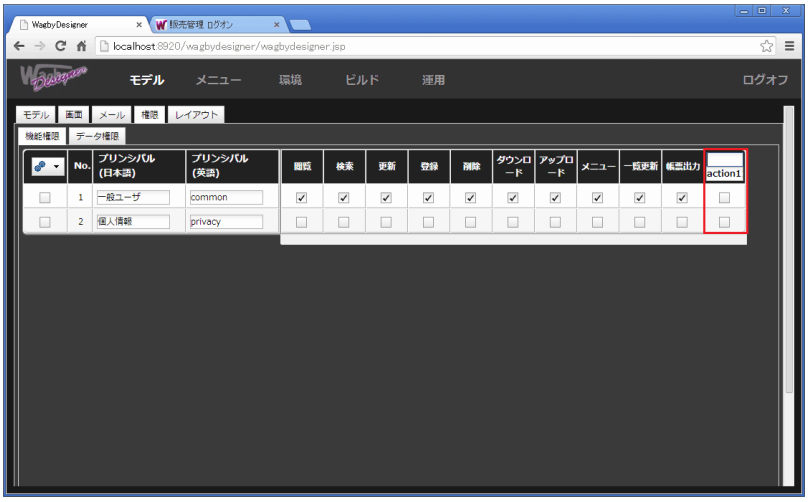


図 パーミッション追加

電子メール項目の閲覧制御を行うためのパーミッション「メール」（英語名　　p_mail）を用意します。「個人情報」プリンシパルで管理します。

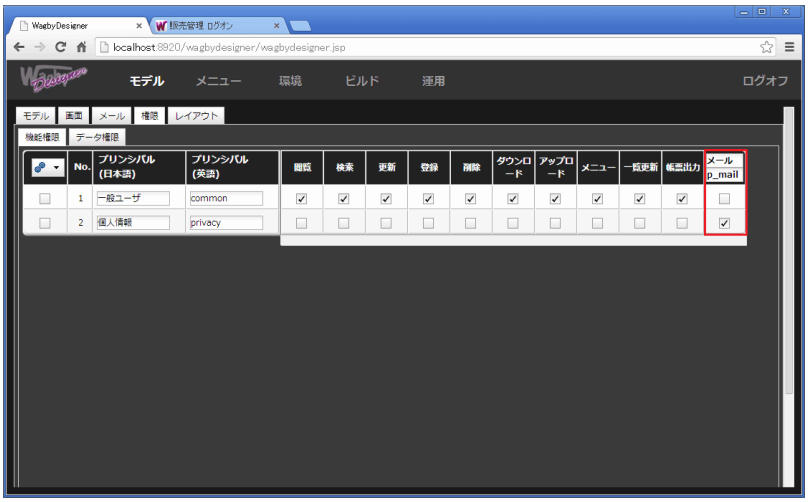


図 パーミッション「メール」を作成する

郵便番号と住所項目の閲覧制御を行うためのパーミッション「住所」（英語名　　p_adr）を用意します。「個人情報」プリンシパルで管理します。

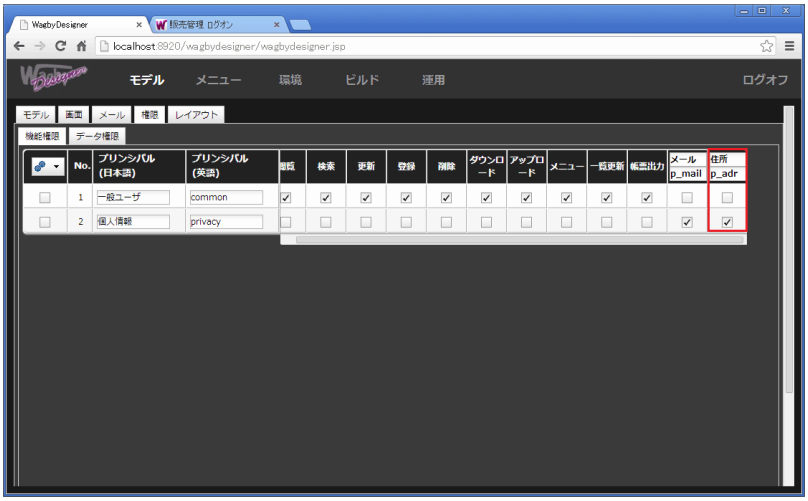


図 パーミッション「住所」を作成する

電子メール項目の詳細定義を開きます。詳細タブ内にある「更新権限」に "p_mail" と設定します。 これによって、この項目を閲覧するためには p_mail パーミッションをもったプリンシパルが必要となります。

図 更新権限p_mailの設定

同じく、郵便番号および住所項目の詳細定義を開きます。詳細タブ内にある「更新権限」に "p_adr" と設定します。 これによって、この項目を閲覧するためには p_adr パーミッションをもったプリンシパルが必要となります。

図 更新権限p_adrの設定

設定後、ビルドを行い、アプリケーションを起動します。

パーミッションを削除する

追加したパーミッションを削除することができます。"項目単位の閲覧権限（閲覧または不可視）> 定義方法 > パーミッションを削除する"をお読みください。

テストアカウントの作成

以下のテストはトライアルキットではお試しいただけません。アカウントの追加処理を行うためです。開発キットをご利用ください。

はじめに、システム管理者でログオンします。管理処理メニューから「アカウント一覧表示」を選択します。

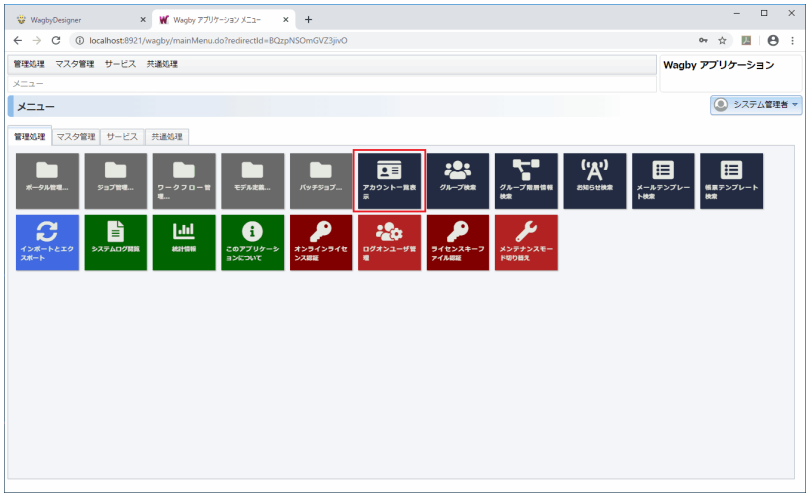


図 アカウント一覧表示へ

アカウントの新規登録を行います。

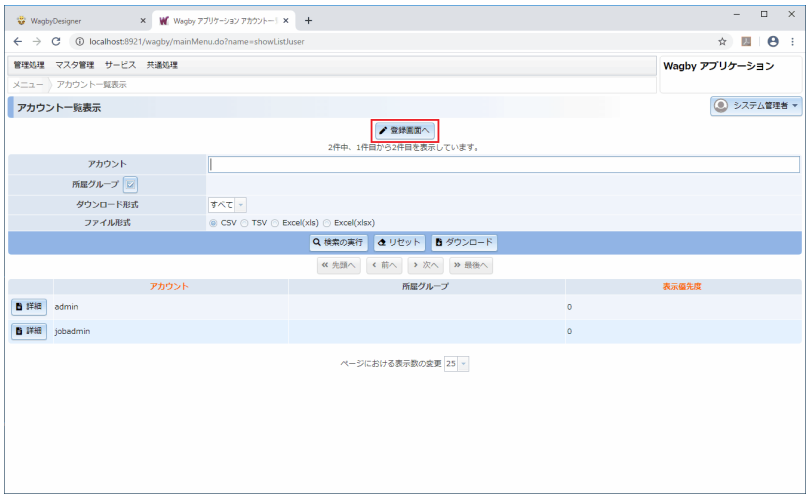


図 アカウント新規登録画面へ

アカウント「user（一般利用者）」と「manager（顧客データ管理者）」を作成します。

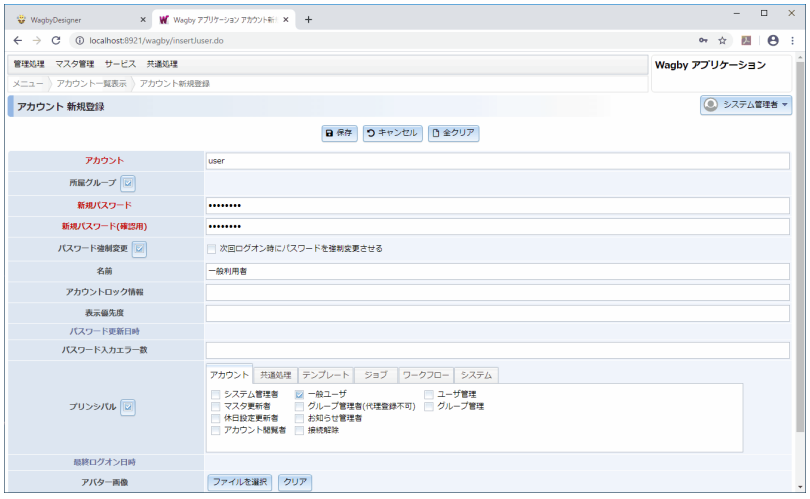


図 一般利用者用アカウントの作成

ユーザ	与えるプリンシパル

user	共通処理、パスワード変更、一般ユーザ
manager	共通処理、パスワード変更、一般ユーザ、個人情報

複数のプリンシパルを選択することができます。このとき「一般ユーザ」かつ「個人情報」のロールを持ちます。

この二つのテスト用アカウントを作成後、システム管理者をログオフします。

例

顧客データ管理者でログオンします。 顧客の新規登録処理を行います。「電子メール」「郵便番号」「住所」の項目は入力可能です。

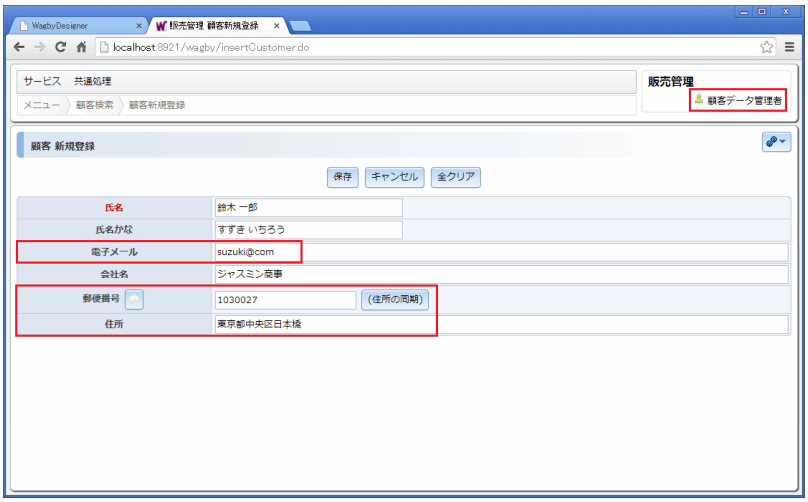


図 顧客データ管理者での操作

次に一般利用者でログオンします。 「電子メール」「郵便番号」「住所」の項目は表示されますが、入力できません。ラベル文字色は青となります。これは権限によって入力を制限されていることを意味します。

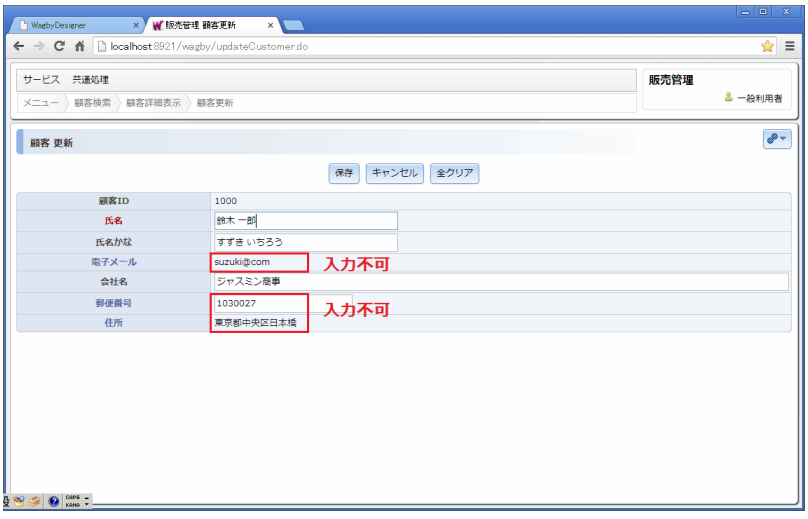


図 一般利用者での更新画面（入力制限項目のラベル文字は青色で示されている）

権限管理の動作テストでは、システム管理者を用いないようにしてください。（システム管理者はすべての権限を持つという設定になっています。）

本設定により影響を受ける機能

画面

登録、更新画面では表示のみとなり、変更できません。一覧表示、詳細表示画面では（更新不可を示すため）ラベルの色が変わりますが、内容は表示されます。検索画面には反映されないため、検索は可能となります。

CSV・Excelダウンロード

権限がない場合、更新することができません。

本設定の影響を受けない機能

固定値

固定値の項目は、権限がない利用者の操作でも値は常に更新されます。

初期値

権限がない利用者の操作でも初期値はセットされます。

仕様・制約

予約されているパーミッション名

項目単位の閲覧権限（閲覧または不可視）> 仕様・制約をお読みください。

項目単位の閲覧更新権限の同時設定

定義方法

ここでは、顧客モデルの「電子メール」項目に閲覧権限と更新権限を同時に設定します。

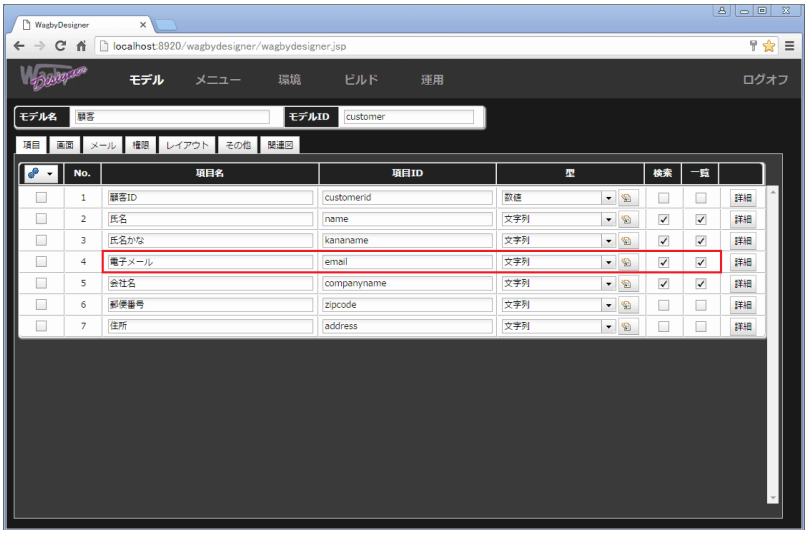


図 モデル定義

「権限」タブ内の「機能権限」タブを開きます。二つのプリンシパル「個人情報R」「個人情報W」を追加しました。これらはそれぞれ、パーミッション「メールR」「メールW」を保持します。

"R" は読み込み、"W" は書き込みを意図して命名しています。

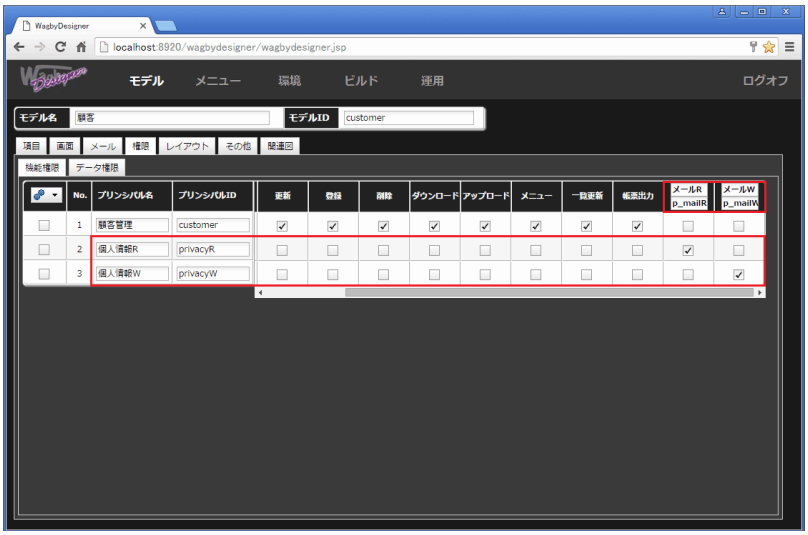


図 個人情報Rと個人情報Wプリンシパルを追加する

「電子メール」項目に、閲覧権限と更新権限を同時に設定します。

項目に「閲覧権限」と「更新権限」を同時に設定時には、閲覧権限のみをもつ場合、閲覧できるが更新はできない（読み込み専用）の項目として機能します。

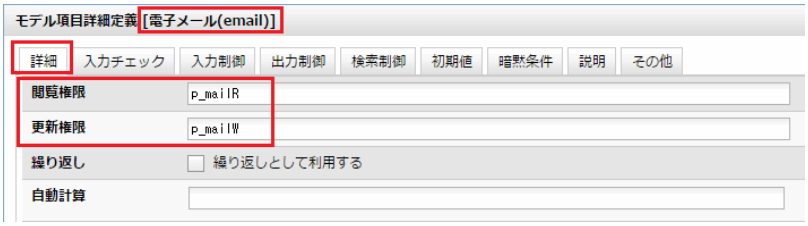


図 閲覧権限と更新権限の設定

例

以下のテストはトライアルキットではお試しいただけません。アカウントの追加処理を行うためです。開発キットをご利用ください。

アカウント「staff」と「manager」を用意しました。

「staff」は "顧客管理" および "個人情報R" プリンシパルをもちます。

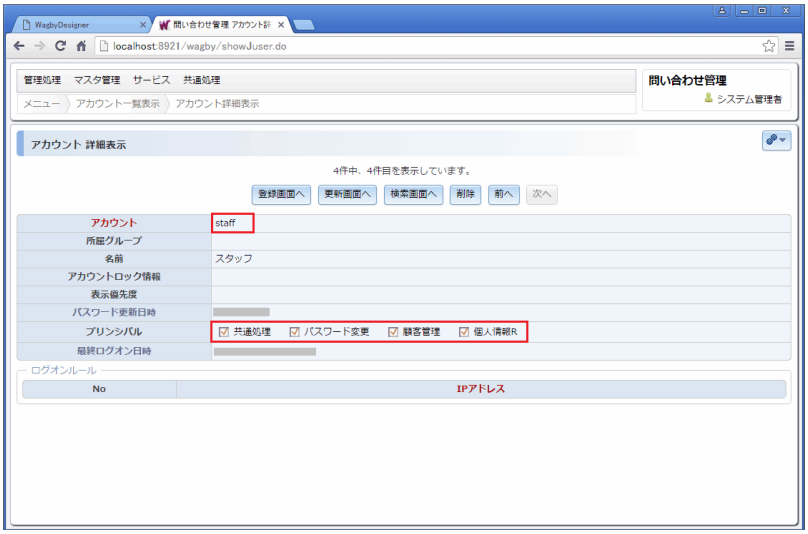


図 アカウントstaffのプリンシパル

「manager」は″顧客管理″および″個人情報R″そして″個人情報W″プリンシパルをもちます。

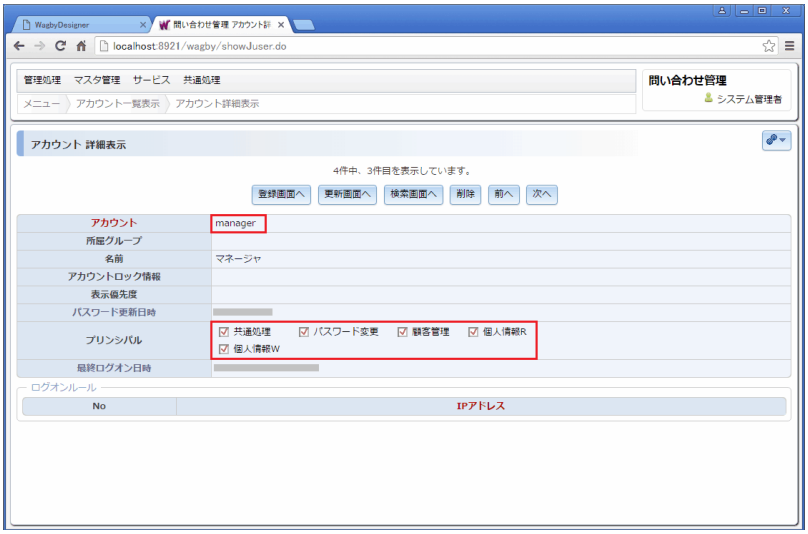


図 アカウントmanagerのプリンシパル

staffでの実行

アカウント staff で、顧客の更新画面を開いた例です。「電子メール」項目は閲覧できますが、更新はできません。

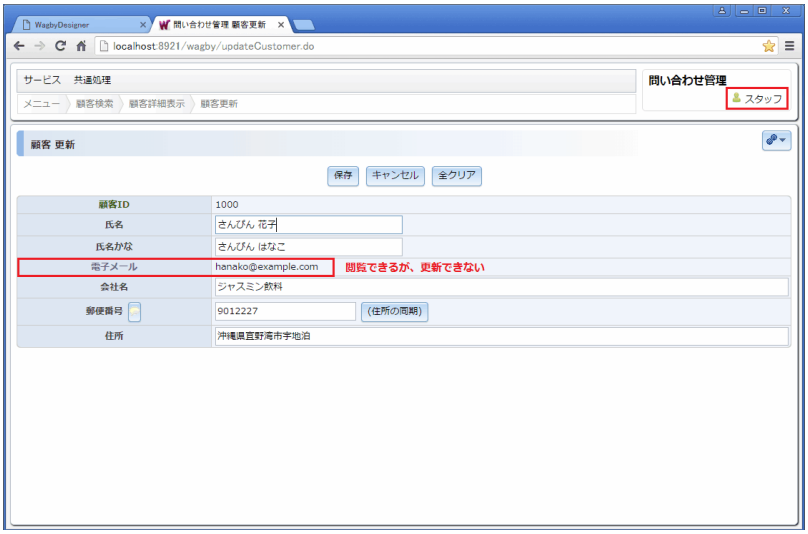


図 アカウントstaffで、顧客の更新画面を開く

managerでの実行

アカウント manager で、顧客の更新画面を開いた例です。「電子メール」項目は閲覧ならびに更新ができます。

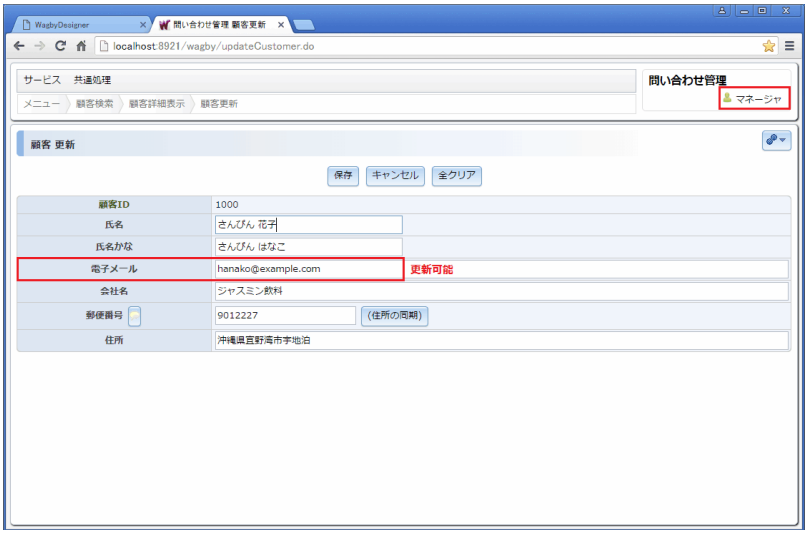


図 アカウントmanagerで、顧客の更新画面を開く

条件によって権限の適用可否を変更する

概要

対象となるモデルの項目の値によって、指定した権限を適用するか、しないかを設定することができます。

閲覧権限に条件式を設定する

権限をもったアカウントで操作し、かつ条件式が true になったとき、当該項目は閲覧可能になります。条件式が false であれば項目は不可視（表示されない）になります。

更新権限に条件式を設定する

権限をもったアカウントで操作し、かつ条件式が true になったとき、当該項目は更新可能になります。条件式が false であれば項目は更新不可(閲覧のみ)になります。

定義方法

ここでは、顧客モデルの「電子メール」項目の更新権限に、条件を設定してみます。

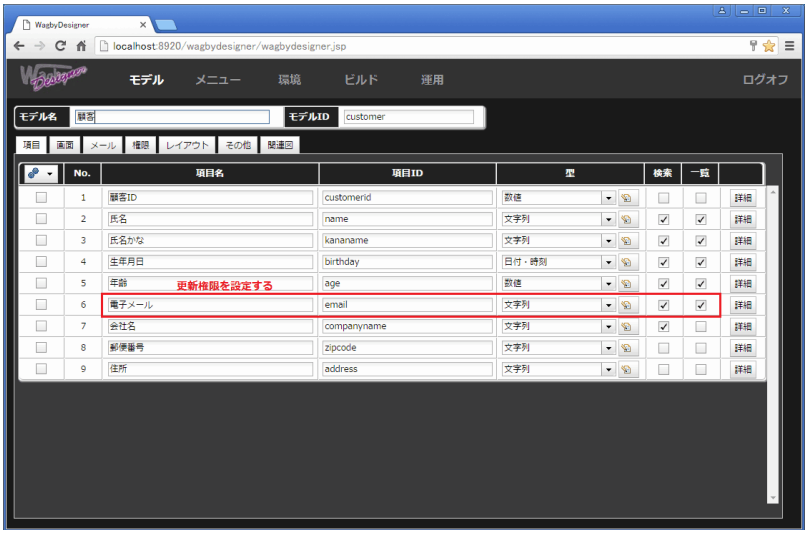


図 モデル定義

「権限」タブ内の「機能権限」タブを開きます。二つのプリンシパル「個人情報R」「個人情報W」を追加しました。これらはそれぞれ、パーミッション「メールR」「メールW」を保持します。

"R" は読み込み、"W" は書き込みを意図して命名しています。

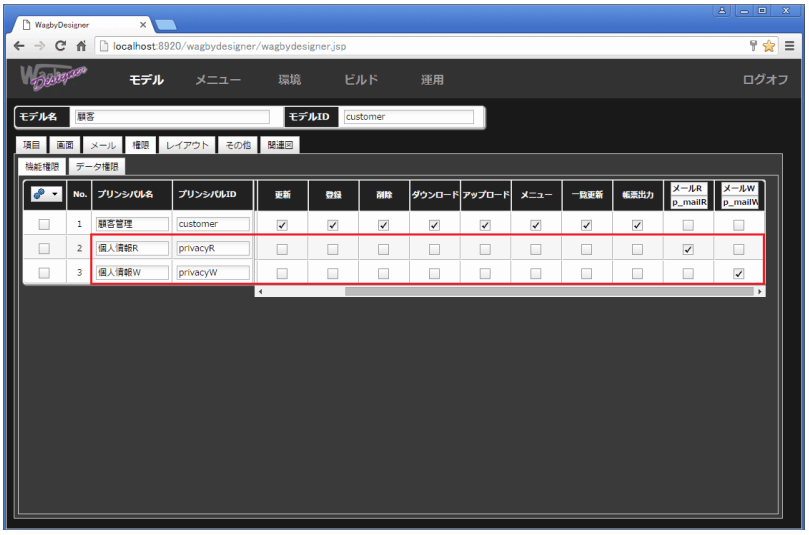


図 個人情報Rと個人情報Wプリンシパルを追加する

「電子メール」項目に、閲覧権限と更新権限を設定します。[詳細...]

ここに条件を指定することができます。具体的には、項目に対する権限指定入力欄に「;(セミコロン)」を記述します。この区切り文字の右側が条件式となります。条件式の戻り値はtrueまたはfalseのいずれかとします。

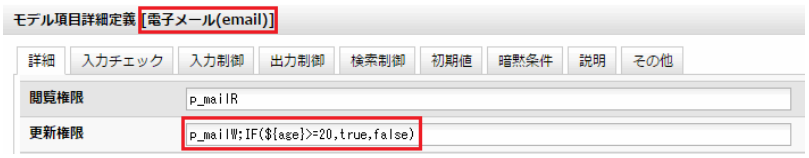


図 条件つき更新権限の設定

この式は、年齢(age)が20以上の場合にtrueを返すとしています。そのため、20歳以上は更新可能で、それ以下は（更新権限を有していても）更新不可となります。

```
p_mailW: IF(${age}>=20,true,false)
```

例

以下のテストはトライアルキットではお試しいただけません。アカウントの追加処理を行うためです。開発キットをご利用ください。

アカウント「manager」を用意しました。「manager」は"顧客管理"および"個人情報R"そして"個人情報W"プリンシパルをもちます。

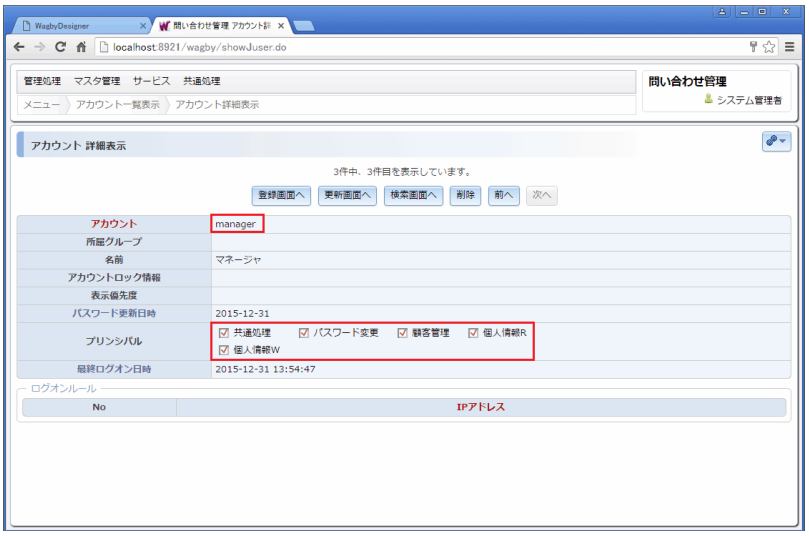


図 アカウントmanagerのプリンシパル

「manager」でログオンし、顧客データを更新した例です。年齢は20歳以上という条件を満たすため、「電子メール」項目は更新できます。

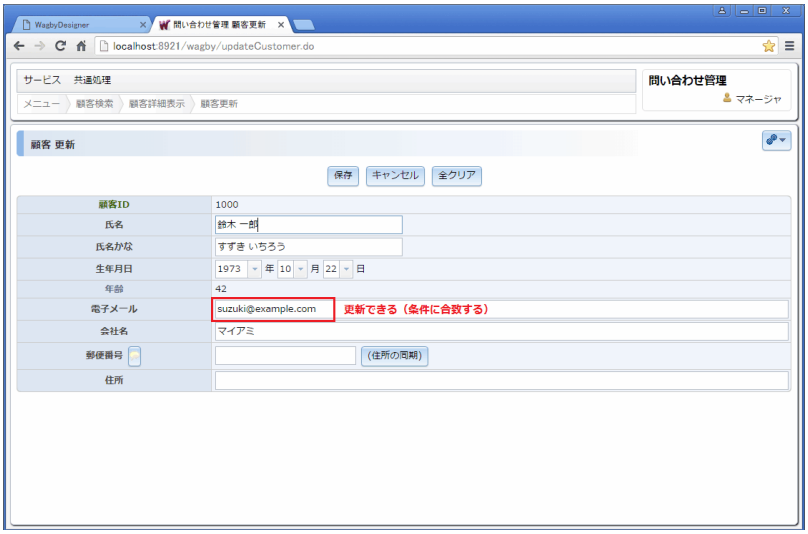


図 顧客データの更新(1)

もう一つの顧客データの更新例です。年齢は20歳以上という条件を満たさないため、「電子メール」項目は更新できません。

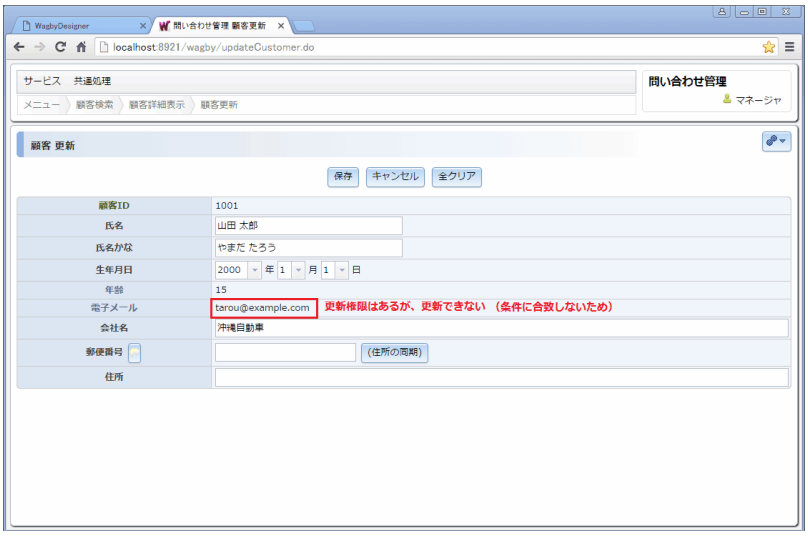


図 顧客データの更新(2)

条件式の対象となる項目（上記の例では「年齢」）の値が変更されたら、画面を再描画するという設定と併用することができます。 これにより、登録・更新画面で値を変更すると自動的に項目の表示が切り替えられるようになります。

登録画面では入力可能だが、更新画面では入力不可としたい

SCREENTYPE関数とHASMODELPERMISSION関数を利用して、画面によって権限をコントロールする例を紹介します。

```
select;
  IF(EXACT(SCREENTYPE(),"insert"),
  true,
  HASMODELPERMISSION("perm_koushin","product"))
```

この式の説明は次のとおりです。

- select は Wagby が提供する標準のパーミッションです。[詳細...]
- 画面種別(SCREENTYPE())が "insert"の場合(登録画面の場合)、true(つまり編集可能)となります。

・　画面種別(SCREENTYPE())が "insert"以外の場合(更新画面の場合)、perm_koushin 権限を有する場合は true(編集可能)、権限を持っていない場合は false(編集不可能)となります。

更新権限が無効になったときの動作 (8.3.0)

更新権限が無効となり、入力不可となった場合に、その値を空白で上書きすることができます。

以下の例で「事業所名」項目に着目してください。この項目は、事業所名が "その他" の場合に入力可としています。

図 初期状態

事業所名を選択すると、都道府県および市町村名が参照連動するようにしています。

図 事業所名から都道府県と市町村名を参照連動で求める

ここで事業所名を "その他" とすると、「事業所名（その他の場合）」が入力できるようにしています。

図 事業所名をその他とした場合

そのあと事業所名を再度、選択します。このとき「事業所名（その他の場合）」の値を空白で上書きしています。

図 事業所名（その他の場合）が再び入力不可となった

設定方法

「事業所名（その他の場合）」に更新権限を指定しています。ここで "空白で上書き" を指定すると、更新権限が無効になったとき、この値を空白で上書きします。

図 空白で上書きを指定する。

仕様・制約

入力値によって条件式を適用したい場合

条件式の対象となる項目の値が変更されたら、画面を再描画するという設定と併用してください。 これにより、登録・更新画面で値を変更すると自動的に項目の表示が切り替えられるようになります。

繰り返し項目・繰り返しコンテナ項目

繰り返し項目および繰り返しコンテナに条件付き権限設定を適用した場合、次のような動作となります。

- 新規登録時は、初期値が指定されていれば、その値がセットされます。
- 更新画面では、登録した値がそのまま使われます。
- 権限が有効になって編集可能となった場合、値が空白であれば初期値がセットされます。値が入っている状態であれば、その値がそのまま使われます。
- 読み込み専用となった状態で、ソートボタンは非表示となります。
- 繰り返しコンテナ内項目の入力チェックと条件付き権限指定を組み合わせたとき、入力可のレコードについてのみ、入力チェックを行います。
- 条件式に繰り返しコンテナ内の項目を指定した場合、ラベル表示の判定は先頭のデータを用いて行われます。

検索条件

条件付き権限設定を行った項目を検索画面の検索条件に加えた場合、条件部は無視されます。

一覧表示

条件付き権限設定を行った項目を一覧表示画面に加えた場合、次のようになります。

- 閲覧権限の条件を満たさない場合は、一覧表示画面に項目が表示されません。
- (閲覧条件を満たすが)更新権限の条件を満たさない場合は、一覧表示画面で列の見出しがすべて表示され、内容が表示されません。
- 一覧表示画面に条件付き権限設定を行った項目（上記のサンプルにおける「メールアドレス」）を含めた場合、条件部の項目（上記のサンプルでは「年齢」項目）も、一覧表示画面に含める必要があります。これは、条件判断処理で必要となるためです。
- 一覧表示画面の条件付き権限管理で、繰り返しコンテナ内の項目を(一覧表示の項目として)出力する場合、条件式はコンテナの先頭のデータを用いて判定されます。

グループ権限

概要

あるグループに所属しているアカウント同士では登録したデータの参照・更新は可能だが、別のグループに所属するアカウントからは参照のみで更新はさせない、といったような使い方ができます。

これはモデル単位で設定します。

グループモデル - jgroup

グループ権限管理は Wagby が提供する「グループ」モデルである jgroup を用います。ログオンアカウントを管理するモデルである juser と連携しており、ログオンアカウントは複数のグループに所属することができます。

グループの作成方法は"アカウント > グループの使い方"をお読みください。

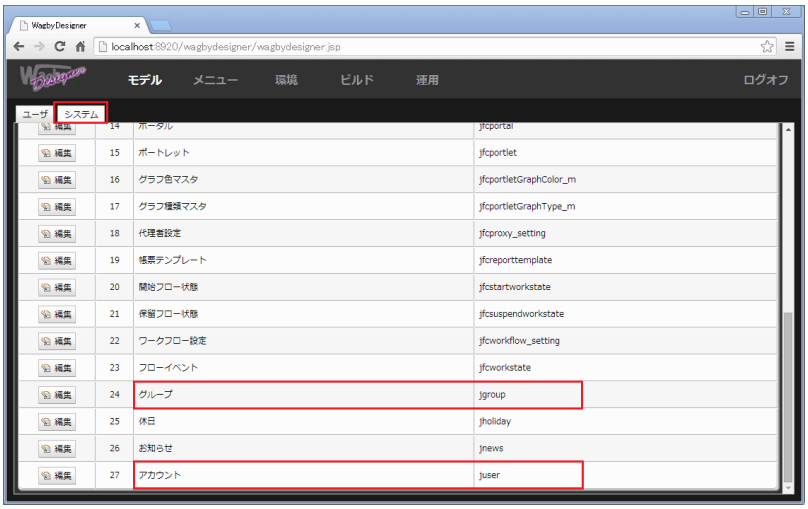


図 グループモデルとアカウントモデル

グループ権限のパターン

モデル毎に「R:読み取り権限」「W：書き込み権限」の2つの権限を組み合わせたパターンを選択します。

図 グループ権限の設定

各パターンはそれぞれ下表のように設定されています。

パターン	データ登録者		同一の所属グループ		その他の所属グループ	
	R	W	R	W	R	W
パターン1	○	○	-	-	-	-
パターン2	○	○	○	-	-	-
パターン3	○	○	○	○	-	-
パターン4	○	○	○	-	○	-
パターン5	○	○	○	○	○	-

パターン6（初期値）	○	○	○	○	○	○

パターンは制約の厳しい順に並んでいます。初期値は「パターン6」（制約なし）が設定されています。

システム管理者は（パターンの設定に関わらず）すべてのデータの閲覧と更新を行うことができます。

グループ権限の動作の仕組み

本機能を使用すると、Wagby　は関連するモデルのテーブル定義に（内部で利用するグループ権限管理用の）項目「データ所有者」及び「データ所有者の所属グループ」を追加します。

そのため、設定変更後はテーブル構造が変わります。既存のテーブルをすべて削除したあと、テーブルの再作成を行ってください。

データ登録後にユーザの所属グループを変更した場合

データ登録後にユーザの所属グループを変更しても既存のデータは影響を受けません。「データ登録時に設定された所有者の所属グループ」が維持されます。

これを「パターン5」のグループ権限を例に説明します。

用意するグループ

グループID（主キー）	グループ名
1000	総務部
1001	営業部
1002	技術開発部

用意するアカウント

アカウント	名前	所属グループ
satou	佐藤	総務部
suzuki	鈴木	総務部
yamada	山田	技術開発部

佐藤がデータを登録した場合

佐藤が顧客データ(ID:1)を登録します。

顧客ID	作成者（データ所有者）	データ所有者の所属グループ	佐藤の操作		鈴木の操作		山田の操作	
			R	W	R	W	R	W
1	satou	1000	○	○	○	○	○	-

佐藤はデータを登録した本人のため、読み書きできます。　鈴木は同一グループのため、読み書きできます。 山田は異なるグループのため、読み取りのみ許可されます。

佐藤の所属グループを変更した後

"satou"（佐藤）の所属グループを技術開発部に変更します。この操作はすでに登録済みのデータには影響がありません。

顧客ID	作成者（データ所有者）	データ所有者の所属グループ	佐藤の操作		鈴木の操作		山田の操作	
			R	W	R	W	R	W
1	satou	1000	○	○	○	○	○	-

佐藤のグループを変更したにも関わらず、鈴木が読み書きでき、山田は読み取りのみです。 これは、登録済みデータは登録時の所属グループのIDを保持しているためです。

佐藤が新しいデータを登録した場合

所属グループ変更後に登録を行ったデータについては、変更後のグループが適用されます。

顧客ID	作成者（データ所有者）	データ所有者の所属グループ	佐藤の操作		鈴木の操作		山田の操作	
			R	W	R	W	R	W
2	satou	1002	○	○	○	-	○	○

佐藤が過去に登録したデータの「データ所有者」を変更したい場合

顧客IDが1のデータは、佐藤の所属グループが変わっても（データ作成者が佐藤であるため）佐藤はそのまま読み書き可能です。運用ポリシー上、データ所有者を変更したいという場合、次の手順を行ってください。

1. システム管理者またはグループ管理者（後述）でログオンします。
2. 当該データの更新画面を開きます。
3. 「データ所有者」という項目が表示されますので、こちらを佐藤以外のアカウントへ修正します。このとき、データ所有者の所属グループも同時に変更されます。

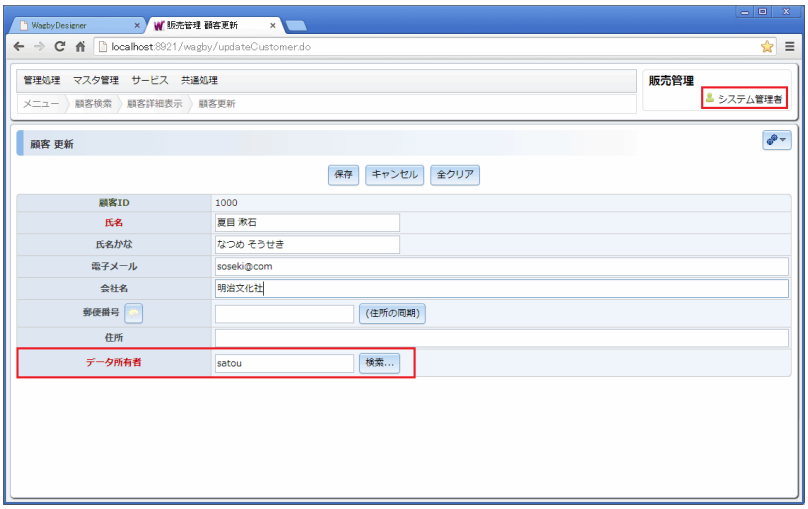


図 データ所有者を変更する

運用上、"データ所有者(総務部)" というアカウントを用意しておくという考え方もあります。データ所有者の変更は、このアカウントに付け替えるというものです。（誰かがこのアカウントを使ってシステムを操作することはない、とします。）

佐藤がデータ所有者であることはそのままに、データ所有者の所属グループだけを変更したい場合 (1)

次の手順で変更します。

1. システム管理者またはグループ管理者（後述）でログオンします。
2. 当該データの更新画面を開きます。
3. 「データ所有者」という項目が表示されますので、こちらをいったん佐藤以外のアカウントへ修正します。（例：admin）(395)
4. 再度、更新画面を開きます。「データ所有者」に再び"satou"を選択し、保存します。

佐藤がデータ所有者であることはそのままに、データ所有者の所属グループだけを変更したい場合 (2) 一括変更

上の説明は一件ごと画面操作で変更するものでしたが、複数のデータを一括変更する場合は次の方法を利用できます。

(395) 「データ所有者の所属グループ」はWagbyの参照連動自モデル保存機能を使っています。この値を修正するために、参照先を変更する必要があります。

1. "satou"の所属グループをAに変更します。
2. "satou"が所有者となっているデータを Excel または CSV ファイルとしてダウンロードします。
3. ダウンロードしたデータの「データ所有者」を"satou"からシステム管理者(admin)へ変更してアップロード更新を行います。なお admin への変更は一時的なものです。
4. 続けて、ダウンロードデータの「データ所有者」を"satou"へ戻し、再度アップロード更新を行います。

グループ管理者

各グループには、グループ内のデータに対して特別な権限を持つ「グループ管理者」を用意することができます。

「グループ管理者」はプリンシパルとして提供されますが、これが選択できるのはグループ管理者の権限が（同一の所属）グループの権限を超える運用を行うときに限定されます。具体的には次のパターンです。

パターン	同一の所属グループ		グループ管理者の権限
	R	W	
パターン1	−	−	RW
パターン1	−	−	R
パターン2	○	−	RW
パターン4	○	−	RW

パターン2を例にあげて説明します。同一グループに属するアカウントは通常、読み取りが許可されています。しかしグループ管理者は通常の権限を強化したものとなっているため、書き込み（データの登録、更新、削除）が行えます。

グループ管理者プリンシパルの選択

グループ管理者はプリンシパルとして用意されます。そのため複数のアカウントをグループ管理者に任命することができます。

図に示すように、プリンシパルは2種類用意されます。

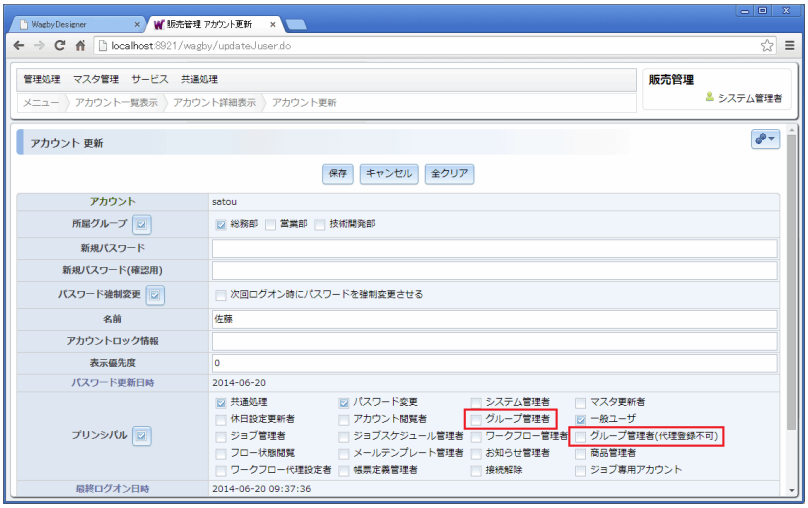


図 二つのグループ管理者プリンシパル

「グループ管理者」は、グループ管理者の権限でデータを操作することに加え、代理登録を行うことができます。 「グループ管理者（代理登録不可）」は、グループ管理者の権限でデータを操作できますが、代理登録を行うことはできません。

代理登録

システム管理者またはグループ管理者は「データ所有者」という項目を変更することにより、データの代理登録を行うことができます。[396]

準備（プリンシパルの設定）

代理登録できるアカウントに、プリンシパル「グループ管理者」と「アカウント閲覧者」を与えます。（なお、システム管理者はこれらを別途、与えなくても常に代理登録が可能です。）

図 グループ管理者とアカウント閲覧者のプリンシパルを与える

代理登録を行う

図で用意したアカウント佐藤で、顧客データの登録画面を開きます。図に示すように「データ所有者」という項目が表示され、かつ変更できるようになっていることがわかります。（この項目はシステム管理者またはグループ管理者以外では不可視です。）

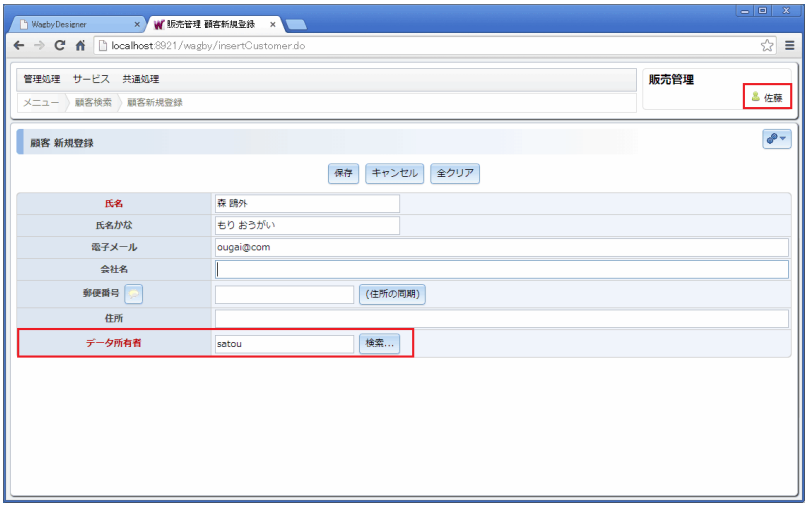


図 データ所有者項目が表示される

別のアカウントに変更することができます。なお、佐藤はグループ管理者のため、所属する同じグループのアカウントに限定されます。

---

[396] グループ管理者の権限が "RW" となっている必要があります。"R" では代理登録はできません。

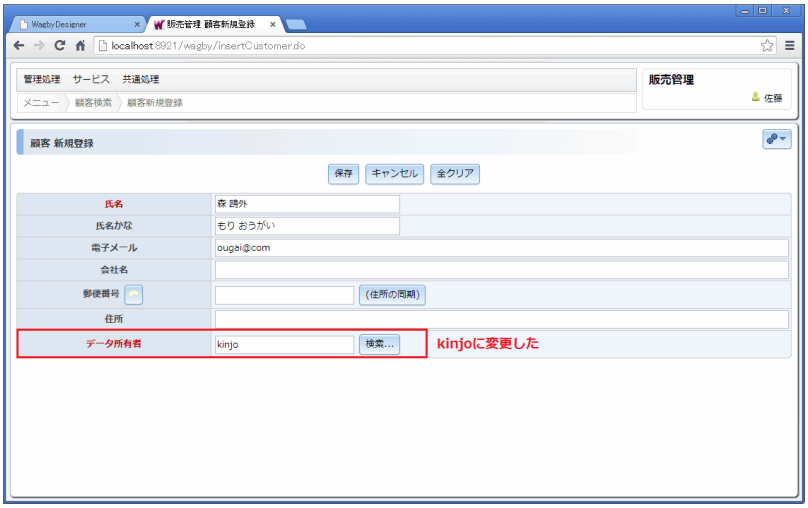


図 データ所有者を変更した

グループ権限設定を解除する

グループ権限管理を解除するには、「権限パターン」の設定をパターン6（初期値；制約なし）にします。

データをグループの持ち物とする

例として、グループ A に所属しているユーザがデータを登録したとします。このモデルの権限パターンが"3" のとき 、登録した本人とグループ A の人々で読み書きが可能です。

このユーザがグループ A から B に異動しました。Wagbyのアカウント情報のグループを A から B に変更します。

グループ B に移動したので、グループ A のデータの閲覧や更新は不可としたいのですが、パターン3ではデータを登録した本人は所属グループに関係なくデータを閲覧、更新可能です。

スクリプトを利用する

この問題を解消し、「データはグループの持ち物」とするためにスクリプトを利用する方法を説明します。 データ登録時や更新時にスクリプトでデータ所有者項目の値を消去します。

```
modelname.useridjshparam = null;
```

(397)

これによってデータ登録者との紐付けがなくなるため、このデータはユーザではなくグループのみで管理されるようになります。

> この値を消去したデータは、グループ管理者で変更できなくなります。（この項目の必須チェックにより、保存が失敗します。）

データ所有者の必須チェックを無効化する（8.1.2）

更新時に上のスクリプトを設定したとき、更新時にデータ所有者が空欄となります。一般の利用者は問題ありませんが、システム管理者やグループ管理者はデータ所有者項目の閲覧権限が適用され、必須チェックによってエラーとなります。

このため、上記スクリプトを設定する場合、あわせて「データ所有者項目の必須チェックを行わない」を有効にしてください。

(397) "modelname" の部分は適切なモデルIDに読み替えてください。

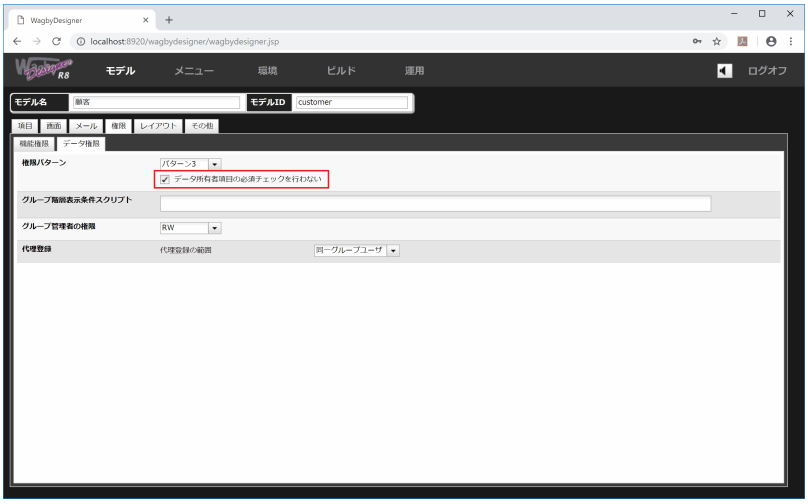


図 データ所有者項目の必須チェックを行わない

グループ階層を利用する

グループ階層を利用すると、上位階層グループの所属ユーザーは、下位階層のグループにも所属していると判定されるようになります。そのため、階層構造の登録前と登録後ではデータの閲覧や編集範囲が広くなります。

ここでは、"アカウント > グループの使い方 > グループ階層" で用意したグループを利用するものとします。設定方法はこちらのリンクにあるマニュアルをお読みください。

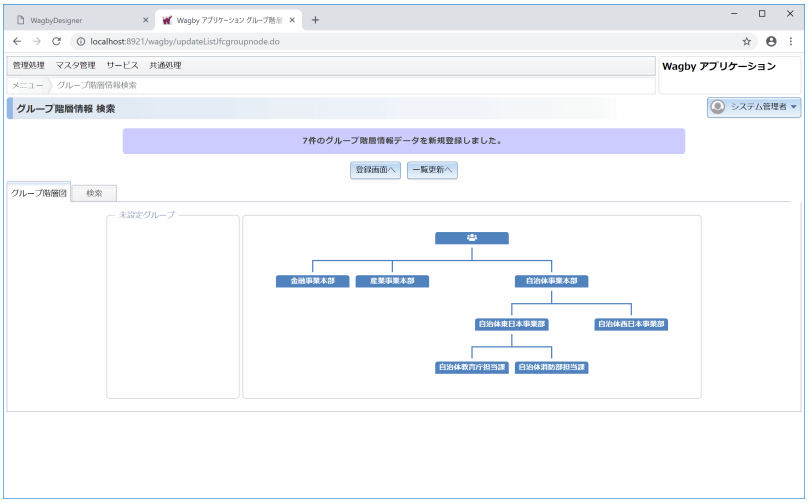


図 用意したグループ階層

階層構造登録を行わない場合、例えば「自治体事業本部」と「自治体東日本事業部」は別々のグループとして扱われます。 「自治体事業本部」グループのみに所属するユーザーは「自治体東日本事業部」グループに所属するユーザーのデータを閲覧/編集することはできません。

そこで「自治体東日本事業部」を「自治体事業本部」の親グループとして登録しました。

すると「自治体事業本部」のみに所属するユーザーであっても（下位グループである）「自治体東日本事業部」に所属するユーザーのデータを閲覧/編集することができるようになります。

更新/削除ボタンの表示制御も所属グループのみだけではなく、下位層グループについても考慮するようになります。そのため、下位層グループのデータについても更新/削除ボタンが表示されるようになります。

テストのためにuser1, user2, user3を用意します。それぞれ第一階層、第二階層、第三階層のグループに所属しているものとします。

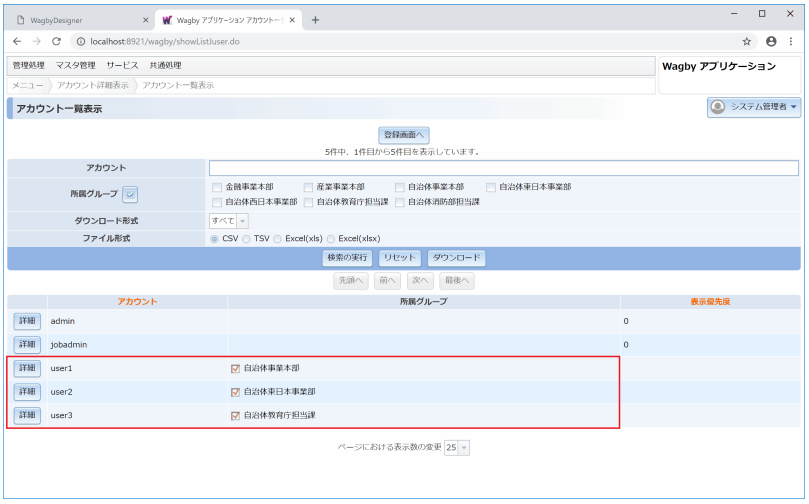


図 ユーザーの登録

テストで用意した「顧客」モデルには、グループ権限3　のパターンを割り当てています。データの登録者および所属グループで読み書きできるというものです。

user1　でログオンし、顧客データを一件、登録します。データ所有者とデータ所属グループは表示されませんが、自動的に設定されます。データ所属グループは自分が所属するグループ（ここでは「自治体事業本部」）が設定されています。

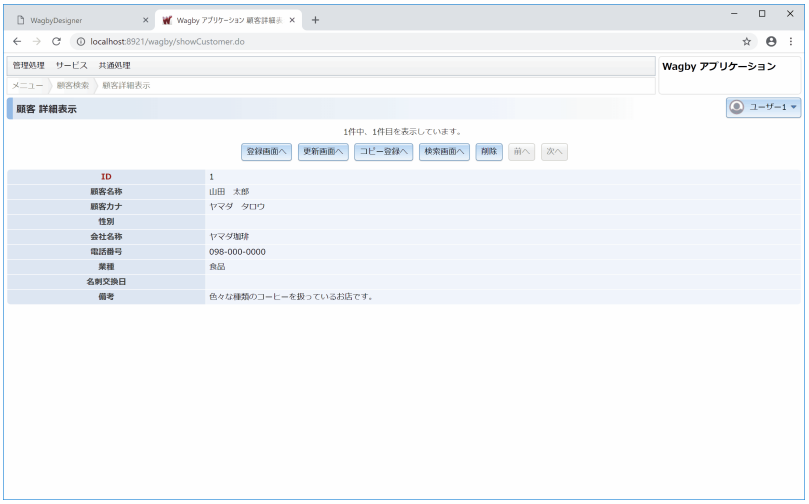


図 データを一件登録する

ダイアログから任意のグループ階層を設定させる（8.1.2）

ここでデータ所有グループを設定できるようにします。階層図から直接、選択できるようにすることができます。

権限 > データ権限　タブに用意された「グループ階層表示条件スクリプト」を設定します。このスクリプトでは、条件を満たしたときに階層図を表示する、というルールを規定します。

ここでは下位階層が存在するユーザーのみ階層図を表示するようにしています。

```
if ($owner.group().children().size() > 0) {
   $owner.group().displayChart();
}
```

スクリプトでは特別な暗黙変数　$owner　が利用できます。この実体は　UserElement　クラスです。詳細は"スクリプトによって承認者・決裁者を動的に決定する > グループ階層"をお読みください。

(398)

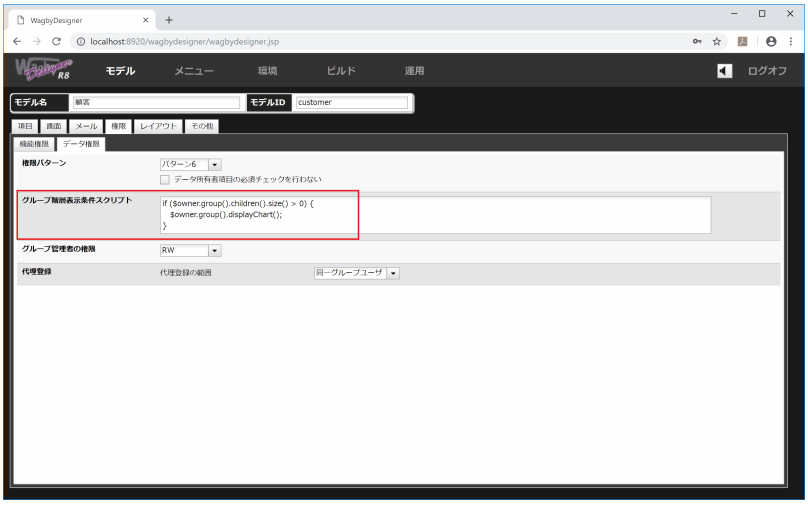


図 スクリプトの設定

(399)

データ所有グループ項目の「検索...」ボタンを押下すると、階層図が表示されます。

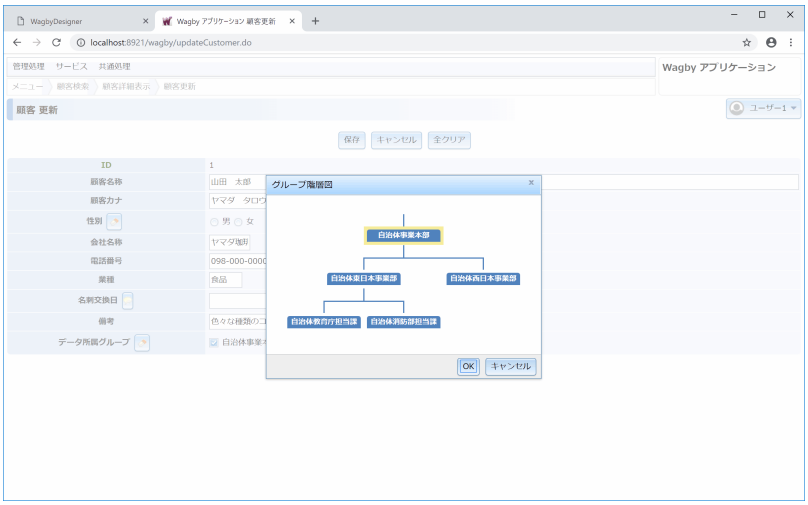


図 階層図の表示

自グループの「下位」のグループにも操作を許諾する場合、これらをマウスで選択します。

(398) 条件を記述せず $owner.group().displayChart(); とすると常に階層図が表示されますが、下位階層グループをもたないユーザーの場合、空のダイアログが表示されるため意味がありません。そのため上のように下位階層がある場合のみ、という条件をつけるとよいでしょう。

(399) 設定したスクリプトは WEB-INF/script/<モデルID>/<モデルID>PHelper_groupSelection.js として保存されます。

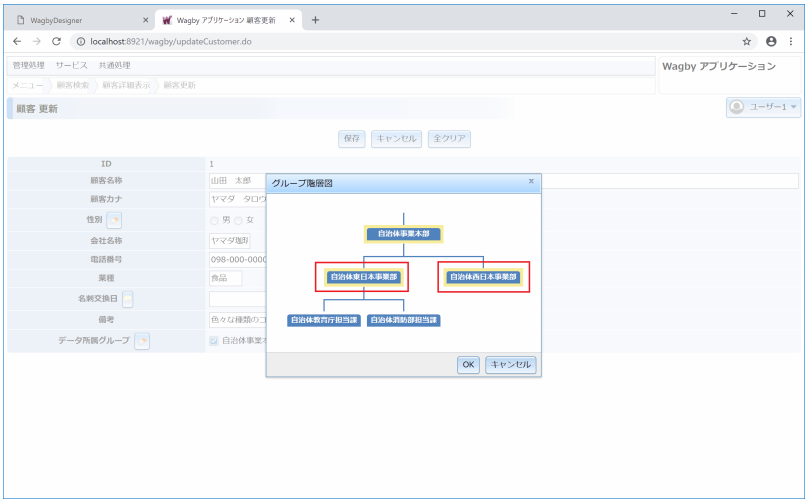


図 下位グループの選択

グループの設定を行いました。チェックボックスだけで操作するよりも、階層構造を理解しながら選択できるようになります。

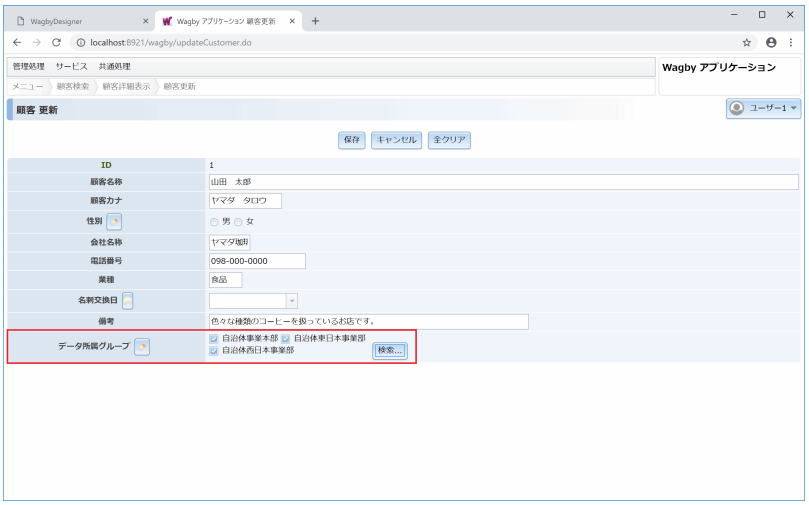


図 下位グループの選択を完了した

権限の動作確認

user2でログオンします。グループ階層で許容されているため、このデータを閲覧することができます。

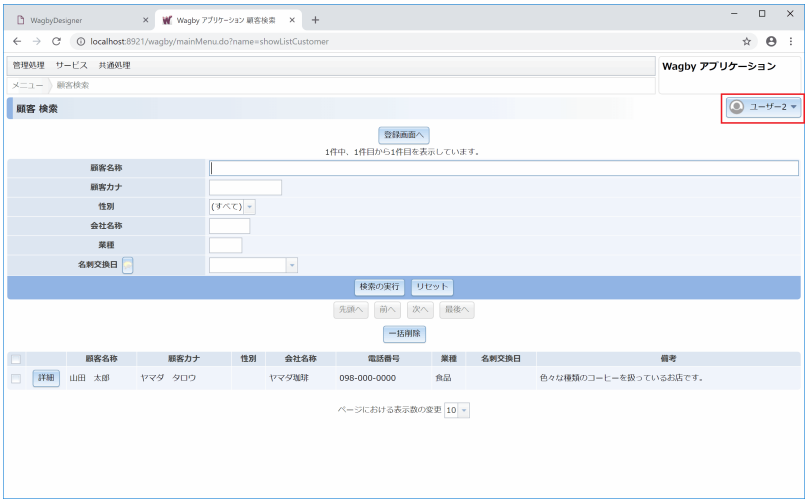


図 user2ではデータを閲覧できる

次にuser3でログオンします。グループ階層で許容されていないため、データを閲覧することができないことがわかります。

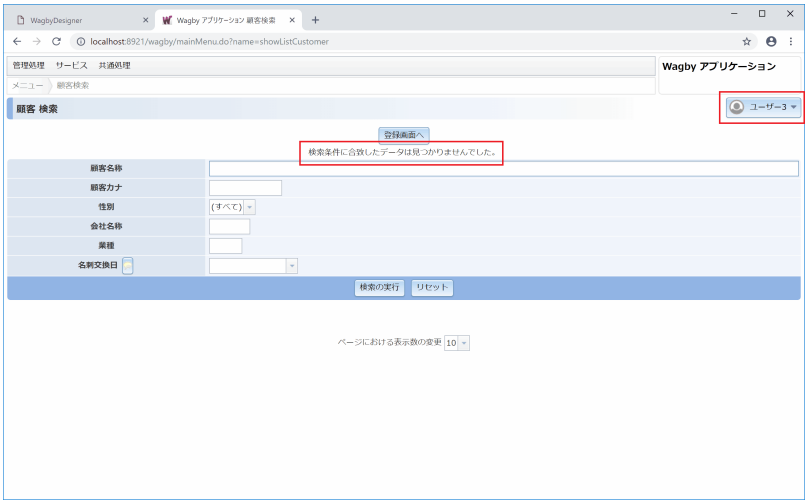


図 user3はデータを閲覧できない

選択させず、常に下位層グループのすべてのユーザーが閲覧可能なデータとする

グループ権限のパターンを2とします。データ登録者は読み書き可能で、グループは閲覧可能というものです。

ここで、データ登録時に、常に自グループおよび下位層グループのすべてのユーザーが閲覧可能なデータとするという要件を想定します。ダイアログを使って選択する必要はない、というものです。

この場合、ヘルパの登録および更新のタイミングに次のスクリプトを記述します。

```
modelname.jgroupidjshparam = $owner.group().descendants().jgroupid();
```

- データ所属グループを表す jgroupidjshparam 項目に、自身の下位層すべてのグループIDを格納しています。

(400)

仕様・制約

- そのモデルの検索・一覧表示画面を用意しなかった場合、グループ権限は利用できません。（設定は無効となります。）
- グループ権限を有効にして更新制限を行っているモデルの一覧表示画面で更新ボタンを用意しないようにしてください。グループ権限利用時、詳細画面に用意される更新ボタンは権限を有するユーザーのみが押せるような制御が働きますが、一覧表示画面には未対応です。
- jgroup モデルの "REST API 機能を有効にする" 設定を有効にしてください。（標準では有効となっています。旧版から移行した場合、手動で設定してください。）

応用例 複数項目の組み合わせパターンに応じたデータの閲覧制御

サンプルの仕様

あるモデルの複数項目の「組み合わせ」によって、アクセスレベルを算定できるものとします。 このとき、当該アクセスレベルをもった利用者のみ、データの閲覧を行わせたいという要件を実現してみます。

(400) "modelname" の部分は適切なモデルIDに読み替えてください。

例を示します。ある工場からの出荷製品について、次のような条件で判定処理を行うものとします。

また、上記 1 ～ 3 に対して、次のようなアクセスレベル表を用意するものとします。

アクセスレベル	条件	説明	処理
レベル4	(1) かつ (2) かつ (3)	全世界へ公開	該当製品を公開する
レベル3	(1) かつ (2)	アジア地域へ公開	
レベル2	(3)	国内へ公開	
レベル1		社内限定	

例

始めにシステム管理者（admin）でログオンし、テストユーザを登録します。 ここではレベル1からレベル4のプリンシパルをもったアカウントをそれぞれ作成しました。

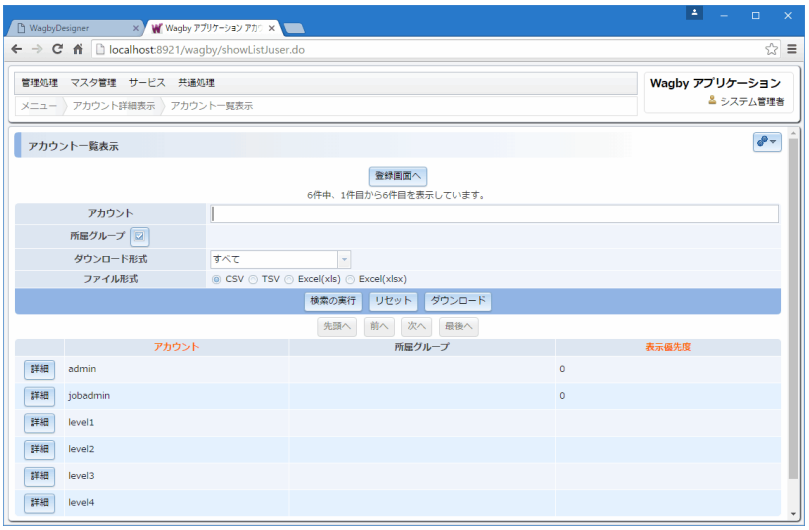


図 テストアカウント一覧

level1アカウントのプリンシパルには「一般ユーザ」と「レベル1」を付与します。同様に level2 アカウントには「レベル2」を、level3 アカウントには「レベル3」を、level4 アカウントには「レベル4」をそれぞれ付与します。

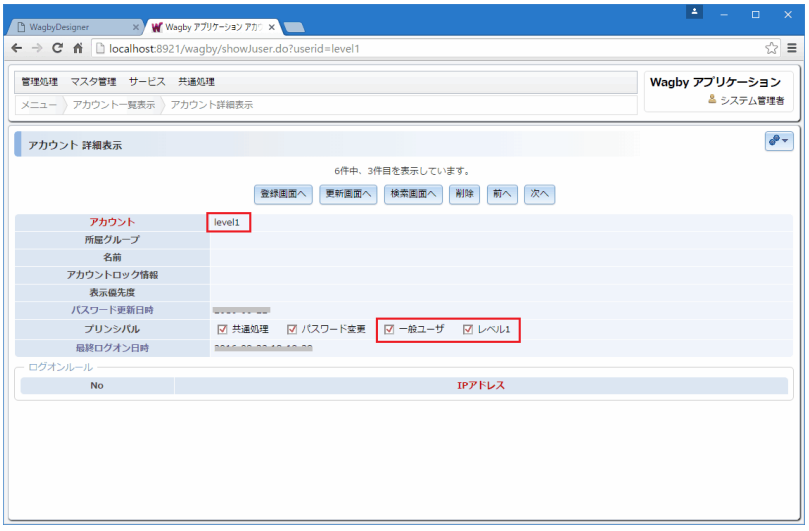


図 level1アカウントの例

続いて製品（テストモデル）を作成します。各項目を入力すると、アクセスレベルの値は自動計算されます。

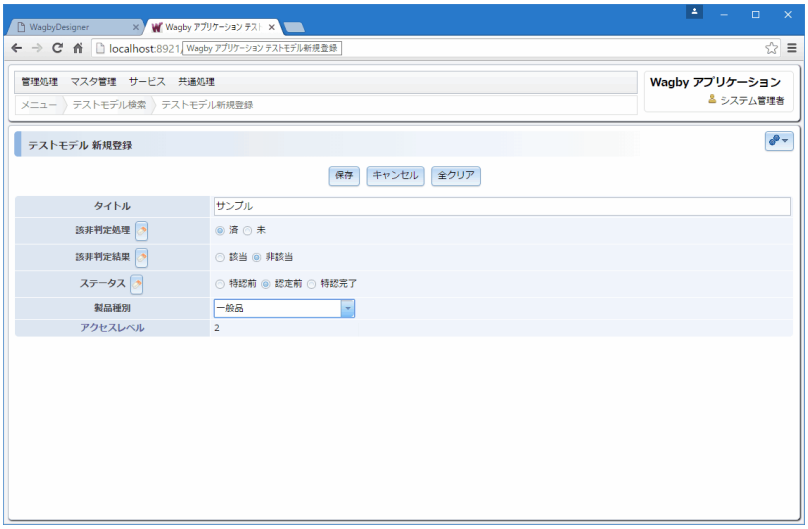


図 テストモデルの登録

アクセスレベル毎のテストモデルを用意します。

図 用意したテストモデル一覧

アクセスレベル1をもったアカウントでログオンします。アクセスレベル1のデータのみが閲覧できます。

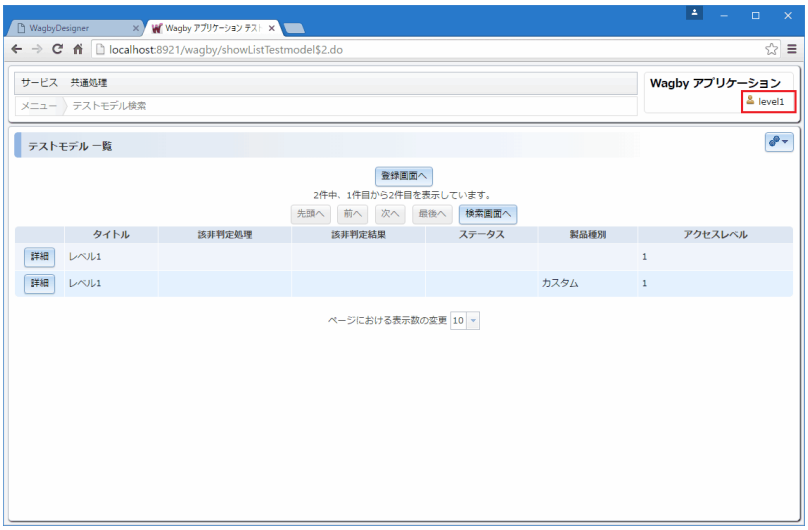


図 アクセスレベル1のデータのみが閲覧できる

アクセスレベル2をもったアカウントでログオンします。 アクセスレベル1およびレベル2データが閲覧できます。

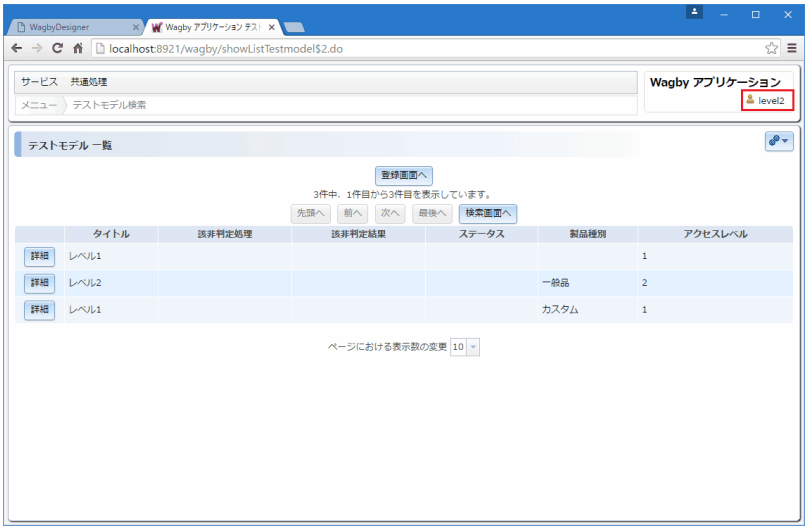


図 アクセスレベル1, 2のデータを閲覧できる

アクセスレベル3をもったアカウントでログオンします。 アクセスレベル1、レベル2、レベル3のデータが閲覧できます。

図 アクセスレベル1, 2, 3のデータを閲覧できる

アクセスレベル4をもったアカウントでログオンします。 すべてのアクセスレベルのデータが閲覧できます。

図 すべてのレベルのデータを閲覧できる

定義方法

サンプルで用意したモデルは次の通りです。

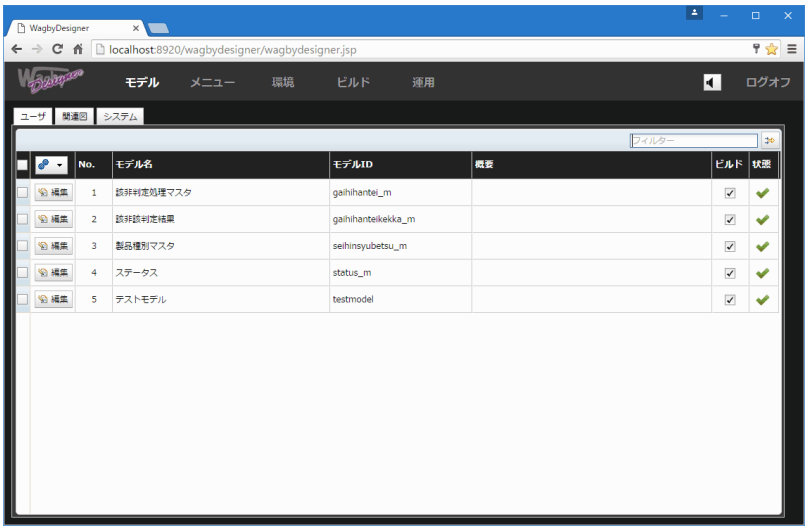


図 モデル定義一覧

本サンプルで利用する項目を示します。

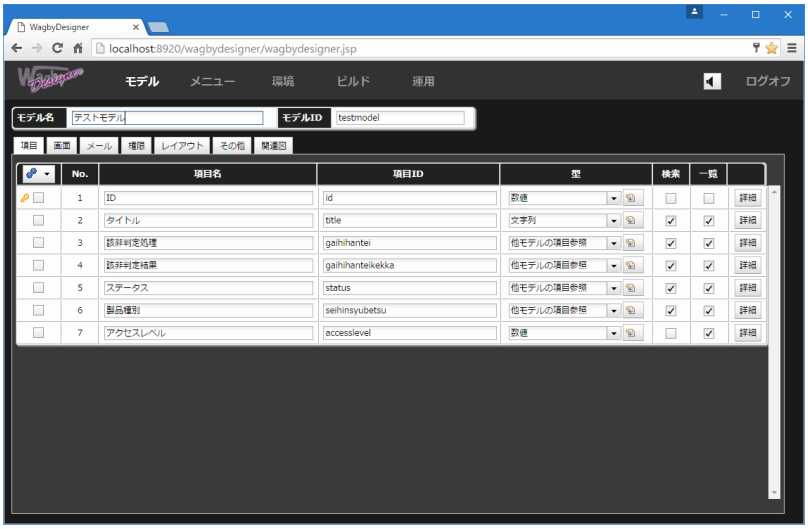


図 利用する項目

「アクセスレベル」を整数型として用意します。次の計算式によって求められるとします。

```
IF(
  AND(
    ${gaihihantei#id}==2,
    ${gaihihanteikekka#id}==2,
    ${status#id}==3,
    ${seihinsyubetsu#id}==1
  ),
  4,
  IF(
    AND(
      ${gaihihantei#id}==2,
      ${gaihihanteikekka#id}==2,
      ${status#id}==3
    ),
    3,
    IF(
      ${seihinsyubetsu#id}==1,
      2,
      1
    )
  )
)
```

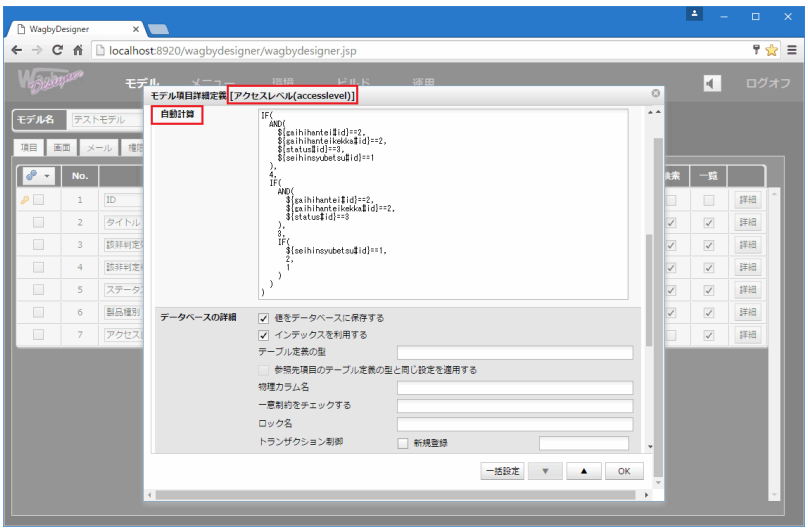


図 計算式の設定

(401)

さらにアクセスレベル項目について、複数の暗黙条件を設定します。

暗黙条件の設定では「指定した権限を満たした場合、この暗黙条件を適用しない（無視する）」というオプションが用意されています。この機能を活用します。

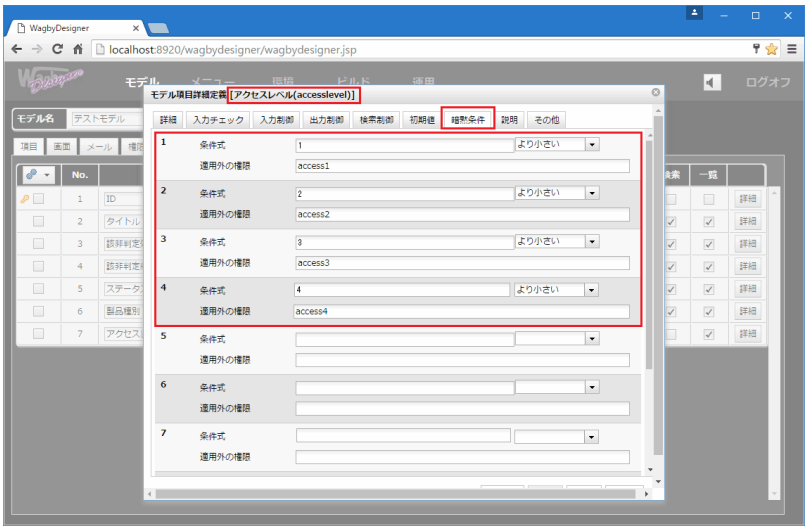


図 アクセスレベルへの暗黙条件の設定

複数の暗黙条件が指定された場合

複数の暗黙条件が指定された場合、これらの暗黙条件は「かつ」条件式として扱われます。 ここで示した例の場合、下記の条件が適用されることになります。

「アクセスレベル」項目は「1より小さい」かつ「2より小さい」かつ「3より小さい」かつ「4より小さい」（「適用外の権限」が考慮されていないとき）

これに「適用外の権限」を考慮すると次のようになります。

access1 権限を持つユーザの場合

暗黙条件は以下のように動作します。

「アクセスレベル」項目は「2より小さい」かつ「3より小さい」かつ「4より小さい」

ただし「2より小さい」という条件は「3より小さい」と「4より小さい」を包含しているので、結果的には「アクセスレベル」項目は「2より小さい」という条件が適用されることになります。

access1, access2 権限を持つユーザの場合

暗黙条件は以下のように動作します。

「アクセスレベル」項目は「3より小さい」かつ「4より小さい」

ただし「3より小さい」という条件は「4より小さい」を包含しているので、結果的には「アクセスレベル」項目は「3より小さい」という条件が適用されることになります。

権限の設定

権限を用意します。利用者毎に割り当てる「プリンシパル」と、各プリンシパルに対応した「パーミッション」を用意します。

(401) 自動計算式で「#id」を付与するとモデルのID値を取得することができます。

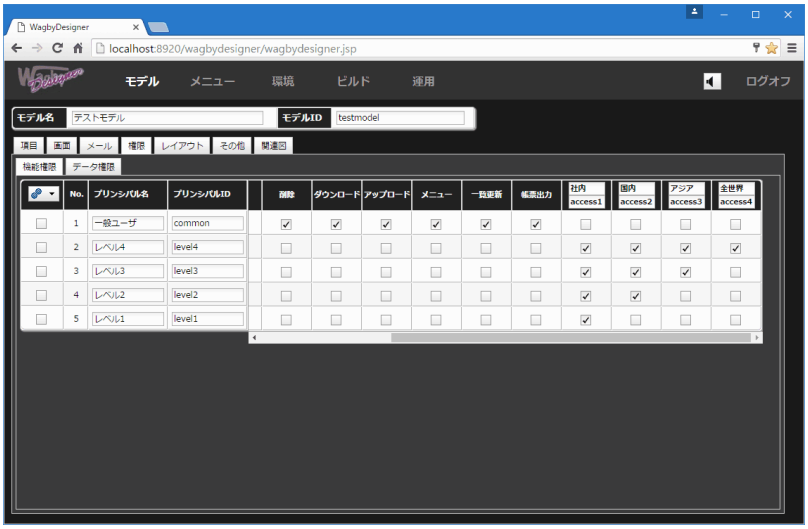


図 プリンシパル、パーミッションの設計

応用例 データベースのビューにグループ権限を適用する

Wagbyが利用するカラムの追加

データベースのビューに次のカラムを用意してください。

- useridjshparamカラム。
- (ビュー名)$jgroupidjshparam テーブル。

useridjshparam カラムとは、用意するビューの SELECT 句へ「useridjshparam」を追加し、データ所有者アカウントを返すようにします。(ビュー名)$jgroupidjshparam テーブルにはデータの所属グループのグループIDを格納してください。グループID は jgroup テーブルの主キー jgroupid を用います。

(402) (403)

グループ権限は WHERE 句に絞り込み条件を指定することで、閲覧/更新の制御をおこなっています。WHERE 句の絞り込み部分は Wagby 側で自動生成していますので、useridjshparamカラムと、xxxx$jgroupidjshparam を用意することで動作するようになります。

アカウント

juserモデルの基本

用意されている項目

項目名	項目ID	説明	サブモデル
アカウント	userid	主キー項目です。juserモデルは他の多くのモデルから参照されます。	○
所属グループ	jgroupid	所属グループ項目でチェックをつけたグループの主キーが	○

(402) 別のモデルで xxxx$jgroupidjshparam テーブルを利用するようにし、これを参考にするとよいでしょう。

(403) (ビュー名)$jgroupidjshparam テーブルを、実際にはテーブルではなく、これもビューとして用意することもできます。

		格納されます。所属グループはグループ権限やワークフローで利用されます。チェックボックス方式のため、実際の値は物理テーブル juser$jgroupid に格納されます。	
現在のパスワード	passwdNow	この値を閲覧、操作することはできません。	×
新規パスワード	passwd	この値を閲覧、操作することはできません。	×
新規パスワード(確認用)	passwd2	この値を閲覧、操作することはできません。	×
パスワード強制変更	compulsoryChange	このフラグを有効にすると、次回の最初のログオンで新規パスワードの入力処理が強制されるようになります。	×
名前	name	ログオンが成功すると、画面右上部に表示されます。	○
アカウントロック情報	valid	通常は空です。ログオン時パスワードの入力ミスでロックされている場合は、Wagby が自動でロック時の日時をセットしています。ここに値がセットされると、このアカウントはロックされ、ログオンできなくなります。	○
表示優先度	level	juserを参照するモデルで指定することができます。	○
パスワード更新日時	passwdChangeDate	最後にパスワードを更新した日時が記録されます。[参考:パスワード有効日数]	×
パスワード更新フラグ	passwdChangeFlag	compulsoryChange項目と同じ目的で用いています。システム内部で管理している値のため、通常は変更しないようにしてください。	×
パスワード入力エラー数	passwdErrrCount	パスワード入力を間違えた回数が記録されます。ログオンが成功すると、この値はクリアされます。[参考:アカウントロック]	×
過去のパスワード	oldPasswds	過去に利用したパスワードの履歴を管理するテーブル juser	×

		$old_passwds が用意されます。運用中にパスワードを変更するごとにレコードが追加されます。	
プリンシパル	jprincipalId	プリンシパル項目でチェックをつけたプリンシパルの主キー(数値)を格納するテーブル juser $jprincipal_id が用意されます。	○
最終ログオン日時	lastLogonDate	最後にログオンした日時が記録されます。	○
アバター画像	avatarImage	アバター画像ファイルが格納されます。	○
ログオンルール	rule	IPアドレスによるログオン制限を行う場合に利用します。juser$rule。	○
プリファレンス	(項目IDは不可視)	"メニュー > 共通処理 > プレファレンス" で設定したものが格納されます。内部では juser $preference_item テーブルが用意されています。	-

(404)

これらの定義済み項目の変更・削除を行うことはできません。

項目を追加する

実行例

juserモデルに「メールアドレス」項目を追加した例を図に示します。

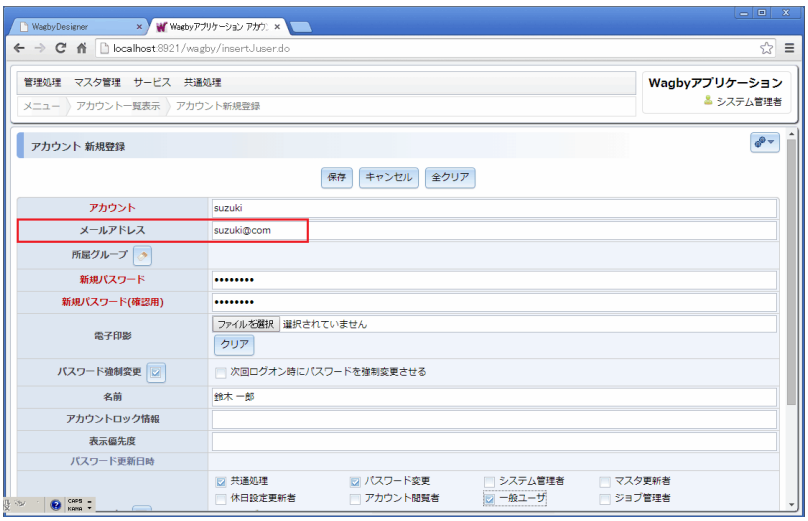


図 メールアドレス項目が追加されたjuserモデル

(404) プリファレンスは juser に紐づいているテーブルですがシステム内部だけで管理されています。Wagby Designer で扱うことはできません。

詳細画面では、メールアドレス部分はリンクになっています。

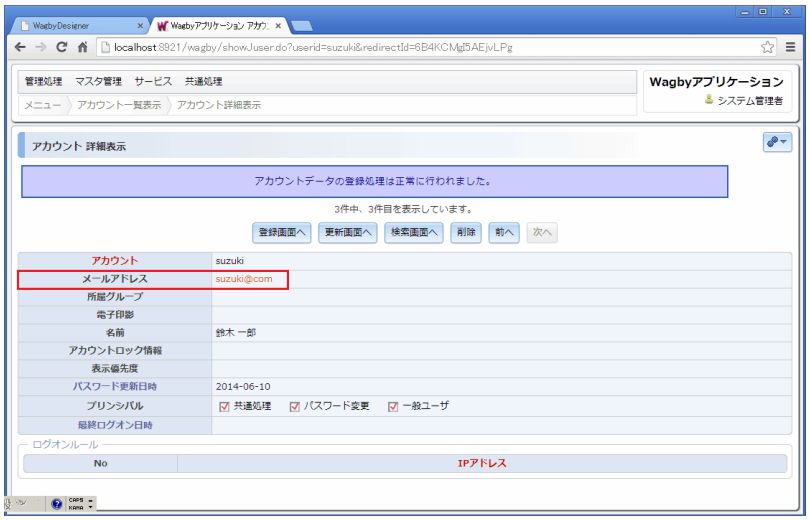


図 juserモデルの詳細画面

定義方法

「アカウント」項目の直下に「メールアドレス」項目を用意する例を示します。挿入したい位置をチェックし、ギアアイコンから新規を選びます。

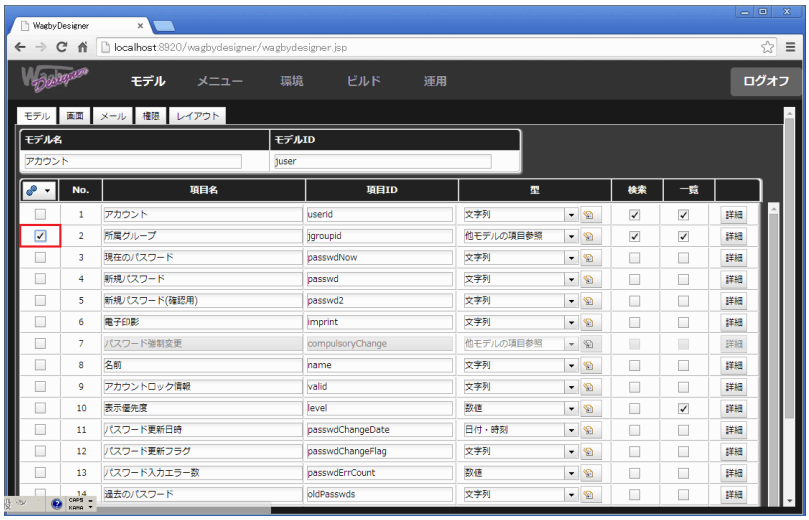


図 挿入したい場所をチェックする

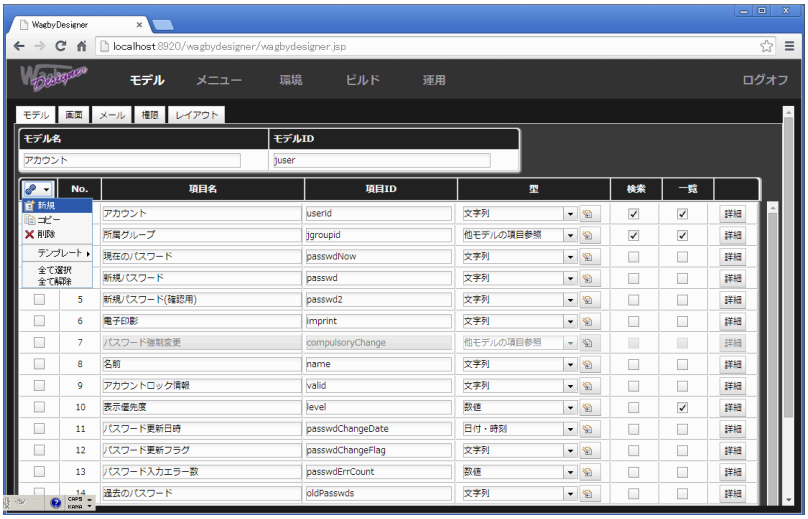


図 ギアアイコンから「新規」を選択する

入力欄が用意されます。ここでチェックを手動で解除します。[405]

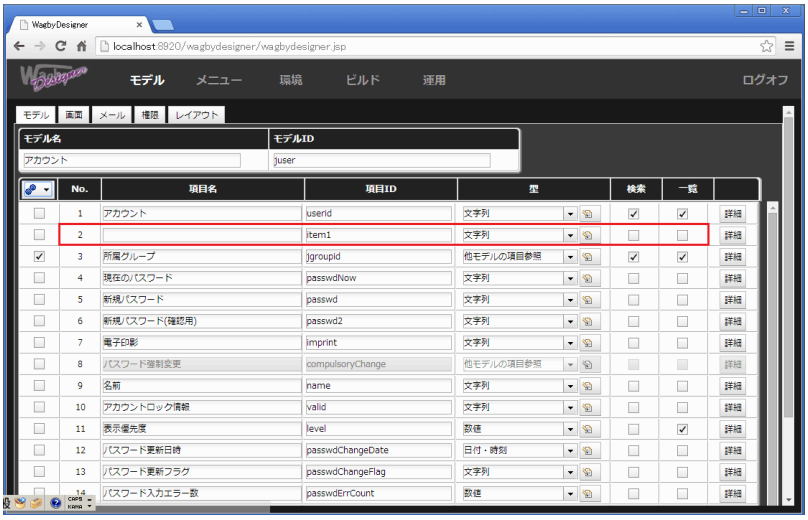


図 新規項目の入力欄が用意される

「メールアドレス」項目を用意しました。項目IDは「mailaddress」とします。検索と一覧もそれぞれ有効にします。

(405) 標準では、チェック状態は維持されます。これは連続した新規操作を実現するためです。

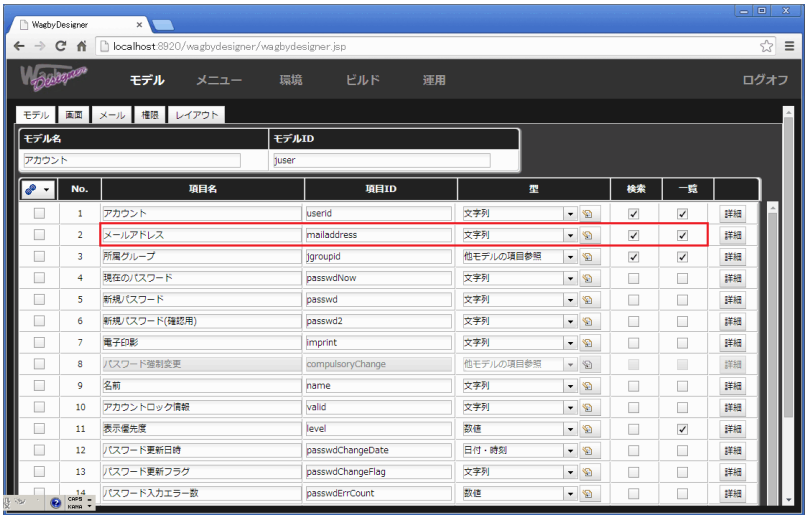


図 メールアドレス項目を用意する

「メールアドレス」項目は文字列型ですが、メールアドレス属性を付与しておきます。

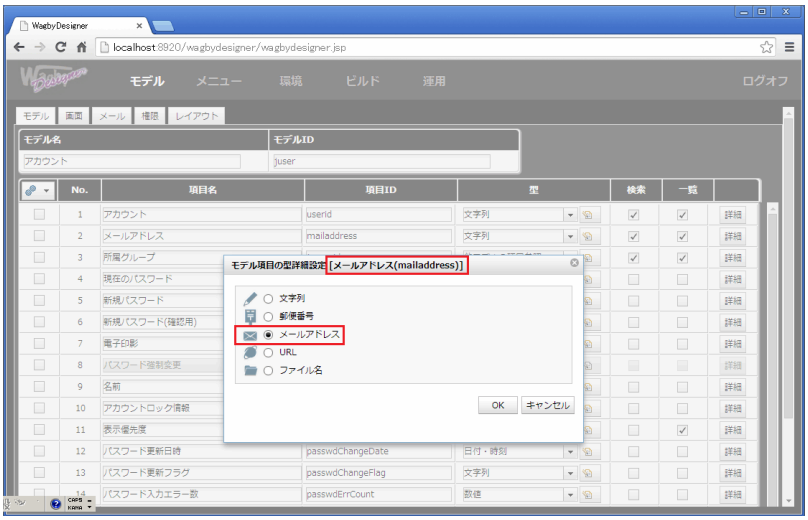


図 メールアドレスに属性を付与する

juserモデルを変更するとテーブル構造が変わります。ビルドの前にWagbyアプリケーションを起動していた場合は、停止させてからビルドを行ってください。

アバター画像

アカウントごとにアバター画像を設定することができます。 アカウントの新規登録（または更新）時に、アバター画像ファイルを登録します。ファイル形式は png としてください。

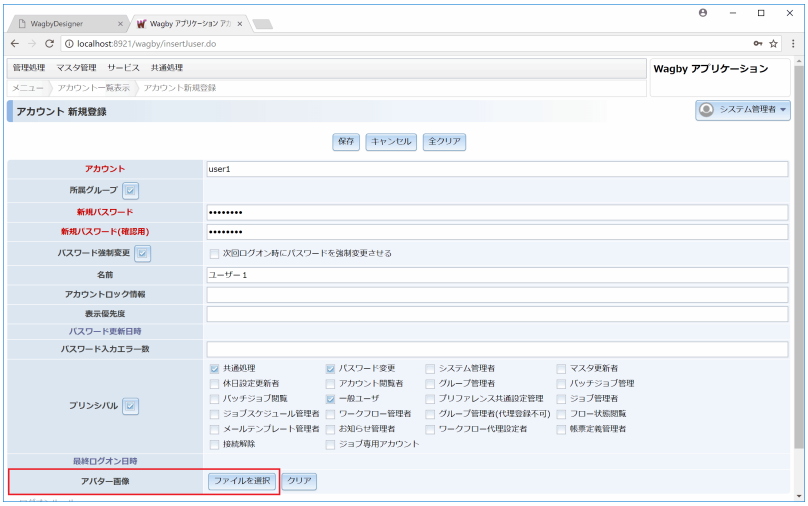


図 アバター画像の登録(1)

登録すると、自動的にサイズが 25x25 ピクセルに縮小されます。

図 アバター画像の登録(2)

ログオンすると、アバター画像が表示されます。

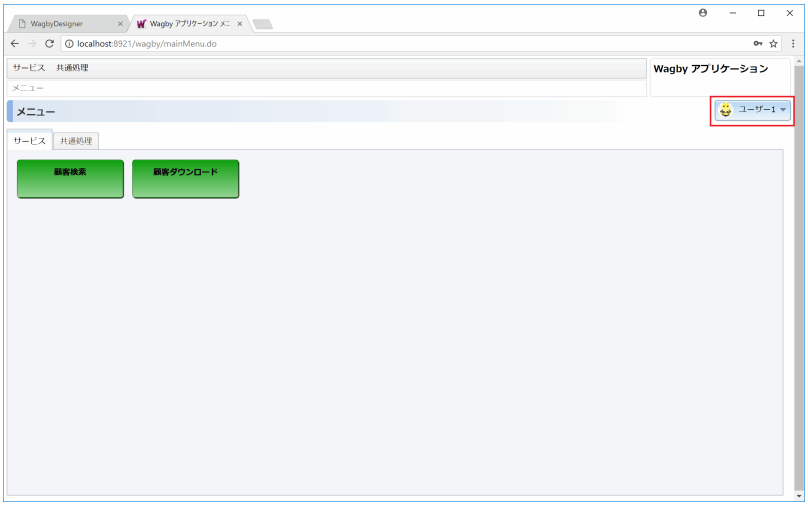


図 アバター画像の表示

仕様・制約

- juserモデルは性質上、カスタマイズできる範囲は制限されています。特に認証・認可に関わる部分は修正することができません。それでも、いくつかの項目を拡張することはできるようになっています。
- juserモデルは他の多くのモデルから参照されます。例えばワークフローやジョブ機能ではjuserのuserid項目を格納する項目があります。運用開始後にjuserモデルの変更を行う場合は、既存データの変換を行う必要が生じる可能性がありますので、ご注意ください。

外部システムから直接、juser テーブルに値を設定する場合

Windows Active Directory / LDAP アカウントと同期するなどの理由で、外部システムから直接、juserテーブルに(SQLなどを使って)値を設定する場合は次の点にご注意ください。

- アカウント (userid) 項目は主キーであるため必須です。
- 所属グループを指定する場合は、juser$jgroupid テーブルへ値を登録してください。グループの主キー(数値)を格納します。
- プリンシパルを指定する場合は、juser$jprincipal_id テーブルへ値を登録してください。プリンシパルの主キー(数値)を格納します。

パスワード

定義方法

「環境」メニューから「アプリケーション」を選択します。 ここで用意された「アカウントパスワード」を設定します。本設定はプロジェクト全体に渡って適用されます。

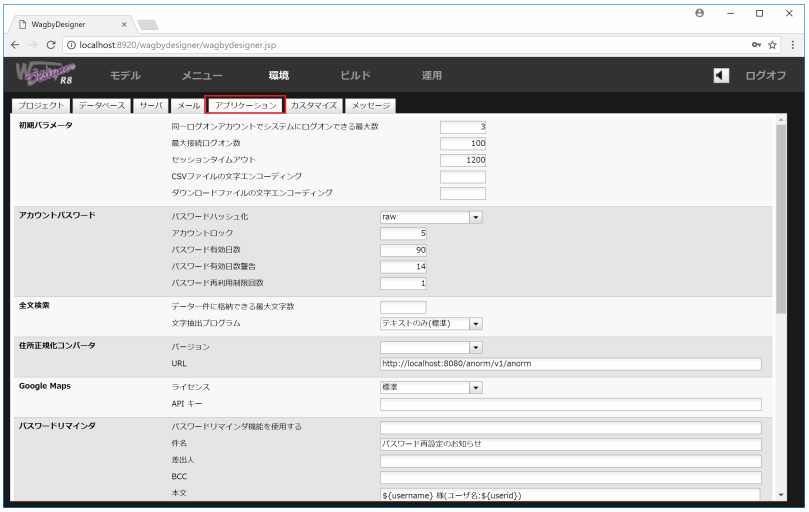


図 アプリケーションの設定

パスワードハッシュ化

DBにパスワードを格納する際にパスワードのハッシュ化を行います。 ハッシュ化されたパスワードは管理者であっても解析することはできません。 この値の標準はraw（パスワードはハッシュ化されず平文で扱う）となっています。

種類	説明
raw	パスワードはハッシュ化されず平文で扱われます。
md5	ハッシュアルゴリズムにMD5を利用します。
sha-1	ハッシュアルゴリズムにSHA-1を利用します。
sha-256	ハッシュアルゴリズムにSHA-256を利用します。

sha-384	ハッシュアルゴリズムにSHA-384を利用します。
sha-512	ハッシュアルゴリズムにSHA-512を利用します。
BCrypt	ハッシュアルゴリズムにBCryptを利用します。 (後述)8.3.0

パスワードハッシュ化設定を指定しない、もしくは上記以外の値を設定した場合にはパスワードはハッシュ化されず平文で扱われます。

Bcryptを利用する (8.3.0)

Bcryptを利用する場合は、同時にBCryptストレッチング回数を指定します。値は 4 から 31 の範囲で、省略時は 10 となります。

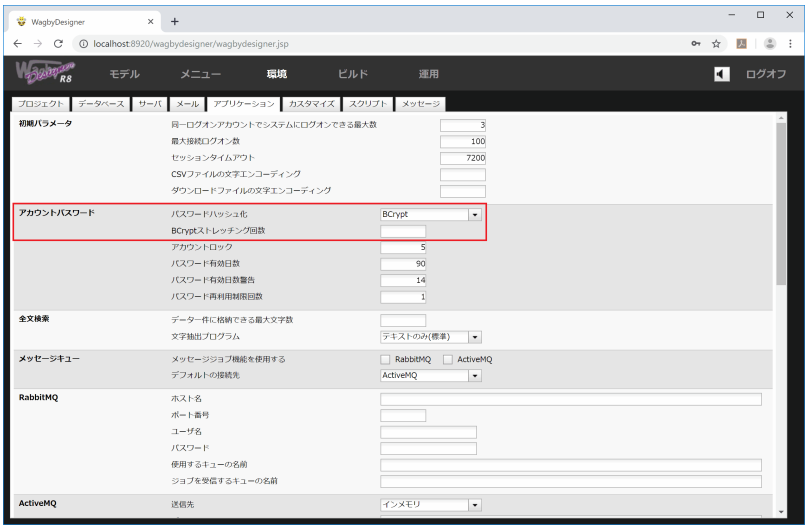


図 Bcryptの設定

運用途中で、パスワードハッシュ化アルゴリズムを変更することはできません。この設定を変更してビルドすると、すでに登録されているアカウントのパスワードは変更前の方式で保存されているため、ログオンできなくなります。

アカウントロック

ログオン画面にてパスワードを連続一定回数以上まちがえるとアカウントがロックされるようになります。このパラメータを指定しない、もしくは1より小さい値を設定した場合には本機能は無効となり、パスワードを連続でまちがえてもアカウントはロックされません。この値の標準は5回となっています。

アカウントロックは「ログオン画面」と「パスワード変更画面」に適用されます。

この設定を有効にした場合でもシステム管理者権限を持つユーザはアカウントロックの対象とはなりません。

アカウントロック機能は「brute force攻撃(パスワード総当り攻撃)」への対策として用意されています。この機能を無効にすると、本攻撃に対するセキュリティが脆弱性な状態となってしまいますのでご注意下さい。

ロック動作の詳細

パスワード誤りが連続して行われているとき、内部ではアカウントロック値を増加させています。しかしエラーメッセージは常に ″ユーザ名またはパスワードが間違っています。パスワードを連続して複数回間違えるとアカウントがロックされる場合があります。ご注意ください。″ となり、いつロックされたかどうかはわからないようになっています。

この理由は、一定回数のパスワード間違いで即座にアカウントロックし、エラーメッセージを変えてしまうと、アカウントの存在そのものが攻撃者に認識されてしまうためです。そのためエラーメッセージは意図的に変えないようにしています。

パスワードを連続して複数回間違え、ログオンできない場合は次のいずれかの方法でロックを解除してください。

- パスワードリマインダー機能を使って、自身でパスワードを変更する。
- システム管理者に依頼して、当該アカウントのパスワードを変更する。

いずれの場合も、このタイミングで、内部で保持しているパスワードエラー回数がリセットされます。

パスワード入力エラー数

パスワード入力を間違えた回数は、juser　モデルの「パスワード入力エラー数」項目内で管理されます。ログオンが成功すると、この値はクリアされます。

パスワード有効日数

ユーザのパスワードに有効期限が設定されます。

パスワードの有効期限が切れたユーザはパスワードを変更するまで他の操作を行うことができなくなります。このパラメータを指定しない、もしくは0より小さい値を設定した場合には本機能は無効となり、パスワードに有効期限は存在しなくなります。この値の標準は90日となっています。

有効日数の扱いは次のようになります。

有効日数	説明
0	パスワードは変更日の24:00 まで有効です(翌日の 0:00 には無効)。
1	パスワードは変更日の1日後の24:00まで有効です。
2	パスワードは変更日の2日後の24:00まで有効です。
...	
90	パスワードは変更日の90日後の24:00まで有効です。
N	パスワードは変更日のN日後の24:00まで有効です。
未指定(または"-1"など、0より小さい値)	パスワード有効期限なし。

パスワード有効日数を90日とした場合、次のような動作となります。

変更日 ... パスワードを変更した日 01日後 ... 89日後 90日後 ... この日の24:00に有効期限が切れます。

ジョブ実行専用アカウント"jobadmin" は、この設定が無効化されます。

パスワード有効日数警告

ユーザのパスワード有効期限が近づくとメニュー画面にその旨を伝えるメッセージが表示され、パスワード変更を促すことができます。この値の標準は14日となっています。

有効日数警告に「N」を指定した場合、期限切れの N 日前に警告が表示されます。 有効日数警告に「1」を指定した場合、パスワード有効日数が切れる1日前(パスワード有効日数が90日の場合、パスワード変更日の90日後の0:00)から警告が表示されるようになります。

以下の場合には本機能は無効になります。

- このパラメータを指定しない
- 0以下の値を指定した場合（負数、および0を含む指定）
- パスワード有効日数の設定値以上の値を指定した場合

パスワード再利用制限回数

ユーザは過去に設定したパスワードを再利用できなくなります。この値の標準は1となっています。 定義方法は次のとおりです。

数値	説明
1以上	現在のパスワードと指定数値の「過去回数のパスワード -1」が利用できない。「1」の場合は、現在のパスワードのみチェックする。
0 または 未設定 または 0以下	チェックしない。

パスワードの一括更新

CSVまたはExcelファイルをアップロード更新することでアカウントとパスワードの一括登録・更新を行うことができます。 詳細は「アカウント>CSVダウンロードとアップロード更新」で説明します。

利用できる文字を制限する

パスワードの文字数の設定

モデル一覧から「システムタブ > アカウント モデル」を選択し、「新規パスワード」項目の詳細定義を開きます。

「文字数チェック」設定欄にてパスワードに入力可能な文字数が指定できます。初期設定値は「文字数 : 最小5文字、最大文字数制限なし」です。

利用できる文字の種類を制限する

「新規パスワード」項目の詳細定義から「許容文字種」に利用できる文字の種類が設定されています。

初期設定値は「許容文字種 : 数字(半角)、アルファベット大文字小文字(半角)、記号(半角)」です。

記号(半角)として利用できる文字は「入力チェック > 許容文字種 > 設定の詳細」をご確認ください。

> 上記の設定を変更される場合は「現在のパスワード」項目及び、「新規パスワード」項目の両方に同じ設定を行ってください。

例: パスワードに1文字以上の英字・数字が混在していること

上で説明した「許容文字種」チェックにて必要最小文字数を指定することができます。 これを用いて、数字とアルファベットを1文字以上とすることで、数字のみ、英字のみのパスワードを許容しないようになります。

ユーザ定義チェックの利用は非推奨

パスワード文字列の入力ルールはユーザ定義チェックではなく、許容文字種の設定を使うことを推奨します。

パスワード項目に「ユーザ定義チェック」を適用する場合、パスワードハッシュ化設定は raw に限定されます。raw 以外を選択した場合、ハッシュ化されたパスワードとのチェックとなるため、意図しない結果となるためです。

パスワードを忘れたとき

システム管理者に依頼する

システム管理者(admin)は、一般利用者の新しいパスワードを再設定することができます。

パスワードリマインダを使う

「パスワードリマインダ」を使って、自分のパスワードを再設定することができます。（この機能を有効にする必要があります。）

juserテーブルの値を直接、編集する

パスワードは juser テーブルの passwd 項目に格納されています。SQLを使って、この値を直接、書き換えることもできます。

(406)

パスワードハッシュ化を適用していた場合、passwd 項目にはハッシュ化された文字列を格納してください。ハッシュ化文字列の作成方法は"Designerの機能・その他のツール ＞ パスワードのハッシュ値を取得する"をお読みください。

ビルド直後の状態に初期化する

init_db.bat スクリプトを使って初期データに戻すことができます。その後、直前にエキスポートしたデータをインポートしてください。

運用中のアプリケーションの挙動を直接、変更する

ここで設定した値は、ビルド後に次のファイルに反映されます。 wagbyapp/webapps/${プロジェクト識別子}/WEB-INF/classes/application.properties

具体的には以下のエントリになります。

R8.4以降

```
server.servlet.context-parameters.userinfo_failedpassword_count=5
server.servlet.context-parameters.userinfo_oldpasswd_save_size=1
server.servlet.context-parameters.userinfo_passwd_change_warn_period=14
server.servlet.context-parameters.userinfo_passwd_digest_algorithm=raw
server.servlet.context-parameters.userinfo_passwd_expire_period=90
```

R8.3.xまで

```
server.context-parameters.userinfo_failedpassword_count=5
server.context-parameters.userinfo_oldpasswd_save_size=1
server.context-parameters.userinfo_passwd_change_warn_period=14
server.context-parameters.userinfo_passwd_digest_algorithm=raw
server.context-parameters.userinfo_passwd_expire_period=90
```

このファイルを直接編集し、Wagbyアプリケーション(wagbyapp)を再起動することで、ビルドすることなく変更した値を適用することができます。

運用上の注意点

- この値を直接編集した場合、該当する(Designerの)設定も同時に行うようにしてください。次回ビルド時に反映されるようになります。
- パスワードハッシュ化アルゴリズム(server.context-parameters.userinfo_passwd_digest_algorithm)は、運用途中で値を変更することはできません。

CSVダウンロードとアップロード更新

ダウンロード

juserモデルは標準で「ダウンロード」機能が有効になっています。

(406) Windows Active Directory / LDAP 利用時を除きます。

図 標準で用意されるダウンロード機能

出力されるファイル仕様は次のとおりです。

項目名	内容
ID	ユーザIDが出力されます。
所属グループ	登録されているグループ名が出力され、アカウントが所属しているグループには「1」、所属しない場合は「0」と出力されます。
アカウントロック情報	アカウントがロックされている場合、文字列が含まれます。ログオン時パスワードの入力ミスでロックされている場合は、Wagby が自動でロック時の日時をセットしています。
表示優先度	Wagbyが内部で用意するアカウント（juser）モデルの表示優先度です。
パスワード更新日	パスワードが更新された日付が出力されます。
プリンシパル（jprincipalId）	定義したアプリケーションが用意するプリンシパル名が出力されます。アカウントが持つプリンシパルには「1」、持たない場合には「0」が出力されます。

パスワードの扱い

パスワード情報は含まれません。

アップロード更新

juserモデルのアップロード更新機能は標準では無効になっています。　本機能を有効にすることで、アップロード更新を行うことができます。

アップロード更新機能を有効にした場合は、メニューへ（アップロード更新の）アイコンを追加するようにしてください。

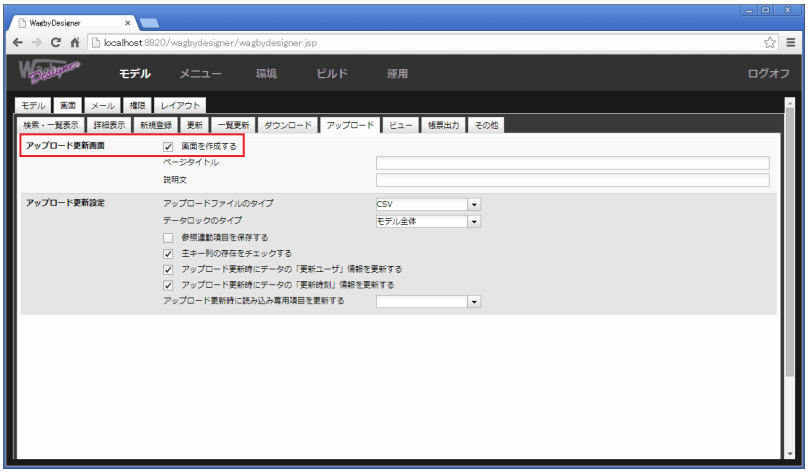


図 アップロード更新機能を有効にする

パスワード項目の扱い

ダウンロード指定に「新規パスワード」列を加えることで、(CSV・Excelファイルを用いて）パスワードの一括更新を行うことができるようになります。

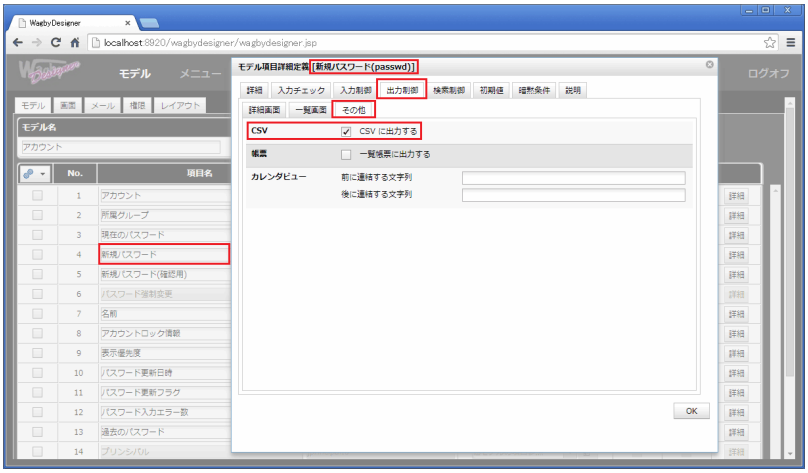


図 新規パスワード列をダウンロードできるようにする

この場合の動作は次のようになります。

「新規パスワード」列をダウンロードに加えない(標準)		「新規パスワード」列をダウンロードに加える	
ダウンロード	アップロード更新	ダウンロード	アップロード更新
「新規パスワード」列は含まれません。	アップロードするファイルに「新規パスワード」列を手動で追加しても無視されます。(407)	ダウンロードファイルに「新規パスワード」列は存在しますが、内容は空欄で出力されます。（パスワードの内容を知ることはできません。）	アップロードファイルの「新規パスワード」列に値を指定すると（その値で）更新されます。空欄のままだと既存の値を維持します。（入力チェック処理も行われません。）(408)

(407) 新規アカウントを追加できますが、管理者が手動で同アカウントのパスワードを設定しない限り、当該アカウントでログオンすることはできません。パスワードの初期値（登録時）を設定することで、初期パスワードを仮登録する運用を行うことができます。

(408) 「新規パスワード（確認用）」はダウンロードに加える必要はありません。

ハッシュ化パスワードの扱い

パスワードハッシュ化の設定を行なっていた場合も、アップロードするパスワードはハッシュ化していない文字列を設定してください。

パスワードに関するその他の項目

以下の項目についても、アップロード更新で値を設定することができます。

- パスワード更新フラグ（passwd_change_flag）
- パスワード入力エラー数（passwd_err_count）
- パスワード強制変更フラグ（compulsoryChange）

よくある質問と回答

「次回更新時にパスワード変更させる」という設定を加えたい

アカウントモデルの「パスワード強制変更(compulsoryChange)」項目は、CSV　出力項目となっていません。この項目はDesignerでは変更できないため、次の手順で CSV 出力項目とします。

1. repository/trunk/common-juser/juser/compulsoryChange/compulsoryChange.txt をテキストエディタで開きます。(Windows OS 付属の「メモ帳」は使えません。 TeraPad などのテキストエディタをご利用ください。)
2. 同ファイルにある、次の行を変更します。 変更前:

```
model/modelitem/@csvOutput=
```

変更後:

```
model/modelitem/@csvOutput=true
```

3. 編集したファイルを保存します。

変更後、テスト用のアカウントを作成し、パスワード強制変更を有効にします。これをダウンロードして、本欄が有効のデータと、無効のデータを目視で比較します。有効のデータの記述を真似て、アップロード更新を行ってください。

(409)

アップロード更新はダウンロードの形式に合わせた形でアップロードする必要があります。上記設定の有効・無効に合わせてアップロードデータの形式も異なるのでご注意ください。

パスワードリマインダ

利用イメージ

この機能を有効にすると、ログオン画面に「※ パスワードを忘れた方はこちら」というリンクが表示されるようになります。

(409) ダウンロード機能の設定で「チェックボックスを区切り文字で区切って1つの列で出力する」を有効にすることでダウンロード形式が異なります。こちらもお試しの上、利用しやすい方法をお使いください。

図 パスワードリマインダへのリンクが表示される

このリンクを押下すると、図の画面が表示されます。

図 パスワード再設定画面

パスワード再設定を行うログオンアカウントを入力し、「送信する」ボタンを押下します。

図 パスワード再設定を行うログオンアカウントを指定する

ログオンアカウントに紐づいたメールアドレスに、パスワードを再設定するための URL が記載されたメールが送信されます。セキュリティの観点から、このURLは1時間だけ有効となっています。

図 メールが送信される

メール文面は次のようになっています。

```
*** 様(ユーザ名:***)

パスワード再設定のリクエストをお受け致しました。
以下のURLからパスワードの再設定を行って下さい。

http://YOURDOMAIN/wagby/setNewPassword.do?id=****

※このURLは2014年08月01日 10時xx分xx秒まで有効です。

**************************************************
なお、この内容にお心当たりの無い場合は、他の方が
メールアドレスを間違えて入力した可能性があります。
その場合、このメールは削除していただけますよう、
お願い申し上げます。
**************************************************
```

メール到着後、文中に記載されたURLのページを開きます。図のような画面が表示されます。

図 パスワード変更画面

新しいパスワードを再設定します。

図 パスワードの再設定を行った

この手順により、ログオンアカウントのパスワードが初期化されます。アカウントがロックされていた場合、ロックも解除されます。

セキュリティ上の問題と対策

この機能によるセキュリティ上の問題点と、その対策方法は次のとおりです。

- 登録メールアドレス以外にはパスワードなどの重要な情報の一切を開示しません。
- 本機能の利用可否や通知メールの送信先はログオンユーザ自身がいつでも変更することができます。
- パスワード変更用のURLの有効期限は1時間としています。
- パスワード変更用のURLを一度操作すると、1時間以内であっても同URLは無効となります。すなわち、このURLは一度利用すると使うことはできません。
- 本機能によってパスワードを変更したという行為はログに記録されます。
- 当該ログオンアカウントが「パスワード変更権限」を持たない場合、本機能は利用できません。
- 送信するメールの文面は自由に修正できます。
- 本機能を利用するとログオンユーザ自身がアカウントのロックを解除できるようになります。従って、退職や卒業等で特定のユーザのログオンに制限をかけたい場合は、アカウント更新画面で「アカウントロック情報」をセットすると共に「パスワード変更」及び「システム管理者」の権限（プリンシパル）を無効にしてください。

これらのシステム的な対策と並行して、利用者には「パスワードリセット詐欺」という攻撃が存在することも伝え、セキュリティ向上の教育を実施されることを推奨します。

- パスワードリセット詐欺［weblio辞書］

具体的にはメールの差出人および、クリックするURLが正しいことを目視で確認することがあげられます。

定義方法

パスワードリマインダの設定

環境メニューから、「アプリケーション>パスワードリマインダ」を開きます。「パスワードリマインダ機能を使用する」の欄に ″USE″ と記載するとともに、「差出人」の欄に適切なメールアドレスを入力します。

（利用者は差出人のメールアドレスを見て、届いたメールが妥当なものか、それとも疑わしいメールなのかを判断します。一般的にはシステム管理者につながるメールアドレスを指定するとよいでしょう。）

「本文」は自由に編集できます。文中の例にあるように、${username} などの表記が利用できます。

表記方法	説明	メール文中に展開される文字列の例
${username}	ログオンアカウント	yamada

${userid}	ユーザー名	山田
${url}	パスワード変更用として一時的に用意されたURL	http://localhost:8921/wagby/setNewPassword.do?id=...
${host}	urlのホスト部分	http://localhost:8921/wagby/
${path}	urlのホスト部から右側の部分	setNewPassword.do?id=... (410)
${limitdate}	パスワード変更用URLの有効期限	2014年09月01日 18時00分00秒

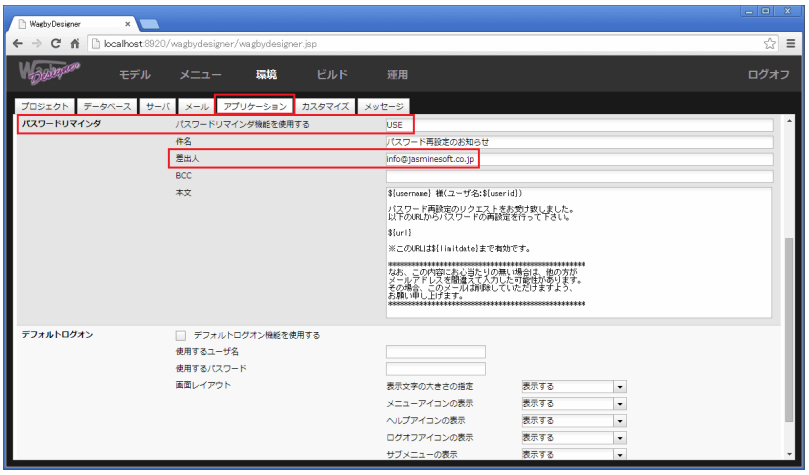


図 パスワードリマインダの設定

メールサーバの設定

パスワードリマインダ機能はメール送信処理を行うため、同じ環境設定の「メール」タブにて、メールホスト名を設定してください。

ご利用のメールホストがポート番号、プロトコル、SMTP認証などを設定する必要がある場合、「メール設定（送信）」欄を適切に記載してください。どのような値を設定するかは、貴社ネットワーク管理者にお尋ねください。

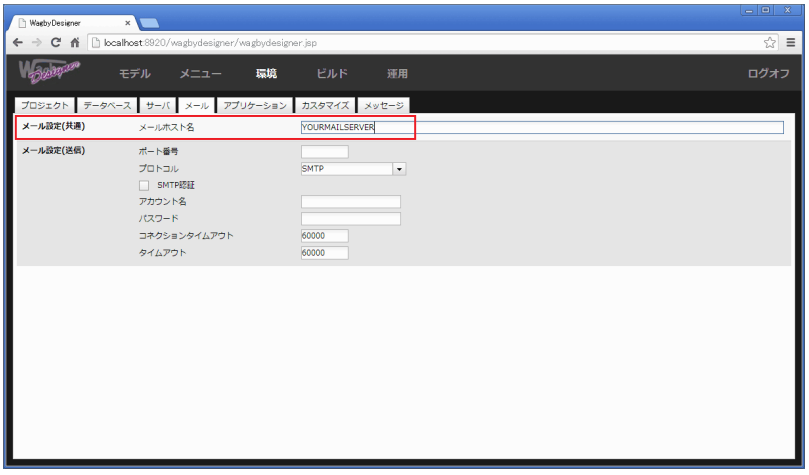


図 メール送信設定

(410) 通常はメール文中に ${url} を指定します。リバースプロキシを使う環境で、ホスト名を別に指定したいときは、固定文字列（ホスト名）の後に ${path} を指定するとよいでしょう。

メールアドレスを利用者に設定させる場合

パスワードリマインダ機能が有効になっているとき、各利用者の「プレファレンス」設定画面にて本機能を利用するかどうかを設定する欄が用意されます。

ここで「パスワードリマインダ」を「利用する」とし、かつ、入力フィールドにメールアドレスを指定することで、この利用者のログオンアカウントが、本機能を利用できるようになります。

メールアドレスの記載がない場合、本機能はご利用いただけません。

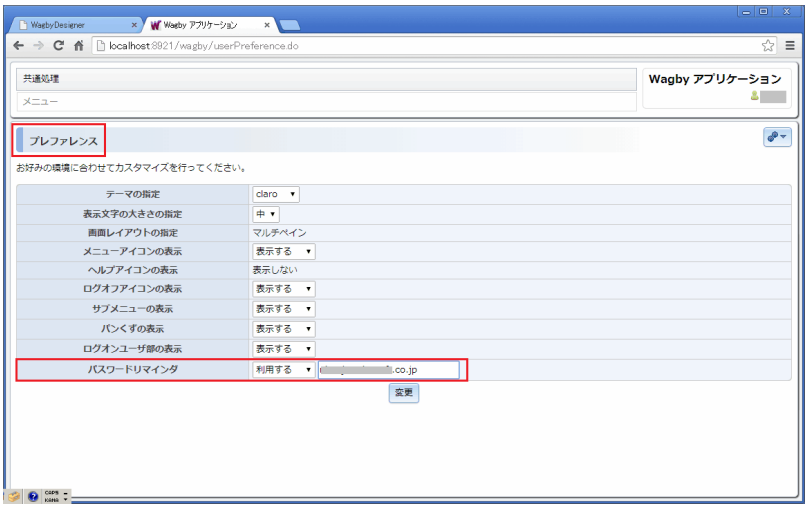


図 プレファレンスの設定

システム内部でログオンアカウントに紐づくメールアドレスを保持している場合

次のような場合は、こちらの方法をご利用されることを推奨します。

1. Wagbyが提供するログオンアカウント（juser）を拡張し、メールアドレス項目を用意している。
2. ログオンアカウントを主キーとする別モデルを定義しており、そこでメールアドレス項目を用意している。

この場合、すでに管理しているメールアドレス情報をパスワードリマインダ機能でも利用することができます。

定義方法

「パスワードリマインダ機能を使用する」の欄に ″USESQL″ と記載します。

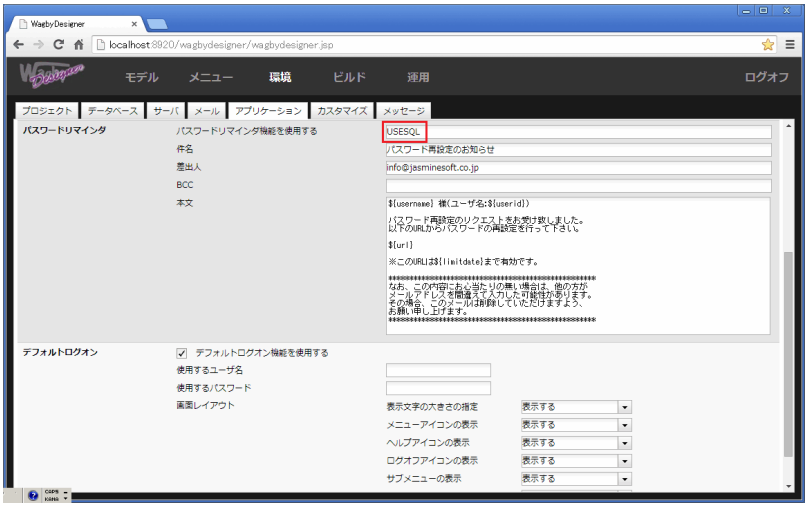


図 パスワードリマインダ機能を使用する

環境設定の「メール」タブにて、メールホスト名を設定します。

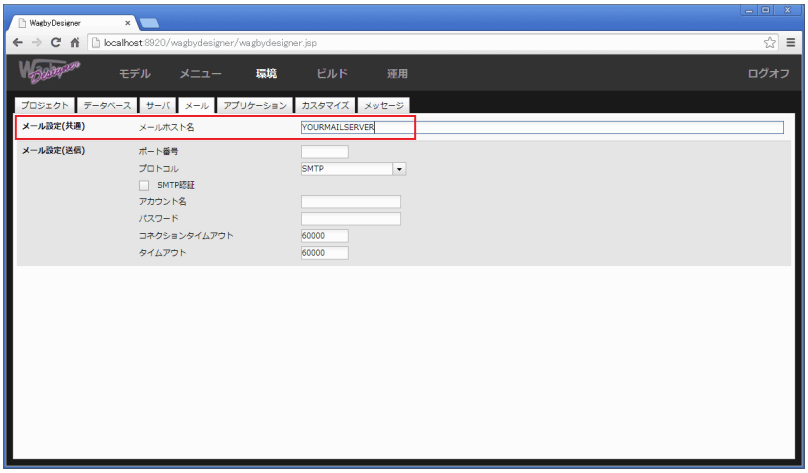


図 メール送信設定

jfcbase.propertiesの設定

「どのようなSQLを使ってメールアドレスを取得するか」を設定します。

wagbydesigner¥bin¥webpage¥WEB-INF¥src¥jfcbase.properties を customize¥resources¥myjfcbase.properties としてコピーします。

テキストエディタ（Windows OS に付属のメモ帳は使用不可です。その他のテキストエディタをご利用ください。）を用いて、コピーした myjfcbase.properties ファイルを編集します。

このファイル中に、次のような記載があります。

```
; jp/jasminesoft/jfc/HibernateUserInfo.java:
; HQL query
;
; case 1 : juser model
;jp.jasminesoft.jfc.HibernateUserInfo.getEmailByUserid.query=¥
; select c.email_ from jp.jasminesoft.jfc.model.juser.Juser as c where c.userid_ = :userid
;
; case 2 : staff model related to juser
;jp.jasminesoft.jfc.HibernateUserInfo.getEmailByUserid.query=¥
; select c.mailaddress_ from jp.jasminesoft.wagby.model.staff.Staff as c where c.userid_
 = :userid
```

"case 1" というエントリが「1. Wagbyが提供するアカウント定義（juser）の拡張」に対応します。この場合、case 1 以下の行について最初のセミコロン（;）を除去し、この設定を有効にします。

```
jp.jasminesoft.jfc.HibernateUserInfo.getEmailByUserid.query=¥
 select c.email_ from jp.jasminesoft.jfc.model.juser.Juser as c where c.userid_ = :userid
```

その上で、必要に応じてこのSQLを書き換えてください。[411]

具体的には select c.email_ ... の "email" 部に、定義ファイル上のメールアドレス項目名（英語）が入ります。[412]

"case 2" というエントリが「2. ログオンアカウントを主キーとする別モデルを定義する」に対応します。この場合、case 2 以下の行について最初のセミコロン（;）を除去し、この設定を有効にします。

```
jp.jasminesoft.jfc.HibernateUserInfo.getEmailByUserid.query=¥
 select c.mailaddress_ from jp.jasminesoft.wagby.model.staff.Staff as c where c.userid_
 = :userid
```

その上で、必要に応じてこのSQLを書き換えてください。具体的には select c.mailaddress_ ... の "mailaddress" 部に、定義ファイル上のメールアドレス項目名（英語）が入ります。from 句にあるモデル名（この例では Staff）も適切に修正してください。

> case 1 と case 2 は、どちらか一方の指定のみを有効にしてください。両方を有効にすることはできません。

今回の修正 *以外* のエントリは削除します。myjfcbase.properties ファイルはビルド時に元のファイルと自動マージされます。（削除した部分は、元のファイル jfcbase.properties の値が使われます。つまり myjfcbase.properties には、元のファイルとの差分情報だけを記述します。）

修正したファイルを保存し、フルビルドを行ってください。

利用者によっては別のメールアドレスを指定したい

上記のようにシステム内部で保持しているメールアドレスを利用する設定を行った場合でも、利用者が個別に「プレファレンス」画面から別のメールアドレスを入力することができます。この場合、プレファレンス画面の設定が優先されます。

パスワードリマインダ機能のためのメールアドレスを一括で登録したい

juser モデルや staff モデルなどに USESQL で参照されるメールアドレス項目を定義すれば、管理者がデータをCSVアップロード更新機能を用いて一括登録することが可能になります。

この場合でも、ユーザがプレファレンス画面でメールアドレスを指定すれば、そちらが優先されます。

パスワードリマインダ機能の利用制御

環境メニュー内の「カスタマイズ>プレファレンス」で、パスワードリマインダに関する「ユーザによる切り替えの可否」を指定することができます。

---

(411) Hibernate が解釈する HQL という書式になっています。

(412) 項目名およびモデル名は Hibernate マッピングファイルを直接参照し、実際のプロパティ名を確認する必要があります。例えば項目名が "mobile_email" のとき、プロパティ名は "mobileEmail_" となります。Hibernate マッピングファイルは wagbydesigner¥webapps¥wagbydesigner¥WEB-INF¥env¥work¥srcgen¥webpage¥WEB-INF¥classes フォルダ以下にモデル毎の ".hbm.xml" という拡張子形式で生成されています。

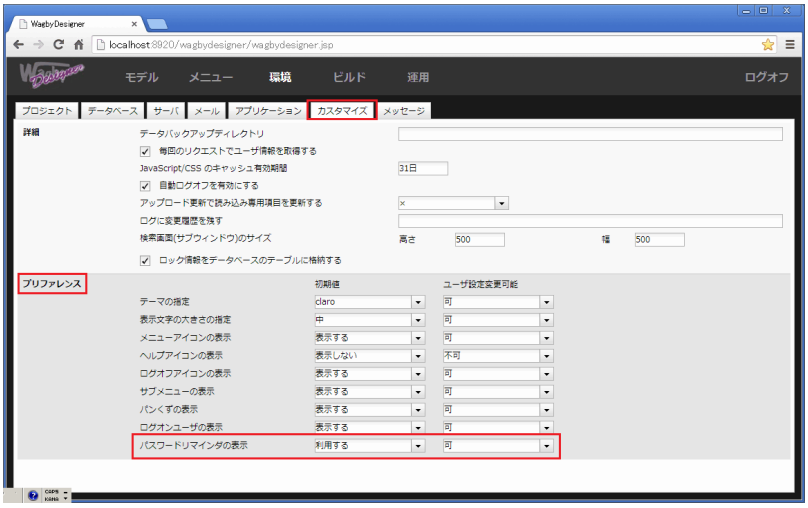


図 プレファレンスの設定

図の設定による挙動の違いは次のとおりです。

ユーザ設定変更可能が「可」の場合

アプリケーションの「プレファレンス」（このページにある図）で「パスワードリマインダ」を「利用する」とした場合は、その設定を優先します。

「利用しない」とした場合は、定義ファイルの「初期値」を優先します。

ユーザ設定変更可能が「不可」の場合

図の初期値が「利用する」となっていた場合、本機能を利用します。 図の初期値が「利用しない」となっていた場合、本機能を利用しません。

オートスケール環境での動作（8.3.1）

オートスケール/クラスタ環境では、このリマインダ情報を jfcreminder テーブルに格納します。（非オートスケール/クラスタ環境では、サーバのメモリで管理します。）

この jfcreminder テーブルは内部利用に限定されるため、Wagbyのモデルとしては定義されません。次の性質をもちます。

- オートスケール/クラスタ設定が有効な場合にのみ、作成される。
- juserモデルのインポート処理時にあわせて、このテーブルがdrop&createされる。
- Wagbyのインポート・エクスポート画面では、このモデルは表示されない。

よくある質問と回答

パスワードリマインダ機能を利用する/しない自体の設定を一括で指定できますか

できません。ユーザ自身がオプトインで機能を有効にするという設計思想になっているため、一括登録の仕組みは用意しない方針となっています。

パスワード初期化を連続して行なった場合はどうなりますか（8.3.1）

初期化のためのURLは1ユーザー1つのみが有効です。すなわちパスワード初期化URLを重複して作成した場合は、後に発行したもののみが有効です。

仕様・制約

パスワードの入力チェック

アカウント(juser)モデルのパスワード項目に設定した入力チェックルールは、パスワードリマインダにも適用されます。

ただし、入力チェックルールの一部は「フロントエンド　(Webブラウザの JavaScript)」で実行されるものがあります。例えば正規表現チェックです。このフロントエンドのチェックは、パスワードリマインダには適用されません。

具体的に説明します。パスワード入力チェックに正規表現チェックを適用した場合、パスワードの入力ルールは次のようになります。

チェックのタイミング	サーバでのチェック	フロントエンドでのチェック
アカウントモデル(juser)の登録・更新画面	○	○
利用者によるパスワード変更画面	○	○
パスワードリマインダ	○	−

(413)　(414)

IPアドレスによるログオン制限

定義方法と運用

アカウントの更新画面を開きます。　画面下部にある「ログオンルール」欄に接続を許可する　IP　アドレスを記述します。

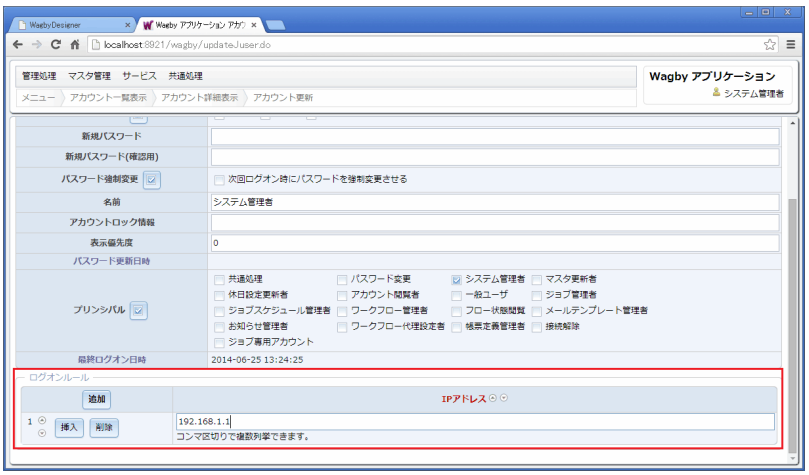


図 接続を許可するIPアドレスを指定する

記述ルールは次のとおりです。

- 本設定欄が未記入の場合は、IPアドレスによるログオン制約は行われません。
- 一つの設定欄に複数のIPアドレスをコンマ区切りで列挙することができます。
- IPアドレス表記は(Javaの)正規表現式に対応しています。

IPv4/IPv6の違い

稼働するOS環境によっては、IPv4形式の表記は受け付けない場合があります。

(413) パスワードリマインダでフロントエンドのチェックが機能しない理由は、固定のHTMLフォームとなっているためです。（自動生成の対象ではない。）

(414) フロントエンドのチェックがなくても、サーバ処理でパスワードの入力チェックはすべて実施されます。フロントエンドのチェックが機能した場合、一部のチェックが先に行われるようになります。これは UI の改善につながりますが、機能的には変わりません。

IPv4 形式のローカルマシン表記 "127.0.0.1" を記載してログオンできない場合は、IPv6 形式の表記 "0:0:0:0:0:0:0:1" で試してください。こちらだと正しくチェックされる場合は、IPv6 形式をご利用ください。

コンマの扱い

- 波括弧{}や角括弧[]内のコンマや、バックスラッシュ¥の後のコンマは区切り文字として扱いません。
- IPv4/IPv6のIPアドレスが指定されている場合は正規表現式として扱いません。

このルールにより、例えば次のような書式が可能です。

```
(0:){1,10}1,::1,127.0.0.1
```

::1はIPv6のIPアドレス(::と書くことで0:0:...を省略できる)、127.0.0.1はIPv4のIPアドレスです。

チェック時のログ出力

log4j.propertiesに下記の設定を行うことで、チェック時のログを出力します。

```
log4j.logger.jp.jasminesoft.jfc.service.LogonService=DEBUG
```

(0:){1,10}1,::1,127.0.0.1 と指定した場合のログの例を示します。

```
2016-xx-xx 0:00:00 [DEBUG jp.jasminesoft.jfc.service.LogonService checkRemoteAddress] not match
 ipaddress 127.0.0.1. rule regex:(0:){1,10}1
2016-xx-xx 0:00:00 [DEBUG jp.jasminesoft.jfc.service.LogonService checkRemoteAddress] not match
 inetaddress /127.0.0.1. rule inetaddress:/0:0:0:0:0:0:0:1
2016-xx-xx 00:00:00 [DEBUG jp.jasminesoft.jfc.service.LogonService checkRemoteAddress] match
 inetaddress /127.0.0.1. rule inetaddress:/127.0.0.1
```

ログオンに失敗した場合

ログオンに失敗した場合、ログオン画面に図のようなメッセージが表示されます。

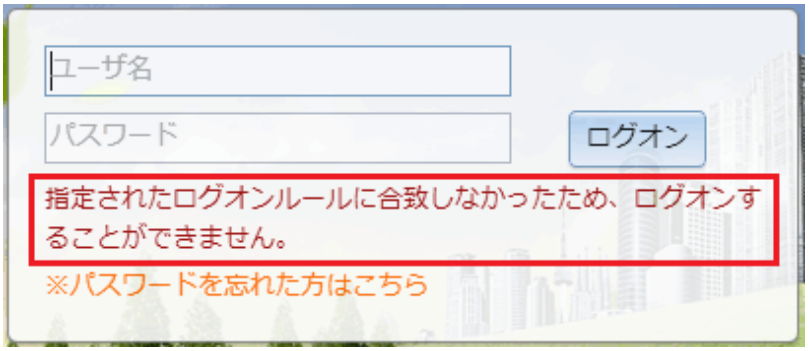


図 ルール違反によるログオン失敗

また、利用者端末のIPアドレスがログに記録されます。

```
JFC-00035:アカウント XXX は IP アドレス xxx の端末からログオンしようとしましたが、ログオンルールに合致しませんでした。
```

テスト

本設定を行った場合、指定したIPアドレス以外からアクセスされないことをテストしてください。 特に正規表現を用いてIPアドレスを記載した場合、正規表現の記述ミスで想定外のマシンからアクセスされることがないかどうか、必ず確認するようにしてください。

juserのサブモデルの作成

制約

juserのサブモデルを作成する場合、次の項目を含めることはできません。

- 現在のパスワード(passwdNow)
- 新規パスワード(passwd) (*1)
- 新規パスワード（確認用）(passwd2)
- パスワード強制変更 (compulsoryChange)
- パスワード更新日 (passwdChangeDate)
- パスワード更新フラグ (passwdChangeFlag)
- パスワード入力エラー数 (passwdErrCount)
- 過去のパスワード (oldPasswds)

サブモデル側でアカウントを作成した場合、パスワードが空白となるため、メインモデル側でパスワードを設定するという運用になります。（または passwd 項目の初期値を設定し、初期パスワードを決めておきます。）

(415) (416)

作成方法

入れ物となるモデルを用意する

はじめに、サブモデルとなる「入れ物」を用意します。新規モデルを作成します。

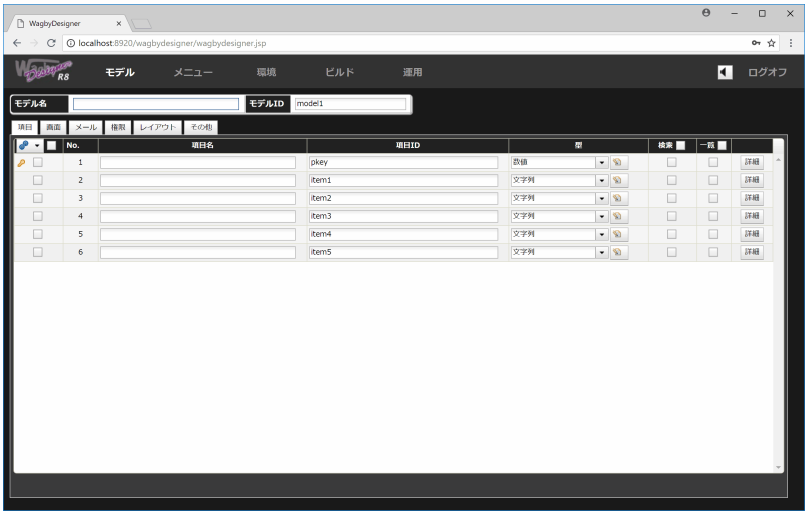


図 新規モデルの作成

モデル名、モデルIDを設定します。任意の名前を指定できます。ここでは "juser_sub" としておきます。

(415) passwdNow や passwd2 に初期値を設定することはできません。

(416) passwd項目は初期値が設定されていれば(juserの)サブモデルに含めることができます。これはサブモデルでのアカウント作成時にパスワードが空欄となるのを防ぐための措置です。しかしサブモデルでパスワードを変更することはできません。

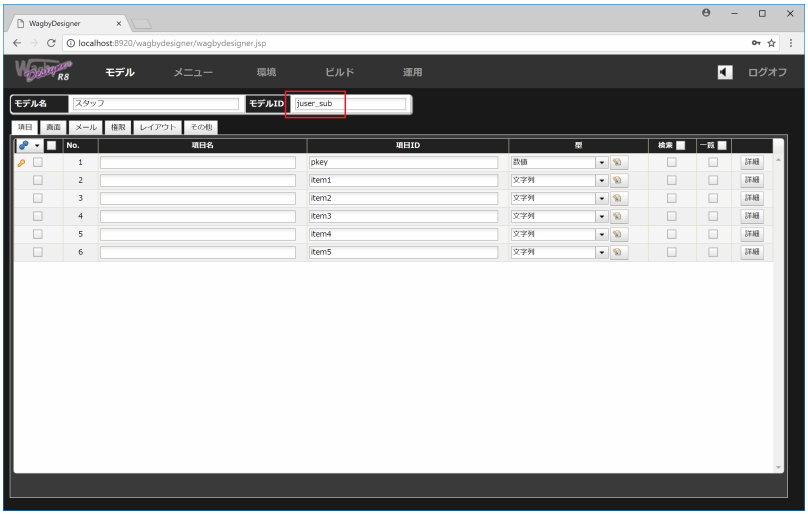


図 モデル名、モデルIDを指定する

最初に用意される項目をすべて消去します。

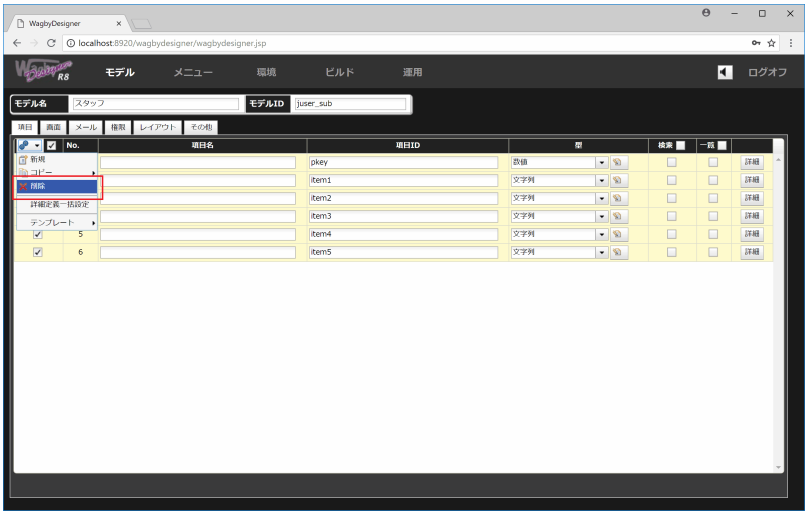


図 項目を消去する(1)

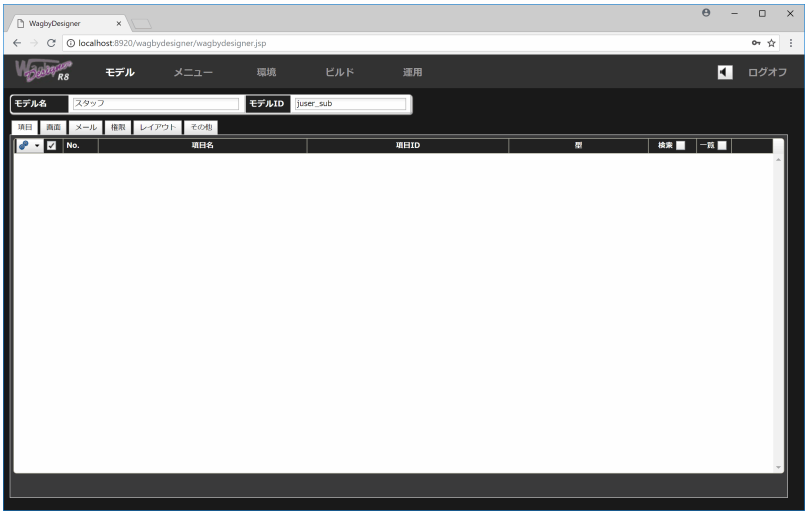


図 項目を消去する(2)

juserモデルから項目をコピーする

モデル ＞ システムから、アカウントモデル(juser)の定義を開きます。

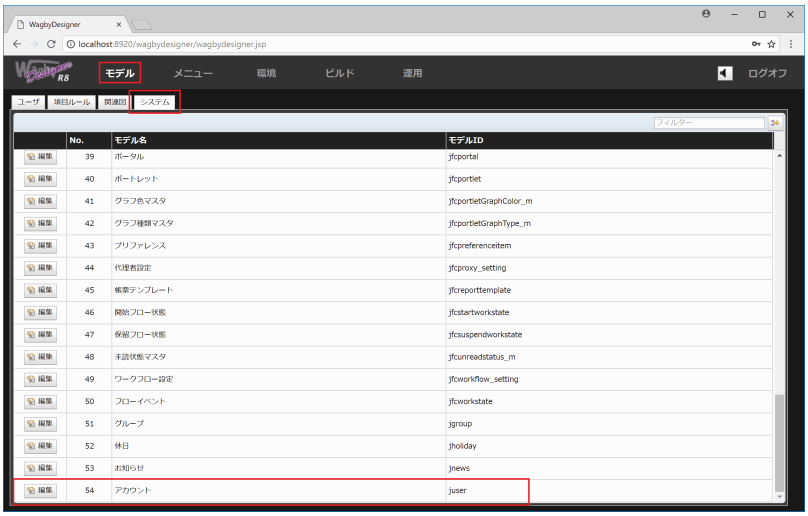


図　juserの定義を開く

すべての項目を選択します。

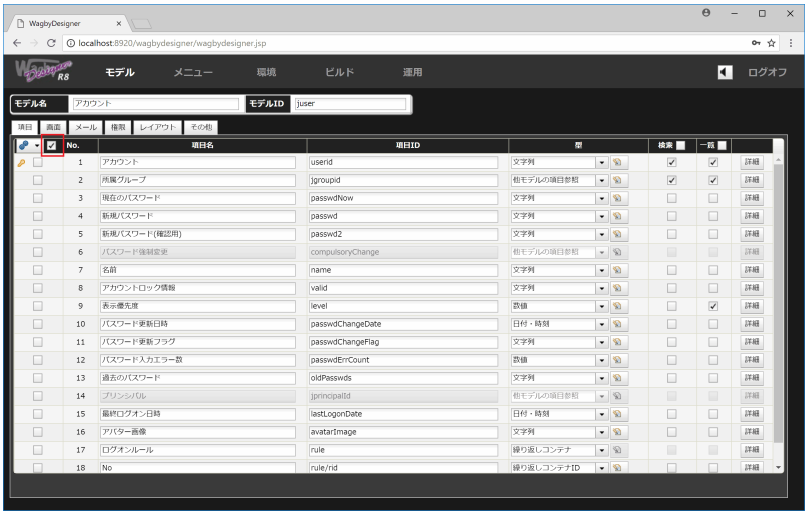


図　juserの項目をすべて選択する(1)

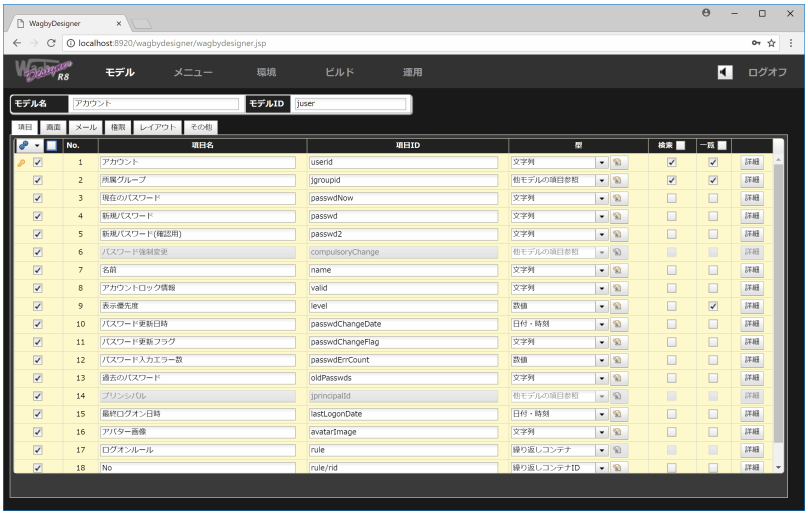
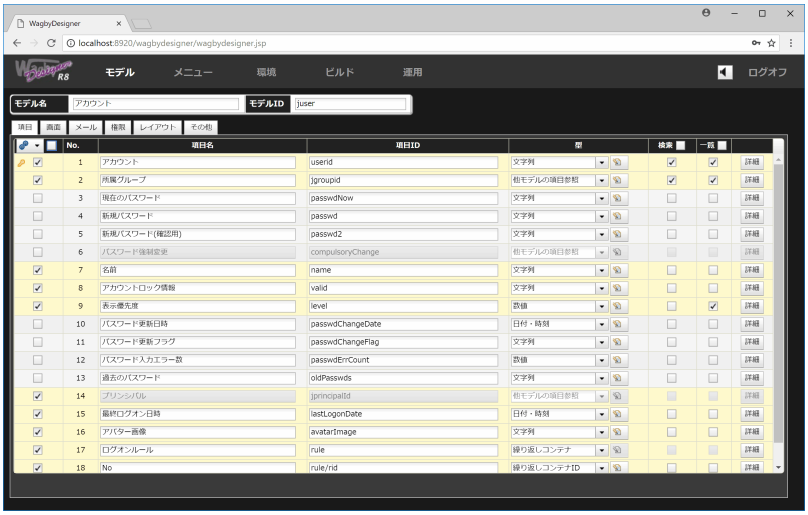


図　juserの項目をすべて選択する(2)

「制約」に合わせて、サブモデルでは保持できない項目を除きます。



Designerの項目コピー機能を使います。対象を ″他モデル″ とします。

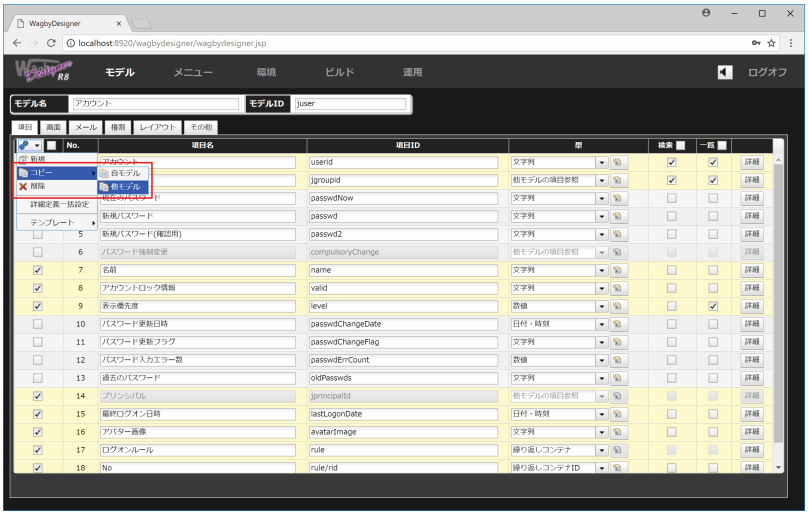


図 項目コピー(1)

対象モデルを選択するウィンドウで、先ほど用意した juser_sub モデルを指定します。

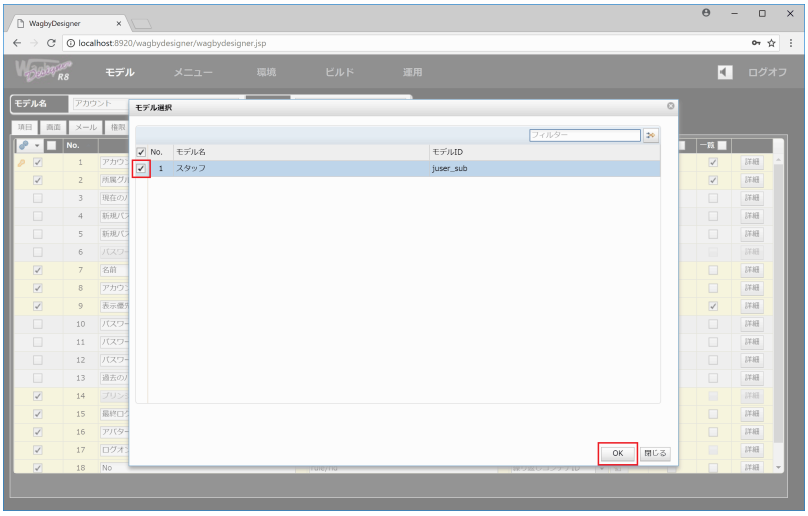


図 項目コピー(2)

確認ダイアログが表示されます。″はい″ ボタンを押下します。

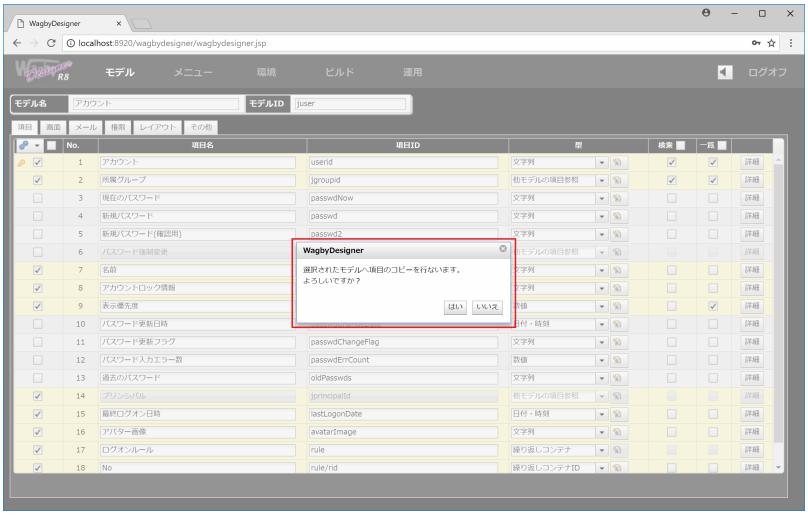


図 項目コピー(3)

juser_sub モデルを確認します。項目がコピーされていることがわかります。

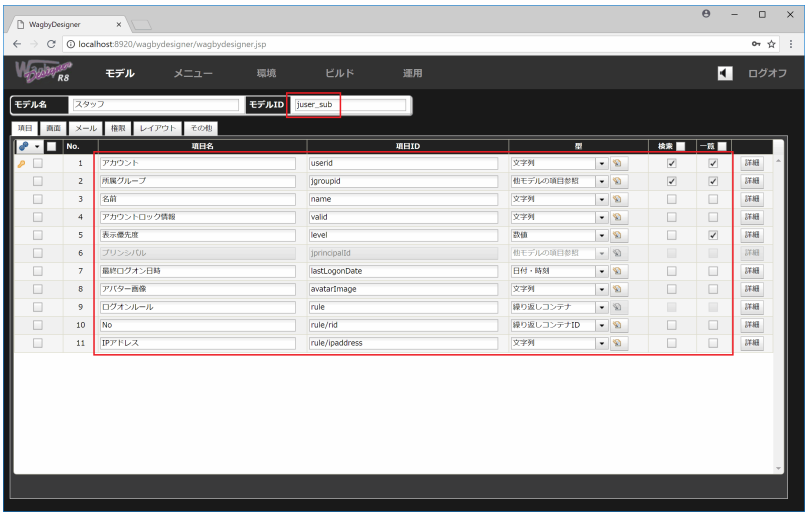


図 項目コピー(4)

(417)

サブモデルの設定を行います。画面 > その他 > モデルの関連性 > メインモデルに、"アカウント(juser)"を指定します。

(417) 項目がコピーされていないときは、リポジトリの再読み込みを行なってください。

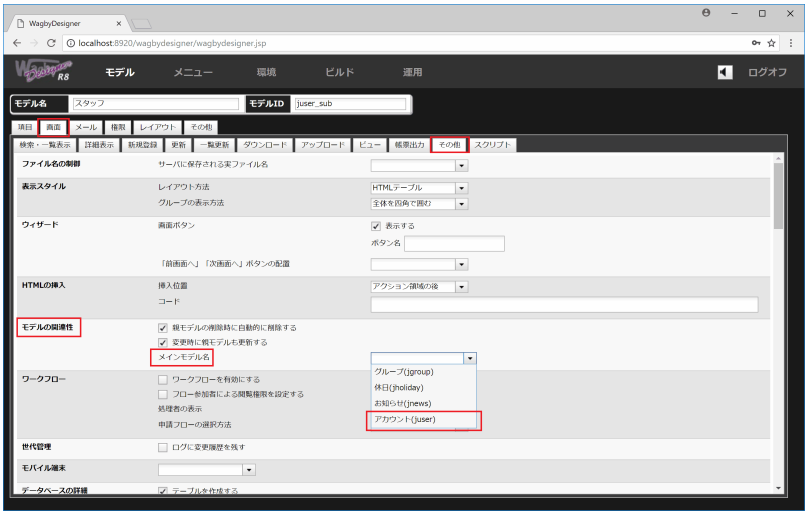


図 サブモデルの設定(1)

juser_sub モデルは、juser モデルのサブモデルになりました。

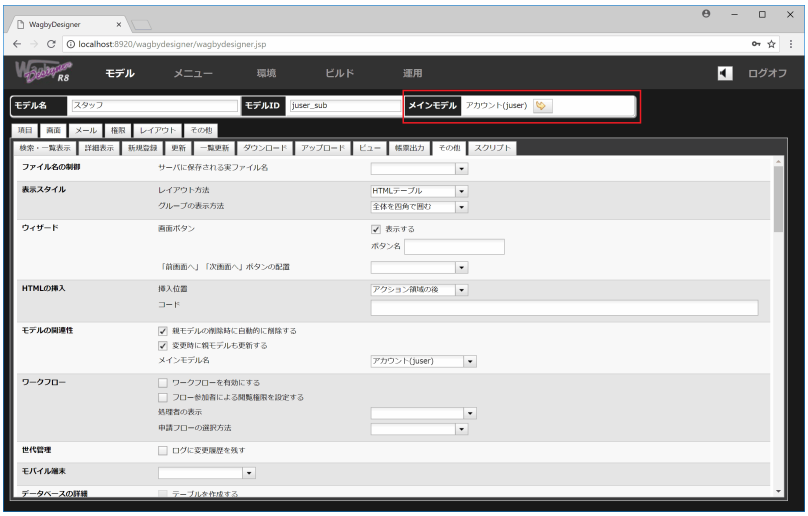


図 サブモデルの設定(2)

動作

ビルドしたアプリケーションを確認します。"スタッフ" というモデル名ですが、実際のデータは juser モデルを参照しています。

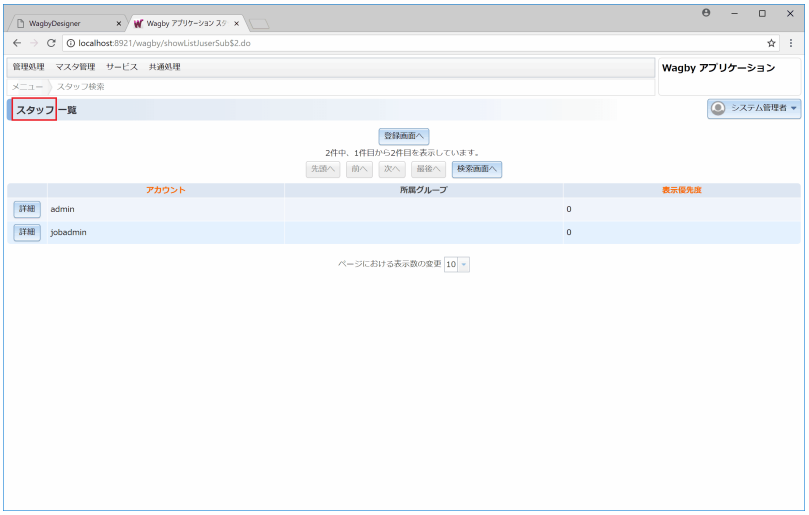


図 juser_subモデルの動作確認(1)

図 juser_subモデルの動作確認(2)

グループの使い方

グループの作成

「グループ」は管理者が作成します。メニューの「管理処理 > グループ検索」を選択します。

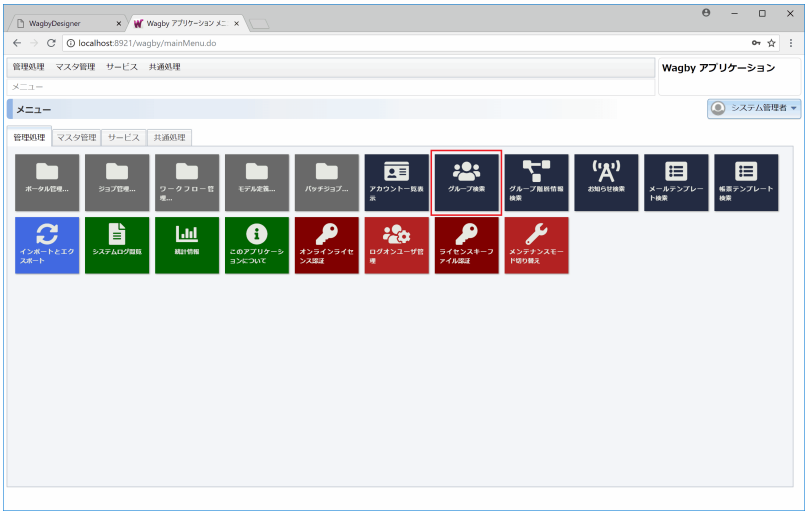


図 サブメニューの操作

「一覧更新」画面を押下します。

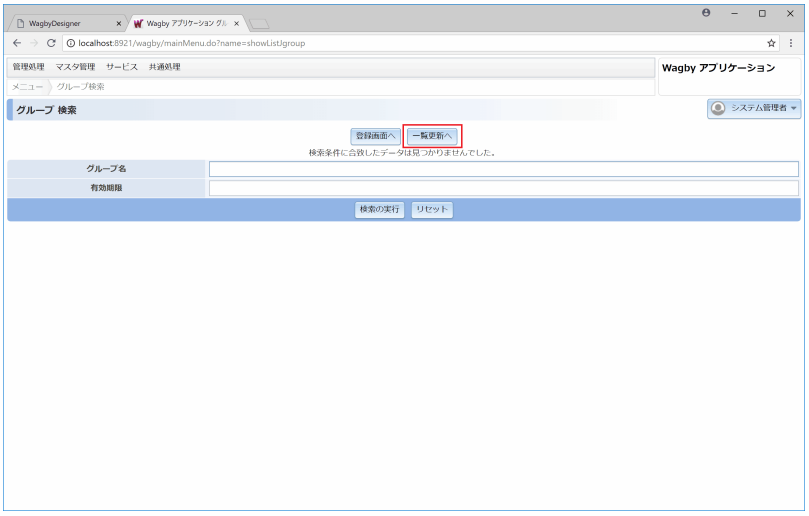


図 一覧更新画面へ

複数のグループを作成できます。 「有効期限」項目は、選択肢を無効にするために使うことができます。詳細は"他モデルの参照 > 選択肢モデル > 選択肢を無効にする"をお読みください。

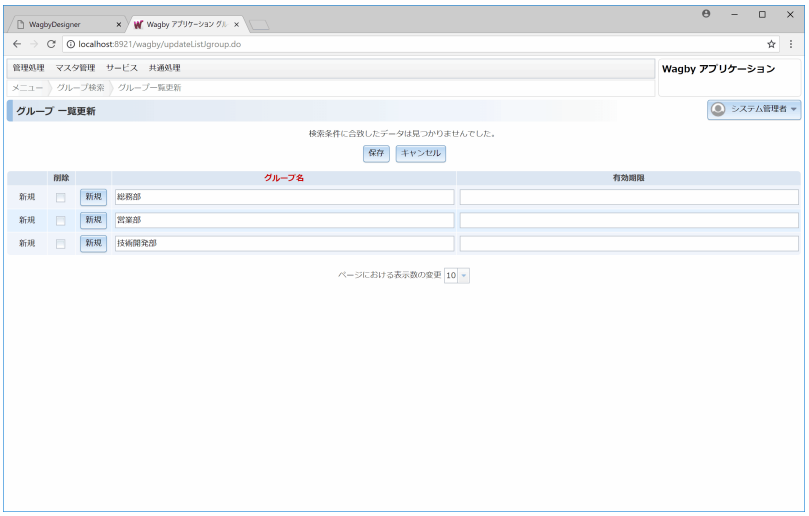


図 グループを作成する

データの登録を行った例です。この検索画面からファイルとしてダウンロードすることもできます。

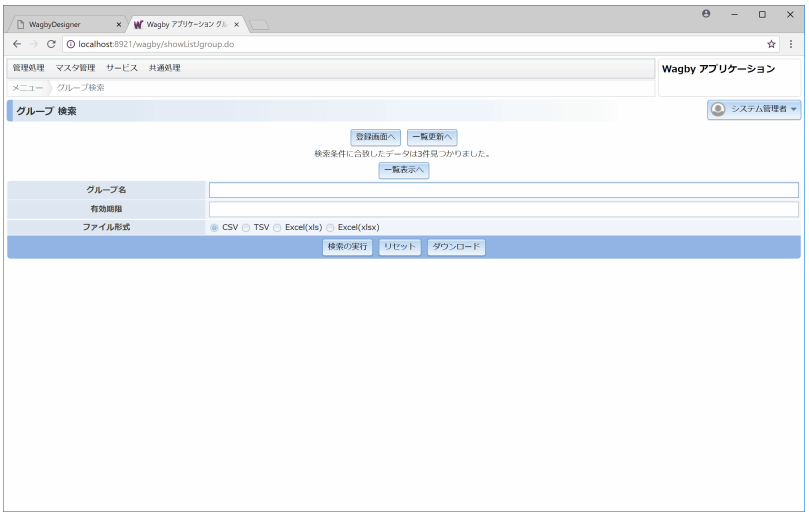


図 検索・一覧・ダウンロード画面

アカウントを登録する

アカウント登録時に、作成したグループを選択することができます。複数のグループに所属させることができます。

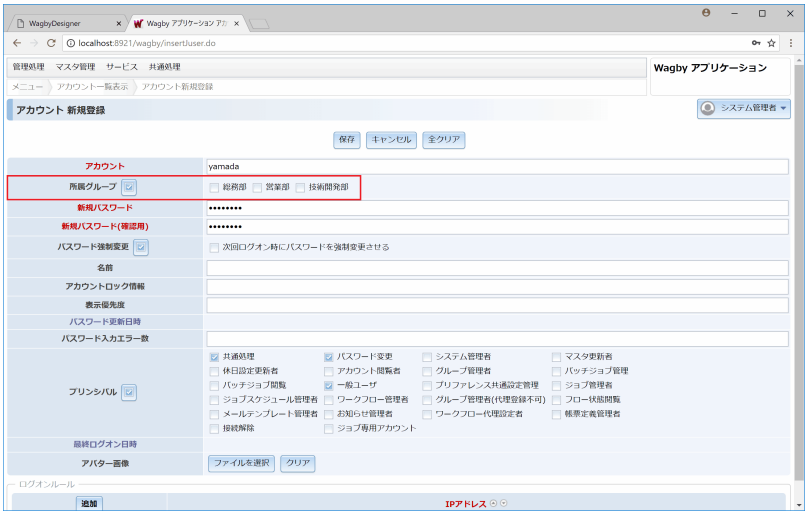


図 グループを指定する

グループの使い方

所属グループごとに権限を設定し、登録したデータの参照・更新の制御を行うことができます。

詳細は"認証・認可 > グループ権限"をお読みください。

グループ階層

Wagbyはグループの階層を設定することができます。指定できる親は一つだけです。親は複数の子グループを持つことができます。

はじめに、グループを事前に登録しておくものとします。

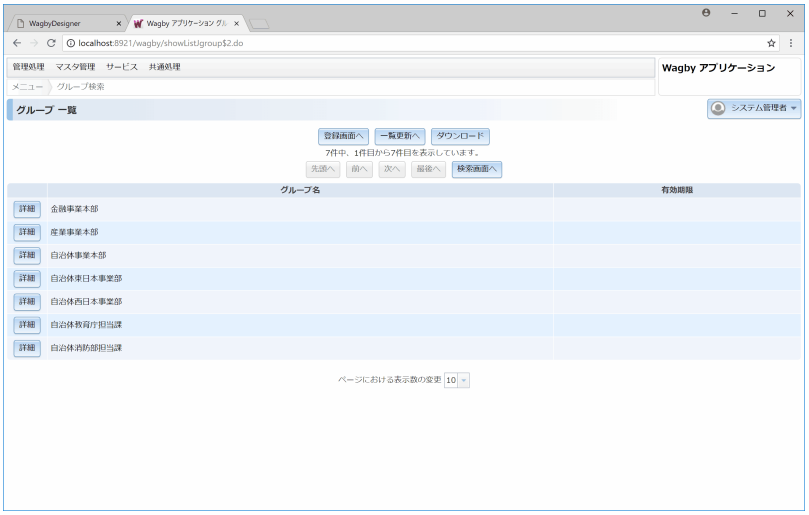


図 事前に登録済みのグループ（サンプル）

メニューから〝グループ階層情報検索〟を選びます。

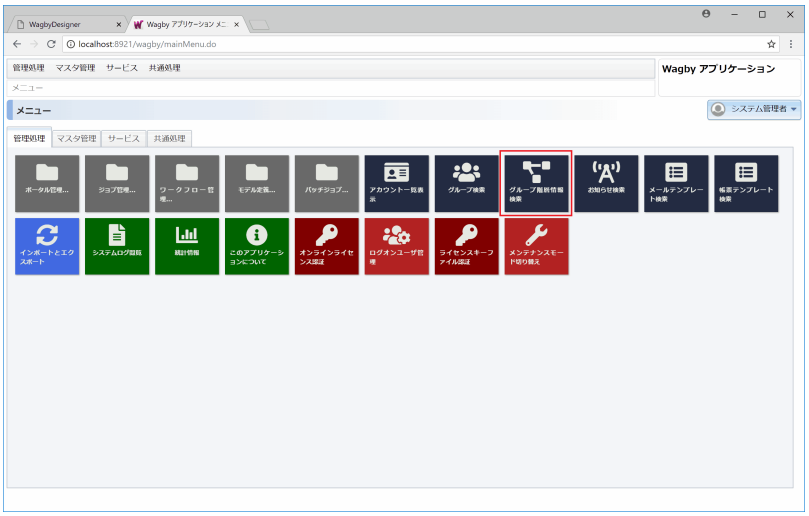


図 グループ階層情報検索

グループ階層情報は一覧更新画面を用意しています。ここでは一覧更新画面を使って登録する例を紹介します。

最初はどのグループも階層を設定しないため、すべてが〝未設定グループ〟に含まれています。この〝未設定〟状態をなくすようにしていきます。

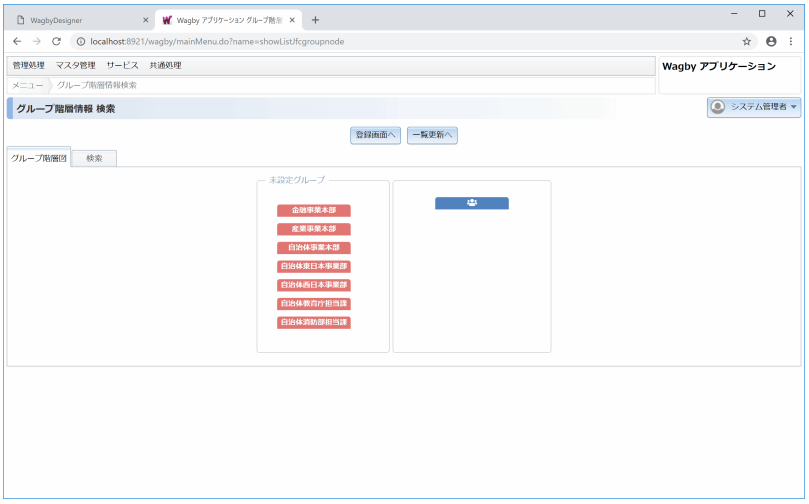


図 一覧更新画面へ

″グループID″ は、階層構造を指定する対象を指定します。Wagbyの参照連動機能で、IDで選択したグループ名が横に表示されます。

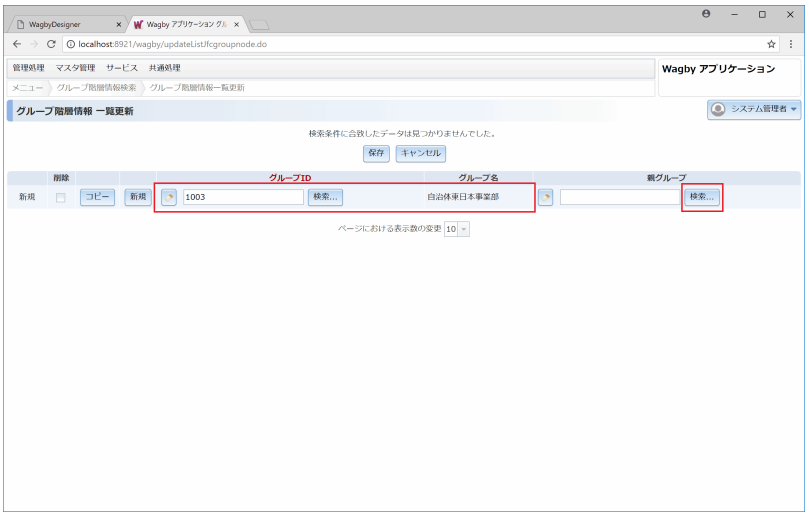


図 グループの決定

（自分の）一つ上の親グループを指定します。

図 親グループの指定(1)

親グループが指定されました。

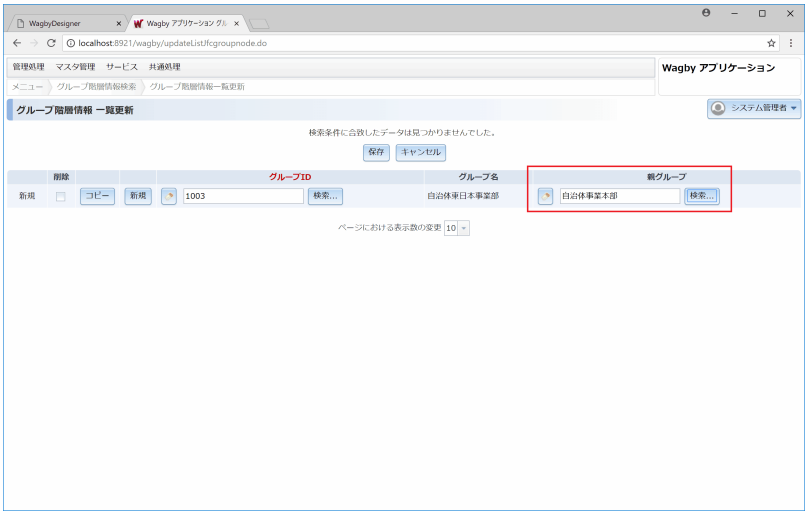


図 親グループの指定(2)

同じように、階層構造を設定していきます。

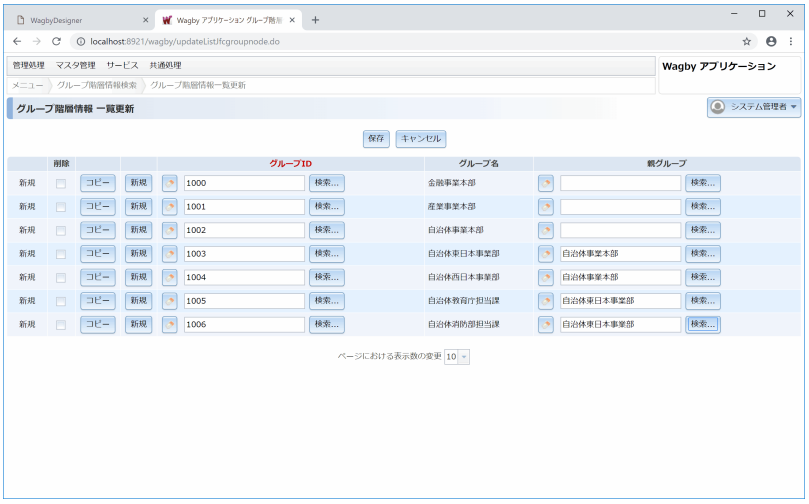


図 親グループの指定(3)

グループの階層構造を設定しました。"未設定グループ" が空になるように設定してください。

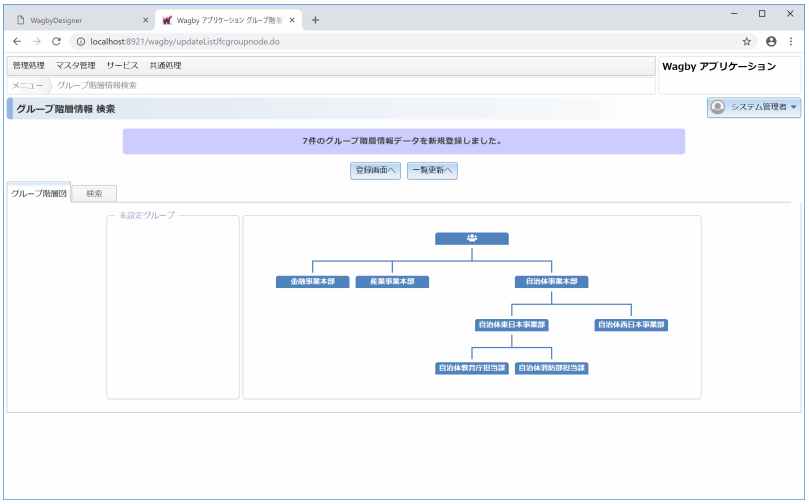


図 グループの階層構造を設定した(1)組織図表示

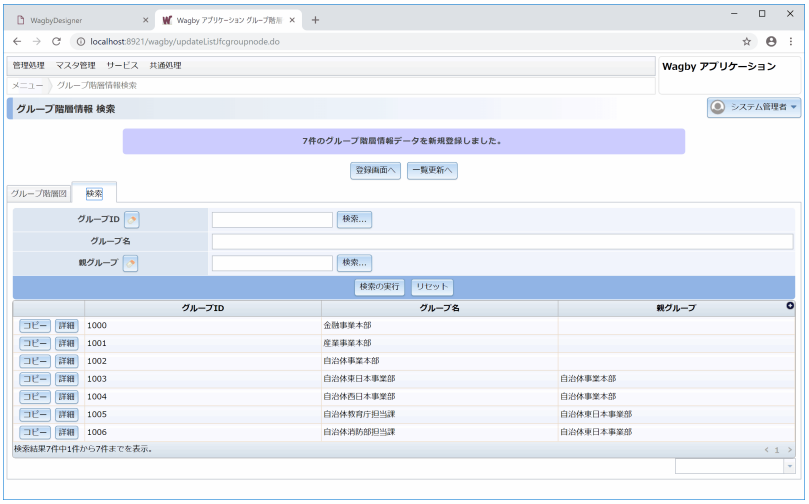


図 グループの階層構造を設定した(2)一覧表示

階層構造の情報を利用する

・ グループの階層構造を設定すると、グループ権限で階層情報を自動的に利用します。

・　ワークフロー設定では、階層構造を利用した承認者、決裁者を指定するスクリプトが記述できるようになります。

仕様・制約

・　グループ階層機能を利用する場合、すべてのグループ（jgroup）を漏れなく登録してください。そうでない場合組織図は表示されません。
・　階層構造の最上位のグループは、親のグループを指定しません。
・　標準では、組織図に表示できるグループ数の上限は 200 となっています。これを変更することができます。

Windows Active Directory/LDAP, SSO による認証

概要

Wagby は標準では（Wagby内部で管理している）アカウントとパスワードを用いたログオン認証を行います。ここで説明する方法を使うと、パスワード認証処理に Windows Active Directory / LDAP 認証, OpenID Connect, シングルサインオンを用いることができます。

認証方式の設定

「環境 > サーバ > 認証」を設定します。

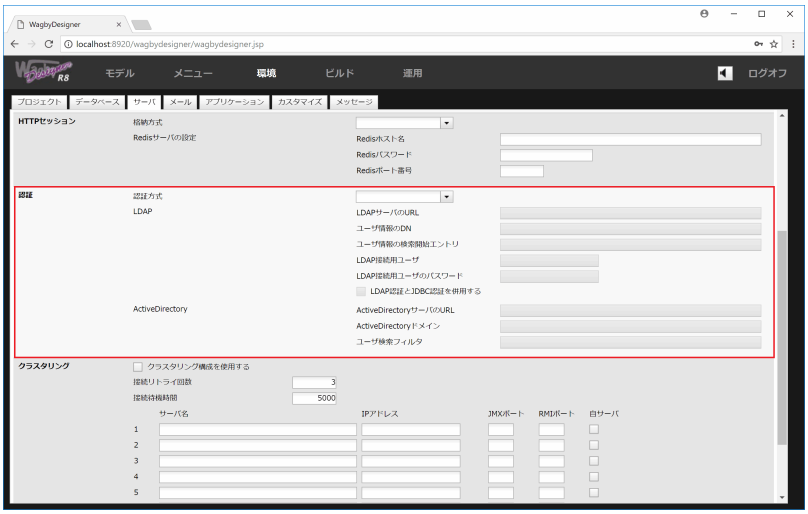


図 認証に関する設定

LDAP

認証方式に "LDAP" を指定します。次の設定欄が有効になります。

設定欄	説明	記述例
LDAPサーバのURL	LDAPサーバのURLを指定します。	ldap://ldap.jasminesoft.co.jp/
ユーザ情報のDN	ユーザ情報が登録されているディレクトリのパスを指定します。ユーザ名は「{0}」の部分に置き換わります。	cn={0}
ユーザ情報の検索開始エントリ	ユーザ情報の検索開始エントリを必要に応じて設定します。	dc=jasminesoft,dc=co,dc=jp

LDAP接続用ユーザ	LDAP 接続ユーザを指定します。匿名アクセスを許可している場合は設定不要です。	
LDAP接続用パスワード	LDAP 接続ユーザのパスワードを指定します。匿名アクセスを許可している場合は設定不要です。	
LDAP認証と juser の認証を併用する	LDAP認証 と juser テーブルを用いた認証（以下、JDBC認証と呼びます）を併用することができます。LDAP, JDBC いずれかのパスワードに合致すればログインできます。(*1) 併用は行うが、特別なアカウント(admin など管理者系)のみ JDBC 認証を行いたい場合は、JDBC 認証を行いたくないアカウントの juser テーブルの passwd カラムを NULL とすることで JDBC 認証ではログオンできなくなります。	

(418)

Active Directory

認証方式に "ActiveDirectory" を指定します。次の設定欄が有効になります。

設定欄	説明	記述例
ActiveDirectoryサーバのURL	ActiveDirectoryサーバのURLを指定します。	ldap://adserver.jasminesoft.co.jp/
ActiveDirectoryドメイン	ActiveDirectoryのドメインを指定します。	jasminesoft.co.jp
ユーザ検索フィルタ	ActiveDirectoryのユーザ検索フィルタを指定します。通常、このフィルタを指定する必要はありません。未設定時は "(&(objectClass=user)(userPrincipalName={0}))" が利用されます。	(&(objectClass=user)(sAMAccountName=XXXXX))

外部認証 (8.3.0)

認証方式に "外部認証(HttpServletRequest#getRemoteUser()を利用)" を指定します。 詳細はシングルサインオンの節で説明します。

アカウントとパスワードの設定

Windows Active Directory / LDAP に登録済みの「アカウント」と同じものをアカウントモデル(juserテーブル)の "userid" 項目に登録してください。なお juser テーブルの「パスワード」はダミーの値を設定してください。それ以外の juser テーブルの項目は任意に設定してください。[参考:juserの関連テーブル...]

(419)

アカウントの同期についての考え方

Windows Active Directory / LDAP 上のアカウントとの「自動同期」には対応していません。

---

(418) 1. そのためjuserテーブルに簡易なパスワードを設定しないように注意する必要があります。

(419) 実際の認証処理では、この（juserテーブルの）パスワードは使われなくなります。

一般的な運用は Windows Active Directory / LDAP をマスタとします。マスタに加わった変更情報を何らかの方法で抽出し、Wagby の juser テーブルへ反映させるようにします。

具体的には Wagby の REST API を呼び出す形の密結合的な同期から、差分情報を CSV 形式ファイルでまとめて、夜間にアップロード更新処理を行って同期するという方法などがあります。

仕様・制約

- LDAP と ActiveDirectory の併用を行うことはできません。
- ActiveDirectory と JDBC認証(juserテーブル利用)の併用を行うことはできません。

技術情報 設定ファイルの詳細

Wagby Designer の設定は application.properties に反映されます。

LDAP 用設定

```
wagby.security.ldap.url=${LDAPサーバのURL}
wagby.security.ldap.userSearchFilter=${ユーザ情報のDN}
wagby.security.ldap.userSearchBase=${ユーザ情報の検索開始エントリ}
wagby.security.ldap.managerDn=${LDAP接続用ユーザ}
wagby.security.ldap.managerPassword=${LDAP接続用ユーザのパスワード}
```

LDAP 認証とJDBC認証(juserテーブル利用)を併用する場合

```
wagby.security.useJdbcAuthentication=true
```

ActiveDirectory 用設定

```
wagby.security.activeDirectory.url=${ActiveDirectoryサーバのURL}
wagby.security.activeDirectory.domain=${ActiveDirectoryドメイン}
```

Spring Security について

Windows Active Directory / LDAP 認証は、Wagby に同梱されている Spring Security という認証・認可のためのオープンソース製品を用いて実現しています。開発者は Spring Security の仕組みを活かした認証処理のカスタマイズを行うこともできます。[技術資料の詳細...]

旧バージョンからの移行

R6/R7 から移行する場合は "R7リポジトリからR8リポジトリへの移行 > CASからSpringSecurityへの移行" をお読みください。

シングルサインオン

シングルサインオンとは、一回の認証（ログオン）で複数のアプリケーションを利用できるようにする仕組みです。シングルサインオンの実現には「認証サーバ」を用意する必要があります。認証サーバを経由することで、Windows Active Directory / LDAP といった認証方式以外にも、多くの認証に対応することができます。

Wagby は「認証サーバ」を同梱していません。すでに組織内で運用中、または新規に導入予定の認証サーバと Wagby を連携させることになります。

外部認証

認証方式に"外部認証(HttpServletRequest#getRemoteUser()を利用)" を指定することができます。

この設定を有効にすると、標準で同梱されているアカウント(juser)による認証は行われなくなり、request.getRemoteUser() から取得したユーザー名でWagbyのログオン処理が行われるようになります。この時、パスワードは不要です。

(420) (421) (422)

トラブルシューティング

ログの取得

想定どおりに動作しない場合、次の手順で詳細なログを取得することができます。

1. ログレベルの変更

wagbyapp/webapps/$(APPNAME)/WEB-INF/classes/log4j.properties の末尾に以下のログ出力設定を追加して下さい。

```
log4j.logger.org.springframework.security.ldap.authentication.ad.ActiveDirectoryLdapAuthenticationProvider=
log4j.logger.org.springframework.security.ldap.SpringSecurityLdapTemplate=DEBUG
```

または次のようにすると、すべてのLDAP関連のログを出力します。

```
log4j.logger.org.springframework.security.ldap=DEBUG
```

2. log4j.properties 変更後、Wagby アプリケーションを再起動します。
3. 下記のケースでログオンを行い、結果のログを確認してください。
   - LDAPに登録してあるID/パスワードを入力した場合。
   - 誤ったパスワードを入力した場合。
   - LDAPに登録していないIDを入力した場合。

エラーメッセージ

メッセージ	説明
Active Directory authentication failed: Supplied password was invalid	パスワードが誤っています。
Active Directory authentication failed: User was not found in directory	ADにユーザーが存在しません。

サポートポリシー

- 外部の認証サーバを利用した設定に関する問い合わせ（接続できない、どう設定するか、など）は Premium Support となります。認証サーバ、お客様環境、Wagbyアプリ、それぞれの環境に依存するためです。
- サポートはお客様の「認証サービス運用担当者」とやりとりします。これは認証サーバ製品に関する知識を有することが前提です。認証サービスの運用は、お客様環境の運用ポリシー、セキュリティポリシーと関わるためです。
- 本ページに記載の内容は、接続のための最低限の事柄を記載したものです。さらに追加設定が必要な環境もあるかも知れません。あらかじめご了承ください。

(420) request.getRemoteUser() からユーザー名が取得できない場合、または取得したユーザーが juser に存在しない場合はログオンできません。

(421) クライアントのWindows認証情報(REMOTE_USER)を使いたいという場合、この設定を有効にし、かつ、独自の Servlet Filter を用意して Wagby が利用している認証用ミドルウェア Spring Security にこれを登録する必要があります。[技術情報...]

(422) IIS や Apache などの Web サーバー側で REMOTE_USER を利用できるように設定している場合は独自 Servlet Filter は不要という場合もあります。詳細は認証処理に詳しいエンジニアにご確認ください。

OpenID Conneect による認証

概要

OpenID Connect (OIDC) は認証に関するオープンな規格の一つです。Google, LINE, Microsoft (Azure, ADFS), Oracle(Oracle Identity Cloud Service), PayPal, リクルートID, RedHat(keycloak), Salesforce, VMware, Yahoo! Japan など多くのサービスで利用できます。

OIDC の認可コードフローを示します。Wagbyは "Relying Party" という役割になります。"OpenID Provider" にはさまざまなベンダーがあります。ここでは Google を例にしています。

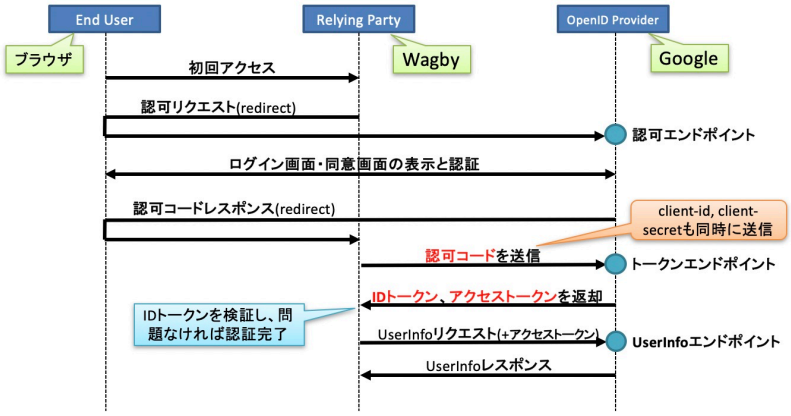


図 OIDC認可コードフロー

OIDC を利用すると、Wagbyのログオン画面は使われなくなります。代わって、Googleの認証画面が表示されます。

図 Googleへのログインを行う

ログオンに成功すると、Wagbyのメニュー画面に遷移します。

図 ログオン後はメニュー画面へ遷移する

WagbyとのID連携に失敗する(アカウントが存在しない等)とエラー画面に遷移します。

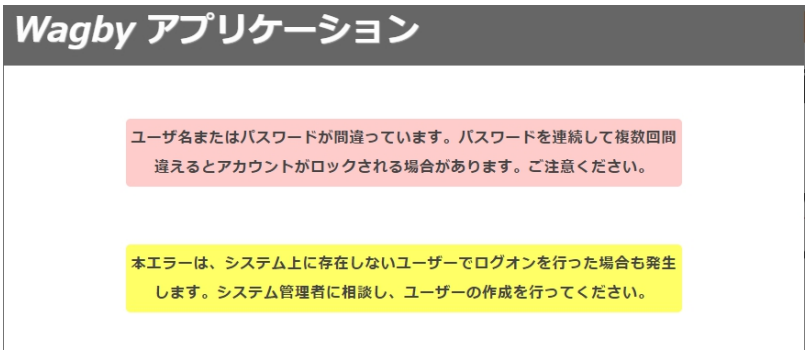


図 ログオンエラー

準備するもの

OIDC を利用するにあたって、次の情報を事前に取得しておきます。

OpenID Connect Provider(OP)で用意するもの
- クライアントID(client-id)
- クライアントシークレット(client-secret)
- 各エンドポイントのURL(認可、トークン、UserInfo)

Relying Party(Wagby)で用意するもの
- フローの種類
  - Webアプリの場合は "認可コードフロー" となる。
- フローの実装コード
  - Wagbyには "Spring Security" が同梱されているので、このままでよい。
- ログイン後(認可コードレスポンス時)にOPからリダイレクトするURL
  - Spring Security利用時は次のようになる。赤字は可変部分。

```
http://localhost:8921/wagby/login/oauth2/code/google
```

Wagby が同梱している Spring Security は OpenID Connect に対応しています。そのため Relaying Party として用意すべきものはほぼ、揃っています。

一方で、OpenID Connect Provider から取得すべき情報は、利用するサービスごとに異なります。ここで取得した情報を Spring Security のパラメータとして図の欄に記載します。

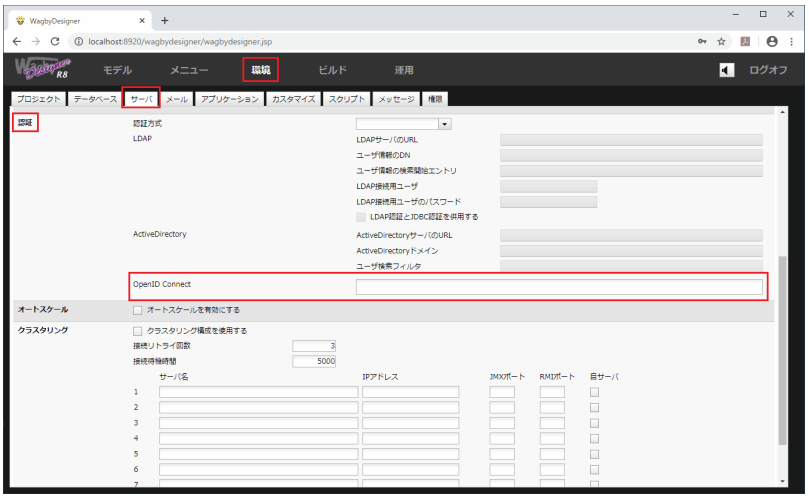


図 OIDCに関する設定を記載する欄

Googleを利用する

OpenID Connect Provider(OP) として Google を利用した例を説明します。次の情報を取得します。

- クライアントID(client-id)
- クライアントシークレット(client-secret)
- 各エンドポイントのURL(認可、トークン、UserInfo)

ただしエンドポイントURLはすでに定められています。次の通りです。

エンドポイント種別	URL
認可エンドポイント	https://accounts.google.com/o/oauth2/v2/auth
トークンエンドポイント	https://www.googleapis.com/oauth2/v4/token
UserInfoエンドポイント	https://www.googleapis.com/oauth2/v3/userinfo
JWK Setエンドポイント	https://www.googleapis.com/oauth2/v3/certs

Spring Security を利用する場合、上記の情報は標準値となっているため、省略できます。そのため実際には「クライアントID」と「クライアントシークレット」だけを取得する必要があります。

Google Developers Console

クライアントIDとクライアントシークレットを取得するため、Google Developers Console を使います。ここではすでに Google のアカウントを持っていることを前提にしています。また Google Developers Console の詳細は割愛します。

はじめに、新規プロジェクトを作成します。

図 新しいプロジェクトの作成(1)

ここではプロジェクト名を "demo001" としています。また、場所は "組織なし" としています。

(423)

図 新しいプロジェクトの作成(2)

OAuth 同意画面とは、図にある（Googleが用意した）画面をいいます。まず入力したプロジェクト名 "demo001" となっていることを確認します。問題なければ "User Type" を指定します。

(423) 続く説明は、適切なプロジェクト名に置換してお読みください。

図 OAuth同意画面を用意する(1)

「アプリケーション名」と「サポートメール」を指定します。ここではアプリケーション名として"Wagby OIDCデモアプリ" としています。サポートメールは Google から送られてくる管理用メールの受け取り先を指定します。

図 OAuth同意画面を用意する(2)

「認証情報」を指定します。"OAuthクライアントID" とします。

図 認証情報の設定

OAuthクライアントIDを作成します。「アプリケーションの種類」を　"ウェブアプリケーション"　とします。「名前」は任意です。ここでは　"ウェブクライアント1"　としています。「承認済みのリダイレクトURI」を "http://localhost:8921/wagby/login/oauth2/code/google" としています。

(424)

図 OAuthクライアントIDの作成

OAuthクライアントIDを作成すると、クライアントIDとクライアントシークレットが取得できるようになります。図のようにポップアップダイアログに表示されますので、コピーしておいてください。

(424) ローカルでの動作検証時は URI に localhost:8921 と指定することができます。本番環境では http://localhost:8921/wagby の部分は適切な値に変更してください。

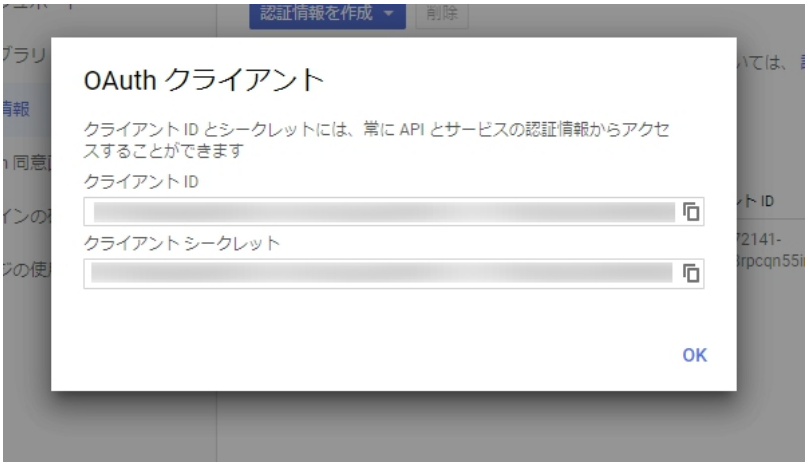


図 クライアントIDとクライアントシークレットが表示される

Wagbyの設定

認証方式

"環境 > サーバ > 認証 > 認証方式" を設定します。 認証方式に "外部認証(HttpServletRequest#getRemoteUser()を利用)" を指定します。

この設定を有効にすると、標準で同梱されているアカウント(juser)による認証は行われなくなり、request.getRemoteUser() から取得したユーザー名でWagbyのログオン処理が行われるようになります。この時、パスワードは不要です。

図 外部認証の指定

OpenID Connect

"環境 > サーバ > 認証 > OpenID Connect" を設定します。この設定欄に記述した内容は、Spring Securityが利用するプロパティファイル application.properties に反映されます。

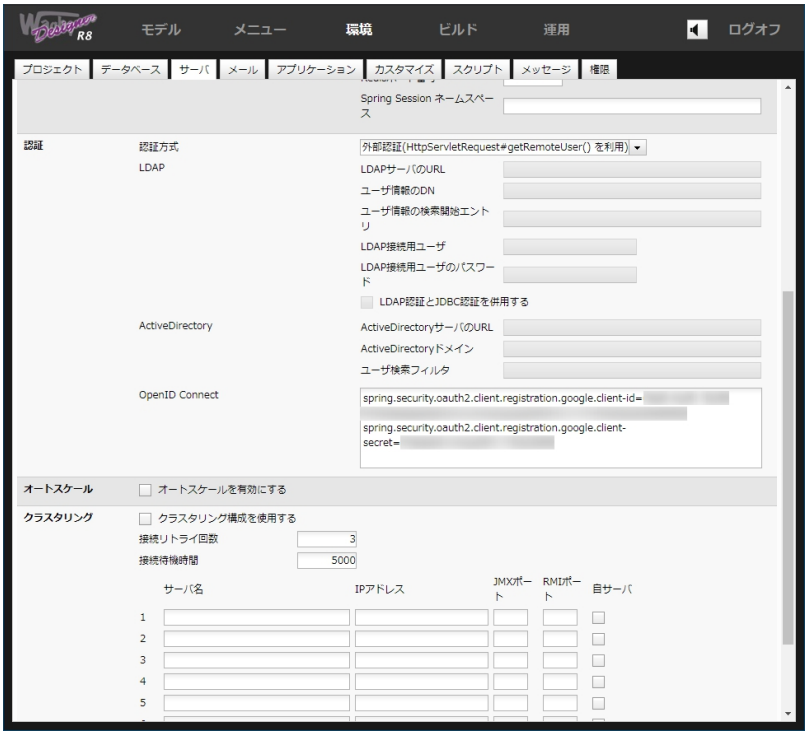


図 OpenIDConnectの設定(Google用)

仕様上は、以下の設定が必要になります。

```
spring.security.oauth2.client.registration.google.client-id=${クライアントID}
spring.security.oauth2.client.registration.google.client-secret=${クライアントシークレット}
spring.security.oauth2.client.registration.google.provider=google
spring.security.oauth2.client.registration.google.scope=openid,profile,email
spring.security.oauth2.client.registration.google.redirect-uri-template={baseUrl}/login/oauth2/
code/{registrationId}
spring.security.oauth2.client.registration.google.client-authentication-method=basic
spring.security.oauth2.client.registration.google.authorization-grant-type=authorization_code
spring.security.oauth2.client.provider.google.authorization-uri=https://accounts.google.com/o/
oauth2/v2/auth
spring.security.oauth2.client.provider.google.token-uri=https://www.googleapis.com/oauth2/v4/
token
spring.security.oauth2.client.provider.google.user-info-uri=https://www.googleapis.com/oauth2/
v3/userinfo
spring.security.oauth2.client.provider.google.jwk-set-uri=https://www.googleapis.com/oauth2/v3/
certs
spring.security.oauth2.client.provider.google.user-name-attribute=email
```

ところが Spring Security には google に関する設定が最初から組み込まれています。そこでクライアントIDとクライアントシークレット以外は省略可能となっています。よって開発者が設定するのは次の二行です。

```
spring.security.oauth2.client.registration.google.client-id=${クライアントID}
spring.security.oauth2.client.registration.google.client-secret=${クライアントシークレット}
```

[重要] クライアントID、クライアントシークレットの扱い ここまで、Designerに設定する方法を説明しましたが、Wagbyのリポジトリはテキスト情報であるため、仮にリポジトリが外部に渡されると、これらの情報を読み取ることができます。セキュリティ上、問題となる懸念があります。 この対策として、実際にはリポジトリには記述せず、ビルド後のアプリケーションに含まれる WEB-INF/classes/application.properties に直接クライアントID、クライアントシークレットを指定すると安全性を高めることができます。

テストアカウントを用意する

認証方式に "外部認証(HttpServletRequest#getRemoteUser()を利用)" を指定したため、認証されたアカウント（Wagbyのユーザー）が juser に存在しない場合はログオンできません。このため、OP で認証が通っても Wagby 側に該当ユーザーが存在しない場合はログオン失敗となります。

内蔵DBを利用している場合は、次のようなカスタマイズファイルを作成しておくとビルドの DB 初期化時に自動的にアカウントがインポートされます。この例では「システム管理者権限」を付与しています。

(425)

customize/webapp/WEB-INF/export/init/juser/item_1000.xml

```
<?xml version="1.0" encoding="UTF-8"?>
  <juser>
  <userid>taro@gmail.com</userid>
  <passwd>dummy</passwd>
  <name>ジャスミン太郎</name>
  <level>0</level>
  <jprincipalId>3</jprincipalId>
</juser>
```

(426) (427) (428)

同期は手動

Wagbyのアカウント(juser)と、OPのアカウントの同期は自動的には行われません。同期の方法は別途、検討する必要があります。詳細はWagby販売代理店へご相談ください。

ユーザ識別子

実は Google では、一人の利用者が複数のメールアドレスを保有することができます。そのため、OP に Google を利用したとき、認証後に渡されるアカウントの情報は、その人を特定する ID として、数字の羅列が返される仕様となっています。これが「ユーザ識別子」です。

しかしこの仕様どおりとすると、Wagby側に用意するアカウントも "taro@gmail.com" ではなく、"123456789..." のような数字が並んだものとなります。これはわかりにくいため、WagbyではOPから返される値のうち、メールアドレスを使うようにしています。これは次の設定となっています。

```
spring.security.oauth2.client.provider.google.user-name-attribute=email
```

この設定が含まれているため、Googleログイン時に用いたメールアドレスと同じものをWagbyアカウントとして用意することで連携できるようにしています。

ユーザ識別子をカスタマイズする

OPから受け取ったユーザ識別子をカスタマイズし、それをWagbyアカウントとして利用することもできます。[詳細...]

Azure Active Directory を利用する

OpenID Connect Provider(OP) として Azure Active Directory (AAD) を利用した例を説明します。次の情報を取得します。

- クライアントID(client-id)
- クライアントシークレット(client-secret)
- テナントID
- グループ名
- 各エンドポイントのURL(認可、トークン、UserInfo)

ただし各エンドポイントのURLは省略できます。Wagby には Microsoft が提供している AAD 接続設定用クラスが同梱されているためです。

---

(425) 外部DB利用時は、以下の内容に沿ったテストアカウントを事前にjuserテーブルに用意してください。

(426) useridの値は、実際に Google に用意された、存在するアカウントである必要があります。

(427) passwdの値は何でもかまいません。この値は利用されません。

(428) jprincipalId "3" は、システム管理者権限を指します。8.4.0

AADはテナントごとに独立しているため、${テナントID} を指定します。これによって AAD が一意に特定されます。

${グループ名}には連携対象のユーザが所属するグループ名を指定します。

クライアントIDとテナントIDの取得

以下のページの "AD テナントへのアプリケーションの登録" を参考にしてください。 これは Azure Active Directory は構築済みという前提で、Open ID Connect 接続のための設定を行う手順となっています。

https://docs.microsoft.com/ja-jp/azure/active-directory/develop/quickstart-register-app#register-a-new-application-using-the-azure-portal

ローカルホストでの動作確認の場合は、"リダイレクト URI" 次のようになります。赤字は可変部分です。

```
http://localhost:8921/wagby/login/oauth2/code/azure
```

(429)

設定後に "アプリケーション（クライアント）ID" と "ディレクトリ（テナント）ID" が 表示されます。これをメモしておいてください。

クライアントシークレットの取得

以下のページの "資格情報を Web アプリケーションに追加する" を参考に "新しいクライアント シークレット" の追加を行います。なお、手順2の「証明書の追加」は今回は不要です。

https://docs.microsoft.com/ja-jp/azure/active-directory/develop/quickstart-configure-app-access-web-apis#add-credentials-to-your-web-application

この対応で "クライアント シークレット" の値が表示されます。これをメモしておいてください。

マニフェストの変更とグループ名の取得

以下のページを参考にマニフェストを編集します。

https://docs.microsoft.com/ja-jp/azure/active-directory/develop/quickstart-configure-app-expose-web-apis#expose-a-new-scope-or-role-through-the-application-manifest

具体的には2つの設定値を false から true に変更して保存します。

```
"oauth2AllowIdTokenImplicitFlow": true,
"oauth2AllowImplicitFlow": true,
```

最後に、認証に用いるグループを確認します。 https://portal.azure.com/ の "Azure サービス > Azure Active Directory > グループ" で表示されるグループ名をメモしておいてください。

Wagbyの設定

認証方式

"環境 > サーバ > 認証 > 認証方式" を設定します。 認証方式に "外部認証(HttpServletRequest#getRemoteUser()を利用)" を指定します。

この設定を有効にすると、標準で同梱されているアカウント(juser)による認証は行われなくなり、request.getRemoteUser() から取得したユーザー名でWagbyのログオン処理が行われるようになります。この時、パスワードは不要です。

(429) ローカルでの動作検証時は URI に localhost:8921 と指定することができます。本番環境では http://localhost:8921/wagby の部分は適切な値に変更してください。

図 外部認証の指定

OpenID Connect

"環境 > サーバ > 認証 > OpenID Connect" を設定します。この設定欄に記述した内容は、Spring Security が利用するプロパティファイル application.properties に反映されます。

これまでにメモした "クライアント ID" と "クライアント シークレット"、"テナントID"、"グループ名" を customize/resources/myapplication.properties に次のように記述します。

```
spring.security.oauth2.client.registration.azure.client-id=${クライアント ID}
spring.security.oauth2.client.registration.azure.client-secret=${クライアント シークレット}
azure.activedirectory.tenant-id=${テナントID}
azure.activedirectory.active-directory-groups=${グループ名}
```

グループ名が複数ある場合は "," で区切って記述します。

[重要] クライアントID、クライアントシークレットの扱い ここまで、Designerに設定する方法を説明しましたが、Wagbyのリポジトリはテキスト情報であるため、仮にリポジトリが外部に渡されると、これらの情報を読み取ることができます。セキュリティ上、問題となる懸念があります。 この対策として、実際にはリポジトリには記述せず、ビルド後のアプリケーションに含まれる WEB-INF/classes/application.properties に直接クライアントID、クライアントシークレットを指定すると安全性を高めることができます。

テストアカウントを用意する

Wagby 側では事前に AAD 認証に用いるアカウントと同じユーザーIDを持つアカウントを作成しておく必要があります。詳細は Google を使った場合の説明を参考にしてください。

動作確認

これらの設定後にビルドし、Wagby アプリケーションにアクセスします。Wagby のログオン画面に変わって Microsoft の認証画面が表示されるようになります。

その他のOPを利用する

Keycloakをはじめとする、ここに記載していない認証サーバでも、Spring Security が対応している製品であれば接続できることがあります。本ページに記載の内容ならびに Spring Security の情報をガイドにお試しください。

(430)

サポートポリシー

- OpenID Connect に関する問い合わせ（接続できない、どう設定するか、など）は Premium Support となります。認証サーバ、お客様環境、Wagbyアプリ、それぞれの環境に依存するためです。
- サポートはお客様の「認証サービス運用担当者」とやりとりします。これは認証サーバ製品に関する知識を有することが前提です。認証サービスの運用は、お客様環境の運用ポリシー、セキュリティポリシーと関わるためです。

(430) Spring Security 5 以上となります。詳細はご利用の Wagby が同梱しているライブラリのバージョンをご確認ください。

・　本ページに記載の内容は、接続のための最低限の事柄を記載したものです。さらに追加設定が必要な環境もあるかも知れません。あらかじめご了承ください。

デフォルトログオン

概要

デフォルトログオンとは、ログオン画面認証をスキップできる特別なアカウントを利用してシステムを操作する機能です。次の用途を想定しています。

特定のデータを一般に公開する

ある特定のデータを公開する場合、閲覧専用のアカウントを用意します。このアカウントはログオン認証なしでデータの検索や閲覧を行うことができるようになります。

申込受付システムを用意する

セミナーの申込受付を行える、登録専用のアカウントを用意します。このアカウントはログオン認証なしでデータの登録（申込）を行うことができるようになります。

本機能は認証をスキップするのではなく、ログオン認証画面をスキップしてシステムにアクセスした場合、予め設定されたアカウントとパスワードでログオン処理を行なわせるものです。

定義方法

「環境 > アプリケーション > デフォルトログオン」を設定します。

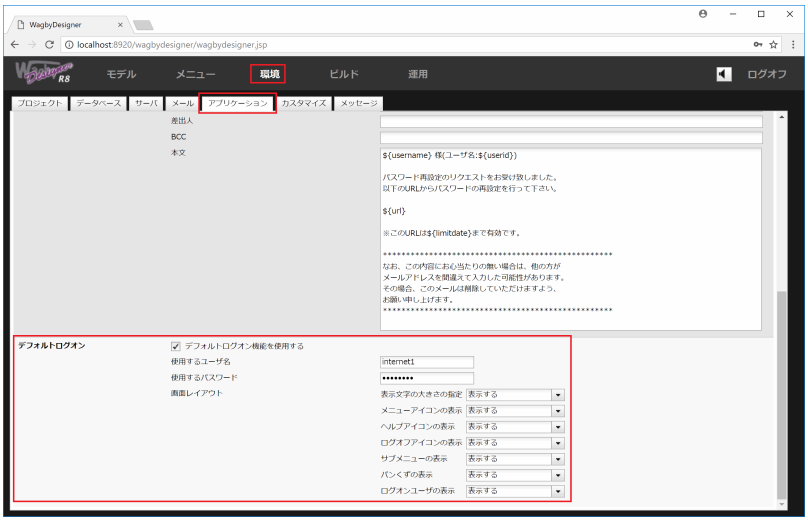


図　デフォルトログオンの設定

・　「デフォルトログオン機能を使用する」を有効にします。
・　「使用するユーザ名」を "internet" ではじまるアカウントします。例えば "internet1" などです。
「使用するパスワード」にパスワードを設定します。

「使用するユーザ名」はデフォルトログオン用であることを意味する "internetXX" としてください。これはインターネットから無記名でシステムを操作するアカウント、という意図です。

デフォルトログオンアカウントを登録する

ビルド後の　Web　アプリケーションに管理者の権限でログオンし、上記で用意したアカウントを作成します。

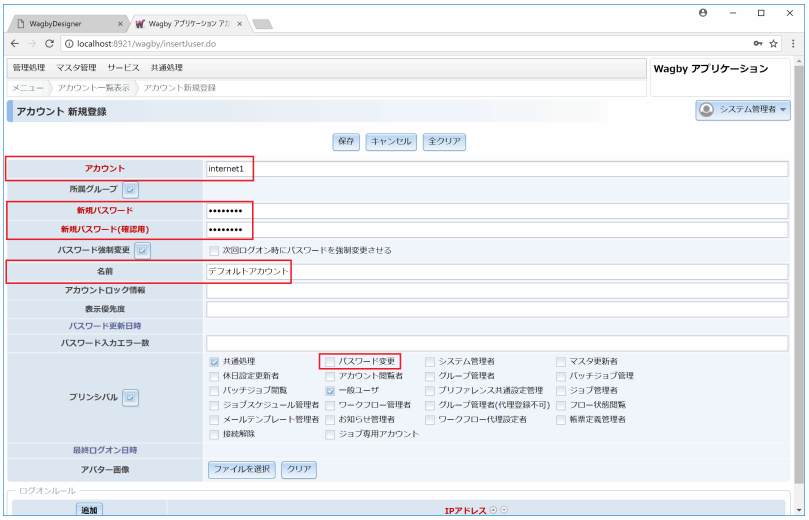


図 デフォルトログオンアカウントの登録

デフォルトログオン用アカウントがパスワードを変更できないよう、プリンシパルで「パスワード変更」の設定は外しておくことを推奨します。

例

メニュー画面を表示する

ログオン処理がスキップすることを確認します。一度ログオフする、または別の Web ブラウザを立ち上げ、アドレスバーに次のように入力します。

```
http://localhost:8921/wagby/mainMenu.do
```

すぐにメニューが表示されます。

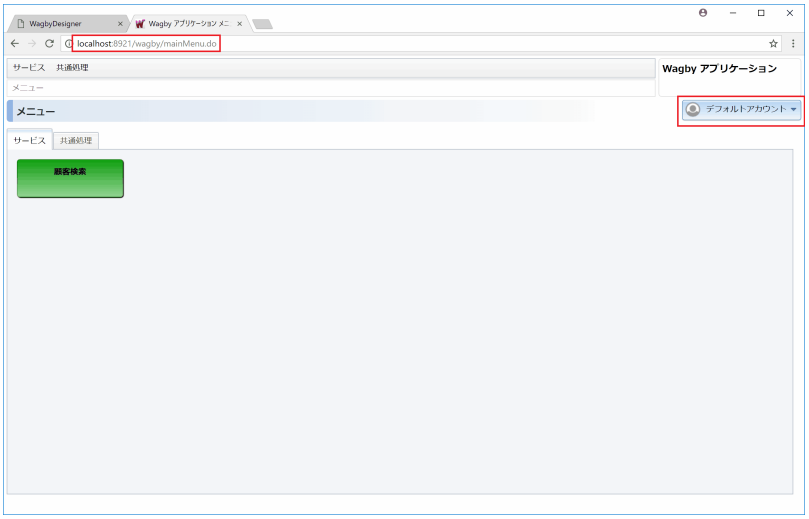


図 ログオン画面をスキップしてメニュー画面を開く

実際には、作成したアカウントで自動ログオンされています。ログファイル（logs/system.log）には次のように記録されます。

```
2018-04-xx 17:00:00 [INFO jp.jasminesoft.jfc.LogonStatusManager addUserName] (admin@localhost) JFC-90000:アカウント internet1 がセッションに追加されました。現在の同時接続数は 1 で、合計セッション数は 1 です。
```

```
2018-04-xx 17:00:00 [INFO jp.jasminesoft.jfc.service.LogonService logon] (admin@localhost)
 JFC-00001:internet1 がログオンしました。利用しているロケール情報は ja です。
```

業務画面を表示する

モデル customer の登録画面を開く例を紹介します。アドレスバーに次のように入力します。

```
http://localhost:8921/wagby/insertCustomer.do?action_New=
```

すぐに顧客登録画面が表示されます。

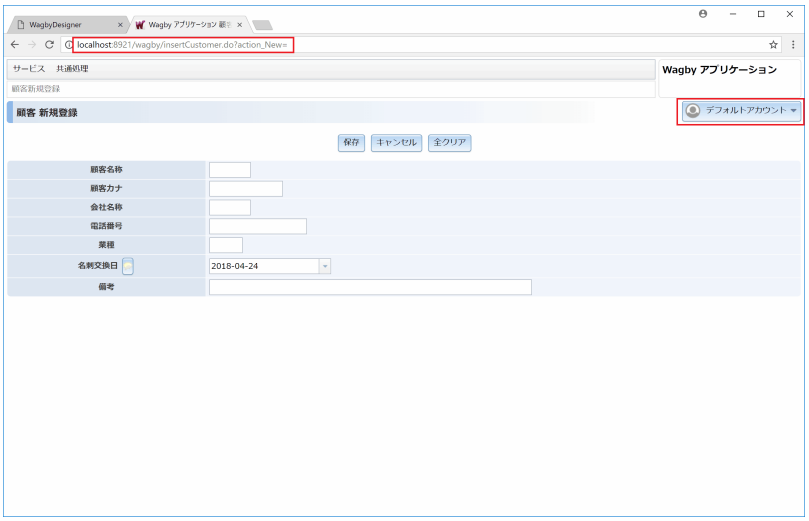


図 ログオン画面をスキップして顧客登録画面を開く

例2 パスワードなしでの利用に限定する

デフォルトログオンを利用して「ラインセンス上の同時接続数の制限を超えたログオンを許容したい」ではなく、単純にログオン画面を通さずに(パスワード入力なしで)システムを利用させることを目的とする場合があります。

この場合は「使用するユーザ名」を　"internet"　ではじまるアカウント以外とします。たとえば"anonymous"とするとよいでしょう。

(431)

運用中にアカウント名、パスワードを変更する

運用後にデフォルトログオンのユーザ名、パスワードを変更する場合は次のようにします。

1. アカウント管理画面で変更します。
2. WagbyDesingerのデフォルトログオン設定も上記に合わせて修正し、再ビルドを行います。

application.propertiesを直接、編集する

設定ファイル application.properties を変更し、Tomcat を再起動することで反映させることもできます。この場合は再ビルドが不要です。

wagbyapp¥webapps¥$(APPNAME)¥WEB-INF¥classes¥application.properties　内の「ignore_logoncheckPassword」「ignore_logoncheckUsername」の値をそれぞれ変更します。

R8.4以降

(431) "anonymous"はデフォルトログオン扱いとならないため、"同一ログオンアカウントでシステムにログオンできる最大数"が適用されます。このためパスワード入力なしでの操作を可能とし、かつ、同一ログオンアカウントでシステムにログオンできる最大数を適用した運用が可能になります。

```
server.servlet.context-parameters.ignore_logoncheckPassword=xxxxxxxx
server.servlet.context-parameters.ignore_logoncheckUsername=internet1
```

R8.3.xまで

```
server.context-parameters.ignore_logoncheckPassword=xxxxxxxx
server.context-parameters.ignore_logoncheckUsername=internet1
```

この方法は再度、ビルドすると application.properties の値が元に戻ります。ビルドせずに動作を確認したいという場合にお試しください。

レイアウト

デフォルトログオンアカウントでシステムを利用する場合、図の設定欄に用意された部品を非表示にすることができます。

- 表示文字の大きさの指定
- メニューアイコン
- ヘルプアイコン
- ログオフアイコン
- サブメニュー
- パンくず
- ログオンユーザ表示

Active Directory と併用する場合

デフォルトログオン用のアカウントは Active Directory 側にも登録してください。

デフォルトログオンを回避するパラメータ (8.4.2)

ブラウザから直接、アドレスバーに https://.../showCustomer.do?id=1000 などと入力したとき、デフォルトログオンが有効な環境では自動的にデフォルトログオンでアクセスされます。

しかし、このモデルはデフォルトログオンを適用したくないという要件があったとき、リクエストパラメータに__jfc_ignore_defaultLogon=trueを付与することで回避できます。上の例では https://.../showCustomer.do?id=1000&__jfc_ignore_defaultLogon=true となります。

仕様・制約

- デフォルトログオンで利用するアカウントにはパスワード有効日数が適用されません。
- デフォルトログオンで利用するアカウントには ″同一ログオンアカウントでシステムにログオンできる最大数″ や ″自動ログオフを有効にする″ 機能は適用されません。これは不特定多数の利用者を想定しているためです。
- 本機能はマルチセッションと併用することはできません。
- REST API ではデフォルトログオン機能は動作しません。（デフォルトログオンを利用しない REST アクセスは利用できます。）

マルチセッション

概要

マルチセッションとは、1つのブラウザでタブを開いたとき、それぞれのタブで Wagby アプリケーションにログオンする仕組みをいいます。ログオン認証は、タブごとに独立して管理されます。

動作例

アカウント user1 でログオンしている状態で、ギアアイコンから「新規ログオン」を選びます。

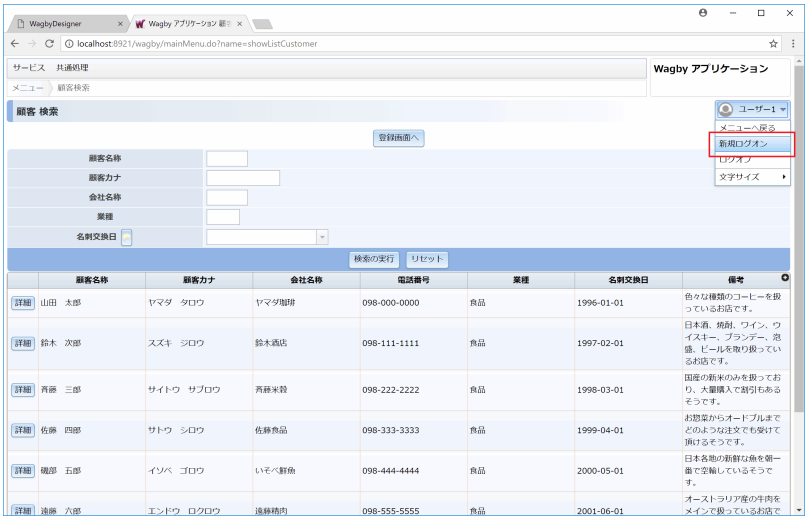


図 新規ログオンを行う

新しいタブが開きます。同じアカウント user1 で自動ログオンされ、同じ画面が開きます。

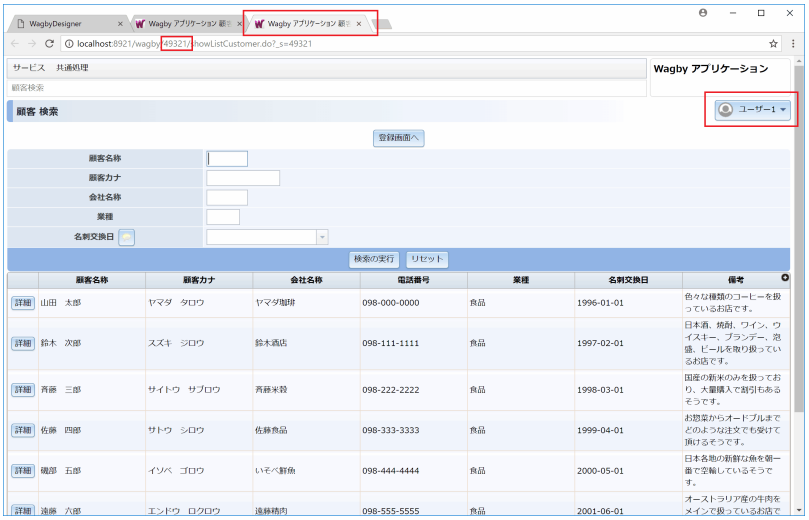


図 新しいタブが開く。

マルチセッションの仕組み

図のアドレスバーに注目します。URLの一部に5桁のランダムな数字が含まれています。これがマルチセッションを実現するための識別子となっています。（新規ログオンを行うたびに、新しい数字が生成されます。）

マルチセッションを利用すると、ユーザごとに内部で割り当てられる「セッション　(*)」が、この識別子で管理されるようになります。このため、複数のタブであっても異なるセッションとして区別できるようになっています。

(432)

定義方法

設定方法

1. "環境 > アプリケーション > HTTPセッション"を設定します。
2. マルチセッション利用時は「同一ユーザーの同時接続数の上限数」を無制限としてください。

(432) ここでいうセッションは、Webアプリケーションが管理する HttpSession オブジェクトを指します。

Redisのインストールと起動

本番運用では Redis を用います。インストールと起動方法...

応用 スクリプトによる画面遷移

独自ボタン押下時、マルチセッションを使ってログオンし、任意の画面に遷移するスクリプトを作成することができます。[詳細...]

仕様・制約

- 「新規ログオン」を行ったときの画面が登録画面または更新画面だった場合、新しいタブの画面はメニュー画面が表示されます。
- 「新規ログオン」を行ったときの画面が一覧表示画面で、このモデルが検索画面と一覧表示画面を別としていた場合、新しいタブの画面は検索画面が表示されます。
- 複数のタブを開いたとき、それぞれのタブで同時認証接続数を消費します。マルチセッションで新しく開いたタブは不要になったとき、手動でログオフ処理を行うことですぐにセッションを解放するため、サーバの負荷軽減に貢献できます。ブラウザのタブを閉じる処理ではサーバのセッションが残り続けます。（これはセッションタイムアウトによって解放されます。）
- 格納方式に Spring Session HashMap は本番運用で利用することはできません。本番運用では Spring Session Redis を指定し、別途 Redis をインストールしてください。 Spring Session HashMap はセッションタイムアウト機能が動作しないという制約があります。そのためセッションタイムアウトが発生してもログオンユーザは残り続けます。

ワークフロー

モデルにワークフロー定義を行う

年休申請モデルの定義

ここでは図に示す「年休申請」モデルを用意します。

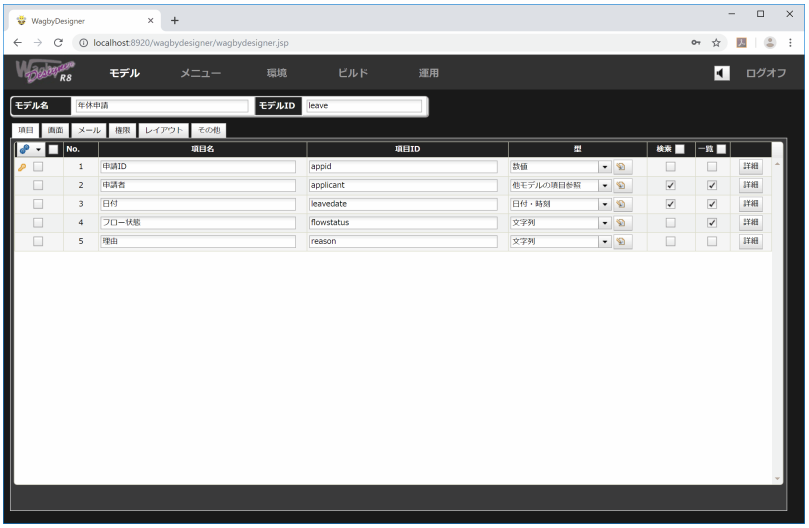


図 年休申請モデル

項目名	項目ID	型	検索	一覧
申請ID	appid	数値		

申請者	applicant	他モデルの項目参照	○	○
日付	leavedate	日付・時刻	○	○
フロー状態	flowstatus	文字列（※計算式）		○
理由	reason	文字列		

申請者

Wagbyが標準で提供するログオンアカウントモデル(juser)を参照します。

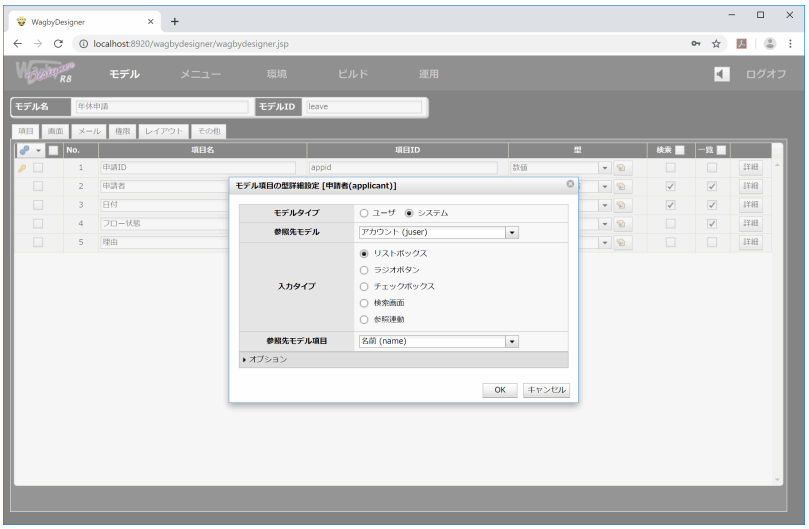


図 juserをリストボックスで参照する

また、登録画面表示時の初期値を「ログオンID」とします。　これによってリストボックスの初期値として、現在ログオンしているアカウントが選択されます。

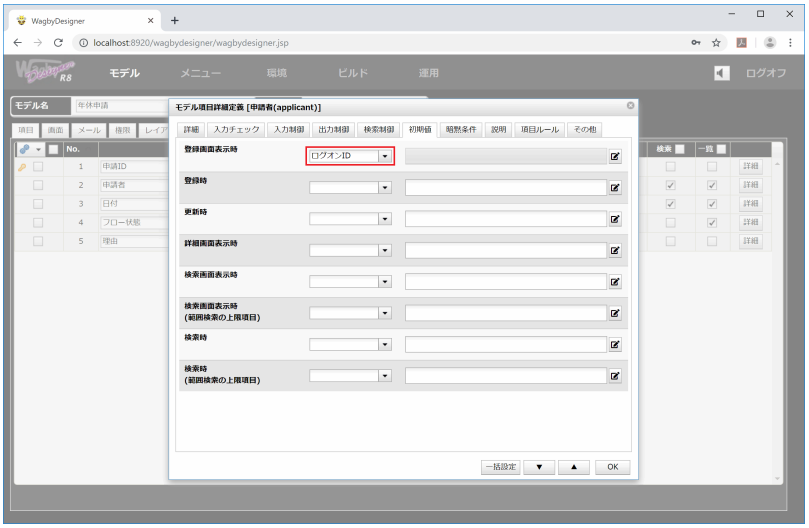


図 登録画面表示時の初期値を設定する

日付

登録画面表示時に今日の日付を自動設定するため、関数 TODAY を用います。

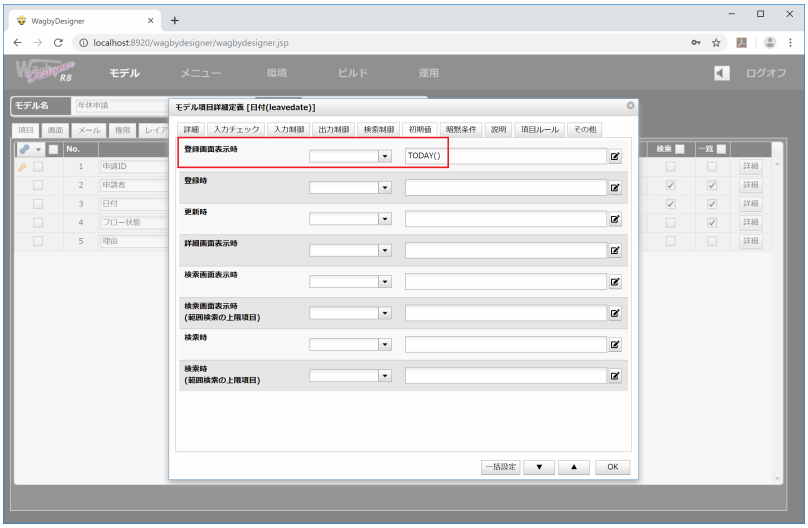


図 関数TODAYを用いる

フロー状態

関数 FLOWSTATUS を用いることで、フローの状態（申請、承認など）を知ることができます。 書式の詳細な定義方法は"関数>ワークフロー>FLOWSTATUS"をお読み下さい。

```
FLOWSTATUS("leave", TOSTR(${appid}))
```

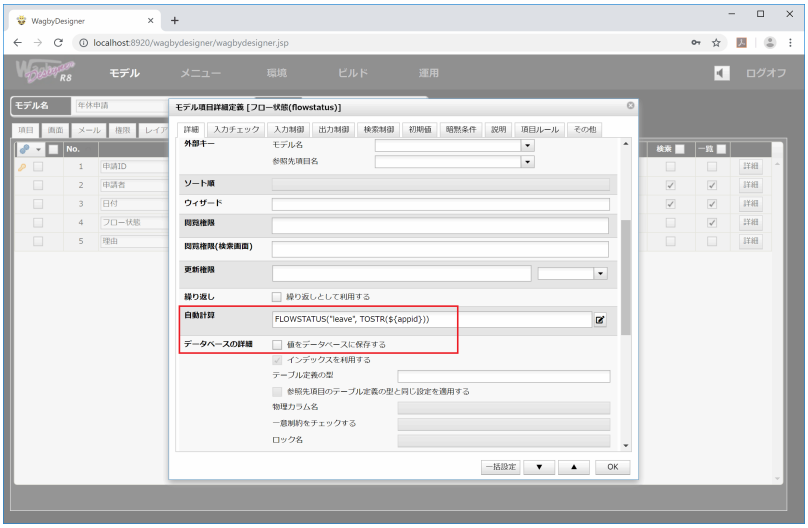


図 関数FLOWSTATUSを用いる

FLOWSTATUS関数を適用した項目はデータベースに保存しないようにしてください。（保存する、を設定しても内部では無視されるようになっています。）

FLOWSTATUS関数を適用した項目を検索条件に指定する場合、特別な設定が必要です。詳細は"モデルのワークフロー状態で検索する"をお読みください。

理由

入力項目の種類をテキストエリアとします。

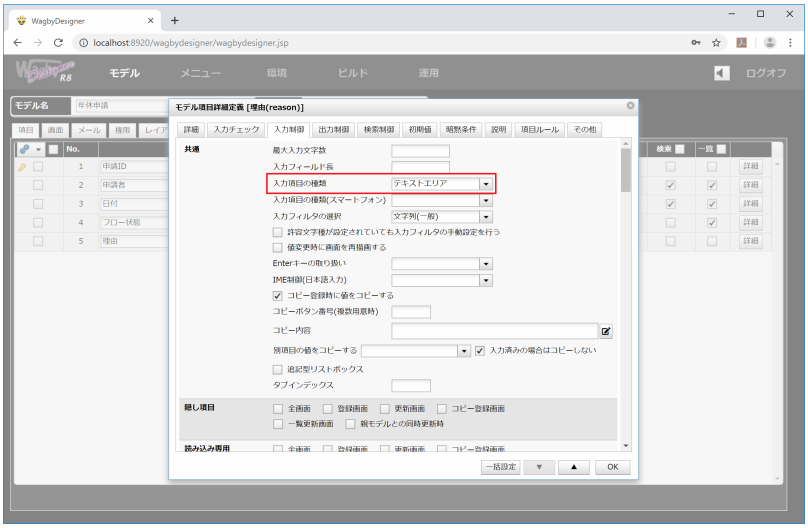


図 入力項目の種類を設定する

ワークフローの適用

このモデルにワークフローを適用します。「画面」タブ内の「その他」に用意された「ワークフロー」設定欄で、「ワークフローを有効にする」のチェックを付与します。

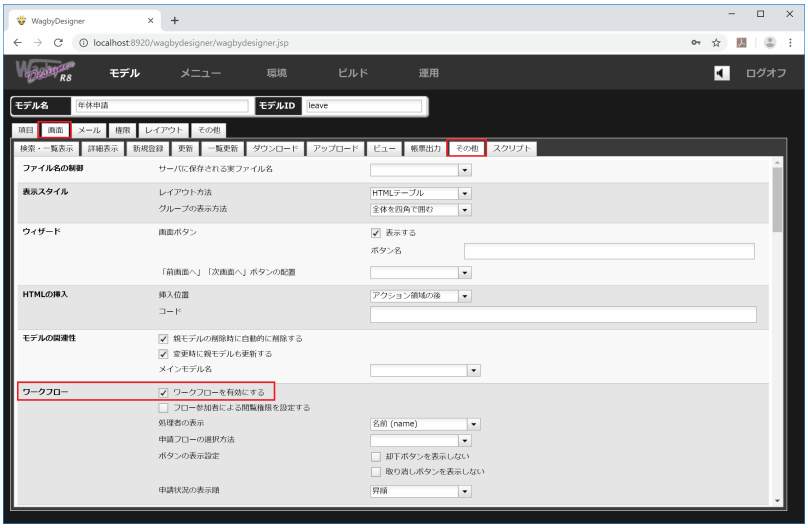


図 ワークフローを有効にする

設定が完了したら、ビルドします。

グループの準備

ビルドしたWebアプリケーションを起動します。システム管理者でログオンし、図に示す3つのグループ（総務部、営業部、技術開発部）を用意します。

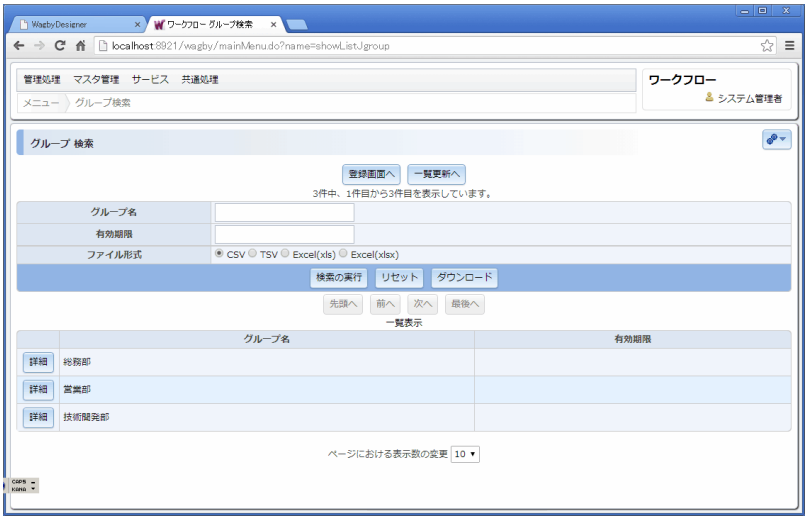


図 デモ用のグループを準備する

アカウントの準備

以下のテストはトライアルキットではお試しいただけません。アカウントの追加処理を行うためです。評価用ライセンスまたは開発キットをご利用ください。（評価用ライセンスは無料で提供しています。Wagby 販売パートナーにお問い合わせください。）

技術開発部グループに所属する3つのアカウント（arakaki,miyagi,murata）を用意しておきます。 プリンシパルは ″共通処理″ ″パスワード変更″ ″一般ユーザ″ に加えて ″フロー状態閲覧″ を加えます。 これらをデモデータとして次ページ以降で説明します。

図 デモ用のアカウントを準備する

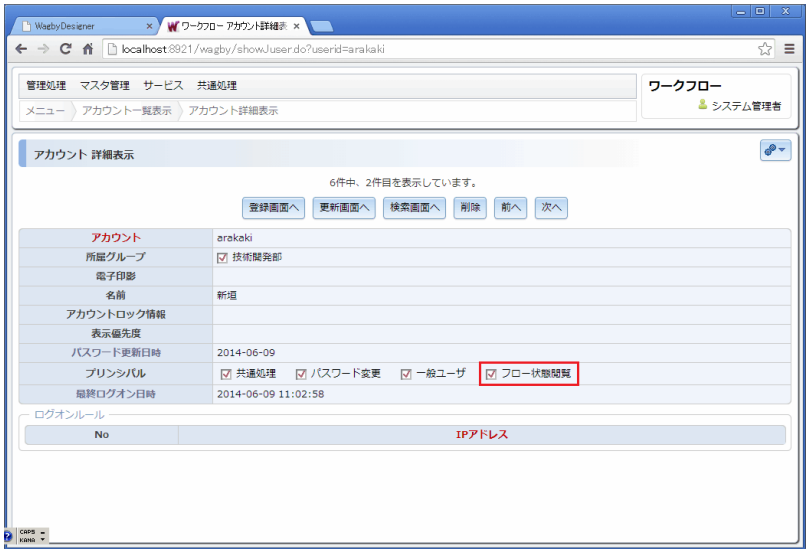


図 デモ用アカウントのプリンシパル

フローパターンの設定を行う

標準で提供されるフローパターン

フローパターンはワークフローの処理の流れを規定します。

Wagby は標準で 11 種類のフローを提供しています。これ以外のフローを独自に設計することもできます。

- 順次・順次フロー （申請→決裁）
- 順次・順次・順次フロー（申請→承認→決裁）
- 順次・合議・順次フロー（申請→合議承認→決裁）
- 合議・順次フロー （複数申請→決裁）
- 合議・順次・順次フロー（複数申請→承認→決裁）
- 順次・合議フロー（申請→合議決裁）
- 合議・合議フロー（複数申請→合議決裁）
- 合議・合議・合議フロー（複数申請→合議承認→合議決裁）
- 合議・合議・順次フロー（複数申請→合議承認→決裁）
- 合議・順次・合議フロー（複数申請→承認→合議決裁）
- 順次・順次・合議フロー（申請→承認→合議決裁）

「順次」と「合議」は次のような意味をもっています。

順次

順番にワークフローを進める場合に指定します。起案者（作成者）から出発し、直属の上長を経由して部長の決裁を頂く、といったシナリオでは、すべてが「順次」ノードになります。

合議

指定した人数が承認／申請した場合に次へ進むことができるノードです。例えばグループ内の誰かが承認した場合に直属の上長の決裁を頂く、といったシナリオで用いることができます。

これら 11 種類のフローは初期データとして用意されており、フローパターン画面で確認できます。

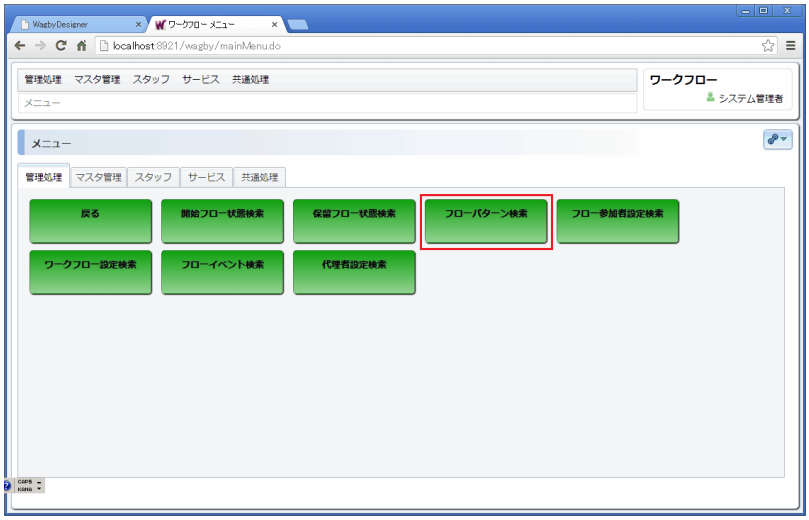


図 管理処理メニューから「フローパターン検索」を選ぶ

図 標準で11パターンが用意されている

独自のフローパターンを設計する

独自のフローパターンを設計する手順を説明します。

これから設計するフローパターンの土台を一つ選びます。　ここでは図の「フローパターン一覧」画面から、3ノードから構成される「順次・順次・順次フロー」を選び、これを拡張して4ノードの「順次・順次・順次・順次フロー」を作成します。

図で、「順次・順次・順次フロー」のコピー登録を行います。図の画面となります。

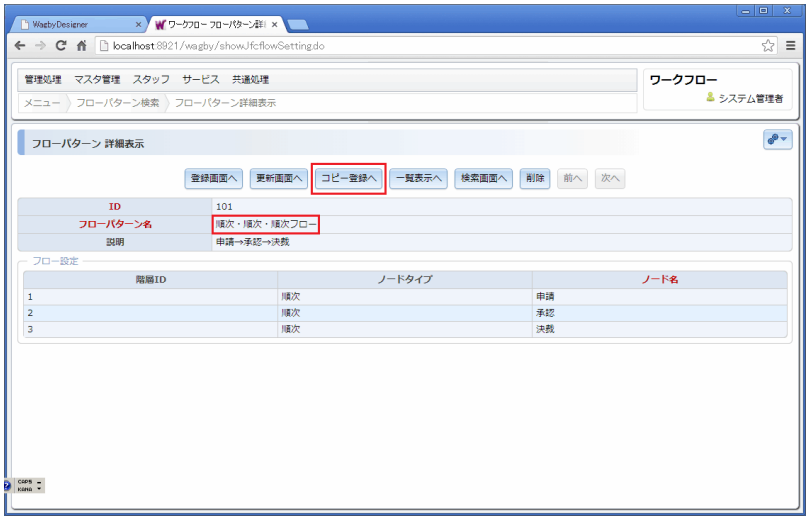


図 コピー登録を行う

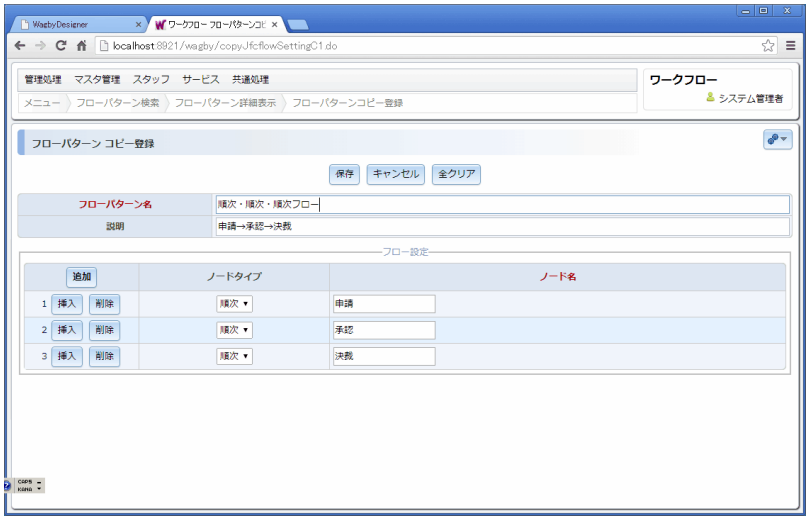


図 コピー登録画面

フローパターン名、説明を修正します。また、ノードを一つ挿入します。

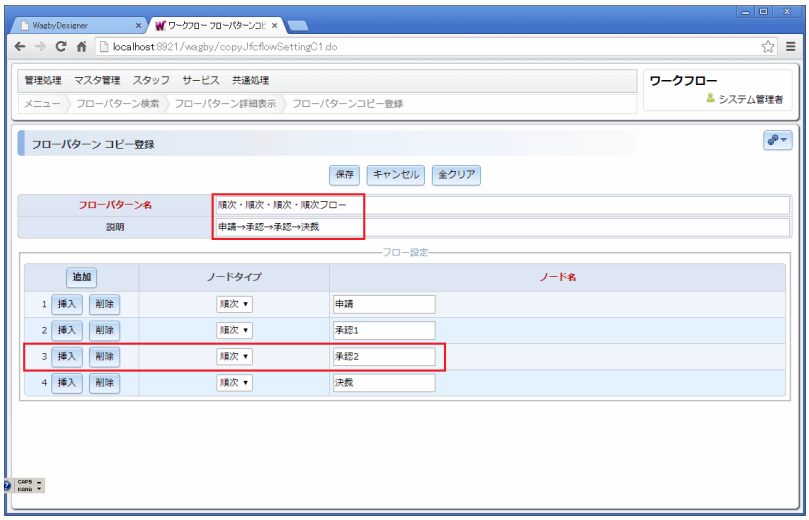
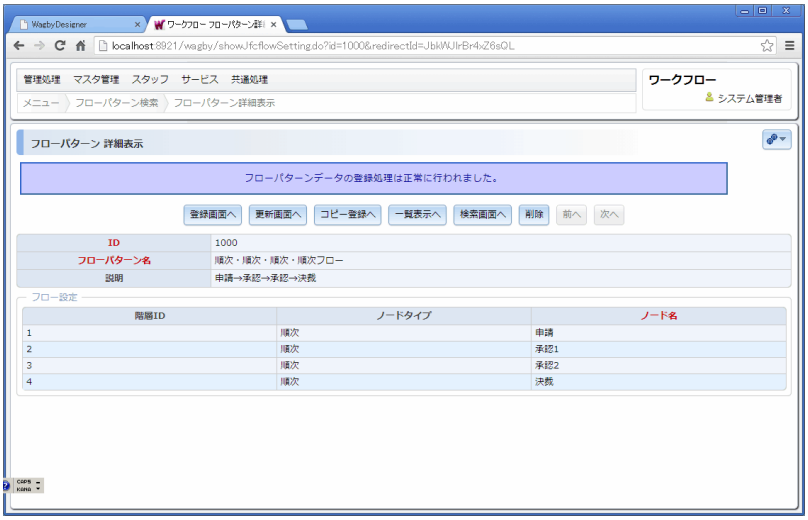


図 フローパターンを修正する

新しい4ノードの順次フローパターンを作成しました。



合議ノードを拡張する

合議ノードでは、申請及び承認時に「〇名以上の申請／承認が必要」と人数を指定することができます。

標準で用意されている「順次・合議・順次」フローパターンを確認します。　ここでは　″承認1″、″承認2″という2つのノードのうち、1名以上が承認した場合に承認となる、と設定されています。

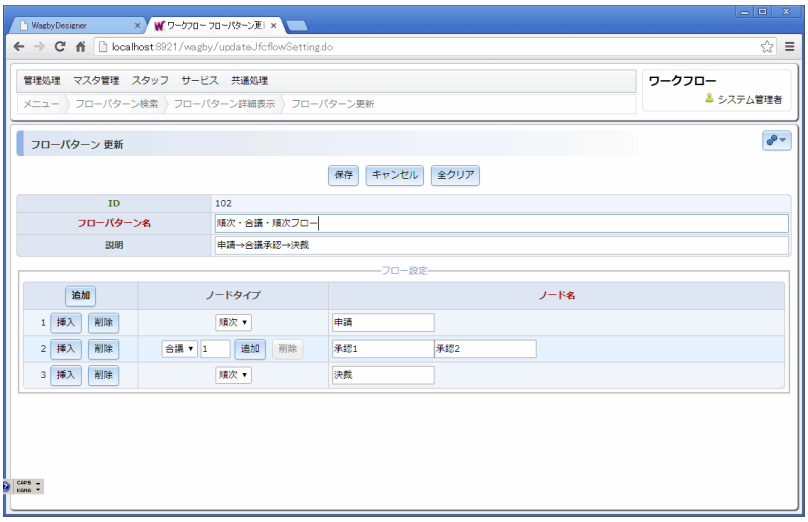


図 標準で用意されている「順次・合議・順次」フローパターン

合議の人数を変更する場合は、数値を変更します。　ノードを追加（削除）する場合は、「追加」（「削除」）のボタンを押下します。 ここでは人数を「2」とし、ノード「承認3」を追加しました。すなわち「承認ノードは 3 つ、2 名以上の承認があれば承認」となります。

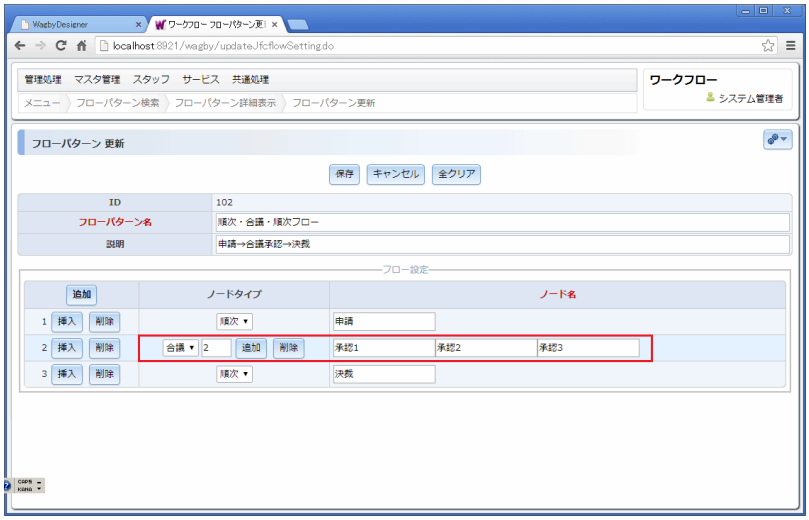


図 合議ノードの変更

フロー定義ファイル

独自のフローパターンを設計すると、以下のフォルダに作成したフロー定義ファイル（XML形式）が生成されます。

```
wagbyapp¥webapps¥$(APPNAME)¥WEB-INF¥workflow-definitions
```

アプリケーションの入れ替え（再ビルド）を行った場合も、フローパターンモデルをインポートするタイミングで上記フォルダ内に設定ファイルが再作成されます。

トラブルシューティング：フロー定義ファイルが読み込めない !-- N20160628_175921 -->

上記フォルダにファイルが存在しているが、実行するとフロー定義ファイルの読み込みに失敗する場合、次のことを確認してください。

起動フォルダ

startup.bat（または startup.sh）を実行するフォルダは wagbyapp/bin である必要があります。 例えば wagbyapp フォルダで起動してしまうと、ワークフロー定義ファイルの取得に失敗します。

フローの参加者を設定する

定義方法

（前ページで説明した）フローパターンを使って、さまざまなワークフローを起こすことができます。Wagbyのワークフローとは、フローパターンに参加者を紐付けたものです。

管理処理メニューから「フロー参加者設定検索」を選びます。

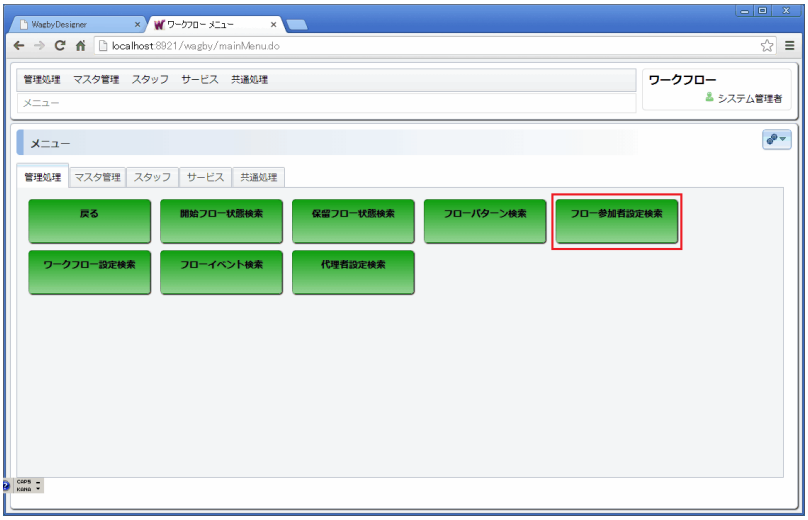


図 フロー参加者設定検索

フロー参加者設定の登録画面を開きます。

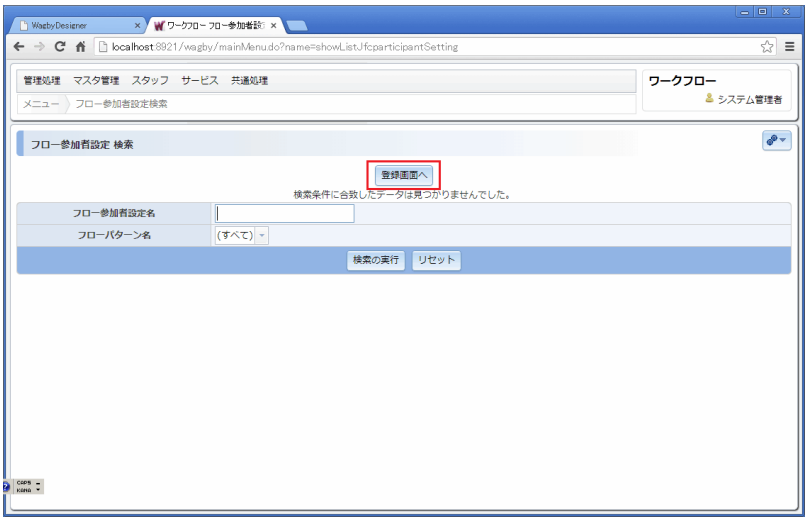


図 フロー参加者設定の登録画面へ

フロー参加者設定名にこのワークフローの名前を記入します。 また、説明欄にワークフローの説明を記入します。

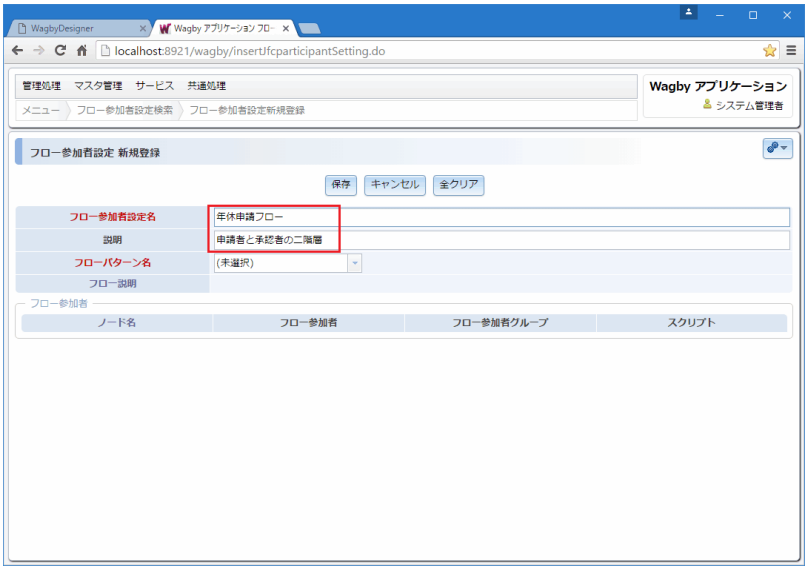


図 ワークフローの名前を決める

フローパターンを決定します。

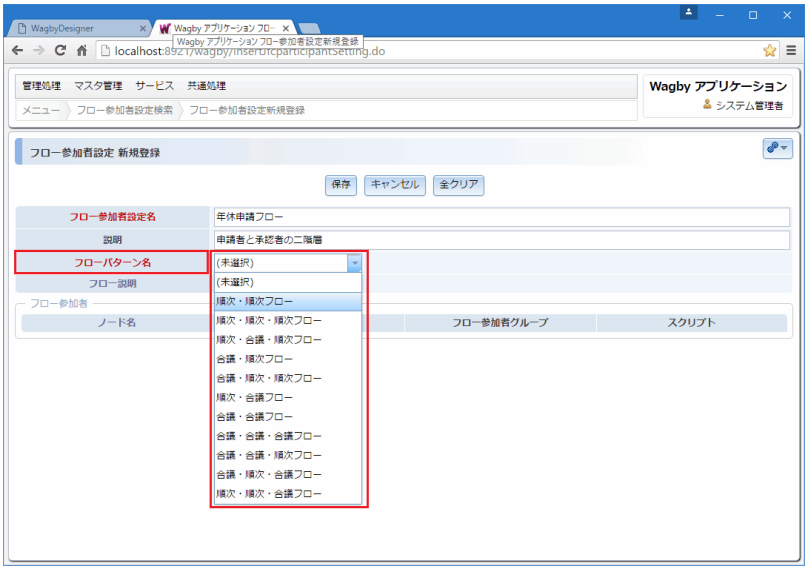


図 フローパターンを決定する

1つのワークフローは、1つのフローパターンと紐づきます。想定されるフローパターンに基づいたワークフローを設計してください。

フローパターンを決定すると、自動的に「フロー参加者」の入力欄が用意されます。

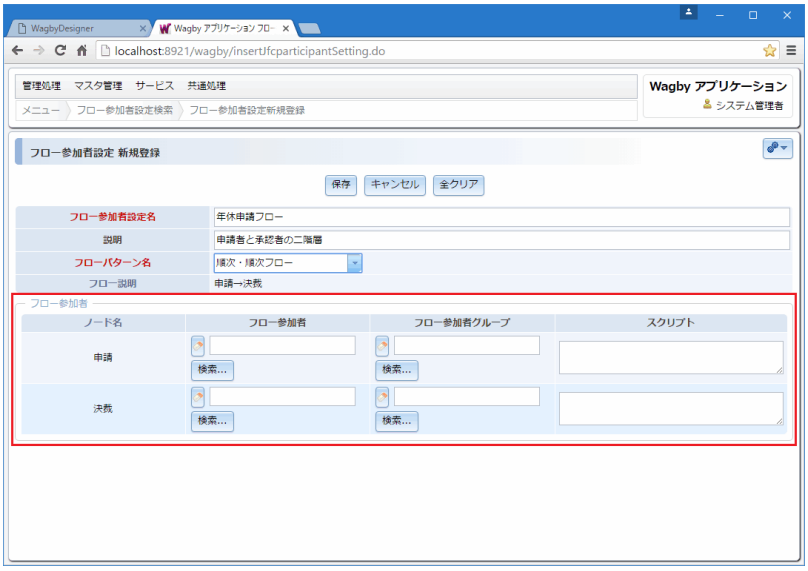


図 フロー参加者の入力欄

ここでは申請ノードとして「技術開発部」グループを指定します。これによってグループに所属するアカウント全員が申請者になることができます。 決裁者は、部の上長となるアカウントを指定しました。

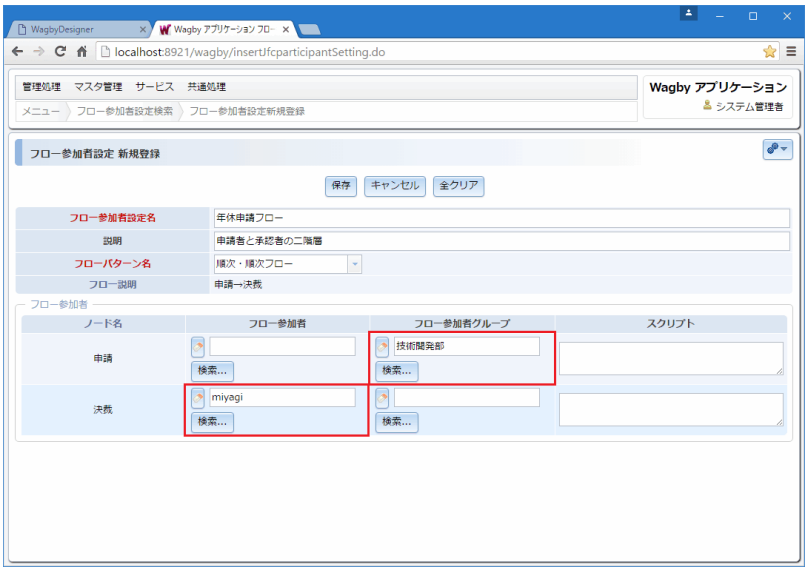


図 フロー参加者の決定

フローを開始するノードは、グループ名を指定することが一般的です。これによってこのグループに所属する人は誰でも、フローを開始することができるようになります。事前にグループを用意しておくとよいでしょう。

設定上の注意

同じアカウント名/グループ名を循環させるような設定はできません。

例：アカウント yamada で承認させたあと、代理承認として再び yamada の承認待ちとする、というような動作を行わせることはできません。

中規模以上の規模での運用（スクリプトを利用する）

小規模な組織ではフロー参加者としてユーザやグループを固定することができますが、中規模以上の組織で、かつ高い頻度で異動や組織改編がある場合は固定での指定による運用は難しいといえます。

このようなケースではスクリプトを使うことができます。詳細は ″スクリプトによって承認者・決裁者を動的に決定する″ で説明します。

モデルにワークフロー定義を設定する

定義方法

（前ページで用意した）ワークフロー定義を、実際のモデルと紐付けます。 1つのモデルは、複数のワークフロー定義を選択して利用できます。

管理処理メニューから「ワークフロー設定検索」を選びます。

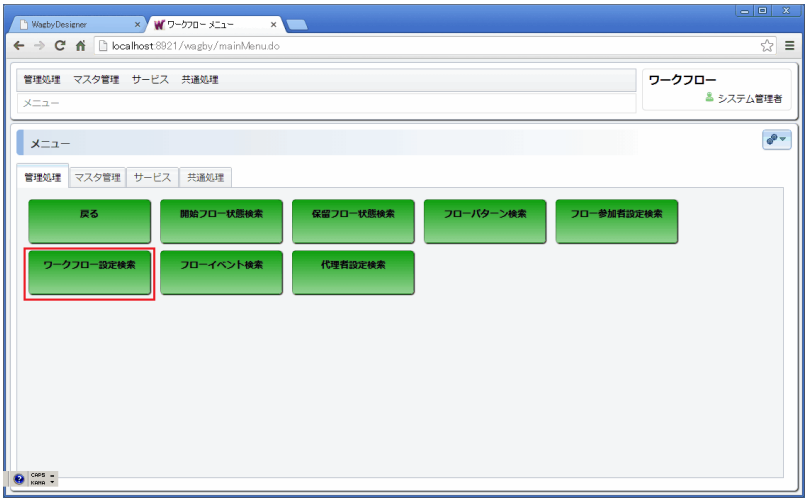
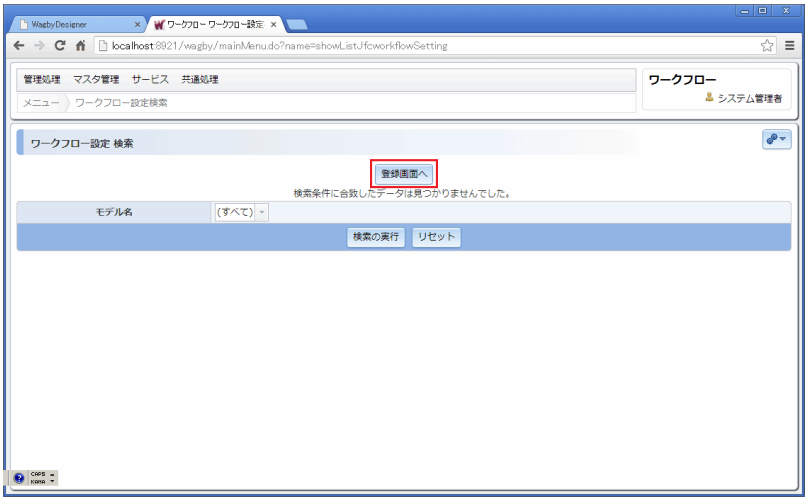


図 ワークフロー設定検索

ワークフロー設定の登録画面を開きます。



「モデル名」は、ワークフローを利用すると設定したモデルから選択します。

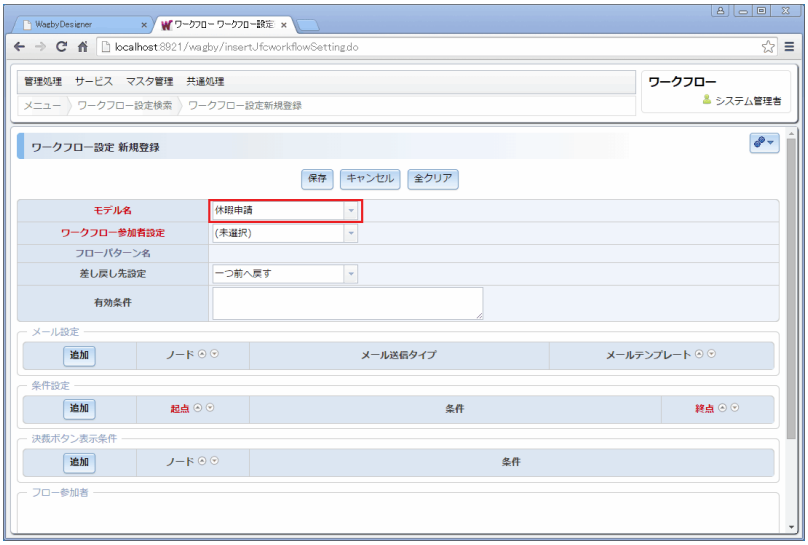


図 モデル名を選択する

「ワークフロー参加者設定」は、フローパターンと参加者を設定済みのワークフロー定義から選択します。

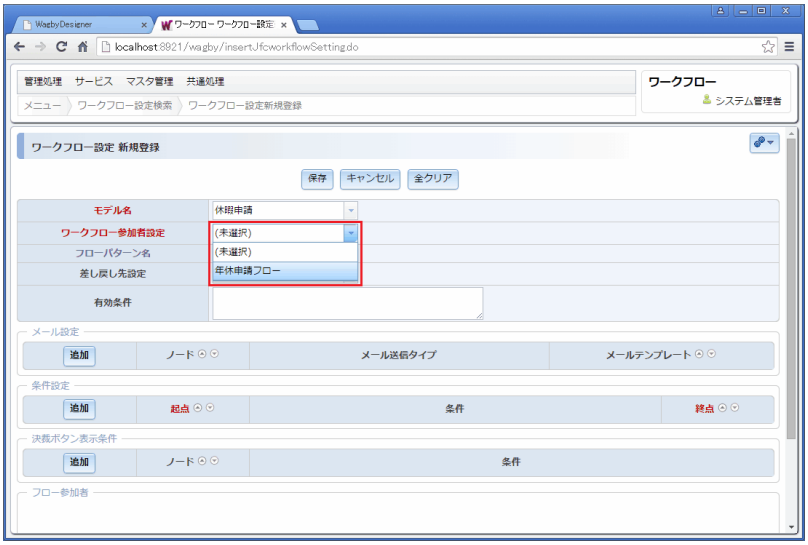


図 ワークフロー参加者設定を選択する

フローパターンと参加者を指定すると、ワークフローの図が表示されます。 これはフローを確認するための図となっており、編集は行えません。

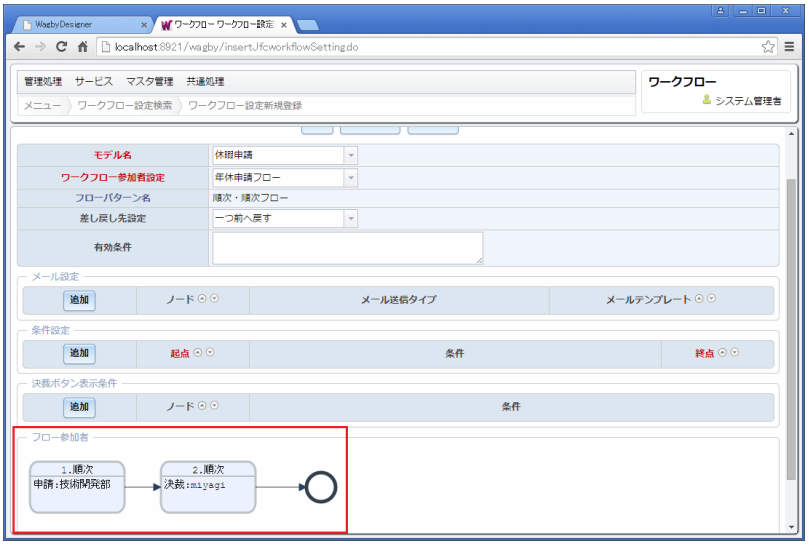


図 ワークフローのダイアグラム表示

差し戻し時の戻し先を設定します。「一つ前へ戻す」または「申請者へ戻す」を選択します。　標準では「一つ前へ戻す」が選択されています。

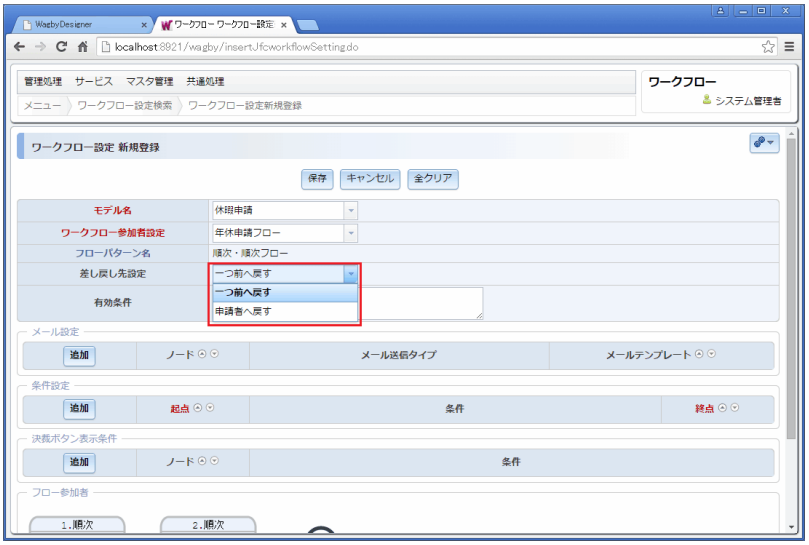


図 差し戻し時の戻し先を設定する

ここまでの設定を保存します。

合議の承認数の確認（8.3.0）

合議の承認数および承認者情報をフロー参加者詳細画面で確認することができます。

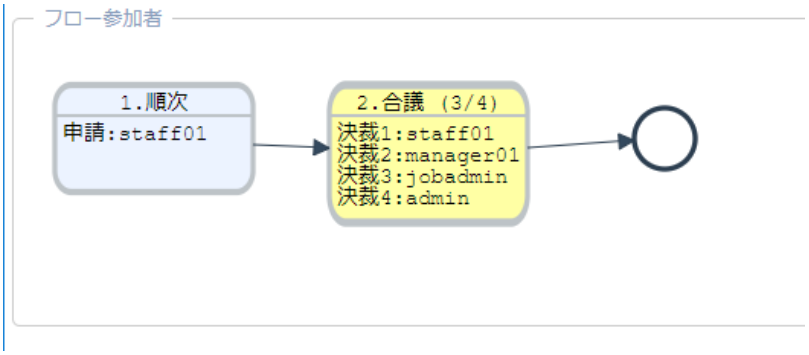


図 合議の承認数と承認者情報の確認

(433)

注意点

モデルに対してワークフロー参加者を設定したあと、ワークフロー参加者に関する情報を変更することはできません。すでに開始されたワークフローと整合性が取れなくなるためです。

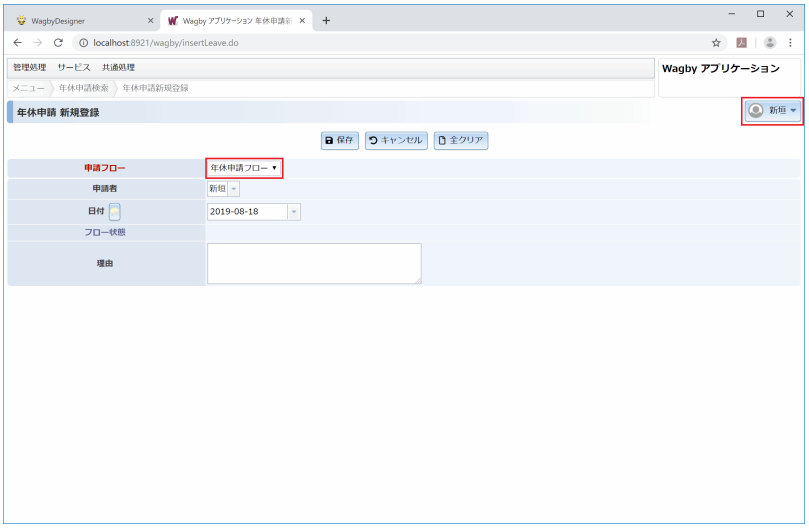
(434)

ワークフローを開始する

実行

ワークフローに対応したモデルのデータを新規登録すると、自動的にワークフローが開始されます。 「休暇申請」モデルに適用した例で説明します。

アカウント ″arakaki″ でログオンし、データを1件登録します。項目「申請フロー」は、ワークフローを用いると定義したモデルの先頭に表示されます。ここには申請可能な申請フローのみ選択肢に用意され、このモデルで使用できるフローを選択することができます。これは「モデルにワークフロー定義を設定する」で登録したフローです。



(433) 合議の承認数ならびに承認者情報は、フロー参加者の登録・更新時にデータベースに保存されます。具体的にはワークフロー設定(jfcworkflow_setting)モデルのフロー参加者コンテナに用意された合議承認数(participants/approval_num)項目です。この値がnullの場合、図の情報は表示されません。この場合は当該フロー参加者データの更新を行なってください。更新を行なったタイミングでこのカラムに値が設定・保存され、必要承認数が詳細画面で閲覧できるようになります。

(434) ワークフローを設定したモデルのデータ A を新規に登録したと想定します。A には「次の承認者」に関する情報がすでに含まれている状態でデータベースに保存されています。ここでワークフローのフロー参加者を変更しても、既存のデータ A には反映されません。

図 休暇申請の新規登録画面

データ登録後、画面下部に「申請状況」という欄が表示され、このデータを申請できるようボタンが用意されます。 まだワークフローは開始していません。この段階ではデータの更新ボタンが表示されています。

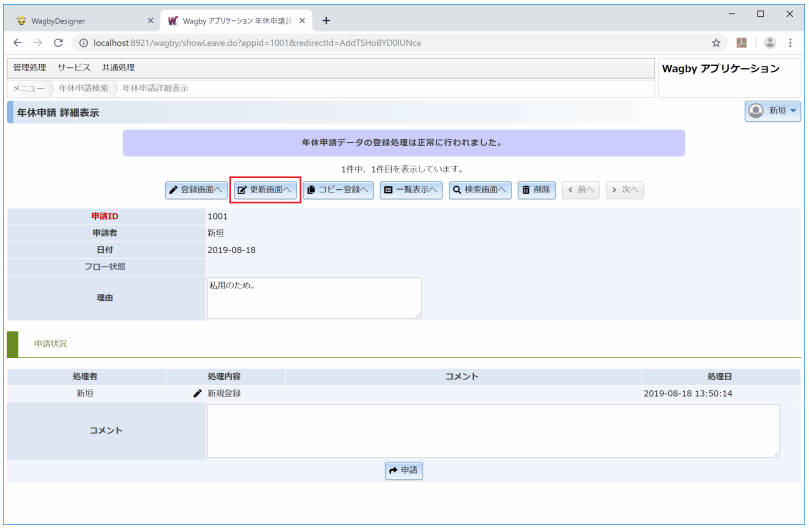


図 休暇申請の詳細画面

画面下部の申請状況欄にコメントを入力し、ワークフローの「申請」ボタンを押下します。

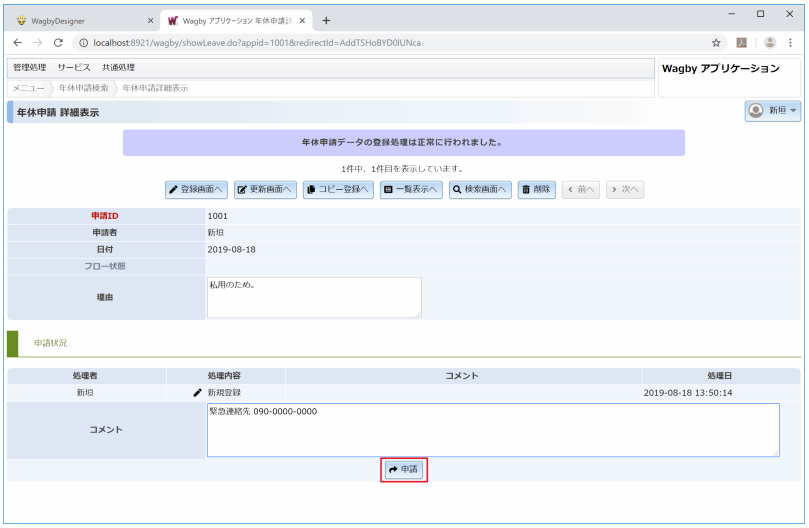


図 ワークフローを開始する(1)

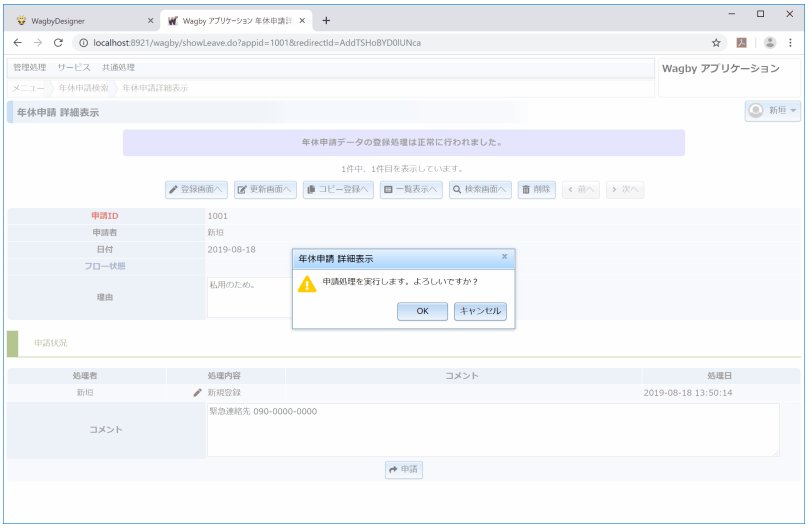


図 ワークフローを開始する(2)

データを登録しても、申請ボタンを押下しない場合はワークフローは開始されません。

「処理内容」が「申請」に変わります。データの更新は行えなくなります。

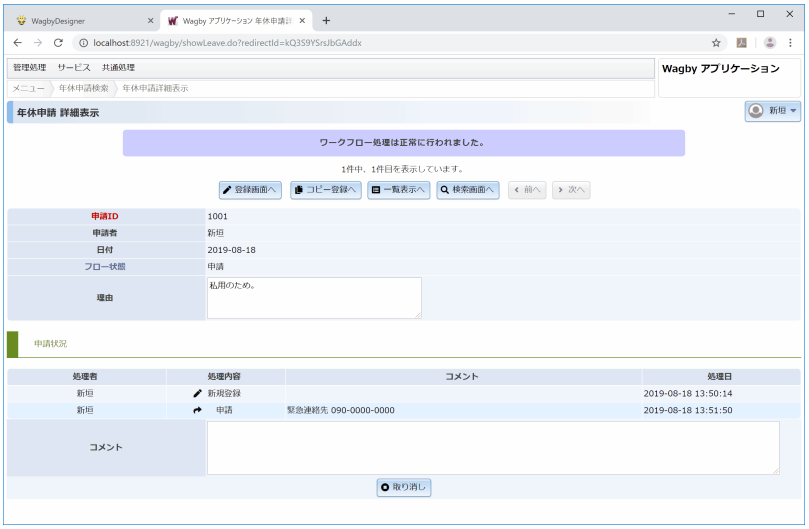


図 申請されたワークフローはデータの更新が行えない

続いて、承認者である"miyagi"でログオンし、図のデータを確認してみます。"miyagi"は承認者なので「承認」「差し戻し」「却下」のボタンが用意されていることが分かります。またデータの更新も可能です。

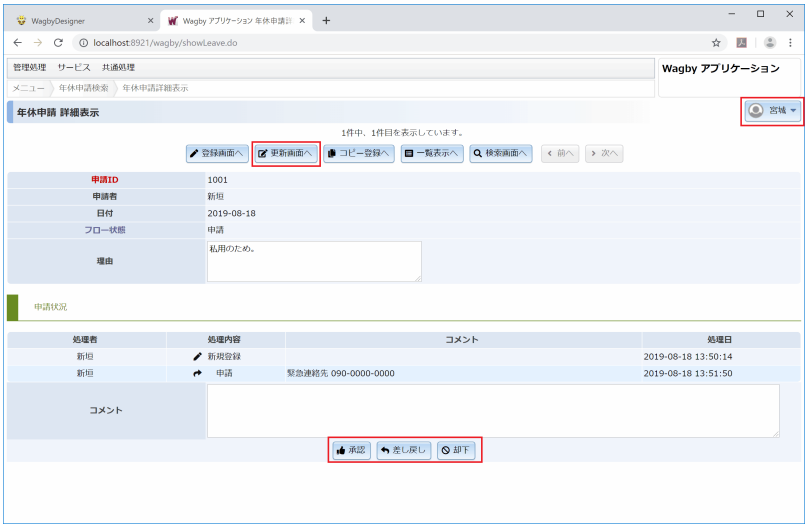


図 アカウントmiyagiでログオンし、データを確認する

コメントを記載し、承認ボタンを押下します。

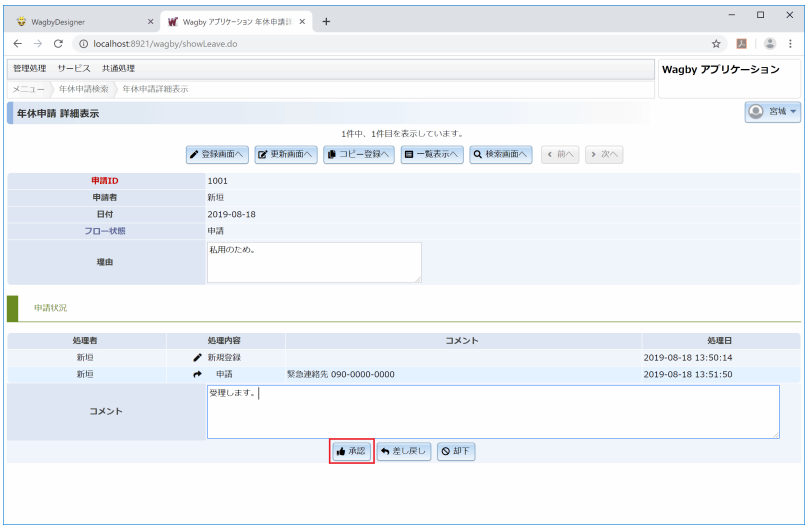


図 申請を承認する(1)

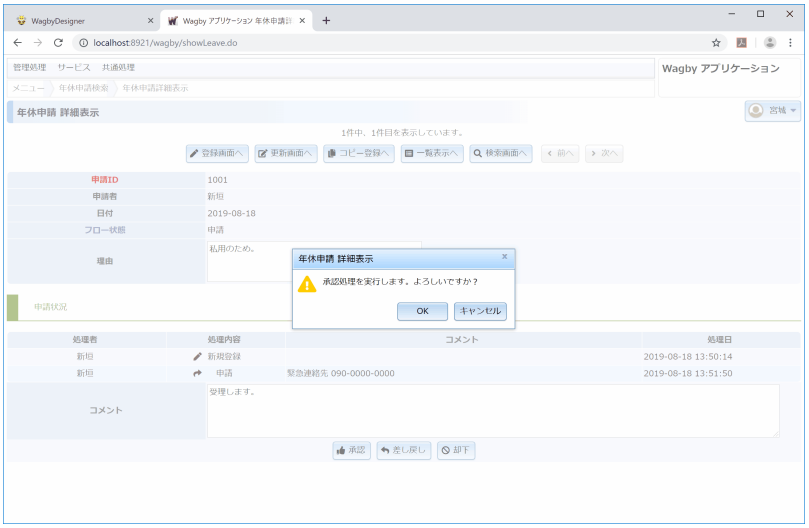


図 申請を承認する(2)

ワークフローに対してアクションを起こす（承認）と、データの更新は行えなくなります。 このワークフローは2ノード（順次→順次）のため、この段階で自動的に決裁となります。

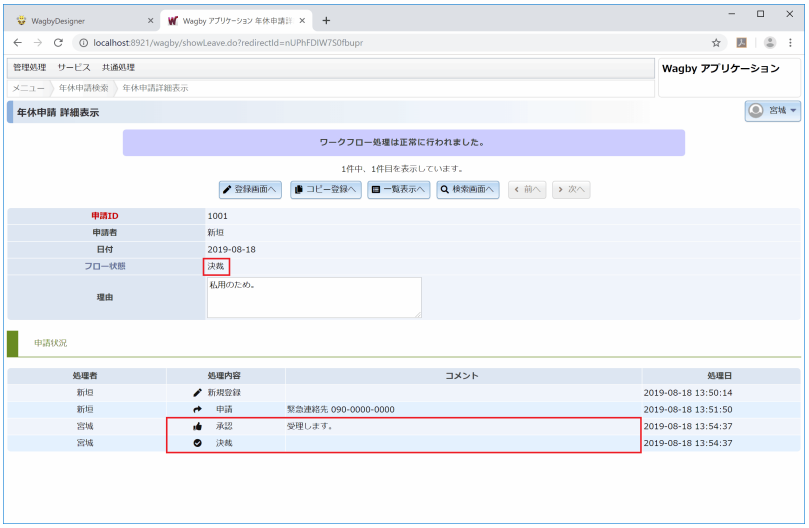


図 ワークフローが決裁された

再びアカウント"arakaki"でログオンします。ワークフローが決裁されていることがわかります。

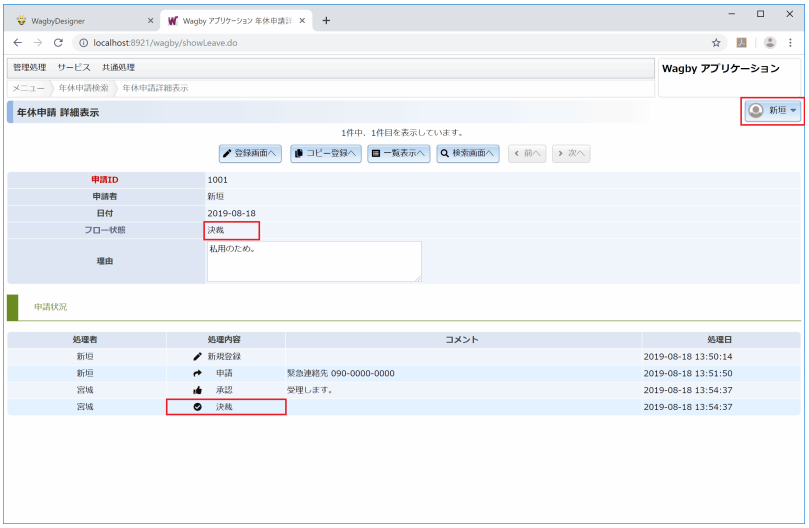


図 アカウントarakakiでログオンし、決裁状態を確認する

ワークフローの状態をFLOWSTATUS関数で取得することができます。図のように一覧表示画面でワークフローの状況を把握できます。

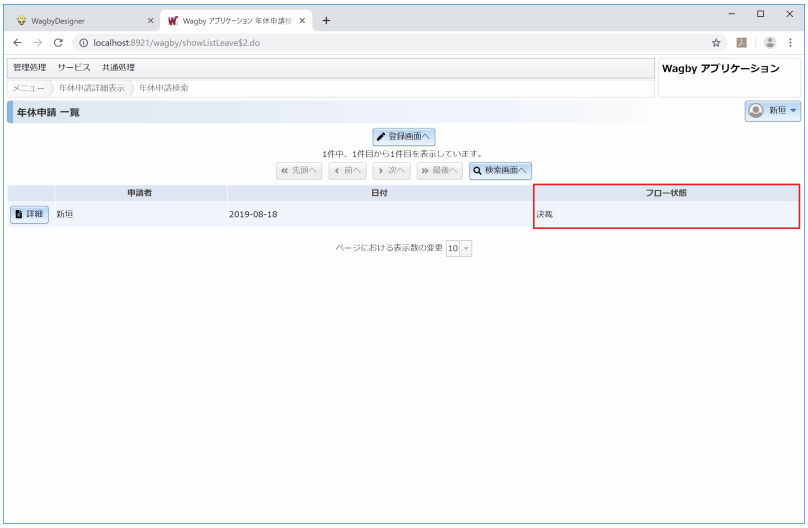


図 一覧表示画面でワークフローの状況を把握する

トラブルシューティング

「申請フロー」の選択肢が空になっている。

図で「申請フロー」の選択肢が空になっている場合、このワークフローの「フロー参加者設定」で、ノード（申請者・担当者・承認者）のフロー参加者またはフロー参加者グループで設定されたユーザー・グループに所属していないアカウントで利用している可能性があります。

ワークフロー定義ファイルの取得に失敗する。

ログ（system.log）に次のように表示された場合は、以下の点をご確認ください。

```
[ERROR jp.jasminesoft.util.VelocityUtils mergeContent] occur ResourceNotFoundException file:s-c.xml
```

- wagbyapp 起動ユーザにワークフロー定義ファイルのフォルダ（WEB-INF/workflow-definitions）へのアクセス権があるか。
- startup.bat（startup.sh）実行時のフォルダが wagbyapp/bin となっているか。(*1)
- 独自のフローパターンを設計した場合、アプリケーションの入れ替え（再ビルド）を行ったときにフローパターンモデルをインポートしているか。

(435)

申請したフローを取り消す

申請後、次の承認者がまだ承認（または差し戻し、却下、決裁などのフローに関する処理）を行っていない間は、「取り消し」ボタンを押下することで、このフローを取り消すことができます。

（フローイベントには「取り消し」を行ったことが記録されます。）

(435) 1. スクリプトで自動起動の設定等を行なっている場合は、上記フォルダへ移動後に startup.bat（startup.sh）を実行するようにしてください。

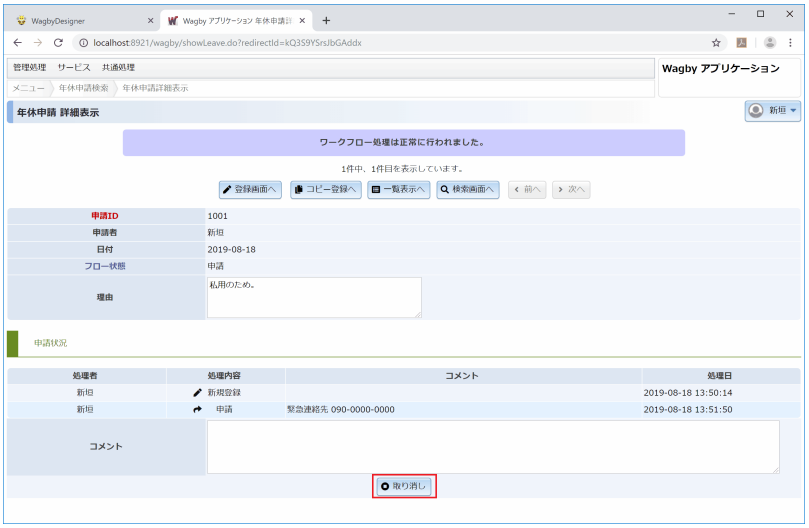


図 取り消しボタン

取り消しボタンが用意される条件

取り消しボタンは次の条件が成立したときに用意されます。

- ワークフローの状態が「申請」状態である。
- ログオンユーザが申請者である。

各処理の動作

フロー実行時の各処理の動作は次のようになります。

申請

登録したデータを承認者に申請し、次のノードにフローを進めます。申請後はデータを更新・削除することはできません。次の承認者がまだ承認処理を行う前であれば、申請を取り消すことができます。

承認

申請フローを承認して次のノードにフローを進めます。最終承認（決裁）の場合は、フローが終了します。 フロー終了後は、申請者、承認者ともにデータを更新・削除することはできません。

差し戻し

一つ前のノードに戻します。差し戻されたデータについて、申請者は再度申請することができます。

却下

申請を却下し、フローを終了させます。フロー終了後は、申請者・承認者ともにデータを更新できませんが、申請者はデータを削除することができます。

処理ボタンの表示ルール

更新ボタン

- ユーザがフローを持っている場合に表示されます。 （例： 申請者による申請後は、詳細画面に更新ボタンが表示されません。承認者がアクセスした場合、更新ボタンが表示されます。）
- 「差し戻し」で申請者にフローが戻ってきた場合は、ボタンが表示され、更新できます。
- フロー終了（決裁）後は、申請者・承認者に関わらず、ボタンが表示されません。
- 「却下」後は、申請者・承認者に関わらずボタンが表示されません。

削除ボタン

- 申請者（データを新規登録したユーザ）のみが使用できます。承認者でアクセスした場合でも削除ボタンは表示されません。
- 申請者による申請後は表示されません。
- 「差し戻し」でフローが戻ってきた場合は、削除ボタンが表示され削除できます。
- フロー終了（決裁）後は、申請者・承認者に関わらず、ボタンが表示されません。
- 「却下」後は、申請者のみ削除できます。

ワークフローの状態とイベント

ワークフローで用意されている「状態」と「イベント」は次のとおりです。

状態

FLOWSTATUS関数で取得できます。

名称	説明
（空白）	未申請の状態です。ワークフローは開始されていません。
申請	申請者がワークフローを開始した状態です。
承認	承認者による承認行為です。多段ノードの場合、いくつもの承認状態に遷移します。
決裁	最終承認者（決裁者）による承認行為です。ワークフローは終了します。
却下	このワークフローが却下された状態です。ワークフローは終了します。再申請のためには、新しいワークフローを起こしてください。
取り消し	申請者による、このワークフローの申請を取り消した状態です。ワークフローは終了します。再申請のためには、新しいワークフローを起こしてください。

イベント

名称	内部表記	説明
開始	Start	ワークフローのデータを新規に登録しました。状態は未申請です。
申請	Application	このワークフローを申請しました。
承認	Admit	このワークフローを承認しました。最終承認者（決裁者）の承認行為も含まれます。
却下	Reject	このワークフローを却下しました。
一つ前へ差し戻す	Back	このワークフローを差し戻ししました。一つ前のノードに戻ります。
申請者へ差し戻す	Rewind	このワークフローを差し戻ししました。申請者（先頭のノード）に戻ります。
取り消し	Cancel	このワークフローを取り消しました。
決裁	End	最終承認を経て、このワークフローが決裁されました。

「却下」「取り消し」ボタンを非表示にする（8.3.0）

ワークフローの「却下」「取り消し」ボタンを非表示にすることができます。（この設定はモデルごとに行います。）

画面 ＞ その他 ＞ ワークフローの設定欄に用意された「却下ボタンを表示しない」「取り消しボタンを表示しない」をそれぞれ有効にします。

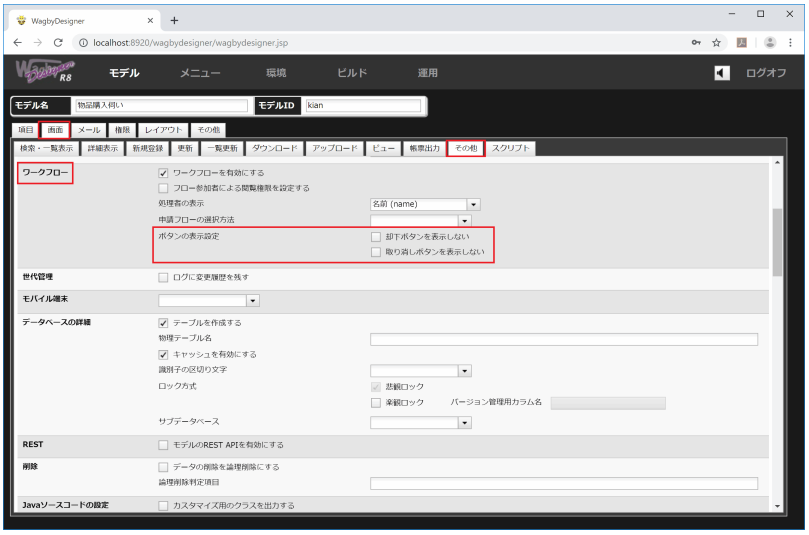


図 「却下」「取り消し」ボタンを非表示にする設定

保留フロー状態の検索画面の却下ボタン

〝承認すべきワークフローを確認する ＞ 却下ボタンを非表示にする〟をお読みください。

ボタン名の変更（8.3.0）

環境タブのメッセージ設定で、ワークフローの操作を行うボタン名とアイコンを変更することができます。

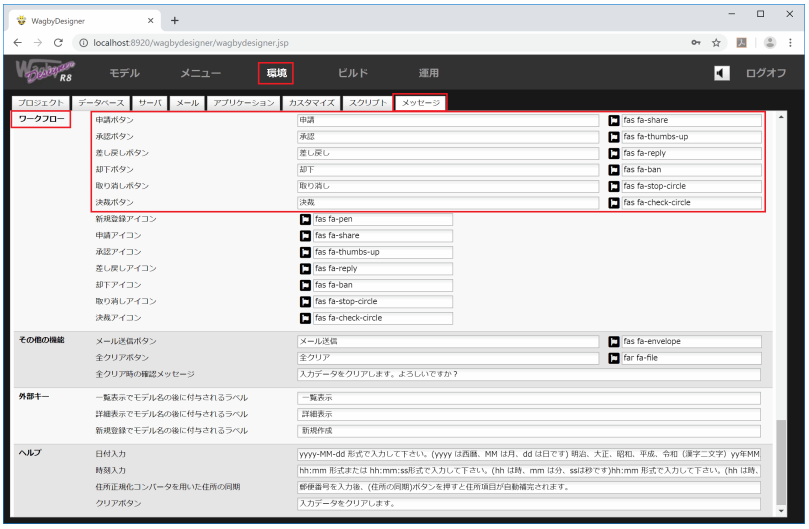


図 ボタン名とアイコンの設定

アイコンの変更（8.3.1）

　環境タブのメッセージ設定で、ワークフロー申請状況の「処理内容」の左横に用意されているアイコンを変更することができます。

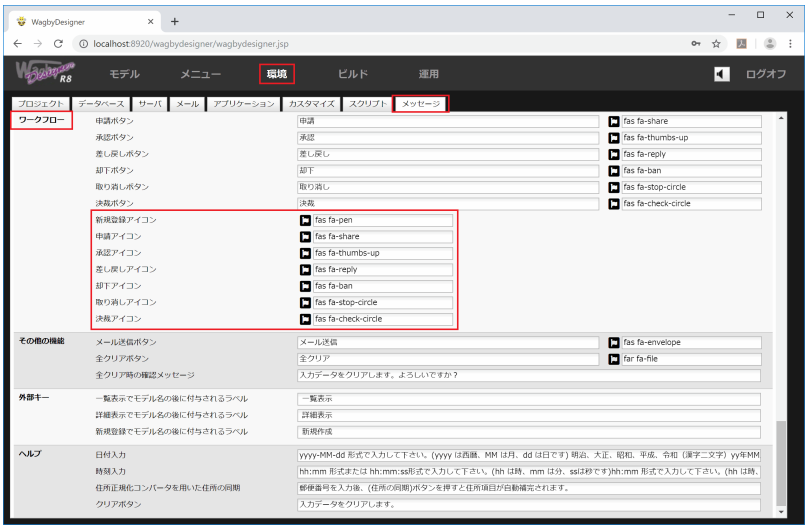


図 アイコン名の設定

申請状況の並び順を指定する（8.3.1）

　″画面>その他″ タブのワークフロー設定欄に用意された「申請状況の表示順」を指定できます。初期値は″昇順″となっています。

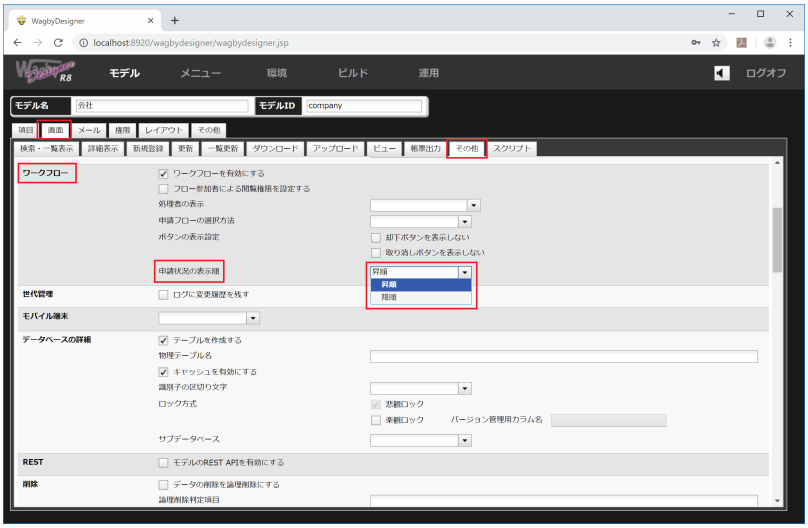


図 申請状況の表示順

PCで申請したワークフローを、スマートフォンで承認・決裁する（8.1.2）

　メインモデルをPC用、サブモデルをスマートフォン用と定義します。このとき、PCで申請したワークフローをスマートフォンで処理できます。

注意点

- スクリプトを使っている場合、メインモデルとサブモデルは区別されます。メインモデルのスクリプトをサブモデルにコピー後、モデルIDを適切に見直してください。（サブモデルを対象にするスクリプトは、サブモデルを指し示すように書き換える必要があります。）
- 代理者の設定を行なっている場合、メインモデルと同じ設定をサブモデルにも行うようにしてください。

処理者に電子印影を用いる

実行

電子印影の登録

ここではWagbyが提供するアカウントモデルを拡張し、「電子印影」という（ファイル型の）項目を用意しています。

各アカウントに、イメージファイルを添付します。

図 アカウント毎に電子印影を用意する

電子印影（電子印鑑）作成のためのソフトウェアは別途、ご用意ください。 ここで使ったのは「クリックスタンパー」という無料のソフトですが、有償製品は（重複しない）独自データを保証するなど、より信頼性の高い印鑑を提供するものもあります。

ワークフロー利用時の動作イメージ

図のように、処理者の欄に名前ではなく電子印影を表示させることができます。

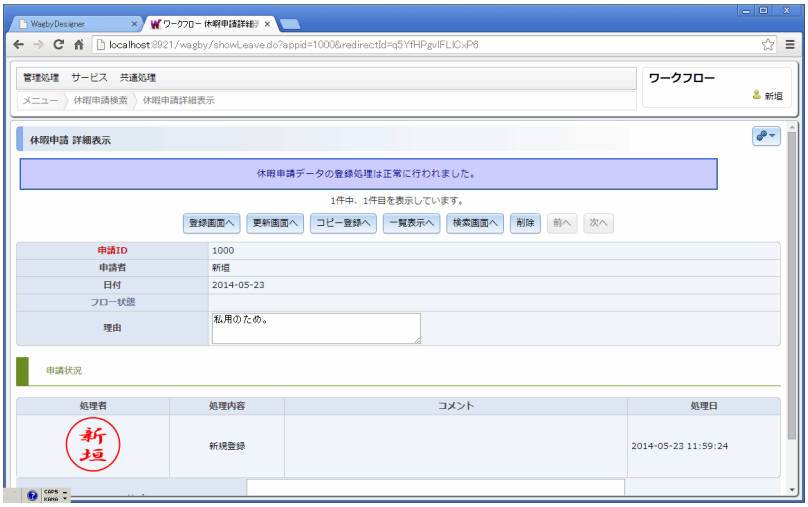


図 処理者の欄に電子印影を表示する

定義方法

アカウント情報(juser)の拡張

モデルの「システム」タブに含まれている「アカウント」の定義を変更します。 ここでは「名前」項目の下に一行、定義を挿入してみます。

図 項目の挿入

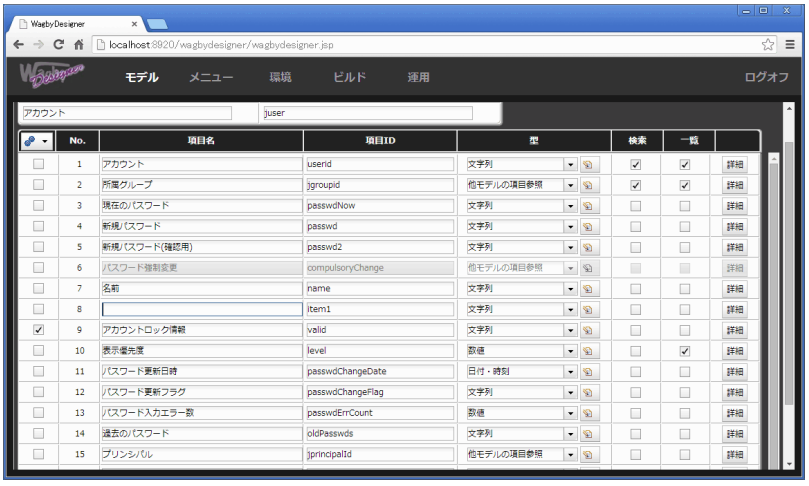


図 新規項目の入力欄が追加される

「電子印影」(imprint) という項目を用意します。型は「文字列 - ファイル名」とします。

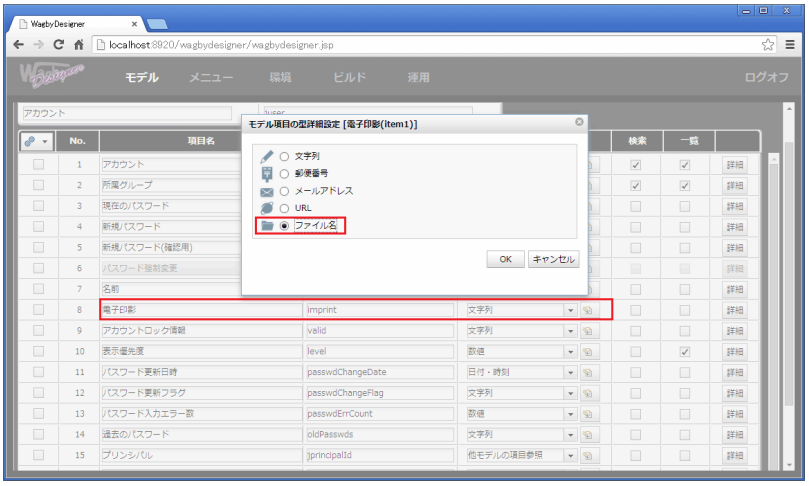


図 電子印影を管理できる項目を設定

このアカウントモデルの「画面>その他」タブを開きます。ファイル名の制御で「オリジナルファイル名をそのまま使う」を選択します。

図 格納ファイルの制御

電子印影の画像ファイルは、半角英数字のみで構成するようにしてください。（例 suzuki_ichirou.png） OSによっては全角英数字や漢字を含むファイル名を正しく認識できないことがあるためです。

ワークフロー利用モデルの修正

ワークフローを設定したモデル（この例では休暇申請モデル）のワークフロー設定を変更します。 処理者の表示に、新しく追加した項目ID "imprint" を指定します。

図 処理者の表示項目を指定する

メールを送信する［1］設定

概要

各ノードにおける「イベント」（承認、差し戻しなど）の発生タイミングで、メールを送信することができます。

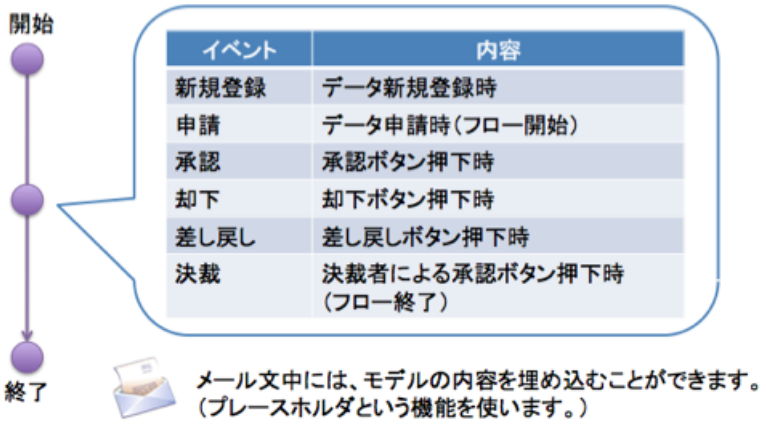


図 メール送信機能

メールの送信元（発信者 "From"）と、送信先（受信者 "To"）を固定するのではなく、ノードに割り当てられたユーザ・グループに送信するように設定できます。ここではその方法を説明します。

アカウントとグループの設定

アカウント

Wagbyが提供するアカウント管理モデルjuserに、メールアドレス項目を加えます。

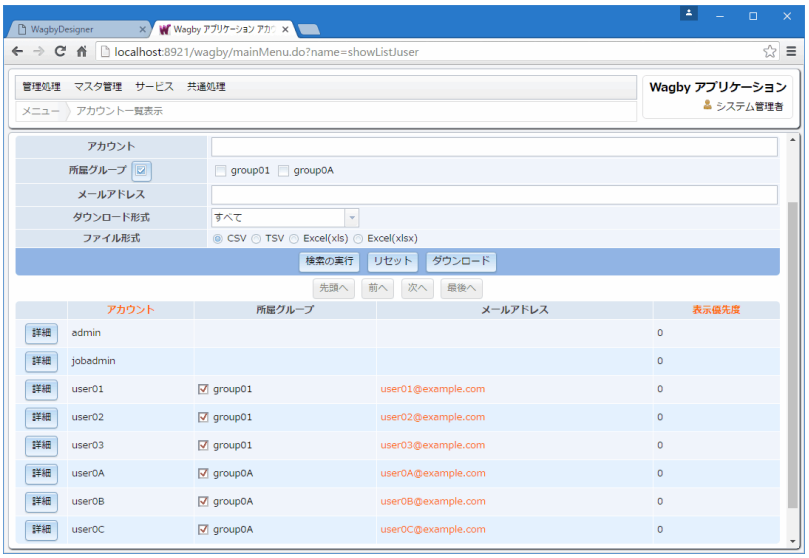


図 メールアドレス項目が追加されたjuserモデル

この設定はアカウント>juserモデルの基本>項目を追加する で説明しています。

グループ

Wagbyが提供するグループモデルjgroupに、メールアドレス項目を加えます。

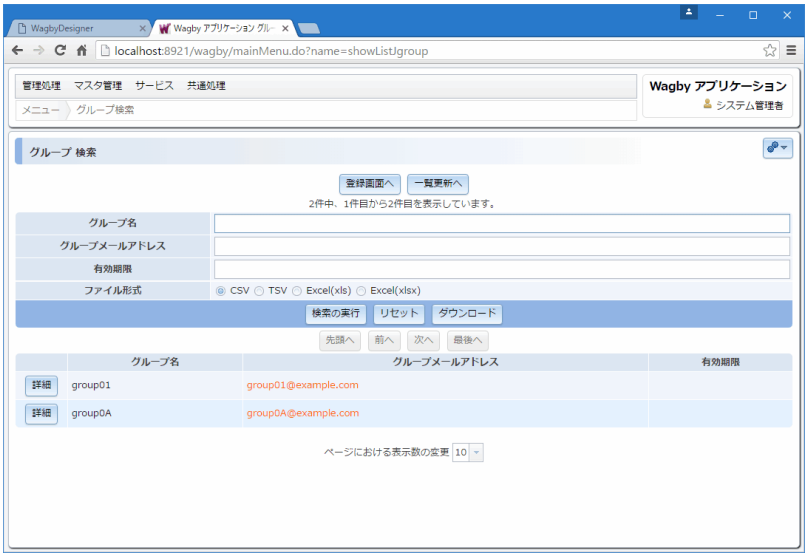


図 メールアドレス項目が追加されたjgroupモデル

グループに設定するメールアドレスは、（複数の）個人メールアドレスへのエイリアスとしてします。社内のメールサーバの設定で、このようなエイリアスを作成することができます。詳細はメールサーバ管理者へお問い合わせください。

モデル定義で、「システム」タブに含まれる「グループ　（jgroup)」を選択します。図に示すように「グループメールアドレス」を挿入します。

項目名	項目ID	型	型への属性
グループメールアドレス	jgroupmailaddress	文字列型	メールアドレス

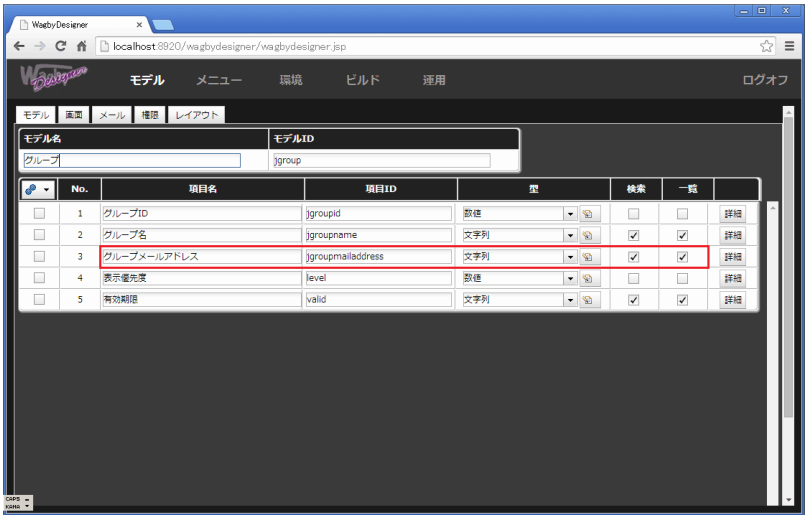


図 jgroupモデルへのメールアドレス項目の追加

ワークフローの設定

フロー参加者設定

今回は「順次・順次・順次フロー」（申請→承認→決裁）　と、「順次・合議・順次フロー」（申請→合議承認→決裁）の二通りの例を使うとします。

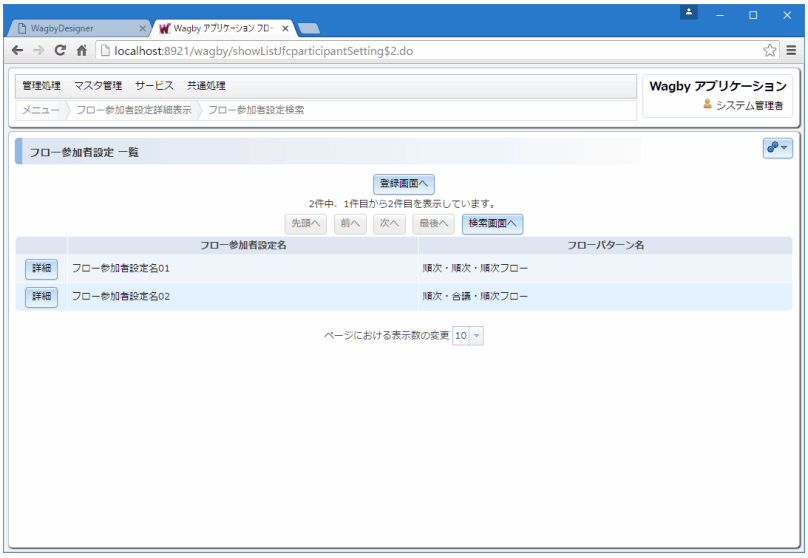


図 フロー参加者設定一覧

「順次・順次・順次フロー」では、申請、承認、決裁いずれもユーザーとしています。

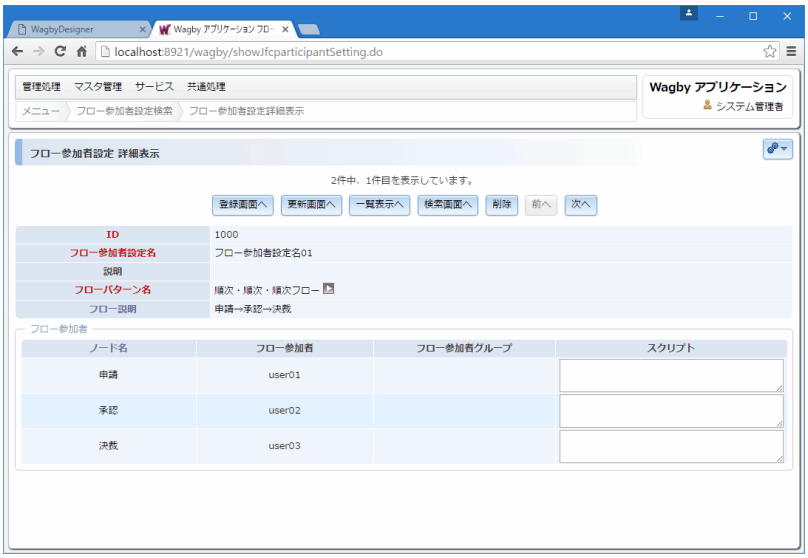


図 順次・順次・順次フローの参加者

「順次・合議・順次フロー」では、申請をグループとします。また、二番目の合議では「承認1～承認4」を用いるとします。

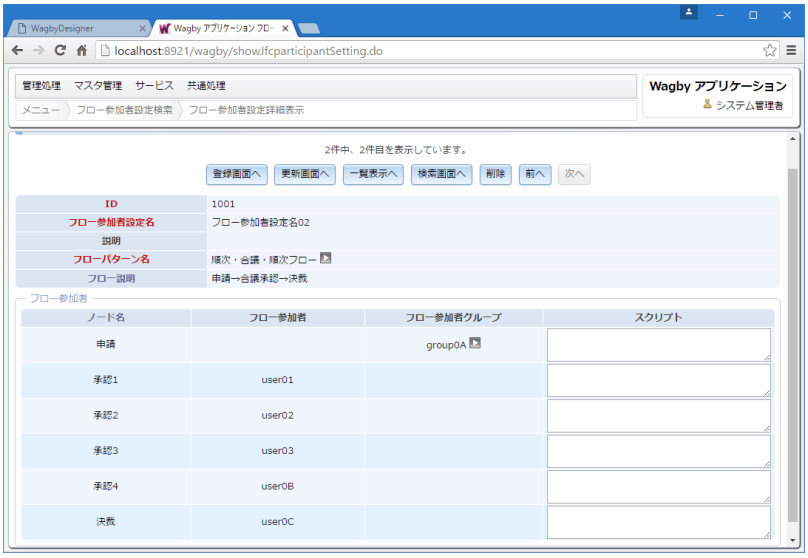


図 順次・合議・順次フローの参加者

フローパターン

なお、この「順次・合議・順次フロー」の合議では、「承認1～承認4」のうち、2人が承認すれば次へ進む、というように設定しているものとします。

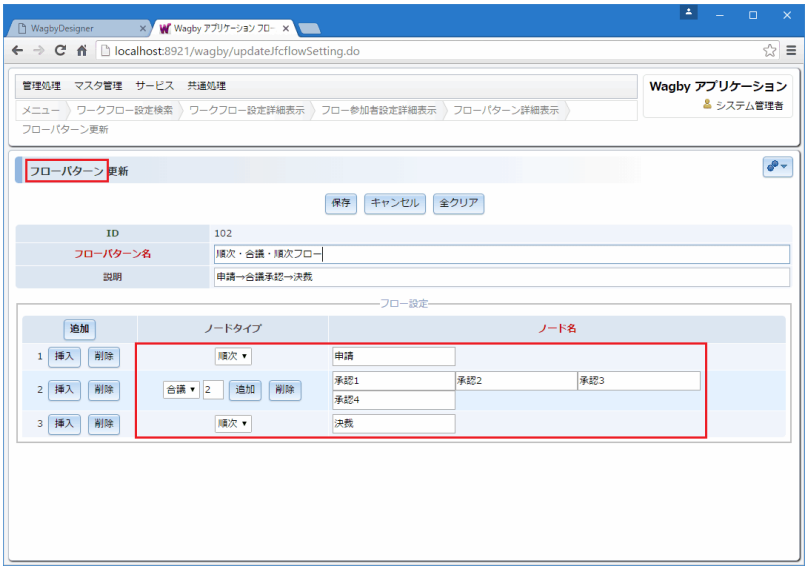


図 順次・合議・順次フローパターン

ワークフロー設定

図のフロー参加者設定を使って、ワークフローを設定します。メールの設定もここで行います。

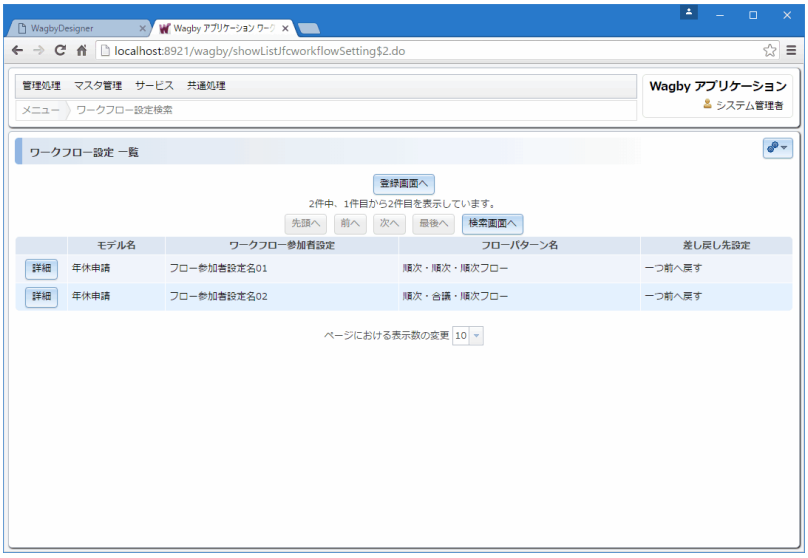


図 フロー参加者設定一覧

「フロー参加者設定名01」は「順次・順次・順次フロー」を用います。申請時にメールを送信するようにします。

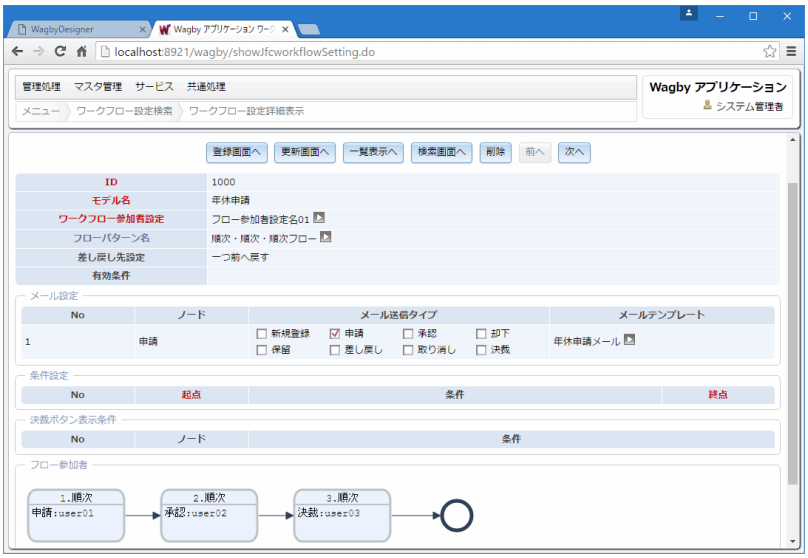


図 フロー参加者設定名01

「フロー参加者設定名02」は「順次・合議・順次フロー」を用います。申請時および合議承認時にメールを送信するようにします。

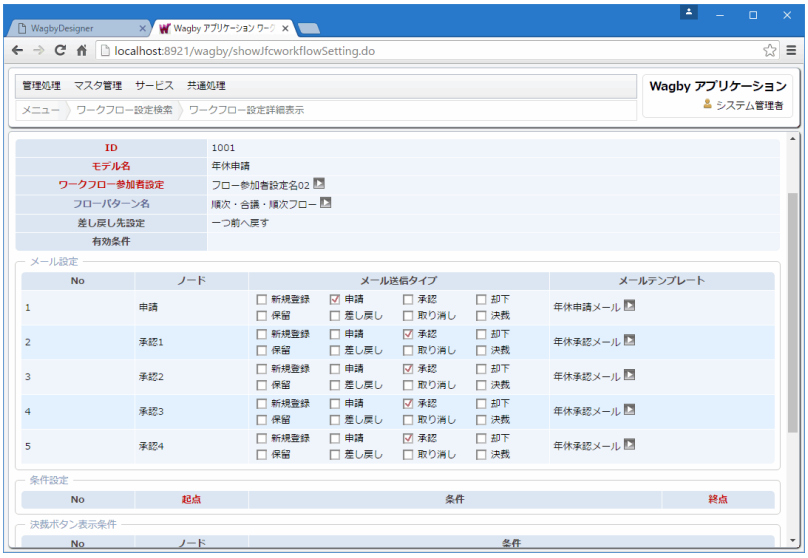


図 フロー参加者設定名02(1)

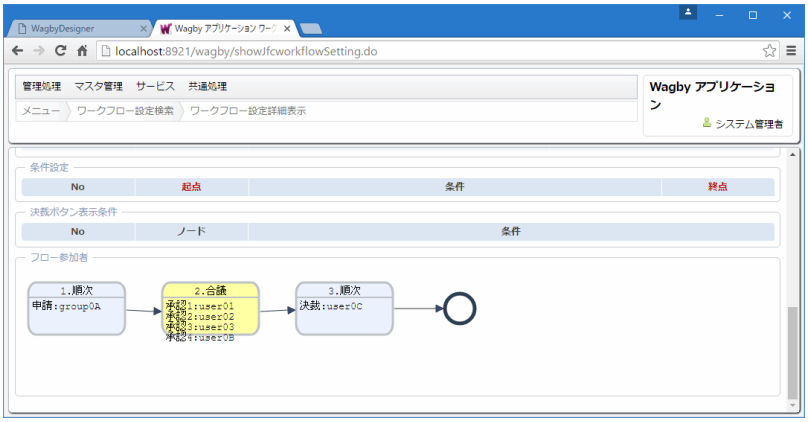


図 フロー参加者設定名02(2)

ワークフロー関数の適用

「年休申請」モデルに次の項目のいずれか1つを追加します。

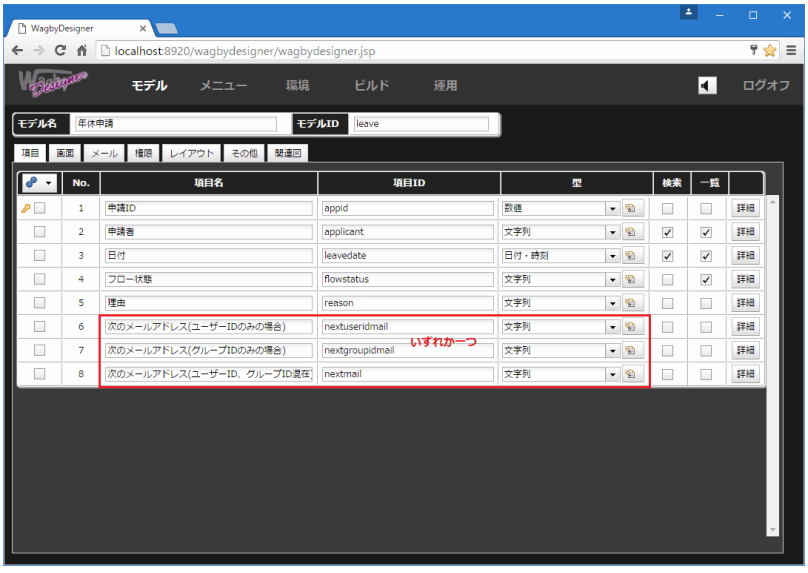


図 年休申請モデルへの項目追加

「申請者」項目の初期値

申請者はログオンIDが自動設定されるようにします。

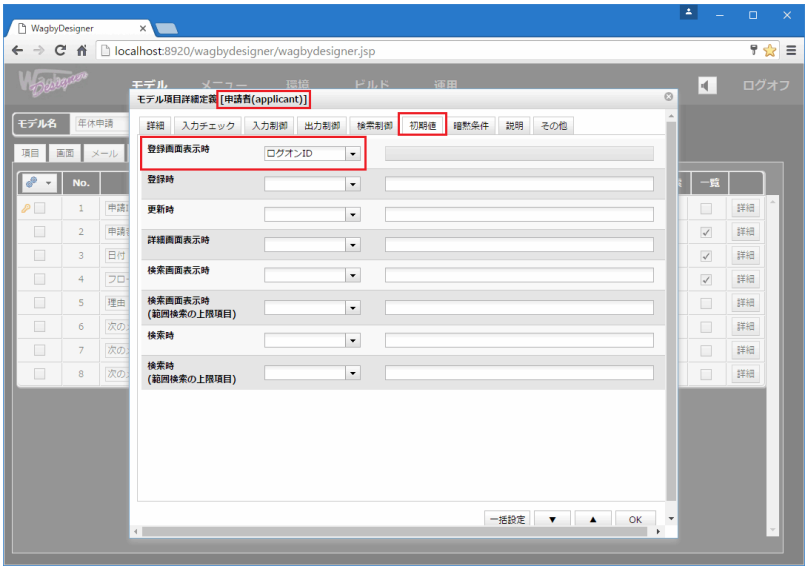


図 初期値の設定

次のメールアドレス項目

申請時であれば次の承認者へ、承認時であれば次の承認者（または決裁者）というように、次のノードのメールアドレスを格納する項目を用意します。

項目名	項目ID	型	式	DB保存	使い分け
次のメールアドレス（ユーザーIDのみの場合）	nextuseridmail	文字列	WF_NEXT_USERID("userid","mailaddress")	×	グループ宛にメールを送る必要がない場合
次のメールアドレス(グループIDのみの場合)	nextgroupidmai	文字列	WF_NEXT_GROUPI("jgroupid","jgroupmailadd	×	ユーザー宛にメールを送る必要がない場合
次のメールアドレス（ユーザーID、グループID混在）	nextmail	文字列	WF_NEXT_MAIL("","userid","mailaddress","jgroup","jgroupid","jgroupmailadd	×	ユーザー、グループのいずれにもメールを送る可能性がある場合

(436) (437)

各関数の引数の詳細はWF_NEXT_USERID_MAIL, WF_NEXT_GROUPID_MAIL, WF_NEXT_MAIL の説明をお読みください。

メールテンプレートの設定

メールのテンプレートを用意します。

(436) このサンプルでは、理解するために3つの項目を用意していますが、実際には上記説明にあるように、使い分けによっていずれか一つの項目を用意するようにしてください。

(437) 上記の項目は、本番運用時には隠し項目として、画面では非表示とすることもできます。

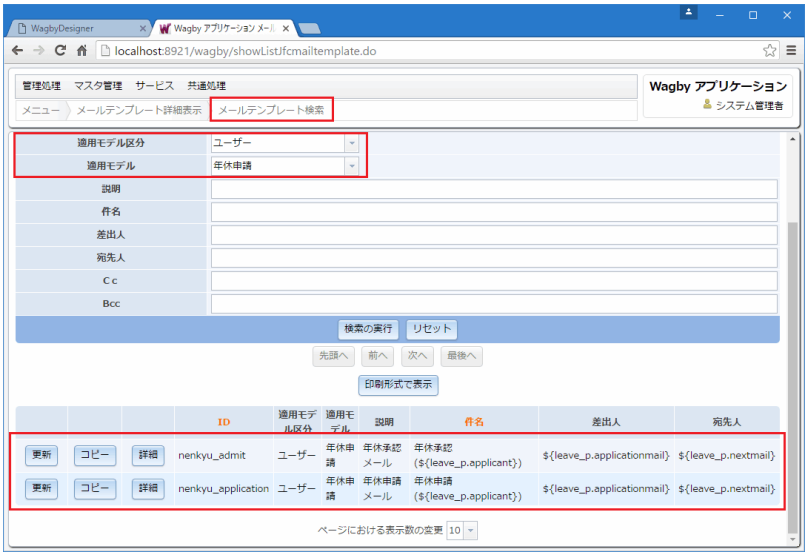


図 メールテンプレート

メールテンプレートを設計します。「ID」は重複しない識別子とします。「適用モデル」に　″年休申請″を選択します。「件名」「差出人」「本文」などの項目にはプレースホルダを含めることもできます。

ここでは″申請″と″承認″の二つのメールテンプレートを設計します。

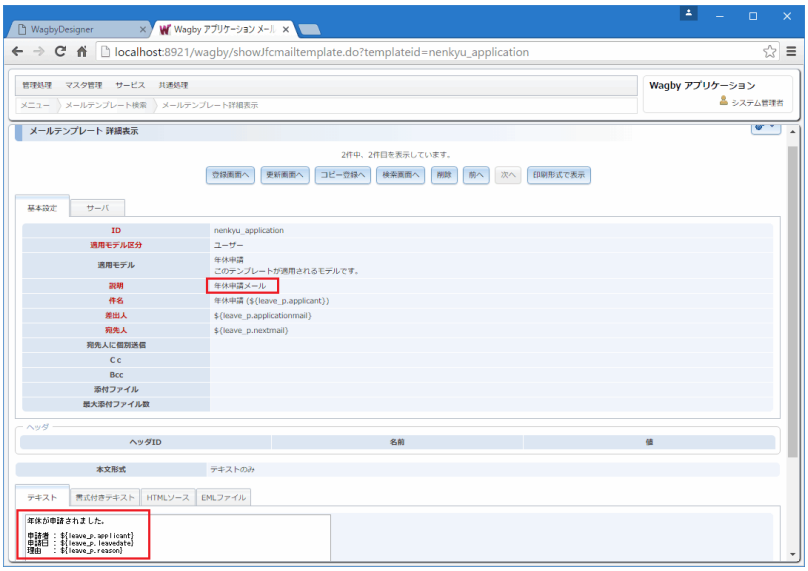


図 申請テンプレート

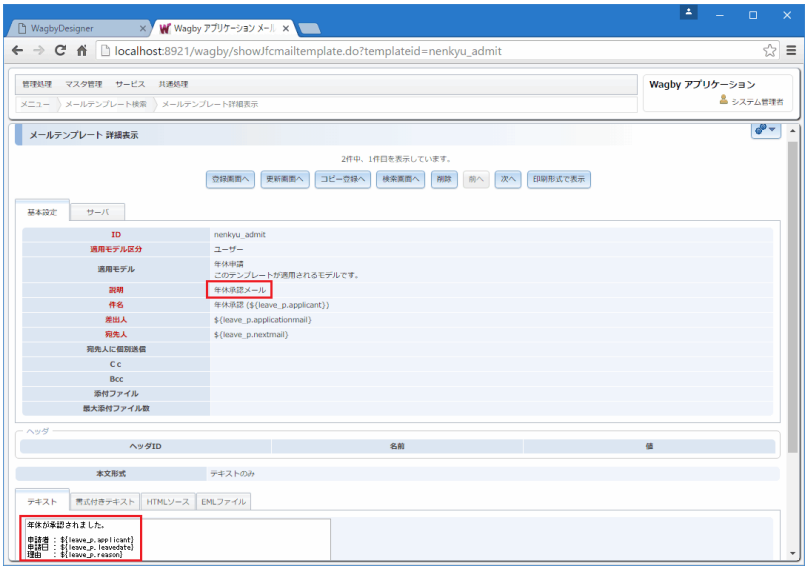


図 承認テンプレート

メール送信タイプとノードの組み合わせ

メール送信タイプ

各メール送信タイプの動作は以下の通りです。

メール送信タイプ	動作
新規登録	ワークフロー対象のモデルでデータを新規登録した場合（保存ボタン押下時）に送信されます。
申請	ワークフロー申請ボタン押下時に送信されます。
承認	ワークフロー承認ボタン押下時に送信されます。
却下	ワークフロー却下ボタン押下時に送信されます。
保留	-（動作しません）
差し戻し	ワークフロー差し戻しボタン押下時に送信されます。
取り消し	-（動作しません）
決裁	ワークフロー決裁者(フローの最終ノードに割り当てられている人)による承認ボタン押下時に送信されます。

[事例] 「担当者→課長→部長→購買部→購買部部長」ルートによるメール設定

ノードとメール送信タイプの組み合わせでメールテンプレートを指定する例を示します。

1. 担当者が「申請」した時、"課長宛てテンプレート"で申請メールを課長に送信。
2. 課長が「承認」した時、"部長宛てテンプレート"で1次承認メールを部長に送信。
3. 部長が「承認」した時、"購買部宛てテンプレート"で2次承認メールを購買部に送信。
4. 購買部が「承認」した時、"購買部部長宛てテンプレート"で3次承認メールを購買部部長に送信。
5. 購買部部長が「決裁」した時、"申請者宛てテンプレート"で決裁完了メールを申請者に送信。

この例ではメール送信タイプ「承認」に、3つのノードが設定されています。それぞれに別のメールテンプレートを割り当てることで、適切な運用を行うことができるようになります。

メール文中にワークフロー内部の情報を利用する

ワークフローの承認や差し戻しをする際にコメントを入力できるテキストエリアの内容をメールテンプレートに記述することができます。 具体的には次のプレースホルダをメールテンプレートの本文中に記載することで、入力されたコメント内容をメールに含めることができます。

```
${jfcworkstate_p.comment}
```

同様に、次のようなプレースホルダを送信メールに含めることができます。

モデル名.項目名 表記	内容
${jfcworkstate_p.modelname}	対象モデル名
${jfcworkstate_p.modelpkey}	対象モデルの主キー
${jfcworkstate_p.participant}	参加者
${jfcworkstate_p.username}	処理者
${jfcworkstate_p.event}	処理内容
${jfcworkstate_p.comment}	コメント
${jfcworkflow_setting_p.participants}	ワークフロー参加者
${jfcworkflow_setting_p.flow}	ワークフローパターン名
${jfcworkflow_setting_p.mail_setting.0.mstype}	メール設定のメール送信タイプ (438)
${jfcworkflow_setting_p.mail_setting.0.mail_te	メール設定のメールテンプレート

(439)

メールを送信する［2］実行

順次・順次・順次フロー

「フロー参加者設定名01」は「順次・順次・順次フロー」を用います。この例では user01→user02→user03 を使って説明します。

申請

はじめに、user01 でログオンします。

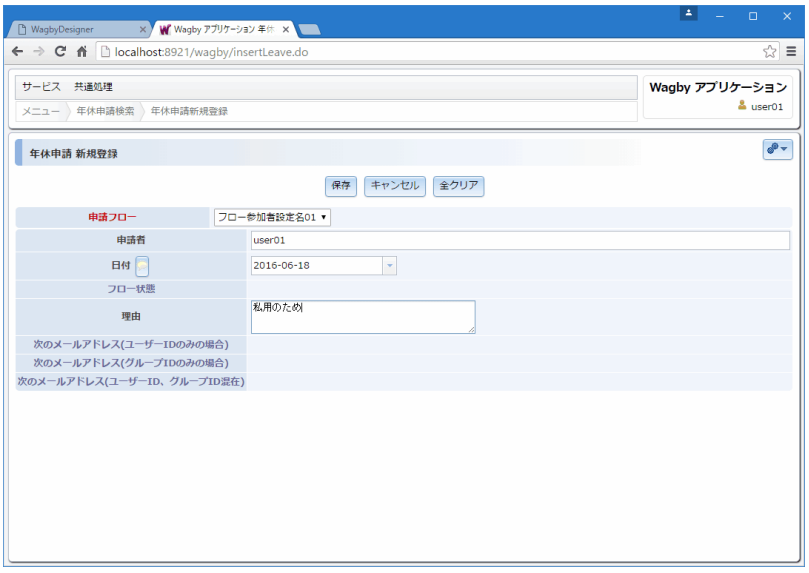


図 user01で年休申請の登録を行う(1)

(438) 繰り返しコンテナのため、".0." や ".1." といった表記で、繰り返しコンテナの対応行を指定します。

(439) この表記を、計算式に用いることはできません。計算式にワークフローの状態を使う場合は「関数>ワークフロー」を利用してください。

データを登録しました。この時点ではまだワークフローは開始していません。次のメールアドレスは申請者となっています。

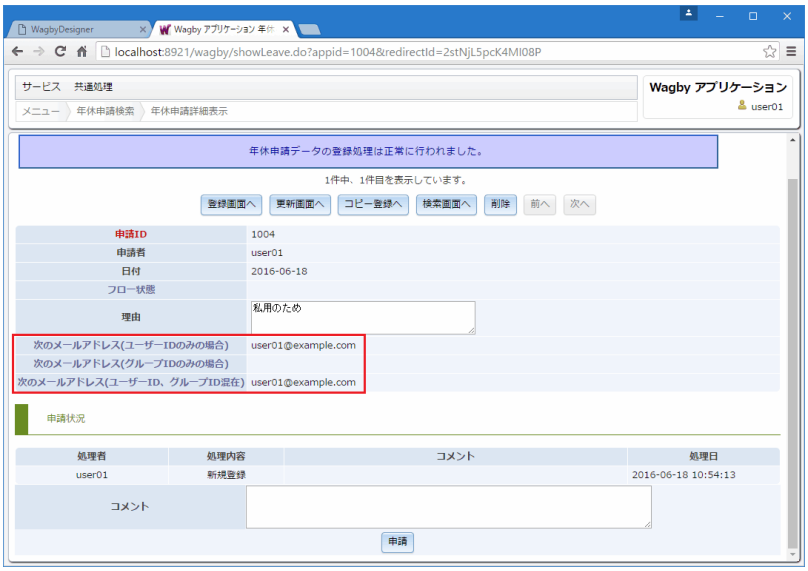


図 user01で年休申請の登録を行う(2)

ワークフローを申請します。

図 ワークフローを申請する(1)

申請すると、次のメールアドレスが user02 に変わります。

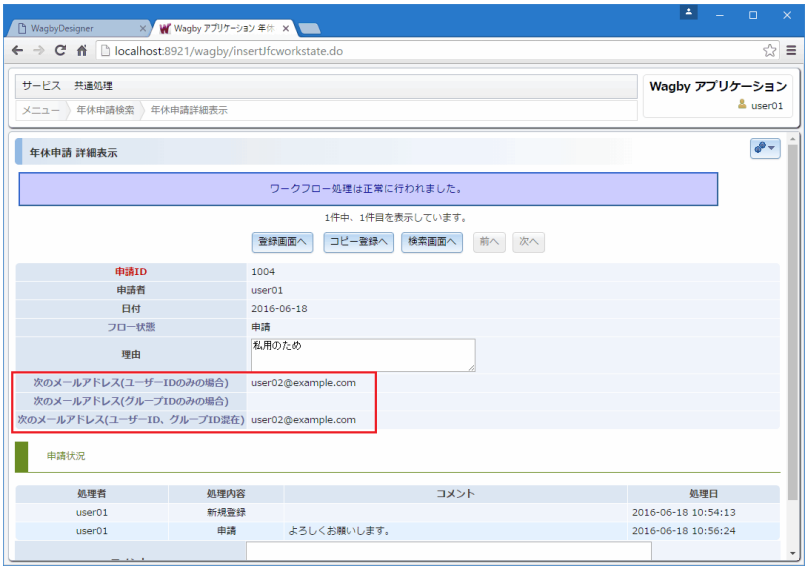


図 ワークフローを申請する(2)

承認

user02 でログオンし、フローを承認します。

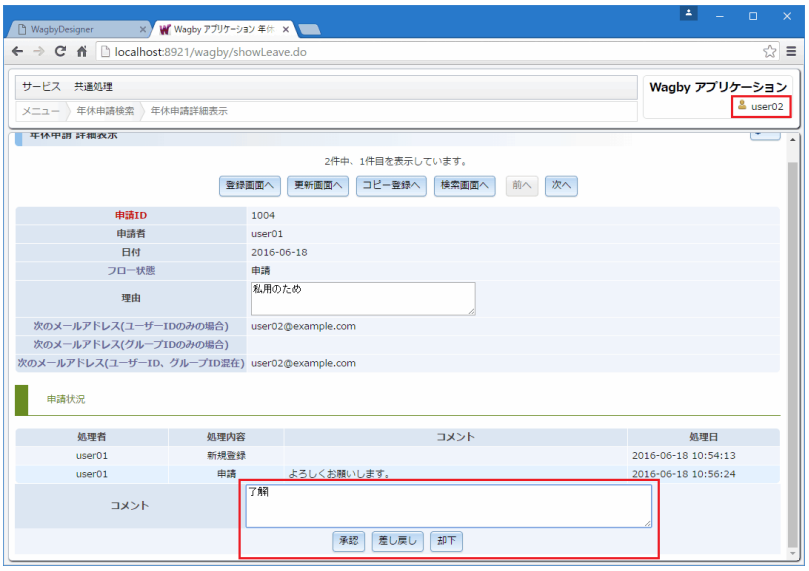


図 user02で承認する(1)

承認すると、次のメールアドレスが user03 に変わります。

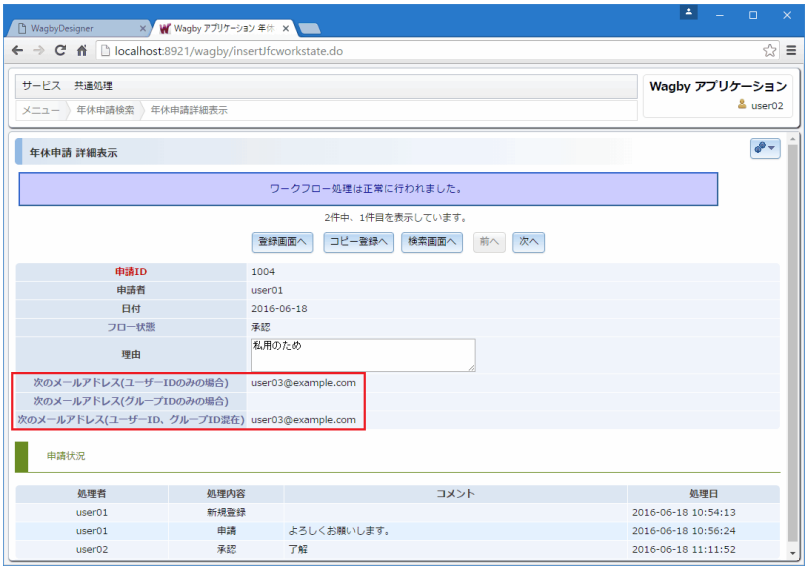


図 user02で承認する(2)

決裁

user03 でログオンし、フローを承認します。

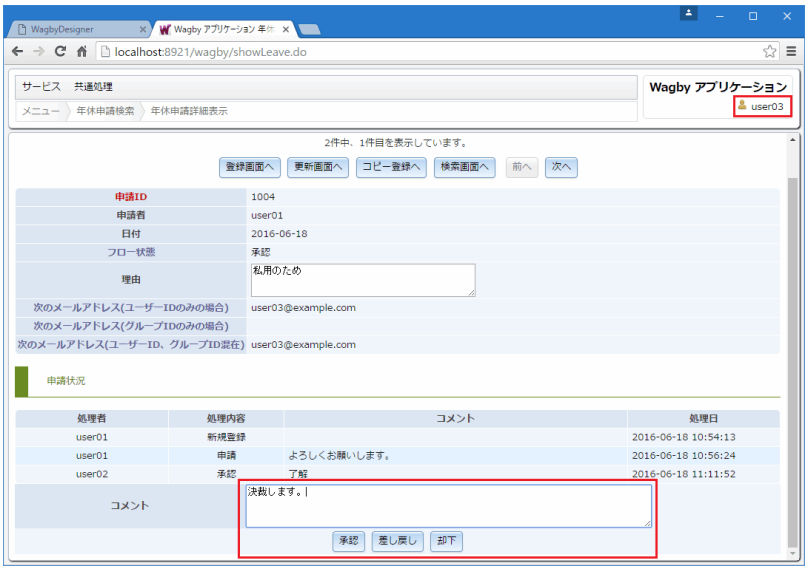


図 user03で承認する(1)

フロー状態が″決裁″になりました。次のメールアドレスは空になります。

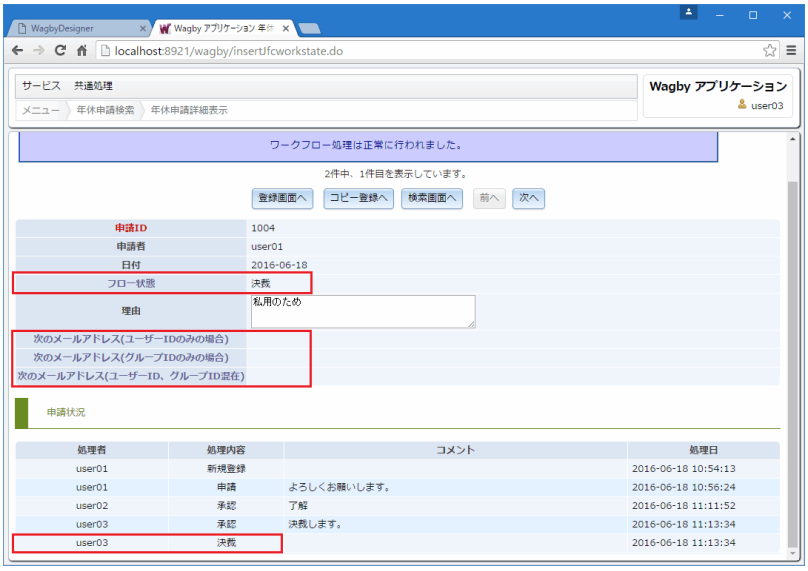


図 user03で承認する(2)

順次・合議・順次フロー

「フロー参加者設定名02」は「順次・合議・順次フロー」を用います。この例では、user0A→(user0B, user01)→user0C を使って説明します。

申請

はじめに、group0A に所属している user0A でログオンします。（今回のフロー設定では申請者は group0A に所属していれば、誰でも可能です。）

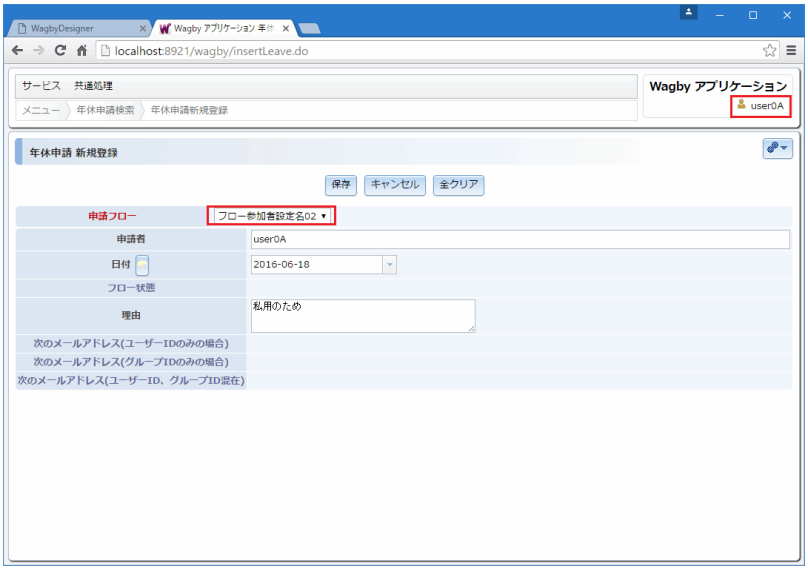


図 user0Aで年休申請の登録を行う(1)

データを登録しました。この時点ではまだワークフローは開始していません。次のメールアドレスは申請者となっています。

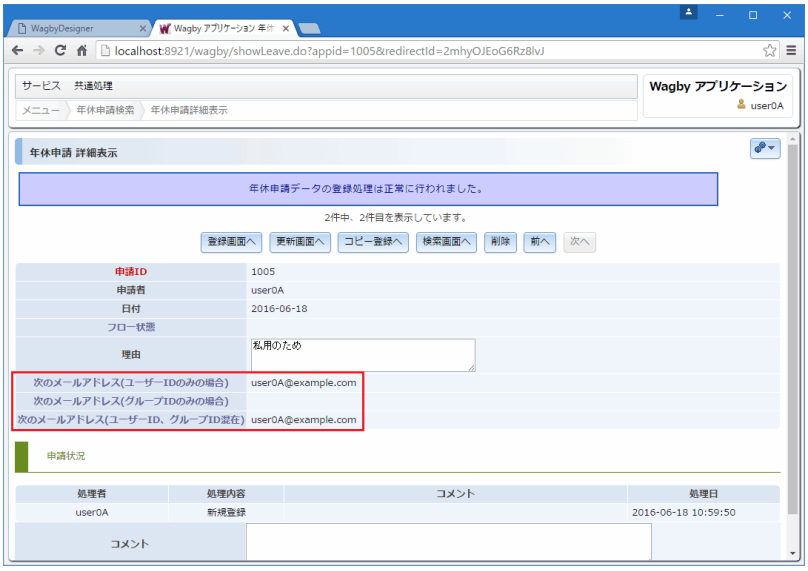


図 user0Aで年休申請の登録を行う(2)

ワークフローを申請します。

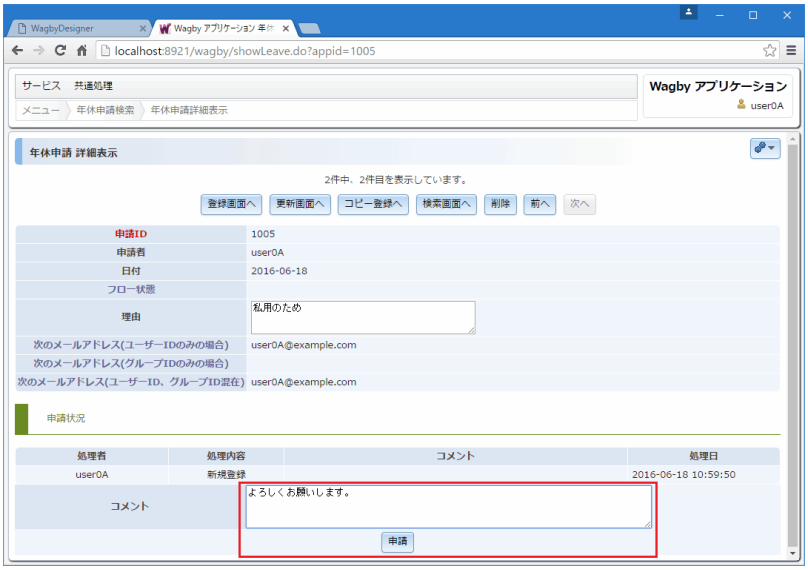


図 ワークフローを申請する(1)

申請すると、次のメールアドレスが user01,user02,user03,user0B に変わります。これは合議に含まれるすべての対象者のメールアドレスです。

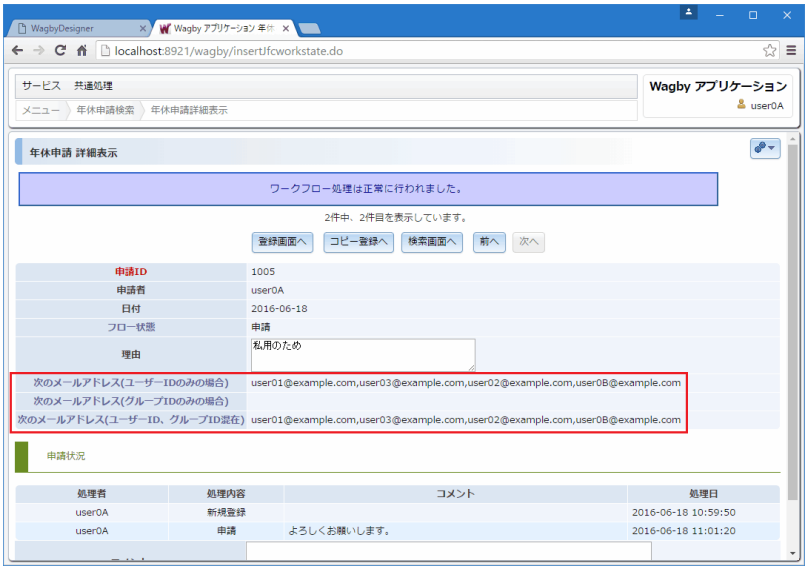


図 ワークフローを申請する(2)

承認 [1]

user0B でログオンし、フローを承認します。

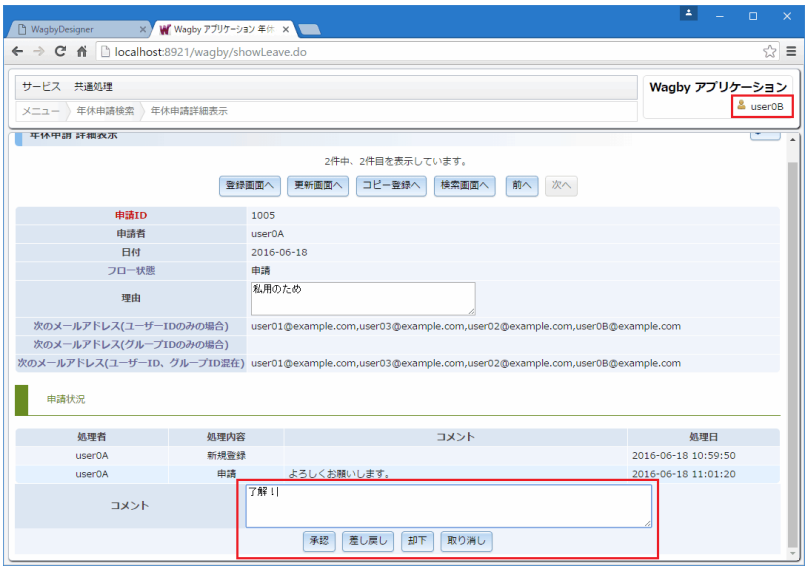


図 user0Bで承認する(1)

承認すると、次のメールアドレスが user01,user03,user03 に変わります。

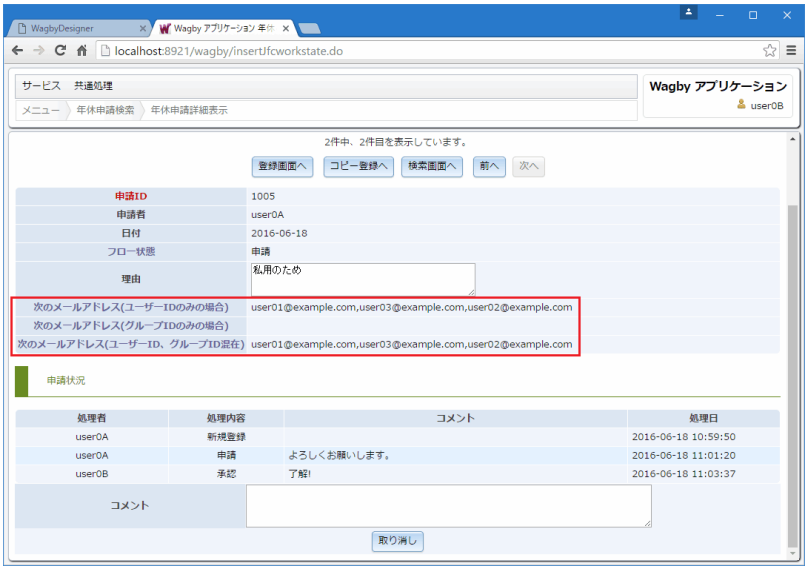


図 user0Bで承認する(2)

承認 [2]

user01 でログオンし、フローを承認します。ここで二回目の承認が行われたため、次のノード（決裁）へ進みます。次のメールアドレスは user0C に変わります。

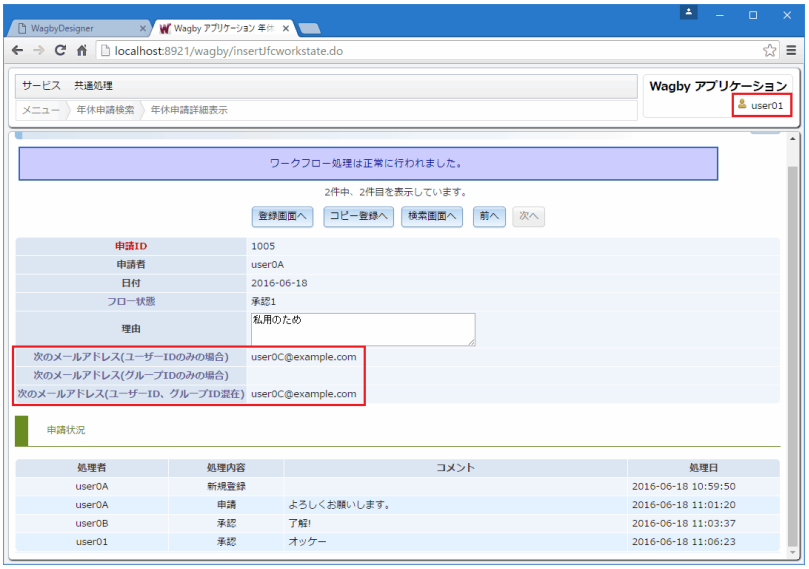


図 user01で承認する

決裁

user0C でログオンし、フローを承認します。

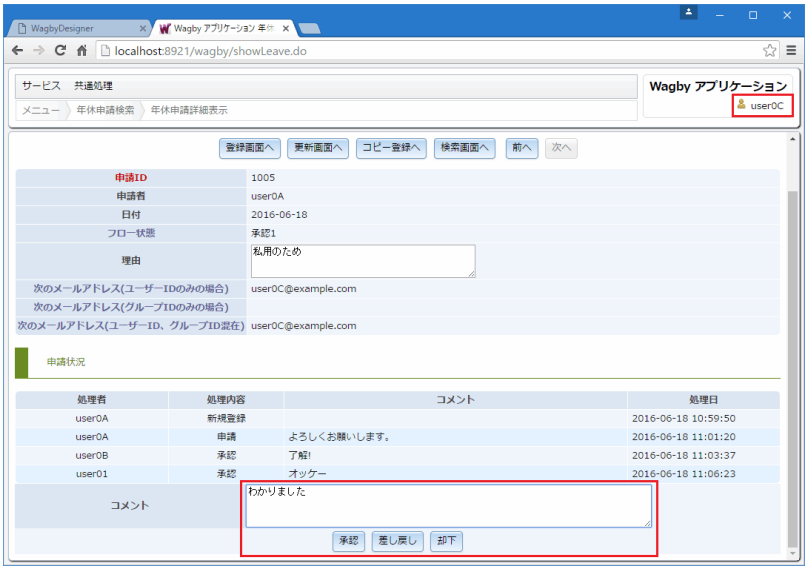


図 user0Cで承認する(1)

フロー状態が"決裁"になりました。次のメールアドレスは空になります。

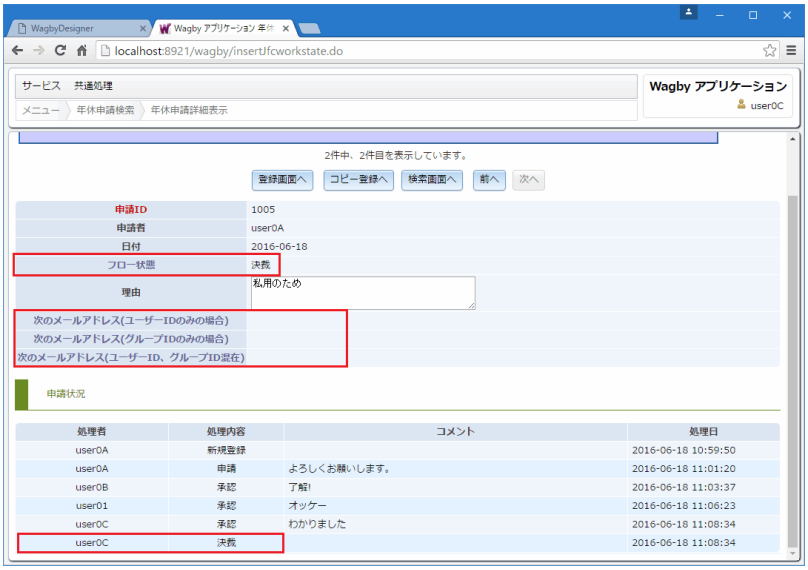


図 user0Cで承認する(2)

メールを送信する［3］自動通知

処理日付時刻項目を用意する

年休申請モデルに「最後の処理日付時刻」項目を追加します。型は日付・時刻型（日付と時刻）とします。

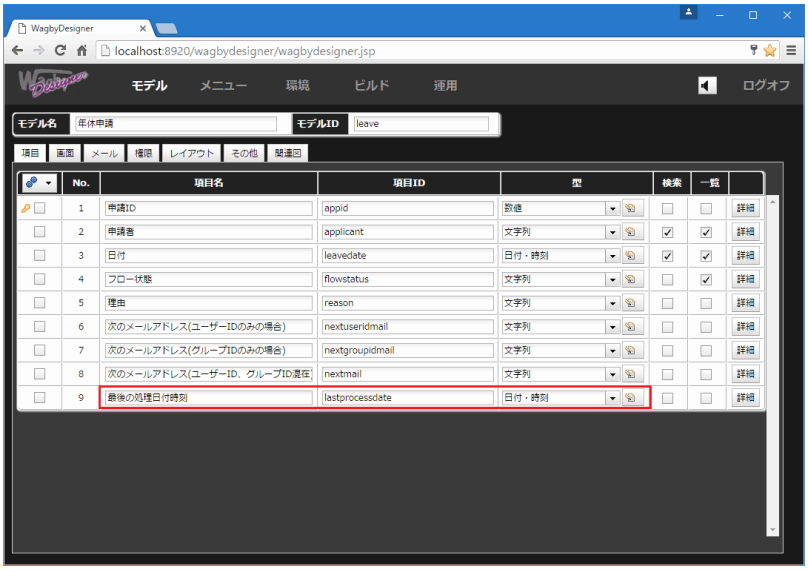


図 最後の処理日付時刻項目

「最後の処理日付時刻」の型は日付・時刻型（日付と時刻）とします。

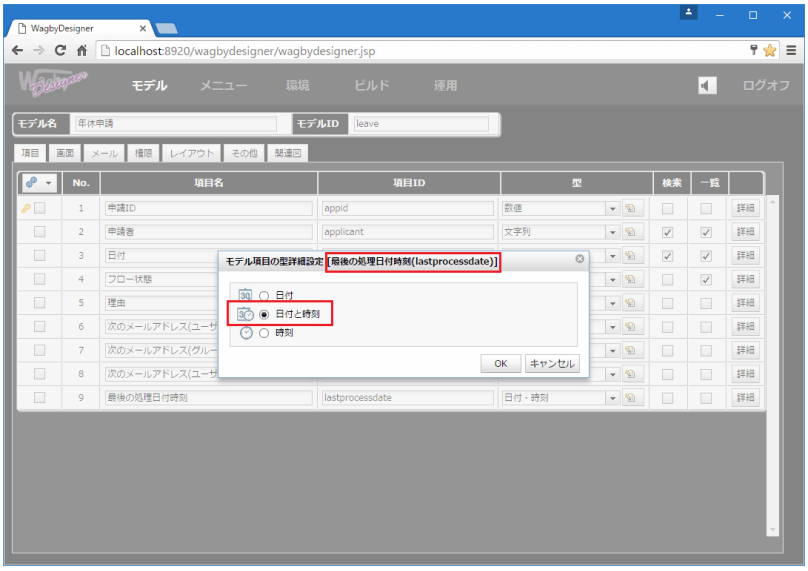


図 日付と時刻型

関数 WF_LASTPROCESSDATE を適用します。最後にワークフローに何らかの処理（承認、決裁など）を行った日付時刻がセットされるようになります。データベースには保存しない、とします。

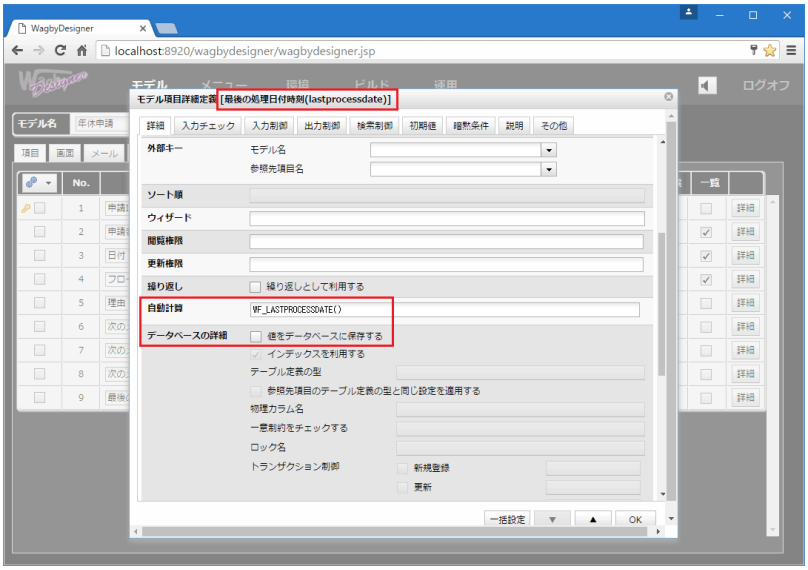


図 WF_LASTPROCESSDATE関数を適用する

メール送信ジョブの定義

メール送信ジョブを用意します。ここではジョブ名を"NotifyRetentionLeaveWorkflowJob"としました。メールテンプレート項目に設定した値は、次節でメールテンプレートIDとして使います。

起動条件に次の式を用意します。この式は、申請または承認状態で、1日以上滞留しているデータのみを選択するものです。

```
AND(
  OR(
    EXACT(${flowstatus},"申請"),
    EXACT(${flowstatus},"承認")
  ),
  DATEDIF(TODAY(),${wflastprocessdate}, "D")>=1
)
```

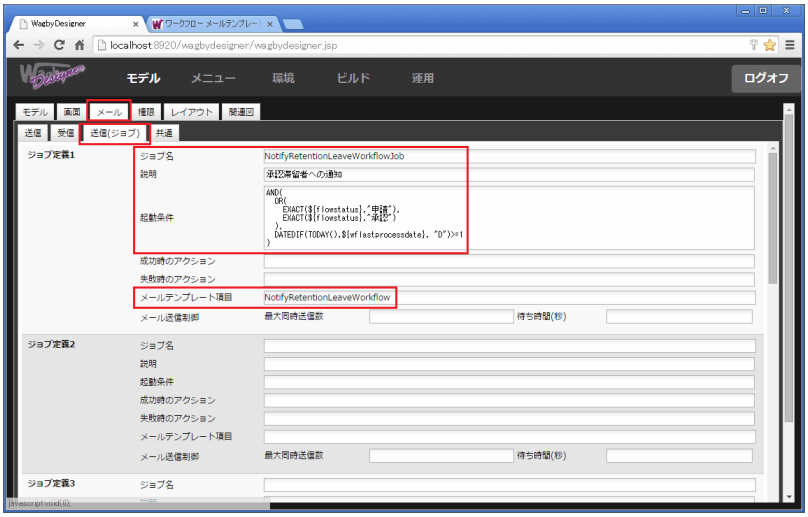


図 メール送信ジョブ

メールテンプレートを用意する

ビルドしたアプリケーションで、送信メールの宛先や文面を用意します。メールテンプレートのIDは、図で指定したメールテンプレート項目と同じとします。（ここでは "NotifyRetentionLeaveWorkflow" としています。）

宛先人の項目に、この休暇申請モデルが保持する、次のメールアドレスをプレースホルダ形式で指定します。件名、差出人、文面は任意に設定してください。

```
${leave_p.wfnextuseridmail}
```

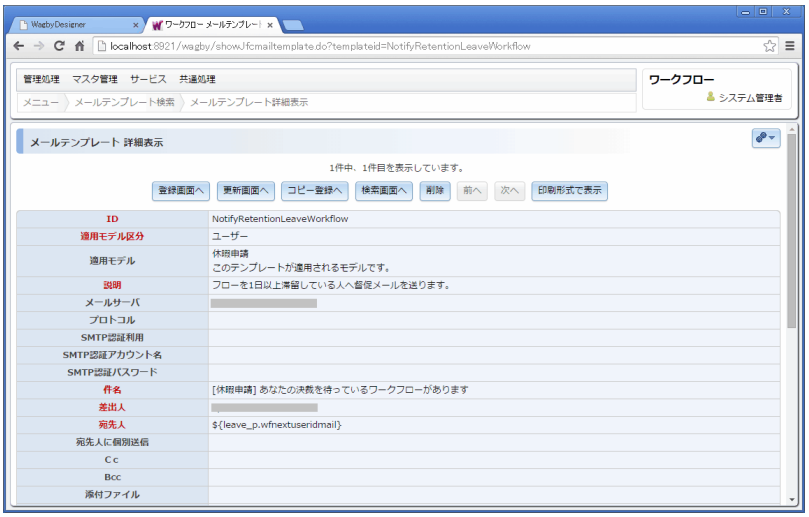


図 メールテンプレートの設定

メール送信ジョブの起動

用意していたメール送信ジョブの時間実行を指定します。実行は jobadmin アカウントが行うものとします。（jobadmin アカウントには、この休暇申請モデルのプリンシパルを与えておくとします。）

(440)

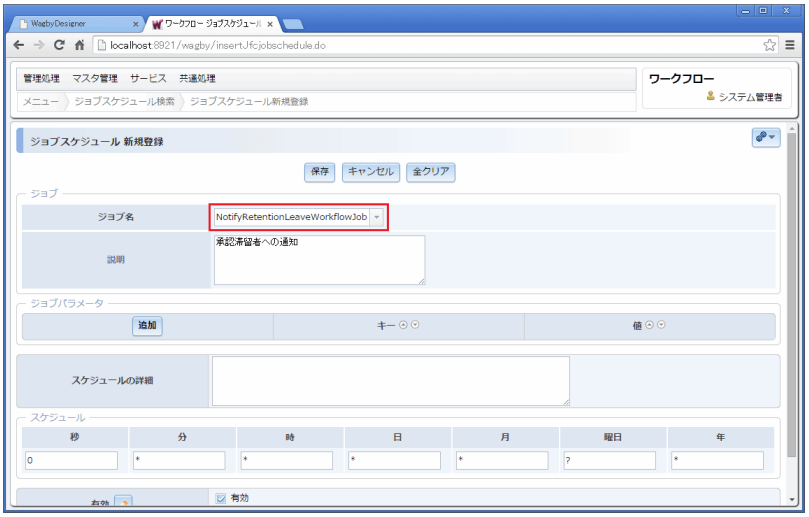


図 メール送信ジョブの起動設定

(440) 図はテストのために毎分実行の例となっていますが、実際には一日おきとするなど、システムに負荷のないように運用してください。

例

年休申請ワークフローを開始した例です。

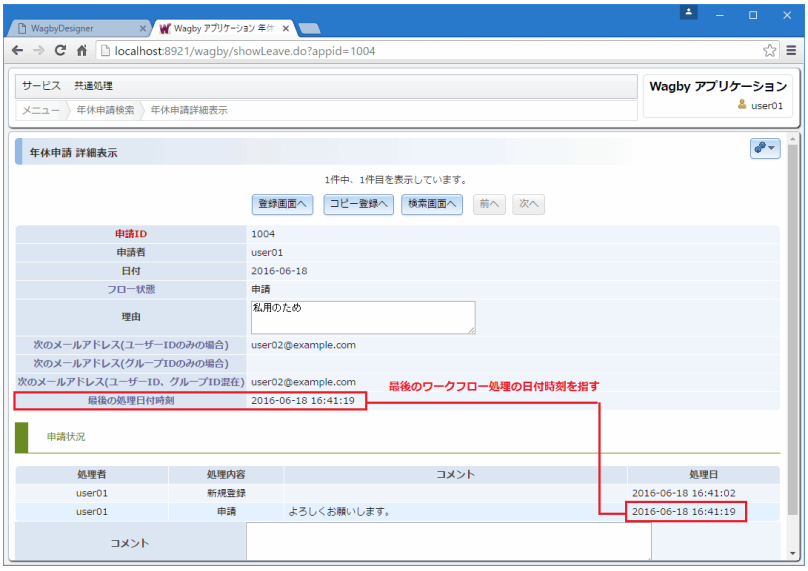


図 年休申請ワークフローを開始した

ここで決裁しなかった場合、未決裁のデータについて、次の承認者へメールが送信されます。

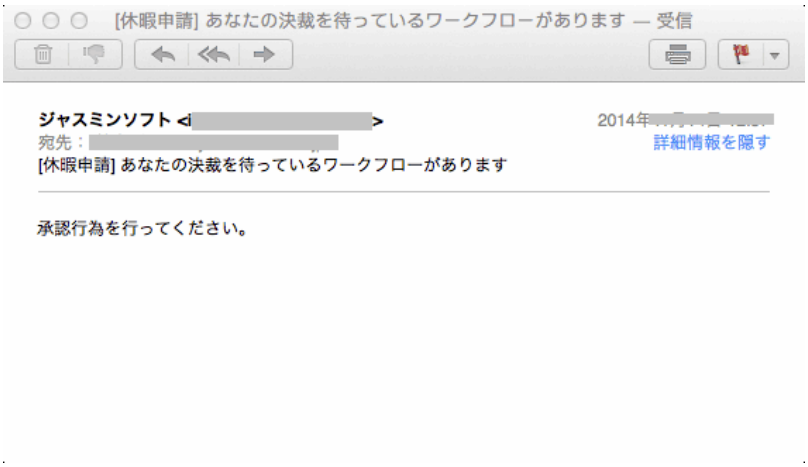


図 送信されたメールの例

代理承認

代理承認のイメージ

承認者の「代理者」を指定することができます。

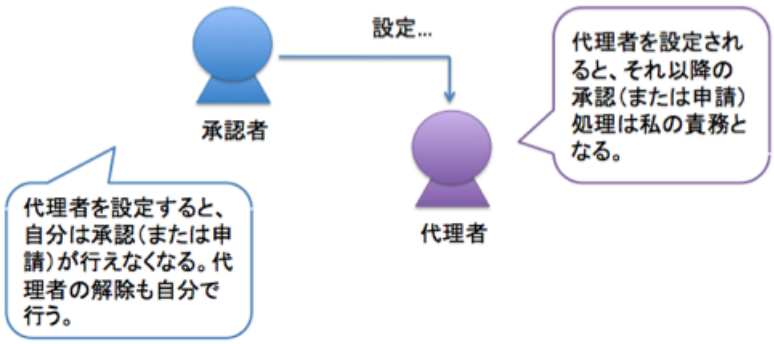


図 代理者のイメージ

準備

承認者に次のプリンシパルを設定しておきます。

- アカウント閲覧者
- ワークフロー代理設定者

図 プリンシパルの追加

申請

申請者でログオンし、休暇申請を行います。するとワークフローが開始されます。

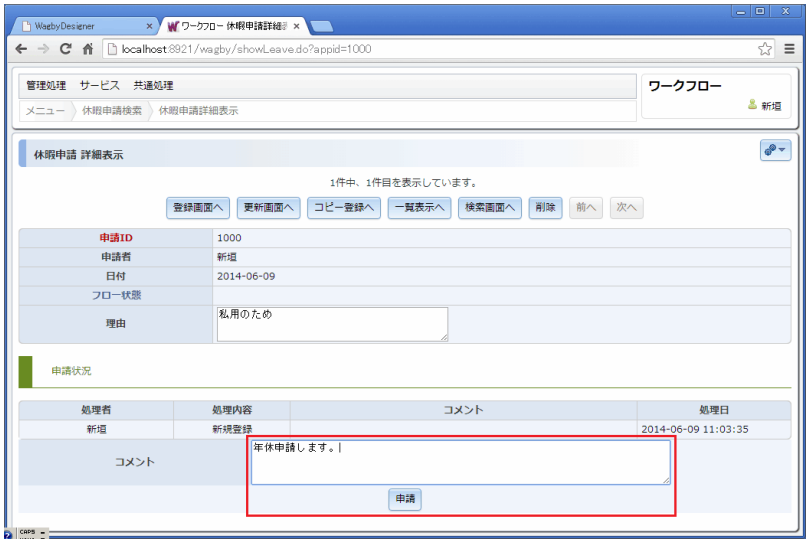


図 申請ボタンを押す

図 ワークフローが開始される

承認内容の確認

承認者でログオンします。承認・差し戻し・却下のボタンが押せるようになっています。

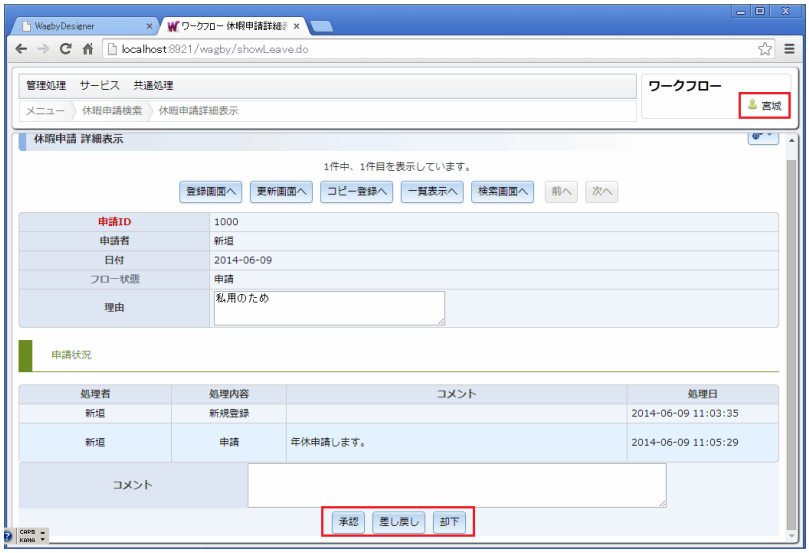


図 承認者による操作画面

代理者の設定

承認の代理者を設定します。先に行った「準備」により、管理処理メニューから「代理者設定」を操作できるようになっています。

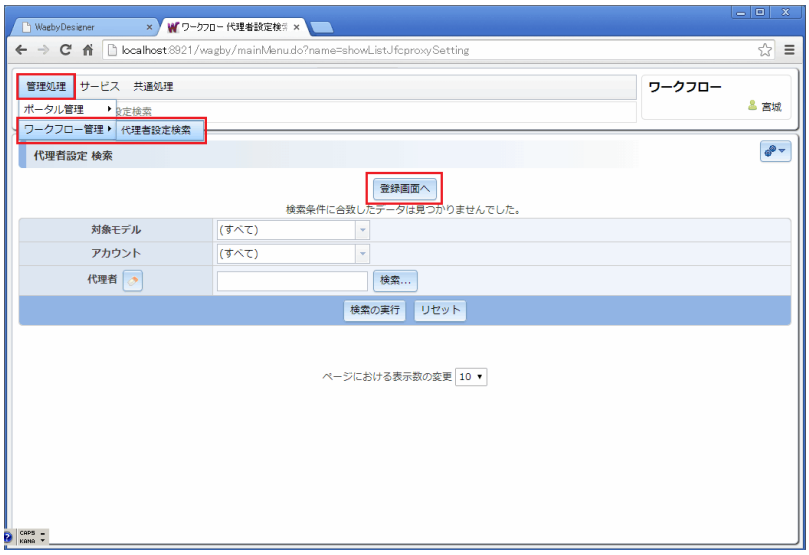


図 代理者設定(1)

代理者設定画面では、対象モデル（どのモデルに紐づくワークフローの処理を代理するのか）と、代理者（個人アカウント）を指定します。

ここでは〝suzuki〟を代理者として設定しました。

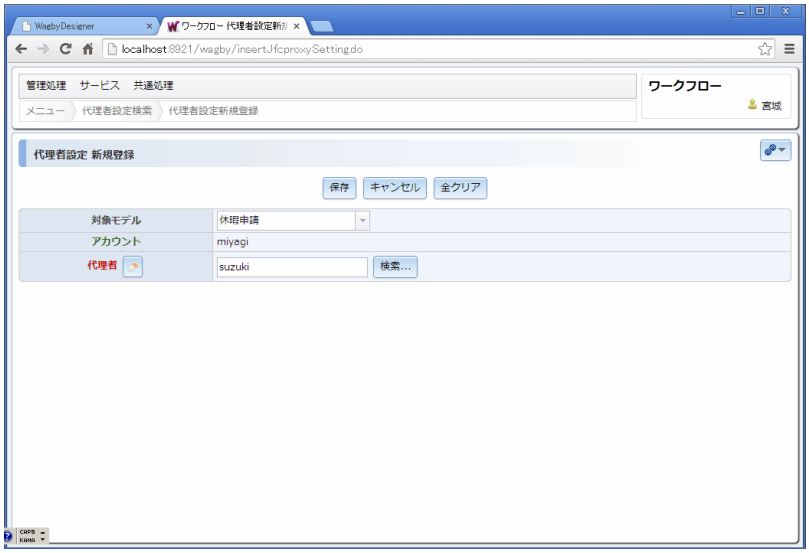


図 代理者設定(2)

代理者を設定すると、自分自身 ("miyagi") は、承認操作が行えなくなります。ボタン群が非表示となります。

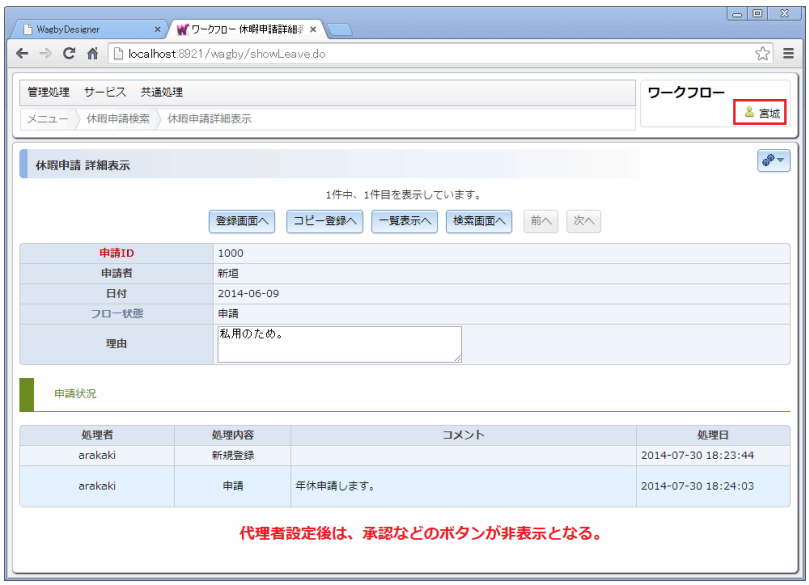


図 代理者設定後はボタンが非表示と鳴る

代理者の画面

代理者と指定された suzuki でログオンします。承認操作が行えるようになっています。

図　代理者による承認操作

[注意] システム管理者が代理者設定を登録・更新する場合、アカウントが編集可能となり、データ所有者が指定できるようになります。このとき、アカウントとデータ所有者は同じになるように設定してください。 システム管理者が代理者設定を登録する場合、初期状態ではアカウント項目とデータ所有者にはシステム管理者のアカウントが設定されています。ここでアカウント項目のみを修正し、アカウント項目とデータ所有者が異なる状態となった場合は、正しく動作しなくなります。

代理者の解除

再び "miyagi" でログオンします。図の代理者設定メニューから、現在の代理者を表示します。このデータを削除することで、代理者の解除を行います。

図　代理者データを削除する

仕様・制約

- 代理者の、さらに代理者を設定することが可能です。ただし代理者を循環させることはできません。上の例では、"miyagi" の代理者を "suzuki" にし、さらに "suzuki" の代理者を "miyagi" に設定することはできません。（そのような設定は行わないようにしてください。）
- グループによる代理承認を行うことができます。

申請したワークフローの状態を確認する

準備

ワークフローを申請できるアカウントに、"フロー状態閲覧" プリンシパルを設定しておきます。

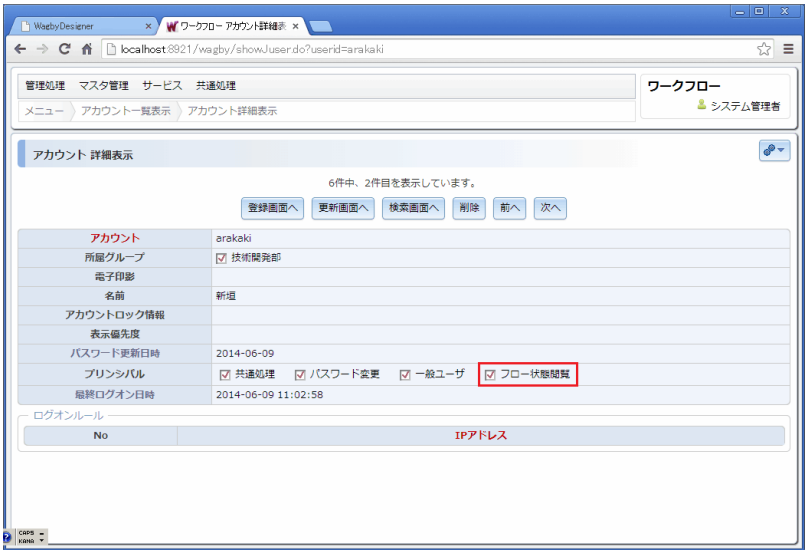


図 申請者にフロー状態閲覧プリンシパルを設定する

実行

休暇申請ワークフローを申請したアカウント（ここでは "arakaki" とする）で、開始フロー状態を検索します。

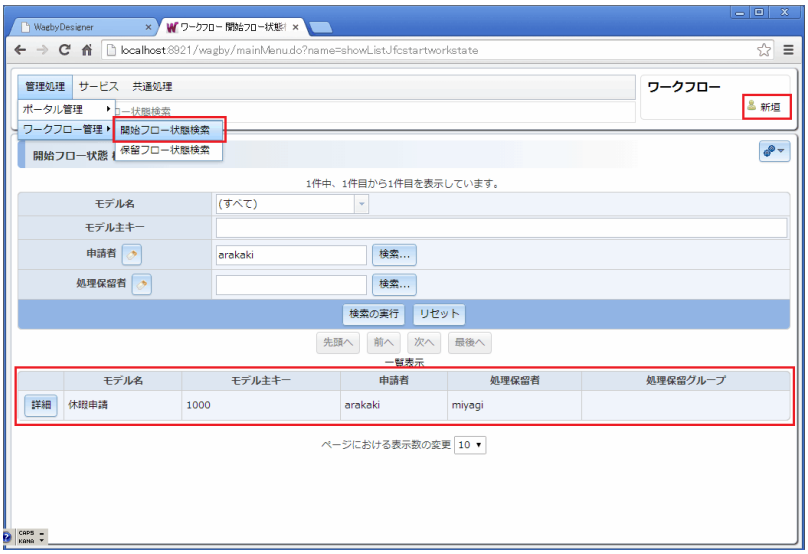


図 開始フロー状態検索

図の画面から、どのモデルのどのデータ（主キーで判断）の「処理保留者」が誰かがわかります。

他者のデータを閲覧する場合（R8.0.4）

自分以外のアカウントが申請者となっているデータを検索したい場合は、図のプリンシパル設定で、さらに〝アカウント閲覧者〟も含めるようにしてください。

カスタマイズ事例 - モデル名以外の検索項目を非表示にする

図の検索画面において、「モデル主キー」「申請者」「処理保留者」を検索条件から非表示にするカスタマイズ例を紹介します。

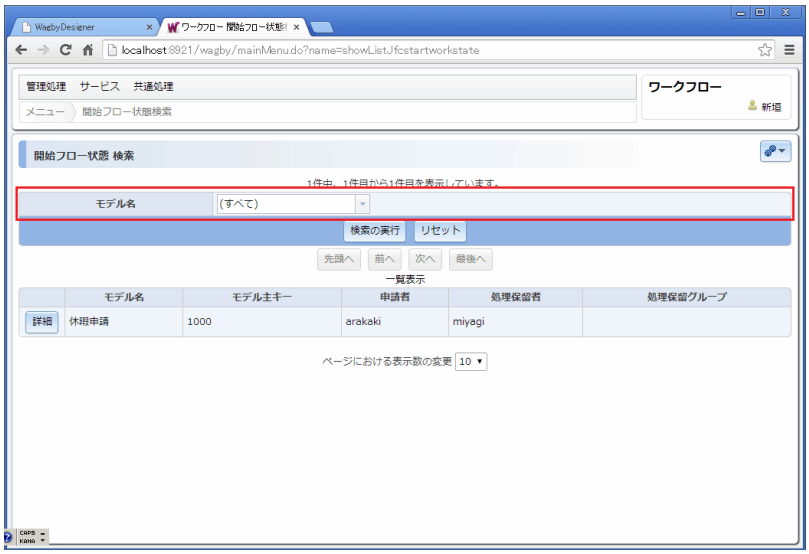


図 検索条件をシンプルにした画面例

定義の変更

「システム」タブを選択します。その中にある「開始フロー状態」モデルを編集します。

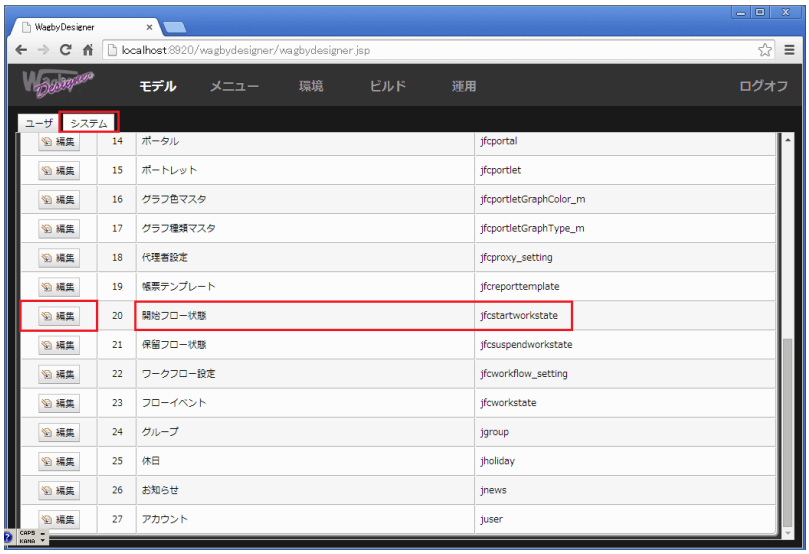


図 開始フロー状態の定義を編集する

システムタブに含まれるモデルは Wagby 内部で利用しています。ここに記載されているカスタマイズ以外は非推奨となりますので、修正作業は慎重に行ってください。

「モデル主キー」「申請者」「処理保留者」はそれぞれ、Wagby内部で検索条件として扱える必要があります。よって、これらの項目の検索機能はそのままに、見た目上、隠し項目とすることで対応します。

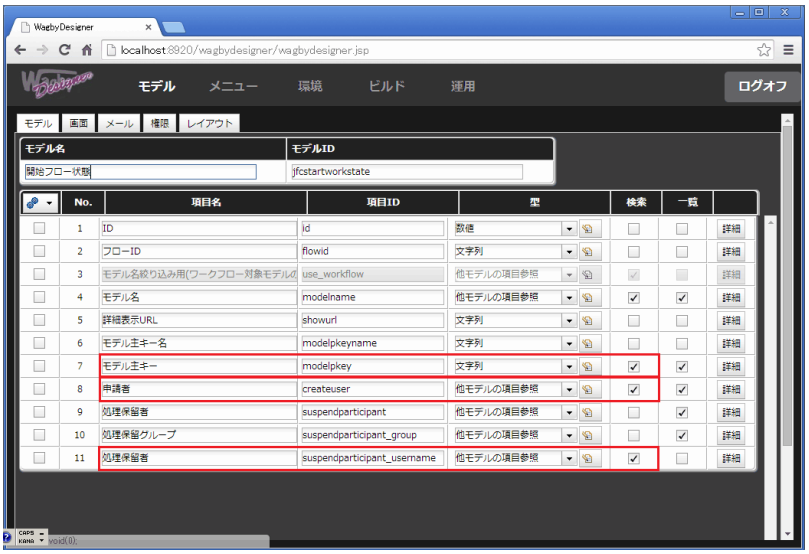


図 開始フローモデルの定義

「モデル主キー」項目の詳細定義を開きます。「検索制御>共通>隠し項目にする」を有効にします。

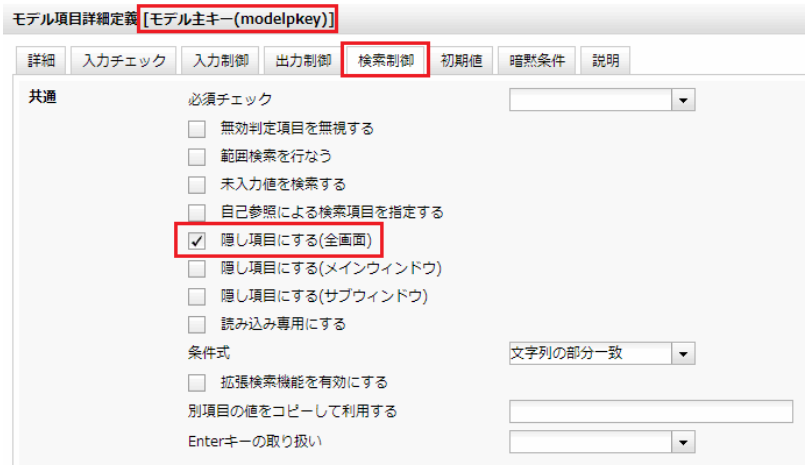


図 モデル主キーの検索項目を隠し項目とする

「申請者」項目の詳細定義を開きます。「検索制御>共通>隠し項目にする」を有効にします。

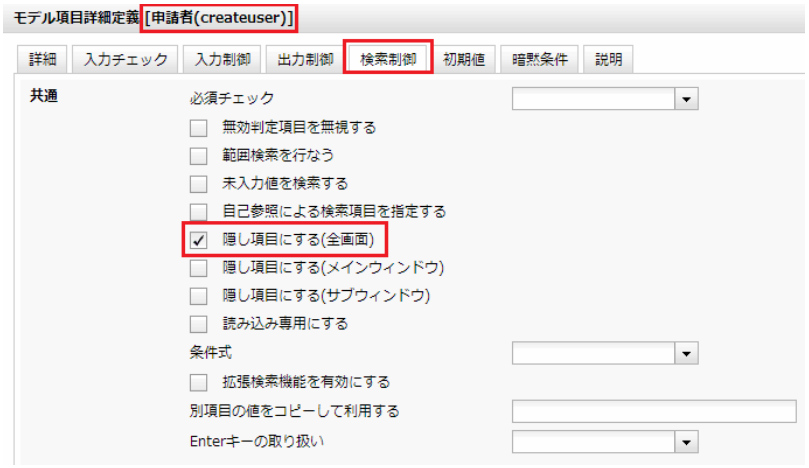


図 申請者の検索項目を隠し項目とする

「処理保留者」項目の詳細定義を開きます。「検索制御>共通>隠し項目にする」を有効にします。

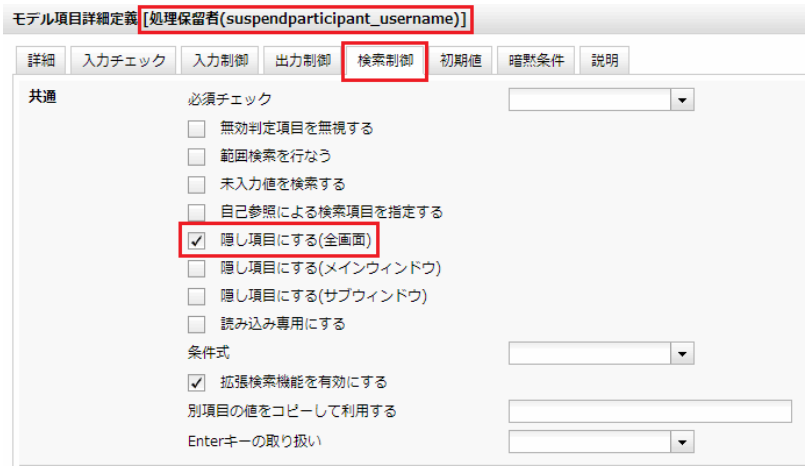


図 処理保留者の検索項目を隠し項目とする

カスタマイズ事例 - 一覧表示項目の変更

一覧表示項目のカスタマイズを行った例を示します。図では「モデル主キー」を隠し、変わって「説明」という項目を付与しています。この説明項目は、休暇申請モデルの「理由」項目を表示するようにしています。

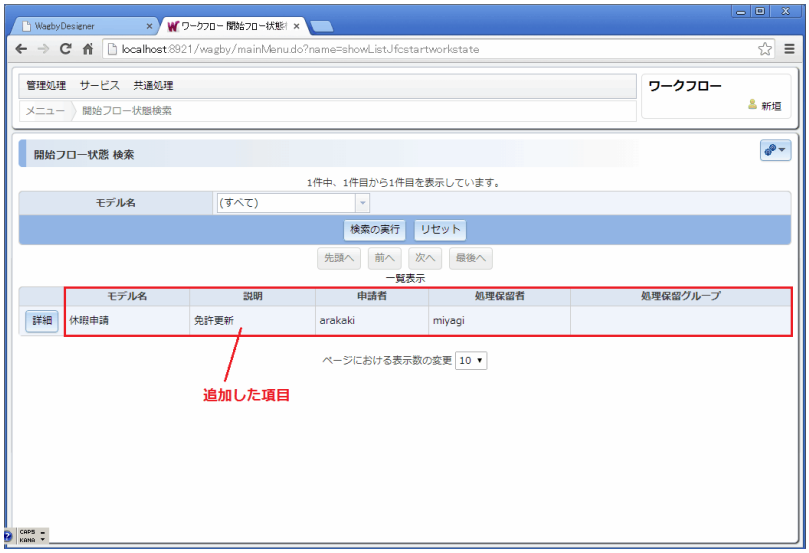


図 一覧表示をカスタマイズした例

定義の変更（1）モデル主キーを隠す

「システム」タブを選択します。その中にある「開始フロー状態」モデルを編集します。

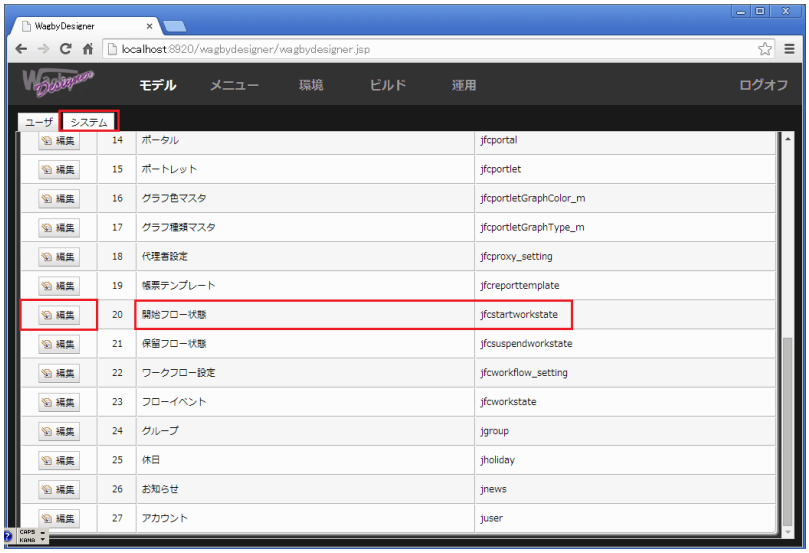


図 開始フロー状態の定義を編集する

システムタブに含まれるモデルは Wagby 内部で利用しています。ここに記載されているカスタマイズ以外は非推奨となりますので、修正作業は慎重に行ってください。

「モデル主キー」項目の詳細定義を開きます。「出力制御>一覧画面>隠し項目にする」を有効にします。

図 モデル主キー項目を隠しとする

(441)

定義の変更（2）説明項目を加える

「開始フロー状態」モデルに「説明」という項目を挿入します。型は文字列とします。

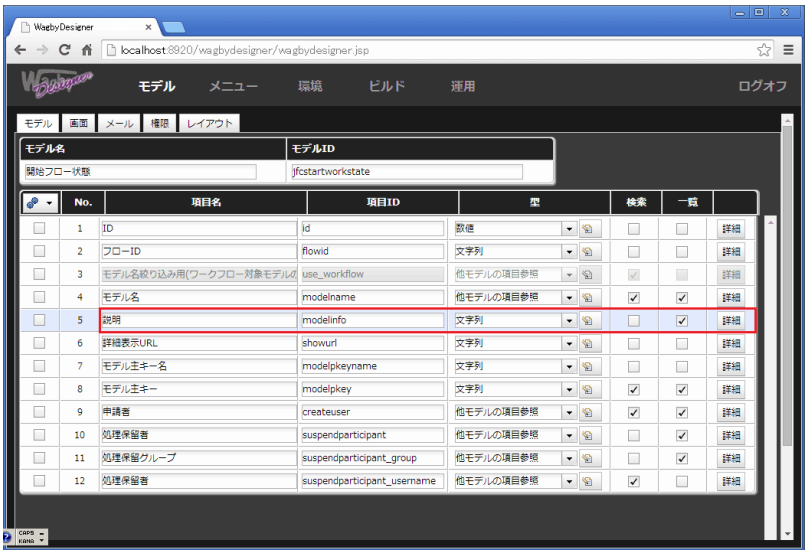


図 説明項目を挿入する

「説明」項目の詳細定義を開きます。「値をデータベースに保存する」設定を解除します。（この項目は常に他のモデルの値が表示されるためです。）

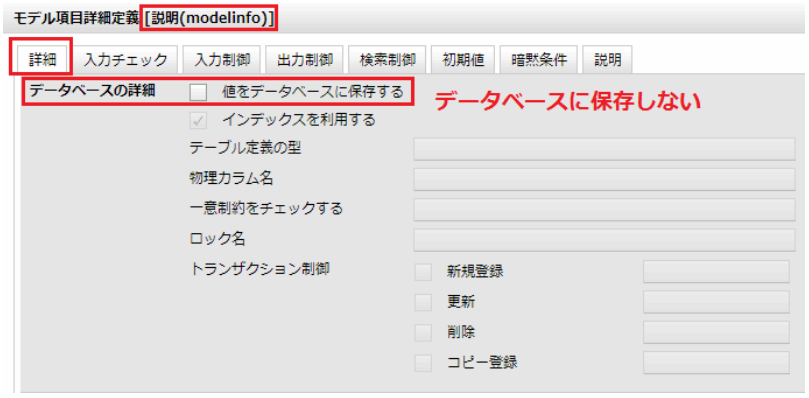


図 データベースに保存する設定を解除する

続いてユーザモデル「休暇申請」の設定を行います。「理由」項目の詳細定義を開きます。

(441) 「モデル主キー」項目は一覧表示に必要です。そのため一覧表示から解除するのではなく、隠し項目としてください。

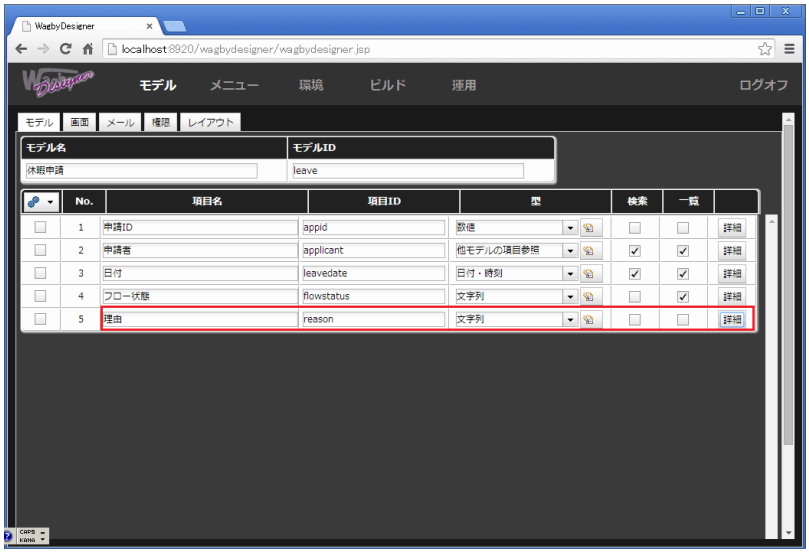


図 休暇申請モデル

「理由」項目の「詳細>ワークフロー」で、「開始フロー状態の一覧項目名」に、図で追加した「説明」項目の項目ID "modelinfo" を指定します。これで紐付けが行われます。

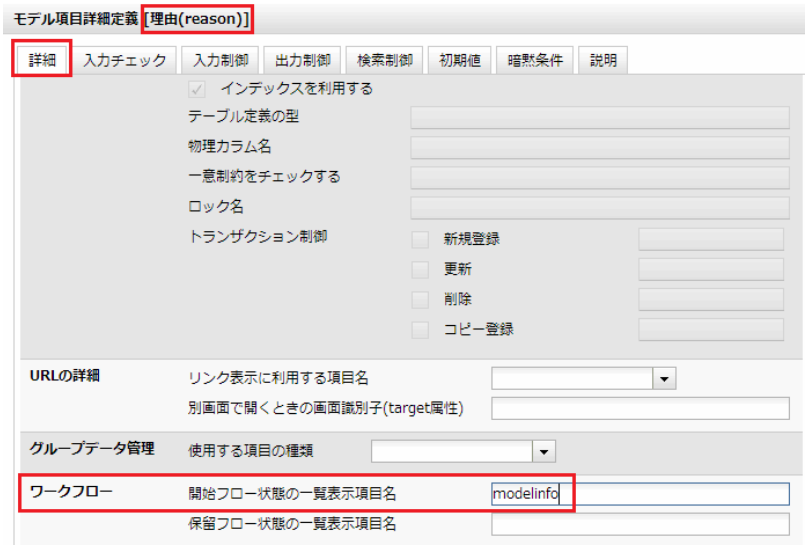


図 開始フロー状態の一覧項目名の設定

開始フロー状態モデルに追加できる（一覧表示用の）項目は1つだけです。ここにさまざまな情報を含めたい場合、元となるモデル（ここでは「休暇申請」モデル）に自動計算式で複数の項目の値を（1つの項目に）まとめ、この項目と紐付けるようにします。

設定上の制約

- ここで追加した項目（開始フロー状態の一覧項目名「modelinfo」）を検索条件に指定することはできません。
- 項目の型は文字列型のみとなります。文字列以外を転記させたい場合、当該モデル（ここでは休暇申請モデル）に転記のための文字列型項目を別途用意し、これを経由させるとよいでしょう。これを隠し項目として用意します。

カスタマイズ事例 - 開始フロー状態専用のプリンシパルを用意する

例

「開始フロー状態検索」を操作する専用のプリンシパルを用意します。申請者と承認者それぞれでメニューを分ける運用が行えます。図では、開始フロー状態検索画面のみ利用できるようになっています。(442)

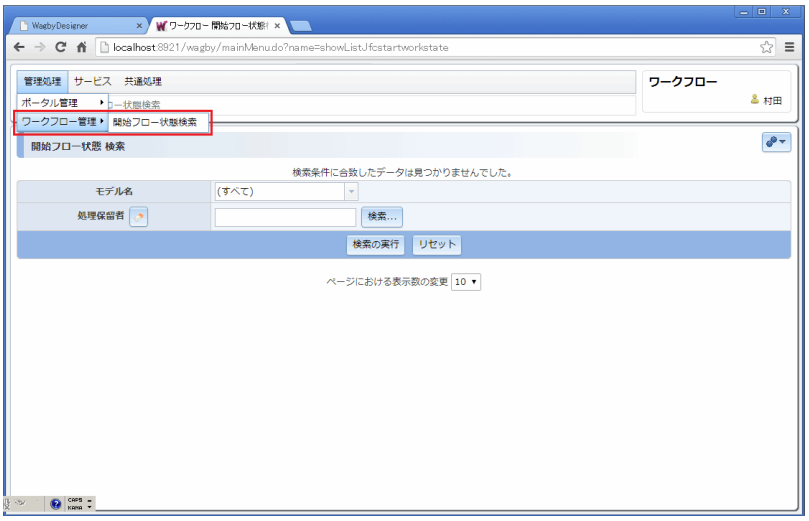


図 メニューを分割し、開始フロー状態検索のみ利用できるように設定

定義の変更

「システム」タブを選択します。その中にある「開始フロー状態」モデルを編集します。

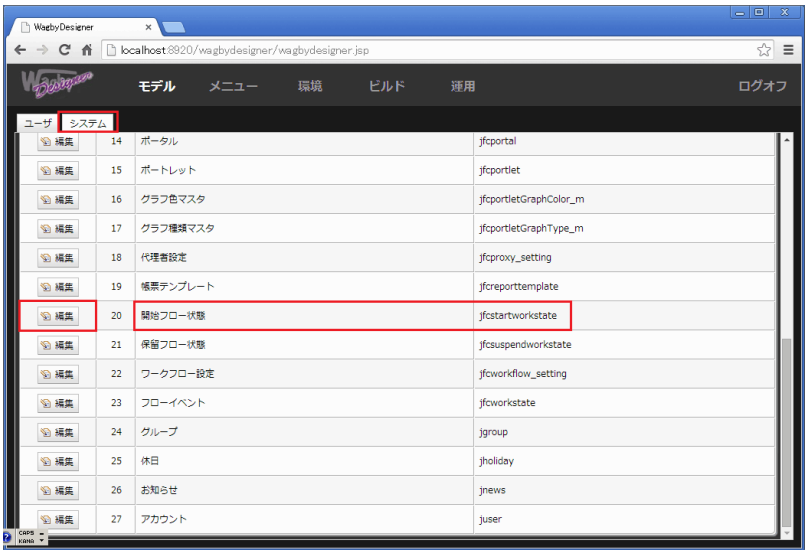


図 開始フロー状態の定義を編集する

システムタブに含まれるモデルは Wagby 内部で利用しています。ここに記載されているカスタマイズ以外は非推奨となりますので、修正作業は慎重に行ってください。

「権限」タブを開きます。新しいプリンシパルを新規で用意します。

(442) 図と比較すると「保留フロー状態検索」画面が利用できないようになっていることがわかります。

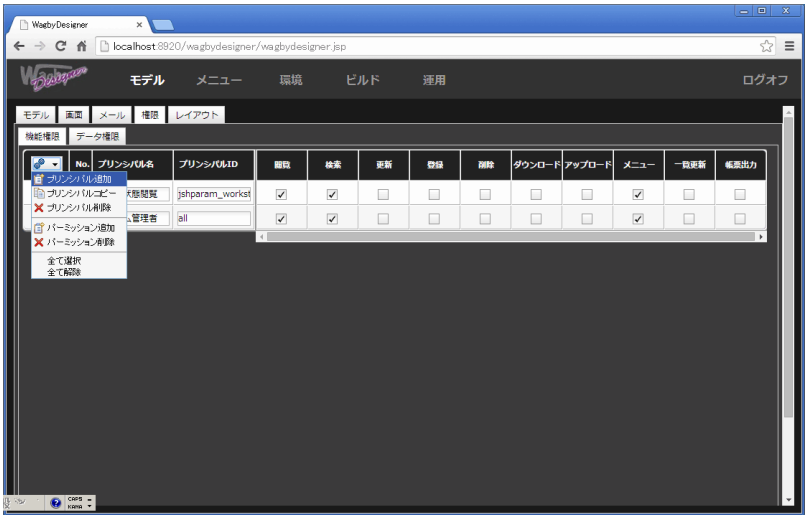


図 新しいプリンシパルを用意する

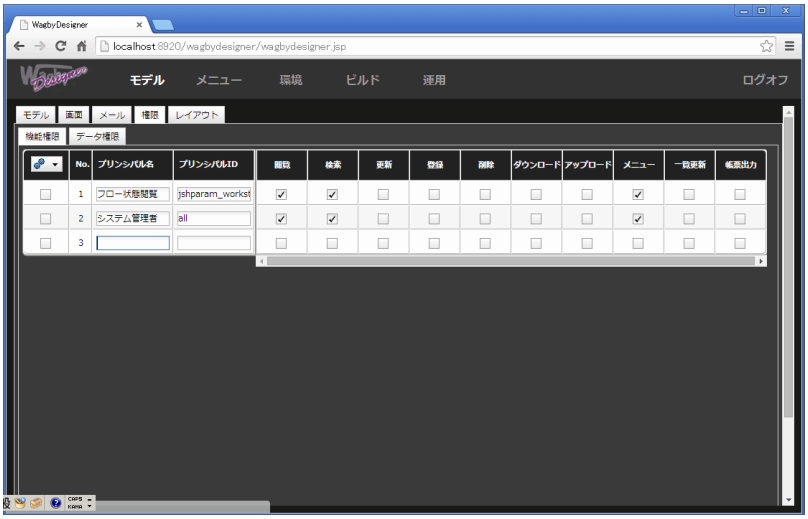


図 プリンシパルの入力欄

「開始フロー状態閲覧」プリンシパルを用意します。[443]

プリンシパル名	プリンシパルID	有効な権限
開始フロー状態閲覧	jshparam_startworkstate_show	閲覧、検索、メニュー

[443] プリンシパル名、プリンシパルIDは任意に名付けることができます。（プリンシパルIDはシステム全体を通してユニークである、すなわち重複しないようにしてください。）

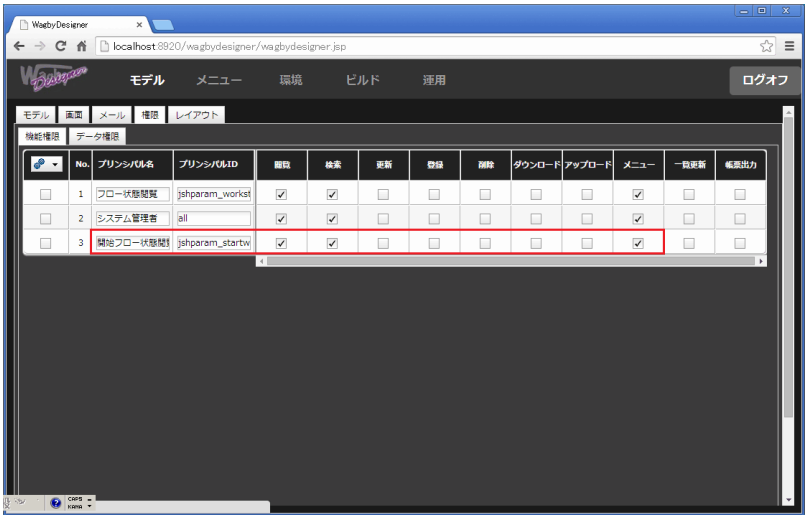


図 開始フロー状態閲覧プリンシパルを用意する

プリンシパルの割り当て

標準で用意されていた「フロー状態閲覧」プリンシパルは、開始フローと保留フローの両方を閲覧できるものでした。今回、標準のプリンシパルを不活性とし、新しく用意した「開始フロー状態閲覧」プリンシパルを使うようにします。

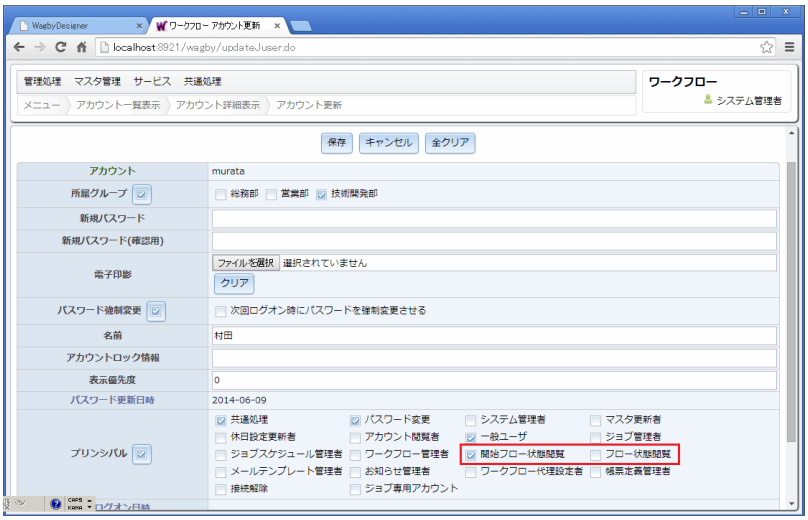


図 「開始フロー状態閲覧」プリンシパルを用いる(1)

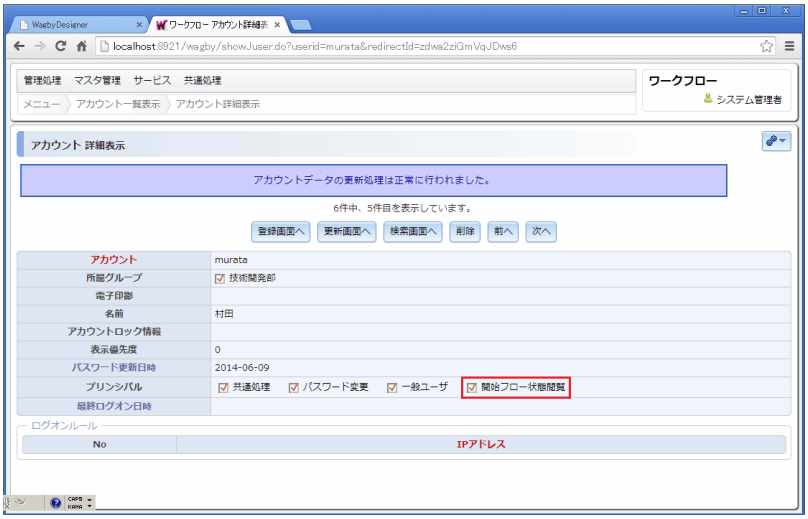


図 「開始フロー状態閲覧」プリンシパルを用いる(2)

仕様・制約

- 開始フロー状態の一覧表示で、並び替え（ソート順制御）を指定することはできません。
- 開始フロー状態の「ページにおける表示数の設定」や「ページネーション」の変更は、カスタマイズでの対応になります。

承認すべきワークフローを確認する

準備

ワークフローを承認できるアカウントに、”フロー状態閲覧” プリンシパルを設定しておきます。

図 承認者にフロー状態閲覧プリンシパルを設定する

実行

休暇申請ワークフローの承認アカウントで、保留フロー状態を検索します。

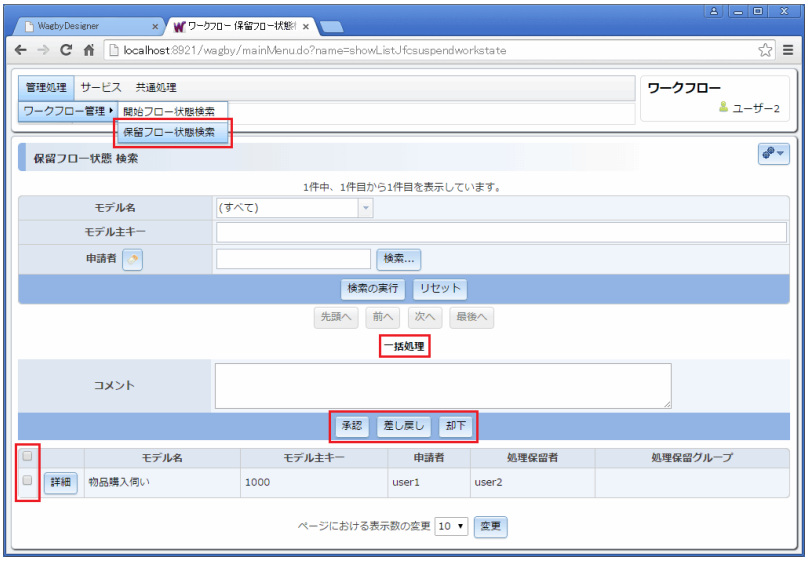


図 保留フロー状態検索

図の画面から、どのモデルのどのデータ（主キーで判断）を承認処理するべきかがわかります。

他者のデータを閲覧する場合（R8.0.4）

自分以外のアカウントが申請者となっているデータを検索したい場合は、図のプリンシパル設定で、さらに〝アカウント閲覧者〟も含めるようにしてください。

一括処理

図において、一覧表示欄の左側のチェックボックスを有効にして「承認」（または「差し戻し」「却下」）ボタンを押下すると一括で承認処理を行なうことができます。コメント欄に入力した文字列が、選択した全ての対象データに設定されます。

保留フローポートレットを利用する

保留フローを確認するためのポートレットを利用できます。自分が承認すべきワークフロー一覧をポータル画面に表示させることができます。

カスタマイズ事例 － モデル名以外の検索項目を非表示にする

図の検索画面において、「モデル主キー」「申請者」を検索条件から非表示にするカスタマイズ例を紹介します。

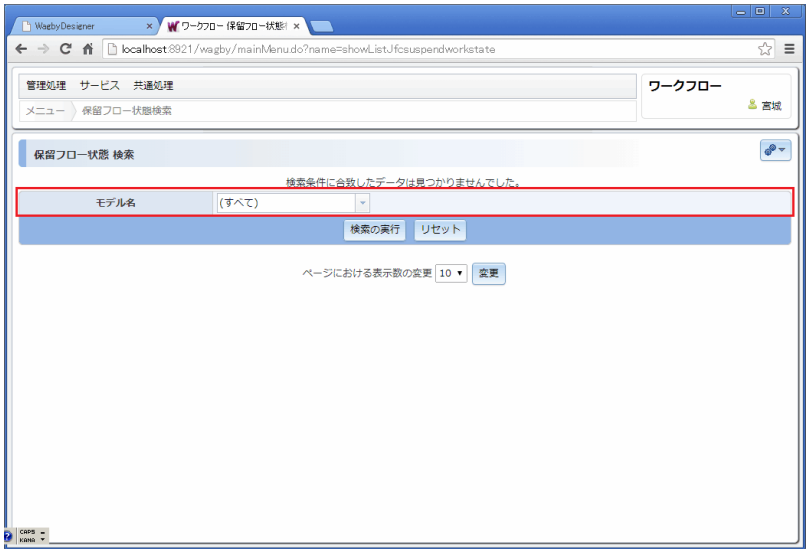


図 検索条件をシンプルにした画面例

定義の変更

「システム」タブを選択します。その中にある「保留フロー状態」モデルを編集します。

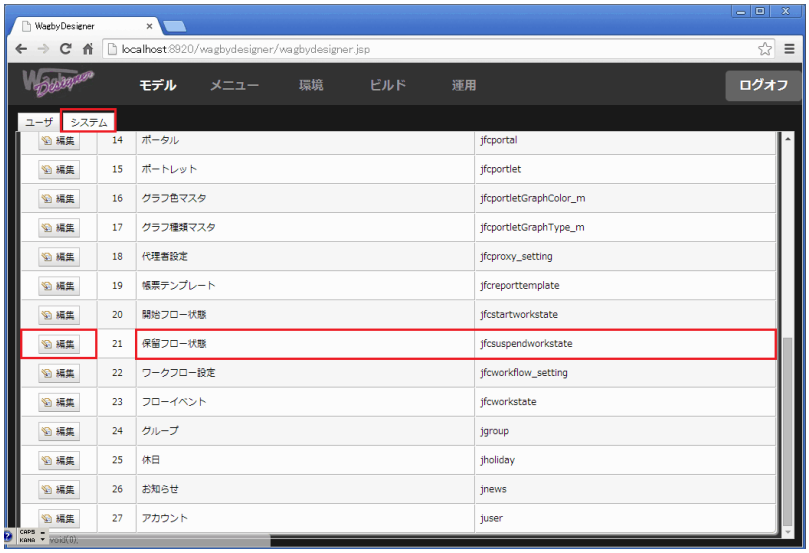


図 保留フロー状態の定義を編集する

システムタブに含まれるモデルは Wagby 内部で利用しています。ここに記載されているカスタマイズ以外は非推奨となりますので、修正作業は慎重に行ってください。

「モデル主キー」「申請者」はそれぞれ、Wagby内部で検索条件として扱える必要があります。よって、これらの項目の検索機能はそのままに、見た目上、隠し項目とすることで対応します。

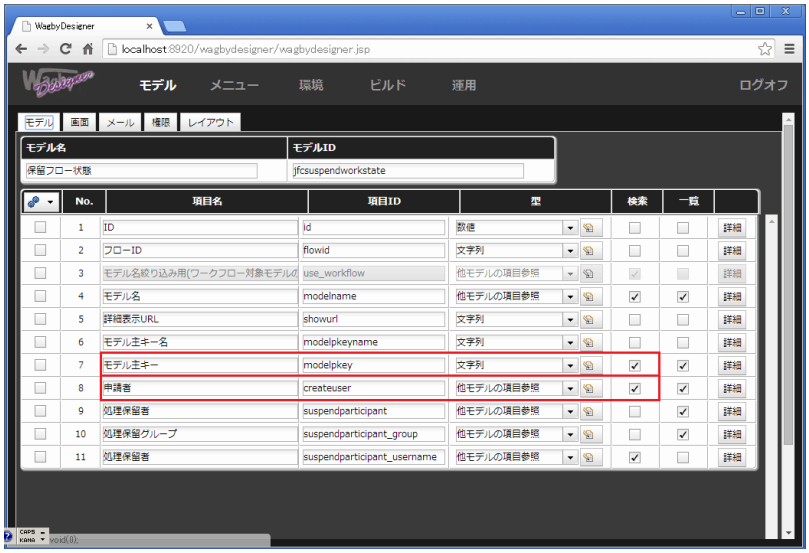


図 保留フローモデルの定義

「モデル主キー」項目の詳細定義を開きます。「検索制御>共通>隠し項目にする」を有効にします。

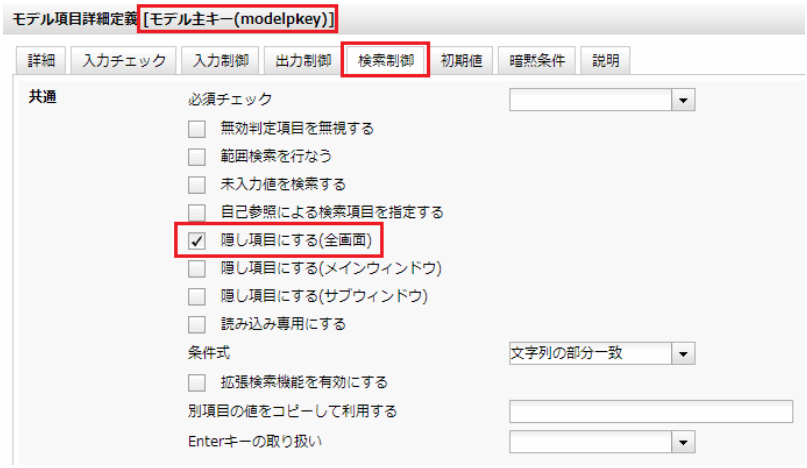


図 モデル主キーの検索項目を隠し項目とする

「申請者」項目の詳細定義を開きます。「検索制御>共通>隠し項目にする」を有効にします。

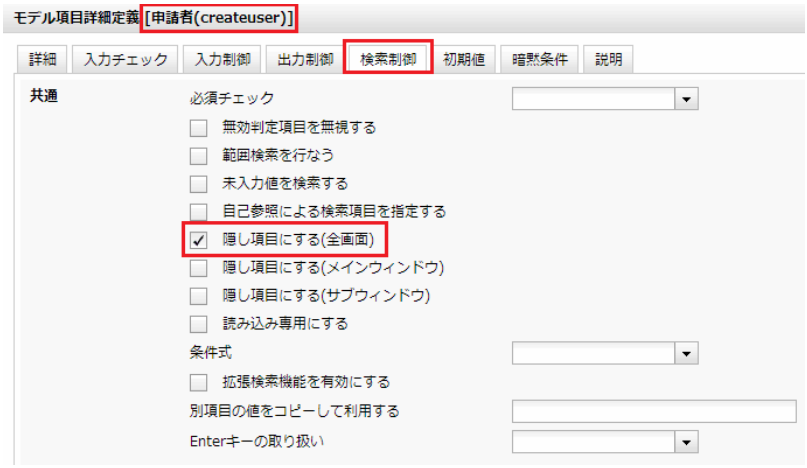


図 申請者の検索項目を隠し項目とする

カスタマイズ事例 - 一覧表示項目の変更

一覧表示項目のカスタマイズを行った例を示します。図では「モデル主キー」を隠し、変わって「説明」という項目を付与しています。この説明項目は、休暇申請モデルの「理由」項目を表示するようにしています。

図 一覧表示をカスタマイズした例

定義の変更（1）モデル主キーを隠す

「システム」タブを選択します。その中にある「保留フロー状態」モデルを編集します。

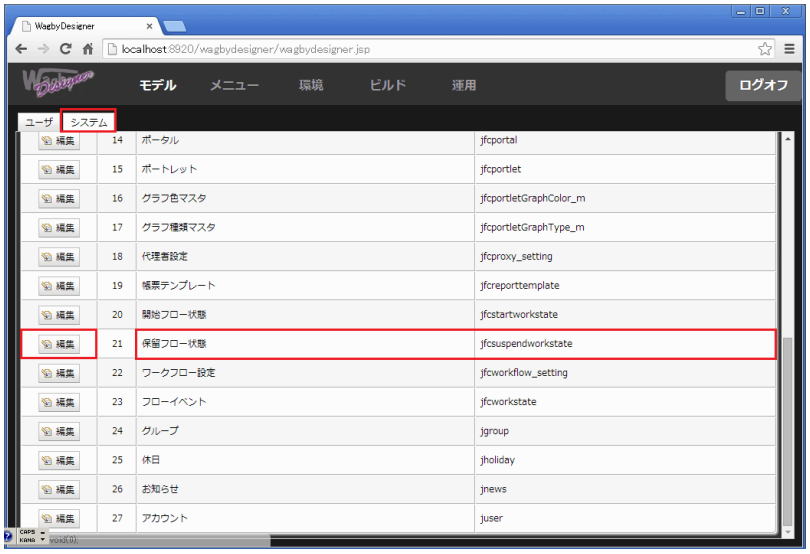


図 保留フロー状態の定義を編集する

システムタブに含まれるモデルは Wagby 内部で利用しています。ここに記載されているカスタマイズ以外は非推奨となりますので、修正作業は慎重に行ってください。

「モデル主キー」項目の詳細定義を開きます。「出力制御>一覧画面>隠し項目にする」を有効にします。

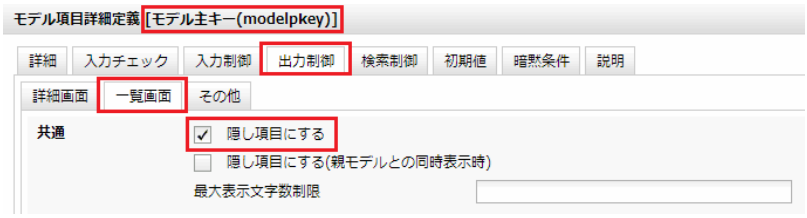


図 モデル主キー項目を隠しとする

(444)

定義の変更（2）説明項目を加える

「保留フロー状態」モデルに「説明」という項目を挿入します。型は文字列とします。

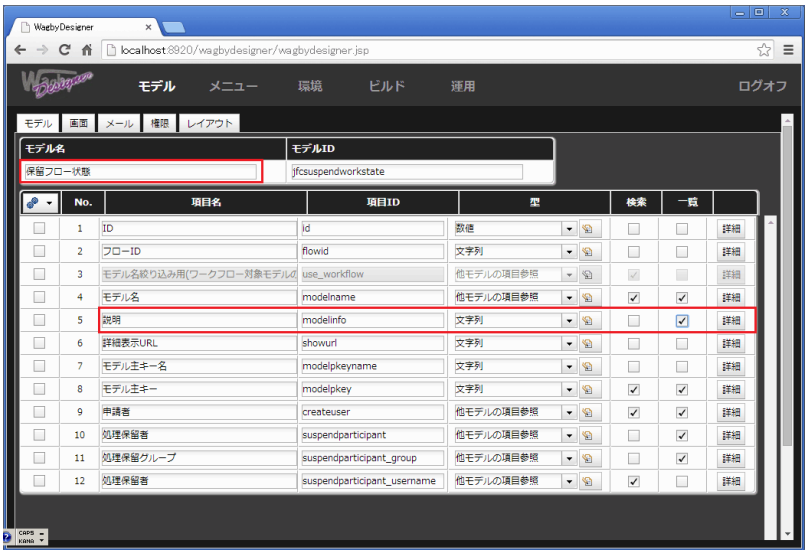


図 説明項目を挿入する

「説明」項目の詳細定義を開きます。「値をデータベースに保存する」設定を解除します。（この項目は常に他のモデルの値が表示されるためです。）

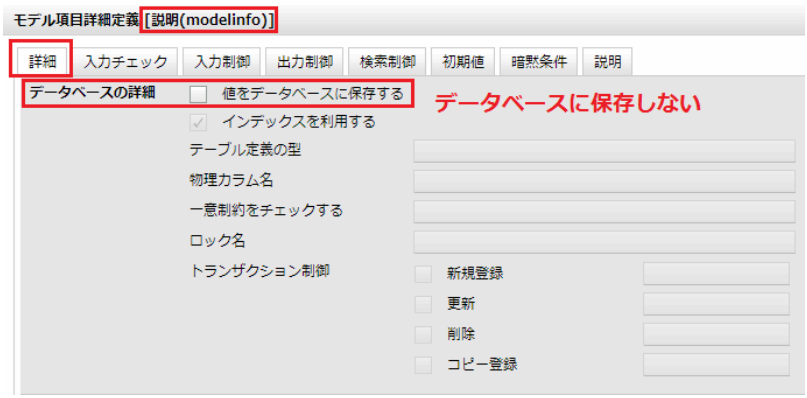


図 データベースに保存する設定を解除する

続いてユーザモデル「休暇申請」の設定を行います。「理由」項目の詳細定義を開きます。

(444) 「モデル主キー」項目は一覧表示に必要です。そのため一覧表示から解除するのではなく、隠し項目としてください。

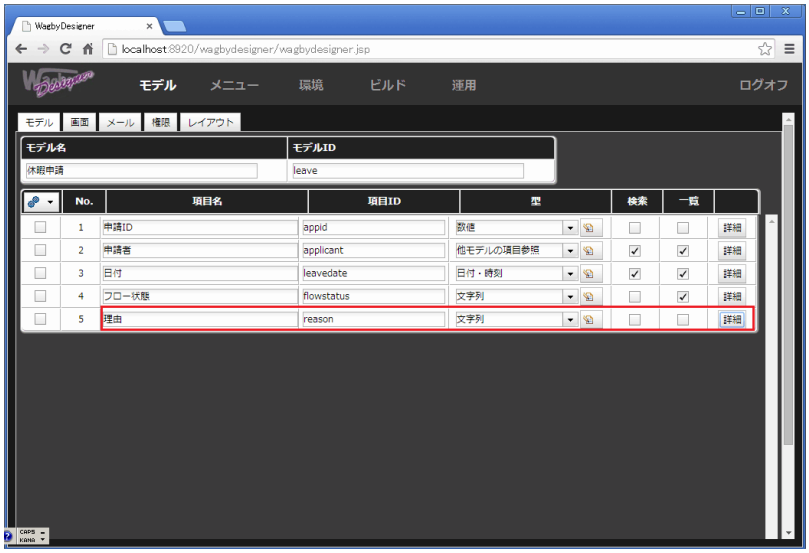


図 休暇申請モデル

「理由」項目の「詳細>ワークフロー」で、「保留フロー状態の一覧項目名」に、図で追加した「説明」項目の項目ID "modelinfo" を指定します。これで紐付けが行われます。

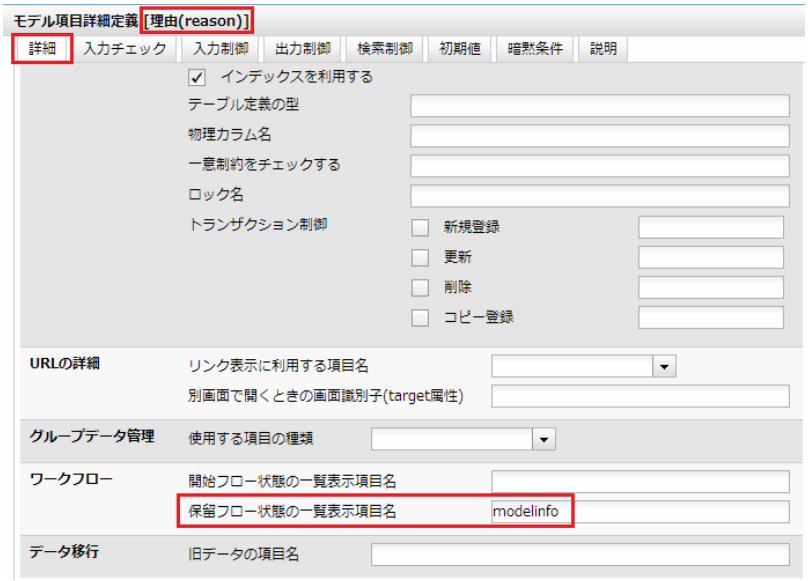


図 保留フロー状態の一覧項目名の設定

保留フロー状態モデルに追加できる（一覧表示用の）項目は1つだけです。ここにさまざまな情報を含めたい場合、元となるモデル（ここでは「休暇申請」モデル）に自動計算式で複数の項目の値を（1つの項目に）まとめ、この項目と紐付けるようにします。

開始フロー状態、保留フロー状態の両方に対応させる

図のように、一つの項目を開始フロー状態の一覧表示項目と、保留フロー状態の一覧表示項目の両方に用いることもできます。

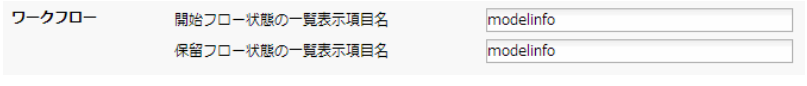


図 保留フロー状態の一覧項目名の設定

設定上の制約

- ここで追加した項目（開始フロー状態の一覧項目名「modelinfo」）を検索条件に指定することはできません。
- 項目の型は文字列型のみが利用できます。文字列以外を転記させたい場合、当該モデル（ここでは休暇申請モデル）に転記のための文字列型項目を別途用意し、これを経由させるとよいでしょう。例：元が日付型項目だった場合は、関数を使って文字列型に変換しておくとします。これを隠し項目として用意します。

カスタマイズ事例 - 保留フロー状態専用のプリンシパルを用意する

例

「保留フロー状態検索」を操作する専用のプリンシパルを用意します。申請者と承認者それぞれでメニューを分ける運用が行えます。図では、保留フロー状態検索画面のみ利用できるようになっています。(445)

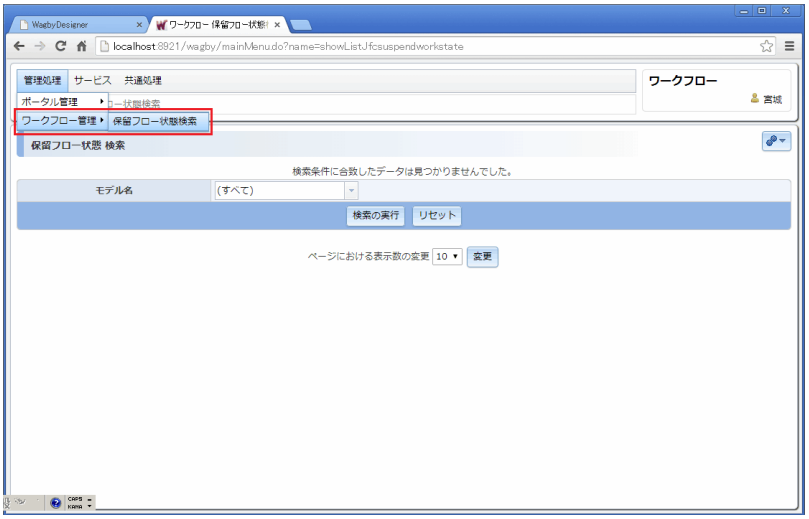


図 メニューを分割し、保留フロー状態検索のみ利用できるように設定

定義の変更

「システム」タブを選択します。その中にある「保留フロー状態」モデルを編集します。

(445) 図と比較すると「開始フロー状態検索」画面が利用できないようになっていることがわかります。

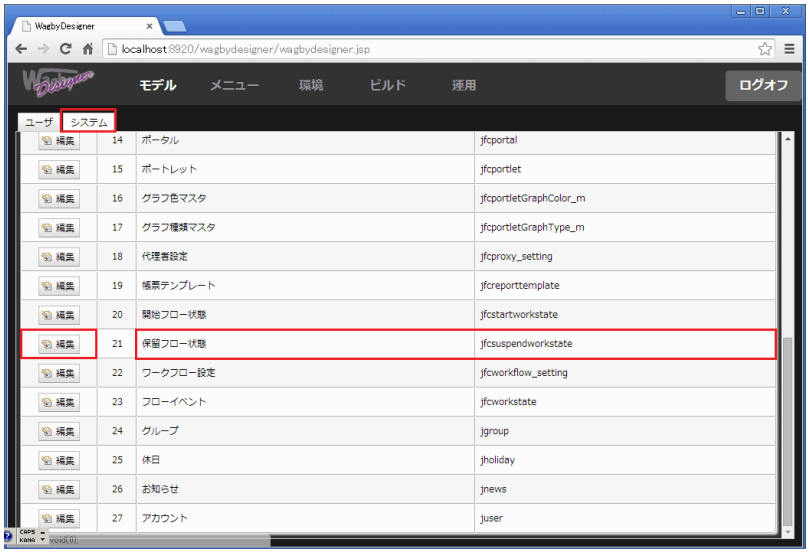


図 保留フロー状態の定義を編集する

システムタブに含まれるモデルは Wagby 内部で利用しています。ここに記載されているカスタマイズ以外は非推奨となりますので、修正作業は慎重に行ってください。

「権限」タブを開きます。新しいプリンシパルを新規で用意します。

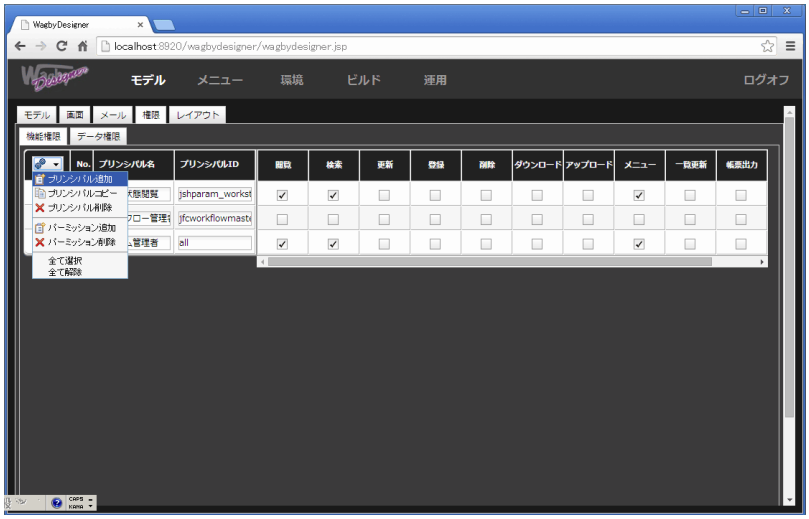


図 新しいプリンシパルを用意する

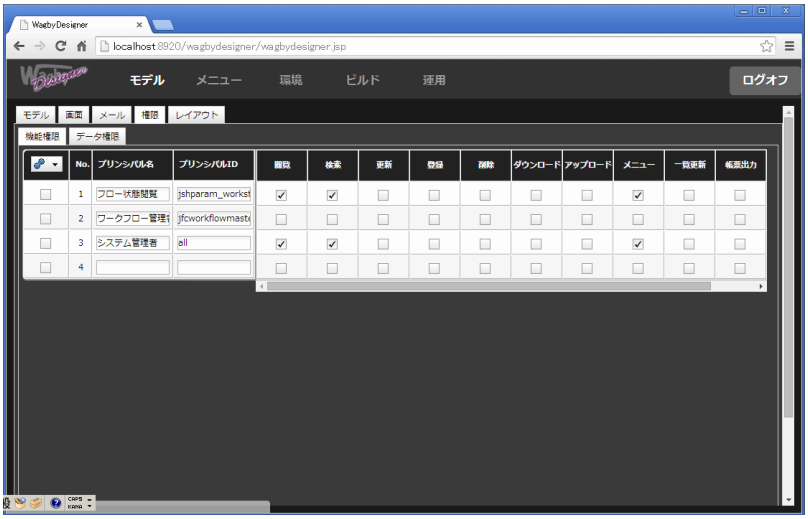


図 プリンシパルの入力欄

「保留フロー状態閲覧」プリンシパルを用意します。[446]

プリンシパル名	プリンシパルID	有効な権限
保留フロー状態閲覧	jshparam_suspendworkstate_show	閲覧、検索、メニュー

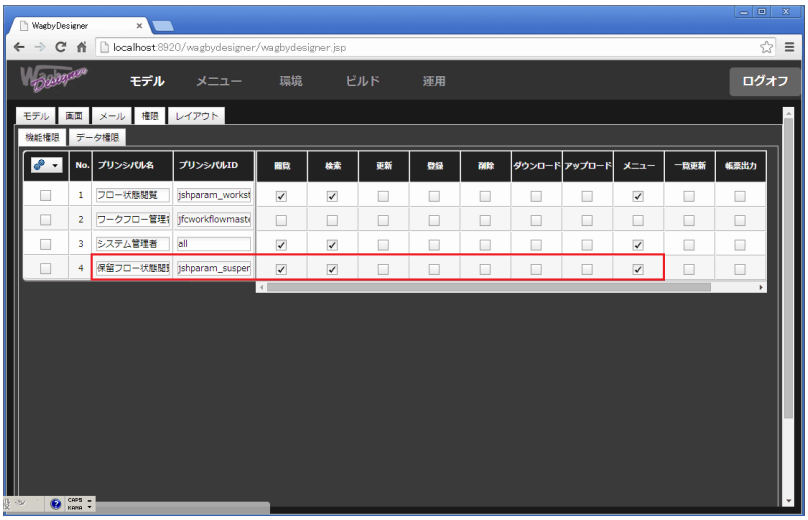


図 保留フロー状態閲覧プリンシパルを用意する

プリンシパルの割り当て

標準で用意されていた「フロー状態閲覧」プリンシパルは、開始フローと保留フローの両方を閲覧できるものでした。今回、標準のプリンシパルを不活性とし、新しく用意した「保留フロー状態閲覧」プリンシパルを使うようにします。

(446) プリンシパル名、プリンシパルIDは任意に名付けることができます。（プリンシパルIDはシステム全体を通してユニークである、すなわち重複しないようにしてください。）

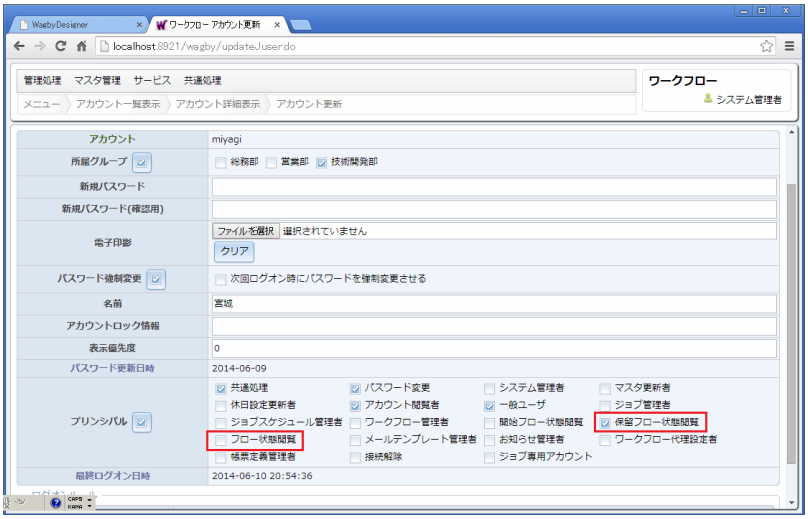


図 「保留フロー状態閲覧」プリンシパルを用いる(1)

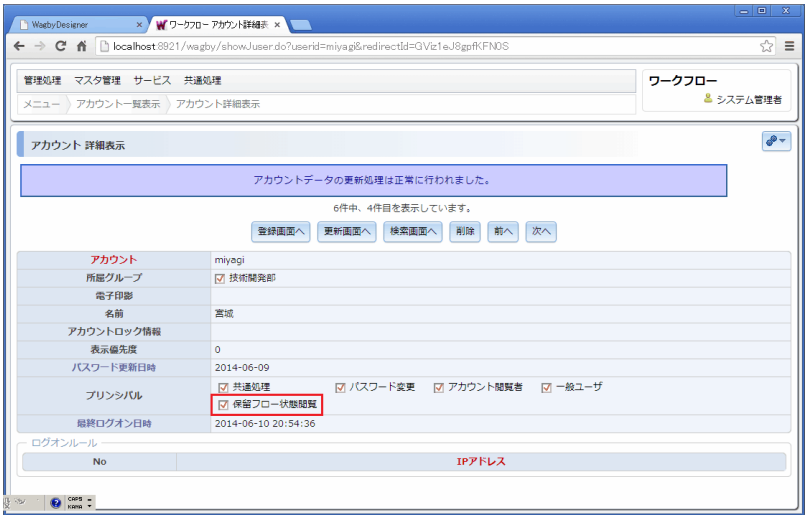


図 「保留フロー状態閲覧」プリンシパルを用いる(2)

カスタマイズ事例 - 却下ボタンを非表示にする

次のスクリプトを jfcsuspendworkstate/MySwhoListJfcsuspendworkstate.js に記述してください。保留フロー状態の検索画面の「却下ボタン」を非表示にすることができます。

```
require(["dojo/dom-construct", "dojo/query"], function(domConstruct, query) {
 query("button[name='btnProcessAllReject']").forEach(function(node) {
   domConstruct.destroy(node);
 });
});
```

仕様・制約

- 本機能は現在、グループ権限に対応していません。将来のWagbyで改善します。
- 保留フロー状態の一覧表示で、並び替え（ソート順制御）を指定することはできません。
- 保留フロー状態の一覧表示で、一括削除、一括メール送信、一括帳票、一括更新の処理を追加することはできません。
- 開始フロー状態の「ページにおける表示数の設定」や「ページネーション」の変更は、カスタマイズでの対応になります。

モデルのワークフロー状態で検索する

例

ワークフローを適用したモデルの検索画面で、「ワークフロー状態」を用いた検索を行うことができます。

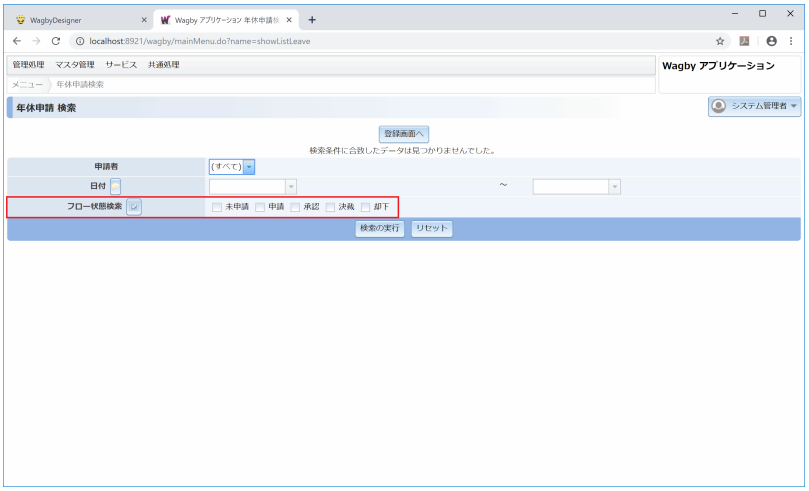


図 ワークフロー状態で検索を行う(1)

「決裁」されたデータのみを絞り込んだ例です。

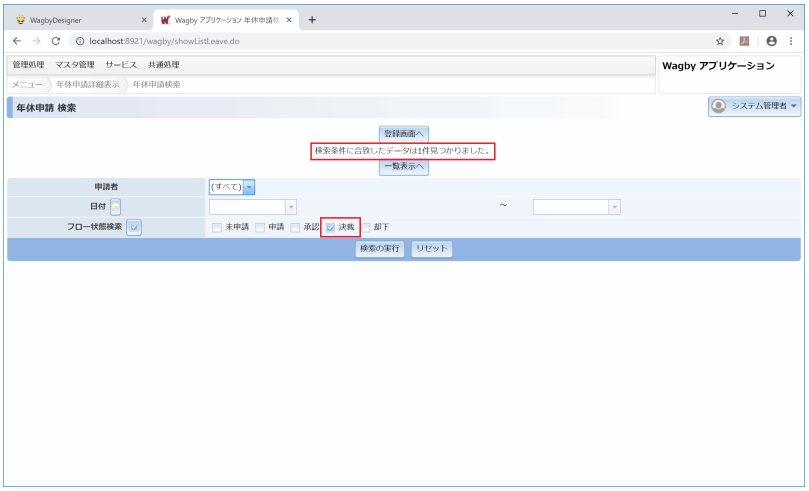


図 ワークフロー状態で検索を行う(2)

フロー状態は4パターンです。「未申請」「申請・承認」「決裁」「却下」で検索できます。

(447)

定義方法

「フロー状態検索」という項目を用意します。（項目名は任意です。）

(447) ワークフローで「取り消し」を行なったデータは、フロー状態検索で「却下」でヒットします。（「取り消し」のみの選択肢は用意されていないため、「却下」での検索は「却下」と「取り消し」の両方がヒットすることになります。）

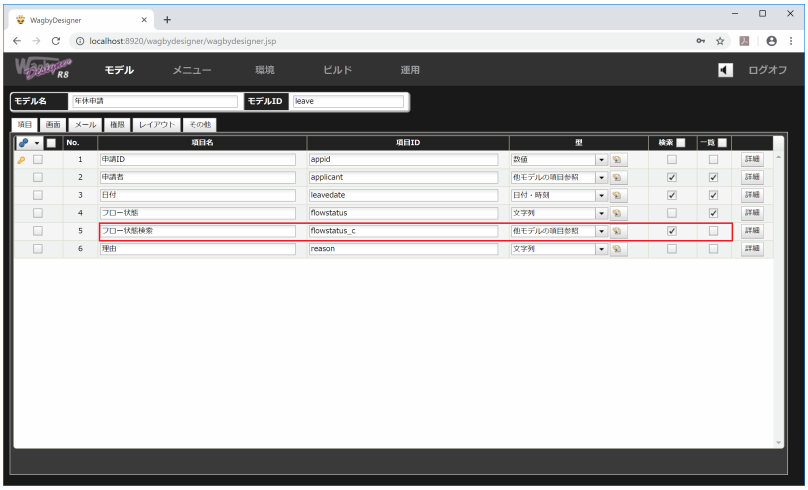


図 フロー状態検索項目を追加する

参照先モデルを〝フロー状態検索項目（jfcflowstatus_search_m)〟とします。これはシステムが標準で提供するモデルです。

入力タイプも選べますが、複数の状態を指定して検索できるため、チェックボックスの利用を推奨します。

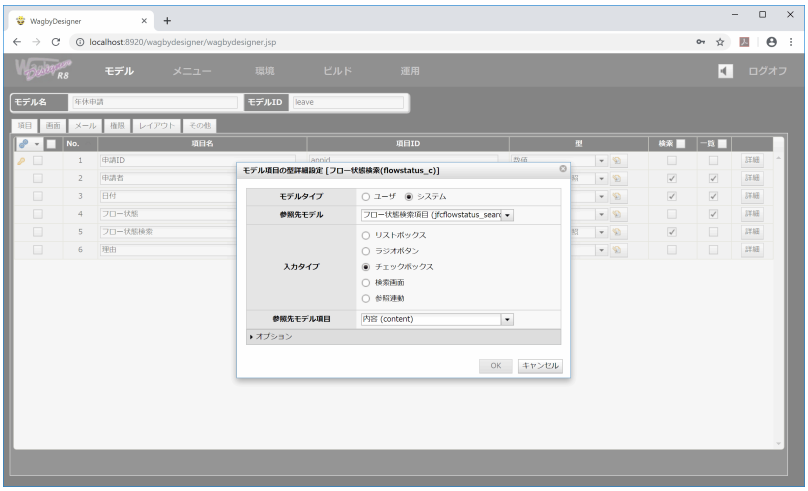


図 参照先モデルの設定

この項目の「検索制御 〉 ワークフロー 〉 フロー状態で検索を行う」を有効にします。

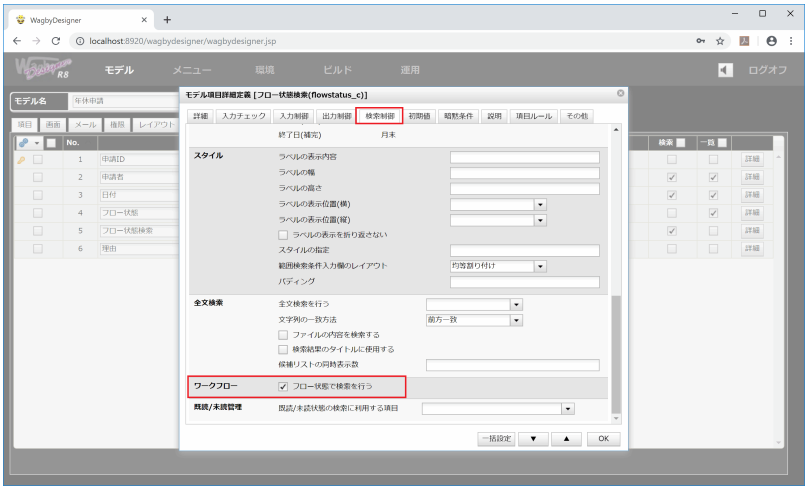


図 フロー状態で検索を行う

「入力制御 > モデル参照 > チェックボックス・ラジオボタンの折り返し数」を "5" とします。これは一行におさまるように表示するためです。

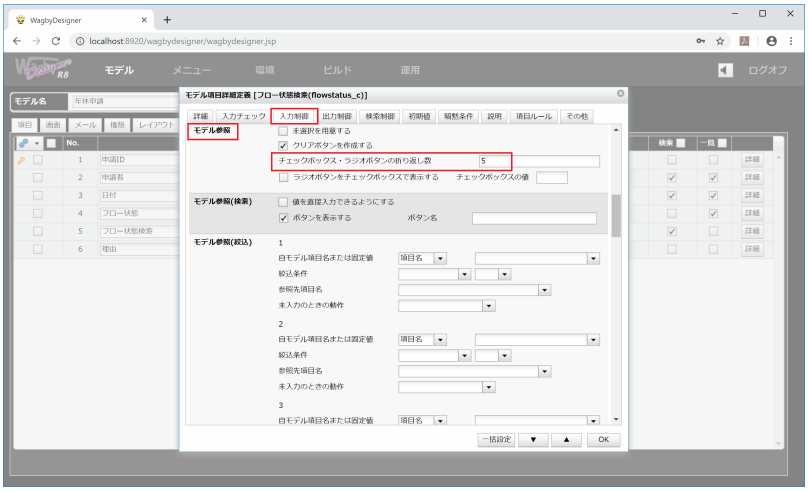


図 チェックボックス・ラジオボタンの折り返し数を5とする

サブモデルを用いて特定のフロー状態のみ検索させる

ここでは、決裁されたデータのみを検索・表示する方法を説明します。このためにサブモデルを使用します。

サブモデルを作成する

「休暇申請」モデルのサブモデルを作成します。はじめに、対象モデルを選択します。

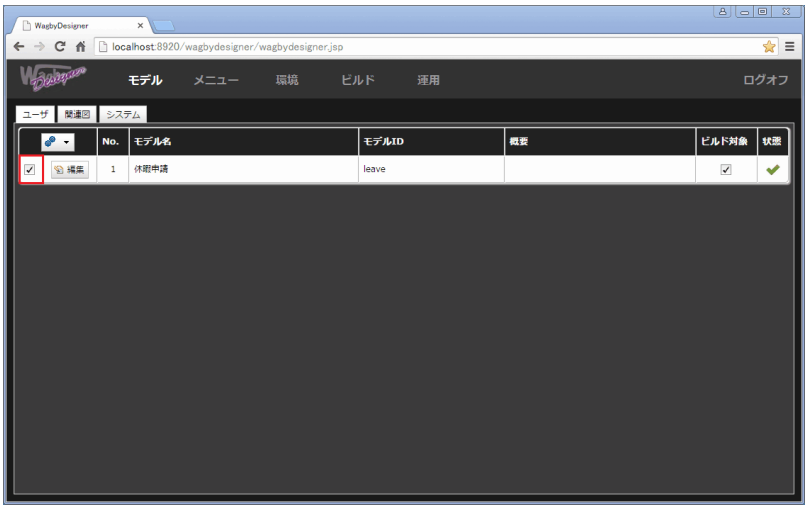


図 サブモデルを作成するモデルを選択する

ギアアイコンから「サブモデル」を選択します。

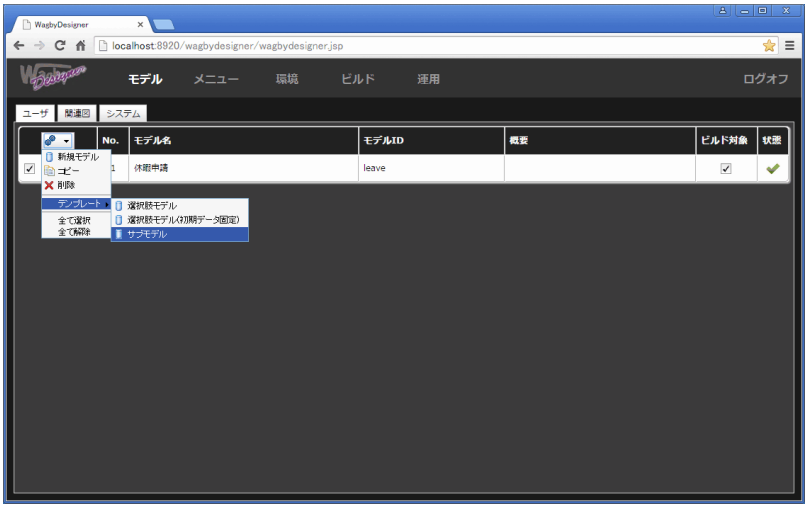


図 サブモデルの作成を選ぶ

サブモデルが用意されました。

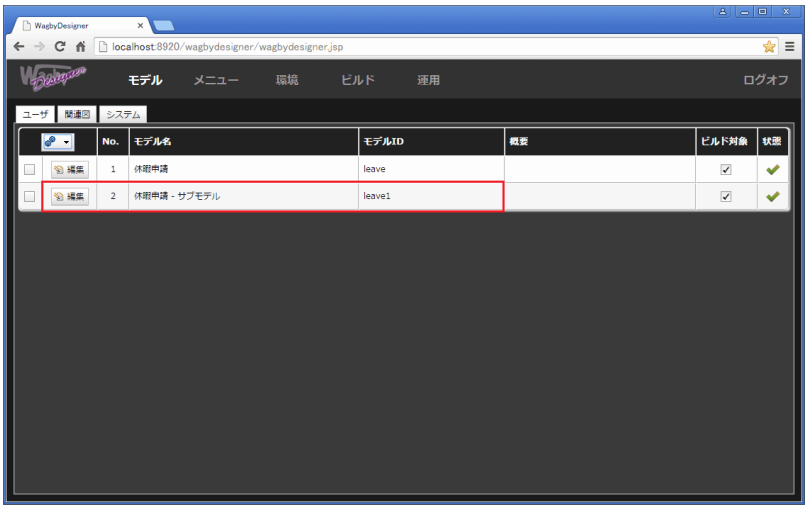


図 サブモデルが作成された

サブモデルの場合、画面にメインモデル情報が明記されています。

図 サブモデルのモデル項目一覧画面

サブモデルのワークフロー設定を解除します。このサブモデルの役割は、あくまでもフロー状態の表示、検索のみに特化したものです。

図 サブモデルではワークフロー設定を解除する

項目の設定

検索の初期値

ここでは決裁データのみで絞り込むことを意図していますので、検索の初期値を "3" で固定します。システムが提供する選択肢モデル "フロー状態検索項目（jfcflowstatus_search_m)" は次の値となっています。

ID	内容
1	未申請
2	申請・承認
3	決裁
4	却下

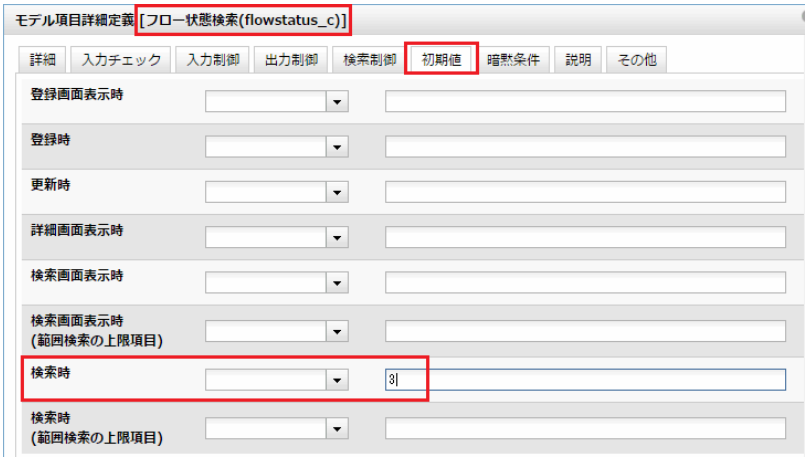


図 検索時の初期値設定

隠し項目

この「フロー状態検索」項目そのものを、検索時は隠し項目とします。これによって検索画面に表示されなくなります。

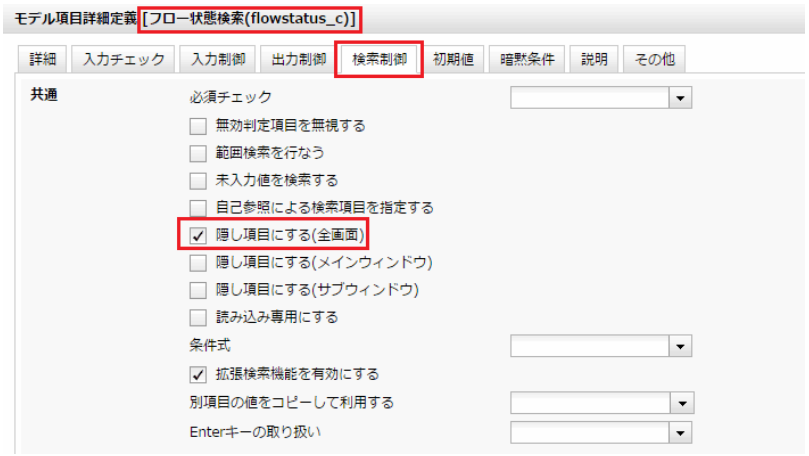


図 検索画面で隠し項目とする

メニューを用意する

このサブモデルをメニューに配置します。タイトルや色も変更しておきます。

図 メニューに配置する(1)

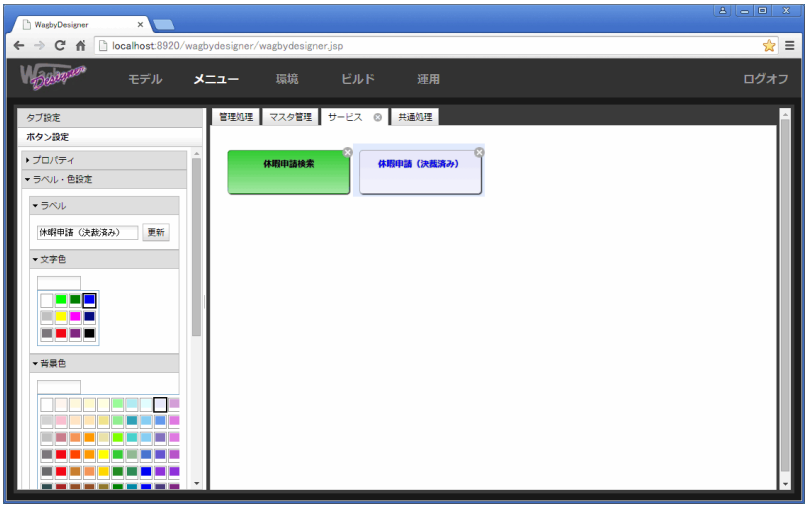


図 メニューに配置する(1)

例

サブモデルでワークフロー状態による検索を行った例です。

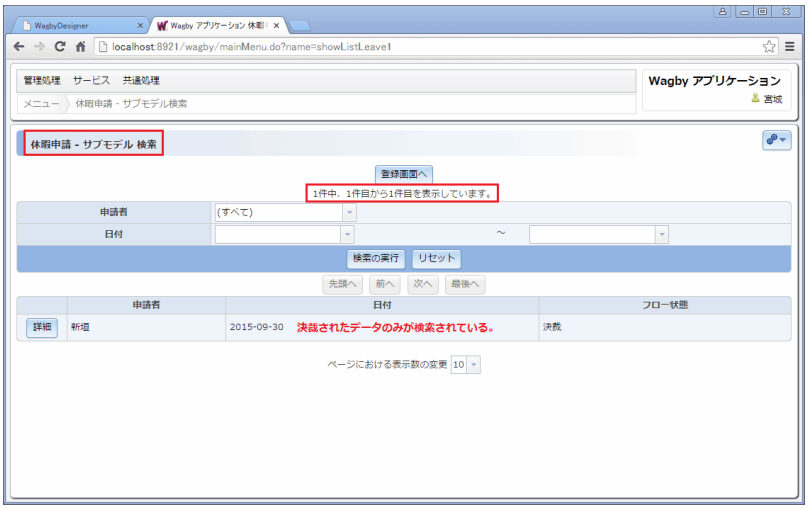


図 決裁されたデータのみが表示される

ワークフローが進行中のデータは権限のあるユーザにメインモデルでのみ閲覧させ、権限のないユーザにはサブモデルで決裁データのみ閲覧させる といったような使い方ができます。

制約

- 検索条件に指定した項目（今回の例における「フロー状態検索」項目）を一覧表示に指定することはできません。指定しても無視されます。一覧表示項目にフロー状態を表示させる場合は、図の「フロー状態」(flowstatus)項目を用いてください。図の定義では「フロー状態」(flowstatus)項目の一覧表示を有効にしています。

条件によってフローを制御する(スクリプト)

物品購入伺いワークフロー

ここでは「物品購入伺い」を例に説明します。図のモデル定義を用います。

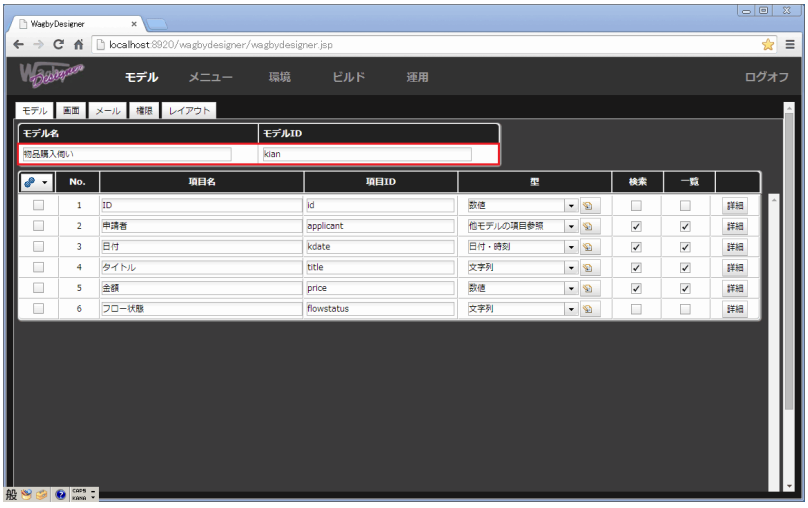


図 物品購入伺いモデル

サンプル（1）後続の承認者をスキップさせる

ルール

次のルールを適用します。金額によって「次の承認者がスキップされる」というものです。

金額	課長承認	部長承認	本部長承認	社長承認（決裁）
50万円未満	○	−	−	−
50万円以上100万円未満	○	○	−	−
100万円以上1,000万円未満	○	○	○	−
1,000万円以上	○	○	○	○

アカウント

次の5つのアカウントを用意します。

アカウント	氏名
user1	申請者
user2	課長
user3	部長
user4	本部長
user5	社長

フローパターン

「順次・順次・順次・順次・順次フロー」を用意します。 説明文に「申請→課長承認→部長承認→本部長承認→決裁」と記載します。最後の「決裁」は「社長承認」を兼ねるとします。

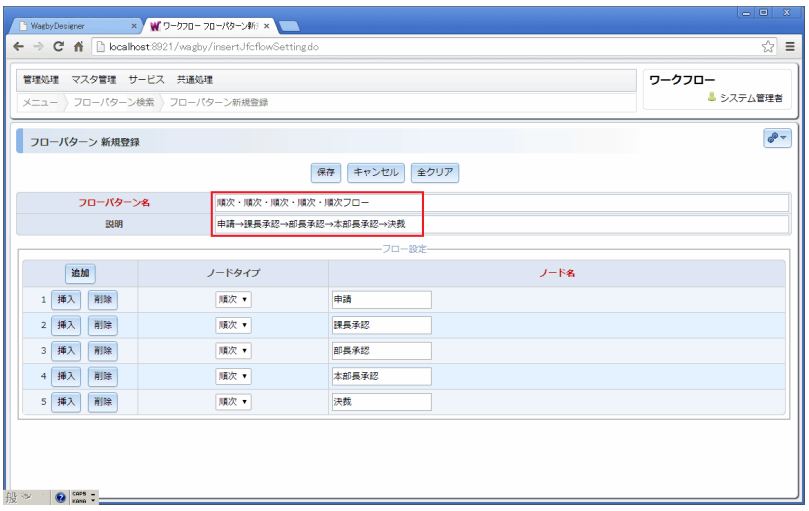


図 フローパターンの追加

フロー参加者

user1からuser5を図のように割り当てます。

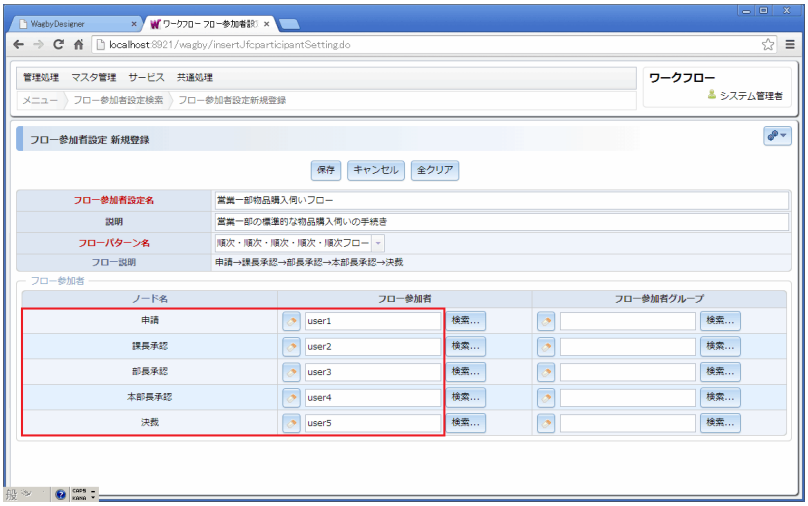


図 フロー参加者の追加

フロー設定で条件を指定する

図で用意した「営業一部物品購入伺いフロー」を選択すると、図のような初期ルートとなります。

図 初期状態のフロー設定

ここで「条件設定」欄を用意します。課長承認後、金額によってすぐに決裁とする条件を加えます。条件部の記述ルールは次のとおりです。

- 条件記述では「起点」と「終点」が必須になります。ノードのIDになります。図において "1.順次" と記載されているノードの "1" の部分がノードIDです。
- 「終点」に、存在しないノードIDを指定すると、自動的に「決裁」へ向かいます。例えば "999" などを使うとよいでしょう。
- 条件部はサーバサイドJavaScriptのコードを記述します。return 文で true または false を返すようにしてください。この条件が成立した（trueが戻された）ときに、終点のノードIDへ移動します。
- 条件が空の場合は、必ず終点のノードIDへ移動します。
- 条件を一つ以上追加すると、標準のルートは消去されます。

```
return kian.price < 500000
```

図 条件を1つ追加する

「挿入」ボタンを押下し、申請者から課長承認へのルートを加えた例を図に示します。ここは条件部を空としています。これによって ″1″ から ″2″ へのルートが描画されます。

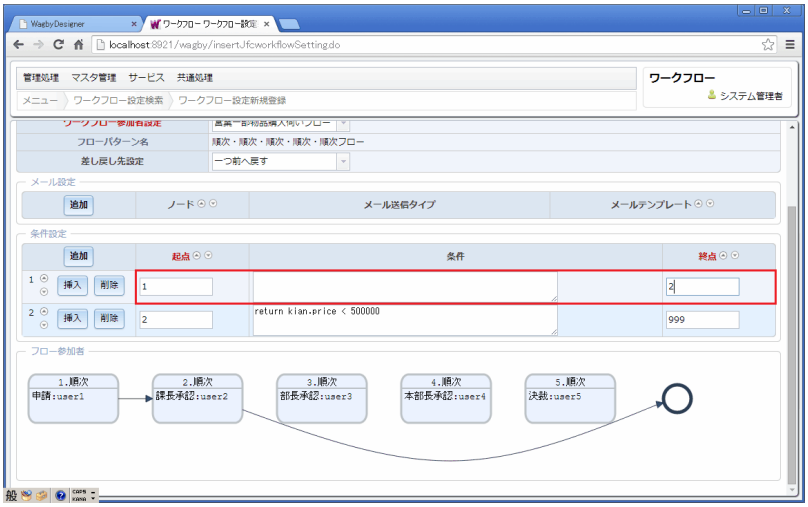


図 条件部が空のルート設定

すべてのルートを設定した状態を図に示します。

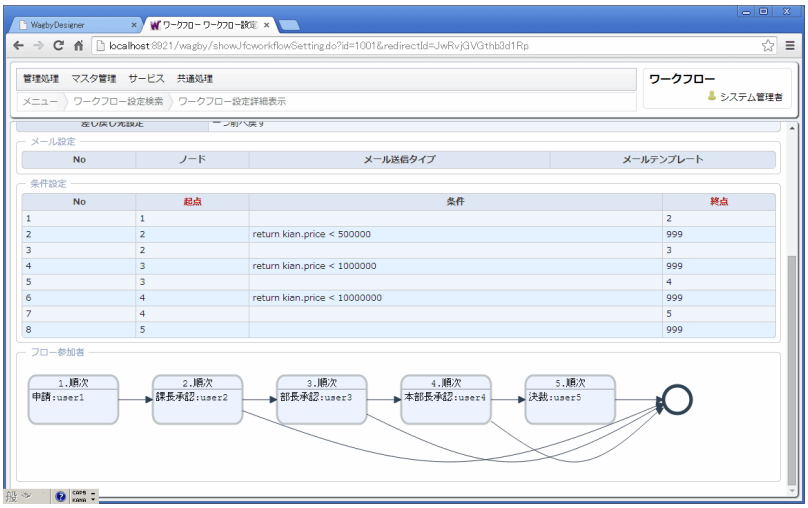


図 すべての条件を設定

実行

課長決裁

50万円未満の購入伺いを行った例を図, 図に示します。

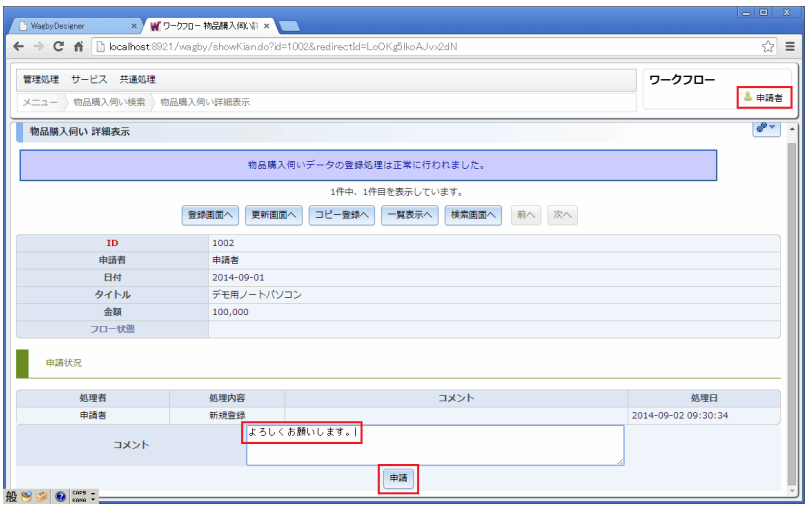


図 50万円未満の購入伺い

図 ワークフローの開始（申請）

課長アカウント(user2)でログオンし、承認します。　　この段階で決裁となり、ワークフローは終了します。

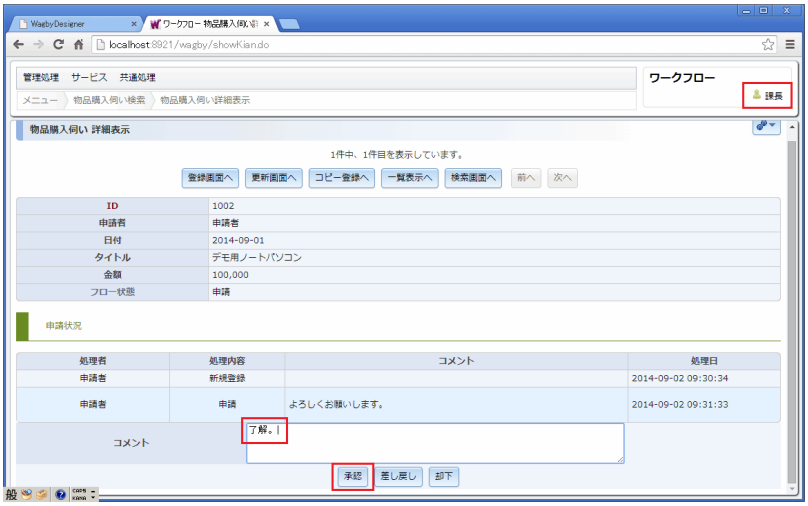


図 課長による承認行為

図 課長承認が決裁となる

部長決裁

50万円以上100万円未満の購入伺いを行った例を図に示します。

図 50万円以上100万円未満の購入伺い

課長による承認直後が図となります。フロー状態は「課長承認」となります。

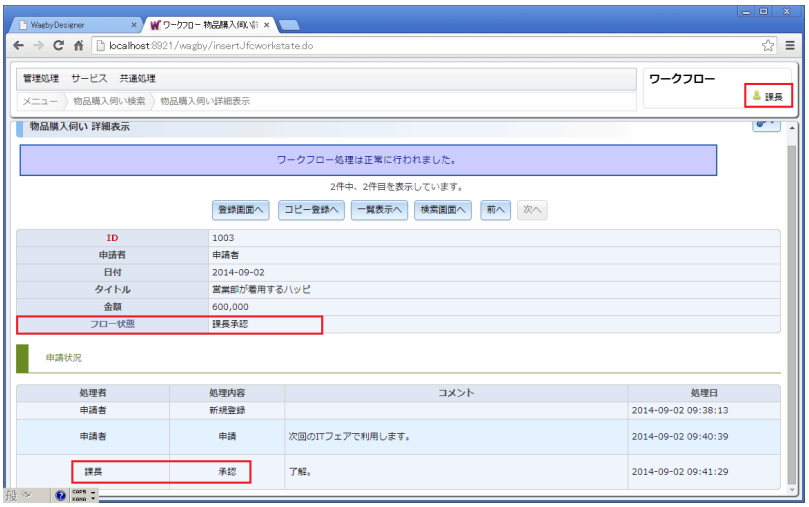


図 課長承認

続いて部長による承認を行ったのが図となります。この段階で決裁となります。

図 部長承認が決裁となる

本部長決裁

100万円以上1,000万円未満の購入伺いを行った例を図に示します。

図 100万円以上1,000万円未満の購入伺い

本部長による承認を行ったのが図となります。この段階で決裁となります。

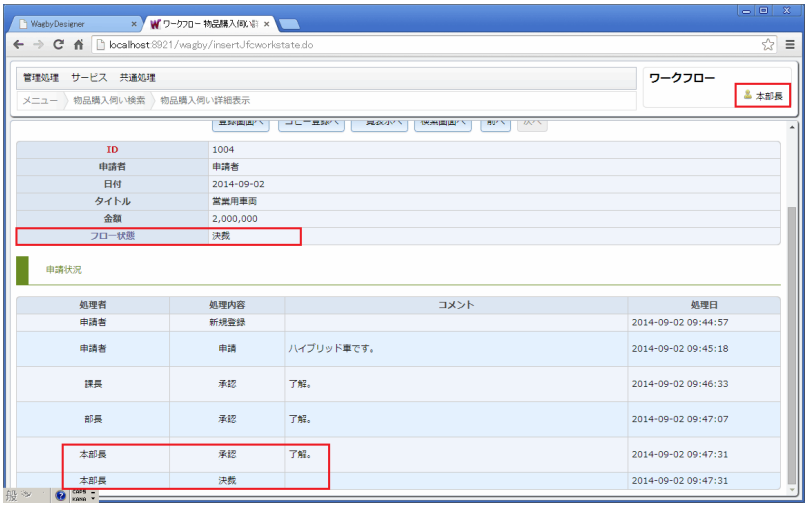


図 本部長による承認が決裁となる

社長決裁

1,000万円以上の購入伺いを行った例を図に示します。

図 1,000万円以上の購入伺い

社長による承認を行ったのが図となります。

図 社長決裁まで行った

サンプル（2）最初の承認者を選択する

ルール

次のルールを適用します。上の例に加えて、金額によって「最初の承認者」が変わるというものです。

金額	課長承認	部長承認	本部長承認	社長承認（決裁）
50万円未満	○	−	−	−
50万円以上100万円未満	−	○	−	−
100万円以上1,000万円未満	−	−	○	−
1,000万円以上	−	−	−	○

設定は図のようになります。

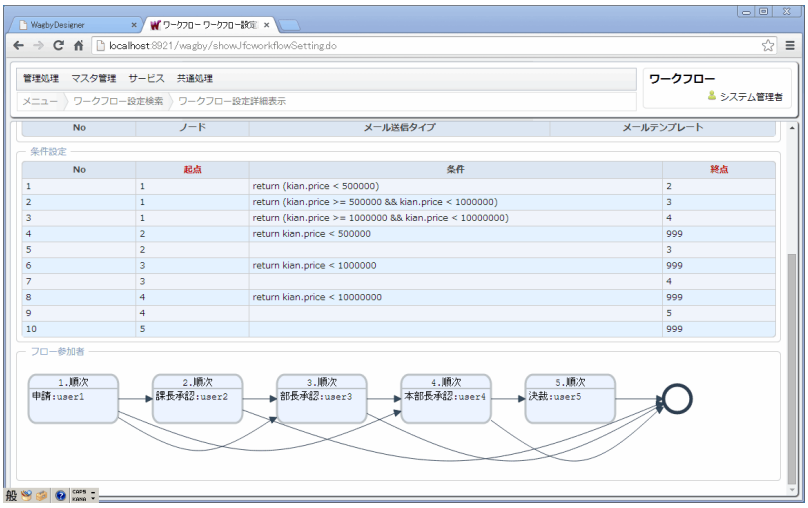


図 条件の設定

仕様・制約

条件部の記述

条件部は複数行にわたって記述することができます。このため複雑な条件を記述できます。例を示します。

```
var p = kian.price;
if (p < 500000) {
  return true;
}
return false;
```

項目名にはキャメル記法が適用されます

「モデルID」は設計情報に記載した名前がそのまま利用できます。（上記例では "kian" という表記になります。）

「項目ID」はキャメル記法が適用されます。 例えば項目IDが "group_cd" の場合、スクリプト内では "groupCd" と表記します。

```
return koubai_model.groupCd == 1
```

循環について

条件記述の誤りで、フローが循環するような場合は正しく動作しません。設定に際しては循環にならないよう、ご注意ください。

利用できる暗黙変数について

条件を記述するスクリプトでは、当該モデル（ストアモデル）のみが参照できます。他のスクリプトで利用できる暗黙変数 p などはすべて利用できません。（このスクリプトではモデルの値を使って条件判定を行うことに特化したものとなっています。）

スクリプトを変更したときの動作

運用中（テスト中を含む）にスクリプトを修正した場合、ワークフロー利用者(申請者、承認者など)をいったんログアウトし、再度ログオンすることで新しいスクリプトが有効になります。

(448)

条件によって申請フローを絞込む(スクリプト)

実行

「物品購入伺い」モデルにワークフローを適用した例です。登録画面を開いたとき、申請フローは "営業一部物品購入伺いフロー1" が選択できるようになっています。これは10万円以下のフローとして定義しています。（後述）

(448) スクリプトを修正しても、再ログオンしない限り、修正したスクリプトは反映されません。

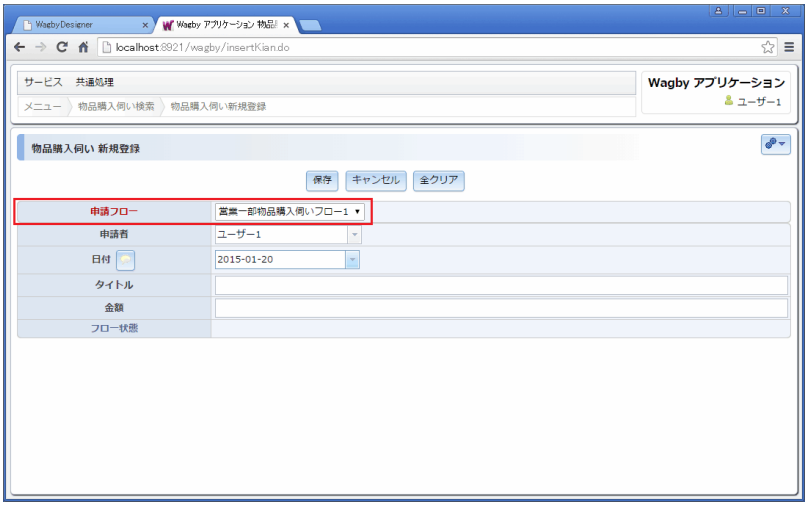


図　初期状態の申請フロー選択肢

金額を入力すると画面が再描画され、申請フローが変わります。ここでは金額が10万円を超えると　″営業一部物品購入伺いフロー2″ に切り替わるようにしています。

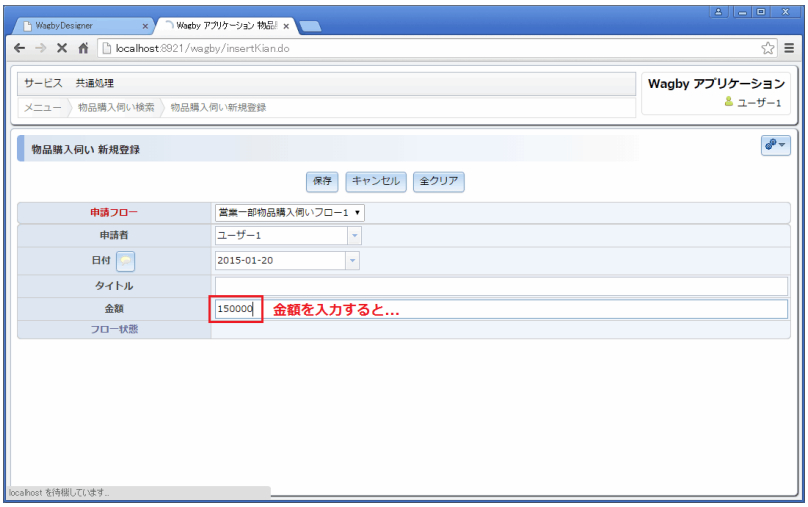


図　金額欄を入力する

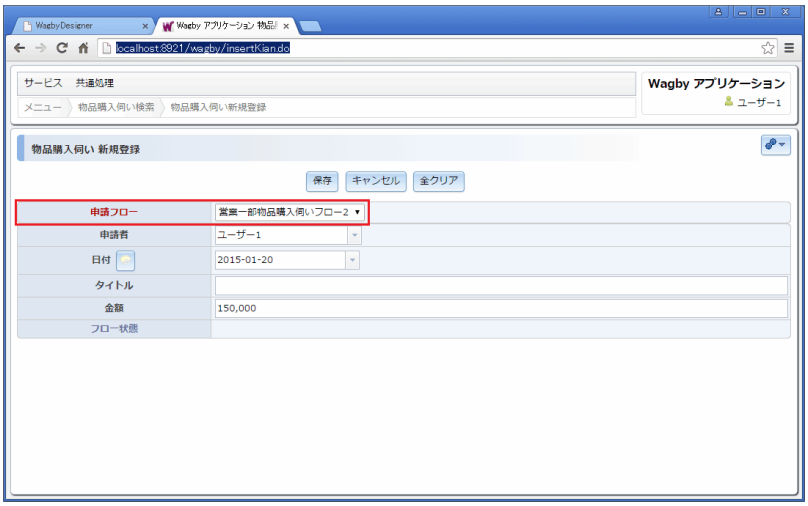


図　申請フローが切り替わる

金額が20万円を超えると ″営業一部物品購入伺いフロー3″ に切り替わります。

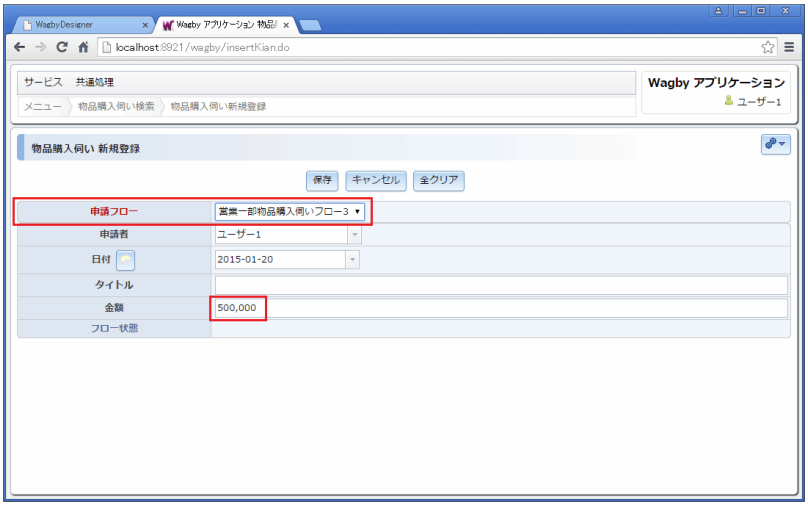


図 申請フローが切り替わる(2)

申請フローの選択肢は常に最低1つ、選択できるように設定します。この項目は必須指定のため、空白の状態で画面が再描画されると入力エラーのメッセージが画面に表示されてしまいます。

定義方法

モデルの定義

「物品購入伺い」モデルの定義は次のとおりです。このモデルにワークフローを適用します。

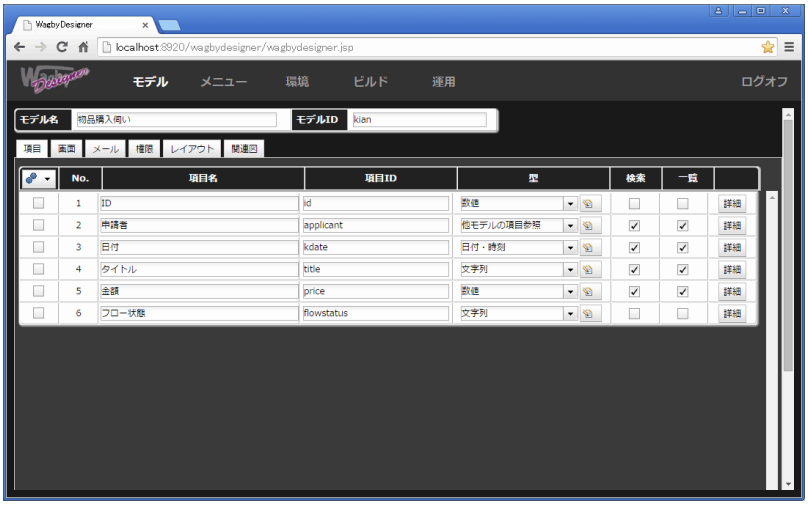


図 物品購入伺いモデル

「金額」項目は、値の変更時に画面を再描画するとします。

図 値の変更時に画面を再描画する設定

アカウント

次の5つのアカウントを用意します。

アカウント	氏名
user1	ユーザー1(申請者)
user2	ユーザー2
user3	ユーザー3
user4	ユーザー4

ルール

申請時に、どのワークフローが適用できるかは次のルールとします。

金額	フロー1	フロー2	フロー3
10万円未満	○	−	−
10万円以上20万円未満	−	○	−
20万円以上	−	−	○

フロー参加者設定

ビルドしたアプリケーションで、「フロー参加者設定」を用意します。上記ルールを適用します。

(449)

(449) "順次・順次・順次・順次フロー" は標準では提供されていないため、別途「フローパターン」モデルを追加してください。

図 フロー参加者設定一覧

"営業一部物品購入伺いフロー1" は、10万円未満用のフローとします。ノードは2つです。

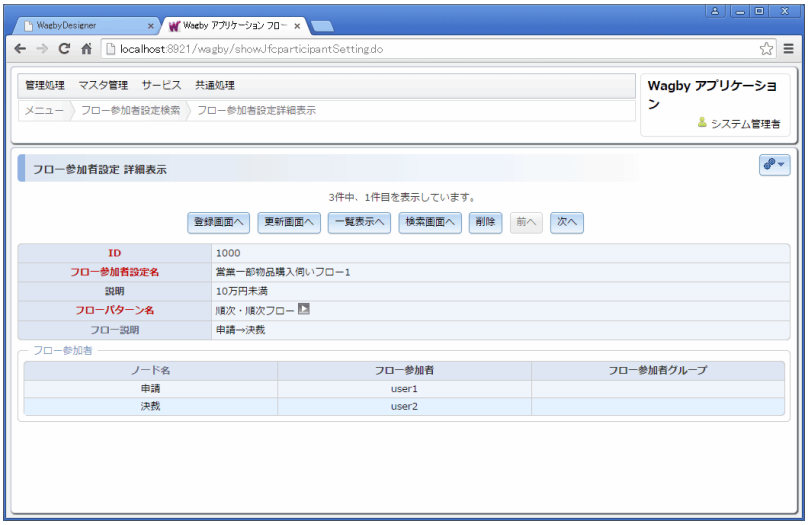


図 順次・順次フローの定義

"営業一部物品購入伺いフロー2" は、10万円以上20万円未満用のフローとします。ノードは3つです。

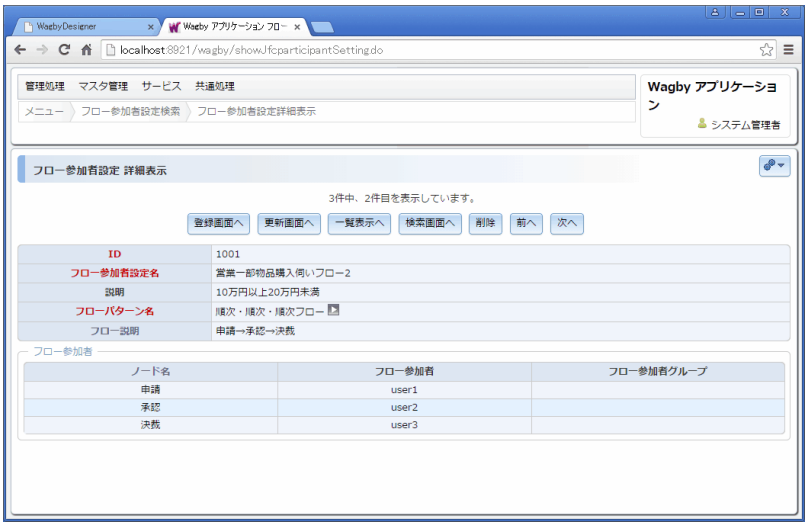


図 順次・順次・順次フローの定義

″営業一部物品購入伺いフロー3″は、20万円以上用のフローとします。ノードは4つです。

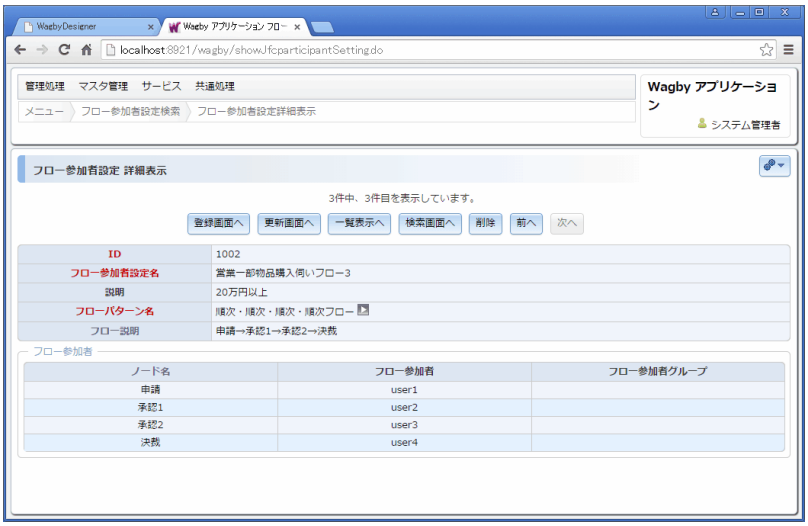


図 順次・順次・順次・順次フローの定義

ワークフロー設定

上の3つのフローを、「物品購入伺い」に紐づけます。

″営業一部物品購入伺いフロー1″の設定で、「有効条件」として次の式を設定します。

```
return kian.price < 100000
```

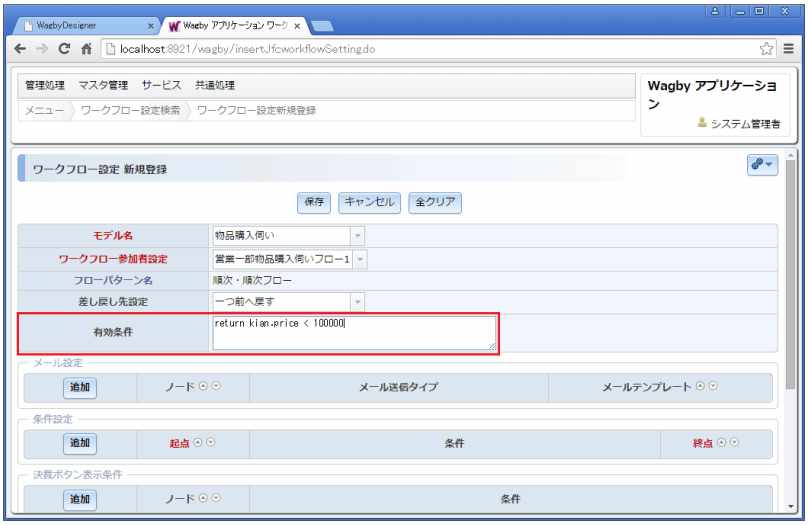


図 営業一部物品購入伺いフロー1の有効条件を設定する

″営業一部物品購入伺いフロー2″の設定で、「有効条件」として次の式を設定します。

```
return kian.price >= 100000 && kian.price < 200000
```

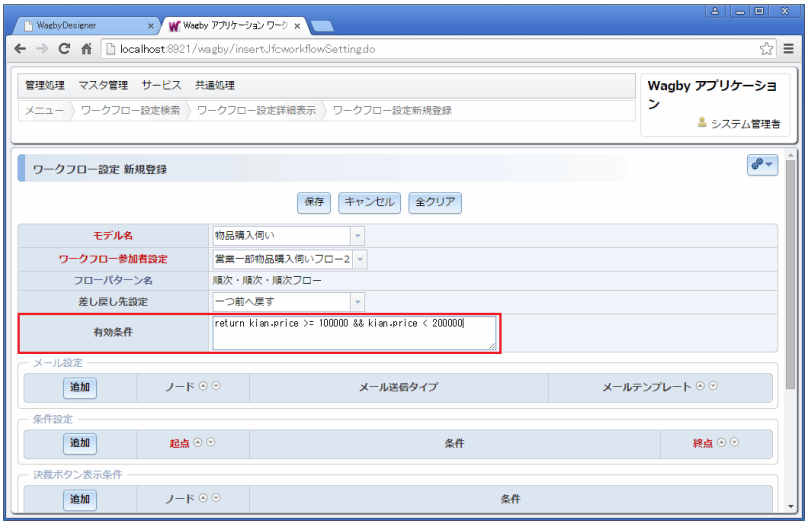


図 営業一部物品購入伺いフロー2の有効条件を設定する

″営業一部物品購入伺いフロー3″の設定で、「有効条件」として次の式を設定します。

```
return kian.price >= 200000
```

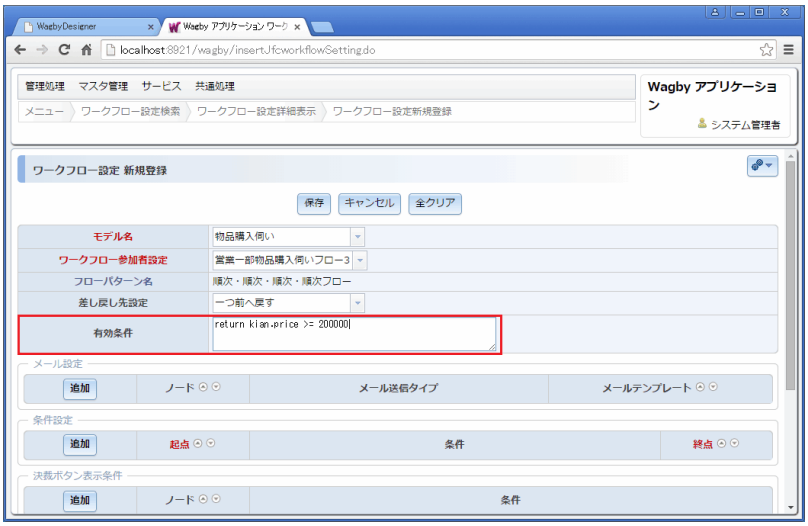


図 営業一部物品購入伺いフロー3の有効条件を設定する

ここまでで、3つのワークフロー設定を用意しました。

図 ワークフロー設定

このように「有効条件」を設定すると、この条件に合致した場合に、申請フローの選択肢として画面に表示されるようになります。

条件の書き方

条件はスクリプトで記述します。

「モデルID」は設計情報に記載した名前がそのまま利用できます。（上記例では ″kian″ という表記になります。）

「項目ID」はキャメル記法が適用されます。　例えば項目IDが　″group_cd″　の場合、スクリプト内では ″groupCd″ と表記します。

```
return koubai_model.groupCd == 1
```

任意のノードに決裁ボタンを用意する(スクリプト)

例(1)

「物品購入伺い」モデルにワークフローを適用した例です。″順次・順次・順次・順次″ の 4 ノードを設定しています。″ユーザー1″ アカウントでフローを開始します。

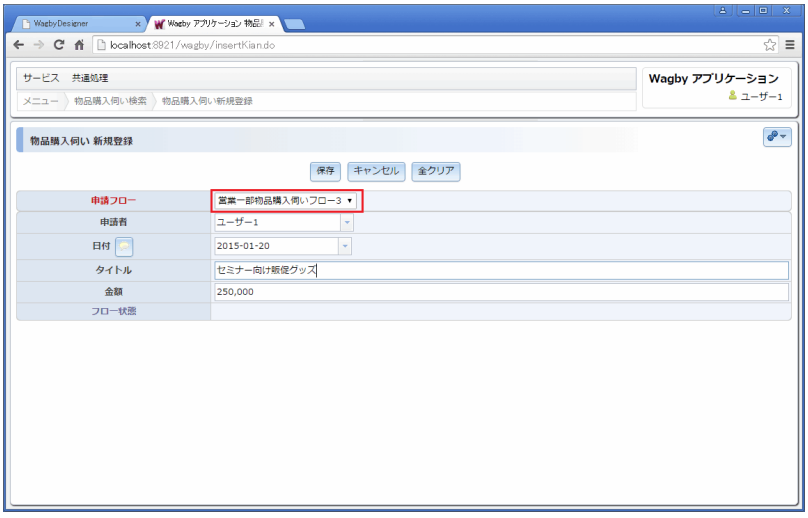


図 物品購入伺いを新規登録する

″ユーザー1″ がワークフローを申請します。

図 ワークフローを申請（開始）する。

ここで金額が30万円未満なら、2ノード目であっても決裁できるようにしました。2ノード目の承認者である〝ユーザー2〟で決裁ボタンが表示されています。

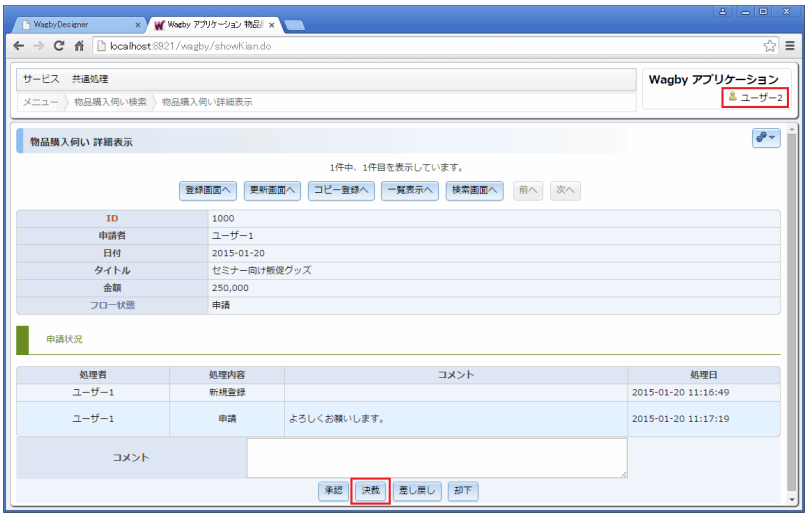


図 2ノード目で決裁ボタンが表示されている

決裁ボタンを押下すると、フローが終了します。3ノード目以降はスキップされます。

図 ワークフローが決裁された

例(2)

次に、金額が30万円以上、50万円未満の伺いを行います。この場合、3ノード目で決裁できるように設定しています。

″ユーザー1″ がワークフローを申請します。金額は40万円とします。

図 購入伺いフローを開始する

2ノード目（ユーザー2）では決裁ボタンは表示されません。

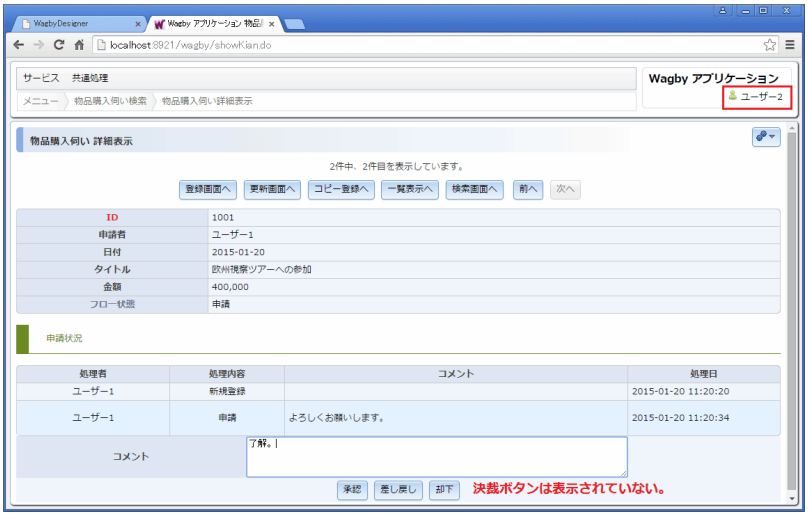


図 ユーザー2は承認処理を行う

3ノード目（ユーザー3）では金額の条件を満たすため、決裁ボタンが表示されます。

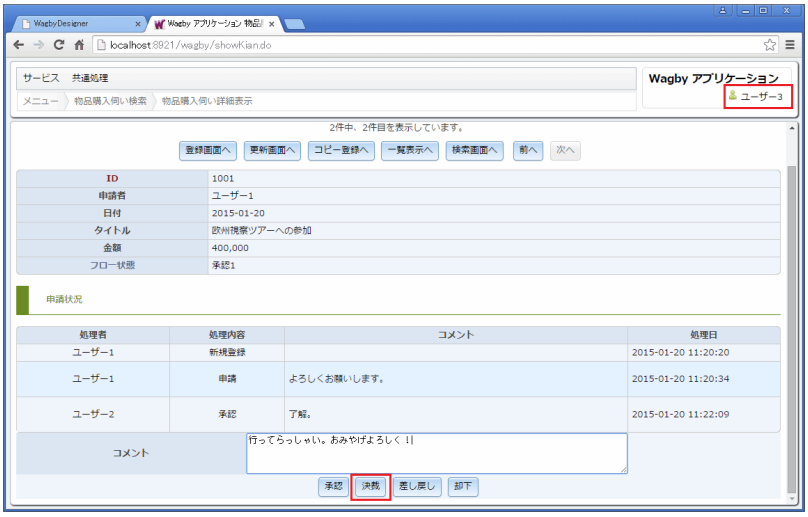


図 ユーザー3は決裁ボタンが表示される

ユーザー3で決裁を行うと、フローは終了します。

図 フロー終了

定義方法

モデルの定義

物品購入伺いモデルの定義ならびにワークフローのフロー参加者設定は「ワークフロー ＞ 条件によって申請フローを絞込む」と同じものを使います。

ルール

「決裁ボタン表示条件」ルールを次のように設定します。

ノード番号	決裁ボタン表示の条件
2	return kian.price < 300000
3	return kian.price < 500000

ワークフロー設定

″営業一部物品購入伺いフロー3″ に、上の「決裁ボタン表示条件」を設定した例です。

図 営業一部物品購入伺いフロー3に決裁ボタン表示条件を設定する

条件の書き方

有効条件はスクリプトで記述します。詳細は"条件によって申請フローを絞込む"をお読みください。

グループ階層とプリンシパルを利用したワークフローの制御(スクリプト)

概要

人事異動や組織改編といったケースに対応したワークフローの運用を行うために、Wagbyが提供するグループ階層とプリンシパルを組み合わせた承認者、決裁者の制御方法を紹介します。ワークフローではスクリプトを用いて承認者や決裁者を動的に決めることができます。

このページのポイントは次の二点です。

- アカウントの所属グループにグループ階層を設定する。
- フロー参加者スクリプトでグループ階層、プリンシパルを条件に承認者、決裁者を動的に決定する。

準備

アカウント

次に示すアカウントを用意しておきます。

アカウント	所属グループ	プリンシパル
amuro	営業一課	年休申請者
kinjo	営業一課	年休承認者
miyagi	営業二課	年休申請者
shinjo	営業二課	年休承認者
onaga	営業部	年休決裁者

グループ階層

営業一課、営業二課、営業部は次のグループ階層とします。グループ階層の作成方法はこちらをお読みください。

図 グループ階層

年休申請・承認・決裁ルール

今回の例は次のルールとします。

申請者
- すべてのユーザ

承認者
- 申請者の同一グループに所属
- 「年休承認者」プリンシパルを持つ

決裁者
- 申請者より1階層上位のグループに所属
- 「年休決裁者」プリンシパルを持つ

このルールに準じたアカウント設定の例を示します。

役割	アカウント	グループ	プリンシパル
申請者	amuro	営業1課	
承認者	kinjo	営業1課	年休承認者プリンシパル有
決裁者	onaga	営業部	年休決裁者プリンシパル有

フロー参加者にスクリプトを設定する

今回は「順次・順次・順次フロー」をベースに、スクリプトを設定した例を示します。

図 スクリプトを使った設定

申請者のスクリプト

全ユーザーを申請可能とする場合は、次のように記述します。

```
return wman.getLogonUserOrApplicant(modelname, modelpkey, p);
```

- 暗黙変数wmanを利用します。wmanは、Wagbyが標準で提供するワークフロー管理クラス(WorkflowManagerクラス)のインスタンスです。
- 暗黙変数 modelname と modelpkey はそれぞれモデルIDと、主キーの値を保持します。
- "すべてのユーザ" を申請者として表現するためにwmanが提供するgetLogonUserOrApplicantメソッドを使います。このメソッドは新規登録の場合、ログオンユーザのuseridを返します。申請済みの場合は申請者のuseridを返します。

承認者のスクリプト

```
return $applicant.group().findUsers(By.principal("nenkyu_authorizer")).name();
```

- 暗黙変数$applicantを利用します。この$applicantは申請者を表すUserElementクラスのインスタンスです。
- 申請者と同一のグループに所属するユーザで、年休承認者(nenkyu_authorizer)のプリンシパルを持つユーザのuseridを返すという式になっています。
  - $applicant.group()は、申請者の所属グループ(GroupElement)を取得します。
  - findUsers(By.principal("nenkyu_authorizer")).name()で、グループに属するユーザーのうち、プリンシパル“nenkyu_authorizer”を持つユーザーのIDを取得します。

決裁者のスクリプト

```
return $applicant.group().parent().findUsers(By.principal("nenkyu_finalist")).name();
```

- 申請者の所属グループの一階層上のグループに所属するユーザで年休決裁者(nenkyu_finalist)のプリンシパルを持つユーザの useridを返す式になっています。
  - $applicant.group().parent()は、申請者の所属グループの1階層上のグループ(GroupElement)を取得します。
  - findUsers(By.principal("nenkyu_finalist")).name()で、グループに属するユーザーのうち、プリンシパル“nenkyu_finalist”を持つユーザーのIDを取得します。

このようなスクリプトを利用することで、一つの参加者設定（式）で複数のグループに対する承認ルールを表現できます。

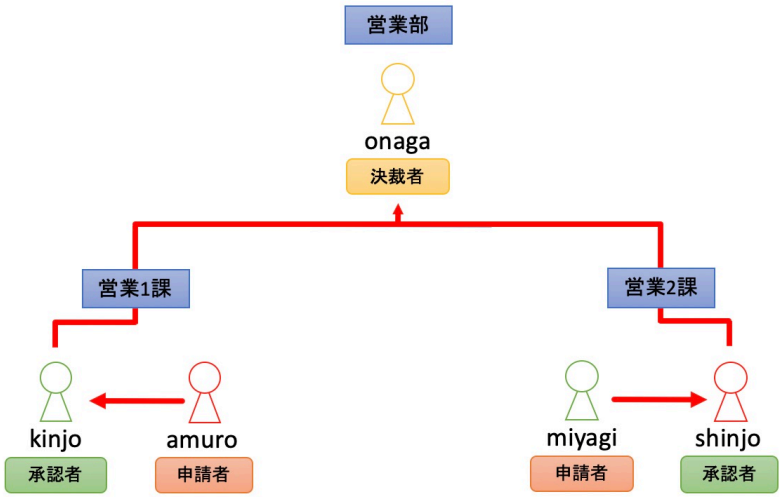


図 スクリプトによって表現した内容

実行

1. 営業一課 amuro による申請

   はじめに amuro が年休を申請しました。

2. 営業一課による承認

   同一グループに所属する承認者 kinjo でログオンすると、承認ボタンが表示されます。

図 承認者kinjoは承認できる

異なるグループ（営業二課）に所属する承認者 shinjo では、承認ボタンは表示されません。

図 承認者shinjoはグループが異なるため承認できない

kinjo が承認しました。次は決裁者になります。kinjo は決裁者スクリプトの条件を満たさないため、承認ボタンは表示されません。

図 kinjoの承認後。kinjoは次の決裁者ではないため承認ボタンは表示されない

3. 営業部による決裁

申請者の一つ上のグループ階層(営業部)に所属し、決裁者プリンシパルを持つ onaga でログオンすると、承認ボタンが表示されます。

図 決裁者onagaは決裁できる

決裁者 onaga による決裁後、ワークフローは完了となります。

図 決裁が完了した

スクリプト内で利用できる暗黙変数

"フロー参加者設定スクリプト" で利用できる暗黙変数は次の通りです。

変数名	説明	例
modelname	当該ワークフローのモデルID。	kianモデルで実行すると print(modelname) で "kian" と出力される。
modelpkey	当該ワークフローモデルデータの主キーの値。	主キー1000のデータで実行すると print(modelpkey) で "1000" と出力される。
p.user.username	ログオンしているユーザーID。	"amuro"
p.request	Javaが提供するrequestオブジェクト。	p.request.getParameter("foo")で、HTTPリクエストパラメータfooの値を取得できる。
p.errors	Wagbyが提供するエラーメッセージを管理するオブジェクト。	-
p.appctx	Springフレームワークが提供するApplicationContextオブジェクト。	p.appctx.getBean("ビーンID")で、Springが管理するオブジェクトを取得できる。

<モデルID>	当該ワークフローモデルのストアモデルオブジェクト。(*1)	"年休申請(nenkyu)"の"理由(reason)"項目の値は nenkyu.reason で参照できる。
<モデルID_p>	当該ワークフローモデルのプレゼンテーションオブジェクト。	
jfcworkstate	当該ワークフローモデルデータの申請状況。申請後のみ利用可能。承認履歴の最も最新の情報を表現するデータを保持している。	
jfcworkstate.comment	直近のコメント	
jfcworkstate.username	直近の処理者のユーザーID	
jfcworkstate.updateDate	直近の処理日時	
wman	Wagbyが提供するワークフロー管理クラスWorkFlowManagerのインスタンス	
JFCUtils,ExcelFunction	使い方はoperation-script-excelfunction.htmlを参照。	
applicant	申請者のユーザーID。実体は juser オブジェクト。	
$applicant	申請者の UserElement オブジェクト	
$rootGroup	最上位のグループを表現するGroupElementオブジェクト	
$allUsers	すべてのアカウントを表現するUserElementオブジェクト	

ストアモデルの詳細 (*1)

標準では、ストアモデルオブジェクトは参照連動や自動計算の処理を行っていません。これはパフォーマンスを優先しているためです。参照連動項目を解決し、計算式の値を取得する場合は、次のように EntityService を経由してください。

```
var entityService = p.appctx.getBean("LeaveEntityService");
var leave = entityService.findById(modelpkey);
```

UserElementの詳細

UserElementは複数または単一のユーザーを表現するクラスとなっています。このクラスが提供する API は次のとおりです。

API	説明
group()	申請者の所属グループ (GroupElement) を取得する。複数の場合もある。
name()	ユーザー名を取得する。複数のユーザー名の場合は区切り文字「|」で連結する。

GroupElementの詳細

上の API で説明した group() で取得できるグループは GroupElementとなっています。GroupElementは複数または単一のグループを表現するクラスです。このクラスが提供する API は次のとおりです。

API	説明
parent()	親グループを取得する。GroupElementが複数のグループの場合は、それぞれの親グループを取得できる。
children()	子グループを取得する。複数の場合もある。
descendants()	現在のグループと、すべての下位階層のグループを取得する。

jgroupid()	グループID(数値)の配列を取得する。
userid()	ユーザIDの配列を取得する。
users()	グループに所属しているユーザーをUserElementとして取得する。GroupElementが複数のグループの場合は、それぞれのグループの所属ユーザーすべてを取得することができる。
findUsers()	グループに所属しているユーザーを検索する。引数に By.principal を指定することができる。[後述]

$applicantの使い方

$applicantは申請者スクリプトでは利用できません。承認者および決裁者スクリプトで利用できます。

次の式は、申請者の1階層上位グループ(GroupElement)を取得します。

```
$applicant.group()
```

次の式は、申請者の所属グループのグループID(数値の配列)を取得します。

```
$applicant.group().jgroupid();
```

(例) 申請者がamuroの場合は1000となります。

図 グループIDの取得

次の式は、申請者と同じグループに属するユーザー(UserElement)を取得します。

```
$applicant.group().users();
```

次の式は、申請者と同じグループに属するユーザーのユーザーIDを取得します。

```
$applicant.group().users().name();
```

(例) 申請者がamuroの場合は”amuro|kinjo”となります。

(450)

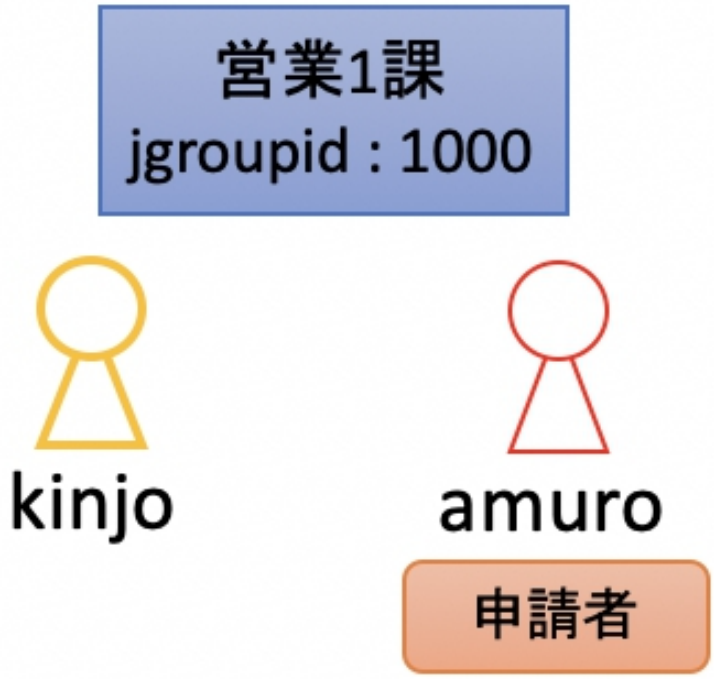


図 グループに属するユーザIDの取得

次の式は、申請者と同じグループに属するユーザーのうち、プリンシパル“nenkyu_authorizer”を持つユーザー（UserElement）のユーザーIDを取得します。

```
$applicant.group().findUsers(By.principal("nenkyu_authorizer")).name();
```

ワークフローのスクリプトで、By.principal を使ってプリンシパルで絞り込みを行うことができます。

```
By.principal("プリンシパル名")
By.principal("プリンシパル名","プリンシパル名",...)
```

複数のプリンシパル名をコンマ区切りで指定することもできます。

(451)

（例） 申請者がamuroの場合は”yamashiro”となります。

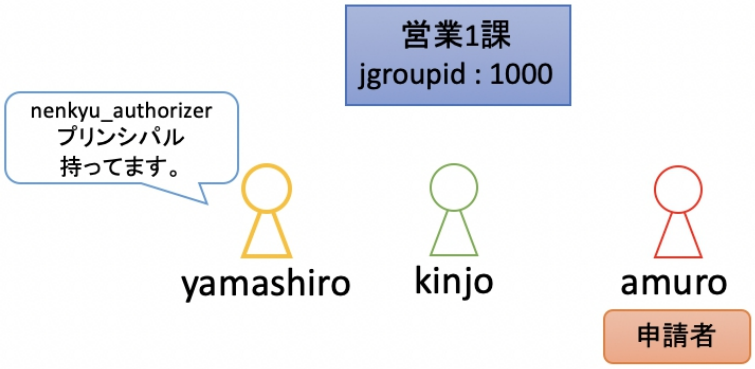


図 プリンシパルを指定したユーザIDの取得

次の式は、申請者の所属グループの1階層上のグループ(GroupElement)のグループIDを取得します。

(450) 複数のユーザーIDは区切り文字「|」で連結される。

(451) プリンシパルの作成方法は リポジトリ > 認証・認可 > モデル単位の制御 をお読みください。

```
$applicant.group().parent().jgroupid();
```

（例） 申請者がamuroの場合は”2000”（営業部）となります。

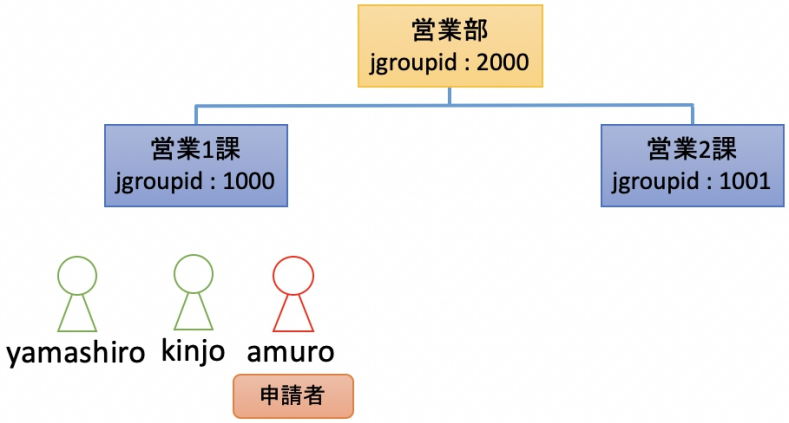


図 上位グループのグループIDの取得

次の式は、申請者の所属グループの1階層上のグループに属するユーザー(UserElement)のユーザーIDを取得します。

```
$applicant.group().parent().users().name();
```

（例） 申請者がamuroの場合は “uehara|onaga” となります。

(452)

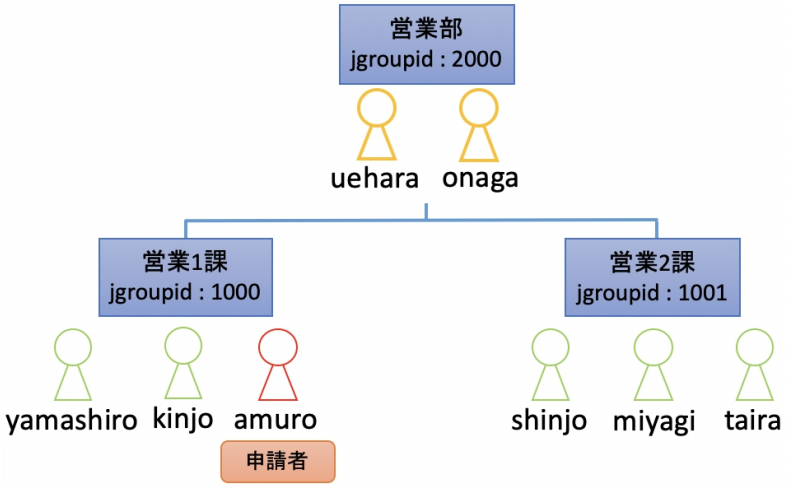


図 上位グループのグループに属するユーザIDの取得

次の式は、申請者の所属グループの1階層上のグループおよび、その下位階層グループ(GroupElement)のグループIDを取得します。

```
$applicant.group().parent().descendants().jgroupid();
```

（例） 申請者がamuroの場合は1000,1001,2000(営業1課、営業2課、営業部)となります。

(453)

(452) 複数のユーザーIDは区切り文字「|」で連結される。

(453) 複数のグループIDは区切り文字「|」で連結される。

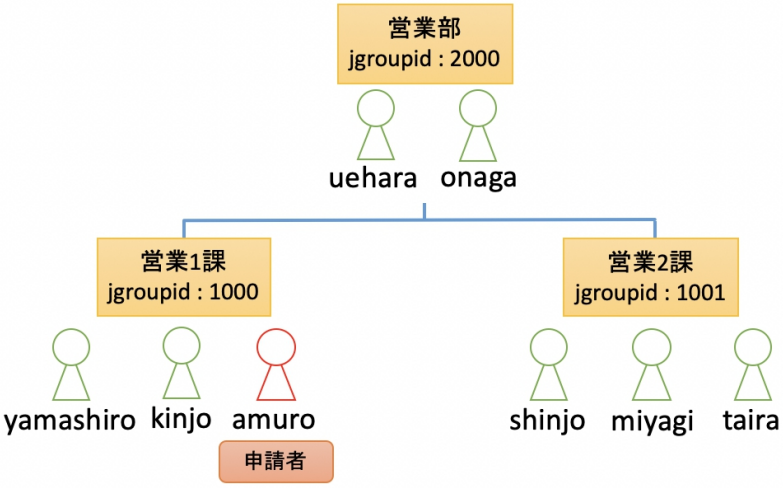


図 上位グループおよび下位グループのグループIDの取得

次の式は、申請者の所属グループの1階層上のグループおよび、その下位階層グループに属するユーザー(UserElement)を取得します。

```
$applicant.group().parent().descendants().users().name();
```

（例） 申請者がamuroの場合は “uehara|onaga|yamashiro|kinjo|amuro|shinjo|miyagi|taira” となります。

(454)

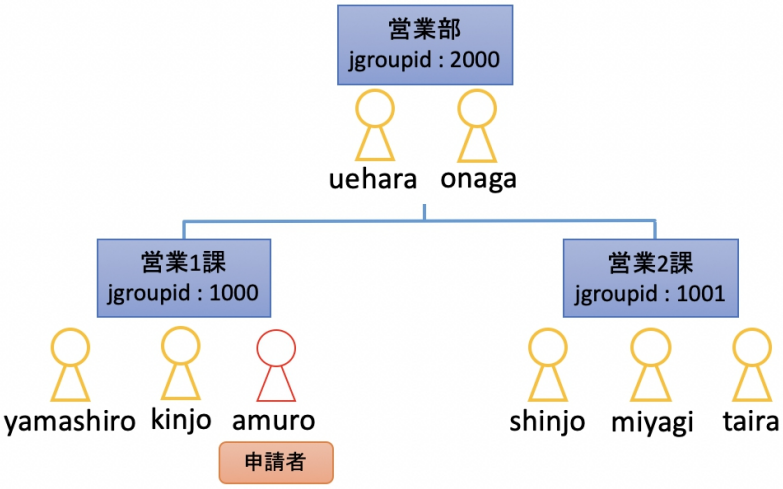


図 上位グループおよび下位グループに属するユーザIDの取得

次の式は、申請者の所属グループの1階層上のグループおよび、その下位階層グループに属するユーザー(UserElement)のうち、プリンシパル"nenkyu_finalist"を持つユーザーを取得します。

```
$applicant.group().parent().descendants().findUsers(By.principal("nenkyu_finalist")).name();
```

（例） 申請者がamuroの場合は”taira”となります。

(454) 複数のユーザIDは区切り文字「|」で連結される。

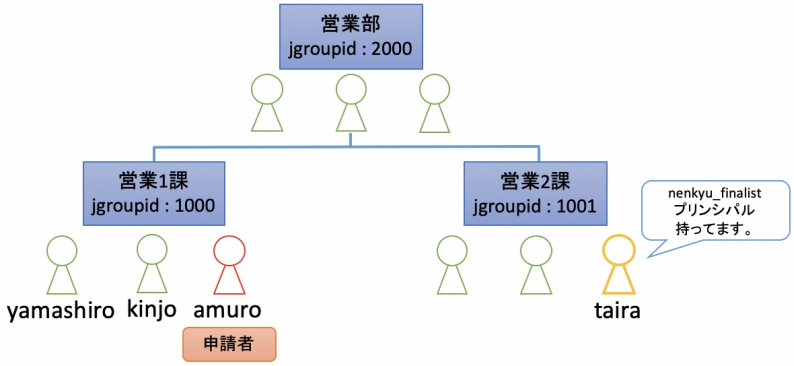


図 上位グループおよび下位グループに属するユーザのうち、指定されたプリンシパルをもつユーザIDの取得

次の式は、申請者の所属グループの1階層上のその下位階層グループ(GroupElement)のグループIDを取得します。

```
$applicant.group().parent().children().jgroupid();
```

（例） 申請者がamuroの場合は1000,1001 （"営業1課"、"営業2課"）となります。"営業部" は含まれません。

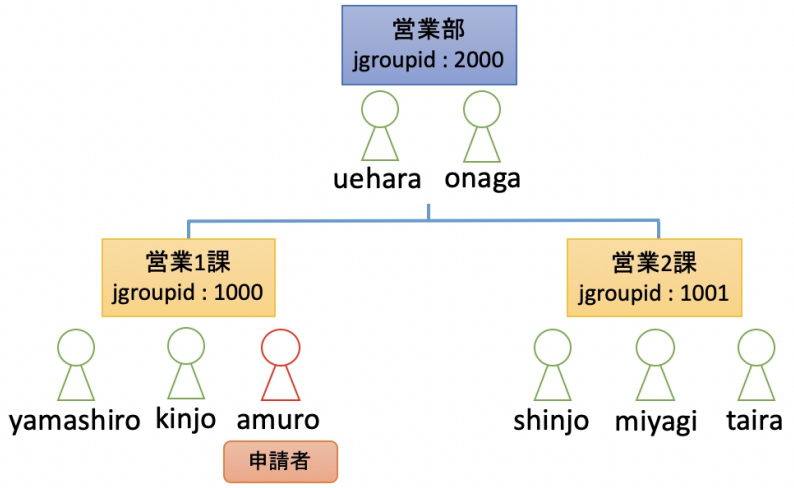


図 上位グループおよび下位グループに属するグループIDの取得

$allUsersの使い方

次の式は、全ユーザーのうちのうち、プリンシパル"nenkyu_finalist"を持つユーザIDを取得します。

```
$allUsers.find(By.principal("nenkyu_finalist"));
```

次の式は、全ユーザーのうちのうち、グループID1000に所属し、かつプリンシパル"nenkyu_finalist"を持つユーザIDを取得します。

```
$allUsers.find(By.jgroupId(1000), By.principal("nenkyu_finalist"));
```

By.principal と同様に、グループIDで絞り込むための By.jgroupid も用意されています。

```
By.jgroupid(グループID)
By.jgroupid(グループID,グループID,...)
```

コンマ区切りで複数のグループIDを指定することもできます。

次の式は、全ユーザーのうちのうち、グループIDが1000,1001,1002に所属し、かつプリンシパル"nenkyu_finalist"を持つユーザIDを取得します。

```
$allUsers.find(By.jgroupId(1000,1001,1002), By.principal("nenkyu_finalist"));
```

(455)

$rootGroupの使い方

$rootGroup は、グループ階層を設定している場合のみ利用できます。

次の式は、全グループを取得します。

```
$rootGroup.descendants();
```

次の式は、最上位グループを取得します。図の例では"管理部"、"第二技術部"、"第一技術部" が該当します。

```
$rootGroup.children();
```

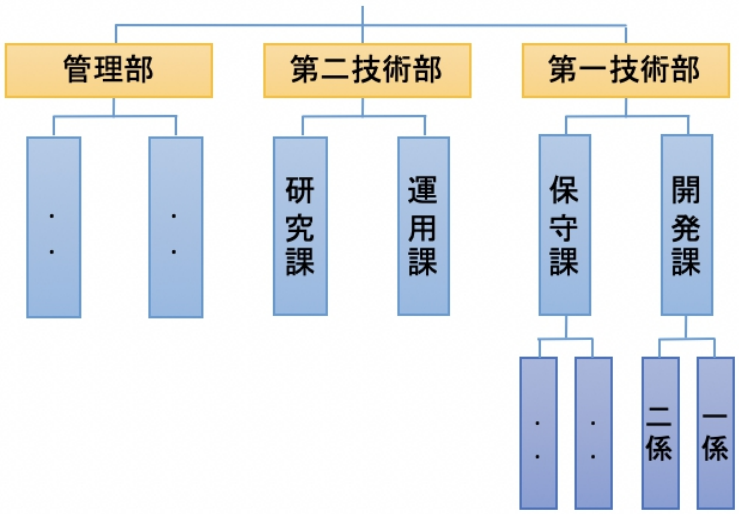


図 最上位グループの取得

仕様・制約

- フロー参加者をスクリプトで設定する場合、スクリプトは必ず return で対象者を返すようにしてください。null（対象者が定まらない）は許容されません。(*1)
- フロー参加者をスクリプトで設定した場合、スクリプトで取得された "次の承認対象者" は、ワークフロー適用モデルの "承認対象者(jfcWorkflowAdmitUser)" 項目に保存されます。このため、申請・承認時にスクリプトで次の承認対象者が確定された後にスクリプトを変更しても、次の承認対象者は変更されません。この仕様から、例えばスクリプトを実行するたびに対象者が変わるような仕組みを実現することはできません。(*2)
- フロー参加者スクリプトのデータベースの型は、各データベースの CLOB に相当する型を用いるようにしています。スクリプトのサイズの最大長は、データベースの CLOB 型の上限値となります。例えば Oracle では NCLOB となります。詳細は、自動生成された jfcparticipant_setting.ddl をご確認ください。
- このスクリプトは、ワークフロー処理中に複数回、実行されることがあります。何回実行されるかはフロー設定により変わるため、複数回実行されても結果が変わらないようにしてください。（具体的には、N回呼び出されることを前提としたり、N回目の呼び出しのときに処理を変える、というような手順を加えないようにしてください。）

(456) (457)

---

(455) find メソッドの引数は可変長となっています。順序は問いません。find(By.principal("xxx"), By.jgroupId(1000)) と記述しても結果は同じです。

(456) 1. R8.3.2まではエラー扱いとせず、nullの場合は内部でワークフローの再処理を行っていたためパフォーマンスの問題を引き起こす場合がありました。R8.3.3以降は、スクリプトがnullを返した場合、画面にエラーメッセージを表示するようにしました。

(457) 2. 例えば対象者がランダムに決まる、曜日によって決まる（月曜日ならAさん、火曜日ならBさん）といったことは実現できません。

その他の情報

- スクリプトでService/Daoクラス、ヘルパ、SQLを利用する場合は、あわせてこちらをお読みください。

フローイベントスクリプトを用意する(スクリプト)

定義方法

ワークフロー設定画面に用意された「フローイベントスクリプト」を定義します。

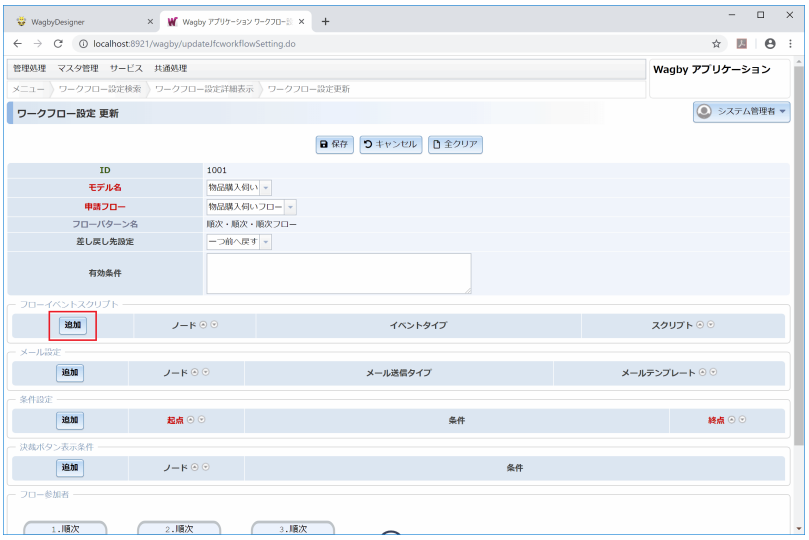


図 フローイベントスクリプトの設定(1)

ノードとイベントタイプに応じたスクリプトを設定することができます。

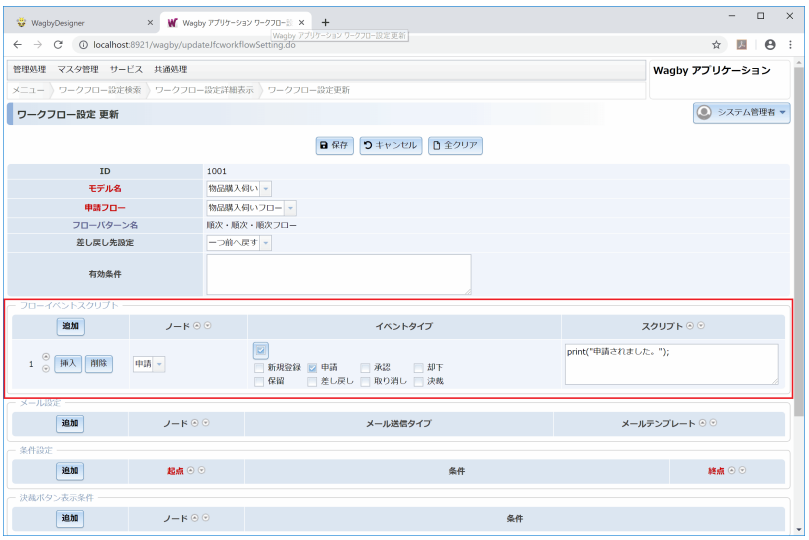


図 フローイベントスクリプトの設定(2)

例外を発生させてワークフローを進めないようにする (8.3.0)

スクリプト内で何らかのチェックを行い、その結果が不正と判断したときに Wagby の例外BusinessLogicException を投げることができます。ワークフローを先へ進めずにエラーと表示するような制御を行うことができます。

例

次のフローイベントスクリプトを「申請」ノード、イベントタイプ「申請」に記載します。(注意:これは必ずエラーとなるようにしたものです。)

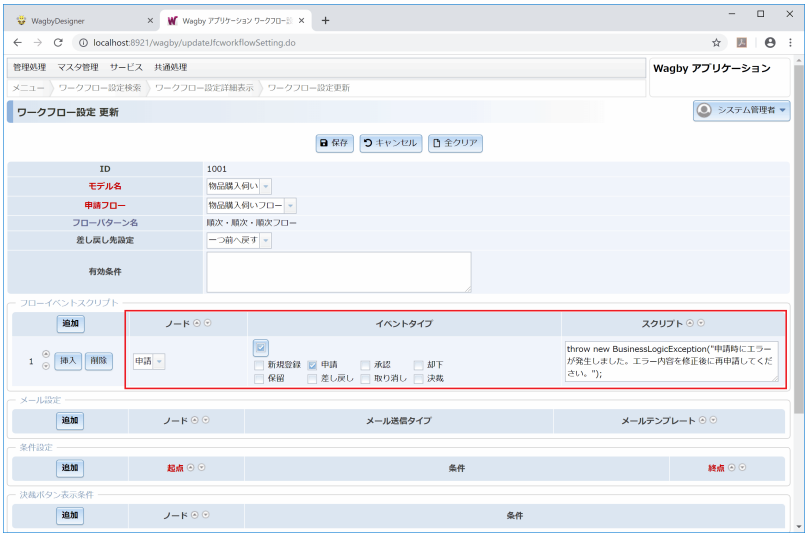


図 フローイベントスクリプトにBusinessLogicExceptionを設定する

```
throw new BusinessLogicException("申請時にエラーが発生しました。エラー内容を修正後に再申請してください。");
```

ワークフローの申請時にエラーとなり、上記エラーメッセージが画面に表示されます。ワークフローは進まないため、エラー発生後に再度申請ボタンが押下可能となっています。

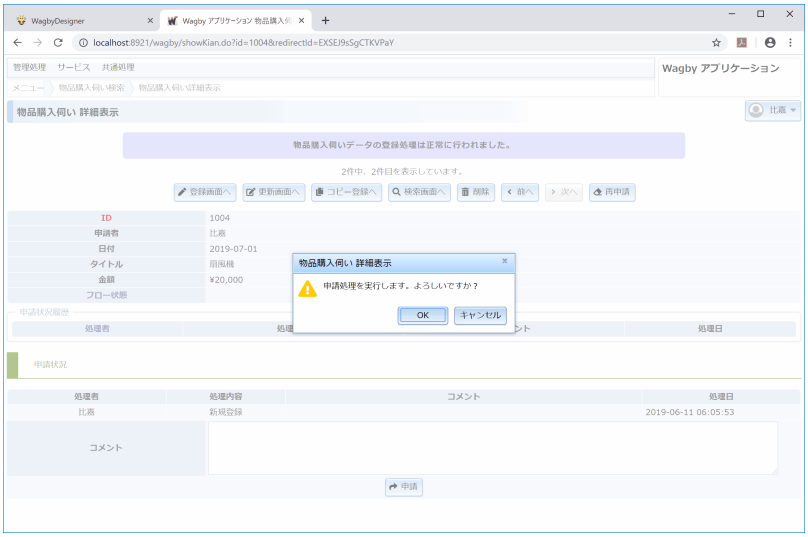


図 ワークフローの申請処理(1)

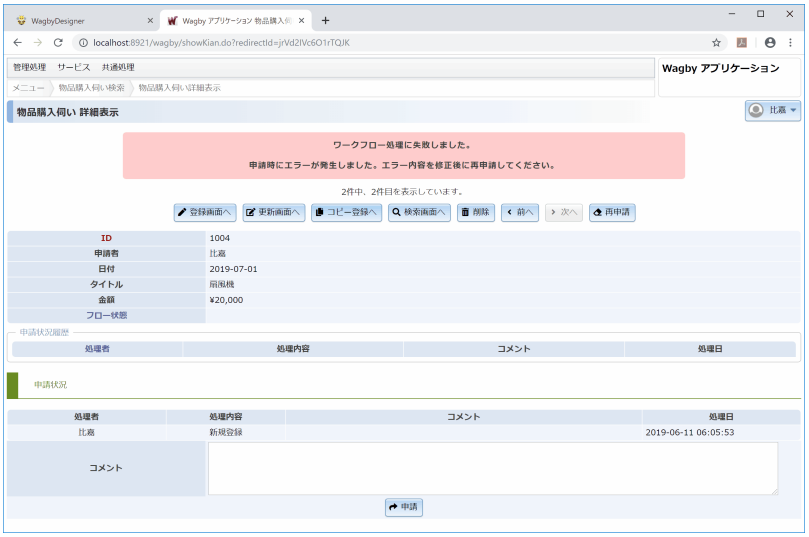


図 ワークフローの申請処理(2)

仕様・制約

- フロー参加者スクリプトのデータベースの型は、各データベースの CLOB に相当する型を用いるようにしています。スクリプトのサイズの最大長は、データベースの CLOB 型の上限値となります。例えば Oracle では NCLOB となります。
- 複数のフローイベントスクリプトを定義し、そのイベントタイプが重複している場合、このスクリプトはワークフロー処理の過程で複数回、実行されることになります。例えば異なるフローイベントスクリプトだがそれぞれで "申請" をチェックした場合や、決裁用のフローイベントスクリプトでイベントタイプに "承認" と "決裁" の両方を指定した場合などです。
- 運用中（テスト中を含む）にスクリプトを修正した場合、ワークフロー利用者(申請者、承認者など)をいったんログアウトし、再度ログオンすることで新しいスクリプトが有効になります。

スクリプトからワークフローを操作する(スクリプト)

申請処理を行うスクリプト

年休申請モデルを例に説明します。Wagbyの標準では年休申請モデルの新規登録後に（画面からの操作で）ワークフローの申請を行います。ここではスクリプトを用いて、新規登録と同時にワークフローの申請を行ってみます。

スクリプトの実行タイミングは、コントローラ > 登録 > 登録の実行 とします。

```
var workstateService = p.appctx.getBean("JfcworkstateService");
var insertJfcworkstateController = p.appctx.getBean("insertJfcworkstateController");

// 新しい workstate を用意する。
var leaveId = leave.appid;
var workstate = workstateService.getStartJfcworkstate("leave", leaveId, p);
workstate.event = "Application";// 申請を意味する
workstate.comment = "TEST";// 任意のコメント文
workstate.mainModelname = "leave";// モデルID

// 画面からの操作と同じように insertJfcworkstateController クラスを使う。
p.request.setAttribute("__jfc_retrieveData_entity", workstate);
p.o = insertJfcworkstateController.retrieveData(p);
insertJfcworkstateController.do_insert(p);
```

- JfcworkstateServiceクラスが提供するgetStartJfcworkstateメソッドを用いることで新しいworkstate オブジェクトを取得することができます。このオブジェクトに event, comment, mailModelname をセットします。それぞれフローイベントの種類、コメント、モデルIDとなります。

- この workstate オブジェクトを内部で利用している特別なキー __jfc_retrieveData_entity で request オブジェクトにセットしたあと、insertJfcworkstateController クラスが提供する retrieveData メソッドを呼び出します。戻り値を p.o にセットします。
- 同じく insertJfcworkstateController クラスが提供する do_insert メソッドを呼び出すとワークフローイベントがシステムに登録されます。これらの処理は画面からワークフローの申請処理を行なったときと同じ操作となっています。

[運用] ワークフローを手動で取り消す

フローイベントとは

ワークフローモデルのデータを新規登録したタイミングで、フローイベントモデルにも同じ主キー値でデータが登録されます。ワークフローが現在、どういう状況になるかは、このフローイベントモデルの最新値で把握できるようになっています。詳細は"ワークフローの状態とイベント"をお読みください。

操作例

user1が申請し、user2が承認、user3が決裁したワークフローの例を図に示します。このワークフローは完了しています。

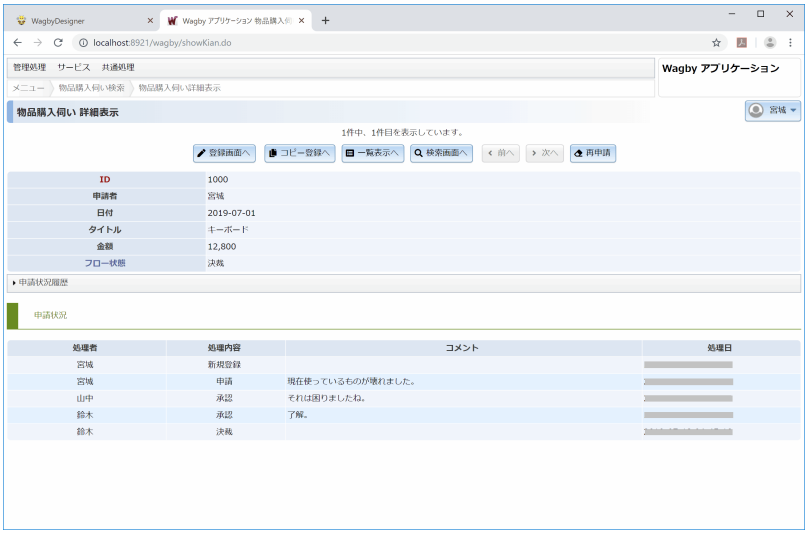


図 完了（決裁済み）のワークフローの例

フローイベントを削除する

システム管理者でログオンします。管理処理メニューから「ワークフロー ＞ フローイベント検索」を開きます。

（ここで対象モデルの主キーや処理者、処理日などで絞り込むことができます。）

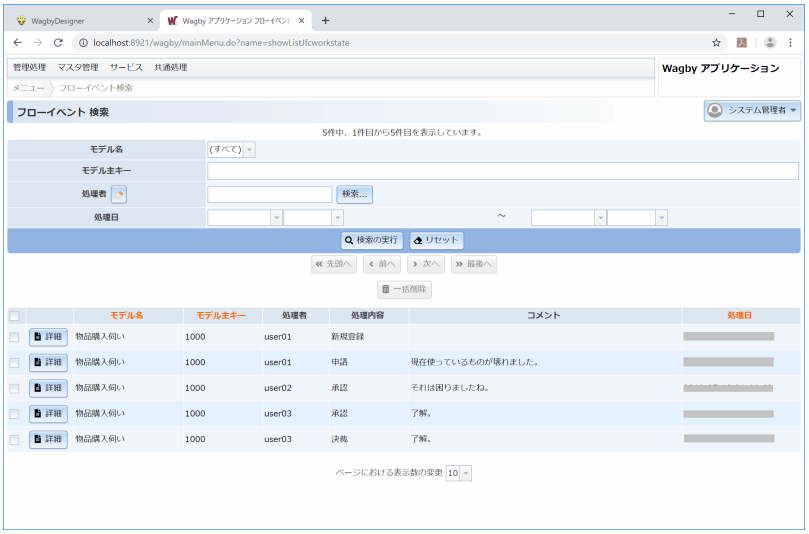


図 フローイベント検索

フローイベントの更新を行うことはできませんが、削除することができます。ここでは申請処理以前まで戻すとします。処理内容 ″申請″, ″承認″, ″決裁″ のフローイベントを選択し、一括削除します。

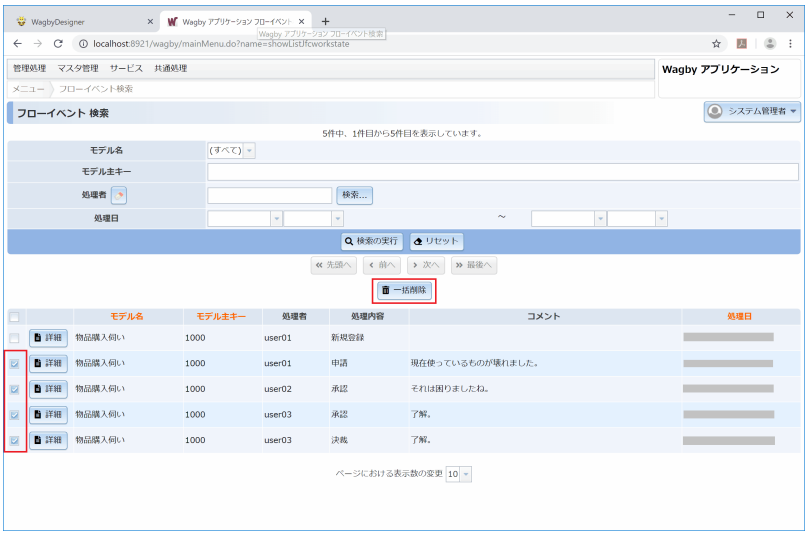


図 削除したいフローイベントを選択する

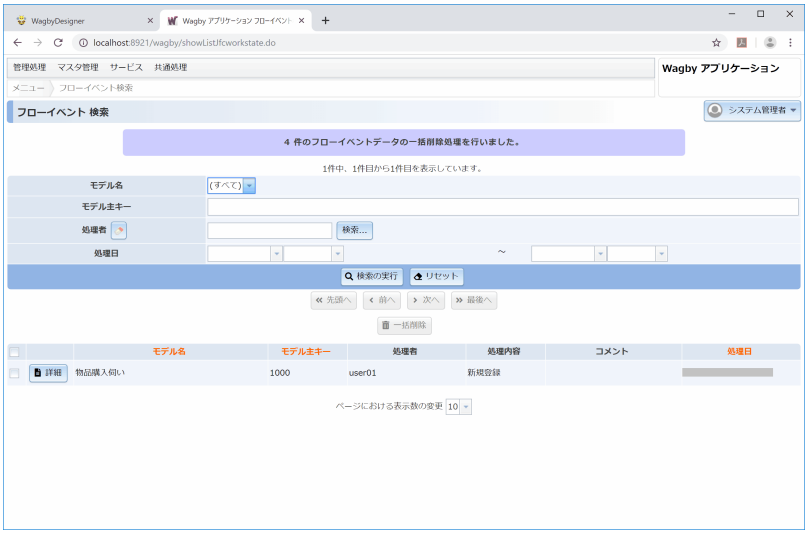


図 一括削除後のフローイベント

確認

再び（ワークフローを申請する）user1　でログオンします。フロー申請前の状態に戻っていることがわかります。

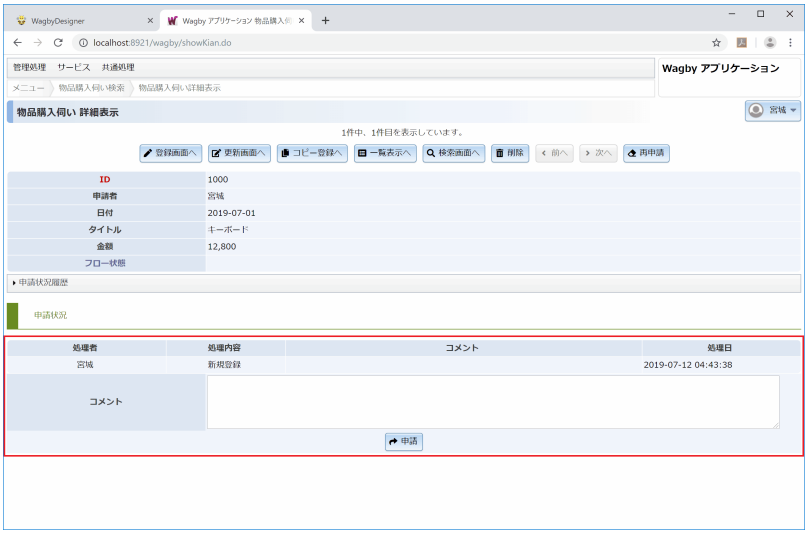


図　フロー申請前の状態

ご注意ください：この方法は、手動でワークフローイベントを削除するため、承認や決裁といった証跡に関する情報が失われます。

スクリプトからの利用

Wagbyが提供するワークフロー操作クラスWorkFlowManagerが提供するメソッドresetFlowEventを呼び出すことで、このモデルの、この主キーに紐づくフローイベントを消去することもできます。

具体的な利用方法は［応用］ワークフロー決裁後に、フローイベントをリセットして再申請させる のページをお読みください。

［応用］ワークフロー決裁後に、フローイベントをリセットして再申請させる（スクリプト）

実行例

図のワークフローは決裁済みのため、更新ボタンは非表示となっています。ここでオリジナルボタン「再申請」を用意しています。このボタンを押下するとフローイベントが消去されます。

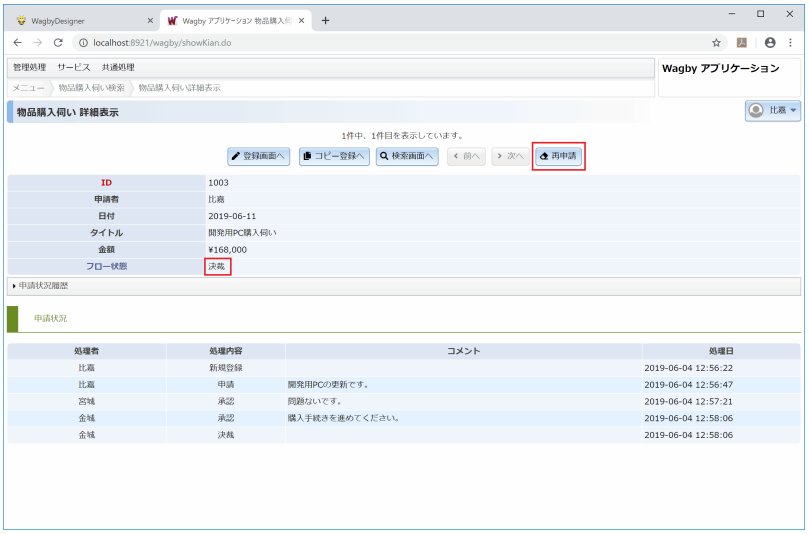


図 決裁済みワークフローに対して「再申請」ボタンを押下する

フローイベントが消去され、新規登録状態に戻ります。このとき、これまでのワークフローの承認記録を「申請状況履歴」という繰り返しコンテナにコピーするようにしています。

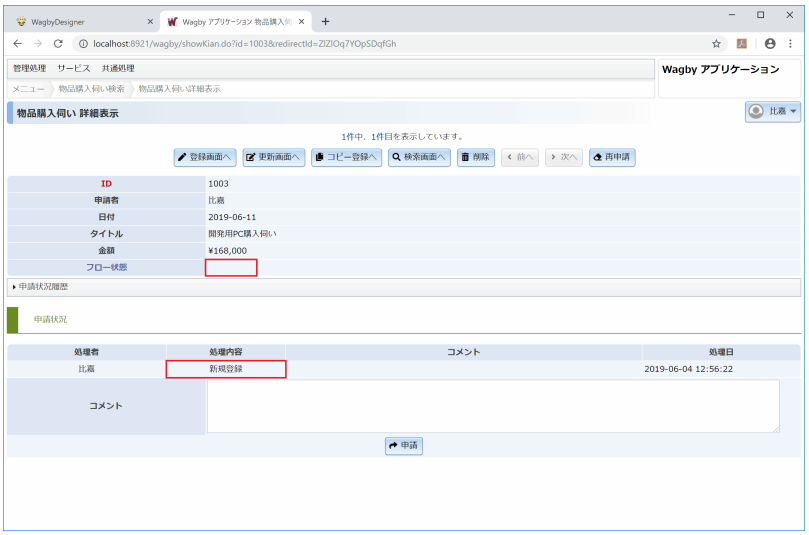


図 フローイベントが消去された

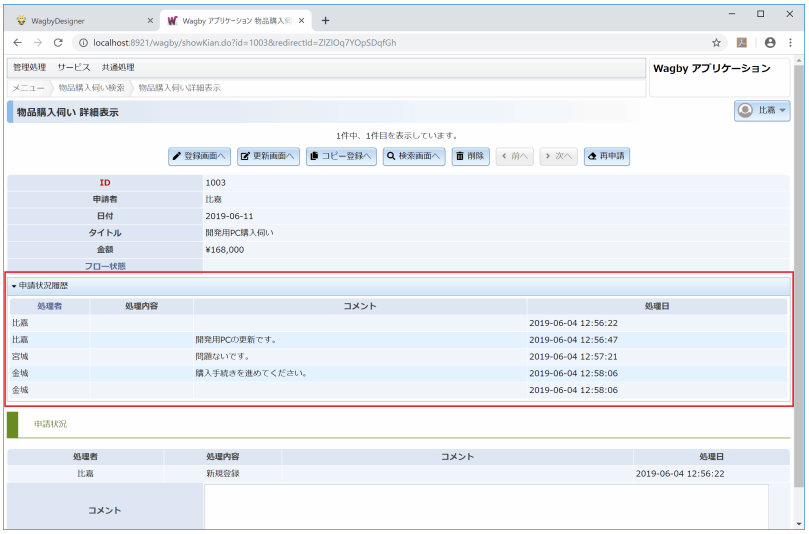


図 フローイベント内容が「申請状況履歴」にコピーされている

未申請となったため更新ボタンが押下できるようになります。ただし「申請状況履歴」は非表示（編集不可）としています。

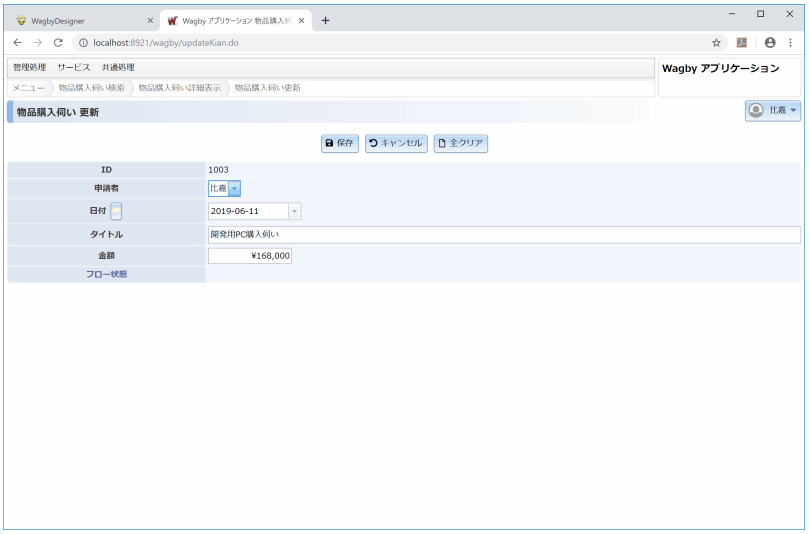


図 更新画面では申請状況履歴は非表示とする

定義方法

サンプル定義「物品購入伺い」のモデル定義を用意します。（このモデルの詳細は割愛します。）

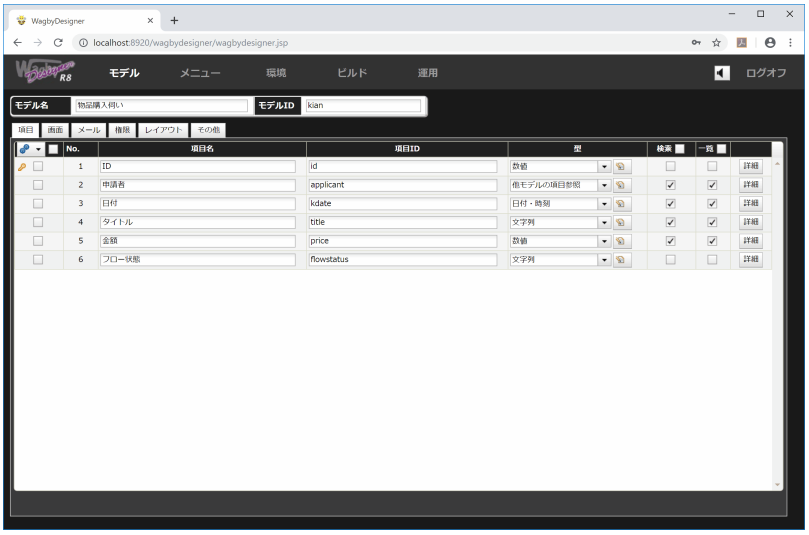


図 サンプルの物品購入伺いモデル

項目テンプレートとして用意されている「ワークフローイベント履歴コンテナ」を加えます。これは再申請処理時に、現在のフローイベントをコピーするための入れ物として用います。

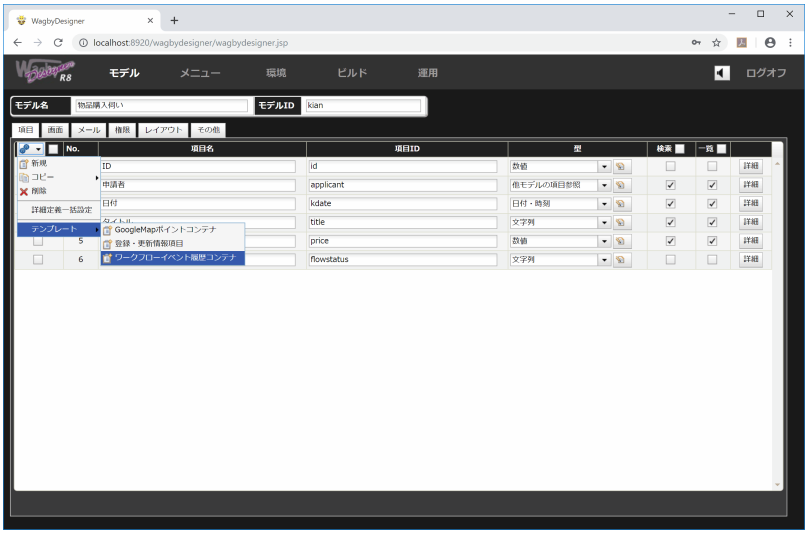


図　「ワークフローイベント履歴コンテナ」繰り返しコンテナを加える(1)

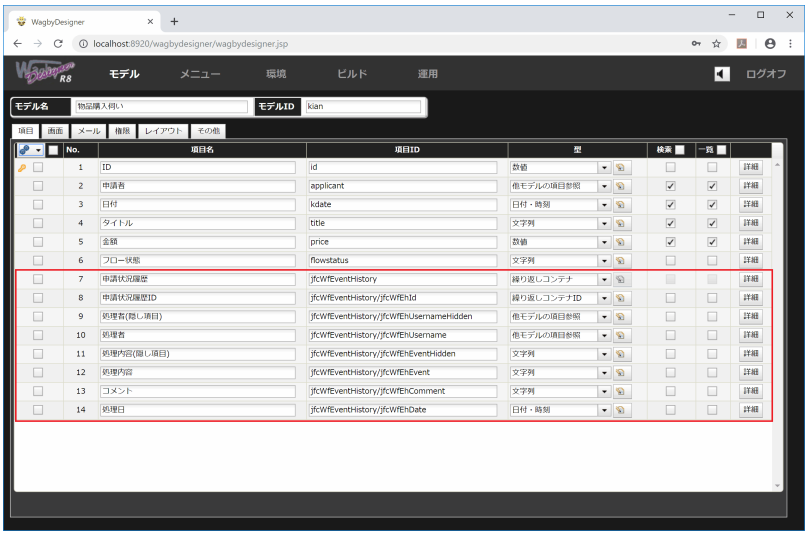


図　「ワークフローイベント履歴コンテナ」繰り返しコンテナを加える(2)

ワークフローイベント履歴コンテナの詳細

「ワークフローイベント履歴コンテナ」繰り返しコンテナの初期表示数は0となっています。

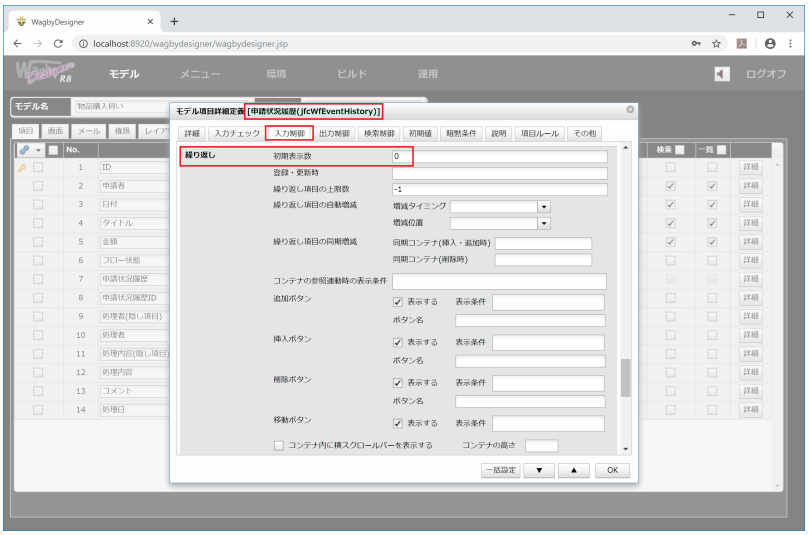


図 「ワークフローイベント履歴コンテナ」繰り返しコンテナの初期表示数

また、「ワークフローイベント履歴コンテナ」繰り返しコンテナは入力時に隠し項目となっています。これによって登録や更新画面では、これらの項目は表示されません。

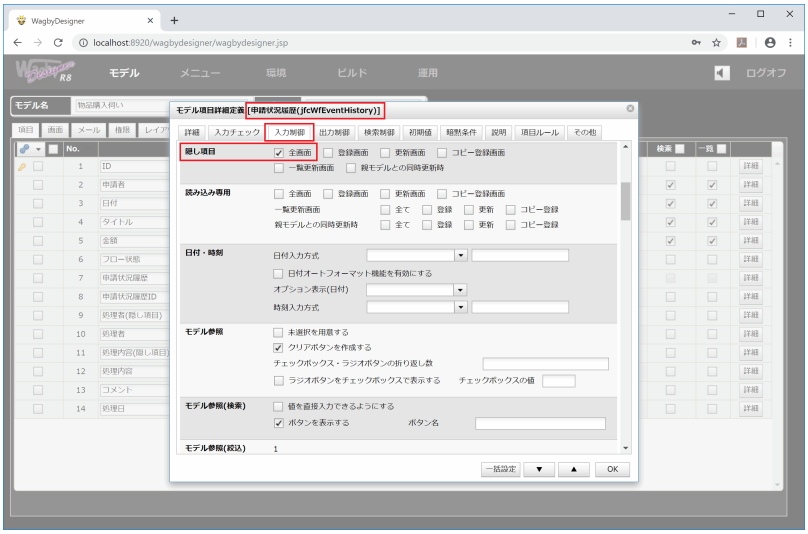


図 「ワークフローイベント履歴コンテナ」繰り返しコンテナは入力時、隠し項目となっている

「ワークフローイベント履歴コンテナ」繰り返しコンテナのID部である「ワークフローイベント履歴コンテナID」は表示時に隠し項目となっています。

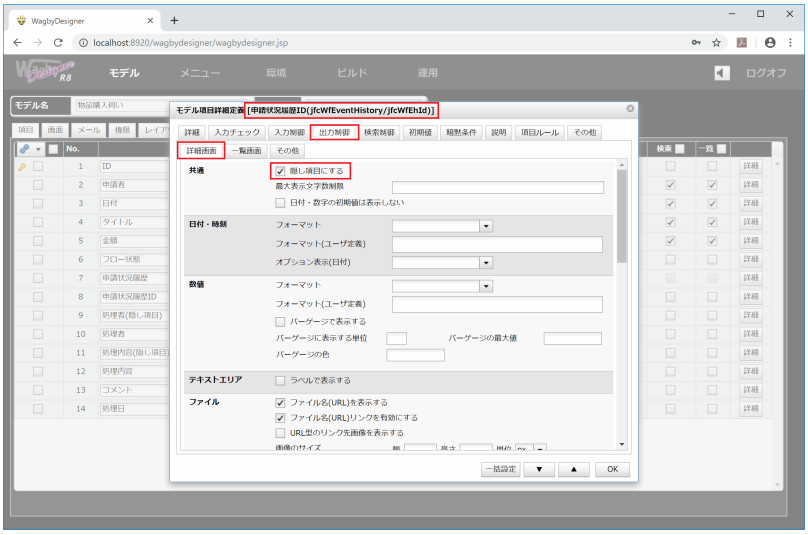


図 「ワークフローイベント履歴コンテナID」っは表示時に隠し項目となっている

「処理者(隠し項目)」はアカウントモデル(juser)と紐付いています。隠し項目とするため、入力タイプは「検索画面」となっています。（隠し項目の場合は「検索画面」とすることがもっともよい方法です。）

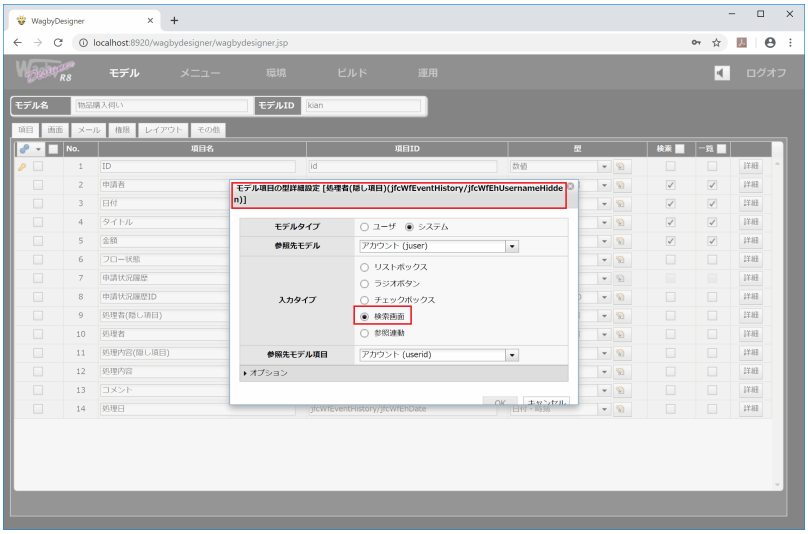


図 「処理者(隠し項目)」の設定

画面に表示する項目として「処理者」を別途、用意しています。参照連動を使ってアカウントモデル(juser)の「名前」が設定されています。この項目は表示専用であるため、データベースに保存されません。

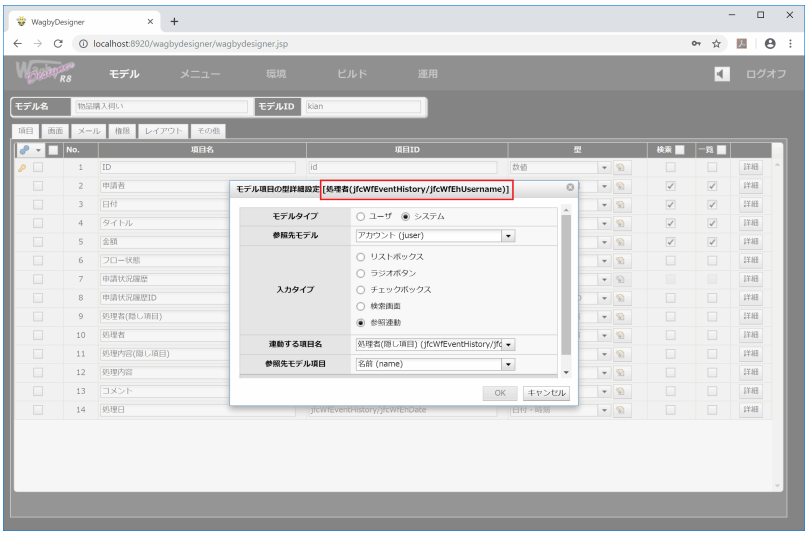


図 「処理者」の設定(1)参照連動

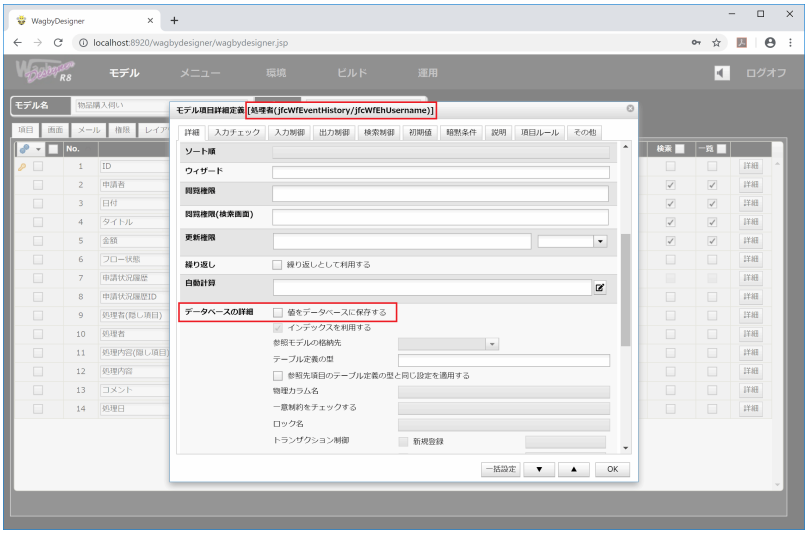


図 「処理者」の設定(2)データベースに保存しない

「処理内容(隠し項目)」は、ワークフローのイベント情報が転記されるものです。ここは Wagby のワークフローエンジンが内部で扱う "Start" や "End" といった文字列が格納されます。画面に表示しないため、隠し項目となっています。

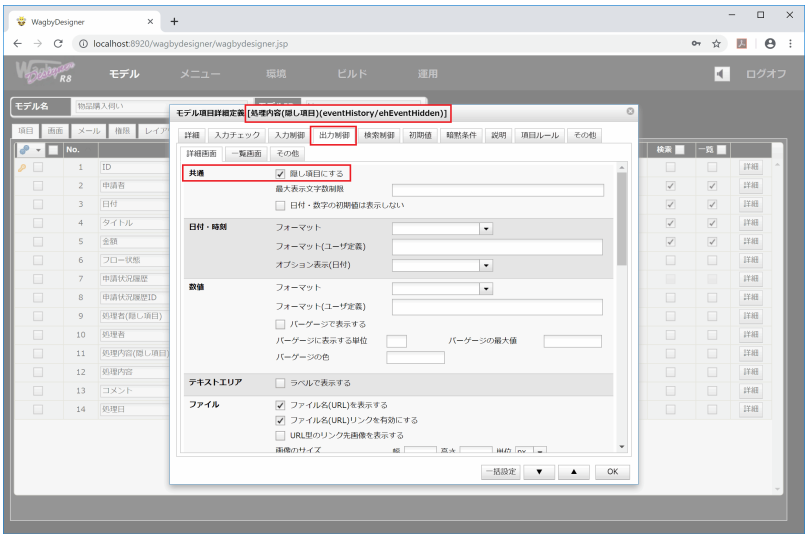


図 「処理内容(隠し項目)」の設定

「処理内容」は、上記ワークフローのイベント情報から、"申請" や "決裁" といった表示用の文字列に変換するために関数 WF_EVENT を使います。この項目は計算で求まるため、データベースに保存しないようになっています。

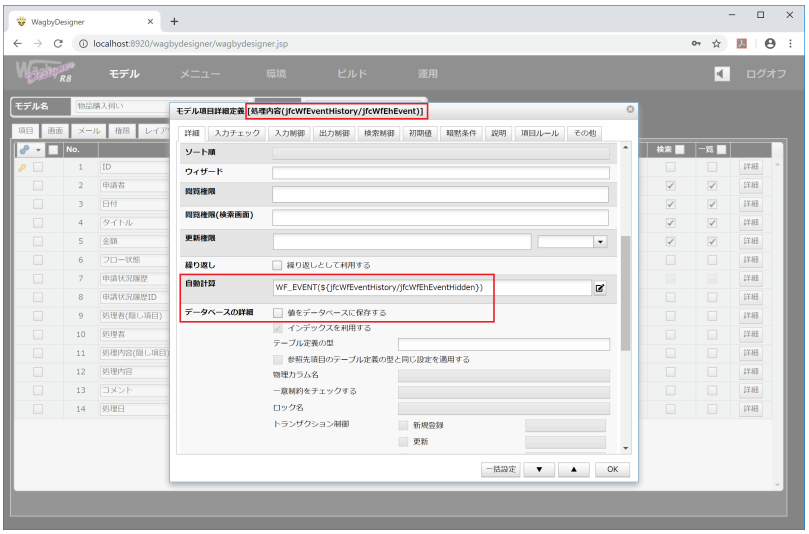


図 「処理内容」はWF_EVENT関数で求める

このモデルの詳細画面に、オリジナルボタンを用意します。ここではボタンのイベントを "Original1" としておきます。

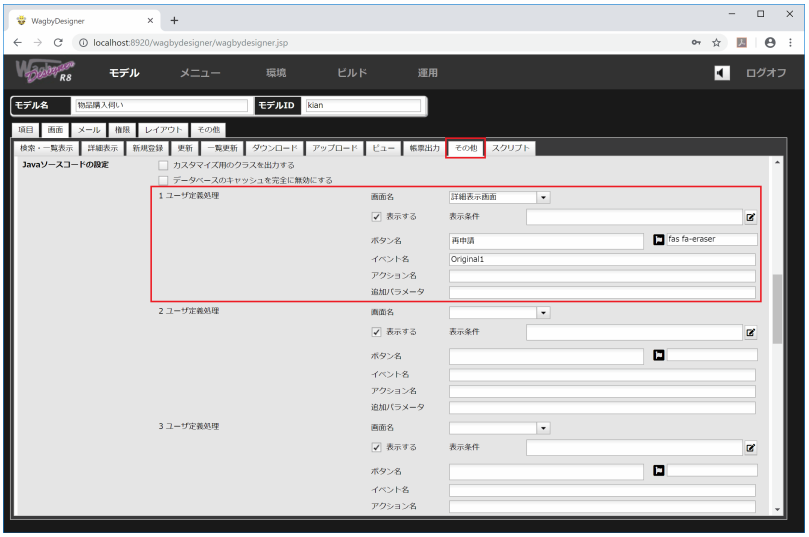


図 再申請を行うためのオリジナルボタンを用意する

再申請を行うスクリプト

オリジナルボタンのスクリプトは Designer ではなく、直接、ファイルを作成します。今回のサンプルはモデル名が kian で、イベント名を Original1 としたため、customize/webapp/WEB-INF/script/kian フォルダを用意し、ここに ShowKian_Original1.js というファイルを作成します。内容は次のとおりです。

```
var WorkFlowManager = Java.type("jp.jasminesoft.wagby.app.WorkFlowManager");
var EventHistoryClass = Java.type("jp.jasminesoft.wagby.model.kian.JfcWfEventHistory");
var Integer = Java.type("java.lang.Integer");
var Timestamp = Java.type("java.sql.Timestamp");

function process() {
  var modelId = "kian";
  var id = Integer.valueOf(p.request.getParameter("id"));
  print("再申請処理 modelId:["+modelId+"], id:["+id+"]");
  var wman = WorkFlowManager.getInstance(p);
  backupFlowEvent(modelId, id, wman, p);
  wman.resetFlowEvent(modelId, id, p);
  return "redirect:showKian.do?id=" + id;
}

function backupFlowEvent(modelId, id, wman, p) {
  print("フローイベントバックアップ処理 modelId:["+modelId+"], id:["+id+"]");
  var workstateList = wman.getJfcworkstateList(modelId, id, p);
  if (!workstateList || workstateList.length == 0) {
    return;
  }
  var kianService = p.appctx.getBean("KianEntityService");
  var kian = kianService.findById(id, true);
  for (var i=0,l=workstateList.length; i<l; i++) {
    var workstate = workstateList[i];
    var eventHistory = new EventHistoryClass();
    eventHistory.jfcWfEhUsernameHidden = workstate.username;
    eventHistory.jfcWfEhEventHidden = workstate.event;
    if (workstate.event !== "End") {
      eventHistory.jfcWfEhComment = workstate.comment || "";
    }
    eventHistory.jfcWfEhDate = Timestamp.valueOf(workstate.insertDate);
    kian.addJfcWfEventHistory(eventHistory);
  }
  kianService.update(kian);
}
```

・　スクリプトで、Wagbyが提供するワークフロー操作クラスWorkFlowManagerが提供するメソッドresetFlowEventを呼び出すことで、このモデルの、この主キーに紐づくフローイベント(jfcworkstateモデル)を消去します。（正確には"新規登録"イベントだけを残し、それ以外を消去します。）
・　フローイベントを消去する前に、現時点のフローイベントを起案モデルの「ワークフローイベント履歴コンテナ」繰り返しコンテナに転記します。このためにWorkFlowManagerクラスが提供するメソッドgetJfcworkstateListを使い、フローイベントを取得しています。

[応用] ワークフロー差し戻し後の再申請処理で、一度承認した人をスキップして承認待ちの方へ進める（スクリプト）

実行例

例として "順次・順次・順次フロー" で説明します。申請者を "higa", 承認者を "miyagi", 決裁者を "kinjo" とします。

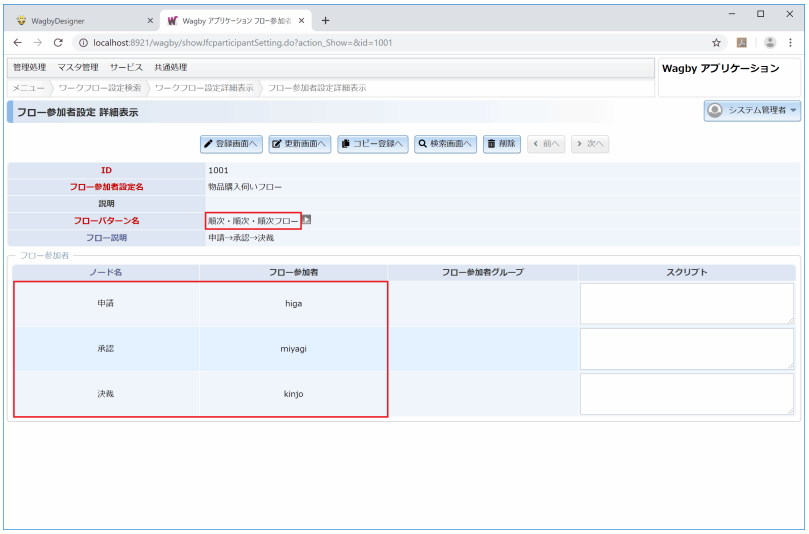


図 サンプルのワークフロー定義

"higa" が申請し、"miyagi" が承認しましたが、決裁者である "kinjo" が差し戻しを行いました。（このワークフローは差し戻し時には申請者へ戻す、としています。）

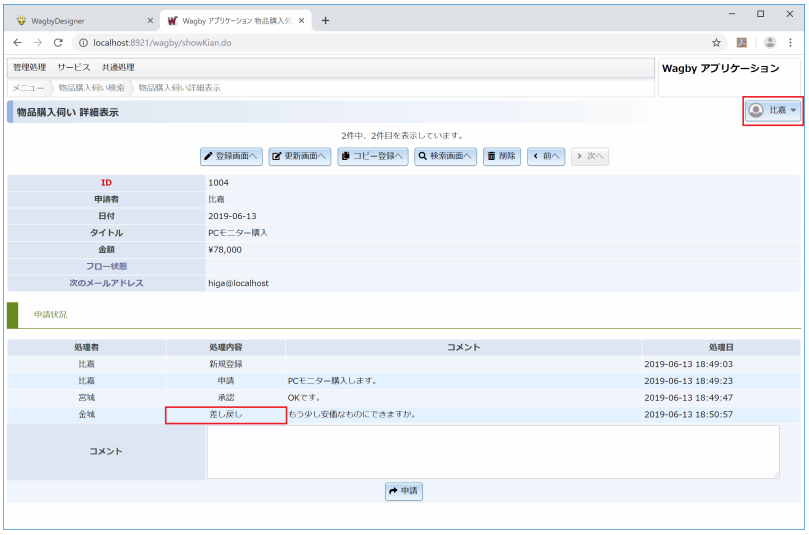


図 決裁者による差し戻しが発生

申請者 "higa" が諸条件を変更し、再び申請処理を行います。

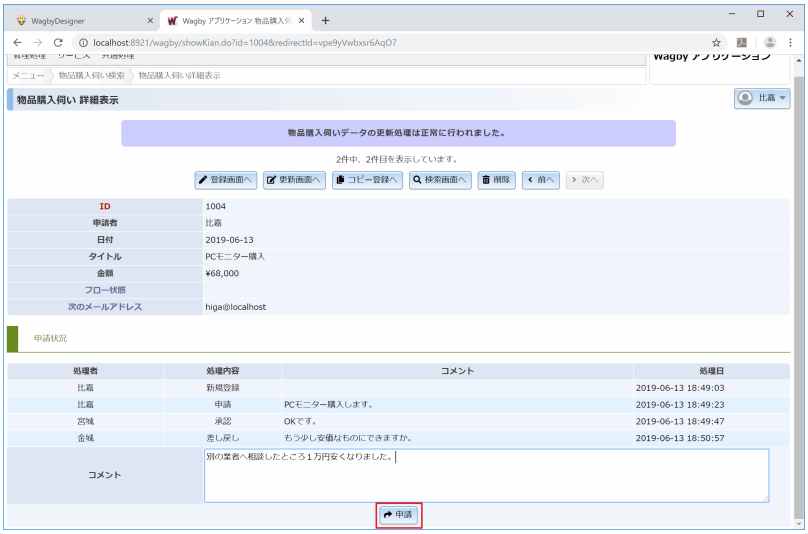


図 再申請処理

先ほど承認済みであった″miyagi″は自動承認されます。再び決裁者″kinjo″まで進みます。

また、このタイミングで″miyagi″には自動承認を行なったというメールが送信されています。

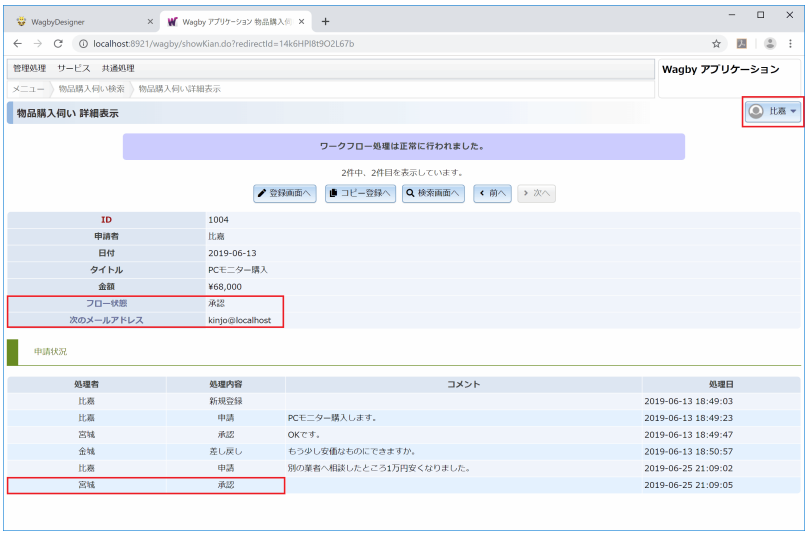


図 miyagiは自動承認され、決裁者kinjoへ進む

アカウント ″miyagi″ でログオンし、このデータを確認します。自動承認であるため、ワークフロー「承認」ボタンは表示されません。

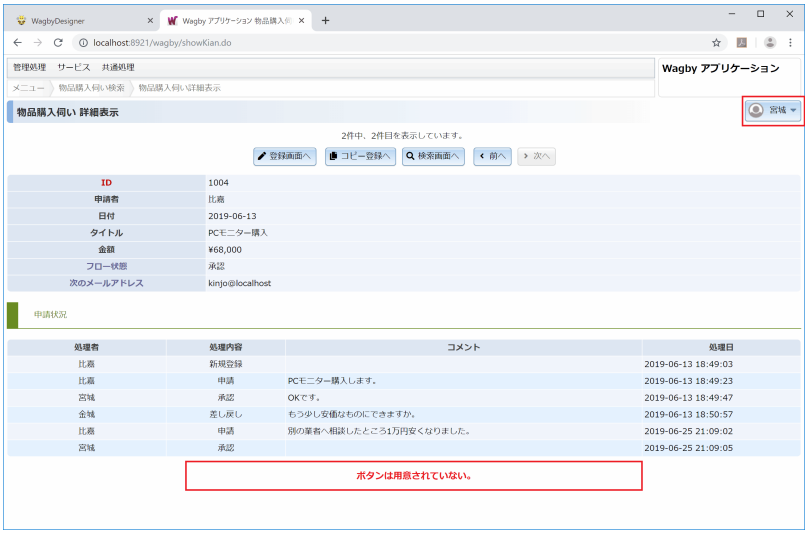


図 自動承認のため、承認ボタンは表示されない

アカウント"kinjo"でログオンし、このデータを確認します。決裁待ちであることがわかります。

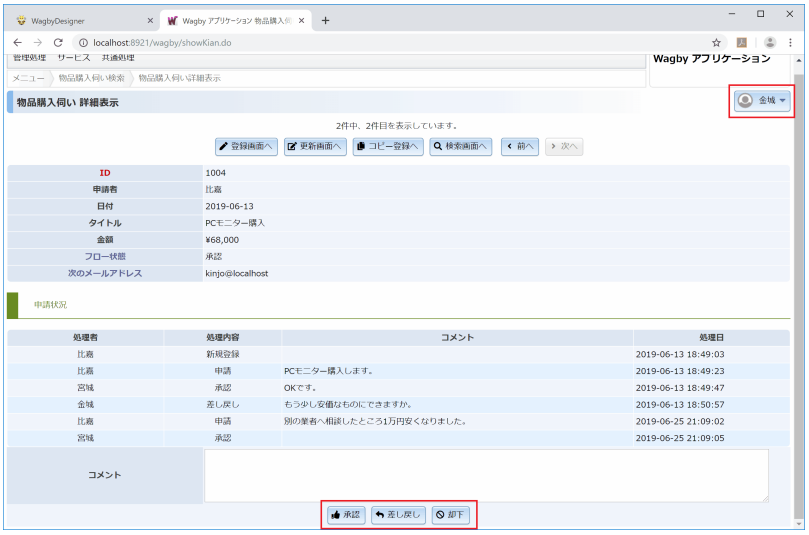


図 kinjoの決裁待ち

定義方法

juserモデルへのメールアドレス項目の追加

ワークフロー処理でメール送信を行うため、juserモデルへメールアドレス項目を追加します。

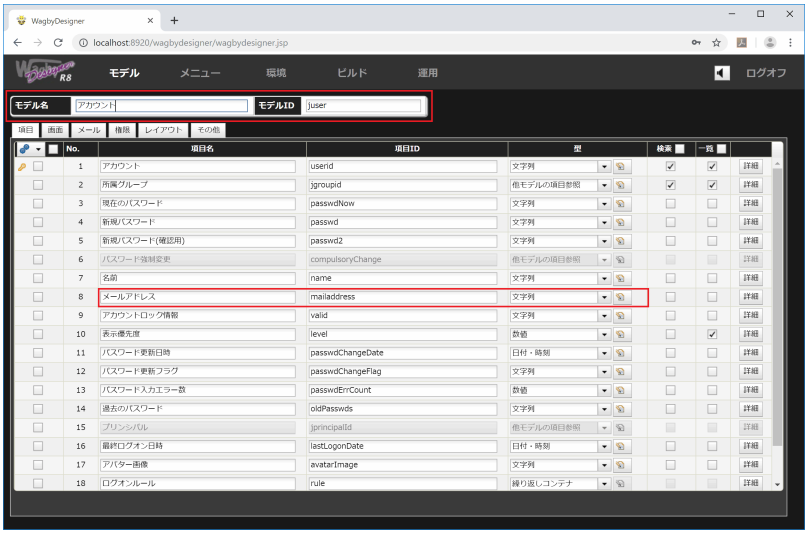


図 メールアドレス項目の追加

物品購入伺いモデルを用意する

上の例で示した物品購入伺いモデルを用意します。

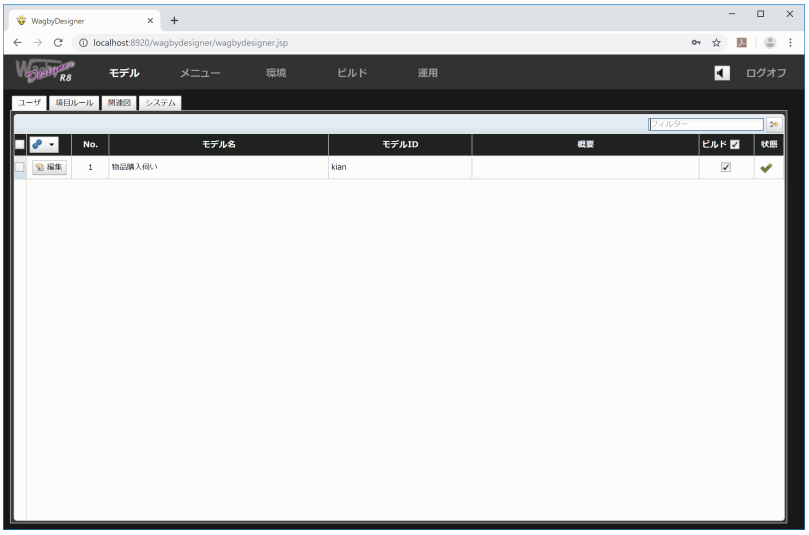


図 物品購入伺いモデルの定義(1)

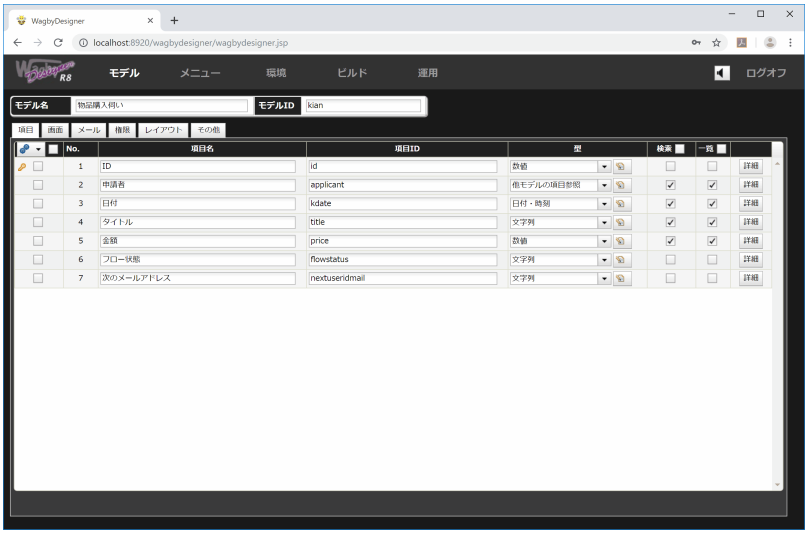


図 物品購入伺いモデルの定義(2)

「フロー状態」項目はワークフロー関数 FLOWSTATUS を適用します。データベースには保存しない項目としてください。

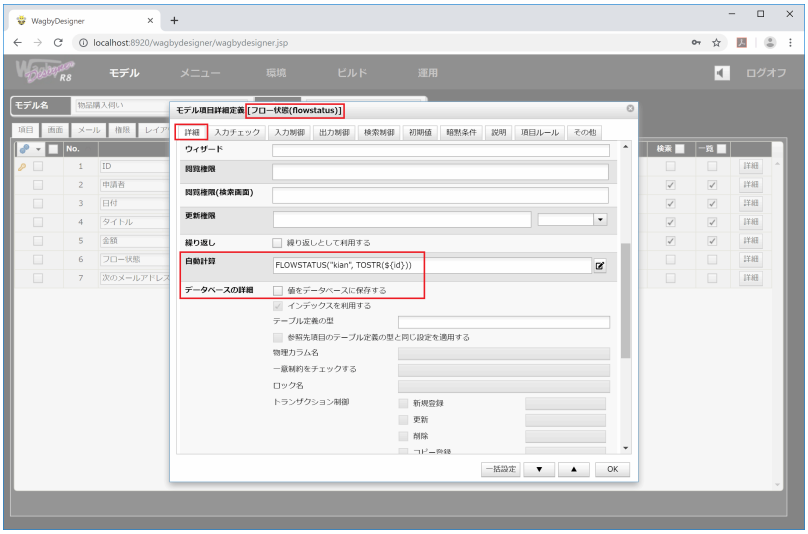


図 フロー状態項目

「次のメールアドレス」項目はワークフロー関数 WF_NEXT_USERID_MAIL を適用します。データベースには保存しない項目としてください。

```
WF_NEXT_USERID_MAIL("juser","userid","mailaddress")
```

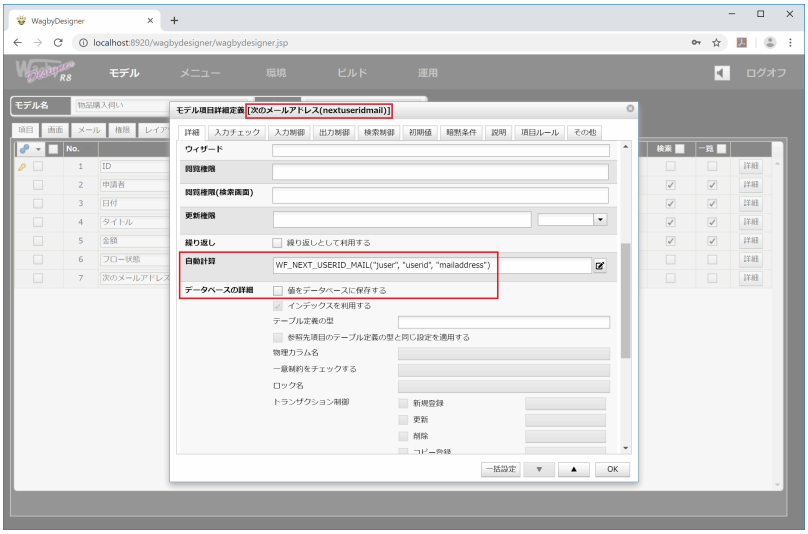


図 次のメールアドレス項目

フローイベントスクリプトを用意する

このワークフローでは、申請ノードでの申請イベント時に、スクリプトを実行するようにします。スクリプトで自動承認処理を記述します。

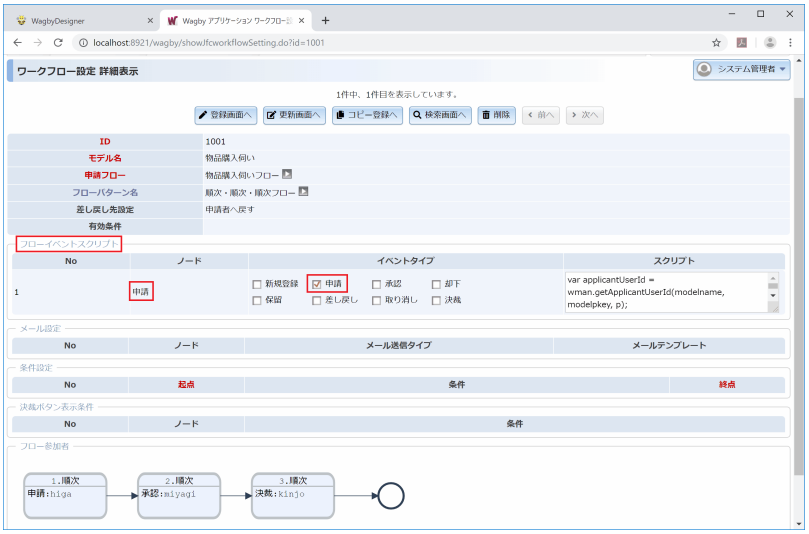


図 フローイベントスクリプトの設定(1)

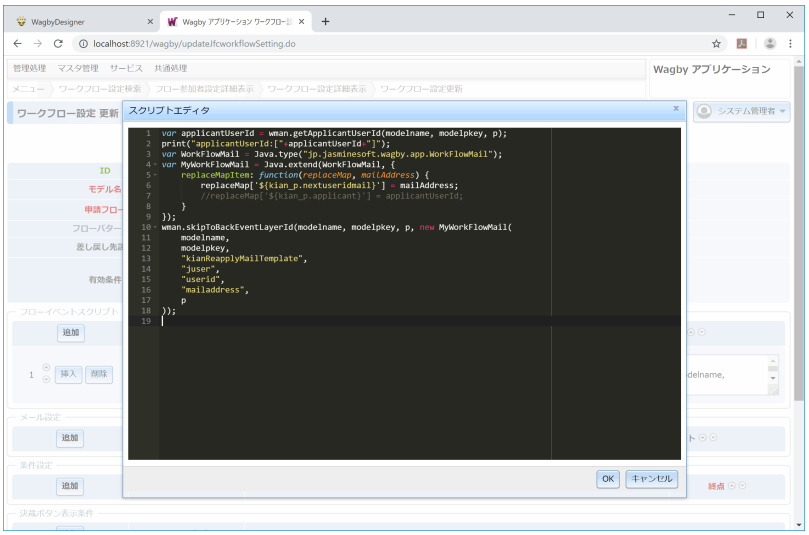


図 フローイベントスクリプトの設定(2)

フローイベントスクリプト (1)

設定したフローイベントスクリプトの例を示します。

```
var applicantUserId = wman.getApplicantUserId(modelname, modelpkey, p);
//print("applicantUserId:["+applicantUserId+"]");
var WorkFlowMail = Java.type("jp.jasminesoft.wagby.app.WorkFlowMail");
var MyWorkFlowMail = Java.extend(WorkFlowMail, {
    replaceMapItem: function(replaceMap, mailAddress) {
        replaceMap['${kian_p.nextuseridmail}'] = mailAddress;
        //replaceMap['${kian_p.applicant}'] = applicantUserId;
    }
});
wman.skipToBackEventLayerId(modelname, modelpkey, p, new MyWorkFlowMail(
    modelname,
    modelpkey,
    "kianReapplyMailTemplate",
    "juser",
    "userid",
    "mailaddress",
    p
));
```

- 暗黙変数 wman はフローイベントスクリプトで利用できます。実体は Wagby が提供する WorkFlowManagerオブジェクトです。
- 暗黙変数 modelname と modelpkey もフローイベントスクリプトで利用できます。それぞれ対象モデルIDと、主キーになります。
- 変数 applicantUserId に対象データに関する申請者のユーザID（例: "higa"）をセットします。
- Wagbyが提供するWorkFlowMailクラスを利用してメール送信の設定を行います。このソースコードは wagbydesigner/webapps/wagbydesigner/WEB-INF/env/work/template_javasrc/jp/jasminesoft/wagby/app/WorkFlowMail.java に用意されています。（パッケージ名を変更している場合は、パッケージ部 jp/jasminesoft/wagby を適切に読み替えてください。）
- このスクリプトで、スクリプトWorkFlowMail クラスが提供する replaceMapItem メソッドをオーバーライドします。（元のクラスでは、このメソッドは空で実装されています。）第一引数 replaceMap には、メールテンプレートで上書きされるプレースホルダが含まれています。第二引数にはメールアドレスが渡されるようになっています。開発者は replaceMap 内の「送信先メールアドレス」に相当する値に、第二引数のメールアドレスを再設定するコードを用意してください。
- 上のスクリプトではコメントとしていますが、変数 applicantUserId も利用できるため、プレースホルダに申請者のユーザIDをセットすることもできます。（メール文中にこのプレースホルダを埋め込んでおくと、申請者のユーザIDに置換されます。）
- 自動承認処理は wman（実体は WorkFlowManager オブジェクト）の skipToBackEventLayerId メソッドを呼び出すことで実現します。第一引数 modelname, 第二引数 modelpkey, 第三引数 p までは固定です。第四引数は上で replaceMapItem メソッドをオーバーライドした MyWorkFlowMail クラスからオブジェクトを生成して渡します。この第四引数がない形でメソッドを呼び出すこともできます。この場合は自動承認処理のみ行い、スキップされた承認者へのメール送信は行われません。
- 第四引数 MyWorkFlowMail オブジェクトを渡す場合、コンストラクタに7つの引数を指定します。最初の二つである modelname と modelpkey は固定です。第三引数は（開発者が事前に用意する）メールテンプレートのIDを指定します。このメールテンプレートで自動承認メールの文面を用意してください。第四引数はメールアドレスが格納されているモデルID、第五引数はそのモデルにある、ユーザIDを示す項目ID、第六引数はそのモデルにある、メールアドレスを示す項目IDです。これらはワークフロー関数 WF_APPLICANT_MAIL と同じ形式です。第七引数には p を指定してください。

フローイベントスクリプト (2)

上のフローイベントスクリプトでは、送信するメールのテンプレートは常に固定でした。

次に示すサンプルでは、ワークフローデータの申請状況を取得し、データが2件以上ある場合は「再申請」と判断するようにしています。再申請についての詳細は "ワークフロー決裁後に、フローイベントをリセットして再申請させる" をお読みください。

ここでは再申請の場合は "kianReapplyMailTemplate" のメールテンプレートが利用され、新規の申請の場合は "kianApplyMailTemplate" が利用されるようにしています。

```
var WorkFlowMail = Java.type("jp.jasminesoft.wagby.app.WorkFlowMail");
var MyWorkFlowMail = Java.extend(WorkFlowMail, {
  replaceMapItem: function(replaceMap, mailAddress) {
      replaceMap['${kian_p.nextuseridmail}'] = mailAddress;
  }
});
var workstateList = wman.getJfcworkstateList(modelname, modelpkey, p);
if (workstateList && workstateList.length > 2) {
  mailTemplateId = "kianReapplyMailTemplate";
  wman.skipToBackEventLayerId(modelname, modelpkey, p, new MyWorkFlowMail(
      modelname,
      modelpkey,
      "kianReapplyMailTemplate",
      "juser",
      "userid",
      "mailaddress",
      p
  ));
} else {
  var myWorkFlowMail = new MyWorkFlowMail(
      modelname,
      modelpkey,
      "kianApplyMailTemplate",
      "juser",
```

```
      "userid",
      "mailaddress",
      p
  );
  myWorkFlowMail.send();
}
```

フローイベントスクリプト (3) コメントの設定 (8.3.1)

WorkFlowManagerのskipToBackEventLayerId関数の第三引数にコメント文を指定できるようにしました。
自動承認時に、指定したコメント文が付与されます。（10行目の第三引数が、これになります。）

```
var WorkFlowMail = Java.type("jp.jasminesoft.wagby.app.WorkFlowMail");
var MyWorkFlowMail = Java.extend(WorkFlowMail, {
  replaceMapItem: function(replaceMap, mailAddress) {
      replaceMap['${kian_p.nextuseridmail}'] = mailAddress;
  }
});
var workstateList = wman.getJfcworkstateList(modelname, modelpkey, p);
if (workstateList && workstateList.length > 2) {
  mailTemplateId = "kianReapplyMailTemplate";
  wman.skipToBackEventLayerId(modelname, modelpkey, "自動承認されました", p, new MyWorkFlowMail(
      modelname,
      modelpkey,
      "kianReapplyMailTemplate",
      "juser",
      "userid",
      "mailaddress",
      p
  ));
} else {
  var myWorkFlowMail = new MyWorkFlowMail(
      modelname,
      modelpkey,
      "kianApplyMailTemplate",
      "juser",
      "userid",
      "mailaddress",
      p
  );
  myWorkFlowMail.send();
}
```

メニュー

ボタン設定

追加

「メニュー」設計画面から「ボタン設計」を開きます。

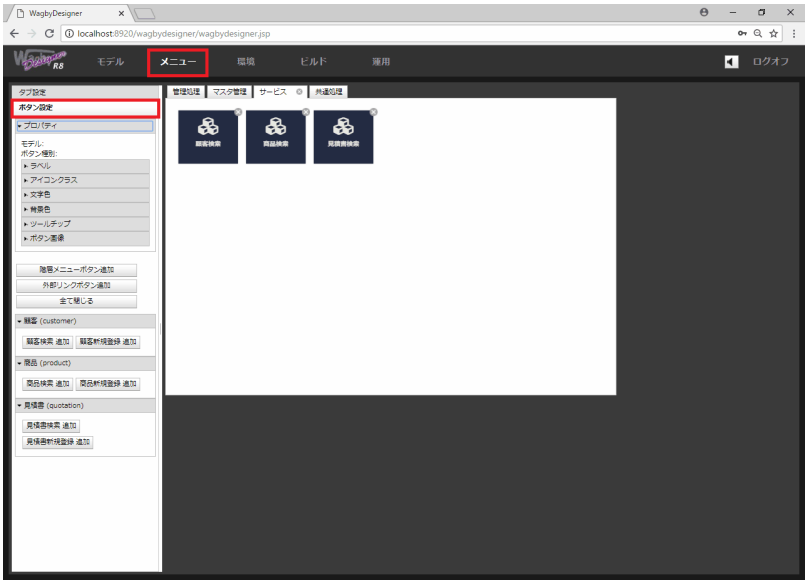


図 ボタン設計

追加できるボタンが、モデル毎に用意されています。ここでは顧客モデルの新規登録ボタンを追加してみます。

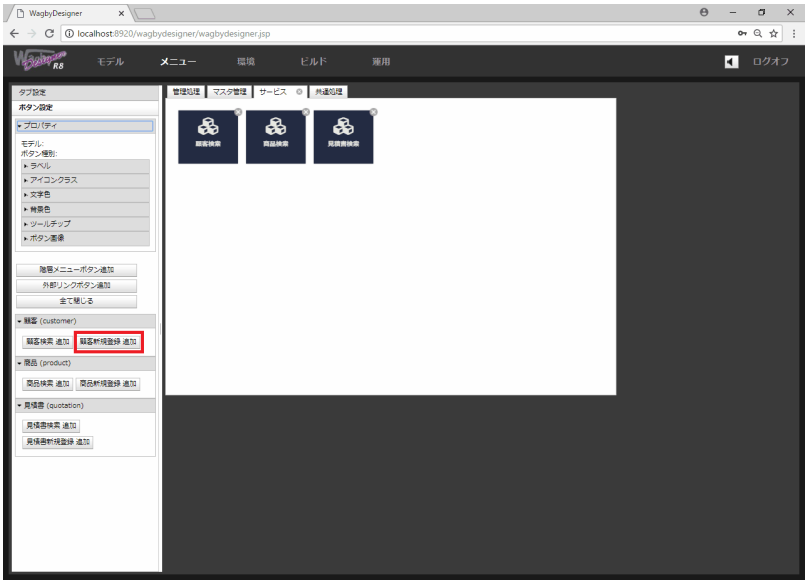


図 新規登録ボタンを追加する

ボタンが追加されます。

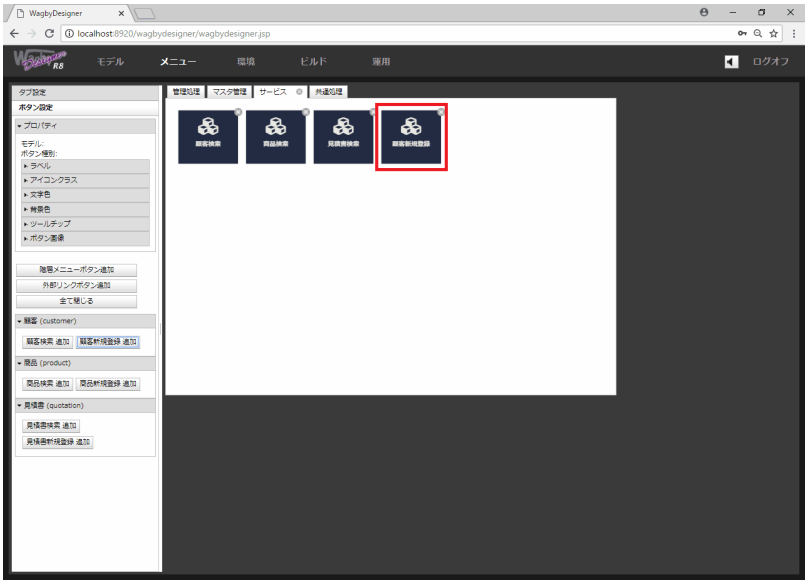


図 ボタンが追加された

詳細、更新、削除のボタンは追加できません。これらの画面はパラメータとして〝主キー〟が必要なため、別画面から遷移させることになります。

仕様

「サービス」タブが生成される条件

「検索・一覧表示」画面を作成する設定を行なっている場合に、標準のタブとして追加されます。

トラブルシューティング

設定したメニューがアプリケーションに反映されない

- 設定を変更したときは「更新」ボタンを押下してください。このボタンを押さないと、設定は反映されません。

図 更新ボタン

- メニューの設定情報の保存は、メニュー設計画面から（モデル等の）他の画面へ移動した際に行なわれます。この時「メニュー」へマウスポインタを重ねると 〝メニューデータ送信中〟 という表示が行なわれます。 メニュー情報が多い場合、この反映処理には十数秒以上の時間を要することがあります。 メニュー設定後、ビルド画面を表示して「ビルド実行」ボタンが押下可能になっていれば反映処理は完了しています。

設定したメニューがアプリケーションで表示されない

表示されないモデルの「権限」が空になっているか、あるいはメニュー権限が無効となっている可能性があります。〝認証・認可 〉 モデル単位の制御 〉 メニューの制御〟をお読みください。

移動

ドラッグ&ドロップでメニューの位置を変更することができます。

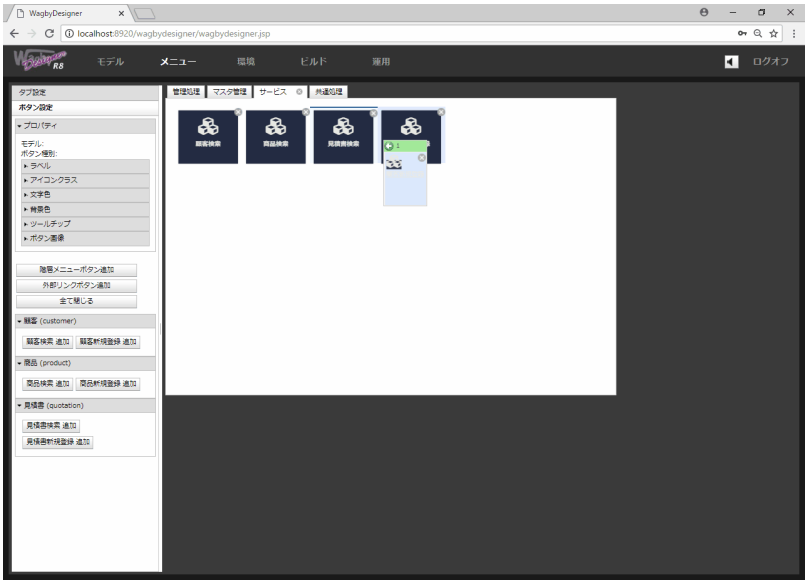


図 メニューボタンの位置を変更する

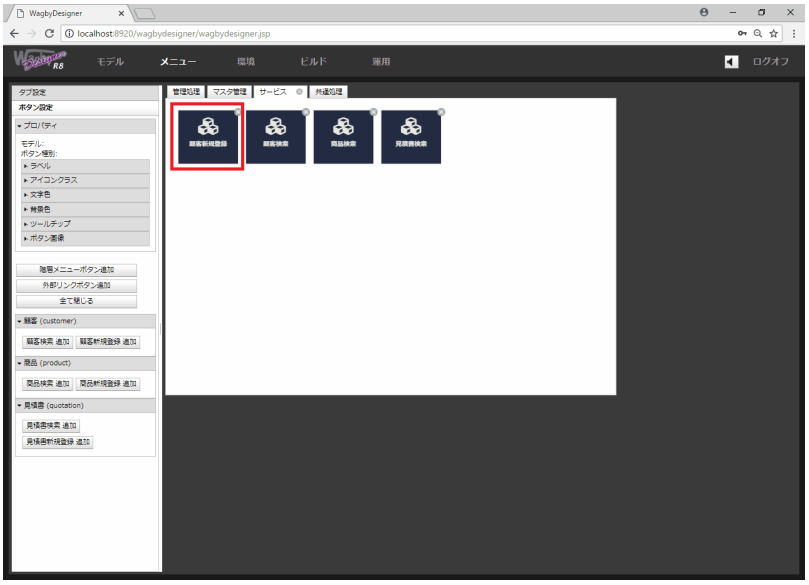


図 顧客新規登録を先頭に移動

タブ間の移動

あるタブに設定しているメニューを、他のタブに移動する場合は、一旦削除してから他のタブにメニューを追加する操作を行ってください。

背景色

はじめに、変更したいメニューボタンを選択します。（選択されたメニューの背景が青くなっています。） 次にボタン設定内の「ラベル・色設定」を開きます。ここで「背景色」を開き、色を選択します。

図 背景色の設定

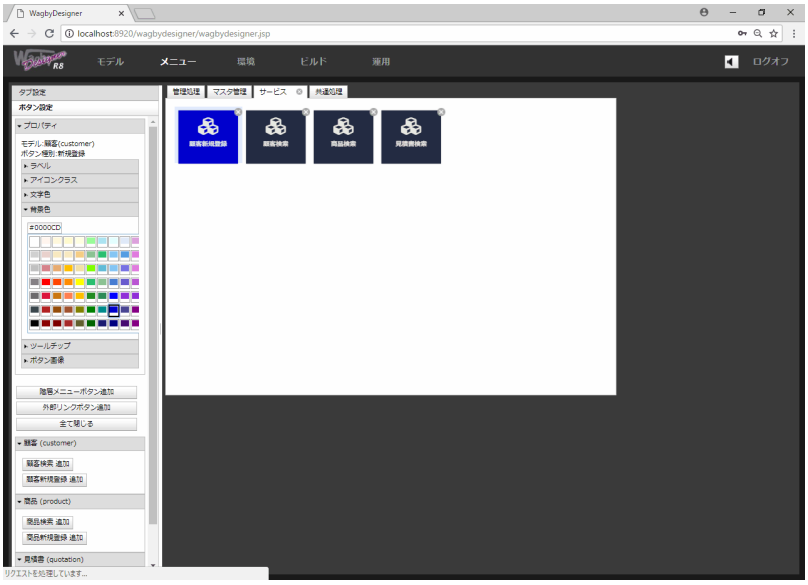


図 背景色が変更された

入力欄に〝#〟で始まる16進数の色を直接、指定することもできます。

文字色

ボタン設定内の「ラベル・色設定」から「文字色」を開きます。色を選択します。

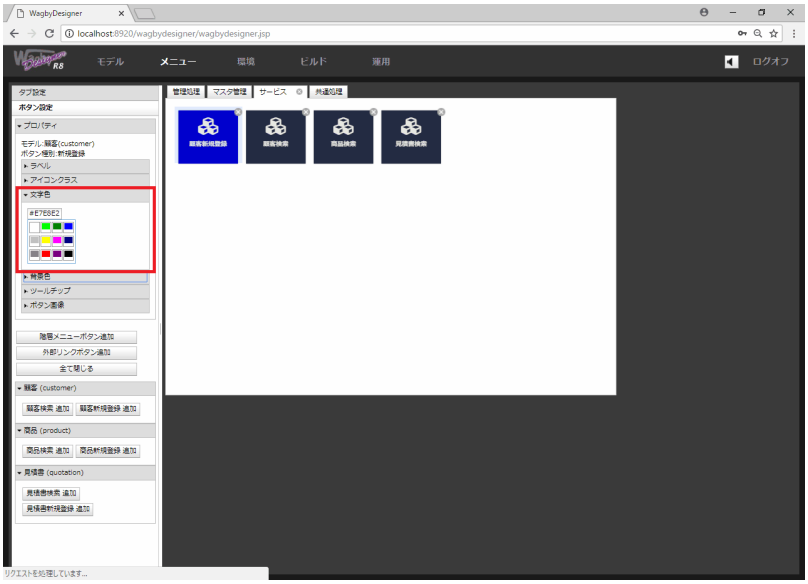


図 文字色の設定

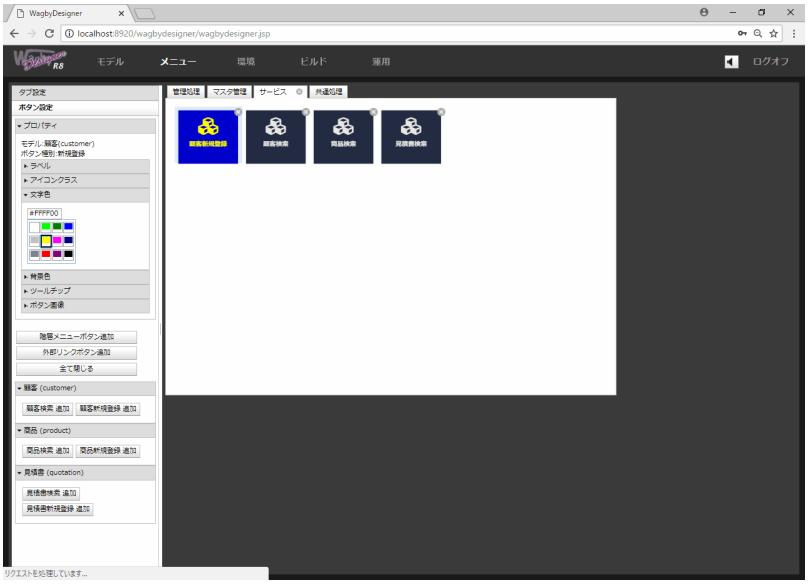


図 文字色が変更された

入力欄に〝#〟で始まる16進数の色を直接、指定することもできます。

文字の変更

ボタン設定内の「ラベル・色設定」から「ラベル」を開きます。文字を入力して「更新」ボタンを押下します。

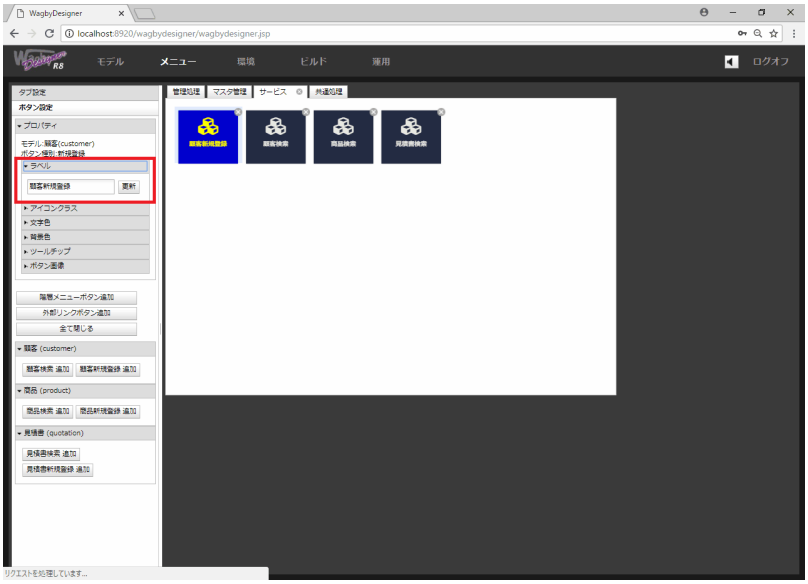


図 ラベルの設定

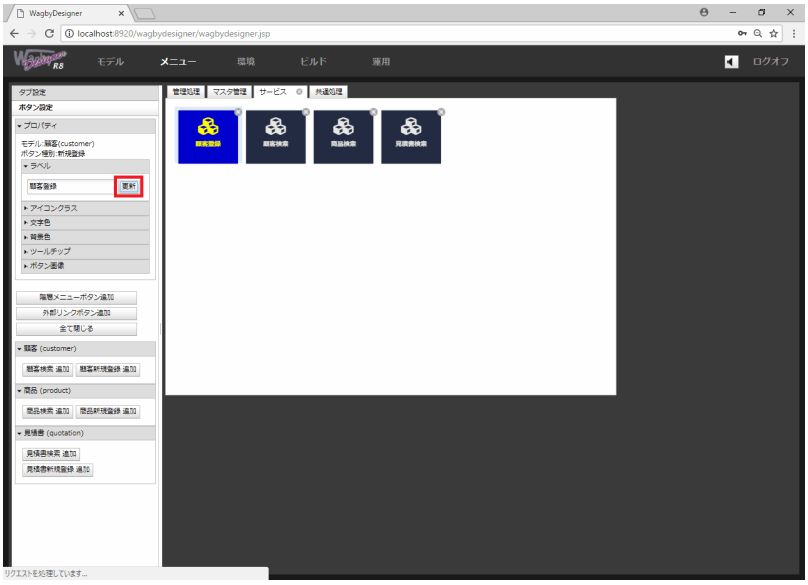


図 文言が変更された

改行を含める

ラベルに改行を含めることもできます。

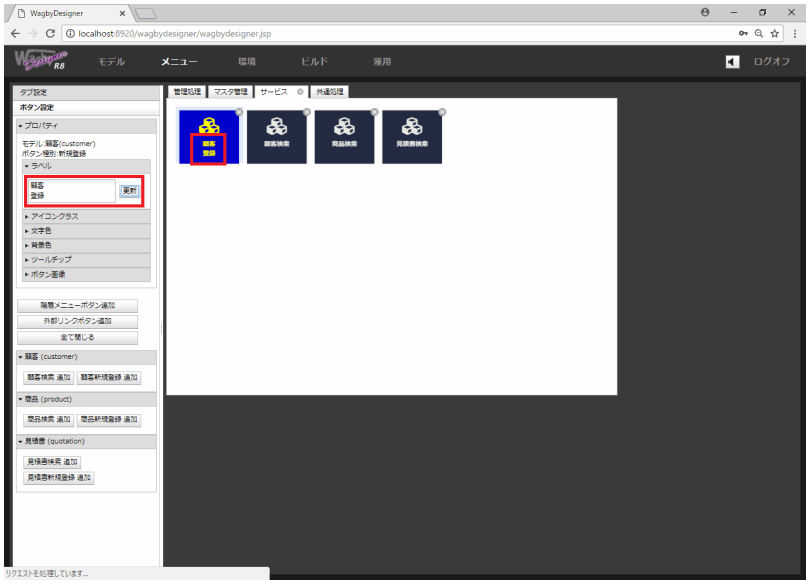


図 ラベルに改行を含める

制約

・　「戻る」ボタンの文言を変更することはできません。

アイコンの変更（8.1.0）

アイコンクラスを指定することで、メニューのアイコンを変更することができます。 Wagbyのメニューアイコンは Font Awesome サイトが提供する Web アイコンフォント（Web ブラウザで文字と同じように表示できるアイコン）を利用しています。

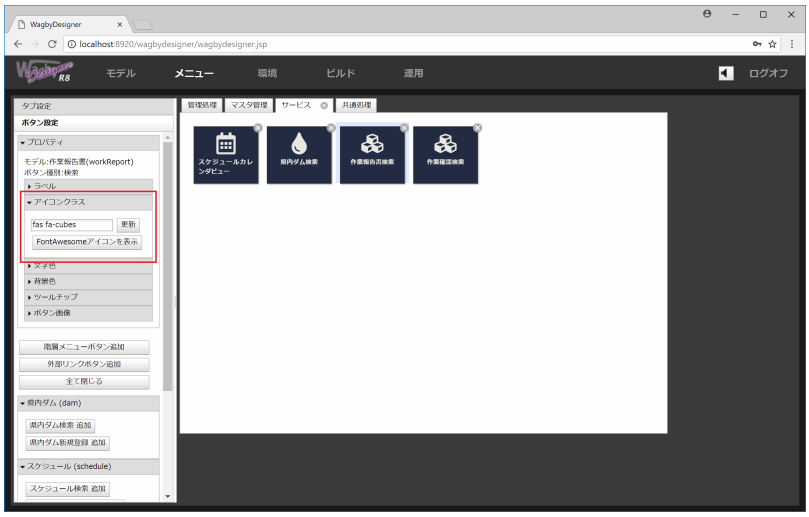


図 アイコンクラスの設定

"FontAwesomeアイコンを表示" ボタンを押下すると、別タブで Font Awesome のサイトが開きます。

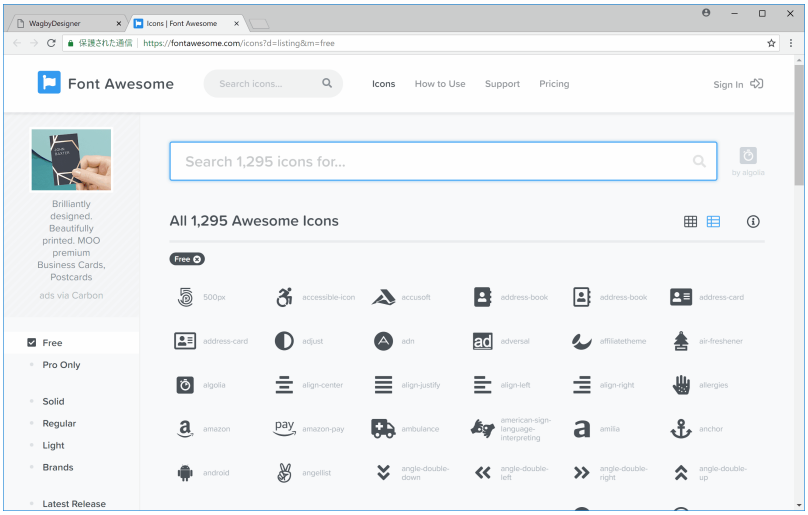


図 FontAwesomeサイト（画像は2018年8月時点のもの。表示内容は変わっている可能性があります。）

ここでは適当なアイコンを選択してみます。

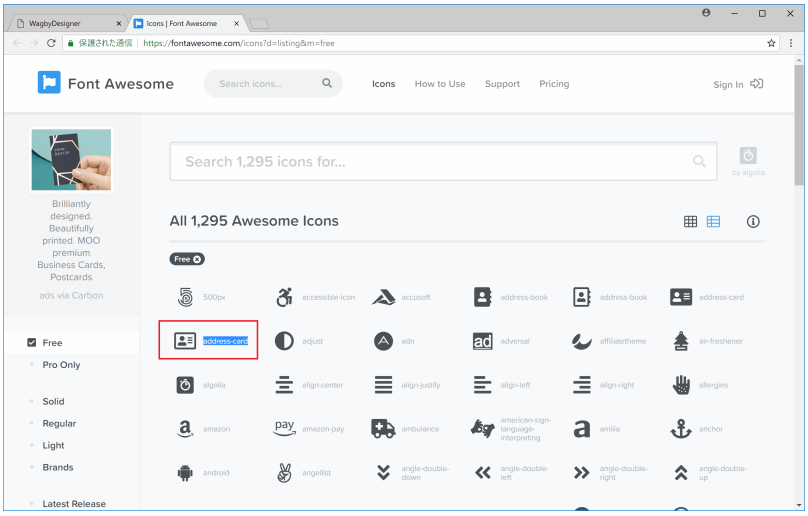


図 アイコンを一つ選択する

アイコンの下部に class ではじまる指定方法が明記されています。この文字列をコピーします。

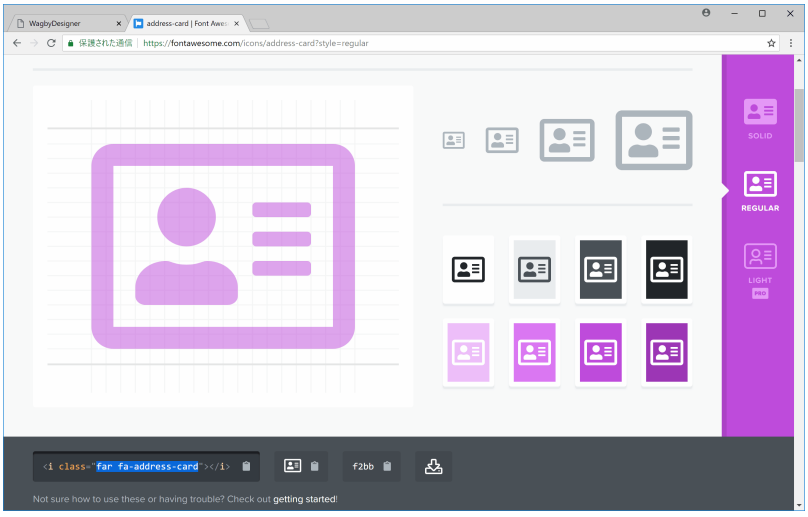


図 class指定をコピーする

Wagby Designer に戻ります。アイコンクラスに上の文字列をはりつけて、更新ボタンを押下します。するとアイコンが変わります。

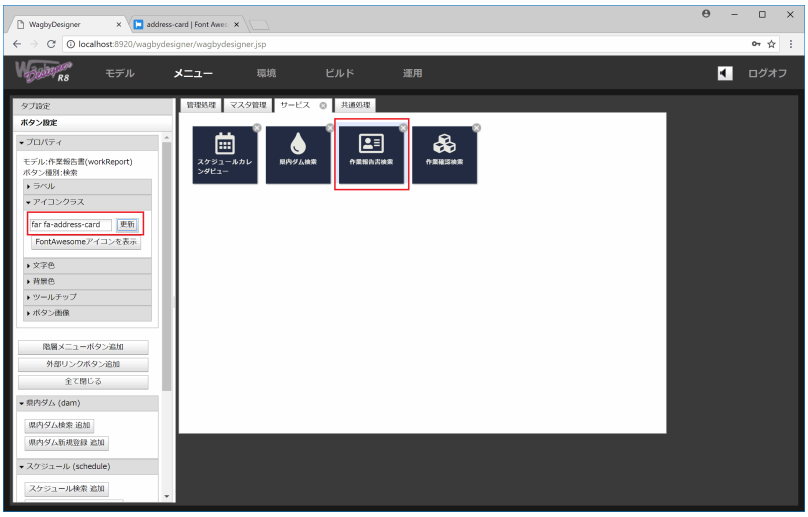


図 アイコンを変更する

ツールチップ

ツールチップとは、メニューにカーソルをのせると、説明文字を表示させる機能です。

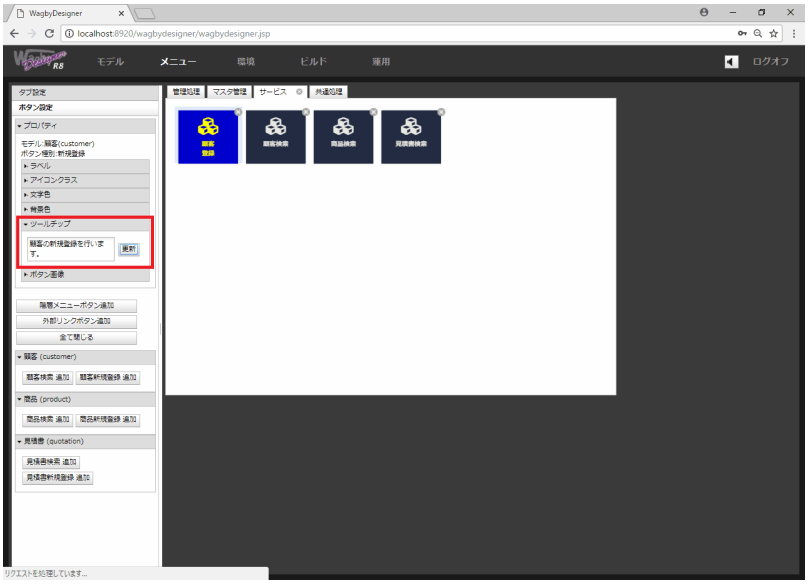


図 ツールチップの指定

画像を指定する

メニューのアイコンを指定することができます。

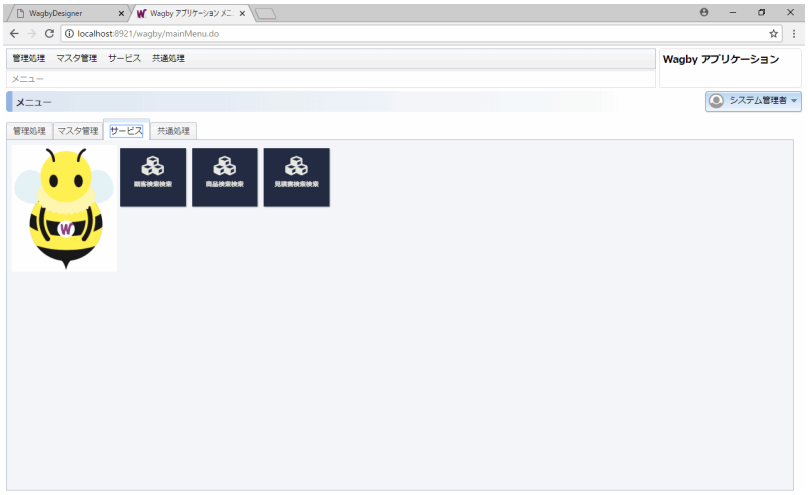


図 アイコンに変更した例

準備

customizeフォルダ内のwebappフォルダに「img」という名前のフォルダを用意します。そこに表示したい画像ファイルをコピーしてください。（例では menuicon1.jpg というファイルを用意しています。）

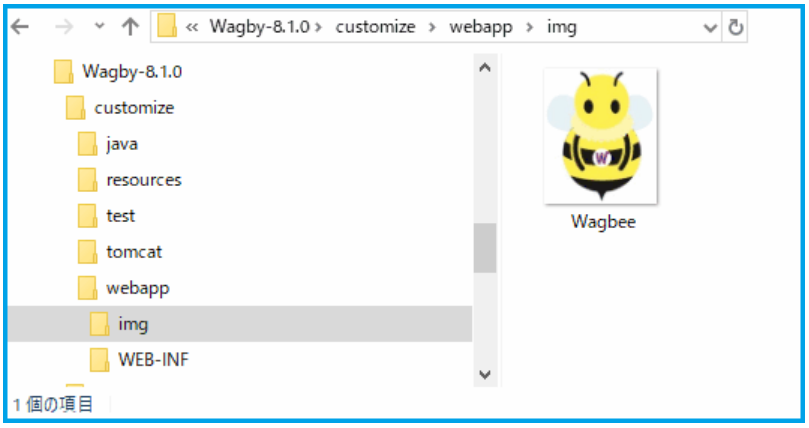


図 メニューアイコンを用意する

ビルド後の img フォルダにはシステムが標準で提供する画像も含まれます。ファイル名が重複しないように配慮してファイル名を決めてください。

定義方法

上記に用意したファイル名を指定します。img フォルダから始まるようにします。

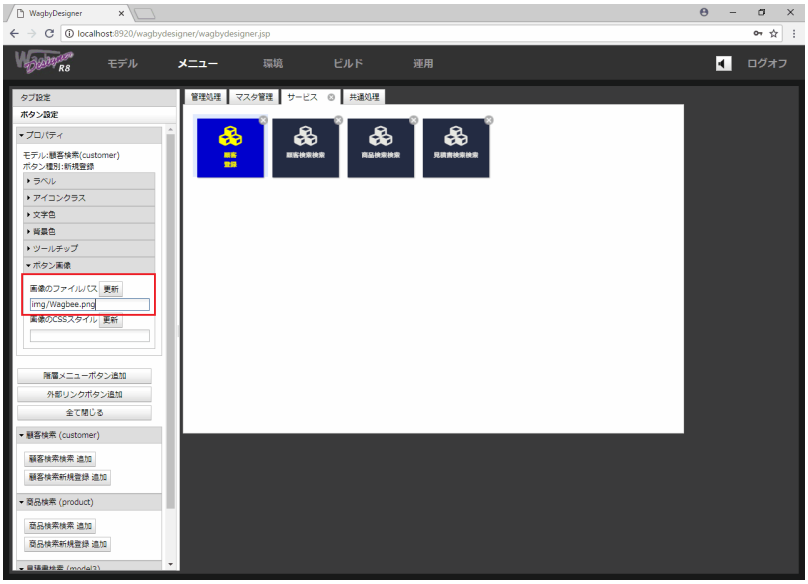


図 パスを指定する

Designer のメニュー設定画面ではアイコンは表示されません。ビルド後のアプリケーションで確認するようにしてください。

CSSを指定する

ここで指定した CSS は、メニューアイコンを表示する img 要素の style 属性に反映されます。例えば width を指定することで、アイコンの表示サイズを変更することができます。

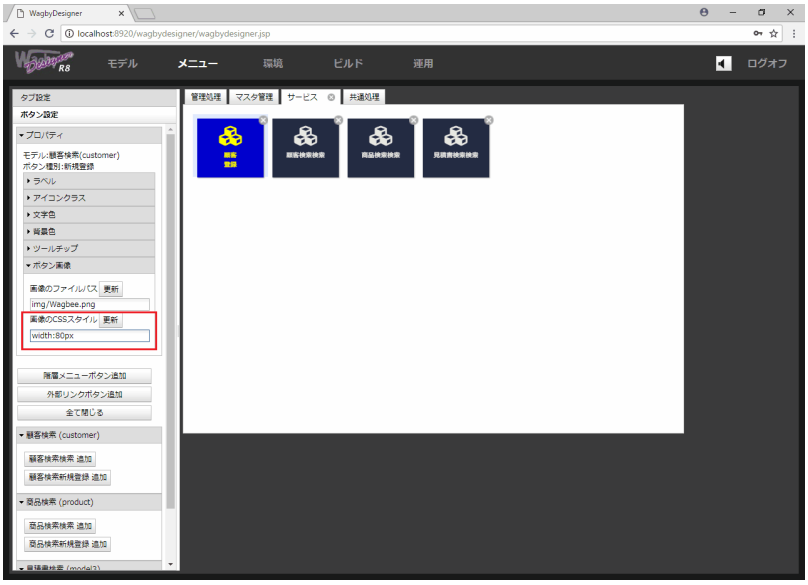


図 widthを80pxに指定する

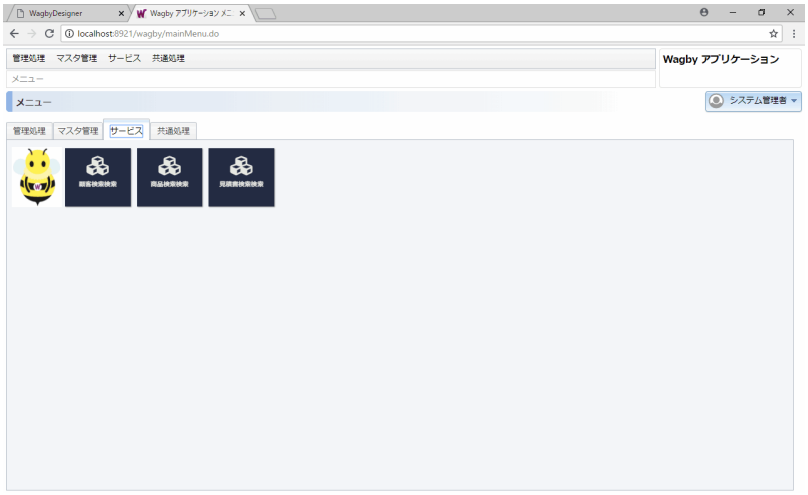


図 アイコンサイズが変更された

制約

・ 「戻る」ボタンに画像を設定することはできません。（「戻る」ボタンは文言を変更することもできません。）

プロパティ

「プロパティ」欄で、どのモデルについてのメニューかを確認することができます。

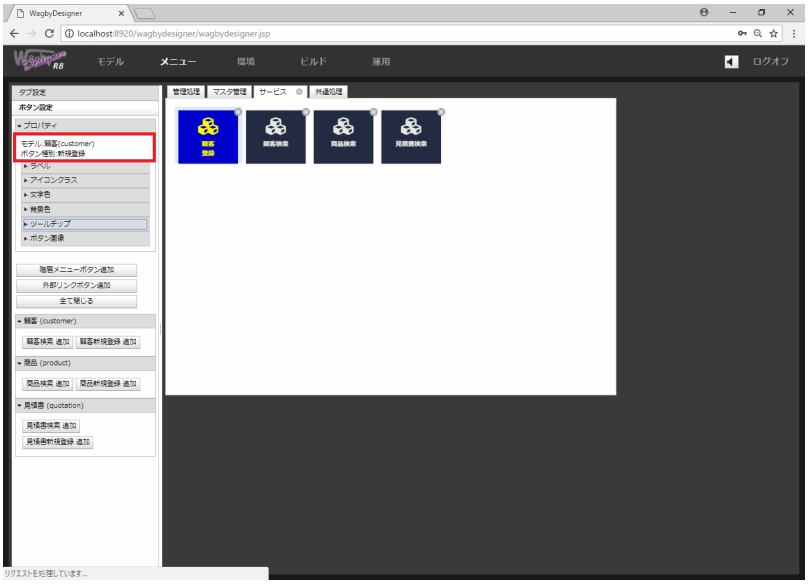


図 プロパティの確認

削除

メニューボタンの右上にある「×」アイコンをクリックするとメニューを削除することができます。図のように確認ダイアログが開きます。

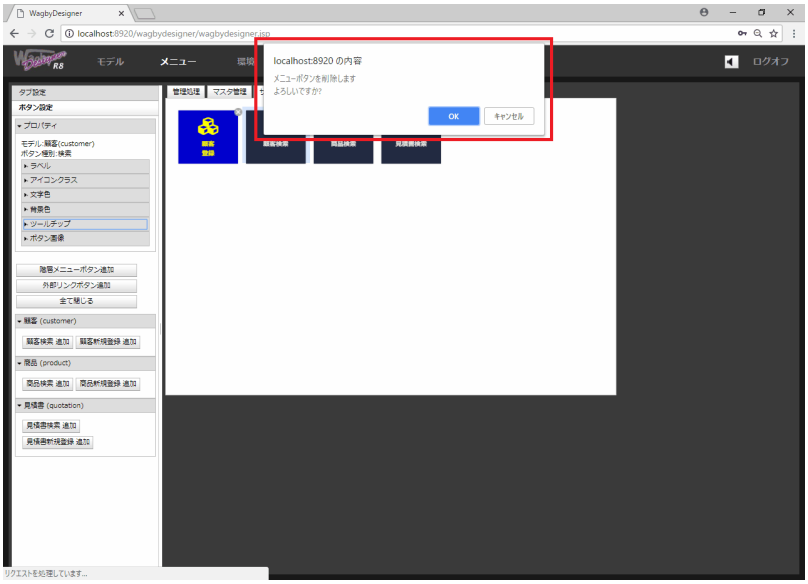


図 メニュー削除の確認

ダイアログで「OK」ボタンを押下すると、メニューが削除されます。

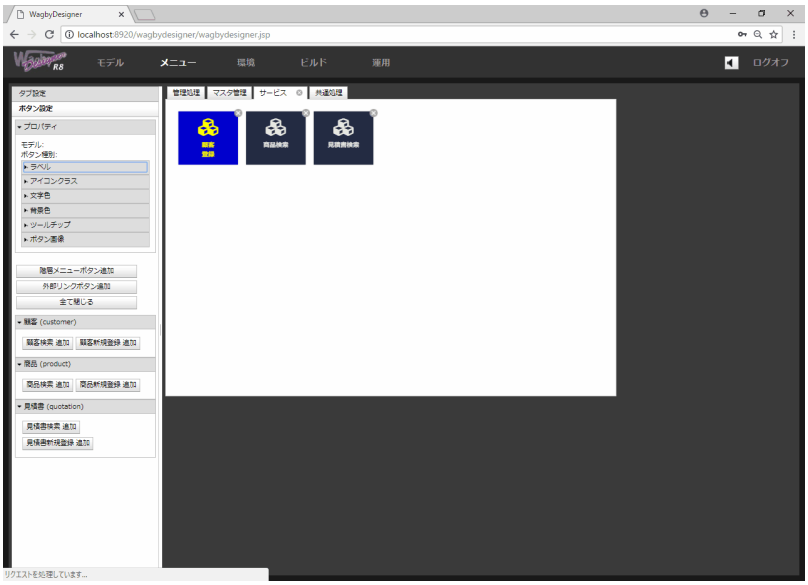


図 メニューが削除された

削除されたメニューを復活させる

メニューを再び追加することで、削除されたメニューを復活させることができます。

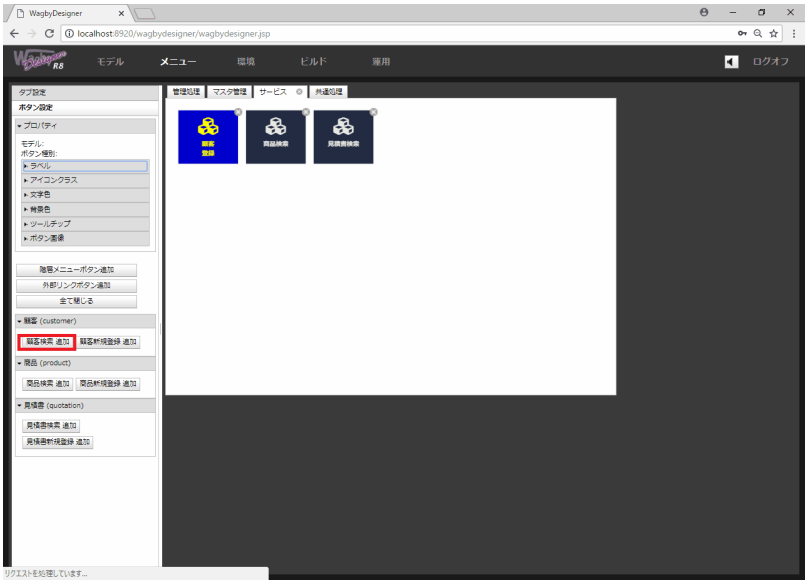


図 再び追加する(1)

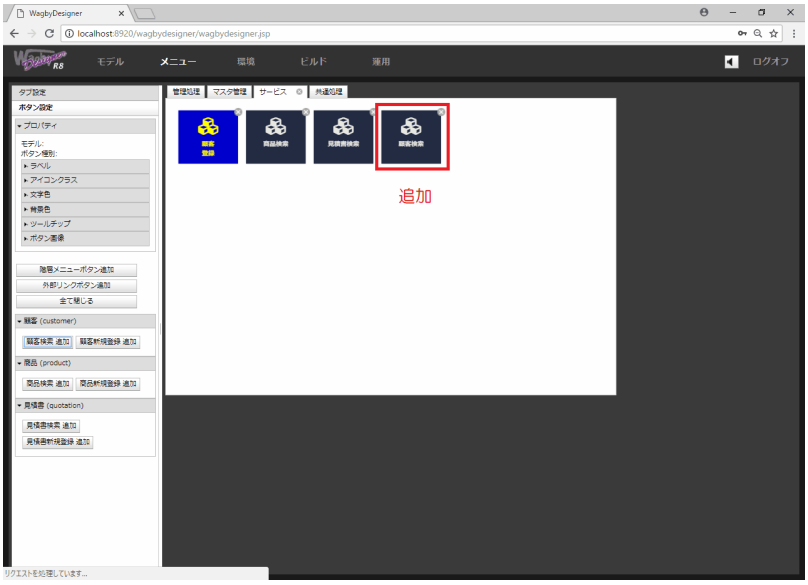


図 再び追加する(2)

例

実行イメージを図に示します。

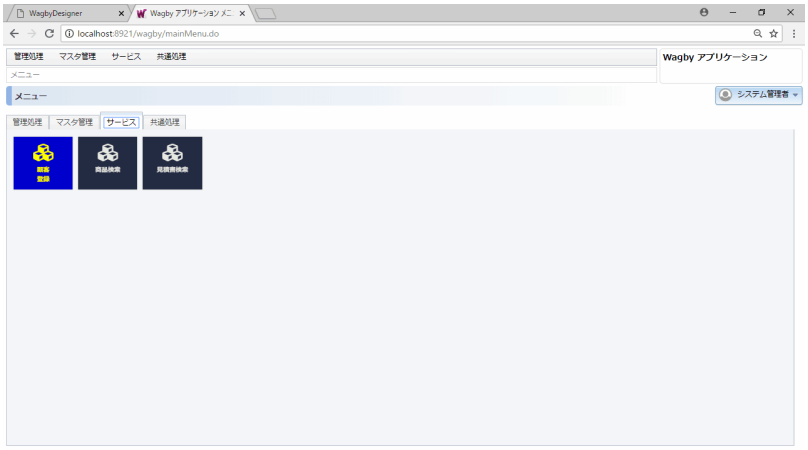


図 例

階層化

準備

以下の例では、「顧客」モデルのダウンロード機能とアップロード更新機能を有効にしています。

階層メニューの追加

階層メニューボタンを配置したいタブを選択しておきます。（ここでは〝サービス〟タブとします。） メニューの「ボタン設定」から「階層メニューボタン追加」を押下します。

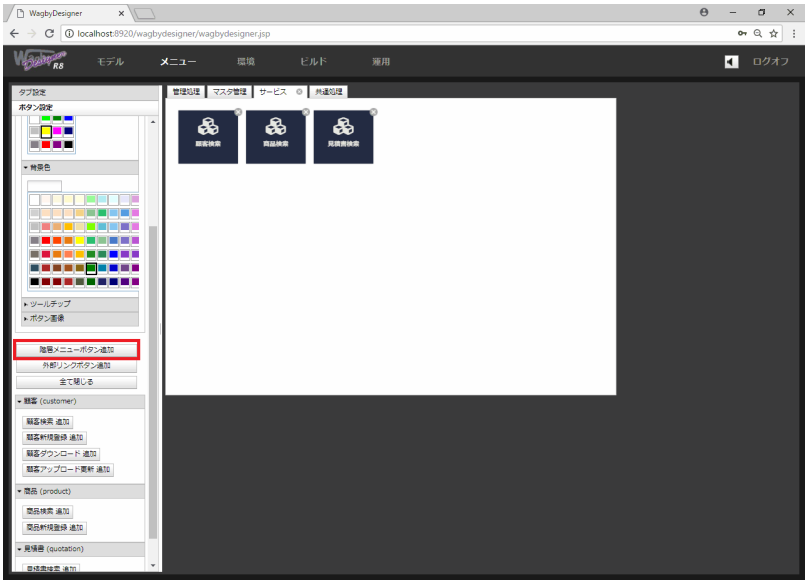


図 階層メニューボタン追加

現在、有効なタブ内に階層メニューのアイコンが用意されます。

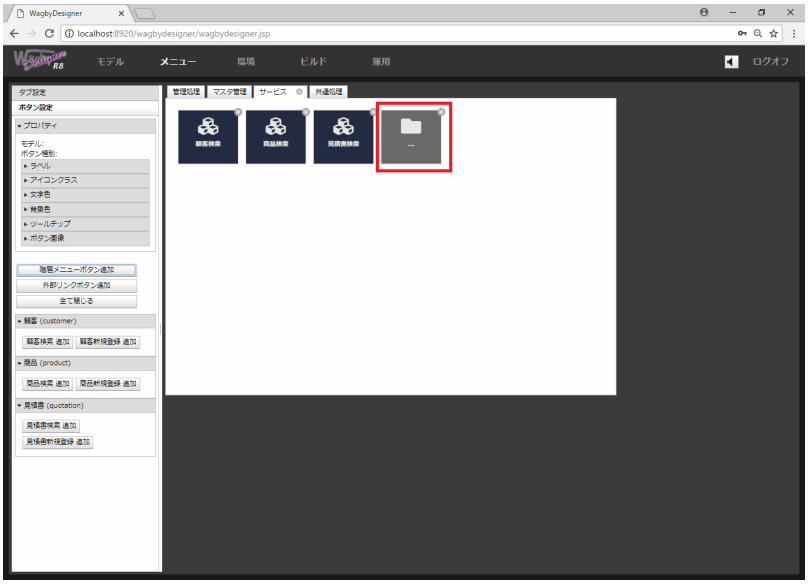


図 階層メニューが配置された

例として、ラベルを〝外部入出力〟とします。文字色と背景色を設定します。

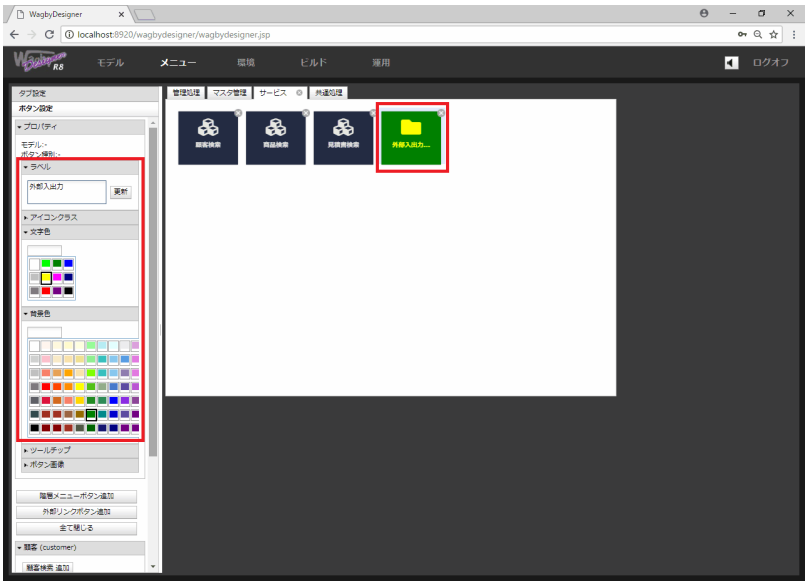


図 ラベル、文字色、背景色の設定

ドラッグ&ドロップで、階層メニューボタンの位置を変更します。

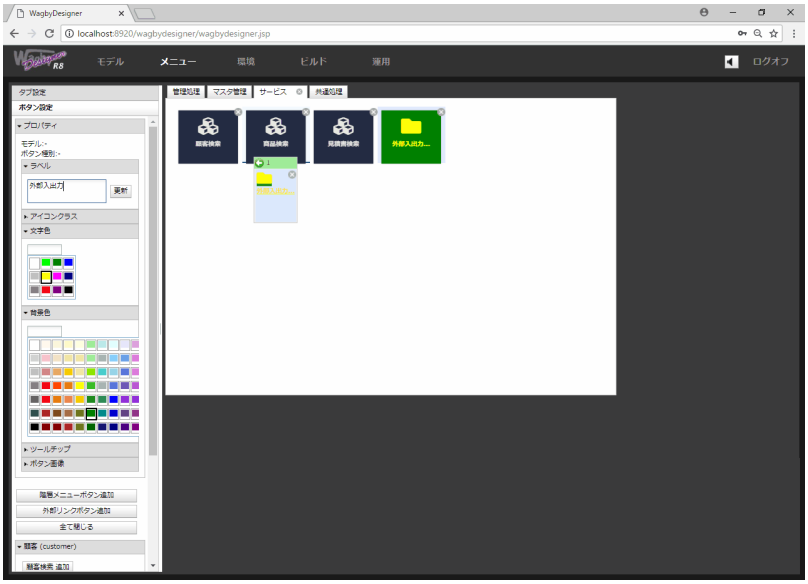


図 階層メニューボタンのドラッグ

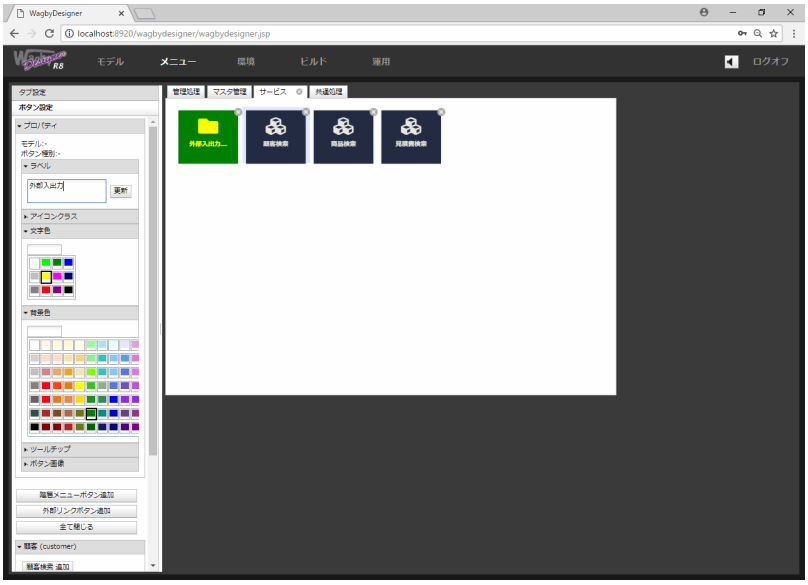


図 階層メニューボタンのドロップ（位置の移動）

追加した〝外部入出力〟メニューのアイコンをダブルクリックします。すると階層の中に移動します。

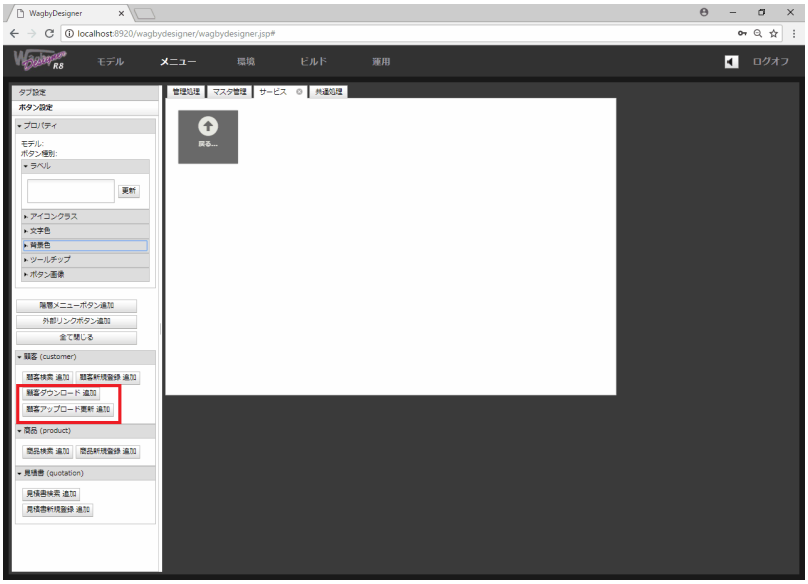


図 階層メニューの中に入る

「顧客ダウンロード」と「顧客アップロード更新」ボタンを追加します。

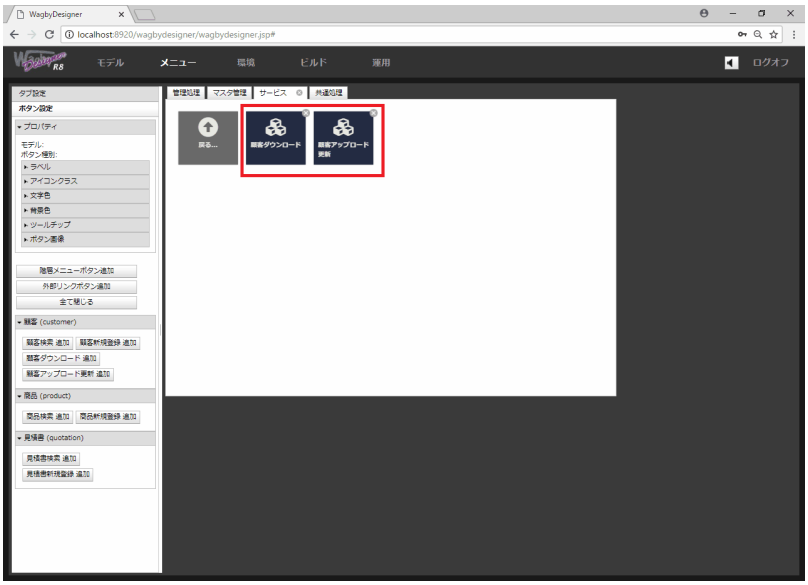


図 ダウンロードとアップロード更新のメニューボタンを追加する

″戻る″ボタンをダブルクリックすると、図の画面に戻ることができます。

″戻る″ボタンのラベルを変更することはできません。文字色と背景色は、階層化メニューボタンと同じものとなります。（変更することはできません。）

階層メニュー間でのメニューの移動

メニューボタンをドラッグしたまま階層メニューへマウスオーバーしたままにすると、この階層メニューが選択されます。

その後、階層メニュー内にメニューボタンを移動するとアイコンが緑に変わり、ドロップ可能となります。このままメニューボタンをドロップすることでメニューをその階層に移動させることができます。

表示方法の切り替え

階層化されたメニューをツリービュー形式で表示することができます。

仕様・制約

- 階層メニューボタンは入れ子にすることができます。
- 階層メニュー設定時、「戻る」ボタンの色および位置を設定することはできません。
- 階層メニューおよび、(階層下からの)戻るボタン押下は、パンくずに記録されません。

タブの管理

ラベルの変更

ラベルを変更したいタブを選択し、左側の設定部で「タブ設定」を選択します。

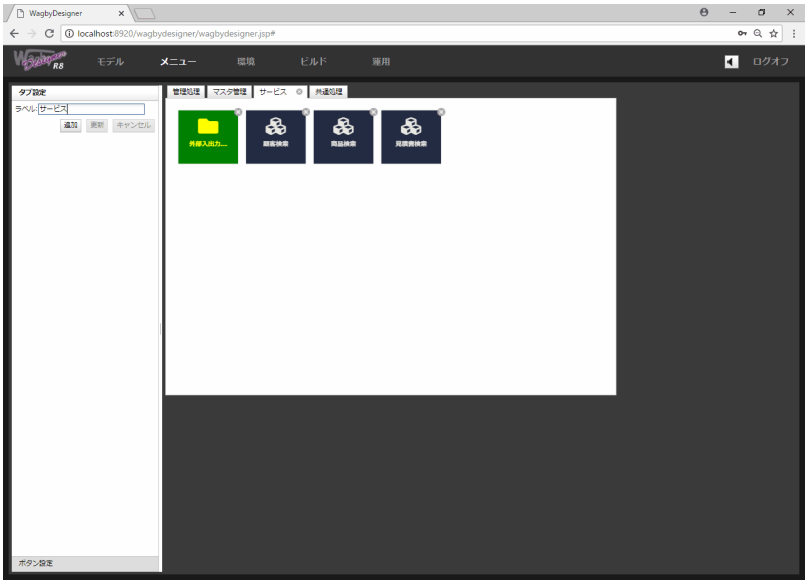


図 タブ設定

新しいラベルを入力し、「更新」ボタンを押下します。

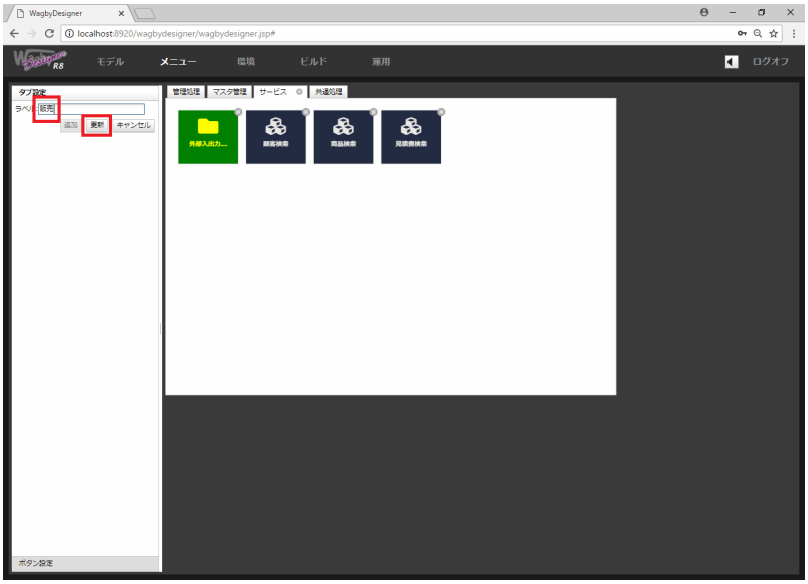


図 ラベルの入力

ラベルが変更されます。

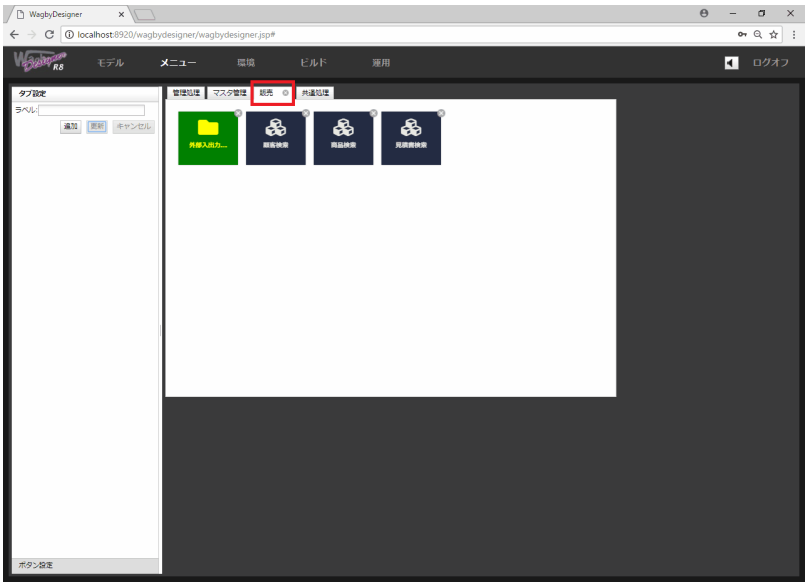


図 ラベルが変更された

タブの移動

ドラッグ&ドロップで、タブの位置を変更することができます。

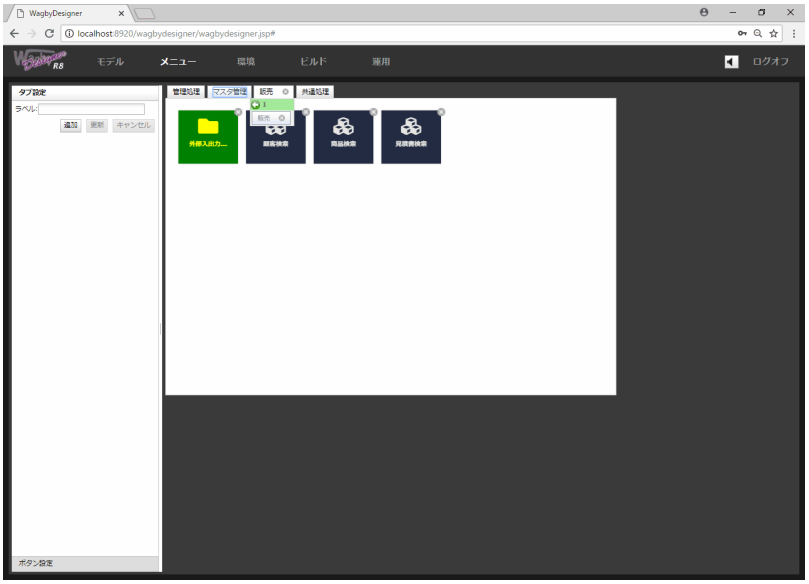


図 タブのドラッグ

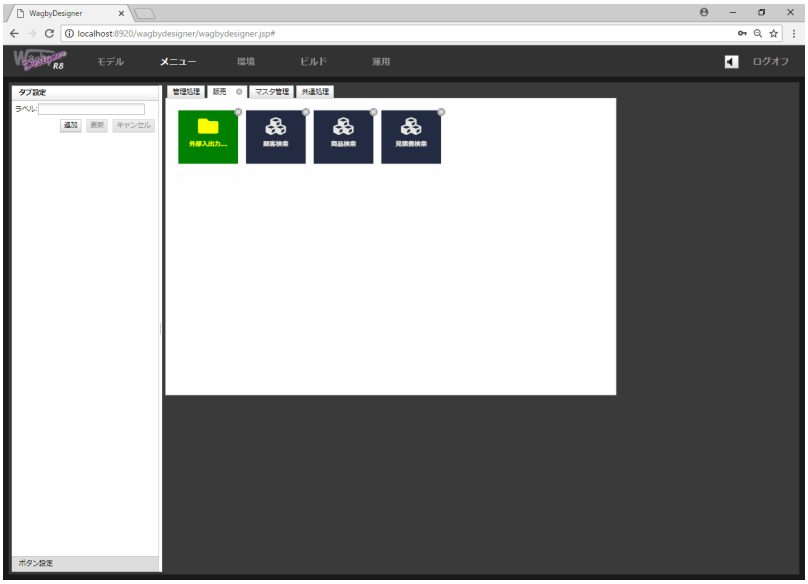


図 タブのドロップ（位置の移動）

タブの追加

任意のタブを選択後、いったん「キャンセル」ボタンを押下します。

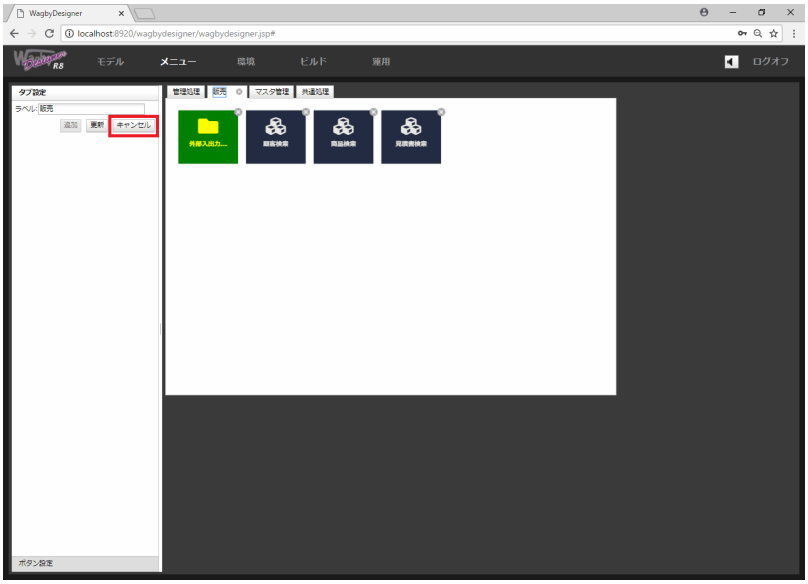


図 キャンセルボタンを押下する

すると「追加」ボタンが押せるようになります。

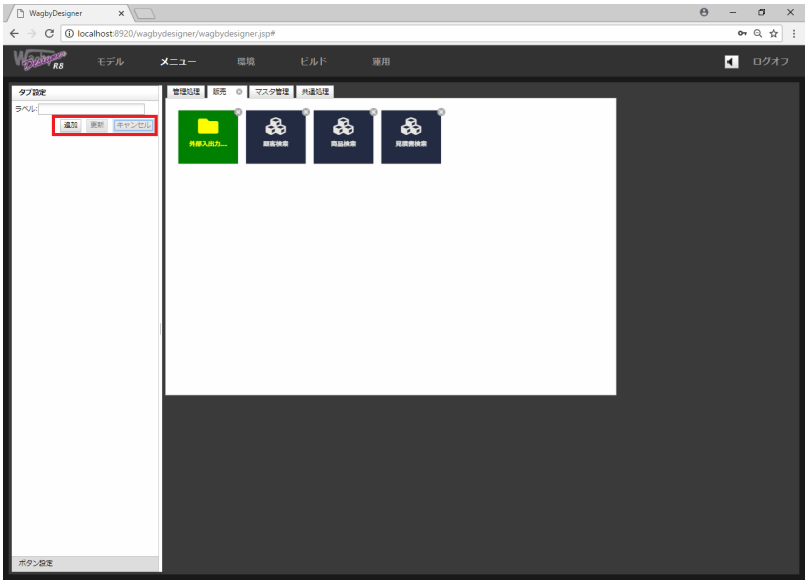


図 追加ボタンが有効になる

新しく追加するタブの名称を入力し、「追加」ボタンを押下します。

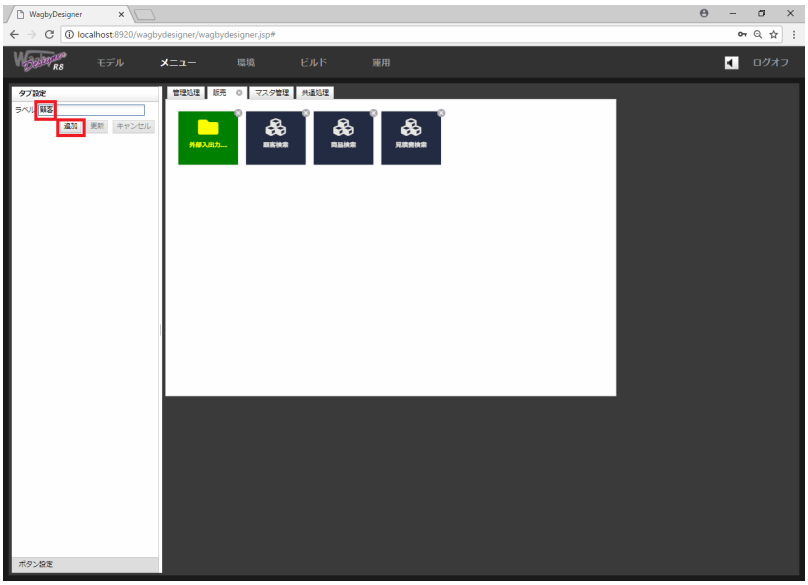


図 新規タブのラベルを入力する

新しいタブが用意されます。

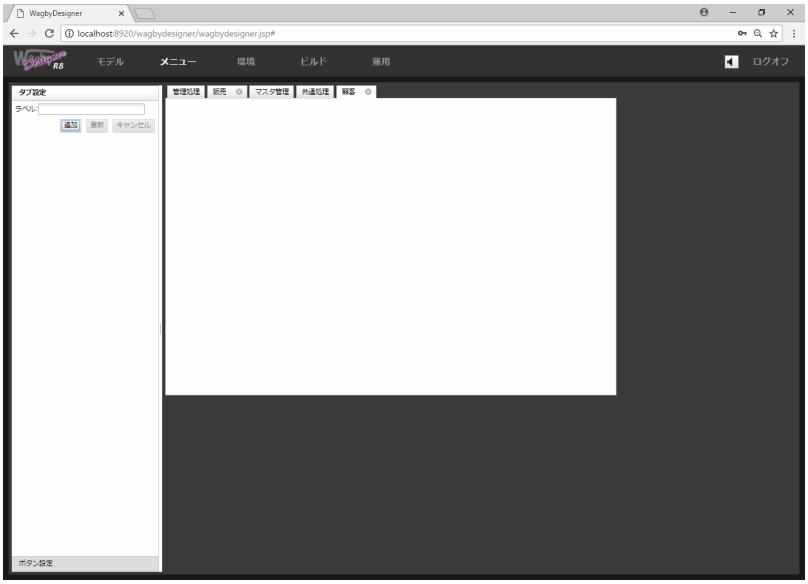


図 新しいタブが用意された

タブ内にメニューアイコンを追加します。

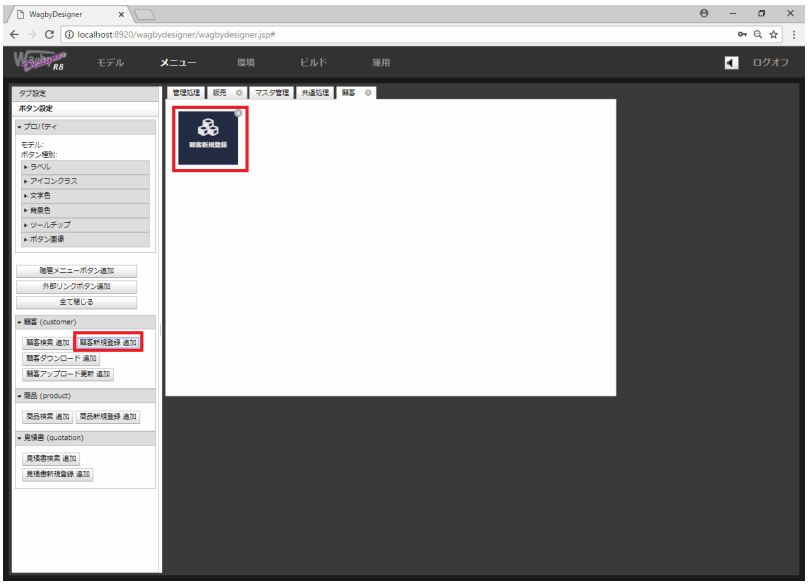


図 新しいタブにメニューアイコンを追加する

タブ追加時は、一つ以上のメニューアイコンを同時に追加してください。メニューアイコンが一つもない場合、タブは無効となります。（ビルド後のアプリケーションには反映されません）

タブ間のメニューの移動

メニューボタンをドラッグしたままメニュータブへマウスオーバーしたままにすると、このタブが選択されます。

その後、タブ内にメニューボタンを移動するとアイコンが緑に変わり、ドロップ可能となります。このままメニューボタンをドロップすることで別タブから移動させることができます。

階層メニューボタンの扱い

階層メニューボタンも上の方法でタブ間を移動させることができます。階層下のメニューとメニューボタンを再帰的にコピーして階層メニュー毎にタブ移動します。

仕様・制約

・　メニューのタブは、タブ内にモデルの検索画面等のメニューを追加することで更新があったと判断しています。タブだけを作成し、中身が空の状態ですと、タブの情報は保存されません。

外部サイトへのリンク

例

　メニューに「外部サイトへ」という大項目（タブ）を用意し、その中に任意の外部サイトへのリンクを用意した例を示します。

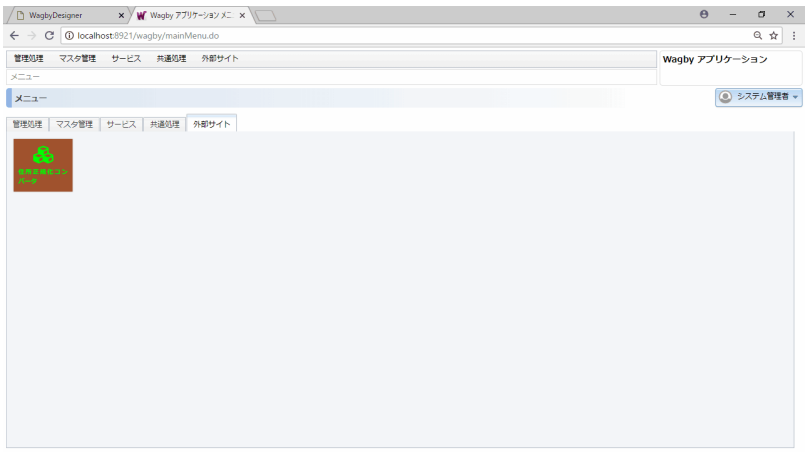


図 外部リンクをメニューに用意する

　メニューをクリックすると、指定したURLへ画面遷移します。

図 指定したURLのページが表示される

定義方法

タブの作成

　ここでは、はじめに「外部リンク」というタブを用意してみます。（なお、タブの作成は必須ではありません。外部リンクは任意のタブに配置することができます。）

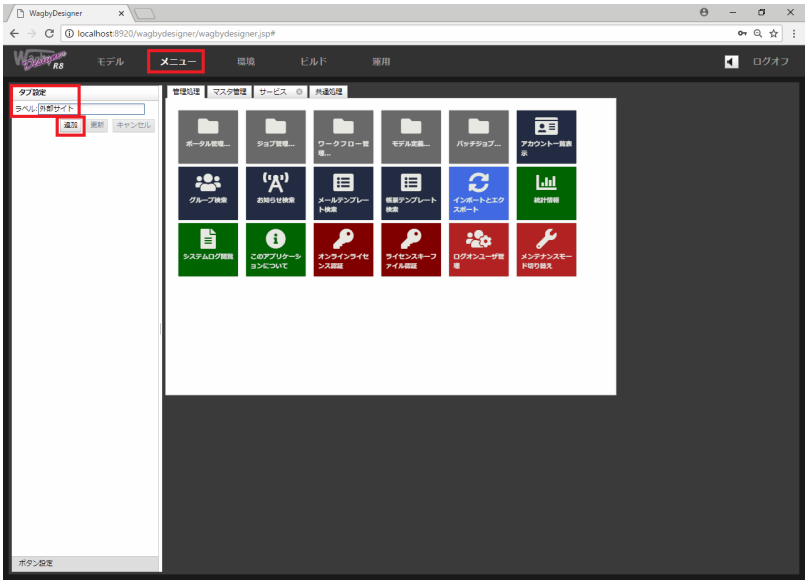


図 タブを作成する

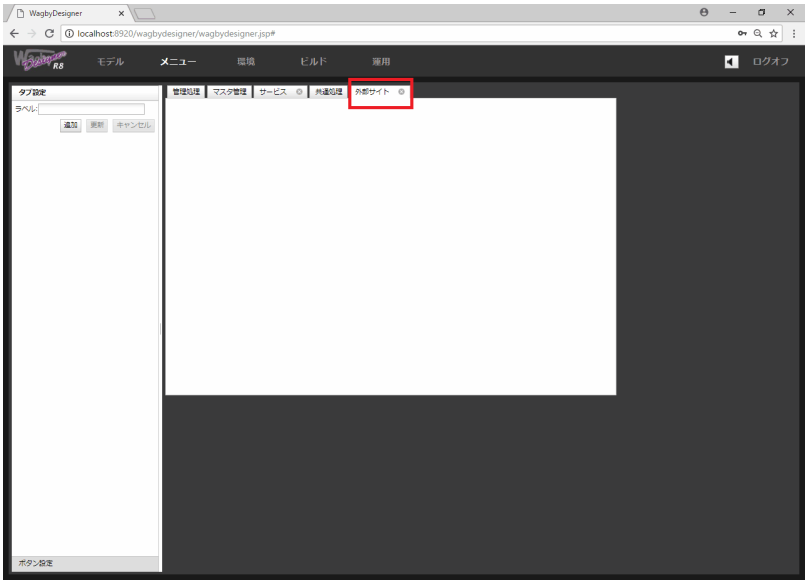


図 タブが作成された

外部リンクの作成

「ボタン設定 > 外部リンクボタン追加」を押下します。

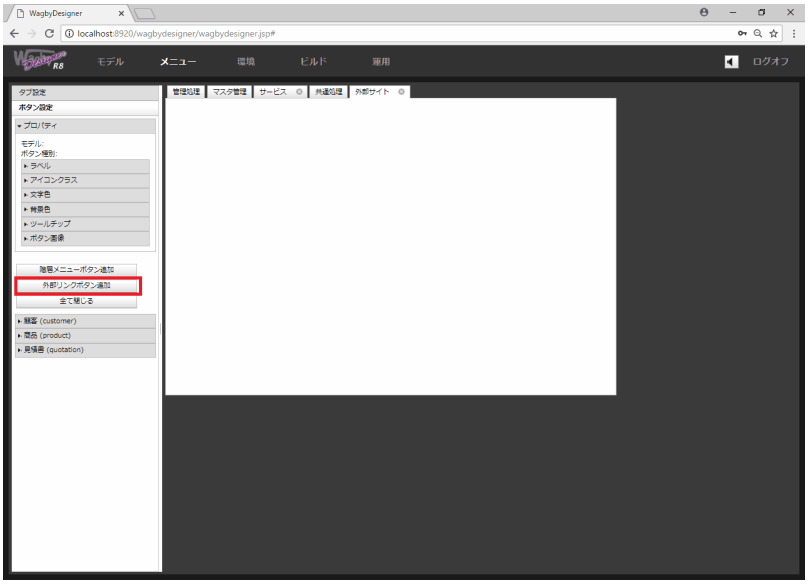


図 外部リンクボタン追加

外部リンクを設定するダイアログが用意されます。

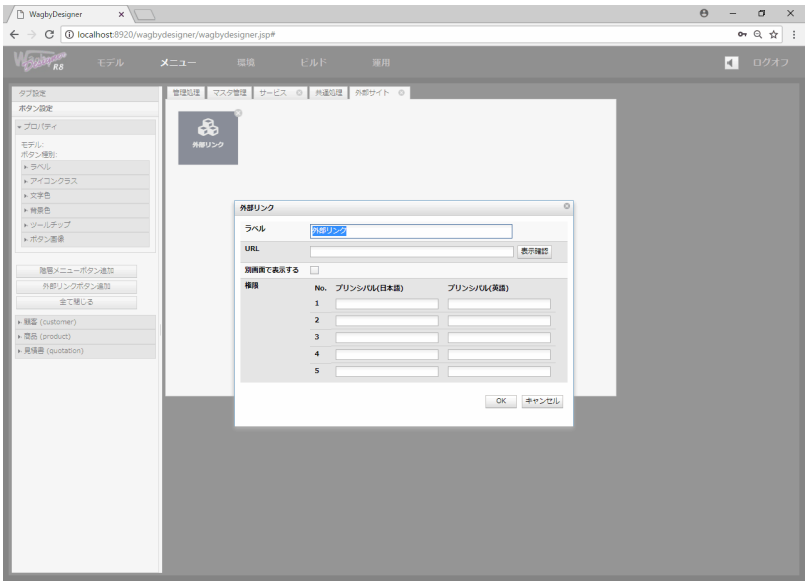


図 外部リンク設定ダイアログ

ラベルとURLを設定します。「別画面で表示する」を有効にすることもできます。多くのブラウザで、別タブが用意されます。

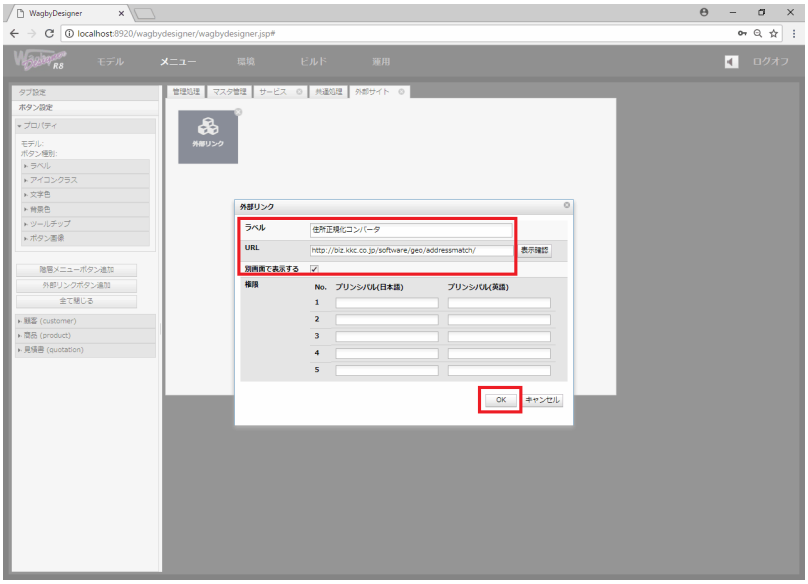


図 外部リンクの設定(1)

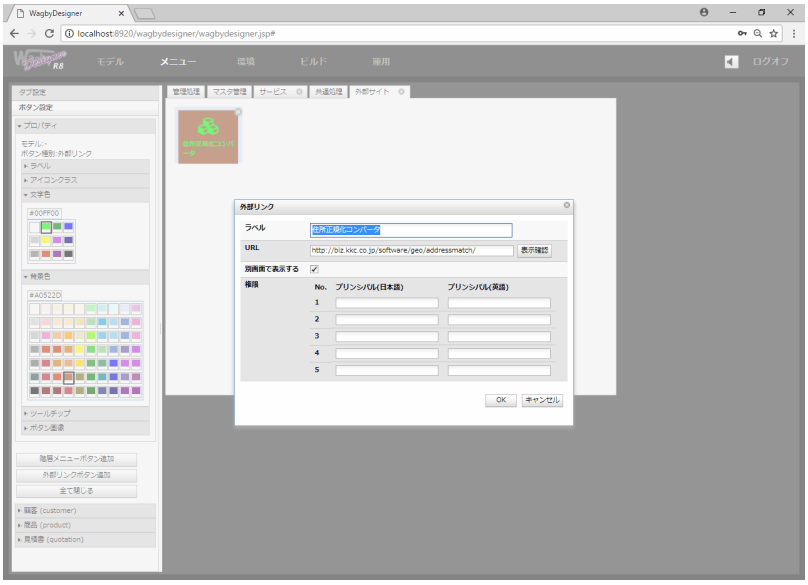


図 外部リンクの設定(2)

他のメニューと同様のボタンが用意されます。

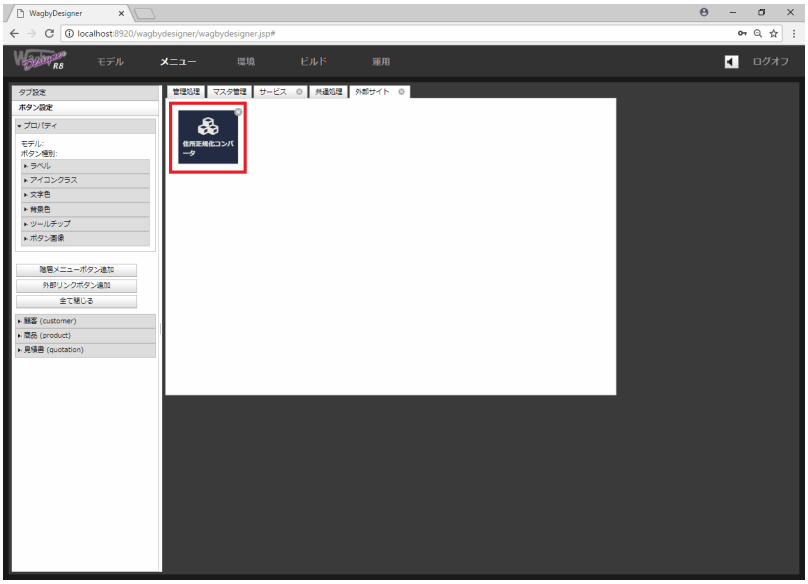


図 外部リンクのためのボタン

文字の色やボタンの背景色を変更することもできます。

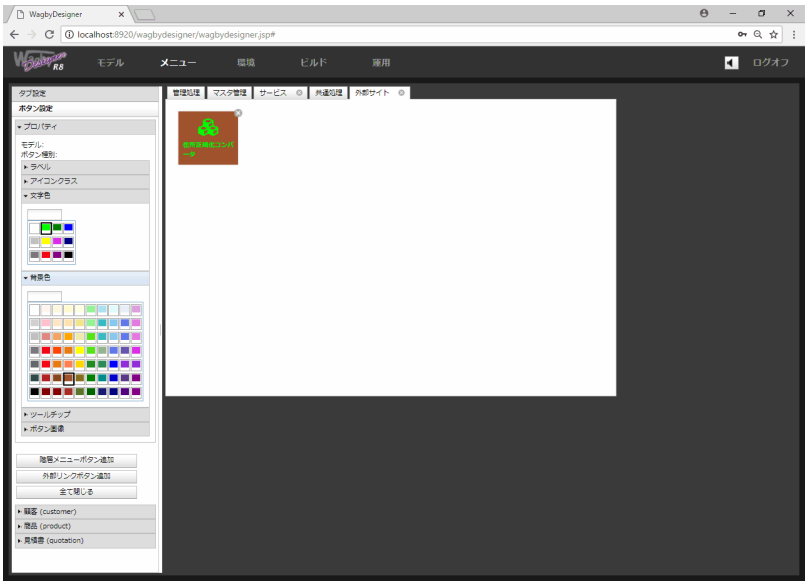


図 文字色やボタン背景色の設定

設定内容を変更する

作成したメニューボタンをダブルクリックすると、再び設定ダイアログが開きます。

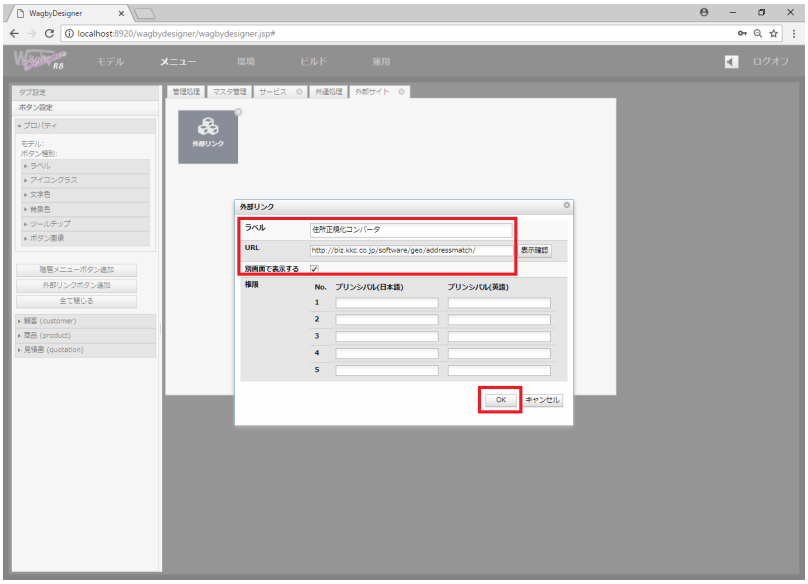


図 外部リンクの設定ダイアログを開く

Wagbyのユーザアカウントをリンクに埋め込む

URL 部に「${username}」と記述することができます。実際のメニューリンクでは、当該部分がログオンIDに置換されます。

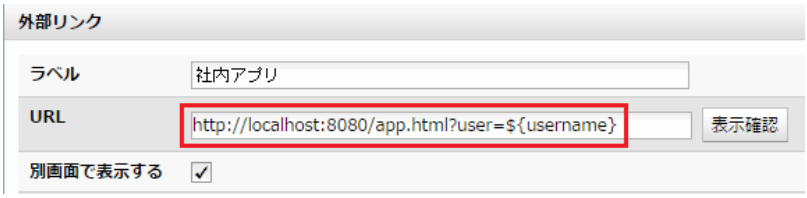


図 URLの設定

```
http://ANOTHER_WEBSITE_URL/index.html?user=${username}
```

受け側の Web アプリケーションでは、Wagby のアカウントを受け取って処理するようにしてください。

権限によってリンクの表示を制御する

各外部リンクに、権限（プリンシパル）を指定することができます。この権限をもっているアカウントのみ、メニューに表示されるようになります。

定義方法

外部リンク設定ダイアログの「権限」欄に、このリンクを有効とするためのプリンシパル名を指定します。

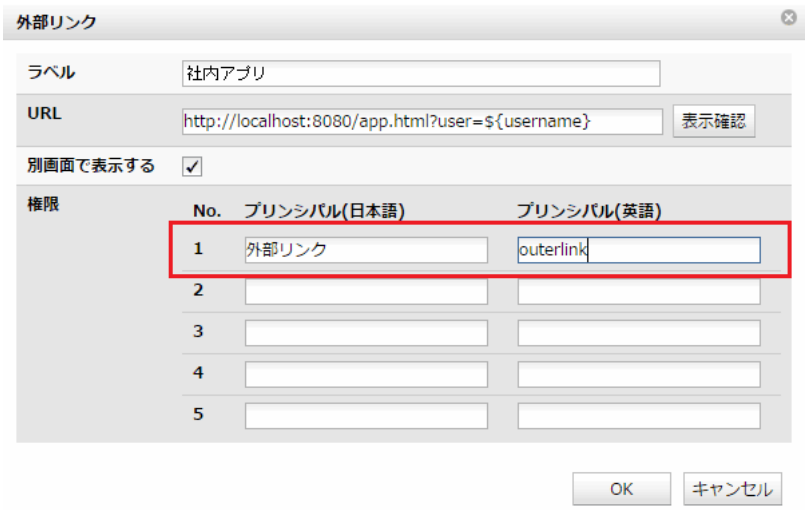


図 プリンシパルの設定

実行

ビルド後のアプリケーションでは、プリンシパルが追加されています。

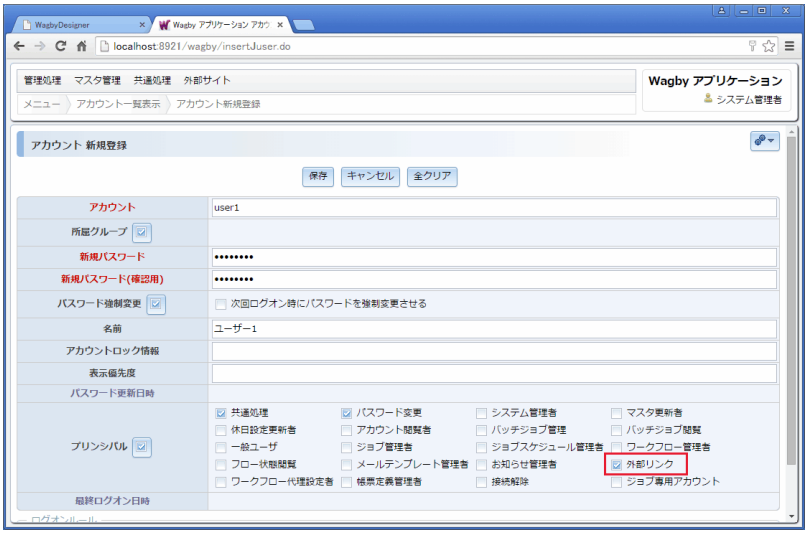


図 アカウントへのプリンシパル追加

外部サイトではなくWagby内部の任意の画面を開く

外部サイトではなく、Wagbyアプリケーション内の特定のモデルの画面へ遷移するためのリンクを用意することもできます。URLを次のように指定します。

```
/<プロジェクト識別子>/アクション名
```

プロジェクト識別子は標準では "wagby" です。プロジェクト識別子を空白とした場合は "/アクション名" のみとなります。

アクション名はモデルと画面の組み合わせから構成される文字列です。アドレスバー(URL)の一部になります。

例:モデル model1 の検索画面へ遷移する

次のような URL を指定します。

```
/wagby/showListModel1.do
```

URLに、遷移先画面へ渡すパラメータやイベント名を加えることもできます。[詳細...]

同じモデルのメニューボタンを異なるタブに重複して配置する

定義方法

例として「顧客検索」メニューを扱います。図にあるように、一度タブに配置すると、同じメニューは追加できません。

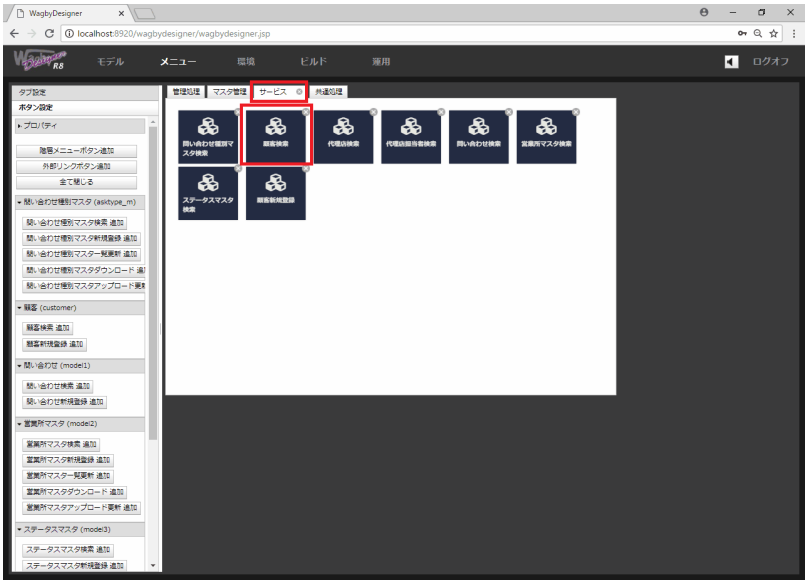


図 顧客検索メニューを用意した

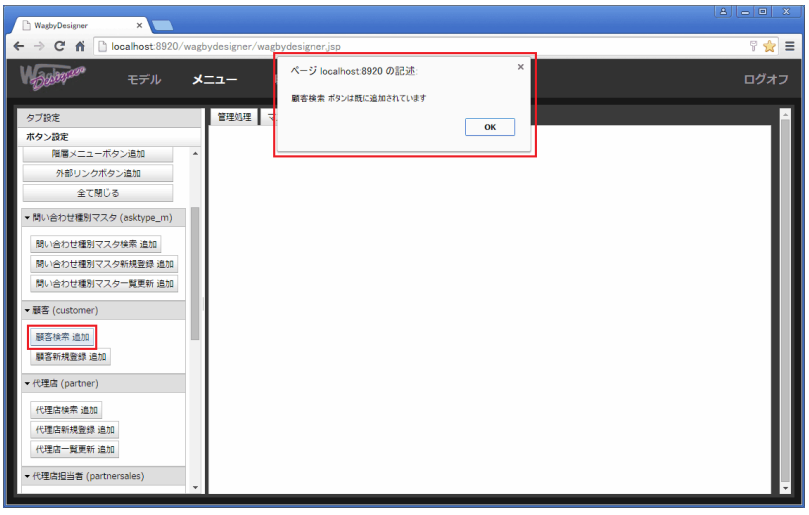


図 同じメニューは追加できない。

外部リンクボタン

外部リンクボタンを使って、同じ「顧客検索」メニューを別のタブに配置する方法を説明します。　最初に、新しいタブを作成します。

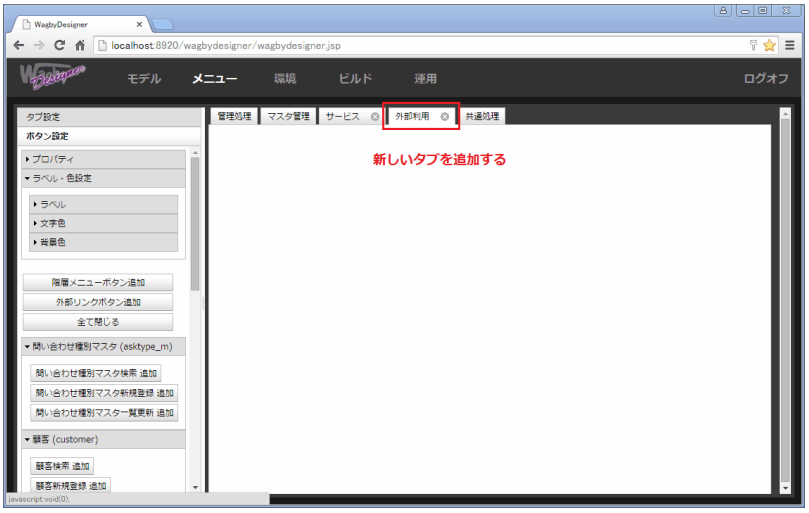


図 タブの新規作成

「外部リンク」を用意します。

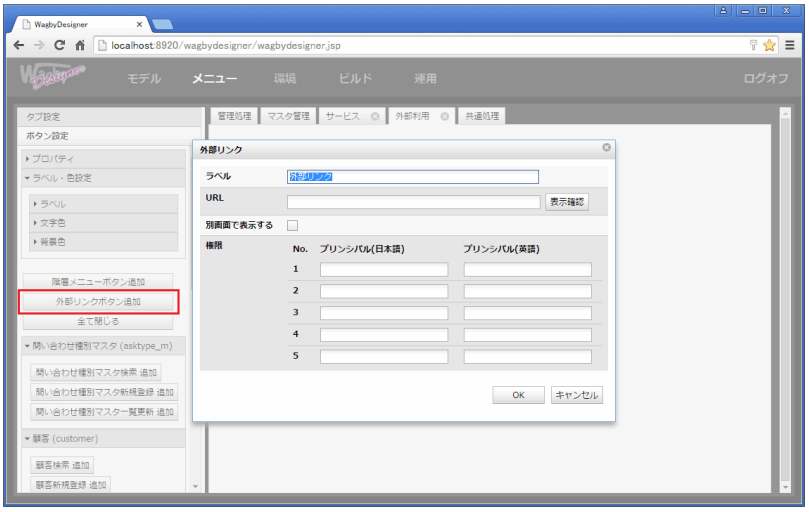


図 外部リンクを用意する（検索一覧画面）

ポイントは、URLの記述にあります。

"/" + プロジェクト識別子 + "/" + 遷移先の画面情報

例を示します。

画面	URL	例
検索・一覧	"showList" + モデル名(キャメル記法) + ".do"	showListCustomer.do
新規登録	"insert" + モデル名(キャメル記法) + ".do?action_New=New"	insertCustomer.do?action_New=New

モデル名はキャメル記法で表現します。先頭を大文字とします。customerはCustomerとなります。

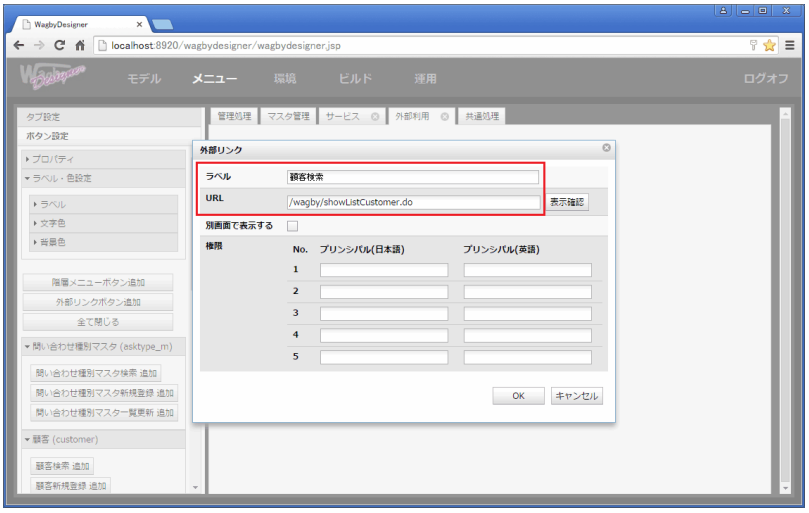


図 外部リンクを用意する（検索一覧画面）

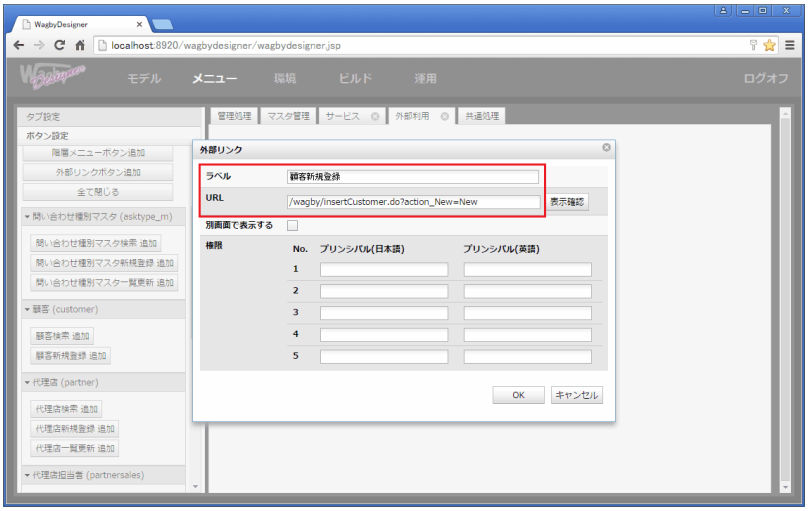


図 外部リンクを用意する（新規登録画面）

例

ビルドしたアプリケーションを確認します。 「サービス」タブと「外部利用」タブに同じメニューアイコンが用意されていることがわかります。

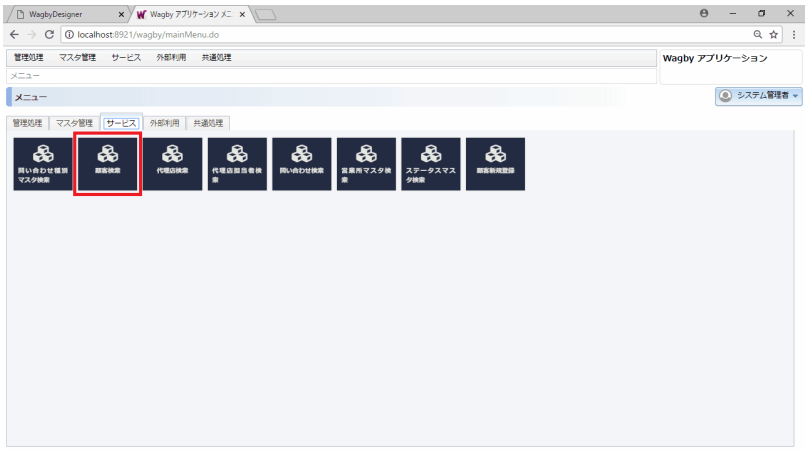


図 通常のメニューアイコン

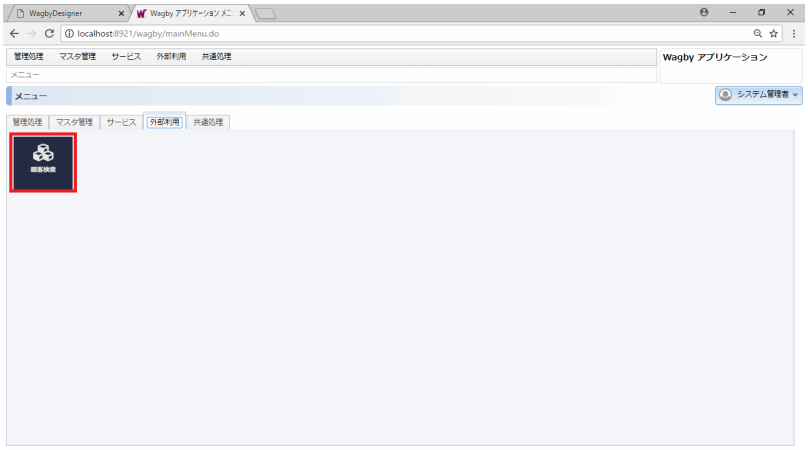


図 外部リンクで実現したメニューアイコン

スマートフォン

スマートフォンの利用

例

スマートフォンに付属の Web ブラウザを使います。Web アプリケーションにログオンすると、図のようなログオン画面が表示されます。（アプリケーションのタイトルは開発者が設定したものが表示されます。）

図 ログオン画面

スマートフォンのWebブラウザから入力する URL は、PC と同一になります。 （例 http://MACHINENAME:8921/wagby/） Wagby が接続機器を識別し、スマートフォンと判定すると自動的にスマートフォン用の画面を表示します。

モバイル版のログオン画面で "モバイル|PC" のリンクから PC 版ログオン画面を表示させた場合、一度ログオンしてログアウトするか、アドレスバーの URL 末尾のpcView=trueをpcView=falseと指定するとモバイル版の画面が表示されます。

アカウントとパスワードを入力し、ログオンボタンを押します。

図 ログオン認証を行う

スマートフォン向けのメニューが表示されます。PC版と異なり、タブごとの表示ではなく、縦に並びます。

図 メニュー

新規登録画面

画面右上の「+」ボタンを押下することで、新規登録を行うことができます。　項目の並び（レイアウト）は項目の定義順となります。（縦並びや横並びのレイアウト指定はありません。）

図 新規登録画面

文字列、数字、日付といった項目に値を入力することができます。

図 新規登録画面への入力例

検索画面（サブウィンドウ）

図で「年休申請ID」項目に用意された「検索...」ボタンを押下すると、画面下からサブウィンドウがせり上がって表示されます。

図 検索画面（サブウィンドウ）

詳細画面

登録後の詳細画面です。

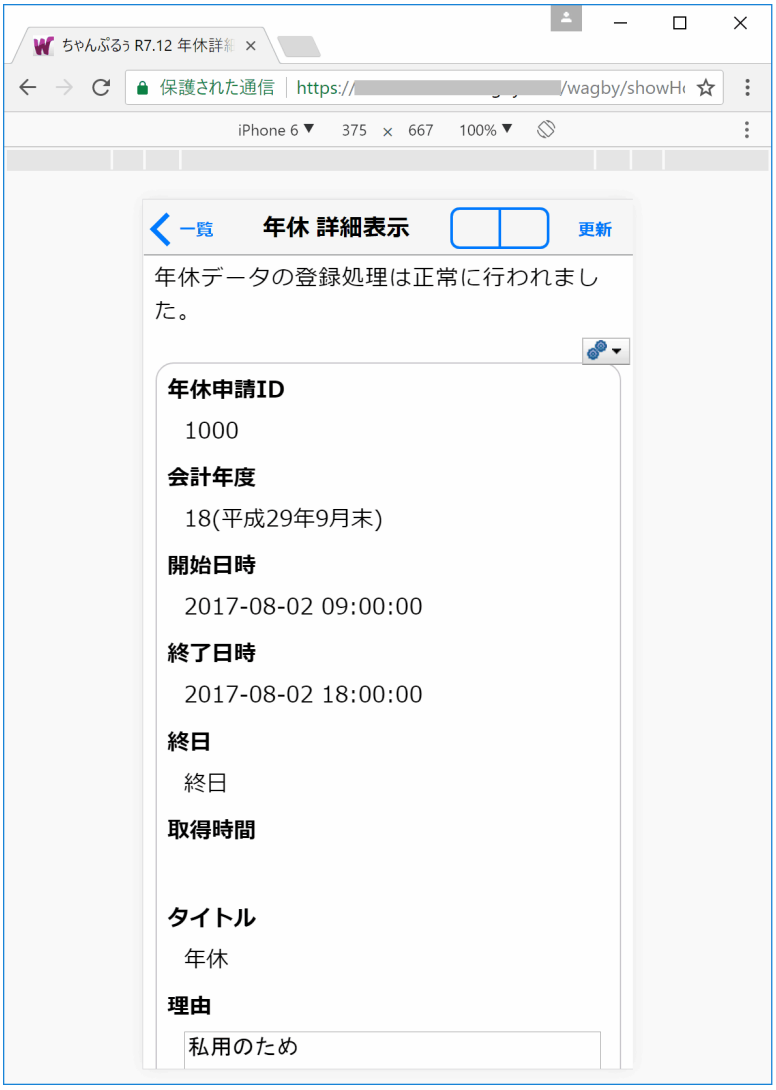


図 詳細画面

詳細画面の右上にボタン(メニュー、ログオフ)が配置されています。

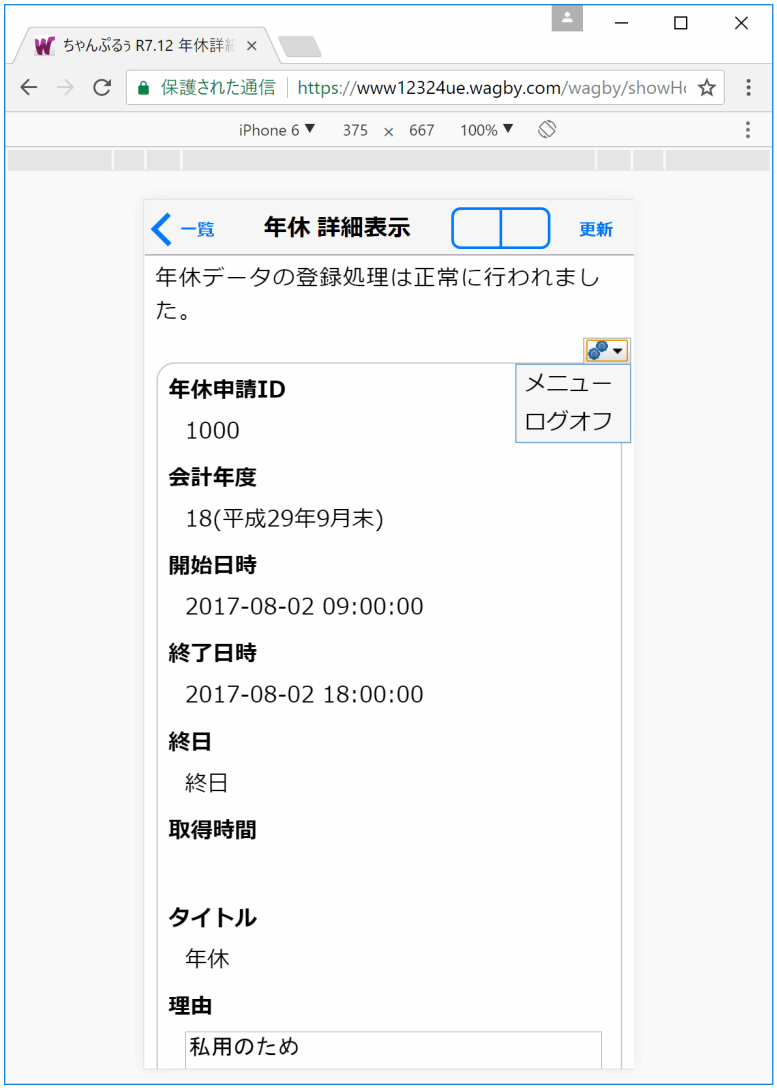


図 メニューとログオフへ遷移するボタン

検索・一覧表示画面

一覧表示画面の右上にある検索ボタンを押下すると、検索入力画面に切り替わります。

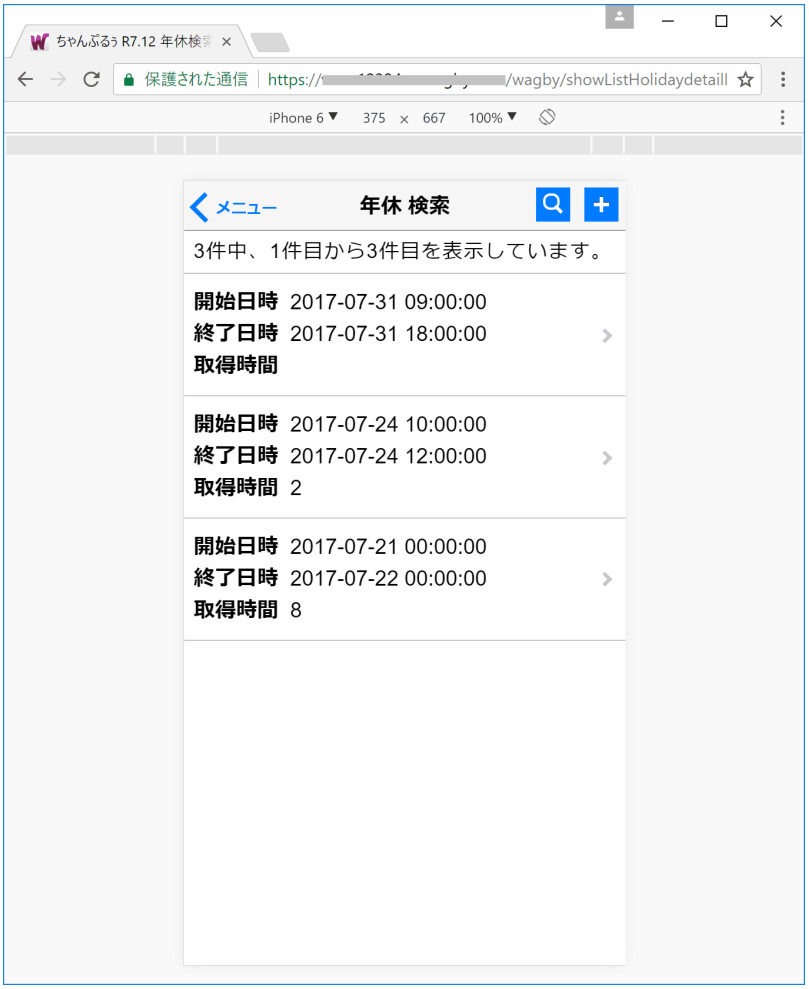

ちゃんぷるぅ R7.12 年休検索
保護された通信
https://
/wagby/showListHolidaydetailI
iPhone 6
375 × 667
100%
メニュー
年休 検索
3件中、1件目から3件目を表示しています。
開始日時 2017-07-31 09:00:00
終了日時 2017-07-31 18:00:00
取得時間
開始日時 2017-07-24 10:00:00
終了日時 2017-07-24 12:00:00
取得時間 2
開始日時 2017-07-21 00:00:00
終了日時 2017-07-22 00:00:00
取得時間 8

図 一覧表示画面

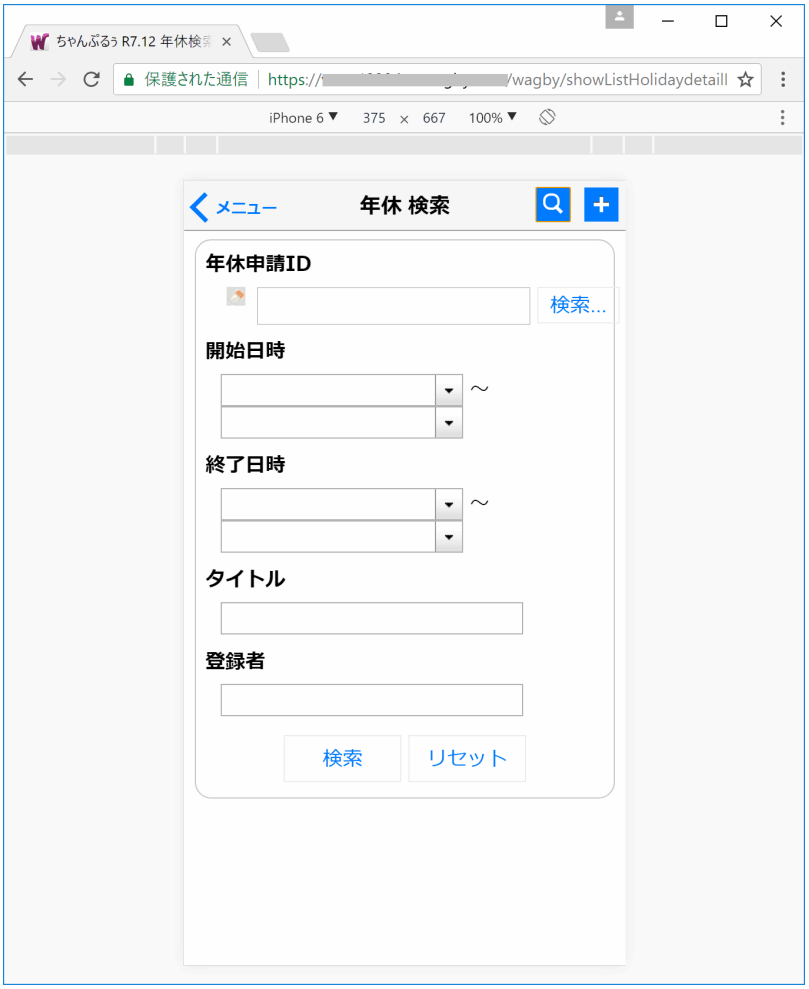


図 検索画面

日付型項目では、PC版と同じカレンダ補助入力インタフェースを利用することができます。

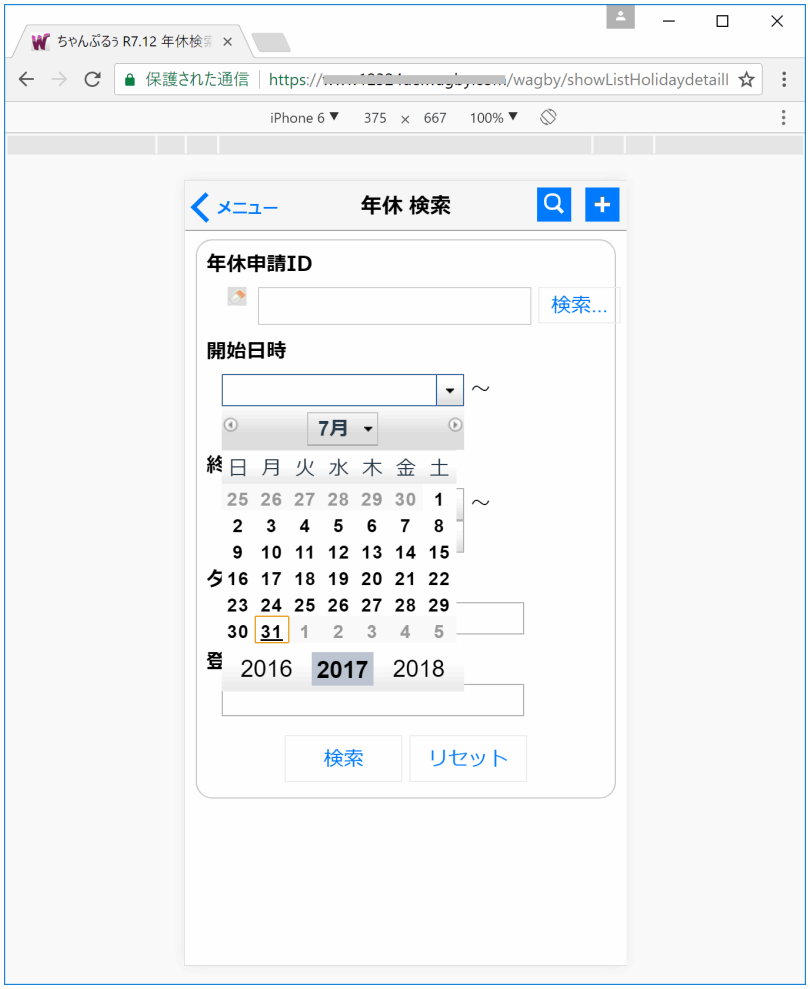


図 日付型項目へのカレンダ補助入力

前後のデータを閲覧する

検索結果により、前後のデータへ移動するボタンが有効になります。図は検索結果の先頭のデータを表示しています。「次」へ移動するボタンが有効になっています。

図 前後のデータへ移動する(1)

次のデータを表示すると、今度は「前」「次」の両方のボタンが押下できるようになっています。

図 前後のデータへ移動する(2)

繰り返しコンテナ

図、図は繰り返しコンテナを表示した例です。レイアウトは項目の定義順に横に並びます。

ちゃんぷるぅ R7.12 交通費精
保護された通信
https://
/wagby/showKoutsuhiMobile.c
iPhone 6
375 × 667
100%
一覧
交通費精算 詳細...
更新
1件中、1件目を表示しています。
精算ID
1000
名前
システム管理者
年月
2017年07月
ガソリン代
15
総合計
¥1,590
車移動の場合
ID
日付
行き先等
距離(km)
駐車料金
高速
その他移動の場合
ID
日付
行き先等
移動手段
料金

図 繰り返しコンテナ(1)

図 繰り返しコンテナ(2)

更新画面では、繰り返しコンテナの値の編集、追加、削除を行うことができます。繰り返しコンテナ部分は横スクロールすることで項目全体を確認できます。

繰り返しコンテナのコンテナ名をタップすると画面スクロールがロックされます。この操作はトグル式になっています。スクロールを有効にするには再度コンテナ名をタップしてください。

図 繰り返しコンテナの編集(1)

図 繰り返しコンテナの編集(2)

外部キー子モデル

詳細画面の下部に、外部キー子モデルを表示した例です。

図 外部キー子モデル

更新画面では、子モデルの同時更新を行うことができます。子モデル部分は横スクロールすることで項目全体を確認できます。

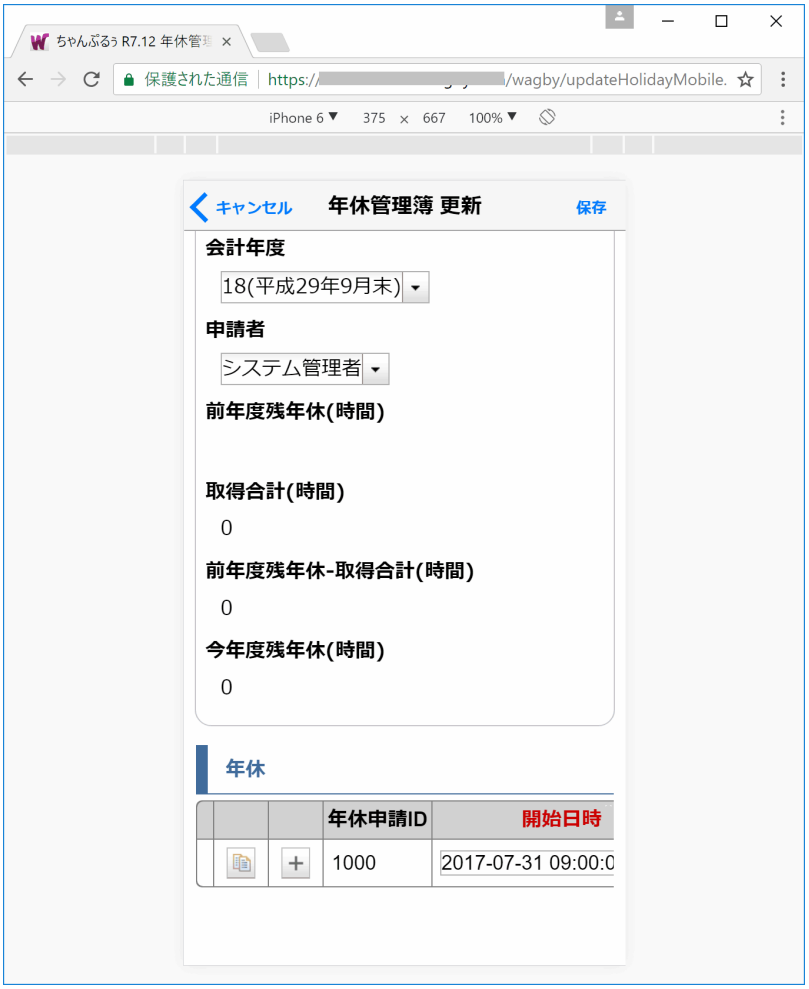


図 外部キー子モデルの更新(1)

図 外部キー子モデルの更新(2)

モダンカレンダビュー

モダンカレンダビューを表示させることができます。PC版と同様の操作方法になります。

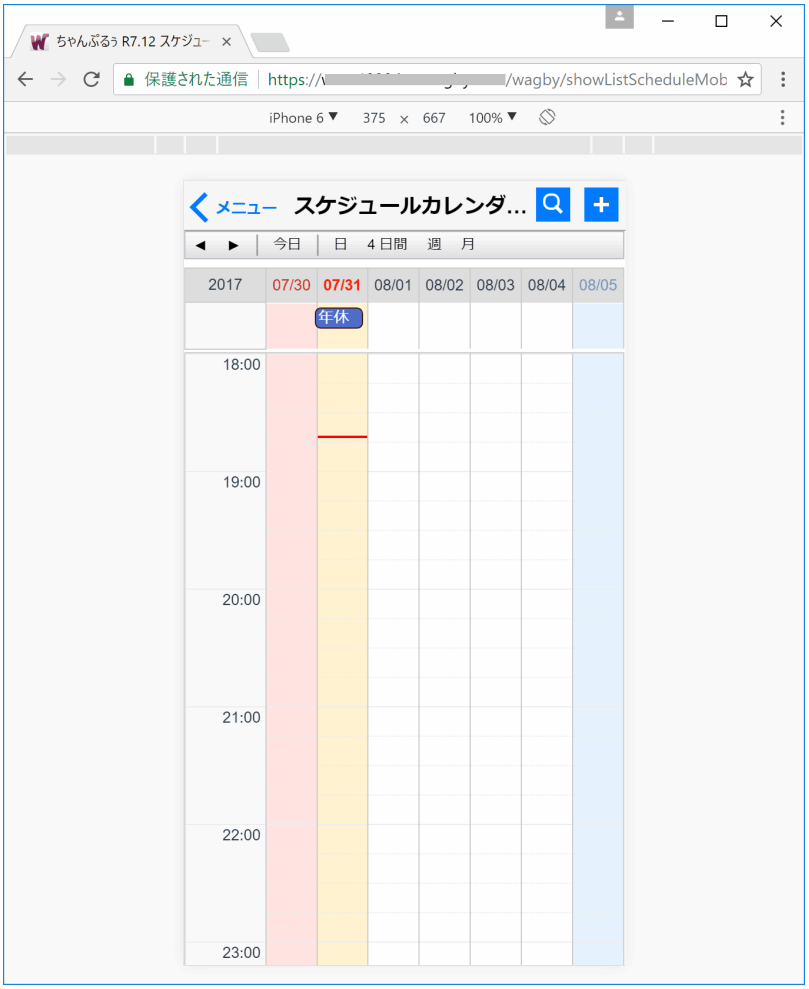


図 カレンダビュー(1)

画面右上の「+」ボタンを押下することで、新規登録を行うことができます。画面下から、基本項目の入力ウィンドウがせり上がって表示されます。

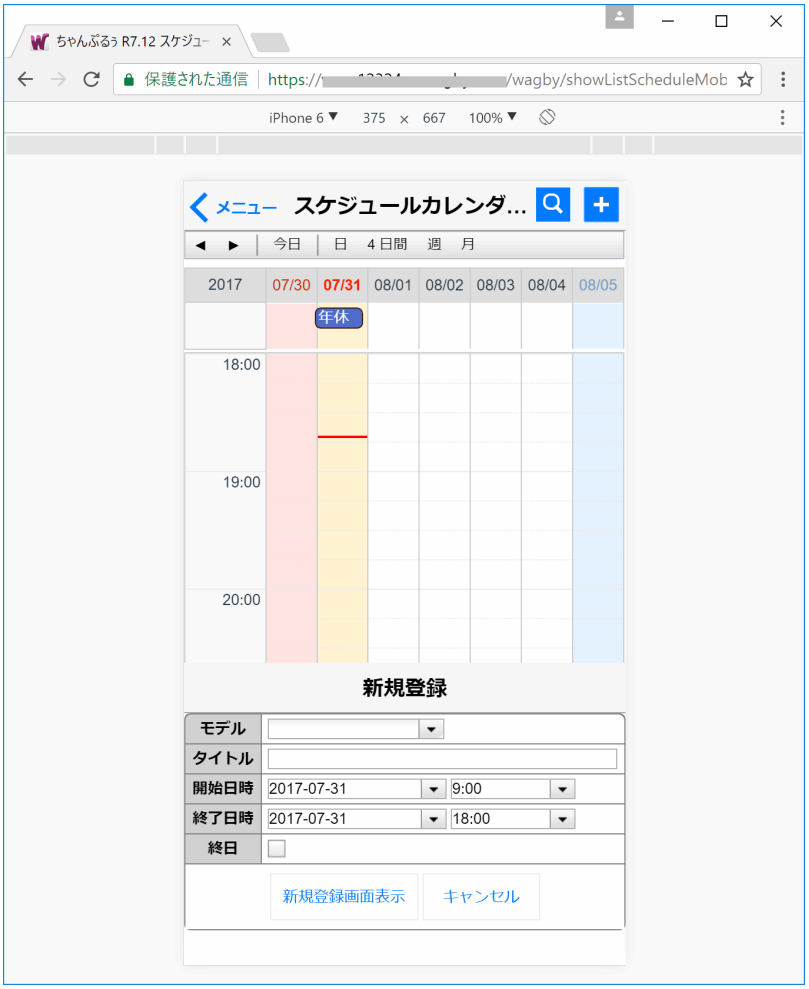


図 カレンダビューモデルへの新規登録(1)

値を入力し、「新規登録画面表示」ボタンを押下します。

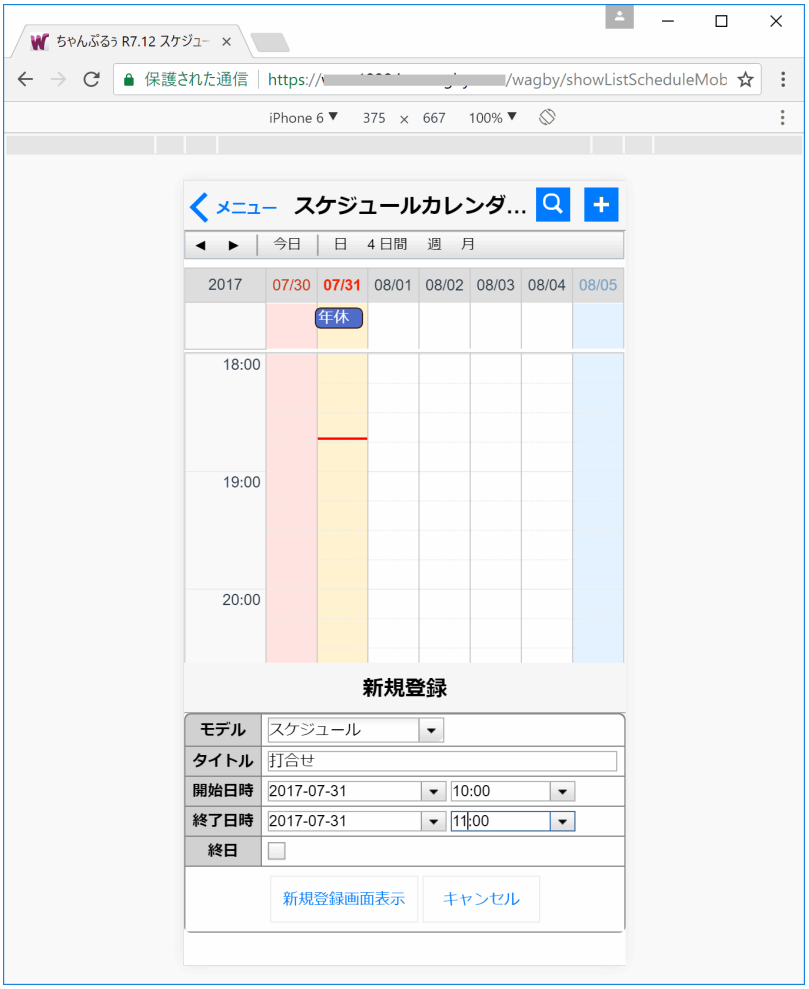


図 カレンダビューモデルへの新規登録(2)

新規登録画面が開きます。

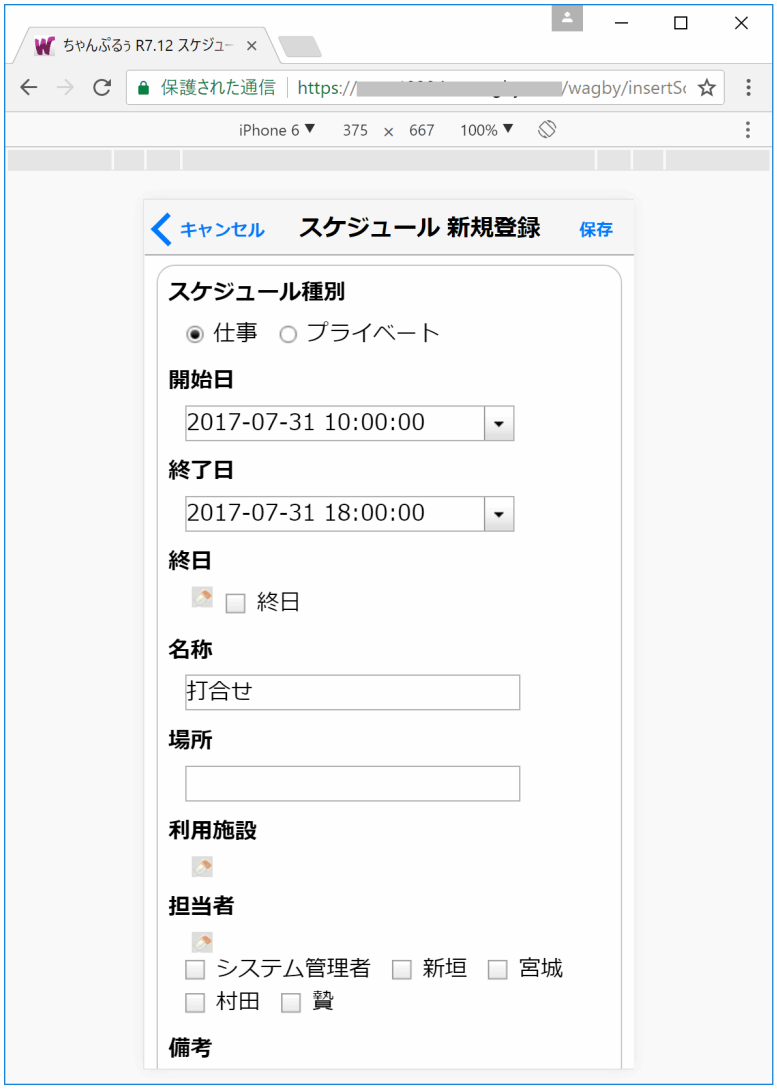


図 カレンダビューモデルへの新規登録(3)

(458)

複数の予定をカレンダ上で確認できます。

(458) 図で入力した値を、図の新規登録画面で受け取るために、モダンカレンダビュー利用モデルに「初期値」を設定する必要があります。 詳細は"画面機能 > カレンダビュー（モダン）>カレンダビューから新規登録画面へ遷移する"をお読みください。

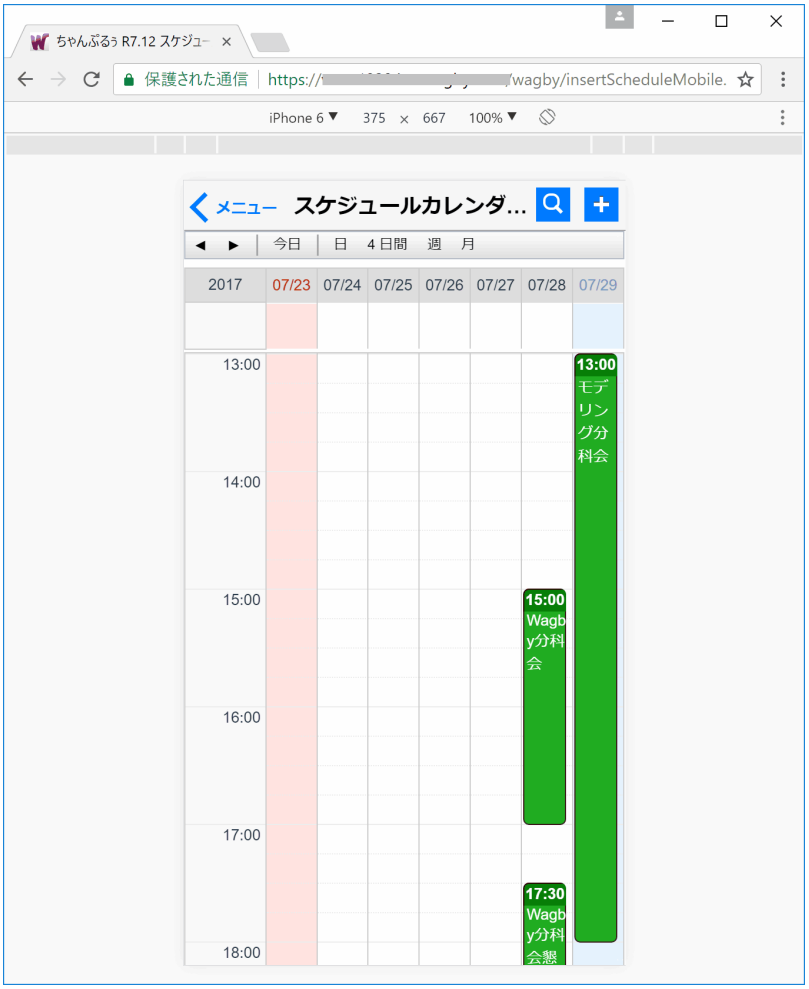


図 複数データの確認

月単位での表示も行うことができます。

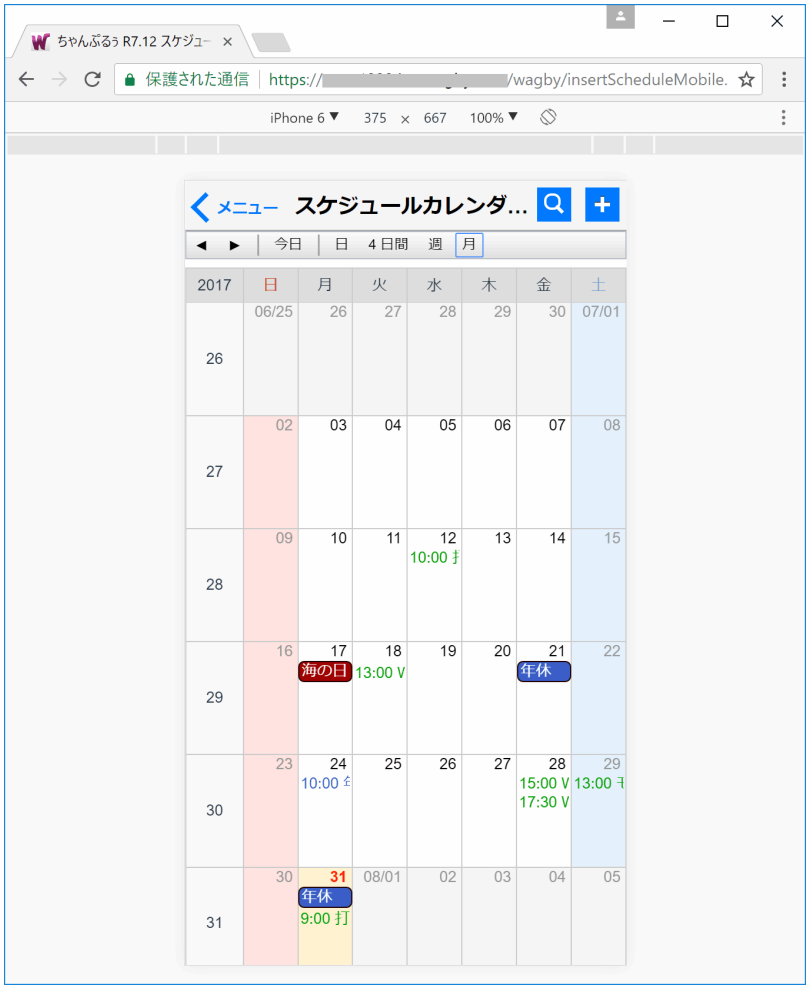


図 月単位での表示

定義方法

PC 向けモデルのサブモデルを新たに用意します。このサブモデルは、スマートフォンの制約を回避するよう配慮して作成します。（制約は後述します。）

サブモデルの作成方法はこちらをお読み下さい。メインモデルとサブモデルは常に次のような関係になります。

メインモデル ＞ サブモデル

図 メインモデルとサブモデルの関係

さらに、サブモデル側の「画面 > その他 > モバイル端末」に「スマートフォン」を選択します。

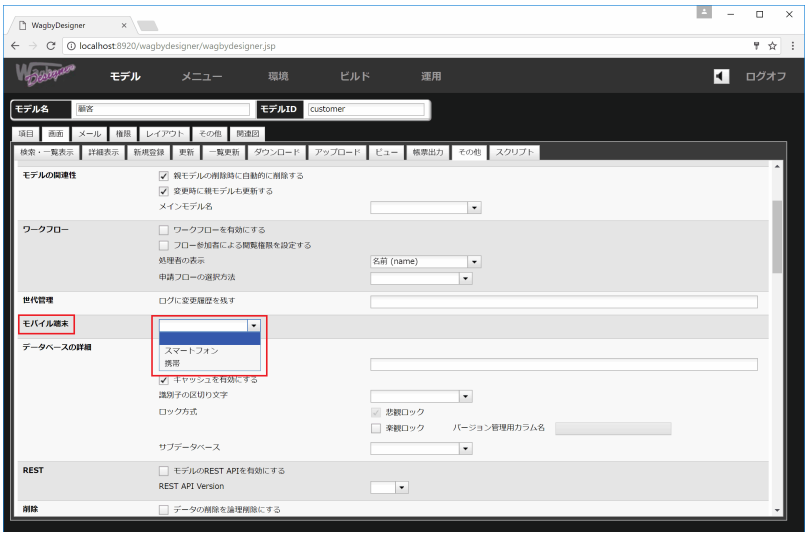


図 モバイル端末の設定

サブモデルを用意せず、一つのモデル定義でPCとスマートフォンの両方を利用できるように設定することもできます。この場合はスマートフォンの制約があるため、PC では使いにくい画面になる懸念があります。そのためスマートフォンから利用する場合は、サブモデルを定義することを推奨します。

テスト方法

開発PCでのテスト

スマートフォンを開発 PC と同一の LAN に（Wi-Fi経由で）接続します。 スマートフォンに付属している Web ブラウザから（開発PC上で起動した）Wagby アプリケーションへ接続してください。

本番環境でのテスト

キャリア （NTTドコモ, KDDI, ソフトバンクなど） が提供する回線を使って外出先から Wagby でビルドしたアプリケーションに接続します。 本番機をインターネットから接続できるよう設定してください。社内LANで閉じている場合は、インターネットからの接続を行うことはできません。

インターネットから接続させるための本番機の設定については、Wagby販売パートナーにご相談ください。

仕様・制約

スマートフォンは PC と異なる画面操作性が必要なため、PC 版とは異なる挙動となります。

動作しない画面動作定義

- 印刷形式で表示
- 一覧表示（グリッド形式）
- 一括更新
- 一括削除
- 一括帳票出力
- 一括メール送信
- CSV・Excel アップロード更新
- マップビュー
- 集計ビュー
- 全文検索
- ウィザード・タブ
- 検索・一覧表示画面のコピー登録ボタン

・　ポータル

その他、動作しない機能
・　横並び、縦並び等の複雑なレイアウト定義
・　管理処理メニュー（管理者用）
・　共通処理（パスワード変更など）
・　パスワードリマインダ機能
・　選択肢モデルの内容を変更するための一覧更新画面

PC版との仕様の差異
・　「ファイル型項目」の扱いが異なります。Android OS では登録・更新画面で写真の選択と送信が可能です。iPhone OS では閲覧は可能ですが登録・更新に未対応です。
・　アクションボタンの表示位置は上下に固定されています。削除・メール送信ボタンは下に表示するようにしています。

iPad利用時のデフォルトをPC画面とする

iPad の Safari を用いて Wagby アプリケーションを操作する場合の標準は「スマートフォン用画面」となっています。これを PC 画面に変える方法を説明します。

動作を試す

次のファイル wagbyapp¥webapps¥プロジェクト識別子¥WEB-INF¥classes¥browsertype.properties をテキストエディタで開きます。以下の3行について、先頭のコメント文字「#」を除去します。

編集前

```
#01.key=.*iPad.*
#01.value=iPad
#01.encoding=default
```

編集後

```
01.key=.*iPad.*
01.value=iPad
01.encoding=default
```

ファイルを保存後、Tomcat の再起動により、変更が反映されます。

変更を反映する

動作確認したファイルを customize/resources フォルダに保存してください。次回ビルド後に、このファイルが常に WEB-INF/classes フォルダにコピーされるようになります。

この変更によるその他の影響

BROWSERTYPE 関数の戻り値が変わります。

スマートフォン利用時のデフォルトをPC画面とする

スマートフォンで Wagby アプリケーションを操作する場合の標準は「スマートフォン用画面」となっています。これを PC 画面に変える方法を説明します。

動作を試す

次のファイル wagbyapp¥webapps¥プロジェクト識別子¥WEB-INF¥classes¥browsertype.properties をテキストエディタで開きます。以下の3行を次のように編集します。

編集前

```
02.key=.*Mobile.*
02.value=MobileSafari
02.encoding=default
```

編集後

```
02.key=.*Mobile.*
02.value=Safari
02.encoding=default
```

02.value の値を "MobileSafari" から "Safari" に変更します。ファイルを保存後、Tomcat の再起動により、変更が反映されます。

変更を反映する

動作確認したファイルを customize/resources フォルダに保存してください。次回ビルド後に、このファイルが常に WEB-INF/classes フォルダにコピーされるようになります。

この変更によるその他の影響

BROWSERTYPE 関数の戻り値が変わります。スマートフォンでアクセスしたときにブラウザタイプが "MobileSafari" ではなく "Safari" となります。

変更履歴

R8.1.2
- PCで申請したワークフローを、スマートフォンで承認・決裁できるようになりました。
- スマートフォンでウィザードを利用できるようになりました。

R8.2.3
- ラジオボタン、チェックボックスの折り返し数の設定が反映されるようになりました。

メッセージの変更・国際化

リソースファイルの編集

リソースファイル

設計情報に含まれる「項目名」などの文字は、すべて「リソースファイル」に格納されています。代表的なリソースファイルを表に示します。

ファイル名	説明
errormsgconv_ja.properties	データベース固有のエラーメッセージを変換します。
errormsg_ja.properties	Wagbyが内部で管理するエラーメッセージです。
jfcapp_ja.properties	アプリケーション全体のメッセージです。
logcode_ja.properties	system.logに出力するメッセージです。

これらのファイルは次のように管理されています。

アプリケーションメッセージ

wagbydesigner/bin/template フォルダに含まれている jfcapp_ja.properties.UTF8 というファイルです。拡張子からわかるように UTF-8 形式のテキストファイルとなっています。

このファイルを土台として、設計情報で用意した項目名がリソースに追加されます。これは wagbydesigner/webapps/wagbydesigner/WEB-INF/env/work/srcgen/webpage/WEB-INF/classes/jfcapp_ja.properties.UTF8 というファイルになります。

エラーメッセージ

wagbydesigner/bin/webpage/WEB-INF/src フォルダに含まれている errormsg_ja.properties.UTF8 というファイルです。拡張子からわかるように UTF-8 形式のテキストファイルとなっています。

最終的なリソースファイル

ビルドされた wagbyapp フォルダ内の WEB-INF/classes フォルダに用意されます。拡張子 UTF8 が除かれた、".properties" ファイルが最終形式です。上記の「テンプレート」に、アプリケーション固有の情報が加わったものとなっています。

ファイル名に "_ja" が付与されているものが日本語リソースです。"_en"など、その他のファイルは「国際化」のページで説明します。

これらのファイルを直接、テキストエディタで変更することはありません。次節以降に説明する方法で書き換えます。

リソースファイルの内容

リソースファイルはテキストファイルです。内部は key-value 形式になっています。

wagbydesigner/webapps/wagbydesigner/WEB-INF/env/work/srcgen/webpage/WEB-INF/classes/jfcapp_ja.properties.UTF8 ファイルの一部を抜粋します。

```
__jfc_common.logon.title=ログオン
__jfc_common.logon.userid=ユーザ名
__jfc_common.logon.password=パスワード
__jfc_common.logon.submit=ログオン
__jfc_common.logon.message={0} さんがログオンしています。
__jfc_common.logon.language=言語:
__jfc_common.logon.class.waku=frmLogon
__jfc_common.logon.class.logonbutton=btnLogon
__jfc_common.logon.display.mobile=表示：モバイル
__jfc_common.search.notfound=検索条件に合致したデータは見つかりませんでした。
__jfc_common.search.action=検索条件を入力してください。
```

ファイル内の 1 行は「キー=メッセージ」というペアで表現されます。キーの部分（「=」の左側）を変更することはできません。

メッセージ部では「プレースホルダ」を使うことができます。これは {0}, {1} という表記です。実際にはこの部分に適切な文字が設定されます。

HTML タグの扱い

リソースの値部分（「=」の右側）は、画面に表示される文字列です。ここに HTML のタグを指定することができますが、これはセキュリティの懸念となる場合があります。開発チームのポリシーで HTML タグを直接、記述することを認めない場合は、「<」「>」「&」「"」といった文字をそれぞれ「<」「>」「&」「"」に置換してください。

(459)

ファイル置換方式（非推奨）

1. wagbyapp/webapps/wagby/WEB-INF/classes/jfcapp_ja.properties.UTF8 をテキストエディタで開きます。

(459) これらの表記は「文字実体参照」と呼ばれるものです。

2. リソースファイルの値部分（イコールの右側）を変更します。

```
...
qa_report.title=タイトル
...
```

> リソースファイルのキー部分（上記では "qa_report.title" の部分）は変更してはいけません。

3. 編集したファイルを customize/resources フォルダに保存します。これによって次回以降のビルド時に、今回編集したファイルが用いられるようになります。

この方法はファイルの置き換えになりますが、非推奨です。このあとの説明にある「差分ファイルの作成」までお読みください。

差分ファイル作成方式（推奨）

リソースファイルの置き換え方式とは別に、差分ファイルを作成する方法を提供しています。差分ファイルとして個別に管理した場合でも、ビルド時にマージ（結合）され、一つの .properties ファイルとしてまとめられます。

1. jfcapp_ja.properties.UTF8 ファイルに対して、接頭語 "my" をつけたファイルである myjfcapp_ja.properties.UTF8 ファイルを新規に作成します。ファイルの中身は空です。このファイルを customize/resources フォルダに保存します。
2. my ではじまるファイルを編集します。このとき、元々あったメッセージと同じ行を記述することで、置き換えることができます。例えば次の行を加えることで、もともと "ログオフ" だったメッセージを書き換えます。

```
__jfc_common.globallink.logoff=ログオフ!
```

3. ビルド時にマージ（結合）されます。

> この "my" をつけたファイルの拡張は、すべてのリソースファイルに対応しています。カスタマイズしたい部分のみを管理することができます。

差分ファイル方式を推奨します

jfcapp_ja.properties.UTF8 は「システムが標準で提供するリソース」と「開発者が追加したモデルに関するリソース」が一つになっています。このファイルを直接、編集すると、Wagbyのバージョンアップ時にシステムが提供するリソースが追加・変更された場合、開発者が手動で修正する必要が生じます。

この手間を省略するため、myjfcapp_ja.properties.UTF8 ファイルを作成することを推奨します。開発者が追加したモデルに関するリソースを、このmyファイルに分離して管理します。

標準で提供される文言を変更する場合も、このmyファイルに記述するとよいです。myファイルに記述したリソースが、標準のリソースを上書きします。

例 モデルごとにコントローラ処理の画面表示メッセージを変更する

myerrormsg_ja.properties.UTF8 を用意します。例えば model1 モデルの新規登録が成功した場合のメッセージを変更したい場合、次のように記述します。

```
model1.success.normal.termination.insert={0}データを登録しました！
```

{0}には、モデル名(日本語)が入ります。

このファイルを customize/resources フォルダに保存し、再ビルドしてください。（差分ビルドです。）

Dojotoolkitに含まれるリソース

JavaScript のダイアログに表示される文言や共通ボタンなどの日本語表記は Wagby が同梱している Dojotoolkit ライブラリが管理しています。この部分の対応は次のようになります。

リソースファイルの場所

リソースファイルはこちらになります。日本語と英語が用意されています。

```
wagbyapp/webapps/<プロジェクトID>/nls/common.js

wagbyapp/webapps/<プロジェクトID>/nls/en/common.js
```

テキストエディタで内容を確認できます。次のような文言が規定されています。

```
  BUTTON_CLOSE: "閉じる",
  BUTTON_DELETE: "削除",
  BUTTON_CANCEL: "キャンセル",
  BUTTON_SAVE: "保存",
  BUTTON_OK: "OK",
  BUTTON_COPY: "コピー",
  BUTTON_DETAIL: "詳細",
  FORMAT_DATE_YOUBI: "曜日",
  ERROR_MSG_CODE: "コード",
  ERROR_MSG_HISSU_1: "「内容」は必須項目です。",
  ERROR_MSG_HISSU_2: "次のコード値に対応する「内容」に値を指定してください。",
  EXCEPTION_MSG_UNEXPECTED: "予期しない値です。",
  EXCEPTION_MSG_FORM_MISSING: "formが取得できませんでした",
  EXCEPTION_MSG_CANTGET_YEAR: "西暦を取得できません。",
  EXCEPTION_MSG_ILLEGAL_MONTH: "不正な月です。",
  EXCEPTION_MSG_ILLEGAL_DATE: "不正な日です。",
  EXCEPTION_MSG_ILLEGAL_MONTH_OR_DATE: "不正な月または日です。",
  EXCEPTION_MSG_ILLEGAL_MONTHDATE: "不正な月日です。",
  EXCEPTION_MSG_ILLEGAL_TIME_OR_MINUTE_OR_SECOND: "不正な時または分または秒です。",
  EXCEPTION_MSG_ILLEGAL_TIMEMINUTESECOND: "不正な時分秒です。",
  EXCEPTION_MSG_UNMATCHED: "日付に完全一致しないです。",
  EXCEPTION_MSG_CAMERA_FAILED: "カメラが起動できませんでした。",
  WARN_MSG_ZIPCODE_NOT_EXIST: "入力された郵便番号は存在しません。",
  WARN_MSG_SESSION_TIMEOUT: "暫く操作されなかったため自動ログオフされました。¥n再ログオンしてください。",
  INFO_MSG_SELECT_BATCH_TARGET: "一括処理対象のデータを選択してください。",
  FLOW_PATTERN_NAME: "フローパターン名",
  NODE_NAME: "ノード名",
  DIAGRAM_TITLE: "フロー参加者",
  SEQUENCE_NODE_NAME: "順次",
  CONSULTATION_NODE_NAME: "合議",
  ERROR_MSG_NO_COMMA_FOR_NODE: "● ノード名にカンマ（,）は入力できません",
  ERROR_MSG_CONFIRM: "入力データに不備があります。以下のメッセージを確認してください。",
  ERROR_MSG_REQUIRE: "● 次の要素は必須項目となっています。",
  ERROR_MSG_SET_PARTICIPANT: "フロー参加者、フロー参加グループ名はいずれか一方を指定するようにしてください。",
  WARNING_KEEP_PASSWORD: "パスワードは入力されていないため、現行のパスワード情報がそのまま利用されます。",
  TIP_MSG_INPUT_DATE: "yyyy-MM-dd 形式で入力して下さい。(yyyy は西暦、MM は月、dd は日です)",
  CHECK_ALL: "全てチェック",
  UNCHECK_ALL: "全てクリア",
  CONFIRM_DELETE: "このデータを削除します。よろしいですか？",
  BUTTON_FILE_SINGLE: "ファイルを選択",
  BUTTON_FILE_MULTIPLE: "複数ファイルを選択",
  ERROR_MSG_SOME_FILES: "複数ファイルを指定できません",
  ERROR_SAVE_LATER: "ファイルのアップロード中です。アップロード後再度行ってください。",
  ERROR_CHECK_FILEMINSIZE: "以下のサイズ以上のファイルである必要があります。",
  ERROR_CHECK_FILEMAXSIZE: "以下のサイズ以下のファイルである必要があります。",
  LABEL_CALENDAR_DIALOG_MODELNAME: "モデル",
  LABEL_CALENDAR_DIALOG_TITLE: "タイトル",
  LABEL_CALENDAR_DIALOG_STARTDATETIME: "開始日時",
  LABEL_CALENDAR_DIALOG_ENDDATETIME: "終了日時",
  LABEL_CALENDAR_DIALOG_ALLDAYFLAG: "終日",
  LABEL_CALENDAR_DIALOG_STAFF: "担当者",
  LABEL_CALENDAR_DIALOG_PLACE: "場所",
```

```
LABEL_CALENDAR_DIALOG_URL: "URL",
LABEL_CALENDAR_DIALOG_MEMO: "備考",
LABEL_CALENDAR_DIALOG_BTN_SHOW: "詳細画面表示",
LABEL_CALENDAR_DIALOG_BTN_UPDATE: "更新画面表示",
LABEL_CALENDAR_DIALOG_BTN_INSERT: "新規登録画面表示",
LABEL_CALENDAR_NEW_REGISTRATION: "新規登録",
ERROR_MSG_DELETE_CALENDAR_ITEM: "カレンダデータの削除に失敗しました",
LABEL_SHOWLIST_CHOICE: "選択",
LABEL_UPDATELIST_DELETE: "削除",
LABEL_UPDATELIST_STATUS: "ステータス",
LABEL_UPDATELIST_RESULTDIALOG_OK: "一覧表示画面へ戻る",
LABEL_UPDATELIST_RESULTDIALOG_CANCEL: "一覧更新を継続する",
LABEL_UPDATELIST_NEW_BUTTON: "新規",
LABEL_UPDATELIST_COPY_BUTTON: "コピー",
LABEL_UPDATELIST_CLEAR_BUTTON: "クリア",
LABEL_UPDATELIST_SEARCH_BUTTON: "検索...",
INFO_MSG_UPDATELIST_NEW: "新規",
INFO_MSG_UPDATELIST_COPY: "コピー",
INFO_MSG_UPDATELIST_UPDATE: "編集有",
ERROR_MSG_UPDATELIST_LOCKERROR: "ロック失敗",
ERROR_MSG_UPDATELIST_ERROR: "エラー",
CONFIRM_MSG_UPDATELIST_SAVE: "入力エラーデータは無視して保存します。<br>よろしいですか？",
ERROR_MSG_GET_GRID_SETTINGS: "グリッド設定取得時にエラーが発生しました",
ERROR_MSG_SET_GRID_SETTINGS: "グリッド設定保存時にエラーが発生しました",
INFO_MSG_SAVE_GRID_SETTINGS_SUCCESS: "グリッド設定が保存されました",
INFO_MSG_DELETE_GRID_SETTINGS_SUCCESS: "グリッド設定が削除されました",
LABEL_CLEAR_GRID_SETTING: "グリッド設定をクリア",
LABEL_BTN_SAVE_GRID_SETTING: "保存",
LABEL_BTN_DELETE_GRID_SETTING: "削除",
LABEL_TOTAL_DISTANCE: "合計距離",
LABEL_SQUARE_METER: "平方メートル",
LABEL_SQUARE_KILOMETER: "平方キロメートル",
LABEL_TOTAL_AREA: "総面積",
TITLE_SCRIPT_EDITOR: "スクリプトエディタ",
CONFIRM_MSG_SCRIPT_EDITOR: "変更を破棄してスクリプトエディタを閉じますか？",
LABEL_JFCJOBSCHEDULE_DIALOG_CHKANYTIME: "任意の時間を指定(全範囲を許容)",
LABEL_JFCJOBSCHEDULE_DIALOG_TIMETYPESPECIFIC: "特定のタイミング",
LABEL_JFCJOBSCHEDULE_DIALOG_CHKONLYWEEKDAYS: "平日のみ(土日を除く)",
LABEL_JFCJOBSCHEDULE_DIALOG_CHKUSEWEEKNUMBER: "週の指定を行う",
LABEL_JFCJOBSCHEDULE_DIALOG_WEEKNUMBER_PREFIX: "第",
LABEL_JFCJOBSCHEDULE_DIALOG_WEEKNUMBER_SUFFIX: "週",
LABEL_JFCJOBSCHEDULE_DIALOG_TIMETYPEINTERVAL: "インターバル",
LABEL_JFCJOBSCHEDULE_DIALOG_TIMETYPE_DURATION: "期間(範囲)指定",
LABEL_JFCJOBSCHEDULE_DIALOG_UNIT_SECOND: "秒",
LABEL_JFCJOBSCHEDULE_DIALOG_UNIT_MINUTE: "分",
LABEL_JFCJOBSCHEDULE_DIALOG_UNIT_HOUR: "時",
LABEL_JFCJOBSCHEDULE_DIALOG_UNIT_DAY: "日",
LABEL_JFCJOBSCHEDULE_DIALOG_UNIT_MONTH: "月",
LABEL_JFCJOBSCHEDULE_DIALOG_UNIT_DOW: "曜日",
LABEL_JFCJOBSCHEDULE_DIALOG_UNIT_YEAR: "年",
LABEL_JFCJOBSCHEDULE_DIALOG_TITLE_HOUR: "時刻(時)",
LABEL_JFCJOBSCHEDULE_DIALOG_INTERVAL_1: "から",
LABEL_JFCJOBSCHEDULE_DIALOG_INTERVAL_2: "おき",
LABEL_JFCJOBSCHEDULE_DIALOG_DURATION_1: " ～ ",
LABEL_JFCJOBSCHEDULE_DIALOG_OPT_LAST: "最終",
LABEL_JFCJOBSCHEDULE_DIALOG_DOW_1: "日",
LABEL_JFCJOBSCHEDULE_DIALOG_DOW_2: "月",
LABEL_JFCJOBSCHEDULE_DIALOG_DOW_3: "火",
LABEL_JFCJOBSCHEDULE_DIALOG_DOW_4: "水",
LABEL_JFCJOBSCHEDULE_DIALOG_DOW_5: "木",
LABEL_JFCJOBSCHEDULE_DIALOG_DOW_6: "金",
LABEL_JFCJOBSCHEDULE_DIALOG_DOW_7: "土",
LABEL_JFCGROUPNODE_UNSETGROUP: "未設定グループ",
LABEL_JFCGROUPNODE_GROUP_HIERARCHY_TAB: "グループ階層図",
```

```
  LABEL_JFCGROUPNODE_SEARCH_TAB: "検索",
  CONFIRM_MSG_SAVE_GROUP_HIERARCHY: "グループ階層を保存しますか?",
```

リアルタイムエラーチェックのリソースファイル

リアルタイム入力チェックで表示されるエラーメッセージ「入力した値は無効です。」を変更する方法を説明します。

このメッセージは dojo-release-1.xx.yy/dijit/form/nls/ja/validate.js というファイルに次の内容で用意されています。

(460) (461)

```
define(
"dijit/form/nls/ja/validate", ({
      invalidMessage: "入力した値は無効です。",
      missingMessage: "この値は必須です。",
      rangeMessage: "この値は範囲外です。"
})
);
```

上記ファイルを直接修正し、customize/webapp/dojo-release-1.xx.yy/dijit/form/nls/ja/validate.js として保存し、ビルドしてください。

確認方法

1. ビルド後にブラウザのキャッシュをクリアします。
2. ブラウザの言語設定で "英語(アメリカ合衆国)" または "英語" を先頭に設定します。

(462)

トラブルシューティング

Windows OS に付属の「メモ帳」利用時のトラブル

Windows OS に付属する「メモ帳」を使って UTF8 ファイルを編集しないようにしてください。TeraPad など、別のテキストエディタをご利用ください。

[詳細] メモ帳を文字コード UTF8 で保存すると BOM と呼ばれる特殊マークがファイルの先頭に付与されます。この影響で、Wagby が正しくファイルの内容を扱えないことが生じます。

なお Windows 10 May 2019 Update 1903 より、メモ帳は「BOMなしUTF-8」で保存できるようになりました。この形式であれば問題ありません。

国際化

リソースファイル

設計情報に含まれる「項目名」などの文字は、すべて「リソースファイル」に格納されています。 リソースファイルの詳細は"リソースファイルの編集"をお読みください。

ファイル名に "_ja" が付与されているものが日本語リソースです。"_en"が英語リソースです。出荷時の製品には日本語と英語のリソースファイルが同梱されています。

(460) xx および yy の部分は Wagby が同梱している Dojotoolkit フレームワークのバージョン番号が入ります。適切に読み替えてください。

(461) "ja"は日本語を表現します。他国語の場合、この部分は適切なロケール表記に変える必要があります。ただし英語だけは基本ロケールなのでフォルダで分けられておらず、dojo-release-1.xx.yy/dijit/form/nls/validate.js となります。ロケール表記の詳細は国際化の説明をお読みください。

(462) ログオン画面の言語設定では切り替わりません。

英語リソースファイルの編集

1. wagbyapp/webapps/wagby/WEB-INF/classes/jfcapp_en.properties.UTF8 をテキストエディタで開きます。（Windowsに付属の「メモ帳」は使いません。TeraPad など別のテキストエディタをご用意ください。）
2. リソースファイルのところどころが日本語表記になっています。これらを英語に翻訳します。

```
...
qa_report.title=title
...
```

リソースファイルのキー部分（上記では "qa_report.title" の部分）は変更してはいけません。

3. 編集したファイルを customize/resources フォルダに保存します。これによって次回以降のビルド時に、今回編集したファイルが用いられるようになります。

この方法はファイルの置き換えになりますが、非推奨です。このあとの説明にある「差分ファイルの作成」までお読みください。

差分ファイルの作成

リソースファイルの置き換え方式とは別に、差分ファイルを作成する方法を提供しています。差分ファイルとして個別に管理した場合でも、ビルド時にマージ（結合）され、一つの .properties ファイルとしてまとめられます。

1. jfcapp_en.properties.UTF8 ファイルに対して、接頭語 "my" をつけたファイルである myjfcapp_en.properties.UTF8 ファイルを新規に作成します。ファイルの中身は空です。このファイルを customize/resources フォルダに保存します。
2. my ではじまるファイルを編集します。このとき、元々あったメッセージと同じ行を記述することで、置き換えることができます。例えば次の行を加えることで、もともと "Logoff" だったメッセージを書き換えます。

```
__jfc_common.globallink.logoff=Log OFF !
```

3. ビルド時にマージ（結合）されます。

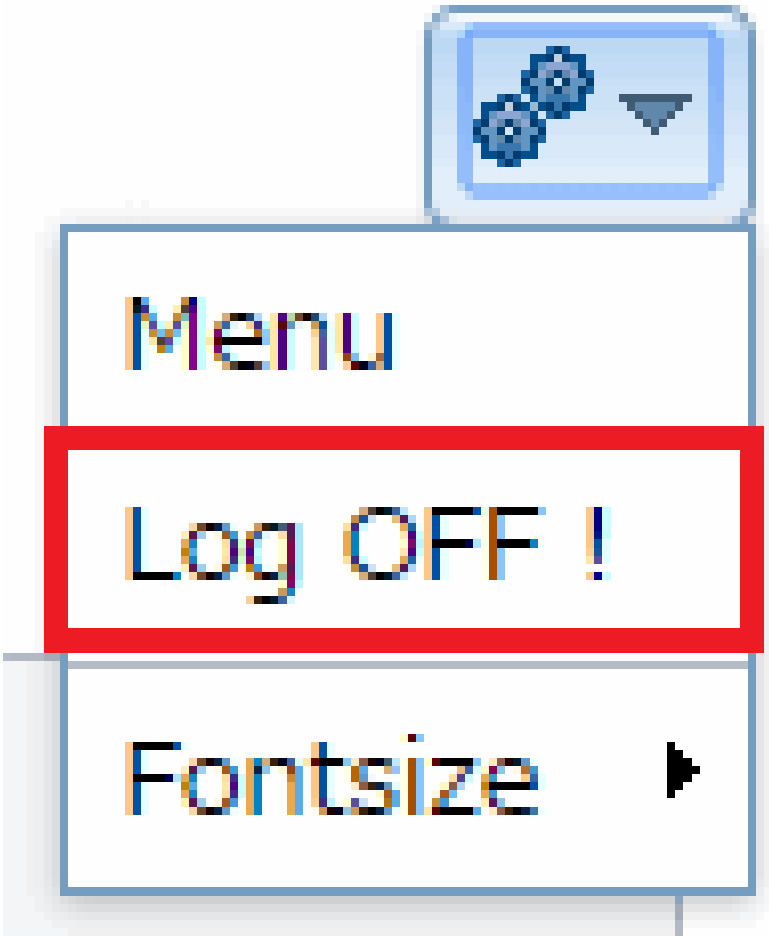


図 既存リソースの置き換え

この "my" をつけたファイルの拡張は、すべてのリソースファイルに対応しています。カスタマイズしたい部分のみを管理することができます。

差分ファイル方式を推奨します

jfcapp_XX.properties.UTF8 は「システムが標準で提供するリソース」と「開発者が追加したモデルに関するリソース」が一つになっています。このファイルを直接、編集すると、Wagbyのバージョンアップ時にシステムが提供するリソースが追加・変更された場合、開発者が手動で修正する必要が生じます。

この手間を省略するため、myjfcapp_XX.properties.UTF8 ファイルを作成することを推奨します。開発者が追加したモデルに関するリソースを、このmyファイルに分離して管理します。

標準で提供される文言を変更する場合も、このmyファイルに記述するとよいです。myファイルに記述したリソースが、標準のリソースを上書きします。

休日ファイルの変更

カレンダビューで表示される「休日」は iCalendar 形式のファイルで記述されています。標準では Thunderbird が提供する日本の公休日情報 "JapanHolidays.ics" を同梱しています。

これを他国用に変更することができます。

1. customize/webpage/WEB-INF フォルダに、その国向けの ics ファイルを保存します。
2. カスタマイズフォルダ customize/resources に myapplication.properties ファイルを用意し、次の行を加えます。 R8.4以降

```
server.servlet.context-parameters.icsfilename=(ファイル名)
```

R8.3.xまで

```
server.context-parameters.icsfilename=(ファイル名)
```

3. ビルドを行い、wagbyappを再起動します。指定した ics ファイルが読み込まれます。

(463)

ユーザ定義入力チェックメッセージ

入力チェック（ユーザ定義チェック）の設定で、メッセージを国際化対応することができます。

ERRORMESSAGE 関数を使います。引数はリソースキーです。

```
ERRORMESSAGE("E0001")
```

myerrormsg_ja.properties.UTF8を編集する

リソースキーに対応するエラーメッセージは、myerrormsg_ja.properties.UTF8 に「キー=値」の書式で記述します。このファイルをcustomize/resources に保存します。

記述例を示します。

```
E0001=このデータは登録できません。
E0002=このデータは {0} のため登録できません。
E0003=このデータは {0} のため {1} できません。
```

ここで {0} や {1} は「プレースホルダ」となります。ERRORMESSAGE 関数の第一引数はリソースキー（上のファイルで E0001 に相当する）ですが、第二引数以降に固定文字列を渡すことができます。

```
ERRORMESSAGE("E0002", "空白が含まれている")

ERRORMESSAGE("E0003", "空白が含まれている", "登録")
```

- メッセージ部分に記述できる関数は ERRORMESSAGE のみです。他の関数は利用できません。
- 引数はすべて固定文字列となります。ダブルクォーテーションで囲ってください。（引数に関数やモデルの値を利用する場合は、スクリプトの利用を検討してください。）

言語パック（英語）

言語パック（英語）をインストールした Wagby は開発から運用まで、すべて英語化されます。一部メッセージの置換も行うため、日本語に戻すことはできません。完全に英語モードとして動作させたい場合に適用してください。

言語パックには次の内容が含まれています。

- WagbyDesignerのメッセージを英語表記とする。(言語の環境変数を英語に変えた起動スクリプトを含む。)
- 製品では一部、日本語が固定となっていた箇所を英語に置き換える。
- ワークフロー、ポータル、ジョブ管理に関する初期データの英語化。

入手方法

wagby.comのダウンロードサイトから「言語パック（英語）ランタイム版」を入手できます。言語パックのインストール方法は、このページにリンクとして用意しています。

日・英以外の国用のリソースファイルの作成

次のリソースファイルを翻訳することで、日・英以外の国に対応することができます。"<ロケール>" は ISO-639 に準拠した指定になります。この詳細は「サポートされているロケール」をお読みください。例えば台湾は zh_TW となります。

ファイル	説明	修正のタイミング

(463) 標準の "ファイル名" は JapanHolidays.ics となっています。

customize/resources/jfcapp_<ロケール>.properties.UTF8	システムメッセージを含む、基本的なリソース。(*1)	モデルの修正の都度。
customize/resources/logcode_<ロケール>.properties.UTF8	ログコード。(*1)	初回のみ。
customize/resources/errormsg_<ロケール>.properties.UTF8	エラーメッセージ。(*1)	初回のみ。
customize/resources/errormsgconv_<ロケール>.properties.UTF8	データベースのメッセージの変換用。(*1)	初回のみ。
customize/webpage/nls/<ロケール>/common.js	Dojotoolkit用の標準メッセージ。(*2)	初回のみ。

(464) (465)

最初に用意する方法

1. アプリケーションビルドする

   翻訳する直前のアプリケーションをビルドします。

2. 英語リソースをコピーする

   ビルドされたアプリケーションに含まれる英語リソースのファイルを、対象となる国向けのファイルとしてコピーします。

   ここでは中国 （ロケールIDをzhとする）のリソースを作成する例で説明します。以下のコマンドを参考にしてください。(*3)

(466)

```
copy wagbyapp/webapps/<プロジェクト識別子>/WEB-INF/classes/jfcapp_en.properties customize/
resources/jfcapp_zh.properties.UTF8
copy wagbydesigner/bin/webpage/WEB-INF/src/logcode_en.properties customize/resources/
logcode_zh.properties.UTF8
copy wagbydesigner/bin/webpage/WEB-INF/src/errormsg_en.properties customize/resources/
errormsg_zh.properties.UTF8
copy wagbydesigner/bin/webpage/WEB-INF/src/errormsgconv_en.properties.UTF8 customize/resources/
errormsgconv_zh.properties.UTF8
copy wagbydesigner/bin/webpage/WEB-INF/src/i18n.txt customize/resources/i18n.txt
mkdir customize/webapp/nls
mkdir customize/webapp/nls/zh
copy wagbydesigner/bin/webpage/nls/common.js customize/webapp/nls/common.js
copy wagbydesigner/bin/webpage/nls/en/common.js customize/webapp/nls/zh/common.js
```

3. myjfcapp の作成

   customize/resources/myjfcapp_zh.properties.UTF8 を用意します。次の行を含めます。

```
__jfc_common.dojo_webpage_dir=dojo-release-1.14.1
```

   実際には Wagby に同梱されている JavaScript フレームワーク Dojotoolkit のバージョン番号を含めます。1.14.1の部分はご利用の Wagby に含まれている Dojotoolkit にあわせて適切に読み替えてください。(*4)

---

(464) 1. テキストエディタで編集するファイルは拡張子 .properties.UTF8 として、UTF-8 を編集できるテキストエディタを使ってください。ビルド時にこのファイルから ".properties" へ変換されます。実際のアプリケーションは（変換された）".properties" ファイルを読み込んでいます。

(465) 2. JavaScript ファイルですが、メッセージが含まれています。

(466) 3. copy はファイルをコピーするコマンドで、第一引数のファイルを第二引数のファイルとしてコピーすると解釈してください。mkdir はフォルダを作成するコマンドです。実際の操作はコマンドラインまたはファイルエクスプローラを使って、以下に記載のコマンドと同様のことを行なってください。Unix系OS利用時は copy を cp に読み替えてください。

(467)

なお上記ファイルに翻訳した文言を含めることができます。ビルドしたアプリケーションは、ここに記載した内容が反映されます。例えば次の行を加えると、メニューの右横にあるボタンの文言 ″ログオフ″ が変わります。

```
__jfc_common.globallink.logoff=注#
```

4. nls/<ロケール>/common.js リソースの修正

customize/webpage/nls/common.js の末尾部分に 'zh': true, を追加します。(*5)

```
...
'ja-jp': true,
'zh': true,
});
```

(468)

5. ログオン画面からの選択

ログオン画面で、今回追加した「国」を選択させる場合は「言語の選択 > 選択できる言語を追加する」を行なってください。

このあとにモデルの追加など、リポジトリの変更が生じた場合は wagbyapp/webapps/<プロジェクト識別子>/WEB-INF/classes/jfcapp_en.properties が変わります。差分をとって過不足の生じたリソースがあれば手動でマージしてください。

仕様・制約

- 一覧表示（グリッド）部品の列名は国際化に未対応です。将来のWagbyのバージョンアップで対応する予定です。

言語の選択

ログオン画面による指定

ログオンフォームに、言語を選択するリストボックスが用意されます。

(467) 4. Dojotoolkitのバージョンは、ビルドしたアプリケーション wagbyapp 内のフォルダ webapps/wagby/dojo-release-1.XX.YY で確認できます。XX と YY には数字が入ります。

(468) 5. JavaScript で利用しているロケール文字は、各画面の _javascript.jsp ファイル内の setLocale("<c:out value='${sessionScope.__jfc_locale}'/>"); 関数で設定されています。${sessionScope.__jfc_locale} の文字列が利用するリソースファイルのフォルダ名と一致している必要があります。(上記例の場合 "zh")。念のため利用しているブラウザで、目的とする「言語」のロケール情報を確認するようにするとよいでしょう。この調査には、WagbyのLOCALE関数を利用できます。

図 ログオンフォーム(1)

標準では「（空白）」「日本語」「英語（English）」の三種類を選択することができます。

図 ログオンフォーム(2)

選択肢は次のとおりです。

選択肢	動作
（空白）	アカウント作成直後など、言語を選択しない状態では、ブラウザの言語情報(Accept-Language)を用いた自動判定を行います。 一度でも何かの言語を選択した場合は、以前に指定した言語となります。つまり、一度選択した言語の情報はユーザー毎に記録されています。
日本語	日本語になります。リソースファイルは "ja" です。
English	英語になります。リソースファイルは "en" です。

プレファレンスによる指定

ログオン後に、プレファレンスから言語を選択することもできます。

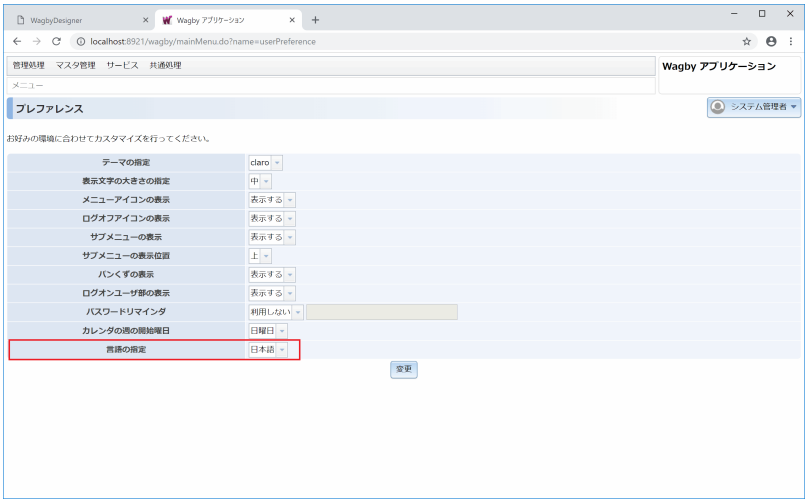


図 プレファレンス

選択するとすぐにメニューやラベルなどが変更されます。

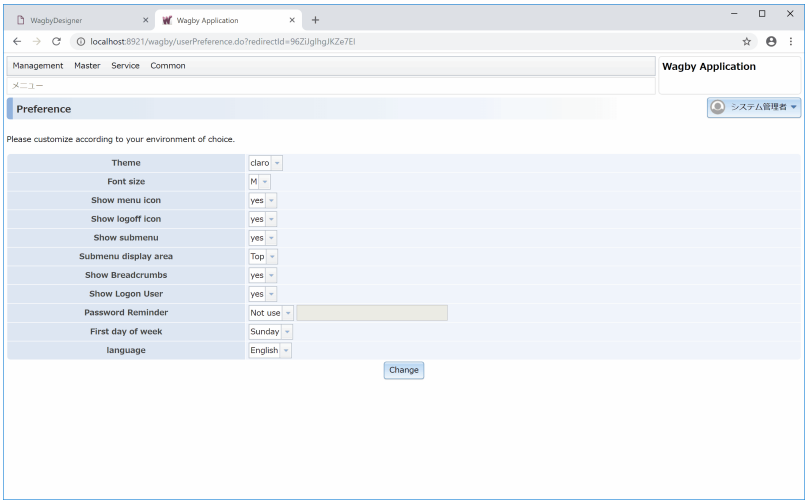


図 プレファレンスで言語を変更した

選択できる言語を追加する

ログオンフォームに用意される言語の選択肢を増やすことができます。

i18n.txtを編集する

1. wagbydesigner/bin/webpage/WEB-INF/src/i18n.txt ファイルを、customize/resources フォルダにコピーします。
2. customize/resources/i18n.txt を編集します。インストール直後は次のようになっています。

```
ja=日本語
en=English
```

3. key-value形式で、行を追加します。なお先頭文字に '#' を加えるとコメントとみなされます。ログオンフォームの選択肢には、この記述順で表示されます。

```
# コメントです。
ja=日本語
en=English
de=Deutsch
```

4. ビルドします。今後は customize/resources/i18n.txt の内容がアプリケーションに反映されます。

選択肢モデルの特別表記

表記方法

選択肢モデルの内容部に言語情報を付与した、次の書式を利用することができます。

```
{ "en":"Description in English","ja":"日本語の説明" }
```

例えば性別を示す選択肢モデル gender を用意したとき、内容部を次のように設定することができます。

```
{ "en":"Man","ja":"男" }
```

Designerでは次のように初期データを設定します。

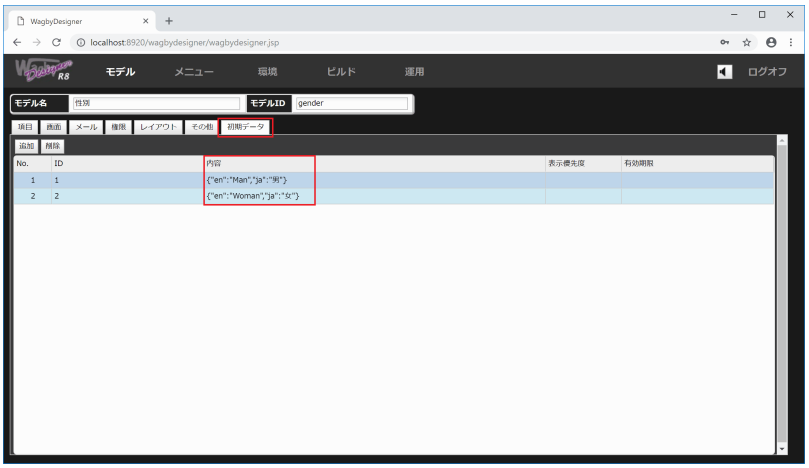


図 Designerでの初期データ設定

例

日本語でログオンした場合、選択肢モデルの内容部は ″ja″ の値が用いられます。

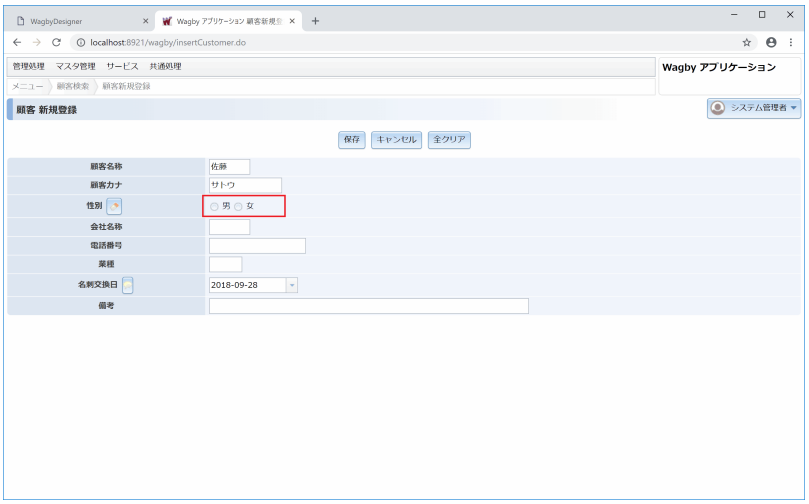


図 選択肢モデルの内容部jaが表示される

英語(English)でログオンした場合、選択肢モデルの内容部は ″en″ の値が用いられます。

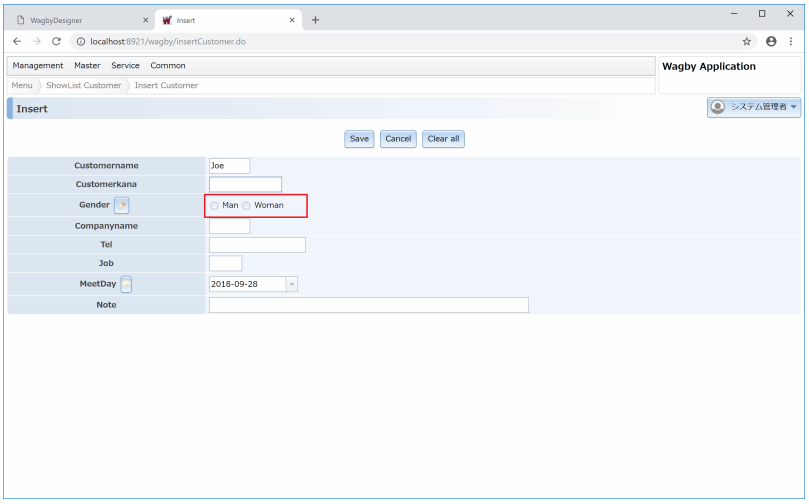


図 選択肢モデルの内容部enが表示される

運用中に選択肢モデルの内容部を変更する

選択肢モデルの一覧表示時は、言語に応じた内容部が表示されます。

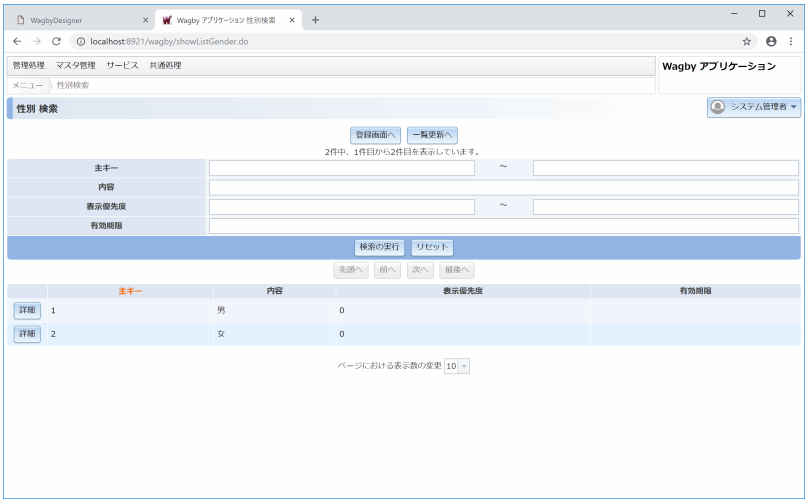


図 選択肢モデルの一覧表示

しかし一覧更新画面では、言語情報込みの表記となります。この形式で値を編集してください。

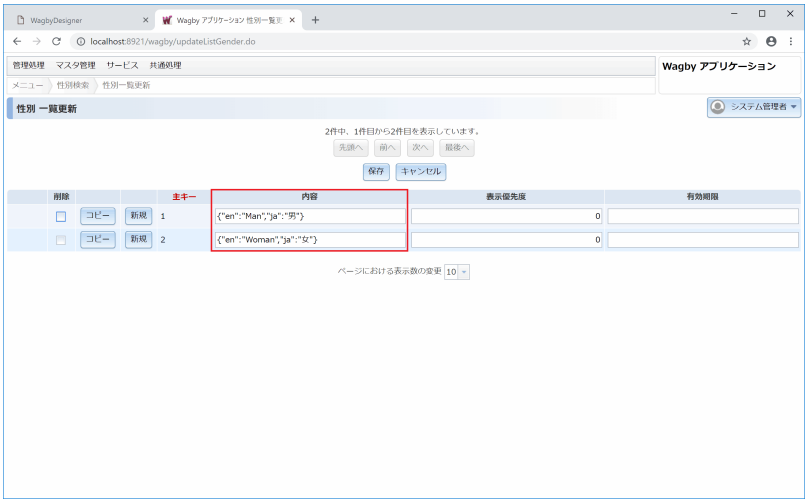


図 選択肢モデルの一覧更新

環境

プロジェクト

プロジェクト情報

「環境」メニューから「プロジェクト」を選択します。

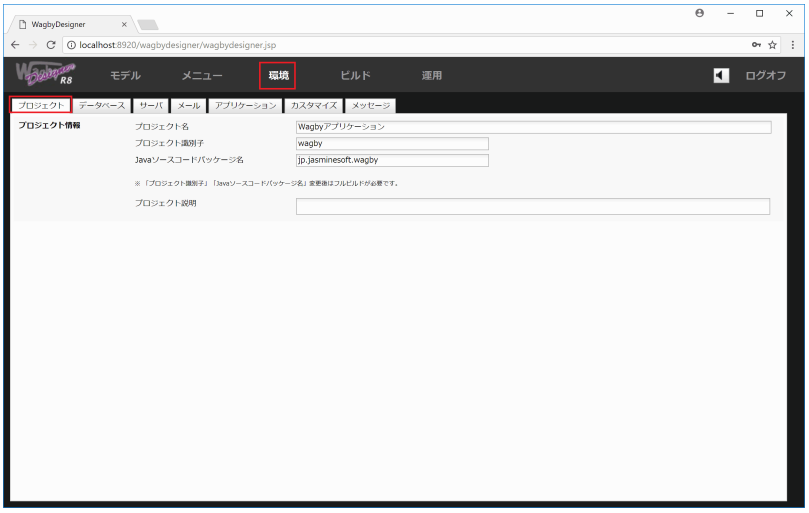


図 プロジェクトの設定

プロジェクト名

構築するアプリケーションの名称を入力します。インストール直後は「Wagbyアプリケーション」になっています。

ここで入力した値はログオン後のアプリケーションのタイトルに反映されます。

プロジェクト識別子

プロジェクト識別子は英数字のみで構成します。インストール直後は「wagby」になっています。

プロジェクト識別子を空白にすることもできます。この場合 Web ブラウザのアドレスバーに入力する際にプロジェクト名が取り除かれます。（例 http://localhost:8921/）

本マニュアル中では、プロジェクト識別子を $(APPNAME) と表現しています。適切に読み替えてください。

Javaソースコードパッケージ名

生成するJavaソースコードのパッケージ名を指定します。

Java ではパッケージ名は「重複しないように」という配慮から、インターネットのドメイン名を逆順に使うことが推奨されています。 例えば foo.bar.com というドメインをもっている組織では "com.bar.foo" となります。 さらに続けて、サブパッケージを使うことができます。例えば "com.bar.foo.myapp" などです。

SIer 様の場合、このパッケージ名には納品先のお客様がもつドメイン名を使うとよいでしょう。

Java によるカスタマイズを行わない場合は、デフォルト値である「jp.jasminesoft.wagby」をそのままお使いください。

Java によるカスタマイズを行いつつ、途中でパッケージ名を変更した場合は、カスタマイズコードのパッケージ名もあわせて変更するようにしてください。

パッケージ名の命名規則

- 先頭は英文字、「_(アンダースコア)」を指定して下さい。
- 二文字目以降は英文字、「_(アンダースコア)」、半角数字を指定して下さい。
- 「_(アンダースコア)」と半角数字のみで構成することはできません。
- 予約語を指定することはできません。（予約語については Appendix > 予約語をお読み下さい。）

このルールにより、プロジェクト名に 「-(ハイフン)」を含めることはできません。

フルビルド

「プロジェクト識別子」「Javaソースコードパッケージ名」変更後はフルビルドが必要です。

データベース

データベースの設定

「環境」メニューから「データベース」を選択します。

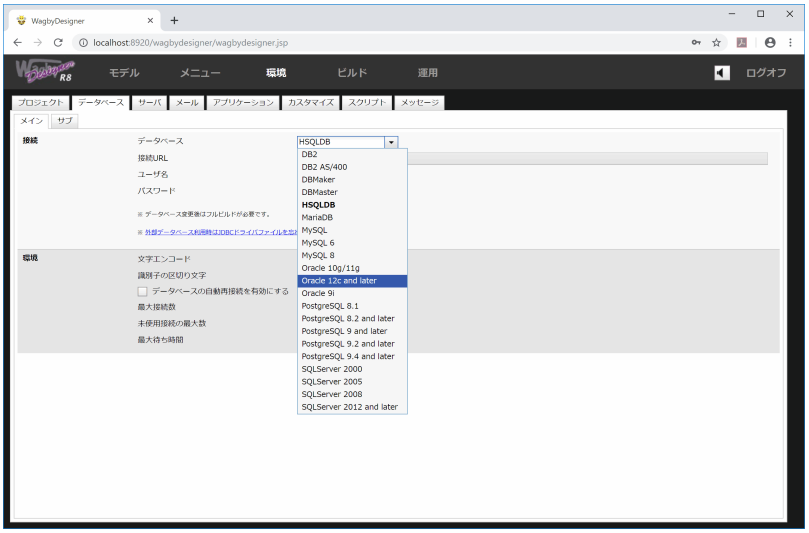


図 データベースの設定

メインデータベースとサブデータベース

1つのWagbyアプリケーションから、複数のデータベースを利用することができます。詳細は〝モデル > サブデータベースを指定する〟をお読みください。

データベース接続

データベース

次のデータベースから一つを選択できます。

- HSQLDB（組み込みDB）Wagby同梱版のみ、変更不可
- PostgreSQL
- Oracle

- SQLServer
- DB2
- DB2 AS/400
- MySQL
- DBMaker/DBMaster

利用できるデータベースのバージョンは「環境 > 本番運用サーバ」のページを参照してください。

上記に記載していないデータベースを利用するために、拡張することもできます。

Wagbyでは、HSQLDB以外のデータベースを「外部データベース」と呼んでいます。外部データベースを利用する場合は、そのデータベースに対応した「JDBCドライバファイル」を用意する必要があります。詳細は"サポート>データベース活用ガイド>接続手順>JDBCドライバの配置"をお読みください。

接続先データベースを変更した場合は「フルビルド」を行う必要があります。

接続URL

リレーショナルデータベースへ接続するためのURLを指定します。 選択されたリレーショナルデータベースによってURLの書式が異なります。 詳細は"サポート>データベース活用ガイド>接続手順>WagbyDesignerにおけるデータベースの設定"をお読みください。

ユーザ名

リレーショナルデータベース側で用意されたユーザ名を指定します。 予め、データベース側にユーザを作成しておいてください。

パスワード

利用するリレーショナルデータベース側で用意されたユーザ名に対応したパスワードを指定します。 予め、データベース側にユーザのパスワードを設定しておいてください。

データベース環境

文字エンコード

外部データベースの文字コードを指定することができます。標準は UTF-8 となっています。

この値は自由入力形式となっています。利用する外部データベースの作成時に指定した文字コードを転記するようにしてください。

データベース製品に固有の文字エンコーディングを指定することはできません。例えばオラクル製品が提供する "JA16SJIS" を指定することはできません。この場合は代わりに "Windows-31J" としてください。(*1)

(469)

識別子の区切り文字

Wagby は標準で、SQL文における識別子の区切り文字を設定するようになっています。

このため、Wagby が作成したテーブルを（外部システムなどで）SQL 文を使って利用する場合はテーブル名、カラム名に区切り文字を付与する必要があります。例えば、次のようになります。

```
select * from "juser" where "id" > 0
```

(469) 1. この文字エンコードの設定は、文字数の入力チェック（バイト指定）でも利用するためです。Javaが許容する文字エンコーディングを用いてください。

この設定を「区切り文字を指定しない」に変えることができます。 この場合、モデルIDやモデル項目IDに利用できないパターンが生じることに配慮する必要があります。 詳細は "Appendix > SQL文における識別子の区切り文字"をお読みください。(*2) [470]

区切り文字を指定しない設定のとき、利用するデータベースによってはテーブル作成に失敗することがあります。これはデータベースの予約語が（テーブルの）列名に使われていたことが原因です。このような場合はエラーとなった列の物理名を指定して回避してください。

データベースの自動再接続を有効にする

Wagby が利用している外部データベースを再起動する場合、Wagby で構築したアプリケーション(wagbyapp) も再起動する必要があります。

本設定を有効にすると、データベースへ問い合わせる度に、接続確認処理を行なうようになります。アプリケーションの再起動は不要になります。(*3)

[471]

この設定を行うと、接続確認処理のためシステムのパフォーマンスが若干、低下します。外部データベースの再起動処理が頻繁に行われないようであれば、本設定は適用しないことを推奨します。

この設定は「Wagbyアプリケーションを起動したあとに、データベースを起動する」というパターンには対応していません。Wagbyアプリケーションは起動時にデータベース接続を行いますが、最初の接続に失敗すると（以降、データベースが回復しても）動作しません。

最大接続数

アプリケーションとデータベースの間の通信に利用される仮想的な「コネクション（接続）」の最大数を設定します。 このコネクションは、複数の利用者（Webブラウザ）で再利用されます。初期値は「100」です。1より大きい数を指定してください。

例えば Wagby の同時認証接続数が 200 の場合、最大 200 名の利用者がログオンします。ここで、コネクションの最大数が 100 の場合、200 名の利用者が 100 本の仮想線を使い回すことになります。Wagby の同時認証接続数に合わせて、この数値を調整することでパフォーマンスが向上します。

この値に -1 を指定すると「無制限」となります。なお、この数値を変更した場合、それに合わせてデータベース側の設定変更（接続受け入れ本数の上限数の指定）が必要になる場合があります。詳細はご利用になる外部データベースのマニュアルをお読みください。

「データベースへの最大接続数」が小さい場合、運用中にデータベース接続エラーが発生することがあります。初期値 100 を下回る設定を行う場合はご注意ください。

未使用接続の最大数

未使用状態になっているコネクションの最大数を指定します。0 を指定すると「無制限」となります。初期値は「10」です。

最大待ち時間

ある利用者によってコネクションが使われたあと、次の利用者が使えるようになるまでの待ち時間を指定します。「-1」を指定すると無制限（永久に待ち続ける）となります。単位はミリ秒です。初期値は「2000ミリ秒」です。

スキーマ

サブデータベースで有効です。(*4)

ここで設定した値は、SQL 発行時にテーブル名の先頭に付与されます。例えばスキーマに "FUNNY" を設定すると、SELECT 文が "FUNNY.テーブル名" となります。

---

[470] 2. HSQLDB(組み込みDB)では、識別子の区切り文字は指定できません。

[471] 3. 本設定は内蔵データベースを用いている場合には適用されません。

(472)

対応する設定ファイル

すでに運用中の本番機の設定を変更する場合、次の説明を参考に設定ファイルを編集してください。

設定	対応する設定ファイル	設定場所
接続URL	wagbyapp¥webapps¥ $(APPNAME)¥META-INF ¥context.xml	Resource/@name="jdbc/jfc" 要素のurl属性
ユーザ名		Resource/@name="jdbc/jfc" 要素のusername属性
パスワード		Resource/@name="jdbc/jfc" 要素のpassword属性
最大接続数		Resource/@name="jdbc/jfc" 要素のmaxActive属性
未使用接続の最大数		Resource/@name="jdbc/jfc" 要素のmaxIdle属性
最大待ち時間		Resource/@name="jdbc/jfc" 要素のmaxWait属性

同様に wagbyapp¥webapps¥$(APPNAME)¥WEB-INF¥export¥conf¥initdb.xml も修正してください。initdb.xmlの詳細な説明はこちらをお読みください。

注意事項

外部データベースを利用する場合は、それぞれのデータベースにおける使用上の注意事項に目を通してください。 例えば DB2 の場合、表スペース・ページ・サイズの変更を行うか、または各モデル項目の型を個別に定義するといった追加作業が必要になることがあります。

- データベース活用ガイド

コネクションプーリング

「データベース環境 > 最大接続数」は、コネクションプーリングという技術を使って管理されます。 標準では Commons DBCP を使います。

- Wagby はコネクションプールからコネクションを取得するという処理を行います。コネクションの接続や解放は、上記コネクションプール機構が行います。
- Commons DBCP は「プール内に空き接続が無ければ、新たに接続を作って返す」というルールとなっています。

詳細は"データベース活用ガイド > コネクションプーリング"をお読みください。

関連するページ

- ビルドせずにデータベースの接続先を変更する

サーバ

サーバの設定

「環境」メニューから「サーバ」を選択します。

(472) 4. メインデータベースでのスキーマ指定は現在、開発中です。将来の Wagby R8 のバージョンアップで対応予定です。

図 サーバの設定

Tomcat

この説明は、ビルドするWebアプリケーション（wagbyappフォルダ）に反映されます。すなわち、運用するアプリケーションのための設定です。[473]

JVM最大ヒープメモリサイズ(MB)

Java Virtual Machine(JVM)の最大ヒープメモリサイズを指定することができます。初期値は"512"です。単位は MB（メガバイト）です。

> 大量のデータを扱う場合や、アプリケーションに同時にログオンする利用者が多い場合は、このメモリサイズを大きくする必要があります。

JVM初期ヒープメモリサイズ(MB)

Java Virtual Machine(JVM)の初期ヒープメモリサイズを指定することができます。初期値は"64"です。単位は MB（メガバイト）です。

HTTPポート番号

Tomcatが使用するHTTPポート番号を指定します。初期値は"8921"です。

> Wagbyでは、標準のポート番号として 8921 を使っています。この値を 80 に変えることで、Webブラウザのアドレスバーに入力するURLから ":8921" というポート番号表記をなくすことができます。ただしこの場合、お使いのマシンですでに別のWebサーバが動作していないことを確認してください。たとえばWindows Server系OSの場合、IISというWebサーバが起動していれば、すでに80番ポートを使用しています。この場合はIISを停止してください。

> IIS や Apache Web Server と（Wagbyが提供する）Tomcatを同じポート80番で共存させるような設定を行うこともできます。定義方法が複雑なため、本マニュアルでは割愛します。詳細についてはWagby販売パートナーへご相談ください。

AJP1.3ポート番号/パスワード (8.4.2)

この設定は Apache Web Server と連携するときに使用します。初期値は空欄となっており、AJP1.3ポートは利用しないようになっています。設定する場合 "8009" が使われることが多いですが、任意です。

---

[473] コンパイル時のメモリ不足など、WagbyDesignerへの変更は"ビルド時のメモリサイズを調整する"をお読みください。

さらにポート番号指定時に "AJP1.3 パスワード" を設定することもできます。指定した場合は server.xml の Connector 要素の secret 属性に値が設定されます。接続元にも同じパスワードを設定してください。

参考資料

- Apache Tomcat 8 Configuration Reference
- Apache Module mod_proxy (mod_proxy_ajpの場合はProxyPassにsecret=xxxと指定する)
- The Apache Tomcat Connectors - Reference Guide (mod_jkの場合はworkers.propertiesのsecretを指定する)

サーバポート番号/サーバシャットダウンパスワード (8.4.2)

Tomcatが使用するサーバポート番号を指定します。初期値は "8005" です。

この値を空白（または負の数）とすると、サーバポートを利用しなくなります。

シャットダウンのパスワード

サーバポートは外部から shutdown 命令を受け付けることができます。このパスワードを設定します。未指定時の初期値は "SHUTDOWN" + プロジェクト識別子（すべて大文字）です。

(474)

リダイレクトポート番号

Tomcatが使用するリダイレクトポート番号を指定します。初期値は "8443" です。

（この設定は通常、変更する必要はありません。）

送信リクエストのセッションIDをチェックする(CSRF対応)

送信リクエストIDのチェックを行うかどうかを選択します。初期値は有効になっています。

この設定はセキュリティレベルを向上させるものです。具体的にはCSRF（Cross Site Request Forgeries）といったログオンユーザの権限を利用したサイト攻撃手法が行えなくなります。

ビルド

Javaのバージョン

自動生成されるJavaソースコードが、どのJavaでコンパイルできるかを指定します。初期値は "1.8" (Java 8) になっています。Java 9 を利用する場合、Javaのバージョンに "9" を指定します。1.9ではないのでご注意ください。

"1.6" や "1.7" を指定することはできません。Java 8 以上が対象になります。

開発の途中でこのバージョンを変更した場合は、フルビルドを行ってください。サーバサイドJavaScriptのコードも再作成されます。

Wagbyのバージョンと、対応するJavaバージョンの対応表

こちらをご確認ください。

キャッシュ

Wagbyは一度データベースから読み込んだ値をメモリにキャッシュします。そのキャッシュサイズを指定することができます。単位はメガバイトとなっており、標準は100メガバイトとなっています。

キャッシュ機構の詳細は "Appendix > キャッシュ処理の詳細" をお読みください。

(474) プロジェクト識別子が "wagby" の場合、シャットダウンパスワードは "SHUTDOWNWAGBY" になります。

HTTPセッション

HttpSession オブジェクトを格納する場所を指定します。「マルチセッション」を利用する場合に設定します。

格納方式	説明
HTTP Session	マルチセッションを利用しません。選択肢が空白の場合、こちらになります。
Spring Session HashMap(テスト用)	マルチセッションを利用します。HttpSessionをメモリ内のハッシュマップで管理します。テスト用であり、本番運用を行うことはできません。
Spring Session Redis	HttpSession を Redis データベースで管理します。マルチセッションを本番で利用する場合は、こちらを使ってください。

Spring Session HashMap の制約

"Spring Session HashMap" は開発中にマルチセッションを試すために用意されています。この方式はセッションタイムアウト（利用者が一定期間操作しなかった場合、自動的にログオフされる機構）に未対応のため、本番運用には用いないでください。(*1)

Spring Session Redis を利用する場合

Redis はキーバリュー型のデータストア製品で、オープンソースで提供されています。次のパラメータを指定することができます。

設定欄	説明
Redisホスト名	Redisサーバのホスト名を指定します。
Redisパスワード	Redisサーバに接続するパスワードを指定します。
Redisポート番号	Redisサーバに接続するポート番号を指定します。
Spring Session ネームスペース8.2.3	複数のWagbyアプリケーションが1つのRedisサーバを利用する場合、個々のWagbyアプリケーションを識別するためのネームスペースを指定することができます。空白時は "spring:session:" が暗黙的なネームスペースとして利用されます。例えばここに "wagbyapp1" と指定することで、"spring:session:" が自動的に付与された "spring:session:wagbyapp1" というネームスペースで利用されます。なお1つのWagbyアプリケーションで1つのRedisサーバを利用する場合は、本設定は空白でも問題ありません。本設定を行う場合は、(複数の)Wagbyアプリケーションをそれぞれ識別するための文字列を指定する、ということになります。

Wagby アプリケーションを起動する前に、Redis を起動しておく必要があります。 また Wagby アプリケーションを再起動する場合、Redis の保存内容を消去しておくようにしてください。詳細は Wagby Developer Network のページで説明します。

(475)

認証

アカウント > Windows Active Directory/LDAP による認証 をお読みください。

(475) 1. マルチセッションは Wagby が内部で利用している Spring framework の機能を使って実現しています。Spring Session HashMap はセッションを格納するデータベースが不要ですぐに動作を確認できる利点がありますが、開発元の Spring チームではこれをテスト用と位置づけており、セッションタイムアウト機構に対応させていません。

クラスタリング

Wagby Developer Network(R8) > クラスタリングの設定 をお読みください。

関連するページ

- ビルドせずに Tomcat のポート番号を変更する

メール

メールの設定

Wagby自身にはメールサーバ機能はありません。すでに貴社内で利用しているメールサーバを使います。以下の設定は（運用中の）メールサーバに合わせて行ってください。

「環境」メニューから「メール」を選択します。

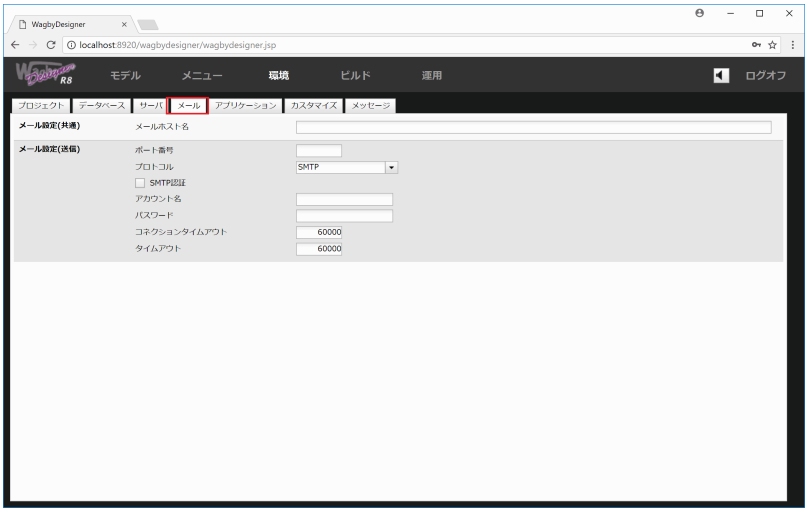


図 メールの設定

メール設定(共通)

メールホスト名

ここで指定するメールホストはプロジェクト全体で共通に利用できます。

各メール送信設定のメールホストを空欄とした場合、ここで指定したメールホストが使われます。

メール設定(送信)

ポート番号

メール送信ポート（SMTP）の番号を指定します。省略時は選択したプロトコルの標準のポート番号が用いられます。SMTPの場合は25番、SMTP over SSLの場合は465番です。

プロトコル

SMTP,SMTP over SSL,STARTTLS からいずれか一つを選択します。

SMTP認証

SMTP認証機能が必要な場合、本設定を有効にします。 SMTP認証は、"LOGIN" と "PLAIN" 方式に対応しています。("CRAM-MD5", "DIGEST-MD5", "NTLM" といった方式には対応していません。)

アカウント名

SMTP認証を有効とした場合、認証用のアカウント名を指定します。

パスワード

SMTP認証を有効とした場合、認証用のアカウントに対するパスワードを同時に指定します。

コネクションタイムアウト

メールサーバとの通信に関するコネクションタイムアウト時間を指定します。単位はミリ秒です。標準値は 60,000 ms (60秒) となっています。省略時は無制限 (タイムアウトなし) となります。

タイムアウト

メールサーバとの通信に関するI/Oタイムアウト時間を指定します。単位はミリ秒です。標準値は 60,000 ms (60秒) となっています。省略時は無制限 (タイムアウトなし) となります。

アプリケーション

アプリケーションの設定

「環境」メニューから「アプリケーション」を選択します。

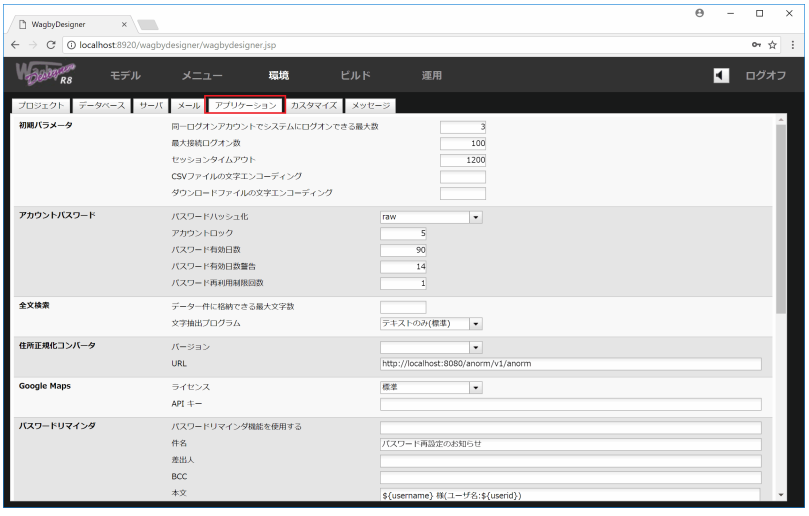


図 アプリケーションの設定

初期パラメータ

同一ログオンアカウントでシステムにログオンできる最大数

同一アカウントでログオンできる最大数を指定することができます。標準値は3となっています。この値を -1 とすると、無制限になります。

ここで指定したログオン回数を超えたログオンを試みた時には、自動ログオフ機能 (もっとも古い時刻のログオンセッションをログオフさせる) により、そのユーザをログオン可能にします。

自動ログオフ機能を無効にすることで、ログオンさせないようにすることもできます。

ジョブを実行するアカウントは、ジョブ実行時のみ、この設定は適用対象外となります。つまりジョブの実行は同時に複数行うことができます。8.1.1

(476)

最大接続ログオン数

アプリケーションへログオンできる最大数を指定します。この数を超えたログオンを行うことはできません。標準値は100となっています。

ログオンできる最大数の調整は、お使いのハードウェアの性能（CPU速度、メモリ量）に依存します。最適値についての一般的な計算式はありません。この値のチューニング方法等については、Web サーバ設定の専門家またはWagby販売パートナーにご相談ください。

セッションタイムアウト

Webアプリケーションサーバのセッションタイムアウトの設定を行うことができます。単位は秒です。利用者がここで指定した時間、何もアクションをおこさなかった場合、自動的にログオフ処理を行います。標準値は「1,200秒（20分）」となっています。

参考情報 - application.properties

この設定は、ビルドしたアプリケーションの次のファイルに反映されます。 wagbyapp/webapps/(プロジェクト名)/WEB-INF/classes/application.properties

カスタマイズフォルダ customize/resources に myapplication.properties ファイルを用意し、次の行を加えてビルドすることで、設定を変更することができます。

R8.4以降

```
server.servlet.context-parameters.session_timeout=1200
```

R8.3.xまで

```
server.context-parameters.session_timeout=1200
```

ビルド後、Webアプリケーションを再起動すると設定が有効になります。

Tomcat 単体ではなく Apache などの Web サーバと連携している場合、Webサーバのタイムアウト設定もあわせて行なってください。（詳細は Web サーバのマニュアルをお読みください。）

参考情報 - session-timeout

この値は、Tomcat が提供する session-timeout 設定に反映されます。そのため、とりうる値の範囲（仕様）は Wagby が同梱している Tomcat（またはご自身で用意したTomcatで運用する場合は、そのバージョン）に準じます。

CSVファイルの文字エンコード

CSV ダウンロード時およびアップロード更新時、CSV ファイルの文字エンコードを指定することができます。 ここで任意の文字コードを指定します。標準は空白となっており、これは「Windows-31J」と解釈されます。

多言語のデータを扱う場合、utf-8 と指定してください。[詳細...]

参考情報 - Windows 31J について

- JSPで特殊文字が文字化けする場合の対処方法 - http://www.atmarkit.co.jp/ait/articles/0501/14/news118.html
- Microsoftコードページ932 - http://ja.wikipedia.org/wiki/Microsoft%E3%82%B3%E3%83%BC%E3%83%89%E3%83%9A%E3%83%BC%E3%82%B8932

(476) 「ジョブ専用アカウント」プリンシパルを持つユーザーが多重ログオン無制限となるわけではありません。ジョブ実行時のみに限定されます。

アカウントパスワード

「アカウント>パスワード」の章で説明します。

全文検索

「検索制御 > 全文検索 > 文字抽出プログラムの指定」の章で説明します。

メッセージキュー

「Wagby Developer Network > メッセージによる非同期処理 > メッセージキューの設定」の章で説明します。

パスワードリマインダ

「アカウント>パスワードリマインダ」の章で説明します。

デフォルトログオン

「アカウント>デフォルトログオン」の章で説明します。

カスタマイズ

カスタマイズの設定

「環境」メニューから「カスタマイズ」を選択します。

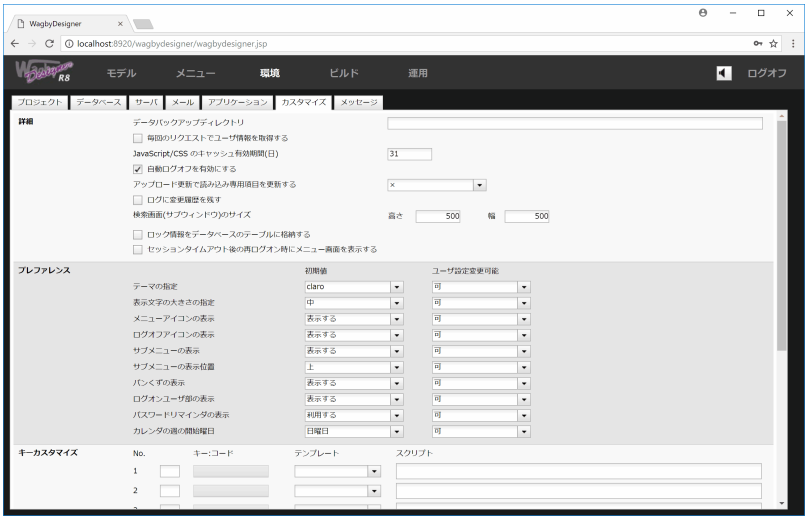


図 カスタマイズの設定

詳細

データバックアップディレクトリ

データのエクスポート機能使用時のデータを保存するディレクトリを指定します。 標準は空白となっており、この場合は wagbyapp と同じ位置の export ディレクトリとなります。 なお、ディレクトリは wagbyapp ¥bin を起点とした相対パスとして指定してください。

毎回のリクエストでユーザ情報を取得する

画面描画毎にログオンアカウントの存在チェックを行います。 ログオン後に抹消されたログオンアカウントはログオフするまでシステムを利用できてしまいますが、この設定を有効にすると、抹消後に何らかの画面操作を行おうとするとすぐに強制ログオフさせることができます。

この設定を有効にすると画面操作の都度、juserテーブルの存在チェックを行うためパフォーマンスが若干低下します。 ログオンアカウントの厳密な運用を行いたい場合に、本設定を有効にしてください。

JavaScript/CSS のキャッシュ有効期間

Wagby ではいくつかのイメージ画像,HTML,JavaScript,CSS ファイルを Web ブラウザにキャッシュすることにより、パフォーマンスを向上させています。このキャッシュの有効期限を指定します。標準は「31」(日)です。

この値を 0 とするとキャッシュは行われなくなります。パフォーマンス向上のため、通常はこの数値を変更せず、デフォルトのままで利用されることを推奨します。

(477) (478)

自動ログオフを有効にする

Wagbyでは、利用者がログオフ処理を行わずにWebブラウザを閉じた場合、これ（ブラウザが閉じられたこと）を検出できません。そのため、定められたセッションタイムアウトまで、この利用者は再ログオンできなくなります。

自動ログオフ機能は、この問題を回避する目的で用意されました。

同一ユーザのログオン処理で、許容されたログオン回数を超えたログオンを試みた時に、過去の最終時刻のセッションを解放し、そのユーザをログオン可能にします。

標準では有効になっています。これを解除すると、上限数を超えた（同一ユーザの）ログオン処理は行えなくなります。

この設定を行う場合は、あわせて同一アカウントでログオンできる最大数を指定して下さい。

「同時接続数を 1」とし、かつ「自動ログオフ機能を無効」とした場合は、ユーザーがログオフせずにブラウザを閉じた場合は、セッションタイムアウトまでユーザーは再ログオンできなくなります。ご注意ください。

この設定を行っても、ご利用頂いているライセンスの上限に達している場合は自動ログオフ機能が働きません。 （正確には、ライセンス上限を超えないログオンであることが確認されたあとに、この自動ログオフ判定が働きます。つまり本機能はライセンス上限を解決するオプションではなく、同一アカウントでログオンできる最大数の上限を解決するオプションという位置づけです。）

アップロード更新で読み込み専用項目を更新する

標準では読み取り専用項目はアップロード更新の対象から除外されます。この設定を「○」とすると、読み取り専用項目であってもアップロード更新の対象となります。

この設定はモデル単位ではなく、アプリケーション全体の挙動変更となります。

ログに変更履歴を残す

(477) 1 キャッシュの有効期限が切れていないファイルを再取得したい場合は、ブラウザのキャッシュクリアを行ってください。（キャッシュをクリアする以外で、ファイルの再取得を行うことはできません。）

(478) 2 すでにブラウザにファイルがキャッシュされている状態で本設定を変更しても、ブラウザのキャッシュが切れるまでは新しい有効期間は反映されません。

Wagbyが出力するログ（logsフォルダ内のsystem.logファイル）は標準ではデータの変更履歴出力は無効となっています。この設定を有効にすると、アプリケーション全体で（すべてのモデルで）変更履歴出力機能が有効になります。

この設定によりログファイルのサイズは巨大になります。アプリケーション全体ではなく、必要なモデル毎に変更履歴を残すように設定することもできます。

検索画面(サブウィンドウ)のサイズ

検索画面(サブウィンドウ)の標準の高さと幅を指定できます。初期値は500ピクセルとなっています。

(479)

ロック情報をデータベースのテーブルに格納する

この設定を有効にすると、外部のプログラムからWagby側で該当データがロック(画面ロック)されているかどうかを確認することができます。

セッションタイムアウト後の再ログオン時にメニュー画面を表示する (8.2.1)

標準では、セッションタムアウト後に再ログオンすると、直前の画面が表示されます。この設定を有効にすると、再ログオン時には通常のログオンと同じように、メニュー画面を表示するようになります。

(480)

プレファレンス

プレファレンスのページで説明します。

キーカスタマイズ

ファンクションキーを含む、キーイベント動作を変更する

プレファレンス

概要

WagbyDesignerでは、プレファレンスの初期値を設定します。このプレファレンスはユーザ毎にカスタマイズすることができます。

(479) この設定はアプリケーション全体で共通です。

(480) 本機能は R8.2.1 以降の Wagby で利用できます。（または R8.1.2 向けのパッチにも含まれています。）

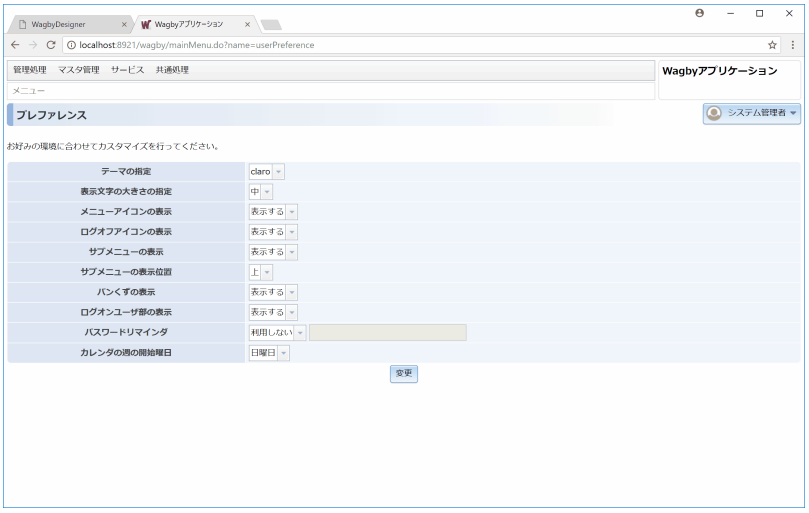


図 共通処理タブに用意されたプレファレンス設定画面（ユーザ毎に設定できる）

「ユーザ設定変更可能」を「不可」とした場合、ユーザによるプレファレンスの変更は行えなくなります。

CSSテーマ

ビルドしたアプリケーションのデザインを選択することができます。標準では "claro" というテーマが選択されています。"flat_purple", "flat_blue", "flat_orange", "soria", "tundra", "nihilo" というテーマが用意されています。

claro（標準）

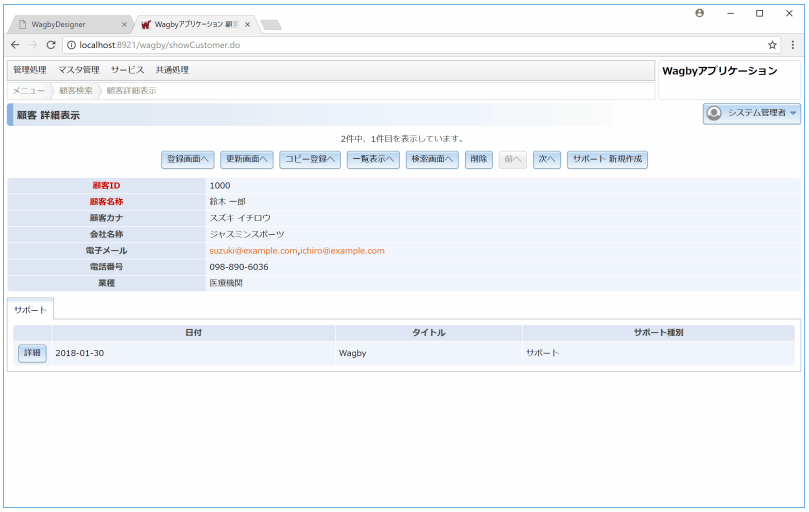


図 CSSテーマ(claro)

flat_purple

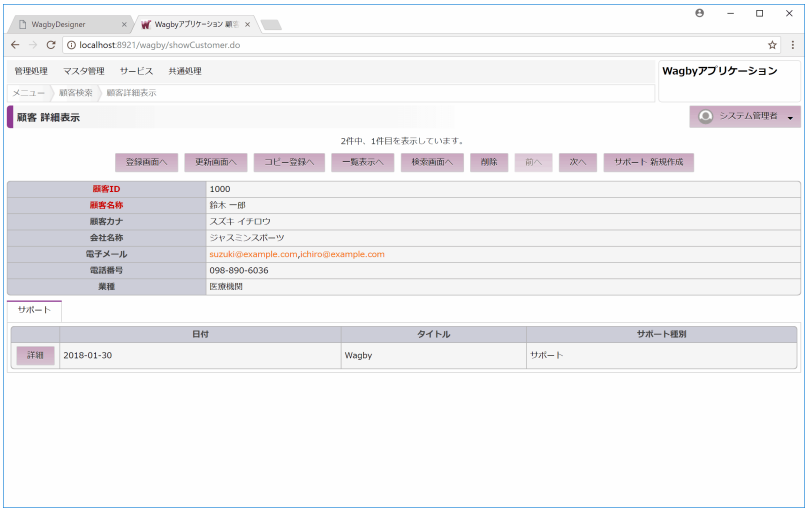


図 CSSテーマ(flat_purple)

flat_blue

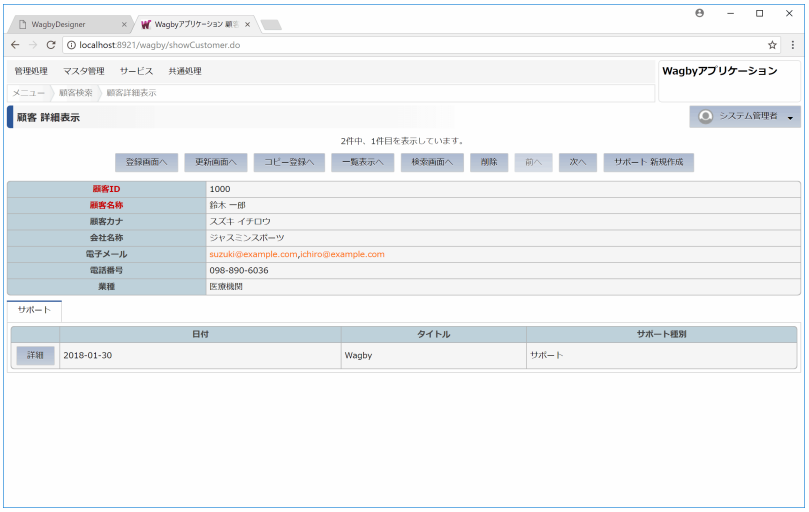


図 CSSテーマ(flat_blue)

flat_orange

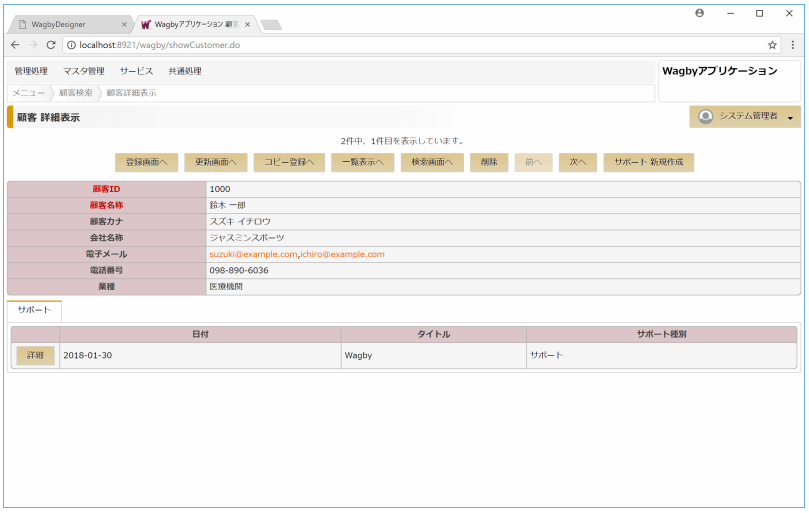


図 CSSテーマ(flat_orange)

soria

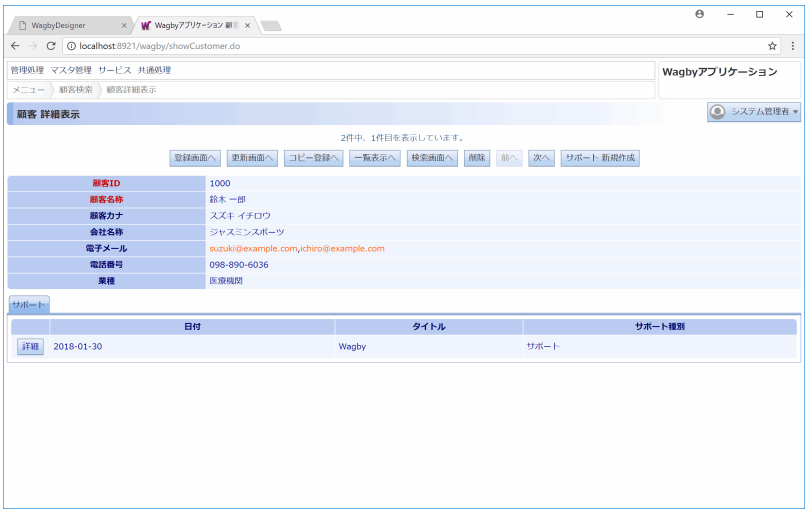


図 CSSテーマ(soria)

tundra

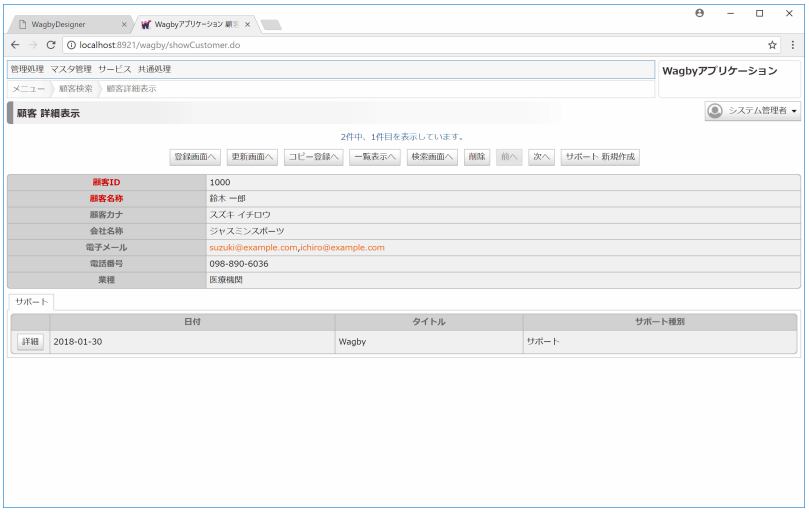


図 CSSテーマ(tundra)

nihilo

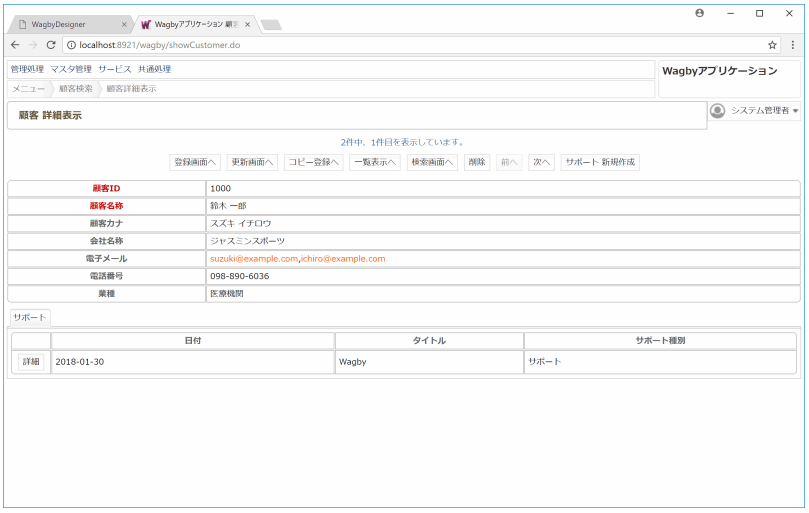


図 CSSテーマ(nihilo)

(481)

表示文字の大きさの指定

文字の大きさを「大」「中」「小」の三段階から指定することができます。標準は「中」です。

「小」の場合

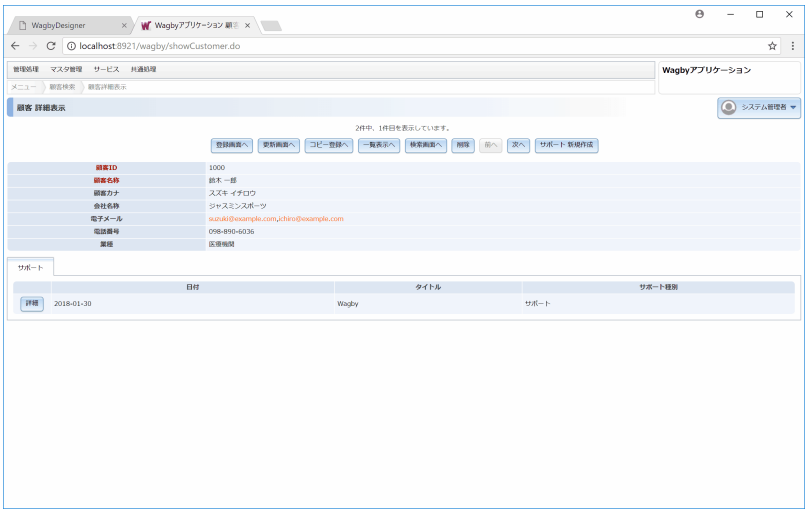


図 フォントサイズ「小」

「中」の場合

(481) nihilo は HTML 帳票の CSS としても利用されます。

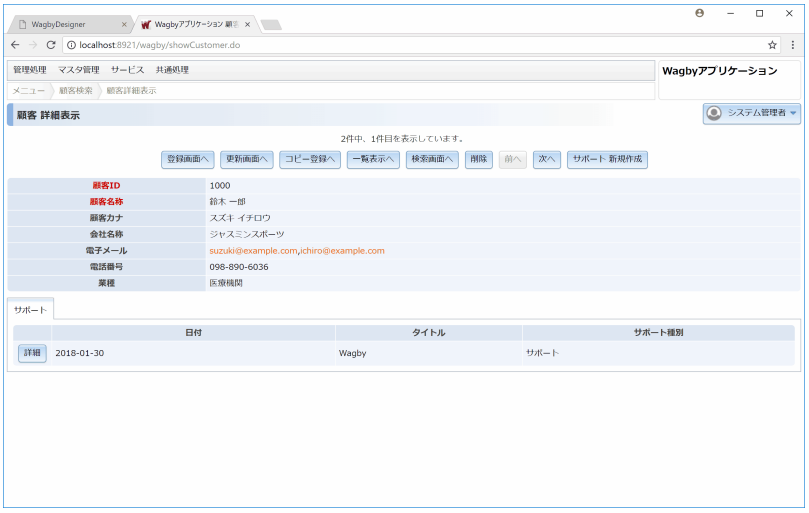


図 フォントサイズ「中」

「大」の場合

図 フォントサイズ「大」

メニューアイコン

ビルドしたアプリケーションに用意されたギアアイコンを押下すると、「メニューへ戻る」というリンクが用意されます。

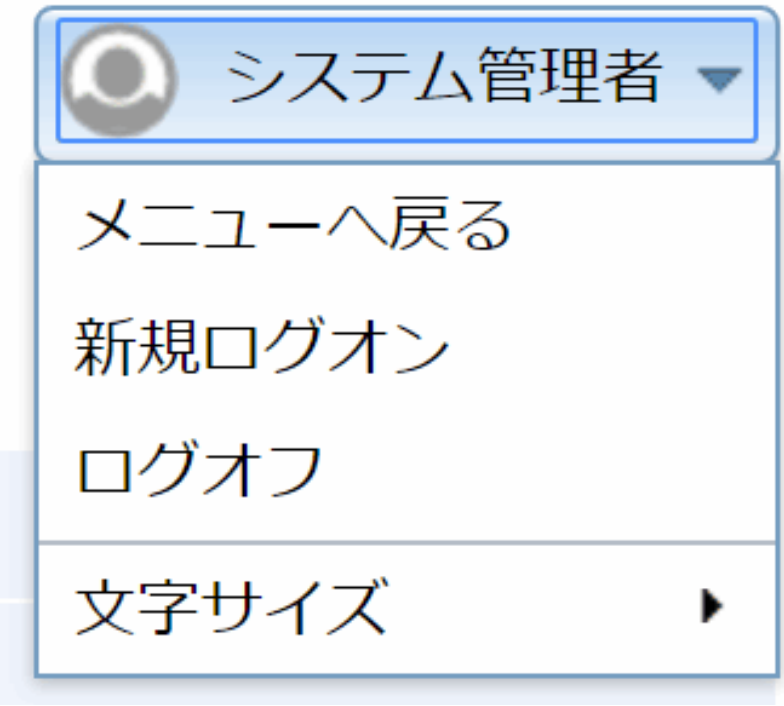


図 メニューへ戻る

または、アプリケーション名をクリックすることでもメニュー画面に戻ることができます。

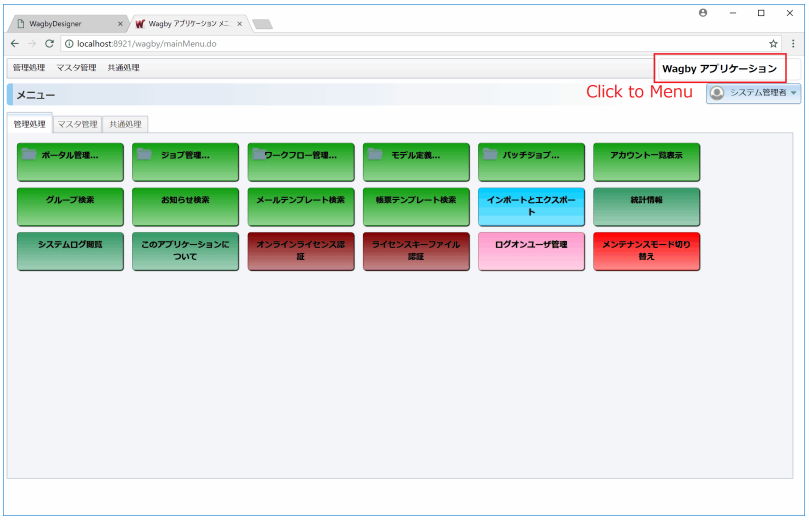


図 アプリケーション名をクリックする(1)

図 アプリケーション名をクリックする(2)

ログオフアイコン

「ログオフ」というリンクが用意されます。

サブメニュー

サブメニューの表示

ビルドしたアプリケーションに用意される「サブメニュー」の表示制御を行います。

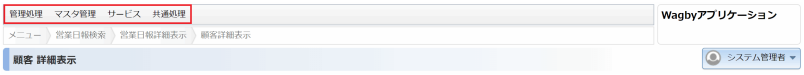


図 サブメニュー

サブメニューの表示位置

ビルドしたアプリケーションに用意される「サブメニュー」の表示位置を指定することができます。

「上」の場合

サブメニューは常に画面上部に表示されます。

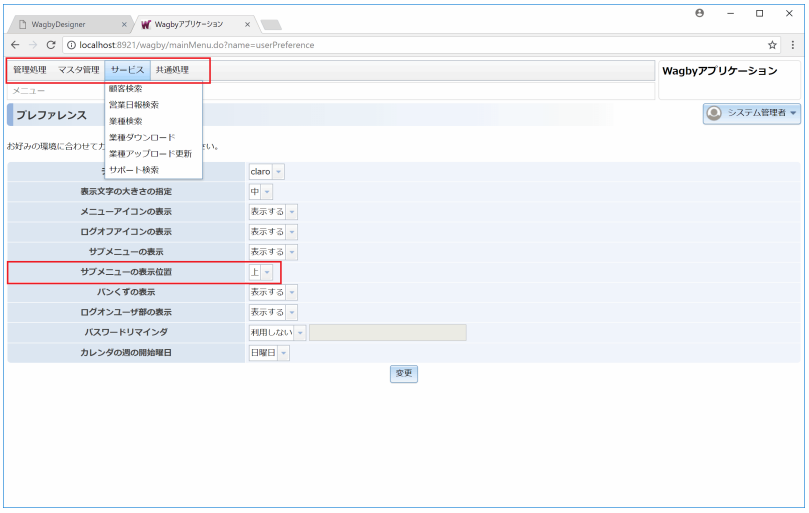


図 サブメニューの表示位置が上

「左」の場合

サブメニューはボタン押下で表示されるようになります。サブメニューの表示幅はマウス操作で変更することができます。

一度サブメニュー（左）を開くと、その後の画面遷移でサブメニューは表示され続けます。表示と非表示の状態は維持されます。

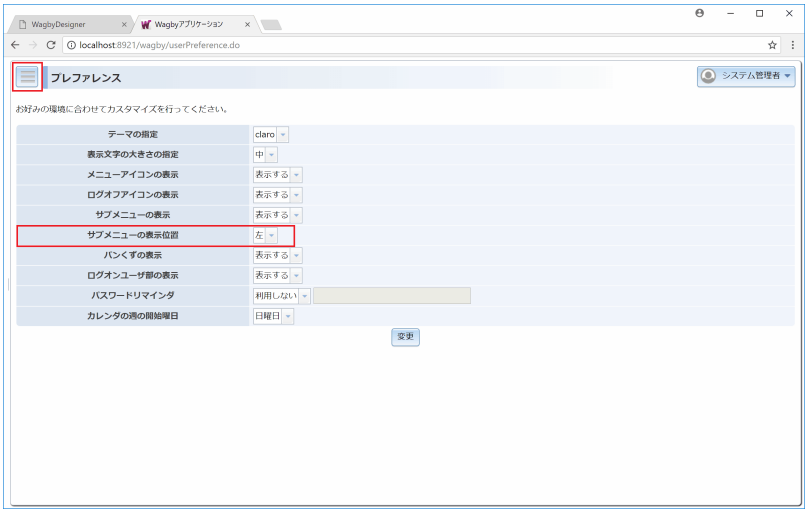


図 サブメニューの表示位置が左(1)

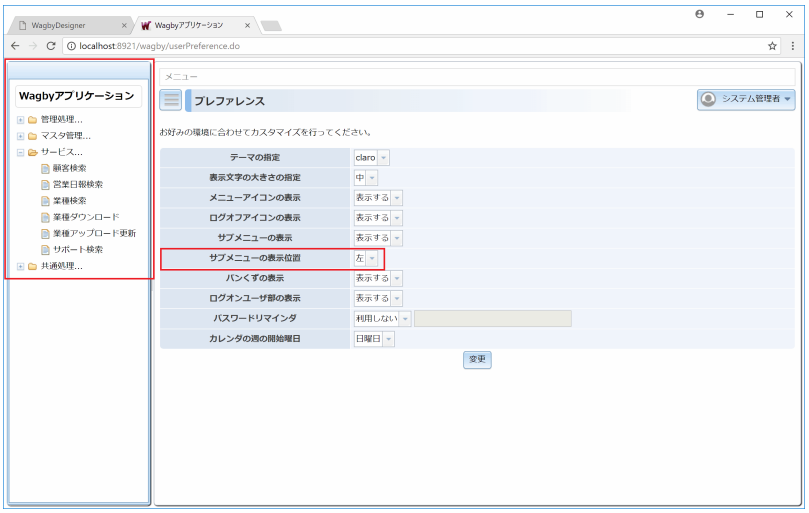


図 サブメニューの表示位置が左(2)

パンくず

「パンくず」は、画面遷移の履歴を表示します。メニューへ遷移すると、これまでのパンくず履歴は消去されます。

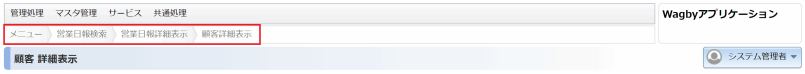


図 パンくず

ログオンユーザ

ビルドしたアプリケーションに用意される「ログオンユーザ」の表示制御を行います。

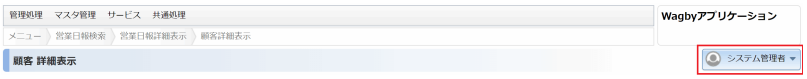


図 ログオンユーザ

パスワードリマインダ

（アカウントの章で説明します。）

カレンダの週の開始曜日

モダンカレンダビューの週の開始曜日を指定することができます。

メッセージ、ボタンアイコン

メッセージの設定

「環境」メニューから「メッセージ」を選択します。

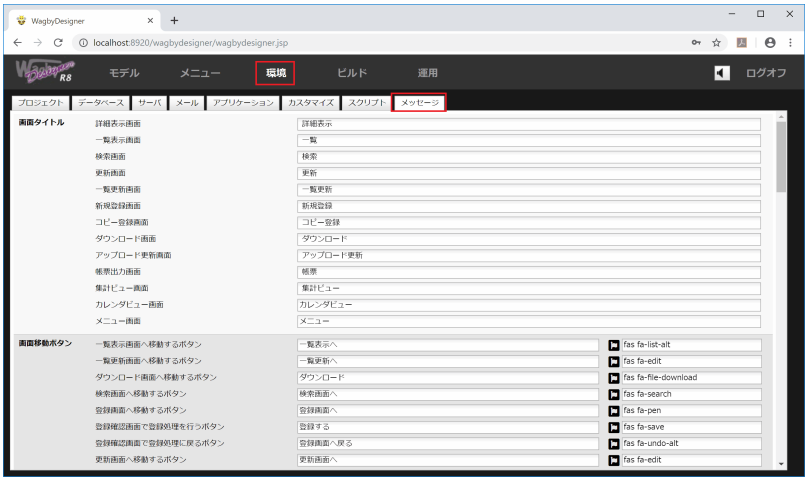


図 メッセージの設定

この文言を修正してビルドすることができます。

この修正は、アプリケーション全体に適用されます。

ボタンアイコンの設定（8.3.0）

ボタンのアイコンを変更することもできます。Wagbyのボタンアイコンは Font Awesome サイトが提供する Web アイコンフォント（Web ブラウザで文字と同じように表示できるアイコン）を利用しています。

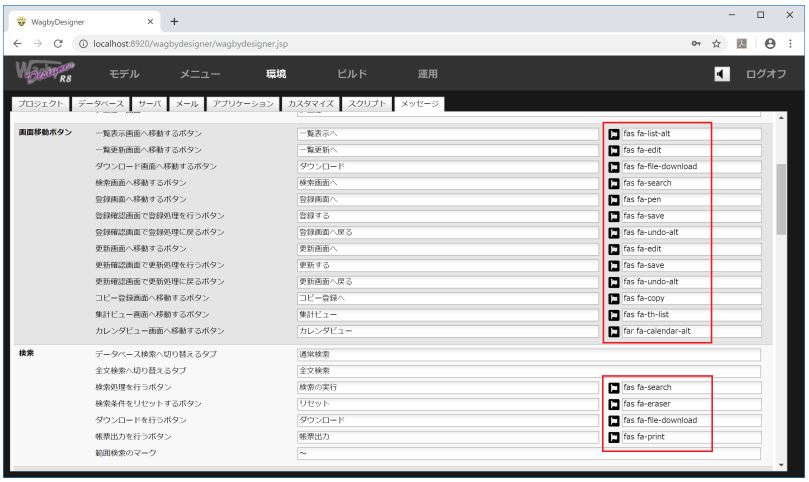


図 ボタンアイコンの指定

用意されたアイコンを押下すると、別タブで Font Awesome のサイトが開きます。この使い方は〝メニュー>アイコンの変更〟をお読みください。

ファンクションキーを含む、キーイベント動作を変更する(JavaScript)

設定方法

環境>カスタマイズタブの「キーカスタマイズ」を開きます。最大で15個のカスタマイズ設定が行えます。

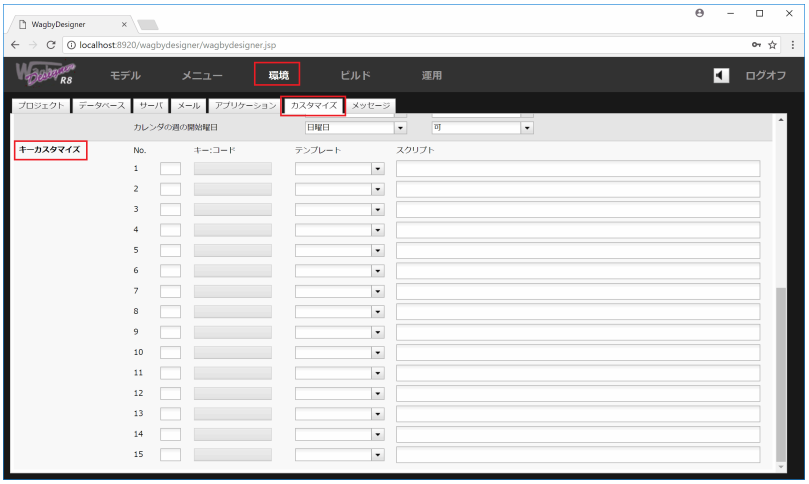


図 キーカスタマイズの設定

最初の欄（No1）のキー入力部で "F1"（ファンクションキーF1）を押下してみます。するとキーコードが自動設定されます。

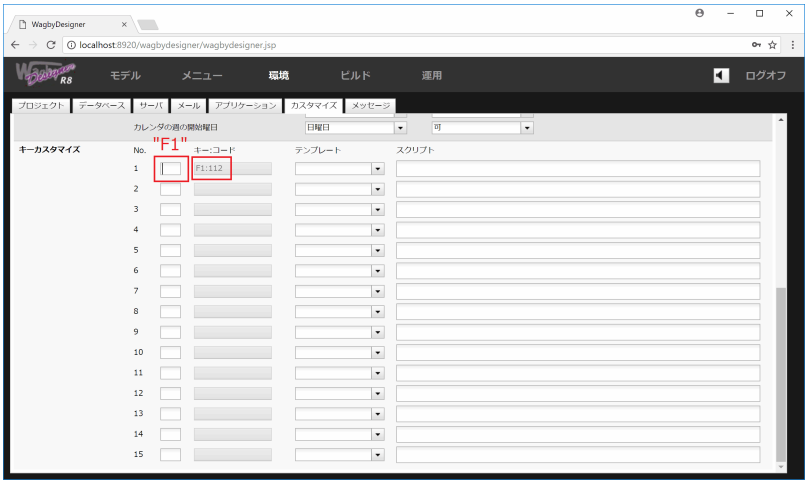


図 ファンクションキーF1を設定する

テンプレートから、希望する動作を選択することができます。ここでは「メインメニューを開く」を指定してみます。

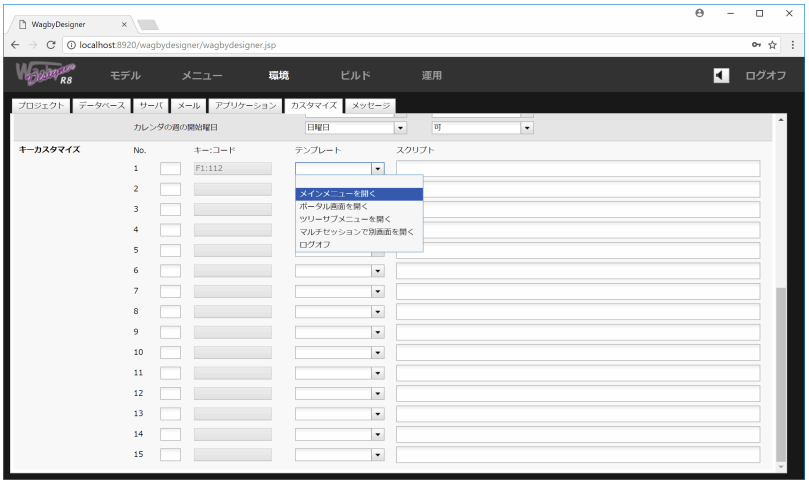


図 メインメニューへ戻る

スクリプトが自動的に設定されました。

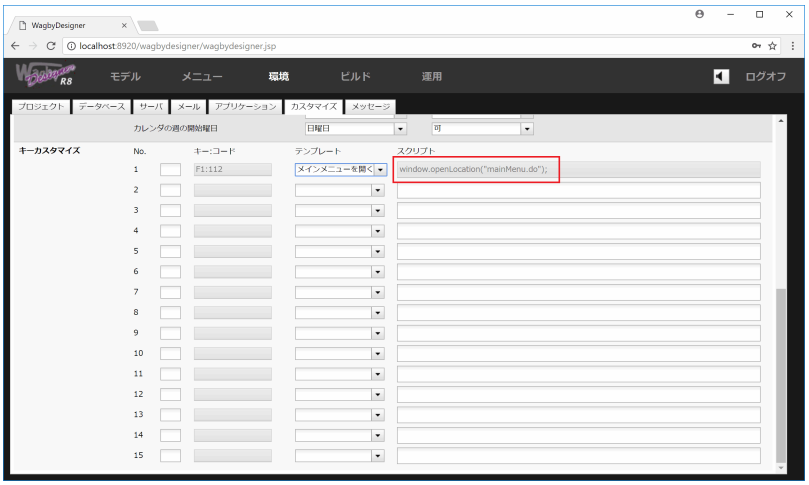


図 スクリプトが設定された

keyEventHandlers.js

この設定を行ったあとビルドを行うことで wagbyapp/webapps/(プロジェクト識別子)/system フォルダに keyEventHandlers.js というファイルが生成されます。これはブラウザ上で動作する JavaScript ファイルです。

(482)

任意のスクリプトを記述する

あらかじめ用意されたテンプレートではなく、任意のスクリプトを記述することができます。keyEventHandlers.js の構造は次のようになっています。

```
function getKeyEventHandlers(keys) {
  var keyEventHandlers = {};
  keyEventHandlers[キー] = handler_keyDown_キー;
  return keyEventHandlers;
}

// キーが押されたときに呼び出されるイベントハンドラ関数
function handler_keyDown_キー(event, dom, registry) {
  処理
}
```

Designerで記述したコードは、上の「処理」部分に転記されます。

Alt,Ctrl,Shiftキーの判定

イベントハンドラ関数に引き渡される event オブジェクトには押下されたキーのコード、シフトキー等同時に押下されたキー情報を参照するための情報が含まれています。

表記	説明
keyCode	押下されたキーのコード(数値)
altKey	Altキーが押されたときはtrueを返す。そうでないときはfalseを返す。
ctrlKey	Ctrlキーが押されたときはtrueを返す。そうでないときはfalseを返す。
shiftKey	Shiftキーが押されたときはtrueを返す。そうでないときはfalseを返す。

イベントハンドラ関数内で、次のようにして同時に押下されたキーにより処理を分けることができます。

```
if (event.altKey) {
 // Alt キー押下
} else {
 // Alt キー押下無し
}
```

特定のモデルの場合のみ有効とする

モデル model1 の画面でのみ、ファンクションキー F1 が有効になる判断を含めた例を示します。

```
function getKeyEventHandlers(keys) {
  var keyEventHandlers = {};
  keyEventHandlers[keys.F1] = handler_keyDown_F1; // メインメニューを開く
  return keyEventHandlers;
}
```

(482) ビルドしたアプリケーション（wagbyapp）に存在する、このファイルを直接編集し、ブラウザを再読み込みさせることで動作を確認することができます。

```
// Model1 の画面で F1 キーを押下した場合にメインメニューを開く
function handler_keyDown_F1(event, dom, registry, query) {

  // デバッグ用のコード。
  //console.log("event: ", event);
  //console.log("pagetitle: ", query(".pagetitle")[0]);

  // CSS クラスを指定してページタイトルの div タグを 取得する
  var pageTitle = query(".pagetitle")[0];
  // ページタイトルの div タグの id 属性に Model1 が含まれる場合、
  // メインメニューを表示する
  if (pageTitle && pageTitle.id.indexOf("Model1") !== -1) {
    window.openLocation("mainMenu.do");
  }
}
```

handler_keyDown_F1 関数に含まれる第四引数 query は指定した id 属性を含む（HTMLの）要素を返します。 このようにページタイトルに含まれる文字列を用いて判定を行うことができます。

保存ボタンを割り当てる

ファンクションキー F12 を「保存ボタン」として扱う例です。

```
function getKeyEventHandlers(keys) {
 var keyEventHandlers = {};
 keyEventHandlers[keys.F12] = handler_keyDown_F12; // 保存を行う
 return keyEventHandlers;
}

// 登録/更新/コピー登録画面で保存ボタンを押下する。
function handler_keyDown_F12(event, dom, registry) {
 var saveButton = registry.byId("btnSend");
 if (!saveButton ) {
   // ボタンが取得できなかった場合は何もしない
   return;
 }
 // ボタンのクリックイベントを実行する
 saveButton.onClick();
}
```

メインメニューを開く

ファンクションキー F1 を割り当てる例を示します。keyEventHandler.js は次のようになります。

```
function getKeyEventHandlers(keys) {
  var keyEventHandlers = {};
  keyEventHandlers[keys.F1] = handler_keyDown_F1; // メインメニューを開く
  return keyEventHandlers;
}

// メインメニューを開く
function handler_keyDown_F1(event, dom, registry) {
  window.openLocation("mainMenu.do");
}
```

- getKeyEventHandlers 関数内で、押下されたキーのコードを連想配列のキーとし、対応するイベントハンドラー関数をセットします。
- openLocation 関数に遷移先の URL を引き渡すことで画面遷移を実現します。
- Designerで設定したコードは、handler_keyDown_F1 関数の内部に反映されます。

(483)

メインメニューのサービスタブを開く

ファンクションキー F2 を割り当てる例を示します。

```
function getKeyEventHandlers(keys) {
  var keyEventHandlers = {};
  keyEventHandlers[keys.F2] = handler_keyDown_F2; // メインメニューのサービスタブを開く
  return keyEventHandlers;
}
// メインメニューのサービスタブを開く
function handler_keyDown_F2(event, dom, registry) {
  // メインメニューの画面を開く前に表示されていたメインメニュータブのIDを
  // 「サービス」のIDに変更する
  window.setMainMenuSelectedTabId("menuTab9");
  window.openLocation("mainMenu.do");
}
```

メインメニューのタブは、メインメニュー画面再表示時に以前に選択されていたタブが開くようにタブのIDをクッキーに保存しています。

メインメニュー表示時に特定のタブを選択するには、メインメニューへの遷移前に上記クッキーの値を書き換える必要があります。 具体的には setMainMenuSelectedTabId 関数に、表示するタブのIDを渡すことで、クッキーの値が更新されます。

タブのIDはタブ名の部分でマウス右クリックから「検証」を選択して表示される Developer Tools の Elements 画面で確認できます。タブ名がセットされた span タグがハイライトされますので、このタグの id 属性値(mainMenuContainer_tablist_menuTab<数字>)の"menuTab<数字>"をセットしてください。

「サービス」タブのIDが mainMenuContainer_tablist_menuTab9 の場合、タブIDは"menuTab9"を指定してください。

(484) (485)

ツリー形式のサブメニューを開く

ファンクションキー F3 を割り当てる例を示します。

```
function getKeyEventHandlers(keys) {
  var keyEventHandlers = {};
  keyEventHandlers[keys.F3] = handler_keyDown_F3; // ツリー形式のサブメニューを開く
  return keyEventHandlers;
}

// ツリー形式のサブメニューを開く
function handler_keyDown_F3(event, dom, registry) {
  // サブメニューの表示位置を「左」に設定した場合に表示されるメニュー表示
  // ボタンをID指定で取得する
  var jfcBtnMenu = registry.byId("jfcBtnMenu");
  if (!jfcBtnMenu) {
    // ボタンが取得できなかった場合は何もしない
    return;
  }
  // ボタンのクリックイベントを実行する
  // ツリー形式のサブメニューの表示・非表示が切り替わる
  jfcBtnMenu.onClick();
}
```

(483) 登録・更新画面で F1 を押下すると、遷移前に入力内容が破棄される旨の確認ダイアログが表示されます。

(484) ご注意ください：メニュー設定を変更すると、このタブのIDも変更されることがあります。その場合は新しい値をセットするようにJavaScriptを修正してください。

(485) 指定したタブIDがないい場合は、一番左側のタブが選択されます。

開発者が定義するイベントハンドラ関数には event, dom, registry の3つの引数があります。

event はキー押下時のイベントオブジェクトです。 dom, registry はそれぞれ Dojotoolkit が提供する関数です。 dom.byId("<ID>") で、指定のID値を持つ要素を取得することができます。 registry.byId("<ID>")で、指定のID値を持つボタン等が取得することができます。

サブメニューの表示位置を左に設定した場合のメニューボタンを押下するスクリプトは上のようになります。 キー押下の度にツリー形式のサブメニューの表示・非表示が切り替わります。

ポータル画面を開く

ファンクションキー F4 を割り当てる例を示します。

```
function getKeyEventHandlers(keys) {
  var keyEventHandlers = {};
  keyEventHandlers[keys.F4] = handler_keyDown_F4; // ポータル画面を開く
  return keyEventHandlers;
}

// ポータル画面を開く
function handler_keyDown_F4(event, dom, registry) {
  // メインメニューの画面を開く前に表示されていたメニュータブのIDを
  // 「ポータル」のIDに変更する
  window.setMainMenuSelectedTabId("menuTab0");
  window.openLocation("mainMenu.do");
}
```

ポータルの場合は "menuTab0" 固定となります。

モデル1の新規登録画面を開く

ファンクションキー F5 を割り当てる例を示します。

```
function getKeyEventHandlers(keys) {
  var keyEventHandlers = {};
  keyEventHandlers[keys.F5] = handler_keyDown_F5; // モデル1の新規登録画面を開く
  return keyEventHandlers;
}

// モデル1の新規登録画面を開く
function handler_keyDown_F5(event, dom, registry) {
  window.openLocation("insertModel1.do?action_New=");
}
```

モデル1の検索画面を開く

ファンクションキー F6 を割り当てる例を示します。

```
function getKeyEventHandlers(keys) {
  var keyEventHandlers = {};
  keyEventHandlers[keys.F6] = handler_keyDown_F6; // モデル1の検索画面を開く
  return keyEventHandlers;
}

// モデル1の検索画面を開く
function handler_keyDown_F6(event, dom, registry) {
  window.openLocation("mainMenu.do?name=showListModel1");
}
```

モデル1のダウンロード画面を開く

ファンクションキー F7 を割り当てる例を示します。

```
function getKeyEventHandlers(keys) {
  var keyEventHandlers = {};
  keyEventHandlers[keys.F7] = handler_keyDown_F7; // モデル1のダウンロード画面を開く
  return keyEventHandlers;
}

// モデル1のダウンロード画面を開く
function handler_keyDown_F7(event, dom, registry) {
  window.openLocation("mainMenu.do?name=downloadModel1");
}
```

マルチセッションで別画面を開く

ファンクションキー F8 を割り当てる例を示します。

```
function getKeyEventHandlers(keys) {
  var keyEventHandlers = {};
  keyEventHandlers[keys.F8] = handler_keyDown_F8; // マルチセッションで別画面を開く
  return keyEventHandlers;
}

// マルチセッションで別画面を開く
function handler_keyDown_F8(event, dom, registry) {
  // 画面右上のドロップダウンボタン内の「新規ログオン」ボタンをID指
  // 定で取得する
  var opennewtabMenuItem = registry.byId("opennewtabMenuItem");
  if (!opennewtabMenuItem) {
    // ボタンが取得できなかった場合は何もしない
    return;
  }
  // ボタンのクリックイベントを実行する
  opennewtabMenuItem.onClick();
}
```

マルチセッションを実現する「新規ログオン」ボタンは opennewtabMenuItem という ID で取得できます。

すべてのキーイベントを一つのファイルにまとめる

一つの keyEventHandler.js ファイルに、複数のキーイベントを登録できます。

```
function getKeyEventHandlers(keys) {
  var keyEventHandlers = {};
  keyEventHandlers[keys.F1] = handler_keyDown_F1; // メインメニューを開く
  keyEventHandlers[keys.F2] = handler_keyDown_F2; // メインメニューのサービスタブを開く
  keyEventHandlers[keys.F3] = handler_keyDown_F3; // ツリー形式のサブメニューを開く
  keyEventHandlers[keys.F4] = handler_keyDown_F4; // ポータル画面を開く
  keyEventHandlers[keys.F5] = handler_keyDown_F5; // モデル1の新規登録画面を開く
  keyEventHandlers[keys.F6] = handler_keyDown_F6; // モデル1の検索画面を開く
  keyEventHandlers[keys.F7] = handler_keyDown_F7; // モデル1のダウンロード画面を開く
  keyEventHandlers[keys.F8] = handler_keyDown_F8; // マルチセッションで別画面を開く
  keyEventHandlers[keys.F9] = handler_keyDown_F9; // ログオフ
  keyEventHandlers[keys.F12] = handler_keyDown_F12; // 保存ボタン
  return keyEventHandlers;
}

// メインメニューを開く
```

```
function handler_keyDown_F1(event, dom, registry) {
  window.openLocation("mainMenu.do");
}

// メインメニューのサービスタブを開く
function handler_keyDown_F2(event, dom, registry) {
  // メインメニューの画面を開く前に表示されていたメインメニュータブのIDを
  // 「サービス」のIDに変更する
  window.setMainMenuSelectedTabId("menuTab9");
  window.openLocation("mainMenu.do");
}

// ツリー形式のサブメニューを開く
function handler_keyDown_F3(event, dom, registry) {
  // サブメニューの表示位置を「左」に設定した場合に表示されるメニュー表示
  // ボタンをID指定で取得する
  var jfcBtnMenu = registry.byId("jfcBtnMenu");
  if (!jfcBtnMenu) {
    // ボタンが取得できなかった場合は何もしない
    return;
  }
  // ボタンのクリックイベントを実行する
  // ツリー形式のサブメニューの表示・非表示が切り替わる
  jfcBtnMenu.onClick();
}

// ポータル画面を開く
function handler_keyDown_F4(event, dom, registry) {
  // メインメニューの画面を開く前に表示されていたメニュータブのIDを
  // 「ポータル」のIDに変更する
  window.setMainMenuSelectedTabId("menuTab0");
  window.openLocation("mainMenu.do");
}

// モデル1の新規登録画面を開く
function handler_keyDown_F5(event, dom, registry) {
  window.openLocation("insertModel1.do?action_New=");
}

// モデル1の検索画面を開く
function handler_keyDown_F6(event, dom, registry) {
  window.openLocation("mainMenu.do?name=showListModel1");
}

// モデル1のダウンロード画面を開く
function handler_keyDown_F7(event, dom, registry) {
  window.openLocation("mainMenu.do?name=downloadModel1");
}

// マルチセッションで別画面を開く
function handler_keyDown_F8(event, dom, registry) {
  // 画面右上のドロップダウンボタン内の「新規ログオン」ボタンをID指
  // 定で取得する
  var opennewtabMenuItem = registry.byId("opennewtabMenuItem");
  if (!opennewtabMenuItem) {
    // ボタンが取得できなかった場合は何もしない
    return;
  }
  // ボタンのクリックイベントを実行する
  opennewtabMenuItem.onClick();
}

// ログオフ
function handler_keyDown_F9(event, dom, registry) {
```

```
  window.openLocation("logoff.do");
}

// 保存ボタン
function handler_keyDown_F12(event, dom, registry) {
 var saveButton = registry.byId("btnSend");
 if (!saveButton ) {
   // ボタンが取得できなかった場合は何もしない
   return;
 }
 // ボタンのクリックイベントを実行する
 saveButton.onClick();
}
```

仕様・制約

- IE では、F1 キーの挙動を変更しても、常にヘルプ画面が立ち上がります。こちらは IE の仕様となります。
- 次の文字はキーイベントに指定することはできません。

```
BACKSPACE, TAB, CLEAR, ENTER, SHIFT, CTRL, ALT, PAUSE, CAPS_LOCK, SPACE, INSERT, DELETE, SELECT,
 NUM_LOCK, SCROLL_LOCK
```

- PrtScn (PrintScreen) キーを扱うことはできません。例えば PrtScn を無効にしたい、ということには対応していません。

更新履歴

- R8.0.0 で keyEventHandlers.js を直接編集することでキーイベントを制御できるようになりました。
- R8.0.1 で Designer からスクリプトを記述できるようになりました。keyEventHandlers.js は自動生成されます。
- R8.0.2 で自動生成される関数 handler_keyDown_キー の第四引数 query が追加されます。

ログオフ

ファンクションキー F9 を割り当てる例を示します。

```
function getKeyEventHandlers(keys) {
  var keyEventHandlers = {};
  keyEventHandlers[keys.F9] = handler_keyDown_F9; // ログオフ
  return keyEventHandlers;
}

// ログオフ
function handler_keyDown_F9(event, dom, registry) {
  window.openLocation("logoff.do");
}
```

対応するデータベースを追加する(SQL)

設定ファイルdatabase.txt

開発者は repository/trunk フォルダに database.txt というファイルを用意してください。これはリポジトリに含まれるため、このプロジェクトのために利用できるデータベースという位置付けになります。

database.txtの記述例を示します。ここでは仮のデータベースとして Oracle 99z という製品を追加した、とします。

```
# oracle 99z
oracle99z.database_label=Oracle 99z
oracle99z.database_type=oracle99z
oracle99z.driver=oracle.jdbc.driver.OracleDriver
oracle99z.url=jdbc:oracle:thin:@localhost:1521:wagbydb
oracle99z.quoteid=¥"
oracle99z.quotewhere=
oracle99z.quoteidinwhere=¥"
oracle99z.quoteidforhibernate=¥"
oracle99z.hibernateDialect=org.hibernate.dialect.Oracle99zDialect
oracle99z.maxTablenameLength=30
oracle99z.maxIndexnameLength=30
oracle99z.maxColumnnameLength=30
oracle99z.maxLengthStringDataType=VARCHAR2(4000)
oracle99z.characterLargeObjectDataType=NCLOB
oracle99z.supportTimeDataType=false
oracle99z.searchStringEscapeInLike=¥¥¥¥
oracle99z.createSequenceDDL=CREATE SEQUENCE $QUOTEID$SEQNAME$QUOTEID $MINVALUE$MAXVALUESTART
 WITH $START NOCACHE CYCLE
oracle99z.sequenceDML=SELECT $QUOTEID$SEQNAME$QUOTEID.NEXTVAL FROM dual
oracle99z.dropIndexDDL=DROP INDEX $QUOTEID$INDEXNAME$QUOTEID
oracle99z.springBatchDDL=batch_oracle.ddl
oracle99z.masterDDL=master_oracle.ddl
oracle99z.validationQuery=SELECT 1 FROM dual
```

各エントリは「データベースを識別するID.設定キー=値」となっています。データベースを識別するIDは、重複しない英数字であれば任意です。ここから設定キーについて説明します。

設定キー	説明	必須	例
database_label	Designerのデータベース選択肢に表示される文字列になります。	○	Oracle 99z
database_type	データベースを識別するIDです。既存の設定と重複しない英数字です。空白を含めることはできません。この識別IDを、各エントリの接頭語として付与してください。	○	oracle99z
driver	データベースベンダーが提供するJDBCドライバのクラス名です。Javaパッケージ名も含めて記述します。大文字小文字も区別します。JDBCは4以上の規格である必要があります。	○	oracle.jdbc.driver.OracleDriver
url	Designerの接続URLの初期値として用いられます。	○	jdbc:oracle:thin:@localhost:1521
quoteid	Quote ID は、データベースへ発行するSQLのテーブル名や列名の前後に付与される文字です。利用するデータベースによって異なるため、データベースの仕様に沿って指定しま	-	¥"

	す。空白にすると Quote ID を使いません。Quote ID で囲むことによって、利用できるテーブル名や列名の名前の自由度を高めることができます。例えば、あるデータベースでは列名にハイフンを使うと SQL がエラーになりますが、Quote ID で囲むことによってハイフンを利用できるようになる、というような場合があります。詳細はデータベース設定に関する識別子の区切り文字をお読みください。		
quotewhere	Quote ID と同じ考え方ですが、WHERE 句全体に適用されるものです。	-	
quoteidinwhere	Quote ID と同じ考え方ですが、WHERE 句の内部の列に適用されるものです。通常は quoteid と同じです。	-	¥"
quoteidforhibernate	Quote ID と同じ考え方ですが、Hibernate が利用するものです。通常は quoteid と同じです。	-	¥"
hibernateDialect	Hibernate の方言 (Dialect) クラスを指定します。Wagby R8 は Hibernate 5.1 を使っています。5.1 に対応した Dialect クラスを指定してください。Hibernate 5.1 自体に含まれていない場合でも、データベースベンダが独自に提供することがあります。詳細は利用されるデータベースベンダにお問い合わせください。ここで指定したクラスファイルを含む jar ファイル (.jar) は、customize/tomcat/lib フォルダに格納しておきます。	○	org.hibernate.dialect.Oracle99zD
maxTablenameLength	利用するデータベースが許容する最大のテーブル名の長さを指定します。Wagby は、この長さを超え	○	30

	ないように調整したテーブル名を用意して、CREATE DDL 文を生成します。		
maxIndexnameLength	利用するデータベースが許容する最大のインデック名の長さを指定します。Wagbyは、この長さを超えないように調整したインデックス名を用意して、CREATE DDL 文を生成します。	○	30
maxColumnnameLength	利用するデータベースが許容する最大のカラム名（列名）の長さを指定します。Wagbyは、この長さを超えないように調整したカラム名を用意して、CREATE DDL 文を生成します。	○	30
maxLengthStringDataTyp	利用するデータベースが許容する文字列型の最大長を指定します。	○	VARCHAR2(4000)
characterLargeObjectDa	利用するデータベースが提供する Character Large Object (CLOB) の型を指定します。	-	NCLOB
supportTimeDataType	利用するデータベースが時刻型をサポートするかどうかを指定します。	-	false
searchStringEscapeInLi	SQLのWHERE句でLIKE演算子を使ったときのエスケープシーケンス文字を指定します。	○	¥¥¥¥
createSequenceDDL	利用するデータベースが順序型を提供するとき、順序を定義するための文法を指定します。[後述]	-	
sequenceDML	利用するデータベースが順序型を提供するとき、順序を利用するための文法を指定します。[後述]	-	
dropIndexDDL	利用するデータベースがインデックスを削除するときの文法を指定します。[後述]	○	
springBatchDDL	Wagby R8 は Spring Batch 3 を同梱しています。Spring Batch 3 のための管理テーブルを作成するための DDL を指定します。注意:Spring	-	

	Batch 3 がサポートするデータベースである必要があります。そうでない場合、ここは空白としてください。		
masterDDL	Wagby が提供する「選択肢モデル」の規定テーブルを生成するための DDL ファイルを指定します。ここに指定した DDL ファイルを wagbydesigner/bin/webpage/WEB-INF/export/conf/createddl フォルダに保存してください。[後述]	○	
validationQuery	データベースの自動再接続を有効にしたとき、データベースに問い合わせる接続確認のためのSQLとして利用されます。	-	SELECT 1 FROM dual

createSequenceDDLの詳細

createSequenceDDL は順序を利用できるデータベースに指定します。次の特殊文字を組み合わせて、利用するデータベースの DDL を用意します。

$QUOTEID

Quote ID を意味します。

$SEQNAME

順序名を意味します。

$MINVALUE

順序の最小値を意味します。型が整数型の場合、Integer.MIN_VALUE となります。

$MAXVALUE

順序の最大値を意味します。型が整数型の場合、Integer.MAX_VALUE となります。

$START

順序の開始数字を意味します。

例えば次のような DDL を指定することができます。

```
CREATE SEQUENCE $QUOTEID$SEQNAME$QUOTEID $MINVALUE$MAXVALUESTART WITH $START NOCACHE CYCLE
```

sequenceDMLの詳細

sequenceDML は上の特殊文字を使って、利用するデータベースが順序を発行するときの DML になります。例えば次のようになります。

```
SELECT $QUOTEID$SEQNAME$QUOTEID.NEXTVAL FROM dual
```

dropIndexDDLの詳細

dropIndexDDL はインデックスを削除する DDL を指定します。次の特殊文字を利用できます。

$QUOTEID

Quote ID を意味します。

$INDEXNAME

インデックス名を意味します。

masterDDLの詳細

masterDDL に指定した ddl ファイルは、選択肢モデルの作成時に用いられます。4つの項目 id,priority,valid,content を定義する必要があります。${tablename} はテーブル名に置換されます。例えばデータベースに Oracle を利用する場合は、次のようになります。

```
CREATE TABLE "${tablename}" (
 "id" NUMBER PRIMARY KEY,
 "priority" NUMBER,
 "valid" VARCHAR2(4000),
 "content" VARCHAR2(4000) NOT NULL
)
```

お使いのデータベースの型にあわせて、上記を修正した ddl ファイルを用意します。これを wagbydesigner/bin/webpage/WEB-INF/export/conf/createddl フォルダに保存してください。

同梱されている設定

標準で同梱されている設定は次の通りです。

ご注意ください。以下の設定にはデータベースベンダがサポートを終了した製品も含まれています。サポート終了となったデータベースは利用しないでください。

HSQLDB

```
hsqldb.database_label=HSQLDB
hsqldb.database_type=hsqldb
hsqldb.driver=org.hsqldb.jdbcDriver
hsqldb.url=
hsqldb.quoteid=¥"
hsqldb.quotewhere=
hsqldb.quoteidinwhere=¥"
hsqldb.quoteidforhibernate=¥"
hsqldb.hibernateDialect=org.hibernate.dialect.HSQLDialect
hsqldb.maxTablenameLength=65535
hsqldb.maxIndexnameLength=65535
hsqldb.maxColumnnameLength=65535
hsqldb.maxLengthStringDataType=VARCHAR(4000)
hsqldb.characterLargeObjectDataType=CLOB
hsqldb.supportTimeDataType=true
hsqldb.searchStringEscapeInLike=¥¥¥¥
hsqldb.springBatchDDL=batch_hsqldb.ddl
hsqldb.masterDDL=master_sql92.ddl
```

PostgreSQL 8.1

```
postgresql81.database_label=PostgreSQL 8.1
postgresql81.database_type=postgresql81
postgresql81.driver=org.postgresql.Driver
postgresql81.url=jdbc:postgresql://localhost/wagbydb
postgresql81.quoteid=¥"
postgresql81.quotewhere=
postgresql81.quoteidinwhere=¥"
postgresql81.quoteidforhibernate=¥"
postgresql81.hibernateDialect=org.hibernate.dialect.PostgreSQL81Dialect
postgresql81.maxTablenameLength=63
postgresql81.maxIndexnameLength=63
postgresql81.maxColumnnameLength=63
postgresql81.maxLengthStringDataType=TEXT
```

```
postgresql81.characterLargeObjectDataType=TEXT
postgresql81.supportTimeDataType=true
postgresql81.searchStringEscapeInLike=¥¥¥¥¥¥¥¥
postgresql81.createSequenceDDL=CREATE SEQUENCE $QUOTEID$SEQNAME$QUOTEID $MINVALUE$MAXVALUESTART
 $START CACHE 1 CYCLE
postgresql81.sequenceDML=select nextval('$QUOTEID$SEQNAME$QUOTEID')
postgresql81.dropIndexDDL=DROP INDEX $QUOTEID$INDEXNAME$QUOTEID
postgresql81.springBatchDDL=batch_postgresql.ddl
postgresql81.masterDDL=master_postgresql.ddl
postgresql81.validationQuery=SELECT 1
```

PostgreSQL 8.2 and later

```
postgresql82.database_label=PostgreSQL 8.2 and later
postgresql82.database_type=postgresql82
postgresql82.driver=org.postgresql.Driver
postgresql82.url=jdbc:postgresql://localhost/wagbydb
postgresql82.quoteid=¥"
postgresql82.quotewhere=
postgresql82.quoteidinwhere=¥"
postgresql82.quoteidforhibernate=¥"
postgresql82.hibernateDialect=org.hibernate.dialect.PostgreSQL82Dialect
postgresql82.maxTablenameLength=63
postgresql82.maxIndexnameLength=63
postgresql82.maxColumnnameLength=63
postgresql82.maxLengthStringDataType=TEXT
postgresql82.characterLargeObjectDataType=TEXT
postgresql82.supportTimeDataType=true
postgresql82.searchStringEscapeInLike=¥¥¥¥¥¥¥¥
postgresql82.createSequenceDDL=CREATE SEQUENCE $QUOTEID$SEQNAME$QUOTEID $MINVALUE$MAXVALUESTART
 $START CACHE 1 CYCLE
postgresql82.sequenceDML=select nextval('$QUOTEID$SEQNAME$QUOTEID')
postgresql82.dropIndexDDL=DROP INDEX $QUOTEID$INDEXNAME$QUOTEID
postgresql82.springBatchDDL=batch_postgresql.ddl
postgresql82.masterDDL=master_postgresql.ddl
postgresql82.validationQuery=SELECT 1
```

PostgreSQL 9 and later

```
postgresql9.database_label=PostgreSQL 9 and later
postgresql9.database_type=postgresql9
postgresql9.driver=org.postgresql.Driver
postgresql9.url=jdbc:postgresql://localhost/wagbydb
postgresql9.quoteid=¥"
postgresql9.quotewhere=
postgresql9.quoteidinwhere=¥"
postgresql9.quoteidforhibernate=¥"
postgresql9.hibernateDialect=org.hibernate.dialect.PostgreSQL9Dialect
postgresql9.maxTablenameLength=63
postgresql9.maxIndexnameLength=63
postgresql9.maxColumnnameLength=63
postgresql9.maxLengthStringDataType=TEXT
postgresql9.characterLargeObjectDataType=TEXT
postgresql9.supportTimeDataType=true
postgresql9.searchStringEscapeInLike=¥¥¥¥¥¥¥¥
postgresql9.createSequenceDDL=CREATE SEQUENCE $QUOTEID$SEQNAME$QUOTEID $MINVALUE$MAXVALUESTART
 $START CACHE 1 CYCLE
postgresql9.sequenceDML=select nextval('$QUOTEID$SEQNAME$QUOTEID')
postgresql9.dropIndexDDL=DROP INDEX $QUOTEID$INDEXNAME$QUOTEID
postgresql9.springBatchDDL=batch_postgresql.ddl
postgresql9.masterDDL=master_postgresql.ddl
postgresql9.validationQuery=SELECT 1
```

PostgreSQL 9.2 and later

```
postgresql92.database_label=PostgreSQL 9.2 and later
postgresql92.database_type=postgresql92
postgresql92.driver=org.postgresql.Driver
postgresql92.url=jdbc:postgresql://localhost/wagbydb
postgresql92.quoteid=¥"
postgresql92.quotewhere=
postgresql92.quoteidinwhere=¥"
postgresql92.quoteidforhibernate=¥"
postgresql92.hibernateDialect=org.hibernate.dialect.PostgreSQL92Dialect
postgresql92.maxTablenameLength=63
postgresql92.maxIndexnameLength=63
postgresql92.maxColumnnameLength=63
postgresql92.maxLengthStringDataType=TEXT
postgresql92.characterLargeObjectDataType=TEXT
postgresql92.supportTimeDataType=true
postgresql92.searchStringEscapeInLike=¥¥¥¥¥¥¥¥
postgresql92.createSequenceDDL=CREATE SEQUENCE $QUOTEID$SEQNAME$QUOTEID $MINVALUE$MAXVALUESTART
 $START CACHE 1 CYCLE
postgresql92.sequenceDML=select nextval('$QUOTEID$SEQNAME$QUOTEID')
postgresql92.dropIndexDDL=DROP INDEX $QUOTEID$INDEXNAME$QUOTEID
postgresql92.springBatchDDL=batch_postgresql.ddl
postgresql92.masterDDL=master_postgresql.ddl
postgresql92.validationQuery=SELECT 1
```

PostgreSQL 9.4 and later

```
postgresql94.database_label=PostgreSQL 9.4 and later
postgresql94.database_type=postgresql94
postgresql94.driver=org.postgresql.Driver
postgresql94.url=jdbc:postgresql://localhost/wagbydb
postgresql94.quoteid=¥"
postgresql94.quotewhere=
postgresql94.quoteidinwhere=¥"
postgresql94.quoteidforhibernate=¥"
postgresql94.hibernateDialect=org.hibernate.dialect.PostgreSQL94Dialect
postgresql94.maxTablenameLength=63
postgresql94.maxIndexnameLength=63
postgresql94.maxColumnnameLength=63
postgresql94.maxLengthStringDataType=TEXT
postgresql94.characterLargeObjectDataType=TEXT
postgresql94.supportTimeDataType=true
postgresql94.searchStringEscapeInLike=¥¥¥¥¥¥¥¥
postgresql94.createSequenceDDL=CREATE SEQUENCE $QUOTEID$SEQNAME$QUOTEID $MINVALUE$MAXVALUESTART
 $START CACHE 1 CYCLE
postgresql94.sequenceDML=select nextval('$QUOTEID$SEQNAME$QUOTEID')
postgresql94.dropIndexDDL=DROP INDEX $QUOTEID$INDEXNAME$QUOTEID
postgresql94.springBatchDDL=batch_postgresql.ddl
postgresql94.masterDDL=master_postgresql.ddl
postgresql94.validationQuery=SELECT 1
```

Oracle 9i

```
oracle9i.database_label=Oracle 9i
oracle9i.database_type=oracle9i
oracle9i.driver=jp.jasminesoft.conint.oracle.OracleDriver
oracle9i.url=jdbc:oracle:thin:@localhost:1521:wagbydb
oracle9i.quoteid=¥"
oracle9i.quotewhere=
oracle9i.quoteidinwhere=¥"
oracle9i.quoteidforhibernate=¥"
```

```
oracle9i.hibernateDialect=org.hibernate.dialect.Oracle9iDialect
oracle9i.maxTablenameLength=30
oracle9i.maxIndexnameLength=30
oracle9i.maxColumnnameLength=30
oracle9i.maxLengthStringDataType=VARCHAR2(4000)
oracle9i.characterLargeObjectDataType=NCLOB
oracle9i.supportTimeDataType=false
oracle9i.searchStringEscapeInLike=¥¥¥¥
oracle9i.createSequenceDDL=CREATE SEQUENCE $QUOTEID$SEQNAME$QUOTEID $MINVALUE$MAXVALUESTART WITH
 $START NOCACHE CYCLE
oracle9i.sequenceDML=SELECT $QUOTEID$SEQNAME$QUOTEID.NEXTVAL FROM dual
oracle9i.dropIndexDDL=DROP INDEX $QUOTEID$INDEXNAME$QUOTEID
oracle9i.springBatchDDL=batch_oracle.ddl
oracle9i.masterDDL=master_oracle.ddl
oracle9i.validationQuery=SELECT 1 FROM dual
```

(486)

Oracle 10g/11g

```
oracle10g.database_label=Oracle 10g/11g
oracle10g.database_type=oracle10g
oracle10g.driver=jp.jasminesoft.conint.oracle.OracleDriver
oracle10g.url=jdbc:oracle:thin:@localhost:1521:wagbydb
oracle10g.quoteid=¥"
oracle10g.quotewhere=
oracle10g.quoteidinwhere=¥"
oracle10g.quoteidforhibernate=¥"
oracle10g.hibernateDialect=org.hibernate.dialect.Oracle10gDialect
oracle10g.maxTablenameLength=30
oracle10g.maxIndexnameLength=30
oracle10g.maxColumnnameLength=30
oracle10g.maxLengthStringDataType=VARCHAR2(4000)
oracle10g.characterLargeObjectDataType=NCLOB
oracle10g.supportTimeDataType=false
oracle10g.searchStringEscapeInLike=¥¥¥¥
oracle10g.createSequenceDDL=CREATE SEQUENCE $QUOTEID$SEQNAME$QUOTEID $MINVALUE$MAXVALUESTART
 WITH $START NOCACHE CYCLE
oracle10g.sequenceDML=SELECT $QUOTEID$SEQNAME$QUOTEID.NEXTVAL FROM dual
oracle10g.dropIndexDDL=DROP INDEX $QUOTEID$INDEXNAME$QUOTEID
oracle10g.springBatchDDL=batch_oracle.ddl
oracle10g.masterDDL=master_oracle.ddl
oracle10g.validationQuery=SELECT 1 FROM dual
```

(487)

Oracle 12c

```
oracle12c.database_label=Oracle 12c
oracle12c.database_type=oracle12c
oracle12c.driver=jp.jasminesoft.conint.oracle.OracleDriver
oracle12c.url=jdbc:oracle:thin:@localhost:1521:wagbydb
oracle12c.quoteid=¥"
oracle12c.quotewhere=
oracle12c.quoteidinwhere=¥"
oracle12c.quoteidforhibernate=¥"
oracle12c.hibernateDialect=org.hibernate.dialect.Oracle12cDialect
oracle12c.maxTablenameLength=30
oracle12c.maxIndexnameLength=30
oracle12c.maxColumnnameLength=30
oracle12c.maxLengthStringDataType=VARCHAR2(4000)
```

(486) JDBC ドライバ jp.jasminesoft.conint.oracle.OracleDriver の説明はこちら。

(487) JDBC ドライバ jp.jasminesoft.conint.oracle.OracleDriver の説明はこちら。

```
oracle12c.characterLargeObjectDataType=NCLOB
oracle12c.supportTimeDataType=false
oracle12c.searchStringEscapeInLike=¥¥¥¥
oracle12c.createSequenceDDL=CREATE SEQUENCE $QUOTEID$SEQNAME$QUOTEID $MINVALUE$MAXVALUESTART
 WITH $START NOCACHE CYCLE
oracle12c.sequenceDML=SELECT $QUOTEID$SEQNAME$QUOTEID.NEXTVAL FROM dual
oracle12c.dropIndexDDL=DROP INDEX $QUOTEID$INDEXNAME$QUOTEID
oracle12c.springBatchDDL=batch_oracle.ddl
oracle12c.masterDDL=master_oracle.ddl
oracle12c.validationQuery=SELECT 1 FROM dual
```

(488)

MySQL

```
mysql.database_label=MySQL
mysql.database_type=mysql
mysql.driver=com.mysql.jdbc.Driver
mysql.url=jdbc:mysql://localhost/wagbydb?
useUnicode=true&relaxautoCommit=true&zeroDateTimeBehavior=convertToNull
mysql.quoteid=`
mysql.quotewhere=
mysql.quoteidinwhere=`
mysql.quoteidforhibernate=`
mysql.hibernateDialect=org.hibernate.dialect.MySQLDialect
mysql.maxTablenameLength=64
mysql.maxIndexnameLength=64
mysql.maxColumnnameLength=64
mysql.maxLengthStringDataType=TEXT
mysql.characterLargeObjectDataType=TEXT
mysql.supportTimeDataType=true
mysql.searchStringEscapeInLike=¥¥¥¥¥¥¥¥
mysql.springBatchDDL=batch_mysql.ddl
mysql.masterDDL=master_mysql.ddl
mysql.validationQuery=SELECT 1
```

MySQL 6

```
mysql6.database_label=MySQL 6
mysql6.database_type=mysql6
mysql6.driver=com.mysql.jdbc.Driver
mysql6.url=jdbc:mysql://localhost/wagbydb?
useUnicode=true&relaxautoCommit=true&zeroDateTimeBehavior=convertToNull
mysql6.quoteid=`
mysql6.quotewhere=
mysql6.quoteidinwhere=`
mysql6.quoteidforhibernate=`
mysql6.version=6
mysql6.hibernateDialect=org.hibernate.dialect.MySQLDialect
mysql6.maxTablenameLength=64
mysql6.maxIndexnameLength=64
mysql6.maxColumnnameLength=64
mysql6.maxLengthStringDataType=TEXT
mysql6.characterLargeObjectDataType=TEXT
mysql6.supportTimeDataType=true
mysql6.searchStringEscapeInLike=¥¥¥¥¥¥¥¥
mysql6.springBatchDDL=batch_mysql6.ddl
mysql6.masterDDL=master_mysql.ddl
mysql6.validationQuery=SELECT 1
```

(488) JDBC ドライバ jp.jasminesoft.conint.oracle.OracleDriver の説明はこちら。

SQLServer 2000

```
sqlserver2000.database_label=SQLServer 2000
sqlserver2000.database_type=sqlserver2000
sqlserver2000.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
sqlserver2000.url=jdbc:sqlserver://localhost;databasename=wagbydb
sqlserver2000.quoteid=¥"
sqlserver2000.quotewhere=
sqlserver2000.quoteidinwhere=¥"
sqlserver2000.quoteidforhibernate=¥"
sqlserver2000.hibernateDialect=jp.jasminesoft.jfc.hibernate.dialect.SQLServerDialect
sqlserver2000.maxTablenameLength=128
sqlserver2000.maxIndexnameLength=128
sqlserver2000.maxColumnnameLength=128
sqlserver2000.maxLengthStringDataType=nvarchar(1000)
sqlserver2000.characterLargeObjectDataType=NVARCHAR(MAX)
sqlserver2000.supportTimeDataType=false
sqlserver2000.searchStringEscapeInLike=¥¥¥¥
sqlserver2000.springBatchDDL=batch_sqlserver.ddl
sqlserver2000.masterDDL=master_sqlserver2000.ddl
sqlserver2000.validationQuery=SELECT 1
```

(489)

SQLServer 2005

```
sqlserver2005.database_label=SQLServer 2005
sqlserver2005.database_type=sqlserver2005
sqlserver2005.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
sqlserver2005.url=jdbc:sqlserver://localhost;databasename=wagbydb
sqlserver2005.quoteid=¥"
sqlserver2005.quotewhere=
sqlserver2005.quoteidinwhere=¥"
sqlserver2005.quoteidforhibernate=¥"
sqlserver2005.hibernateDialect=jp.jasminesoft.jfc.hibernate.dialect.SQLServer2005Dialect
sqlserver2005.maxTablenameLength=128
sqlserver2005.maxIndexnameLength=128
sqlserver2005.maxColumnnameLength=128
sqlserver2005.maxLengthStringDataType=nvarchar(1000)
sqlserver2005.characterLargeObjectDataType=NVARCHAR(MAX)
sqlserver2005.supportTimeDataType=false
sqlserver2005.searchStringEscapeInLike=¥¥¥¥
sqlserver2005.springBatchDDL=batch_sqlserver.ddl
sqlserver2005.masterDDL=master_sqlserver2000.ddl
sqlserver2005.validationQuery=SELECT 1
```

(490)

SQLServer 2008

```
sqlserver2008.database_label=SQLServer 2008
sqlserver2008.database_type=sqlserver2008
sqlserver2008.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
sqlserver2008.url=jdbc:sqlserver://localhost;databasename=wagbydb
sqlserver2008.quoteid=¥"
sqlserver2008.quotewhere=
sqlserver2008.quoteidinwhere=¥"
sqlserver2008.quoteidforhibernate=¥"
sqlserver2008.hibernateDialect=jp.jasminesoft.jfc.hibernate.dialect.SQLServer2008Dialect
sqlserver2008.maxTablenameLength=128
sqlserver2008.maxIndexnameLength=128
sqlserver2008.maxColumnnameLength=128
```

(489) Hibernate Dialect クラス jp.jasminesoft.jfc.hibernate.dialect.SQLServerDialect の説明はこちら。

(490) Hibernate Dialect クラス jp.jasminesoft.jfc.hibernate.dialect.SQLServer2005Dialect の説明はこちら。

```
sqlserver2008.maxLengthStringDataType=nvarchar(1000)
sqlserver2008.characterLargeObjectDataType=NVARCHAR(MAX)
sqlserver2008.supportTimeDataType=false
sqlserver2008.searchStringEscapeInLike=¥¥¥¥
sqlserver2008.springBatchDDL=batch_sqlserver.ddl
sqlserver2008.masterDDL=master_sqlserver2000.ddl
sqlserver2008.validationQuery=SELECT 1
```

(491)

SQLServer 2012/2014/2016

```
sqlserver2012.database_label=SQLServer 2012/2014/2016
sqlserver2012.database_type=sqlserver2012
sqlserver2012.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
sqlserver2012.url=jdbc:sqlserver://localhost;databasename=wagbydb
sqlserver2012.quoteid=¥"
sqlserver2012.quotewhere=
sqlserver2012.quoteidinwhere=¥"
sqlserver2012.quoteidforhibernate=¥"
sqlserver2012.hibernateDialect=jp.jasminesoft.jfc.hibernate.dialect.SQLServer2012Dialect
sqlserver2012.maxTablenameLength=128
sqlserver2012.maxIndexnameLength=128
sqlserver2012.maxColumnnameLength=128
sqlserver2012.maxLengthStringDataType=nvarchar(1000)
sqlserver2012.characterLargeObjectDataType=NVARCHAR(MAX)
sqlserver2012.supportTimeDataType=false
sqlserver2012.searchStringEscapeInLike=¥¥¥¥
sqlserver2012.createSequenceDDL=CREATE SEQUENCE $QUOTEID$SEQNAME$QUOTEID $MINVALUE$MAXVALUESTART
 WITH $START NO CACHE CYCLE
sqlserver2012.sequenceDML=SELECT NEXT VALUE FOR $QUOTEID$SEQNAME$QUOTEID
sqlserver2012.springBatchDDL=batch_sqlserver.ddl
sqlserver2012.masterDDL=master_sqlserver2000.ddl
sqlserver2012.validationQuery=SELECT 1
```

(492)

DB2

```
db2.database_label=DB2
db2.database_type=db2
db2.driver=com.ibm.db2.jcc.DB2Driver
db2.url=jdbc:db2://localhost:50000/wagbydb
db2.quoteid=¥"
db2.quotewhere=
db2.quoteidinwhere=¥"
db2.quoteidforhibernate=¥"
db2.hibernateDialect=org.hibernate.dialect.DB2Dialect
db2.maxTablenameLength=128
db2.maxIndexnameLength=18
db2.maxColumnnameLength=30
db2.maxLengthStringDataType=VARCHAR(4000)
db2.characterLargeObjectDataType=CLOB
db2.supportTimeDataType=true
db2.searchStringEscapeInLike=¥¥¥¥
db2.createSequenceDDL=CREATE SEQUENCE $QUOTEID$SEQNAME$QUOTEID $MINVALUE$MAXVALUESTART WITH
 $START NO CACHE CYCLE
db2.sequenceDML=VALUES NEXTVAL FOR $QUOTEID$SEQNAME$QUOTEID
db2.dropIndexDDL=DROP INDEX $QUOTEID$INDEXNAME$QUOTEID
db2.springBatchDDL=batch_db2.ddl
db2.masterDDL=master_db2.ddl
```

(491) Hibernate Dialect クラス jp.jasminesoft.jfc.hibernate.dialect.SQLServer2008Dialect の説明はこちら。

(492) Hibernate Dialect クラス jp.jasminesoft.jfc.hibernate.dialect.SQLServer2012Dialect の説明はこちら。

```
db2.validationQuery=SELECT 1 FROM sysibm.sysdummy1
```

DB2 AS/400

```
db2400.database_label=DB2 AS/400
db2400.database_type=db2400
db2400.driver=com.ibm.as400.access.AS400JDBCDriver
db2400.url=jdbc:as400://localhost/wagbydb;naming=system;errors=full
db2400.quoteid=¥"
db2400.quotewhere=
db2400.quoteidinwhere=¥"
db2400.quoteidforhibernate=¥"
db2400.hibernateDialect=org.hibernate.dialect.DB2400Dialect
db2400.maxTablenameLength=10
db2400.maxIndexnameLength=10
db2400.maxColumnnameLength=8
db2400.maxLengthStringDataType=VARCHAR(4000)
db2400.characterLargeObjectDataType=CLOB
db2400.supportTimeDataType=true
db2400.searchStringEscapeInLike=¥¥¥¥
db2400.createSequenceDDL=CREATE SEQUENCE $QUOTEID$SEQNAME$QUOTEID $MINVALUE$MAXVALUESTART WITH
 $START NO CACHE CYCLE
db2400.dropIndexDDL=DROP INDEX $QUOTEID$INDEXNAME$QUOTEID
db2400.masterDDL=master_db2.ddl
db2400.validationQuery=SELECT 1 FROM sysibm.sysdummy1
```

DB Master

```
dbmaster.database_label=DBMaster
dbmaster.database_type=dbmaster
dbmaster.driver=dbmaster.sql.JdbcOdbcDriver
dbmaster.url=jdbc:dbmaster:wagbydb
dbmaster.quoteid=¥"
dbmaster.quoteidinwhere=¥"
dbmaster.quoteidforhibernate=¥"
dbmaster.hibernateDialect=org.hibernate.dialect.DBMasterDialect
dbmaster.maxTablenameLength=128
dbmaster.maxIndexnameLength=128
dbmaster.maxColumnnameLength=128
dbmaster.maxLengthStringDataType=VARCHAR(3992)
dbmaster.characterLargeObjectDataType=NCLOB
dbmaster.supportTimeDataType=true
dbmaster.searchStringEscapeInLike=¥¥¥¥
dbmaster.createSequenceDDL=CREATE SEQUENCE $QUOTEID$SEQNAME$QUOTEID $MINVALUE$MAXVALUESTART
 $START CACHE 1 CYCLE
dbmaster.dropIndexDDL=DROP INDEX $QUOTEID$INDEXNAME$QUOTEID FROM $QUOTEID$TABLENAME$QUOTEID
dbmaster.masterDDL=master_dbmaster.ddl
dbmaster.validationQuery=SELECT 1
```

DB Maker

```
dbmaker.database_label=DBMaker
dbmaker.database_type=dbmaker
dbmaker.driver=dbmaker.sql.JdbcOdbcDriver
dbmaker.url=jdbc:dbmaker:wagbydb
dbmaker.quoteid=¥"
dbmaker.quoteidinwhere=¥"
dbmaker.quoteidforhibernate=¥"
dbmaker.hibernateDialect=org.hibernate.dialect.DBMakerDialect
dbmaker.maxTablenameLength=128
dbmaker.maxIndexnameLength=128
dbmaker.maxColumnnameLength=128
```

```
dbmaker.maxLengthStringDataType=VARCHAR(3992)
dbmaker.characterLargeObjectDataType=NCLOB
dbmaker.supportTimeDataType=true
dbmaker.searchStringEscapeInLike=¥¥¥¥
dbmaker.createSequenceDDL=CREATE SEQUENCE $QUOTEID$SEQNAME$QUOTEID $MINVALUE$MAXVALUESTART
 $START CACHE 1 CYCLE
dbmaker.dropIndexDDL=DROP INDEX $QUOTEID$INDEXNAME$QUOTEID FROM $QUOTEID$TABLENAME$QUOTEID
dbmaker.masterDDL=master_dbmaker.ddl
dbmaker.validationQuery=SELECT 1
```

同梱されている設定を上書きする

開発者が作成する database.txt に、標準で同梱されていた設定を再定義することができます。これによって標準の挙動を変更できます。

再定義する場合は、このデータベースに関するすべてのエントリを含めてください。（再定義しないエントリは、標準と同じ値をそのまま用います。）

ジャスミンソフトが提供するクラスファイルについて

JDBC ドライバ jp.jasminesoft.conint.oracle.OracleDriver

このファイルは Oracle が提供する JDBC ドライバクラスのラッパークラスです。connect メソッド呼び出し時に日付時刻に関するフォーマットを指定する機能を追加しています。具体的なソースコードは次のとおりです。

```
package jp.jasminesoft.conint.oracle;

import java.sql.Connection;
import java.sql.Driver;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Enumeration;
import java.util.Properties;

public class OracleDriver extends oracle.jdbc.OracleDriver {
    public Connection connect(String url, Properties info)
        throws SQLException {

        Connection con = super.connect(url, info);
        Statement s = con.createStatement();
        s.execute("ALTER SESSION SET NLS_DATE_FORMAT='YYYY-MM-DD HH24:MI:SS'");
        s.execute("ALTER SESSION SET NLS_TIME_FORMAT='HH24:MI:SS.FF3'");
        s.execute("ALTER SESSION SET NLS_TIMESTAMP_FORMAT='YYYY-MM-DD HH24:MI:SS.FF3'");
        s.close();
        return con;
    }

    static
    {
        try {
            Enumeration en = DriverManager.getDrivers();
            while(en.hasMoreElements()) {
                Object obj = en.nextElement();
                String driverName = obj.getClass().getName();
                if ("oracle.jdbc.OracleDriver".equals(driverName)
                        || "oracle.jdbc.driver.OracleDriver".equals(driverName)) {
                    DriverManager.deregisterDriver((Driver)obj);
                }
            }
```

```
            DriverManager.registerDriver(new OracleDriver());
        }
        catch(RuntimeException runtimeexception) { }
        catch(SQLException sqlexception) { }
    }
}
```

Hibernate Dialect クラス jp.jasminesoft.jfc.hibernate.dialect.SQLServerDialect

このクラスは Hibernate が提供する Dialect の継承クラスとなっており、CHARおよびVARCHARにそれぞれ nchar, nvarchar を利用するようにしたものです。SQLServerのDialectごとに用意しています。

```
package jp.jasminesoft.jfc.hibernate.dialect;

import java.sql.Types;

public class SQLServerDialect extends org.hibernate.dialect.SQLServerDialect {

    public static final int NVARCHAR_MAX_LENGTH = 4000;

    public SQLServerDialect() {
        super();
        registerColumnType(Types.CHAR, "nchar(1)");
        registerColumnType(Types.VARCHAR, "nvarchar(max)");
        registerColumnType(Types.VARCHAR, NVARCHAR_MAX_LENGTH, "nvarchar($l)");
    }
}
```

クラスタリング環境で運用する

クラスタリング環境とは

オンプレミス環境もしくはAWS や Azure などのパブリッククラウド環境で、2台(以上)のインスタンスによる「クラスタリング」運用を行うことができます。負荷分散、耐障害性の向上につなげることができます。

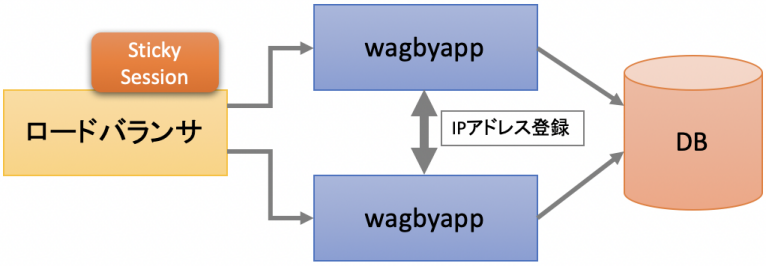


図 Wagbyのクラスタリング環境

特徴は次のとおりです。

ロードバランサは Sticky Session の設定が必要

ロードバランサでは Sticky Session を有効にしてください。利用者が最初にログオンしたインスタンスを記憶し、以降はログオフするまで同じインスタンスを使う必要があります。

セッション情報を共有しない

Wagby ではログオンする利用者ごとに「セッション」を作成します。セッションには利用者固有の情報が格納されており、ログオフするまでこれらは保持されています。

クラスタリング環境では、このセッション情報の共有を行いません。そのため、あるインスタンスがダウンしたとき、そのインスタンスの利用者は再ログオンを行って別のインスタンスに接続しなおす必要があります。

キャッシュの整合性

クラスタリング環境では、個々のインスタンスに割り当てられるIPアドレスを事前に指定します。インスタンスごとに保持しているキャッシュ情報の整合性を保つためにインスタンス同士で直接、メッセージをやりとりします。

設定方法

″環境 > サーバ > クラスタリング″ 欄を設定します。

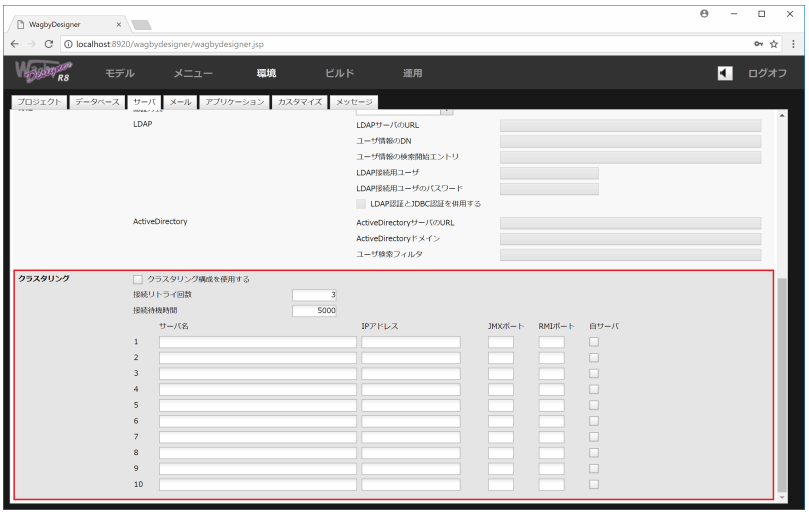


図 クラスタリングの設定

設定欄	説明
クラスタリング構成を使用する	クラスタリングを行う場合、有効にしてください。
接続リトライ回数	クラスタリングサーバ同士で通信を行なっています。その通信が失敗したときのリトライ回数の上限を指定します。リトライがこの上限に達した場合、失敗したマシンはクラスタリング対象から除外されます。（後述する「クラスタリング管理」メニューで復帰させることができます。）
接続待機時間	リトライを行うためのインターバル時間を指定します。単位はミリ秒です。
サーバ名	クラスタリングを行うマシン名（ホスト名）を記載します。重複しないようにしてください。
IPアドレス	上記マシンのIPアドレスを指定します。ドメイン名を指定することはできません。
JMXポート	1024から65535の範囲で、適当な番号を割り当ててください。RMIポートと重複することはできません。(*1)(*2)
RMIポート	1024から65535の範囲で、適当な番号を割り当ててください。JMXポートと重複することはできません。(*1)(*2)
自サーバ	ビルドしたアプリケーションが稼働するサーバを指定してください。このサーバはビルドしたアプリケーションをそのまま利用できます。このアプリケーションをコピーする2台目以降の

	サーバは手動で設定を変更する必要があります。[後述]

(493) (494)

動作の確認

3台のマシンでクラスタリングを行うテストの設定例を図に示します。

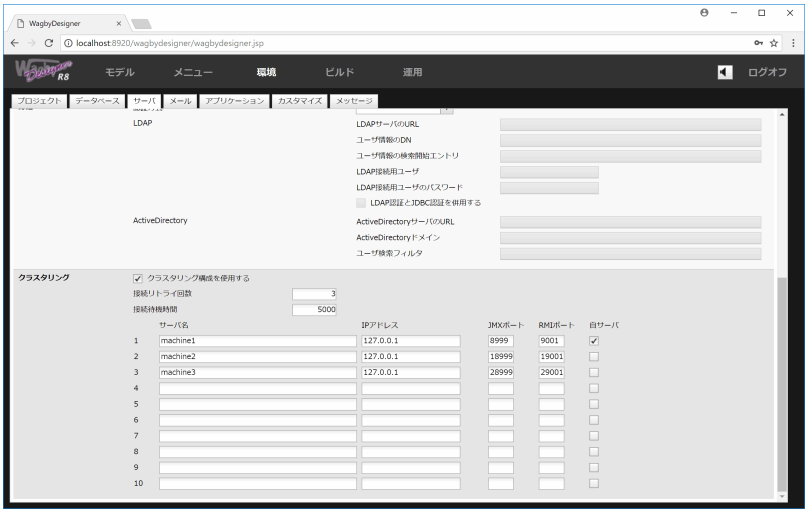


図 クラスタリングの設定例

次の手順で設定します。

1. ビルドされた wagbyappフォルダをコピーし、それぞれ wagbyapp2, wagbyapp3 とする。
2. ポート番号を変更する。wagbyapp2,wagbyapp3 の修正箇所は次の通り。

wagbyapp2

wagbyapp2/conf/server.xml

- Server/@port=8005を18005に変更
- Connector/@port=8921を18921に変更
- Connector/@port=8009を18009に変更
- Connector/@redirectPort=8443を18443に変更

wagbyapp2/webapps/wagby/WEB-INF/classes/jfccluster.properties

- cluster_list.thismachine.1=trueをfalseに変更
- cluster_list.thismachine.2=falseをtrueに変更

wagbyapp3

wagbyapp3/conf/server.xml

- Server/@port=8005を28005に変更
- Connector/@port=8921を28921に変更
- Connector/@port=8009を28009に変更
- Connector/@redirectPort=8443を28443に変更

wagbyapp3/webapps/wagby/WEB-INF/classes/jfccluster.properties

- cluster_list.thismachine.1=trueをfalseに変更
- cluster_list.thismachine.3=falseをtrueに変更

(493) 1. サーバ OS のファイアウォール機能を有効にしている場合、JMX ポート番号と RMI ポート番号の通信は 2 台のサーバ間で許可するようにして下さい。

(494) 2. すでに同一ネットワーク内で利用済みのポート番号は指定できません。どのポート番号が利用できるかの詳細は貴組織のネットワーク管理者にご相談ください。

動作テスト

1. 3つの wagbyapp を起動します。
2. 異なるブラウザ（例 Chrome,Edge,IE）にて次のURLにアクセスし、adminでログオンします。

- http://localhost:8921/wagby/
- http://localhost:18921/wagby/
- http://localhost:28921/wagby/

ロックテスト

1. Chromeにてmodel1のデータを作成し、更新画面を開く。
2. Edgeにて管理処理のロック情報管理にアクセスし、該当のデータがロックされていることを確認する。
3. Edge,IEにてmodel1の更新画面を開こうとする。
4. ロックされているというメッセージ表示される。[OK]

キャッシュテスト

1. Chromeにてmodel1のデータを変更する。
2. Edge,IEにてmodel1の詳細画面を再度表示する。
3. キャッシュが自動的にクリアされて、Chromeにて修正した値が表示される。[OK]

クラスタリング環境での運用

クラスタリングの設定を有効にすると、メニューの「管理処理」に次の機能が追加されます。

クラスタ管理

現在のクラスタ構成のマシンの状況を表示します。　自マシンや他のマシンのロックをすべて解除できます。　自マシンが稼動していることを通知することができます。　自マシンにて認識している他のマシンの稼動情報を修正できます。

図 クラスタ管理画面

ロック情報検索

現在どのマシンがどのデータをロックしているかを表示します。　ロック情報を1件づつ削除することができます。

ログオンユーザ管理

マシン名が表示されるようになります。

ロックの扱い

クラスタリング環境でロック方式を「悲観ロック」としたモデルは、悲観ロック情報をテーブルで管理するようになります。

ジョブスケジュールの制御

Wagbyが提供するジョブスケジュールは、指定時刻になったとき、起動されているすべてのインスタンスでジョブが実行されます。（いずれか一つのインスタンスだけでジョブを実行するという制御は行えません。なおオートスケール環境ではこの制御に対応しています。）

仕様・制約

- Wagbyのクラスタリングはセッション情報の共有を行いません。よって片方のインスタンスがダウンしたとき、そのサーバを利用していた方は再ログオンが必要となります。
- 最大で10台までのクラスタリング構成を行うことができます。
- クラスタリング環境で、パスワードリマインダ機能を使うことはできません。（なおオートスケール環境はパスワードリマインダに対応しています。）
- クラスタリングの利用は別途、追加ライセンスが必要になる場合があります。詳細はWagby購入先へお問い合わせください。

オートスケール環境で運用する

オートスケール環境とは

AWS や Azure などのパブリッククラウド環境が提供する「オートスケール」サービスを利用することができます。ビルドした Wagby アプリケーション（wagbyapp）が1つ以上のインスタンスで実行されるため負荷分散、耐障害性の向上につなげることができます。

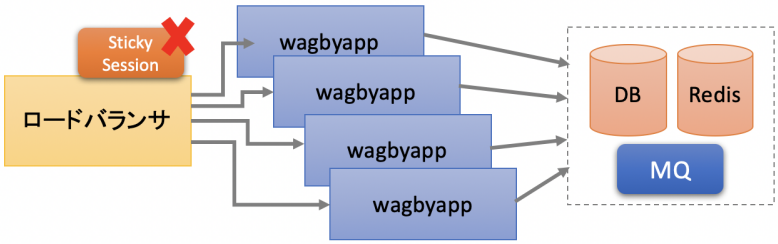


図 Wagbyのオートスケール環境

特徴は次のとおりです。

ロードバランサ

ロードバランサが必要です。なおロードバランサでは Sticky Session を有効にしません。利用者はどのwagbyappにつながってもよい状態となっています。

(495)

セッション情報の共有

Wagby ではログオンする利用者ごとに「セッション」を作成します。セッションには利用者固有の情報が格納されており、ログオフするまでこれらは保持されています。

(495) 利用者 X が、ログオン時にはインスタンス A につながり認証されたとします。Sticky Session がないため、その後のアプリケーション画面操作でインスタンス B につながるかも知れません。このような場合でも適切に動作します。

オートスケール環境では、このセッションに格納されている情報はキーバリューストアデータベースである Redis に格納されています。そのためどのインスタンスを使ってもセッション情報は維持されています。

(496)

キャッシュの整合性

オートスケール環境では、個々のインスタンスに割り当てられるIPアドレスを事前に知ることができません。そのためインスタンスごとに保持しているキャッシュ情報の整合性を保つためのメッセージ送受信はメッセージキューを使います。

設定方法

パブリッククラウドの設定

あらかじめ次のサービスを設定してください。（設定方法の詳細は割愛します。詳細はWagby代理店へお問い合わせください。）

- インスタンスのオートスケールを有効にする。AWSの場合はEC2インスタンスのオートスケール設定を使う。上限数なども指定できる。
- メッセージキューの設定。AWSの場合は Amazon MQ を使う。
- Redisの設定。[詳細...]

Designerの設定

オートスケールを有効にする

"環境 > サーバ > オートスケール" 欄に用意されている "オートスケールを有効にする" をチェックします。

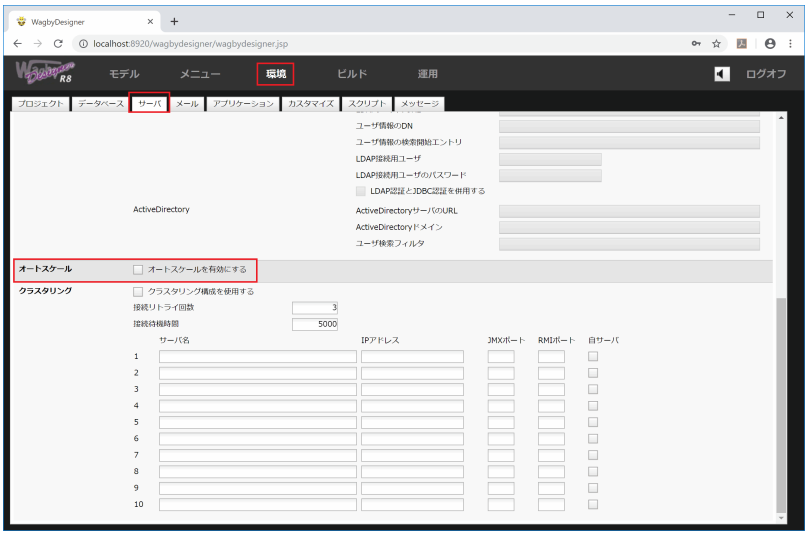


図 オートスケールを有効にする

メッセージキューの設定

"オートスケールを有効にする" を設定したとき、メッセージキューサービスを有効にする必要があります。オートスケールは ActiveMQ を使います。"環境 > アプリケーション > ActiveMQ" 欄を設定します。

(496) REST API 利用限定であれば Redis が不要なケースもあります。[詳細...]

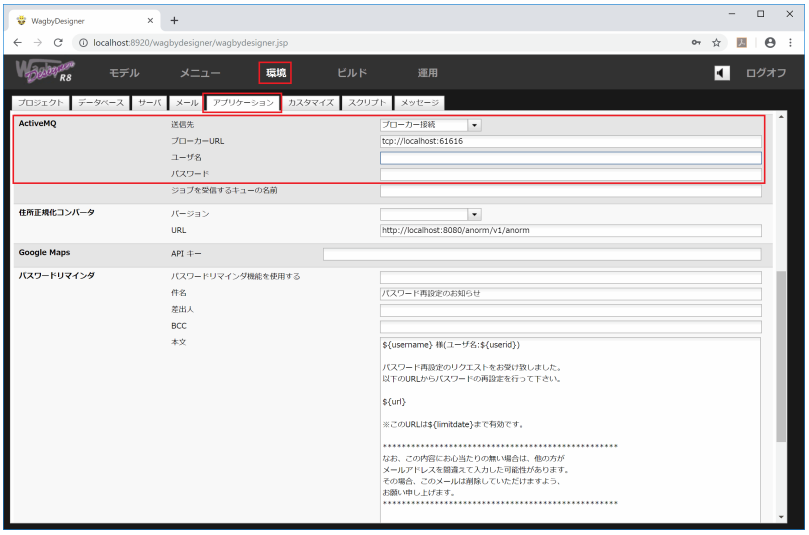


図 ActiveMQの設定

設定欄	説明	例
送信先	"ブローカー接続" を指定します。（インメモリでは動作しません）	"ブローカー接続"
ブローカーURL	接続先のメッセージキューサービスのURLを設定します。例えば AWS であれば、Amazon MQ を有効にしたとき、そのサービスにアクセスするための URL がわかります。	
ユーザ名	メッセージキューに登録したユーザ名を指定します。	
パスワード	上記ユーザのパスワードを指定します。	

(497)

HTTPセッションの設定

セッション情報の共有を行うため、HTTPPセッションに "Spring Session Redis" を指定します。同時にRedis サーバのパラメータを設定します。

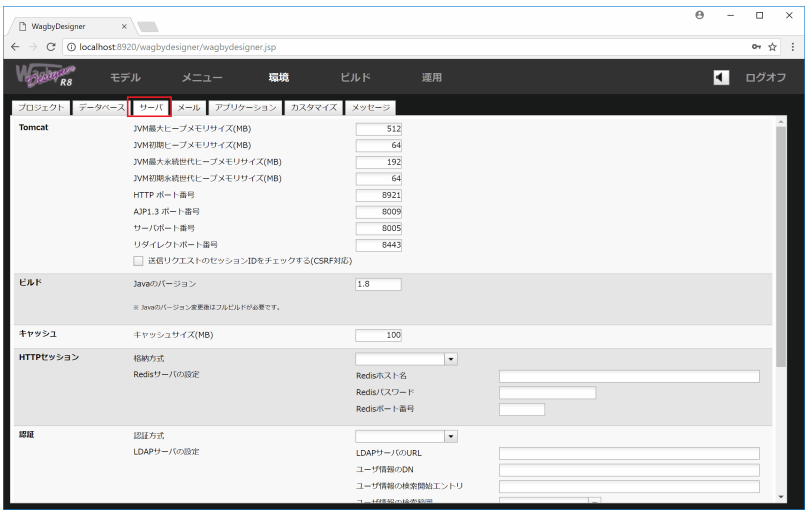


図 Redisの設定

(497) "ジョブを受信するキューの名前" は空白としてください。オートスケールのために利用するキューは自動設定されます。

動作の確認

オートスケール環境の動作テストシナリオの例を示します。

1. 事前に jgroup のデータを1件登録しておく。
2. インスタンスA にて jgroup の主キー 1000 の詳細データを取得する。(この時 A には主キー 1000 のデータのキャッシュが保持される。)
3. 同様に、インスタンスB でも jgroup の主キー 1000 の詳細データを取得する。(B にも主キー 1000 のデータのキャッシュが保持される。)
4. インスタンスA で jgroup の主キー 1000 のデータの更新処理を行う。このときキャッシュ整合性を担保するため、メッセージキューを介してAとBにキャッシュクリアを指示するメッセージが発行され、いずれのインスタンスもキャッシュがクリアされる。
5. インスタンスB で jgroup の主キー 1000 の詳細データを取得する。インスタンスAの変更結果が反映されていれば、キャッシュクリアの同期が行われていると判断できる。(キャッシュクリアの同期に失敗していた場合は、手順3と同じデータが取得されることになる。)

REST API サーバとして利用する

オートスケール環境に対応した Wagby アプリケーション (wagbyapp) を REST API サーバ専用として用いることで、サーバ部分の処理を Wagby に集約させることができます。

Redisの必要性

- リクエストヘッダに “X-Wagby-Authorization” をセットする使い方に限定する場合は、Redis は不要です。(セッションを維持する必要がないため)
- REST API 呼び出し時に、リクエストヘッダにセッションIDを付与してセッションを維持する場合は、Redis が必要です。

ロックの扱い

オートスケール環境に対応した Wagby アプリケーション (wagbyapp) で、ロック方式は「悲観ロック」「楽観ロック」いずれも利用できます。

Wagbyの標準は悲観ロック方式です。オートスケール環境では、悲観ロックはテーブルで管理する設定が自動的に有効になります。

(498)

悲観ロックテーブルは別途、データベースに用意する必要があります。[詳細...]

ジョブスケジュールの制御

オートスケール環境では、Wagbyが提供するジョブスケジュールの起動方式として次のいずれかを指定することができます。

- ″1インスタンス″ : 指定時刻になったとき、いずれか一つのインスタンスだけでジョブが実行される。(どのインスタンスかを指定することはできない。)
- ″全インスタンス″: 指定時刻になったとき、起動されているすべてのインスタンスでジョブが実行される。

前者はエクスポート処理やプログラムによるデータ操作処理など、複数のインスタンスで実行する必要がない場合に有効です。後者はメンテナンスモード切り替えなど、複数のインスタンスで起動されることが必要な場合に有効です。

設定方法

ジョブスケジュールで実行対象式を指定します。通常は ″1インスタンス″ を指定します。

(498) Wagbyが提供するシステムモデルは悲観ロック方式です。そのためオートスケール環境では開発者が用意したモデルが楽観ロックでも、システムモデルは悲観ロックが適用されます。この悲観ロックはすべてテーブルで管理されるものとなります。

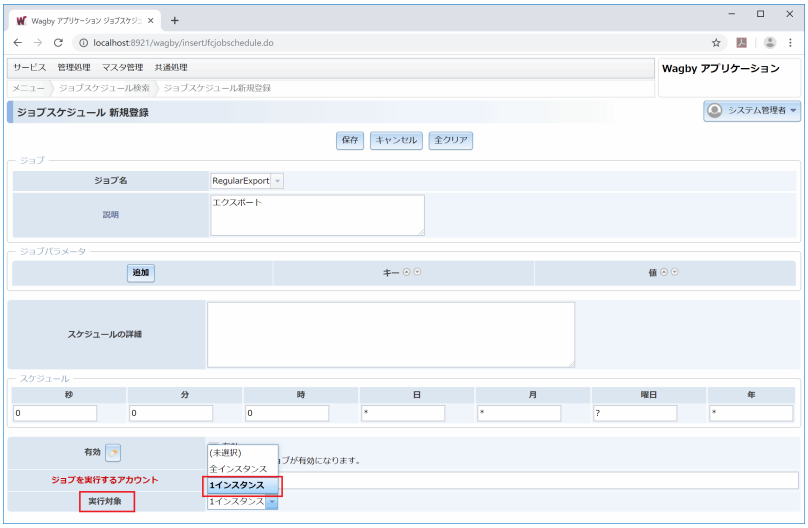


図 ジョブスケジュールで実行対象を指定する

システムログ閲覧

オートスケール環境ではログ出力は wagbyapp/logs フォルダ内の system.log 等のログファイルに出力は行わず、標準出力に出力するように変更します。これは下記の理由によるものです。

- 複数サーバで実行された際に各サーバにログファイルが保存されている場合、閲覧するには各サーバにアクセスする手間が発生します。
- 各クラウドサービスにて、標準出力されたログを一括して管理する方法があります。Dockerではfluentd(ログ管理)、AWSではAmazon CloudWatch Logs、Azureでは「Azure Container Instances でコンテナーのログとイベントを取得する」にて、ログを取得できます。

ログを標準出力するようにするため、ビルド前にカスタマイズファイルを追加します。この修正方法についてはDocker向けのカスタマイズファイルをご覧ください。（このページの後半部で、修正ファイルをダウンロードできます。）

注意:この変更によって "管理処理 > システムログ閲覧"、"管理処理 > 統計情報"、ログを監視するジョブ(AlterMailFromLog)機能は利用できなくなります。（system.log ファイルが生成されないためです。）

仕様・制約

ライセンスの制約

オートスケールはUnlimited/Corporateライセンスのみで動作します。Projectライセンスではご利用いただけません。

技術情報

- Wagbyのオートスケールに関する技術情報(PDF)

オートスケールに関連するその他の機能 (8.3.1)

- パスワードリマインダ機能にも対応しています。パスワードリマインダで発行される一時URLは、どのインスタンスに接続しても有効です。
- ログオンユーザ管理機能の拡張。オートスケール運用時でも、ログオンユーザー覧表示やセッションの強制切断機能を利用できます。
- メンテナンスモード機能の拡張。一つのノードをメンテナンスモードとしたとき、他のすべてのノードもメンテナンスモードに切り替わります。
- 全文検索インデックスを保存できるフォルダを変更できます。Docker利用時に設定しやすくすることを意図しています。

ファイル操作に関する注意点

オートスケール環境で、あるインスタンスのファイルを操作しても、他のインスタンスへ自動反映されるわけではありません。インスタンスはそれぞれ独立しており、また停止される場合もあります。停止されたインスタンス内のフォルダに保存したファイルは通常、インスタンスの停止時に消失します。これらのことから、ファイルの操作は次の点を配慮してください。

- 共有フォルダを用意し、すべてのインスタンスが同じフォルダにアクセスできるようにする。AWS であれば Amazon EFS になります。Azure であれば Storage になります。
- 共有フォルダではなく、データベースを使う。Wagbyでモデル定義を行えば、そのモデル内のデータはすべてのインスタンスで共有できます。

よくある質問と回答

オンプレミス環境でもオートスケールを試すことはできますか

可能ですが非推奨です。オートスケールに対応したパブリッククラウド環境でお試しください。

(499)

Wagbyのオートスケールでは Redis や ActiveMQ といったサービスは必須ですか

Redis は必須ではありません。[詳細...]

MQ はキャッシュ以外にもジョブやアップロード更新などで利用しており、今後も利用するケースを増やす計画があります。そのため、オートスケール環境では MQ は必須であるとお考えください。

Designerの機能・その他のツール

ビルド処理の詳細

ビルド対象モデルを指定する

モデル一覧の「ビルド」欄がチェックされているもが、ビルド対象モデルとなります。これを細かく指定することができます。

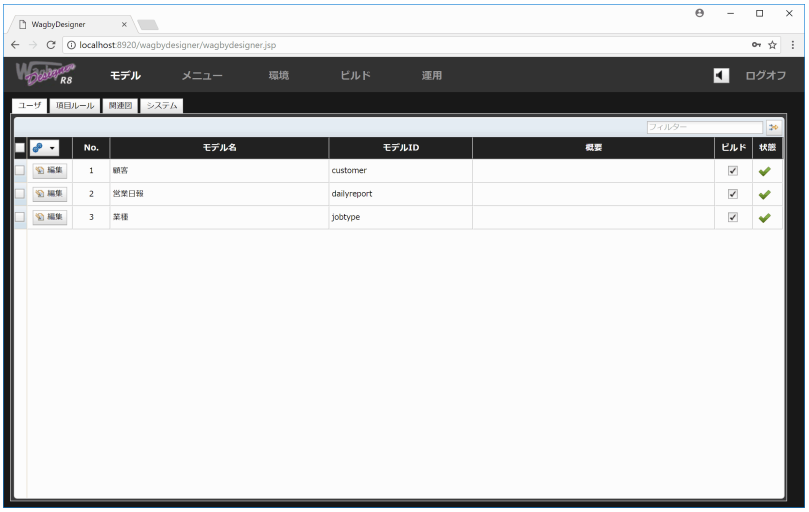


図 ビルド対象モデル

仕様・制約

(499) オンプレミスでもリソース監視と仮想環境の自動構築技術を組み合わせた環境を自前で構築することで可能です。この構築技術はインフラストラクチャに精通したエンジニアにご相談ください。Wagbyのサポート範囲外となります。

- 内部的には、モデルごとに用意されたリポジトリキーmodel/@buildの値がtrueとなっているものがビルド対象です。
- あるモデルがビルド対象のとき、このモデルのビルドのために必要な他のモデルは自動的にすべてビルドの対象になります。そのため、ビルド対象モデルが一つだけであっても、実際には複数の関連するモデルがあわせてビルドされることはあります。
- 関連するビルドを探す処理では、カスタマイズコードおよびスクリプト内部の解析は含まれません。このような場合は開発者が手動でビルド対象モデルを追加して、ビルドエラーが生じないようにしてください。
- システムタブに含まれるモデル（システムモデル）は常にビルド対象です。これを取捨選択することはできません。

コマンドラインからビルドを行う

ここでは Windows OS での利用方法を説明します。Linux OS で操作される場合はコンソールをお使いください。

「スタートメニュー>プログラムとファイルの検索」欄に、"cmd" と入力して実行します。

図 スタートメニュー

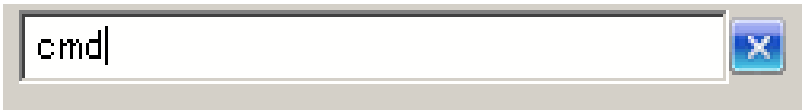


図 cmdを実行する

図に示すような、コマンドプロンプトが表示されます。

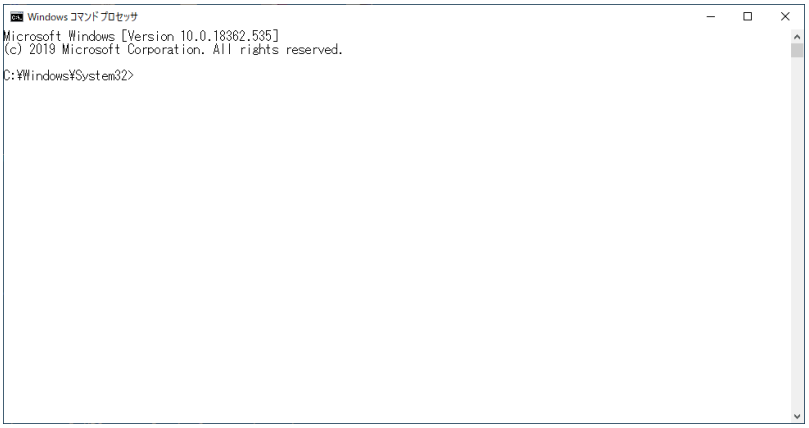


図 コマンドプロンプト

スタートメニューを経由せず、直接「コマンドプロンプト」を実行することもできます。

コマンドプロンプトが起動したら、cd コマンドでカレントフォルダを移動します。移動先は Wagbyインストールフォルダ¥misc です。（インストール先フォルダは適切に読み替えてください。）

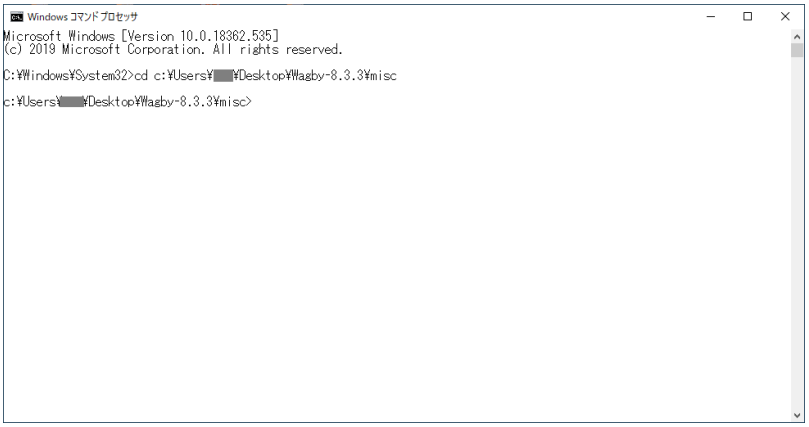


図 カレントフォルダを変える

misc フォルダに、build.xml を用意しています。misc フォルダで、次のように実行します。

```
lib/ant full
```

(500)

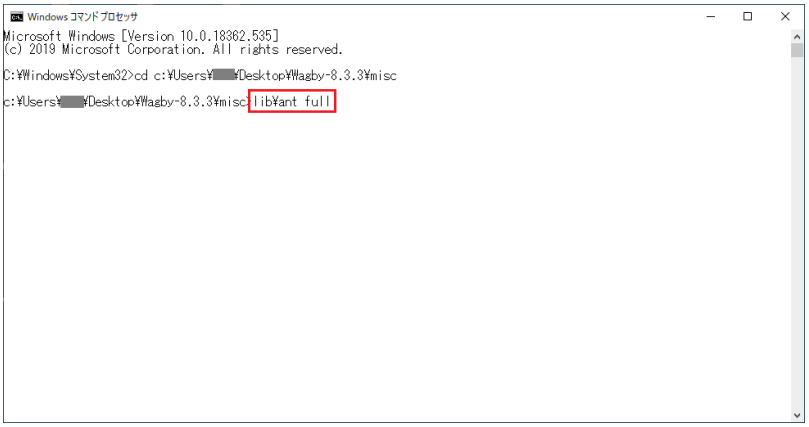


図 antから呼び出す(1)

(500) Wagby には Apache Ant というビルドツールが同梱されています。

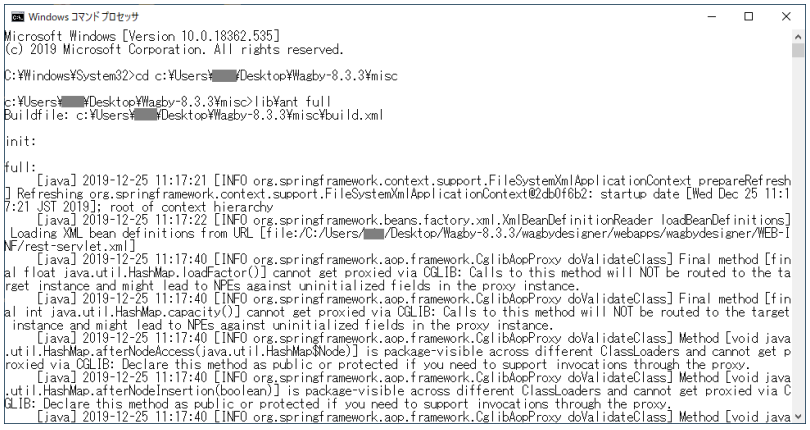


図 antから呼び出す(2)

「フルビルド」が行われます。最後に "BUILD SUCCESSFUL." と表示されれば、ビルドが終了となります。wagbyapp フォルダが再作成されています。

図 ビルド終了

この説明では Wagby に同梱している Ant を使いましたが、開発者が別途、インストール済みの Ant を使うこともできます。Ant のバージョンは 1.9 以上を利用してください。

ビルド時のメモリサイズを指定する

misc フォルダに用意されている（Apache Ant 用の）build.xml に次のように記述しています。プロジェクトの規模が大きい場合、mxmem によるメモリ指定を適切に変更してください。

```
  <target name="init">
...
    <property name="mxmem" value="4096m"/>
...
  </target>
```

現在のビルド対象モデルを表示する

```
lib/ant targetlist
```

出力は次のようになります。

```
> lib/ant targetlist
Buildfile: C:¥Wagby-8.3.4¥misc¥build.xml

init:
```

```
targetlist:
...
     [java] model/@build Enabled Model
     [java] ModelId:customer     ModelName:顧客
     [java] ModelId:sales        ModelName:売上
     [java] model/@build Disabled Model
     [java] none
     [java] Enable/disablemodel system properties: -Denablemodel=customer,sales -Ddisablemodel=

BUILD SUCCESSFUL
Total time: 9 seconds
```

見出し文字列 "model/@build Enabled Model" に続いて、ビルド対象モデルが表示されます。 同じく見出し文字列 "model/@build Disabled Model" に続いて、ビルド対象ではないモデルが表示されます。

"none" という表記は、対象モデルがないことを示します。上の例では、ビルド対象ではないモデルは存在しない（つまり、すべてビルド対象である）ことを示しています。

ビルド対象モデルを指定する (1)

```
lib/ant full -Denablemodel=[モデルID、コンマ区切りで複数指定可] -Dbuild=false
```

例えば customer モデルだけをビルドする場合、次のように入力します。

```
> lib/ant full -Denablemodel=customer -Dbuild=false
```

このとき、内部では次のような動作となります。

- -Ddisablemodel=* という指定が暗黙的に加わります。これによって、すべてのモデルのmodel/@buildの値をfalseに書き換えます。
- あわせて、-Denablemodel に含まれるモデルのmodel/@buildの値だけをtrueにします。

ビルド処理を行わない

-Dbuild=false 指定により、model/@buildの値を書き換えるのみで、ビルド処理は行われません。この設定を省略すると、ビルド対象モデル変更後、すぐにフルビルドが行われます。

ビルド対象モデルを指定する (2)

上の方法は -Denablemodel で指定されたモデル以外は自動的にビルド対象外となっていました。そうではなく、現在の状態を維持したまま、ビルド対象モデルを追加する場合は次のようにします。

```
lib/ant full -Denablemodel=[モデルID、コンマ区切りで複数指定可] -Ddisablemodel= -Dbuild=false
```

例えば現在の状態を維持しながら customer モデルをビルド対象に加える合、次のように入力します。

```
> lib/ant full -Denablemodel=customer -Ddisablemodel= -Dbuild=false
```

-Ddisablemodel= という指定を明示することで、model/@buildの値をfalseに書き換える処理がスキップされます。

すべてのモデルをビルド対象にする

次のようにします。

```
> lib/ant full -Denablemodel=* -Ddisablemodel= -Dbuild=false
```

ビルド対象にしないモデルを指定する (1)

```
lib/ant full -Ddisablemodel=[モデルID、コンマ区切りで複数指定可]
```

例えば現在の状態から customer モデルだけをビルド対象から除く場合、次のように入力します。

```
> lib/ant full -Ddisablemodel=customer
```

このとき、内部では次のような動作となります。

・ -Denablemodel= という指定が暗黙的に加わります。これによって、指定されていないモデルのmodel/@buildは維持されます。
・ -Ddisablemodel に含まれるモデルのmodel/@buildの値だけをfalseにします。

ビルド対象にしないモデルを指定する (2)

上の方法は -Ddisablemodel で指定されたモデル以外で、ビルド対象となっていたモデルは維持されていました。そうではなく、-Ddisablemodel で指定したモデル以外は、すべてビルド対象とする場合は次のようにします。

```
lib/ant full -Ddisablemodel=[モデルID、コンマ区切りで複数指定可] -Denablemodel=*
```

例えば customer モデルだけをビルド対象から除き、それ以外のモデルをすべてビルド対象とする場合、次のように入力します。

```
> lib/ant full -Ddisablemodel=customer -Ddisablemodel=*
```

-Ddisablemodel= という指定を明示することで、model/@buildの値をfalseに書き換える処理がスキップされます。

仕様・制約

コマンドライン操作によるビルド処理はフルビルドのみ対応しています。差分ビルドを行うことはできません。（差分ビルドは、差分情報を Designer が管理しているため、Designer による操作が必要です。）

リモートのDesignerのリポジトリを転送する (8.3.4) (8.4.0)

2台のPCにWagby(Designer)がインストールされているとします。このとき、開発機 A で起動している Designer のリポジトリを開発機 B に転送することができます。

(501)

開発機 B は Designer を起動する必要はありません。misc フォルダからコマンドラインで操作します。

コマンド	内容	対応
lib/ant clone	A の最新リポジトリを B に展開する標準的な方法。backup, get のあと、(A に作成された) バックアップファイルを削除する。	R8.4.0
lib/ant backup	A の repository/trunk を zip 形式とする。A の repository/branches にバックアップファイル (wrb_XX.zip) が作成される。	R8.4.0
lib/ant get	A にある repository/branches にある最新のバックアップファイルを転送し、B の repository/trunk に展開する。	R8.3.4
lib/ant backup get	A にある repository/trunk を zip 形式としたあと、このファイル（最新のバックアップファイル）を転送し、B の repository/trunk に展開する。	R8.4.0
lib/ant getzip	B の repository/branches にある任意のバックアップファイルを B の repository/trunk に展開する。	R8.3.4

(501) 開発機 A, B いずれも R8.3.4 以上の Wagby である必要があります。

各コマンドの詳細を説明します。

ant backup (8.4.0)

書式

```
lib/ant backup -DrepoServerHost=開発機Aのホスト名:ポート番号 -DrepoServerUsername=アカウント -DrepoServerPassword=パスワード -Dmemo=バックアップ時のメモ
```

(502)

例

開発機 A のホスト名が hostA で、標準のポート番号 8920 で起動している Wagby Designer の現在のリポジトリを zip 化する場合、次のようにします。

```
> lib/ant backup -DrepoServerHost=hostA:8920
```

ant get (8.3.4)

書式

```
lib/ant get -DrepoServerHost=開発機Aのホスト名:ポート番号 -DrepoServerUsername=アカウント -DrepoServerPassword=パスワード
```

(503)

例

開発機 A のホスト名が hostA で、標準のポート番号 8920 で起動している Wagby Designer からリポジトリを転送する場合、次のようにします。

```
> lib/ant get -DrepoServerHost=hostA:8920
```

(504)

その他のオプション

- -DdownloadOnly=trueを指定すると、ダウンロードしたファイルをrepository/branchesにダウンロードする処理のみを行って、終了します。repository/trunkへの展開は行いません。
- -DnoMigration=trueを指定すると、移行処理は行わず、転送したファイルをそのままrepository/trunkに展開します。

ant getzip (8.3.4)

リモート/ローカルという関係ではなく、1つの(ローカルの) Wagby Designer で、branches にあるリポジトリファイルを trunk に展開します。言い換えると、Designerのバックアップリポジトリ読み込み操作をコマンドラインで行います。

書式

```
lib/ant getzip -Drepofile=../repository/branches/バックアップファイル.zip
```

例

コマンドライン操作で初期リポジトリに戻す場合、次のようにします。(Wagby Designer は起動しているものとします。)

```
> lib/ant getzip -Drepofile=../repository/branches/wrb_20180130_0000001.zip
```

ant clone (8.4.0)

---

(502) -DrepoServerUsernameと-DrepoServerPasswordおよび-Dmemoは省略できます。repoServerUsernameは省略するとuserとなります。repoServerPasswordは省略するとwagbyとなります。-DrepoServerHostを省略することはできません。

(503) -DrepoServerUsernameと-DrepoServerPasswordは省略できます。repoServerUsernameは省略するとuserとなります。repoServerPasswordは省略するとwagbyとなります。-DrepoServerHostを省略することはできません。

(504) 開発機 A の Wagby Designer で管理しているリポジトリが旧バージョンの場合、転送のタイミングで移行処理が動作します。これは通常、Wagby Designer でリポジトリを読み込むと自動的に移行処理が動作するのと同じです。

リモートマシンの最新のリポジトリのバックアップと転送(取得)、リモートマシンで取得したバックアップの削除、ローカルマシンでのビルドを1回のコマンドで行います。

書式

```
lib/ant clone -DrepoServerHost=hostA:8920
```

その他のオプション

ant get と同じオプションを利用できます。

(505)

ビルドタブを非表示にする (8.4.0)

Designer 画面の上部に用意された「ビルド」タブを非表示にすることができます。これにより、その Designer ではブラウザからの操作によるビルド処理が行えなくなります。

設定方法

1. repository/trunk 直下の設定ファイル wdsettings.txt をテキストエディタで開きます。
2. 次の行を増やします。

```
build.disable=true
```

3. "運用>リポジトリ"のリポジトリ再読み込みボタンを押下し、wdsettings.txt の再読み込みを行います。今回の設定内容が画面に反映されます。

フィルタ機能

操作例

モデル一覧表示画面の右上に「フィルター」という欄が用意されています。

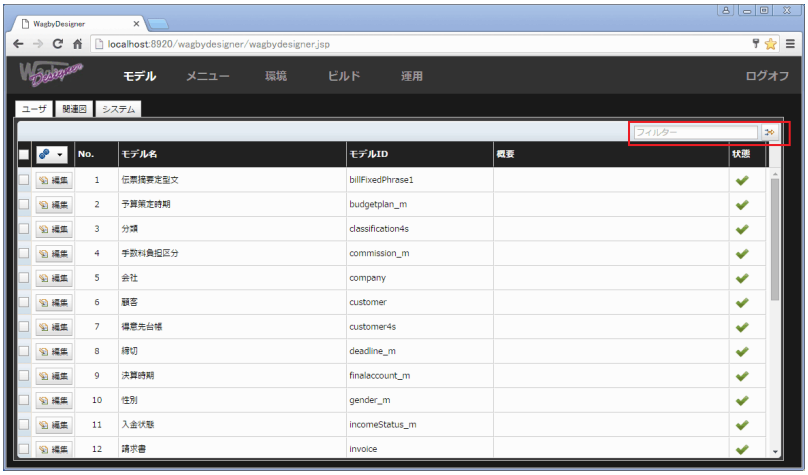


図 フィルター入力欄

モデルIDで絞り込む

モデルIDで絞り込むことができます。図は "cu" と入力した例です。モデルIDに"cu"を含むものだけが表示されます。

(505) いずれのコマンドも "full" と組み合わせることができます。例えば lib/ant clone full とすることで、取得とフルビルドを一緒に行うことができます。

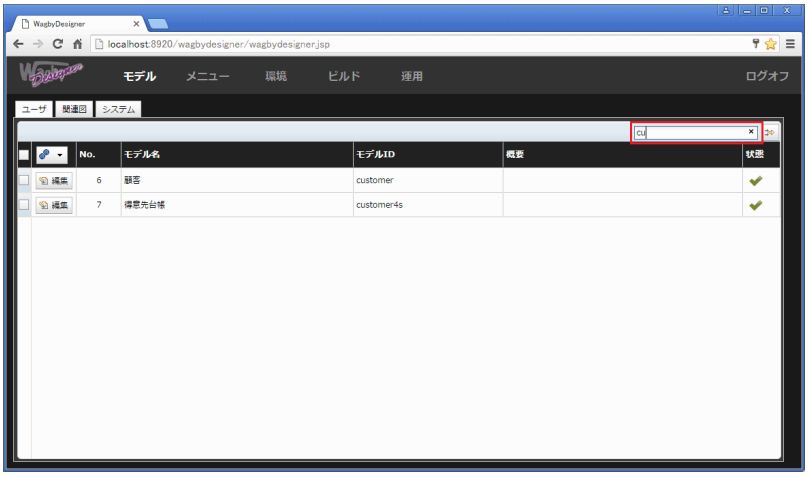


図 モデルIDで絞り込んだ例

モデル名で絞り込む

モデル名で絞り込むこともできます。図は ″書″ と入力した例です。モデル名に″書″を含むものだけが表示されます。

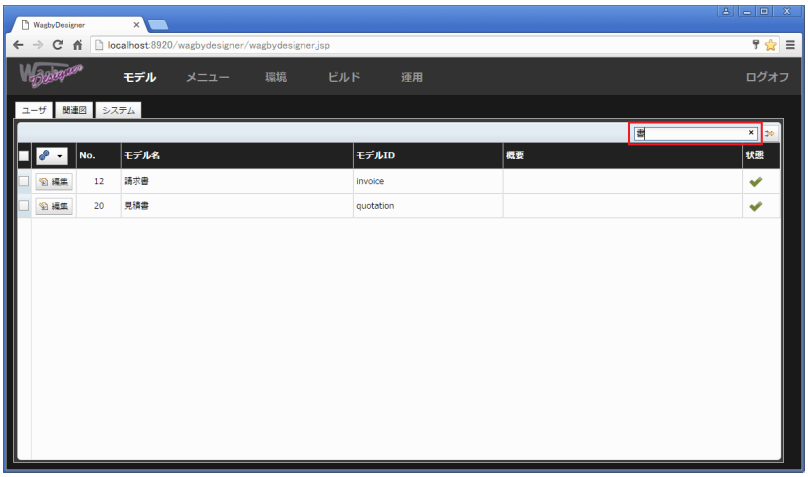


図 モデル名で絞り込んだ例

仕様・制約

仕様

- フィルター欄にはモデルID/モデル名のいずれを入力することもできます。
- 入力した文字を含んだものが選択されます。（部分一致検索となります。）

制約

別のプロジェクトに切り替えるためにリストアを行う場合、フィルタに入力していた文字は事前に消去しておいてください。文字が残ったままリストアを行うと、モデル一覧の表示が乱れることがあります。（フィルタ文字を消去することで、乱れは解消されます。）

モデル項目の一括設定

基本的な操作

ここでは「会社」モデルを例に、一括設定の基本操作を説明します。

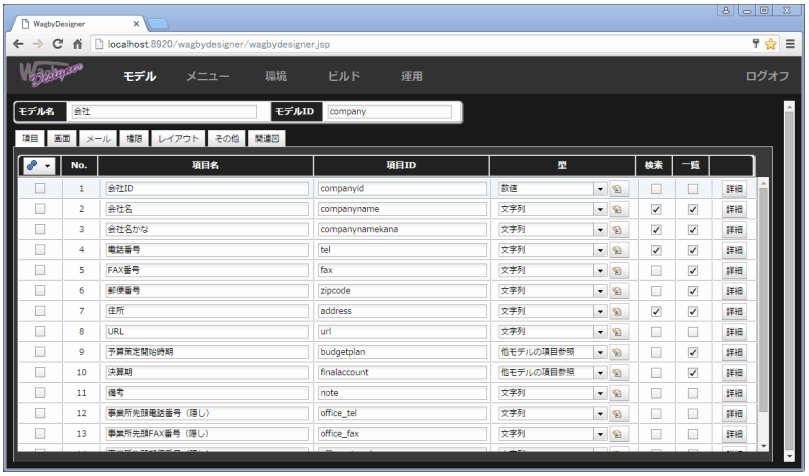


図 サンプル：会社モデル

設定するリポジトリキーを指定する

任意の項目から、モデル項目詳細定義のダイアログを開きます。図では"会社名"項目を使っています。 次に、一括設定したい「名称」部分をマウスで選択します。

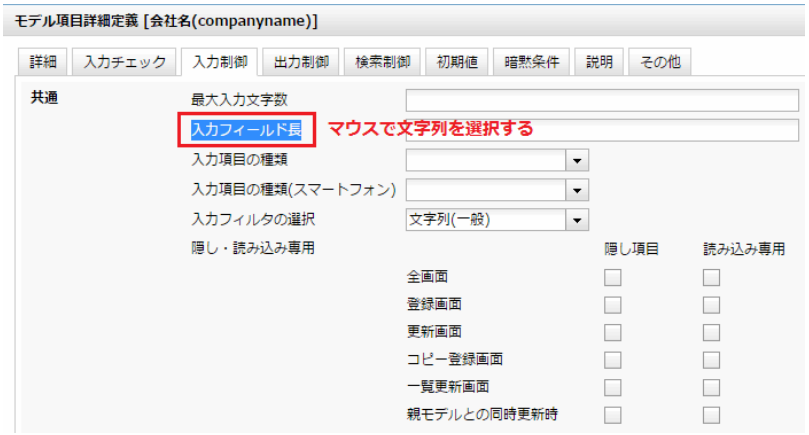


図 一括設定の対象を選択する

一括設定の対象を選択した状態で、ダイアログ下部に用意された「一括設定」ボタンを押下します。

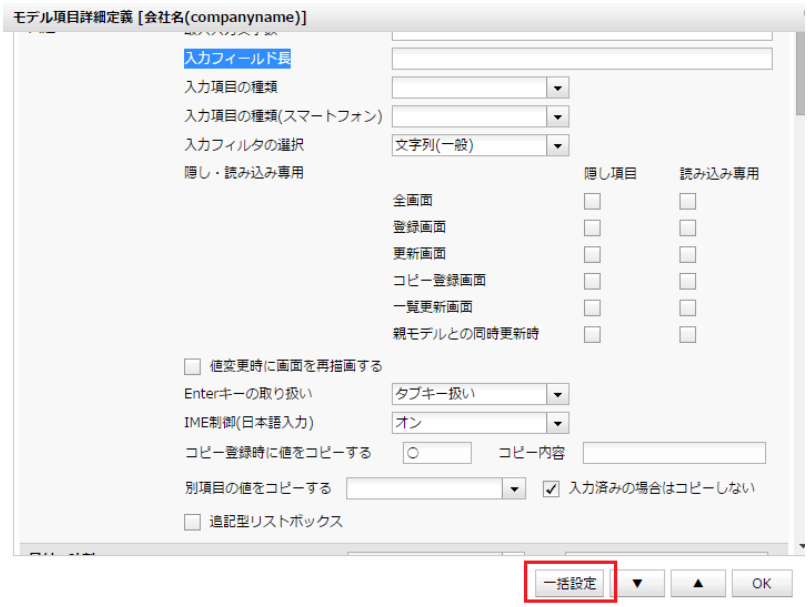


図 一括設定ボタン

図のような一括設定ダイアログが開きます。ここでリポジトリキーの部分は、先に選択した文字列を含む設定が用意されています。複数の候補が表示されることもありますので、適切なものを選択してください。

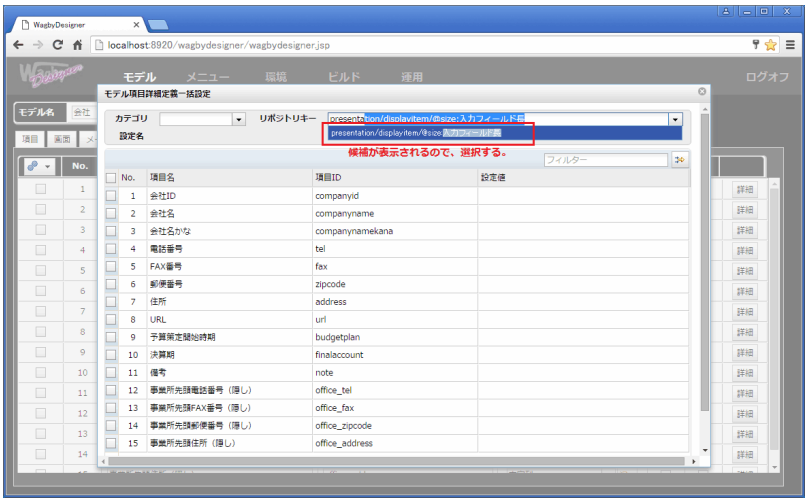


図 一括設定ダイアログ

値を入力する

項目ごとに、値を入力することができます。入力後、「保存」ボタンを押下します。

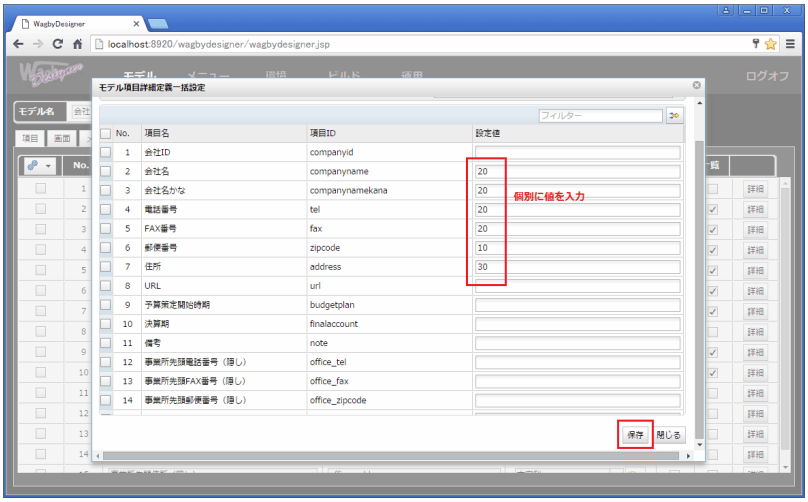


図 値を入力し、保存ボタンを押下する

設定確認ダイアログが表示されます。「はい」を押すと設定が反映されます。

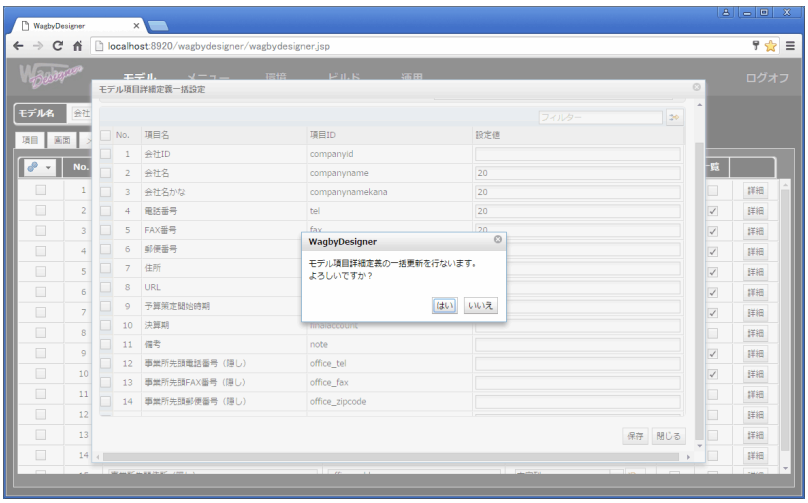


図 設定確認ダイアログ

設定が終了しました。

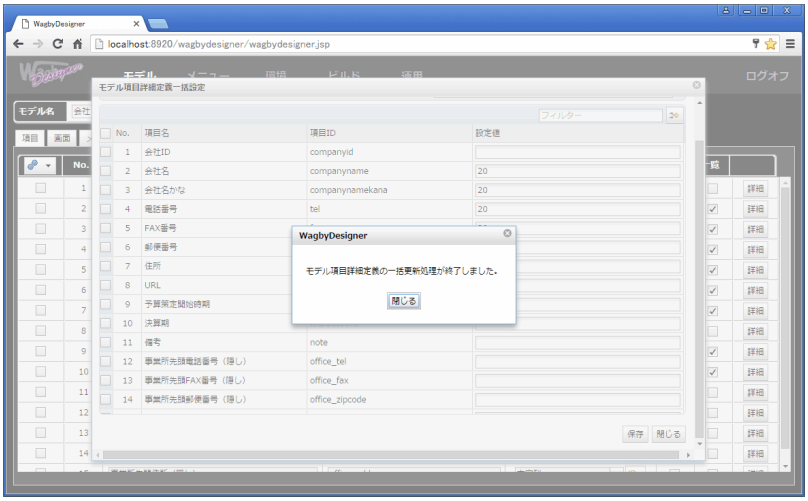


図 設定が終了した

「閉じる」ボタンを押下してダイアログを閉じます。閉じるボタンを押さずに、一括設定の操作を継続することもできます。

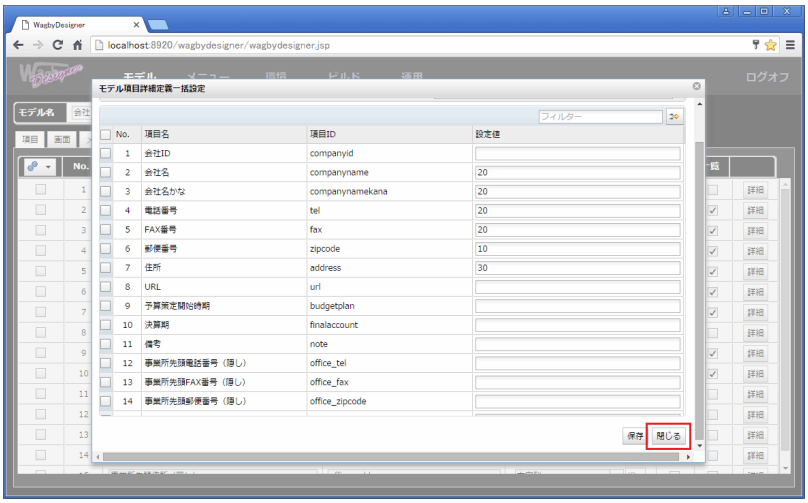


図 一括設定ダイアログを閉じる

全項目の一括設定

項目名の上部に用意されたチェックボックスを使うと、全項目を選択することができます。

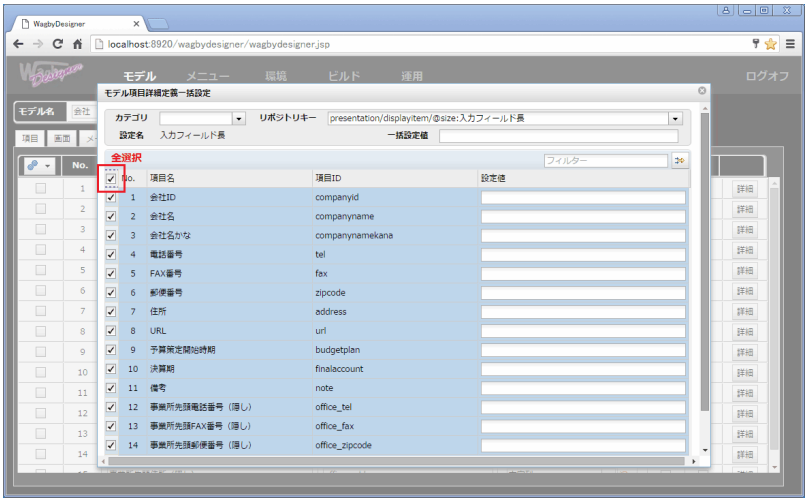


図 全項目を選択する

この状態で「一括設定値」の欄に値を入力することができます。

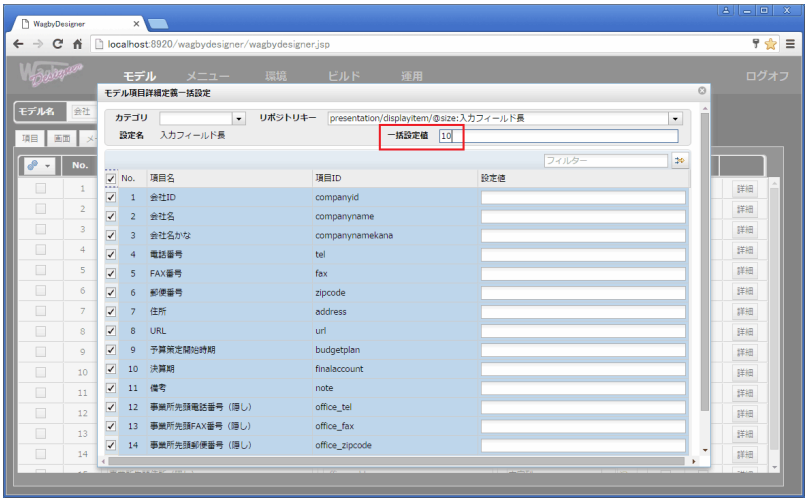


図 一括設定値を入力する

チェック状態にあるすべての項目に、値が一斉にセットされます。保存ボタンを押す前に、個別に値を変更することもできます。

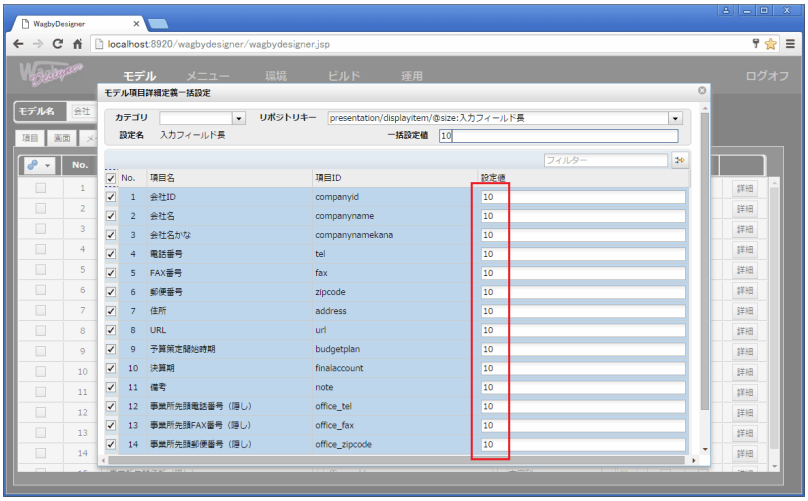


図 項目に値がセットされた

選択形式の場合

選択形式となっているリポジトリの場合、定義方法が変わります。 ここでは「必須」指定のリポジトリを操作する例を紹介します。

図 入力チェック>必須指定

一括設定値は、図に示すようにリストボックス形式となっています。

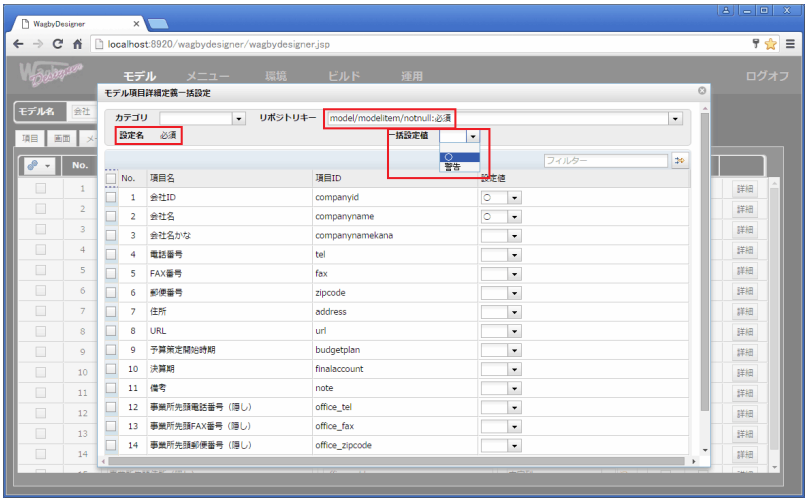


図 リストボックスによる選択

対象項目をチェックした状態で、選択肢から値を選ぶと、設定値がセットされます。

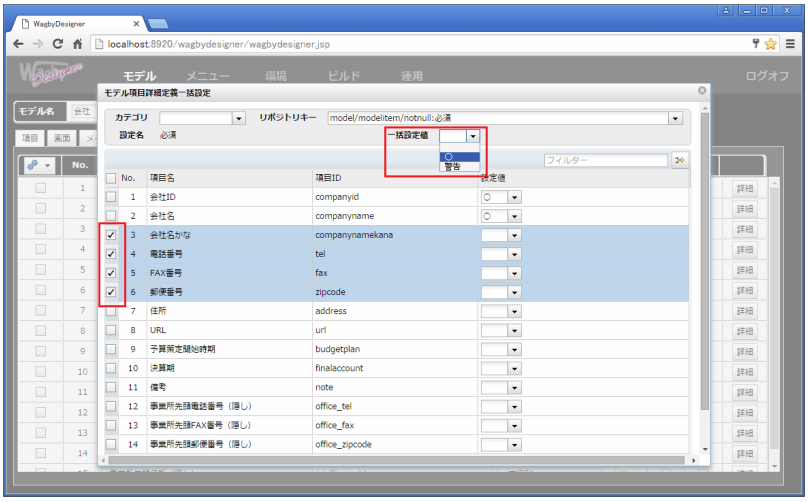


図 対象項目をチェックする

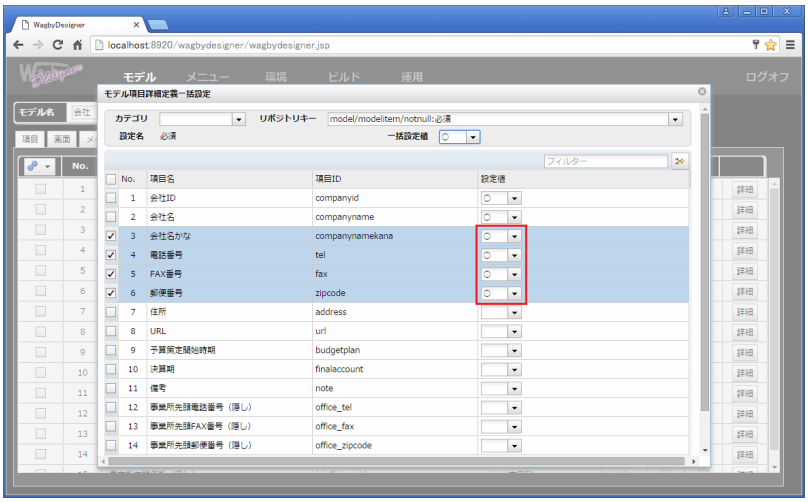


図 設定値がセットされた

リポジトリキーを直接指定する

リポジトリキーを直接、指定することもできます。はじめに、カテゴリを指定します。

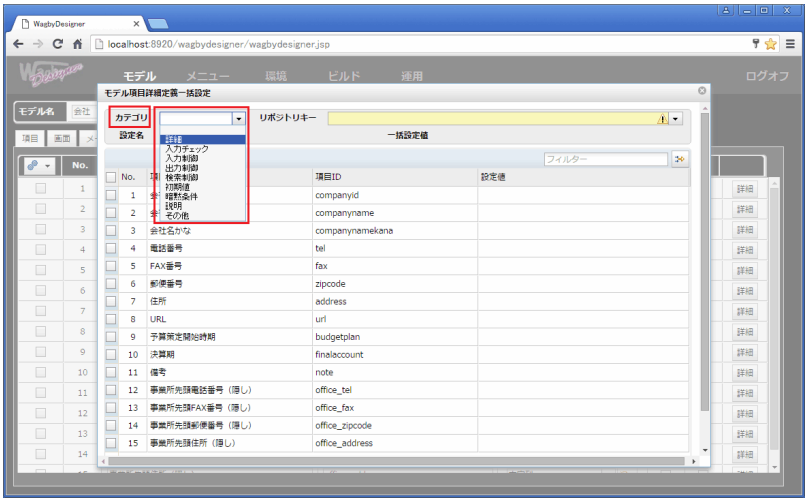


図 カテゴリを指定する

カテゴリに属する、すべてのリポジトリキーのリストが用意されます。設定したいリポジトリキーを選択してください。

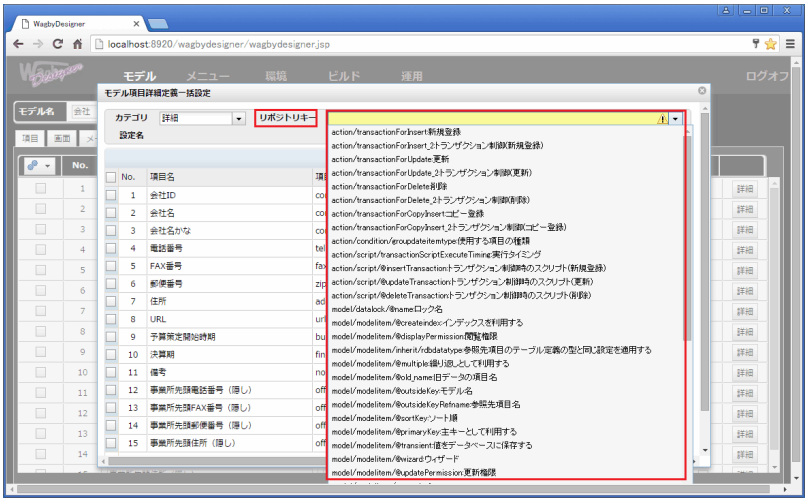


図 リポジトリキーを選択する

リポジトリキーの詳細は「リポジトリー覧」を参照してください。

関連:項目ルール

ここで紹介した方法とは別に、項目ルールを設定することもできます。項目ルールはデータ辞書として利用でき、ルール名によって判断が行いやすいという特徴があります。こちらの利用もご検討ください。

プロジェクト・モデル・モデル項目の説明文を用意する

プロジェクトの説明

「環境 > プロジェクト」を開きます。プロジェクト説明欄に、自由に文章を記述できます。

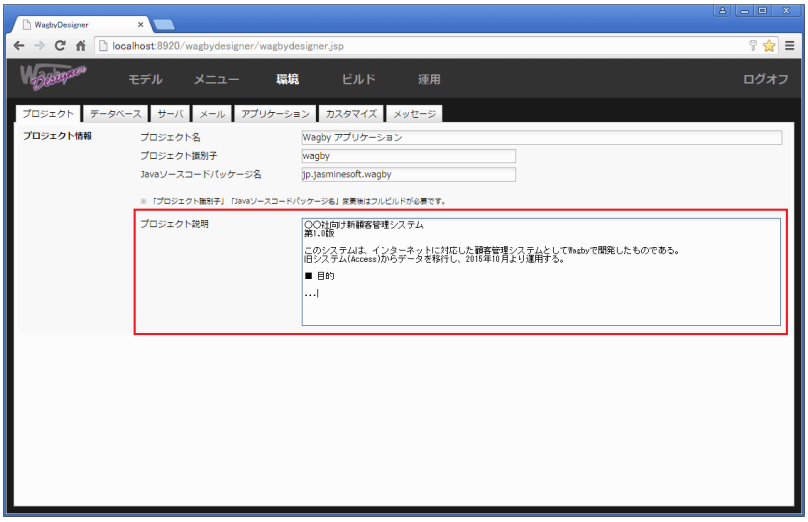


図 プロジェクトの説明

モデルの説明

「モデル ＞ その他 ＞ モデル情報」を開きます。「概要」と「説明」を用意しています。概要は10文字前後を目安としてください。

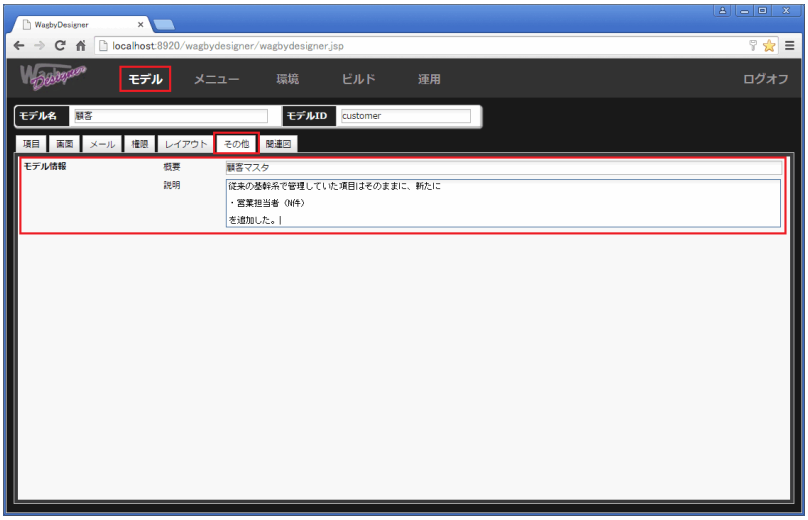


図 モデルの説明

モデル一覧で、概要（の文章）を確認できます。

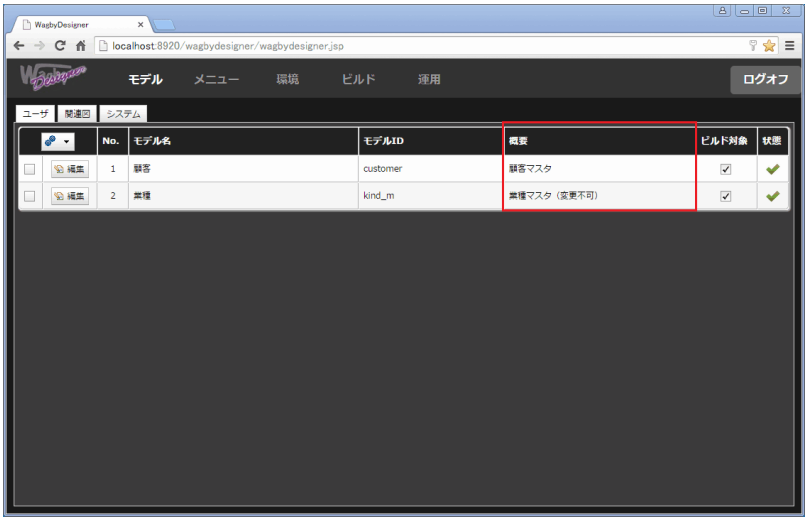


図 概要がモデル一覧に表示される

モデル項目の説明

モデル項目設定ダイアログにて「その他」を開きます。「モデル項目説明」に記述できます。

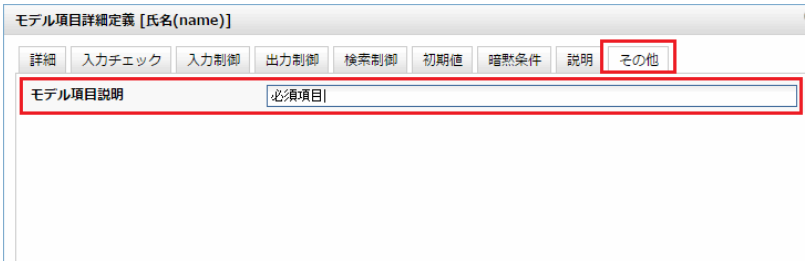


図 モデル項目の説明

モデル項目一覧画面で、該当するモデル項目の「詳細」ボタンにマウスを合わせると、説明文がツールチップ形式で表示されます。

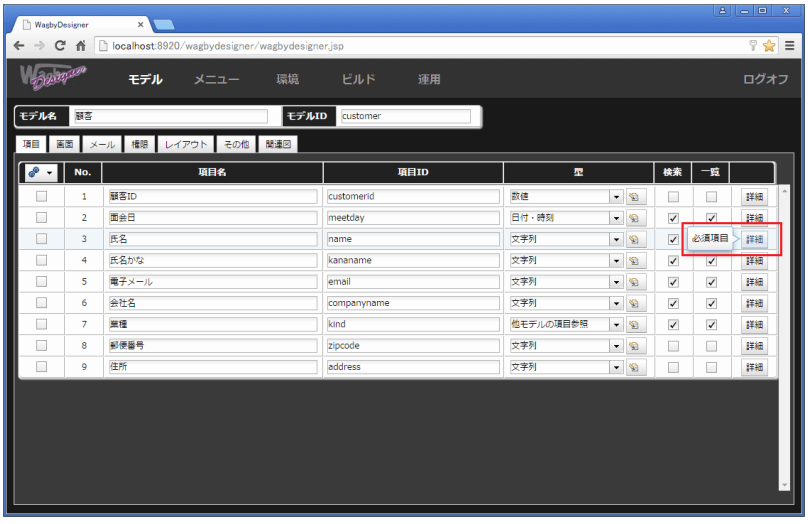


図 説明文がツールチップで表示される

リポジトリのバックアップとリストア

リストア（リポジトリの復元）

「運用」メニュー内の「リポジトリ」タブで、リポジトリのバックアップとリストアを行うことができます。

多くのバックアップファイルがある場合、図の赤枠部分に任意の文字列を入力することで、フィルタ処理による表示の絞り込みを行うことができます。

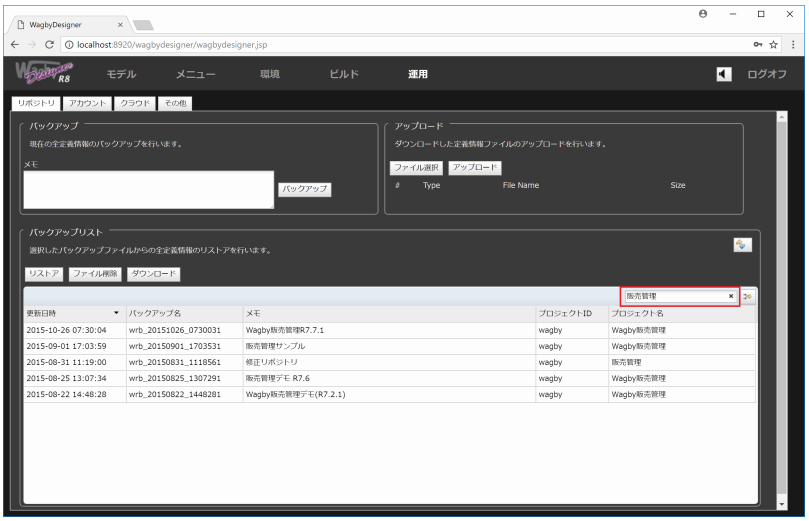


図 運用メニューのリポジトリタブ

バックアップリストから任意のリポジトリを選択し、「リストア」ボタンを押下すると、その時点のリポジトリに戻す（復元する）ことができます。

図 任意の時点のリポジトリをリストアする

旧バージョンのリポジトリを読み込む

旧バージョンでバックアップされた zip 形式のリポジトリファイルを直接、読み込むことができます。読み込み時に移行処理が適用されます。

(506)

リポジトリ読み込み後のビルド

新しいリポジトリを読み込んだあとは「フルビルド」を行ってください。

(506) 「システム」タブに含まれるモデルは原則として移行の対象になりません。常に新しいモデル定義になります。（一部例外あり）システムタブに含まれるモデル定義を変更していた場合は移行後、手動で同じ修正を行うようにしてください。詳細はこちらをお読みください。

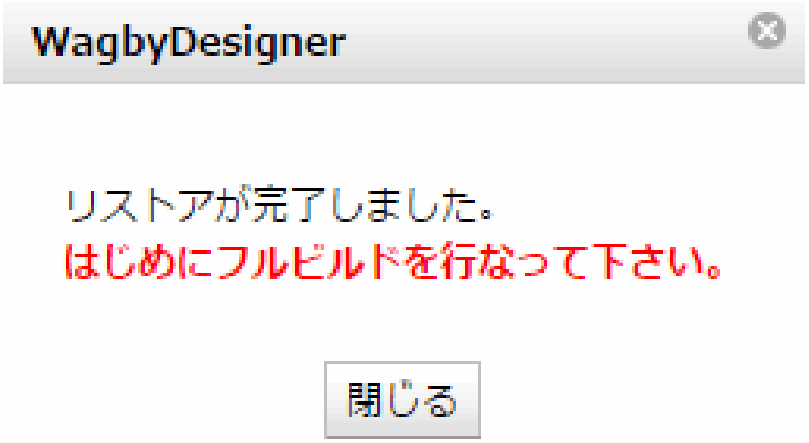


図 リポジトリをリストアしたときに表示されるダイアログ

手動バックアップ

現在のリポジトリのバックアップを作成することができます。メモを残すと、リストア時に探しやすくなります。

図 リポジトリの手動バックアップ

自動バックアップ

ビルド画面では標準で「ビルド時にバックアップを行う」が有効になっています。メモを残すことで、保存リポジトリの一覧から探しやすくなります。

このチェックをはずすと、リポジトリの自動バックアップは行われません。

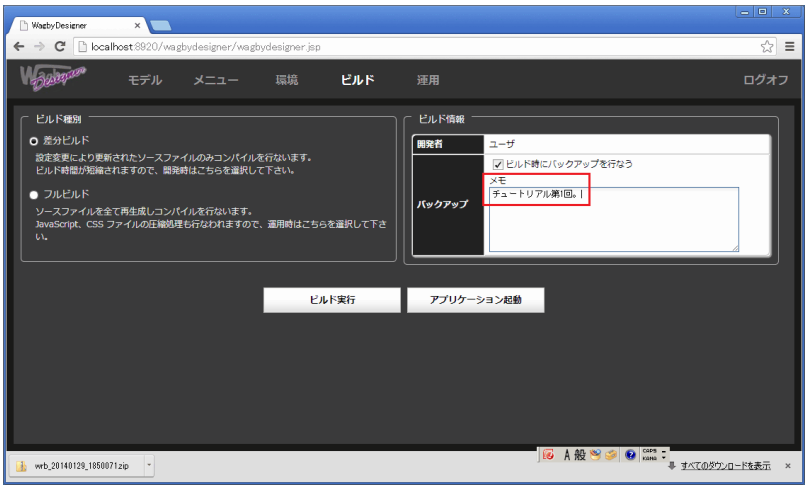


図 リポジトリの自動バックアップ

リポジトリをファイルとして取得する

バックアップリストから任意のリポジトリを選択し、「ダウンロード」ボタンを押下すると、そのリポジトリファイル一式をzip圧縮ファイルとしてダウンロードすることができます。

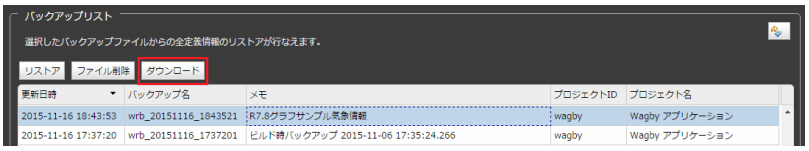


図 ダウンロード

ダウンロードファイルは ″wb_″ という接頭語のあとに日付時刻情報が付与された zip 圧縮形式です。

図 ダウンロードされたファイル

リポジトリファイルを読み込む

zip圧縮形式のリポジトリファイルをアップロードすることで、WagbyDesignerに取り込むことができます。 「ファイル選択」ボタンを押下して、リポジトリファイル（″wb_″ ではじまる zip 圧縮形式）を指定します。

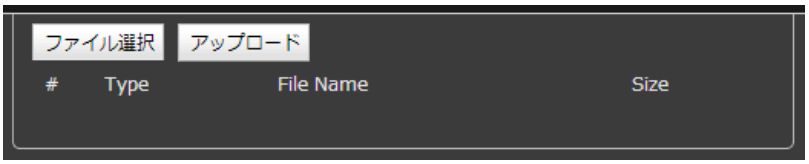


図 ファイル選択とアップロード

ファイル選択後、「アップロード」ボタンを押下することで、バックアップリストに格納されます。

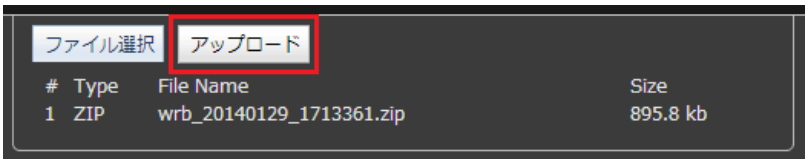


図 アップロード

操作例 他者が作成したリポジトリを読み込む

別の開発者によって設計されたリポジトリを、自分の開発PCにロードする（読み込む）手順を説明します。

1. 運用メニューの「アップロード」欄から「ファイル選択」ボタンを開きます。ファイルダイアログが表示されるので、読み込むリポジトリファイル（通常は ″wrb_″ で始まる、zip 形式ファイル）を指定します。

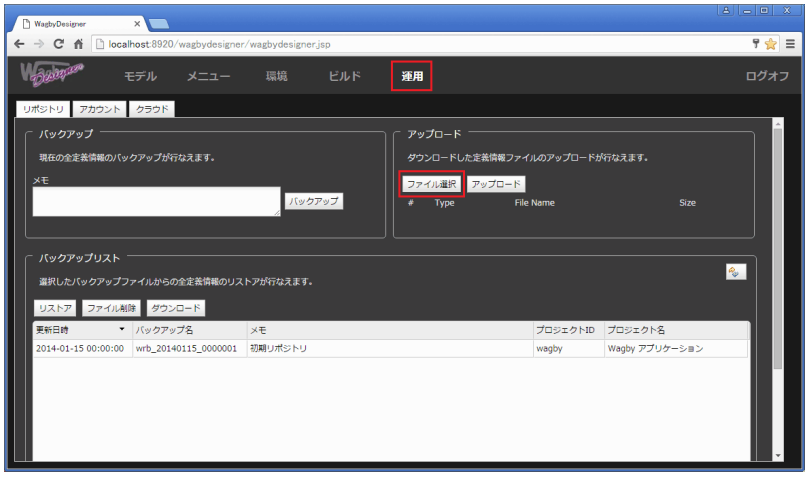


図 ファイルの選択

2. ファイル選択後、「アップロード」ボタンを押下します。

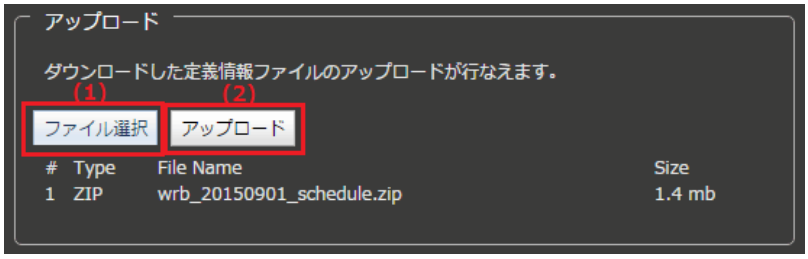


図 アップロード

3. アップロード後、すぐにリストアを行うかどうかを確認するメッセージが表示されます。ここでは、リストアするを選択した場合を説明します。

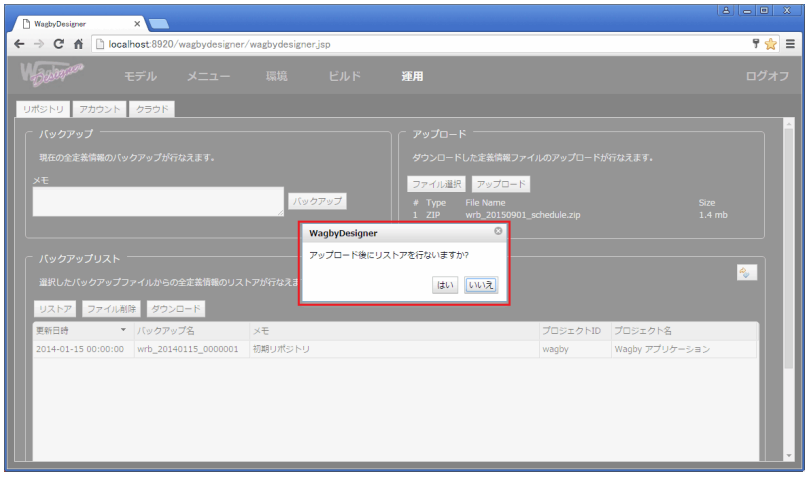


図 すぐにリストアを行うかどうかを確認するダイアログ

4. リストア処理中です。

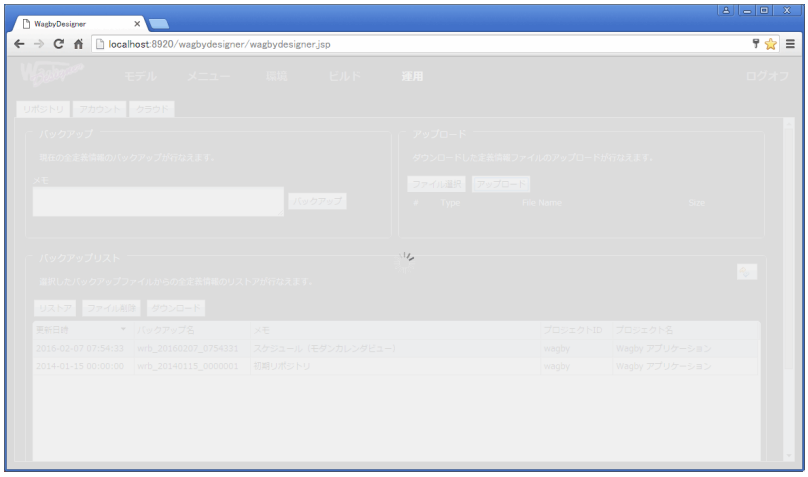


図 リストア処理中

5. リストアが完了すると、モデルが表示されます。フルビルドを行うと、アプリケーションが生成されます。

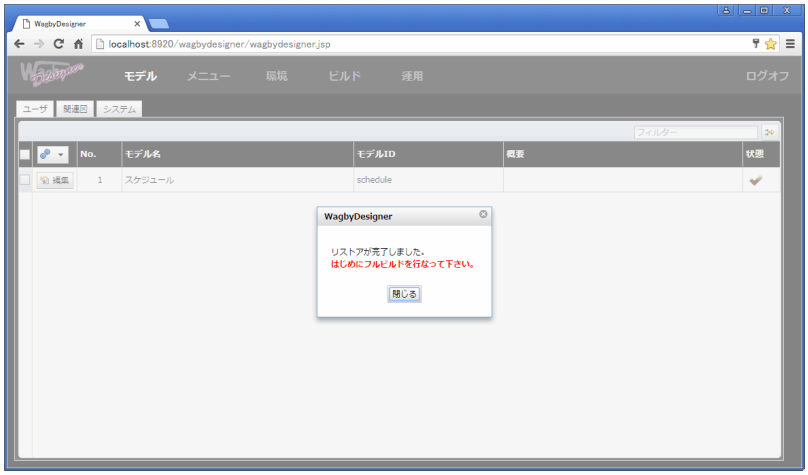


図 リストア完了

6. 運用タブを確認すると、リストアしたリポジトリが「バックアップリスト」に含まれていることがわかります。

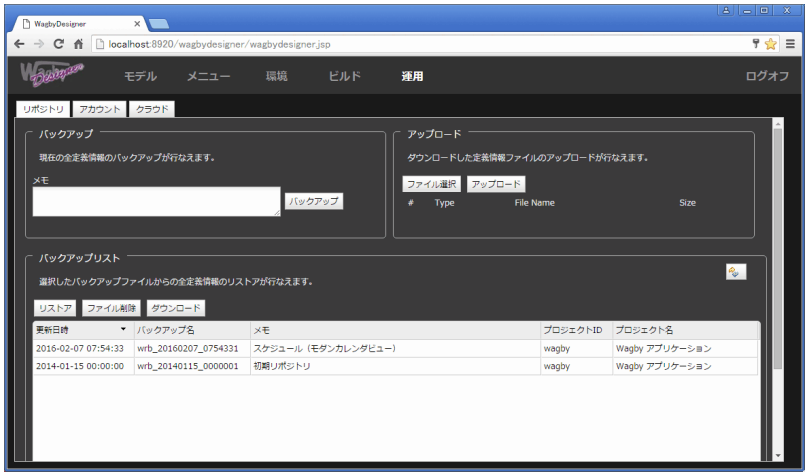


図 バックアップリストを確認する

このリストア処理により、それまでのリポジトリは消去されます。開発途中のプロジェクトがある場合は、リストアの前に現在のリポジトリのバックアップを取得してください。

トラブルシューティング

リストアに失敗する。

WagbyDesignerを起動している Tomcat コンソールまたは logs/catalina.out にエラーメッセージが記録されます。この内容を確認してください。あわせて、次の点をご確認ください。

- repository/branches フォルダに初期リポジトリファイル wrb_20180130_0000001.zip は含まれていますか。リストア時に、このファイルを参照します。
- repository/branches フォルダに含まれている初期リポジトリファイル wrb_20180130_0000001.zip は、古いものを使っていませんか。このファイルはインストールされているバージョンの Wagby によって内容が異なります。必ず、インストール時に含まれているファイルをお使いください。
- 無制限開発ライセンスと、プロジェクトライセンスが混在していませんか。混在している場合、リストアを行うことはできません。
- 無制限開発ライセンス利用時は、購入時に変換されたリポジトリファイルを用いる必要があります。詳細は販売パートナーにお問い合わせください。

ビルド時のバックアップファイルがDesignerから参照できない。

ウィルス対策ソフトが原因となっていることがあります。ファイル除外設定を行ってください。

もっと詳しく

- "よくある質問と回答 > WagbyDesignerのエラー > リポジトリのエラー"をお読みください。

技術情報

wrb.properties

作成されたzip形式のバックアップファイルには、wrb.propertiesというファイルが含まれます。これはモデル名など、バックアップファイルの内容を説明するための内部情報となっており、リストア時に用いています。

リストア後、すなわち repository/trunk フォルダに展開されたあとは wrb.properties は利用されません。そのためバージョン管理の対象外としてください。

- 参考情報:リポジトリ構造とチーム開発 > リポジトリフォルダの構造

リポジトリの再読み込み

リポジトリの再読み込み方法

「運用」メニュー内の「リポジトリ」タブに用意された「再読み込み（リロード）」を押します。

図 リポジトリ再読み込みボタン

再読み込みが完了すると、図のようなダイアログが表示されます。

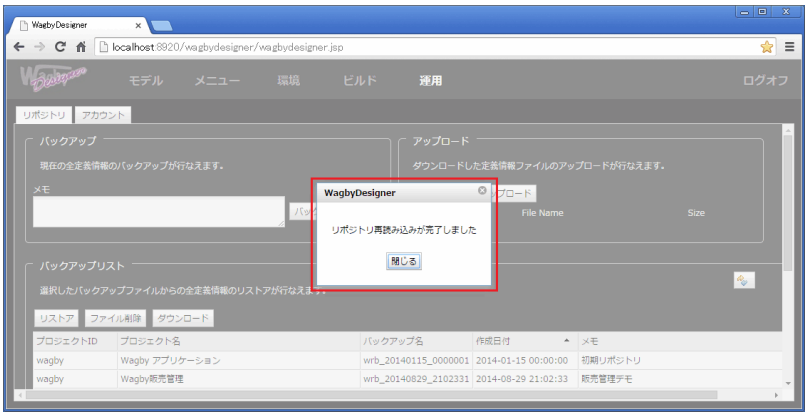


図 再読み込みが完了した

再読み込みが必要なケース

Designerで入力した値が反映されていないとき

Designer で入力した値が反映されていないと感じたとき、再読み込み処理によって ″どこまで反映されたか″ を確認することができます。

具体的には入力エラーが生じたとき、エラー発生後の入力は無視される場合があります。

エラー状態を解消したいとき

何らかの入力エラーが生じたとき、モデル一覧で「状態」に警告アイコン（オレンジ色のマーク）が表示されます。入力エラーを解消しても警告アイコンが残っている場合、再読み込み処理によってエラー状態を初期化することができます。

repository フォルダを直接、編集したとき

repository フォルダを直接、編集することがあります。次のような場合です。

- バージョン管理システムと同期して、他者が変更した情報を取得する。
- 他者が開発したモデルを「フォルダ単位」で受け取り、repository/trunk フォルダ直下に保存する。

このように Wagby Designer を介さずに repository フォルダを変更した場合は、最新の情報を (Wagby Designer に) 読み込ませる必要があります。

関連図の表示

関連図

「関連図」タブを開くと、すべてのモデル名と主キーがアイコンで表示されます。

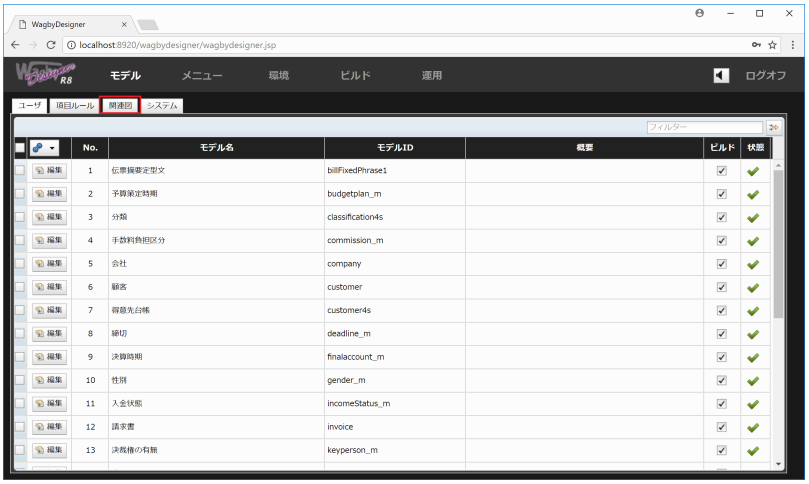


図 関連図タブを開く(1)

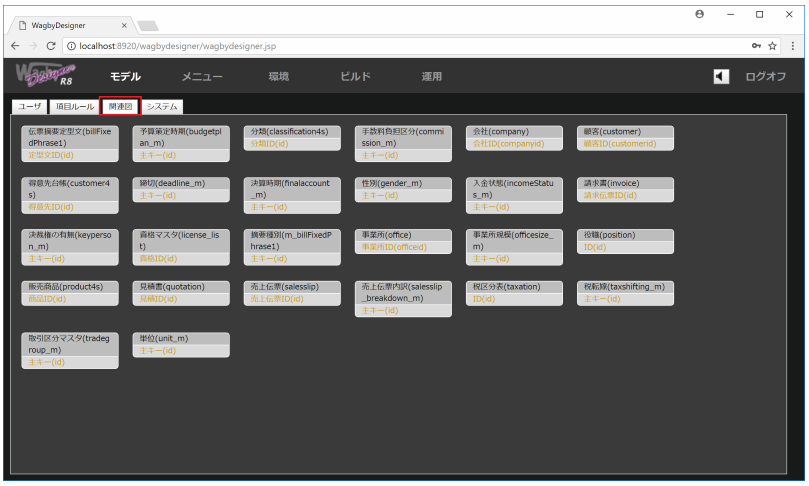


図 関連図タブを開く(2)

任意のモデルをマウスで選択すると、そのモデルに関連する他のモデルも含めて画面に描画されます。 図の中心に配置されるのが、選択したモデルです。参照されているモデルが左側に、参照しているモデルが右側に表示されます。

図 自分自身および、関連するモデル

別モデルを選択する

ここで別のモデルを選択すると、選択したモデルを中心にして、同じルールで再描画されます。

図 再描画された関連図

表示ルール

- 参照されているモデルが左側に、参照しているモデルが右側に表示されます。
- メインモデルは上部、外部キー子モデルは下部に表示されます。
- 関連図のモデルのボックスをクリックすると当該モデルの関連図が表示されるようになります。
- 左上の←→ボタンで上記操作で遷移したモデルの関連図の表示を切り替えることができます。
- システムタブに含まれるモデル（juserなど）は表示対象外となります。

モデル、モデル項目のテンプレートを用意する

項目テンプレートの設定

「システム」タブに、「ユーザ定義項目テンプレート」（モデルIDが"jfcDesignerUserTemplateModelItem" ではじまる 9 つのモデル）が予め、用意されています。これが項目テンプレートの入れ物になります。

（「登録・更新情報項目」は、事前に定義済みです。）

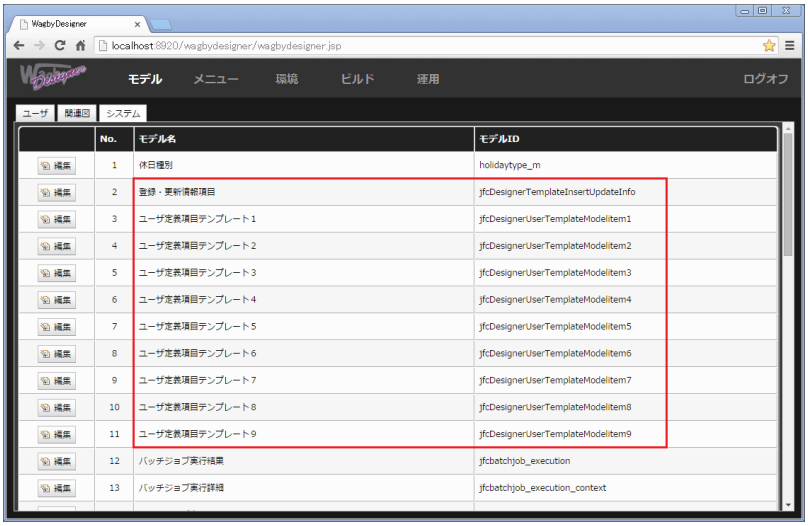


図 ユーザ定義項目テンプレート

ユーザ定義項目テンプレートは最大で 9 つ、用意することができます。

「ユーザ定義項目テンプレート1」を選択したケースを図に示します。

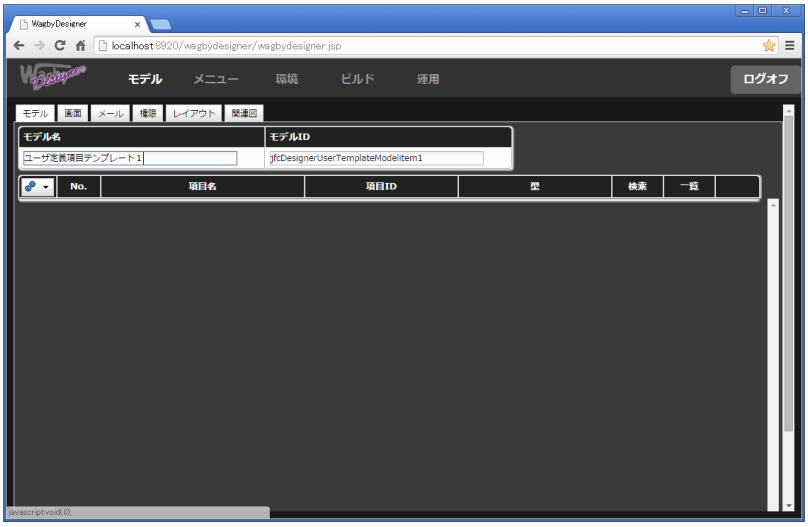


図 ユーザ定義項目テンプレート1を設定する

モデル名は変更できます。モデルIDを変更することはできません。 ここではモデル名を「登録・更新情報項目（和暦版）」とします。4つの項目を設定してみます。

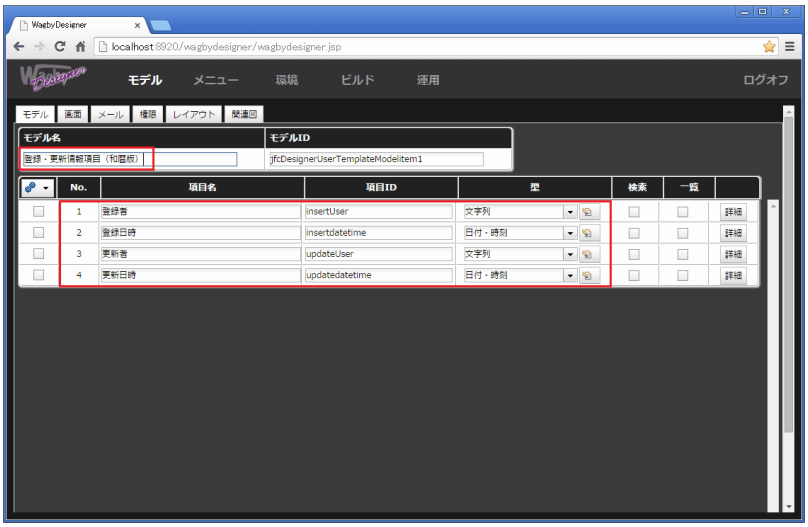


図 登録・更新情報項目（和暦版）を定義する

設定できる内容

設定できる内容はモデル項目のみです。「画面」「メール」「権限」「レイアウト」は設定できません。

項目テンプレートを使う

開発者が設定したモデル（ここでは「顧客」モデル）に、上で用意した項目テンプレートを加えてみます。 ギアアイコンから「テンプレート」を選択します。図で用意したモデル名が、選択肢に含まれていることがわかります。

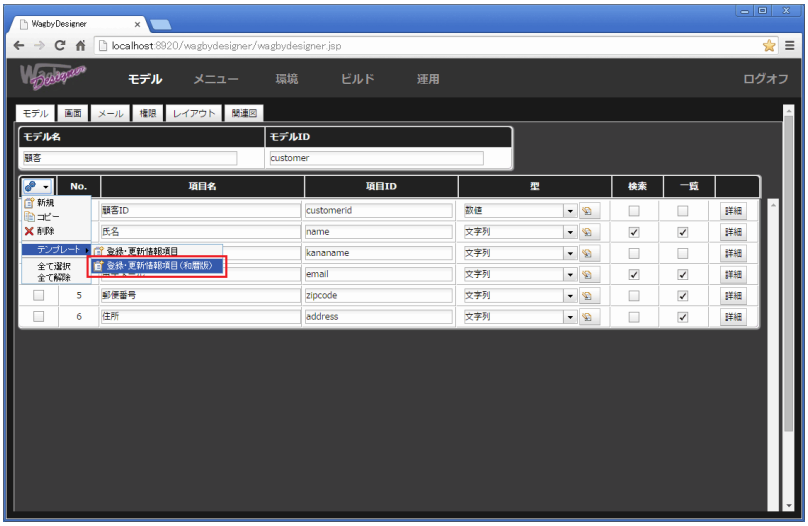


図 項目テンプレートを選択する

項目が追加されました。

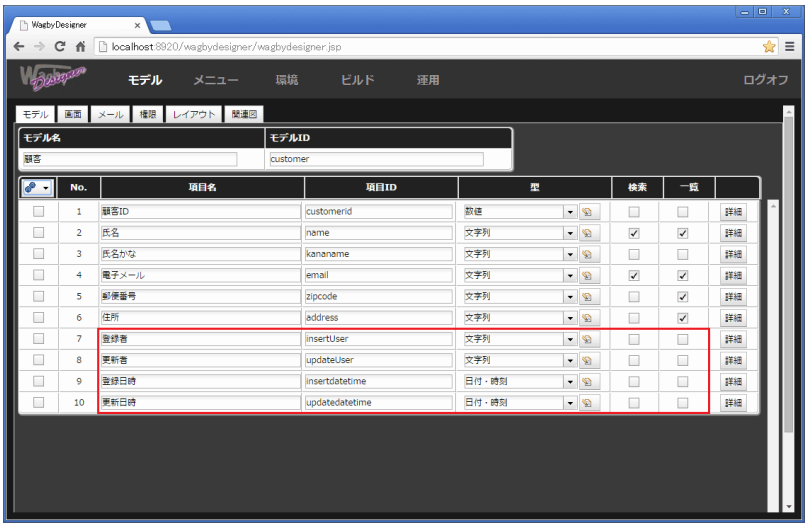


図 項目テンプレートが追加された

モデルのテンプレートを自作する

独自テンプレートモデルを用意する

モデル項目ではなくモデル自体のテンプレートを、開発者が用意することもできます。

あらかじめ、開発者が必要となる初期設定を組み込んだモデルを定義しておきます。これを「独自テンプレートモデル」と呼びます。 新しくモデルを追加するとき、この独自テンプレートモデルをコピーします。

具体的な手順は次のとおりです。

1. 独自テンプレートモデルを作成します。モデル名は任意ですが、わかりやすいように何らかの接頭語を付与するとよいです。
2. 1. で作成した独自テンプレートモデルに、必要な初期設定を行います。
3. このモデルがビルド対象とならないよう、モデル一覧にて〝ビルド〟のチェックを外します。

独自テンプレートモデルを利用する

新規にモデルを作成する場合、この独自テンプレートモデルをコピーします。対象モデルを選択し、ギアアイコンから「コピー」を選びます。

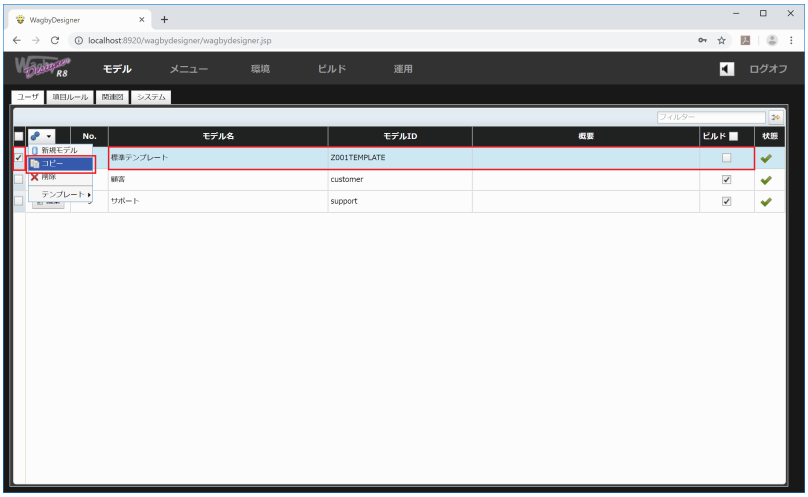


図 独自テンプレートモデルをコピーする(1)

コピーされました。このモデルに名前を付与します。

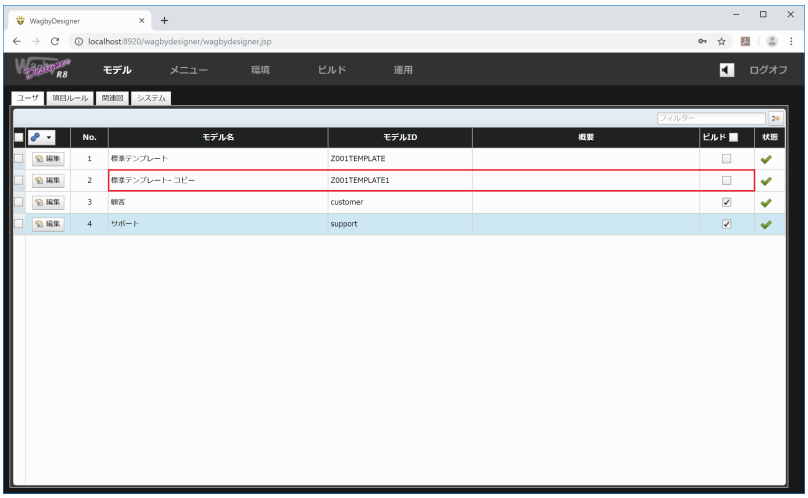


図 独自テンプレートモデルをコピーする(2)

適切な名前を設定し、ビルド対象とします。

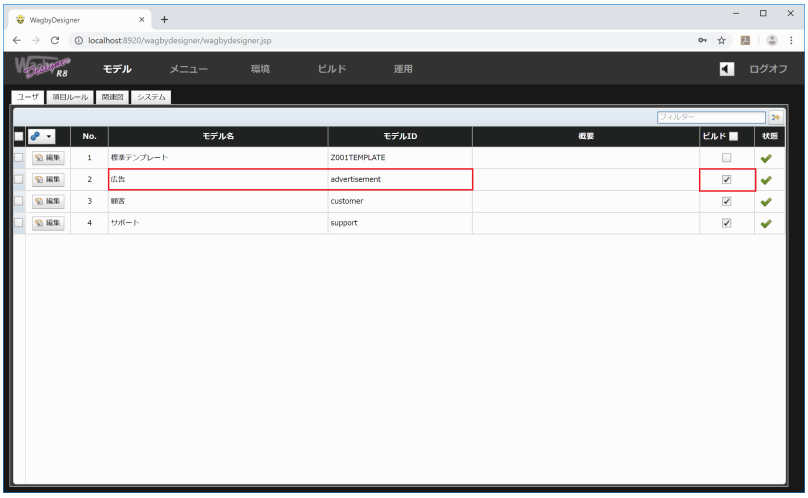


図 名前を設定し、ビルド対象とする

関連する機能

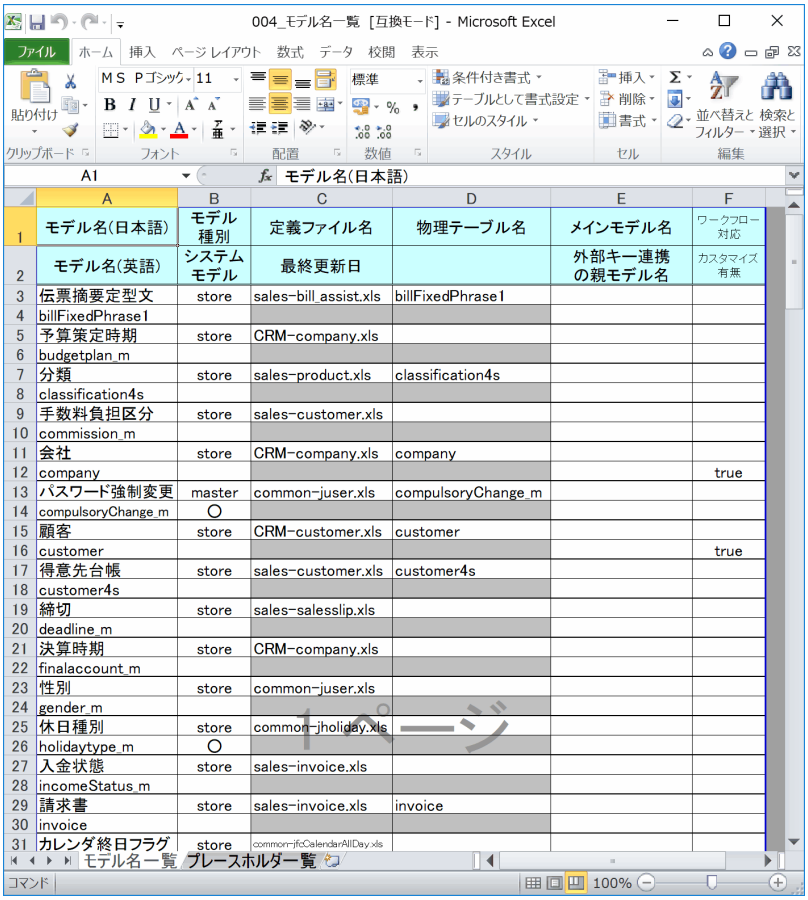
・　　モデル項目 ＞ 項目ルールを定義する（データ辞書としての利用）

設計書の出力

設計書出力機能とは

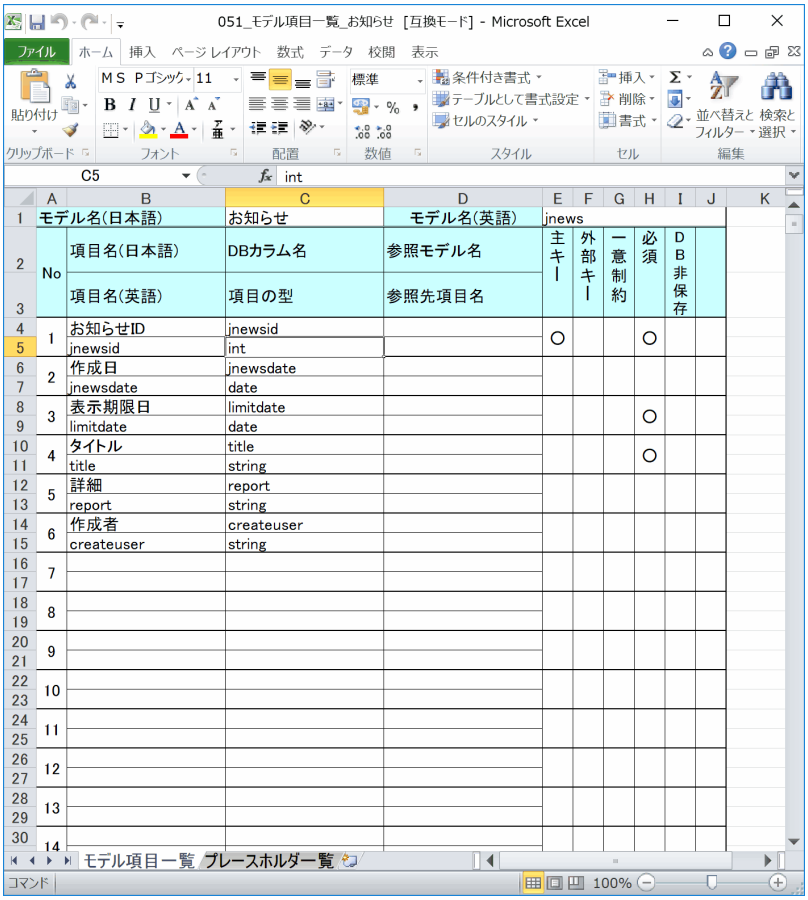
アプリケーションのリポジトリ（設計情報）を Excel 形式のファイルで出力する機能です。印刷してお使いいただくことができます。

本ツールが出力するExcelファイルの例を図～図に示します。



	A	B	C	D	E	F
1	モデル名(日本語)	モデル種別	定義ファイル名	物理テーブル名	メインモデル名	ワークフロー対応
2	モデル名(英語)	システムモデル	最終更新日		外部キー連携の親モデル名	カスタマイズ有無
3	伝票摘要定型文	store	sales-bill_assist.xls	billFixedPhrase1		
4	billFixedPhrase1					
5	予算策定時期	store	CRM-company.xls			
6	budgetplan_m					
7	分類	store	sales-product.xls	classification4s		
8	classification4s					
9	手数料負担区分	store	sales-customer.xls			
10	commission_m					
11	会社	store	CRM-company.xls	company		
12	company					true
13	パスワード強制変更	master	common-juser.xls	compulsoryChange_m		
14	compulsoryChange_m	○				
15	顧客	store	CRM-customer.xls	customer		
16	customer					true
17	得意先台帳	store	sales-customer.xls	customer4s		
18	customer4s					
19	締切	store	sales-salesslip.xls			
20	deadline_m					
21	決算時期	store	CRM-company.xls			
22	finalaccount_m					
23	性別	store	common-juser.xls			
24	gender_m					
25	休日種別	store	common-jholiday.xls			
26	holidaytype_m	○				
27	入金状態	store	sales-invoice.xls			
28	incomeStatus_m					
29	請求書	store	sales-invoice.xls	invoice		
30	invoice					
31	カレンダ終日フラグ	store	common-jfcCalendarAllDay.xls			

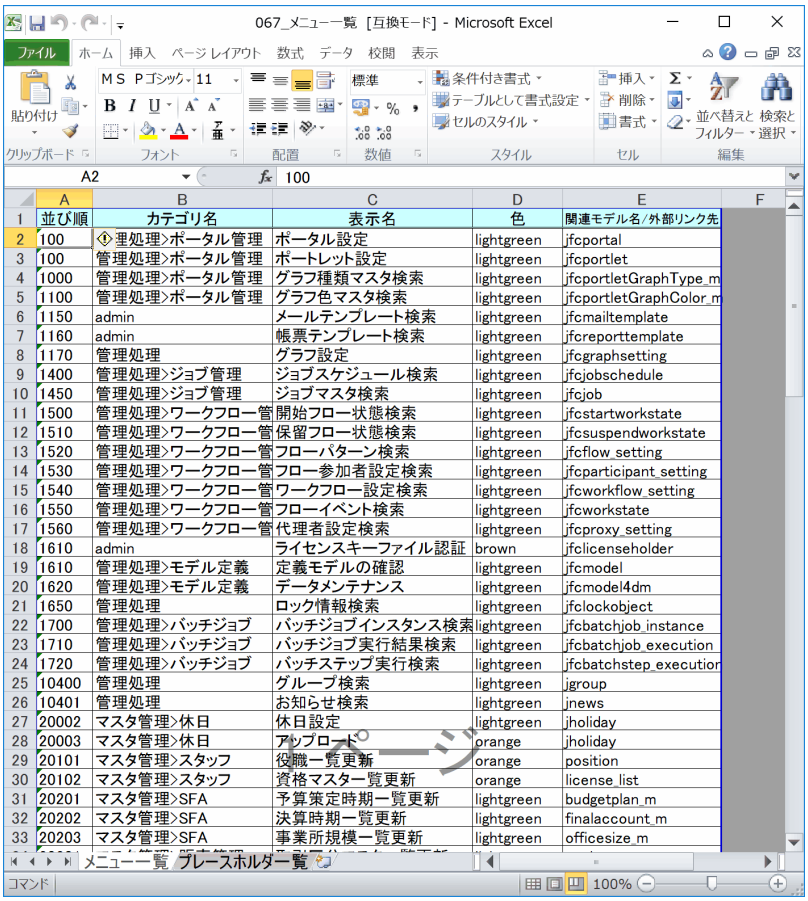
図 モデル名一覧



051_モデル項目一覧_お知らせ [互換モード] - Microsoft Excel

	A	B	C	D	E	F	G	H	I	J	K
1	モデル名(日本語)		お知らせ	モデル名(英語)	jnews						
2	No	項目名(日本語)	DBカラム名	参照モデル名	主キー	外部キー	一意制約	必須	DB非保存		
3		項目名(英語)	項目の型	参照先項目名							
4	1	お知らせID	jnewsid		○			○			
5		jnewsid	int								
6	2	作成日	jnewsdate								
7		jnewsdate	date								
8	3	表示期限日	limitdate					○			
9		limitdate	date								
10	4	タイトル	title					○			
11		title	string								
12	5	詳細	report								
13		report	string								
14	6	作成者	createuser								
15		createuser	string								
16	7										
17											
18	8										
19											
20	9										
21											
22	10										
23											
24	11										
25											
26	12										
27											
28	13										
29											
30	14										

図 モデル項目一覧（モデル毎に用意される）



067_メニュー一覧 [互換モード] - Microsoft Excel

	A	B	C	D	E	F
1	並び順	カテゴリ名	表示名	色	関連モデル名/外部リンク先	
2	100	運処理>ポータル管理	ポータル設定	lightgreen	jfcportal	
3	100	管理処理>ポータル管理	ポートレット設定	lightgreen	jfcportlet	
4	1000	管理処理>ポータル管理	グラフ種類マスタ検索	lightgreen	jfcportletGraphType_m	
5	1100	管理処理>ポータル管理	グラフ色マスタ検索	lightgreen	jfcportletGraphColor_m	
6	1150	admin	メールテンプレート検索	lightgreen	jfcmailtemplate	
7	1160	admin	帳票テンプレート検索	lightgreen	jfcreporttemplate	
8	1170	管理処理	グラフ設定	lightgreen	jfcgraphsetting	
9	1400	管理処理>ジョブ管理	ジョブスケジュール検索	lightgreen	jfcjobschedule	
10	1450	管理処理>ジョブ管理	ジョブマスタ検索	lightgreen	jfcjob	
11	1500	管理処理>ワークフロー管	開始フロー状態検索	lightgreen	jfcstartworkstate	
12	1510	管理処理>ワークフロー管	保留フロー状態検索	lightgreen	jfcsuspendworkstate	
13	1520	管理処理>ワークフロー管	フローパターン検索	lightgreen	jfcflow_setting	
14	1530	管理処理>ワークフロー管	フロー参加者設定検索	lightgreen	jfcparticipant_setting	
15	1540	管理処理>ワークフロー管	ワークフロー設定検索	lightgreen	jfcworkflow_setting	
16	1550	管理処理>ワークフロー管	フローイベント検索	lightgreen	jfcworkstate	
17	1560	管理処理>ワークフロー管	代理者設定検索	lightgreen	jfcproxy_setting	
18	1610	admin	ライセンスキーファイル認証	brown	jfclicenseholder	
19	1610	管理処理>モデル定義	定義モデルの確認	lightgreen	jfcmodel	
20	1620	管理処理>モデル定義	データメンテナンス	lightgreen	jfcmodel4dm	
21	1650	管理処理	ロック情報検索	lightgreen	jfclockobject	
22	1700	管理処理>バッチジョブ	バッチジョブインスタンス検索	lightgreen	jfcbatchjob_instance	
23	1710	管理処理>バッチジョブ	バッチジョブ実行結果検索	lightgreen	jfcbatchjob_execution	
24	1720	管理処理>バッチジョブ	バッチステップ実行検索	lightgreen	jfcbatchstep_execution	
25	10400	管理処理	グループ検索	lightgreen	jgroup	
26	10401	管理処理	お知らせ検索	lightgreen	jnews	
27	20002	マスタ管理>休日	休日設定	lightgreen	jholiday	
28	20003	マスタ管理>休日	アップロード	orange	jholiday	
29	20101	マスタ管理>スタッフ	役職一覧更新	orange	position	
30	20102	マスタ管理>スタッフ	資格マスター覧更新	orange	license_list	
31	20201	マスタ管理>SFA	予算策定時期一覧更新	lightgreen	budgetplan_m	
32	20202	マスタ管理>SFA	決算時期一覧更新	lightgreen	finalaccount_m	
33	20203	マスタ管理>SFA	事業所規模一覧更新	lightgreen	officesize_m	

図 メニュー一覧

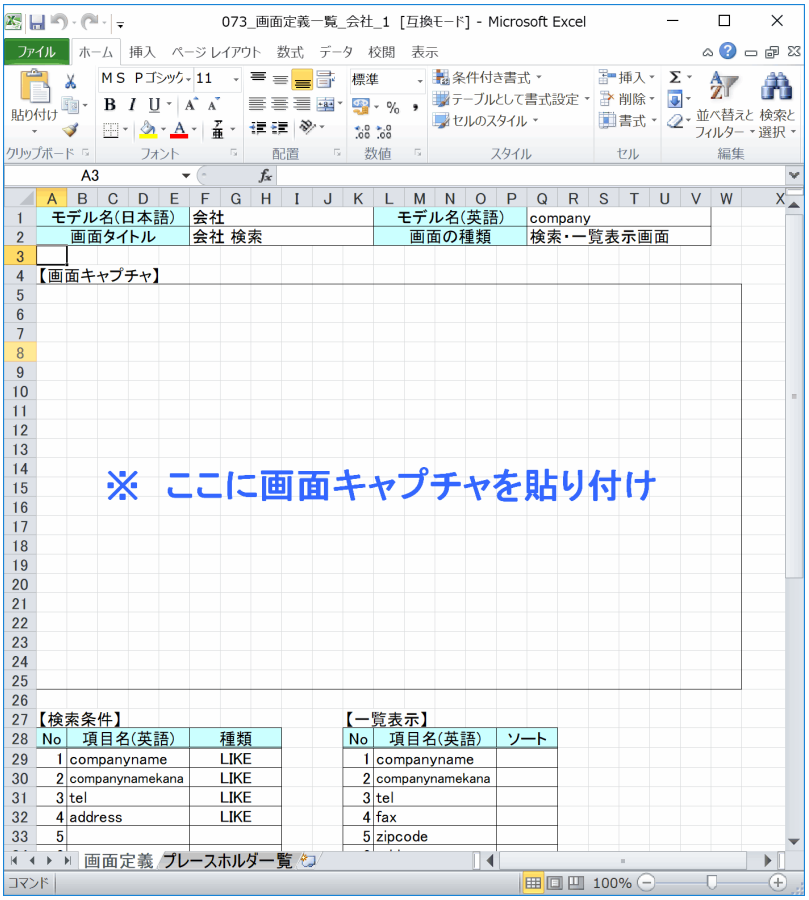


図 画面定義一覧（モデル毎に用意される）

変更履歴

R8.4.0

- ER図出力機能で実行時エラーが生じる場合があったため、これを修正する。
- 出力ファイル名にスラッシュが含まれていると、パスの区切り文字と解釈されて出力エラーになったため、ファイル名に混じったスラッシュ '/' をアンダースコア '_' に変更する。
- 出力ファイルの先頭に付与される連番の桁数を変更できるようにする。(例 "0"詰め4桁にする)

R8.3.3

- ER図出力機能で、選択肢モデルを含める設定を用意する。（設定ファイルの変更で対応）

R8.3.2

- テンプレートファイル wagbydesinger/bin/spec/template/front.xls を編集し、更新日のプレースホルダを ${createdateYMD} とすると日付を西暦にすることができる。

R8.3.0

- すべてのモデルを「ビルド対象」に含めないと実行時エラーになる場合がある不具合を修正。
- ER図出力の画面配置アルゴリズムを修正。選択肢モデルはER図に含めないようにする。
- 画面遷移図、スクリプト一覧表を生成する。
- モデル名一覧のモデル種別を "store","master" という表記から、"システム","ユーザ" へ変更する。"システム" は内部で利用しているモデルで、"ユーザ" は開発者が作成したモデルを意味する。

R8.2.0

- Designerで設定した「モデルの概要」「モデルの説明」「モデル項目の説明」を設計書に出力できるようにする。設計書出力のテンプレートファイルmodel_list.xlsに"model[N]/@outline","model[N]/@note"を指定できる。またmodelitem_list.xlsに"model/@outline","model/@note","modelitem[N]/@note"を指定できる。
- 一覧表示画面をつくらないモデルを含んだプロジェクトで設計書出力を行うと実行時エラーが発生する不具合を修正。
- 設計書出力処理を連続で行うと、出力ファイル名に "(2)","(3)",... といったサフィックスが含まれてしまう不具合を修正。

R8.0.2

- 一覧表示項目として指定しないが、一覧帳票に出力する設定を行うと「画面定義一覧」に含まれてしまっていたため、これを除く。

使い方

準備

設計書を出力する前に、ビルド処理を行ってください。直前にビルドされた状態の設計書を出力します。

設計書の作成

「運用 ＞ その他 ＞ 設計書」タブを開きます。「設計書出力」ボタンを押下すると、設計書の作成が開始されます。

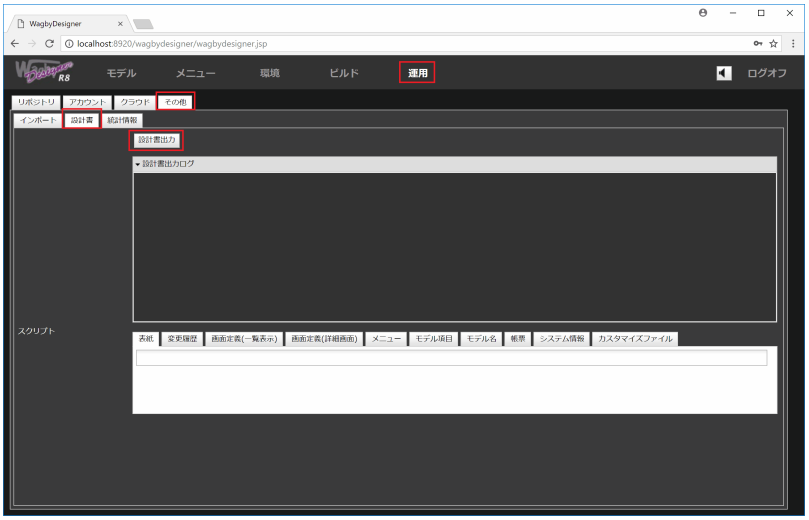


図 設計書出力ボタン

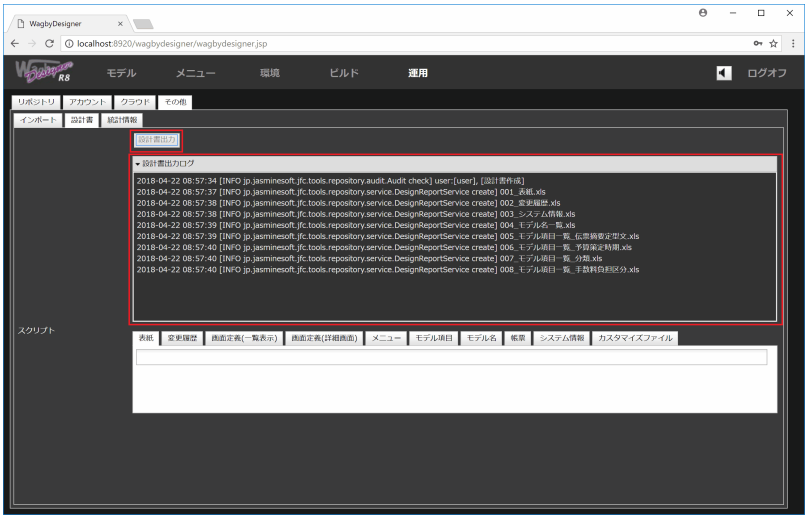


図 設計書作成中

図に示すように、作成された設計書がダウンロードされます。ファイルは zip 形式で圧縮されています。

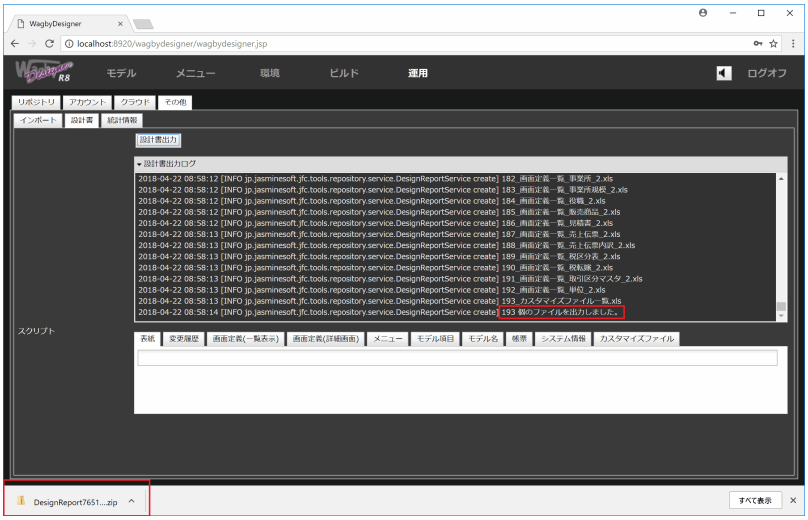


図 設計書が作成された

ダウンロードしたファイルを展開してください。複数の Excel ファイルが含まれています。

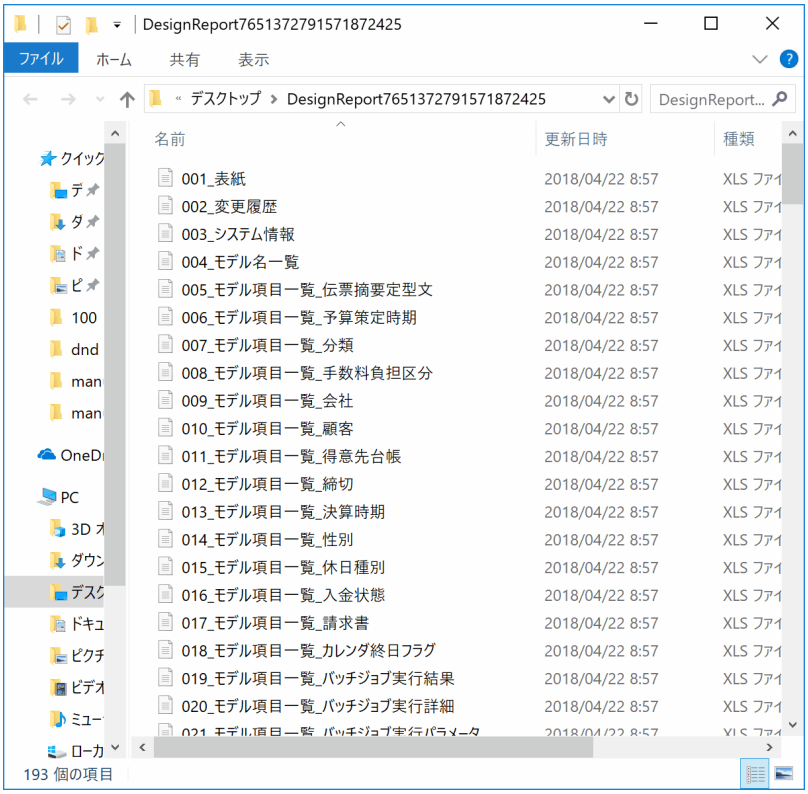


図 複数のExcelファイル

自動生成される設計書の内容

次のドキュメントが自動生成されます。

- 表紙
- 変更履歴
- システム情報
- モデル名一覧
- 各モデル項目毎の定義
- メニュー一覧
- 帳票一覧
- 各モデル毎の画面定義

・　　カスタマイズファイル一覧
・　　ER図（後述）

画面キャプチャの自動実行は行いません。生成された「画面定義書」に、スクリーンショットを貼り付けて完成させてください。

ER図,画面遷移図,スクリプト一覧表（8.3.0）

ER図

ファイル名 "ER_Diagram.xlsx" には、ER図が含まれます。ファイルの拡張子は ".xlsx" です。例を示します。

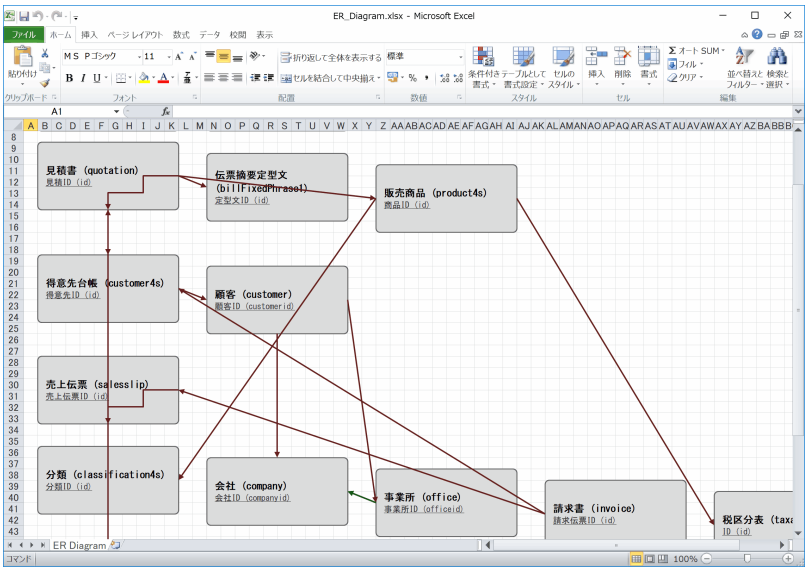


図 ER図(1)

エンティティ同士の関連は「コネクタ」となっています。そのため、あるエンティティを選択して移動したとき、コネクタも一緒に移動します。

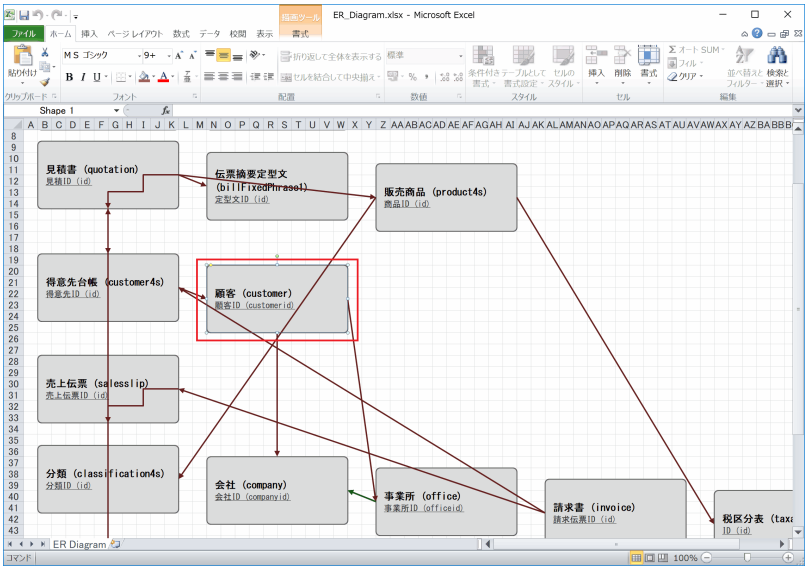


図 ER図(2)

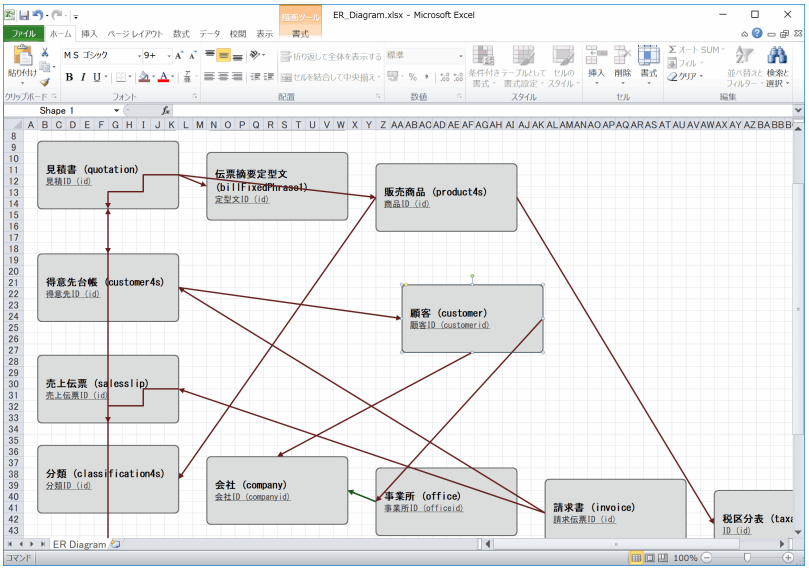


図 ER図(3)

設計書出力機能では、エンティティは単純な並びですがコネクタは相互に関連づけているため、開発者はエンティティを適切に再配置して、よりみやすいER図を作成することができます。

コネクタの色

コネクタの配色ルールは次の通りです。

- 外部キー関係は緑色。
- 他モデル参照は紫色。

エンティティの配置

エンティティの配置ルールは次の通りです。

- 参照数の多いもの(結合度の高いもの)を左に配置する。
- 上記ノード(モデル)から参照されているノードを右に配置する。
- juserなどシステム系は、このプロジェクトで作成したモデルで参照されているもののみを対象にする。
- 選択肢モデルは配置しない。

選択肢モデルの出力 (8.3.3)

標準では選択肢モデルは出力されません。選択肢モデルも出力に含める場合は、次の手順で設定ファイルを修正してください。

1. Designerを停止します。
2. wagbydesigner/webapps/wagbydesigner/WEB-INF/classes/poi_diagram.properties ファイルをテキストエディタで開きます。
3. 次の行を見つけます。

```
erd.display.optionalmodel = false
```

4. 設定値を false から true に変更します。

```
erd.display.optionalmodel = true
```

5. ファイルを保存し、Designerを再起動します。

画面遷移図

モデル毎に、利用している画面を表示します。また画面間で遷移可能な矢印を表示します。

ボタン名を設定していた場合、画面機能（黒字）の下に、ボタン名を青字で表記しています。

オリジナルボタンはボタン名とパラメータを記載します。番号（連番）を付与し、番号によって遷移状態を表現します。

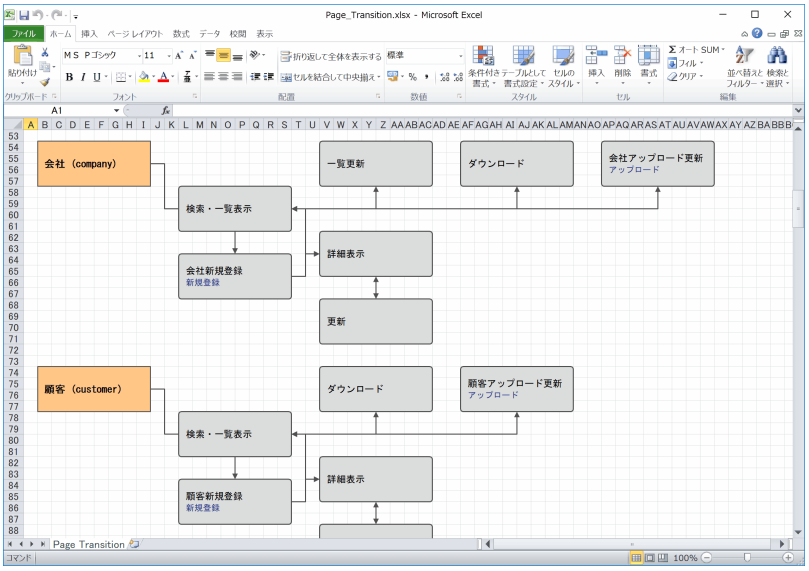


図 画面遷移図

スクリプト一覧表

スクリプト一覧表は、スクリプトの「ドキュメンテーションコメント」を抽出して作成します。

ドキュメンテーションコメントの詳細はこちらをお読みください。

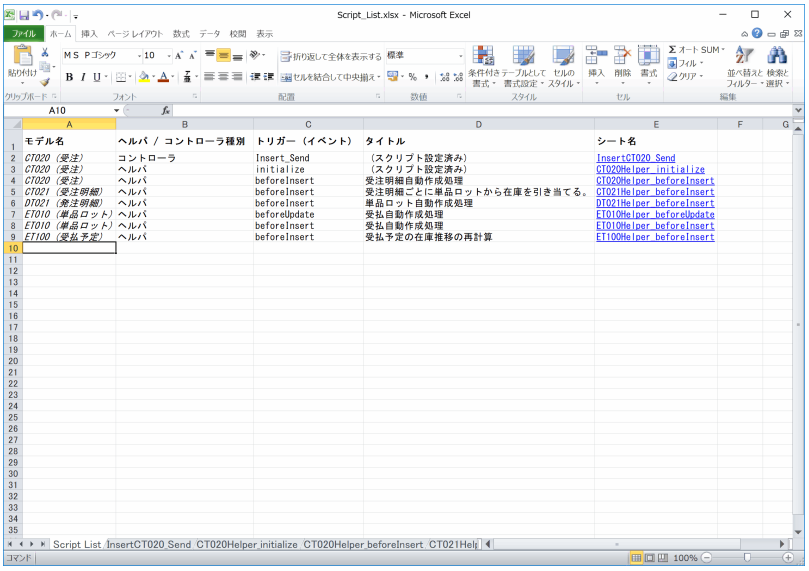


図 スクリプト一覧表

テンプレートファイルを修正する

すべてのドキュメントは Excel 形式で出力されます。テンプレートとなるファイルを修正することで、レイアウトをカスタマイズすることができます。

テンプレートファイルはwagbydesigner¥bin¥spec¥templateフォルダ以下に用意されています。

例 モデル項目の出力欄を追加する

Wagby出荷時は、モデル項目の出力は100項目までとなっています。次の手順でこれを増やすことができます。

1. wagbydesigner¥bin¥spec¥templateにあるmodelitem_list.xlsをExcelで開きます。

2. 標準では、100項目までのテンプレートを用意しています。必要な数だけ、この欄を増やしてください。

例 自動計算式を出力する

modelitem_list.xls で利用できます。次のプレースホルダを含めてください。（標準では含まれていません）

```
${modelitem[番号]/autocalc}
```

例 ユーザ定義入力チェック式を出力する

modelitem_list.xls で利用できます。次のプレースホルダを含めてください。（標準では含まれていません）

```
${modelitem[番号]/generic_check[1-5]}
${modelitem[番号]/generic_check/errormsg[1-5]}
${modelitem[番号]/generic_check/@warning[1-5]}
```

generic_check1からgeneric_check5までを指定できます。 generic_check/errormsgは、エラーメッセージ文字列です。 generic_check/@warningは、警告として利用するかどうかを表します。

autocalc と generic_check は、ここで紹介した簡易表記が使えます。これ以外のリポジトリ情報を使う場合は、次の節をお読みください。

スクリプトを使ったカスタマイズ

先の例を題材に説明します。次の2点を改良することを検討します。

- 先の方法では、値があるときは "int 10 (char)" と出力されるが、ない場合は "int ()"となってしまう。
- テンプレートを修正して、各行にプレースホルダを追加するのが面倒である。

この問題を解決するために、JavaScript言語を使って、出力文字列を加工するプログラムを自作することができます。この方法を説明します。

スクリプトの設定

今回のケースは「モデル項目」の出力に関わるため、スクリプト設定の「モデル項目」をマウスで選択します。するとエディタが起動します。

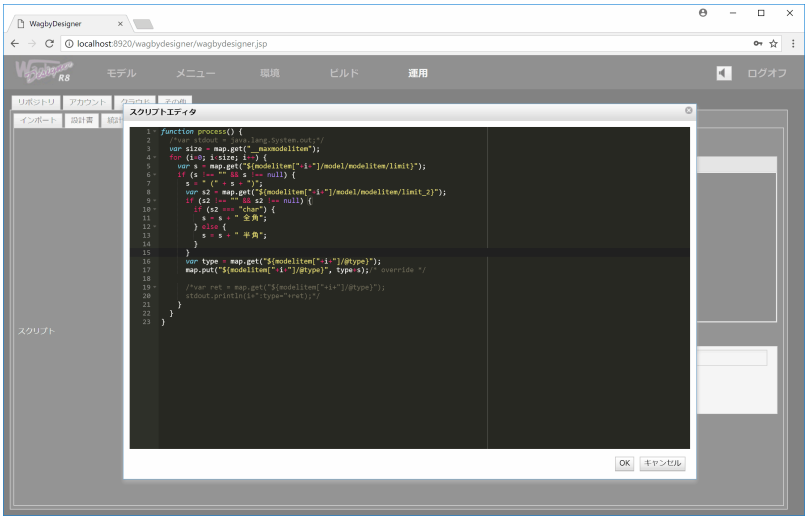


図 スクリプトの入力(1)

エディタにスクリプトを入力します。「OK」ボタンを押すと設定されます。

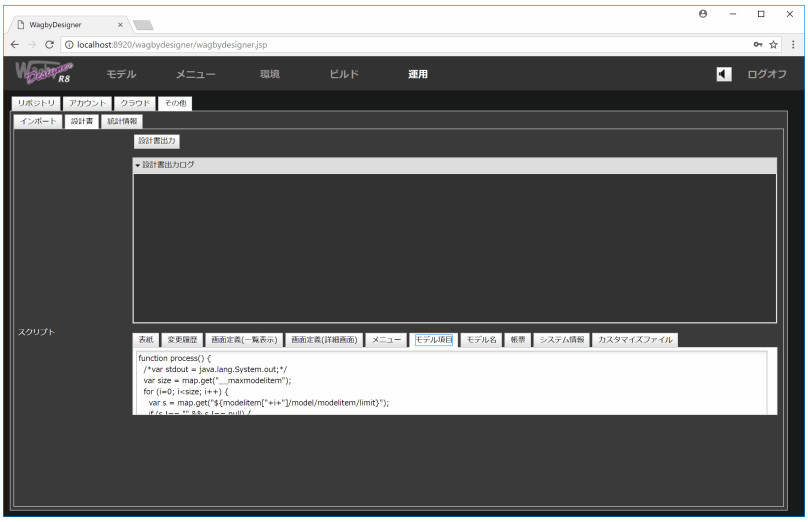


図 スクリプトの入力(2)

例 どのようなプレースホルダが利用できるかを調べる

関数 process を用意します。引数はありません。この関数の内部では map 変数が利用できます。map には、出力しようとしている値がマップ形式で格納されており、この内容を確認します。

```
function process() {
  var keys = map.keySet();
  var it = keys.iterator();
  while (it.hasNext()) {
    var key = it.next();
    var value = map.get(key);
    print(key+"="+value);
  }
}
```

例 型情報の後ろに最大入力文字数と単位（全角/半角）を加える

関数 process を用意します。引数はありません。この関数の内部では map 変数が利用できます。map には、出力しようとしている値がマップ形式で格納されており、これを開発者が加工します。

```
function process() {
  var size = map.get("__maxmodelitem");
  for (i=0; i<size; i++) {
    var s = map.get("${modelitem["+i+"]/model/modelitem/limit}");
    if (s !== "" && s !== null) {
      s = " (" + s + ")";
      var s2 = map.get("${modelitem["+i+"]/model/modelitem/limit_2}");
      if (s2 !== "" && s2 !== null) {
        if (s2 === "char") {
          s = s + " 全角";
        } else {
          s = s + " 半角";
        }
      }
      var type = map.get("${modelitem["+i+"]/@type}");
      map.put("${modelitem["+i+"]/@type}", type+s);/* override */

      /*var ret = map.get("${modelitem["+i+"]/@type}");
      print(i+":type="+ret);*/
    }
  }
```

```
}
```

設計書出力時に、このスクリプトが実行されます。これによって modelitem[N]/@type という値が書き換えられます。具体的には、標準では "string" だった出力値が "string(10) 全角" などとなります。

既存のプレースホルダをそのままに値を再セットするため、テンプレートファイルを修正する手間を減らすこともできます。その他の説明は次のとおりです。

- function() に渡されるのは map という名前の変数のみです。それ以外の変数は開発者が宣言してご利用ください。
- mapには "__" で始まる特別な値が含まれます。これは格納されているキーの個数を示します。[後述]

格納されている数

格納数は "__" ではじまる変数で取得できます。

ファイル名	説明	キー名
list_screen.js	画面定義一覧 - 一覧画面系	__maxcondition : 検索条件の数。condition[N]/@name など。__maxlist : 一覧表示の数。list[N]/@name など。
menu_list.js	メニュー一覧	__maxmenu : メニューの数。menu[N]/@desc など。
modelitem_list.js	モデル項目一覧	__maxmodelitem : モデル項目の数。modelitem[N]/@name など。
model_list.js	モデル名一覧	__maxmodel : モデルの数。model[N]/@modelId など。
report_list.js	帳票一覧	__maxprintlayout : 帳票の数。printlayout[N]/@filename など。
show_screen.js	画面定義一覧 - 詳細画面系	__maxmodelitem : モデル項目の数。modelitem[N]/@name など。

ファイル名に付与される連番の桁数を変更する (8.4.0)

ファイル名の先頭に付与される連番は標準で "0" 詰め3桁です。これを4桁に変更する方法を説明します。

1. wagbydesigner/webapps/wagbydesigner/WEB-INF/classes/build.properties ファイルをテキストエディタで開きます。(Windows OS のメモ帳は利用しないようにしてください。)
2. 次の行を加えます。

```
designreport.docnumformat=%1$04d
```

3. 保存後、Designerを再起動します。

(507)

サンプルのダウンロード

「リポジトリダウンロードサービス > 販売管理」で公開しているリポジトリに、本ツールを適用して得られた設計情報をダウンロードできます。出力結果のサンプルとしてご確認ください。

- サンプルアプリケーション（販売管理）の設計書出力サンプル（R8.3.0利用版）

トラブルシューティング

ファイルを解凍したとき、複数のExcelファイルのファイル名が文字化けする。

(507) この設定を行わない場合の標準フォーマットは %1$03d です。このフォーマットは java.lang.String クラスが提供するメソッド format の第一引数として適用されます。具体的には String.format(フォーマット文字列, 整数値の番号) として用いられます。

お使いの解凍ツールの問題の可能性があります。Windows OS に付属の標準の解凍ツールを使ってお試しください。

実行すると処理が中断され、エラー画面が表示される。

リポジトリのサイズが大きい場合、メモリ不足となっている可能性があります。Designerの稼働環境のメモリサイズを増やしてお試しください。

仕様・制約

- 設計書がダウンロードされない場合、wagbydesinger/temp フォルダにダウンロードされるzipファイルが生成されているかどうかを確認してください。ファイルはあるがダウンロードされない場合、WagbyDesingerを起動したときのパスがbinではない可能性があります。WagbyDesingerはbinフォルダを起点にして起動するようにしてください。
- 設計書出力時は、すべてのモデルを「ビルド対象」に含めてください。ビルド対象外のモデルを指定した場合、設計書出力が正常に動作しないことがあります。 R8.3.0 で解消されました。

本番運用サーバ転送用のzipファイルの作成

createWagbyappZipコマンド

ビルドしたアプリケーション「wagbyapp」フォルダを zip 形式で圧縮します。wagbyapp/binフォルダに含まれるいくつかの実行ファイル（拡張子 .bat または .sh）の実行権限が保持されるという特徴があります。

Windows OS に標準で付属する zip ツールで wagbyapp フォルダを圧縮すると、実行権限が消えるという問題があります。本ツールはこれを解消するために用意されました。

使い方

準備

フルビルドを行い、wagbyapp フォルダを作成しておいてください。以下の手順は本番運用のための wagbyapp フォルダが作成されていることが前提です。

実行

ここでは Windows OS での利用方法を説明します。Linux/Mac OS Xで操作される場合はコンソールをお使いください。

「スタートメニュー>プログラムとファイルの検索」欄に、"cmd" と入力して実行します。

図 スタートメニュー

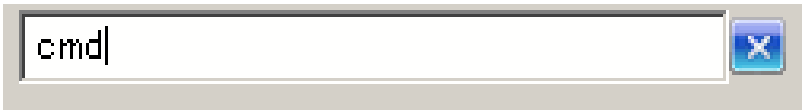


図 cmdを実行する

図に示すような、コマンドプロンプトが表示されます。

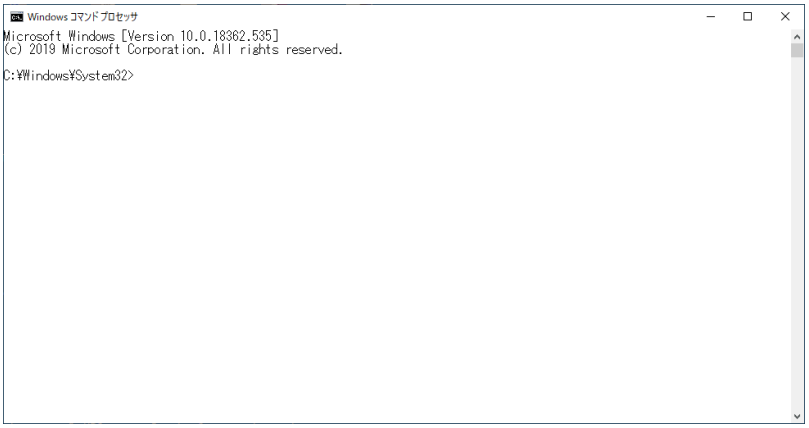


図 コマンドプロンプト

スタートメニューを経由せず、直接「コマンドプロンプト」を実行することもできます。

コマンドプロンプトが起動したら、cd コマンドでカレントフォルダを移動します。移動先は Wagbyインストールフォルダ¥misc です。（インストール先フォルダは適切に読み替えてください。）

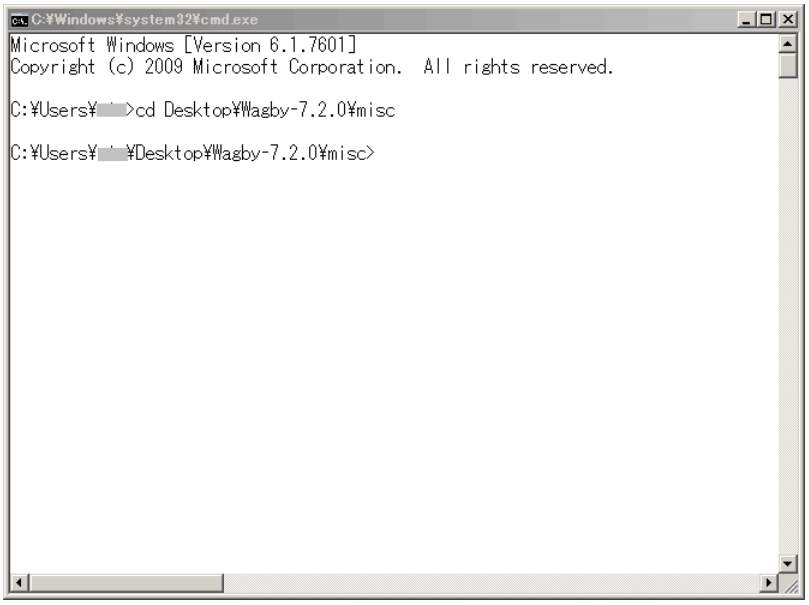


図 カレントフォルダを変える

同フォルダに含まれる createWagbyappZip.bat を実行します。

```
createWagbyappZip.bat
```

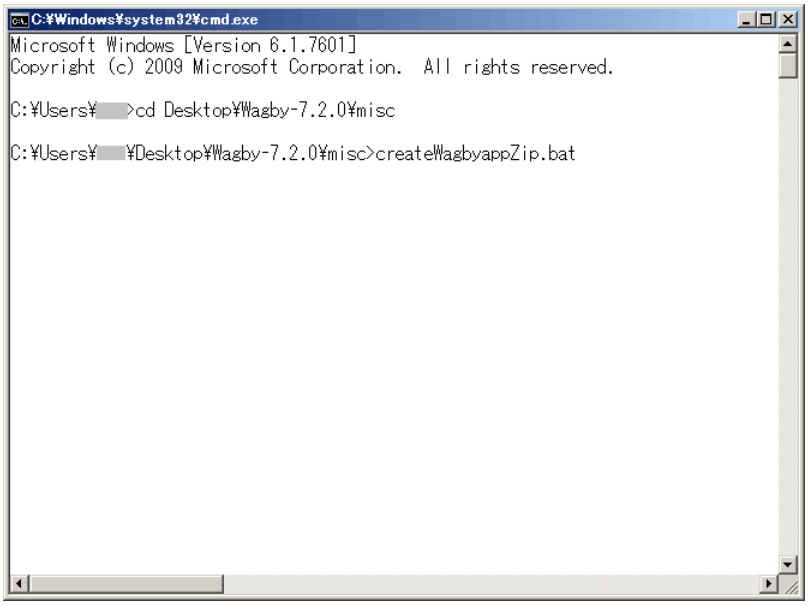


図 createWagbyappZip.batを実行する

Wagbyインストールフォルダに　wagbyapp.zip　が用意されます。このファイルを本番運用サーバへコピーし、展開してください。

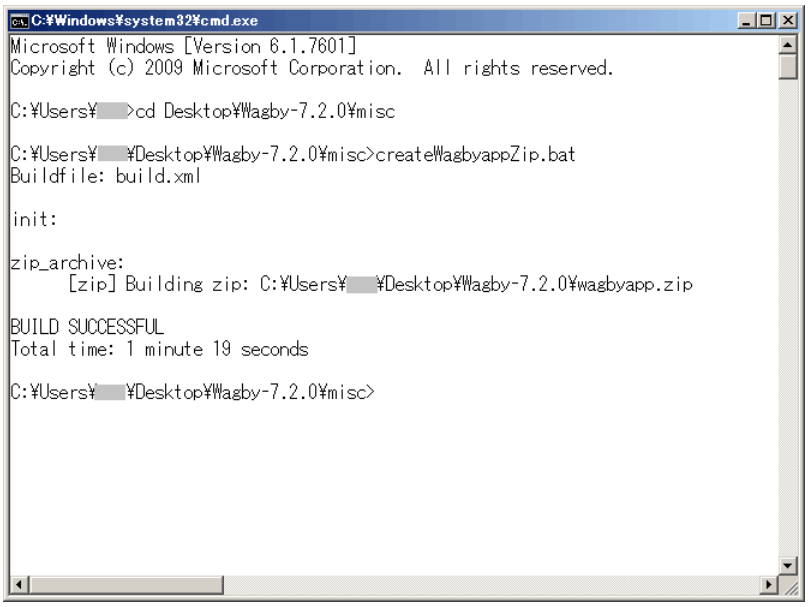


図 wagbyapp.zipが作成される

仕様

- このコマンドは内部で misc/lib フォルダに含まれている Apache Ant のライブラリを使っています。そのため開発者の方で別途、Ant を用意する必要はありません。
- クラスファイルが65,536を超える場合は、R8.3.2 以降に同梱されている misc/build.xml をご利用ください。

既存テーブル定義からリポジトリを生成する

概要

既存システムのリレーショナルデータベースからテーブル定義情報を読み込み、Wagby のリポジトリを半自動生成することができます。 読み込むことができるのは、次の情報です。

- テーブル名
- 項目名
- 項目の型
- 外部キー

Wagby が対応している、すべてのデータベースを利用できます。さらに Microsoft Access の MDB ファイルにも対応しています。

使い方

「運用 > その他」のタブにある「インポート」を選択します。

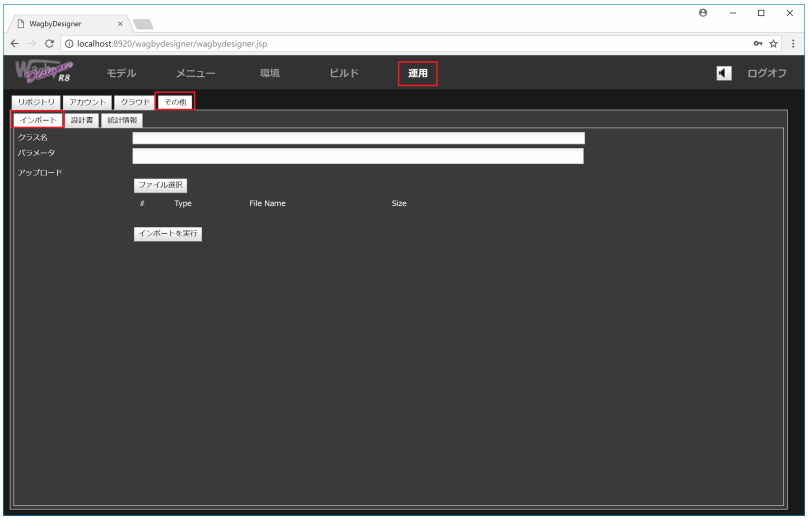


図 リポジトリインポート画面

1. JDBCドライバファイルの配置

   Wagby-8.x.x¥customize¥tomcat¥lib に、移行対象となるDB向けのJDBCドライバファイルを置きます。

2. 接続先データベースを指定する

   「パラメータ」の欄に、接続先データベースの情報を指定します。図に例を示します。

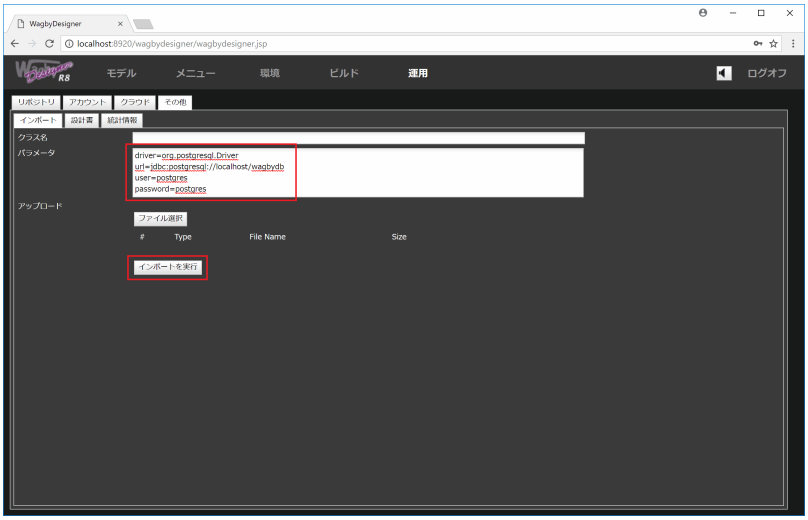


図 接続先データベースの設定

```
driver=org.postgresql.Driver
url=jdbc:postgresql://localhost/wagbydb
user=postgres
password=postgres
```

上記パラメータにそれぞれ、適切な値を指定します。

もう一つの方法 ： サブデータベースを指定する

サブデータベースの設定を再利用することもできます。図に、サブデータベースの設定例を示します。

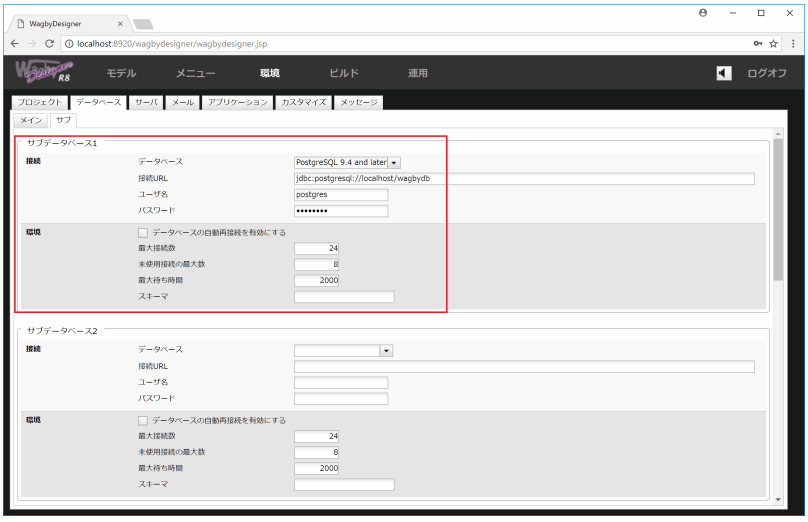


図 サブデータベースの設定(1)

サブデータベース1 の情報を利用する場合、パラメータに ″subdatabase1″ または省略形である ″s1″ と指定します。

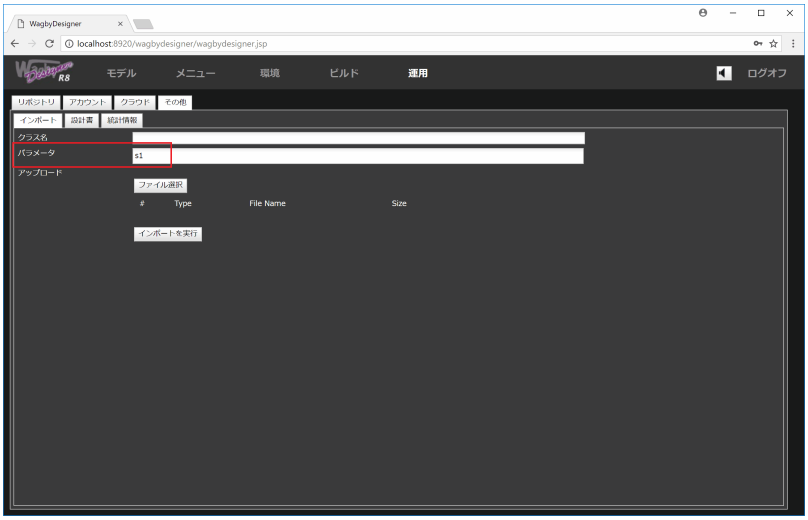


図 サブデータベースの設定(2)

(508)

実行

「インポートを実行」ボタンを押すと、接続先データベースのすべてのテーブル定義を読み込みます。

指定できるパラメータ

リポジトリのインポートでは次のパラメータを指定することができます。パラメータは「キー=値」の形式で指定します。複数記述する場合は、改行で区切ります。

パラメータ	説明	例
driver	JDBCドライバクラスを指定します。driver にはファイル名ではなく Java のクラス名を指定してください。	driver=org.postgresql.Driver
url	データベースの接続先情報を指定します。詳細は利用するデータベースのJDBC接続手順方法に関するドキュメントをお読みください。（製品ごとに記述が異なります。）	url=jdbc:postgresql://localhost/wagbydb
user	データベース接続に必要なアカウントを指定します。	user=postgres
password	user指定に対応したパスワードを指定します。	password=postgres
subdatabseXX	環境>データベースの設定にある、サブデータベースの接続情報を再利用します。XXの部分には、サブデータベースの設定番号が入ります。この指定を行う場合、driver,url,user,passwordは指定しないようにしてください。	subdatabse1
sXX	subdatabseXX の省略形です。	s1
includeModel	取り込むテーブル名を指定します。複数指定する場合はコンマで区切りま	includeModel=A,B

(508) サブデータベース2 の情報を利用する場合は "subdatabase2" または省略形 "s2" と指定します。

	す。excludeModel との併用はできません。記載したテーブル名は正規表現として扱われます。そのため A と記載すると、テーブル A1 や A2 もマッチします。	
excludeModel	取り込みを除外するテーブル名を指定します。複数指定する場合はコンマで区切ります。includeModel との併用はできません。記載したテーブル名は正規表現として扱われます。そのため A と記載すると、テーブル A1 や A2 もマッチします。	excludeModel=C
ignoreNotnull	移行対象テーブルが NOT NULL DEFAULT となっていた場合、取り込み後の定義はすべて必須項目になります。これを避けるための設定です。(*1) 複数指定する場合はコンマで区切ります。正規表現を使うこともできます。	ignoreNotnull=A,B
keepType	テーブル定義の列の型定義を「テーブルの物理型」設定に転記するようにします。デフォルトはfalseです。	keepType=true
debug	trueを指定すると、Tomcat コンソールにデバッグ用のメッセージを表示します。(Designerの画面には表示されません。)	debug=true

(509)

Microsoft Excel の場合

Microsoft Excel からリポジトリを生成することができます。詳細は「チュートリアル > ドラッグ&ドロップによるアプリケーション開発」をお読みください。

Microsoft Access の場合

Microsoft Access のファイルからリポジトリを生成する場合は、次の手順で行ってください。

JDBCドライバファイルの配置

次のファイルを Wagby-8.x.x¥customize¥tomcat¥lib に保存します。

- hsqldb.jar
- jackcess-2.1.2.jar
- ucanaccess-3.0.2.jar

ここからダウンロードできます。

- RepositoryConverterAccessDriver.zip [byte]

パラメータの指定は不要

(509) これによって Wagby では必須項目と扱われませんが、実テーブルは NOT NULL であるため、空欄の項目で入力しても空白に置換させる処理を加える必要があります。

パラメータの指定は不要です。空白としてください。

MDBファイルをアップロードする

対象となる MDB ファイルをアップロードし、「インポートを実行」ボタンを押下してください。

仕様

- モデルの検索、一覧表示、詳細表示、登録、更新、削除を行う機能が有効になります。
- 検索と一覧表示は同一画面として扱います。また一覧表示はグリッド形式となります。
- 検索画面への遷移を行うメニューボタンが用意されます。メニューボタンは「サービス」タブに含まれます。
- 型は「文字」「数字」「日付」「モデル参照（検索）」を判別します。モデル参照は、外部キー指定(foreignkey) の場合に有効です。
- DBのカラムサイズを判断材料に使います。文字列型で255文字以上なら、テキストエリアとして扱います。
- 浮動小数点型の場合、桁数情報があればそれを使い、入力フォーマットに反映させます。
- NOT NULL 制約を引き継ぎます。
- 日付型は登録画面を開いたときの初期値に関数 TODAY を、日付時刻型は関数 NOW を指定します。
- モデルID(英語)と項目ID(英語)が日本語の場合、それぞれ「modelXX」「itemXX」を自動で設定します。
- モデルID(英語)と項目ID(英語)がWagbyの名前付ルールに合致しないとき、それぞれ「modelXX」「itemXX」を自動で設定します。

手動で修正すること

- 外部キーはすべて「モデル参照（検索）」となってしまいますが、実際には繰り返しコンテナまたは外部キー（親子モデル関係）として扱うことが適切なケースがあります。自動判定できないため、開発者がこれらの構造を適切に見直してください。
- 複合キーの場合の「モデル参照（検索）」も同様です。
- モデルIDや項目IDに日本語が設定される場合があります。これらは手動で修正し、英数字に置き換えてください。
- 入力チェックや出力フォーマットなどは適切に修正してください。

謝辞

Microsoft Access のファイルを読み込むため、オープンソースの UCanAccess を利用しています。

変更履歴

R8.0.5

- テーブル名と項目名が日本語の場合、これを物理テーブル/カラム名に反映させるようにしました。
- Oracle からのテーブル定義読み込み時、NOT NULL制約が引き継がれない（全て必須項目になってしまう）問題を修正しました。
- Oracle からのテーブル定義読み込み時、コメントとして記載している項目名日本語を読み込むようにしました。

kintoneの設計情報からリポジトリを生成する

概要

サイボウズ社が提供する「kintone」アプリケーションの設計情報を、本ページに記載する方法で記述することで、Wagbyのリポジトリへと変換できます。

用意するファイル	必須	説明

モデルID.csv	○	1ファイル=1モデル定義です。
モデルID-コンテナ名.csv	-	モデルに含まれる明細情報の定義を行います。
master.csv	-	Wagbyの選択肢モデルに該当します。

次節より、それぞれのファイルの詳細なフォーマットを説明します。

モデルID.csvファイル

「モデルID.csv」ファイルは、1ファイルで1つのモデル定義を構成します。内容は次のとおりです。

列	説明
label	項目名。日本語表記。
type	型。(後述)
code	項目ID。英数字表記。

先頭行はヘッダとして扱われます。「label,type,code」と記載してください。 2行目以降から、モデル項目情報を記載します。例を示します。

```
label,type,code
氏名,SINGLE_LINE_TEXT,ApplicantName
氏名かな,SINGLE_LINE_TEXT,ApplicantNameKana
```

メニューへの配置

モデルID.csv に用意されたモデルは、メニュータブ「サービス」内に配置されます。

その他のルール

- ファイル名はモデルIDとなるため、Wagbyが「モデルIDとして利用できる名前」ルールに準拠します。例：customer.csv。詳細は"モデルIDの命名規則"をお読みください。
- Wagbyの「モデル名（日本語)」は、モデルIDと同じとなっています。変換後、Wagby Designer を用いてモデル名を指定してください。
- 「label」をシングルクォートで囲っていた場合、シングルクォートを除いた部分が有効になります。例："氏名"と記載していた場合、前後のシングルクォートは除かれます。
- 「code」は英数字表記としてください。日本語で表記されていた場合、内部で自動的に「item+数字(連番)」に置換されます。
- 「code」はモデル項目IDとなるため、Wagbyが「項目IDとして利用できる名前」ルールに準拠します。詳細は"項目IDの命名規則"をお読みください。

「型」について

「モデルID.csv」ファイルに記載する「型」は次のいずれかを指定してください。

表記	Wagbyへ取り込む場合の型
RECORD_NUMBER	数値（整数)、主キー。
SINGLE_LINE_TEXT	文字列型
NUMBER	数値（整数)
MULTI_LINE_TEXT	文字列型（テキストエリア)
RICH_TEXT	文字列型（テキストエリア・リッチ)
CHECK_BOX	選択肢モデルの参照（チェックボックス）※後述
RADIO_BUTTON	選択肢モデルの参照（ラジオボタン）※後述
DROP_DOWN	選択肢モデルの参照（リストボックス）※後述
FILE	文字列型（ファイル)
LINK	文字列型（URL)
DATE	日付型
TIME	時刻型
DATETIME	日付時刻型

CREATOR	固定値 - 登録ユーザ
CREATED_TIME	固定値 - 登録時刻
MODIFIER	固定値 - 更新ユーザ
UPDATED_TIME	固定値 - 更新時刻
SUBTABLE	繰り返しコンテナ ※後述

(510) (511)

選択肢モデルの扱い

master.csv

Wagbyの選択肢モデルを用意することができます。 ファイル名を「master.csv」とします。内容は次のとおりです。

列	説明
label	項目名。日本語表記。
type	型。
code	項目ID。英数字表記。
options	選択肢の値。

例を示します。1行目は「label,type,code,options」というヘッダ行とします。2行目以降に実際のデータを記述していきます。

```
label,type,code,options
性別,DROP_DOWN,Gender,男
性別,DROP_DOWN,Gender,女
職業,DROP_DOWN,Job,会社員
職業,DROP_DOWN,Job,学生
職業,DROP_DOWN,Job,主婦
```

このように、label,type,code が同じで、optionsが異なる行を連ねてください。

モデルID.csv との関係

モデルID.csvのcode部と、master.csvのcode部が一致していた場合、当該項目と (master.csvで用意された)選択肢モデルとの紐付けが行われます。

例えば customer.csv に次の行があった場合：

```
性別,DROP_DOWN,Gender
```

master.csvの "Gender" と紐づくため、「男」「女」という選択肢を、リストボックス形式で選べるようになります。

メニューへの配置

master.csv に用意された選択肢モデルは、メニュータブ「マスタ管理」内に配置されます。

サブテーブルの扱い

kintoneの「サブテーブル」は、Wagbyの「繰り返しコンテナ」として移行されます。

モデルID-コンテナ名.csv

「コンテナ名」は英数字になります。

(510) RECORD_NUMBER が主キーとなります。一つだけ用意してください。

(511) 上記で示した以外の形式は、未サポートです。

列	説明
fields_name	モデルID.csv に記載された項目名。
label	項目名。日本語表記。
type	型。
code	項目ID。英数字表記。

例を示します。ファイル名を customer-operator.csv とします。 1行目は「fields_name,label,type,code,options」というヘッダ行とします。2行目以降に実際の定義を記述していきます。

```
fields_name,label,type,code
OperatorTable,操作日時,DATETIME,OperatorDate
OperatorTable,操作内容,MULTI_LINE_TEXT,OperatorMemo
```

モデルID.csv との関係

モデルID.csvの型が SUBTABLE で、かつcode部と、モデルID-コンテナ名.csvのfields_name部が一致していた場合、モデルID.csvの当該位置に、サブテーブル項目が「繰り返しコンテナ」として挿入されます。

例えば customer.csv に次の行があった場合：

```
,SUBTABLE,OperatorTable
```

customer-operator.csv の "OperatorTable" として記載された 4 つの項目が、customer.csv の当該位置に含まれるようになります。

注意事項

- サブテーブル定義（上記では customer-operator.csv）の code 部は、Wagbyの「項目ID」に相当するため、重複してはなりません。具体的には、モデルID.csv のどの項目名（code部）とも重複しないようになっている必要があります。

使い方

「運用 > その他」のタブにある「インポート」を選択します。

パラメータ欄に "kintone" と記述します。上で説明した csv ファイルを圧縮した zip ファイルをアップロードし、「インポート実行」ボタンを押下します。

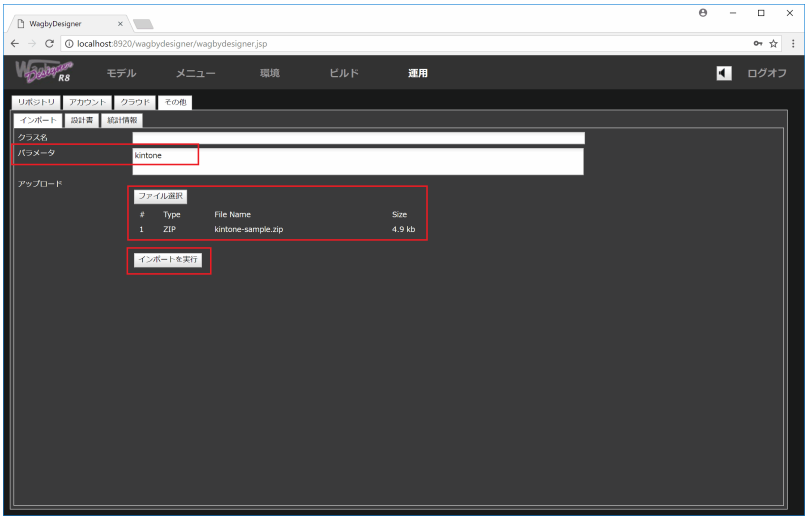


図 リポジトリインポート画面

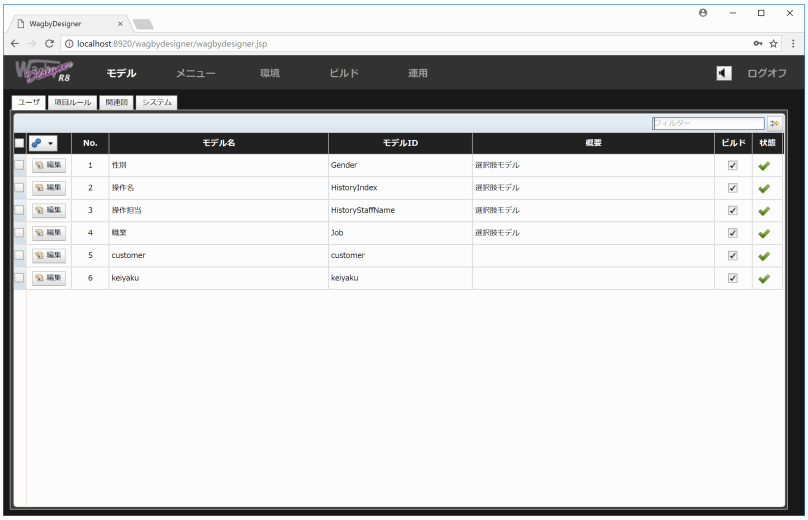


図 リポジトリが読み込まれた

サンプルの設定情報

kintone_sample.zip を用意しています。お試しください。

手動で修正すること

- フィールド長や NOT NULL 制約、式などは移行されません。Wagby Designer にて改めて設定してください。
- モデルID(英語)と項目ID(英語)が日本語の場合、それぞれ「modelXX」「itemXX」を自動で設定します。XXは数字(連番)です。
- モデルID(英語)と項目ID(英語)がWagbyの名前付ルールに合致しないとき、それぞれ「modelXX」「itemXX」を自動で設定します。XXは数字(連番)です。

ファイル名の扱いを「オリジナルファイル名を残す」方式に変更する

概要

システム運用の途中で「ファイル名にオリジナルファイル名をそのまま使う」ように設定を変更した場合、その前まで登録されたファイルは拡張子が .tmp のファイル名のまま残り、変更後に登録したファイルはオリジナルファイル名として保存されるようになります。

ここでは、過去に登録された .tmp ファイル名を、オリジナルファイル名に戻す方法を説明します。

準備

これから説明する操作を行う前に、データのバックアップならびに upload_dir フォルダのバックアップを行っておいてください。処理に失敗した場合、速やかに元に戻せるようにします。

また、本設定（オリジナルファイル名を使う）を行う場合、本番機の OS は Windows である必要があります。

操作方法

1. 管理者でログオンし、対象となるモデルに関するデータをエクスポートします。このとき、"Zip形式で扱う" を無効にしてエクスポートしてください。

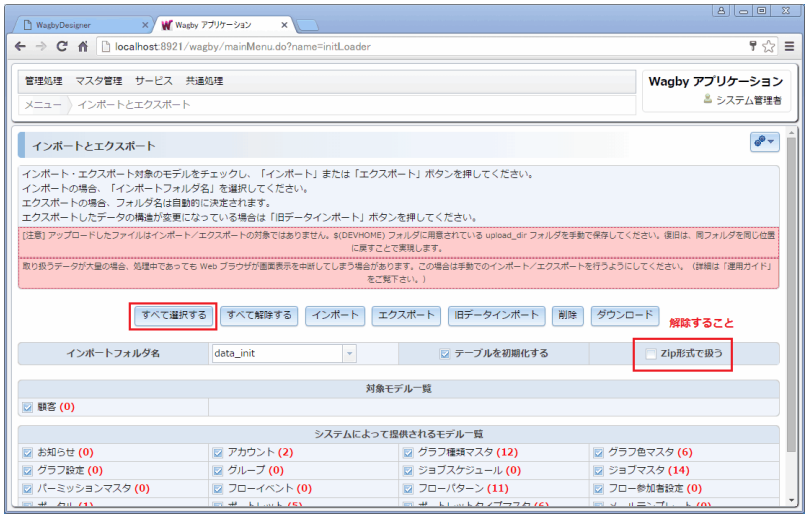


図 Zip形式で扱うを無効にしてエクスポートする

2. Wagby で構築した Web アプリケーションを停止します。
3. wagbyapp¥binフォルダにて、次のコマンドを実行します。

```
java -jar UploadFilenameConverter.jar
```

4. すると同ディレクトリに「renamefile.bat / renamefile.sh」ファイルが生成されます。また、export/data_changed フォルダが生成されます。
5. 生成された renamefile.bat (Linux 系では .sh) を実行します。.tmp ファイルがオリジナルファイル名に変更されます。なお、実行前に同ファイルの内容をテキストエディタで確認するとよいでしょう。
6. Web アプリケーションを起動します。
7. 管理者でログオンし、export/data_changed 以下のモデルをインポートします。

仕様・制約

- モデル内に登録されたファイル名が重複していた場合、一つにまとめられます。
- 本ツールは Wagby Cloud には適用できません。
- 「Zip 形式でのエクスポート」には対応していません。本機能は非 Zip 形式のエクスポートフォルダに対して行ってください。

Designerにログオンできるアカウントの追加・変更・削除

アカウントの扱いについて

Unlimited Enterprise の場合

Designerにログオンできるアカウントの追加・変更・削除が可能です。登録するアカウントの上限はありません。一つの Designer に複数名の開発者がログオンし、モデルを編集できるようになります。

Unlimited Enterprise ではない場合

Designerにログオンできるアカウントは "user" 固定となります。アカウント名の変更や削除を行うことはできません。一度に Designer にログオンできる開発者は一名のみとなります。

(512)

(512) このとき "運用 > アカウント" タブ自体が表示されません。

新しいアカウントの登録

1. 「運用」の「アカウント」タブを開きます。

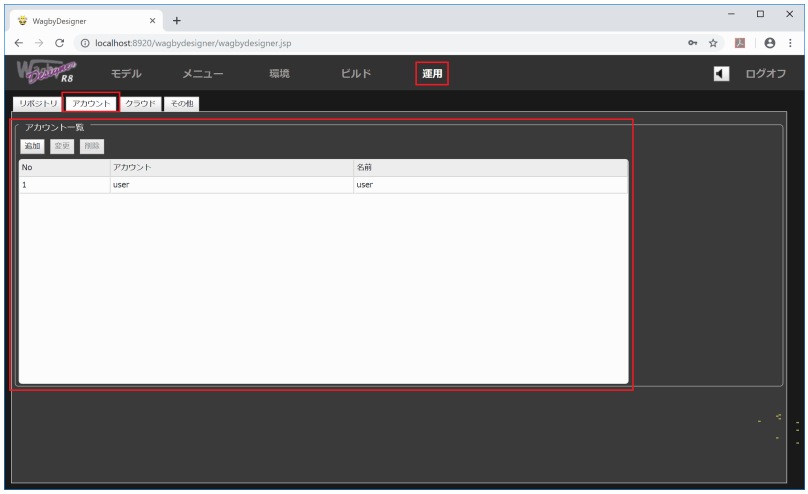


図 運用>アカウントタブ

2. 新しいアカウントを追加します。「追加」ボタンを押下します。するとアカウント情報ダイアログが表示されます。

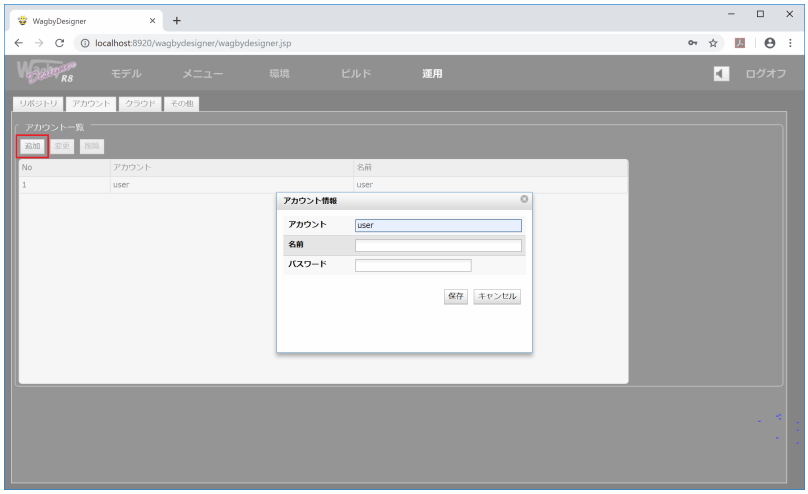


図 アカウントの追加(1)

3. アカウント情報ダイアログで新しい「アカウント」「名前」「パスワード」を入力して「保存」ボタンを押下します。

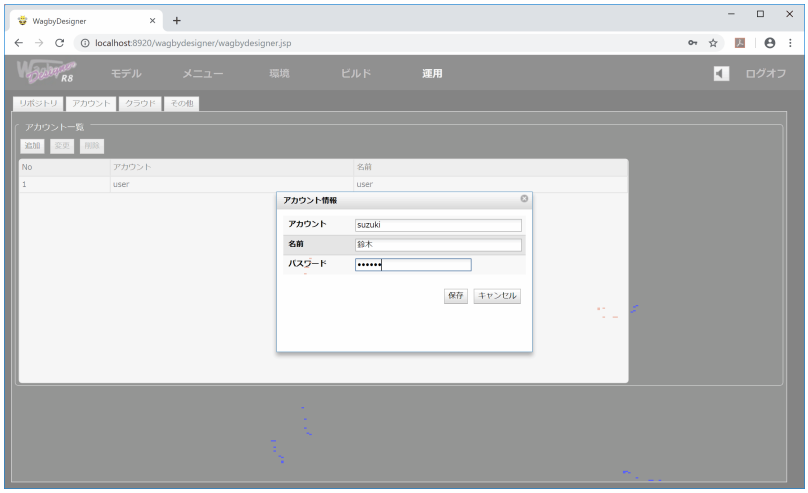


図 アカウントの追加(2)

4. 新しいアカウントが追加されました。

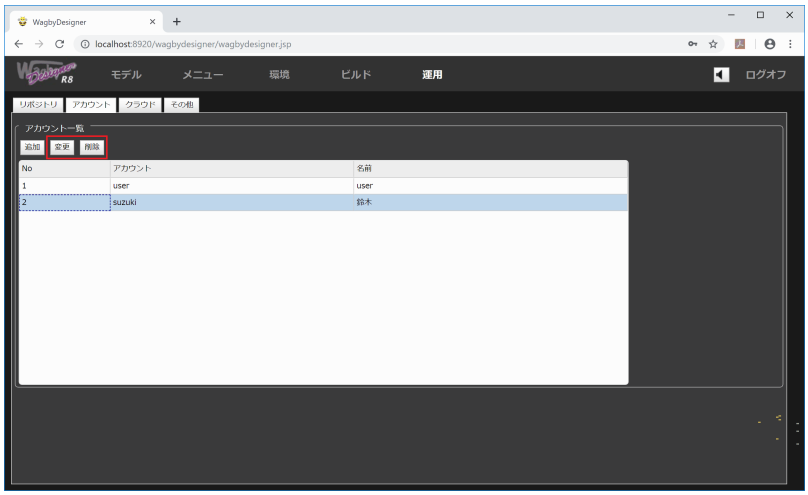


図 アカウントの追加(3)

5. いったんログオフし、新しいアカウントで Designer にログオンします。ビルドタブ内の開発者に、現在のアカウントが表示されています。

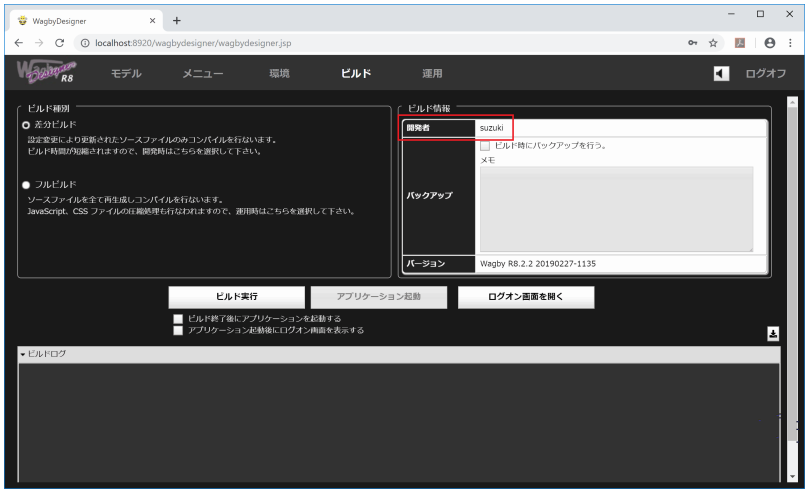


図 開発者アカウントの確認

アカウントの変更

図で対象アカウントを選択します。すると「変更」ボタンが押せるようになります。上述の説明と同じくアカウント情報ダイアログを操作して変更してください。

アカウントの削除

図で対象アカウントを選択します。すると「削除」ボタンが押せるようになります。削除ボタンを押すとアカウント情報が削除されます。

仕様

- コントロールキーを押下することで削除対象アカウントを複数選択できます。
- 登録済みのアカウントが1名のみの場合は削除することはできません。2名以上登録されている場合に削除機能が有効になります。

Unlimited Enterprise ではない版でのアカウントとパスワードの変更方法

Unlimited Enterprise ではない場合に、アカウントのパスワードを変更する方法を説明します。

application-customize.propertiesの編集

Designer に含まれる次のファイルをテキストエディタで開きます。

```
wagbydesigner/webapps/wagbydesigner/WEB-INF/classes/application-customize.properties
```

標準では次のようになっています。

```
#designer.username=new_user
#designer.password=$2a$12$ocl/e/uliBuf9Umugvup2ejonLAchuZMCYB6KfFDc3bZBHi8UvCWO
```

ここで先頭にある文字 "#" はコメントを意味します。1行目と3行目のコメントを取り除きます。

```
designer.username=suzuki
designer.password=(...Bcyptでハッシュ化されたパスワード...)
```

1行目はログオンアカウントのIDです。イコールの右側を書き換えることができます。（英数字のみ、漢字などは利用不可）

3行目はパスワードです。Bcrypt でハッシュ化された値を指定してください。Bcrypt パスワードは次のようなサイトで作成することができます。

- http://www.tekboy.net/bcrypt-calculator
- https://www.browserling.com/tools/bcrypt

(513)

BCrypt 形式でないパスワードを記述した場合はログオンに失敗します。system.log には次のようなログが出力されます。 [WARN org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder matches] Encoded password does not look like BCrypt

起動時のオプション指定

Designer 起動時のオプションに-Dspring.profiles.active=customizeを追記します。具体的にはsetenv.bat (Unix系OSであれば setenv.sh) の環境変数 CATALINA_OPTS の設定に追記します。その後、Designer を起動してください。

パスワードのハッシュ値を取得する

PasswordEncoderによるカスタマイズを行う場合

Spring が提供する PasswordEncoder の仕組みに準拠した、パスワードハッシュ化のカスタマイズを行うことができます。[詳細...] 8.2.2

利用できるハッシュ化アルゴリズム

「アカウント ＞ パスワード ＞ パスワードハッシュ化」は使われません。カスタマイズした内容が使われます。

PasswordEncoderによるカスタマイズを行わない場合

R8.2.1 までの Wagby をご利用、または R8.2.2 以降の Wagby でも PasswordEncoder を使ったパスワードハッシュ化のカスタマイズを行っていない場合は、製品に同梱されている gethash コマンドを使って、ハッシュ値を事前に求めることができます。

(513) ストレッチング回数(Rounds) は 4 以上の値を指定してください。数値が大きいほどセキュリティの強度が向上します。

BCryptへの対応 (8.3.0)

gethashコマンドは BCrypt には未対応です。BCrypt については、次のようなサイトを利用してハッシュ値を作成してください。

- http://www.tekboy.net/bcrypt-calculator
- https://www.browserling.com/tools/bcrypt

ハッシュ値の参考

文字列 "wagby" と "admin" について、gethash コマンドで求められた値を掲載しておきます。これらの文字列であればコマンドを使わずとも、下記の内容をそのままご利用いただけます。

文字列 "wagby" のハッシュ値

アルゴリズム	値
md5	ri49lYC9pPakRAjwacg7XO
sha-1	rguixRbEDEZQGhx2CJydZd2zGuE
sha-256	SfXpY49wuPzTjccA4w6zjvjvPbLwdV4zhPm3euckgvg
sha-384	Z9h2ko-3w-iFgzsHGnzloWrkmaNoNaxAGhqiPLFHxASpTl4O57PLb2IBNUHD4DFB
sha-512	YdK+GSi6R6ZVlVz5cPWLKKqO5tFPls6Bg7mCjsTI4ZsO7ZQRvoDrPFgH6QW

文字列 "admin" のハッシュ値

アルゴリズム	値
md5	8icLanFnFqt3ykEeiEOvMM
sha-1	Q3fyaKd8HHlC3Y8k2KMRxgNdGps
sha-256	z6BSVrl111mZWgyZjuUlTX5DGsxP-4KUG1ZLaHh8Ghw
sha-384	DaqkGga5M3h3b9lggxJXDrRs3QJCs-1vrrPBw5aW8egAw46lrKz9EKOv
sha-512	NWR4OWRSaBSwF5bVW5jZMu3DFiEUOlYzY+GNSOKj7thOoQTWNNPjjHMRQmGT+UGgO1+nkhfmNRedNFToTF1TXO

gethashコマンドの使い方

ここでは Windows OS での利用方法を説明します。Linux/Mac OS Xで操作される場合はコンソールをお使いください。

「スタートメニュー>プログラムとファイルの検索」欄に、"cmd" と入力して実行します。

図 スタートメニュー

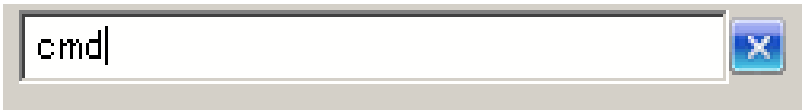


図 cmdを実行する

図に示すような、コマンドプロンプトが表示されます。

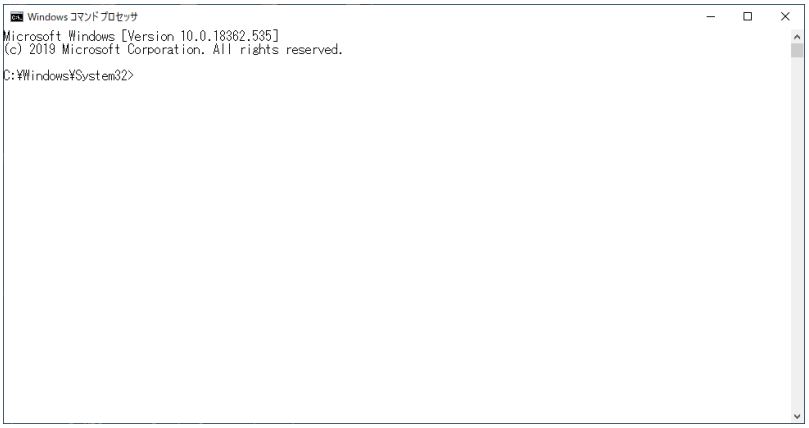


図 コマンドプロンプト

スタートメニューを経由せず、直接「コマンドプロンプト」を実行することもできます。

コマンドプロンプトが起動したら、cd コマンドでカレントフォルダを移動します。移動先は Wagbyインストールフォルダ¥wagbydesigner¥bin です。（インストール先フォルダは適切に読み替えてください。）

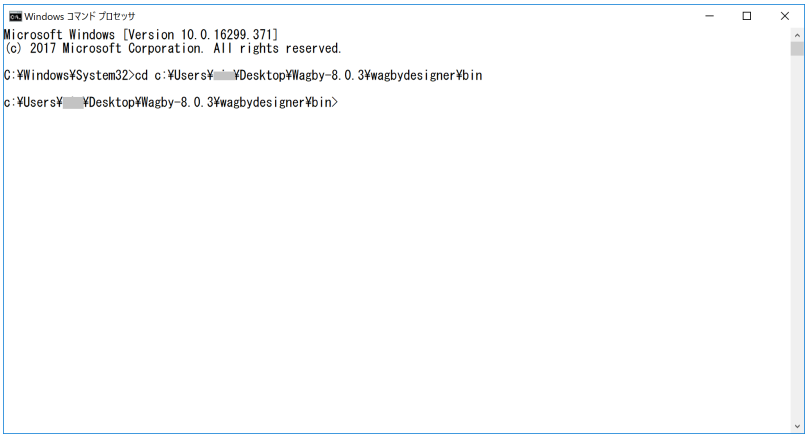


図 カレントフォルダを変える

次のようにコマンドを実行します。

```
gethash [パスワード] [ハッシュアルゴリズム]
```

例えば "admin" という文字列のハッシュ値を sha-1 アルゴリズムを使って取得する場合は次のようになります。

```
gethash admin sha-1
```

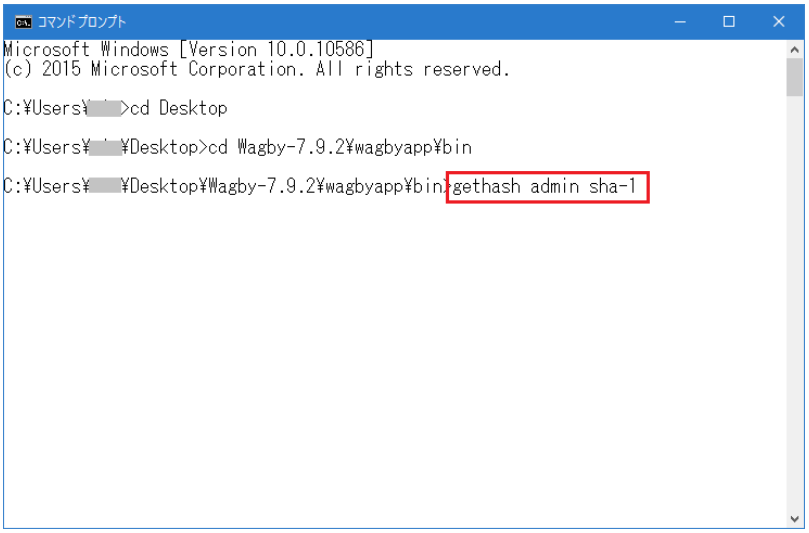


図 コマンドの実行

実行結果が表示されます。

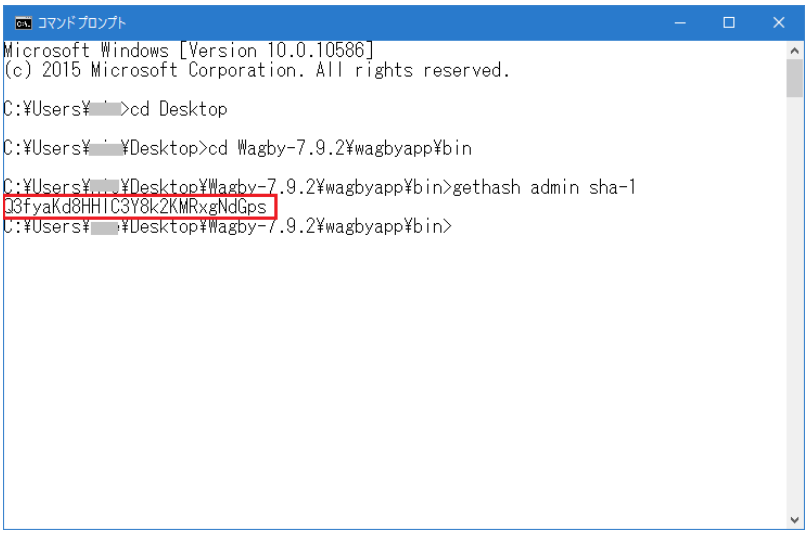


図 実行結果

指定できるハッシュ化アルゴリズム

「アカウント > パスワード > パスワードハッシュ化」と同じです。

ハッシュ値について

ハッシュ値の変換処理は不可逆変換のため、ハッシュ値から元の値を導き出すことはできません。

関連するページ

- アカウント > パスワード > パスワードを忘れたとき

統計情報

実行方法

準備

統計情報を出力する前に、ビルド処理を行ってください。直前にビルドされた状態の統計情報を出力します。

統計情報の作成

「運用 ＞ その他 ＞ 統計情報」タブを開きます。「統計情報を更新」ボタンを押下すると、最新の統計情報が作成されます。

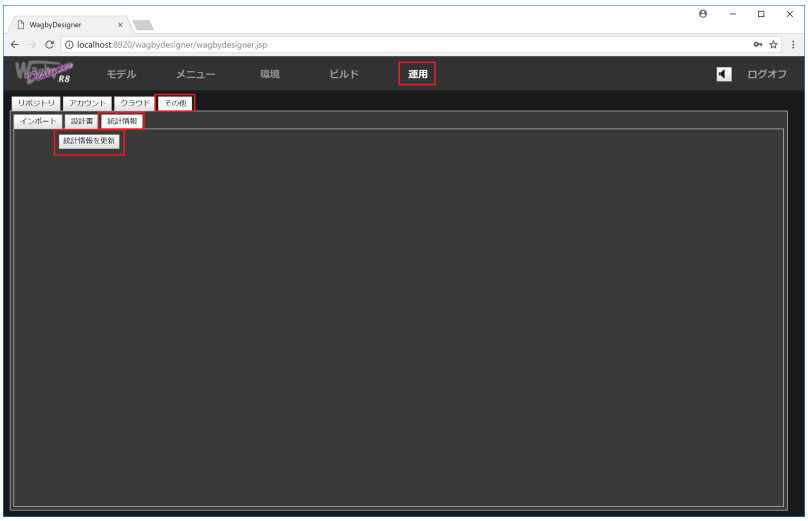


図 統計情報を更新する

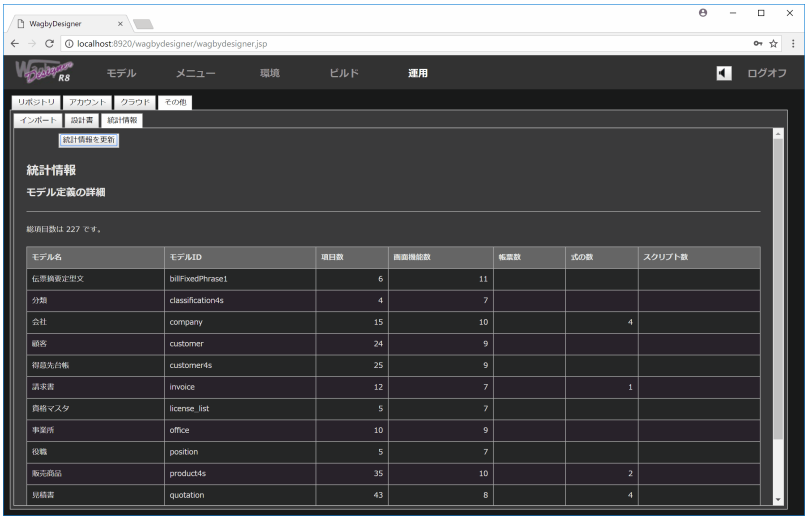


図 統計情報(1)

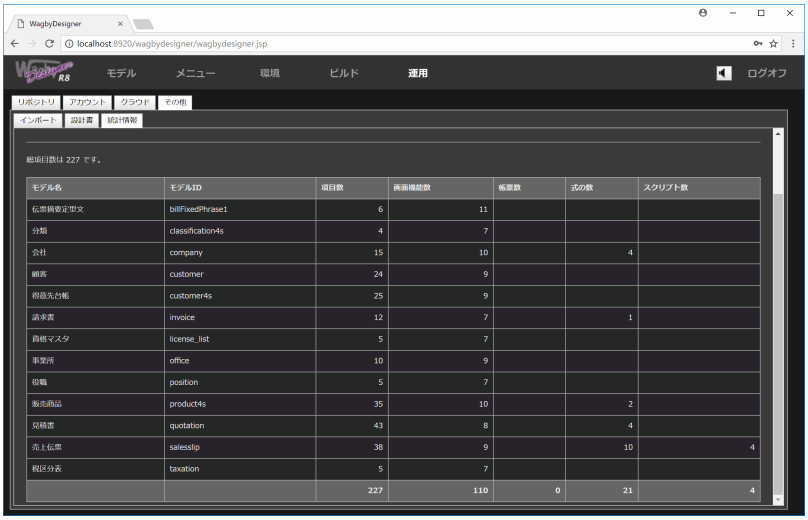


図 統計情報(2)

内容の詳細

統計情報には次の内容が含まれます。

- モデルごとの項目数
- モデルごとの画面機能数
- モデルごとの帳票数
- モデルに含まれる「計算式」の数
- モデルに設定された「スクリプト」の数

関連するページ

この統計情報は >製品 > Wagbyを用いた見積手法″に適用することができます。

影響分析

実行方法

モデル項目の詳細ダイアログから「影響分析」タブを開きます。

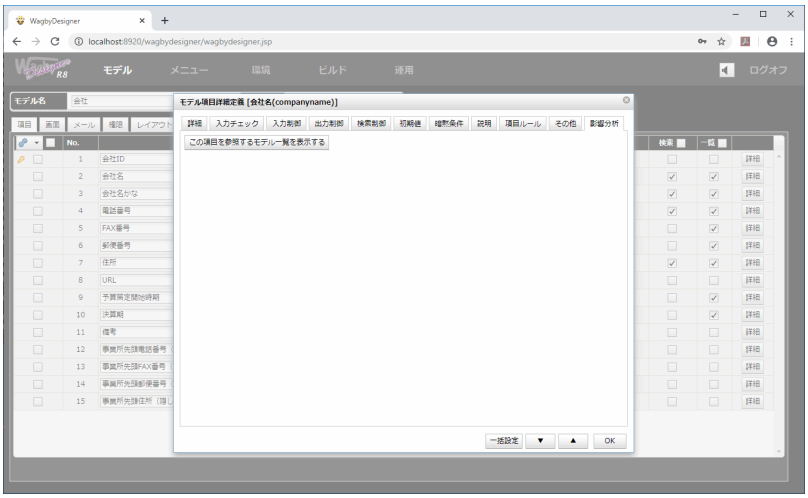


図 影響分析(1)

「この項目を参照するモデル一覧を表示する」ボタンを押下します。

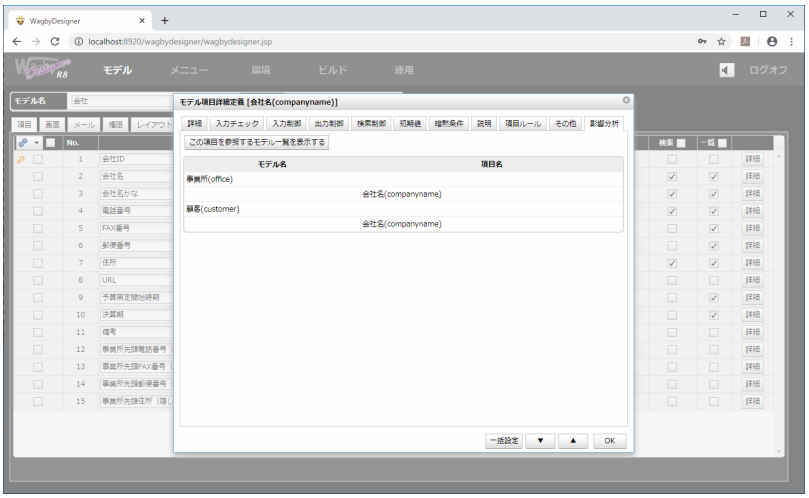


図 影響分析(2)

仕様・制約

- モデル項目の関連情報を確認しています。
- 計算式内部で指定した場合や、スクリプトで利用されているもの、カスタマイズコード内で利用されているものは除きます。（これは将来のWagbyで対応する計画です。）

リポジトリ構造とチーム開発

モデル項目の詳細

項目の並び順

モデル一覧はモデルIDの辞書順でソートされますが、モデル項目は定義した順で維持されます。

内部では、モデル項目のリポジトリに並び順が保持されています。 例えば customer モデルの name 項目については repository/trunk/customer/customer/name/name.txt 内に

```
model/modelitem/@__linenumber=2
```

という形式で、並びが保持されています。

トラブルシューティング

モデル項目の並びが変わっている

次の点をご確認ください。

- Wagby Designerを使わず、直接、項目フォルダをコピー＆ペーストして作成し、上記 __linenumber を編集しなかったため、重複してしまっている。
- チーム開発で、誰かが並びを変えてコミットした。または、項目の追加や削除、並びの変更を行ったにもかかわらず、順序が変更された全てのリポジトリファイルをコミットしなかった。

対応方法

上記の値を「重複しないよう」手動で編集することもできますが、移行ツールを用いると自動で並び替えを行います。

現在、利用しているバージョンのWagbyをもう一つ、インストールします。（インストールフォルダは別にします。）

同バージョン間で移行を行ってください。

メニュー

.menuと.menuorderフォルダ

.menuフォルダ

モデル毎に、リポジトリキー action/@menuNameXXX, action/@menucolorXXX, @action/@menurefXXX の情報を格納したフォルダが用意されます。

タブ内へのメニューの配置、メニューのタイトル、色に関する変更を行った場合、当該モデルに関する上記ファイルが変更されます。

.menu/.menuorderフォルダ

モデル毎に、action/@menuorderXXX の情報を格納したフォルダが用意されます。

タブ内へのメニューの並びを変更した場合、当該モデルならびに影響をおける他のすべてのモデルのファイルが変更されます。

> WagbyDesignerではドラッグ&ドロップによりメニューの並びを変更できます。このとき内部では左上側から順番に、数字を再割り当てします。（リナンバリング処理）このため、他のモデルの menuorder 値も書きかわることがあります。

__project.txt

管理処理メニューならびに共通処理メニューに含まれるタブに関する情報が格納されています。

チーム開発時の注意点

開発者がメニューの並びを変更した場合、他のモデルが変更される場合があります。具体的には action/@menuorderXXX の値が書き換わります。これは WagbyDesigner が数字を再割り当て（リナンバリング処理）を行うためです。

この変更による影響は .menu/.menuorder フォルダ以下のファイルだけに限定されます。そのためチーム開発では、このフォルダ以下のファイル群のみ取り扱い方針を規定するとよいでしょう。 例えば「このファイルが変わった場合、他の開発者は必ずマージを行う」ルールや「メニュー並びはあるタイミングで決定するようにし、通常の開発では触らない」ルールなどが考えられます。

Appendix

DDL

DDLの作成ルール

Wagby はモデル定義から create table 文、drop table 文といった DDL を自動生成します。これは Hibernate の機能を用いています。具体的には Hibernate マッピングファイル（.hbm.xml）から生成されるものです。

DDL の項目の並びは、モデル定義の項目順ではありません。Hibernate マッピングファイルの記載順となります。

DDLの場所

自動生成されたDDLファイルは以下のフォルダに保存されています。

```
wagbydesigner/webapps/wagbydesigner/WEB-INF/env/work/dbschema/createddl/
```

制約に関する命名

"drop constraint FKXXXX" の、FKXXXXX に相当する部分は Hibernate が自動生成しています。Wagby では特に制御していません。

インデックス

検索条件に対応したインデックス指定は、次のファイルで確認できます。

```
wagbyapp/webapps/wagby/WEB-INF/export/conf/initdb.xml
```

テキストエディタでファイルの内容を確認することができます。次の DDL が含まれています。

```
CREATE INDEX "jfcidx_ModelID_ModelitemName" ON "ModelID"
```

文字エンコード

Wagby が生成する DDL の文字エンコードは、Wagby Designer を動作させている OS に依存します。

例えば Windows OS 環境では Shift JIS になります。そのためテーブル名やカラム名に日本語を指定するとエラーになります。

回避策

wagbydesigner/bin/setenv.bat に下記の行を追加することで、回避できます。DDLがUTF-8で出力されるようになります。

```
set CATALINA_OPTS=%CATALINA_OPTS% -Dfile.encoding=UTF-8
```

ただし、この設定を行うと、Tomcatのコンソール出力が文字化けします。

SQL文における識別子の区切り文字

区切り文字一覧

Wagbyでは各データベースに対して次に示す区切り文字を利用します。標準では次のようになっています。

データベース	区切り文字(quoteid)
HSQLDB	"(ダブルクォート)
PostgreSQL	"(ダブルクォート)
MySQL	`(バッククォート)
Oracle	"(ダブルクォート)
SQLServer	"(ダブルクォート)
DB2	"(ダブルクォート)

SQL の例

利用者がデータベースに対して直接 SQL を発行する場合も、この区切り文字を考慮するようにしてください。

例えばデータベースに Oralce を利用している場合、次のような SQL になります。

```
select * from "juser" where "id" > 0
```

区切り文字を設定しない方法とその影響

区切り文字を付与する理由は、各データベース製品の予約語を回避するためです。

環境 > データベース > 識別子の区切り文字の設定で、これを無効にすることはできます。この場合、利用するデータベースによってモデル名、モデル項目名（英語）に利用できない名前が生じることがあります。具体的には次のようなものがあります。

データベース	利用できない名前（英語）
（共通）	select, update, delete, insert, create, from, to, where, order
Oracle	level, size

(514)

このような理由から、新規にシステム開発を行う場合はトラブル回避のため、区切り文字を付与されることを推奨します。

予約語と利用できる文字

Javaの予約語

```
abstract
assert
boolean
break
byte
case
catch
char
class
const
continue
default
do
double
enum
else
extends
false
final
finally
float
for
goto
if
implements
import
instanceof
int
interface
long
native
new
null
package
page
private
```

(514) 上記に示した項目はすべてではありません。詳細は、各データベースのマニュアルをご覧下さい。

```
protected
public
return
short
static
strictfp
super
switch
synchronized
this
throw
throws
transient
true
try
void
volatile
while
```

(515) (516)

その他の予約語

```
action
and (JSTLで利用)
application
choose
content
css
customize
div (JSTLで利用)
empty (JSTLで利用)
error
errorcode
exception
eq (JSTLで利用)
ge (JSTLで利用)
gt (JSTLで利用)
help
hour_m
id
img
item
jpermission_m
jprincipal
jshparamを含む名前
juser
label
lt
minute_m
mod (JSTLで利用)
movestep_m
ne (JSTLで利用)
not (JSTLで利用)
now
out
or (JSTLで利用)
pageContext
request
```

(515) アルファベット大文字小文字を区別しません。abstractもAbstractもABSTRACTも予約語となります。

(516) 将来のJavaで、予約語が追加される可能性があります。予めご了承ください。

```
response
session
system
xmlで始まる名前
WEB-INF
```

Windows OS で利用できない名前

Windows OS で Wagby Designer を利用する場合、次の名前を利用することができません。

```
con
aux
prn
com1
lpt1
```

Wagby ではリポジトリ（設計情報）は「名前.txt」というファイル名で管理されます。Windows OS では con という名前は予約されているため con.txt というファイル名がエラーとなります。

利用できる文字の制限

Wagby ではモデル名、モデルID は XML の要素名としても使用します。このため、XMLの名前に使用できない文字を指定することはできません。

具体的には次のルールが適用されます。

- 1文字目は半角英字、全角ひらがな、全角カタカナ、漢字、「_(アンダースコア)」などの文字を使用することができます。
- 2文字目以降は1文字目で使える文字に加えて、半角数字などを使用することができます。
- 半角カタカナや全角数字、全角英字、記号は使用できません。

XMLの名前として使用できる文字の詳細については、XML 1.0のW3C勧告(http://www.w3.org/TR/2004/REC-xml-20040204/#CharClasses)をご覧下さい。

クラス名の重複を回避する仕組み

開発者が作成したモデルIDや項目IDが、Wagbyですでに使っている Java のクラス名と重複したとき、生成されるクラス名に "1" などを付与して重複を避ける仕組みになっています。

(517)

例えば "member" というモデルIDのとき、ストアモデルのクラスは Member1.java となります。これは java.lang.reflect.Member と重複するためです。

システムモデルとユーザモデル

システムモデル一覧

モデル名	モデルID	用途
バッチジョブ実行結果	jfcbatchjob_execution	Spring Batch を利用した場合に有効です。ジョブの実行結果が格納されます。登録処理は Spring Batch フレームワークが行います。更新や削除を行うことは（Spring Batch が）想定していませ
バッチジョブ実行詳細	jfcbatchjob_execution_context	
バッチジョブ実行パラメータ	jfcbatchjob_execution_params	
バッチジョブインスタンス	jfcbatchjob_instance	
バッチステップ実行	jfcbatchjob_execution	
バッチステップ実行詳細	jfcbatchjob_execution_context	

(517) 上で説明した「予約語」ではないためモデルIDや項目IDで利用することはできます。

		ん。運用時は閲覧専用になります。
共通設定プリファレンス	jfccspreferenceitem	一覧表示グリッドの設定を複数の利用者で共通利用する場合に用います。共通設定を作成したタイミングでデータが登録されます。運用時は閲覧専用になります。更新や削除は行う場合は直接、データベースを操作してください。
フローパターン	jfcflow_setting	ワークフロー利用時に用いられます。フローパターンを格納します。運用時に独自のフローパターンを追加することもできます。
グラフ設定	jfcgraphsetting	グラフ表示の設定を保存し、他の利用者が共通で利用することができます。共通設定を作成したタイミングでデータが登録されます。運用時は閲覧専用になります。更新や削除は行う場合は直接、データベースを操作してください。
ジョブマスタ	jfcjob	Wagby は Spring Batch とは別に、時間指定によるジョブを用意しています。そのジョブリストを格納しています。その内容は、Wagby インストール時にあらかじめ用意されています。
ジョブスケジュール	jfcjobschedule	jfcjob で用意されたジョブを実行するスケジュールが格納されます。運用者はジョブスケジュールを追加・編集することができます。
ライセンスファイル	jfclicenseholder	本番運用ライセンスファイルに関する情報が格納されます。通常、1レコードを保持します。
メールテンプレート	jfcmailtemplate	メール送信設定が保存されます。設定した分だけレコードが追加されます。
定義モデル	jfcmodel	定義済みのシステムモデルおよびユーザモデルの名称ならびにデータ量を保持しています。開発者や運用者が操作するモデルではなく、閲覧専用として用いられます。
定義モデル	jfcmodel4dm	
フロー参加者設定	jfcparticipant_setting	ワークフロー利用時に用いられます。フローに参加するユーザまたはグループの情報を格納します。運用時に任意に追加することができます。
フロー参加者コンテナ	jfcparticipant_setting_node	
ポータル	jfcportal	ポータルを利用する場合に用いられます。ポータルを利用するかしないか、という1レコードのみを保持します。
ポートレット	jfcportlet	ポータルに表示できる部品情報を保持します。開発者が独自にポートレットを拡張して追加することもできます。
ポートレット（グラフ色）	jfcportletGraphColor_m	グラフポートレットで用います。インストール時に確定さ

ポートレット（グラフ種類）	jfcportletGraphType_m	れており、拡張することはできません。
プリファレンス	jfcpreferenceitem	利用者（ログオンアカウントjuser）毎に用意されます。一覧表示のソート情報や一覧表示グリッドの列並び情報など、ログオフしても値を保持すべき情報が格納されています。再ログオンすると、前回の情報を復元するために用意されています。登録済みアカウント数 x 保持する情報（key-valueのペア）の数だけレコードが追加されます。レコードの追加、編集、削除はシステム内部で自動的に行われます。
代理者設定	jfcproxy_setting	ワークフロー利用時に用いられます。代理承認の設定を行なったときにレコードが追加されます。
帳票テンプレート	jfcreporttemplate	Excel/PDF帳票利用時に用いられます。設定した数だけレコードが追加されます。
開始フロー状態	jfcstartworkstate	ワークフロー利用時に用いられます。自分が申請したワークフローが現在、誰の承認待ちになっているかを知るための管理情報です。システムによってレコードが用意されます。ワークフローが終了してもこのレコードは記録としてデータベースに保持されるため、運用に伴ってデータ量は増加します。管理者が手動で古いデータを削除することが可能です。（削除されたデータは検索対象外になります。）
保留フロー状態	jfcsuspendworkstate	ワークフロー利用時に用いられます。自分が承認すべきワークフローを調べるための管理情報です。システムによってレコードが用意されます。ワークフローが終了してもこのレコードは記録としてデータベースに保持されるため、運用に伴ってデータ量は増加します。管理者が手動で古いデータを削除することが可能です。（削除されたデータは検索対象外になります。）
ワークフロー設定	jfcworkflow_setting	ワークフロー利用時に用いられます。ワークフローを利用するモデルと、ワークフロー定義を結びつけます。設定した数だけレコードが作成されます。
フローイベント	jfcworkstate	ワークフロー利用時に用いられます。ワークフローに対するイベント（承認、差し

		戻し、決裁など）を記録します。システムによってレコードが用意されます。ワークフローが終了してもこのレコードは記録としてデータベースに保持されるため、運用に伴ってデータ量は増加します。管理者が手動で古いデータを削除することが可能です。（削除されたデータは検索対象外になります。）
グループ	jgroup	システム内部で利用するグループ情報です。登録したグループの数だけレコードが追加されます。
グループ階層	jfcgroupnode	グループ階層を実現する情報です。登録したグループ階層の数だけレコードが追加されます。
休日	jholiday	システム内部で利用する休日情報です。初期値は日本の公休日情報 "JapanHolidays.ics" が登録されています。任意の休日を追加登録することもできます。
お知らせ	jnews	利用者へのお知らせをポータル画面に表示することができます。お知らせポートレットを利用した場合にレコードが追加されます。
プリンシパル	jprincipal	利用者（ログオンアカウント juser）毎に用意されます。Wagbyの認可機能を実現します。アカウントの登録と編集にあわせて、システム内部で自動的に管理されます。
アカウント	juser	利用者（ログオンアカウント）です。登録したアカウントの数だけレコードが用意されます。
順序	seq	モデル毎の順序（主キー）を保持します。OracleやPostgreSQLなど、データベースが sequence（順序）サービスを提供する場合には用意されません。MySQLなど、sequence を提供しないデータベースを利用した場合にのみ用意されます。システムが自動的に管理します。

システムモデルの制約

システムモデルには次の制約があります。

- モデルIDを変更してはいけません。
- モデルを削除してはいけません。
- モデルに含まれるモデル項目はID、型を含めすべて変更してはいけません。

以下のモデルは、モデル項目の拡張ができます。

定義済み項目の変更は行えません。具体的には項目名の変更や、型の変更を行うことはできません。

モデル名	モデルID	用途
グループ	jgroup	グループ権限管理で用いる。アカウント(juser)と連携している。
アカウント	juser	ログオンアカウント。
お知らせ	jnews	利用者へのお知らせを行うことができるモデル。

ここに記載している以外のモデルは変更することはできません。インストール時の状態でお使いください。 Wagbyのバージョンアップでは、移行ツール適用時に（上記モデル以外の「システム」に属するモデルは）最新版のモデルが適用されます。

その他の情報

次の ″テーブル″ は、モデル定義としては用意されていません。システム内部で利用するテーブルが別途、用意されます。

テーブル名	用途
jfcreminder	オートスケール環境でのみ用意される。パスワードリマインダ機能のための一時URLが保存される。

複合キーの内部的な表現

テーブル定義

主キーが複合キー形式となっている「商品」モデルを用意します。

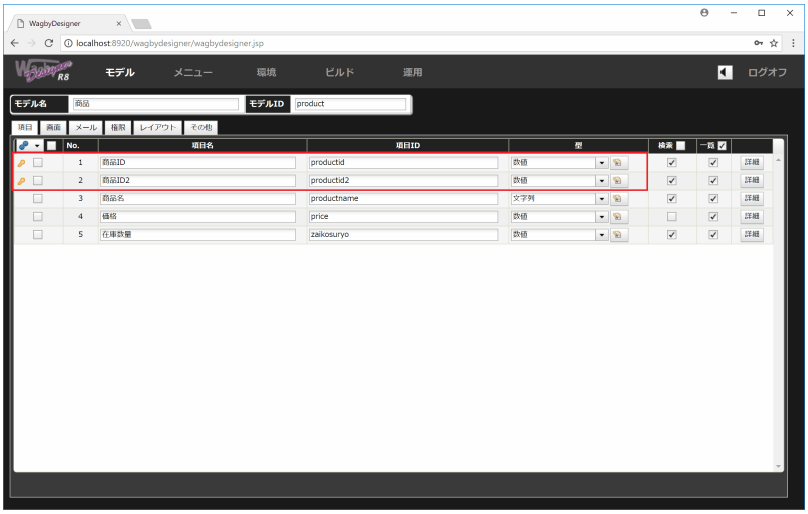


図 商品モデル（主キーが複合キー形式）

「売上伝票」モデルの「明細」部（繰り返しコンテナ）で、商品モデルを参照します。

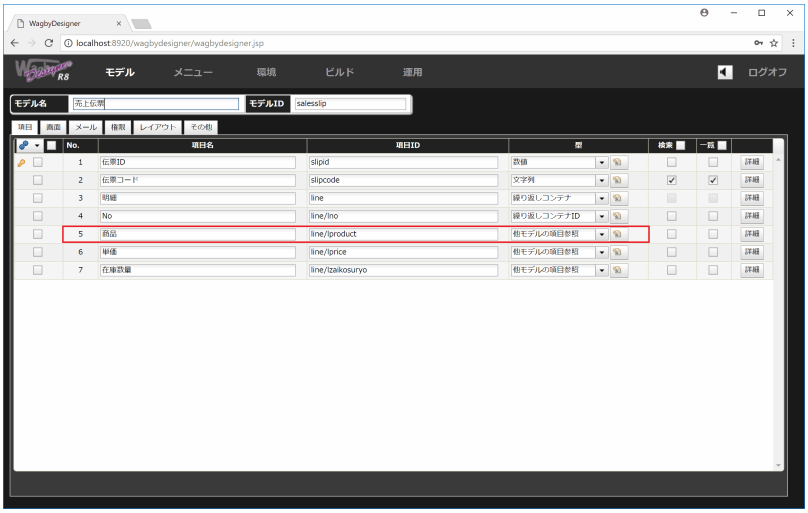


図 売上伝票モデル

Wagbyの設計情報の「商品」項目は、「売上伝票」モデルのテーブル定義においては二つの列が用意されます。

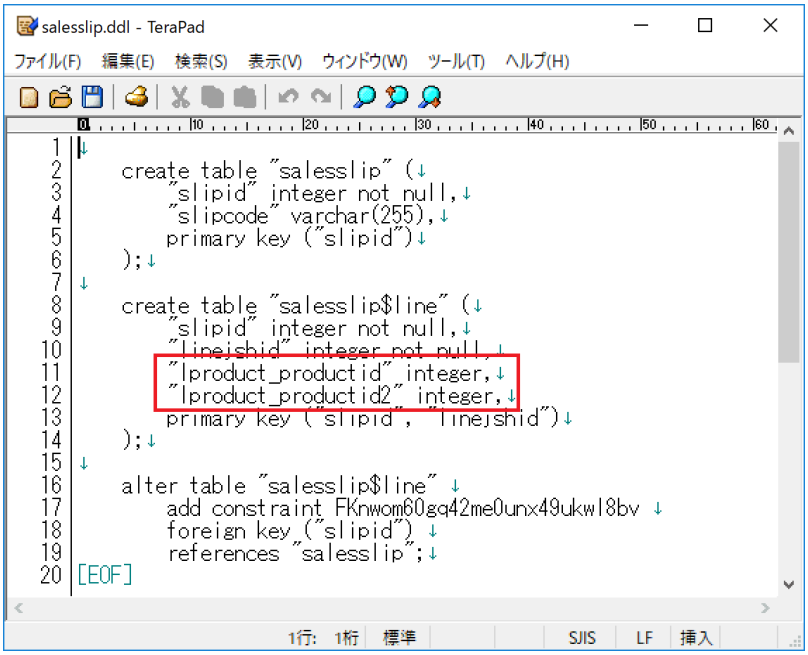


図 売上伝票モデルのテーブル定義

生成される列の命名規則は次のとおりです。

```
(Wagbyの)項目名_参照先モデルの主キー項目名
```

Webフォームのパラメータ

複合キーを利用した場合、Wagbyが生成するWebフォームでは次のような形式でデータを扱います。

```
value1$SEP$value2$SEP$...
```

ここで value1, value2 はそれぞれ主キー1の値、主キー2の値です。"SEP" が値の区切り文字となります。

ロック処理の詳細

ロックの実装方法

楽観ロックの実装

Wagbyが利用する Hibernate というミドルウェアが提供する機能を用いています。ただしタイムスタンプ方式ではなく、バージョン方式のみを採用しています。（タイムスタンプ方式は厳密性に欠けると判断しました。）

悲観ロックの実装

Wagby独自のロックマネージャを用います。ロック取得時のキーはメモリに保持されています。特別な内部テーブル jfclockobject でロックキーを管理させることもできます。この場合、外部システムから「どのレコードがロックされているか」を知ることができます。[詳細は次節で説明します。]

データベースの SELECT FOR UPDATE 構文について

いくつかのデータベースには "SELECT ... FOR UPDATE" 構文が提供されています。しかし Wagby のロックは、この機能を使っていません。その理由を説明します。

"SELECT ... FOR UPDATE" はトランザクション内でのみ有効なロックです。 しかし本節で説明しているロック処理は、ユーザの操作を伴うもので、次のような動作が求められるものです。

1. 更新画面を開く。(ここでデータを SELECT する。)
2. ユーザがデータを編集する。
3. 保存ボタン押下でデータを更新する。(データを UPDATE する。)

ここで 1. と 3. ではトランザクションが異なります。このように複数のトランザクションをまたがるような処理は一般に「ロングトランザクション」として扱います。

(518)

ロングトランザクションでは、"SELECT ... FOR UPDATE" を使うことができません。 この構文は画面処理ではなく、ある完結した業務処理ロジックの中で使うことを想定したものです。

(519)

Wagbyではこのようなロングトランザクションでも一貫性を保つため、楽観ロックならびに（独自のロックマネージャを使った）悲観ロックの両方に対応しています。

(悲観)ロックテーブル

Wagbyの「悲観ロック」は内部でロックマネージャをもっています。このロックマネージャはメモリ内にロックを保持します。この情報は外部から知ることはできません。

このロック情報を（リレーショナルデータベースの）テーブルに書き込むオプションを用意しています。これによって、外部プログラムから「どのデータがロックされているか」を知ることができるため、ロックされているデータは更新しないといった制御を行うことができます。

さらに、外部プログラム側からロックをかけることで、Wagby側からも更新ができないように制御することもできます。

定義方法

「環境 > カスタマイズ > 詳細 > ロック情報をデータベースのテーブルに格納する」を有効にします。

---

(518) データベースのトランザクションとは異なるものです。

(519) 「保存ボタンを押下した直後に "SELECT ... FOR UPDATE" を実行して...」という考え方ではロストアップデートを防ぐことができず、厳密な悲観ロックとはいえません。

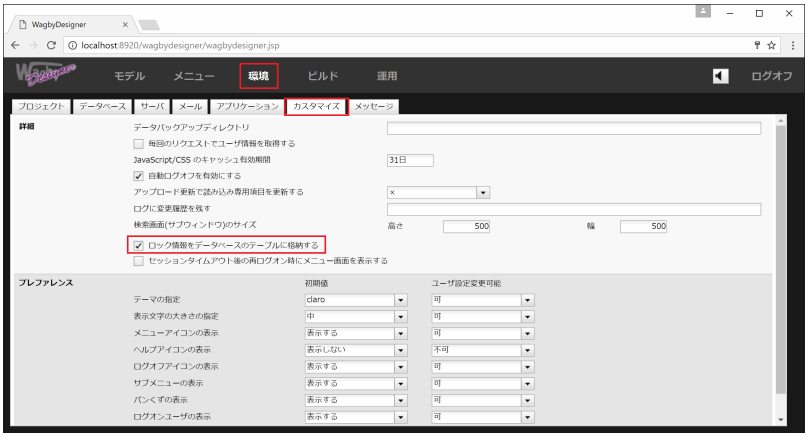


図 ロック情報をデータベースのテーブルに格納する

テーブルの作成

ビルド後に init_db.bat を行うことで外部データベースに jfclockobject テーブルが作成されます。

または手動で jfclockobject テーブルだけを（外部データベースに）作成することもできます。コマンドを実行するカレントディレクトリを wagbyapp¥bin とします。

```
java -jar InitLoader.jar create -t jfclockobject -c ..¥webapps¥wagby¥WEB-INF¥export¥conf
¥initdb.xml
```

・ 参考:インポートとエクスポート機能活用ガイド > InitLoader > 操作例

ロック情報を検索する

システム管理者でログオンし、管理処理タブにある「ロック情報検索」画面を開くと、図のように現在、ロックされているデータが表示されます。

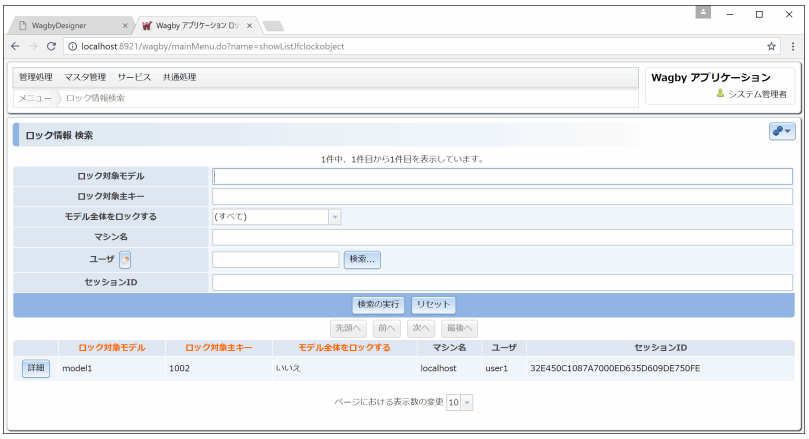


図 ロック情報を検索する

jfclockobjectテーブル

設定を有効にすると、利用するリレーショナルデータベース内に jfclockobject テーブルが用意されます。

項目名	主キー	説明
modelname	○	ロック対象のモデル名（英語）
pkey	○	ロック対象データの主キーの値。複合キーの場合は"$SEP

		$"を区切り文字として値を連結する。
lockForAll	○	数値型。"1" が個別データのロック。"2" がモデル全体のロックを意味する。
userid		ロックを取得した(juserアカウントの)userid値。
username		ロックを取得した(juserアカウントの)username値。
machinename		ロックを取得したホストマシンの値。クラスタリング、オートスケール運用で利用する。
sessionid		ロックを取得したユーザのセッションID値。

あるデータの更新画面を開くタイミングで、jfclockobject テーブルを検索し、該当データに関するロックが存在するかどうかを確認します。すでにロックが存在していた場合、更新画面を開くことができません。存在しなかった場合、同テーブルにロック情報を書き込みます。更新処理が終了したタイミングで、ロック情報を削除します。

外部からロックを取得する

外部プログラムから jfclockobject テーブルにロック情報（レコード）を追加することで、Wagbyの画面から編集できないようにすることができます。[520]

jfclockobject テーブルに用意されたすべての列を埋める必要があります。いずれか一つでも、未設定(null) であった場合、ロックは取得されません。

項目 modelname, pkey, lockForAll は対象データを指し示すために必要です。 項目 userid, username, sessionid は「誰がロックを取得したか」という情報を記録するために必要です。

なお項目 sessionid は Web アプリケーションサーバが発行するセッションIDが格納されます。しかし外部プログラムからこの値を設定する場合、セッションID を生成することは困難です。例えば Web アプリケーションサーバに Tomcat を利用した場合、セッションIDは32文字のランダム文字列です。原則として、ユニークな文字列であれば動作します。

外部プログラムから値を設定する場合は、何らかのルールを決めると良いでしょう。例えばこの値を次のようにします。

```
$HOSTNAME$APPLICATIONNAME$SEQUENCEVALUE
```

このようなルールに基づいて擬似的なセッションID文字列をセットすることで、ユニーク性を保ちつつ、誰がロックしたかも視認しやすくなります。

ロックキーのカスタマイズ

Wagbyのロックマネージャの仕組みを理解し、ロックキーをカスタマイズすることができます。（悲観ロック利用時。楽観ロックの場合は、本内容は適用されません。）

Wagby の悲観ロックでは、Wagby内部のロックマネージャを用います。ユーザーがデータの更新画面を開いたタイミングで対象データをロックします。

このとき、対象データのモデル名を「ロック名」とし、主キー項目を「ロック項目」として扱います。ロック名とロック項目の値から「ロックキー」を生成し、これをロックマネージャで管理しています。

ロックキーのカスタマイズとは、このロック名やロック項目を任意に設定できるようにすることです。 さらに、ひとつの更新画面で複数のロックキーを扱えるようにします。

この機能を利用すると、トランザクション処理のためのカスタマイズコードを記述するときに対象データのロック管理を柔軟に制御することができるようになります。

定義方法

[520] 事前に同テーブルにロック対象データがないか確認してからレコードを追加するようにしてください。

「モデル項目詳細定義>詳細>データベースの詳細>ロック名」を指定します。

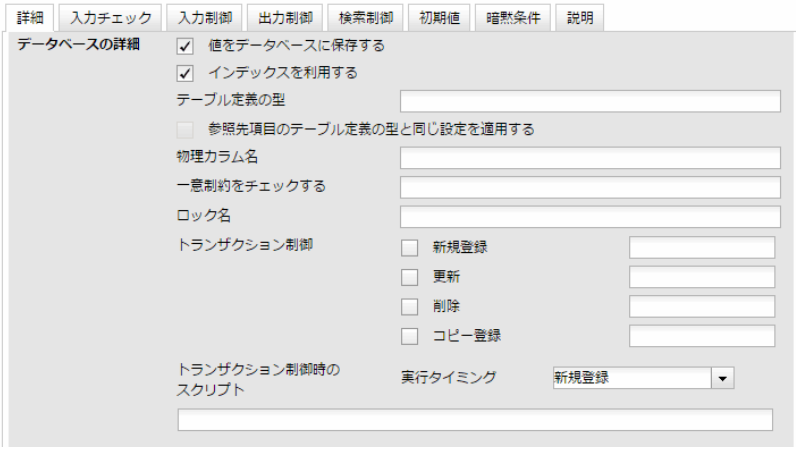


図 モデル項目詳細定義

標準は未指定（空白）となっています。対象となるモデルのすべての項目で未指定の場合は、モデル名をロック名とし、主キー項目をロックキーとします。これが通常の動作となります。

値を記載すると、この項目がロックキーとして用いられます。また記載した値がロック名となります。

サンプル

次のモデルを例に示します。

モデル a

項目	主キー
ida1	○
ida2	

モデル b

項目	主キー
idb1	○
idb2	

また、モデルaおよびモデルbのデータが次のようになっているとします。

モデルaのデータ：2つのデータ

	項目 ida1	項目 ida2
1	1000	1001
2	1001	1002

モデルbのデータ：2つのデータ

	項目 idb1	項目 idb2
1	1000	1001
2	1001	1002

例1:モデル a のデータを更新する際に、モデル b のデータもロックする。

1000番のモデル a のデータを編集すると、1000番のモデル b のデータもロックする。

モデル a 項目 ida1 のモデル項目定義シートの「モデルの関連性 - ロック名」に「b」と指定します。これにより、モデル a のデータを編集する場合でも、ロック名はモデル b と同じになるため、モデル b のデータもロックされることとなります。

例2:モデル a のデータを更新する際に、ida2の値でモデル b もロックする。

1000番のモデル a のデータを編集する場合、ida2は1001なので、1001番のモデル b のデータもロックする。

モデル a 項目 ida2 のモデル項目定義シートの「モデルの関連性 - ロック名」に「b」と指定します。この設定により、モデル a のデータを更新する場合、ロック名 b ロックキー ida2 の値となるロックを行います。

例3:下記のような複合キーのあるモデル c がある場合に、モデル a を更新する際に、ロックキーとして ida1 と ida2 を使い、モデル c をロックする。

モデル c

項目	主キー
idc1	○
idc2	○

モデル a 項目 ida1 のモデル項目定義シートの「モデルの関連性 - ロック名」に「c」と指定します。

モデル a 項目 ida2 のモデル項目定義シートの「モデルの関連性 - ロック名」に「c」と指定します。

一つのモデルにて同じロック名が指定されている場合は、複合キーとして扱います。
ida1:1000,ida2:1001のモデル a のデータを更新する場合、idc1:1000,idc2:1001のモデル c のデータをロックします。

例4:モデル a を更新する際に、モデル b とモデル c のデータをロックする。

モデル a 項目 ida1 のモデル項目定義シートの「モデルの関連性 - ロック名」に「b c」と指定します。

モデル a 項目 ida2 のモデル項目定義シートの「モデルの関連性 - ロック名」に「c」と指定します。

スペース区切りで複数のロック名を指定することができます。

ida1:1000,ida2:1001のモデル a のデータを更新する場合、idb1:1000のモデル bのデータと、idc1:1000,idc2:1001のモデル c のデータをロックします。

繰り返しコンテナ項目の値を使う場合

繰り返しコンテナ内の項目を使ったロック名の指定を行うことができます。繰り返しコンテナの数に合わせて、ロックを行います。

[注意] 更新画面では繰り返しコンテナの追加/削除/挿入は行わないようにしてください。（ボタン表示制御で、SCREENTYPE 関数を利用して更新画面の場合にこれらのボタンを非表示にする、といった対応を行なってください。）

一つの繰り返し項目、繰り返しコンテナ内の項目に指定した場合

この場合は、n 個のロックオブジェクトが作成されます。

例: 下記のようにcont1/c1contentにロック名が指定されたモデルがあるとします。

項目	主キー	ロック名
id	○	
cont1		
cont1/c1id		
cont1/c1content		test1

この場合、ロック名をtest1、ロックデータキーにcont1/c1contentを用いてロックを作成します。

(521)

具体的に説明します。このモデルに対応する下記のデータがあるとします。

(521) cont1のすべてのデータに対応するロックオブジェクトを用意します。

```
<test1>
  <id>1000</id>
  <cont1><c1id>1</c1id><c1content>a</c1content></cont1>
  <cont1><c1id>2</c1id><c1content>b</c1content></cont1>
</test1>
```

この場合、次の2つのロックオブジェクトが内部で用意されます。

ロック名:test1 ロックキー:a ロック名:test1 ロックキー:b

一つの項目と、一つの繰り返しコンテナ内の項目に指定した場合

この場合は、n 個の複合キーとなるロックオブジェクトが作成されます。

例: 下記のようにidとcont1/c1contentにロック名が指定されたモデルがあるとします。

項目	主キー	ロック名
id	○	test1
cont1		
cont1/c1id		
cont1/c1content		test1

この場合、ロック名をtest1、ロックデータキーにidとcont2/c2contentを用いてロックを作成します。

(522)

具体的に説明します。このモデルに対応する下記のデータがあるとします。

```
<test1>
  <id>1000</id>
  <cont1><c1id>1</c1id><c1content>a</c1content></cont1>
  <cont1><c1id>2</c1id><c1content>b</c1content></cont1>
</test1>
```

この場合、次の2つのロックオブジェクトが内部で用意されます。

ロック名:test1 ロックキー:1000$a ロック名:test1 ロックキー:1000$b

複数の繰り返し項目、繰り返しコンテナ内の項目を指定した場合

この場合は、n*m 個の複合キーとなるロックオブジェクトが作成されます。

例: 下記のようにcont1/c1contentとcont2/c2content項目にロック名が指定されたモデルがあるとします。

項目	主キー	ロック名
id	○	
cont1		
cont1/c1id		
cont1/c1content		test1
cont2		
cont2/c2id		
cont2/c2content		test1

この場合、ロック名をtest1、ロックデータキーにcont1/c1contentとcont2/c2contentを用いてロックを作成します。

(523)

具体的に説明します。このモデルに対応する下記のデータがあるとします。

```
<test1>
```

(522) 通常項目と繰り返し項目等に同一のロック名を指定した場合、id項目の値と、cont2のすべての項目値を組み合わせてロックオブジェクトを用意します。

(523) 複数の繰り返し項目などに同一のロック名を指定した場合、cont1のすべてのデータと、cont2のすべてのデータを掛け合わせて総当たりでロックオブジェクトを用意します。

```
  <id>1000</id>
  <cont1><c1id>1</c1id><c1content>a</c1content></cont1>
  <cont1><c1id>2</c1id><c1content>b</c1content></cont1>
  <cont2><c2id>1</c2id><c2content>c</c2content></cont2>
  <cont2><c2id>2</c2id><c2content>d</c2content></cont2>
</test1>
```

この場合、次の4つのロックオブジェクトが内部で用意されます。

ロック名:test1 ロックキー:a$c ロック名:test1 ロックキー:a$d ロック名:test1 ロックキー:b$c ロック名:test1 ロックキー:b$d

仕様・制約

- ここで説明したロックキーのカスタマイズについてのお問い合わせは Premium Support の対象となります。
- モデル参照（チェックボックス）にロック名を指定することはできません。
- 複数のモデルでロック名を共用する場合に、ロックキーとなる項目の型が一致する必要があります。上記、例1の設定では、モデル a 項目 ida1 とモデル b 項目 idb1 の項目の型が一致しています。
- 複合キーとして扱われた場合は、設定されている順番と項目の型が一致する必要があります。例3の設定では、項目 ida1 と idc1、項目 ida2 と idc2 の項目の型がそれぞれ一致しています。
- ロックキーに指定した項目は更新画面、一覧更新画面で読み込み専用項目となります。
- ロックは更新時のみ行われます。新規登録やコピー登録の際には行われません。(*)

(524)

キャッシュ処理の詳細

キャッシュについて

Wagbyはデータベースから読み込んだ値をメモリ内にキャッシュしています。これによりシステム全体のパフォーマンスを向上させています。

モデルごとにキャッシュを使うかどうかを指定する

モデルごとにキャッシュを有効にするかどうかを指定することができます。 ただし、ワークフローを有効にしたモデルは "データベースのキャッシュを完全に無効にする" をチェックすることはできません。

サイズの設定

アプリケーション全体のキャッシュサイズを指定することができます。

モデルごとにキャッシュサイズを指定することはできません。内部ではオープンソースの ehcache ライブラリを用いており、LRU アルゴリズムによって、よく使われるデータが優先してキャッシュします。

(525)

一時ファイル

wagbyapp/temp フォルダに一時ファイルが生成されることがあります。Wagbyアプリケーション停止後であれば、wagbyapp/temp フォルダに残った一時ファイルを手動で削除することができます。

キャッシュの種類と性質

キャッシュは「ストアモデル」と「選択肢」があります。

---

(524) Wagbyは新規登録時には排他制御は行いません。カスタマイズを行う場合でも登録と更新が競合しない設計方針となるよう、留意ください。

(525) JVMヒープメモリ領域に空きが少なく、キャッシュ用のメモリ領域を確保できない場合に "CacheManager configuration: You've assigned more memory to the on-heap than the VM can sustain, please adjust your -Xmx setting accordingly" というエラーメッセージがコンソールに表示されます。この場合はヒープメモリ領域を増やすか、キャッシュ用のメモリ領域を減らしてください。（ヒープメモリ領域を増やす場合は、指定したサイズよりも大きな物理メモリがあることが前提となります。）

区分	説明	Time To Live	Time To Idle
ストアモデル	データベースから読み込んだ1データ。スクリプトに関するマニュアルでは「エンティティ」と表記することもある。	600 sec(10分)	600 sec(10分)
選択肢	登録、更新、詳細、一覧画面のモデル参照項目で表示される「内容」部の文字列。	無制限	259,200 sec(72時間)

TTL値とTTI値は、WEB-INF/classes/ehcache.xml で指定しています。

よくある質問と回答

誰かが操作中にキャッシュクリア処理を行なっても問題ありませんか。

Wagbyではジョブ実行時にキャッシュをクリアする指定を行うことができます。または REST API を利用してキャッシュクリアを指示することもできます。

もし誰かが操作途中にキャッシュがクリアされた場合、キャッシュからのデータを取得できなければデータベースから取得するようになっているため、問題ありません。

関連するページ

- 環境 > サーバ > キャッシュ, キャッシュサイズの指定
- 繰り返し > 繰り返しコンテナ - 独立したモデルとして扱う, キャッシュを無効にする設定が必要。
- モデル > テーブル操作に関する注意点まとめ > キャッシュ, キャッシュを定期的にクリアする方法について。
- モデル > サブデータベースを指定する > キャッシュの扱い, サブデータベース利用時のキャッシュについて。
- モデル > データベースのビューを利用する, キャッシュの扱いについて。
- 画面機能 > メインモデルとサブモデル > メインモデルとサブモデルという関係ではなく、物理テーブルを共用することはできますか。, キャッシュの考慮について。
- 環境 > クラスタリング環境で運用する > 動作の確認, キャッシュのテストについて。
- 環境 > オートスケール環境で運用する > 動作の確認, キャッシュのテストについて。
- Wagbyのオートスケールに関する技術情報, キャッシュの整合性を保つ方法の解説。
- 管理者ガイド > モデルのキャッシュを消去する, ジョブの設定方法。
- スクリプト > Service/Daoクラス、ヘルパ、SQLを利用する > SQLやストアドプロシージャでデータを更新する > キャッシュのクリア
- パフォーマンスチューニング > ビューを使う場合でもキャッシュを有効にする

項目の型とデータベースの関係

文字列

データベースに格納される文字列型の標準サイズは次のとおりです。

データベース	定義
Oracle	varchar2(255 char)
SQL Server	nvarchar(255)
DB2	varchar(255)
MySQL	varchar(255) (*1)
PostgreSQL	varchar(255)
HSQLDB (内蔵DB)	varchar(255)

(526)

数値

データベースに格納される数値型の標準サイズは次のとおりです。

データベース	型	定義
Oracle	整数	number(10,0)
	1バイト整数	number(3,0)
	2バイト整数	number(5,0)
	4バイト整数	number(10,0)
	8バイト整数	number(19,0)
	4バイト浮動小数	float
	8バイト浮動小数	double precision
SQL Server	整数	int
	1バイト整数	tinyint
	2バイト整数	smallint
	4バイト整数	int
	8バイト整数	numeric(19,0)
	4バイト浮動小数	float
	8バイト浮動小数	double precision
DB 2	整数	integer
	1バイト整数	smallint
	2バイト整数	smallint
	4バイト整数	integer
	8バイト整数	bigint
	4バイト浮動小数	float
	8バイト浮動小数	double
MySQL	整数	integer
	1バイト整数	tinyint
	2バイト整数	smallint
	4バイト整数	integer
	8バイト整数	bigint
	4バイト浮動小数	float
	8バイト浮動小数	double precision
PostgreSQL	整数	int4
	1バイト整数	int2
	2バイト整数	int2
	4バイト整数	int4
	8バイト整数	int8
	4バイト浮動小数	float4
	8バイト浮動小数	float8
HSQLDB (内蔵DB)	整数	int4
	1バイト整数	int2
	2バイト整数	int2
	4バイト整数	int4
	8バイト整数	int8
	4バイト浮動小数	float4
	8バイト浮動小数	float8

テキストエリア

文字列型の項目をテキストエリアに変更した場合（またはその逆に、テキストエリア指定を解除した場合）Wagby は関連するモデルのテーブル定義を変更することがあります。例えば外部データベースに MySQL

(526) 1. varchar 型で指定する数は MySQL のバージョンによって変わります。4.1 以前は「バイト」単位ですが、それ以降は「文字」単位です。詳細はMySQLのマニュアルでご確認ください。

を用いている場合、通常の文字列型は varchar 型を適用しますが、テキストエリアの場合は text 型を用いるようになります。

テキストエリアを指定した項目には「格納できる文字数」の上限があります。これはデータベースによって変動します。 これを変更する場合は"テーブル定義の型"を直接、指定してください。

データベース	文字列	文字列（テキストエリア）
HSQLDB(組み込みDB)	varchar(4000)	varchar(4000)
PostgreSQL	varchar(255)	text (*1)
Oracle	varchar2(255)	varchar2(4000)
SQL Server	nvarchar(255)	nvarchar(1000)
DB 2	varchar(255)	varchar(4000)
MySQL	varchar(255)	text (*2)

(527) (528)

選択肢モデル/リストボックス/ラジオボタン

モデルを参照する側（参照元モデル）の項目では、参照先モデルのID値を保持します。内容部ではありません。

(529)

チェックボックス

図は、標準で用意されているアカウントモデル(juser)のテーブル構成を抜粋したものです。「所属グループ(jgroupid)」項目はグループモデル(jgroup)をチェックボックスとしてモデル参照しています。この項目は juser テーブルには存在しません。代わって「juser$jgroup」という名前のテーブルが作成されます。

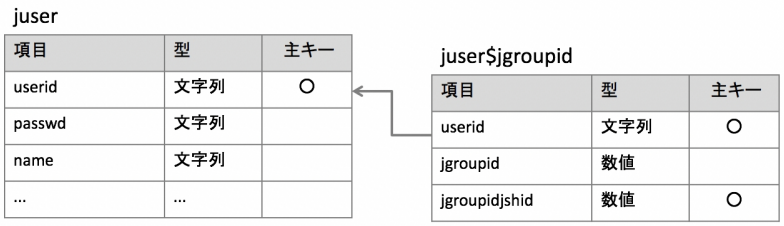


図 テーブルイメージ

- テーブル名は「モデル名$項目名」です。
- 所属先となるjuserテーブルの主キーを持ちます。
- 「項目名jshid」という列が自動的に付加されます。これも主キーの一部になります。これは内部管理上、インデックス列として用意されるものです。
- 「項目名jshid」の値は 0 から開始されます。また飛び番号はありません。

一つも選択していない状態は、別テーブルに一件もデータがないという状態になります。実際に保持する値（上の例では jgroupid 列の値）は、参照先モデル（jgroup）の主キーになります。

データベースを直接操作する場合、「項目名jshid」は 0 から開始かつ飛び番号なしというルールを遵守してください。不整合が生じた場合、チェックボックス項目は機能しなくなります。

(527) 1. PostgreSQL の text 型は 1G バイトです。

(528) 2. MySQL の text 型は 65,535 バイトです。

(529) データベース上のテーブルを確認すると、ID値が格納されていることがわかります。

繰り返し項目

繰り返し属性を指定した項目は、別テーブルに分割されます。（1:Nの関係になります。）そのため登録できるデータ数の上限はありません。

(530)

生成されるテーブルは「モデル名$項目名」となります。

繰り返しコンテナ

繰り返しコンテナは、別テーブルに分割されます。（1:Nの関係になります。）そのため登録できるデータ数の上限はありません。

(531)

生成されるテーブル名は「モデル名$（繰り返しコンテナ）項目名」となります。このテーブルの命名規則は固定です。

(532)

繰り返しコンテナID

「繰り返しコンテナID」は、標準では"<コンテナ名>jshid"という名前のカラムが用意されます。型は整数型です。

- 「繰り返しコンテナID」はデータベースに保存する必要があります。データベース非保存とすると正しく動作しません。
- 「繰り返しコンテナID」はデータベースには　"0"　から開始される連番として値が保存されています。nullは許容しません。また、飛び番号も認められません。

この「繰り返しコンテナID」の物理カラムを指定することができます。

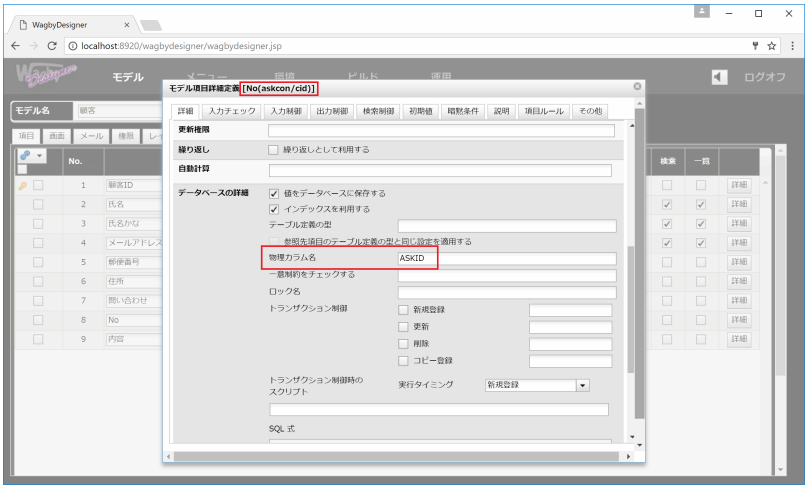


図　「繰り返しコンテナID」の物理カラムを指定する

(530) 厳密には採用する外部データベースの、1テーブルに保持できるレコード数が理論的な上限となります。よって実運用での制約は、なしとみなすことができます。

(531) 厳密には採用する外部データベースの、1テーブルに保持できるレコード数が理論的な上限となります。実運用での制約は、なしとみなすことができます。

(532) 繰り返しコンテナ項目は、物理カラムを指定しても無視されます。

郵便番号辞書を手動で用意する

手順

株式会社アイビス様が提供する郵便番号データと、日本郵政公社が提供する事業所の個別郵便番号データを用いて、zipcode.dat を作成する方法を説明します。

1. 株式会社アイビス様が提供する「zipcloud」にアクセスし、「ダウンロード」リンクをクリックします。同データは無償で公開されています。

図 zipcloud

2. ダウンロードした "ken-allYYYYMM.zip" (YYYYは年号,MMは月) を "ken_all.zip" に改名し、wagbyapp/webapps/wagby/WEB-INF/jmaster に保存します。（ken-all というハイフンは ken_all というアンダースコアに変更してください。）
3. 日本郵政公社の郵便番号検索サイト http://www.post.japanpost.jp/zipcode/index.html にアクセスし、「郵便番号データのダウンロード」リンクをクリックします。（毎月一回、月末に更新されます。）

図 郵便番号検索サイト

4. 事業所の個別郵便番号をダウンロードするためのリンクをクリックします。

図 事業所の個別郵便番号ダウンロード(1)

5. zip形式でのダウンロードへのリンクをクリックします。

図 事業所の個別郵便番号ダウンロード(2)

6. 最新データのダウンロードリンクをクリックすると、"jigyosyo.zip" というファイルがダウンロードされます。（ファイルサイズは年月によって異なります。）

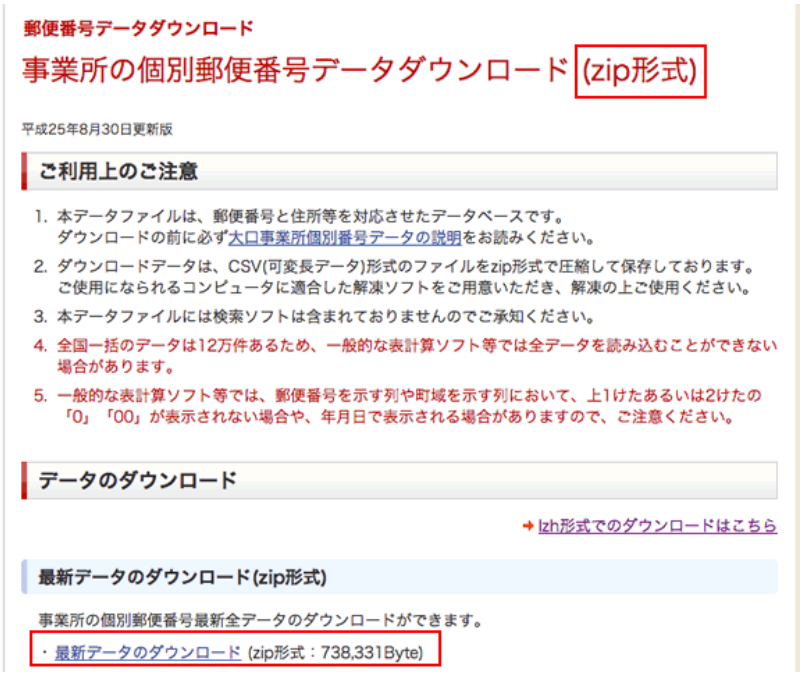


図 事業所の個別郵便番号ダウンロード(3)

7. ダウンロードした "jigyosyo.zip" も、wagbyapp/webapps/wagby/WEB-INF/jmaster に保存します。

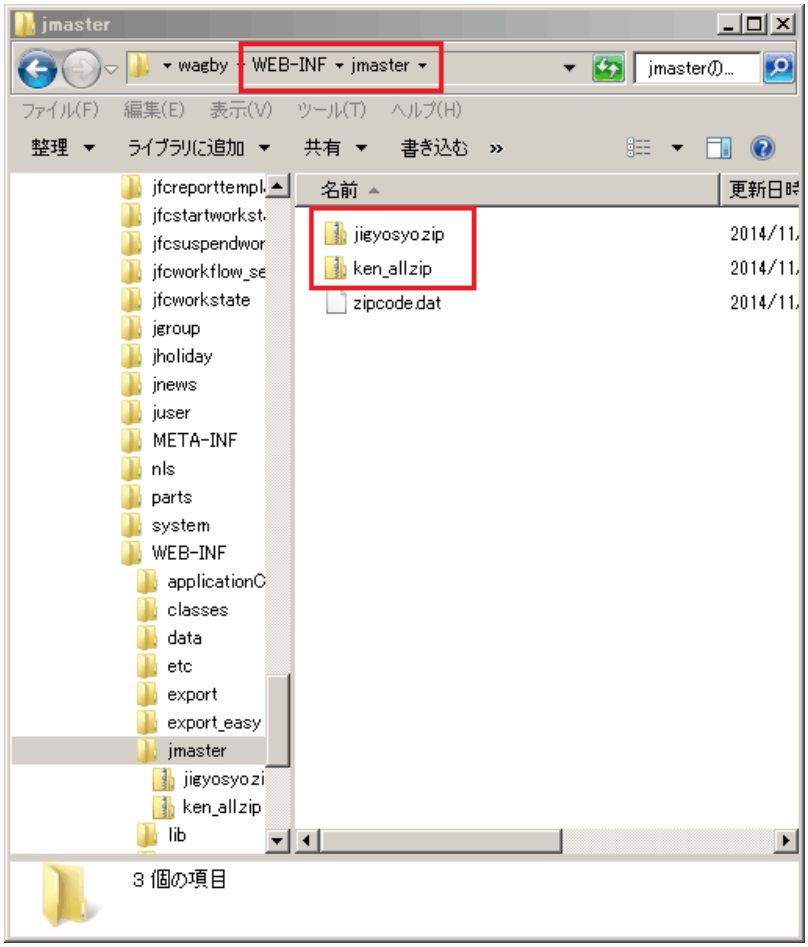


図 jmasterフォルダ

8. Wagbyアプリケーションを停止します。
9. コマンドプロンプトを起動し、wagbyapp/webapps/wagby/WEB-INF フォルダに移動します。 次のコマンドを入力し、実行します。

```
java -cp lib/j_util.jar;lib/commons-lang.jar jp.jasminesoft.util.address.GenerateMasterData
```

(533)

次のように出力されます。

```
C:¥Wagby-8.x.x¥wagbyapp¥webapps¥wagby¥WEB-INF>java     -cp     lib/j_util.jar;lib/commons-lang.jar
 jp.jasminesoft.util.address.GenerateMasterData
Generate Master Data Program. compaction=true
Reading postal file...done.
Creating zip file...done.
```

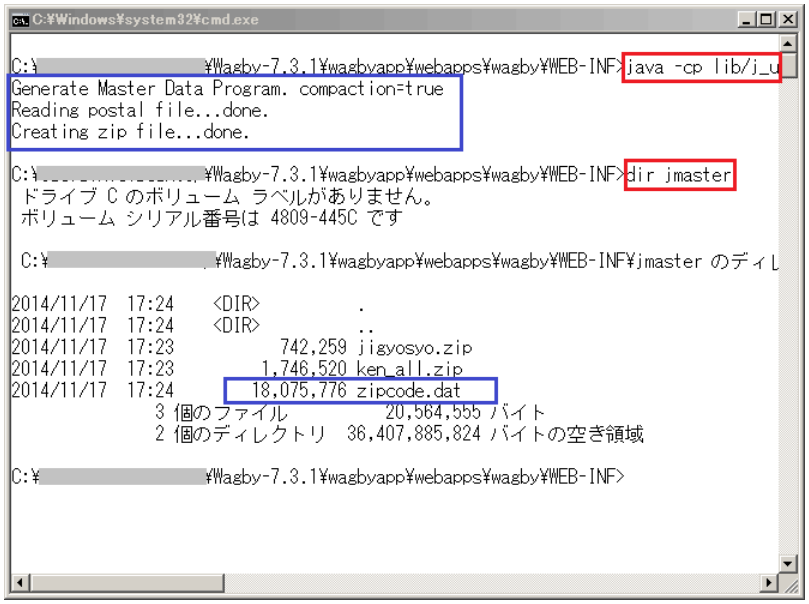


図 コマンド実行

10. wagbyapp/webapps/wagby/WEB-INF/jmaster/zipcode.dat が更新されたことを確認します。
11. コマンドプロンプトを終了します。
12. Wagbyアプリケーションを再起動します。

このあとにビルド処理を行った場合は、ここで作成した zipcode.dat を再度、wagbyapp/webapps/wagby/WEB-INF/jmasterフォルダに保存してください。

住所正規化コンバータ(R6)との連携方法

Tomcat の設定

Apache Tomcat のサイトから最新の Tomcat を入手します。

ダウンロードして展開した apache-tomcat フォルダを "AnormR6Tomcat" に改名しました。

webappsフォルダの設定

標準で添付されている webapps フォルダ以下のファイルを削除します。

(533) javaコマンドが動作しない場合、「Javaのインストール」手順に従って、Java本体のインストールを行ってください。

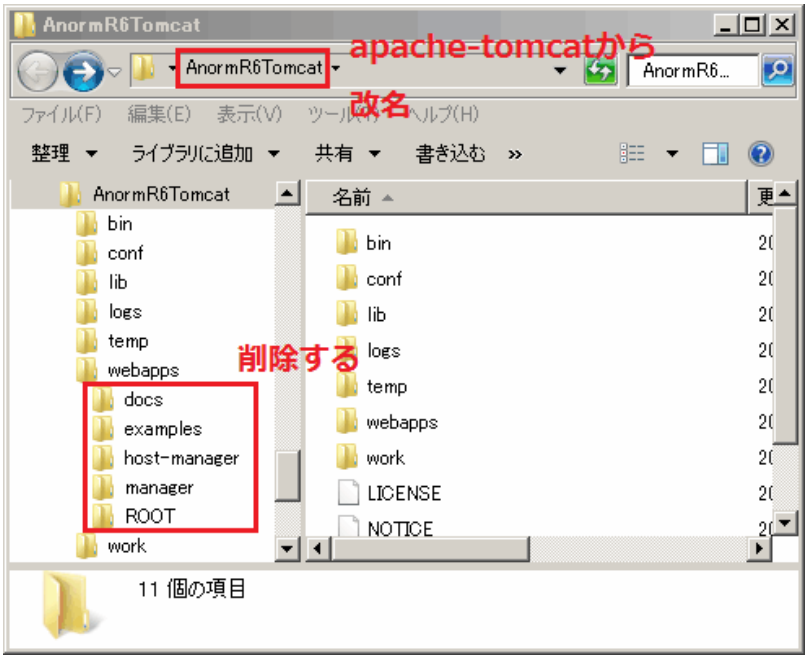


図 webappsフォルダ以下の標準ファイルを削除する

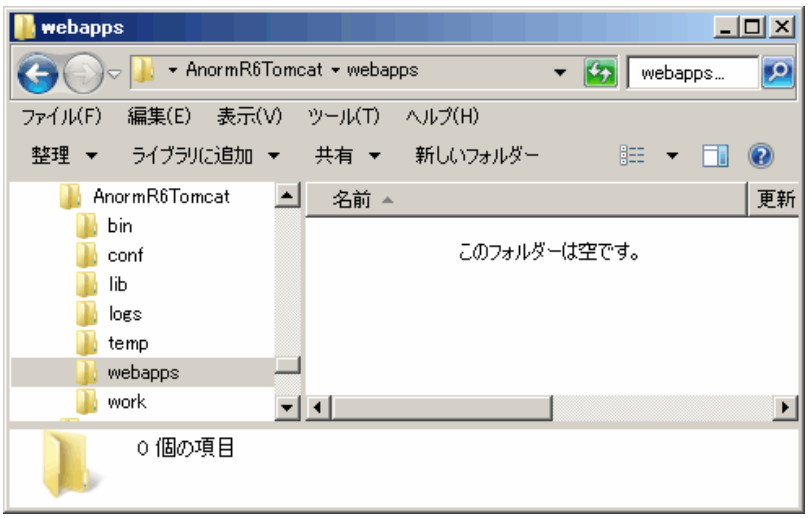


図 空になったwebappsフォルダ

入手した住所正規化コンバータサーバモジュールを webapps フォルダに展開します。

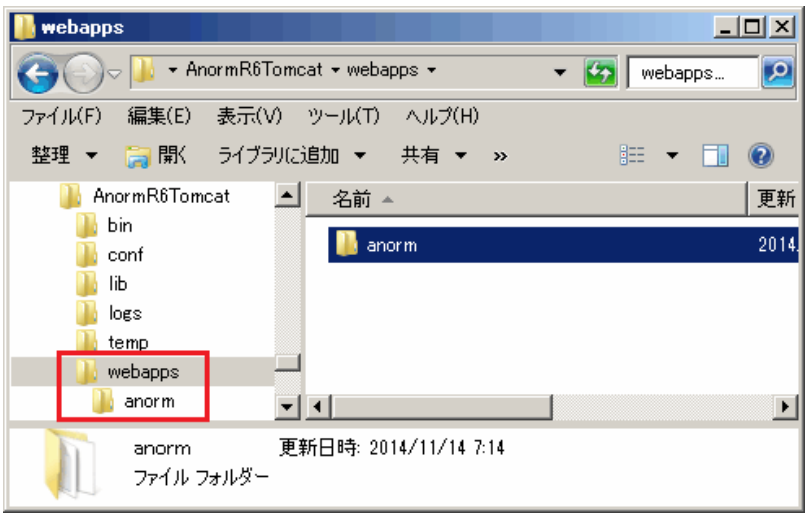


図 webappsフォルダにanormが展開された

展開後のwebappsフォルダを示します。

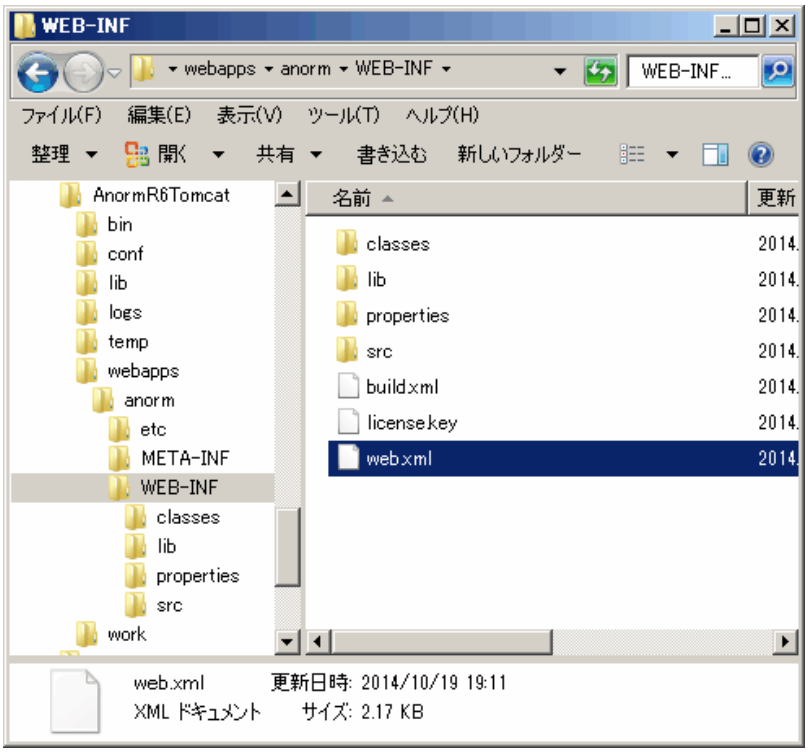


図 展開後のwebappsフォルダ

web.xmlの編集

anorm/WEB-INF/web.xml を編集します。(Windows OS 付属のメモ帳は使えません。TeraPadなど別のテキストエディタを使ってください。)

"jmasterDirPath" を適切に編集します。住所正規化コンバータが提供する住所マスタである jmaster のパスを指定してください。(Windows OS では、フォルダの区切り文字は円マークを二つ入力します。)

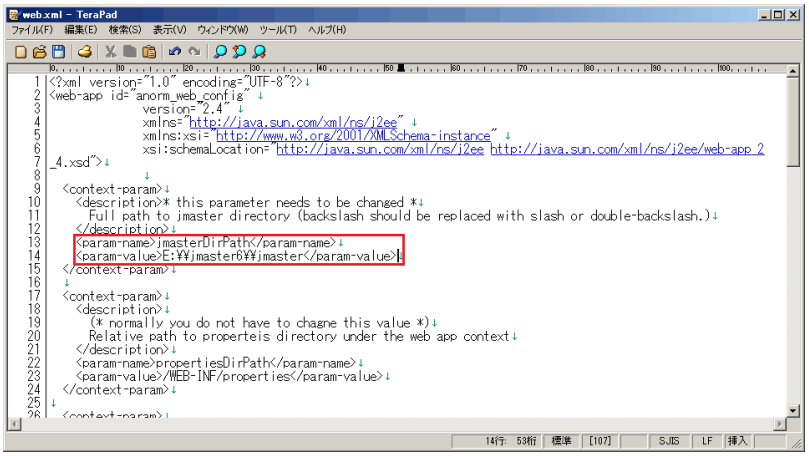


図 jmasterのパスを指定する

catalina.batの編集

binフォルダに含まれているcatalina.batファイルを編集します。

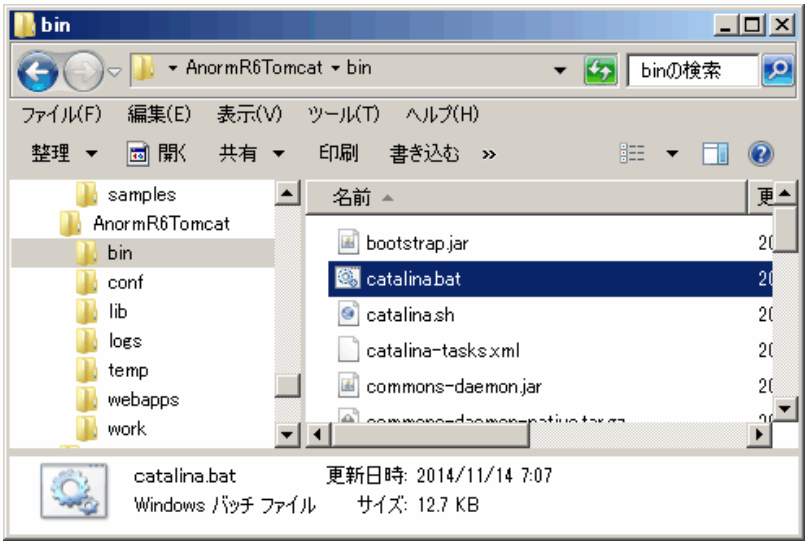


図 catalina.batファイル

コメントが終わった部分に、メモリサイズを指定します。

```
set JAVA_OPTS=-Xmx1024m -Xms1024m
```

住所正規化コンバータR6は、標準で1Gバイトのヒープメモリが必要のため、これを指定しています。

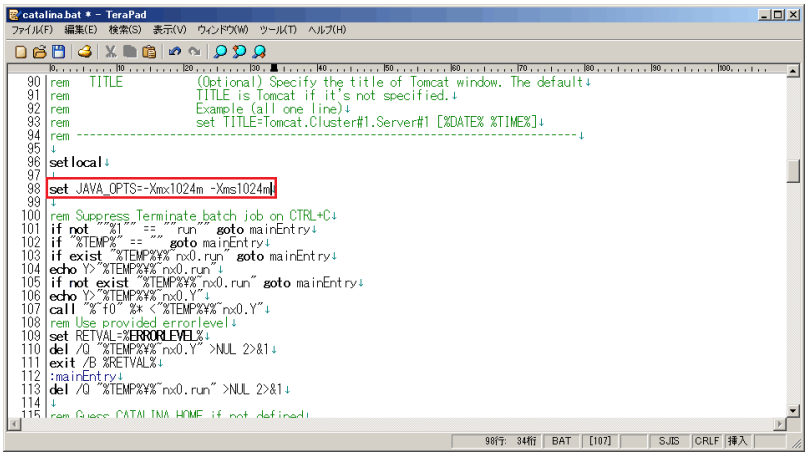


図 メモリサイズの設定

server.xmlの編集

住所正規化コンバータサーバとWagbyの開発機が「別である」場合は、この手順をスキップすることができます。

ここでは同一マシンで住所正規化コンバータのサーバとWagbyDesignerを動作させている場合を想定し、住所正規化コンバータサーバのTomcatのポート番号を変更する方法を説明します。

ポート番号が重複していると、サーバの起動に失敗します。

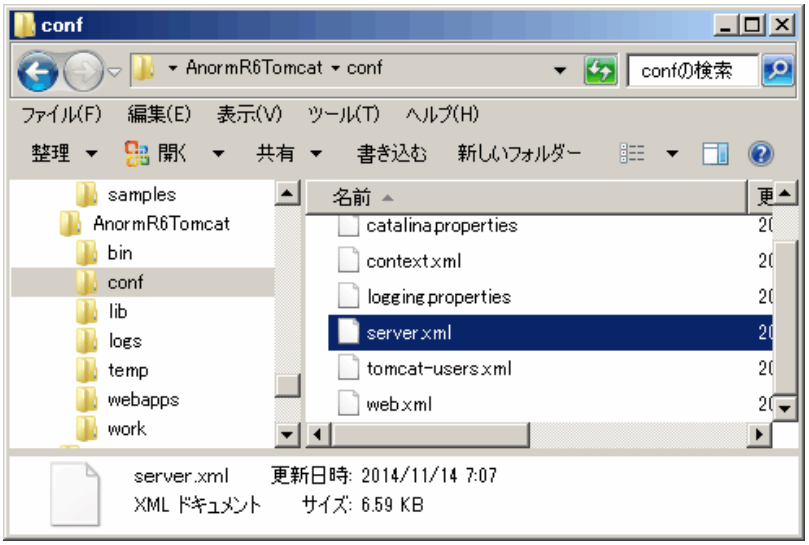


図 server.xmlを開く

ポート番号 "8005" の部分を、"18005" に変更した例です。

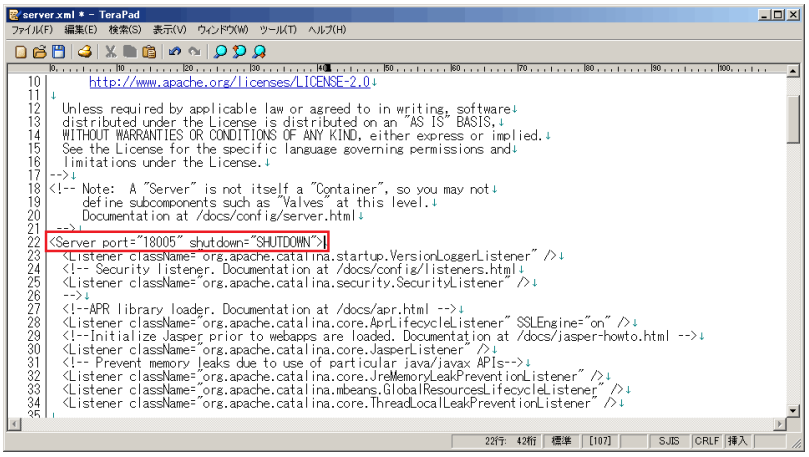


図 8005番ポートの変更

ポート番号 "8443" の部分を、"18443" に変更した例です。

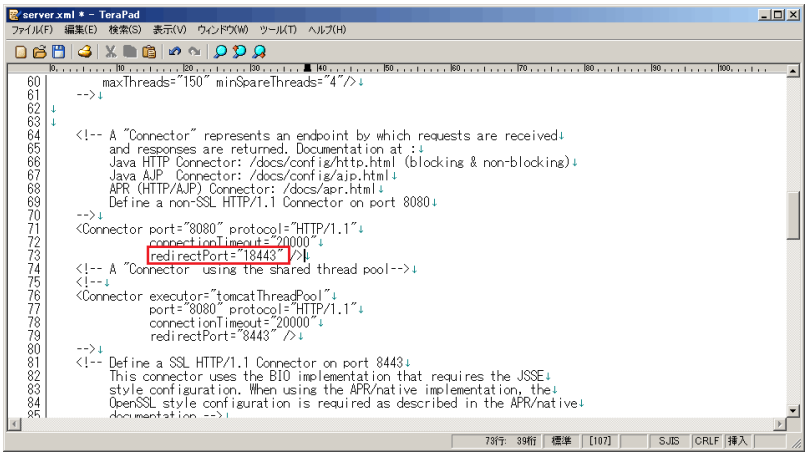


図 8443番ポートの変更

ポート番号 "8009" の部分を、"18009" に変更した例です。

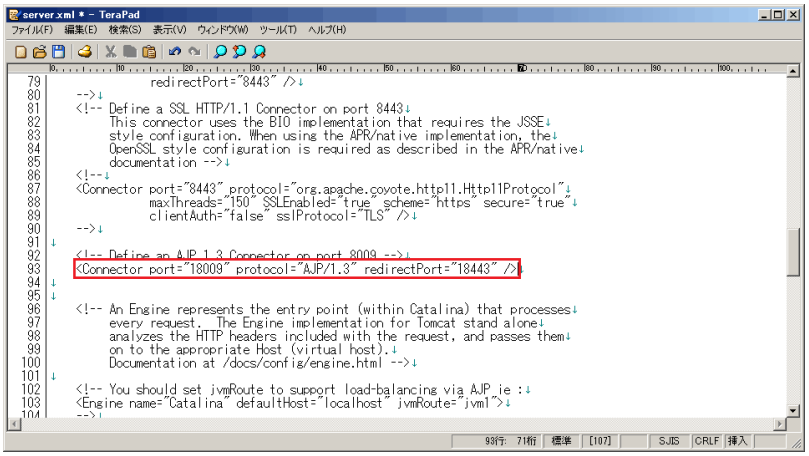


図 8009番ポートの変更

住所正規化コンバータサーバ(Tomcat)の起動

binフォルダのstartup.batを実行します。

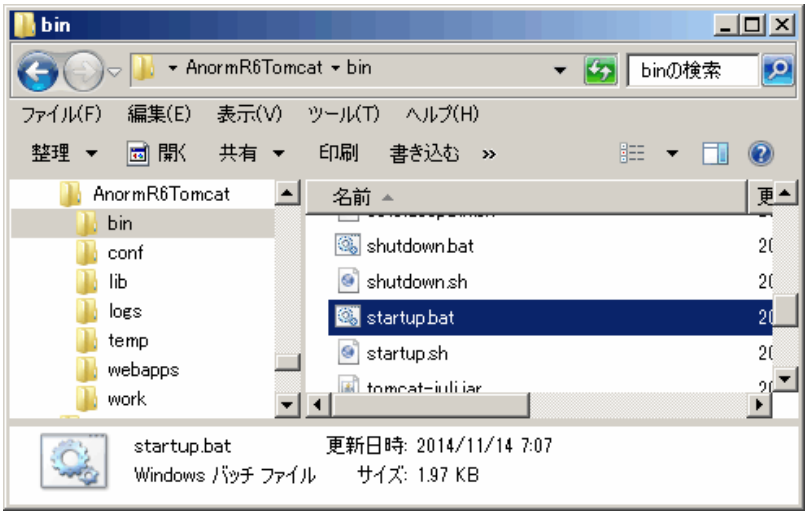


図 startup.batファイル

起動後、住所マスタの読み込みが開始されます。コンソールにはマスタの読み込みに関する表示があります。"Server startup in XXX ms" という表示が出たところで、起動に成功しています。

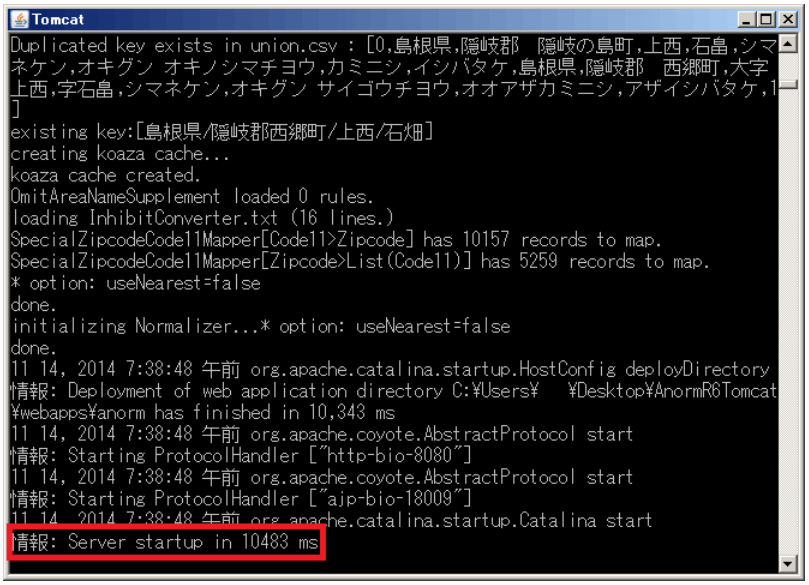


図 起動に成功した

住所マスタの読み込みメッセージがないまま起動した場合、住所正規化コンバータは正しく動作しません。web.xmlの設定（住所マスタフォルダの位置）を見直してください。

動作テスト

Webブラウザから住所正規化コンバータサーバを開きます。

```
http://localhost:8080/anorm/
```

図 テスト用画面を開く

住所文字列を入力し、「正規化処理を開始」ボタンを押下します。正規化が行われているかどうか確認してください。

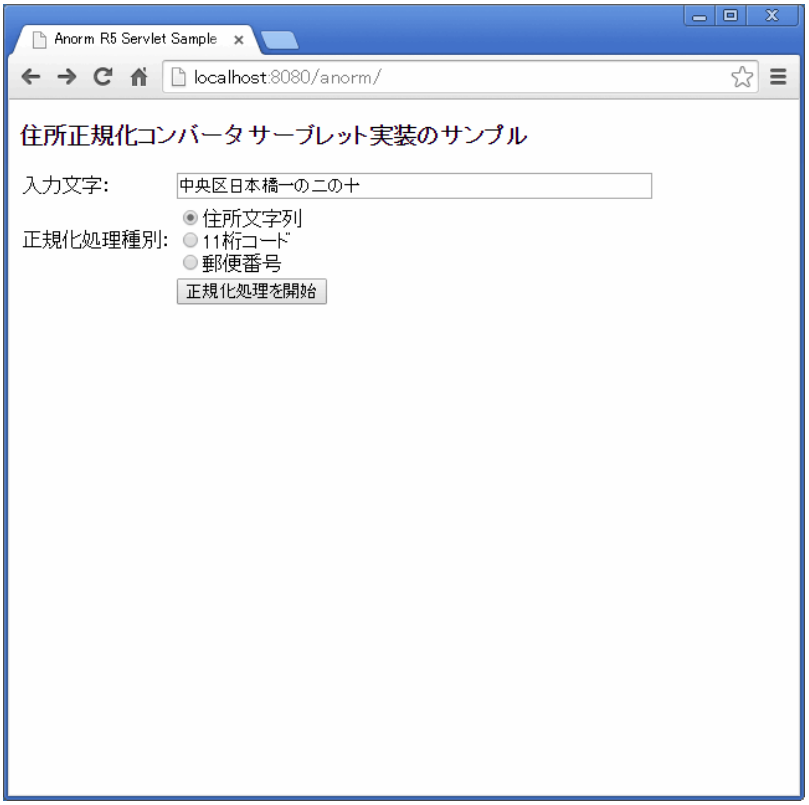


図 動作テスト(1)

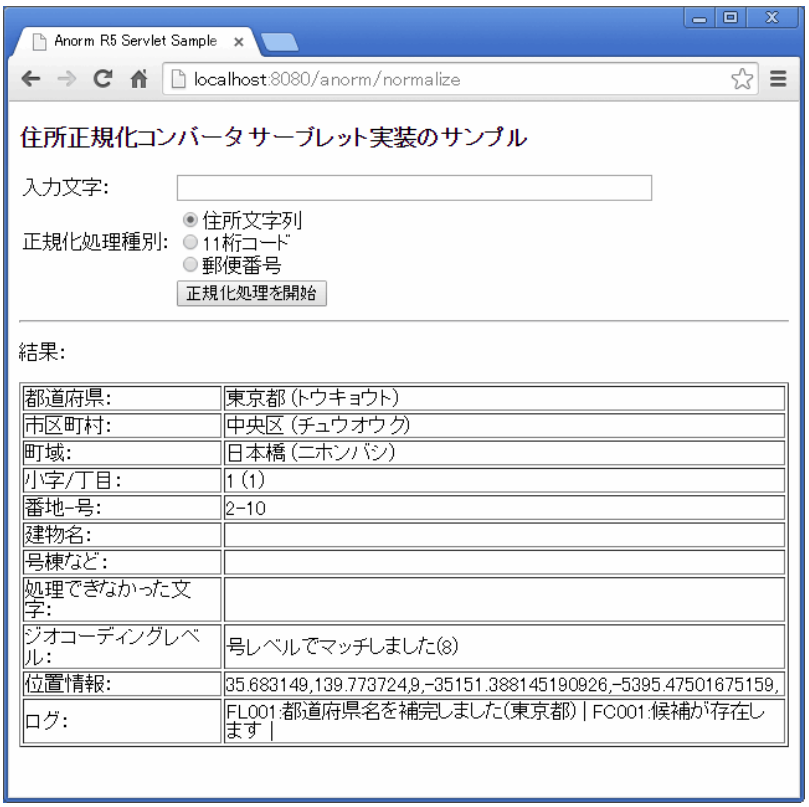


図 動作テスト(2)

続いてREST APIのテストを行います。アドレスバーに直接、次のように入力します。（入力パラメータは郵便番号です。）

```
http://localhost:8080/anorm/v1/anorm?zipcode=1030027
```

結果がJSON表記で返されれば動作テストは終了です。

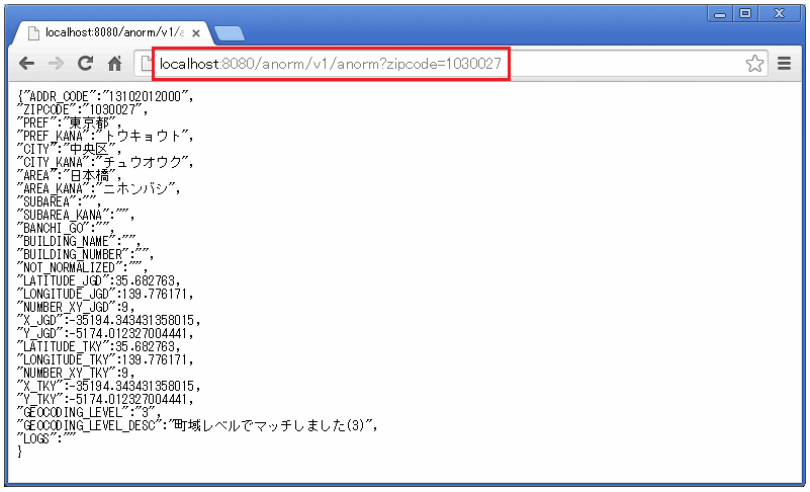


図 RESTAPIのテスト

Wagbyから住所正規化コンバータを利用する場合、この REST API 機能を使います。

ファイル型の詳細

内部の構造

Wagby でファイル型の項目を定義するとテーブルには二つのカラムが用意されます。

- 一つ目のカラムには、オリジナルのファイル名を保存する。
- もう一つのカラムには、upload_dir フォルダ内の物理的なファイル名を記録する。

ファイル本体はデータベースに格納されません。upload_dirというフォルダ内にモデルIDのフォルダが用意され、その中に一時ファイル名が付与されて保存されます。

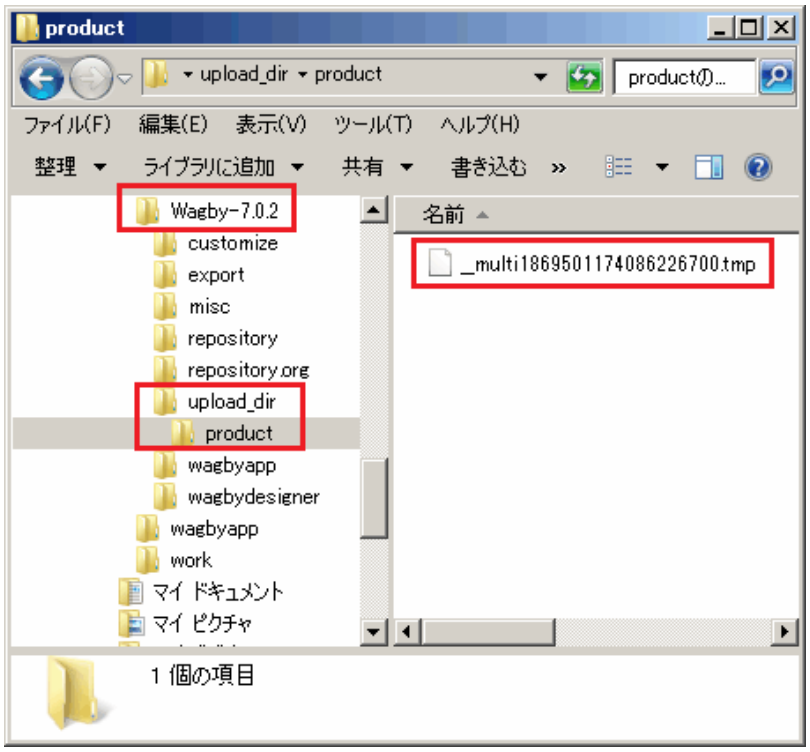


図 upload_dirフォルダ

ファイル型項目を作成すると、対応する（データベースの）テーブルには二つの項目が用意されます。例を示します。

モデル product にファイル型項目 photo を用意したとします。このとき、テーブル product が作成されますが、項目は次のように二つ用意されます。

項目名	説明	例
photo	ファイル名が格納されます。	duke3D.jpg
photo_jshfilename	ファイルの存在するパスを示します。起点は wagbyapp/bin となっています。	../../upload_dir/product/__multi1869501174086226700.tmp

(534)

オリジナルファイル名をそのまま使う

「オリジナルファイル名をそのまま使う」設定時の upload_dir フォルダを図に示します。アップロード時のファイル名で格納されています。

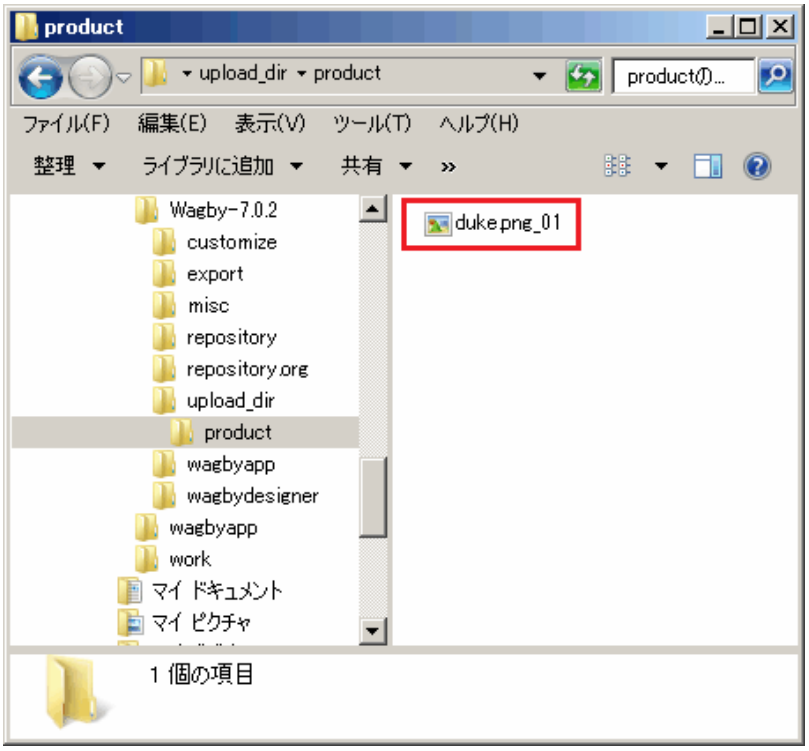


図 オリジナルファイル名をそのまま使う設定にした場合のupload_dirフォルダの内容

「オリジナルファイル名をそのまま使う」設定の注意点

- 本番環境（サーバ）では、日本語名を扱える OS （例：Windows OS）を用いる必要があります。
- 対象項目を検索条件とした場合、全角、半角を区別してご利用ください。入力フィルタは適用されません。（フィルタは内部で強制的に「文字列（最小限）」となります。）
- オリジナルファイル名利用時に同名のファイルが登録された場合、自動的に "_1", "_2", "_3",... とファイル名を区分するための接尾語が付与されます。

運用途中でファイル名の扱いを「オリジナルファイル名を残す」方式に変更する場合

運用後に「ファイル名にオリジナルファイル名をそのまま使う」ように設定を変更して再ビルドし、アプリケーションを入れ替えた場合は次のようになります。

- その前まで登録されたファイルは拡張子が .tmp のファイル名のまま。

(534) この例では upload_dir に保存される実際のファイルは一時ファイル名(拡張子tmp)となっています。これをアップロードしたファイル名自身を使うようにすることもできます。次の節で説明します。

・　アプリケーション入れ替え後に登録したファイルはオリジナルファイル名として保存される。

過去に登録された .tmp ファイル名を、オリジナルファイル名に戻すためのツールを提供しています。 詳細は"ファイル名の扱いをオリジナルファイル名を残す方式に変更する"をお読みください。

ファイルの実体の扱い

ファイルの削除処理を行なった場合、データベースからファイル名の情報は削除されますが、ファイルの実体は残っています。

ファイルの実体を upload_dir フォルダに残す理由は次のとおりです。

・　コピー登録機能を用いた場合、コピー元のデータを削除したときにコピーされたデータからファイルの実体が存在しなくなった、ということを防ぐ必要がある。
・　ファイル型項目の参照連動・自モデル保存の場合、連動元と連動先でファイルの二重管理を行う必要がある。連動元をコピーしても連動先から参照するファイルの実体は残っている必要がある。

物理的なファイルを定期的に削除する

残っているファイル（の実体）を定期的に削除するジョブが用意されています。 "サポート > 管理者ガイド(R8) > 参照されていないファイルの削除ジョブ"をお読みください。

物理的なファイルを同時に削除する

データ削除のタイミングでファイルの実体を削除する方法も用意されています。この方法は上述した「ファイルの実体を upload_dir フォルダに残す理由」に該当しない場合にのみ行なってください。

カスタマイズフォルダ customize/resources に myapplication.properties ファイルを用意し、次の行を加えます。

R8.4以降

```
server.servlet.context-parameters.file_delete_mode=UnsavedFile
```

R8.3.xまで

```
server.context-parameters.file_delete_mode=UnsavedFile
```

上記設定を追加後、ビルドを行なってください。この設定はアプリケーション全体に適用されます。

(535)

制約

一覧更新、アップロード更新、一括更新、メール受信、メール送信でのファイル削除には未対応です。また、カスタマイズコードで Dao/Service を使った場合も未対応です。これらは将来の Wagby で対応する計画です。

保存先フォルダの変更

自動生成された　WEB-INF¥classes¥savedir.properties.UTF8　ファイルをテキストエディタで開きます。次のように記載されています。

```
# ライセンスファイル
savedir.jfclicenseholder=../../upload_dir/jfclicenseholder
# メールテンプレート
savedir.jfcmailtemplate=../../upload_dir/jfcmailtemplate
# 帳票テンプレート
savedir.jfcreporttemplate=../../upload_dir/jfcreporttemplate
# 顧客
```

(535) 運用途中に直接 application.properteis ファイルを変更することもできます。変更（し、再起動した）後の操作で、ファイルの削除処理が機能します。ただし、変更前のファイルは upload_dir フォルダに残ったままとなります。過去データについては、テーブルに格納されているファイル型項目を直接、SQL で検索し、参照されていないファイルを手動で削除してください。

```
savedir.customer=../../upload_dir/customer
```

標準では、起点ディレクトリ (wagbyapp¥bin) からの相対パスとなっています。保存先はモデル毎に指定できるようになっています。

ここで、変更したいモデルのみエントリを残して編集します。変更が不要なモデルは、もともとのファイルから記述を削除します。 例えば「顧客」モデルのみ保存先を変更したい場合は、次のようになります。

```
# 顧客
savedir.customer=C://Users//YOURNAME//Desktop//newfolder
```

編集したファイルを customize¥resources フォルダに、mysavedir.properties.UTF8 というファイル名で保存します。

カスタマイズファイル名の注意点

カスタマイズした場合は mysavedir.properties として保存してください。接頭語 my の詳細はこちらをお読みください。

savedir.properties ファイルを直接書き換えてカスタマイズフォルダに含めてしまうと、新規モデルを追加したときに、そのモデルのリソース(savedir.XXXXX)が含まれないこととなり、実行時にエラーとなります。

運用開始後に変更した場合

運用途中に mysavedir.properties.UTF8 ファイルを作成した場合、次のルールとなります。

- 変更前に登録したファイルは、変更前に指定されていたフォルダを参照します。
- 変更後に登録したファイルは、新しく指定されたフォルダを参照します。

このため、変更前に指定されていたフォルダは消さないようにご注意ください。

その他の情報

"サポート > Wagby Developer Network(R8) > Javaを用いたカスタマイズ > リソースファイル"をお読みください。

MIMEタイプ (8.3.1)

ファイル項目のダウンロードに設定されるMIMEタイプはWEB-INF/mime.typesに設定されています。このルールは次の通りです。

- pdf,rtf,tsv,text/plain,Microsoft Officeファイル,画像ファイルはブラウザで表示することを許容する。
- それ以外のファイル（例:HTMLファイル,XMLファイルなどが含まれる）を添付（アップロード）したとき、この添付したファイルは画面に表示せず、クリック時にダウンロードされるようにする。
- Flash形式(拡張子がflv)はインラインで表示しない。

クロスサイトスクリプティング脆弱性につながる懸念があったため、原則は許可されたファイル以外の動作はダウンロードに統一しています。

一意制約の詳細

データベースによる挙動の違い

一意制約設定を行った項目の一部を未入力(null)として保存した場合の動作は、データベースによって異なります。

項目A, B, Cに一意制約を定義します。ここで A:aaa, B:null, C:ccc というデータを登録します。 その後、A:aaa, B:null, C:ccc のデータを再び登録しようとすると、次のようになります。

内蔵データベース(HSQLDB)、PostgreSQL

登録可能。

SQLServer

一意制約違反で登録に失敗する。

詳細は、ご利用のデータベースのマニュアルをお読みください。

生成されるDDL

ここで指定した一意制約の設定は、次のファイルで確認できます。

```
wagbydesigner/webapps/wagbydesigner/WEB-INF/env/work/dbschema/createddl/モデルID.ddl
```

テキストエディタでファイルの内容を確認することができます。次の DDL が含まれています。

```
alter table "モデルID"
  add constraint jfcuk_モデルID_設定した名前 unique ("項目名");
```

init_db.bat を実行すると、この alter 文も同時に実行されます。

一意制約名の自動調整

一意制約名は、利用するデータベースが許可する制約名の最大長に合わせて調整されます。

具体的には最大長を超える名前の場合、短縮されます。（短縮の結果、重複する場合には、これを重複しないように一意な名前を割り当てます。これはテーブル名や項目名でも同様です。）

仕様・制約

- 繰り返しコンテナ内項目に一意制約を設定することはできません。（この要件では、一意制約のチェックはスクリプトなどで別途、行うようにしてください。）

入力フィルタの詳細

標準のフィルタルールを変更する

project_filter.xls を変更することで、標準のフィルタルールをカスタマイズすることができます。

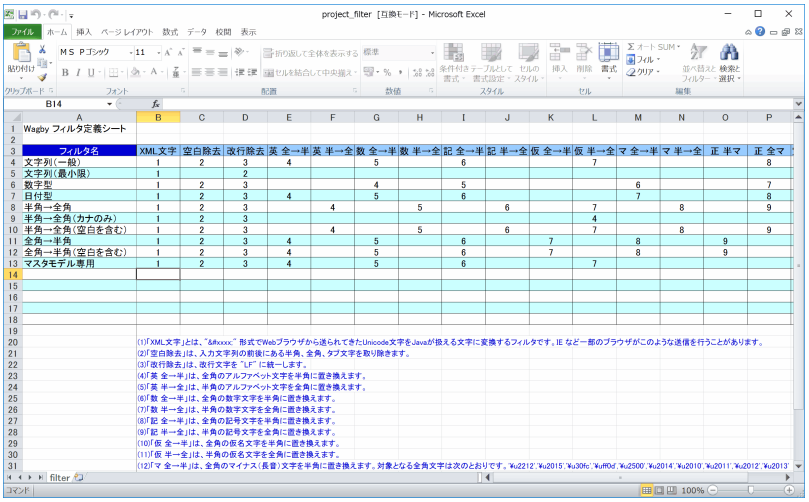


図 project_filter.xls

ダウンロード

・　project_filter.zip (9,349 bytes)

ダウンロードしたファイルを解凍し、customize/resources フォルダに保存します。下記の仕様にあわせて、標準のフィルタルールを変更することができます。ビルドを行うと、カスタマイズしたフィルタルールが適用されます。

仕様

project_filter.xls の仕様は次の通りです。

・　新しいフィルタを定義することはできません。既存のフィルタ（「フィルタ名」に含まれるもの）の適用ルールを変更することができます。
・　各数値は、適用の順番を意味します。この数値を変えることで、ルールが変わります。
・　数値を消去すると、当該フィルタルールは使われなくなります。このとき、数字の並びが順序よく整列されるようにしてください。

全文検索のインデックス

格納場所

全文検索のために利用する検索用インデックスファイルは次の場所に格納されます。

```
wagbyapp¥webapps¥wagby¥WEB-INF¥idxdir¥モデル名
```

格納場所を指定する (8.3.1)

設定ファイル application.properties に wagby.MakeTextIndex.indexDir を指定することができます。次の例はexportフォルダなどと同じ位置にインデックスフォルダを作成します。

```
wagby.MakeTextIndex.indexDir=../../idxdir
```

実際に変更する場合はカスタマイズフォルダ customize/resources に myapplication.properties ファイルを用意し、上の行を含めるようにしてください。（差分ファイル方式を推奨します。）

(536)

変更されるタイミング

インデックスファイルは次のタイミングで変更されます。

・　データの変更(新規登録、更新、削除)
・　インポート処理
・　インデックスの最適化処理

データベースを直接更新した場合

全文検索（およびサジェスト）対象の項目の内容を、SQLなどで直接編集した場合、Wagbyはこれを検知することができません。そのためインデックスを更新することができません。この場合は手動でインデックスファイルの再作成を行なってください。

ファイルサイズ

データをインデックスに登録すると、データ1件に使用されるサイズは、登録対象サイズの最大約7.2倍になります。 このことから、1モデルで使用するインデックスサイズは次のように見積りできます。

```
インデックスサイズ(バイト) = データ件数 × 平均登録対象サイズ × 7.2
```

(536) wagbyapp/webapps/<プロジェクト識別子>/WEB-INF/classes/application.properties に反映されます。

平均登録対象サイズは、全文検索の対象となる項目に格納される文字列数および添付文書に含まれるテキスト文字列数からおおよその数値を出してください。

インデックスファイルのサイズは、扱うデータのサイズより大きくなります。 特に添付ファイルの全文検索機能を有効にするとインデックスのサイズは巨大になります。 データベース用のディスク領域とあわせてインデックス用のディスク領域も十分確保されているかを確認してください。

生成ルール

全文検索で利用するインデックスファイルと、サジェストで利用するインデックスファイルは別に用意されています。

全文検索

N-gram インデックス方式を採用しています。N の値は 2 (bigram) を用いています。 これは、文字列の先頭から1文字ずつずらしながら、2文字ずつ抜き出したものを検索語として登録するものです。

例えば ″東京都″ という文字列の場合「東京」「京都」が検索語になります。具体的な検索動作は次のようになります。

- ″東″ ではヒットしない。
- ″東京″ ではヒットする。
- ″京″ ではヒットしない。
- ″京都″ ではヒットする。
- ″東京都″ ではヒットする。

ただし半角アルファベット文字についてはスペース等の区切り文字で分割したものを登録します。 例えば ″orange pineapple banana″ の場合、″orange″, ″pineapple″, ″banana″ に分割し、それぞれを検索語として登録します。このため検索語に ″apple″ を入力した場合はヒットしませんが、″pine″ はヒットします。

サジェスト

文字列を分割せずにそのまま検索語として登録します。 なお、文字列が 64 文字を超える場合は、先頭から64 文字を登録します。

サジェストの場合は先頭からの一致検索になります。例えば ″北九州″ という文字列に対して ″北″ はヒットしますが、″九州″ はヒットしません。

最適化処理

インデックスファイルは定期的に最適化処理を行うことで、全文検索の実行速度を常に最良に保つことができます。 Wagbyは次のタイミングで最適化処理を行います。

- Web アプリケーションが起動した時。
- インデックス最適化を行うジョブが起動した時。

アプリケーションを無停止で稼働させる場合は、定期的にインデックス最適化ジョブを実行するようにしてください。

手動によるインデックスファイルの再作成

何らかの理由でインデックスファイルが破損した場合、次の手順で再作成することができます。

インデックスファイルは検索用のファイルですので、破損してもデータがおかしくなることはありません。

1. Webアプリケーションを停止します。
2. インデックスファイルを削除します。

特定モデルのインデックスを最適化する場合、例えばsampleモデルのインデックスを最適化する場合には、次のフォルダを削除します。

```
wagbyapp¥webapps¥wagby¥WEB-INF¥idxdir¥sample
```

またインデックス全体を最適化する場合には次のフォルダ全体を削除します。

```
wagbyapp¥webapps¥wagby¥WEB-INF¥idxdir
```

3. Webアプリケーションを起動します。初期化処理時にインデックスの作成の後、インデックスの最適化が行われます。

運用時の注意 (R8.0.5)

全文検索を有効にしたとき、Wagbyアプリケーション起動時に検索用インデックスファイルを作成するようになります。この処理が終わらないうちに（全文検索を使った）モデルの登録・更新・削除を行うと、検索用インデックスファイルが作成完了するまで待機します。また、処理が終わらない状態で検索を行なっても（インデックスファイルへの反映が行われていないため）正しく検索されないこともあります。

このような動作のため、全文検索を有効とした場合は Wagby アプリケーションをメンテナンスモードで起動させ、ログ(system.log)に"インデックス作成完了"と出力されるまで待つという運用を推奨します。

具体的にはインデックスファイルの作成が完了すると次のようなログが出力されます。

```
[INFO jp.jasminesoft.jfc.textsearch.servlet.MakeTextIndexThread run] (admin@localhost) finished making textsearch index.
```

このログを確認後、メンテナンスモードを終了させ、一般利用者に解放するとよいでしょう。

組織図の表示

表示できるグループ数の上限 (8.1.2)

1. wagbyapp/webapps/<プロジェクトID>/jfcgroupnode フォルダ内の showListJfcgroupnode_javascript.jsp をテキストエディタで編集します。
2. 331行目と349行目にグループとグループ階層情報を取得する処理があります。

```
request("rest/jgroup/list?pagesizejshparam=200&action_Search_Condition=", {
...
request("rest/jfcgroupnode/list?pagesizejshparam=200&action_Search_Condition=", {
```

pagesizejshparam の値が取得件数となっています。200の部分を書き換え、取得する件数を変更してください。 変更後、ファイルを保存して画面を再描画し、動作を確認してください。

3. 変更したファイルを wagbydesigner/bin/webpage/jfcgroupnodeフォルダに保存します。変更直後はフルビルドを行ってください。（次回以降は差分ビルドとなります。）

選択肢の無効化マーク

選択肢の無効化マークを変更する

1. WagbyDesignerを停止します。
2. wagbydesigner/webapps/wagbydesigner/WEB-INF/classes/gencompatibility.properties ファイルをテキストエディタで開きます。
3. リソースキー #jp.jasminesoft.jfc.gen.JavaSourceGenerator.invalidMark= (-) の行の先頭の「#」を除きます。あわせて、標準値 " (-)" を変更します。
4. 修正したファイルを保存後、WagbyDesignerを再起動します。
5. フルビルドを行います。

[ご注意ください] この設定で、イコールの右側を空文字とすると、無効化マークが表示されなくなります。この場合、利用者はこの選択肢が無効状態かどうかを判断できなくなるため、空文字とすることは非推奨です。

関連する設定

- 「他モデルの参照 > 選択肢モデル > 参照先モデルのデータが削除されていた場合」

プリンシパルの詳細

プリンシパルを追加したときに行うこと

開発者が追加したプリンシパルは、データベースに用意された "jprincipal" というテーブルに格納されます。このモデルは通常、外部から操作することはできませんが、テーブル情報としては存在しています。アカウント情報（juser）のインポート・エクスポート時には、jprincipal も操作されます。

jprincipalの主キー(プリンシパルID)はビルドのタイミングで自動採番されます。（"プリンシパル名（英語）" のソート順になります。）

外部データベース利用時は、ビルド後に import_db.bat を実行することで、追加したプリンシパルがアプリケーションに反映されます。 詳細は"サポート > データベース活用ガイド(R8) > テーブルの作成 >二回目以降"の手順をお読みください。

内蔵データベース(HSQLDB)利用時は、この手順は自動的に行われます。

詳細な仕組み

「ログオンアカウントにプリンシパルを割り当てる」にあるように、アカウントには複数のプリンシパルを紐づけることができます。内部ではjuserは、複数のプリンシパルIDを保持します。

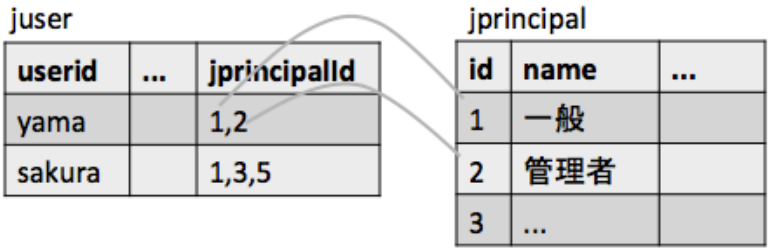


図 juserとjprincipalの関係

このように juser は jprincipal の ID 値を参照する仕組みですが、新たに jprincipal が追加された場合、juser がもっている ID 値に変更がないにもかかわらず、jprincipal の ID 値が降り直されるため、「ずれ」が生じる可能性があります。

そこで Wagby では、プリンシパルの追加に伴いプリンシパルIDが変わった場合でも、通常のアプリケーション入れ替えルール（エクスポートとインポート処理）を行うことで、整合性が維持されるようにしています。

1. ビルド前のアプリケーションで、データをエクスポートしておきます。
2. ビルド後のアプリケーションで、データをインポートします。インポート時に、インポート対象データのプリンシパル情報と、（ビルド後の）アプリケーションのプリンシパル情報を比較して、インポート対象データのアカウント(juser)のプリンシパルIDを自動的に変換します。

最小限の操作

この仕組みから、プリンシパルの追加時は、アカウントモデル(juser)のインポートを行うことで反映されます。具体的には次の操作になります。

1. ビルド前のアプリケーションでアカウントモデル(juser)をエクスポートしておく。
2. ビルド及びアプリケーションの入れ替えを行う。
3. 新しいアプリケーションで 1. のアカウントモデル(juser)をインポートする。

3. のタイミングで、追加されたプリンシパルが自動的に反映されます。

(537)

(537) import_db.bat を使って juser のみのインポートを行うことはできません。juser のみを対象とするインポートは、ビルド後の Wagby アプリケーションを使うことができます。システム管理者で Wagby アプリケーションにログオンし、"管理処理 > インポートとエクスポート" 画面から "アカウント" のみにチェックをつけインポート処理を行うことができます。このとき juser と jprincipal テーブルのインポートが行われ、プリンシパルIDが再セットされます。

プリンシパルの削除または名前の変更

上で説明したようにプリンシパルの追加はエクスポートとインポートによってプリンシパルIDのずれを吸収しますが、プリンシパルの削除もしくはプリンシパル名の変更は手動で対応する必要があります。

具体的にはアカウントモデル(juser)がもつプリンシパル情報をすべて消去し、新しくプリンシパルを再設定してください。

(538)

プリンシパルモデルの詳細

プリンシパルモデルは Wagby システム内部で管理されており、開発者がこの定義を変更することはできません。内部では次のようになっています。

R8.4.0以降

項目	型	説明
id	整数	順序による自動採番
principalId	文字列	識別用の文字
content	文字列	表示用の文字
category	整数	カテゴリID
isShowInSelect	真偽値	開発者による権限設定で選択肢として表示するかどうか
isEditable	真偽値	プリンシパル設計画面で編集可能かどうか

R8.3.xまで

項目	型	説明
id	整数	順序による自動採番
name	文字列	識別用の文字
content	文字列	表示用の文字

(539)

内部表現

プリンシパルをXML形式で表現すると次のようになります。

```
<?xml version="1.0" encoding="UTF-8"?>
<jprincipal id="1000">
  <principalId>productmgr</principalId>
  <content>{"en":"productmgr","ja":"商品管理者"}</content>
  <category>7</category>
  <isShowInSelect>true</isShowInSelect>
  <isEditable>true</isEditable>
</jprincipal>
```

- id部は自動的に割り当てられます。
- content部は日本語と英語を含む国際化の特別表記形式となっています。詳細は "メッセージの変更・国際化 > 選択肢モデルの特別表記" をお読みください。8.1.2
- category部にはカテゴリIDが適用されます。

プリンシパルIDのルール (8.4.2)

(538) プリンシパルIDの変更は、主キーの変更と同じインパクトがあります。そのため追加は可能ですが削除や変更は影響が大きいものとなっています。

(539) R8.4.0以降のWagbyへ移行すると、nameは削除されます。

システム系のプリンシパルのIDは1番から999番までとし、これは予約します。ユーザが独自に作成したプリンシパルIDは1000番から割り当てられます。なお、このプリンシパルIDは内部で自動採番されるものであり、開発者が通常、意識することはありません。

外部システムから直接、Wagby の任意の画面を開く

URLを指定する

外部システムから直接、Wagby 上の画面を開くことができます。URL に次の情報を含めます。

キー	説明
forwardUrl	遷移先画面に関する情報

例えば顧客モデル(customer)の一覧表示画面を開く場合、次のようになります。ホスト名は適切に読み替えてください。

```
http://localhost:8921/wagby/logon.do?forwardUrl=showListCustomer
```

最初にログオン画面が表示されます。ログオン後、メニュー画面をスキップして顧客モデル(customer)の一覧表示画面が開きます。

- パラメータ　forwardUrl　は「画面名」を記載してください。ここで、画面名とはログオン後のメニュー選択時に URL に表示される部分に相当します。
- 遷移先画面用の追加パラメータを指定しない場合、末尾の ".do" は不要です。
- forwardUrl パラメータを省略した場合は、メインメニュー画面へ遷移します。
- パラメータにログオンアカウントおよびパスワードを指定することはできません。

例:新規登録画面を開く

URL に action_New パラメータを加えます。値部分も同じく "New" とします。

```
http://localhost:8921/wagby/logon.do?forwardUrl=insertCustomer.do?action_New=New
```

接頭語 "action_" のあとに続く文字列は「イベント」として扱われます。 詳細は"画面遷移パラメータまとめ"をお読みください。

例:検索条件を付与する

forwardUrl パラメータ内に含める&は、"%26" に置換してください。これは forwardUrl パラメータ以降の文字列をひとまとまりとして扱えるようにするためです。

```
http://localhost:8921/wagby/logon.do?forwardUrl=showListCustomer.do?
action_Search_Condition=Search_Condition%26customer_cp$002f整数型項目1jshparam=整数
値%26customer_cp$002f整数型項目2jshparam=整数値
```

- 上例で、検索条件を格納するモデル名は customer_cp と表現しています。詳細は「コンディションモデル」をお読みください。
- 上例の「整数型項目」は適切に読み替えてください。Wagbyでは整数型項目は範囲を指定することができます。開始と終了を、それぞれ接尾語 "1jshparam", "2jshparam" を付与することで区別します。
- コンディションモデル名と項目名の間は "$002f" で区切ります。これは '/' 文字を utf-8 でエンコードした表現となっています。

例:日本語文字を含む検索条件を付与する

遷移先を showListStaff.do とします。また、検索条件部を次のようにします。

```
?action_Search_Condition=Search_Condition&staff_cp$002fname=鈴木 一郎
```

まずパラメータのマルチバイト文字列「鈴木　一郎」のみをURLエンコードします。文字エンコーディングは utf-8 としてください。

```
?action_Search_Condition=Search_Condition&staff_cp$002fname=%E9%88%B4%E6%9C
%A8%20%E4%B8%80%E9%83%8E
```

次に「画面URL」と「パラメータ」を連結します。

```
showListStaffSub.do?action_Search_Condition=Search_Condition&staff_cp$002fname=%E9%88%B4%E6%9C
%A8%20%E4%B8%80%E9%83%8E
```

この文字列をもう一回URLエンコードします。

```
showListStaffSub.do%3Faction_Search_Condition%3DSearch_Condition%26amp%3Bstaff_cp%24002fname%3D
%25E9%2588%25B4%25E6%259C%25A8%2520%25E4%25B8%2580%25E9%2583%258E
```

このようにして作成された文字列を logon.do の forwardURL のパラメータ値として指定します。

```
http://localhost:8921/wagby/logon.do?forwardUrl=showListStaffSub.do%3Faction_Search_Condition
%3DSearch_Condition%26amp%3Bstaff_cp%24002fname%3D%25E9%2588%25B4%25E6%259C
%25A8%2520%25E4%25B8%2580%25E9%2583%258E
```

(540)

まとめますと、日本語文字(マルチバイト文字)は、2回URLエンコードした値に置換してください。

- [参考] TECH-UNLIMiTED すぐに使える便利なWEBツール URLエンコード・デコード

スマートフォンからの操作

スマートフォンの場合は forwardUrl 指定は無効となります。つまりログオン認証後、指定画面へリダイレクトされず、メニュー画面が表示されます。

JSPおよびエラー画面のカスタマイズ

JSPのカスタマイズ

標準で同梱されている、または Wagby が生成した JSP ファイルの修正は、常に上書き保存になります。保存先フォルダの起点は次のとおりです。

```
customize¥webapp
```

このフォルダを起点に、生成されたアプリケーションのフォルダ構成に対応したカスタマイズファイルを保存してください。

> 自動生成されたファイルをカスタマイズしたあとにモデル定義を変更した場合、追加した定義にあわせてカスタマイズファイルも手動で修正する必要があります。そのため上書き保存方式によるカスタマイズはモデルの構造が固まったあとで行うことを推奨します。

エラー画面(JSP)のカスタマイズ

Wagbyが提供する標準エラー画面は次のとおりです。これをカスタマイズすることで表示内容を変更することができます。

ファイル (*1)	説明
error/4xx.jsp	HTTPステータスコードが400系の場合に表示されます。
error/5xx.jsp	HTTPステータスコードが500系の場合に表示されます。

(540) forwardUrl に指定するパラメータ値のうち「=」「$」「?」「.」はURLエンコードしなくても動作します。つまりURLエンコードが必要な文字は「&」「%」及びマルチバイト文字となります。

(541)

仕様・制約

401,402などのコードにあわせた個別のメッセージを表示する場合は、4xx.jsp内にif文を記述して振り分けるようにしてください。

ビルド時のメモリサイズの指定

設定方法

ソースコードの生成・コンパイル・データベースのテーブル作成といったビルド処理の途中で、次のようなエラーメッセージがコンソールに表示されることがあります。

```
システム・リソースが不足しています。
詳細は次のスタック・トレースで調査してください。
java.lang.OutOfMemoryError: GC overhead limit exceeded
```

この場合、Wagby Designer 用のメモリを増やしてください。具体的な手順は次のとおりです。

1. Wagbyをインストールしたフォルダ内にある wagbydesigner/bin フォルダを開きます。
2. その中に含まれる setenv.bat (Linux,Mac OS X系は setenv.sh) をテキストエディタで開きます。
3. -Xmxと-Xmsの後ろにある数字を大きくします。数字の後ろについている "m" はメガという単位を意味します。初期値である 4096m は、4096メガバイトすなわち4Gバイトのメモリを使うことを意味します。

```
set CATALINA_OPTS=-server -Xmx4096m -Xms1024m
```

例えば、8Gバイトのメモリを使う設定は次のようになります。

```
set CATALINA_OPTS=-server -Xmx8192m -Xms1024m
```

4. 同ファイルの編集を保存し、WagbyDesignerを再起動します。

32 bit Java での利用はサポートしていません。32 bit Java をご利用の場合、-Xmx 指定の初期値は 1024m が上限となります。ビルド処理のパフォーマンスが著しく低下するため、64 bit Java を利用してください。（稼働 OS も 64bit 版である必要があります。）

ログコード一覧

ログコード

コード	区分	説明
JFC-00001	INFO	ログオンに成功したときに記録されます。利用しているロケール情報も記録されます。
JFC-00002	INFO	ログオンしていない状態であることを伝えています。
JFC-00003	INFO	ログオン認証時、アカウント名が指定されていません。
JFC-00004	INFO	ログオン認証時、アカウント名に空白が含まれています。（アカウント名に空白を含めることはできません。）
JFC-00005	INFO	ログオンボタンが連続して複数回、押下された可能性があります。

(541) 1. これらのファイルはビルドしたアプリケーション wagbyapp/webapps/$APPNAME 直下に用意されています。

JFC-00006	INFO	ログオン認証に失敗しました。
JFC-00007	ERROR	ログオンコンテキストの生成に失敗しました。[詳細] wagbyapp/conf/app_jaas.config の読み込みができていない可能性があります。通常、この設定は wagbyapp/bin/setenv.{bat,sh} 内で環境変数を指定して行なっています。Wagbyが提供するTomcat環境ではなく、独自に環境を構築した場合、Wagbyが起動時に設定しているすべての環境変数を同じように指定するようにしてください。
JFC-00008	INFO	アカウントの有効期限が切れています。システム管理者により、このアカウントを復帰させる必要があります。
JFC-00010	INFO	ログオン認証を試みたが、登録済みのパスワードと一致しません。利用者によるパスワード忘れです。
JFC-00011	ERROR	ログオン認証エラー時にパスワードエラー回数をセットするが、その書き込みに失敗しました。 データベース接続に問題がある可能性があります。
JFC-00012	ERROR	ログオン認証後、ユーザ情報の読み込みに失敗しました。データベース接続に問題があるか、ユーザ情報が破損している可能性があります。
JFC-00013	ERROR	予期しないエラーにより、ログオン認証に失敗しました。メモリ不足など、その失敗の前後に不正な状態になっている可能性があります。
JFC-00014	ERROR	アカウント情報がシステムに存在していません。 Wagby のアカウント情報は juser というテーブルに保存されています。データベースを直接参照して、内容を確認してください。 古いデータのまま処理を行なったなどの可能性があります。
JFC-00015	INFO	パスワードのエラー回数が許容範囲を超えたため、アカウントがロックされました。詳細はアカウントロックの説明をお読みください。
JFC-00016	ERROR	予期しないエラーにより、ログオン認証に失敗しました。メモリ不足など、その失敗の前後に不正な状態になっている可能性があります。
JFC-00017	ERROR	ログオン認証後、ユーザ情報の読み込みに失敗しました。データベース接続に問題があ

		るか、ユーザ情報が破損している可能性があります。
JFC-00018	INFO	接続解除ユーザとして機能していません。詳細はログオン接続の解除を行う特別なアカウントを用意するをお読み下さい。
JFC-00019	INFO	お使いの Wagby の同時認証接続ライセンスの制限です。ライセンスの追加により解消されます。
JFC-00020	INFO	同一アカウントで同時にログオンできる許容数を超えました。初期パラメータ > 同一ログオンアカウントでシステムにログオンできる最大数を調整してください。
JFC-00021	INFO	同一アカウントでの多重ログオンが禁止されています。初期パラメータ > 同一ログオンアカウントでシステムにログオンできる最大数を調整してください。
JFC-00022	INFO	同一アカウントで同時にログオンできる許容数を超えました。初期パラメータ > 同一ログオンアカウントでシステムにログオンできる最大数を調整してください。
JFC-00023	INFO	ログオン後、何も操作をしなかったため、自動的にログアウトされました。このメッセージが頻繁に表示される場合、初期パラメータ > セッションタイムアウトを調整してください。
JFC-00024	INFO	パスワードの有効期限が切れています。
JFC-00025	INFO	パスワードの有効期限が近づいています。
JFC-00026	ERROR	セッションIDを取得できないという内部エラーです。
JFC-00027	INFO	ログオフ時に表示されるメッセージです。
JFC-00028	INFO	ログオフ時に表示されるメッセージです。
JFC-00029	WARN	ログオフ処理に失敗したという内部エラーです。システム運用は継続されます。
JFC-00030	ERROR	パスワードの有効期限の取得に失敗したという内部エラーです。このエラーの前後でデータベース接続エラーやメモリ不足といった異常状態が検出されていないか、ログを確認してください。
JFC-00031	ERROR	無効なセッション ID を使おうとしました。 これは長時間アクセスがなかったユーザをシステムが自動ログオフさせた後、そのユーザがそのことに気づかずにそのまま操作を

		行った場合に出力されます。該当ユーザはログオン画面に遷移され、画面上部には下記のメッセージが表示されます。″接続の有効時間が切れました(もしくは管理者によって接続が解除されました)。再ログオンしてください。″自動ログオフ以外にも、システム管理者がログオンユーザ管理画面よりユーザの接続を解除した後、解除されたユーザがそのまま画面操作を行った場合にも同ログが出力されます。その際も操作ユーザの画面には上記と同じメッセージが表示されます。
JFC-00032	INFO	システムに登録された直後のログインでは、プレファレンス情報が用意されていません。これは問題ではありません。ログオン後、自動的にプレファレンス情報が作成されます。
JFC-00034	ERROR	ログオンルールの検出中に内部エラーが発生しました。ルールの書式誤りの可能性があります。
JFC-00035	INFO	ログオンルールに合致しませんでした。詳細はIPアドレスによるログオン制限の説明をお読みください。
JFC-00036	INFO	ジョブ実行アカウントでログオンしようとしました。詳細はジョブ実行アカウントの説明をお読みください。
JFC-00037	ERROR	パスワード再設定画面で入力されたユーザへのメール送信が失敗しました。詳細はパスワードリマインダの説明をお読みください。
JFC-00038	INFO	パスワードリマインダ機能で、ユーザのメールアドレスが取得できませんでした。パスワードリマインド機能の処理は停止されました。
JFC-00039	INFO	パスワードリマインダ機能で、ユーザのメールアドレスが不正な形式でした。パスワードリマインド機能の処理は停止されました。
JFC-00040	INFO	パスワードリマインダ機能で、無効になったユーザでパスワード変更処理を行おうとしました。パスワードリマインド機能の処理は停止されました。
JFC-00041	ERROR	パスワードリマインダ機能で、不正な値でパスワード変更処理を行おうとしました。パスワードリマインド機能の処理は停止されました。

JFC-00042	INFO	パスワードリマインダ機能で、無効になった値でパスワード変更処理を行おうとしました。パスワードリマインド機能の処理は停止されました。
JFC-00043	INFO	パスワードリマインダ機能で、新しいパスワードを登録しました。処理が正しく成功したことを示します。
JFC-00044	ERROR	パスワードリマインダ機能で、新しいパスワードの登録に失敗しました。内部エラー発生もしくは画面に表示されるエラーメッセージが一つ以上ある場合にこのログが記録されます。前者（内部エラー発生）時は、ログにその原因を示すスタックトレースが記録されます。
JFC-00045	INFO	パスワードリマインダ機能で、アカウントにパスワード変更権限がなかったため、この処理を停止しました。
JFC-00046	INFO	パスワードリマインダ機能で、存在しないアカウントを処理しようとしました。この処理を停止しました。
JFC-00047	INFO	パスワードリマインダ機能で、指定したアカウントは本機能が有効となっていません。この処理を停止しました。
JFC-00048	ERROR	パスワード強制変更フラグがセットされていますが、パスワード変更権限が付与されていない状態です。システム管理者が当該アカウントの設定を適切に変更する必要があります。
JFC-00049	ERROR	このアカウントにプリンシパルが設定されていないため、メニューの表示に失敗しました。システム管理者がこのアカウントのプリンシパルを行う必要があります。
JFC-00063	WARN	juser テーブルの操作中に問題が発生しました。テーブル情報が誤っているか、データベースとの接続が行えていない可能性があります。
JFC-00064	WARN	jprincipal テーブルの操作中に問題が発生しました。テーブル情報が誤っているか、データベースとの接続が行えていない可能性があります。
JFC-00065	WARN	lastaccesstime テーブルへの書き込み処理中に問題が発生しました。テーブル情報が誤っているか、データベースとの接続が行えていない可能性があります。

JFC-00066	WARN	プリンシパル情報の操作中に問題が発生しました。プリンシパル名が古いか、スペルミスの可能性があります。
JFC-00067	WARN	パーミッション情報の操作中に問題が発生しました。パーミッション名が古いか、スペルミスの可能性があります。
JFC-00068	ERROR	サブジェクトポリシーの生成に失敗しました。subjectpolicy.xml ファイルが存在していないか、破損している可能性があります。
JFC-00069	WARN	ロックされているオブジェクトを再ロックしようとしました。システム内部エラーです。ブラウザの「戻る」ボタン等で、システムが想定していないタイミングで更新処理が行われた可能性があります。正常な操作にも関わらず本エラーが発生する場合は、サポートへお問い合わせ下さい。
JFC-00070	WARN	ロックされているオブジェクトを再ロックしようとしました。システム内部エラーです。ブラウザの「戻る」ボタン等で、システムが想定していないタイミングで更新処理が行われた可能性があります。正常な操作にも関わらず本エラーが発生する場合は、サポートへお問い合わせ下さい。
JFC-00071	WARN	ロックされていることを期待されていましたが、ロックされていませんでした。システム内部エラーです。ブラウザの「戻る」ボタン等で、システムが想定していないタイミングで更新処理が行われた可能性があります。正常な操作にも関わらず本エラーが発生する場合は、サポートへお問い合わせ下さい。
JFC-00072	WARN	ロックされていることを期待されていましたが、ロックされていませんでした。システム内部エラーです。ブラウザの「戻る」ボタン等で、システムが想定していないタイミングで更新処理が行われた可能性があります。正常な操作にも関わらず本エラーが発生する場合は、サポートへお問い合わせ下さい。
JFC-00073	WARN	ダウンロード対象ファイルが存在しないか、破損しているため、正しく読み込むことができません。

JFC-00074	WARN	ダウンロードファイルの読み込み処理が失敗しました。ディスク障害などの可能性があります。
JFC-00075	WARN	ダウンロードファイルをブラウザへ送信している途中で処理が失敗しました。サーバに何らかの障害が発生した可能性があります。
JFC-00076	WARN	ダウンロードファイルをブラウザへ送信している途中で処理が失敗しました。ネットワーク障害などの可能性があります。
JFC-00077	WARN	ロックされていることを期待されていましたが、ロックされていませんでした。システム内部エラーです。ブラウザの「戻る」ボタン等で、システムが想定していないタイミングで更新処理が行われた可能性があります。正常な操作にも関わらず本エラーが発生する場合は、サポートへお問い合わせ下さい。
JFC-00078	ERROR	メニュー情報の読み込みに失敗しました。menu.xmlファイルが存在しないか、破損している可能性があります。
JFC-00079	WARN	指定したメニューの情報が見つかりません。古いmenu.xmlファイルを利用している可能性があります。
JFC-00080	WARN	メニュー情報が利用できません。menu.xmlファイルが誤っている可能性があります。
JFC-00081	WARN	メニュー操作処理が失敗しました。jp.jasminesoft.jfc.menu.JFCAppMenuのログレベルをDEBUGとすると詳細な情報を得ることができます。
JFC-00100	WARN	システム内部で処理できなかった例外が発生しました。ログファイルをサポートに送付してください。
JFC-00101	WARN	セキュリティ例外が発生しました。開発者が意図したセキュリティ設定の範囲内であれば問題ありません。そうでない場合は、サポートにお問い合わせください。
JFC-00102	WARN	認証されていないアクセスの可能性があります。
JFC-00103JFC-00104	ERROR	画面表示用の文字列情報がリソースファイルに含まれていません。開発者によってリソースファイルを手動で修正した場合は、リソースキーが意図通りかどうかを確認ください。フルビルドを行うことで解決する可能性もあります。

JFC-00105	ERROR	画面表示用の文字列情報がリソースファイルに含まれていません。開発者によってリソースファイルを手動で修正した場合は、リソースキーが意図通りかどうかを確認ください。フルビルドを行うことで解決する可能性もあります。
JFC-00108	WARN	セキュリティエラーが発生しました。利用者の使い方を確認してください。
JFC-00110JFC-00111JFC-00112JFC	ERROR	データベース接続不良です。データベースサービスが停止している可能性があります。またはネットワーク環境に問題が発生している可能性があります。
JFC-00204	ERROR	ファイルアップロード処理で問題が発生しました。ログに記録されているスタックトレースを確認してください。
JFC-00205	WARN	画面遷移時に問題が発生しました。開発者によりカスタマイズされている場合は、内容を再確認してください。
JFC-00206	WARN	画面遷移時に問題が発生しました。次の画面遷移先がnullとなっています。開発者によりカスタマイズされている場合は、内容を再確認してください。
JFC-00207	WARN	画面処理中に予期しない例外が発生しました。開発者によりカスタマイズされている場合は、内容を再確認してください。
JFC-00208	WARN	セッションがタイムアウト状態になった可能性があります。この場合は問題ではなく、仕様どおりの動作となります。
JFC-00210	WARN	指定されたメールIDに対応するメールテンプレートが見つかりません。メール送信処理はスキップします。
JFC-00211	WARN	休日データの読み込みに失敗しました。
JFC-00212	WARN	アップロード更新の処理に失敗しました。
JFC-00213	WARN	アップロード更新の後処理(processEndメソッド)に失敗しました。
JFC-00214	ERROR	jfcbase.properties ファイルのパラメータ jp.jasminesoft.jfc.ResultBean.DEFAULT_CA の指定に誤りがあります。
JFC-00215JFC-00216JFC-00217JFC	WARN	検索結果集合を扱う ResultBeanクラスで予期しないエラーが発生しました。
JFC-00222JFC-00223JFC-00224	WARN	全文検索処理（またはサジェスト処理）で、ユーザが不

		適切なキーワードを入力したか、またはシステム側に問題が発生しています。許容できないキーワードが入力された場合は、ユーザがキーワードを変えることで引き続きシステムを利用できます。
JFC-00225JFC-00226JFC-00227JFC	WARN	データベースの順序を扱うSequenceクラスで予期しないエラーが発生しました。
JFC-00228JFC-00229JFC-00230JFC	WARN	データベースの順序をテーブルで管理するSeqクラスで予期しないエラーが発生しました。
JFC-00237JFC-00239JFC-00241JFC	WARN	メール送信処理中に内部エラーが発生しました。
JFC-00238	WARN	所定のディレクトリが存在しません。
JFC-00240	WARN	所定のファイルが存在しません。
JFC-00243JFC-00244JFC-00250JFC	WARN	Excel帳票処理で内部エラーが発生しました。
JFC-00249	WARN	パスワードリマインダ機能は無効となっています。
JFC-00601	INFO	統計情報を確認しました。
JFC-00756	INFO	指定したエクスポートディレクトリの削除に成功しました。
JFC-00757	INFO	指定したエクスポートファイル(Zip形式)の削除に成功しました。
JFC-00758	ERROR	指定したエクスポートディレクトリの削除に失敗しました。
JFC-00759	ERROR	指定したエクスポートファイル(Zip形式)の削除に失敗しました。
JFC-00800..JFC-00820	ERROR	クラスタリング処理中に内部エラーが発生しました。
JFC-00900	INFO	ジョブは正常に終了しました。
JFC-00901	INFO	ジョブは失敗しました。
JFC-00902	INFO	ジョブを開始しました。
JFC-00903	WARN	ジョブは実行中です。
JFC-00904	WARN	ジョブを実行しようとしましたが、ジョブコマンドが指定されていませんでした。（ジョブはスキップされます。）
JFC-00905	WARN	ジョブを実行しようとしましたが、指定された作業ディレクトリが存在しませんでした（ジョブはスキップされます。）
JFC-00907JFC-00908JFC-00909JFC	ERROR	ジョブ実行中に内部エラーが発生しました。
JFC-00912,913,914,915	WARN	メール送信処理が失敗しました。
JFC-90008JFC-90009JFC-90010JFC	ERROR	ご利用いただいているライセンスキーを作成したときの機器と、異なる機器で動作させようとしています。Wagby が

		許容しているライセンス条項の範囲内であれば、ライセンスキーの再作成を行なってください。
JFC-90011	ERROR	ライセンスキーファイルの更新に失敗しました。Wagby販売パートナーにご連絡ください。
JFC-90012	INFO	Web経由でライセンスキーファイルの更新を行ないます。
JFC-90013	INFO	通常モードからメンテナンスモードへ入ります。
JFC-90014	INFO	メンテナンスモードから通常モードへ戻ります。
JFC-90015	INFO	現在ログオンしている利用者数を示します。
JFC-90016	WARN	アカウントライセンスが0となっています。現在、システムにログオンすることはできません。
JFC-90017	INFO	登録ユーザー数がアカウントライセンスの上限を超えているため、ログオンに失敗しました。
JFC-90020	INFO	登録ユーザー数がアカウントライセンスの上限に達しているため、ユーザーの追加は行うことができません。
JFC-90021	WARN	実行前にメンテナンスモードに入ろうとしましたが、ログオンしているユーザがいたため、ジョブを実行することができません。
JFC-90022	INFO	実行前にログオンしていたアカウントを強制ログアウトさせました。
JFC-90023	ERROR	実行前にアカウントを強制ログアウトさせようとしましたが失敗しました。

変更履歴

R8.3.1

- JFC-00033を廃止し、JFC-00015に集約する。これまでのJFC-00033のメッセージはJFC-00015が担う。JFC-00033はこれまで出力されることがなかった。

パフォーマンスチューニング

パフォーマンスチューニング

大量の選択肢を持つリストボックス・ラジオボタンをつくらない

問題点

Wagbyが提供するリストボックスやラジオボタンは、大量データの選択には向いていません。ユーザーインタフェースの操作性から、選択肢として画面に表示される数は100件以下が妥当といえます。

このリストボックス（またはラジオボタン）を読み込み専用または隠し項目とし、かつ100件を超えるデータを保持させるような設計を行った場合、内部では選択肢を用意するため大量の SQL が発行されます。このような項目を多く含むモデル定義では、選択肢のデータ量が増加するに従い、パフォーマンスに問題が生じます。

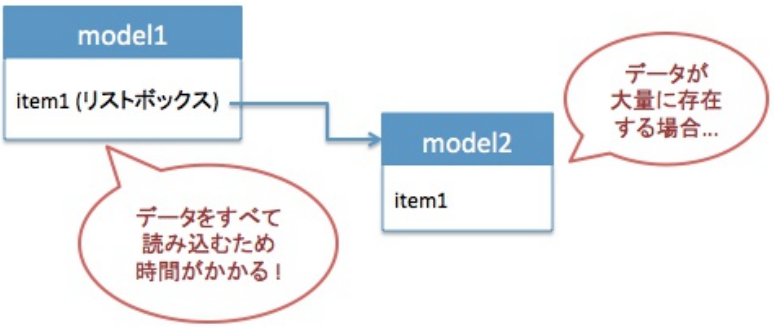


図 大量の選択肢をもったリストボックス

検索条件にも含めない !-- 2019.12.15 -->

参照先モデルのデータ数が多いリストボックス（またはラジオボタン）を検索条件に含めた場合、そのモデルの一覧表示画面を開くときに時間がかかります。親子モデル関係にある親モデルの子モデル同時表示では、子のモデルの処理が遅くなることから、結果的に親モデルの画面を開くときに時間がかかるようになります。

解決策

大量の選択肢を持つモデル参照項目はリストボックスやラジオボタンではなく「検索画面」として定義してください。特に、もともと読み込み専用または隠し項目として設定していた場合は、見た目の影響もありません。

「検索画面」は選択肢を事前に作成しないため、SQL発行数を削減することができます。

リストボックスの絞込設定の方式を見直す

問題点

Wagby では登録・更新画面でのリストボックス等の絞込方法として2通りの定義方法を提供しています。

- [a] 参照モデル同士の関係性で選択肢を絞り込む
- [b] 他項目の値で選択肢を絞り込む

[a] 方式は、いったん全ての選択肢データをデータベースから取得してから絞り込みを行います。そのため、大量の選択肢をもつ場合には不向きです。

[b] 方式は SQL の where 句を使って絞り込みを行うため、大量の選択肢があった場合でもパフォーマンスの問題が発生しにくいようになっています。

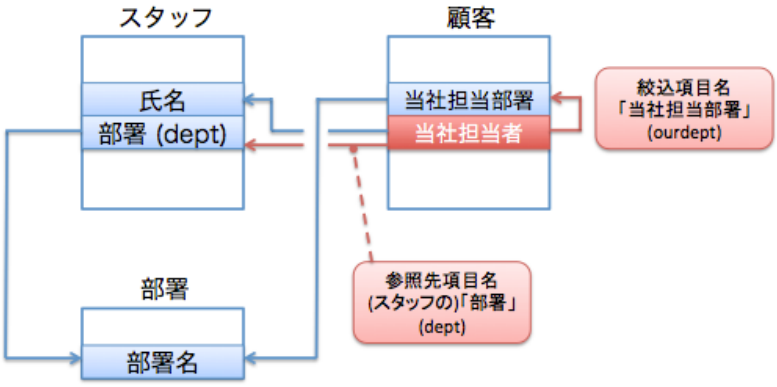


図 「当社担当部署」を指定すると「当社担当者」の選択肢が変わる例

解決策

選択肢の数が多い場合は［b］方式の利用を推奨します。

被参照モデルのサブモデルを作成する

問題点

モデルAが、モデルBを参照している関係を例に説明します。ここでモデルBはマスター用途のモデルとします。

被参照モデル(B)に繰り返しコンテナ、チェックボックス項目などが定義されているとき、モデルAから当該モデルを参照するタイミングで、モデルBのこれら項目全てを SQL を使って読み込みます。しかし、モデルA 側では、これらの項目を利用しないことがわかっている場合、この SQL 処理は不要なものです。

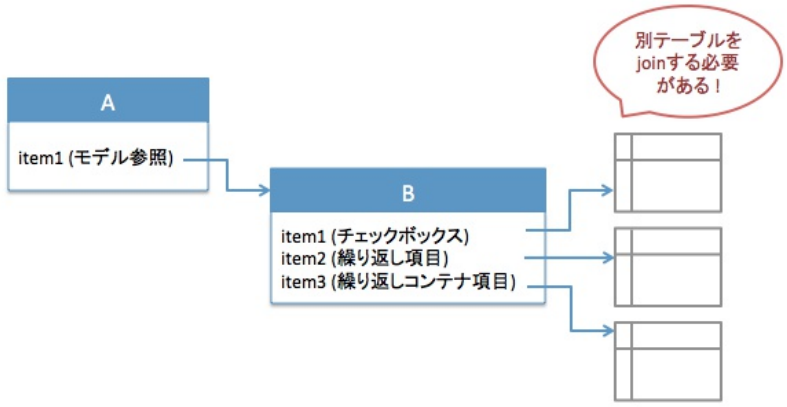


図 別テーブルのjoinが必要な場合

例えば、被参照モデルに「データの変更履歴」を保持させたとします。しかし多くの場合、これを参照するモデルでは、変更履歴情報を使うことはありません。しかし処理上は、被参照モデルをアクセスするタイミングで必ずデータの変更履歴を保持する繰り返しコンテナ（これは別テーブルです）も SQL で操作するため、パフォーマンスの観点からは好ましくありません。

解決策

被参照モデルに、必要最低限の項目のみを保持したサブモデルを用意します。

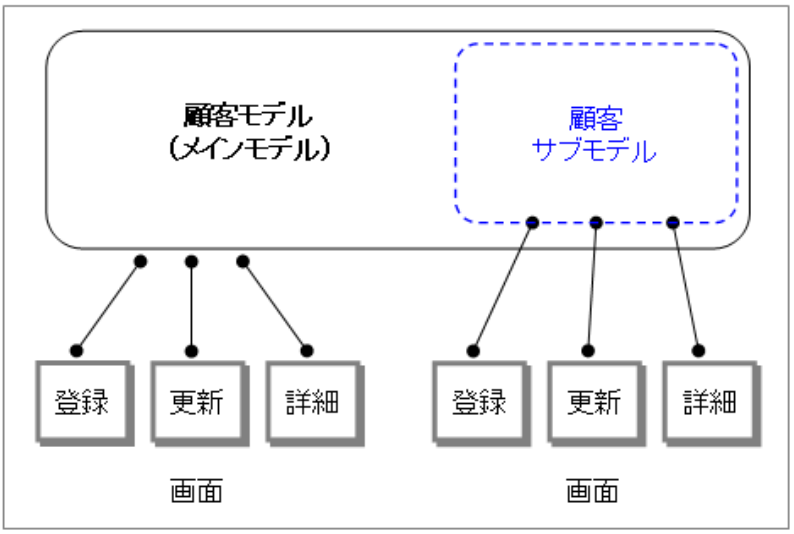


図 メインモデルとサブモデル

サブモデルには主キー、非参照項目、表示優先度項目、無効判定項目、絞込項目等のモデル参照の際に利用される最低限の項目のみを定義します。参照モデルからは、このサブモデルを使うことで、不要な処理を抑制することができます。

ビューを使う場合でもキャッシュを有効にする

問題点

外部データベースにあらかじめビューを用意しておき、Wagbyのモデルがこのビューを参照するような設計を行うことができます。

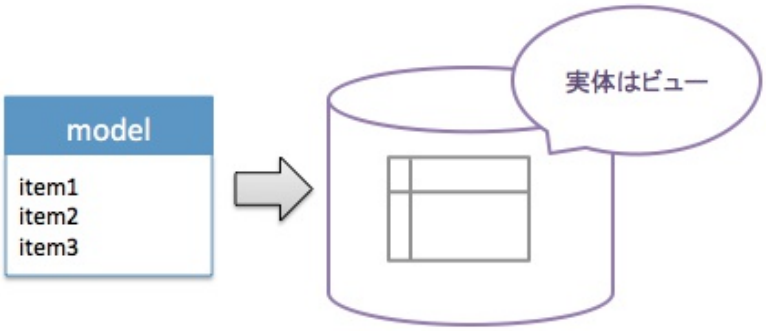


図 ビューを参照させる

ここでビューのデータが不定期に更新される場合、Wagbyのモデルキャッシュ機構によって過去のデータを表示してしまう懸念が生じます。そのためモデルキャッシュを無効とすることができますが、これは当該モデルへの参照の都度、SQL を発行するため、パフォーマンス低下の要因になります。

解決策

次の方法をご検討ください。

1. Wagbyのモデルキャッシュを有効とし、ジョブを使ってキャッシュ情報を定期的に消去する。
2. データベースが提供する「マテリアライズドビュー」を使う。

詳細は"モデル > ビューの利用"をお読みください。

前後のデータにアクセスするボタンの動作

詳細画面で「前後のデータにアクセスする」ボタンを用意することができます。

Wagbyの一般的な画面操作は、(1) 検索 (2) 一覧表示 (3) 詳細表示となります。 ここで前後のデータにアクセスするボタンは、(2) の一覧表示の処理で取得した検索結果を使って、自分の前と後のデータを把握します。

この手順を踏まずに、直接 (3) の詳細表示画面へ遷移する場合、改めて内部で一覧データを取得する処理を行います。そのため、詳細画面の表示に時間を要します。

前後のデータにアクセスするボタンを無効とすることで、詳細画面の表示パフォーマンスを改善することができます。（標準では、本機能は有効となっています。）

SQL式の見直し

複雑なSQL式はパフォーマンス低下の原因になることがあります。また、SQL式の項目を一覧表示のソート項目にしないような工夫も必要です。

集合処理関数の見直し

親子モデル関係で、子モデルの要素全体に対する集合処理（COUNT や SUM など）を行なっており、第二引数に条件を指定する場合、モデル参照項目のID部を使って判断する処理は時間がかかります。子モデル一覧表示に ID 部を保持する隠し項目を用意し、この項目の値を使って判断するようにすることでパフォーマンスの向上につなげることができます。

またはWagbyが提供するCOUNTやSUM関数を使わず、代わりにスクリプトを使って計算結果を求めるという方法もあります。スクリプトでは適切な検索条件をセットしてDaoを呼び出すか、または直接、SQLを記述します。(利用するデータベースが提供するCOUNT,SUM関数を使うようにします。)

スクリプトのリロード方式の変更 (8.2.2)

Wagby のスクリプトは実行時に毎回、スクリプトファイルの日付をチェックし、更新があった場合は再読み込みを行うようになっています。このチェック処理を行わないようにすることで、パフォーマンスを向上させることができます。

アプリケーションの起動時間を短縮する (8.3.2)

ビルドされたアプリケーション(wagbyapp)は、生成されたソースコードをコンパイルした .class ファイルが WEB-INF/classes フォルダに格納されています。これを一つの jar ファイルにまとめることで、起動時間を短縮することができます。

boostコマンド

Wagbyをインストールしたフォルダ内に misc フォルダが含まれています。コマンドプロンプトを開き、misc フォルダに移動します。次のコマンドを実行します。

```
boost
```

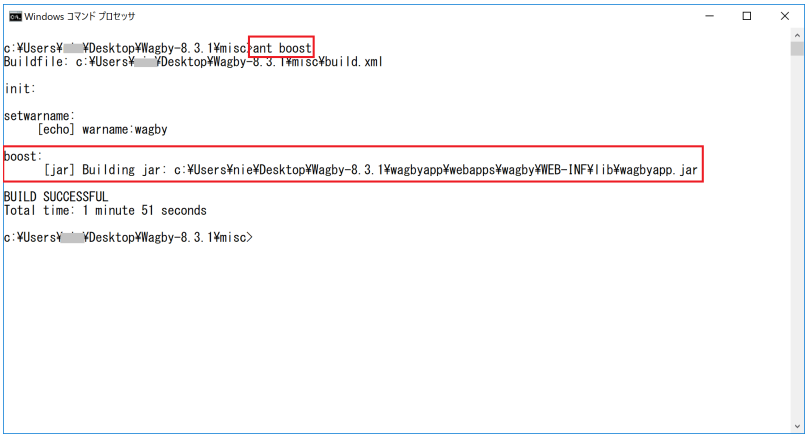


図 boostの実行

このタスクは WEB-INF/classes のファイル群をまとめ、WEB-INF/lib フォルダに wagbyapp.jar を生成します。WEB-INF/classes 以下の .class ファイルは削除されます。これによって起動時間が高速になります。

その後のビルド処理

その後、Designer に戻ってモデル定義を変更し、差分ビルドを行うと WEB-INF/lib/wagbyapp.jar が再び WEB-INF/classes に展開されてからコンパイル処理が行われます。（注意:この展開のための時間が余分にかかります。）

フルビルドの場合は WEB-INF/classes および WEB-INF/lib/wagbyapp.jar の両方を消去し、クリーンな状態にします。

仕様・制約

boost 後は、check_db.bat などデータベース運用のための CUI コマンドが実行できません。これらのコマンドは boost 前に実行するようにしてください。または次のコマンドで、boost 状態を元に戻すことができます。

```
unboost
```