実践 3『登録画面（紐付け）』を作成する

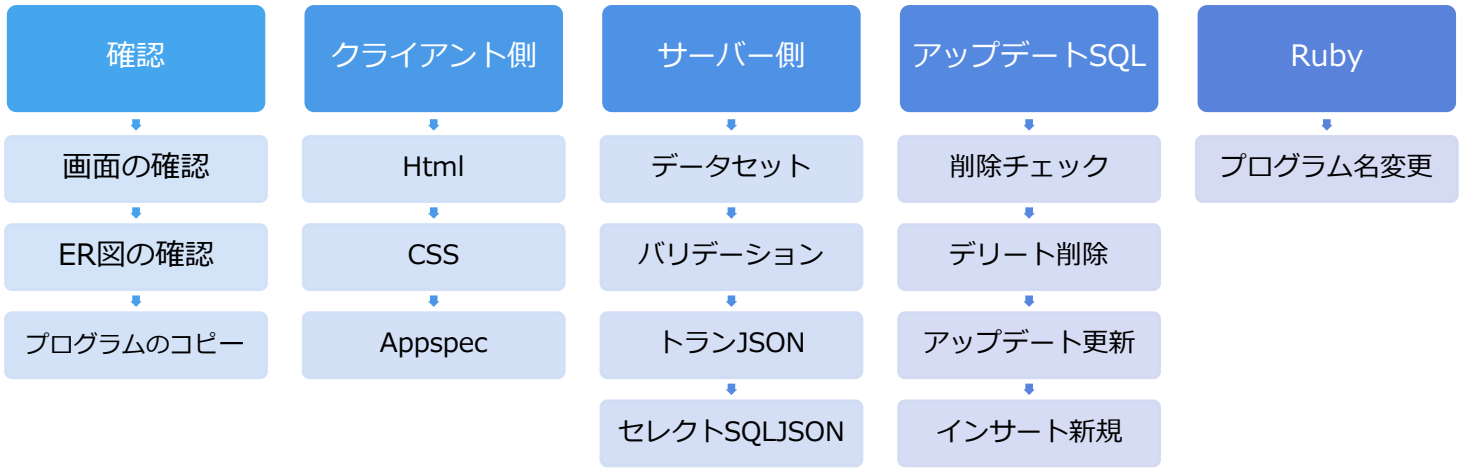
事前確認

手順

手順は実践 2 とほとんど同じです。紐付け部分が違うので、紐付け箇所を中心に作成箇所を確認します。

実践 1 の手順で『REST 一覧画面』を作成します。

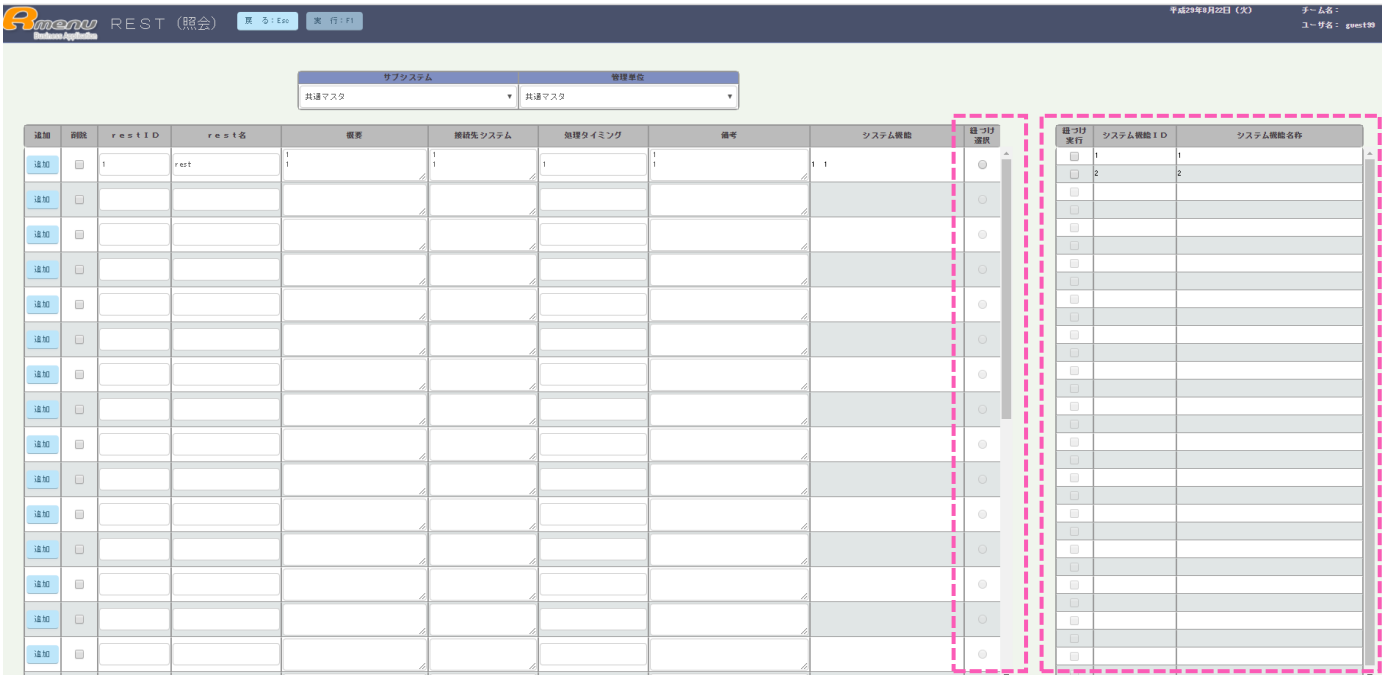
その後実践 3 で『REST 登録画面（紐付け）』を作成します。



画面の確認する『REST 登録画面』

［画面・REST 一覧］

［画面・REST 登録訂正/照会］



実践 3 で作成するのはこの登録画面です。

実践 2 と手順はほとんど同じですが、点線内が紐付け画面で追加されている機能です。

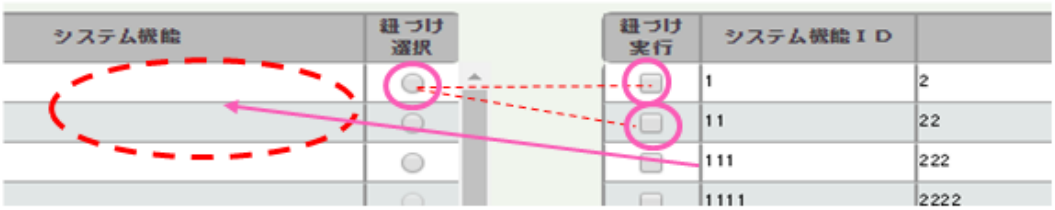
ここを実践 3 で作成します。

紐付けしたい行のチェックを行うと右のテーブルのデータのどれを紐付けるのかを

選択でき、実行をクリックすると DB へ紐付けた内容が登録されます。

［作成する画面の確認］

「基本の更新画面で作成したメインテーブルの横に紐付け用のテーブルの有る画面を作成する」



「メインテーブルで紐付けにチェックをしてから、紐付けテーブルで紐付けたい行にチェックを入れると紐付けができる
紐付けできたデータはメインテーブル側に内容が表示される仕組み」

※紐付け選択チェック欄では「入出力要素no(rest)」をkeyにする
紐付け実行チェック欄では「プロセス要素no(機能)」をkeyにする　この二つを紐付ける機能

ER 図を確認する

作成する画面の『REST』テーブルを中心に ER 図を確認。

紐付けは『**REST**』画面から『**システム機能**』と紐付けをおこないます。

上記 2 つのテーブルの間に『**システム機能_入出力要素**』テーブルがあります。

ここに紐付けしたデータが入ります。

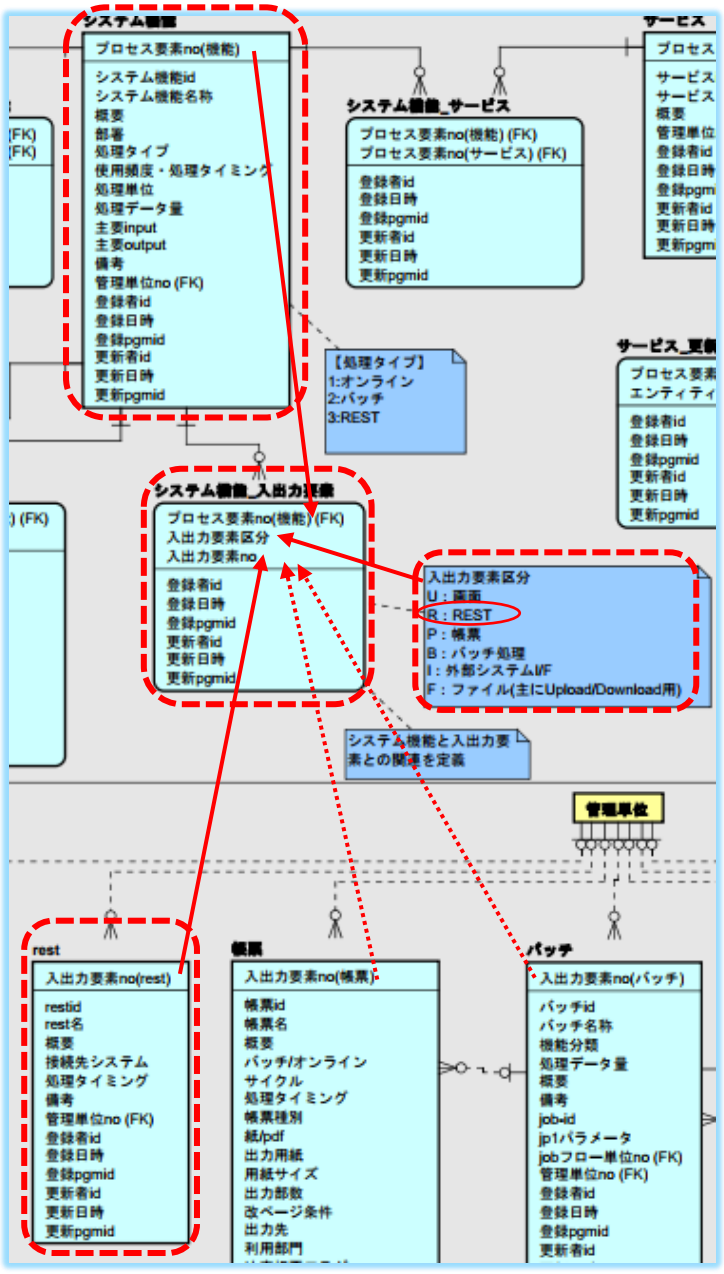
このテーブルは **REST 以外のテーブル**からも**システム機能**へ紐付けデータを残すため、

どれを紐付けたか仕分けが必要になり、『**入出力要素区分**』でどのテーブルの

入出力要素ｎｏなのかを仕分け管理しています。

ＳＱＬを作成するときは入出力要素区分にきをつけて作成します。

（コピー元のままで作成いてしまうとデータが混ざってしまう為）



「シーケンス　＝　サロゲートキー　とは」

記で使用している［入出力要素ｎｏ］はシーケンスになります。

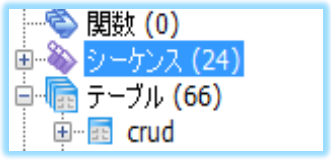
このレコードはデータが増えた時に自動でナンバーを作ってくれる役割です。

ｎｅｘｔｖａｌ　＝　自動的にＮＯを作ってくれるｋｅｙ　（今持っているＮＯに+1 追加していく）

ポスグレを開き、てーぶるを確認すると下記図の様に nextval があります。

```
CREATE TABLE public."画面"
(
  "入出力要素no(画面)" integer NOT NULL DEFAULT nextval('"画面_入出力要素no(画面)_seq"
  "画面id" character varying(20) NOT NULL,
  "画面名" character varying(60),
```

ここでも確認できます。シーケンスの中で確認できる。



自動的に NO を作成する様子が SQL ビューで確認できます。

スタートに今までカウントしている件数がわかります。

次に入れたデータは 5 になります。

SQLビュー

```
1   -- Sequence: public."画面_入出力要素no(画面)_seq"
2
3   -- DROP SEQUENCE public."画面_入出力要素no(画面)_seq";
4
5   CREATE SEQUENCE public."画面_入出力要素no(画面)_seq"
6     INCREMENT 1
7     MINVALUE 1
8     MAXVALUE 9223372036854775807
9     START 4
10    CACHE 1;
11  ALTER TABLE public."画面_入出力要素no(画面)_seq"
12    OWNER TO postgres;
13
```

インサート時はプラス１でｎｏを作成

アップデート時は４は４のままで内容だけを更新できる。

プログラムのコピー

「紐付け登録画面を作成する。プログラム名は **DG_620R** と **DG_630U** です。

DG_620R は一覧画面作成と同じ。DG_630U は登録画面の紐付け機能付きです。」

1.紐付け画面の雛型［DG_600R］から［**DG_620R**］をコピーする

2.紐付け画面の雛型［DG_610U］から［**DG_630U**］をコピーする

DG_620R は実践 1 を参考に作成します。

DG_630U を実践 3 で作成します。

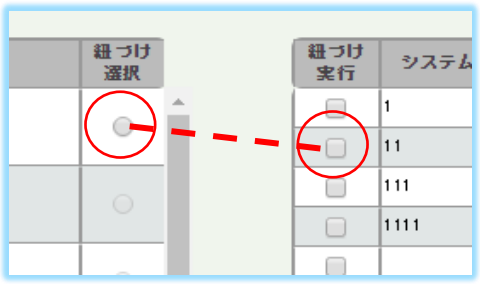
※コピーの手順は実践 1 に記載済

クライアント側を作成する

Html

紐付け部分のクライアント側を確認します。

※他は実践 1 と同じ手順で作成



紐付け部分があるためテーブルが 2 つに分かれて作られています。

［テーブル 1 の Html］

```
<ul class="nav nav-pills rmenu-ul"> <!-- menu-ul スタート -->↓
  <li class="rmenu-li"> <!-- menu-li スタート -->↓

    <div id="scrollable-TableH1"> <!-- テーブル スクロール スタート -->↓
      <table id="mainTableH1" class="table table-condensed rmenuTable">↓
        <thead>↓
          <tr>↓
            <th class="width50">追加</th>↓
            <th class="width50">削除</th>↓
            <th class="width100">ｒｅｓｔＩＤ</th>↓
            <th class="width150">ｒｅｓｔ名</th>↓
            <th class="width200">概要</th>↓
            <th class="width150">接続先システム</th>↓
            <th class="width150">処理タイミング</th>↓
            <th class="width220">備考</th>↓
            <th class="width210">システム機能</th>↓
            <th class="width50">↓
              <div>紐づけ</div>↓
              <div>選択</div>↓
            </th>↓
          </tr>↓
        </thead>↓
      </table>↓
    </div> <!-- テーブル スクロール エンド -->↓

    <div id="scrollable-TableD1"> <!-- テーブル スクロール スタート -->↓
      <table id="mainTable1" class="table table-condensed rmenuTable">↓
        <tbody>↓
          <tr>↓
            <td class="width50">↓
              <input name="行追加" type="button" class="行追加 btn color4 btn-sm rfocusblue" value="追加">↓
            </td>↓
            <td class="width50">↓
              <input name="削除" type="checkbox" class="削除 rfocusblue" value="削除">↓
            </td>↓
            <td class="width100">↓
              <input name="ｒｅｓｔＩＤ" type="text" value="" class=" form-control input-sm rfocusblue rmenuLeft ｒｅｓｔＩＤ">↓
            </td>↓
            <td class="width150">↓
              <input name="ｒｅｓｔ名" type="text" value="" class=" form-control input-sm rfocusblue rmenuLeft ｒｅｓｔ名">↓
            </td>↓
            <td class="width200">↓
              <textarea name="概要" rows="3" class="form-control input-sm rfocusblue rmenuLeft 概要"></textarea>↓
            </td>↓
            <td class="width150">↓
              <textarea name="接続先システム" rows="3" class="form-control input-sm rfocusblue rmenuLeft 接続先システム"></textarea>↓
            </td>↓
            <td class="width150">↓
              <input name="処理タイミング" type="text" value="" class=" form-control input-sm rfocusblue rmenuLeft 処理タイミング">↓
            </td>↓
            <td class="width220">↓
              <textarea name="備考" rows="3" class="form-control input-sm rfocusblue rmenuLeft 備考"></textarea>↓
            </td>↓
            <td class="width210 rmenuLeft システム機能">↓
            </td>↓
            <td class="width50">↓
              <input name="紐づけ選択" type="radio" class="紐づけ選択 rfocusblue" value="1">↓
            </td>↓
          </tr>↓
        </tbody>↓
      </table>↓
    </div> <!-- テーブル スクロール エンド -->↓
    ↓
  </li> <!-- menu-li エンド -->↓
</ul> <!-- menu-ul エンド -->↓
<div class="rmenu-float-clear"> <!-- float 解除 -->↓
```

［テーブル 2 の Html］

```
<ul class="nav nav-pills rmenu-ul"> <!-- menu-ul  スタート -->↓
  <li class="rmenu-li"> <!-- menu-li  スタート -->↓

    <div id="scrollable-TableH2"> <!-- テーブル  スクロール  スタート -->↓
      <table id="mainTableH2" class="table table-condensed rmenuTable">↓
        <thead>↓
          <tr>↓
            <th class="width50">↓
              <div>紐づけ</div>↓
              <div>実行</div>↓
            </th>↓
            <th class="width110">システム機能ＩＤ</th>↓
            <th class="width250">システム機能名称</th>↓
          </tr>↓
        </thead>↓
      </table>↓
    </div> <!-- テーブル  スクロール  エンド -->↓

    <div id="scrollable-TableD2"> <!-- テーブル  スクロール  スタート -->↓
      <table id="mainTable2" class="table table-condensed rmenuTable">↓
        <tbody>↓
          <tr>↓
            <td class="width50">↓
              <input name="紐づけ実行" type="checkbox" class="紐づけ実行 rfocusblue" value="1">↓
            </td>↓
            <td class="width110 rmenuLeft システム機能ＩＤ">↓
            </td>↓
            <td class="width250 rmenuLeft システム機能名称">↓
            </td>↓
          </tr>↓
        </tbody>↓
      </table>↓
    </div> <!-- テーブル  スクロール  エンド -->↓
    ↓
  </li> <!-- menu-li  エンド -->↓
</ul> <!-- menu-ul  エンド -->↓
<div class="rmenu-float-clear"> <!-- float  解除 -->↓
</div>↓
```

点線内が紐付けを作成しています。

※コピー元と同じの為項目名の変更箇所だけ作成で OK です

CSS

※実践 1 と同じ

幅を調節します。

Appspec

紐付け機能の動きを確認します。

※コピー元と同じの為そのままで OK です。

```
74      //----------------------------------------------------------------↓
75      // 権限情報の設定（使用不可のロールを設定する）↓
76      //----------------------------------------------------------------↓
77      ,roleInfo: [↓
78          ["#戻る",              [""]]↓
79          ,["#実行",             [10, 50]]↓
80      ]↓
81      // バリデーションのイベントを定義する↓
82      ,validationEvent: [↓
83          ["#mainForm", "focus", "onFocus"]↓
84          ,["#mainForm", "blur", "onBlur"]↓
85      ]↓
86      // その他セレクターイベントを定義する↓
87      ,selectorEvent: [↓
88      // ここから追加処理↓
89          ["#検索サブシステム選択",  "change", "onChange検索サブシステム選択"]↓
90          ,["#検索管理単位選択",     "change", "onChange検索管理単位選択"]↓
91          ,[".削除",                 "change", "onChange削除"]↓
92          ,[".画面種類",             "change", "onChange画面種類"]↓
93          ,[".紐づけ選択",           "change", "onChange紐づけ選択"]↓
94          ,[".紐づけ実行",           "change", "onChange紐づけ実行"]↓
95      ]↓
```

onchange 紐づけ選択/実行が作成される。

これをコントローラで利用します。

コントローラ

DG_630U のコントローラを開きます。

ここには同じ紐づけの指示がありません。

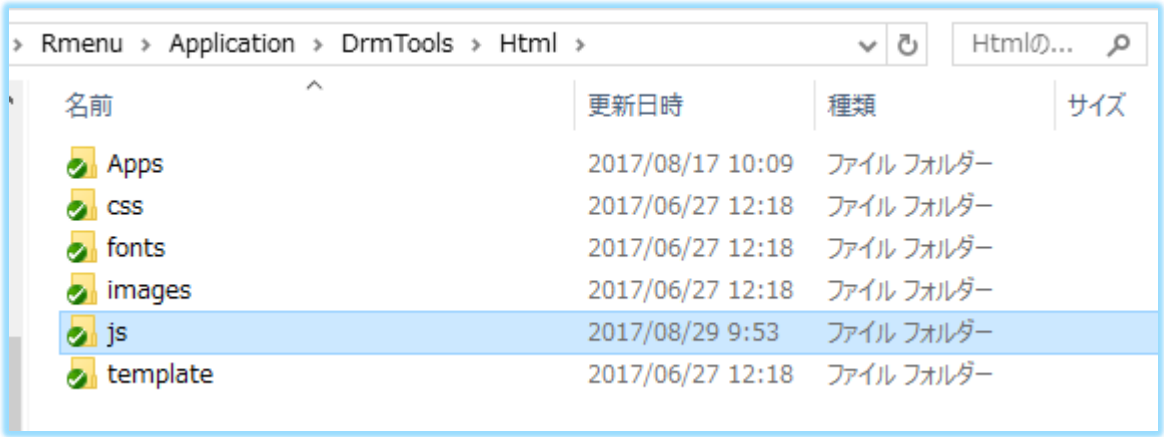
コピー元など複数の画面で同じ操作を行う為、共通の mixin に指示が書かれています。

ここで利用している mixin の確認方法は点線内の書かれています。

```
(function($, $R){
    //名前空間を設定する
    var App = $R.Application.DG_630U;

    //インスタンスプロパティを追加する
    var Controller = App.Controller = new $R.Class($R.Controller)
    Controller.fn.init = function(appspec, model, view) {
        $R.log("Controllerr init : start");

        this.appspec = appspec;
        this.model   = model;
        this.view    = view;
        this.createPubSubEvent();

        $R.log("Controllerr init : end");
    };

    //モジュールを追加する
    Controller.include($R.Library.EnterTabPFKeyMixin);
    Controller.include($R.Library.ControllerMixin);

    //マスターメンテパターンミックスインを追加する
    Controller.include($R.Library.MasterMainte2ControllerMixin);
    Controller.include($R.Library.PgwBrowserControllerMixin);
```

図のライブラリーを開きます。

mixin は j s フォルダーの中に保管されています。



共通の mixin を開くと［on　change 紐づけ選択/実行］があります。

※ここは触らないこと。変えてしまうとこの mixin を使う画面すべてに影響します。

```
↓
//----------------------------------------------------↓
//紐づけ選択□チェンジイベント処理（追加処理）↓
//----------------------------------------------------↓
,onChange紐づけ選択:function(event){↓
    $R.log("Controller onChange紐づけ選択 : start");↓

    //Win7・IE11の判定↓
    var status = this.isIE11OfWin7();↓
    if (status) {↓
        //var row = event.currentTarget.parentNode.parentNode.rowIndex + 1;↓
        var row = event.currentTarget.parentNode.parentNode.rowIndex;↓
    }↓
    else {↓
        var row = event.currentTarget.parentNode.parentNode.rowIndex;↓
    }↓

    var dataSet = this.model.onChange紐づけ選択(event);↓
    this.view.setチェックOf紐付け実行(event, row, dataSet);↓

    $R.log("Controller onChange紐づけ選択 : end");↓
}↓
↓
//----------------------------------------------------↓
//紐づけ実行□チェンジイベント処理（追加処理）↓
//----------------------------------------------------↓
,onChange紐づけ実行:function(event){↓
    $R.log("Controller onChange紐づけ実行 : start");↓

    //Win7・IE11の判定↓
    var status = this.isIE11OfWin7();↓
    if (status) {↓
        //var row = event.currentTarget.parentNode.parentNode.rowIndex + 1;↓
        var row = event.currentTarget.parentNode.parentNode.rowIndex;↓
    }↓
    else {↓
        var row = event.currentTarget.parentNode.parentNode.rowIndex;↓
    }↓

    var arg = this.model.on紐づけ実行チェック(row);↓
    if (!arg["status"]) {↓
        $R.log("Controller on紐づけ実行 : end");↓
        return;↓
    }↓
    ↓
    arg = this.model.on紐づけ実行判定(arg, row);↓
    if (arg["status"] == "紐づけ") {↓
        var dataSet = this.model.on紐づけ実行_紐づけ(arg);↓
    }↓
    else {↓
        var dataSet = this.model.on紐づけ実行_解除(arg);↓
    }↓

    this.view.on紐づけ実行(dataSet, arg);↓

    $R.log("Controller onChange紐づけ実行 : end");↓
}↓
```

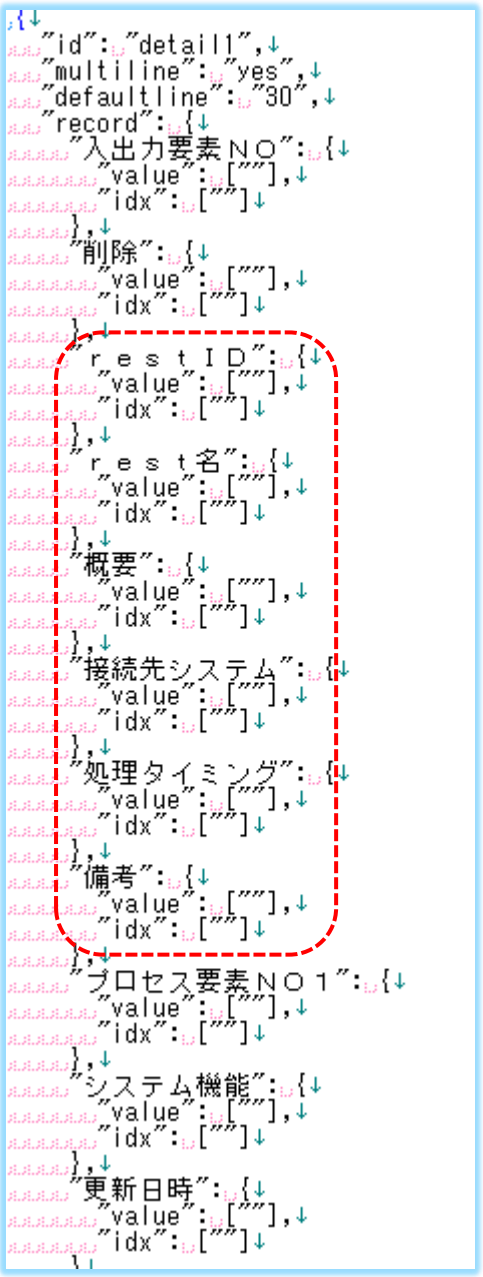
サーバー側を作成する

データセットＪＳＯＮ

※実践 2 と同様

コピー元と違う部分の項目名を作成します。

主にテーブル 1 の作成のため、［**id : detail1**］を Html と合わせて作成します。



バリデーションＪＳＯＮ

※実践 2 と同様

①データセットを確認し、項目名を作成します。

②ポスグレで文字数を確認して［**min**］と［**max**］を作成します。

```
{
"id": "detail1",
"multiline": "yes",
"defaultline": "30",
"record": {
  "入出力要素ＮＯ": {
    "validation": ["nonrequired",
    "integerP"],
    "min": "1",
    "max": "11"
  },
  "ｒｅｓｔＩＤ": {
    "validation": ["nonrequired",
    "free"],
    "min": "1",
    "max": "20"
  },
  "ｒｅｓｔ名": {
    "validation": ["nonrequired",
    "free"],
    "min": "1",
    "max": "60"
  },
  "概要": {
    "validation": ["nonrequired",
    "free"],
    "min": "1",
    "max": "900"
  },
  "接続先システム": {
    "validation": ["nonrequired",
    "free"],
    "min": "1",
    "max": "60"
  },
  "処理タイミング": {
    "validation": ["nonrequired",
    "free"],
    "min": "1",
    "max": "60"
  },
  "備考": {
    "validation": ["nonrequired",
    "free"],
    "min": "1",
    "max": "900"
  },
  "プロセス要素ＮＯ１": {
    "validation": ["nonrequired",
    "free"],
    "min": "1",
    "max": "11"
  }
```

トランＪＳＯＮ

※実践 2 と同様

同様にデータセットから項目名を作成する。

トランはセレクトとアップデート 2 つあります。

select（セレクト）SQL JSON

※実践 2 と同じ

①アウトプットをデータセットから作成

②AS をアウトプットから作成

③SQL を作成

※ER 図を確認するで確認した入出力要素区分の作成に注意する

④ジェネレート SQL の場合、［**field/table/funct**］が必要な場合は作成する

アップデート
updateSQL ＪＳＯＮ

アップデート SQL は実践 2 とほとんど同じですが、紐付け部分が違うので確認します。

下記図の紐付けが追加されています。

［updatesqlの構成を確認する］

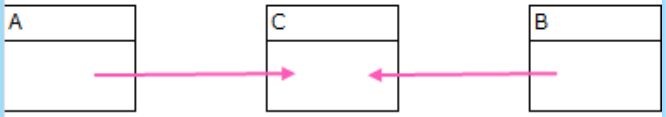
削除処理	id": "delete1	テーブルのデータが削除されたときにDBデータを削除する処理	インプットのみ
削除処理	id": "delete2	テーブルのデータが削除されたときに紐付け内容も一緒に削除する処理	
更新処理	id": "update	テーブルでデータが変更されたときにDBへデータを上書更新する処理	
削除処理	id": "delete_IO	紐付けテーブルで削除が押下された時の処理	アウトプットなし
登録処理	id": "insert_IO	紐付けテーブルで紐付けが押下された時の処理	書き込む目的の為
新規処理	id": "insert	テーブルで新規登録がされたときにDBへデータを書き込む処理	
照会処理	id": "detail1	実行された時にセレクトでテーブルのデータを**表示する処理**	アウトプットあり

※アウトプットなし　＝　インプットで DB へ入れるため

※アウトプットあり　＝　更新した DB からデータを出すため

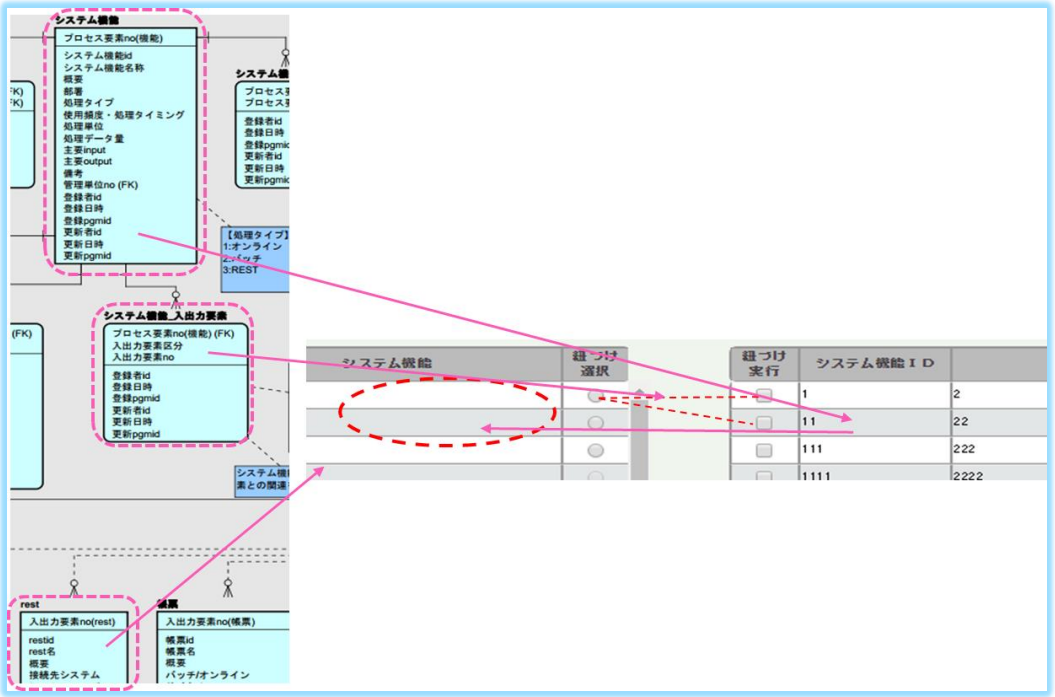
［紐付けの仕組み］

［紐付けの仕組み簡素図］



「Bがメインテーブル
Aのデータを紐付けたい為に
Cのテーブルでどれとどれを紐付けたのか記録する」

［画面と ER 図］



［照会処理を作成］

［updatesqlの構成を確認する］

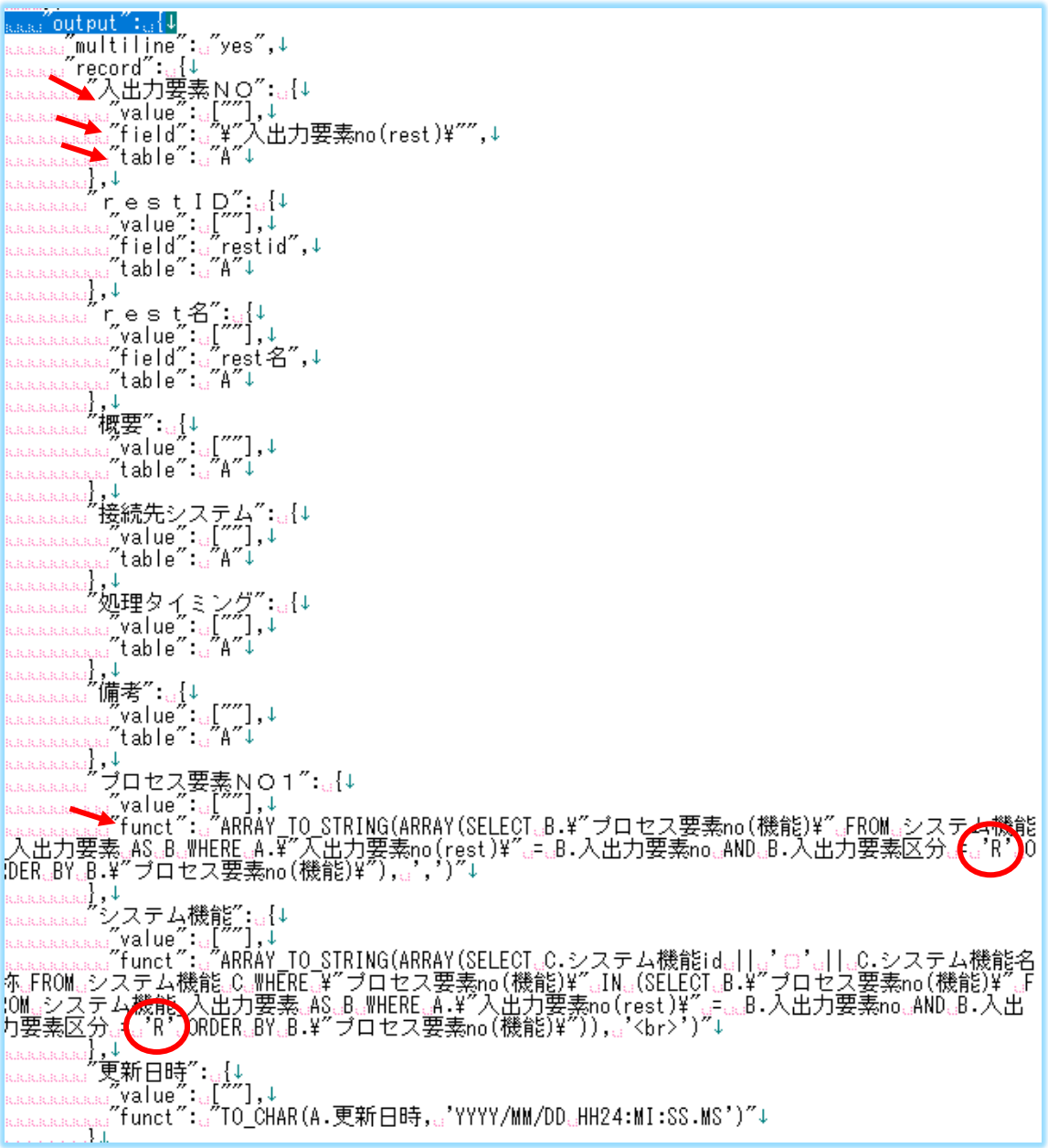
削除処理	id": "delete1	テーブルのデータが削除されたときにDBデータを削除する処理	インプットのみ アウトプットなし 書き込む目的の為
削除処理	id": "delete2	テーブルのデータが削除されたときに紐付け内容も一緒に削除する処理	
更新処理	id": "update	テーブルでデータが変更されたときにDBへデータを上書更新する処理	
削除処理	id": "delete_IO	紐付けテーブルで削除が押下された時の処理	
登録処理	id": "insert_IO	紐付けテーブルで紐付けが押下された時の処理	
新規処理	id": "insert	テーブルで新規登録がされたときにDBへデータを書き込む処理	
照会処理	id": "detail1	実行された時にセレクトでテーブルのデータを表示する処理	アウトプットあり

①アウトプットを作成する

（ジェネレート SQL 時は［**field/table/funct**］注意）

②［**Funct**］は freesql の為内容に注意

入出力要素区分がコピー元と違う為、ER 図を確認して［**R**］を作成する。



```
"output": {
    "multiline": "yes",
    "record": {
        "入出力要素ＮＯ": {
            "value": [""],
            "field": "¥"入出力要素no(rest)¥"",
            "table": "A"
        },
        "ｒｅｓｔＩＤ": {
            "value": [""],
            "field": "restid",
            "table": "A"
        },
        "ｒｅｓｔ名": {
            "value": [""],
            "field": "rest名",
            "table": "A"
        },
        "概要": {
            "value": [""],
            "table": "A"
        },
        "接続先システム": {
            "value": [""],
            "table": "A"
        },
        "処理タイミング": {
            "value": [""],
            "table": "A"
        },
        "備考": {
            "value": [""],
            "table": "A"
        },
        "プロセス要素ＮＯ１": {
            "value": [""],
            "funct": "ARRAY_TO_STRING(ARRAY(SELECT B.¥"プロセス要素no(機能)¥" FROM システム機能
入出力要素 AS B WHERE A.¥"入出力要素no(rest)¥" = B.入出力要素no AND B.入出力要素区分 = 'R' O
RDER BY B.¥"プロセス要素no(機能)¥"),',')"
        },
        "システム機能": {
            "value": [""],
            "funct": "ARRAY_TO_STRING(ARRAY(SELECT C.システム機能id || '□' || C.システム機能名
称 FROM システム機能 C WHERE ¥"プロセス要素no(機能)¥" IN (SELECT B.¥"プロセス要素no(機能)¥" F
ROM システム機能入出力要素 AS B WHERE A.¥"入出力要素no(rest)¥" = B.入出力要素no AND B.入出
力要素区分 = 'R' ORDER BY B.¥"プロセス要素no(機能)¥")),'<br>')"
        },
        "更新日時": {
            "value": [""],
            "funct": "TO_CHAR(A.更新日時,'YYYY/MM/DD HH24:MI:SS.MS')"
        }
```

［削除処理を作成］

［updatesqlの構成を確認する］

削除処理	id": "delete1	テーブルのデータが削除されたときにDBデータを削除する処理	インプットのみ アウトプットなし 書き込む目的の為
削除処理	id": "delete2	テーブルのデータが削除されたときに紐付け内容も一緒に削除する処理	
更新処理	id": "update	テーブルでデータが変更されたときにDBへデータを上書更新する処理	
削除処理	id": "delete_IO	紐付けテーブルで削除が押下された時の処理	
登録処理	id": "insert_IO	紐付けテーブルで紐付けが押下された時の処理	
新規処理	id": "insert	テーブルで新規登録がされたときにDBへデータを書き込む処理	
照会処理	id": "detail1	実行された時にセレクトでテーブルのデータを表示する処理	アウトプットあり

①

```
"sqls":[
  {
    "comment":"ｒｅｓｔ 削除処理",
    "id":"delete1",
    "before":"setDeleteDataOfDrmTools('detail1','delete1','入出力要素ＮＯ')",   →Ruby①へ
    "after":"",
    "sql":[
      {
        "type":"delete",   ←デリートします
        "freesql":"",
        "genesql":{
          "dist":"",
          "from":"rest",   ←メインテーブル
          "where":"¥"入出力要素no(rest)¥"=?",   ←条件
          "order":""
        }
      }
    ],
    "input":{
      "multiline":"yes",
      "record":{
        "入出力要素ＮＯ":{
          "value":[""],
          "field":"¥"入出力要素no(rest)¥""
        }
      }
    }
  },
  {
    "comment":"システム機能_入出力要素 削除処理",
    "id":"delete2",
    "before":"setDeleteDataOfDrmTools('detail1','delete2','入出力要素ＮＯ')",   →Ruby②へ
    "after":"",
    "sql":[
      {
        "type":"delete",   ←デリートします
        "freesql":"",
        "genesql":{
          "dist":"",
          "from":"システム機能_入出力要素",   ←紐付けテーブル
          "where":"入出力要素区分='R' AND 入出力要素no=?",
          "order":""
        }
      }
    ],   ↑のRは紐付けテーブルにrestの紐付けデータである事を「R」で区分して管理している
    "input":{
      "multiline":"yes",
      "record":{
        "入出力要素ＮＯ":{
          "value":[""],
          "field":"入出力要素no"
```

［更新処理を作成］

［updatesqlの構成を確認する］

削除処理	id": "delete1	テーブルのデータが削除されたときにDBデータを削除する処理	インプットのみ アウトプットなし 書き込む目的の為
削除処理	id": "delete2	テーブルのデータが削除されたときに紐付け内容も一緒に削除する処理	
更新処理	id": "update	テーブルでデータが変更されたときにDBへデータを上書更新する処理	
削除処理	id": "delete_IO	紐付けテーブルで削除が押下された時の処理	
登録処理	id": "insert_IO	紐付けテーブルで紐付けが押下された時の処理	
新規処理	id": "insert	テーブルで新規登録がされたときにDBへデータを書き込む処理	
照会処理	id": "detail1	実行された時にセレクトでテーブルのデータを表示する処理	アウトプットあり

```
  "comment": "ｒｅｓｔ 更新処理",
  "id": "update",
  "before": "setUpdateDataOfDrmTools('detail1', 'update', '入出力要素ＮＯ')",   →Ruby③へ
  "after": "",
  "sql": {
    "type": "update",   ←更新します
    "freesql": "",
    "genesql": {
      "dist": "",
      "from": "rest",
      "where": "¥"入出力要素no(rest)¥" = ?",   ←メインテーブル
      "order": ""
    }
  },
  "input": {
    "multiline": "yes",
    "record": {
      "ｒｅｓｔＩＤ": {
        "value": [""],
        "field": "restid"
      },
      "ｒｅｓｔ名": {
        "field": "rest名",
        "value": [""]
      },
      "概要": {
        "value": [""]
      },
      "接続先システム": {
        "value": [""]
      },
      "処理タイミング": {
        "value": [""]
      },
      "備考": {
        "value": [""]
      },
      "更新日時": {
        "value": [""],
        "funct": "CURRENT_TIMESTAMP"
      },
      "更新者id": {
        "value": [""],
        "fromtype": "request",
        "fromid": "login",
        "fromio": "",
        "fromname": "ユーザＩＤ"
      },
      "更新pgmid": {
        "value": ["DG_630U"]
      },
      "入出力要素ＮＯ": {
        "value": [""],
        "field": "¥"入出力要素no(rest)¥""
      }
    }
  },
```

←インプットへテーブルの項目名を作成する

［紐付けテーブルの処理を作成］

［updatesqlの構成を確認する］

削除処理	id": "delete1	テーブルのデータが削除されたときにDBデータを削除する処理	インプットのみ
削除処理	id": "delete2	テーブルのデータが削除されたときに紐付け内容も一緒に削除する処理	
更新処理	id": "update	テーブルでデータが変更されたときにDBへデータを上書更新する処理	
削除処理	id": "delete_IO	紐付けテーブルで削除が押下された時の処理	アウトプットなし 書き込む目的の為
登録処理	id": "insert_IO	紐付けテーブルで紐付けが押下された時の処理	
新規処理	id": "insert	テーブルで新規登録がされたときにDBへデータを書き込む処理	
照会処理	id": "detail1	実行された時にセレクトでテーブルのデータを表示する処理	アウトプットあり

※入出力要素区分を作成忘れないように注意する

①［紐付けテーブルの紐付けデータを削除する］

```
"comment": "システム機能_入出力要素 削除処理",
"id": "delete_IO",
"before": "setUpdateDataOfDrmTools('detail1','delete_IO','入出力要素ＮＯ')",  →Ruby④へ
"after": "",
"sql": {
  "type": "delete",  ←デリートします
  "freesql": "",
  "genesql": {
    "dist": "",
    "from": "システム機能_入出力要素",  ←紐付けテーブルから
    "where": "入出力要素区分 = 'R' AND 入出力要素no = ?",  ←条件を
    "order": ""
  }
},
"input": {
  "multiline": "yes",
  "record": {
    "入出力要素ＮＯ": {
      "value": [""],
      "field": "入出力要素no"
    }
```

②［紐付けテーブルへ紐付けデータを更新］

```
"comment": "システム機能_入出力要素　登録処理",↓
"id": "insert_IO",↓
"before": "setInsertシステム機能_入出力要素OfDG_630U('detail1', 'insert_IO')",↓
"after": "",↓
"sql": {↓
  "type": "insert",↓
  "freesql": "",↓
  "genesql": {↓
    "dist": "",↓
    "from": "システム機能_入出力要素",↓
    "where": "",↓
    "order": ""↓
  }↓
},↓
"input": {↓
  "multiline": "yes",↓
  "record": {↓
    "プロセス要素ＮＯ１": {↓
      "value": [""],↓
      "field": "¥"プロセス要素no(機能)¥""↓
    },↓
    "入出力要素区分": {↓
      "value": ["R"]↓
    },↓
    "入出力要素ＮＯ": {↓
      "value": [""],↓
      "field": "入出力要素no"↓
    },↓
    "登録日時": {↓
      "value": [""],↓
      "funct": "CURRENT_TIMESTAMP"↓
    },↓
    "登録者id": {↓
      "value": [""],↓
      "fromtype": "request",↓
      "fromid": "login",↓
      "fromio": "",↓
      "fromname": "ユーザＩＤ"↓
    },↓
    "登録pgmid": {↓
      "value": ["DG_630U"]↓
    },↓
    "更新日時": {↓
      "value": [""],↓
      "funct": "CURRENT_TIMESTAMP"↓
    },↓
    "更新者id": {↓
      "value": [""],↓
      "fromtype": "request",↓
      "fromid": "login",↓
      "fromio": "",↓
      "fromname": "ユーザＩＤ"↓
    },↓
    "更新pgmid": {↓
      "value": ["DG_630U"]↓
    }↓
```

→Ruby⑤へ

←インサート更新する

←紐付けテーブル

←インプットへテーブルの項目名を作成する
（入出力要素区分を作成する）

［新規処理を作成］

［updatesqlの構成を確認する］

削除処理	id": "delete1	テーブルのデータが削除されたときにDBデータを削除する処理	インプットのみ アウトプットなし 書き込む目的の為
削除処理	id": "delete2	テーブルのデータが削除されたときに紐付け内容も一緒に削除する処理	
更新処理	id": "update	テーブルでデータが変更されたときにDBへデータを上書更新する処理	
削除処理	id": "delete_IO	紐付けテーブルで削除が押下された時の処理	
登録処理	id": "insert_IO	紐付けテーブルで紐付けが押下された時の処理	
新規処理	id": "insert	テーブルで新規登録がされたときにDBへデータを書き込む処理	
照会処理	id": "detail1	実行された時にセレクトでテーブルのデータを表示する処理	アウトプットあり

```
"comment": "ｒｅｓｔ 新規処理",
"id": "insert",
"before": "setInsertDataOfDrmTools('detail1', 'insert', '入出力要素ＮＯ')",  →Ruby⑥へ
"after": "",
"sql": {
    "type": "insert",  ←インサート更新する
    "freesql": "",
    "genesql": {
        "dist": "",
        "from": "rest",  ←メインテーブル
        "where": "",
        "order": ""
    }
},
"input": {
    "multiline": "yes",
    "record": {
        "ｒｅｓｔＩＤ": {
            "value": [""],
            "field": "restid"
        },
        "ｒｅｓｔ名": {
            "value": [""],
            "field": "rest名"
        },
        "概要": {
            "value": [""]
        },
        "接続先システム": {
            "value": [""]
        },
        "処理タイミング": {
            "value": [""]
        },
        "備考": {
            "value": [""]
        },
        "検索管理単位ＮＯ": {
            "value": [""],
            "field": "管理単位no",
            "fromtype": "request",
            "fromid": "header",
            "fromio": "",
            "fromname": "検索管理単位ＮＯ"
        },
        "登録日時": {
            "value": [""],
            "funct": "CURRENT_TIMESTAMP"
        },
        "登録者id": {
            "value": [""],
            "fromtype": "request",
            "fromid": "login",
            "fromio": "",
            "fromname": "ユーザＩＤ"
        },
        "登録pgmid": {
            "value": ["DG_630U"]
        },
        "更新日時": {
            "value": [""],
            "funct": "CURRENT_TIMESTAMP"
        },
        "更新者id": {
            "value": [""],
            "fromtype": "request",
            "fromid": "login",
            "fromio": "",
            "fromname": "ユーザＩＤ"
        },
        "更新pgmid": {
            "value": ["DG_630U"]
        }
```

Ruby

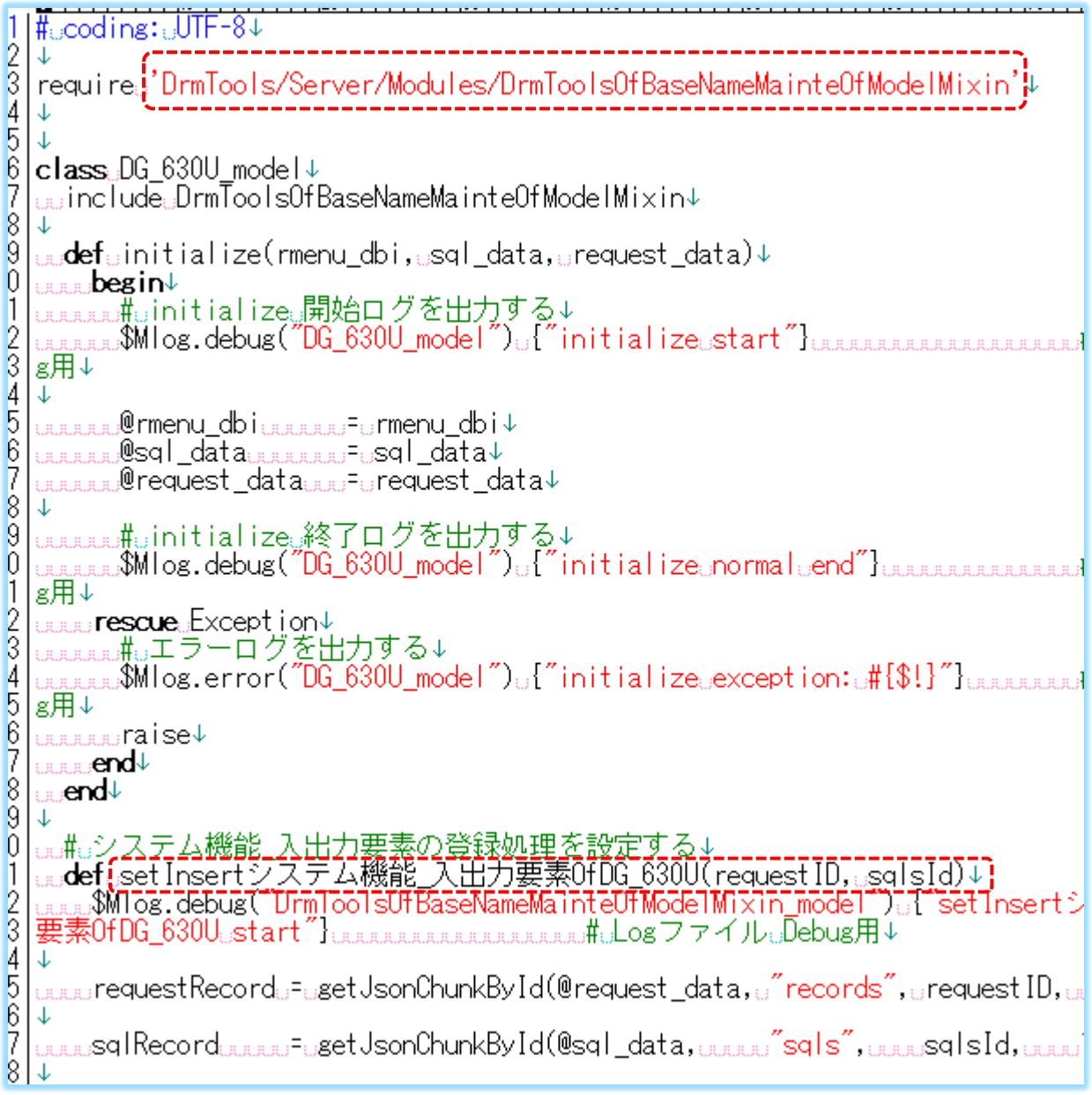
updateSQL の before に書かれていた Ruby を確認する。モデル ruby を開きます。

①Ruby の中のプログラム名がコピー元のままで変換されない為、

プログラム名［**DG_630U**］を作成します。

②Mixin を読み込んでいるため、内容を確認します。

［DG_630U_model.rb］



```
#␣coding:␣UTF-8↓
↓
require␣'DrmTools/Server/Modules/DrmToolsOfBaseNameMainteOfModelMixin'↓
↓
↓
class␣DG_630U_model↓
␣␣include␣DrmToolsOfBaseNameMainteOfModelMixin↓
↓
␣␣def␣initialize(rmenu_dbi,␣sql_data,␣request_data)↓
␣␣␣␣begin↓
␣␣␣␣␣␣#␣initialize␣開始ログを出力する↓
␣␣␣␣␣␣$Mlog.debug("DG_630U_model")␣{"initialize␣start"}
g用↓
↓
␣␣␣␣␣␣@rmenu_dbi␣␣␣␣␣=␣rmenu_dbi↓
␣␣␣␣␣␣@sql_data␣␣␣␣␣␣=␣sql_data↓
␣␣␣␣␣␣@request_data␣␣=␣request_data↓
↓
␣␣␣␣␣␣#␣initialize␣終了ログを出力する↓
␣␣␣␣␣␣$Mlog.debug("DG_630U_model")␣{"initialize␣normal␣end"}
g用↓
␣␣␣␣rescue␣Exception↓
␣␣␣␣␣␣#␣エラーログを出力する↓
␣␣␣␣␣␣$Mlog.error("DG_630U_model")␣{"initialize␣exception:␣#{$!}"}
g用↓
␣␣␣␣␣␣raise↓
␣␣␣␣end↓
␣␣end↓
↓
␣␣#␣システム機能_入出力要素の登録処理を設定する↓
␣␣def␣setInsertシステム機能_入出力要素OfDG_630U(requestID,␣sqlsId)↓
␣␣␣␣$Mlog.debug("DrmToolsOfBaseNameMainteOfModelMixin_model")␣{"setInsertシ
要素OfDG_630U␣start"}␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣#␣Logファイル␣Debug用↓
↓
␣␣␣␣requestRecord␣=␣getJsonChunkById(@request_data,␣"records",␣requestID,␣
↓
␣␣␣␣sqlRecord␣␣␣␣␣=␣getJsonChunkById(@sql_data,␣␣␣␣"sqls",␣␣␣␣sqlsId,␣␣␣␣
↓
```

［updateSQL の before 一覧］

下記を確認すると 4 種類の ruby が使われていることがわかる

①“comment": " ｒ ｅ ｓ ｔ　削除処理",

"id": "delete1",

"before": "**setDeleteDataOfDrmTools**('detail1', 'delete1', '入出力要素ＮＯ')",

②"comment": "システム機能_入出力要素　削除処理",

"id": "delete2",

"before": "**setDeleteDataOfDrmTools**('detail1', 'delete2', '入出力要素ＮＯ')",

③"comment": " ｒ ｅ ｓ ｔ　更新処理",

"id": "update",

"before": "**setUpdateDataOfDrmTools**('detail1', 'update', '入出力要素ＮＯ')",

④"comment": "システム機能_入出力要素　削除処理",

"id": "delete_IO",

"before": "**setUpdateDataOfDrmTools(**'detail1', 'delete_IO', '入出力要素ＮＯ')",

⑤"comment": "システム機能_入出力要素　登録処理",

"id": "insert_IO",

"before": "**setInsert システム機能_入出力要素 OfDG_630U**('detail1', 'insert_IO')",

⑥"comment": "システム機能_入出力要素　登録処理",

"id": "insert_IO",

"before": "**setInsert システム機能_入出力要素 OfDG_630U**('detail1', 'insert_IO')",

⑦ "comment": " ｒ ｅ ｓ ｔ　新規処理",

"id": "insert",

"before": "**setInsertDataOfDrmTools**('detail1', 'insert', '入出力要素ＮＯ')",

［model.rb を開く］

［def］に続き、指定のプログラムがある

下記は⑤⑥の sql で使われている ruby です。

プログラム名・項目名を作成します。

```
↓
  #　システム機能_入出力要素の登録処理を設定する↓
def　setInsertシステム機能_入出力要素OfDG_630U(requestID,　sqlsId)↓
    $Mlog.debug("DrmToolsOfBaseNameMainteOfModelMixin_model")　{"setInsertシステム機能_入出力要素OfDG_630U　start"}
#　Logファイル　Debug用↓
↓
    requestRecord　=　getJsonChunkById(@request_data,　"records",　requestID,　"record")↓
    sqlRecord　=　getJsonChunkById(@sql_data,　"sqls",　sqlsId,　"input",　"record")↓
↓
    maxLength　=　requestRecord["削除"]["value"].length　-　1↓
    k　=　0↓
    emptySW　=　"PASS"↓
    ↓
    for　i　in　0..maxLength　do↓
        #　削除データは処理対象外↓
        if　requestRecord["削除"]["value"][i]　==　"9"↓
            next↓
        end↓
        ↓
        #　新規データは処理対象外↓
        if　requestRecord["入出力要素ＮＯ"]["value"][i]　==　""↓
            next↓
        end↓
↓
        #　システム機能未設定データは処理対象外↓
        if　requestRecord["プロセス要素ＮＯ１"]["value"][i]　==　""↓
            next↓
        end↓
↓
        #　SQLのINPUT項目にリクエストデータを設定する↓
        w_プロセス要素ＮＯ配列　=　requestRecord["プロセス要素ＮＯ１"]["value"][i].split(",")↓
        maxSize　=　w_プロセス要素ＮＯ配列.length　-　1↓
        for　j　in　0..maxSize　do↓
            sqlRecord["プロセス要素ＮＯ１"]["value"][k]　=　w_プロセス要素ＮＯ配列[j]↓
            sqlRecord["入出力要素ＮＯ"]["value"][k]　=　requestRecord["入出力要素ＮＯ"]["value"][i]↓
            ↓
            k　=　k　+　1↓
            emptySW　=　"OK"↓
        end↓
    end↓
↓
    $Mlog.debug("DrmToolsOfBaseNameMainteOfModelMixin_model")　{"setInsertシステム機能_入出力要素OfDG_630U　end　status　=　#{emptySW}"}
↓
    return　emptySW↓
end↓
```

［共通の mixin.rb を開く］

※共通の mixin は触らないので確認のみ。

削除データを設定

下記は①②の sql で使われている ruby です。

```
  #_基本名称メンテナンス用□モジュール↓
  #_削除データを設定する↓
  def_setDeleteDataOfDrmTools(requestID,_sqlsId,_idName)↓
    $Mlog.debug("DrmToolsOfBaseNameMainteOfModelMixin_model")_{"setDeleteDataOfDrmTools_start"}
#_Logファイル_Debug用↓
↓
    requestInfo_____=_getJsonChunkById(@request_data,_"records",_requestID)↓
    requestRecord_=_requestInfo["record"]↓
↓
    sqlInfo_____=_getJsonChunkById(@sql_data,_"sqls",_sqlsId)↓
    sqlRecord_____=_sqlInfo["input"]["record"]↓
↓
    deleteCheck____=_requestRecord["削除"]["value"]↓
    idData_____=_requestRecord[idName]["value"]↓
    maxLength_____=_deleteCheck.length_-_1↓
    j_____=_0↓
    emptySW_____=_"PASS"↓
    ↓
    ↓
    for_i_in_0..maxLength_do↓
      if_deleteCheck[i]_!=_"9"↓
        next↓
      end↓
      if_idData[i]_==_""↓
        next↓
      end↓
↓
      requestRecord.each{|key,_value|↓
        if_sqlRecord.key?("#{key}")↓
          sqlRecord[key]["value"][j]_=_value["value"][i]↓
          emptySW_=_"OK"↓
        end↓
      }↓
      j_=_j_+_1↓
    end↓
↓
    $Mlog.debug("DrmToolsOfBaseNameMainteOfModelMixin_model")_{"setDeleteDataOfDrmTools_end_status_=_#{emptySW}"}
↓
    return_emptySW↓
  end↓
```

更新データを設定

下記は③④の sql で使われている ruby です。

```
# 基本名称メンテナンス用□モジュール
# 更新データを設定する
def setUpdateDataOfDrmTools(requestID, sqlsId, idName)
    $Mlog.debug("DrmToolsOfBaseNameMainteOfModelMixin_model"){"setUpdateDataOfDrmTools start"}
Logファイル Debug用

    requestInfo    = getJsonChunkById(@request_data, "records", requestID)
    requestRecord = requestInfo["record"]

    sqlInfo        = getJsonChunkById(@sql_data, "sqls", sqlsId)
    sqlRecord      = sqlInfo["input"]["record"]

    deleteCheck    = requestRecord["削除"]["value"]
    idData         = requestRecord[idName]["value"]
    maxLength      = deleteCheck.length - 1
    j              = 0
    emptySW        = "PASS"
    for i in 0..maxLength do
      if deleteCheck[i] == "9"
        next
      end
      if idData[i] == ""
        next
      end

      requestRecord.each { |key, value|
        if sqlRecord.key?("#{key}")
          sqlRecord[key]["value"][j] = value["value"][i]
          emptySW = "OK"
        end
      }
      j = j + 1
    end

    $Mlog.debug("DrmToolsOfBaseNameMainteOfModelMixin_model"){"setUpdateDataOfDrmTools end status = #{emptySW}"}

    return emptySW
end
```

登録データを設定

下記は⑦の sql で使われている ruby です。

```
# 基本名称メンテナンス用□モジュール↓
# 登録データを設定する↓
def setInsertDataOfDrmTools(requestID, sqlsId, idName)↓
    $Mlog.debug("DrmToolsOfBaseNameMainteOfModelMixin_model"){"setInsertDataOfDrmTools start"}
Logファイル Debug用↓

    requestInfo   = getJsonChunkById(@request_data, "records", requestID)↓
    requestRecord = requestInfo["record"]↓

    sqlInfo       = getJsonChunkById(@sql_data, "sqls", sqlsId)↓
    sqlRecord     = sqlInfo["input"]["record"]↓

    deleteCheck   = requestRecord["削除"]["value"]↓
    idData        = requestRecord[idName]["value"]↓
    maxLength     = deleteCheck.length - 1↓
    j             = 0↓
    emptySW       = "PASS"↓
    for i in 0..maxLength do↓
        if deleteCheck[i] == "9"↓
            next↓
        end↓
        if idData[i] != ""↓
            next↓
        end↓

        requestRecord.each { |key, value|↓
            if sqlRecord.key?("#{key}")↓
                sqlRecord[key]["value"][j] = value["value"][i]↓
                emptySW = "OK"↓
            end↓
        }↓
        j = j + 1↓
    end↓

    $Mlog.debug("DrmToolsOfBaseNameMainteOfModelMixin_model"){"setInsertDataOfDrmTools end status = #{emptySW}"}

    return emptySW↓
end↓
```