RMENU

Rmenuサイト

実践5『エクセル印刷機能』作成

実践5『エクセル印刷機能』作成
事前確認

手順　画面確認　プログラムのコピー

◆ 事前確認

作成する画面タイトル：『REST（一覧）』の印刷ボタン

作成するプログラム名：DG_620R　へJSON4つ

コピー元プログラム名：DG_600R　からJSON4つ

習得内容：作成したプログラムに印刷機能を追加する

手順

事前確認

- 手順
- 画面の確認
- プログラムのコピー

エクセルの用意

- テンプレートの作成
- 出力先フォルダー作成

クライアント側の作成

- Html
- Appspec
- コントローラ
- モデル

サーバー側の作成

- データセットJSON
- トランJSON
  - •印刷一括
  - •印刷個別
  - •afterの作成
- SQLJSON
  - •両SQL共通の作成
  - •印刷一括
  - •印刷個別

Rubyの作成

- モデル作成
  - •コピー元からコピペ
  - •印刷一括のbefore作成
  - •印刷個別のbefore作成
- ビュー作成
  - •コピー元からコピペ
  - •テンプレートの設定
  - •エクセルを作成する

画面の確認

画面タイトル：『**REST一覧**』
プログラム名：［**DG_620R**］

作成箇所：ヘッダー上部に赤線の「**印刷ボタン**」を作成します。

説明：この画面は2つの印刷ボタンがあり、2種類のエクセルを印刷できます

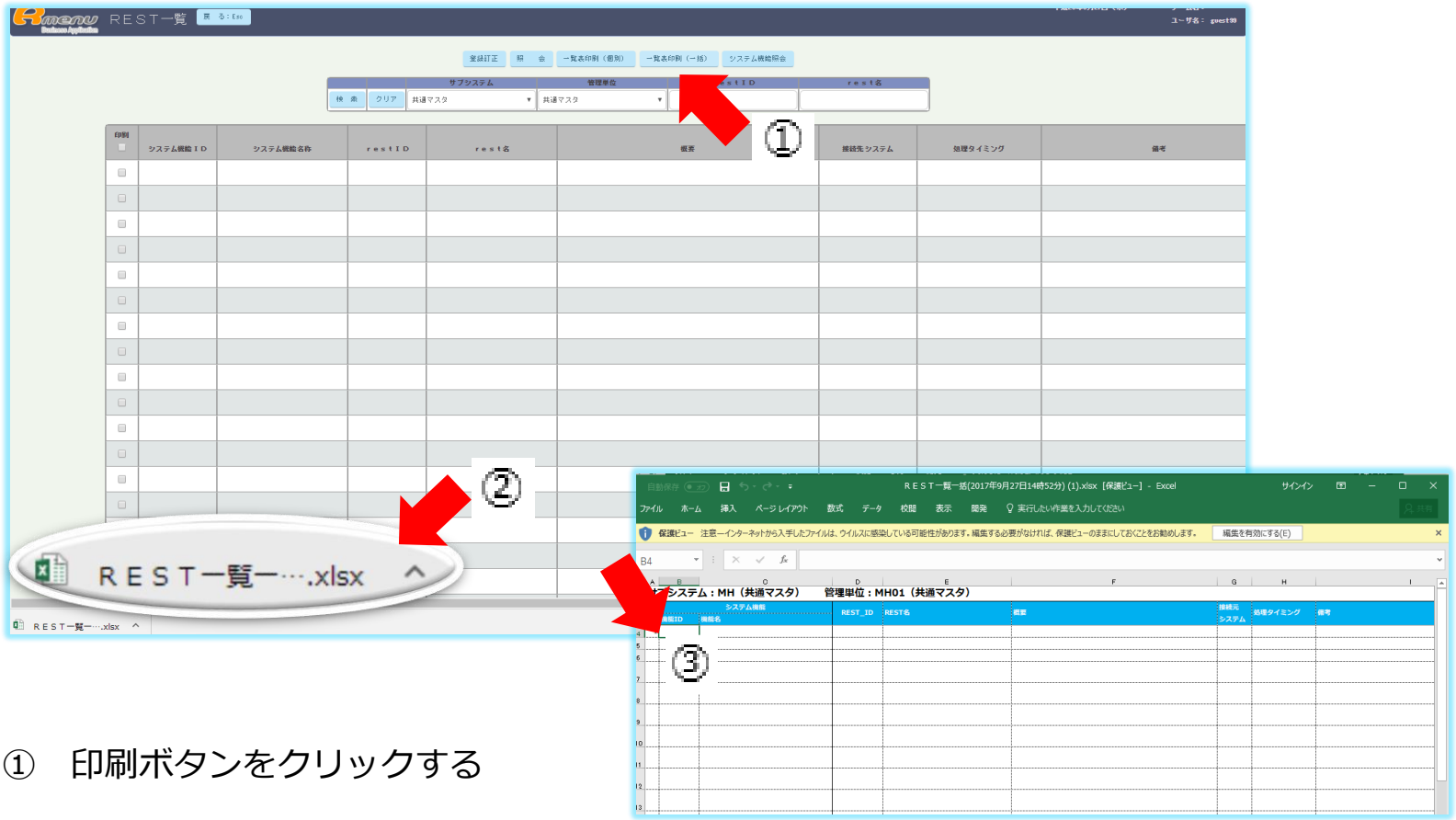
- ［**一覧表印刷（個別）ボタン**］
  印刷したい行にチェックを入れて選択したデータのみを印刷

- ［**一覧表印刷（一括）ボタン**］
  一覧画面に表示されている全てのデータを印刷

※2つのボタン＝2つのデータ作成方法があるため、プログラムを2つ作成します
・**個別**印刷時は［**printchecksql/printchecktran**］が動く
・**一括**印刷時は［**printallsql/printalltran**］が動く

【詳細説明】
個別印刷の場合は、
出力したいデータにチェックを入れる。
チェックボタンにチェックが1つも入っていない場合は、
エラーが表示される。

画面確認（ボタン操作時）



① 印刷ボタンをクリックする

② 左下にエクセルが出力される

③ エクセルを開くと表示されているテーブルのデータが印刷できます

プログラムのコピー

印刷のプログラムを雛型からコピーし作成します

①コピー元プログラム名：［DG_600R］の［JSON］を開き、4つのJSONをコピーします
　［**printallsql**］［**printalltran**］［**printchecksql**］［**printchecktran** ］

②作成先プログラム名：［DG_620R］のJSONを開き、貼り付けます
　図の様にプログラム名を手打ちで書き換えます［**DG_600R**］→［**DG_620R**］へ

③プログラム内の**プログラム名**を一括で変換する
　操作方法：秀丸を利用する
　［**新規ページ**］を開く→ツールバー［**検索**］クリック→［**grepの実行**］クリック→
　　条件を入力→［**置換を実施**］クリック→ ［DG_600R］を［**DG_620R**］へ置き換え→完了

実践5『エクセル印刷機能』作成
エクセルの用意

エクセル確認 → テンプレート作成 → 出力先フォルダー作成

- 作成したいエクセルを準備します

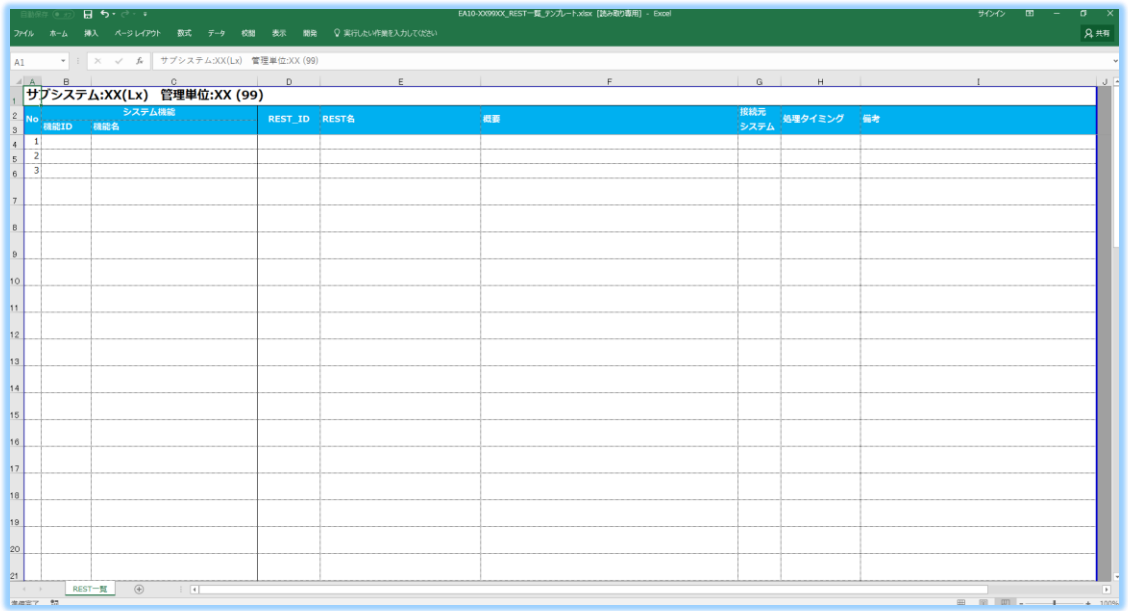
①エクセルをもらいます

②Rmenu へエクセルのテンプレートを作成します

③生成されたエクセルの出力先を作成します

エクセルの確認

REST一覧_テンプレート.xlsx

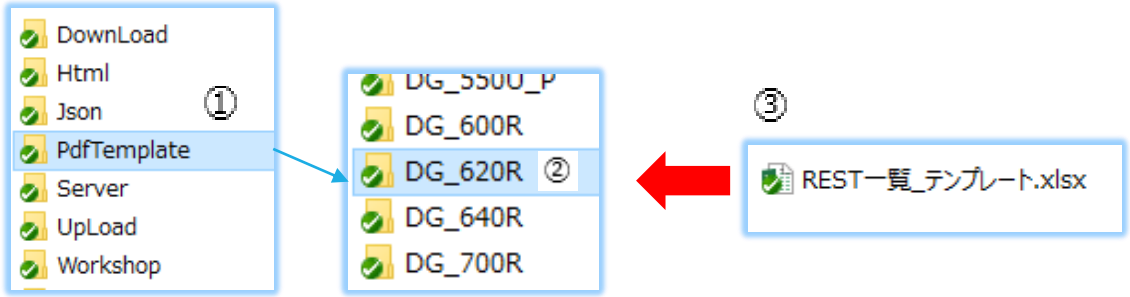


エクセルを用意し、内容を確認します

出力したいエクセルワークシートは**事前に作成したものを用意**し、
このワークシートに画面のデータが入り、印刷ができるように作成します。

※エクセルの拡張子は［.xlxs］である事と
　隠れているセルがないか確認します。（エラーを防ぐため）

テンプレート作成



プログラムでデータを作成した時の出力先エクセルワークシートを
先に指定のフォルダーへ準備します。

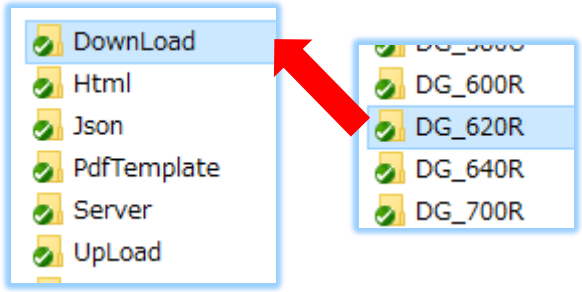
①Html・JSONの下に［**PdfTemplate**］があるので開く

②プログラム名のファルダーを新規作成します［**DG_620R**］

③作成したフォルダーへエクセルを貼り付け完了

注意
※元のエクセルシートの**拡張子**を確認すること（マクロが付いていると使用不可）
※元のエクセルシートの**全ての列**を確認すること（隠れている列があるとデータの出力先がズレます）

出力先フォルダー作成



プログラムが生成したエクセルワークシートの出力先を作成する。
[Download]へエクセルの出力先になる[空のフォルダー]を作成します。

①Htmlの上に［ Download ］があるので開く

②プログラム名のファルダーを新規作成します　［**DG_620R**］

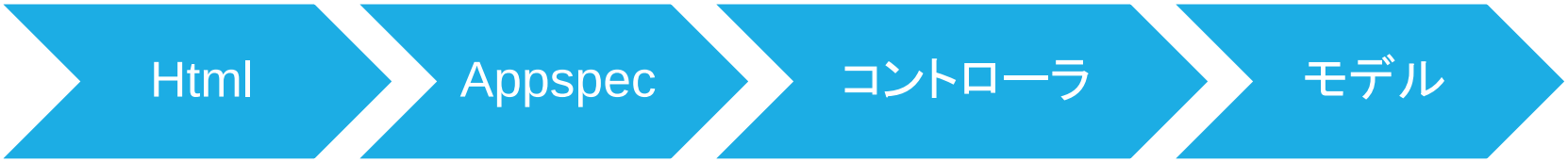
エクセルの用意ができたのでプログラムを作成していきます。

【詳細説明】

印刷ボタンを押すと、
この中にファイルが生成されて、
作ったエクセルを、この中に書き込む。
書き込んだエクセルファイルを
サーバーに
ここにあると情報を渡す。

ダウンロードしたら不要なため
自動的に消えるようになっている。

実践5『エクセル印刷機能』作成
クライアント側の作成



- クライアント側から作成します

Html：印刷ボタンを作成
Appspec：イベント設定
コントローラ：動きの設定、モデルへ指示を出す
モデル：モデルで処理をしコントローラへデータを返す
（モデルは後日Mixinへプログラムを移しました。）

Html

```
<div class="row">↓
  <div class="col-sm-12 text-center"> <!-- センタリング　スタート -->↓
↓
              <div class="btn-group">↓
                <button id="登録訂正"     class="btn color4 btn-sm" type="button" name="登録" >登録訂正</button>↓
              </div>↓
              <div class="btn-group">↓
                <button id="照会"     class="btn color4 btn-sm" type="button" name="照会" >照　　会</button>↓
              </div>↓
              <div class="btn-group">↓
                <button id="一覧表印刷個別"     class="btn color4 btn-sm" type="button" name="一覧表印刷個別" >一覧表印刷（個別）</button>↓
              </div>↓
              <div class="btn-group">↓
                <button id="一覧表印刷一括"     class="btn color4 btn-sm" type="button" name="一覧表印刷一括" >一覧表印刷（一括）</button>↓
              </div>↓
              <div class="btn-group">↓
                <button id="システム機能照会"     class="btn color4 btn-sm" type="button" name="システム機能照会" >システム機能照会</button>↓
              </div>↓
```

Htmlで印刷ボタンを作成します
ボタンの作成は実践2で作成済の為、項目名を確認します。

①［id］［name］［画面に表示されるテキスト］の3か所を確認します

※黄色の部分を統一させる

Appspec

コピー元Appspec

```
//----------------------------------------------------------------↓
//トランザクション・リクエストチェック・レスポンス編集・エラーのコールバック関数を定義する↓
//----------------------------------------------------------------↓
,requestInfo: [↓
  ["initfirst",        "on初期処理最初OfCheckRequestData",        "on初期処理最初OfEditResponseData",        "onErrorResponseData"]↓
, ["initreload",       "on初期処理リロードOfCheckRequestData",    "on初期処理リロードOfEditResponseData",    "onErrorResponseData"]↓
, ["control",          "on管理単位処理OfCheckRequestData",        "on管理単位処理OfEditResponseData",        "onErrorResponseData"]↓
, ["selectmaintable",  "onメインテーブル照会OfCheckRequestData",  "onメインテーブル照会OfEditResponseData",  "onErrorResponseData"]↓
, ["printcheck",       "on一覧表印刷個別OfCheckRequestData",      "on一覧表印刷個別OfEditResponseData",      "onErrorResponseData"]↓
, ["printall",         "on一覧表印刷一括OfCheckRequestData",      "on一覧表印刷一括OfEditResponseData",      "onErrorResponseData"]↓
]↓
// ----------------------------------------------------------------↓
```

```
//----------------------------------------------------------------↓
//イベント設定はセレクタ・イベント・コールバック関数の順に指定する↓
//----------------------------------------------------------------↓
//NAVボタンのイベントを定義する↓
navButtonEvent: [↓
  ["#戻る",          "click", "on戻る"]↓
, ["#登録訂正",      "click", "on登録訂正"]↓
, ["#照会",          "click", "on照会"]↓
, ["#最初のページ",  "click", "on最初のページ"]↓
, ["#前のページ",    "click", "on前のページ"]↓
, ["#次のページ",    "click", "on次のページ"]↓
, ["#最後のページ",  "click", "on最後のページ"]↓
, ["#mainTable td",  "click", "onテーブル行クリック"]↓
//ここから追加処理↓
, ["#検索",          "click", "on検索"]↓
, ["#クリア",        "click", "onクリア"]↓
, ["#一覧表印刷個別", "click", "on一覧表印刷個別"]↓
, ["#一覧表印刷一括", "click", "on一覧表印刷一括"]↓
, ["#システム機能照会", "click", "onシステム機能照会"]↓
]↓
```

```
//----------------------------------------------------------------↓
//権限情報の設定（使用不可のロールを設定する）↓
//----------------------------------------------------------------↓
roleInfo: [↓
  ["#戻る",          ""]]↓
, ["#登録訂正",      [10, 50]]↓
, ["#照会",          ""]]↓
, ["#最初のページ",  ""]]↓
, ["#前のページ",    ""]]↓
, ["#次のページ",    ""]]↓
, ["#最後のページ",  ""]]↓
//ここから追加処理↓
, ["#検索",          ""]]↓
, ["#クリア",        ""]]↓
, ["#一覧表印刷個別", ""]]↓
, ["#一覧表印刷一括", ""]]↓
, ["#システム機能照会", ""]]↓
]↓
//バリデーションのイベントを定義する↓
validationEvent: [↓
  ["#mainForm", "focus", "onFocus"]↓
, ["#mainForm", "blur",  "onBlur"]↓
]↓
//その他セレクターイベントを定義する↓
selectorEvent: [↓
//ここから追加処理↓
  ["#検索サブシステム選択", "change", "onChange検索サブシステム選択"]↓
, ["#検索管理単位選択",     "change", "onChange検索管理単位選択"]↓
, ["#H印刷",               "change", "onChangeH印刷"]↓
]↓
```

[on印刷]を作成します

①コピー元プログラムのAppspecを開き、**印刷ボタンのイベント設定をコピペで持ってきます。**

②イベント処理、トランザクション関数定義などボタン2つ分の2行づつ作成する（黄色の行）

※注意「カンマ」や「閉じカッコ」に気を付けること

Appspec 【詳細説明】

ボタンイベントの設定

```
67      //------------------------------------------------------------------↓
68      //イベント設定はセレクタ・イベント・コールバック関数の順に指定する↓
69      //------------------------------------------------------------------↓
70      //NAVボタンのイベントを定義する↓
71      ,navButtonEvent:[↓
72          ["#戻る",          "click", "on戻る"]↓
73      ,   ["#登録訂正",      "click", "on登録訂正"]↓
74      ,   ["#照会",          "click", "on照会"]↓
75      ,   ["#最初のページ",  "click", "on最初のページ"]↓
76      ,   ["#前のページ",    "click", "on前のページ"]↓
77      ,   ["#次のページ",    "click", "on次のページ"]↓
78      ,   ["#最後のページ",  "click", "on最後のページ"]↓
79      ,   ["#mainTable td",  "click", "onテーブル行クリック"]↓
80      //ここから追加処理↓
81      ,   ["#検索",          "click", "on検索"]↓
82          ["#クリア"         "click", "onクリア"]↓
83      ,   ["#一覧表印刷個別",  "click", "on一覧表印刷個別"]↓
84      ,   ["#一覧表印刷一括",  "click", "on一覧表印刷一括"]↓
85      ,   ["#システム機能照会",  "click", "onシステム機能照会"]↓
86      ,   ["#画面照会",      "click", "on画面照会"]↓
87      ,   ["#バッチ照会",    "click", "onバッチ照会"]↓
88      ]↓
```

Index.htmlに記述されているidは、
左記：appspecのボタンイベント設定の項目と同じで
なければいけない。

画面上でボタンが押されると、この部分へ指示が出される。

Appspec 【詳細説明】

トランザクションの定義

```
35      //------------------------------------------------------------------------------↓
36      //トランザクション・リクエストチェック・レスポンス編集・エラーのコールバック関数を定義する↓
37      //------------------------------------------------------------------------------↓
38      ,requestInfo:[↓
39          ["initfirst","on初期処理最初OfCheckRequestData","on初期処理最初OfEditResponseData","onErrorResponseData"]↓
40          ,["initreload","on初期処理リロードOfCheckRequestData","on初期処理リロードOfEditResponseData","onErrorResponseData"]↓
41          ,["control","on管理単位処理OfCheckRequestData","on管理単位処理OfEditResponseData","onErrorResponseData"]↓
42          ,["selectmaintable","onメインテーブル照会OfCheckRequestData","onメインテーブル照会OfEditResponseData","onErrorResponseData"]↓
43          ,["printcheck","on一覧表印刷個別OfCheckRequestData","on一覧表印刷個別OfEditResponseData","onErrorResponseData"]↓
44          ,["printall","on一覧表印刷一括OfCheckRequestData","on一覧表印刷一括OfEditResponseData","onErrorResponseData"]↓
45      ]↓
```

印刷に関するトランザクションを追加する。
トランザクションとは、サーバーとのやり取りのことであり、トランザクションに記述された名前と
同じファイルが必ずjsonファイルに存在する。

Jsonファイルに、上記で追加したトランザクションの内容を追加する。
コピー元からJSONをコピーし、ファイル名をプログラム名変更する。

最初に行ったプログラムのコピー作業を指します

コントローラ

コントローラで3つの処理を作成します

①イベント処理
（後日、共通プログラムへ移した為**作成不要**になりました）
「**ボタンクリック時の動き**」を作成しています

②リクエストデータ処理
（後日、共通プログラムへ移した為**作成不要**になりました）
「モデルへ指示を出す動き」を作成しています

③レスポンスデータ処理
「モデルから帰ってきたデータを受け取る動き」を作成しています

コントローラ

①イベント処理（後日共通プログラムへ移した為**作成不要**になりました）

```
// ---------------------------------------
// ボタン・ファンクションキー イベント処理
// ---------------------------------------
on一覧表印刷個別: function(event) {
  $R.log("Controller on一覧表印刷個別 : start");
  this.ajaxExecute("printcheck");              →プリントチェックのプログラムを実行する指示
  $R.log("Controller on一覧表印刷個別 : end");
}
,on一覧表印刷一括: function(event) {
  $R.log("Controller on一覧表印刷一括 : start");
  this.ajaxExecute("printall");                →プリントオールのプログラムを実行する指示
  $R.log("Controller on一覧表印刷一括 : end");
}
```

ボタンクリック時の動きを作成します。
コピー元からイベント処理をかたまりごとコピペし［**on一覧印刷個別**］と［**on一覧印刷一括**］を作成する。

※赤文字の指示は（）の中のプログラムを実行する指示です

［**個別印刷**］ボタンがクリックされると［**printchecksql/printchecktran**］が動き
［**一括印刷**］ボタンがクリックされると［**printallsql/printalltran**］が動くように
コントローラのイベント処理で作成されています。

コントローラ

②リクエストデータ処理（後日共通プログラムへ移した為**作成不要**になりました）

```
// ------------------------------------
// リクエストデータ　編集・チェック処理
// ------------------------------------
,on一覧表印刷個別OfCheckRequestData: function(requestData, mode) {
  $R.log("Controller on一覧表印刷個別OfCheckRequestData : start");
  var status = this.model.on一覧表印刷個別OfCheckRequestData(requestData, mode);
  if (status) {
    status = this.checkRequestData(requestData);
  }
  $R.log("Controller on一覧表印刷個別OfCheckRequestData : end");
  return status;
}
,on一覧表印刷一括OfCheckRequestData: function(requestData, mode) {
  $R.log("Controller on一覧表印刷一括OfCheckRequestData : start");
  var status = this.model.on一覧表印刷一括OfCheckRequestData(requestData, mode);
  if (status) {
    status = this.checkRequestData(requestData);
  }
  $R.log("Controller on一覧表印刷一括OfCheckRequestData : end");
  return status;
}
```

コピー元からイベント処理をかたまりごとコピペし、［**on一覧印刷個別**］と［**on一覧印刷一括**］を作成する。

コントローラ

③レスポンスデータ処理

（ﾚｽﾎﾟﾝｽは指示にプログラム名を含み、共通プログラムでは処理できない為、各プログラムで作成する。）

```
//------------------------------------
//　レスポンスデータ□編集処理
//------------------------------------
,on一覧表印刷個別OfEditResponseData: function(responseData, mode) {
    $R.log("Controller on一覧表印刷個別OfEditResponseData : start");

    //this.editResponseData(responseData);
    this.model.on一覧表印刷OfEditResponseData(responseData, mode, "ＲＥＳＴ一覧個別");

    $R.log("Controller on一覧表印刷個別OfEditResponseData : end");
}

,on一覧表印刷一括OfEditResponseData: function(responseData, mode) {
    $R.log("Controller on一覧表印刷一括OfEditResponseData : start");

    //this.editResponseData(responseData);
    this.model.on一覧表印刷OfEditResponseData(responseData, mode, "ＲＥＳＴ一覧一括");

    $R.log("Controller on一覧表印刷一括OfEditResponseData : end");
}
```

1.コピー元からイベント処理をかたまりごとコピペし、［**on一覧印刷個別**］と［**on一覧印刷一括**］を作成する

2.赤枠のプログラム名部分を作成する［**REST**］

コントローラ【詳細説明】

ボタンイベント処理

印刷機能というのは、複数のプログラムで使用している為、共通プログラムに記述している。

mastermaintelist2.controller.mixin-2.1.1.js

```
133 ,on一覧表印刷個別: function(event) {↓
134 $R.log("Controller on一覧表印刷個別 : start");↓
135 ↓
136 this.ajaxExecute("printcheck");↓
137 ↓
138 $R.log("Controller on一覧表印刷個別 : end");↓
139 }↓
140 ↓
141 ,on一覧表印刷一括: function(event) {↓
142 $R.log("Controller on一覧表印刷一括 : start");↓
143 ↓
144 this.ajaxExecute("printall");↓
145 ↓
146 $R.log("Controller on一覧表印刷一括 : end");↓
147 }↓
148 ↓
```

this.ajaxExcute(“～”)とは、
このトランザクションを実行しなさい。
という意味である。

画面でボタンを押されると、
appspecのボタンイベント定義に記述された内容を元に、
左記のイベントを発生させる。

コントローラ【詳細説明】

リクエストデータ　編集　チェック処理

```
282     ,on一覧表印刷個別OfCheckRequestData: function(requestData, mode) {↓
283         $R.log("Controller on一覧表印刷個別OfCheckRequestData : start");↓
284 ↓
285         var status = this.model.on一覧表印刷個別OfCheckRequestData(requestData, mode);↓
286         if (status) {↓
287             status = this.checkRequestData(requestData);↓
288         }↓
289 ↓
290         $R.log("Controller on一覧表印刷個別OfCheckRequestData : end");↓
291         return status;↓
292     }↓
293 ↓
294     ,on一覧表印刷一括OfCheckRequestData: function(requestData, mode) {↓
295         $R.log("Controller on一覧表印刷一括OfCheckRequestData : start");↓
296 ↓
297         var status = this.model.on一覧表印刷一括OfCheckRequestData(requestData, mode);↓
298         if (status) {↓
299             status = this.checkRequestData(requestData);↓
300         }↓
301 ↓
302         $R.log("Controller on一覧表印刷一括OfCheckRequestData : end");↓
303         return status;↓
304     }↓
```

トランザクションを実行する前に行うチェック内容を記述する。

この場合は、Modelに同じ関数名で記述された処理を呼び出している。

コントローラ【詳細説明】

レスポンスデータ　編集処理　※パターンではなくDG_640Rに記述

```
70  //----------------------------------------↓
71  //　レスポンスデータ□編集処理↓
72  //----------------------------------------↓
73  ,on一覧表印刷個別OfEditResponseData: function(responseData, mode) {↓
74  $R.log("Controller on一覧表印刷個別OfEditResponseData : start");↓
75  ↓
76  //this.editResponseData(responseData);↓
77  this.model.on一覧表印刷OfEditResponseData(responseData, mode, "帳票一覧個別");↓
78  ↓
79  $R.log("Controller on一覧表印刷個別OfEditResponseData : end");↓
80  }↓
81  ↓
82  ,on一覧表印刷一括OfEditResponseData: function(responseData, mode) {↓
83  $R.log("Controller on一覧表印刷一括OfEditResponseData : start");↓
84  ↓
85  //this.editResponseData(responseData);↓
86  this.model.on一覧表印刷OfEditResponseData(responseData, mode, "帳票一覧一括");↓
87  ↓
88  $R.log("Controller on一覧表印刷一括OfEditResponseData : end");↓
89  }↓
90  ↓
91  });↓
```

但し、この情報をmodelへ引き渡すことにより
modelは共通プログラムで処理を行う事ができる。
※詳しくは、modelで説明

トランザクションを実行し、DBから戻ってきた時に行う処理を記述する。
この場合は、Modelに同じ関数名で記述された処理を呼び出している。

このレスポンス編集処理では、エクセルのタイトル部分に名前をセットしている。
タイトルにはシステム名を明記しないと何の情報かというのがわからない。
そのため共通プログラムではなく個々のプログラムで記述する必要がある。

モデル

① リクエストデータ処理

② レスポンスデータ処理

上記2つの処理を作成します

（後日、共通プログラムmixinへ移した為**作成不要**になりました）

①リクエストデータ処理
「印刷チェック行の管理NO」と「エラーメッセージ」を作成しています

②レスポンスデータ処理
「生成するエクセルのファイル名」と「印刷実行日時」を作成しています

モデル

①リクエストデータ処理

```
// ---------------------------------
// リクエストデータ　チェック処理
// ---------------------------------
on一覧表印刷個別OfCheckRequestData: function(requestData, mode) {
 $R.log("Model on一覧表印刷個別OfCheckRequestData : start");
 var dataSet       = this.dataset.getData();
 var detailRecord  = this.appspec.getJSONChunkByIdAtRecords(dataSet, "detail")["record"];
 var maxSize       = detailRecord["入出力要素ＮＯ"]["value"].length;
 var j             = 0;
 var requestRecord = this.appspec.getJSONChunkByIdAtRecords(requestData, "header")["record"];
 var w_選択        = [];
 for (var i = 0; i < maxSize; i++) {
  if ( $($(".印刷")[i]).is(':checked') ) {
    w_選択[j] = detailRecord["入出力要素ＮＯ"]["value"][i];
    j++;
  }
 }
 if (j > 0) {
  requestRecord["印刷入出力要素ＮＯ"]["value"] = w_選択;
  return true;                          ←「印刷入出力要素NO」を作る　項目名ができたので追加が必要
 }
 var arg = {};
 arg["title"]   = "印刷選択エラー";      ←選択なしで個別印刷を行った時のエラーを表示する（個別印刷のみ）
 arg["status"]  = "ERROR";
 arg["message"] = "印刷が選択されていません。選択して下さい。";
 this.pubsub.publish("alertDialog", arg);
 $R.log("Model on一覧表印刷個別OfCheckRequestData : end");
 return false;
}
on一覧表印刷一括OfCheckRequestData: function(requestData, mode) {
 $R.log("Model on一覧表印刷一括OfCheckRequestData : start");      ←一括にはエラーメッセージの指示なし
 $R.log("Model on一覧表印刷一括OfCheckRequestData : end");
 return true;
}
```

手順：
1.コピー元モデルから図の処理をコピペする

説明：
「印刷チェック行の管理NO」と
「エラーメッセージ」を作成しています

●印刷チェック行の管理NO
チェック時にどのデータが選択されたのかを管理するＮＯ［**印刷入出力要素NO**］をつくり印刷する行を管理します。
ここで新しい項目名ができたので、
データセットJSONに記述がいります。

●エラーメッセージ作成
［一覧印刷個別］の方が指示が多い。
チェックした項目だけを印刷する為に、
チェック無し状態で印刷ボタンがクリックされた時はエラーメッセージを表示する設定があるため。

モデル

②レスポンスデータ処理

```
// ------------------------------------------------------
// レスポンスデータ　編集処理
// ------------------------------------------------------
on一覧表印刷個別OfEditResponseData: function(responseData, mode) {
 $R.log("Model on一覧表印刷個別OfEditResponseData : start");
 var date     = new Date();
 var datetime = date.getFullYear()    + "年"
          + (date.getMonth() + 1) + "月"
          + date.getDate()        + "日"
          + date.getHours()       + "時"
          + date.getMinutes()     + "分";
 var headerRecord  = this.appspec.getJSONChunkByIdAtRecords(responseData, "header");
 var downloadfile  = headerRecord["record"]["downloadfile"]["value"][0];
 var argHash  = new Object();
 argHash["file"]     = downloadfile;
 argHash["type"]     = "xlsx";                    ↓日付けを作成したものを入れる
 argHash["download"] = "ＲＥＳＴ一覧個別(" + datetime + ").xlsx";
 this.postDownloadRack(argHash);                  ←エクセルのタイトルを作成（個別）
 $R.log("Model on一覧表印刷個別OfEditResponseData : end");
}
on一覧表印刷一括OfEditResponseData: function(responseData, mode) {
 $R.log("Model on一覧表印刷一括OfEditResponseData : start");
 var date     = new Date();
 var datetime = date.getFullYear()    + "年"
          + (date.getMonth() + 1) + "月"
          + date.getDate()        + "日"
          + date.getHours()       + "時"
          + date.getMinutes()     + "分";
 var headerRecord  = this.appspec.getJSONChunkByIdAtRecords(responseData, "header");
 var downloadfile  = headerRecord["record"]["downloadfile"]["value"][0];
 var argHash  = new Object();
 argHash["file"]     = downloadfile;
 argHash["type"]     = "xlsx";                    ↓日付けを作成したものを入れる
 argHash["download"] = "ＲＥＳＴ一覧一括(" + datetime + ").xlsx";
 this.postDownloadRack(argHash);                  ←エクセルのタイトルを作成（一括）
 $R.log("Model on一覧表印刷一括OfEditResponseData : end");
}                                                 ※作られたエクセルはダウンロードフォルダーへ出力される
```

手順：
1.コピー元のモデルから図の処理をコピペする

2.赤枠内の項目名を作成する
これは生成されるエクセル名です
［**REST一覧個別**］と［**REST一覧一括**］

3.エクセル名に印刷日時を追記します
［**+datetime+**］
（コピー元にあるのでそのままで使います）

説明：
「生成するエクセル名を作成」と
「印刷実行日時」を作成しています

● 「印刷実行日時」
［Var datetime＝］で印刷時刻を作成しています

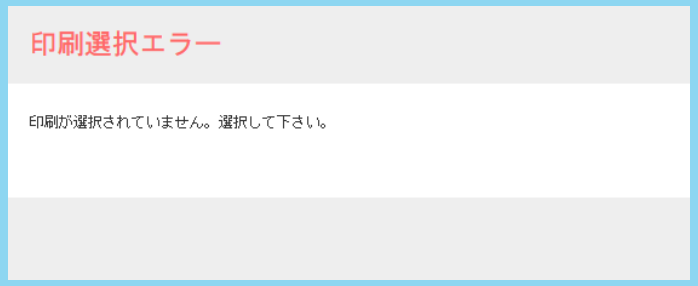
● 「生成するエクセル名を作成」
［argHash］でdownloadフォルダーへこのタイトル+日付け名のエクセルを生成します

モデル【詳細説明】

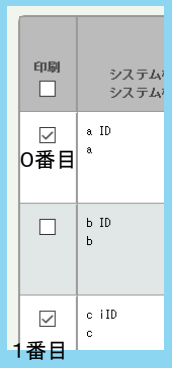
リクエストデータ　編集　チェック処理

まず画面上で一覧表印刷（個別）が押されると、まずmodelで下記チェックを行う。

チェックなしで実行時



チェックありで実行時



個別印刷は画面上で印刷にチェックを入れたデータのみ
印刷を行う機能のため、画面上でのチェックは必須である。

チェックをせずに一覧表印刷（個別）ボタンを押した場合、
左記のエラーを表示させる。

チェックされたデータの入出力要素NOを
まずW_選択にセットし、その内容を次に
requestRecord：印刷入出力要素NOにセットしている。

Index.html(tbody)

```
<tbody>↓
    <tr>↓
        <td class="width50 rmenuCenter">↓
            <input name="印刷" type="checkbox" class="rfocusblue 印刷">↓
        </td>↓
        <td class="width150 rmenuLeft システム機能ＩＤ名称"></td>↓
        <td class="width100">↓
            <div class="rmenuLeft 帳票ＩＤ"></div>↓
            <div class="rmenuLeft 帳票名"></div>↓
        </td>↓
        <td class="width200">↓
            <div class="rmenuLeft 概要"></div>↓
            <div class="rmenuLeft バッチオンライン"></div>↓
        </td>↓
```

モデル【詳細説明】

リクエストデータ　編集　チェック処理

コントローラから呼び出された関数。
トランザクションを実行する前に行うチェック内容を記述する。

コントローラと同じく共通プログラムに記述している。
mastermaintelist2.model.mixin-2.1.1.js

```
303     //--------------------------------------↓
304     //  リクエストデータ□チェック処理↓
305     //--------------------------------------↓
306     ,on一覧表印刷個別OfCheckRequestData: function(requestData, mode) {↓
307         $R.log("Model on一覧表印刷個別OfCheckRequestData : start");↓
308 ↓
309         var dataSet         = this.dataset.getData();↓
310         var detailRecord    = this.appspec.getJSONChunkByIdAtRecords(dataSet,"detail")["record"];↓
311         var maxSize         = detailRecord["入出力要素ＮＯ"]["value"].length;↓
312         var j               = 0;↓
313         var requestRecord   = this.appspec.getJSONChunkByIdAtRecords(requestData,"header")["record"];↓
314         var w_選択          = [];↓
315         ↓
316         for (var i = 0; i < maxSize; i++) {↓
317             if ( $($("印刷")[i]).is(':checked') ) {↓
318                 w_選択[j] = detailRecord["入出力要素ＮＯ"]["value"][i];↓
319                 j++;↓
320             }↓
321         }↓
322         ↓
323         if (j > 0) {↓
324             requestRecord["印刷入出力要素ＮＯ"]["value"] = w_選択;↓
325             return true;↓
326         }↓
327         ↓
328         var arg = [];↓
329         arg["title"]   = "印刷選択エラー";↓
330         arg["status"]  = "ERROR";↓
331         arg["message"] = "印刷が選択されていません。選択して下さい。";↓
332         this.pubsub.publish( alertDialog ,arg);↓
333         ↓
334         $R.log("Model on一覧表印刷個別OfCheckRequestData : end");↓
335         return false;↓
336     }↓
337 ↓
338     ,on一覧表印刷一括OfCheckRequestData: function(requestData, mode) {↓
339         $R.log("Model on一覧表印刷一括OfCheckRequestData : start");↓
340         ↓
341         ↓
342         $R.log("Model on一覧表印刷一括OfCheckRequestData : end");↓
343         return true;↓
344     }↓
345     ↓
```

For文で印刷ボタンがチェックされているかを確認している。

印刷ボタンにチェックされたデータの入出力要素noを上から順番に数える。[i]

W_選択という空の配列を用意し、
上から順番に0番目、1番目という形で配列の中に追加している。

印刷ボタンにチェックされた数が0以上であれば、
配列:W_選択の内容をrequestRdcord:印刷入出力要素NOにセットしている。
チェックの数が0件の場合は、エラーを表示させる。

一括印刷は、印刷部分のチェック有無にかかわらず画面上に表示されているデータを全件処理する為、
個別印刷のようなチェックは必要ない。

上記の項目がOKであれば、トランザクションを実行する。

モデル【詳細説明】

レスポンスデータ　編集処理

コントローラのリクエストチェック処理・モデル編集処理、モデルのリクエストチェック処理については、
個別印刷・一括印刷とそれぞれの関数が記述されていた。
しかしモデルのレスポンス編集処理において記述された処理は1つである。

レスポンスデータ編集処理には、
トランザクションを実行し、DBから戻ってきた時に行う処理を記述する。

サーバーでは、もともと存在するエクセルのテンプレート情報を読み込み、
そこへDBから取得したデータ情報をエクセルに貼り付けている。
つまりこのレスポンスデータ編集処理を行う時点で、エクセルの中身はできている状態である。

ここでは完成したエクセルをパソコンでダウンロードするときに、タイトル部分に名前をセットしている。

mastermaintelist2.model.mixin-2.1.1.js

```
399     //-------------------------------------------------------------↓
400     //  一覧表印刷□レスポンスデータ□編集処理↓
401     //-------------------------------------------------------------↓
402     ,on一覧表印刷OfEditResponseData: function(responseData, mode, listname) {↓
403         $R.log("Model on一覧表印刷OfEditResponseData : start ");↓
404     ↓
405         var date     = new Date();↓
406         var datetime = date.getFullYear()     + "年"↓
407                      + (date.getMonth() + 1)  + "月"↓
408                      + date.getDate()         + "日"↓
409                      + date.getHours()        + "時"↓
410                      + date.getMinutes()      + "分";↓
411     ↓
412         var headerRecord = this.appspec.getJSONChunkByIdAtRecords(responseData, "header");↓
413         var downloadfile = headerRecord["record"]["downloadfile"]["value"][0];↓
414     ↓
415         var argHash = new Object();↓
416         argHash["file"]     = downloadfile;↓
417         argHash["type"]     = "xlsx";↓
418         argHash["download"] = listname + "(" + datetime + ").xlsx";↓
419         this.postDownloadRack(argHash);↓
420     ↓
421         $R.log("Model on一覧表印刷OfEditResponseData : end");↓
422     }↓
423     ↓
424 };↓
```

Controllerのレスポンスデータ編集処理を確認。

datetimeに日付をセットしている。

いつダウンロードしたデータなのかをわかるようにするため、
日付と時刻を設定し、ファイル名にしている。

ここでタイトルをセットしている。
Listnameに帳票一覧個別、rest一覧一括など
controllerから受け取った情報がセットされる。

今回はエクセルファイルだが、CSV、画像などいろんな種類のデータをダウンロードする処理がある。
'type"とはダウンロードするファイルの種類を表し、csv,jpgなどに変更することでダウンロードするファイルの種類も変更する。

モデル【詳細説明】

レスポンスデータ　編集処理

エクセルデータ

帳票一覧個別(2017年9月19日13時3分).xlsx - 保護ビュー - Excel

保護ビュー 注意―インターネットから入手したファイルは、ウイルスに感染している可能性があります。編集する必要がなければ、保護ビューのままにしておくことをお勧めします。

サブシステム：MH（共通マスタ）　管理単位：MH01（共通マスタ）

No	システム機能		帳票ID	帳票名	概要	バッチ/オンライン	画面/バッ
	機能ID	機能名					
1	a ID	a	ID a	a	a	オンライン	ID a a
2	b ID	b	ID b	b	b	オンライン	ID b b

上記のmodelは共通プログラムである。
その為、個別印刷・一括印刷かだけではなくこの関数を必要とする全てのプログラムの処理を行う。

Controller:レスポンスデータ　編集処理

```
70  //----------------------------------↓
71  //　レスポンスデータ□編集処理↓
72  //----------------------------------↓
73  ,on一覧表印刷個別OfEditResponseData: function(responseData, mode) {↓
74  $R.log("Controller on一覧表印刷個別OfEditResponseData : start");↓
75  ↓
76  //this.editResponseData(responseData);↓
77  this.model.on一覧表印刷OfEditResponseData(responseData, mode, "帳票一覧個別");↓
78  ↓
79  $R.log("Controller on一覧表印刷個別OfEditResponseData : end");↓
80  }↓
81  ↓
82  ,on一覧表印刷一括OfEditResponseData: function(responseData, mode) {↓
83  $R.log("Controller on一覧表印刷一括OfEditResponseData : start");↓
84  ↓
85  //this.editResponseData(responseData);↓
86  this.model.on一覧表印刷OfEditResponseData(responseData, mode, "帳票一覧一括");↓
87  ↓
88  $R.log("Controller on一覧表印刷一括OfEditResponseData : end");↓
89  }↓
90  ↓
91  });↓
```

そこで個々のcontrollerから受け取った情報をlistnameにセットするようにしている。

そうすることで、modelは共通プログラムに記述することができ、なおかつプログラムは1つで済む。

実践5『エクセル印刷機能』作成
サーバー側の作成

データセット JSON → トランJSON → SQLJSON

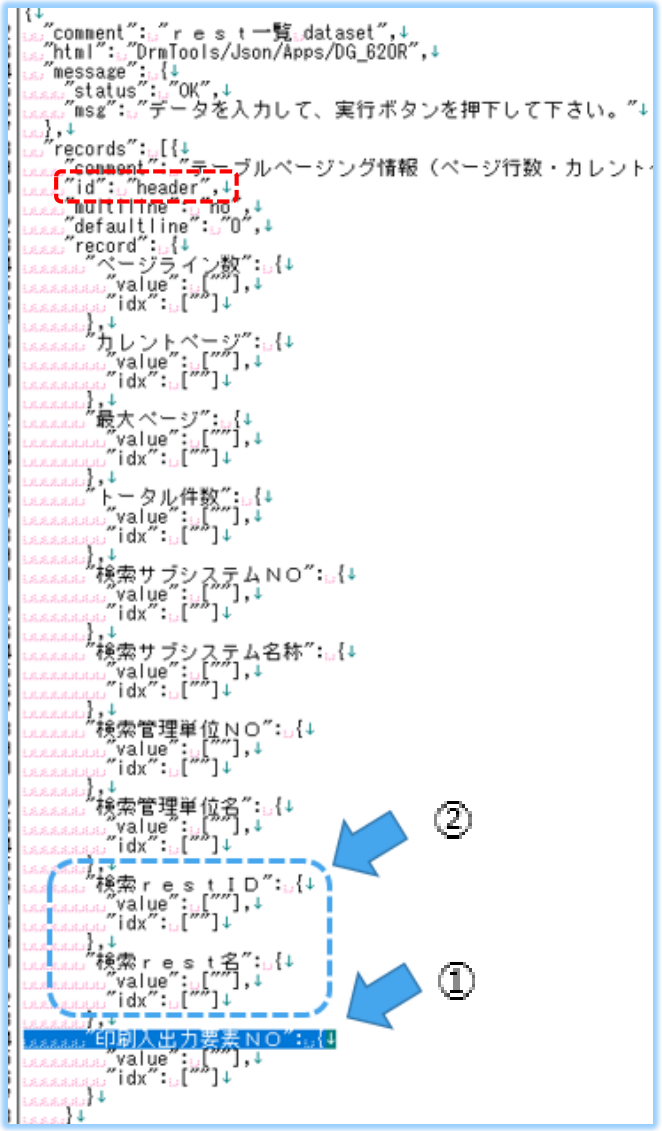
- サーバー側を作成します

データセットJSON：モデルで作成した項目を追加作成

トランJSON： printchecktranJSON/printalltranJSONの2つを作成します

SQLJSON： printchecksqlJSON/printallsqlJSONの2つを作成します

データセットJSON



データセットJSONを作成します

①モデルでできた項目名をデータセットに作成する。
［**印刷入出力要素NO**］の項目ができたのでデータセットできるように追加する。

このNOは印刷個別時にチェックされた行を認識するためのＮＯです

②個別印刷と一括印刷の両方で使用する

トランJSON

トランJSONを3ステップで作成します

データセットJSONを参考に項目を作成します

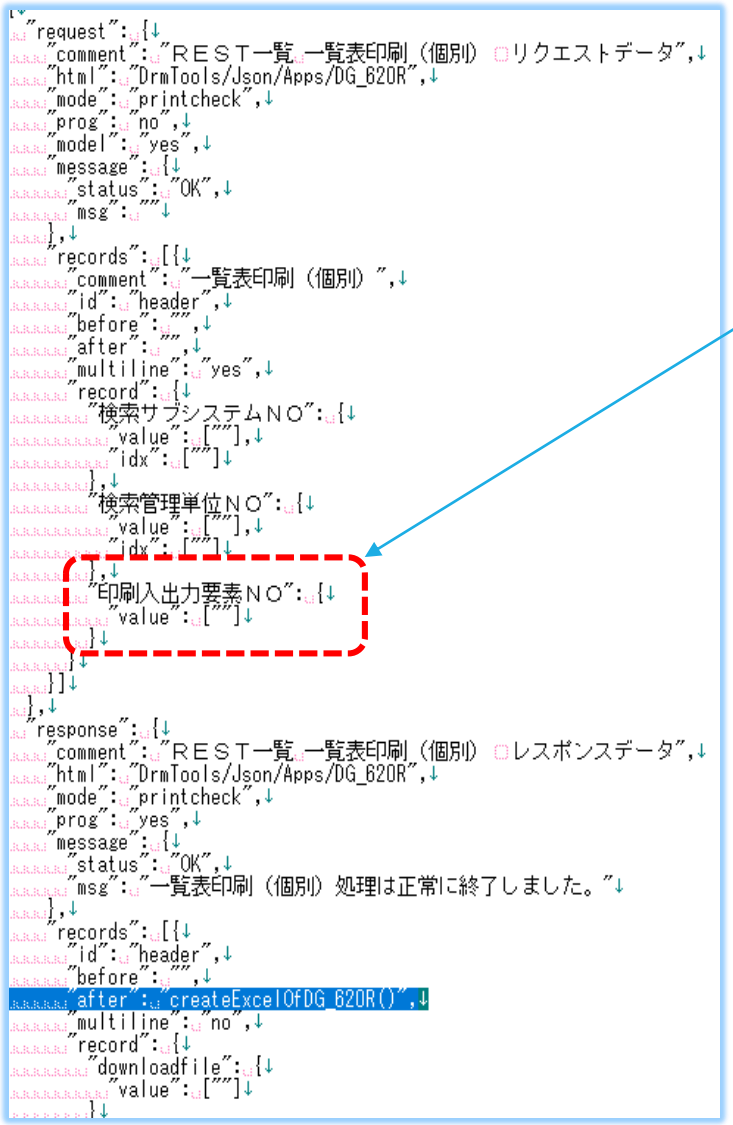
①printchecktranJSONの作成

②printalltranJSONの作成

③Afterの作成

トランJSON

①printchecktranJSONを作成する



```
{
  "request": {
    "comment": "ＲＥＳＴ一覧_一覧表印刷_(個別) □リクエストデータ",
    "html": "DrmTools/Json/Apps/DG_620R",
    "mode": "printcheck",
    "prog": "no",
    "model": "yes",
    "message": {
      "status": "OK",
      "msg": ""
    },
    "records": [{
      "comment": "一覧表印刷（個別）",
      "id": "header",
      "before": "",
      "after": "",
      "multiline": "yes",
      "record": {
        "検索サブシステムＮＯ": {
          "value": [""],
          "idx": [""]
        },
        "検索管理単位ＮＯ": {
          "value": [""],
          "idx": [""]
        },
        "印刷入出力要素ＮＯ": {
          "value": [""]
        }
      }
    }]
  },
  "response": {
    "comment": "ＲＥＳＴ一覧_一覧表印刷_(個別) □レスポンスデータ",
    "html": "DrmTools/Json/Apps/DG_620R",
    "mode": "printcheck",
    "prog": "yes",
    "message": {
      "status": "OK",
      "msg": "一覧表印刷（個別）処理は正常に終了しました。"
    },
    "records": [{
      "id": "header",
      "before": "",
      "after": "createExcelOfDG_620R()",
      "multiline": "no",
      "record": {
        "downloadfile": {
          "value": [""]
        }
```

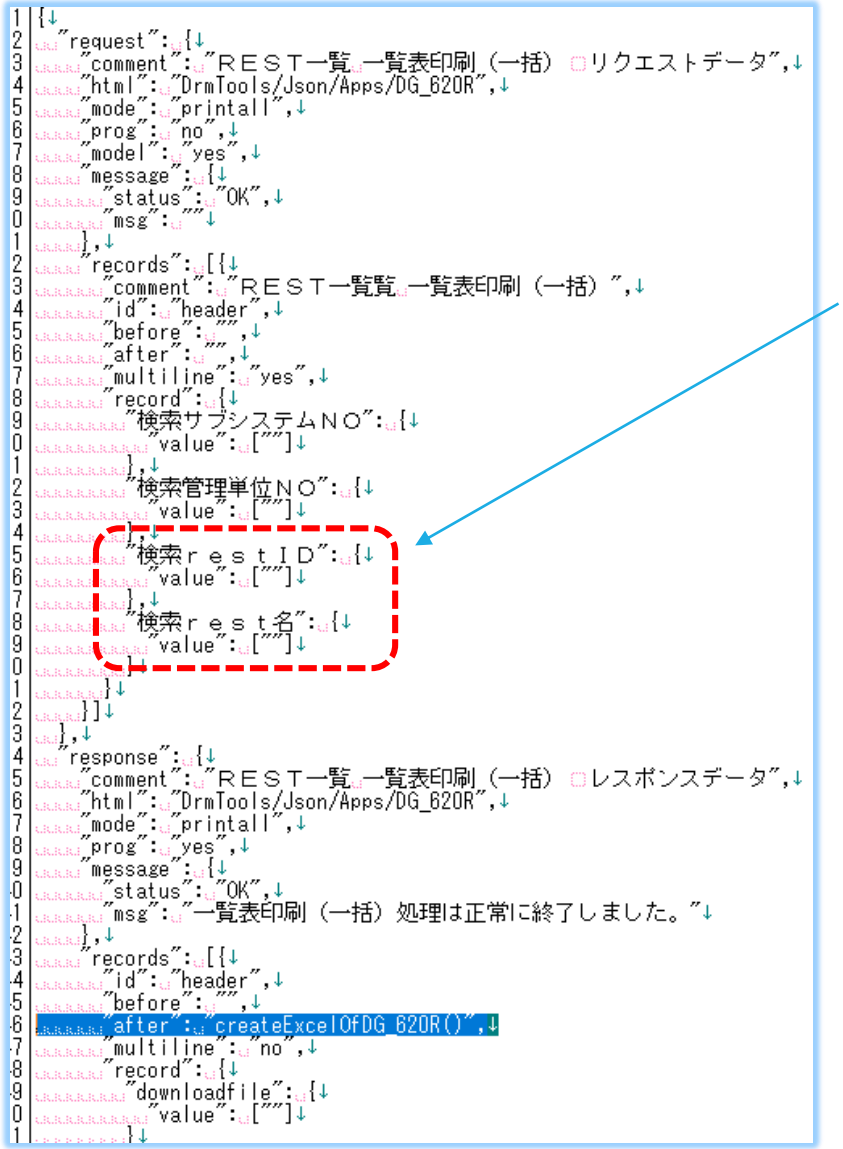
データセットJSONを参考に項目を作成します

Checkは選択された項目だけを印刷する為

［**印刷入出力要素NO**］があるものを印刷する
そのため、トランに項目を用意する

トランJSON

②printalltranJSONを作成する



```
{
  "request": {
    "comment": "ＲＥＳＴ一覧_一覧表印刷（一括）□リクエストデータ",
    "html": "DrmTools/Json/Apps/DG_620R",
    "mode": "printall",
    "prog": "no",
    "model": "yes",
    "message": {
      "status": "OK",
      "msg": ""
    },
    "records": [{
      "comment": "ＲＥＳＴ一覧_一覧表印刷（一括）",
      "id": "header",
      "before": "",
      "after": "",
      "multiline": "yes",
      "record": {
        "検索サブシステムＮＯ": {
          "value": [""]
        },
        "検索管理単位ＮＯ": {
          "value": [""]
        },
        "検索ｒｅｓｔＩＤ": {
          "value": [""]
        },
        "検索ｒｅｓｔ名": {
          "value": [""]
        }
      }
    }]
  },
  "response": {
    "comment": "ＲＥＳＴ一覧_一覧表印刷（一括）□レスポンスデータ",
    "html": "DrmTools/Json/Apps/DG_620R",
    "mode": "printall",
    "prog": "yes",
    "message": {
      "status": "OK",
      "msg": "一覧表印刷（一括）処理は正常に終了しました。"
    },
    "records": [{
      "id": "header",
      "before": "",
      "after": "createExcelOfDG_620R()",
      "multiline": "no",
      "record": {
        "downloadfile": {
          "value": [""]
        }
```

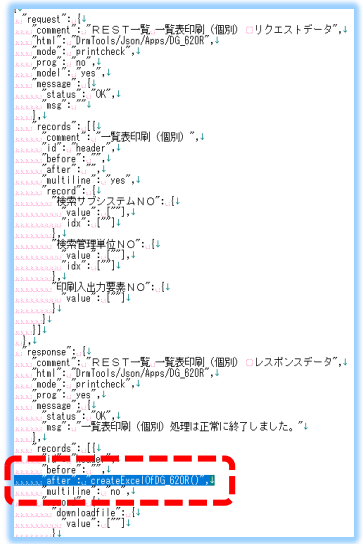
データセットJSONを参考に項目を作成します

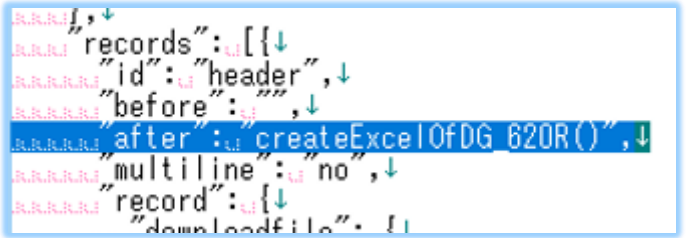
Allはすべてを印刷する為

画面に［**検索restID**］［**検索rest名**］があるものを全て印刷するのに、2つの項目が必要になる

- トランJSON

③Afterの作成





[ａｆｔｅｒ]のプログラム名の作成

[ａｆｔｅｒ(アフター)]はｒｕｂｙのviewへ飛ぶ指示です。
プログラム名を[DG_620R]で作成する

createExcel(クリエイトエクセル)は
「レスポンス結果はエクセルの項目へrubyのviewを使って書き込んで」の指示。
※レスポンスに項目名を書かず**rubyへデータを送り、rubyに項目名を作成します**。

両方のトランJSONの下部にAfterがありますこれはRubyのビューを使う指示です

- トランJSON【詳細説明】

作成中

SQLJSON

SQLJSONを3ステップで作成します

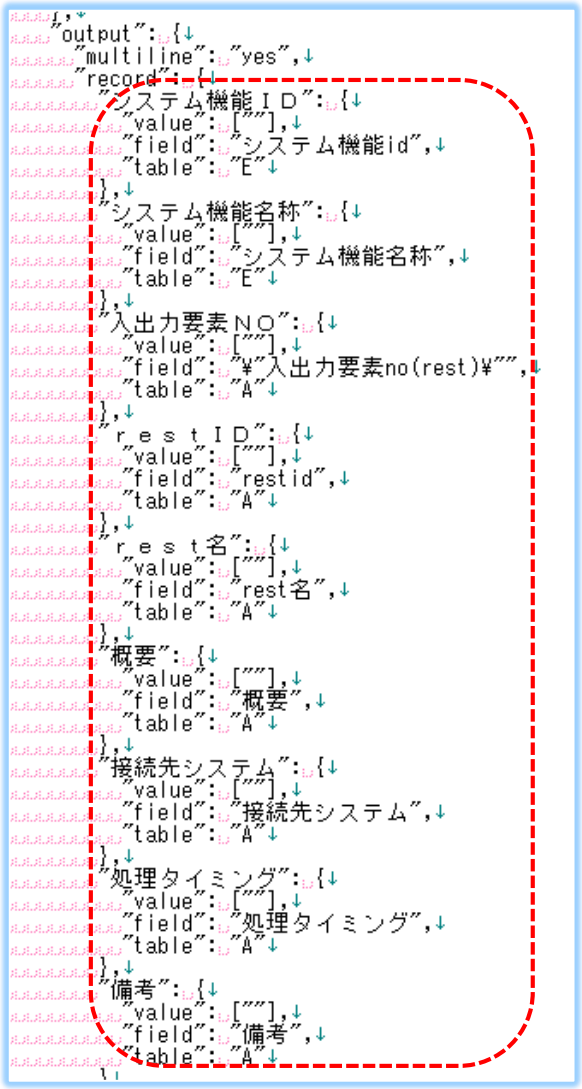
①両sqlの作成

②printchecksqlJSON
　［印刷個別］ボタンクリック時のＳＱＬを作成します

③printallsqlJSON
　［印刷一括］ボタンクリック時のＳＱＬを作成します

- SQLJSON

①両ＳＱＬのアウトプットを作成



```
},
"output": {
  "multiline": "yes",
  "record": {
    "システム機能ＩＤ": {
      "value": [""],
      "field": "システム機能id",
      "table": "E"
    },
    "システム機能名称": {
      "value": [""],
      "field": "システム機能名称",
      "table": "E"
    },
    "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": [""],
      "field": "概要",
      "table": "A"
    },
    "接続先システム": {
      "value": [""],
      "field": "接続先システム",
      "table": "A"
    },
    "処理タイミング": {
      "value": [""],
      "field": "処理タイミング",
      "table": "A"
    },
    "備考": {
      "value": [""],
      "field": "備考",
      "table": "A"
    }
```

1.アウトプットをそれぞれ作成する

「SQLは印刷一括の[**printallsql**]と印刷個別[**printchecksql**]があり必要項目名をそれぞれ作成。」

[**id：明細**]の**アウトプット作成する。**
（上部2つはコピー元と同じの為そのままで良い）

ジェネレートSQLの為、
アウトプットの［**field**］と［**table**］に注意して作成する

※【Printallsqlとprintchecksqlの構造】3つのSQLで構成されている

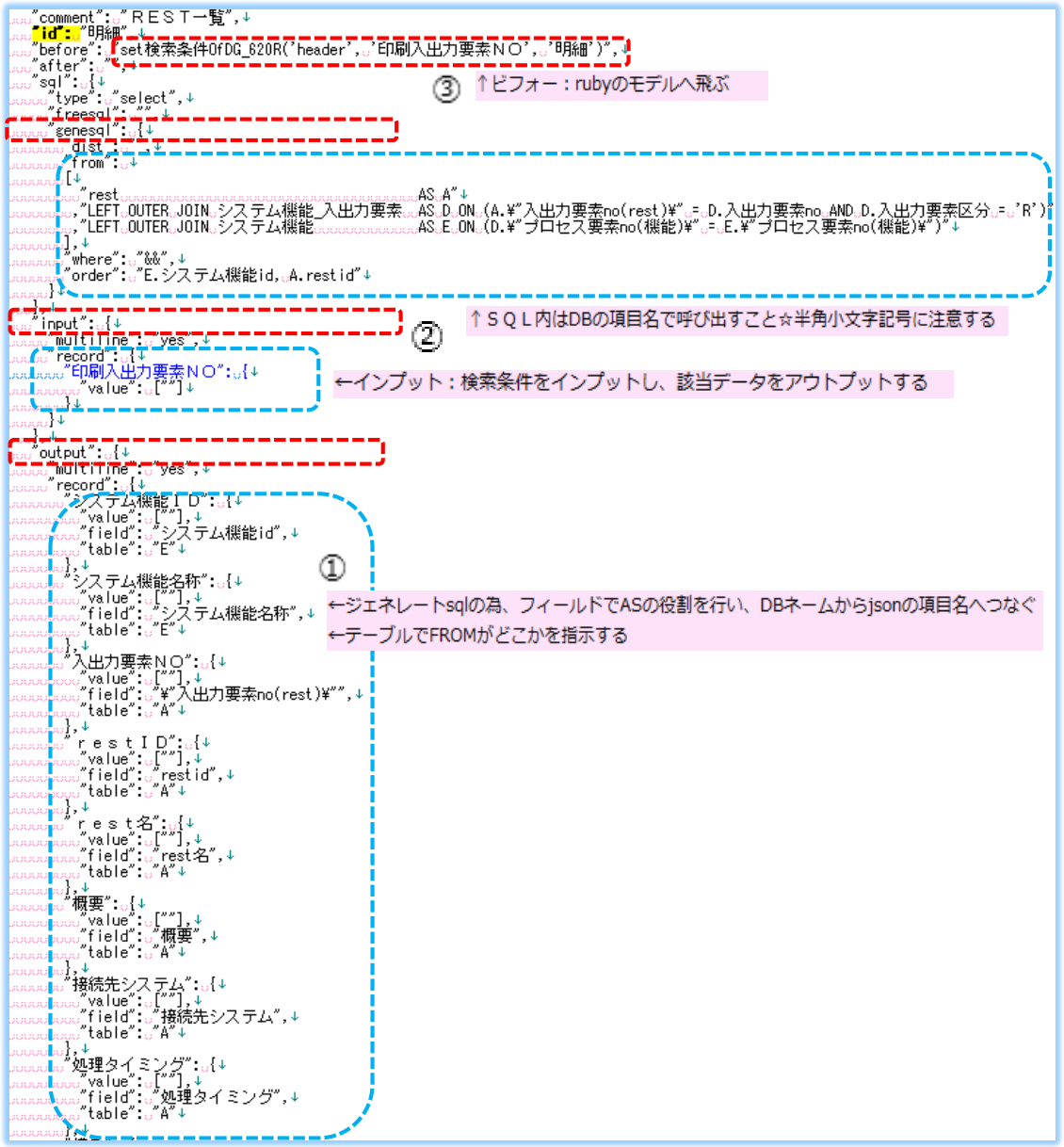
"id": "サブシステム",（ヘッダーの条件セレクトBOX1：エクセルシートのヘッダーに印字する）

"id": "管理単位",（ヘッダーの条件セレクトBOX2：エクセルシートのヘッダーに印字する）

"id": "明細",（ディティールのデータ：エクセルシートの明細に印字する）

SQLJSON

②printchecksqlJSON



※(赤枠を確認し、青枠内を作成します。)

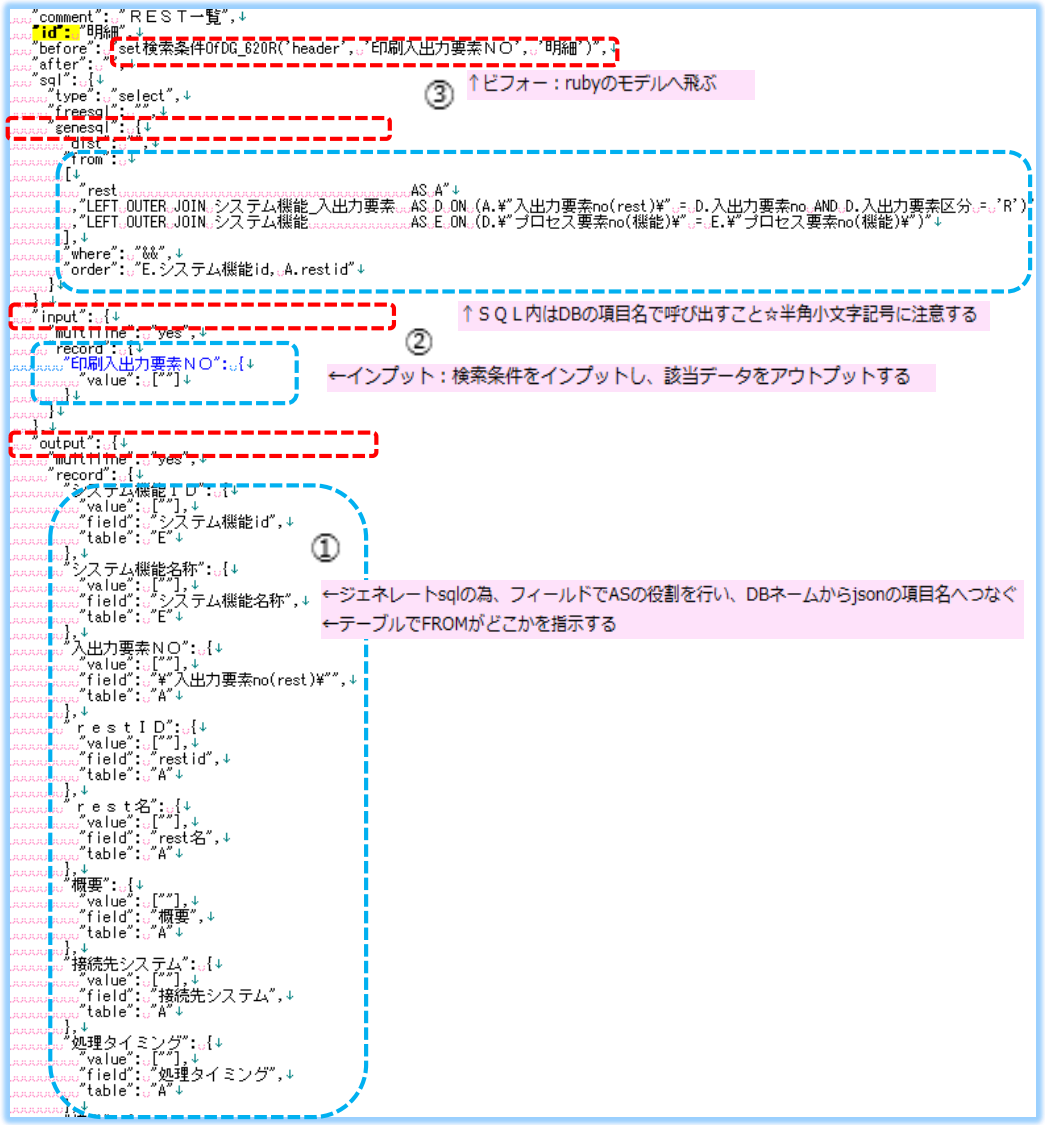
1.**ジェネレートSQL**を作成する
項目名の作成と、[**入出力要素区分**]を忘れないように注意する。

2.**インプットの項目名**を［**印刷入出力要素NO**］へ変更する。（選択した行をインプットさせる）

3.[**before**]に指示があるのでrubyのモデルを使用している

SQLJSON

③printallsqlJSON



※(赤枠を確認し、青枠内を作成します。)

1.**ジェネレートSQL**を作成する

※項目名の作成と、[**入出力要素区分**]を忘れないように注意する。

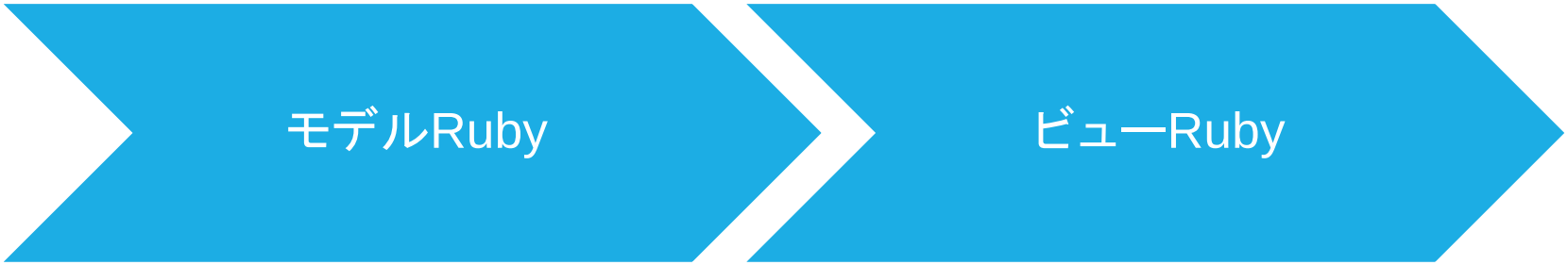
2.**インプットの項目名**を
［**検索restID/名**］へ変更する。

3.[**before**]に指示があるので
rubyのモデルを使用している。

- SQLJSON【詳細説明】

作成中

実践5『エクセル印刷機能』作成
Rubyの作成



- Rubyを作成します

プログラムの場所：
server→プログラム名→各ruby（コントローラ/モデル/ビュー）にあります。

モデルRuby：SQLJSONのbeforeから来る

ビューRuby：トランJSONのAfterから来る

モデルRuby

上記2つのRubyを作成します

①コピー元からプログラムをコピーして作成

②printcheckSQLのbeforeを作成

③printallSQLのbeforeを作成

モデルRuby

①コピーして作成

```
#SQL文の検索条件を変更する（検索データ項目名無し）
def setSql_DG800R0fDrmTools(idname1, idname2,genesql_freesql)
  $Nlog.debug("DG_800R_model") "setSql_DG800R0fDrmTools start"                    #Logファイル Debug用
  requestInfo = getJsonChunkById(@request_data, "records", idname1)
  sqlInfo = getJsonChunkById(@sql_data, "sqls", idname2)
  searchString01 = requestInfo["record"]["検索画面ＩＤ"]["value"][0]
  searchString02 = requestInfo["record"]["検索画面名"]["value"][0]
  searchString03 = requestInfo["record"]["検索管理単位ＮＯ"]["value"][0]
  searchString04 = requestInfo["record"]["検索サブシステムＮＯ"]["value"][0]
  rep = ""
  #検索画面ＩＤ
  if searchString01 != ""
    rep = rep + " AND A.画面id LIKE '%#{searchString01}%'"
  end
  #検索画面名
  if searchString02 != ""
    rep = rep + " AND A.画面名 LIKE '%#{searchString02}%'"
  end
  #検索管理単位ＮＯ
  if searchString03 != ""
    rep = rep + " AND A.管理単位no = #{searchString03}"
  end
  #検索サブシステムＮＯ
  if searchString04 != ""
    rep = rep + " AND B.サブシステムno = #{searchString04}"
  end
  #検索条件の置き換え
  if genesql_freesql == 'genesql'
    sqlInfo["sql"]["genesql"]["where"].sub!(/&&/, rep)
  elsif genesql_freesql == 'freesql'
    strJoin = sqlInfo["sql"]["freesql"].join("¥t")
    strJoin.sub!(/&&/, rep)
    strSplit = strJoin.split("¥t")
    sqlInfo["sql"]["freesql"].each_with_index do |temp, idx|
      sqlInfo["sql"]["freesql"][idx] = strSplit[idx]
    end
  end
  $Nlog.debug("DG_800R_model") "setSql_DG800R0fDrmTools end"                      #Logファイル Debug用
  return "OK"
end

#SQL文の検索条件を変更する（検索データ項目名有り）
def set検索条件0fDG_800R(idname1, itemname1, idname2)
  $Nlog.debug("DG_800R_model") "set検索条件0fDG_800R start"                       #Logファイル Debug用

  requestRecord = getJsonChunkById(@request_data, "records", idname1, "record")
  sqlRecord = getJsonChunkById(@sql_data, "sqls", idname2)

  if requestRecord[itemname1]["value"][0] == ""
    $Nlog.debug("DG_800R_model") "set検索条件0fDG_800R end"
    return "PASS"
  end

  maxSize = requestRecord[itemname1]["value"].length - 1
  rep = "A.¥"入出力要素no(画面)¥" IN ("
  for num in 0..maxSize do
    if num == 0
      rep = rep + requestRecord[itemname1]["value"][num].to_s
    else
      rep = rep + "," + requestRecord[itemname1]["value"][num].to_s
    end
  end
  rep = rep + ")"
  #where条件を置き換え
  sqlRecord["sql"]["genesql"]["where"].sub!(/&&/, rep)
  $Nlog.debug("DG_800R_model") "set検索条件0fDG_800R end"                         #Logファイル Debug用
  return "OK"
end
```

SQLの[**before**]に記載があった場合、
[**server**]→[**モデルRuby**]のファイルを開き、
指示を確認する。

[**before**]の後の文書が指示の飛ぶ場所が書かれています。
個別と一括の2つあります。

```
"comment": "ＲＥＳＴ一覧",
"id": "明細",
"before": "set検索条件0fDG_620R('header','印刷入出力要素ＮＯ','明細')",
"after": "",
"sql": {
  "type": "select",
```

③ ↑ビフォー：rubyのモデルへ飛ぶ

コピー元から上記の［**def～end**］部分をコピペし、
[**プログラム名**]と[**項目名**]を変更する。

モデルRuby

②printcheckSQLのbeforeを作成

```
#SQL文の検索条件を変更する（検索データ項目名有り）
def set検索条件OfDG_620R(idname1, itemname1, idname2)
  $Mlog.debug("DG_620R_model"){"set検索条件OfDG_620R start"}                    #Logファイル Debug用
  requestRecord = getJsonChunkById(@request_data,"records",idname1,"record")
  sqlRecord     = getJsonChunkById(@sql_data,"sqls",idname2)
  if requestRecord[itemname1]["value"][0] == ""
    $Mlog.debug("DG_620R_model"){"set検索条件OfDG_620R end"}
    return "PASS"
  end
  maxSize = requestRecord[itemname1]["value"].length - 1
  rep     = "A.¥"入出力要素no(rest)¥" IN ("
  for num in 0..maxSize do
    if num == 0
      rep = rep + requestRecord[itemname1]["value"][num].to_s
    else
      rep = rep + "," + requestRecord[itemname1]["value"][num].to_s
    end
  end
  rep = rep + ")"
  #where条件を置き換え
  sqlRecord["sql"]["genesql"]["where"].sub!(/&&/,rep)
  $Mlog.debug("DG_620R_model"){"set検索条件OfDG_620R end"}                      #Logファイル Debug用
  return "OK"
end
```

printchecksql（個別）で指示されているｒｕｂｙを作成する

1.コピー元の[**def**]を検索し、
[**“before”: “set検索条件OfDG_620R(‘header’, ‘印刷入出力要素ＮＯ’, ‘明細’)”,**]部分をコピペする

2.プログラム名[**DG_620R**]の作成と、［**項目名**］を作成する。
※ここではSQL文を作成しているため、DBの項目名を使用しないとエラーになるので注意

モデルRuby

③printallSQLのbeforeを作成

```
# SQL文の検索条件を変更する（検索データ項目名無し）
def setSql_DG620ROfDrmTools(idname1, idname2 ,genesql_freesql)
  $Mlog.debug("DG_620R_model") {"setSql_DG620ROfDrmTools start"}                    # Logファイル Debug用

  requestInfo     = getJsonChunkById(@request_data,"records",idname1)
  sqlInfo         = getJsonChunkById(@sql_data,"sqls",idname2)

  searchString01 = requestInfo["record"]["検索ｒｅｓｔＩＤ"]["value"][0]
  searchString02 = requestInfo["record"]["検索ｒｅｓｔ名"]["value"][0]
  searchString03 = requestInfo["record"]["検索管理単位ＮＯ"]["value"][0]
  searchString04 = requestInfo["record"]["検索サブシステムＮＯ"]["value"][0]

  rep = ""

  # 検索ｒｅｓｔＩＤ
  if searchString01 != ""
    rep = rep + " AND A.restid LIKE '%#{searchString01}%'"
  end

  # 検索ｒｅｓｔ名
  if searchString02 != ""
    rep = rep + " AND A.rest名 LIKE '%#{searchString02}%'"
  end

  # 検索管理単位ＮＯ
  if searchString03 != ""
    rep = rep + " AND A.管理単位no = #{searchString03}"
  end

  # 検索サブシステムＮＯ
  if searchString04 != ""
    rep = rep + " AND B.サブシステムno = #{searchString04}"
  end

  # 検索条件の置き換え
  if genesql_freesql == 'genesql'
  sqlInfo["sql"]["genesql"]["where"].sub!(/&&/,rep)
  elsif genesql_freesql == 'freesql'
  strJoin = sqlInfo["sql"]["freesql"].join("¥t")
  strJoin.sub!(/&&/,rep)
  strSplit = strJoin.split("¥t")
  sqlInfo["sql"]["freesql"].each_with_index do |temp,idx|
    sqlInfo["sql"]["freesql"][idx] = strSplit[idx]
  end
  end

  $Mlog.debug("DG_620R_model") {"setSql_DG620ROfDrmTools end"}                     # Logファイル Debug用
  return "OK"
end
```

ＳＱＬと繋がるところ

DBの項目名を使用しないとエラーになるので注意

Printallsql（一括）で指示されているｒｕｂｙを作成する。

1.コピー元の[**def**]を検索し、[**“before”:"setSql_DG620ROfDrmTools('header', '明細', 'genesql')",**]部分をコピペ

2.プログラム名[**DG_620R**]の作成と、［**項目名**］を作成する。

※ＳＱＬ文になる所はDBの項目名を使用しないとエラーになるので注意

ビューRuby

用意できたデータをビューRubyでアウトプットを作成します

①コピー元からプログラムをコピーして作成

②テンプレートの設定

③テンプレートへデータを作成する

ビューRuby

①コピーして作成

```
#エクセルを作成する
def createExcelOfDG_620R()                    ←トランのａｆｔｅｒ：から来た箇所
  $Vlog.debug("DG_620R_view"){"createExcelOfDG_620R start"}

  subsystemRecord = getJsonChunkById(@sql_data, "sqls", "サブシステム", "output", "record")
  controlRecord = getJsonChunkById(@sql_data, "sqls", "管理単位", "output", "record")
  detailRecord = getJsonChunkById(@sql_data, "sqls", "明細", "output", "record")

  #テンプレートファイル名設定
  excelFile = $Rconfig['apps_path'] + "/" + @response_data["html"].gsub(/¥/Json¥//, '/PdfTemplate/') + "/REST一覧_テンプレート.xlsx"

  #エクセルテンプレートを読み込む                エクセルシートの場所を指示している    エクセルの名前を設定する
  workbook = RubyXL::Parser.parse(excelFile)

  #ＲＥＳＴ一覧を作成する
  createPrintOfDG_620R(workbook, subsystemRecord, controlRecord, detailRecord)

  #当日日付とカレント時刻を設定する
  w_date = Date.today
  w_time = Time.now
  yyyymmdd = w_date.strftime("%Y%m%d")
  hhmmss = w_time.strftime("%H%M%S")            エクセルに入れる日付け作成

  #ディレクトリが無ければ作成する（有れば何もしない）
  downloadDir = $Rconfig['apps_path'] + "/DrmTools/DownLoad/Apps/DG_620R"
  newDir = $Rconfig['apps_path'] + "/DrmTools/DownLoad/Apps/DG_620R" + "/d" + yyyymmdd
  createDir(newDir)

  #当日よりも古いディレクトリーを削除する        ←作成したエクセルをダウンロードフォルダーへ出力するが
  deleteDir(downloadDir, newDir)                古いものを削除する指示

  #エクセルファイル出力
  excelFile = newDir + "/" + "data" + "#{hhmmss}.xlsx"
  workbook.write(excelFile)

  #エクセルファイル情報設定
  excelPath = "DrmTools/DownLoad/Apps/DG_620R" + "/d" + yyyymmdd + "/"
  filename = "data" + "#{hhmmss}.xlsx"

  responseInfo = getJsonChunkById(@response_data, "records", "header")
  responseInfo["record"]["downloadfile"]["value"][0] = excelPath + "/" + filename

  $Vlog.debug("DG_620R_view"){"createExcelOfDG_620R normal end"}
  return "OK"
end
```

トランJSONのAfterにある［ createExcelOFDG_620G ］をビューrubyへ作成します。

1.コピー元のビューRubyを開き、［ **createExcelOFDG_620G** ］をdef～endまでをコピペする

2.③で利用するプログラムを一緒にコピペする
テンプレートへデータを入れるために赤枠内の[**createprintOFDG_620G**]が必要なので一緒にコピペする

3.［プログラム名］［項目名］を作成する

- ビューRuby

②テンプレートの設定

```
#エクセルを作成する
def createExcelOfDG_620R()
    $Vlog.debug("DG_620R_view").{"createExcelOfDG_620R start"}

    subsystemRecord    = getJsonChunkById(@sql_data,      "sqls",      "サブシステム",    "output",  "record")
    controlRecord      = getJsonChunkById(@sql_data,      "sqls",      "管理単位",        "output",  "record")
    detailRecord       = getJsonChunkById(@sql_data,      "sqls",      "明細",            "output",  "record")


    #テンプレートファイル名設定
    excelFile = $Rconfig['apps_path'] + "/" + @response_data["html"].gsub(/¥/Json¥//,'/PdfTemplate/') + "/REST一覧_テンプレート.xlsx"

    #エクセルテンプレートを読み込む
    workbook = RubyXL::Parser.parse(excelFile)


    #PDFシート一覧を作成する
    createPrintOfDG_620R(workbook, subsystemRecord, controlRecord, detailRecord)

    #エクセルファイル出力
    excelFile = newDir + "/" + "data" + "#{hhmmss}.xlsx"
    workbook.write(excelFile)

    #エクセルファイル情報設定
    excelPath = "/DrmTools/DownLoad/Apps/DG_620R" + "/d" + yyyymmdd + "/"
    filename = "data" + "#{hhmmss}.xlsx"

    responseInfo = getJsonChunkById(@response_data, "records", "header")
    responseInfo["report"]["down_path_file"]["value"][0] = excelPath + "/" + filename

    $Vlog.debug("DG_620R_view").{"createExcelOfDG_620R normal end"}
    return "OK"
end
```


```
#テンプレートファイル名設定
excelFile = $Rconfig['apps_path'] + "/" + @response_data["html"].gsub(/¥/Json¥//,'/PdfTemplate/') + "/REST一覧_テンプレート.xlsx"

#エクセルテンプレートを読み込む
workbook = RubyXL::Parser.parse(excelFile)
```


エクセルシートの場所を指示している

エクセルの名前を設定する

テンプレートの場所を指示し読み込み

→初めに作成したテンプレート名[**REST一覧_テンプレート.xlsx**]を出力先にする指示を作成する。

- ビューRuby

③-1テンプレートへデータを作成する

```
#ＲＥＳＴ一覧を作成する
def createPrintOfDG_620R(workbook, subsystemRecord, controlRecord, detailRecord)
    $Vlog.debug("DG_620R_view") {"createPrintOfDG_620R start"}

    worksheet = workbook["REST一覧"]
    maxSize = detailRecord["入出力要素ＮＯ"]["value"].length
    #ヘッダーのサブシステム・管理単位を設定
    w_ヘッダー１ = "サブシステム：" + subsystemRecord["サブシステムＩＤ"]["value"][0] + " (" + subsystemRecord["サブシステム名称"]["value"][0] + ") "
    w_ヘッダー２ = "管理単位：" + controlRecord["管理単位ＩＤ"]["value"][0] + " (" + controlRecord["管理単位名"]["value"][0] + ") "
    w_ヘッダー３ = w_ヘッダー１ + "　□　" + w_ヘッダー２
    worksheet[0][0].change_contents(w_ヘッダー３, worksheet[0][0].formula)

    #明細を設定する
    #52行を超える時、行を追加する
    if maxSize > 52
        addSize = maxSize - 52
```

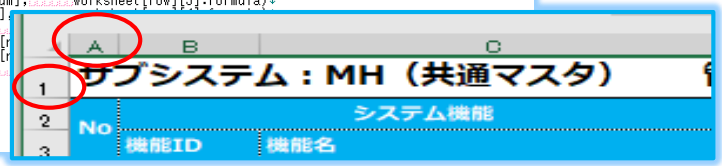
```
#ヘッダーのサブシステム・管理単位を設定
w_ヘッダー１ = "サブシステム：" + subsystemRecord["サブシステムＩＤ"]["value"][0] + " (" + subsystemRecord["サブシステム名称"]["value"][0] + ") "
w_ヘッダー２ = "管理単位：" + controlRecord["管理単位ＩＤ"]["value"][0] + " (" + controlRecord["管理単位名"]["value"][0] + ") "
w_ヘッダー３ = w_ヘッダー１ + "　□　" + w_ヘッダー２
worksheet[0][0].change_contents(w_ヘッダー３, worksheet[0][0].formula)
```

←ワークシートの0-0は1-Aを指す

```
worksheet[row][1].change_contents(detailRecord["システム機能ＩＤ"]["value"][num], worksheet[row][1].formula)
worksheet[row][2].change_contents(detailRecord["システム機能名称"]["value"][num], worksheet[row][2].formula)
worksheet[row][3].change_contents(detailRecord["ｒｅｓｔＩＤ"]["value"][num], worksheet[row][3].formula)
worksheet[row][4].change_contents(detailRecord["ｒｅｓｔ名"]["value"][num]
worksheet[row][5].change_contents(detailRecord["概要"]["value"][num],
worksheet[row][6].change_contents(detailRecord["接続先システム"]["value"][r
worksheet[row][7].change_contents(detailRecord["処理タイミング"]["value"][r
worksheet[row][8].change_contents(detailRecord["備考"]["value"][num],

    end

    $Vlog.debug("DG_620R_view") {"createPrintOfDG_620R end"}
end
```



1.エクセルシートへ**タイトルを表示する位置を指定**します。
　シートのA列1行は[0][0]で指示する。（セルの位置）

- ビューRuby

③-2テンプレートへデータを作成する

```
# ＲＥＳＴ一覧を作成する↓
def createPrintOfDG_620R(workbook, subsystemRecord, controlRecord, detailRecord)↓
    $Vlog.debug("DG_620R_view") {"createPrintOfDG_620R start"}↓
    ↓
    worksheet = workbook["REST一覧"]↓
    maxSize = detailRecord["入出力要素ＮＯ"]["value"].length↓
    ↓
    # ヘッダーのサブシステム・管理単位を設定↓
    w_ヘッダー１ = "サブシステム：" + subsystemRecord["サブシステムＩＤ"]["value"][0] + "（" + subsystemRecord["サブシステム名称"]["value"][0] + "）"↓
    w_ヘッダー２ = "管理単位：　　" + controlRecord["管理単位ＩＤ"]["value"][0] + "（" + controlRecord["管理単位名"]["value"][0] + "）"↓
    w_ヘッダー３ = w_ヘッダー１ + "　　" + w_ヘッダー２↓
    worksheet[0][0].change_contents(w_ヘッダー３, worksheet[0][0].formula)↓
    ↓
    # 明細を設定する↓
    # 52行を超える時、行を追加する↓
    if maxSize > 52↓
        addSize = maxSize - 52↓
        for j in 1..addSize do↓
            worksheet.insert_row(52)↓
        end↓
    end↓
    #データセットの開始行とデータの終了行を設定
    maxSize = maxSize - 1↓
    for num in 0..maxSize do↓
        row = 3 + num↓
        ↓
        lineNo = row - 2↓
        worksheet[row][0].change_contents(lineNo,
        worksheet[row][1].change_contents(detailR
        worksheet[row][2].change_contents(detailR
        worksheet[row][3].change_contents(detailR
        worksheet[row][4].change_contents(detailR
        worksheet[row][5].change_contents(detailR
        worksheet[row][6].change_contents(detailR
        worksheet[row][7].change_contents(detailR
        worksheet[row][8].change_contents(detailR
        ↓
    end↓
    ↓
    $Vlog.debug("DG_620R_view") {"createPrintOfDG_620R end"}↓
end↓
```

```
# 明細を設定する↓
# 52行を超える時、行を追加する↓
if maxSize > 52↓
    addSize = maxSize - 52↓
    for j in 1..addSize do↓
        worksheet.insert_row(52)↓
    end↓
end↓
↓
```

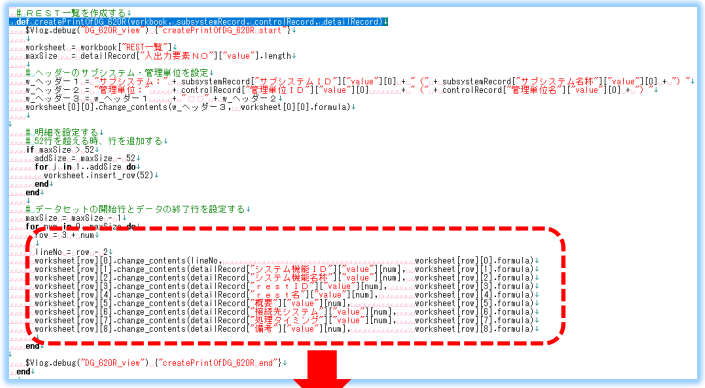
←ワークシートの最大行数を指示52行目にデータが入ると、
次は2枚目の1行目からデータを入れる。行数を間違えると入れる場所がなく、エラーになる。

2.エクセルシートの**MAX行数を設定**する
（エクセルシートを何行まで表示するのかを指示します。）

図の場合は52行目までを1枚に印刷します。
52行以上になると次のページへデータを回し印刷します。

ビューRuby

③-3テンプレートへデータを作成する



3.**データをどのセルへ出力するか設定**する
トランの**レスポンスと同じ。**
データの出力先をココで指示します。

［row］［数字］で横列のセルの位置を指定します。
※0,1,2,3,4,5＝3-A,3-B,3-C,3-D,3-E,,,

［項目名］［Value］でセルへ入れる物を指示する。

```
#データセットの開始行とデータの終了行を設定する
maxSize = maxSize - 1
for num in 0..maxSize do
  row = 3 + num
  ↓ワークシートのどこにレスポンスデータを入れるのかを指示する
  lineNo = row - 2
  worksheet[row][0].change_contents(lineNo,                                                    worksheet[row][0].formula)
  worksheet[row][1].change_contents(detailRecord["システム機能ＩＤ"]["value"][num],           worksheet[row][1].formula)
  worksheet[row][2].change_contents(detailRecord["システム機能名称"]["value"][num],           worksheet[row][2].formula)
  worksheet[row][3].change_contents(detailRecord["ｒｅｓｔＩＤ"]["value"][num],             worksheet[row][3].formula)
  worksheet[row][4].change_contents(detailRecord["ｒｅｓｔ名"]["value"][num],               worksheet[row][4].formula)
  worksheet[row][5].change_contents(detailRecord["概要"]["value"][num],                     worksheet[row][5].formula)
  worksheet[row][6].change_contents(detailRecord["接続先システム"]["value"][num],           worksheet[row][6].formula)
  worksheet[row][7].change_contents(detailRecord["処理タイミング"]["value"][num],           worksheet[row][7].formula)
  worksheet[row][8].change_contents(detailRecord["備考"]["value"][num],                     worksheet[row][8].formula)
end
```

	A	B	C	D	E	F	G	H
1	サブシステム:XX(Lx)　管理単位:XX (99)							
2	No	システム機能		REST_ID	REST名	概要	接続元システム	処理タイミング	備考
3		機能ID	機能名						
4	1								
5	2								
6	3								

完成