Edit Script について

- Edition Flex の呼び出し時に、editScript パラメータを与える事で、スクリプトによるページの編集を可能としました。
- 処理タイミングは、差し込み処理後、編集画面を出す前です
- 言語仕様は、groovy で、そのまま、GroovyShellで実行します。
- 処理は、前処理と、ページ毎の処理を行いますが、スクリプトは全ページに同じものを適用します。
    - 例えば、ページ数が 2 の場合は、同じ editScript が、前処理とページごとで、合計 3 回呼ばれます。
    - ページ毎に処理を分けたい場合は、if(page.no == 1) {} の様に、条件判断で分けてください。　switch(page.no) { case 1: ; } というような記述も可能です。
    - 少なくとも前処理は処理を分けた方が良いです。前処理が不要な場合は、先頭に、if(page.no == 0) return; と入れてください。
- エラーが起きた場合は、ページごとにスクリプトの実行を停止して、エラーは無視します。　※但し、env.debug = true; とした場合は、停止してエラーを表示します。
- EditScriptは、テンプレートに同梱される場合と、APIのパラメータで指定される場合があります。
- テンプレート同梱のスクリプトは、APIのパラメータの enableIncludedEditScript が true の場合のみ実行されます。
- テンプレート同梱スクリプトと、パラメータ指定のスクリプトの両方が有効な場合は、パラメータ指定のスクリプトの後ろに、テンプレート同梱スクリプトを結合して実行します。

produce API 呼び出し時の Edit Script 利用に関する注意事項

- produce API呼び出し時は、原則として処理フローは同じですが、利用の仕方が異なります。
  - produce API のEditScriptでは、原則として、前処理のみを利用して、前処理中に、setCurrentPage() でカレントページを切り替えながら、renderPage()で、カレントページの内容を（動的に）PDFのページとして追加出力していくというフローになります。
    - そのために、produce API では、前処理完了時点でEditScript処理を中断する為に、"exit=true" を、与えられた EditScriptの先頭に自動挿入します。
      - produce API で、ページごとの処理が必要な場合は、exit=false を EditScript中に記載して、前処理中に exit が false になるようにしてください。
  - 原則としてページごとの処理はせず、前処理だけですので、if(page.no == 1) のようなページ判定処理の記述は不要となります。
  - 前処理での処理となりますので、getPartsByLabel() などのページ上のパーツを操作するような関数呼び出しをする場合には、setCurrentPage(ページ番号)を呼び出して、処理対象のページをカレントにする必要があります。（一時的にページ処理状態に切り替えます。このとき、前処理用の関数は使えなくなります。）
    - setCurrentPage(ページ番号) 呼び出し後は、指定したページ番号のページを操作することができますが、逆に前処理だけでしか使えない関数呼び出しはできなくなります。前処理の関数を呼び出したい場合は、setCurrentPage(0) を呼び出して前処理の状態に戻してください。
    - 一度も renderPage() が呼ばれない場合は生成されるものが無いので、no page 等のエラーとなります。
  - EditScriptを省略した場合は、自動で全ページを生成するEditScriptが設定されます。

```
exit = true;
        for(int p = 1; p <= ${maxPage}; p++) {
        renderPage(p);
        }
```

※例えばテンプレートとなるページに、順次データを与えて差し込み処理をしながらマルチページのPDFを生成するような場合、概ね次のようなフローとなります。

```
//ページを呼び出す。
        setCurrentPage(1);

        //CSVの先頭行をラベル名と見なす
        def labelList = getRecordTitle();

        //データで回す（データはCSVファイルで指定）
        for(def i=1; i <= getRecordCount(); i++) {
        def record = getRecord(i); //CSVから１行取得
        labelList.each { label ->
        def p = getPartsByLabel(label);
        if(p) {
        // 該当ラベルのパーツがあれば、
        p.injectionData(record[label]); // レコードの該当データを差し込む
        }
        }
        //差し込みが終わったら、PDFページを出力する
        renderPage();

        // ※この時点で、カレントのページは差し込み処理で変更されていますが、保存はされていません。

        //もし1レコード目の差し込みの状態を2レコード目の差し込み時にリセットしたい場合は、更に次の呼び出しをする
        setCurrentPage(1, true); //disposePreviousを true にすることで、1レコード目の差し込みを保存せずに再度1ページ目を呼び出す（＝リセット）
        }
```

@import : ライブラリのインポート

- よく使う処理や関数は、EditScriptの断片として、Edition FlexのEditScriptライブラリに登録して利用することができます。
- 登録されたをEditScriptの断片は、 @import 文で、読み込む事ができます。
- "@import" + 半角スペース + "登録名"：この行を見つけると、その場所に、登録されたEditScriptの断片を挿入してから EditScriptが実行されます。

前処理

- ページごとの処理の前に、前処理を行います。ページ数に関わらず必ず最初に１回実行されます。
- 通常は、 if(page.no == 0) { 前処理 } else { ページごとの処理}; や、switch 文で処理を分岐してください。
- この時、page.no は0 になります。また、page.no も０になりますが、page のその他の値はダミーの値です
- 特定のページを対象とした処理ではありませんので、ページやパーツに対する処理はすべて、原則としてエラーとなりますが、なるべくエラーを出さないように、空の処理（なにもしない処理）を行うようにします。

全体で利用可能なメソッド

- デバッグ用ログ出力　：　trace(String message)
    - サーバー上のログ（editscriptDebug.log）に、文字列を出力する。（日時も出力されます）
- URLエンコード：URLEncode(String str)
    - str を URLエンコードします。
- 差し込みデータMapの取得　：　getInjectionDataMap()
    - JSON形式で injectionData が、layout or wizard API 呼び出しのパラメータ指定されている場合のみ利用可能
    - 指定された injectionData を Map に変換したものを取得して、injectionData を参照します
- 差し込みデータの更新　：　setInjectionData(String label, value)
    - JSON形式で injectionData が、layout or wizard API 呼び出しのパラメータ指定されている場合のみ利用可能
    - 指定された injectionData を 更新します。 injectionData は label 対 値（値はMapの場合もあります）の構成なので、指定 labelの値を value にセットします。
        - value が、String の場合は、テキストパーツの文字、画像パーツのtrackingId になります。
        - パーツの属性等も指定したい場合は、value を Map にして指定します。 ["value":流し込む値, "font":"MSゴシック"] など
            - injectionData についての詳細は、新 injectionData(JSON)（差し込み設定パラメータ）についてを参照してください。
- ラベルモード変更　：　setLabelMode(String mode) *Ver.3.0
    - getPartsByLabel , getPartsSetByLabel のラベル検索モードを変更します。
    - mode
        - "full"(デフォルト)
            - 完全一致で検索します。
        - "path"
            - 階層構造に従って検索します。
            - 先頭に / を付けます。
            - 末尾に / を付けません。

```
例）
        * パーツ
        str1
        str2
        3333
        group1(グループパーツ)
        str4(グループパーツ内のパーツ)
        str5(グループパーツ内のパーツ)
        6666(グループパーツ内のパーツ)
        * 検索値
        /group1/str5
        * 検索結果
        str5が見つかります。
```

- "regex"
  - 正規表現(ECMAScript準拠)で検索します。

```
例）
        * パーツ
        str1
        str2
        3333
        group1(グループパーツ)
        str4(グループパーツ内のパーツ)
        str5(グループパーツ内のパーツ)
        6666(グループパーツ内のパーツ)
        * 検索値
        str.*
        * 検索結果
        str1,str2が見つかります。
```

- "pathRegex"
  - 正規表現(ECMAScript準拠)で検索します。
  - 階層構造に従って検索します。
  - 先頭に / を付けます。
  - 末尾に / を付けません。

```
例）
        * パーツ
        str1
        str2
        3333
        group1(グループパーツ)
        str4(グループパーツ内のパーツ)
        str5(グループパーツ内のパーツ)
        6666(グループパーツ内のパーツ)
        * 検索値
        /str.*
        * 検索結果
        str1,str2が見つかります。
```

```
例）
        * パーツ
        str1
        str2
        3333
        group1(グループパーツ)
        str4(グループパーツ内のパーツ)
        str5(グループパーツ内のパーツ)
        6666(グループパーツ内のパーツ)
        * 検索値
        /group1/str.*
        * 検索結果
        str4,str5が見つかります。
```

前処理で、参照（及び設定）可能な変数

- exit：true を処理中に設定すると、ページごとの処理を行わない。
    - 全てを前処理で行ってしまう場合に、exit = true として前処理を終われば、ページごとの処理は行いません。
- env：Map：環境変数
    - 初期値は、['debug':false] です。任意の値を設定し、ページごとの処理で参照します
    - 前処理からページごとの処理に渡したい値、ページをまたがる値が必要な場合にスクリプト中で設定・参照します。
    - スクリプト中（通常先頭行）で、 env.debug = true とした場合、スクリプト実行時エラーがあった時に、処理を中断してエラーを表示します。
- pageCount：int
    - ページ数

前処理で利用可能なメソッド

- ページの新規追加　： void addPage([pageName [,width = 100,[height = 100, [marginTop = 0, [marginBottom = 0, [marginLeft = 0, [marginRight = 0]]]]]])
    - pageName：String：ページ名
    - width：double：幅(mm)（省略時 100）
    - height：double：高さ(mm)（省略時 100）
    - marginTop: double：上余白(mm)（省略時 0）
    - marginBottom: double：下余白(mm)（省略時 0）
    - marginLeft: double：左余白(mm)（省略時 0）
    - marginRight: double：右余白(mm)（省略時 0）

- ページのコピー　： void copyPage([pageNo = 1][,trackingId = null])
    - pageNo: int：コピーするページ番号（省略時１）
    - trackingId: String：コピー元のテンプレートの trackingId（省略時 null）
    - コピーされたページは、ページ一覧の最後に追加します。
    - trackingId が指定されていない場合は、現在編集中のテンプレート（自分自身）のページをコピーします
    - trackingId が指定されていた場合は、テンプレートプロバイダの trackingId で指定されるテンプレートのページをコピーします。
    - pageNo が、0 の場合は、全ページをコピーします。

- ページの削除：void removePage([pageNo = 1])
    - pageNo: int：削除するページ番号（省略時１）

- データセット情報の取得 Map getDatasetInfo(injectionDataSetName)
    - injectionDataSetName：String 差し込みデータセット名（名前で検索し、同名のものがあれば最初に見つかった方）
        - 戻り値 Map
            - trackingId：String：データセットのtrackingId
            - vpath：String：データセットが存在するフォルダパス
            - tags：String[]：データセットに設定されたタグ
            - dispname：String：データセット名
            - lastModified：Date：最終更新日
            - dataCount：int：データ数
            - label：String[]：ラベル一覧

- データセットのデータの取得　Map findData(injectionDataSetName, keyLabel, keyValue [,valueLabel])
    - injectionDataSetName：String 差し込みデータセット名（名前で検索し、同名のものがあれば最初に見つかった方）
        - keyLabel：String：検索対象のキー
        - keyValue：String：keyLabel の値。（keyLabelで指定した項目の値が keyValueで指定した値と一致する最初のレコード（行）を取得)
        - valueLabel（省略可能）：String：指定した場合は検索結果はレコードではなく、その中の指定ラベルの値（文字列）となる

- データセットのデータの取得（シーケンス番号で取得）　Map getData(injectionDataSetName, seqNo [,valueLabel])
    - injectionDataSetName：String 差し込みデータセット名（名前で検索し、同名のものがあれば最初に見つかった方）
        - seqNo：int: データセットのデータの、１～レコード数の連番　※どう扱うかは差し込みデータプロバイダ次第
        - valueLabel（省略可能）：String：指定した場合は検索結果はレコードではなく、その中の指定ラベルの値（文字列）となる

- レコードセットの項目名リストの取得　Map getRecordTitle()
    - CSVの項目名（String）のリストを取得する

- レコードセットのレコード数の取得 int getRecordCount()
    - CSVの行数（ヘッダー行を除く）を取得する

- レコードセットのデータの取得（レコード番号で取得）　Map getRecord(recordNo)
    - produce API 呼び出し時に、recordCSVで与えたCSVファイルをレコードセットとして、その１行をレコードとして読み込む
        - このAPI自体は、CSVである事を前提としていないので、何らかのリストから指定番目のデータを取り出す処理を行っている。
        - 将来的に、EditScript実行時にrecordSet として与えるデータを、CSV以外のものにすることも可能で、例えば、Edition CMSのデータを事前に一括読み込みしてレコードセットとして使う事で、シーケンス単位に読み込むより高速化される。
    - recordNo：int: レコードセットのデータの、１～レコード数の連番

- 差し込み処理 injectionData(pageNo, data)
    - 指定ページに対して、injectionData 指定での差し込み相当の差し込み処理を行います
    - カセットのtrackingIdによる入れ替えも行います（ただしネストはしません）
    - data: Map：差し込みデータ（ラベルと差し込み値のペア）
    - 戻り値：boolean：差し込み処理が行われて何らかの変化があれば、true

- URLを指定して画像をダウンロード Map wgetImage(url[, filter [,pdfVersions]]);
    - （編集画面のURL指定画像アップロードに相当）指定されたURLから画像をダウンロードし、ドキュメントに取り込みます。
    - url：String：画像のURL ex. "http://...."
    - filter：String："jpegConvertToCMYK" もしくは "jpegConvertToGrayScale" URLがJPEG画像の場合、CMYK変換、あるいはグレースケール変換をする
    - pdfVersions：String：PDFのバージョン番号をカンマ区切りで指定する。 ex. "1.3,1.4" ※指定した場合指定以外のバージョンではEditScriptがエラーで終了します。

- カセットの取得　parts getCassetteInfo([trackingId = null [,pageNo = 1 [,layer = 0 [,withMargin = false [,enableIncludedEditScript = false [,editScript = null]]]]]])
    - trackingId: String：カセットの識別コード（テンプレート、ドキュメントも指定可能）　null の場合は編集中のドキュメントのページ
    - pageNo：追加するカセットが複数ページで構成されている場合に追加するページ（通常はカセットは１ページ構成ですが、テンプレートをカセットと見立てて追加するような場合に、ページ番号が指定できるようにしました。）
    - layer：指定したレイヤーだけ取得する（0の場合は全レイヤー） ※最新版では マイナス値の場合に指定レイヤー以外
    - withMargin：余白含めてカセットとする（通常はfalse）
    - enableIncludedEditScript：取得時にカセットに同梱された EditScriptをカセットに対して実行する（page.no == 0 と page.no == 1 の処理が走ります）
    - editScript：追加時にカセットに対して実行するスクリプト。カセット同梱スクリプトが実行される場合は、その前に結合されて実行されます。
    - 戻り値： 取得したカセット情報（cassetteInfo）
    - ※カセットとテンプレートは構造が同じである為、カセットプロバイダによっては、テンプレート（のいずれかのページ）をカセットとして取得することができます。（テンプレートプロバイダ次第）

- 現在のページの選択 void setCurrentPage([int pageNo = 0, [boolean disposePrevious = false]])
    - 前処理中に現在の対象ページを選択して、ページごとの処理で利用可能なメソッドをそのページに対して利用可能とします。
        - 現在のページが設定されている状態では、setCurrentPage() 以外の前処理でのみ利用可能なメソッドは利用不可となります。
    - pageNo：現在のページとして設定するページ番号
        - pageNo = 0 の場合は、前処理状態に戻り、前処理でのみ利用可能なメソッドが利用可能となり、ページごとの処理でのみ利用可能なメソッドは利用不可となります。
        - カレントのページが、pageNo で指定されたページと同じ場合は、disposePreviousに関わらずカレントページの状態は破棄されて読み込み直しとなる。カレントのページと pageNo で指定されたページが異なる場合は、カレントのページは保存される。（disposePreviousがtrueの場合は保存されない）
    - disposePrevious：true の場合は、呼びだされた時点でカレントになっているページを保存しない。

- ページ出力（product API のみ） void renderPage([pageNo = 0], [int rotate = 0])
    - product API 呼び出しの場合に、指定ページ（あるいはカレントページ）を出力します。
        - renderPageは、ベースシートを無視して、ページをそのまま出力します。renderNextPage呼び出しは不要です。呼出し後に renderAddSheetは呼べます。
    - pageNo：保存するページ番号（無指定あるいは0の場合は、現在のページを出力します。現在のページが無い場合は何も出力しません。）
    - rotate：回転角度 0/90/180/270 ページを回転させる

- ベースシート追加（product API のみ） int renderAddBase(void)
    - product API 呼び出しの場合に、カレントページをベースシート（PDF生成時の各ページの最背面固定部）として追加する
    - 戻り値は追加したベースシートのインデックスで、追加した順に、1からの連番になる。　追加エラー時は 0
    - 追加したシートがカレントのベースシートとなり、次の renderNextPage で背景として使われる

- カレントページ出力の確定と次ページ準備（product API のみ） void renderNextPage(int basePageIndex = 0, [int rotate = 0])
    - product API 呼び出しの場合に、現在出力中のページを確定（サンプルや透かしの描画はこのタイミングで行われます）して、次のページの準備を行う。ベースシートが設定されていれば、それが出力された状態となる
    - basePageIndex：ベースシートのインデックス　※0を指定した場合はカレントのまま変えない
    - rotate：回転角度 0/90/180/270 次のページを回転させる

- シート（ベースシートに重ねるページ）出力（product API のみ） void renderAddSheet(void)
    - product API 呼び出しの場合に、カレントページをベースページに重ねます出力します。

ページごとの処理

ページごとの処理で、参照（及び設定）可能な変数

- exit：true を処理中に設定すると、以降のページの処理を行わない。
    - EditScriptは、存在する全ページについて各１回呼ばれる事が前提ですが、処理すべきページが最初の数ページのみで、ページ数が多い場合などには、必要なページの処理が終わった時点で、exit = true　とすれば、以降のページ処理が行われません。
- env：Map：環境変数
    - 初期値は、['debug':false] です。任意の値を設定し、別のページの処理で参照します
    - ページをまたがる値が必要な場合にスクリプト中で設定・参照します。
    - 例えば、１ページ目のテキストパーツの「会社名」を、２ページ目以降の同じラベルのパーツに設定するなど
    - スクリプト中（通常先頭行）で、 env.debug = true とした場合、スクリプト実行時エラーがあった時に、処理を中断してエラーを表示します。

- pageCount：int：ページ数
- pageNo：int：ページ番号（参照のみ） ※ page.no と同じ

- page：Map
    - no：int：ページ番号（参照のみ）
    - name：String：ページ名（参照と設定）
    - size：Map：ページサイズ（参照と設定）
        - totalWidth：double：余白込みの幅
        - totalHeight：double：余白込の高さ
        - width：double：余白無しの幅
        - height：double：余白無しの高さ
        - marginTop：double：上余白
        - marginLeft：double：左余白
        - marginBottom：double：下余白
        - marginRight：double：右余白
    - delete：boolean：スクリプト中で、delete = true と設定された場合は、そのページを削除する

ページごとの処理で、利用可能な、メソッド

- URLを指定して画像をダウンロード Map wgetImage(url[, filter [,pdfVersions]]);
    - 前処理で利用可能なメソッドと同じです。詳細は前処理の方を参照してください。

- LayoutLogicシートの取得 getLayoutSheet() *Ver.5.0
    - 生のLayoutLogic シートを返します。使用注意ですが、EditScriptで関数化されていない低レベルのAPIを利用するときに使います。
        - LayoutLogicに関する理解と知識が必要ですが、仕様を公開していませんので、これを利用する場合は、原則、レゾロジックに開発依頼が必要です。
    - 戻り値： シート

- 全パーツの取得 List<parts> getPartsAll(typeId, parentParts) *Ver.3.0
    - typeId：null 以外を指定した場合は、指定したタイプのパーツのみを取得します。
    - parentParts：parts：親となるグループパーツ（カセット）を指定します。 省略時は null で、親なし（ページ直下）のパーツセット取得
    - 戻り値： パーツのList（指定したパーツが無い場合は空のリスト、１つだけの場合は要素１つのリスト）
    - ラベルが無いパーツも取得します

- パーツセットの取得 List<parts> getPartsSetByLabel(label [, parentParts = null])
    - label：String：ラベル文字
    - parentParts：parts：親となるグループパーツ（カセット）を指定します。 省略時は null で、親なし（ページ直下）のパーツセット取得
        - パーツセットの取得は、parentParts で指定されたグループ内のパーツセットを取得します。parentParts が null の場合は、ページ直下のパーツセットを取得します。
    - 戻り値： パーツのList（指定したパーツが無い場合は空のリスト、１つだけの場合は要素１つのリスト）

- パーツの取得　parts getPartsByLabel(label [,index = 1, [parentParts = null]])
    - label：String：ラベル文字
    - index：int：同一ラベルのパーツが複数ある場合に、１～の番号でパーツを指定する。（省略時１）
    - parentParts：parts：親となるグループパーツ（カセット）を指定します。 省略時は null で、親なし（ページ直下）のパーツ
        - パーツの取得は、parentParts で指定されたグループ内のパーツを取得します。parentParts が null の場合は、ページ直下のパーツを取得します。
    - 戻り値： パーツ

- パーツの削除　void deletePartsByLabel(label [,index = 1])
    - label：String：ラベル文字
    - index：int：同一ラベルのパーツが複数ある場合に、１～の番号でパーツを指定する。（省略時１）
    - 戻り値：無し

- パーツの削除（直接） void deleteParts(parts) *Ver.3.0
    - parts：削除するパーツ

- グルーピング：parts groupingParts(partsList) *Ver.3.0
    - partsList：グルーピングするパーツのリスト
    - 戻り値： グループパーツ（getPartsByLabel()で取得したパーツと同等に使えます）

- グループ解除： List<parts>ungroupingParts(parts) *Ver.3.0
    - parts：グループ解除するパーツ
    - 戻り値： グループに含まれていた子パーツのリスト

- 一番上の図形でマスク：parts maskingParts(partsList) *Ver.5.0
    - partsList：マスクするパーツのリスト（一番上がマスク図形、マスク図形以外のパーツはマスク図形と重なりを持つ必要がある）
    - 戻り値： グループパーツ（getPartsByLabel()で取得したパーツと同等に使えます）

- グループ解除： List<parts>unmaskingParts(parts) *Ver.5.0
    - parts：マスク解除するパーツ
    - 戻り値： マスクパーツに含まれていた子パーツのリスト（マスク図形も含む）

- パーツの追加　parts addParts(typeId [,layer = 1])
    - typeId：String：パーツのタイプID（parts.laml に記載がある、”矩形”などの日本語表記も利用可能）
    - layer：追加するレイヤー（１～）　省略時は１
    - 戻り値： パーツ
    - パーツ追加時の位置に関する注意
        - パーツ追加直後は、位置計算がされず、transform情報がありません。この状態ではパラメータの、x,y,width,height,cx,cy などで位置を指定します。（回転やスケール変更はできません）
            - パーツの、x,y,width,height,cx,cyなどの情報はこの段階でしか設定できません。一度 transform が計算されてしまいますと、以降は、サイズ変更はスケール（拡大縮小）になってしまいますので、幅や高さはこの段階で指定するようにしてください。
        - 追加後、getBoundBox()を呼び出すとその時点のパラメータに基づいて位置計算が行われ、transform が計算されます。　以降は、transformで位置指定を行います。

- カセットの追加　parts addCassette(trackingId [,layer = 1 [,enableIncludedEditScript = false [,editScript = null [,pageNo = 1 [,withMargin = false]]]]])
    - trackingId: String：カセットの識別コード
    - layer：追加するレイヤー（１～）　省略時は１
    - enableIncludedEditScript：追加時にカセットに同梱された EditScriptをカセットに対して実行する（page.no == 0 と page.no == 1 の処理が走ります）
    - editScript：追加時にカセットに対して実行するスクリプト。カセット同梱スクリプトが実行される場合は、その前に結合されて実行されます。
    - pageNo：追加するカセットが複数ページで構成されている場合に追加するページ（通常はカセットは１ページ構成ですが、テンプレートをカセットと見立てて追加するような場合に、ページ番号が指定できるようにしました。）
    - withMargin：余白含めてカセットとする（通常はfalse）
    - 戻り値： 追加したカセット
    - ※位置　カセットの場合は、追加時に位置計算されます。初期位置は、(0,0) で、追加後に、transform で位置指定をしてください
    - ※カセットとテンプレートは構造が同じである為、カセットプロバイダによっては、テンプレート（のいずれかのページ）をカセットとして追加することができます。（backstageはtrackingIdで指定した場合にテンプレートとカセットを区別なく返します）

- カセットの入れ替え parts replaceCassette(cassette, trackingId [,enableIncludedEditScript = false [,editScript = null [,pageNo = 1 [,withMargin = false]]]])
    - cassette: parts：入れ替え元のカセット
    - trackingId: String：入れ替えるカセットの識別コード
    - enableIncludedEditScript：追加時にカセットに同梱された EditScriptをカセットに対して実行する（page.no == 0 と page.no == 1 の処理が走ります）
    - editScript：追加時にカセットに対して実行するスクリプト。カセット同梱スクリプトが実行される場合は、その前に結合されて実行されます。
    - pageNo：追加するカセットが複数ページで構成されている場合に追加するページ（通常はカセットは１ページ構成ですが、テンプレートをカセットと見立てて追加するような場合に、ページ番号が指定できるようにしました。）
    - withMargin：余白含めてカセットとする（通常はfalse）
    - 戻り値： 入れ替えたカセット
    - ※カセットとテンプレートは構造が同じである為、カセットプロバイダによっては、テンプレート（のいずれかのページ）をカセットとして追加することができます。（backstageはtrackingIdで指定した場合にテンプレートとカセットを区別なく返します）

```
注意）
        追加したばかりのパーツは、位置もパラメータも何も指定されていませんので、
        パーツを追加する際は、addParts() の戻り値を変数へ格納し、位置やパラメータを設定してください。
        位置指定は、transform ではなく、param.x, param.y, param.width, param.height で指定してください。
```

- データセット情報の取得 Map getDatasetInfo(injectionDataSetName)
    - injectionDataSetName：String 差し込みデータセット名（名前で検索し、同名のものがあれば最初に見つかった方）
        - 戻り値 Map
            - trackingId：String：データセットのtrackingId
            - vpath：String：データセットが存在するフォルダパス
            - tags：String[]：データセットに設定されたタグ
            - dispname：String：データセット名
            - lastModified：Date：最終更新日
            - dataCount：int：データ数
            - label：String[]：ラベル一覧

- データセットのデータの取得　Map findData(injectionDataSetName, keyLabel, keyValue [,valueLabel])
    - injectionDataSetName：String 差し込みデータセット名（名前で検索し、同名のものがあれば最初に見つかった方）
        - keyLabel：String：検索対象のキー
        - keyValue：String：keyLabel の値。（keyLabelで指定した項目の値が keyValueで指定した値と一致する最初のレコード（行）を取得)
        - valueLabel（省略可能）：String：指定した場合は検索結果はレコードではなく、その中の指定ラベルの値（文字列）となる

- データセットのデータの取得（シーケンス番号で取得）　Map getData(injectionDataSetName, seqNo [,valueLabel])
    - injectionDataSetName：String 差し込みデータセット名（名前で検索し、同名のものがあれば最初に見つかった方）
        - seqNo：int: データセットのデータの、１～レコード数の連番　※どう扱うかは差し込みデータプロバイダ次第
        - valueLabel（省略可能）：String：指定した場合は検索結果はレコードではなく、その中の指定ラベルの値（文字列）となる

- カセットの取得　parts getCassetteInfo([trackingId = null [,pageNo = 1 [,layer = 0 [,withMargin = false [,enableIncludedEditScript = false [,editScript = null]]]]]])
    - trackingId: String：カセットの識別コード（テンプレート、ドキュメントも指定可能）　nullの場合は編集中のドキュメントのページ
    - pageNo：追加するカセットが複数ページで構成されている場合に追加するページ（通常はカセットは１ページ構成ですが、テンプレートをカセットと見立てて追加するような場合に、ページ番号が指定できるようにしました。）
    - layer：指定したレイヤーだけ取得する（0の場合は全レイヤー）
    - withMargin：余白含めてカセットとする（通常はfalse）
    - enableIncludedEditScript：取得時にカセットに同梱された EditScriptをカセットに対して実行する（page.no == 0 と page.no == 1 の処理が走ります）
    - editScript：追加時にカセットに対して実行するスクリプト。カセット同梱スクリプトが実行される場合は、その前に結合されて実行されます。
    - 戻り値： 取得したカセット情報（cassetteInfo）
    - ※カセットとテンプレートは構造が同じである為、カセットプロバイダによっては、テンプレート（のいずれかのページ）をカセットとして取得することができます。（テンプレートプロバイダ次第）

- ページ出力（product API のみ） void renderPage([pageNo = 0])
    - product API 呼び出しの場合に、指定ページ（あるいはカレントページ）を出力します。
    - pageNo：保存するページ番号（無指定あるいは0の場合は、現在のページを出力します。現在のページが無い場合は何も出力しません。）

- （カセット情報で）カセットの追加 addCassetteByInfo(cassetteInfo [,layer = 1])
    - getCassetteInfo() で取得したカセット情報を使ってカセットをページに追加する
    - trackingId指定ではなく、取得済みのカセット情報を使うという部分以外は、addCassette() と同じです。
    - カセット情報は一度取得すれば何度でも使えEditScriptも取得時に実行されるだけなので、面付け処理の効率化が図れます。

- （カセット情報で）カセットの入れ替え replaceCassetteByInfo(cassette, cassetteInfo, adjustSize = true)
    - getCassetteInfo() で取得したカセット情報を使ってカセットを入れ替えます
    - trackingId指定ではなく、取得済みのカセット情報を使うという部分以外は、replaceCassette() と同じです。
    - adjustSize：サイズ調整：通常は省略もしくは true にしてください。 falseを指定すると、サイズ調整は行われなくなり、位置とラベルのみ維持になります。（高速化の為）

- レイヤー数を取得する getLayerCount(void)
- レイヤーを（一番上に）追加する addLayer(void)
- 指定レイヤーを削除する removeLayer(layer = 0)
    - layer： １～のレイヤー番号、0 だと一番上のレイヤー　※レイヤーは再背面が1です
- 指定されたレイヤーを（編集画面の初期）編集対象レイヤーにする　selectLayer(layer = 1)
    - layer： １～のレイヤー番号 ※EditScript処理中は影響ありません
- （編集画面の初期）編集対象レイヤーを取得する　getSelectedLayer(void
    - 戻り値：レイヤー番号　※存在しない場合は 0
- レイヤーの表示・非表示を切り替える　showLayer(layer = 1, bShow = true)
    - layer： １～のレイヤー番号
    - bShow ： true:表示 false:非表示

- レイヤーの表示・非表示状態を取得する　isShowLayer(layer = 1)
    - layer： 1～のレイヤー番号

- シート（ページ）のパラメータを取得する String getSheetParamValue(String key)
    - key：パラメータのキー
        - グリッド関係パラメータ key：grid_show / grid_type / grid_adsorb / grid_vert / grid_horz / grid_snap / grid_x / grid_y
        - 補助線関係パラメータ key：guide_show / guide_snap / guide_lock / guide_adsorb
    - 戻り値：String keyに対する値
- シート（ページ）のパラメータの設定する boolean setSheetParamValue(String key, String value)
    - key：パラメータのキー
    - value：パラメータの値 (bool 値は、 true = "1" / false = "0" になります。
    - 戻り値：boolean 成否

パーツのプロパティ

- パーツは次のプロパティを持ちます。
    - partsId：パーツID（タイプID＋サブID）
    - typeId：パーツのタイプID
    - subId：パーツのサブID
    - display：表示状態 none: 非表示 / inline:表示
    - logic：logic要素
        - logic.mode：String：許可属性文字列
        - logic.label：String：ラベル文字列
    - transform：基準点の位置・拡大/縮小・基準点周りの回転
        - translateX：double：基準点の横位置（ページ左が0で mm 単位）
        - translateY：double：基準点の縦位置（ページ上が0で mm 単位）
        - rotate：double：回転角度（0～360度）
        - scaleX：double：X軸拡大率（1.0 = 100%）
        - scaleY：double：Y軸拡大率（1.0 = 100%）
    - param：パーツのパラメータ
        - パーツの各種パラメータ（パーツタイプごとに定義されています）　取得時の型は、parts.laml に従って判断します
        - param.xxxxx の、 xxxxx は、本来のパラメータ名に加えて、parts.laml で定義されている代理名も使えます。例： parts.param.文字列 = "これはテキスト"
    - boundBox：パーツのバウンダリボックスの位置とサイズ（パーツを囲む最小矩形サイズ） ※ 参照のみ　getBoundBox(true, true) と同じ
        - x：double：バウンダリボックスの左上のX座標
        - y：double：バウンダリボックスの左上のY座標
        - width：double：バウンダリボックスの幅
        - height：double：バウンダリボックスの高さ

パーツのメソッド

- 差し込み処理 parts.injectionData(data, recalcParts = false)
    - 対象のパーツに対して、injectionData 指定での差し込み相当の差し込み処理を行います
    - ただし、カセットの入れ替え処理は行いません。
    - カセットの入れ替えは、replaceCassette() を呼び出して下さい
    - data: Map or String：差し込みデータ（対象パーツが text , group の場合はMapが指定できます）
        - 対象パーツが image の場合にも、Map を指定できますが、原則として wgetImage() の戻り値を指定します。 path / name / version といったドキュメントに取り込んだ画像を示す値が必要です。
        - image パーツへの trackingId での画像の差し込みは、 data として、 String trackingId を指定してください。
    - recalcParts：boolean：デフォルトは false：true を指定すると、差し込み後にパーツの矩形領域を再計算する。通常は、parts#getBoundBox() で再計算させるので、false で良いが、即座に再計算させたいときに trueとする。
    - 戻り値：boolean：差し込み処理が行われて何らかの変化があれば、true
- 表示制御 parts.setDisplay(String state)
    - state："none" の場合はそのパーツを非表示にします。 "inline" の場合は、そのパーツを表示にします。
- 基準点設定 parts.editReferencePoint(String referencePoint, boolean keepReferencePointPosition = false)
    - referencePoint：基準点パラメータ文字列 center-center など
    - keepReferencePointPosition：trueの場合は、基準点の位置を変更前の位置に移動する。（パーツを移動せずに基準点を移動させたい場合はfalse）
- バウンドボックスの取得： getBoundBox(boolean bTranslate = true, recalcParts = true)
    - bTranslate：true の場合は、回転とスケールを考慮した矩形を取得。 false は、パーツ自体の矩形
    - recalcParts：取得前にパーツのアップデート処理を行う
    - このメソッドが呼ばれると、transform / param / logic が更新される

利用可能なユーティリティ

- format：差し込み用フォーマットユーティリティ
  - nvl(String value)：value が null なら "" そうでなければ value をそのまま返す
  - date(Object obj, String format)：obj を、format で日付フォーマットする。例： parts.param.text = format.date(new Date(), "yyyy/MM/dd");
  - comma：数値を3桁カンマ整形する。　例： format.comma(123456) ==> "123,456" になります。
  - jWeek：曜日を取得　日月火水木金土 の何れか
- StringUtils：文字列操作（Apache CommonsのStringUtils）
  - http://commons.apache.org/lang/api-2.5/org/apache/commons/lang/StringUtils.html
- LogicTable：テーブルパーツのデータ生成
- LAMLBuilder / GLAMLTool / LamlElement：LAML生成

例１：（パーツの削除とか属性の変更など）

```
        if(page.no == 1) {
        deletePartsByLabel("rect1", 2)
        def p1 = getPartsByLabel("rect1", 1)
        def p2 = getPartsByLabel("氏名")

        p1.logic.mode = "exd"
        p1.logic.label = "矩形です"
        p1.param.fill = "#84EF5C"
        p1.param.stroke = "#0000FF"
        p1.param.rx = p1.param.ry = 5

        p2.transform.translateX -= 30
        //      p2.transform.rotate = 5
        def p2p = p2.param;
        p2p.__extra__ += " Edition Flex"
        p2p.color = "blue"
        }


        例２：（パーツの追加や、ページ情報の参照）

        def p = addParts("矩形");
        p.param.width = 30;
        p.param.height = 50;
        p.param.x = (page.size.width - p.param.width) / 2.0;
        p.param.y = 50;
        p.param.fill = "#FF0000"

        def p2 = getPartsByLabel("氏名")
        p2.param.text += "_${page.no}:${page.name}";
```

データプロバイダにアクセスする例：

```
/* データプロバイダのデータは次のような感じに入っています。
        http://demo.reso.co.jp/backstage/ の、"design" ユーザが、editOnlyユーザのプロバイダに登録されています。
        ------------------------------------------------------------------
        商品番号, 商品, 金額, 単位, 商品画像
        ------------------------------------------------------------------
        1,ガーリックチキンフライ,"1,860",円,83cb6a0eac1082770c208cb572ca66ca
        2,炒飯,"2,230",ペソ,83ccdde2ac1082772df4bef73312a356
        3,天津五目麺,550,ドル,83ccdc2cac10827718b8b3dfe332fc9d
        4,沖縄ソーキそば,"99,800",マルク,83ccda38ac108277664e15e2536a0b9e
        5,卵豆腐,50,マルカ,83ccd883ac1082776f402cc3fef0ee0f
        6,ゆでダコ,0,ルーブル,83ccd670ac10827773d9d3b8f5fb0226
        ------------------------------------------------------------------
        */

        if(page.no == 1) {
        //プロバイダからデータ取得
        def record = findData("商品マスタ", "商品番号", "3");
        //レコードから"商品"というラベルの値（文字列）を取得する場合
        //def strData = findData("商品マスタ", "商品番号", "3", "商品");

        /*
        * プライスカード（グループパーツ）の中身のパーツに、
        * データプロバイダから取得したレコードデータを差し込む
        * その際、横コネクト編集を行う
        */

        //対象パーツ取得
        def pPrice = getPartsByLabel("プライス");
        //差込データの record['align'] を horizontal にすることで横コネクト編集
        record.align = "horizontal";
        //差し込み処理
        pPrice.injectionData(record);

        /*
        * 画像の差込
        */

        //対象パーツ取得
        def pImage = getPartsByLabel("商品画像");
        pImage.injectionData(record.商品画像);  //trackingIdがデータに入っている

        /*
        * 部分文字列差込
        */

        //対象パーツ取得
        def pSpec = getPartsByLabel("商品スペック");
        //テキストパーツにMAPデータを指定すれば、部分差込となる
        pSpec.injectionData(record);

        /*
        * テキストの差込
        */
        def pName = getPartsByLabel("商品");
        pName.injectionData(record.商品);
        }
```

- 指定ラベルのパーツを非表示とする関数例

```
def hidePartsByLabel(label, cassette = null) {
        def pList = getPartsSetByLabel(label, cassette);
        pList.each { parts ->
        parts.setDisplay("none");
        }
        }
```

RESOLOGIC|2020.04.22