JSPの実装

JSPで利用可能なタグリブとEL関数

- spring-form JSP Tag Library
- spring JSP Tag Library
  - spring:evalタグはガイドで示している用途以外では原則禁止 (個別画面での使用は禁止)
  - 次のタグは使用しない想定 (使用したい場合はAP基盤に相談してください)
    - escapeBody
    - htmlEscape
    - theme
    - transform
- java-time-jsptags
- JSTL (ＪＳＴＬリファレンス)
  - 使用可能なタグは次の通り
    - c:set
    - c:remove (殆ど使用しない)
    - c:if
    - c:choose c:when c:otherwise
    - c:forEach
    - c:forTokens (殆ど使用しない)
    - c:import (殆ど使用しない。JSPの実装#ファイルのinclude方法 を参照のこと)
    - fmt:formatNumber
  - EL関数(fn:containsなど) は fn:escapeXml 以外全て使用可能
- TERASOLUNA提供のEL関数
  - f:h
  - f:js, f:hjs (クライアントサイド開発コーディング規約上、基本的に使用することは無い)
  - f:br
  - f:cut
  - f:u (使う機会は無いはず。使いたい場合はAP基盤に相談することを推奨)
  - 他のEL関数は使用不可 (使用したい場合はAP基盤に相談)

留意事項

- フォーム系のタグには spring-form JSP Tag Library を使用する
- URLの構築には spring:url タグを使用する (JSTLの c:url ではなく)
  - form:formタグのaction属性に渡すURLも spring:url タグで構築する
- HTMLエスケープについては 全体実装方針#HTMLエスケープ を参照のこと
- 日時のフォーマットには javatime:format タグを使用する
  - java-time-jsptags
  - JSTLのfmt:formatDateはjava.timeに対応していないので使用不可

ファイルのinclude方法

- 単純に分割しただけのJSPファイルは includeディレクティブ (e.g. <%@ include file="foo.jsp" %>)を使用する
- 動的にincludeしたいケースがある場合は基本的にincludeアクション(jsp:include)ではなく、JSTLのimportタグ (c:import) を使用する
  - 但し、JSPファイルからの相対パスで指定したい場合はincludeアクションを使用する
  - 動的にincludeしたいケースは殆ど無い認識 (思いつかない)
  - 画面共通部品を作るのであれば、動的にincludeしてパラメータ渡しをするのではなくtagファイルを使うこと (see 画面共通部品)

画面遷移リンク・ボタンの実装方法

- 以下に示す方法で実装出来ないケースがあった場合はAP基盤に相談すること
- JavaScriptで無理やり遷移させたり、無駄にリダイレクトさせたりしないこと

画面遷移ボタンがaタグの場合

- hrefにurlを指定するだけ
- URLはspring:urlタグで構築すること (クエリ文字列を文字列連結で構築しないこと)

実装例

```
<spring:url var="detailUrl" value="/sample/IIAP1200/init" htmlEscape="true">
  <spring:param name="receiptNo" value="${receiptNo}" />
</spring:url>

<a href="${detailUrl}">詳細</a>
```

画面遷移ボタンがinput or buttonタグの場合 (csm only)

- 以下の実装例のようにdata属性にurlを指定するだけ
- URLはspring:urlタグで構築すること (クエリ文字列を文字列連結で構築しないこと)
- 単なる画面遷移の場合は、直接遷移すること
  - POST
    (やGET)で自画面表示Controllerにリクエスト投げ、リダイレクトすることで画面遷移するのは無駄なのでしないこと

実装例

```
<spring:url var="detailUrl" value="/sample/IIAP1200/init" htmlEscape="true">
  <spring:param name="receiptNo" value="${receiptNo}" />
</spring:url>

<input type="button" data-ati-toggle="linkButton" data-ati-url="${detailUrl}">詳細</input>
```

詳細は次のリンクを参照のこと

- 画面共通部品#画面遷移ボタン
- 画面共通部品#別ウィンドウ表示ボタン

フォーム部品の実装例

JSPでフォーム部品を実装するサンプルを示す。
代表例としてテキストボックス、チェックボックス、セレクトボックスについてJava側と合わせてサンプルを記載する。
(全体的にソースのコメントは省略しているが、 実装時には必ずコメントを記載すること)

テキストボックス実装例

- text (<input type="text">)、label(<label>)
  - Java(Controller)

```
@PostMapping("yyy")
public String yyy(XxxForm form, Model model) {
    String str = xxxService.getFoo();
        form.se"123"); // <1>
        model.addAttr"xxxForm", form);
    return "xxx";
}
```

    - <1> 画面に初期値を設定する場合
  - Java(Model)

```
public class XxxForm implements Serializable {
    private static final long serialVersionUID 1L;
    private String foo;

    // foo getter/setter
}
```

- JSP

```
<spring:url var="zzzUrl" value="/zzz" />
<form:form action="${zzzUrl}" modelAttribute="xxxForm"> <!-- <2> -->
    <form:label path="foo">input: </form:label>
    <form:input path="foo"/> <!-- <3> -->
</form:form>
```

  - <2> formタグのaction属性に渡すURLも spring:url タグで構築する
  - <3> <1>で初期値を設定している場合、<1>で設定した値がテキストボックスに入力済みの状態となる
- レンダリング結果(※見やすくするために改行を入れています)

```
<form id="xxxForm" action="/zzz" method="post">
    <label for="foo">input: </label> <!-- <4> -->
    <input id="foo" name="foo" type="text" value="123"/> <!-- <5> <6> -->
</form>
```

  - <4> <form:label>タグのpath属性の指定がfor属性に設定されている
  - <5> <form:input>タグのpath属性の指定がidおよびnameに設定されている
  - <6> XxxForm#setFoo(String)に値を設定しているため、初期値としてvalue属性に設定されている

チェックボックス実装例

- checkbox (<input type="checkbox">)、checkboxes (<input type="checkbox">)
  - Java(Controller)

```
@PostMapping("yyy")
public String yyy(XxxForm form, Model model) {
    List<Foo> fooList = getFooList();
        model.addAttr"fooList", fooList); // <1>
    List<String> foobarList new ArrayList<String>();
        foobarLis"1");l(
        form.setFoobarList(foobarL// <2>
        model.addAttr"xxxForm", form);
    return "xxx";
}
```

    - <1> Listを設定しているがオブジェクトの配列でも可
    - <2> 画面に初期値を設定する場合
  - Java(Model)

```
public class Foo implements Serializable {
    private static final long serialVersionUID 1L;
    private String id;
    private String name;

    // id,name getter/setter
}
```

```
public class XxxForm implements Serializable {
    private static final long serialVersionUID 1L;
    private List<String> foobarList;

    // foobarList getter/setter
}
```

- JSP

```
<!--  form:checkbox  -->
<spring:url var="zzzUrl" value="/zzz" />
<form:form action="${zzzUrl}" modelAttribute="xxxForm">
    <c:forEach var="bar" items="${fooList}">
        <form:checkbox path="foobarList" value="${bar.id}" label="${bar.name}" /> <!--  <3>  <4>  <5>  -->
    </c:forEach>
</form:form>
```

or

```
<!--  form:checkboxes  -->
<spring:url var="zzzUrl" value="/zzz" />
<form:form action="${zzzUrl}" modelAttribute="xxxForm">
    <form:checkboxes path="foobarList" items="${fooList}" itemLabel="name" itemValue="id" delimiter=" " />
<!--  <3>  <4>  -->
</form:form>
```

  - <3> 配列も可。<1>で配列を設定した場合も実装方法は同じ。
  - <4> <2>で初期値を設定している場合、<2>で設定した値とidの値が一致するチェックボックスが選択済みとなる
  - <5>
    ラベルを属性(label)に指定した場合はHTMLエスケープされるがタグの外でラベルを指定する場合は明示的にHTMLエスケープ(f:h())する必要がある

```
<form:checkbox path="foobarList" value="${bar.id}" />${f:h(bar.name)}
```

- レンダリング結果(※見やすくするために改行を入れています)

```
<!--  form:checkbox  -->
<input id="foobarList1" name="foobarList" type="checkbox" value="1" checked="checked"/><label for="foobarList1">
xxx1</label>  <!--  <6>  <7>  -->
<input type="hidden" name="_foobarList" value="on"/>
<input id="foobarList2" name="foobarList" type="checkbox" value="2"/><label for="foobarList2">xxx2</label>
<input type="hidden" name="_foobarList" value="on"/>
<input id="foobarList3" name="foobarList" type="checkbox" value="3"/><label for="foobarList3">xxx3</label>
<input type="hidden" name="_foobarList" value="on"/>

<!--  form:checkboxes  -->
<span><input id="foobarList1" name="foobarList" type="checkbox" value="1" checked="checked"/><label for="
foobarList1">xxx1</label></span>  <!--  <7>  <8>  -->
<span> <input id="foobarList2" name="foobarList" type="checkbox" value="2"/><label for="foobarList2">xxx2
</label></span>
<span> <input id="foobarList3" name="foobarList" type="checkbox" value="3"/><label for="foobarList3">xxx3
</label></span>
<input type="hidden" name="_foobarList" value="on"/>
```

  - <6>
    <form:checkbox>タグのpath属性はname属性と数値をした形でid属性に、value属性はvalue属性に、label属性はlabelタグとして設定されている
  - <7> XxxForm#setFoobarList(String)に値を設定しているため、checked="checked"が設定されている
  - <8>
    <form:checkboxes>タグのpath属性はname属性と数値をした形でid属性に、itemValue属性はvalue属性に、itemLabel属性はlabelタグとして設定されている
  - ※<form:checkbox>と<form:checkboxes>の差異
    - 一つのチェックボックスでフラグ的に使用する場合は<form:checkbox>、複数のチェックボックスの値を配列やリストで受け渡す場合は<form:checkboxes>を使用すると実装しやすい
    - <form:checkboxes>にはspanタグが付与される
    - また、<form:checkbox>では一つずつ、<form:checkboxes>では全体で一つhiddenタグが付与される
      hiddenタグが付与される理由
      ⇒hiddenタグが存在することにより、一つもチェックされない場合でも空の配列またはリストが設定される

セレクトボックス実装例

- select (<select>)、option (<option>)、options (<option>)
  - Java(Controller)

```
@PostMapping("yyy")
public String yyy(XxxForm form, Model model) {
    List<Foo> fooList = getFooList();
        model.addAttr"fooList", fooList); //  <1>
        form.setl"3"); //  <2>
        model.addAttr"xxxForm", form);
    return "xxx";
}
```

    - <1> Listを設定しているがオブジェクトの配列でも可
    - <2> 画面に初期値を設定する場合
  - Java(Model)

```
public class Foo implements Serializable {
    private static final long serialVersionUID 1L;
    private String id;
    private String name;

    // id,name getter/setter
}
```

```
public class XxxForm implements Serializable {
    private static final long serialVersionUID 1L;
    private String hoge;

    // hoge getter/setter
}
```

  - JSP

```
<spring:url var="zzzUrl" value="/zzz" />
<form:form action="${zzzUrl}" modelAttribute="xxxForm">
    <form:select path="hoge">
        <form:option value="" label="選択してください" />
        <c:forEach var="bar" items="${fooList}">
            <form:option value="${bar.id}" label="${bar.name}" /> <!-- <3> <4> <5> -->
        </c:forEach>
    </form:select>
</form:form>
```

    or

```
<spring:url var="zzzUrl" value="/zzz" />
<form:form action="${zzzUrl}" modelAttribute="xxxForm">
    <form:select path="hoge">
        <form:options items="${fooList}" itemLabel="name" itemValue="id" /> <!-- <3> <4> -->
    </form:select>
</form:form>
```

    - <3> 配列も可。<1>で配列を設定した場合も実装方法は同じ。
    - <4> <2>で初期値を設定している場合、<2>で設定した値とidの値が一致するセレクトボックスが選択済みとなる
    - <5>
      ラベルを属性(label)に指定した場合はHTMLエスケープされるがタグのボディ部でラベルを指定する場合は明示的にHTMLエスケープ(f:h())する必要がある

<form:option value="${bar.id}">${f:h(bar.name)}</form:option>

- ※配列やリストなどでControllerから渡された値以外に選択していないことを示す初期値(「選択してください」など)を設定したい場合は<form:option>、
  不要な場合は<form:options>を使用する。(JSPへ渡す前に選択していないことを示す値を設定すれば<form:options>でも同様のことは実現可能)

◦ レンダリング結果(※見やすくするために改行を入れています)

```
<!--  form:option  -->
<select id="hoge" name="hoge">  <!--  <6>  -->
<option value="-">選択してください</option>  <!--  <7>  -->
<option value="1">xxx1</option>
<option value="2">xxx2</option>
<option value="3" selected="selected">xxx3</option>  <!--  <8>  -->
</select>

<!--  form:options  -->
<select id="hoge" name="hoge">  <!--  <6>  -->
<option value="1">xxx1</option>  <!--  <9>  -->
<option value="2">xxx2</option>
<option value="3" selected="selected">xxx3</option>  <!--  <8>  -->
</select>
```

- <6> <form:select>タグのpath属性がid、name属性に設定される
- <7> <form:option>タグのvalue属性がvalue属性、label属性が<option>タグのボディ部に設定される
- <8> XxxForm#setHoge(String)に値を設定しているため、selected="selected"が設定されている
- <9> <form:options>タグのitemValue属性がvalue属性、itemLabel属性が<option>タグのボディ部に設定される

その他フォーム部品

上記以外のフォーム部品についてもテキストボックス もしくはチェックボックス と同様の方法で実装可能。

- radio (<input type="radio">)

  <form:radiobutton path="foo"/>

  <form:radiobuttons path="foo" items="${bar}"/>
- password (input type="password">)

  <form:password path="foo"/>
- hidden (<input type="hidden">)

  <form:hidden path="foo"/>
- textarea (<textarea>)

  <form:textarea path="foo" rows="n" cols="n"/>
- error

  <form:errors path="*" cssClass="foo"/>

  ◦ path="*" - displays all errors
- form (<form>)

```
<form:form action="/foo">
```

http://docs.spring.io/spring/docs/current/spring-framework-reference/html/view.html

JSTL

JSTLを利用した実装サンプルを示す。

金額の3桁カンマ付加表示の実装例

<fmt:formatNumber>タグを使用する。

```
<%--　１．3桁カンマ区切り　-%>
<fmt:formatNumber value="1234" pattern="#,###" /><br>　<!--　実行結果　　1,234　-->
<fmt:formatNumber value="-1234" pattern="#,###" /><br> <!--　実行結果　　-1,234　-->
<fmt:formatNumber value="1234" pattern="-#,###" /><br> <!--　実行結果　　-1,234　-->

<%--　２．桁数合わせでゼロを表示させる　-%>
<fmt:formatNumber value="12.34" pattern="000.000" /><br> <!--　実行結果　　012,340　-->

<%--　３．パーセント変換　-%>
<fmt:formatNumber value="0.278" pattern="##.#%" /><br>　<!--　実行結果　　27.8%　-->

<%--　４．セミコロンを指定し、正と負の場合の変換方法を指定　-%>
<fmt:formatNumber value="-1000" pattern="△#,###;▼#,###" /><br> <!--　実行結果　　▼1,000　-->
<fmt:formatNumber value="1000" pattern="△#,###;▼#,###" /><br> <!--　実行結果　　△1,000　-->
```

JSテンプレート内で変換したい場合は Number Helper を参照

その他

- url構築
  - JSP

```
<spring:url value="hoge/{fuga}" var="hogeUrl" htmlEscape="true">
    <spring:param name="fuga" value="${hogehoge}"/>
</spring:url>
<form:form action="${hogeUrl}">
...
```

  - Java(Controller)

```
@PostMapping(value="/hoge/{fuga}")
public String　xx@PathVariable("fuga") String　fuga,　Model　model)　{
    //　...
}
```

- 改行コード変換（DBで改行コードを含む文字列をHTML上で改行表示したい等）
  - JSP

```
${f:br(f:h(hoge))}
```

  - see http://terasolunaorg.github.io/guideline/5.3.0.RELEASE/ja/ArchitectureInDetail/WebApplicationDetail/TagLibAndELFunctions.html#f-br