画面共通部品の実装

タグファイル、タグハンドラの実装について説明する。
どちらの形態で実装するかは基本的に次の様に判断する。

- デザイン要素を多く含むUI系の画面共通部品 → タグファイル
- 多少複雑な処理を必要とする画面共通部品 → タグハンドラ

タグファイル

実例 をベースに実装方法を説明する。
次のコード例は実例を抜粋したもの。

```
<%@ tag pageEncoding="UTF-8" description="結果メッセージ(ResultMessages)を表示します。> <!--1-->
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix='> <!--2-->
<%@ taglib uri="http://www.springframework.org/tags" prefix="sprir>;"%
<%@ attribute name="id" description="コンテナ要素のHTML'id'属ｲ> <!--3-->
<
%@ attribute name="messagesAttribute" description="結果メッセージ。デフォルトでresultMessagesを利用します。" type="org.ter
asoluna.gfw.common.message.ResultMessages"%>

<c:set var="msgAttr" value="${messagesAttribute == null ? resultMessages : messagesA" />ute}
<c:if test="${msgAttr != nι">}
  <spring:eval var="resultMessagesItr" expression="msgAttr.iterator()" /> <!--4-->
  <!-- error_box START -->
  <div <c:if test="${not empty ">id=${id}</c:if> class="asw-notice-massagｃ'
    <ul>
      <c:forEach var="resultMessage" items="${resultMessagesItr}">
        <li><spring:message code="${resultMessage.code}" arguments="${resultMessage.args}" htmlEscape="true" /></li>
      </c:forEach>
    </ul>
  </div>
  <!-- error_box END -->
</c:if>
```

- <1> tagディレクティブを例のように設定する
- <2> 各画面JSPファイルと異なり、使用するタグリブのtaglibディレクティブを記述する必要がある
  - taglibディレクティブはinclude.jspで定義されてあるものの中から使用すること
    - atd用 include.jsp
    - ati用 include.jsp
- <3> タグの入力パラメータを定義可能
- <4> なるべく避けた方が良いが、画面共通部品という事でspring:evalタグを使用しても良い
  - 但し、安全のためAP基盤の承認を得ること

このタグファイルを各画面JSPで使うには、

- include.jspに <%@ taglib tagdir="/WEB-INF/tags" prefix="ati-ui" %> というようにtaglibディレクティブを追加し (例)
- 各画面JSPで <ati-ui:messagesPanel id="foo" /> というようにタグハンドラと同じ形で使用する
  - タグファイル名がタグ名になる

プロパティの参照方法

タグファイルでプロパティを参照したい場合は以下の様に、TagConfigクラスを作成し、タグファイルからspring:evalタグで参照する。
なお、urlなどのプロパティについてはTagConfigクラスを作らずに プロパティ管理
に記載のatc:propertyタグで直接プロパティ参照可能。

TagConfigクラスのコード例

```
package jp.co.anas.atp.xyz.web.{division}.tags.config; //  <1>
```

```
@Component //  <2>
public class SnsShareBtnGroupTagConfig  //  <3>

    @Value("${sns.line-url}") //  <4>
    private String  lineUrl;

    @Value("${sns.twitter-url}")
    private String  twitterUrl;

    public String  getLineUrl()  {
        return  lineUrl;
          }

    public String  getTwitterUrl()  {
        return  twitterUrl;
          }
}
```

- <1> 規約に従ったパッケージ (jp.co.anas.atp.xyz.web.{division}.tags.config) に作成すること (Springに一括登録するため)
- <2> Springに一括登録するため @Component アノテーションを付ける
- <3> クラス名は分かりやすさのため「タグファイル名のUpperCamelCase + TagConfig」とする (上記のコード例は snsShareBtnGroup.tag で使用するクラス)
- <4> プロパティ管理 の通り、@Valueアノテーションを使用する

TagConfigクラスを使用するタグファイルのコード例 (/WEB-INF/tags/snsShareBtnGroup.tag)

```
<%@  tag  pageEncoding="UTF-8"  description="SNSシェアボタングループタ>"%

<spring:eval var="snsConfig" expression="@snsShareBtnGroupTagConfig" /> <!--1-->
<ul>
  <li><a href="${snsConfig.lineUrl}">...</a> <!--2-->
  <li><a href="${snsConfig.twitterUrl}">...</a>
</ul>
```

- <1> spring:evalタグでTagConfigのインスタンスを取得する
- <2> getter経由でプロパティを参照する

タグハンドラ

- org.springframework.web.servlet.tags.RequestContextAwareTag を継承して作成
  - HTMLエスケープを使用する場合は org.springframework.web.servlet.tags.HtmlEscapingAwareTag を継承
  - 必要に応じて、より深いサブクラスを利用する
- 実装についてはSpringが提供するタグハンドラの実装を参考にすること

プロパティの参照方法

タグファイルと同様、タグハンドラでプロパティを参照したい場合は以下の様に、TagConfigクラスを作成する。
配置先パッケージやその他の規約は上記を参照のこと。(タグファイル用のTagConfigクラスと同じ)

```
package jp.co.anas.atp.xyz.web.{division}.tags.config;

@Component
public class FooTagConfig  {

  @Value("${プロパティキー}")
  private String  bar;

  public String  getBar()  {
    return  bar;
     }

}
```

プロパティの参照は次の様に、アプリケーションコンテキストからTagConfigクラスを取得し行う。

```
package jp.co.anas.atp.xyz.web.{division}.tags;

public class FooTag extends  HtmlEscapingAwareTag //  <1>

    private String  value;

    public void  setValu(String  value)  {
        this.value  =  value;
          }

    @Override
    protected int  doStartTagInternal(throws Exception  {

                    FooTagConfig  config  =  getRequestContext().getWebApplicationContext().getBean(FooTa//  <2>g.class);
        String  bar  =  config.getB//  <3>

        String  result  =  htmlEscape(value)  +  bar;

                    pageContext.getOut().print(result);

        return  SKIP_BODY;
          }

}
```

- <1> SpringのRequestContextAwareTag(及びサブクラス)を継承する
- <2> getRequestContext().getWebApplicationContext()でSpringのアプリケーションコンテキストを取得し、getBeanでTagConfigクラスを取得する
- <3> TagConfigクラスからプロパティを取得する

参考: tldファイルの例

```
<?xml  version="1.0"  encoding="UTF-8"?>
<taglib xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  http://java.sun.com/xml/ns/javaee/web-jsptaglibrary_2_1.xs" version="
2.1">

  <description>ATD  JSP  Tag  Lib</description>
  <tlib-version>1.0</tlib-version>
  <short-name>atp</short-name>

  <tag>
    <description>タグの説明文</description>
    <name>foo</name>
    <tag-class>jp.co.anas.atp.xyz.web.{division}.tags.FooTag</tag-class>
    <body-content>empty</body-content>
    <attribute>
      <description>属性の説明文</description>
      <name>value</name>
      <required>true</required>
      <rtexprvalue>true</rtexprvalue>
    </attribute>
  </tag>

</taglib>
```

参考: 各画面JSPでタグハンドラを使えるよるための設定

- tldファイルの指定 (web.xml)

```
<jsp-config>
  <taglib>
    <taglib-uri>http://www.anas.co.jp/asw/atp-xyz/tags</taglib-uri>
    <taglib-location>/WEB-INF/tld/atp-xyz.tld</taglib-location>
  </taglib>
  <!--  ...  -->
</jsp-config>
```

- include.jsp

```
<%@  taglib  uri="http://www.anas.co.jp/asw/atp-xyz/tags"  prefix="at
```