XperienCentral JSPs

This topic covers the details of the 6 steps you performed in the Design in 6 Steps. It will explain more about JSPs and their relation to XperienCentral.

In This Topic

Contents of a JSP

A JSP file contains simply the (X)HTML (or XML) that is requested. JSP directives allow the use of so-called tag libraries. To use the core functions of the JSP Standard Tag Library (JSTL), the following directive should be placed at the beginning of the file:

```
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
```

This URI is not a website URL, but it specifies the tag library to be used. In XperienCentral's `web.xml`, the URI tag library is mapped to the real location of the TLD path. In a default installation this would be `<xperiencentral-root>/webmanager-webapps/webmanager-backend-webapp/target/webmanager-backend-webapp-1.0-SNAPSHOT/WEB-INF/wm/tld`. The prefix (in this case "c") determines the tag name preceding the colon of a request. The JSTL core functions offer, among other things, the possibility for iteration. An example:

```
<c:forEach var="index" begin="1" end="5">
        ...
</c:forEach>
```

To gain access to Java methods, the `presentationcontext` (lowercase), is added to the request. This object can be requested in the JSP using Expression Language (EL). For example:

```
${presentationcontext}
```

The EL offers a simple way to gain access to objects and associated methods in JSPs. An EL expression is always of the `${...}` form.

The Java method `presentationcontext` consists of a list of methods that can be requested in the JSP. To see a list of all available methods and properties, refer to the XperienCentral Javadoc,

Methods cannot be used immediately but have to be transformed following these rules:

- If the method generates a boolean, remove the "is" prefix.
- If the method generates anything other than a boolean, remove the "get" prefix.
- Replace the initial capital letter with a lowercase letter.
- Remove the opening and closing brackets.

The method `getWebsite()` will become website, for instance. To store the website object from the `presentationcontext` in a new variable, you can use:

```
<c:set var="myWebsite" value="${presentationcontext.website}" />
```

To request the object ID of the channel, use:

```
${presentationcontext.website.id} or ${myWebsite.id}
```

The best way to become familiar with these expressions is to experiment with printing various properties of the `presentationcontext`, website, page and `pageVersion` objects (see the XperienCentral Javadoc).

JSP Guidelines

Because the JSP and XperienCentral API also connect to other XperienCentral modules, for example the caching mechanism, it is important to learn more about what you should and shouldn't do while working with JSPs. If you neglect this the performance of your site may be significantly slower because the caching or personalization mechanisms do not perform well with the JSPs.

XperienCentral's guidelines to use JSPs are the following:

- Start every JSP that should in any way be able to cache with the JSP directive:

```
<%@ page language="java" session="false" buffer="none" %>
```

- Include only those tag libraries that are necessary.
- Avoid as much as possible non-XML compliant JSPs, such as:

```
<a href="<c:out value="${link.url}"/>">title</a>
```

- GX uses JSP Version 2.0 which contains an improved version of the Expression Language (EL), and allows the above to be written as follows:

```
<a href="${link.url}">title</a>
```

- The content of JSPs has to conform to the following JSP code conventions

  http://java.sun.com/developer/technicalArticles/javaserverpages/code_conventio
  http://www.oracle.com/technetwork/java/codeconvtoc-136057.html

Don'ts

- JSPs are designed to contain HTML and scripting code, not Java code. In combination with the special `<% %>` tag (the so-called scriptlets), it is possible to include Java code in the JSP. Try to avoid using these scriptlets because this conflicts with the separation of content, business logic and the design template and it may conflict with the caching mechanism.
- Don't use the `<jsp:include>` tag to include a JSP. Instead use the XperienCentral alternative: `<wm:includeUrl url=""/>`.
- Don't mix XperienCentral sessions with common Java sessions.

Back to top

XML Descriptor File

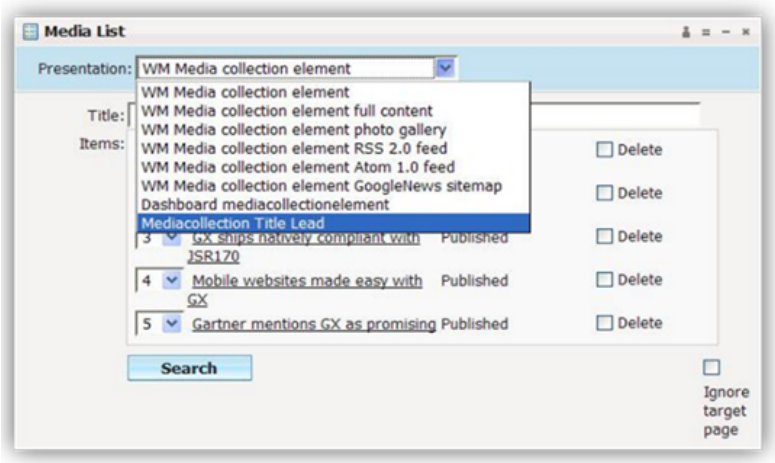
To connect a JSP to XperienCentral, a design template descriptor has to be defined. The design template descriptor – in XML format – identifies the JSP for XperienCentral. Almost every JSP has its own descriptor. XperienCentral reads the design template descriptors when you upload the plugin.

A design template descriptor for a JSPF that implements the design of a paragraph looks like this:

```
<presentation>
        <name>helloworld design template textelement</name>
        <display-name>HWP textelement</display-name>
        <scope>TextElement</scope>
</presentation>
```

The XML file has the same name as the JSP but with the extension `.xml`. XperienCentral uses the scope (in the example above `TextElement`) to categorize all the design templates. The scope names determine for which part in XperienCentral the design template is intended. All (known) scope names can be viewed in the scope design template pull-down list on the `[Design templates]` tab from the **Configuration > Design Template** panel.

It is possible to define several design template descriptors for the same element. If more than one design template descriptor is defined, for an element, for example, a drop-down for selecting one of the design templates for the element appears automatically in XperienCentral. The names of the design templates, as defined in the descriptors (the display name), appears in the drop-down list. This gives editors of the website the possibility to choose from various design templates of every element (provided that there is more than one design template available). For example:



The following fields can be used in the XML descriptor files associated with each JSP. The options are:

Tag	Note	Description
`<name>`	Required	The unique internal name for the design template.
`<display-name>`	Required	The name shown in the drop-down list for the element.
`<scope>`	Several are possible	Specifies for which part in XperienCentral the descriptor is intended.
`<ssi>` `<presentation>`	Optional Optional	Disk location of the handling JSPF for the server side include version.
`<property>`	Optional and several are possible	Definitions of the options to be configured in XperienCentral.
`<channel>`	Optional and several are possible	Definitions of the channels.

Property Tags, Design Template Variants and Styles

In order to add the options for editors to a JSP, properties have to be specified in the design template descriptor. This is done using a property tag under the design template tag. For example:

```
<property>
        <propertyname>showLead</propertyname>
        <display-name>Show the lead</display-name>
        <datatype>Boolean</datatype>
        <default>true</default>
</property>
```

In the JSP this property can be used as follows:

```
<wm:presentationProperty var="showLead" label="showLead" />

<c:if test="${showLead}">
        <p>${mediaItemVersion.lead}</p>
</c:if>
```

If at least one property is specified in a design template descriptor, a design template variant can be created and added to the default value. In a design template variant the default values, if specified, are overruled. In order to make everything more flexible, you can create several design template variants for the same design template. All the created variants will then appear in a drop-down list next to the relevant part in XperienCentral. It is not possible to export variants.

The possible property tags are:

Tag	Note	Description
`propertyname`	Required	The name as it is used as identifier in the JSP.
`display-name`	Required	The name as appears in XperienCentral drop-down lists.
`datatype`	Required	The options are "Boolean", "String", and "Integer".
`default`	Optional	If no style property is connected to a design template property, the default value is used.

The `integer`, `string`, and `boolean` data types make it possible to configure the JSPs without having to use Java code. If possible, create a `pagemetadata` plugin if you want to assign specific objects to a page.

Special case: separator properties

The Community Edition "Content" `pagepart` design template contains two design template properties (`fromSeparator` and `toSeparator`) that specify the range of the elements to be printed. These properties are related to the separators. It is not necessary to retrieve the values of these properties using the `<wm:presentationProperty-tag`, when you retrieve the elements - the method will automatically use these values.

Channel Tag

XperienCentral can render a page as a PDF document send pages as newsletters and send special pages to mobile devices. Using channel design templates, it is possible to create a single page and use multi-channel publishing to make the content viewable by all possible devices.

A channel can be viewed using the `?channel=…` query argument on a page, for example `http://localhost:8080/web/show/?channel=pdf`.

Example uses are:

Channel-name	Usage
`text`	Generates plain text.
`textmail`	Generates plain text used by the Newsletter component.

`htmlmail`	Generates HTML used by the Newsletter component.
`pdf`	Generates the design template in PDF.
`mobile`	Generates HTML for mobile devices.

If a design template does not specify a channel, this item will be skipped when rendering. For example, if a table element should not be rendered for a newsletter, do not add channel `textmail` or `htmlmail` to the table element design template(s).

The styles included in XperienCentral has tags with the following meaning:

Tag	Note	Description
`name`	Required	The frame identifier.
`presentation`	Required	The relative path to the JSP.

If the website uses HTML frames, the content of the frames is specified using a frame tag in the JSP's design template descriptor that contains the frame set. The XML looks as follows (using sample entries):

```
<frame>
        <name>top</name>
        <location>/WEB-INF/project/jsp/page/top.jsp</location>
</frame>
```

The tags have the following meaning:

Tag	Note	Description
`name`	Required	The frame identifier.
`location`	Required	Relative path to the JSP.

To use such a frame in the JSP, create a link in the URL with `frame=<frame identifier>`. For the XML example above, the link is as follows:

```
<wm:link var="topFrame" frame="top" passOnAllParameters="true" />
```

The frame attribute is a so-called dynamic attribute, which means that this attribute is not specified in the TLD. The implementation of `<wm:link>` is such that all dynamic attributes are placed in the query string as key value pairs. The `passOnAllParameters` attribute passes on all the parameters from the current URL if it is set to `true`. In the HTML `frameset` tag, this looks as follows:

```
<frame name="top" scrolling="NO" src="${topFrame.url}" frameborder="NO" /
```

Static CSS style sheets can be connected to a JSP using a CSS tag. The XML looks as follows (using sample entries):

```
<css>
        <name>main</name>
        <location>/static/project/css/mystyle.css</location>
        <pagedependent>true</pagedependent>
</css>
<css>
        <name>sectionNews</name>
        <location>/static/project/css/news.css</location>
        <pagedependent>true</pagedependent>
</css>
```

The tags have the following meanings:

Tag	Note	Description
`name`	Required	The stylesheet identifier.
`presentation`	Required	The relative path to the stylesheet.
`pagedependent`	Required	A boolean that determines whether the content of the stylesheet is determined by limitations set in the XperienCentral style and that are valid for that page.

This does not mean that the stylesheet is automatically included in the source. This is accomplished by requesting the current page from the `presentationcontext`, and then requesting all the connected stylesheets from that page. This generates an array of stylesheet objects that each generate the URL using the `getUrl()` method.

In practice it should be used as follows:

```
<c:set var="styleSheets" value="${presentationcontext.pageVersion.styleSheets}" />
<c:forEach var="styleSheet" items="${styleSheets}">
        <link rel="stylesheet" href="${fn:escapeToXml(styleSheet.url)}" type= "text/css" />
</c:forEach>
```

Back to top