Design in 6 Steps

In This Topic

This topic explains in a few steps how any design/website can be placed in XperienCentral and how it can be made dynamic. This is primarily intended to ensure that XperienCentral's page content is shown in the new design. This chapter will show in a few simple steps the power of the XperienCentral SDK. You may not completely understand all the details immediately after taking these steps. The details are explained in detail in XperienCentral JSPs.

Step 1: Collect and Check the HTML, CSS and Javascript

Every project based on XperienCentral requires an HTML version of the website before the design of the site can be implemented. This step uses this HTML version in such a way that it can be the starting point for a new design template of a page in XperienCentral. The static HTML website HTML is usually provided by the designer who has already applied the GX Software plugin guidelines.

It is a good practice to check the design to ensure that it meets the following reqluirements:

- Does the HTML validate against its declared doctype? (http://validator.w3.org)
- Does the CSS validate against the latest CSS version? (http://jigsaw.w3.org/css-validator/)
- If it is a government website, does it conform to the government's guidelines? See http://www.cynthiasays.com/ and http://www.webrichtlijnen.nl/.
- If JavaScript frameworks (such as jQuery) are used, are they the latest versions? Is inline editing supported?
- Has the design has been checked by a SEO (Search Engine Optimization) professional?
- Are editors able to exactly r-create the examples by adding content into the elements? For example, an image inside a table element will be difficult to create with the default elements.
- If possible, check whether the HTML that has no content (for example a `<div class="rounded-corner">&nsbp;</div>` can be removed.
- If long texts or titles (navigation) are used, does the design still look good (cross-browser)?
- Links that are added by an editor will by default always have a CSS class assigned (`<a class="normal">` …).
- Bold text should be formatted using the `<strong>` tag and italic using the `<em>` tag.
- Can all page sections be moved around?
- Are all non-static images rendered by an image element?
- Are page headers and footers re-usable so that only one main page design template is needed?

Back to Top

Step 2: Create a Design Template Plugin

In order to deploy your custom design template to an XperienCentral installation, you have to use a design template plugin (XperienCentral component bundle). The plugin is a JAR file that contains all your JSPs and static files. Once it is deployed, the JSPs and static files are copied to the correct directories in the XperienCentral installation. Follow these steps:

1. Open a Command prompt in a folder outside your XperienCentral installation folder where you would like to create your design template plugin, `C:\GX\plugins\` for example..
2. Run the following Maven command and replace "version" with your version of XperienCentral, the "artifact ID" with a name of your choice and be sure to add the correct path to your installation's settings.xml file.. The command is one string.

```
mvn archetype:generate -DinteractiveMode=false -DarchetypeGroupId=nl.gx.webmanager.archetypes -
DarchetypeArtifactId=webmanager-presentation-archetype -DarchetypeVersion=10.36 -DgroupId=com.gxwebmanager.
helloworld -DartifactId=helloworldDesign template
-Dclassprefix=HelloWorld -s C:\GX\XperienCentral\settings.xml
```

- You have to fill in the XperienCentral version that you are using with the property ‘DarchetypeVersion’. In the above example it’s version R36 — the version number of XperienCentral used in code must take the form 10.x where x is the major/minor version number. For example, XperienCentral version R35 is declared as 10.35 in code and XperienCentral version R35.1 is declared as 10.35.1.
- In this example the plugin will be named "helloworldDesign template" and all examples in this topic will use that name.

You should see a ‘Build successful’ message from the Maven command and the helloworldDesign template folder should be created. The created folder structure is:

```
/src/main/..
        /java/nl/gx/product/Activator.java
        /resources/Design templatetype/..
                /jsp/
                /static/helloworldDesign
```

3. Create several subdirectories in the `/static/helloworldDesign template` folder for storing the various types of static files (img, css, js, and so forth).
4. Copy your static files from the design to the `/static/helloworldDesign template` subfolder.
5. Create several subdirectories in the `/jsp/` folder for storing the various types of design templates (page, element, forms, pageparts, and so forth).

If a Design template plugin depends on design template JSPs that are part of the original XperienCentral platform, these JSPs should be copied to the design template plugin in the directory: `/src/main/resources/Design templatetype/jsp/wm`.

6. Copy the received HTML file from the design into the `../jsp/page/` folder, and rename it to (for example) `content.jsp`.

Back to Top

Step 3: Convert the HTML to JSP

JSP files contain not just the the website’s HTML code but also functions for using data from XperienCentral dynamically on the page. This makes it possible to, for example, place a code in the JSP to print a paragraph or an image’s URL. These functions originate in the XperienCentral API. This API can be used in the JSPs via a taglib. This taglib has to be imported at the beginning of the JSP.

Each XperienCentral JSP file has an associated descriptor file. This descriptor file is an XML file that contains more information about the JSP file. It contains, for example, the name, a description, and possible additional dynamic parameters such as design template properties.

1. Open both the `page.jsp` and `content.jsp` with an IDE such as Eclipse.

   Additional instructions about XperienCentral plugins and Eclipse can be found in Using Eclipse.

2. From the `page.jsp` file, copy the first lines that include the taglibs:

```
<%@ page language="java" session="false" buffer="none" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<%@ taglib uri="http://www.gx.nl/taglib/wm" prefix="wm" %>
<%@ taglib uri="http://www.gx.nl/taglib/functions" prefix="wmfn" %>
```

3. Paste the copied JSP code in the `content.jsp` file above the first line in the original HTML code

4. Create a new `mydesign.xml` file and place it in the same folder as `mydesign.jsp`. This XML-descriptor (explained in detail in Example Design Layout) should contain the following:

```
<design template>
        <name>helloworldDesign template mydesign</name>
        <display-name>HWP MyDesign</display-name>
        <scope>Page</scope>
</design template>
```

5. Adjust the name and display name in the XML descriptor file. The name-value will be used by XperienCentral to identify the design template, and should be unique. The value of the `name` attribute in the descriptor of a JSP should also be prefixed by the plugin ID as required by the development guideline G139. The display name will be visible for editors when selecting the page design template for a page.
6. Now it is time for the deployment of the design template plugin to XperienCentral. In a Command prompt, run the following command in the helloworldDesign template folder:

   ```
   mvn clean package -s C:\GX\XperienCentral\settings.xml
   ```

7. This command will create a target folder that contains the plugin: `helloworldDesign template-1.0.0.jar`.
8. You can either upload this plugin using the Plugin Management Console in XperienCentral or copy the file to the `C:\GX\XperienCentral\work\deploy` folder manually.
9. It takes a few seconds for XperienCentral to extract the plugin, read the contents and the XML descriptor files and then make the design template available. It is possible to check whether the new design template is available by logging in to the edit environment and opening the **Configuration > Design templates** panel. Set the scope to "page" and you should see your new design template.

If you don't see your design template in the panel something might have gone wrong.

Back to Top

Step 4: Change the Paths to Dynamic Paths

Static files (javascript, CSS, etc.) should be requested from the web server (for instance `http://localhost:8080/web/static/project/images/myimage.jpg`) when viewing the page. All the paths to static files have to be modified in the JSPs. The method below takes care of most structural problems. The JSP code functions in the local development environment as well as in the production environment, where the host settings can be entirely different and, therefore, requesting a file should be completely different.

Go to XperienCentral and assign the new design template to a new page. Perform the following steps:

1. Log into XperienCentral.
2. Create a new page.
3. Set the status of this page to "Published".
4. Enter some text on the page and an image element.
5. Open the properties of the page.
6. Check the "Define other settings for this page" check box and click [Apply].
7. Next to "Design Template", select the design template you created, click [Apply] and after that [Close].
8. Preview the page: Open a new browser window and navigate to the newly created page with the new design.
9. Modify the following line in the `mysite.jsp` in your plugin.

```
<c:set var="staticFilesUrl" value="${Design templatecontext.website.staticFilesUrl}/static
/helloworldDesign template" />
```

> ⚠ By default `${Design templatecontext.website.staticFilesUrl}` is empty, but don't forget to add it because a production server might use it. The value of the variable `static_files_url` can be defined in the XperienCentral Setup Tool (`/web/setup`) on the General (R30 and older) tab. In most situations the default setting will work fine for local development and production environments.

10. Replace the existing paths to the style sheets with dynamic paths based on the `staticFilesUrl`. For example:

```
<link href="${staticFilesUrl}/css/style.css" rel="stylesheet" type="text/css">
```

Repeat this step for external JavaScript paths, images, etc. The added path starts with a '/'. If `${staticFilesUrl}` is empty, then that means the file you are requesting is served by the web server (Apache/JBoss) and not by the application server (XperienCentral).

11. Check whether there are references to pictures in the CSS files. If so, update their paths too.
12. Deploy and upload the plugin.
13. Check in the browser to see what the page now looks like. It should be an exact replica of the original.

Back to Top

Step 5: Change the Content to Dynamic Content

The web page saved in step 1 is now used as the design for a page (design template) in XperienCentral. At this time, however, only the static content from the original HTML code is shown. This should, of course, be replaced with paragraphs and other XperienCentral content elements on the page.

In this step, the content of the page is removed from the JSP and replaced with a request to XperienCentral that generates the content of the page and prints it in the JSP.

1. Use the one distributed with the installed version of XperienCentral that has already been extracted at `C:\GX\XperienCentral\webmanager-webapps\webmanager-backend-webapp\target\webmanager-backend-webapp-1.0-SNAPSHOT\WEB-INF\project.`
2. Copy the following files into your plugin under `/src/main/resources/Design templatetype/jsp/wm`:

    - `pagepart/content.jspf / xml`
        - `pagepart/contentTextVersion.jspf`
        - `XML/contentPDF.jspf`
    - `pagepart/databaseEntity.jspf / xml`
        - `XML/databaseEntityPDF.jspf`
    - `pagepart/mediaItemContent.jspf / xmlpagepart/sitemap.jspf / xml`
        - `pagepart/mediaItemContentTextVersion.jspf`
        - `XML/mediaItemContentPDF.jspf`
    - `pagepart/searchResults.jspf / xml`

3. In `content.jsp`, locate the content (text + images) of the page saved in step 1 and replace it with:

```
<wm:pagePart label="WM content" />
<wm:pagePart label="WM databasepage"/>
<wm:pagePart label="WM mediaItemContent"/>
<wm:pagePart label="WM sitemap" />
<wm:pagePart label="WM searchresults"/>
```

The default design templates for rendering dynamic content have now been copied and are in use by your page design template. The page should now display the content that has been added by editors.

Back to Top

Step 6: Apply Other "Quick Wins"

At this point, the content page part in the design template is dynamic: The content shown is generated by XperienCentral. To make an even bigger impact on visitors, other parts of the page can also be easily made dynamic. The task of an engineer working with the SDK is simply making all items from the HTML design that should be dynamic, dynamic (for example, navigations and page sections).

Examples of "Quick Wins"

Printing the Title of a Page in the Title Bar of the Browser

In `content.jsp`, locate the title of the page (= the part between `<title>` and `</title>`) and replace it with:

```
<title>${fn:escapeXml(website.title)} - ${fn:escapeXml(title)}</title>
```

Rendering a Breadcrumbs Navigation Path

In `content.jsp`, add the following code just above the content page part:

```
<wm:pagePart label="WM path" />
```

And copy the `pagepart/path.jsp` and `xml`. The style of the page has now changed. This is necessary because certain tags that are defined in this style sheet will be used.

Rendering the Publication Date of a Page

To print the page's publication date, use the following code:

```
<fmt:setLocale value="${Design templatecontext.pageVersion.language.metaTagValue}"/>
<fmt:formatDate value="${Design templatecontext.pageVersion.publicationDate}" pattern="d MMMM yyyy" />
```