EMBEDDING APPLICATIONS INTO THEMES

EMBEDDING APPLICATIONS

- It’s often necessary for developers to embed applications into their themes.
- Embedding applications into the `portal_normal.ftl` will render them on each site page.
- Let’s look at a couple of ways we can embed applications.

TAGLIBS USED IN EMBEDDING APPLICATIONS

- There are three taglibs that we can use in our theme to embed applications or content:
  - `<@liferay_portlet["runtime"]`
  - `<@liferay_journal["journal-article"]`
  - `<@liferay_ui["asset-display"]`
- Each of these taglibs takes different parameters.

USING PORTLET PROVIDER CLASS NAME

- The `<@liferay_portlet["runtime"]` expects two parameters:
  1. `portletProviderAction` requests the portlet provider to perform an action for display.
     - Using portletProviderAction.VIEW for the first parameter most commonly used displays the default application view.
  2. The `portletProviderClassName` requires the fully qualified class name of the entity on which we want to perform the action.
- The `portletProviderClassName` is always coupled with the `portletProviderAction`.

```
<@liferay_portlet["runtime"]
    portletProviderAction=portletProviderAction.VIEW
    portletProviderClassName="CLASS.NAME"
/>
```

APPLICATIONS IN THE TAGLIB

- Using the above method will work for a set of applications that can be found in the source code.
- To find which applications work, follow these steps:
  1. You can go to *https://github.com/liferay/liferay-portal*
  2. Search the code for *extends BasePortletProvider*
- From there, you will be able to find the list of applications and what actions you can use with them.

EMBEDDING WEB CONTENT USING LIFERAY PORTLET TAGLIB

- For other applications, such as Web Content, you would need to pass in the *portletName*.
- Here is an example of adding Web Content using `<@liferay_portlet["runtime"]`.

```
<#assign VOID = freeMarkerPortletPreferences.setValue
("portletSetupPortletDecoratorId", "barebone") />
<#assign VOID = freeMarkerPortletPreferences.setValue
("groupId", "${group_id}") />
<#assign VOID = freeMarkerPortletPreferences.setValue
("articleId", "ARTICLE_ID") />

<@liferay_portlet["runtime"]
  defaultPreferences="${freeMarkerPortletPreferences}"
  portletProviderAction=portletProviderAction.VIEW
  instanceId="INSTANCE_ID"
  portletName="com_liferay_journal_content_web_portlet_JournalContentPortlet"
/>

<#assign VOID = freeMarkerPortletPreferences.reset() />
```

THE PORTLET NAME ATTRIBUTE

- The `portletName` is the application id, written as the string reference of the application class path.

```
<@liferay_portlet["runtime"]
    portletName="CLASS_NAME"
/>
```

- For example, the Web Content application would be `com_liferay_journal_content_web_portlet _JournalContentPortlet`.

PORTLET PREFERENCES FOR EMBEDDED APPLICATIONS

- It is also possible to set preferences in an application using `${freeMarkerPortletPreferences}`, as we can see in the Web Content example.
- This allows you to change the application preferences and have it immediately display in the theme.

```
<#assign VOID = freeMarkerPortletPreferences.setValue(
"portletSetupPortletDecoratorId", "barebone") />

<@liferay_portlet["runtime"]
    defaultPreferences="${freeMarkerPortletPreferences}"
    portletName="com_liferay_login_web_portlet_LoginPortlet"
/>

<#assign VOID = freeMarkerPortletPreferences.reset() />
```

PORTLET RUNTIME ATTRIBUTES

- Let's look at some of the additional attributes that can be used:
  - **defaultPreferences**: This is a string of Portlet Preferences for the application that will be rendered. It could include look and feel configurations.
  - **instanceId**: If the application is instanceable, this allows for the instance id to be set.
  - **persistSettings**: This attribute will have an application use its default settings, which will persist across layouts. By default the attribute is set to *true*.
  - **settingsScope**: This attribute specifies which settings the application is to use. The default setting is `portletInstance` but can be set to `group` or `company`.

USING THE JOURNAL ARTICLE TAGLIB

- You can also embed Web Content using the `<@liferay_journal["journal-article"]` taglib.

```
<@liferay_journal["journal-article"]
  articleId="ARTICLE_ID"
  ddmTemplateKey="TEMPLATE_KEY"
  groupId=${group_id}
 />
```

- The `<@liferay_journal["journal-article"]` taglib requires the following:
  - **Article ID**: The id of the Web Content Article you wish to display
  - **Template Key**: The id of any Web Content Template you want to identify
  - **groupId**: The Site id where the content is available

USING THE ASSET DISPLAY TAGLIB

- Finally, you can also embed other specific assets, such as wiki articles or blogs, using the `<@liferay_ui["asset-display"]` taglib.

```
<@liferay_ui["asset-display"]
  className="JAVA_CLASS_NAME"
  classPK="CLASS PK (RESOURCE PK) OF ASSET"
  template="full_content"
/>
```

- The `<@liferay_ui["asset-display"]` taglib requires the following:
  - **Class Name**: The Java Class Name of the asset
    - This would be the content type, such as blogs or documents
  - **Class PK**: The Primary Key id of the specific asset to display
    - This would be the specific blog or document you want to display
  - **Template**: This identifies the template used to display the asset

EMBEDDING WITH THE RIGHT TAGLIB

- In the past, the only option was to embed applications themselves.
- With these taglibs, you can choose to embed applications, specific web content, or any other asset you'd like on display.
- This gives you the flexibility to choose the option that best fits your requirements.

Notes: