Plugin Online Help

The plugin online help service was removed from XperienCentral in version R39.

In This Topic

Each plugin may provide its own online help. This online help can be written in either XHTML format or a JSP XML document that generates XHTML. All online help files provided by plugins are collected and merged to a single help application by the online help plugin. As a result each element or panel in a plugin that provides online help is able to show a context sensitive help button. When the help button is clicked, the online help plugin will open the help panel and navigate to the relevant help topic in the merged help file.

Providing proper online help in your plugin is required for certain component types, as described in the guidelines G108 and G147.

Enabling Online Help in a Plugin

The online help plugin is a plugin that is part of the XperienCentral platform and is responsible for collecting and displaying the online help files provided by all plugins that provide help. The online help plugin consists of the online help panel, an `OnlineHelpProvider` service interface and a online help listener service. The listener service listens to all services being registered that implement the `OnlineHelpProvider` service interface. If a plugin containing a service that implements this interface becomes available, the listener will receive a notification and retrieve the online help file from that plugin.

To enable online help in a plugin a service must be created that implements the `OnlineHelpProvider` service interface. Define a service component in the Activator of the plugin as shown below.

```
ServiceComponentDefinitionImpl definition = new
ServiceComponentDefinitionImpl(false);

definition.setId(WCB_HELPPROVIDER_ID);
definition.setName(WCB_HELPPROVIDER_NAME);
definition.setDescription("Implements the " + WCB_BUNDLE_ID + " online help provider service");
definition.setTypeId(ServiceComponentType.class.getName());
definition.setProperties(new Hashtable<String, String>());
definition.setImplementationClassName(OnlineHelpProviderImpl.class.getName());
definition.setInterfaceClassNames(new String[]{OnlineHelpProvider.class.getName()});
definition.setDependencies(new ComponentDependency[0] );
```

The plugin must provide the implementation class of the `OnlineHelpProvider` interface as well. This implementation must also implement `ServiceComponent`. The easiest way to do so is by extending `SimpleServiceComponent`, as illustrated below.

```
public class OnlineHelpProviderImpl extends SimpleServiceComponent implements OnlineHelpProvider {
}
```

⚠ Since the `OnlineHelpProvider` interface does not define any methods, it is not necessary to define any methods in the implementation. The implementation of the service itself can be empty.

Back to Top

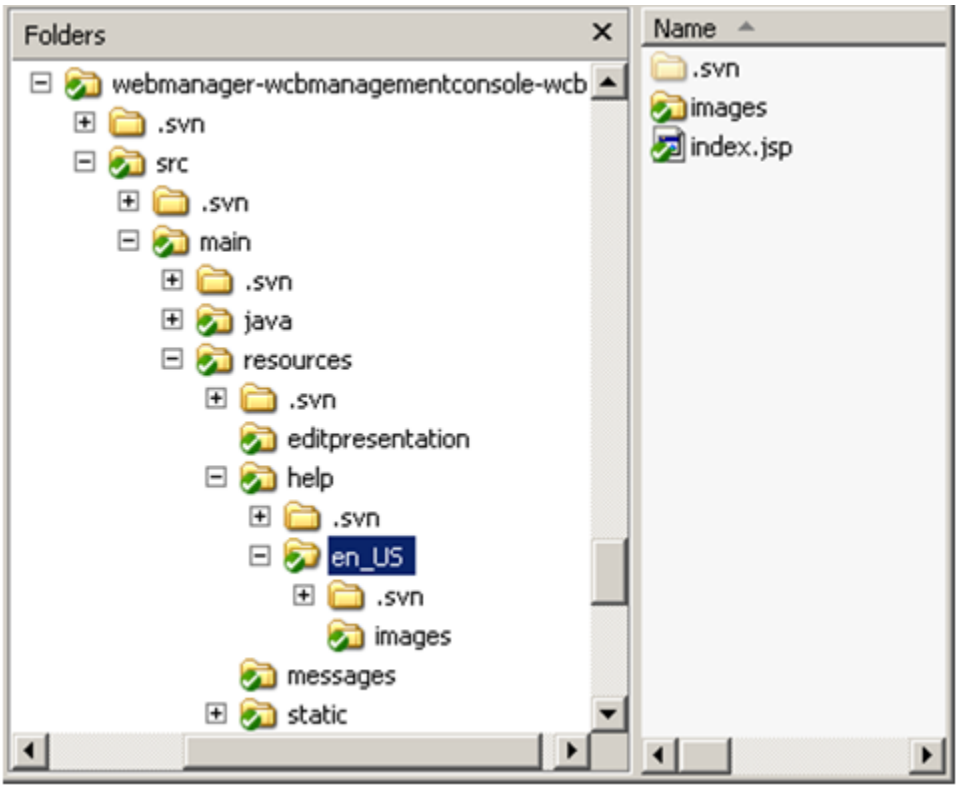
Providing the Online Help Files

The online help files should be written in XHTML format or in an XML JSP Document format that generates XHTML. Badly formed XHTML will cause the help to be undisplayable. You should always validate the XHTML using a validator such as http://validator.w3.org/. A stylesheet for rendering the online help file is contained in the online help plugin and cannot be customized. online help files are written in a particular locale. The online help framework supports providing online help in multiple locales.

The online help files of a plugin must be located in the `/help` directory of the plugin's resources. Typically the help file sources will be located in `/src/main/resources/help`. This is a fixed location and cannot be configured. Within that help directory, a subdirectory must be created for each supported locale using the Java i8N notation. Some examples of this notation are:

Locale Identifier	Description
`en_US`	English (U.S)
`en_GB`	English (United Kingdom)
`fr_FR`	French (France)
`de_DE`	German (Germany)
`nl_NL`	Dutch (Netherlands)
`es_ES`	Spanish (Spain)

Within that subdirectory either a file called `index.html` or `index.jsp` should exist. If both files are present, `index.jsp` will be used. This file represents the online help file in that particular locale. The help directory may contain all kinds of resources used in the help file like images or ZIP files. These can be referred to by the XHTML or XML JSP document. Paths must be relative to the location of the XHTML or XML JSP Document file. The following is an example online help directory structure:



A plugin contains only one single help file for each locale. It is not possible to provide multiple separate help files for the individual components contained by the plugin.

Back to Top

Writing the Online Help Files

online help files should be written in XHTML or a XML JSP Document that generates XHTML. The online help plugin generates a table of contents from all online help files provided by all online help-enabled plugins. The table of contents offers the end-user a way to navigate through the available help topics.

This table of contents is generated by parsing the XHTML or XML JSP Document file. Three special tags contained by that XHTML make up the table of contents:

- The `<title>` attribute in the `<head>` becomes the first level navigation
- The `<h1>` tag in the `<body>` becomes the second level navigation
- The `<h2>` tag in the `<body>` becomes the third level navigation

To properly enable navigation through the table of contents these tags must be provided with anchors. Within the title attribute of the header the ID must be provided that indicates the ID of the anchor. For example:

```
<head>
        <title id="title">Plugin Management Console</title>
        ...
</head>
```

Within the body of the document an anchor should be added with a name attribute to the text contained by the `h1` and `h2` tags that hold an identifier that is unique within that online help file. For example:

```
<body>
        <h1><a name="introduction">Introduction</a></h1>
        <p>Text here...</p>
        <h2><a name="configuration">Configuration</a></h2>
        <p>Text here...</p>
</body>
```

The IDs in all translations of the help file should match. If anchor IDs are different per language, the context sensitive help will not work properly.

Including Images

Take the next steps to include an image to the help:

- Create an image and place it in the images subfolder of the help document;
- Add an `image` tag in the help file: `<p><img src="images/import-users.png" alt="Import users" /></p>`

> ⚠ If you don't add the `<p> … </p>` around the image tag, the image will be placed inline with the text. If you want to add some whitespace above and underneath the image, use `<p> … </p>` around the `image` tag.

Creating a Reference to a Character in an Image

In some images a special arrow with a character inside it is used to point out the interesting part(s) within the image. In the text surrounding the image it's useful to refer to this arrow by referring to this character. To add special formatting, do so using HTML tags. For example: `<span class="imageletterreference">A</span>`

Creating a Link to an External URL

When creating a link to an external website, use the `externallink` class to ensure the link opens in a new window:

```
<a href="http://en.wikipedia.org/wiki/Role-based_access_control" class="externallink">Wikipedia RBAC page</a>
```

Creating a Table

The following HTML shows how to create a table:

```
<dl>
        <dt>Permission category</dt>
        <dd>A permission category is a collection of permissions that belong to one and the same application
component. For example:<p />
                <table>
                        <tr>
                                <th>Application component</th>
                                <th>Permission category</th>
                                <th>Permissions</th>
                        </tr>
                        <tr>
                                <td>FAQ</td>
                                <td>FAQ</td>
                                <td>Maintain FAQ sets<br />
                                Maintain FAQ questions<br />
                                Insert, edit and delete FAQ elements
                                </td>
                        </tr>
                </table>
        </dd>
</dl>
```

Back to Top

Adding the Help Button to a Panel

By default, the online help is displayed in the online help panel, but there is no relation with the panel for which the online help was written. To do so a help button must be added to the panel. This must be done in the `configurePanel()` method of the panel implementation class. This class typically extends `PanelBase`.

To add the help button to the panel, simply create an additional PanelButton of type `BUTTONTYPE.HELP` for it. For example:

```
public void configurePanel() {
  PanelButton close = new PanelButtonImpl(PanelButton.BUTTONTYPE.CLOSE);
  PanelButton apply = new PanelButtonImpl(PanelButton.BUTTONTYPE.APPLY);
  PanelButton help = new PanelButtonImpl(PanelButton.BUTTONTYPE.HELP);

  PanelButton[] panelButtons = {help, apply, close};
  setButtons(panelButtons);
}
```

The panel will now show an additional help button which opens the help dialog when clicked. However, the button is still not context sensitive, so when the online help panel is opened, it will not navigate to the relevant topic within the online help. To enable context sensitive help, the context of the help that should be displayed must be passed to the help button.

To do so the proper action must be set on the help button using the `setAction` method. The syntax of this action must be: `<plugin ID>/<Anchor id>`. So for example if the ID of a plugin is defined statically in the Activator and the anchor ID is "overview" then the action would be set as follows:

```
helpButton.setAction(Activator.WCB_BUNDLE_ID + "/overview");
```

Since the action to set depends on the context within the panel, setting this property must be done in a way that the proper action is set when the user is able to click on the help button. For example, if the context depends on the selected tab within the panel, we can make the online help context sensitive to the selected tab by overriding the `showForm` of the controller as follows:

```
public ModelAndView showForm(HttpServletRequest request, HttpServletResponse response, BindException errors,
Map controlModel)
        throws Exception {
        ModelAndView modelAndView =  super.showForm(request, response, errors, controlModel);

        // Set the help context
        String selectedTabId = myPanel.getTabset().getSelectedTabId();
        if (selectedTabId == null || selectedTabId.equals(OVERVIEW_TAB_ID)) {
            String action = Activator.WCB_BUNDLE_ID + "/overview"; myHelpButton().setAction(action);
            }
            else {
                        String action = Activator.WCB_BUNDLE_ID + "/errorlog"; myHelpButton().setAction(action);
            }
                return modelAndView;
}
```

Back to Top

Adding the Help Button to an Element

To add a help button to an element, simply implement the `getHelpTopicsId` method in the form backing object of the element. The method must return the ID of the help topic that has to be displayed when the help button is clicked. This is typically the class that extends `ElementFBO`. For example, to enable context sensitive help in a custom element plugin, add the following method to the implementation of the form backing object:

```
public class
CustomElementFBO extends ElementFBO {
        ...
        public String getHelpTopicsId() {
                return Activator.OH_ANCHOR_INTRODUCTION;
        }
}
```


In the above example, the ID of the actual help topic is defined statically (as `public static final String`) in the Activator. This is the proper location for such definitions since the Activator usually defines also namespaces, bundle and component IDs.

When the user of XperienCentral activates the help of the element, the value of `Activator.OH_ANCHOR_INTRODUCTION` is looked up and returned by the method. Suppose the `Activator.java` class contains the next definition:

```
public static final String OH_ANCHOR_INTRODUCTION = "introduction";
```

Then the help file must contain an anchor with the name “introduction”:

```
<h1><a name="introduction">Introduction</a></h1>
```

Back to Top

Adding the Help Button to a Custom Media Item

To add a help button to the custom metadata part of a custom media item, simply add the `getHelpTopicsId` method to the form backing object of the custom media item. The method must return the ID of the help topic that has to be displayed when the help button is clicked. This is typically the class that extends `MediaItemVersionFBO`. For example, to enable context sensitive help in a custom media item plugin, the following method is added to the implementation of the form backing object:

```
public class CustomMediaItemVersionFBO extends MediaItemArticleVersionFBO {
        ...
        public String getHelpTopicsId() {
            return Activator.OH_ANCHOR_INTRODUCTION;
        }
}
```

In this example the ID of the actual help topic is defined statically (as `public static final String`) in the Activator. This is the proper location for such definitions since the Activator usually also defines namespaces, bundle and component IDs.

When the user of XperienCentral activates the help of the custom meta data in the media item, the value of `Activator.OH_ANCHOR_INTRODUCTION` is looked up and returned by the method. Suppose the `Activator.java` class contains the next definition:

```
public static final String OH_ANCHOR_INTRODUCTION = "introduction";
```

The help file must contain an anchor with the name “introduction”:

```
<h1><a name="introduction">Introduction</a></h1>
```