CHAPTER 3

Extending the Administration Console

You can extend the Administration Console to provide a graphical user interface for administering your add-on component. You can use any of the user interface features of the Administration Console, such as tree nodes, links on the Common Tasks page, tabs and sub-tabs, property sheets, and JavaServer Faces (JSF) pages. ~~sub-tabs, and JavaServer Faces (JSF) pages that serve as property sheets.~~ ~~You provide a single console provider for your add-on component, and you specify an integration point for each user interface feature in a configuration file~~ Your add-on component implements a marker interface and provides a configuration file which describes your integration points for all your user interface customizations. The Console Add-On Component Service for the Administration Console locates all the console providers and creates a user interface that incorporates all of them.

This chapter refers to a simple example called `console-sample-ip` that illustrates how to provide Administration Console features for a hypothetical add-on component. For instructions on how to obtain and use this example, go to (`http://wiki.glassfish.java.net/Wiki.jsp?page=V3SampleIpProject`). When you check out the code, it is placed in a directory named `glassfish-samples/v3/plugin/adminconsole/console-sample-ip/` in your current directory. In this chapter, path names for the example files are relative to this directory.

The following topics are addressed here:

- "Administration Console Architecture" on page 32
- "About Administration Console Templates" on page 33
- "About Integration Points" on page 34
- "Specifying the ID of an Add-On Component" on page 34
- "Adding Functionality to the Administration Console" on page 35
- "Adding Internationalization Support" on page 46
- "Changing the Theme or Brand of the Administration Console" on page 47
- "Creating a New Integration Point Type" on page 49

Administration Console Architecture

The Administration Console is a web application that is composed of OSGi bundles. These bundles provide all the features of the Administration Console, such as the Web Application, Update Center, and Security content. ~~support all the features of the Administration Console, such as the Update Center, JDBC resources, and security~~. To provide support for your add-on component, create your own OSGi bundle that implements the parts of the user interface that you need. Place your bundle in the `modules` directory of your Enterprise Server installation, along with the other Administration Console bundles. ~~By convention, the Administration Console bundle names begin with "`console-`"~~.

To learn how to package the Administration Console features for an add-on component, go to the `modules` directory of your Enterprise Server installation and examine the contents of the files named `console-`*componentname*`-plugin.jar`. Place the `console-sample-ip` project bundle in the same place in order to deploy it and examine the changes it makes to the Administration Console.

The Administration Console includes a Console Add-On Component Service. The Console Add-On Component Service is an HK2 service that acts as a façade to all theAdministration Console add-on components. The Console Add-On Component Service queries the various console providers for integration points so that it can perform the actions needed for the integration (adding a tree node or a new tab, for example).

For details about the Hundred Kilobytes Kernel (HK2), see "Hundred-Kilobyte Kernel" on page 18 and "HK2 Component Model" on page 23.

Remark 3–3 Reviewer **We use the term "add-on component" instead of "plugin" in our documentation. Is it okay to rename this service here? I know the API interface is called** `ConsolePluginService`**.** See p32 Remark 3-3

See p32 paragraph 5 comment.

~~To create a console provider, write a Java class that implements the `org.glassfish.api.admingui.ConsoleProvider` interface. To enable your implementation to perform actions for the HK2 runtime, annotate the declaration of your implementation with the `@Service` annotation. The console provider allows the Enterprise Server runtime to efficiently locate all add-on components. It also defines the interface in which add-on components define their integration points. Each console provider must implement the following method:~~

~~`public URL getConfiguration()`~~

You may want to insert the first diagram in section 4.1.1 of the one pager http://wiki.glassfish.java.net/Wiki.jsp?page=GFV3PluggabilityOnePager

Implementing a Console Provider

~~Implement the `org.glassfish.api.admingui.ConsoleProvider` interface to fulfill the contract required by `@service` implementations that display information in the Enterprise Server Administration Console.~~

glassfish.api.admingui.ConsoleProvider
The ~~ConsoleProvider~~ interface has one required method, getConfiguration. ~~The getConfiguration method retrieves and processes the contents of the console-config.xml file~~. The getConfiguration method returns the location of this file as a java.net.URL. If getConfiguration returns null, the default location, META-INF/admingui/console-config.xml, is used. The console-config.xml file is described in "About Integration Points" on page 34.

To implement the console provider for your add-on component, write a Java class that is similar to the following example.

EXAMPLE 3–1 Example ConsoleProvider Implementation

This example shows a simple implementation of the ConsoleProvider interface:

```
package org.glassfish.admingui.plugin;

import org.glassfish.api.admingui.ConsoleProvider;
import org.jvnet.hk2.annotations.Service;

import java.net.URL;

@Service
public class SamplePlugin implements ConsoleProvider {

    public URL getConfiguration() { return null; }
}
```

In the example, you can find this code in the file project/src/main/java/org/glassfish/admingui/plugin/SamplePlugin.java.

Remark 3–4 Reviewer **Under what circumstances would it not be a no-op? For the sample this seems to work just fine. What kind of URL might it return and what would be the use of the URL? It seems as if an empty** getConfiguration **method is sufficient to retrieve and process the contents of the** console-config.xml **file.** See p.33 Remark 3-4

About Administration Console Templates

I would suggest moving this section right before "Adding a Page to the Administration Console"
Enterprise Server includes a set of templates that make it easier to create JSF pages for your add-on component. These templates use Templating for JavaServer Faces Technology (also known as JSFTemplating), which is described in https://jsftemplating.dev.java.net/.

For more details on the templates, see Appendix B, "Template Reference."

About Integration Points

The integration points for your ~~add-on component~~ Add-On Component are the individual Administration Console user interface features that your ~~component requires~~ Add-On Component wishes to extend. You can implement the following kinds of integration points:

- Nodes in the navigation tree
- Elements on the Common Tasks page of the Administration Console
- JSF pages ~~where users can specify component properties~~
- Tabs and sub-tabs

Specify all the integration points in a file named `console-config.xml`. In the example, this file is in the directory `project/src/main/resources/META-INF/admingui/`. The following sections describe how to create this file.

In addition, create JSF files that contain JSF code fragments to implement the integration points. In the example, these files are in the directory `project/src/main/resources/`. ~~These files typically reference another JSF file that implements a page with a property sheet.~~ The content of these files depends on the integration point you are implementing (see integration point wiki for details). The following sections describe how to create these JSF files.

For reference information on integration points, see Appendix A, “Integration Point Reference.”

Specifying the ID of an Add-On Component

The `console-config.xml` file consists of a `console-config` element that encloses a series of `integration-point` elements. The `console-config` element has one attribute, `id`, which specifies a name or ID value for the ~~add-on component~~ Add-On Component.

In the example, the element is declared as follows:

```
<console-config id="sample">
    ...
</console-config>
```

You will also specify this ID value when you ~~construct the URL and image URL for your JSF pages~~ construct URLs to images, resources and pages in your Add-On Component. See“Adding a Node to the Navigation Tree” on page 36 for an example.

Remark 3–5 Reviewer **Please provide more details about component IDs if necessary.**

And example URL to an image named my.gif might be:

<sun:image url="/resource/sample/images/my.gif" />

"/resource" is required for all resource URLs so that it can located properly.
"sample" is the Add-On Component id. It is very important to choose a unique ID value.
"images" is a folder inside the root of their Add-On Component jar file.

Adding Functionality to the Administration Console

The integration-point elements in the console-config.xml file specify attributes for the user interface features you choose to implement. The example file provides examples of all the possible kinds of integration points. Your own add-on component can use some or all of them.

, at the time of this writing.

For complete details about the Administration Console user interface features, see the API documentation for Project Woodstock at http://webdev2.sun.com/woodstock-tlddocs/index.html.

This Woodstock paragraph should move to one or more of the "Creating a JSF File for..." sections.

For each integration-point element, specify the following attributes.

id
: An identifier for the integration point.

parentId
: The ID of the integration point's parent.

type
: The type of the integration point.

priority
: A numeric value that specifies the relative ordering of integration points with the same parentId. A lower number specifies a higher priority (for example, 100 represents a higher priority than 400). You may need to experiment in order to place the integration point where you want it. This attribute is optional.

, this is typically a JSF file.

content
: The ~~JSF file that contains the~~ content for the integration point. In the example, you can find the JSF files in the directory project/src/main/resources/.

Note – The order in which these attributes are specified does not matter, and in the example console-config.xml file the order varies. To improve readability, this chapter uses the same order throughout.

The following topics are addressed here:

- "Adding a Node to the Navigation Tree" on page 36
- "Adding Tabs to a Page" on page 38
- "Adding a Task to the Common Tasks Page" on page 40
- "Adding a Task Group to the Common Tasks Page" on page 42
- "Adding Items to a Page" on page 44
- "Adding a Page to the Administration Console" on page 46

Adding a Node to the Navigation Tree

You can add a node to the navigation tree, either at the top level or under another node. To add a node, use an integration point of type ~~`treeNode`~~. Use the `parentId` attribute to specify where the new node should be placed, using any of the following values.

This is not correct, check the Integration Point wiki.

`tree`
At the top level

`registration`
Under the Registration node

These are example values, any TreeNode id (including ones added by other plugins) may be used.

`applicationServer`
Under the Application Server node

`applications`
Under the Applications node

`resources`
Under the Resources node

`configuration`
Under the Configuration node

Remark 3–6 Reviewer **What happens if you don't specify a `parentId`? One of the integration points in `v3/admingui/web/src/main/resources/META-INF/admingui/console-config.xml` doesn't have one (the one with the id `webAppLink`).** The new content will be added to the root of the integration point. In this case the Tree root.

EXAMPLE 3–2 Example Tree Node Implementation Point

For example, the following `integration-point` element uses a `parentId` of `tree` to place the new node at the top level.

```
<integration-point
        id="sampleNode"
        parentId="tree"
        type="org.glassfish.admingui:treeNode"
        priority="200"
        content="sampleNode.jsf"
/>
```

This example specifies the following values in addition to the `parentId`:

- The `id` value, `sampleNode`, specifies the integration point ID.
- The `type` value, `org.glassfish.admingui:treeNode`, specifies the integration point type as a tree node.
- The `priority` value, `200`, specifies the order of the node on the tree. ~~This value is optional.~~
- The `content` value, `sampleNode.jsf`, specifies the JSF file that displays the node.

The example `console-config.xml` file provides other examples of tree nodes under the Resources and Configuration nodes.

Creating a JSF File for Your Node

A JSF file for a tree node uses the JSFTemplating ~~custom~~ tag `sun:treeNode`. This tag provides all the capabilities of the Project Woodstock tag `webuijsf:treeNode`.

EXAMPLE 3–3 Example JSF File for a Tree Node

In the example, the `sampleNode.jsf` file has the following content:

```
<sun:treeNode id="treeNode1"
        text="SampleTop"
        url="/sample/page/testPage.jsf?name=SampleTop"
        target="main"
        imageURL="resource/sample/images/sample.png"
        >
        <sun:treeNode id="treeNodeBB"
        text="SampleBB"
        url="/sample/page/testPage.jsf?name=SampleBB"
        target="main"
        imageURL="resource/sample/images/sample.png"
        />
</sun:treeNode>
```

This file uses the `sun:treenode` ~~custom~~ tag to specify both a top-level tree node and another node nested beneath it. In your own JSF pages, use the attributes of this ~~custom~~ tag as follows:

`id`
: Use this attribute to specify a unique identifier for the tree node.

`text`
: Use this attribute to specify the node name that appears in the tree.

`url`
: Use this attribute to specify the location of the JSF page that appears when you click the node. In the example, most of the integration points use a very simple JSF page called `testPage.jsf`, which is in the `src/main/resources/page/` directory. Specify the integration point `id` value as the root of the URL; in this case, it is `sample` (see "Specifying the ID of an Add-On Component" on page 34). The rest of the URL is relative to the `src/main/resources/` directory, where `sampleNode.jsf` resides.

  The `url` tag in this example passes a `name` parameter to the JSF page.

`target`
: Use this attribute to specify the frame in which to display the JSF page specified by the `url` tag. Normally, the value is `main`.

`imageURL`

Use this attribute to specify the location of a graphic to display next to the node name. In the example, the graphic is always `sample.png`, which is in the `src/main/resources/images/` directory. The URL for the deployed `images` directory is relative to `resource/`*idval*`/`, where *idval* is the integration point `id` value (see "Specifying the ID of an Add-On Component" on page 34).

This would be a good spot to refer to the Woodstock documentation.

Adding Tabs to a Page

You can add a tab to an existing tab set, or you can create a tab set for your own page. One example of adding a ~~To add a~~ tab or tab set, is to use an integration point of type `serverInstTab`. You can also create sub-tabs. Once again, the `parentId` element specifies where to place the tab or tab set.

EXAMPLE 3–4 Example Tab Integration Point

In the example, the following `integration-point` element adds a new tab on the main Application Server page of the Administration Console:

```
<integration-point
    id="sampleTab"
    parentId="serverInstTabs"
    type="org.glassfish.admingui:serverInstTab"
    priority="500"
    content="sampleTab.jsf"
/>
```

This example specifies the following values:

- The `id` value, `sampleTab`, specifies the integration point ID.
- The `parentId` value, `serverInstTabs`, specifies the tab set associated with the server instance. The Application Server page is the only one of the default Administration Console pages that has a tab set.
- The `type` value, `org.glassfish.admingui:serverInstTab`, specifies the integration point type as a tab associated with the server instance.
- The `priority` value, `500`, specifies the order of the tab within the tab set. This value is optional.
- The `content` value, `sampleTab.jsf`, specifies the JSF file that displays the tab.

EXAMPLE 3–5 Example Tab Set Integration Points

The following `integration-point` elements add a new tab with two sub-tabs, also on the main Application Server page of the Administration Console:

EXAMPLE 3–5 Example Tab Set Integration Points *(Continued)*

```
    <integration-point
        id="sampleTabWithSubTab"
        parentId="serverInstTabs"
        type="org.glassfish.admingui:serverInstTab"
        priority="300"
        content="sampleTabWithSubTab.jsf"
    />

    <integration-point
        id="sampleSubTab1"
        parentId="sampleTabWithSubTab"
        type="org.glassfish.admingui:serverInstTab"
        priority="300"
        content="sampleSubTab1.jsf"
    />
    <integration-point
        id="sampleSubTab2"
        parentId="sampleTabWithSubTab"
        type="org.glassfish.admingui:serverInstTab"
        priority="400"
        content="sampleSubTab2.jsf"
    />
```

These examples specify the following values:

- The `id` values, `sampleTabWithSubTab`, `sampleSubTab1`, and `sampleSubTab2`, specify the integration point IDs for the tab and its sub-tabs.
- The `parentId` of the new tab, `serverInstTabs`, specifies the tab set associated with the server instance. The `parentId` of the two sub-tabs, `sampleTabWithSubTab`, is the `id` value of this new tab.
- The `type` value, `org.glassfish.admingui:serverInstTab`, specifies the integration point type for all the tabs as a tab associated with the server instance.
- The `priority` values specify the order of the tabs within the tab set. This value is optional. In this case, the priority value for `sampleTabWithSubTab` is `300`, which is higher than the value for `sampleTab`. That means that `sampleTabWithSubTab` appears to the left of `sampleTab` in the Administration Console. The priority values for `sampleSubTab1` and `sampleSubTab2` are `300` and `400` respectively, so `sampleSubTab1` appears to the left of `sampleSubTab2`.
- The `content` values, `sampleTabWithSubTab.jsf`, `sampleSubTab1.jsf`, and `sampleSubTab2.jsf`, specify the JSF files that display the tabs.

Creating JSF Files for Your Tabs

A JSF file for a tab uses the JSFTemplating ~~custom~~ tag `sun:tab`. This tag provides all the capabilities of the Project Woodstock tag `webuijsf:tab`.

EXAMPLE 3–6 Example JSF File for a Tab

In the example, the `sampleTab.jsf` file has the following content:

```
<sun:tab id="sampleTab" immediate="true" text="Sample First Tab">
    <!command
        setSessionAttribute(key="serverInstTabs" value="sampleTab");
        redirect(page="#{request.contextPath}/page/
tabPage.jsf?name=Sample%20First%20Tab");
    />
</sun:tab>
```

Note – In the actual file there are no line breaks in the `redirect` value.

In your own JSF pages, use the attributes of this ~~custom~~ tag as follows:

`id`
Use this attribute to specify a unique identifier for the tab, in this case `sampleTab`.

`immediate`
Use this attribute to specify that event handling for this component should be handled immediately (in the Apply Request Values phase) rather than waiting until the Invoke Application phase.

`text`
Use this attribute to specify the tab name that appears in the tab set.

The JSFTemplating page displays tab content differently from the way the page for a node displays node content. It invokes two handlers for the `command` event, `setSessionAttribute` and `redirect`. The `redirect` handler has the same effect for a tab that the `url` attribute has for a node. It invokes a simple JSF page called `tabPage.jsf`, in the `src/main/resources/page/` directory, passing the text “Sample First Tab” to the page in a `name` parameter.

The `sampleSubTab1.jsf` and `sampleSubTab2.jsf` files are almost identical to `sampleTab.jsf`.

Woodstock TLD reference here too?

Adding a Task to the Common Tasks Page

You can add either a single task or a group of tasks to the Common Tasks page of the Administration Console. To add a task or task group, use an integration point of type `commonTask`. You can add a single task to either the Deployment task group or the Monitoring task group, but not to any other group.

Is this true? It might be -- I don't know, but I am suprised if it is.

Remark 3–7 Reviewer **Is this true? Can you add a task to other task groups?**

See “Adding a Task Group to the Common Tasks Page” on page 42 for information on adding a task group.

EXAMPLE 3–7 Example Task Integration Point

In the example `console-config.xml` file, the following `integration-point` element adds a task to the Deployment task group:

```
<integration-point
        id="sampleCommonTask"
        parentId="deployment"
        type="org.glassfish.admingui:commonTask"
        priority="200"
        content="sampleCommonTask.jsf"
/>
```

This example specifies the following values:

- The `id` value, `sampleCommonTask`, specifies the integration point ID.
- The `parentId` value, `deployment`, specifies that the task is to be placed in the Deployment task group.

  Specify a value of `monitoring` to place the task in the Monitoring task group.
- The `type` value, `org.glassfish.admingui:commonTask`, specifies the integration point type as a common task.
- The `priority` value, `200`, specifies the order of the task within the task group.
- The `content` value, `sampleCommonTask.jsf`, specifies the JSF file that displays the task.

Creating a JSF File for Your Task

A JSF file for a task uses the JSFTemplating ~~custom~~ tag `sun:commonTask`. This tag provides all the capabilities of the Project Woodstock tag `webuijsf:commonTask`.

EXAMPLE 3–8 Example JSF File for a Task

In the example, the `sampleCommonTask.jsf` file has the following content:

```
<sun:commonTask
        text="Sample Application Page"
        toolTip="Sample Application Page"
        infoLinkUrl="/com_sun_webui_jsf/help/
helpwindow.jsf?&windowTitle=Help+Window&helpFile=applications.html"
        onClick="admingui.nav.selectTreeNodeById('form:tree:deployment:ejb');
        parent.location='#{facesContext.externalContext.requestContextPath}/sample/
```

EXAMPLE 3–8 Example JSF File for a Task *(Continued)*

```
page/testPage.jsf?name=Sample%20Application%20Page'; return false;">
</sun:commonTask>
```

Note – In the actual file there are no line breaks in the `infoLinkUrl` and `parent.location` attribute values.

This file uses the `sun:commonTask` ~~custom~~ tag to specify the task. In your own JSF pages, use the attributes of this custom tag as follows:

- Use the `text` attribute to specify the task name that appears on the Common Tasks page.
- Use the `toolTip` attribute to specify the text that appears when a user mouses over the task name.
- Use the `infoLinkUrl` attribute to specify the URL for the link that is displayed at the bottom of the task's information panel.
- Use the `onClick` attribute to specify scripting code to be executed when a user clicks the task name.
- ~~Use the `parent.location` attribute to specify WHAT?~~

Remark 3–8 Reviewer **What is `parent.location`? I found the others in the Woodstock javadoc for `webuijsf:commonTask`, but not that one.**

That's part of the "onClick" attribute. I think it is really "window.parent.location" (but window is implicit). It causes the browsers "location" to change.

Adding a Task Group to the Common Tasks Page

You can add a new group of tasks to the Common Tasks page to display the most important tasks for your add-on component. To add a task group, use an integration point of type `commonTask`.

EXAMPLE 3–9 Example Task Group Integration Point

In the example `console-config.xml` file, the following `integration-point` element adds a new task group to the Common Tasks page:

```
<integration-point
     id="sampleGroup"
     parentId="commonTasksSection"
     type="org.glassfish.admingui:commonTask"
     priority="500"
     content="sampleTaskGroup.jsf"
 />
```

This example specifies the following values:

- The id value, sampleGroup, specifies the integration point ID.
- The parentId value, commonTasksSection, specifies that the task group is to be placed on the Common Tasks page.
- The type value, org.glassfish.admingui:commonTask, specifies the integration point type as a common task.
- The priority value, 500, specifies the order of the task group on the Common Tasks page. The low value places it at the end of the page.
- The content value, sampleTaskGroup.jsf, specifies the JSF file that displays the task.

Creating a JSF File for Your Task Group

A JSF file for a task group uses the JSFTemplating custom tag sun:commonTasksGroup. This tag provides all the capabilities of the Project Woodstock tag webuijsf:commonTasksGroup.

EXAMPLE 3–10 Example JSF File for a Task Group

In the example, the sampleTaskGroup.jsf file has the following content:

```
<sun:commonTasksGroup title="My Own Sample Group">
    <sun:commonTask
        text="Go To Sample Resource"
        toolTip="Go To Sample Resource"
        infoLinkUrl="/com_sun_webui_jsf/help/
helpwindow.jsf?&windowTitle=Help+Window&helpFile=jdbcconnectionpoolnew1.html"
        onClick="admingui.nav.selectTreeNodeById(’form:tree:resources:treeNode1’);
        parent.location=’#{facesContext.externalContext.requestContextPath}/sample/
page/testPage.jsf?name=name=Sample%20Resource%20Page’; return false;">
    </sun:commonTask>
    <sun:commonTask
        text="Sample Configuration"
        toolTip="Go To Sample Configuration"
        infoLinkUrl="/com_sun_webui_jsf/help/
helpwindow.jsf?&windowTitle=Help+Window&helpFile=jdbcconnectionpoolnew1.html"
        onClick="admingui.nav.selectTreeNodeById(’form:tree:resources:treeNode1’);
        parent.location=’#{facesContext.externalContext.requestContextPath}/sample/
page/testPage.jsf?name=Sample%20Configuration%20Page’; return false;">
    </sun:commonTask>
</sun:commonTasksGroup>
~~/sample/page/testPage.jsf?name=Sample%20Configuration%20Page~~
```

Remark 3–9 Reviewer **Why is the /sample/page/ path placed at the bottom of the file, outside of the XML tags? Is this an error? When I click on one of the tasks in the group I get the following server.log entry: "Oct 14, 2008 2:16:47 PM com.sun.jsftemplating.util.LogUtil info INFO: JSFT0006: WARNING: Failed**

Yes, it appears to be an error.

to set property (rendered) with (null) value. This occured on the component named (sun_commonTask72) of type (com.sun.webui.jsf.component.CommonTask)."

Note – In the actual file there are no line breaks in the `infoLinkUrl` and `parent.location` attribute values.

This file uses the `sun:commonTasksGroup` ~~custom~~ tag to specify the task group, and two `sun:commonTask` tags to specify the tasks in the task group. The `sun:commonTasksGroup` tag has only one attribute, `title`, which specifies the name of the task group.

Adding Items to a Page

You can add an item for your add-on component to an existing top-level page, such as the Configuration page or the Resources page. ~~When a user clicks on this item, it usually has the same effect as clicking on the node for the component~~. To add an item, use an integration point of type `configuration` or `resources`.

EXAMPLE 3–11 Example Resources Page Implementation Point

In the example `console-config.xml` file, the following `integration-point` element adds a new component to the top-level Resources page:

```
<integration-point
        id="sampleResourceLink"
        parentId="propSheetSection"
        type="org.glassfish.admingui:resources"
        priority="100"
        content="sampleResourceLink.jsf"
/>
```

This example specifies the following values:

- The `id` value, `sampleResourceLink`, specifies the integration point ID.
- The `parentId` value, `propSheetSection`, specifies that the item is to be a section of a property sheet on the page.
- The `type` value, `org.glassfish.admingui:resources`, specifies the integration point type as the Resources page.

  To add an item to the Configuration page, specify the `type` value as `org.glassfish.admingui:configuration`.

Remark 3–10 Reviewer **Are there any other possibilities for the type?** Currently, I do not think so. When more are added we will add documentation of additional ones on the wiki.

- The `priority` value, `100`, specifies the order of the item on the Resources page. The high value places it at the top of the page.

- The content value, sampleResourceLink.jsf, specifies the JSF file that displays the item on the Resources page.

Another integration-point element in the console-config.xml file places a similar item on the Configuration page.

Perhaps this should be called "Page Content" vs "Page Item"?

Creating a JSF File for Your Page Item

A JSF file for a page item commonly uses the JSFTemplating ~~custom~~ tag sun:property, to specify an item on a property sheet. This tag provides all the capabilities of the Project Woodstock tag webuijsf:property.

EXAMPLE 3–12 Example JSF File for a Resource Page Item

In the example, the sampleResourceLink.jsf file has the following content:

```
<sun:property>
    <sun:hyperlink
        toolTip="Sample Resource"
        url="/sample/page/testPage.jsf?name=Sample%20Resource%20Page" >
        <sun:image url="/resource/sample/images/sample.png" />
        <sun:staticText text="Sample Resource" />
    </sun:hyperlink>
</sun:property>

<sun:property>
    <sun:hyperlink
        toolTip="Another"
        url="/sample/page/testPage.jsf?name=Another" >
        <sun:image url="/resource/sample/images/sample.png" />
        <sun:staticText text="Another" />
    </sun:hyperlink>
</sun:property>
```

The file specifies two simple properties on the property sheet, one above the other. Each consists of a sun:hyperlink element (a link to a URL). Within each sun:hyperlink element is nested a sun:image element, specifying an image, and a sun:staticText element, specifying the text to be placed next to the image.

Each sun:hyperlink element uses a toolTip attribute and a url attribute. Each url attribute references the testPage.jsf file used elsewhere in the example, specifying different content for the name parameter.

You can use many other kinds of user interface elements within a sun:property element.

Right before this section might be the best place to discuss templates.

Adding a Page to the Administration Console

Your add-on component may require new configuration tasks. In addition to implementing commands that accomplish these tasks (see Chapter 4, “Extending the asadmin Utility”), you can provide property sheets that enable users to configure your component or to perform tasks such as creating and editing resources for it.

EXAMPLE 3–13 Example JSF File for a Property Sheet

Most of the user interface features used in the example reference the file testPage.jsf or (for tabs) the file tabPage.jsf. Both files are in the src/main/resources/page/ directory. The testPage.jsf file looks like this:

```
<!composition template="/templates/propertySheetTemplate.tpl"
              pageTitle="TEST Sample Page Title"
              helpText="Shows you what my page looks like"
              >

    <!define name="properties">
       <sun:property id="prop1" labelAlign="left" noWrap="#{true}"
                     overlapLabel="#{false}" label="page Name:" >
            <sun:staticText text="$pageSession{pageName}" >
                <!beforeCreate
                    getRequestValue(key="name" value=>$page{pageName});
                />
            </sun:staticText>
        </sun:property>
    </define>
</composition>
```

The page uses the composition directive to specify that the page uses the propertySheetTemplate.tpl template and to specify a page title and inline help text. (See “Property Sheet Template” on page 102 for more information about this template and more examples.) The page uses additional directives, events, and custom tags to specify its content.

Adding Internationalization Support

To add internationalization support for your add-on component to the Administration Console, use the i18nBundle attribute of the composition directive for your JSF page to specify the name of the resource bundle.

For example, in the tabPage.jsf file, the composition directive specifies org.glassfish.admingui.core as the bundle name:

```
<!composition template="/templates/propertySheetTemplate.tpl"
              pageTitle="TEST Sample Page Title"
              helpText="Shows you what my page looks like"
              i18nBundle="org.glassfish.admingui.core"
              >
```

The bundle name specifies the package where the resource files are located.

Alternatively, at the top of your page you can do the following:

<!initPage   setResourceBundle(key="yourI18NKey" bundle="bundle.package.BundleName") />

Changing the Theme or Brand of the Administration Console

To change the theme or brand of the Administration Console for your add-on component, use the integration point type `org.glassfish.admingui:customtheme`. This integration point affects the CSS files and images used in the Administration Console.

EXAMPLE 3–14 Example Custom Theme Integration Point

For example, the following integration point specifies a custom theme:

```
<integration-point
        id="myOwnBrand"
        type="org.glassfish.admingui:customtheme"
        priority="2"
        content="myOwnBrand.properties"
/>
```

Place the properties file at the top level of your bundle.

It looks correct. You should check with Ana.

Remark 3–11 Reviewer **Is that right? That's where it is in the `console-branding-plugin.jar` file.**

Additional integration point types also affect the theme or brand of the Administration Console:

`org.glassfish.admingui:masthead`
: Specifies the name and location of the include masthead file, which can be customized with a branding image. This include file will be integrated on the masthead of the Administration Console.

`org.glassfish.admingui:loginimage`
: Specifies the name and location of the include file containing the branding login image code that will be integrated with the login page of the Administration Console.

`org.glassfish.admingui:loginform`
: Specifies the name and location of the include file containing the customized login form code. This code also contains the login background image used for the login page for the Administration Console.

`org.glassfish.admingui:versioninfo`
Specifies the name and location of the include file containing the branding image that will be integrated with the content of the version popup window.

~~`org.glassfish.admingui:upsellframepe`
Specifies the name and location of the include file containing the frame source code that will be used to display or hide content in the bottom frame of the Administration Console for the community distribution.~~

I think the 2 "upsell" types were merged and renamed. We should NOT publish "upsell" anywhere.

~~`org.glassfish.admingui:upsellframeee`
Specifies the name and location of the include file containing the frame source code that will be used to display or hide content in the bottom frame of the Administration Console for the enterprise distribution.~~

EXAMPLE 3–15 Example of Branding Integration Points

For example, you might specify the following integration points. The content for each integration point is defined in an include file.

```
<integration-point
        id="myOwnBrand"
        type="org.glassfish.admingui:masthead"
        priority="80"
        content="branding/masthead.inc"
/>
<integration-point
        id="myOwnBrand"
        type="org.glassfish.admingui:loginimage"
        priority="80"
        content="branding/loginimage.inc"
/>
<integration-point
        id="myOwnBrand"
        type="org.glassfish.admingui:loginform"
        priority="80"
        content="branding/loginform.inc"
/>
<integration-point
        id="myOwnBrand"
        type="org.glassfish.admingui:versioninfo"
        priority="80"
        content="branding/versioninfo.inc"
/>
<integration-point
        id="myOwnBrand"
        type="org.glassfish.admingui:upsellframepe"
        priority="80"
        content="branding/upsellpe.inc"
```

I don't recommend using the same "id" for everything.

EXAMPLE 3–15 Example of Branding Integration Points *(Continued)*

```
/>
<integration-point
        id="myOwnBrand"
        type="org.glassfish.admingui:upsellframeee"
        priority="80"
        content="branding/upsellee.inc"
/>
```

To provide your own CSS and images to modify the global look and feel of the entire application ~~(not just the Administration Console~~), use the theming feature of Project Woodstock (see `https://woodstock.dev.java.net/docs/specs/ThemeFS.html`).

Creating a New Integration Point Type

~~If your add-on component requires user interface features that the current Administration Console does not support, you can create a new integration point type to support your feature~~.

Remark 3–12 Reviewer **Information needed on this topic.**

See p49 Remark 3-12