Sun Java System Portal Server 7.1 Developer Sample Guide

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–5026
June 2006

070314@17039

Contents

Figures

Tables

Examples

Preface

The *Sun Java System Portal Server Developer Sample Guide* includes sample customizations and detailed instructions for customizing the Sun Java™ System Portal Server software Developer Sample portal.

Who Should Use This Book

The *Developer Sample Guide* is intended for use by administrators and other individuals responsible for customizing the Developer Sample desktop.

Audience for this guide should already understand the following technologies:

- Lightweight Directory Access Protocol (LDAP)
- Java technology
- JavaServer Pages™ (JSP™) technology
- Hypertext Transfer Protocol (HTTP)
- Hypertext Markup Language (HTML)
- Extensible Markup Language (XML)

Related Books

The `http://docs.sun.com/coll/1483.1` SM web site enables you to access Sun technical documentation online. You can browse the archive or search for a specific book title or subject.

Related Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Documentation, Support, and Training

The Sun web site provides information about the following additional resources:

- Documentation (`http://www.sun.com/documentation/`)
- Support (`http://www.sun.com/support/`)
- Training (`http://www.sun.com/training/`)

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–1 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX® system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	Prompt
C shell	`machine_name%`
C shell for superuser	`machine_name#`
Bourne shell and Korn shell	`$`
Bourne shell and Korn shell for superuser	`#`

Default Paths and File Names

The following table describes the default paths and file names used in this book.

TABLE P–3 Default Paths and File Names

Term	Description
PortalServer-base	Represents the base installation directory for Sun Java System Portal Server software. The Portal Server software default base installation and product directory depends on your specific platform: Solaris™ systems:`/opt/SUNWportal` Linux systems: `/opt/sun/portal`
AccessManager-base	Represents the base installation directory for Sun Java System Identity Server software. The Identity Server software default base installation and product directory depends on your specific platform: Solaris systems: `/opt/SUNWam`
DirectoryServer-base	Represents the base installation directory for Sun Java System Directory Server software. The Directory Server software default base installation is `/opt/SUNWdsee`.
ApplicationServer-base	Represents the base installation directory for Sun Java System Application Server software. The Application Server software default base installation is `/opt/SUNWappserver`.
WebServer-base	Represents the base installation directory for Sun Java System Web Server software. The Web Server software default base installation is `/opt/SUNWwbsvr7`.

TABLE P–3 Default Paths and File Names *(Continued)*

Term	Description
PortalServer-DataDir	Represents the directory where JavaServer™ Pages (JSP), templates and property files, and tag libraries are installed. By default, this is: ■ `/var/opt/SUNWportal` on Solaris ■ `/var/opt/sun/portal` on Linux

◆ ◆ ◆ CHAPTER 1

Introduction to Customizing the Developer Sample Portal Desktop

This chapter provides an introduction to customizing the Sun Java™ System Portal Server software Desktop. It describes the different kinds of customizations and who should make those customizations. This chapter also provides an overview of the display profile, sample desktops included with the product, and how you create and deploy a new desktop and provider.

This chapter contains the following sections:

Types of Desktop Customizations

The Developer Sample can be customized by end users, administrators, and developers. Though this guide covers only administrator customizations, it includes an overview of end user and developer customizations, and where to go for more information on those customizations. This guide refers to administrator customizations such as changes and modifications made to the sample portal that involve modifications to the display profile, JavaServer™ Pages and template files, search provider, and online help.

End User Customizations

End users can customize the sample portal desktop in the following ways.

- Setting the channel time out
- Selecting column layout from the Layout page
- Moving channels up and down, as well as side to side

- Arranging channels by width
- Re-sizing the channel window
- Adding and removing certain channels from the Content page
- Customizing channels by using the channel Edit page
- Selecting theme from a set of preset themes or customize the theme by changing color scheme and font type for the Desktop channels
- Creating, removing, and editing tabs

Users customize channels by using the Edit Channel icon for a particular channel (as long as the administrator has made it available). Users customize the look and feel of the Desktop through the Themes page. See the Portal Server software End User Desktop Online Help for more information.

Users can also configure:

- Time zone, language, and password by using the User Info channel Edit page.
- Personal information by using the Sun Java System Access Manager software administration console.

Developer Customizations

The Portal Server software developers can customize the desktop by creating:

- Provider classes
- Desktop templates
- Tag libraries
- JavaServer Pages™

Developers can also use the Provider Application Programming Interface (PAPI) and the Search service APIs to extend the Portal Server software. See the *Sun Java System Portal Server 7.1 Developer's Guide* for more information.

Administrator Customizations

The Portal Server software administrators can customize the Desktop by:

- Using supplied providers to define additional content channels.
- Creating and customizing the display profile, which involves creating or modifying provider, channel, and container channel objects. When you modify the display profile, you use the appropriate XML tag definitions for providers, channels, and container channels.
- Create new preset themes in the display profile.
- Using supplied JavaServer Pages and template files to modify the user interface.

- Customizing the search provider.
- Customizing the Desktop end user online help.

By performing these customizations, you can arrive at:

- A site-specific look and feel of the Desktop: whether it uses tabs or frames, what channels are available to users and how they are situated out on the Desktop, what applications are available to end users, what kind of online help is available, and so on.
- Different Desktops for different LDAP roles or organizations.
- Desktop behavior based on user roles.

This guide describes these administrator customizations.

What Are the Areas for Customizing the Desktop?

In general, the Portal Server software documentation divides user interface customization into the following areas:

Desktop pages	You can modify the look and feel of the templates, including images, structure, and color.
Desktop layout	You can modify the look and feel of the Desktop, customize the top container used by the organization, control Desktop themes, and so on. This chapter describes the Desktop customization tasks.

Using the Display Profile

Much of your work in customizing the Portal Server software involves creating or editing the display profile to provide the kind of Desktop you want for your site. The display profile is an XML document that defines the Desktop structure and content. The display profile Document Type Definition file (DTD) defines valid syntax for the display profile XML documents. See `/etc/opt/SUNWportal/dtd/psdp.dtd` for more information.

The hierarchical structuring of the display profile document does not define the visual layering of channel on the portal Desktop. The display profile exists only to provide property values for channels on the Desktop.

The display profile contains definitions that enable you to construct the Desktop. These definitions include providers, channels, containers, and properties. Some of these definitions create the Desktop containers (such as the frames, tables, and tabs that arrange the content of the Desktop) and others create channels for the Desktop through the respective providers. A display profile provider definition is a template for building channels based on that provider.

Note – A provider is a Java class responsible for converting the content in a file, or the output of an application or service into the proper format for a channel. A number of providers are shipped with the Portal Server software. As the desktop is imaged, each provider is queried in turn for the content of its associated channel. Some providers are capable of generating multiple channels based upon their configuration.

Using the display profile mechanism, you can create a new portal desktop or modify the sample portal provided with the product. See Chapter 2 and Chapter 3 for more information.

The display profile documents themselves consist of display profile objects. The display profile DTD defines the XML tags that represent the allowable display profile objects.

Like display profile objects are grouped within their appropriate XML tag pairs. That is, providers are grouped within `<Providers></Providers>` tags, channels within `<Channels></Channels>` tags, properties within `<Properties></Properties>` tags.

For a complete discussion of the display profile, how the merging works, and a description of the display profile DTD syntax, see the *Sun Java System Portal Server 7.1 Technical Reference*.

Using JavaServer Pages and (HTML) Template Files

To generate the rendered desktop user interface, the Portal Server software makes use of either JavaServer Pages (JSP™) or template files. JSPs are preferred because they enable a much easier customization process without having to change the provider Java classes. JSPs also provide a way to enable a strict separation of business and presentation logic. Specifically, this means having the business logic in the provider classes and presentation logic in JSPs.

Note – In general, a three-tier architecture consists of presentation logic, business logic, and the data. Tag libraries or Enterprise JavaBeans™ provide the business logic, a database contains the data, and JavaServer Pages (JSPs) or templates provide the presentation logic. However, this view is based on a “small” system where the entire system is contained in one server, or perhaps only the data is on another server.

The Portal Server software takes the “larger” system view, where all of the product is presentation. The business logic resides in some back end resource server that a content provider accesses. The data is on yet another back end server.

The default set of JSPs and template files are installed in *PortalServer-DataDir*`/portals/`*portal-ID*`/desktop/default` directory. The sample portal JSPs and template files are installed in

PortalServer-DataDir/portals/*portal-ID*/desktop/developer_sample directory. The Desktop Type attribute in the Desktop attributes page of the Portal Server management console specifies from what subdirectory to retrieve either the JSP or template files for the Desktop.

Note – How the JSPs and Template files are referenced by the providers is provider-specific. Some providers specify the file in the display profile, other providers specify fixed names.

For example, for JSPProvider, there are display profile properties such as contentPage, editPage, and so on, that reference Desktop files under the *PortalServer-DataDir*/portals/*portal-ID*/desktop/default directory. For other providers, such as BookmarkProvider, the name of the template file is fixed, for example, display.template and that name is mentioned in the display profile document for that provider.

Using the Desktop Tag Library

Desktops based on JSPs enable a customization process without the necessity of changing the provider Java classes. The implementation of the JSP-based Desktop uses a tag library which the Portal Server software supplies. Not all Desktop channels need to be JSP-based. There are also channels using HTML-based templates.

A tag library is exposed through Tag Library Descriptors (TLD) files, so tags are in their appropriate functional area. For more information, see the *Sun Java System Portal Server 7.1 Technical Reference*.

◆ ◆ ◆ CHAPTER 2

Modifying the Developer Sample Portal

This chapter contains the following sections:

Introduction to Developer Sample

The Portal Server software includes a sample portal that demonstrates the Portal Server software features. The sample portal requires customization before deploying the portal because the sample portal does not have the custom content that you may require in your deployment environment.

Editing the Developer Sample Portal Files

Do not directly edit any of the files that make up the sample portal (display profile XML, JSP, and template files). Instead, make a copy of the sample portal files to modify to a new directory and then modify those copied files. In this way you preserve the integrity of the sample portal. Additionally, if you later apply a patch to the Portal Server, you won't loose any changes you might have made to the sample portal files, as the patch would only overwrite the initially installed sample files.

Changing the Desktop Type

You should also create a custom desktop type for your users. The desktop type attribute of the desktop service is a comma-separated string. It is still a string type, but the desktop uses it as an ordered desktop type list. The list is used by the desktop lookup operation when searching for templates and JSPs. The lookup starts at the first element in the list and each element represents a sub directory under the desktop template base directory. If a template is not found in the first directory, then it proceeds to the next one in the list. This continues until the item is found (or not), for all desktop type elements in the list.

If the `default` directory is not included in the list, it will be added at the end of the list implicitly. For example, if the desktop type is `developer_sample`, the target template will be searched in the `developer_sample` sub directory, then the `default` sub directory.

By default, if the developer sample is installed, then the desktop type attribute is set to `developer_sample`, meaning files are retrieved from the `developer_sample` subdirectory. If the Developer Sample is not installed, then the desktop type attribute value is set to `default`. The authless user is created as part of the sample portal, and the desktop type for the authless user is set to `developer_anonymous_sample,developer_sample`.

You can define a new set of templates by creating a new directory under the *PortalServer-DataDir*`/portals/`*portal-ID*`/desktop/` directory, placing your template files in this directory, and making this directory the Desktop Type attribute for that organization.

▼ To change the Desktop type

1 **Create a new subdirectory in the** *PortalServer-DataDir*`portals/`*portal-ID*`/desktop` **directory.**

For example:

`mkdir` *PortalServer-DataDir*`/portals/`*portal-ID*`/desktop/sesta`

2 **Manually copy only the template files that you wish to modify to the new directory location.**

For example, if your Desktop type will modify `content.jsp` file for JSPProvider, copy this file to *PortalServer-DataDir*`/portals/`*portal-ID*`/desktop/sesta/JSPProvider/content.jsp`, and customize the file for the new Desktop type in that location.

You only need to copy the files that you have changed from the sample installation to the new directory tree. This structure enables you to tell at a glance which files have been modified from the original distribution. It also eliminates the need to back up copies of the original sample files.

3. **Use the Portal Server software management console to change the value of the Desktop Type attribute for the subdirectory created in step 1.**

   As this attribute is a Dynamic Access Manager service attribute, you need to change it everywhere that it appears (organization, sub-organization, role, and user). Changing the Desktop Type at the organization level will not necessarily be reflected at the user level. This will be the case only if the user has not overwritten the Desktop Type in which case the Desktop Type value will be inherited from the organization level. If the user defines the Desktop Type at the user level, the value will remain the same even if the Desktop Type is changed at the organization level.

   In this example, in the Portal Server management console, you would specify `sesta` as the value for the Desktop Type attribute.

Restoring the Default Settings

The Developer Sample portal display profile file goes in the `o=DeveloperSample,dc=sun,dc=com` DN and the display profile file is in *PortalServer-base*`/par-src/developer_sample/dp` directory.

▼ To Reload the Global Sample Display Profile

1. **Create the default portal PAR file.**

   For example, type*PortalServer-base*`/bin/psadmin create-par --dir` *PortalServer-base*`/par-src/default-portal/ /tmp/default-portal.par`

2. **Import the default portal PAR file.**

   For example, type *PortalServer-base*`/bin/psadmin import -u` *amadmin* `-f` *password-file* `-p portal1 --overwrite /tmp/default-portal.par`.

▼ To Restore Developer Sample Display Profile

1. **Copy Developer Sample configuration files to a custom directory.**

   Copy the template files from *PortalServer-base*`/samples/portals/shared` to a custom directory, remove the `.template` extension, and substitute the tokens in each file. For example, type:

```
mkdir /tmp/mydir
cp /opt/SUNWportal/samples/portals/shared/input.properties.template
 /tmp/mydir/input.properties
cp /opt/SUNWportal/samples/portals/shared/password.properties.template
 /tmp/mydir/password.properties
```

2 Specify values for the `input.properties` and `password.properties` files.

Edit the `input.properties` file and replace all the tokens that begin and end with `%` with the appropriate Portal Server software settings. Edit the `password.properties` file and replace all the tokens that begin and end with `%` with the appropriate administrator password value.

3 Run ant to install the Developer Sample.

For example, to install the Developer Sample:

- On Solaris, type:

```
export JAVA_HOME=/usr/jdk/entsys-j2se
/usr/sfw/bin/ant -buildfile PortalServer-base/samples/portals/developer/build.xml
-Dconfig.location PortalServer-DataDir/tmp -logfile
PortalServer-DataDir/tmp/dslog.txt
```

- On Linux, type:

```
export JAVA_HOME=/usr/jdk/entsys-j2se
/opt/sun/bin/ant -buildfile
PortalServer-base/samples/portals/developer/build.xml
-Dconfig.location /var/opt/sun/portal/tmp -logfile
/var/opt/SUNWportal/tmp/dslog.txt
```

◆ ◆ ◆ CHAPTER 3

Creating a New Desktop

Creating a new Desktop involves the following:

Developing the Container

When developing a container, you can do the following:

- Define a `<Container>` element in the display profile that references an existing `<Provider>` element. For example, to create a custom Container definition that uses the JSPTabContainerProvider:

```
<Container name="MyTabContainer" provider="JSPTabContainerProvider">
        <Properties/>
        <Available/>
        <Selected/>
        <Channels/>
</Container>
```

- Define a `<Provider>` element in the display profile that references an existing container provider class. You must also define the container. For example, create the custom Provider definition that uses the JSPTabContainerProvider container provider class:

```
<Provider name="MyTabContainerProvider" class="com.sun.portal.providers.containers.jsp.tab.JSPTabContainerProvider">
        <Properties/>
</Provider>

<Container name="MyTabContainer" provider="MyTabContainerProvider">
        <Properties/>
        <Available/>
        <Selected/>
        <Channels/>
</Container>
```

- Define a container provider class that extends an existing container provider such as JSPTabContainerProvider. You must also define the container in the display profile. For example:

```
public class MyTabContainerProvider extends JSPTabContainerProvider {
    }

<Container name="MyTabContainer" provider="MyTabContainerProvider">
        <Properties/>
        <Available/>
        <Selected/>
        <Channels/>
</Container>
```

- Define a container provider class from scratch that extends ContainerProviderAdapter. You must also define the container in the display profile. If you create a container by extending the container provider class, then it also needs to implement the ContainerProvider interface:

```
public class MyTabContainerProvider extends ContainerProviderAdapter
        implements ContainerProvider {
}

<Provider name="MyTabContainerProvider" class="package.MyContainerProvider">
        <Properties/>
</Provider>

<Container name="MyTabContainer" provider="MyTabContainerProvider">
        <Properties/>
        <Available/>
        <Selected/>
        <Channels/>
</Container>
```

Note – You cannot create a container provider by just extending the Provider class. By definition, a container must implement the ContainerProvider interface. ContainerProviderAdapter does this. See the *Sun Java System Portal Server 7.1 Developer's Guide* for more information on creating custom container providers.

If you write a new class file, it must reside in the *PortalServer-DataDir*`/portals/`*portal-ID*`/desktop/classes` directory. You can change this location by editing the *PortalServer-DataDir*`/portals/`*portal-ID*`/config/desktopconfig.properties` file. The `providerClassBaseDir` setting in the `desktopconfig.properties` file determines the directory where the classes will be picked up.

Editing the Display Profile

The developer needs to edit the display profile document and specify the `<Container>` and `<Provider>` display profile elements where applicable. The tags that should be modified are:

```
<Provider name="provider" class="provider class">
<Container name="container" provider="provider">
```

For JSP files, the `<Properties>` tag for the provider contains the following property tag, which references the JSP Content page:

```
<String name="contentPage" value="value">
```

The `<Properties>` tag for the channel can have values that override the properties set in the `<Provider>` tag. Thus, if desired, you could set the JSP `contentPage` value here. You do not reference template-based providers, or other providers you might develop, in this way.

The `<Available>` and `<Selected>` tags are required for all containers in the display profile.

The JSP-based tab, table, and frametab containers have additional properties requirements.

Note – There is a distinction between a provider element in the display profile and the Java class for the provider.

Provider element

```
<Provider name="JSPTableContainer" class=com.sun.portal.providers.containers.jsp.table.JSPTableContainerProvider>
```

Java class

```
com.sun.portal.providers.containers.jsp.table.JSPTableContainerProvider
```

You can modify display profile objects by performing one of the following:

- Using the Portal Server software administration console to download the display profile and uploading it after modifying it.
- Manually editing an existing display profile document and then loading it at the appropriate LDAP node by using the `psadmin modify-display-profile` subcommand.
- Creating a new display profile document from scratch and then loading it at the appropriate LDAP node by using the `psadmin add-display-profile` subcommand.

For more information on display profile and the `psadmin` subcommands to manage the display profile, see the *Sun Java System Portal Server 7.1 Command Line Reference*.

Developing and Deploying JSP or Template Files

The Developer Sample JSP and template files are located in the *PortalServer-DataDir*`/portals/`*portal-ID*`/desktop/developer_sample` directory. The JSP and template files can be copied to a custom directory and modified by the developer or the developer can create new JSP or template files for their deployment.

For the JSPs, you can find compilation and runtime errors in the desktop debug log at *PortalServer-DataDir*`/portals/`*portal-ID*`/logs/portal-instance/`*portal.0.0*`.log`. Also, all JSPProvider based Desktop channels have a property called `showExceptions`. By default, this property is set to `false`; setting it to `true` causes the JSP exception to show up as the content of the channel.

When you create a container, you need to create a new subdirectory for your newly created container in the *PortalServer-DataDir*`/portals/`*portal-ID*`/desktop/`*desktopType* directory. That is, the newly created container should be placed based on what the *desktopType* is. If the Developer Sample portal is installed, then the *desktopType* is, by default, `developer_sample`; so, create a new directory for the container under `developer_sample` so that any JSP and template

that is being added can adopt the same look and feel that as defined in the sample portal. If sample portal is not installed, and if you have set up a custom *desktopType*, for example, `foo`, then the new container directory must be created directly under `foo`.

Either copy the modified JSP or template files here, or place your newly created files here. If you use a sample container without changing any content or file names, you do not need to create a new subdirectory nor copy any files there. (In the example that follows, a new subdirectory is needed, because a new container is created.)

For example, let's say you create a new container called newSingleContainer whose display profile definition is the following:

```
<Container name="newSingleContainer" provider="JSPSingleContainer">
    <Properties>
        <String name="helpURL" value="desktop/newSingle.html"/>
        <String name="title" value="A new single container"/>
        <String name="contentPage" value="newsinglecontent.jsp"/>
        <Boolean name="isEditable" value="true"/>
        <String name="editType" value="edit_subset"/>
    </Properties>
    <Available/>
    <Selected/>
    <Channels/>
</Container>
```

Because the file specified for the `contentPage` property is different from the `contentPage` value for the provider definition, you need to create a new directory under the *PortalServer-DataDir*`/portals/`*portal-ID*`/desktop/developer_sample` directory called `newSingleContainer`. You then only need to copy the `newsinglecontent.jsp` file to this new directory. The system is able to locate all other JSPs referenced by the JSPSingleContainer provider.

If desired, rather than customizing the sample portal JSP and template files directly, you can create a separate directory for your organization's customized files, and perform customizations on those files. This preserves the initially installed portal JSP and template files.

▼ To Create Customized Organization JSP and Template Files

1 Change directories to the Desktop JSP or template directory.

For example,

`cd` *PortalServer-DataDir*`/portals/`*portal-ID*`/desktop`

2 **Create a new directory for your organization's JSPs and templates.**

For example,

```
mkdir sesta
```

3 **Copy the JSPs and templates that you wish to modify into the new directory location, maintaining the same directory structure.**

For example, if your new Desktop type will modify *PortalServer-DataDir*`/portals/`*portal-ID*`/desktop/default/JSPProvider/content.jsp`, copy this file to *PortalServer-DataDir*`/portals/`*portal-ID*`/desktop/sesta/JSPProvider/content.jsp`, and customize the file for the new Desktop type in that location.

4 **Customize the JSPs templates in the `sesta` directory as required.**

5 **Change the dynamic Desktop Type attribute in the Portal Server software administration console to use the newly created directory.**

Loading the Display Profile at the Appropriate LDAP Nodes

The display profile document for the Developer Sample exist at DNs similar to the following:

```
o=DeveloperSample,dc=sun,dc=com
```

Load the display profile at the appropriate LDAP node using the `psadmin` subcommands. You can also use the Edit Display Profile XML from the Portal Server software management console using the two links to download the display profile and upload the display profile. Download the display profile, make the necessary changes, and then upload it again through the Portal Server software management console.

See the *Sun Java System Portal Server 7.1 Command Line Reference* for more information on the `psadmin` subcommands.

(Optional) Creating a Resource Bundle

If you created a new provider, you may need to create a resource bundle file with the same name as the provider. The provider resource bundle should be in *PortalServer-DataDir*`/portals/`*portal-ID*`/desktop/classes` directory.

Accessing the Developer Sample Portal Desktop

Access the Desktop in one of the following ways.

▼ To Access the Desktop

- **Use a specific container or channel reference by using the provider argument to the Desktop login URL:**

  `http://`*hostname:port*`/`*portal-ID*`/dt?provider=`*provider-name*

  where *provider-name* is one of the providers. For example, to access the JSP-based tab Desktop, in a browser, type:

  `http://`*hostname:port*`/`*portal-ID*`/dt?provider=JSPTabContainer`

  - **If no channel is referenced, the Desktop looks in the session for the last channel or container that was displayed. (This is stored in the session.)**
  - **If no channel is stored in the session, the Desktop looks in a Desktop service attribute for the top-level container to display (Default Channel Name attribute). This happens after an initial login.**
  - **Once this top-level container is determined, that container draws the containers or channels that it references (through the Selected list), until all of its leaves have been reached.**

Debugging the Desktop

Use the following to help debug the Developer Sample portal desktop environment:

The Desktop debug file is located at *PortalServer-DataDir*`/portals/`*portal-ID*`/logs/`*portal-instance*`/portal.0.0.log`.

If you get an error message page on the desktop, you can view the source to look at the stack trace.

◆ ◆ ◆ CHAPTER 4

Customizing Container Tabs

This chapter provides a variety of tasks to customize the container tabs. It contains the following sections:

Note – The order that you list the tabs in the display profile is the order that tabs are displayed in the Desktop. So, to make a tab the first tab in the user's Desktop, you need to move it to be first in the selected list in the display profile.

Adding a Tab to JSPTabContainer

A tab can be any container type, but, by default, the sample portal uses table container. To add a new tab, you must first define the container, then register that container in JSPTabContainer, which "houses" the tabs.

▼ To Add a Tab to JSPTabContainer

1 **Create the necessary display profile.**

a. **Define the new collection within** `<Collection name="TabProperties">` **in JSPTabContainer. For example:**

```
...
<Collection name=" NewTabPanelContainer" >
    <Boolean name="removable" value="false"/>
    <Boolean name="renamable" value="true"/>
    <Boolean name="predefined" value="true"/>
</Collection>
...
</Collection> ...
```

b. **Add entries to the** `<Available>` **and** `<Selected>` **tags. For example:**

```
...
<Available>
    <Reference value="NewTabPanelContainer"/>
    ...
</Available>
...
<Selected>
    <Reference value="NewTabPanelContainer"/>
    ...
</Selected>
...
```

c. **Define a container for NewTabPanelContainer. For example:**

```
<Container name="NewTabPanelContainer" provider="JSPTableContainerProvider">
        <Properties>
            <String name="title" value="New Container Channel"/>
            <String name="contentPage" value="tabtable.jsp"/>
            <String name="description" value="This is a test for front table containers"/>
            <String name="Desktop-fontFace1" value="Sans-serif"/>
            <Collection name="categories">
                <String value="Personal Channels"/>
                <String value="Sample Channels"/>
            </Collection>
            <Collection name="Personal Channels">
                <String value="UserInfo"/>
                <String value="MailCheck"/>
            </Collection>
            <Collection name="Sample Channels">
                <String value="SampleJSP"/>
                <String value="SampleXML"/>
```

```
            </Collection>
        </Properties>
        <Available>
            <Reference value="UserInfo"/>
            <Reference value="MailCheck"/>
            <Reference value="SampleJSP"/>
            <Reference value="SampleXML"/>
        </Available>
        <Selected>
            <Reference value="UserInfo"/>
            <Reference value="MailCheck"/>
            <Reference value="SampleJSP"/>
            <Reference value="SampleXML"/>
        </Selected>
        <Channels>
            ...
        </Channels>
    </Container>
```

d. If predefined property value is true in the `TabProperties` collection ("Adding a Tab to JSPTabContainer" on page 37), then it is recommended to define a Provider for the container channel which is meant to be used as a predefined tab.

For example, see below for the PredefinedNewTabPanelContainerProvider Display Profile Definition

```
<Provider name="PredefinedNewTabPanelContainerProvider" class="com.sun.portal.providers.containers.jsp.table.JSP
    <Properties>
        <ConditionalProperties condition="locale" value="en" >
            <ConditionalProperties condition="locale" value="US" >
                <String name="title" value="New Sample"/>
                <String name="description" value="New Tab"/>
            </ConditionalProperties>
        </ConditionalProperties>
        <String name="title" value="New Sample"/>
        <String name="description" value="New Tab"/>
        <String name="contentPage" value="tabtable.jsp"/>
        <String name="presetThemeChannel" value="JSPPresetThemeContainer" advanced="true"/>
        <String name="customThemeChannel" value="JSPCustomThemeContainer" advanced="true"/>
        <String name="parentTabContainer" value="JSPTabContainer" advanced="true"/>
        <String name="Desktop-fontFace1" value="Sans-serif"/>
        <String name="refreshTime" value="" advanced="true"/>
        <String name="width" value="thin" advanced="true"/>
        <String name="fontFace1" value="Sans-serif"/>
        <String name="productName" value="Sun Java System Portal Server"/>
        <String name="maximizedChannel" value=""/>
        <Integer name="timeout" value ="240"/>
        <Integer name="layout" value="1"/>
        <Boolean name="showExceptions" value="false"/>
```

```
<Boolean name="parallelChannelsInit" value="false"/>
<Boolean name="refreshParentContainerOnly" value="false" advanced="true"/>
<Boolean name="isEditable" value="true" advanced="true"/>
<String name="editType" value="edit_complete" advanced="true"/>
<String name="editContainerName" value="JSPEditContainer" advanced="true"/>
<Integer name="thin_popup_height" value="200"/>
<Integer name="thin_popup_width" value="500"/>
<Integer name="thick_popup_height" value="300"/>
<Integer name="thick_popup_width" value="600"/>
<Integer name="fullwidth_popup_height" value="500"/>
<Integer name="fullwidth_popup_width" value="600"/>
<Boolean name="defaultChannelIsMinimizable" value="true"/>
<Boolean name="defaultChannelIsMaximizable" value="true"/>
<Boolean name="defaultChannelIsMinimized" value="false" advanced="true"/>
<Boolean name="defaultChannelIsDetached" value="false" advanced="true"/>
<Boolean name="defaultChannelIsDetachable" value="true"/>
<Boolean name="defaultChannelIsRemovable" value="true"/>
<Boolean name="defaultChannelHasFrame" value="true" advanced="true"/>
<Boolean name="defaultChannelIsMovable" value="true"/>
<Boolean name="defaultBorderlessChannel" value="false" advanced="true"/>
<String name="defaultChannelColumn" value="1" advanced="true"/>
<String name="defaultChannelRow" value="1" advanced="true"/>
<Collection name="categories">
    <String value="Sample Channels"/>
</Collection>
<Collection name="Sample Channels">
    <String value="SampleRSS"/>
    <String value="SampleURLScraper"/>
    <String value="Notes"/>
    <String value="SampleSimpleWebService"/>
</Collection>
<Collection name="channelsColumn" advanced="true">
    <String name="SampleURLScraper" value="2"/>
    <String name="Notes" value="2"/>
    <String name="SampleSimpleWebService" value="2"/>
</Collection>
<Collection name="channelsRow" advanced="true">
    <String name="SampleURLScraper" value="2"/>
    <String name="Notes" value="3"/>
    <String name="SampleSimpleWebService" value="4"/>
</Collection>
<Collection name="channelsIsMinimized" advanced="true"/>
<Collection name="channelsIsDetached" advanced="true"/>
<Collection name="channelsHasFrame" advanced="true"/>
<Collection name="channelsIsMinimizable"/>
<Collection name="channelsIsMaximizable"/>
<Collection name="channelsIsMovable"/>
<Collection name="channelsIsRemovable"/>
```

```
        <Collection name="channelsIsDetachable"/>
        <Collection name="borderlessChannels"/>
    </Properties>
</Provider>
```

e. Define the container channel based on the PredefinedNewTabPanelContainerProvider.

When the user creates a new tab based on the predefined tab, all the properties for this tab are picked up from the Provider definition. For example, see below for PredefinedNewTabPanelContainer Channel Properties.

```
<Container name="PredefinedNewTabPanelContainer" provider="PredefinedNewTabPanelContainerProvider">
    <Properties/>
    <Available>
        <Reference value="SampleRSS"/>
        <Reference value="SampleURLScraper"/>
        <Reference value="Notes"/>
        <Reference value="SampleSimpleWebService"/>
    </Available>
    <Selected>
        <Reference value="SampleRSS"/>
        <Reference value="SampleURLScraper"/>
        <Reference value="Notes"/>
        <Reference value="SampleSimpleWebService"/>
    </Selected>
    <Channels>
    </Channels>
</Container>
```

2 Load the display profile into LDAP by using the `psadmin` subcommand.

3 Bring up the Desktop and verify that the tab was added.

Creating a Tab Within a Tab

This is similar to "Adding a Tab to JSPTabContainer" on page 37, except that instead of defining the tab based on JSPTableContainerProvider, you base the new tab on JSPTabContainerProvider.

Stretching a Tab Across an Entire Container

This section includes instructions for creating a tab that spans an entire container. The top-level container is a tab container. Nested within it is a table container, and nested within that is a tab container, in which the tab spans the entire container.

▼ To Stretch a Tab Across an Entire Container from the Command Line

1. **Edit the display profile and make the width of the Nested Tab Container `full_top` in the Table Container so that it stretches across the entire page.**
2. **Load the display profile into LDAP by using the `psadmin` subcommand.**

▼ To Stretch a Tab Across an Entire Container from the Portal Server Management Console

1. **Log in to the Portal Server management console and select Portals and *portalID*.**
2. **Select DeveloperSample [Org] from the Select DN drop down menu and select Manage Containers and Channels.**

   Ensure that JSPTabContainer is selected.
3. **Click on New Channel or Container and create a new container by specifying the Container Provider (for example, JSPTableContainerProvider) and Name for the new container (for example, TestContainer).**
4. **Select the container you just created (for example, TestContainer) and click New Channel or Container (in the container you just created).**
5. **Create a new container by specifying the Container provider (for example, JSPTabContainerProvider, and Name for container (for example, TestTabContainer).**
6. **Select the new tab container you just created (for example, TestTabContainer), specify the `width` property as `full_top`, and save the values.**
7. **Create a new tab (for example, TestNewTab) for the tab container you just created (for example, TestTabContainer) by selecting New Tab.**

Changing the Tab Image for JSP-based Tab Containers

You can customize the look of tabs as they use images.

▼ To Change the Tab Image for JSP-based Tab Containers

1. **Log in to the Portal Server management console.**
2. **Edit the display profile for the Developer Sample organization.**
   To edit the display profile:
   a. **Select Portals from the menu bar.**
   b. **Select a portal server and the desired organization or suborganization under Current Location.**
   c. **Click Manage Containers and Channels.**
   d. **Scroll to View Type and select DP XML tree.**
3. **Select Global Themes and the theme you wish to modify in the DP XML Tree.**
4. **Change the value of the** `tabNotchImage` **property to the new image name.**
   By default, the value for this property is `tabNotch.gif`.
5. **Copy the new image into** *PortalServer-base*`/web-src/desktop/tabs/images` **directory.**
6. **Run the** *PortalServer-base*`/bin/psadmin redeploy --adminuser` *amadmin* `--passwordfile` *passwordfile* `--portal` *portal-ID* **subcommand to deploy the new image.**
7. **Reload the Desktop to verify the change.**

Changing the Color of Tabs

The background color of tabs are part of the themes.

▼ To Change the Color of Tabs

1. **Log in to the Portal Server administration console.**
2. **Edit the display profile for the Developer Sample organization.**
   To edit the display profile:
   a. **Select Portals from the menu bar.**
   b. **Select a portal server and the desired organization or suborganization under Current Location.**
   c. **Click Manage Containers and Channels.**
   d. **Scroll to View Type and select DP XML tree.**
3. **Select Global Themes and the theme you wish to modify in the DP XML Tree.**
4. **Change the value of the `titleBarColor` property to change the color of the selected tab and/or change the value of `tabColor` property to change the color of an unselected tab.**
   The selected tab background color is the same as the title bar color.
5. **Reload the Desktop to verify the change.**

Making a Tab the Start Tab

The "Start tab" is the tab that is highlighted when user first logs in.

▼ To make the tab the start tab

1. **Edit the display profile for the appropriate container.**
2. **Change the `startTab` property to the tab to highlight when the user logs in. For example:**
   ```
   <String name="startTab" value="MyFrontPageTabPanelContainer"/>
   ```
3. **Load the display profile into LDAP by using the `psadmin` command.**

Adding a Channel to a User-defined Tab

Users can add a new tab to their Desktop by using the Tabs link and then by clicking the Make a New Tab link. The channel list that gets displayed on the content page which is shown when the user selects to create a new tab from scratch is picked up from the JSPTabCustomTableContainer's Available list.

◆ ◆ ◆ CHAPTER 5

Customizing Channels

This chapter describes how to customize channels. This chapter contains the following sections:

Customizing Channel Refresh Times and Container Caching

The `refreshTime` property controls how often a channel's content is reloaded. When `refreshTime` is set to `0` (the default) for the container, the browser refresh (or reload) causes the page to be reloaded and the `getContent()` method is called again for every channel.

The following applies to a single channel:

- It is not possible to refresh only the content of the single channel within a container because a channel is an HTML table cell.
- It is possible to use the `DesktopURL()` method in the PAPI. The provider can use `getDesktopURL()` to get the Desktop servlet's URL, append arguments to it, and generate a new URL (or link).

The following applies to controlling and configuring container caching:

- Use the `refreshTime` property for the container along with the `refreshTime` for individual channels within the container.
- If the `refreshTime` for the container is blank, it is calculated to be the minimum time for all of the contained channels. If you want to override that calculated time, set a `refreshTime` for the container and then the content for the whole container will be cached.

Note – If you have a large number of channels, utilize the provider caching by setting the `refreshTime` to a large number so that the portal page can use cached content. This makes sense when most of your channels have static content. The way the `refreshTime` works is if the container's `refreshTime` is set, it will use it. If `refreshTime` is set to an empty string, it will try to get and use the minimum of the `refreshTime` of its selected channels.

Customizing Window Preference

For channels that include links that launch another browser, you can control how this browser window is opened.

▼ To Customize the Channel Window Preference

1 **Define the display profile (either for the channel, to make the change for only that channel, or for the provider, to make the change for every channel that uses the provider) so that it includes the `windowPref` property.**

For example:

```
<Properties>
    ...
    <String name="windowPref" value="all_new"/>
    ...
</Properties>
```

Note –

The values are:

- `all_new` (New window is opened for every link)
- `one_new` (All links open on the same new window)
- `same` (Desktop window)

2 **Load the display profile into LDAP using the `psadmin` subcommand or from the Portal Server management console.**

Note – The intelligence has to be built with the help of JavaScript for that particular channel.

▼ To Customize the Channel Window Preference from the Portal Server Management Console

1. **Log in to the Portal Server management console and select the user, organization, or role for which the `windowPref` has to be changed.**

2. **Select Manage channels and containers and click on the concerned channel. On the right frame, change the `windowPref` property value for the channel.**

   The values can be:

   - `all_new` (New window is opened for every link)
   - `one_new` (All links open on the same new window)
   - `same` (Desktop window)

Removing a Button

▼ To Remove a Button From All Channels in a Container

1. **Find the container you want to work with. If you are working with one of the sample portals, you need to modify the appropriate "contained" container, which is part of the top-level container.**

2. **Add the appropriate property (within the `<Properties></Properties>`) tags from "Removing a Button" on page 49 to the container's display profile for the button you want to remove. This two column table lists the button in the first column and the property to hide the button in the second column.**

   The order of the buttons in this table corresponds to the order they appear in the channel, from left to right: Minimize, Maximize, Help, Edit, Detach, and Remove.

Button	Property to Hide the Button
Minimize	`<Boolean name="defaultChannelIsMinimizable" value="false"/>`
Maximize	`<Boolean name="defaultChannelIsMaximizable" value="false"/>`
Help	`<String name="helpURL" value=""/>`
Edit	`<Boolean name="isEditable" value="false"/>`

Button	Property to Hide the Button
Detach	`<Boolean name="defaultChannelIsDetachable" value="false"/>`
Remove	`<Boolean name="defaultChannelIsRemovable" value="false"/>`

Note – For the Help and Edit buttons, insert the respective property for each channel. You cannot insert the property within the container's `<Properties></Properties>` tags.

Make sure the following properties are not defined in the container:

```
<Collection name="channelsIsRemovable">..</Collection>
<Collection name="channelsIsMinimizable"/>..</Collection>
<Collection name="channelsIsMaximizable"/>..</Collection>
<Collection name="channelsIsDetachable"/>..</Collection>
```

3 **Load the display profile into LDAP using the `psadmin` subcommand or from the Portal Server management console.**

▼ To Remove a Button From All Channels in a Container From the Portal Server Management Console

1 **Log in to the Portal Server management console and select the user, organization, or role in which the container is defined.**

2 **Select Manage Channels and Containers and click on the contained container.**

3 **Change the `DefaultChannelIsMinimizable`, `DefaultChannelIsMaximizable`, `helpURL`, `isEditable`, `DefaultChannelIsDetachable`, and `DefaultChannelsIsRemovable` properties to `false`.**

4 **Select Save to save the new values.**

▼ To Remove a Button From a Single Channel

1 **For the channel from which you want to remove a button, add the appropriate property to a `Collection` tag in the container that contains the channel. See "Removing a Button" on page**

49, for the button you want to remove. This two column table lists the button in the first column and the property to hide the button in the second column

The order of the buttons in this table corresponds to the order they appear in the channel, from left to right: Minimize, Maximize, Help, Edit, Detach, and Remove.

Button	Property to Hide the Button

Minimize	`<Collection name="channelsIsMinimizable">`
	`<Boolean name="channelname" value="false"/>`
	`</Collection>`

Maximize	`<Collection name="channelsIsMaximizable">`
	`<Boolean name="channelname" value="false"/>`
	`</Collection>`

Detach	`<Collection name="channelsIsDetachable">`
	`<Boolean name="channelname" value="false"/>`
	`</Collection>`

Remove	`<Collection name="channelsIsRemovable">`
	`<Boolean name="channelname" value="false"/>`
	`</Collection>`

2 **For the channel in which you want to remove a button, add the appropriate property to a Collection tag in the controlling container.**

For example, use the following XML to hide the Remove button for the Sample JSP channel in the JSP table container, MyFrontPageTabPanelContainer, whose container is JSPTabContainer.

```
<Container name="MyFrontPageFramePanelContainer" provider="JSPTableContainerProvider">
    <Properties>
        ...
        <Collection name="channelsIsRemovable">
        <Boolean name="SampleJSP" value="false"/>
        </Collection>
    </Properties>
    ...
```

3 **Load the display profile into LDAP by using the `psadmin` subcommand or from the Portal Server management console.**

▼ To Remove a Button from a Single Channel From the Portal Server Management Console

1. **Log in to the Portal Server management console and select the user, organization, or role in which the container is defined.**
2. **Select Manage Channels and Containers and click on the contained container.**
3. **Change the** `channelsIsMinimizable`**,** `channelsIsMaximizable`**,** `channelsIsDetachable`**, and** `channelsIsRemovable` **properties as follows:**
   a. **Select the property (for example,** `channelsIsMinimizable`**) and click on New Property from the Properties table.**
   b. **Create a boolean type property, specify the channel name that does not want that button, and set the value to be** `false`**.**
   c. **Follow the steps to finish the wizard.**
      There will be a new boolean property (for example, for the `channelsIsMinimizable` property) in the Properties table for the specified channel.

Changing the Channel Layout for a Table Container

▼ To Change the Channel Layout for a Table Container

1. **You can change the layout for a particular table container by modifying (or adding) the following property in the table container's display profile:**

   ```
   <Integer name="layout" value="value"/>
   ```

   where value is:

   1 = Thin-wide, two columns

   2 = Wide-thin, two columns

   3 = Thin-wide-thin, three columns

2. **Reload the display profile to LDAP by running the** `psadmin` **with the** `modify-display-profile` **sub-command, loading it at the top-most node in the directory by using the** `-g` **option.**
   See the *Sun Java System Portal Server 7.1 Command Line Reference* for more information on the `psadmin` subcommand.

Removing the Title Bar from a Channel

▼ To Remove the Title Bar from a Channel

1. **Add the following to the table container display profile in which the channel is present.**

```
<Collection name="channelsHasFrame">
<Boolean name="channelname" value="false"/>
</Collection>
```

2. **Load the display profile into LDAP by using the `psadmin` subcommand or from the Portal Server management console.**

Changing the Channel Border Width and/or Color

You can change the `borderWidth` property and `borderColor` property for the GlobalThemes collection. This changes the width and the color of the channel borders respectively for a theme. Users can then select the theme from the Themes page.

▼ To Change the Border Width and Color for all Channels in a Container

1. **Log in to the Portal Server management console and select Portals, *portal-ID*, Developer Sample (from the Select DN pull-down menu), and Manage Channels and Containers.**
2. **Select DP XML Tree in the View drop-down menu.**
3. **Select DP_Root, GlobalThemes, and SunTheme.**
4. **Modify the `borderWidth` and `borderColor` settings and save.**

◆ ◆ ◆ CHAPTER 6

Customizing Authentication

Portal Server supports a number of authentication schemes, including LDAP, anonymous, membership, UNIX, and more.

This chapter contains instructions for:

Using UNIX Authentication with LoginProvider

▼ To Use UNIX Authentication with LoginProvider

1 **Change directories to the `default/LoginProvider` directory.**

For example:

```
cd PortalServer-DataDir/portals/portal-ID/desktop/default/LoginProvider
```

2 **Copy the `display_UnixAuth.template` file to `display.template`.**

For example,

```
cp display_AuthUnix.template display.template
```

3 **Add the service from the Access Manager administration console.**

4 **Setup authentication configuration for UNIX**

Note – To use LDAP authentication, the authentication module is already enabled for the default organization. You only need to copy `display_AuthLDAP.template` to `display.template`.

Configuring LDAP Authentication for UserInfoProvider

Out of the box, the UserInfo channel allows the user to edit and maintain their Membership password (change their own password). To change the user's authentication module to only LDAP, the administrator has to customize the UserInfoProvider to acknowledge LDAP authenticated users.

▼ To Enable End User Password Maintenance for LDAP Authentication

1. **Create an LDAP `passwordHandler` template. The template name format is `passwordHandler-`*authType*`.template`.**

   You can copy an existing template in the Userinfo template directory. For example,

   ```
   cd PortalServer-DataDir/portals/portal-ID/desktop/default/UserInfo/html
   cp passwordHandler-Membership.template passwordHandler-LDAP.template
   ```

2. **Optionally, modify the descriptive text within `passwordHandler-`*authType*`.template`.**

   For example, in the `passwordHandler-LDAP.template` file, change the `Membership` to `LDAP`.

3. **Add the authentication module name to the channel's `authTypes` display profile Collection.**

   Use the `psadmin` utility subcommand to add the entry to the UserInfoProvider `<Provider>` element. For example:

   a. **Add the entry `LDAP` to the `authTypes` collection for the UserInfoProvider as shown (in bold) below:**

```
<Collection name="authTypes" advanced="true"> <String value="Membership"/> <String value="LDAP"/> </Collection>
```

      Here, based on the user's SSOToken authentication type, the appropriate authType will be used.

   b. **Import the modified display profile document using the `psadmin modify-display-profile` subcommand.**

4. **Restart the web container.**

5. **Access the portal desktop as an LDAP authenticated user and edit the user info channel.**

   Verify that the password field is displayed.

6. **Modify the user's password and select finished**

7. **Logout and login to the Desktop with the new credentials.**

◆ ◆ ◆ CHAPTER 7

Modifying the Desktop Layout

This chapter describes how to modify the channel arrangement in the Desktop. It contains the following sections:

Deriving More Desktop Layouts

The Desktop Layout page provides a way for users to set the arrangement of the channels by moving them up or down, and right or left. The Desktop Layout page also provides users with the option to set column layout, where they can arrange columns by channel width.

Channel widths are defined as thin, wide, full_top, and full_bottom. A thin channel takes less Desktop area than a wide channel. A full_top channel spans the entire Desktop width above all the other channels. A full_bottom channel spans the entire Desktop width below all other channels. The available layouts, which use different combinations of channel widths, are:

`thin-wide`	Two columns
`wide-thin`	Two columns
`thin-wide-thin`	Three columns
`full_top`	One column spans the width of the Desktop at the top of the page
`full_bottom`	One column spans the width of the Desktop at the bottom of the page

You can derive more Desktop layouts from the existing layouts by modifying display profile properties and the JSPs for the table container, when one of the contained channel's width is specified as either full_top or full_bottom.

For example, you could come up with the following:

```
full_top-thin-wide/full_top-wide-thin/full_top-thin-wide-thin
thin-wide-full_bottom
fulltop-thin-wide-full_bottom
```

Note – The `fulltop-thin-wide-full_bottom` layout is a combination of the first two with `full_top` at the top and `full_bottom` at the bottom.

To do so involves modifying the appropriate display profile. That is, to derive more desktop layouts, use the appropriate display profile for the desired layout. After making a change to the display profile, load the display profile into LDAP by using the `psadmin` subcommand.

To use a full_top-thin-wide/full_top-wide-thin/full_top-thin-wide-thin layout, modify a channel's width in the display profile as follows:

```
<Channel name="Search" provider="SearchProvider">
    <Properties>
        <String name="title" value="Search"/>
        <String name="description" value="This is a search provider example" />
        <String name="searchServer" value=""/>
        <String name=”width” value=”full_top”/>
    </Properties>
</Channel>
```

To use a thin-wide-full_bottom layout, modify a channel's width in the display profile as follows:

```
<Channel name="Search" provider="SearchProvider">
    <Properties>
        <String name="title" value="Search"/>
        <String name="description" value="This is a search provider example" />
        <String name="searchServer" value=""/>
        <String name="width" value="full_bottom"/>
    </Properties>
</Channel>
```

To use a full_top-thin-wide-full_bottom layout, modify the width of one of the channels as full_bottom and one of the other channels as full_top in the display profile.

Changing Content Layout to Support Categorizing the Available and Selected Lists

For sites with a great number of channels, and a need to categorize and sub-categorize these into a hierarchical kind of structure, to make channel selection easier to navigate, and channels easier to find, customization options include:

- "Customizing Existing JSPs" on page 59
- "Writing a New Content Channel" on page 59

Customizing Existing JSPs

You can customize `contentedit.jsp`, which generates the Content page, and `contentdoedit.jsp`, which processes the page. The Content page uses categories to group channels into different sections. If you want this page to look different, for example, to use a pull-down menu instead of showing all the channels for one category, to save on page space, you can customize the two JSPs mentioned. (The main reason for a JSP-based content and layout channels is to support ease of customization.)

Writing a New Content Channel

You can write a new content channel by extending JSPSingleContainerProvider to support your site's needs. You would then only need to modify the Content link in `table.jsp` under JSPTableContainerProvider to point to this new content channel. You can also create your own JSP to do whatever specific implementation you need, and change the link from `table.jsp` to `new.jsp`.

Adding New Layouts

The following example is intended to show some of the customization possibilities for the Desktop. Details are not provided. See the *Sun Java System Portal Server 7.1 Developer's Guide* for more information on how to add new layouts.

In this scenario, the Desktop has three rows. The first row contains one full-width channel; the second row contains 2 channels, a thin plus a thick channel; and the third row contains 3 thin channels of equal width.

To create such a Desktop requires a custom container channel, created from the JSPs for the table container (JSPTableContainer).

To enable the Layout link to work with this container, you need a new layout channel with a customized JSP for editing and processing the Edit page. You build this by starting from the `layoutedit.jsp` and `layoutdoedit.jsp` files.

You also need to create a new custom table container provider by extending JSPTableContainerProvider.

Changing the Desktop Column Layout

▼ To Change the Desktop Column Layout from the Command Line

1 Modify the `width` property to change the layout of the channel column. Specify one of the following:

thin	The channel appears in one of the thin columns.
thick	The channel appears at the top spanning the entire horizontal space.
full_top	The channel appears at the top spanning the entire horizontal space.
full_bottom	The channel appears at the bottom spanning the entire horizontal space.

For example, the following specifies that a channel appears at the top spanning the entire horizontal space:

```
<String name="width" value="full_top”/>
```

2 Load the display profile into LDAP by using the `psadmin` subcommand.

See “Editing the Display Profile” on page 31 and *Sun Java System Portal Server 7.1 Command Line Reference*.

◆ ◆ ◆ CHAPTER 8

Branding the Desktop

This chapter describes how to brand the Desktop with your site's logo and name. It contains the following sections:

Changing the HTML Title (Title That Appears in the Browser)

The title is in the `productName` property in the display profile definition for all the providers and channels. Edit this property to change the HTML title.

Changing the Logo (Image) in the Banner Header

The logo image is defined in the themes in the display profile. The related theme properties are:

`brandImage`	The brand image on the left of the header.
`brandImage2`	The brand image in the center of the header; if there's no need to have the second image, then use `spacer.gif`.
`brandImageBgColor`	The background color for the left image file.
`brandImage2BgColor`	The background color for the center image file.
`brandImageWidth`	The width of the left image file.
`previewImage`	The image that is displayed in the Theme/preset Themes page.

▼ To Change the Logo (image) in the Banner

1. **Log in to the Sun Java System Portal Server management console.**
2. **Download the display profile XML fragment.**
3. **Modify the relevant theme properties and upload the display profile.**
4. **Copy the new image into** *PortalServer-base*`/web-src/images` **directory.**
5. **Run** *PortalServer-base*`/bin/psadmin redeploy --adminuser` *amadmin* `--passwordfile` *passwordfile* `--portal` *portal-ID* **command to deploy the new image.**
6. **Reload the Desktop to verify the change.**

Changing the Header and Footer of the Theme, Content, and Layout Pages

Depending on the Desktop, the particular header and footer files for in the Theme, Content, and Layout pages are determined by the container that contains the Theme, Content, and Layout pages.

For example, when you access the Content page for JSPTabContainer, JSPContentContainer is the container that is used to include the header and footer files that the tab container is using. The `contentedit.jsp` file, located in the JSPContentContainer directory, uses logic, based on the container, to access the appropriate header and footer files.

To brand the header and footer of the Theme, Content, and Layout pages:

Use "Changing the Header and Footer of the Theme, Content, and Layout Pages" on page 62 to determine the appropriate header and footer JSP files to edit. This two column table lists the Desktop containers in the first column and the corresponding header and footer files in the second column.

TABLE 8–1 Header and Footer Files for Theme, Content, and Layout Pages

Desktop Container	Header and Footer Files for Theme, Content, and Layout Pages
FrameTabContainer	`framePreferenceHeader.jsp` and `framePreferenceMenubar.jsp`
JSPSingleContainer	`singlePreferenceHeader.jsp` and `singlePreferenceMenubar.jsp`
JSPTabContainer	`tabPreferenceHeader.jsp` and `tabPreferenceMenubar.jsp`

TABLE 8–1 Header and Footer Files for Theme, Content, and Layout Pages *(Continued)*

Desktop Container	Header and Footer Files for Theme, Content, and Layout Pages
JSPTableContainer	`tablePreferenceHeader.jsp` and `tablePreferenceMenubar.jsp`

Containers not listed in "Changing the Header and Footer of the Theme, Content, and Layout Pages" on page 62 use `defaultHeader.jsp` and `defaultMenubar.jsp` files. These two files are actually the same as `singlePreferenceHeader.jsp` and `singlePreferenceMenubar.jsp` files. If you want a default look and feel for the container's header and menubar, customize these two JSPs. Currently, the sample portal does not use `defaultHeader.jsp` and `defaultMenubar.jsp` files.

▼ To Change the Header and Footer of the Theme, Content, and Layout Pages

1. **Change to the appropriate directory.**

   That is, change to *PortalServer-DataDir*`/portals/`*portal-ID*`/desktop/developer_sample` or change to the specific desktop type subdirectory associated with the target user or organization.

2. **Edit the JSP files.**

   For example, change the HTML title and logo in the header file, and change the product name in the footer.

3. **Run the `touch` command.**

   For example, type **`touch *.jsp`**.

4. **Reload the Desktop to verify the change.**

◆ ◆ ◆ CHAPTER 9

Changing Desktop Colors

This chapter describes how to change the color for various Desktop components, such as header, footer, font color in the header and footer, and so on.

This chapter contains the following:

Changing Desktop Colors

Most of these colors are part of the global theme attributes. See *Sun Java System Portal Server 7.1 Technical Reference* for more information.

▼ To Change the Desktop Colors

1 **Use the following table to determine what you want to change and what file you need to change. This two column table lists the Desktop component to customize in the first column, the files to edit in the second column, and the corresponding theme attribute in the third column.**

Desktop Component to Customize	File to Edit in *PortalServer-DataDir*/portals/*portal-ID*/desktop/default/ Directory	Theme Attribu
Header background color	JSP-based containers: ■ ./JSPSingleContainerProvider/header.jsp ■ ./JSPTabContainer/header.jsp ■ ./JSPTableContainerProvider/header.jsp ■ ./PredefinedFrontPageFramePanelContainerProvider/header.jsp ■ ./PredefinedFrontPageTabPanelContainerProvider/header.jsp ■ ./PredefinedSamplesTabPanelContainerProvider/header.jsp Frame-based containers: ■ ./FrameTabContainer/banner.jsp ■ ./PredefinedSamplesFramePanelContainerProvider/header.jsp	brandBgCol
Footer background color	JSP-based containers: ■ ./JSPSingleContainerProvider/menubar.jsp ■ ./JSPTabContainer/menubar.jsp ■ ./JSPTableContainerProvider/menubar.jsp ■ ./PredefinedFrontPageFramePanelContainerProvider/menubar.jsp ■ ./PredefinedFrontPageTabPanelContainerProvider/menubar.jsp ■ ./PredefinedSamplesTabPanelContainerProvider/menubar.jsp Frame-based containers: ■ ./FrameTabContainer/menubar.jsp ■ ./PredefinedSamplesFramePanelContainerProvider/menubar.jsp	brandBgCol
Font color in the header and footer	The related JSPs are the header.jsp and menubar.jsp that listed in header background color and footer background color.	headerFont
Selected tab color	JSP-based containers: ./JSPTabContainer/selectedTab.jsp Frame-based containers: ./FrameTabContainer/selectedTab.jsp	titleBarCo
Content Page color	./JSPContentContainer/contentLayoutBar.jsp ./JSPEditContainer/contentLayoutBar.jsp ./JSPLayoutContainer/contentLayoutBar.jsp ./TabJSPEditContainer/contentLayoutBar.jsp	(none)
Layout Page color	./JSPLayoutContainer/layoutedit.jsp	(none)
Desktop body	./JSPTableContainerProvider/tabtable.jsp	tableBgCol

2 Edit the appropriate file.

In almost all case, make modifications to the bgcolor=value statement to change the color. In the case of the font color in the header and footer, change the color inside of the FONT tag for the specific link.

3. **In the directory where you make the change, run the following command:**

   ```
   touch *.jsp
   ```

   (Or, if you know the parent JSP file, just run the `touch` command on that file.)

4. **Reload the Desktop.**

Changing the Default Color Scheme for an Organization

There are two ways in which to provide a new color scheme and layout for an organization:

- Define a new set of templates - You can define a new set of templates in *PortalServer-DataDir*`/portals/`*portal-ID*`/desktop/new/` and make this directory (new) the Desktop Type attribute for that organization.
- Define a new theme - In the display profile, you can define your own theme in the GlobalThemes collection. See Chapter 10 for more information.

◆ ◆ ◆ CHAPTER 10

Customizing the Global Themes

This chapter contains the following:

Customization Overview

There are two levels of customization for the themes:

The number of themes and theme attributes are configurable by the administrators. Theme and theme attributes are display profile properties; so they can be edited in the display profile directly. The theme properties are defined as global properties in the organization level in the sample portal. So, when a new theme is created, all users in the organization will see it.

The end user can select one of the preset themes that are defined by the administrator, or customize some theme attribute values inside of the theme page in the Desktop. When the theme changes, it applies to all the containers in the Desktop, and also, the changed property will be stored in the user level display profile.

There are tag library functions defined to allow JSPs to retrieve the theme related values from the display profile. Behind the scene, the tag library functions use the Theme Java class to get the theme properties. For more information on the Theme Java class, please see the Java docs for `com.sun.portal.providers.context.Theme`.

When you add (or customize) a global theme, all channels see the change, as themes are a global property for all channels.

See the *Sun Java System Portal Server 7.1 Technical Reference* for more information on the Global themes

Adding a Theme to the Sample Portal

▼ To add a Theme to the Developer Sample

1. **Develop the display profile XML definition for the new theme and ensure that the new collection has all thirty eight (38) properties defined in the display profile.**

   The collection can be added either using the Portal Server management console, or using the `psadmin` subcommand.

2. **Copy new images into the** *PortalServer-base*`/web-src`**.**

   For example:

   - `activeBulletImage`, `inactiveBulletImage`, `brandImage`, `brandImage2`, `previewImage`: these images must be copied in to the *PortalServer-base*`/web-src/images` directory.
   - `helpImage`, `removeImage`, `minimizeImage`, `maximizeImage`, `normalImage`, `attachImage`, `detachImage`, `editImage`: these images must be copied in to the *PortalServer-base*`/web-src/desktop/images` directory.
   - `tabNotchImage` must be copied in to the *PortalServer-base*`/web-src/desktop/tabs/images` directory.

3. **Run the** *PortalServer-base*`/bin/psadmin redeploy --adminuser` *amadmin* `--passwordfile` *passwordfile* `--portal` *portal-ID* **command to deploy the image files.**

   See the *Sun Java System Portal Server 7.1 Command Line Reference* for more information on the `psadmin` subcommand.

4. **Verify that the new theme shows up on the Desktop's Theme page.**

Customizing the Current Themes

Change the theme values in the display profile. You can modify the theme properties from the administration console, or by using the `psadmin` subcommand to load the XML fragment.

Changing in the administration console is easier, since you want to pin point some specific properties, and change the values.

To Change the Text

The font families and font sizes are combined and defined in the following theme attributes.

- `headerText`
- `titleText`

- `contentLayoutText`

The value of these attributes is actually a class defined in the desktop *PortalServer-base*`/web-src/desktop/css/style.css` file. In the Desktop `style.css` file, there are predefined font family and font size as follows:

```
.sansSerif12Font { font-family: sans-serif; font-size: 12pt }
.sansSerif11Font { font-family: sans-serif; font-size: 11pt }
.sansSerif10Font { font-family: sans-serif; font-size: 10pt }
.sansSerif9Font { font-family: sans-serif; font-size: 9pt }
.sansSerif8Font { font-family: sans-serif; font-size: 8pt }
.sansSerif6Font { font-family: sans-serif; font-size: 6pt }
.monospace12Font { font-family: monospace; font-size: 12pt }
.monospace11Font { font-family: monospace; font-size: 11pt }
.monospace10Font { font-family: monospace; font-size: 10pt }
.monospace9Font { font-family: monospace; font-size: 9pt }
.monospace8Font { font-family: monospace; font-size: 8pt }
.monospace6Font { font-family: monospace; font-size: 6pt }
.serif12Font { font-family: serif; font-size: 12pt }
.serif11Font { font-family: serif; font-size: 11pt }
.serif10Font { font-family: serif; font-size: 10pt }
.serif9Font { font-family: serif; font-size: 9pt }
.serif8Font { font-family: serif; font-size: 8pt }
.serif6Font { font-family: serif; font-size: 6pt }
.verdana12Font { font-family: verdana; font-size: 12pt }
.verdana11Font { font-family: verdana; font-size: 11pt }
.verdana10Font { font-family: verdana; font-size: 10pt }
.verdana9Font { font-family: verdana; font-size: 9pt }
.verdana8Font { font-family: verdana; font-size: 8pt }
.verdana6Font { font-family: verdana; font-size: 6pt }
```

To change the font for the header text, title text, and content and layout text, please use one of the predefined class name, or, add new class definition in the `style.css` file, and then use it.

◆ ◆ ◆ CHAPTER 11

Customizing the Service Providers

This chapter provides common customization tasks for modifying the Search provider and Discussion provider.

This chapter contains the following sections:

Overview of Customizing the Service Providers

The Portal Server software includes a search service and discussion service provider.

Overview of Customizing the Search Provider

The Search provider (SearchProvider) furnishes a basic reference user interface that contains both search and browse functionality. Search functionality includes basic search mode, and advanced search for more complex searches. You can perform specific field searches in advanced search mode. For example, while in advanced mode, you can search within the title, URL, last modified date, author, and so on.

SearchProvider provides a link for category browsing. In addition, you can create a taxonomy for the Search Engine along with category filter rules. You can browse through the taxonomy tree and view documents within a category through the Search provider interface.

Search Provider Design

The Search provider uses JSPProvider to access the Portal Server back end services. The Search provider users JavaServer Pages™ (JSP™) helper tag libraries to avoid using Java™ scriptlets.

The `searchServer` is a global service list type attribute that is configured and updated at installation time. The Search provider is responsible for directing the search request to the appropriate back end Search Engine server.

See the *Sun Java System Portal Server 7.1 Technical Reference* for the display profile properties you can set for the provider.

Search JavaServer Pages and Tag Libraries

The Search provider consists of two stages (input form and results) and the JSPs used by the Search provider fall into one of those two stages.

The following Example 11–1explains the JSP layout for `searchContent.jsp`. In the input stage (Stage 1), `searchContent.jsp` makes use of `searchMenu.jsp` and `psSearch.jsp` to set up the initial interface. The `basicSearch.jsp` file is used for a basic search and `advancedSearch.jsp` file for an advanced search. The description menu (that is, the Full, Brief, and Title menus) is displayed for both basic and advanced searches by `descMenu.jsp` file. The `browseHeader.jsp` file defines the browse interface.

In the results stage (Stage 2), one of three JSPs is used: `browseOnly.jsp` file sets and executes the parameters for category browsing using the Search tag library, and includes the `browseResults.jsp` page; `browseSearch.jsp` file sets and executes the parameters for searching and browsing within categories using the Search tag library and includes `browseSearchResults.jsp`; and `searchOnly.jsp` file sets and executes the parameters for search using search the Search tag library and includes `results.jsp` and `score.jsp` for the match relevance. The `pageFooter.jsp` file displays the list of pages, Next, and Previous links.

EXAMPLE 11–1 JSP Layout for `searchContent.jsp`

```
searchContent.jsp
    |
    |------>searchMenu.jsp
    |
    |------>psSearch.jsp
    |
    |------>basicSearch.jsp----->descMenu.jsp (if search mode is basic)
    |
    |------>advancedSearch.jsp-->descMenu.jsp (if search mode is advanced)
    |
    |------>browseHeader.jsp (if browse mode is selected)
    |
    |                    STAGE 1 INPUT
    ____________________________________________________________________________
    |    |
    |    |                STAGE 2 INPUT
    |    |
    |    |--->browseOnly.jsp----->browseResults.jsp (category browsing in browse mode)
    |
    |------>browseSearch.jsp--->browseSearchResults.jsp (only for category search)
    |
    |------>searchOnly.jsp----->results.jsp--->score.jsp(displays the match relevance)
    |
    |------>pageFooter.jsp
```

See the *Sun Java System Portal Server 7.1 Technical Reference* for more information on the Search JSP files.

The Search JSPs use the following tag libraries, which ship with the Portal Server software:

- Jakarta tag library, for any generic tags
- Desktop tag library, for the Portal Server software related information
- Search tag library to cover all Search Engine server access functionality and to provide a tag-based wrapper for the existing public search and SOIF API

See the *Sun Java System Portal Server 7.1 Technical Reference* for more information.

Overview of Customizing the Discussion Provider

The DiscussionProvider is JSPProvider based and hence customizable. It uses the Desktop themes. It retrieves data from the back end Search service using search taglibs and API. The discussions and comments are stored as separate Resource Descriptors (RDs) in the discussion database.

The DiscussionProvider includes features such as discussion threads, starting discussions based on documents or new topics, searching discussions, and rating discussions. By default, the Discussions channel is available on the sample portal for anonymous users. However, an anonymous user cannot subscribe to a discussion or edit the Discussion channel.

The DiscussionProvider supports a full view (through the Discussions channel) and a lite view (through the DiscussionLite channel.) It has the following main functions:

- Start a new discussion from the discussion channel.
- Start a new discussion based on web documents from the search channel.
- Add a comment to an existing discussion or post a reply to an existing discussion.
- Rate all discussions and comments. Note that the displayed ratings are based on an algorithm such that the rating for any comment goes up gradually. For example, a comment has to be rated important three times before it is marked as important.
- Search all discussions and search within a discussion. These functions are routed to the search provider. The `displaySearch` property can be disabled if the search feature is not required in the discussion channel. Users can also search by rating in Advance Search.
- Authenticated users can choose to subscribe to a particular discussion by selecting the subscribe link. The request is handled by the SubscriptionProvider. The `displaySubscription` property can be disabled if the feature is not required. By default, the value is true.

A Discussion Lite view retrieves main posts sorted by last-modified date and has pagination so users can access older discussions. View discussion displays each discussion subtree. The main item is displayed in detail and the subtree is displayed below the main item.

View discussion includes:

- Several filters on the page. A document display can be based on filters such as document rating (irrelevant, routine, interesting, important, and must read).
- Display preference can be set to threaded or flat display.

Expansion threshold helps to control displayed items in the subtree. The users can choose to expand only highly rated documents, or expand all or collapse all. Default value is collapse all. Expand all will expand all the filtered comments. It will also show a description of the discussion, provide a menu for rating the discussion, and allow the user to post a reply.

Discussion Service Channels

The DiscussionLite channel and the Discussions channel are based on the DiscussionProvider.

DiscussionLite Channel

The DiscussionLite channel displays the top twenty discussion titles (which can be reconfigured) and the date. The discussions are sorted by creation date (last modified) and the

newest discussion is displayed first. The DiscussionLite channel view has links to view each discussion, view all discussions, and start a new discussion. All these links target the Discussions channel which gets displayed in the JSPDynamicSingleContainer.

Properties for this channel can be configured from the administration console. By default, there are no user editable properties for this channel.

Discussions Channel

The Discussions channel includes a full view that:

- Shows detailed descriptions for the top eight discussions sorted in descending order. This can be reconfigured from the channel edit page.
- Includes pagination so that users can see all the discussions.
- Supports search. The search returns discussion and comment results.

Discussion JavaServer Pages and Tag Libraries

Similar to the search channel JSPs, the discussion channel JSPs have a query portion, a display portion, and use Desktop themes.

For more information on the channel specific JSP files, see the *Sun Java System Portal Server 7.1 Technical Reference*.

Tips for Customizing the Service Providers

This section provides some basic tips for customizing the search and discussion providers.

Debugging the Service Providers

The Portal Server software provides files to help debug the Search and Discussion providers.

The following directory contains various search log files:

PortalServer-DataDir`/searchservers/search1/logs`

The following search log file records the search query sent to the Search Engine by the Search server:

PortalServer-DataDir`/searchservers/search1/logs/rdm.0.0.log`

Location of JavaServer Pages

JavaServer Pages for the Search channel are in the *PortalServer-DataDir*/portals/*portal-ID*/desktop/default/SearchProvider directory.

JavaServer Pages for the DiscussionLite channel are in the *PortalServer-DataDir*/portals/*portal-ID*/desktop/default/DiscussionLite directory.

JavaServer Pages for the Discussions channel are in the *PortalServer-DataDir*/portals/*portal-ID*/desktop/default/DiscussionProvider directory.

Modifying JavaServer Pages

When you modify statically included JavaServer Pages, be sure to run the touch command, otherwise no changes are reflected. You need to either run the touch command on the top-level JSP file or on all JSP files. For example,

```
touch searchContent.jsp
```

or

```
touch *.jsp
```

See also "JavaServer Page Caching Information" on page 91 and "Recompiling JSPs" on page 90.

Accessing Channels Directly

You can access the search channel directly at the following URL:

```
http://server:port/portal/dt?provider=JSPDynamicSingleContainer&JSPDynamicSingleContainer.selectedChannel=Search&last=fa
```

Modify all the links to use these extra parameters in the URL. For example, edit searchMenu.jsp file as follows:

```
<nobr>  <a class=noUnderline href="<%=dpurl%>?mode=basic">Basic Search</a>  </nobr>
```

Replace the bold portion with:

```
http://server:port/portal/dt?provider=JSPDynamicSingleContainer&JSPDynamicSingleContainer.selectedChannel=Search&last=fa
```

You can access the Discussion channel directly at the following URL:

```
http://server:port/portal/dt?provider=JSPDynamicSingleContainer&JSPDynamicSingleContainer.selectedChannel=Discussions&la
```

Customizing the Search Provider

This section describes how to perform some common customizations on the Search provider.

▼ To Add last-modified to the Search Result Display

1. **Modify `searchOnly.jsp` file by adding last-modified to the list of `viewAttributes`.**

   For example:

```
<search:setViewAttributes viewAttributes="hl-url,hl-title,hl-description,score,content-length,hl-classification,
```

2. **Modify `results.jsp` file to display the last-modified date for document results using the SOIF `getValue` tag.**

   For example:

```
<% if (formbean.getDescription().equals("full")) { %>
    <FONT color=<%=tFontColor%> face=<%=tFontFace%>><search:getValue soifAttribute="description" escape="false"/>
    <FONT color=#707070 face=<%=tFontFace%>><search:getURL escape="true"/><BR>
    <search:getValue soifAttribute="content-length" id="sz"/>
    <search:getValue soifAttribute="last-modified"/><BR>
<% } %>
```

3. **Run the `touch` command.**

   For example, type **`touch *.jsp`**.

4. **Reload the Desktop to verify the change.**

▼ To Remove content-length from Search Results

1. **(Optional) Modify `searchOnly.jsp` file by removing content-length from the list of `viewAttributes`.**

   The line to modify is the following:

```
<search:setViewAttributes viewAttributes="hl-url,hl-title,hl-description,score,content-length,classification hl-
```

   Remove content-length from this line.

2. **Modify `results.jsp` file by removing the line that displays the content-length.**

   Comment the lines as shown here:

```
<%--
    <search:getValue attribute="content-length" id="sz"/>
    <jx:declare id="sz" type="java.lang.String"/>
    <%
```

```
                int sizek = 1;
                try {
                    sizek = Math.max(1, (Integer.parseInt(sz) + 512)/1024 );
                } catch (Exception e) {
                }
            %>
            <i18n:message key="size" bundle="searchjsp"/>
            <%= sizek %>k 
        --%>"
```

3 **Run the `touch` command.**

For example, type `touch *.jsp`.

4 **Reload the Desktop to verify the change.**

▼ To Remove author from the Advanced Search Interface

1 **Comment out or remove the author related HTML from the `advancedSearch.jsp` file.**

For example:

```
<!--     -->
<!-- To disclude the "author" row, remark out the following section -->
<!--     -->
<TR>
    <td valign=middle align=right height=40><FONT color=<%=tFontColor%> face=<%=tFontFace%>><nobr> <LABEL FOR="advAuth
    <SELECT NAME="authorOp">
    <OPTION VALUE=<%=SearchContext.CONTAIN%> <%=formbean.authorOpSelection(SearchContext.CONTAIN)%>>does</OPTION>
    <OPTION VALUE=<%=SearchContext.NOTCONTAIN%> <%=formbean.authorOpSelection(SearchContext.NOTCONTAIN)%>>does not</OP
    </SELECT>contain </FONT></nobr></TD>
    <td valign=middle align=left height=40><INPUT TYPE="text" NAME="authorVal" id="advAuthor" VALUE= "<%=SearchContext
</TR>
```

2 **Comment out author- related lines in `advQuery.jsp` file.**

```
if (!formbean.getAuthorVal().equals("")) {
    h = new HashMap();
    h.put(SearchContext.OPERAND, "author");
    h.put(SearchContext.OPERATION, formbean.getAuthorOp());
    h.put(SearchContext.VALUE, formbean.getAuthorVal());
    l.add(h);
}
```

3 **Run the `touch` command.**

For example, type **`touch *.jsp`**.

4 **Reload the Desktop to verify the change.**

▼ To Add a New Field to Advanced Search

1 **Uncommment the keywords section in** `advancedSearch.jsp` **file.**

```
<!--     -->
<!-- To Include the "Keywords" row, unremark the following section -->
<!--
<TR>
<td valign=middle align=right height=40><FONT color=<%=tFontColor%> face=<%=tFontFace%>><nobr> <LABEL FOR="advKey
<SELECT NAME="keywordsOp">
<OPTION VALUE=<%=SearchContext.CONTAIN%> <%=formbean.keywordsOpSelection(SearchContext.CONTAIN)%>>does</OPTION>
<OPTION VALUE=<%=SearchContext.NOTCONTAIN%> <%=formbean.keywordsOpSelection(SearchContext.NOTCONTAIN)%>>does not
</SELECT>contain </FONT></nobr></TD>
<td valign=middle align=left height=40><INPUT TYPE="text" NAME="keywordsVal" id="advKeywords" VALUE= "<%=SearchC
</TR>
-->
```

Remove the `<!--` and `-->` comment marks from this section.

2 **Add the keywords to** `advQuery.jsp` **file.**

```
if (!formbean.getKeywordsVal().equals("")) {
  h = new HashMap();
  h.put(SearchContext.OPERAND, "Keywords");
  h.put(SearchContext.OPERATION,formbean.getKeywordsOp());
  h.put(SearchContext.VALUE, formbean.getKeywordsVal());
  l.add(h);
}
```

3 **Run the** `touch` **command.**

For example, type **`touch *.jsp`**.

4 **Reload the Desktop to verify the change.**

Customizing the Discussion Channels

This section contains the following:

▼ To Display Additional Fields in the List View of Discussions

1. **Change directories to** *PortalServer-DataDir*`/portals/`*portal-ID*`/desktop/default/DiscussionProvider`**.**

2. **Modify** `query.jsp` **file and add** `xxx` **field to** `viewAttributes`**.**

   For example, add content-length as follows:

```
<search:setViewAttributes viewAttributes= "url,title,description,rd-rating,author,last-modified,rd-last-changed,rd-ref
```

3. **Add the new fields in** `fullDiscussionDisplay.jsp` **file wherever appropriate.**

   For example:

```
<search:getValue soifAttribute="content-length"/>
```

▼ To Modify the Sort Order in List All Discussions Page

- **By default, discussions are sorted by the last-modified date/time. That is, discussions are displayed in a descending order with the latest or most recent discussion shown first.**

  To modify the sort order in list All Discussions page, modify the `viewOrder` property in `fullDiscussion.jsp` file. For example, you can reset the value below to `author` or `rd-last-changed`:

```
<jx:set var="viewOrder" value="-last-modified"/>
```

▼ To Inherit Classification and readACL

If you have classified only the parent discussion manually or modified the access control for the parent discussion, you may want to inherit those values in discussion replies as follows:

1. **Change directories to** *PortalServer-DataDir*`/portals/`*portal-ID*`/desktop/default/DiscussionProvider`**.**

2. **Edit** `feedbackProcess.jsp` **file and modify the values of** `inheritClassification` **and** `inheritReadACL`**.**

   By default, these are set to `false`. Reset them to `true` if you want comments to inherit the parent's classification field and `readACL` field. Note that comments are automatically protected in this case.

3. **Save the file.**

4. **Run the `touch` command.**

   For example, type `touch *.jsp`.

▼ To Control Access to Discussions

This can be accomplished by one of the following two ways:

- **Modify the `dbname` property in the display profile for Discussions channel for each role to point to a different database.**

  In this case users in one role cannot view discussions created by users in a different role.

◆ ◆ ◆ CHAPTER 12

Customizing the Desktop End-User Online Help

This chapter provides information on customizing the Portal Server Developer Sample desktop online help.

This chapter contains the following sections:

Overview of the Desktop End-User Online Help

The desktop end-user online help is a collection of HTML files that is referenced in the display profile. Each provider, including the various container providers, has a display profile entry for a corresponding help file. In the display profile, each provider definition has the default value for the help file. If a channel that uses the provider has a different help file, the `helpURL` property can be defined in the channel definition also, overriding the provider's value.

The display profile entries for the online help are defined in the provider properties. The entries define the string name, `helpURL`, and its value.

The `helpURL` property is a conditional property. Multiple values can be associated with the `helpURL` property and the display profile API returns the proper value depending on the client type and locale. If your portal server is configured to serve multiple clients (such as HTML, WML) in multiple locales (such as english, french), the `helpURL` property will allow you to set up multiple help files based on the type of client and type of locale you are serving.

EXAMPLE 12–1 HelpURL Property in Display Profile Definition

```
<ConditionalProperties condition="client" value="WML">
    <ConditionalProperties condition="locale" value="en">
        <String name="helpURL" value="en/wml/help.wml"/>
```

EXAMPLE 12–1 HelpURL Property in Display Profile Definition *(Continued)*

```
    </ConditionalProperties>
    <ConditionalProperties condition="locale" value="fr">
        <String name="helpURL" value="fr/wml/help.wml"/>
    </ConditionalProperties>
</ConditionalProperties>
<ConditionalProperties condition="client" value="HTML">
    <ConditionalProperties condition="locale" value="en">
        <String name="helpURL" value="en/html/help.html"/>
    </ConditionalProperties>
    <ConditionalProperties condition="locale" value="fr">
        <String name="helpURL" value="fr/html/help.html"/>
    </ConditionalProperties>
</ConditionalProperties>
```

The value of the `helpURL` property can be either a relative path or an absolute path. The location shown in the above example is relative to the Desktop static content root.

The static content root is the install directory of static content. By default the static content root is *PortalServer-base*`/`.

The relative path will be generated as:

static-content-root`/`*doc-root*`/`*locale*`/`*helpURL-value*

The doc root is defined in the display profile also. For example, if the doc root is `docs`, and the user's locale is `en_US`, with the `helpURL` value, the final value for the help location will be:

PortalServer-base`/web-src/docs/en_US/desktop/userinfo.htm`

The following is an example of an absolute URL that defines a help file location. Use a similar format for using an absolute URL to define the location of an online help file.

```
<String name="helpURL" value="http://sesta.com/docs/desktop/userinfo.htm"/>
```

Location of the Desktop End-User Online Help HTML Files

The source location of the Desktop end-user online help files for the sample Desktop is:

PortalServer-base`/web-src/docs/`*locale*`/desktop`

Modifying the Desktop End-User Online Help HTML files

You can customize the end-user online help by editing the existing HTML online help files or by creating new HTML files.

Editing An Existing Help File

You can edit an existing help file to customize the content to meet specific requirements of your organization. For example, you can remove or change the `SunONE.jpg` image or meta text that is currently displayed on the sample help files, or replace the help file completely with a file of the same name.

After modifying the file, run the *PortalServer-base*`/bin/psadmin redeploy --adminuser` *amadmin* `--passwordfile` *passwordfile* `--portal` *portal-ID* command to deploy the file into the web-app location.

This method of modifying the online help files is useful if you use the sample providers that are shipped with the Portal Server software.

For example, if you use the UserInfoProvider that ships with the Portal Server product, the display profile for that provider already defines the `helpURL` value as `desktop/userinfo.htm`. By editing the help file `userinfo.htm`, no changes to the display profile are necessary.

Creating a New Help File

You can create a new help file by creating a new HTML file. This method of customizing the online help is useful if, for example, you add a new provider to the Desktop. When you create a new help file, you must modify the display profile to contain the new `helpURL` value.

▼ To Create a New Online Help File and to Define the `helpURL` Value

1. **Create an HTML file for the provider you want to document.**

2. **Place your file in the appropriate directory.**

   You can place your custom help files under the web server document root, in the directory specified as root by the display profile:

   PortalServer-base`/web-src/docs/`*locale*`/desktop`

   Or, you can deploy them in a custom web application archive. See the web server documentation for information on how to deploy a web application archive.

3. **Run the** *PortalServer-base*`/bin/psadmin deploy` **subcommand to deploy the file.**

4. **Define the `helpURL` value for that file in the display profile.**

   To define the `helpURL` value for a new online help file, use the format described in the section "Overview of the Desktop End-User Online Help" on page 85.

5. **Use the `psadmin` subcommand to load the display profile into LDAP.**

6. **Verify that the new help file is displayed correctly.**

◆ ◆ ◆ CHAPTER 13

Miscellaneous JSP and Template Information

This chapter contains the following sections:

This chapter contains miscellaneous information on using JavaServer Pages™ (JSP™) and templates when customizing the Desktop.

Performing JSP Redirects

To perform redirects from the `doedit.jsp` page back to the `edit.jsp` page, use the following URL:

```
response.sendRedirect(p.getProviderContext().getDesktopURL(request).toString() +
 "?action=edit&provider=ipsdtJSPEditChannel" + "&targetprovider=" +
 p.getName() + "&reedit=true");
```

To return back to the Edit page, use the following, which does not hardcode the edit channel name.

```
String editChannel = request.getParameter("editChannelName");
response.sendRedirect(p.getProviderContext().getDesktopURL(request).toString() +
 "?action=edit&provider=<%=editChannel%>&targetprovider=" + p.getName() +
 "&redit=true");
```

JSP or Theme Color

The colors of a theme can be changed in two places: in the JSP or template file that uses the theme tag, and in the display profile using the GlobalThemes attribute.

If you change the color value in the JSP or template file, the Desktop uses this value and not what is in the GlobalThemes attribute.

For example, if you manually change the following background color in a JSP file

```
BGCOLOR="<dttheme:getAttribute name="borderColor"/>"
```

to

```
BGCOLOR="#FFFFFF"
```

then the Desktop shows white for border color, and the theme color is not used. There is no way the theme or the Desktop can detect this.

Recompiling JSPs

Every time you make a modification to a JSP file, you need to recompile. You do this by running the `touch` command on the modified container's top-level JSP file.

For example, type **`touch tab.jsp`**.

Note – A typical desktop will include content from several JSPs. However, if you make a modification to a non-toplevel JSP that has been included in a top level JSP, the included JSP will not be recomplied. The end result is your desktop changes will never be reflected.

If the top-level JSP is touched, the JSP engine recompiles all the relevant JSPs. If you cannot find the top level JSP, run the `touch` command on all JSPs in the directory, for example,

```
touch *.jsp
```

This modifies all JSPs, including the top-level JSP.

See "JavaServer Page Caching Information" on page 91 for information on how to find the top-level (parent) JSP.

JavaServer Page Caching Information

When the system compiles JavaServer Pages™ (JSP), the result is only one Java™ class per parent JSP. There is no compiled class for the static included JSPs. Thus, when you change an included JSP, you need to run the touch command on the parent JSP to recompile the parent JSP with the changed JSP. The Portal Server software JSP engine checks the last modification time on the compiled class and the JSP file to see if the JSP needs to be recompiled. In this way, the change takes effect immediately.

To find a JSP's parent JSP, search in the JSP for the string `<%@ include file="filename.jsp" %>`. Some JSPs are dynamically included by using the `<jsp:include page="header.jsp" flush="true"/>` syntax instead of `<%@ include file="header.jsp" %>`. This syntax compiles `header.jsp` and generates a separate Java class.

The path to cached JSPs is constructed in such a way so that the compiled JSPs do not conflict with each other in multi-server instances, when multiple Desktop types contain the same JSPs and for multiple clientTypes and locales. So when JSPs are dynamically included, the touch command does not need to be run for the parent JSP.

Debugging JSPs

The JSP classes are created at *PortalServer-DataDir*`/portals/`*portal-ID*`/desktop/compiled` directory.

You can find compilation and runtime errors in the Desktop debug log at *PortalServer-DataDir*`/portals/`*portal-ID*`/logs/`*portal-instance*`/portal.0.0.log`.

Also, all JSPProvider based Desktop channels have a property called `showExceptions`. This property, by default, is set to `false`; setting it to `true` causes the JSP exception to show up as the content of the channel.

Dynamic Template Reloading

If you make changes to the Desktop templates, note that these templates are dynamically reloaded. The reload interval is by default set to thirty seconds. You can change the reload interval in the `desktopconfig.properties` file at *PortalServer-DataDir*`/portals/`*portal-ID*`/config/` directory.

▼ To Change the Template Reload Interval

1 **Log in to the Portal Server host and change directories to** *PortalServer-DataDir*`/portals/`*portal-ID*`/config`**.**

2. **Open the `desktopconfig.properties` file and reset the `templateScanInterval` property value.**

3. **Save and close the file.**