Creating and Using Page Templates in Oracle WebCenter Portal Applications

Oracle WebCenter Portal 11g R1 PS5 (11.1.1.6.0)
Objectives

After completing this module, you should be able to:

• Create and use page templates both at design- and run time
• Create an editable page template
• Create a portal resource from an existing page template
• Explain the round-trip development process for page templates
• Describe the differences between page templates for an Oracle WebCenter Portal Framework application and Oracle WebCenter Portal: Spaces
Agenda

• Working with Page Templates
• Page Templates for WebCenter Portal Framework Applications
• Page Templates for WebCenter Portal: Spaces
What is a Page Template

A page template controls the layout and common content of the application’s pages.
Typical Elements on a Page Template:
Layout Elements

Header

Sustainability through natural building

Content area

Design & Build

El Piju’s Design & Build project delivery method offers the security of a single-source of responsibility for both the design and construction of your project. On a Design & Build project, El Piju engages the client prior to the start of design professionals and contractors, creating a “win-win” partnership environment. Key benefits include an early guarantee of project costs, a shorter overall project schedule, and fewer burdens on your management resources.

Footer
Typical Elements on a Page Template: Brand-Specific Elements

- Logo
- Slogan
- Copyright message
Typical Elements on a Page Template: Navigation

Global navigation

Additional navigation
Typical Elements on a Page Template: Additional Links and Actions

- Login/Logout
- Pop-up menus
- Global links
Typical Elements on a Page Template: Conditional Elements

Elements of the page template can be displayed conditionally.

- Depending on whether the user is public or authenticated

- Depending on the user’s role and privileges
Working with Templates at Design Time

Create Template (1)

1. New Template

2. Define name, create page definition
Working with Templates at Design Time
Create Template (2)

3. Create facet(s)

4. Create attribute(s)

JSF Page Template wizard is not reentrant, but you can edit the template source to add more facets and attributes.
Structure of the Template Source

<?xml version='1.0' encoding='UTF-8'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1" ... 
xmlns:cust="http://xmlns.oracle.com/adf/faces/customizable">
<jsp:directive.page contentType="text/html;charset=UTF-8"/>
<af:pageTemplateDef var="attrs">
<af:facetRef facetName="content"/>
<af:xmlContent>
<component xmlns="http://xmlns.oracle.com/adf/faces/rich/component">
<display-name>Corporate Template</display-name>
<facet><facet-name>content</facet-name></facet>
<attribute>
<attribute-name>contentWidth</attribute-name>
<attribute-class>java.lang.String</attribute-class>
<default-value>960px</default-value>
</attribute>
</component>
</af:xmlContent>
</af:pageTemplateDef>
</jsp:root>
Working with Templates at Design Time: Populate Template

- Add components to the template (similar to populating pages)
  - Drag-and-drop from the component palette
  - Edit the source if necessary
- Add facet reference where the page content will be displayed
  
  <af:facetRef facetName="content"/>
Template Attributes

• Define attributes in the template:
  name, class, and optionally, a default value

  <attribute>
    <attribute-name>contentWidth</attribute-name>
    <attribute-class>java.lang.String</attribute-class>
    <default-value>960px</default-value>
  </attribute>

• Default value can be an EL expression
• Default can be overridden,
  – in the page, where the template is used
  – At run time by Composer, when the page is displayed
• Refer to an attribute:

  <trh:tableLayout id="globalPageTable"
    width="#{attrs.contentWidth}"
Page Template as Portal Resource

• A page template can become a portal resource.
  – Packed in a resource archive (EAR, MAR)
  – Stored in MDS

• Rules:
  – Create under
    /oracle/webcenter/portalapp/pagetemplates
  – Must have a facet called content
  – Must have a reference to the content facet
Page Template as Portal Resource: Creating the Resource

1. Create resource

2. Define properties
   - Name
   - optionally
   - Description
   - Icon URI, …
Page Template as Portal Resource: Working with Resources

- Update, remove

- Export
  - Provide location (EAR file)
  - Include the content directory, if necessary
Page Template as Portal Resource: Importing a Resource

1. Menu on the project

2. Provide location (EAR file)

After importing a page template, you need to edit `pagetemplate-metadata.xml` to register the newly imported template for JDeveloper.
Round-Trip Development Process for Page Templates

- Create page template in JDeveloper
- Edit page template in JDeveloper
- Test page template in running application
- Export page template as a resource
- Upload page template resource
- Create page template in Resource Manager
- Edit page template with Composer
- Test page template in running application
- Download page template

Development team

Portal resource archive

Design Time

Run Time
Working with Templates at Run Time

Use: Administration > Resources > Page Templates
- Create, Upload, Copy
- Delete
- Download
- Edit, Edit Source, Edit Properties, Show
Page Template Resource Archive

- Wrapper EAR (ZIP) file
- MDS archive

Corporatetemplate.ear

transport.mar

/o racle/webcenter/portalapp/pagetemplates
  - CorporateTemplate.jsp
  - CorporateTemplatePageDef.xml
/o racle/webcenter/portalapp/shared
  - ...
/o racle/webcenter/siteresources/scopedMD/id
  - generic-site-resources.xml
Template Content Tips (1)

- Use CSS or JavaScript
  - Embedded in the template
    ```
    <af:resource type="css"> ... </af:resource>
    <af:resource type="javascript"> ... </af:resource>
    ```
  - Add CSS to the skin
  - Use external files
    ```
    <af:resource type="css" source="url" />
    <af:resource type="javascript" source="url" />
    ```

- Use JavaSever Pages Standard Tag Library
  - Add namespace
    ```
    xmlns:c="http://java.sun.com/jsp/jstl/core"
    ```
  - Add source
    ```
    <c:if test="..."> ... </c:if>
    ```
  - Drag from Component Palette
Template Content
Tips (2)

• Use variables
  
  `<c:set var="contextRoot"`
  
  `value="${facesContext.externalContext.requestContextPath}"`
  
  `scope="session"/>

  `<af:goLink id="home"`
  
  `destination="/#{contextRoot}/spaces/
  #{$spaceContext.currentSpaceName}" />

• Conditional display
  – ADF tags, use rendered attribute
    
  – Other tags, surround with conditional JSTL tags

  `<c:if test="..."> ... </c:if>

  `<c:choose>
    <c:when test="..."> ... </c:when>
    <c:otherwise> ... </c:otherwise>

  </c:choose>`
Template Content: Use Additional Files

Page template might refer to external files: images, JavaScript, CSS.

- Deploy separately, use fix context root
- Deploy together with the application
  - Place under the context root
  - Use PortalWebAssets project
  - Does not work with packaged resources
- Use the `/oracle/webcenter/portalapp/shared` folder
  - Shared folder content packed into the resource archive
  - In PS3, PS4, only images files are accessible
  - In PS5, you can use other file types, for example CSS or JavaScript; you can use folder hierarchy
Editable Templates

A page template can be edited at run time if it contains editable components.

Panel Customizable

- Container with horizontal or vertical layout
- Holds other components

Show Detail Frame

- Chrome for an actual component
- Add actions, like show, hide, move
- Enables editing the frame or the embedded component properties
Editable Templates: Design Time

- Add from the Oracle Composer component group
- Must not add Page Customizable
- Add at least one Panel Customizable
- Add ADF components surrounded by Show Detail Frame
  - Output Text
  - Rich Text Editor
  - Go Image Link
  - Go Link
  - …
Editable Templates: Source Example

1. Add Panel Customizable

2. Wrap components into Show Detail Frame

3. Set this attribute that Show Detail Frame will customize the embedded component

4. Use ADF component, Output Text

5. Rich Text Editor component No need for the attribute here
Editing Templates at Run Time

• Resource Manager

• Composer
Editing Templates: Run-Time Example
Editing Templates: Run-Time Example
Editing Templates: Run-Time Example
Editing Templates: Run-Time Example
Editing Templates: Run-Time Example

- Add Box
- Edit properties
- Edit HTML fragment

- Resource Catalog
- Rich Text Editor
Editable Templates: Resource Catalog

Web development components

- Panel Customizable
- Output Text
- Go Link
- Go Image Link
- Show Detail Frame
- Rich Text Editor
- Inline Frame
Page Layout

The biggest challenge in page template design is how to lay out components:

• elements of the template
• page content

There are two basic strategies:

• Flow layout:
  – Components have fixed sizes and are arranged side by side.
  – If necessary, the browser displays scroll bar(s).

• Stretch layout:
  – Components will be stretched to occupy the available space on the page.
  – If necessary, individual components may have scroll bars.
Page Template Layout: Vertical Behavior

• Stretching:
  – Header & footer always visible
  – Height of the page is determined by the browser window
  – Content stretched vertically
  – Content might have scroll bar

• Flowing:
  – Content never stretched vertically
  – Height of the page is calculated based on the components
  – Browser might display scroll bar
  – Header and/or footer not necessarily visible
Page Template Layout: Horizontal Behavior

- **Stretching**
  - Content stretched horizontally
  - Content might have scroll bar
  - Other components have fixed width

- **Flowing**
  - Content width is fixed
  - Browser might show scroll bar
  - Some components might be stretched to fill up existing space
Page Layout Usage

- Majority of web sites use flow-type layout.
- “Application” type web sites prefer stretch-type layout.
- As a template developer, you can control whether the content facet is in a stretching or flowing region of the page.
- Page content must be created taking the layout strategy into consideration.
- Since a page template can be changed dynamically, create pages and design custom components that display properly in stretching and flowing context.
- We recommend using ADF Faces containers to create the layout.
Page Template Layout
ADF UI Components (1)

Stretching layouts:

• Build outer structure with containers that can be stretched and stretch its children.
  – PanelStretchLayout, PanelSplitter, ...

• Create flowing islands.
  – Use PanelGroupLayout with type="scroll" or type="vertical"
    this can be stretched, but will not stretch its children

• You should not embed stretching components inside flowing islands.

• Never try to stretch something vertically when inside of a flowing container. Do not use height with percent unit.
Page Template Layout
ADF UI Components (2)

Flowing layouts:

• Use non-stretching containers
  – PanelGroupLayout, PanelBorderLayout, ...
  – PanelBorderLayout can be used to approximate HTML table component

• To avoid multiple scroll bars, do not nest scrolling PanelGroupLayout components. Consider type="vertical"

• Most stretchable ADF components also work in flowing context with dimensionsFrom="auto"

• To stretch a component horizontally, use styleClass="AFStretchWidth" instead of inlineStyle="width:100.0%"
Page Template Layout
ADF UI Components (3)

Customizable components:

• In PanelCustomizable, use layout="auto" to detect whether to stretch its children or not.
• In order to support flow and stretch layouts, use ShowDetailFrame with stretchChildren="auto"
Creating Pages Using Page Templates: Design Time

- Use New Gallery > JSF > JSF Page
  
  ![New Gallery screenshot](image1)

- Create pages under /oracle/webcenter/portalapp/pages
- Choose page template
- Find available templates in `pagetemplate-metadata.xml`
Creating Pages Using Page Templates:
Page Source

•  *main.jspx*

```xml
<af:form id="f1">
    <af:pageTemplate id="pt1"
        viewId="/oracle/webcenter/portalapp/pagetemplates/CorporateTemplate.jspx"
        value="${bindings.pageTemplateBinding}"
    
    <f:attribute name="contentWidth" value="1000px"/>
    <f:facet name="content"/>
    </af:pageTemplate>
</af:form>
```

•  *mainPageDef.xml*

```xml
<executables>
    <variableIterator id="variables"/>
    <page path="oracle.webcenter.portalapp.pagetemplates.CorporateTemplatePageDef"
        id="pageTemplateBinding" Refresh="ifNeeded"/>
</executables>
```
Creating Pages Using Page Templates: Run Time

- Use Administration > Resources > Pages > Create Page

- Choose page template
- Choose page style
Agenda

• Working with Page Templates
• Page Templates for WebCenter Portal Framework Applications
• Page Templates for WebCenter Portal: Spaces
Default Page Template

- There is a preference bean that stores default values for the application UI.
  - Page Template
  - Navigation Model, Resource Catalog, Skin
  - others
- To use the default template, change the source of the page.
  - main.jspx
    <af:pageTemplate id="pt1"
      value="#{bindings.pageTemplateBinding.templateModel}"/>
  - mainPageDef.xml
    <page path="${preferenceBean.defaultPageTemplate}"
      id="pageTemplateBinding" Refresh="ifNeeded"/>
Default Page Template:
Setting the Default Template

• Design time: edit `adf-config.xml`

```xml
<portal:preference
    id="oracle.webcenter.portalapp.pagetemplate.pageTemplate"
    desc="Default Page Template"
    value="/oracle/webcenter/portalapp/pagetemplates/CorporateTemplate.jspx"
    resourceType="Template" display="true"/>
```

• Run time: Administration > Configuration
Out-of-the-Box Templates

A WebCenter Portal application has two OOTB page templates:

- **Globe**: `pageTemplate_globe.jspx`
- **Swooshy**: `pageTemplate_swooshy.jspx`
## Out-of-the-Box Templates: Template Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>contentWidth</td>
<td>String</td>
<td>960px</td>
</tr>
<tr>
<td>showNavigation</td>
<td>Boolean</td>
<td>true</td>
</tr>
<tr>
<td>showGreetings</td>
<td>Boolean</td>
<td>#{securityContext.authenticated}</td>
</tr>
<tr>
<td>showLogin</td>
<td>Boolean</td>
<td>true</td>
</tr>
<tr>
<td>showAdmin</td>
<td>Boolean</td>
<td>#{securityContext.authenticated}</td>
</tr>
<tr>
<td>isAdminPage</td>
<td>Boolean</td>
<td>false</td>
</tr>
<tr>
<td>isEditingTemplate</td>
<td>Boolean</td>
<td>false</td>
</tr>
</tbody>
</table>
Out-of-the-Box Templates: Template Attributes (continued)

- In OOTB login.jspx, error.jspx
  showNavigation = false, showLogin = false
- In built-in administration pages
  isAdminPage = true
- When editing the template, Resource Manager sets
  isEditingTemplate = true

Use these attributes in your custom application template if the template will be used as default template or you intend to use the OOTB login page.
Hiding the Template from Page Editing

Hide the template content, **except the facet reference**, when editing the page.

- For example, use the rendered attribute:

  rendered = "#{!composerContext.inEditMode or attrs.isEditingTemplate}"

- Essential only when the template has editable components
Login/Logout

• Use ADF security – redirect to login page
  <af:goLink id="pt_glnk3" text="Login"
    destination="/adfAuthentication?success_url=/"
    rendered="#{!securityContext.authenticated}"/>
  <af:goLink id="pt_glnk4" text="Logout"
    destination="/adfAuthentication?logout=true&end_url=/"
    rendered="#{securityContext.authenticated}"/>

• Use utility Java Bean: o_w_s_l_LoginBackingBean
  – Set attributes: username, password
  – Invoke methods: doLogin, doLogout
  <af:commandLink id="pt_logincb" text="Login"
    action="#{o_w_s_l_LoginBackingBean.doLogin}"
    rendered="#{!securityContext.authenticated}"/>
  <af:commandLink id="pt_glnk4" text="Logout"
    action="#{o_w_s_l_LoginBackingBean.doLogout}"
    rendered="#{securityContext.authenticated}"/>
Agenda

• Working with Page Templates
• Page Templates for WebCenter Portal Framework Applications
• Page Templates for Oracle WebCenter Portal: Spaces
Run-Time Resource Management

Global templates:

Space templates:

- Global templates inherited in a space
- Template created in JDeveloper uploads as global template
Out-of-the-Box Templates
PS5 (11.1.1.6.0)

In PS5 release, new OOTB page templates will be introduced:

*Flowing layout:*
- Top Navigation
- Side Navigation
- Collaborative with Top Navigation
- Portal-Centric with Top Navigation

*Stretching Layout:*
- Top Navigation (Stretch)
- Side Navigation (Stretch)
- Fusion Side Navigation
- Fusion Top Navigation
- Public Pages Template
Creating a Page Using Page Templates

• Pages are created only at run time.
• Pages always use a template defined by Spaces.
  – Defined at application level: Administration > Configuration.
  – Defined in each Space
  – Defined in the navigation
Template Content: Spaces Components

Examples
- Login/Logout
- User profile/User preferences
- Edit Page/Create Page/Manage Pages
- Home space
- Spaces Switcher
- Contact Administrator
- Copyright Message, …
Spaces Components: Design Time (1)

- Add JSP Library to project:
  
  `<JDev_HOME>/jdeveloper/webcenter/modules/oracle.webcenter.framework_11.1.1/spaces-components.jar`
Spaces Components: Design Time (2)

- Drop from Spaces Declarative Components

Result in source:
- Added namespace definition
  
  ```
  xmlns:wcdc="http://xmlns.oracle.com/webcenter/spaces/taglib"
  ```
- Added tags
  ```
  <wcdc:spacesAction id="..." type="login" text="Login"/>
  <wcdc:spacesAction id="..." type="logout" text="Logout"/>
  <wcdc:editPage ... />
  ```
Spaces Components: Run Time

- Add from Resource Catalog > Template Development
Template Content: Miscellaneous Tips

- No need to hide the template when editing the page
- Surround template with special tags:
  
  `<wcddc:siteTemplateMetadata type="start"/>
  
  `<wcddc:siteTemplateMetadata type="end"/>

- If you use `wcddc:spacesAction`, add the following:
  
  `<af:resource type="javascript">
  
  function wcNavigate(event) {
    window.location =
    event.getSource().getProperty('wcDestination');
    event.cancel(); }
  
  function wcLaunchWindow(event) {
    window.open(event.getSource().
    getProperty('wcDestination'));
    event.cancel(); }
  
  </af:resource>`
Summary

In this module you should have learned how to:

- Create and use page templates both at design- and run time
- Create an editable page template
- Create a portal resource from an existing page template
- Explain the round-trip development process for page templates
- Describe the differences between page templates for an Oracle WebCenter Portal Framework application and Oracle WebCenter Portal: Spaces