Preface ............................................................................................................................................................ xix
   Intended Audience ............................................................................................................................. xix
   Documentation Accessibility ............................................................................................................. xx
   Related Documents ........................................................................................................................... xx
   Conventions ........................................................................................................................................... xxi
   Browser Recommendations ............................................................................................................... xxii

Part I       Portlet Overview

1  Understanding Portlets
   1.1 Introduction to Portal Development ........................................................................................ 1-1
   1.2 Understanding Portlets ............................................................................................................. 1-1
   1.3 Portlet Anatomy ....................................................................................................................... 1-3
   1.4 Portlet Resources ..................................................................................................................... 1-4
      1.4.1 Out-of-the-Box Portlets ....................................................................................................... 1-4
      1.4.2 Other Sources of Pre-Built Portlets .................................................................................... 1-5
      1.4.3 Web Clipping .................................................................................................................... 1-6
      1.4.4 OmniPortlet ....................................................................................................................... 1-9
      1.4.5 Portlet Builder .................................................................................................................... 1-10
      1.4.6 Programmatic Portlets ....................................................................................................... 1-10
      1.4.7 Deciding Which Tool to Use ............................................................................................ 1-11
   1.5 Summary .................................................................................................................................. 1-12

2  Portlet Technologies Matrix
   2.1 The Portlet Technologies Matrix ............................................................................................ 2-1
   2.2 General Suitability ................................................................................................................... 2-4
      2.2.1 Web Clipping ..................................................................................................................... 2-4
      2.2.2 OmniPortlet ....................................................................................................................... 2-4
      2.2.3 Java Portlets ....................................................................................................................... 2-5
      2.2.4 Portlet Builder .................................................................................................................... 2-5
      2.2.5 PL/SQL Portlets ................................................................................................................... 2-5
   2.3 Expertise Required .................................................................................................................... 2-6
      2.3.1 Web Clipping ..................................................................................................................... 2-6
      2.3.2 OmniPortlet ....................................................................................................................... 2-6
      2.3.3 Java Portlets ....................................................................................................................... 2-6
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.4</td>
<td>Portlet Builder</td>
<td>2-7</td>
</tr>
<tr>
<td>2.3.5</td>
<td>PL/SQL Portlets</td>
<td>2-7</td>
</tr>
<tr>
<td>2.4</td>
<td>Deployment Type</td>
<td>2-7</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Web Providers</td>
<td>2-7</td>
</tr>
<tr>
<td>2.4.2</td>
<td>WSRP Producers</td>
<td>2-8</td>
</tr>
<tr>
<td>2.4.3</td>
<td>Database Providers</td>
<td>2-9</td>
</tr>
<tr>
<td>2.4.4</td>
<td>Provider Architecture</td>
<td>2-10</td>
</tr>
<tr>
<td>2.4.5</td>
<td>Provider Registration</td>
<td>2-11</td>
</tr>
<tr>
<td>2.5</td>
<td>Caching Style</td>
<td>2-12</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Web Clipping, OmniPortlet, and Portlet Builder</td>
<td>2-13</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Java Portlets</td>
<td>2-13</td>
</tr>
<tr>
<td>2.5.3</td>
<td>PL/SQL Portlets</td>
<td>2-13</td>
</tr>
<tr>
<td>2.6</td>
<td>Development Tool</td>
<td>2-13</td>
</tr>
<tr>
<td>2.6.1</td>
<td>Web Clipping, OmniPortlet, and Portlet Builder</td>
<td>2-13</td>
</tr>
<tr>
<td>2.6.2</td>
<td>Java Portlets</td>
<td>2-13</td>
</tr>
<tr>
<td>2.6.3</td>
<td>PL/SQL Portlets</td>
<td>2-13</td>
</tr>
<tr>
<td>2.7</td>
<td>Portlet Creation Style</td>
<td>2-14</td>
</tr>
<tr>
<td>2.7.1</td>
<td>OmniPortlet and Web Clipping</td>
<td>2-15</td>
</tr>
<tr>
<td>2.7.2</td>
<td>Java Portlets</td>
<td>2-15</td>
</tr>
<tr>
<td>2.7.3</td>
<td>Portlet Builder</td>
<td>2-15</td>
</tr>
<tr>
<td>2.7.4</td>
<td>PL/SQL Portlets</td>
<td>2-15</td>
</tr>
<tr>
<td>2.8</td>
<td>User Interface Flexibility</td>
<td>2-15</td>
</tr>
<tr>
<td>2.8.1</td>
<td>Web Clipping</td>
<td>2-15</td>
</tr>
<tr>
<td>2.8.2</td>
<td>OmniPortlet</td>
<td>2-15</td>
</tr>
<tr>
<td>2.8.3</td>
<td>Java Portlets and PL/SQL Portlets</td>
<td>2-15</td>
</tr>
<tr>
<td>2.8.4</td>
<td>Portlet Builder</td>
<td>2-16</td>
</tr>
<tr>
<td>2.9</td>
<td>Ability to Capture Content from Web Sites</td>
<td>2-16</td>
</tr>
<tr>
<td>2.9.1</td>
<td>Web Clipping</td>
<td>2-16</td>
</tr>
<tr>
<td>2.9.2</td>
<td>OmniPortlet</td>
<td>2-16</td>
</tr>
<tr>
<td>2.9.3</td>
<td>Java Portlets</td>
<td>2-16</td>
</tr>
<tr>
<td>2.9.4</td>
<td>PL/SQL Portlets</td>
<td>2-16</td>
</tr>
<tr>
<td>2.10</td>
<td>Ability to Render Content Inline</td>
<td>2-16</td>
</tr>
<tr>
<td>2.10.1</td>
<td>Web Clipping</td>
<td>2-16</td>
</tr>
<tr>
<td>2.10.2</td>
<td>OmniPortlet</td>
<td>2-17</td>
</tr>
<tr>
<td>2.10.3</td>
<td>Java Portlets</td>
<td>2-17</td>
</tr>
<tr>
<td>2.10.4</td>
<td>Portlet Builder</td>
<td>2-17</td>
</tr>
<tr>
<td>2.10.5</td>
<td>PL/SQL Portlets</td>
<td>2-17</td>
</tr>
<tr>
<td>2.11</td>
<td>Charting Capability</td>
<td>2-18</td>
</tr>
<tr>
<td>2.11.1</td>
<td>Web Clipping</td>
<td>2-18</td>
</tr>
<tr>
<td>2.11.2</td>
<td>OmniPortlet</td>
<td>2-18</td>
</tr>
<tr>
<td>2.11.3</td>
<td>Java Portlets</td>
<td>2-18</td>
</tr>
<tr>
<td>2.11.4</td>
<td>Portlet Builder</td>
<td>2-18</td>
</tr>
<tr>
<td>2.11.5</td>
<td>PL/SQL Portlets</td>
<td>2-18</td>
</tr>
<tr>
<td>2.12</td>
<td>Public Portlet Parameters Support</td>
<td>2-18</td>
</tr>
<tr>
<td>2.13</td>
<td>Private Portlet Parameter Support</td>
<td>2-19</td>
</tr>
<tr>
<td>2.13.1</td>
<td>OmniPortlet, Web Clipping, and Portlet Builder</td>
<td>2-19</td>
</tr>
<tr>
<td>2.13.2</td>
<td>Java Portlets and PL/SQL Portlets</td>
<td>2-19</td>
</tr>
</tbody>
</table>
3 Creating Portlets with OmniPortlet

3.1 What is OmniPortlet? ................................................................. 3-1
3.2 The OmniPortlet Wizard ............................................................ 3-2
3.2.1 Type ..................................................................................... 3-3
3.2.2 Source .................................................................................. 3-4
3.2.2.1 Proxy Authentication .............................................................. 3-4
3.2.2.2 Connection Information ......................................................... 3-5
3.2.2.3 Spreadsheet ........................................................................ 3-6
3.2.2.4 SQL ...................................................................................... 3-6
3.2.2.4.1 SQL Connection Information .............................................. 3-7
3.2.2.4.2 Using Stored Procedures ..................................................... 3-8
3.2.2.5 XML ................................................................................... 3-9
3.2.2.6 Web Service ......................................................................... 3-10
3.2.2.7 Web Page ............................................................................ 3-11
3.2.3 Filter .................................................................................... 3-11
3.2.4 View .................................................................................... 3-12
3.2.5 Layout .................................................................................. 3-13
3.2.5.1 Tabular Layout .................................................................... 3-13
3.2.5.2 Chart Layout ....................................................................... 3-14
3.2.5.3 News Layout ....................................................................... 3-17
3.2.5.4 Bullet Layout ...................................................................... 3-18
6.1.1.1 Shared Screen Mode (View Mode for JPS) .......................................................... 6-2
6.1.1.1.1 HTML Guidelines for Rendering Portlets .................................................. 6-3
6.1.1.1.2 Cascading Style Sheet Guidelines for Rendering Portlets ....................... 6-4
6.1.1.2 Edit Mode (JPS and OracleAS Portal) ............................................................. 6-4
6.1.1.2.1 Guidelines for Edit Mode Operations ...................................................... 6-4
6.1.1.2.2 Guidelines for Buttons in Edit Mode ...................................................... 6-5
6.1.1.2.3 Guidelines for Rendering Personalization Values .................................... 6-5
6.1.1.3 Edit Defaults Mode (JPS and OracleAS Portal) ............................................. 6-5
6.1.1.3.1 Guidelines for Edit Defaults Mode Options ............................................ 6-5
6.1.1.3.2 Guidelines for Buttons in Edit Defaults Mode ........................................ 6-6
6.1.1.3.3 Guidelines for Rendering Personalization Values .................................... 6-6
6.1.1.4 Preview Mode (JPS and OracleAS Portal) .................................................... 6-6
6.1.1.5 Full Screen Mode (OracleAS Portal) ............................................................. 6-7
6.1.1.6 Help Mode (JPS and OracleAS Portal) ......................................................... 6-7
6.1.1.7 About Mode (JPS and OracleAS Portal) ...................................................... 6-7
6.1.1.8 Link Mode (OracleAS Portal) ....................................................................... 6-8
6.1.2 Guidelines for Navigation within a Portlet ....................................................... 6-8
6.1.2.1 Types of Links for Portlets ........................................................................... 6-8
6.1.2.1.1 Intraportlet Links .................................................................................... 6-9
6.1.2.1.2 Portal Links .......................................................................................... 6-9
6.1.2.1.3 External Links ......................................................................................... 6-9
6.1.2.1.4 Internal/Resource Links .......................................................................... 6-9
6.1.3 Guidelines for Mobile Portlets ......................................................................... 6-10
6.1.3.1 Declare Capabilities .................................................................................... 6-11
6.1.3.2 Declare a Short Title .................................................................................. 6-11
6.1.3.3 Implement Personalization of the Short Title ............................................ 6-11
6.1.3.4 Implement Link Mode ................................................................................ 6-11
6.1.3.5 Heed Device Information .......................................................................... 6-12
6.1.3.6 Tailor Personalization Pages ...................................................................... 6-12
6.2 Introduction to Java Portlet Specification (JPS) and WSRP .................................. 6-12
6.3 Configuring Your Application Server to Run JPS-Compliant Portlets .................. 6-14
6.3.1 Configuring OC4J in Oracle Application Server ............................................ 6-14
6.3.1.1 Configuring with the Portal and Wireless Option ..................................... 6-14
6.3.1.2 Configuring without the Portal and Wireless Option ............................... 6-18
6.3.2 Configuring OC4J Standalone ...................................................................... 6-22
6.4 Building JPS-Compliant Portlets with Oracle JDeveloper .................................. 6-24
6.4.1 Installing the Portal Extension for Oracle JDeveloper .................................... 6-24
6.4.2 Building JPS-compliant Portlets .................................................................... 6-24
6.4.2.1 Creating a Portlet ....................................................................................... 6-25
6.4.2.2 Adding Portlet Logic ................................................................................ 6-33
6.4.2.3 Deploying Your Portlet to an Application Server ..................................... 6-33
6.4.2.3.1 Creating a Connection to Your Application Server ......................... 6-33
6.4.2.3.2 Deploying the WAR File ................................................................. 6-35
6.4.2.4 Registering and Viewing Your Portlet ....................................................... 6-36
6.5 Introduction to PDK-Java .................................................................................... 6-41
6.6 Building PDK-Java Portlets with Oracle JDeveloper ........................................ 6-42
6.6.1 Installing the Portal Extension for Oracle JDeveloper .................................... 6-42
7 Enhancing Java Portlets

7.1 Enhancing JPS Portlets ................................................................. 7-1
  7.1.1 Adding Personalization .......................................................... 7-2
    7.1.1.1 Assumptions ................................................................. 7-2
    7.1.1.2 Implementing Personalization ...................................... 7-2
  7.2 Enhancing PDK-Java Portlets .................................................. 7-4
    7.2.1 Adding Show Modes .......................................................... 7-5
      7.2.1.1 Assumptions ................................................................. 7-5
      7.2.1.2 Implementing Extra Show Modes .............................. 7-5
      7.2.1.3 Updating the XML Provider Definition .................... 7-6
      7.2.1.4 Viewing the Portlet ................................................... 7-6
    7.2.2 Adding Personalization ..................................................... 7-8
      7.2.2.1 Assumptions ................................................................. 7-9
      7.2.2.2 Implementing Personalization for Edit and Edit Defaults Pages 7-10
        7.2.2.2.1 Reviewing the Generated Code ............................ 7-10
        7.2.2.2.2 Modifying the Generated Code ......................... 7-11
      7.2.2.3 Implementing Personalization for Show Pages ............ 7-12
      7.2.2.4 Preference Information Within the XML Provider Definition 7-12
      7.2.2.5 Viewing the Portlet ................................................... 7-13
      7.2.3 Passing Parameters and Submitting Events .................... 7-13
        7.2.3.1 Assumptions ................................................................. 7-13
        7.2.3.2 Adding Parameters to Your Portlets ....................... 7-14
        7.2.3.3 Passing Portlet URL Parameters ............................ 7-16
          7.2.3.3.1 Portlet URL Types ............................................ 7-16
          7.2.3.3.2 URL Parameters .............................................. 7-16
          7.2.3.3.3 Building Links with the Portlet URL Types .......... 7-18
          7.2.3.3.4 Building Forms with the Portlet URL Types ........ 7-19
          7.2.3.3.5 Implementing Navigation within a Portlet .......... 7-21
        7.2.3.4 Submitting Events .................................................. 7-24
      7.2.4 Using JNDI Variables ...................................................... 7-28
        7.2.4.1 Declaring JNDI Variables ........................................ 7-28
          7.2.4.1.1 Variable Types ............................................... 7-28
          7.2.4.1.2 Variable Naming Conventions .......................... 7-28
          7.2.4.1.3 Examples ......................................................... 7-29
        7.2.4.2 Setting JNDI Variable Values ................................ 7-29
          7.2.4.2.1 Setting Values in Oracle Enterprise Manager 10g .. 7-29
          7.2.4.2.2 Setting Values Manually ................................ 7-30
        7.2.4.3 Retrieving JNDI Variables ....................................... 7-30
8 Creating PL/SQL Portlets

8.1 Guidelines for Creating PL/SQL Portlets ................................................................. 8-2
8.1.1 Portlet Show Modes ............................................................................................ 8-2
8.1.2 Recommended Portlet Procedures and Functions ............................................. 8-3
8.1.3 Guidelines for Mobile Portlets ........................................................................... 8-4
8.2 Building PL/SQL Portlets with the PL/SQL Generator ......................................... 8-4
8.2.1 Creating the Input XML File .............................................................................. 8-4
8.2.2 Running the PL/SQL Generator ....................................................................... 8-7
8.2.3 Publishing the Generated PL/SQL Portlet ......................................................... 8-8
8.2.3.1 Installing the Packages in the Database ..................................................... 8-8
8.2.3.2 Registering the Database Provider ........................................................... 8-9
8.2.3.3 Adding Your Portlet to a Page .................................................................... 8-9
8.3 Building PL/SQL Portlets Manually ...................................................................... 8-9
8.3.1 Implementing the Portlet Package ................................................................. 8-10
8.3.2 Implementing the Provider Package ............................................................ 8-11
8.3.3 Adding Your Portlet to a Page ................................................................. 8-15
8.4 Implementing Information Storage ...................................................................... 8-15
8.4.1 Implementing a Preference Store .................................................................... 8-15
8.4.1.1 Using a Preference Store ................................................................. 8-16
8.4.1.2 Creating and Accessing a Preference Store ........................................... 8-16
8.4.2 Implementing a Session Store ....................................................................... 8-20
Part III  Content Management APIs

9  Content Management API Introduction

9.1  Overview ................................................................................................................. 9-1
9.2  Content Management APIs .................................................................................... 9-3
9.2.1  Secure Content Repository Views ........................................................................ 9-3
9.2.2  Terminology ......................................................................................................... 9-3
9.3  Providing Access to the APIs and Secure Views ....................................................... 9-4
9.4  Guidelines for Using the APIs .................................................................................. 9-5
9.4.1  Using a Separate Schema ..................................................................................... 9-5
10 Getting Started with Content Management APIs

10.1 Setting the Session Context ................................................................. 10-1
10.2 API Parameters ..................................................................................... 10-1
10.3 Finding an Object ID ........................................................................... 10-2
10.3.1 Finding a Page Group ID ................................................................. 10-2
10.3.2 Finding a Page ID ............................................................................ 10-3
10.3.3 Finding Region IDs ........................................................................ 10-4
10.3.4 Finding an Item ID ........................................................................ 10-4

11 Performing Simple Content Management Tasks

11.1 Editing Page Properties ................................................................. 11-1
11.1.1 Creating the WWSBR_USER_PAGES View ................................... 11-6
11.2 Editing Content ............................................................................... 11-6
11.2.1 Setting Item Attributes ................................................................. 11-6
11.2.2 Editing an Item ............................................................................. 11-11
11.2.3 Checking Items Out and In ......................................................... 11-12
11.2.4 Using Version Control ................................................................. 11-13
11.3 Reorganizing Content ................................................................... 11-13
11.3.1 Moving an Item to a Different Page ............................................. 11-14
11.3.2 Moving Pages .............................................................................. 11-14
11.3.3 Moving Categories and Perspectives ........................................... 11-15

11.4 Copying Content .......................................................................... 11-15
11.4.1 Copying Items ............................................................................ 11-16
11.4.2 Copying Pages ............................................................................ 11-16

11.5 Deleting Content ............................................................................ 11-17
11.5.1 Deleting Items ........................................................................... 11-17
11.5.2 Deleting Pages ........................................................................... 11-18

12 Extending Your Portal

12.1 Creating a Page Group ................................................................. 12-1
12.2 Creating Pages .............................................................................. 12-2
12.3 Creating Categories and Perspectives ........................................... 12-3
12.4 Creating Items .............................................................................. 12-4
12.5 Setting Perspectives Attributes of Pages and Items ....................... 12-8
12.6 Approving and Rejecting Items ................................................................. 12-9

### 13 Searching Portal Content

13.1 Searching For Items Across All Page Groups ............................................. 13-2
13.2 Searching For Pages in Specific Page Groups .............................................. 13-3
13.3 Searching For Items By Attribute .............................................................. 13-4
13.4 Transforming Search Results into XML ...................................................... 13-5
13.4.1 Creating a Directory for the XML File .................................................... 13-5
13.4.2 Creating an XML File from a CLOB ...................................................... 13-6
13.4.3 Generating Search Results in XML ....................................................... 13-7
13.4.4 Workaround for get_item_xml ............................................................. 13-7
13.5 Displaying Search Results ......................................................................... 13-9
13.5.1 Displaying XML Search Results in OmniPortlet ................................. 13-9
13.5.2 Displaying Search Results in a Dynamic Page ....................................... 13-10

### 14 Creating Multi-Lingual Content

14.1 Introduction to Multi-Lingual Support ....................................................... 14-1
14.2 Querying the Default Language ............................................................... 14-1
14.3 Setting the Session Language ................................................................. 14-2
14.4 Modifying an Existing Translation ........................................................... 14-2
14.5 Creating a Translation for an Item ........................................................... 14-3
14.6 Translations and Item Versioning ............................................................ 14-4

### 15 Implementing Content Security

15.1 Retrieving Object Privileges ................................................................. 15-1
15.2 Setting Page Level Privileges ................................................................. 15-3
15.2.1 Granting Page Level Privileges ............................................................ 15-3
15.2.2 Removing Page Level Privileges ........................................................... 15-4
15.3 Setting Item Level Privileges ................................................................. 15-5
15.3.1 Granting Item Level Privileges ............................................................ 15-6
15.3.2 Removing Item Level Privileges ........................................................... 15-7
15.3.3 Inheriting Item Level Privileges from the Page .................................... 15-8

### 16 Using the Content Management Event Framework

16.1 What Is the Content Management Event Framework? ............................ 16-1
16.2 How Does the Content Management Event Framework Work? ............... 16-1
16.2.1 Enqueuing Messages ............................................................................ 16-2
16.2.2 Subscribers and Dequeuing Messages ................................................. 16-3
16.2.2.1 Adding a Subscriber to the WWSBR_EVENT_Q Queue ..................... 16-3
16.2.2.2 Subscriber Queue Management ....................................................... 16-4
16.2.2.3 Dequeuing Messages ........................................................................ 16-4
16.2.3 Exception Handling ............................................................................. 16-6
16.2.4 Listening for Messages ........................................................................ 16-6
16.3 Using the Content Management Event Framework ................................... 16-8
16.3.1 Creating Subscriber Code .................................................................... 16-8
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.3.2 Adding a Subscriber to WWSBR_EVENT_Q</td>
<td>16-9</td>
</tr>
<tr>
<td>16.3.3 Enabling CMEF Events at the Page Group Level</td>
<td>16-9</td>
</tr>
<tr>
<td>16.3.4 Examining CMEF Events</td>
<td>16-11</td>
</tr>
<tr>
<td>16.3.5 Running a CMEF Subscriber</td>
<td>16-11</td>
</tr>
<tr>
<td>16.3.6 CMEF Message Payload</td>
<td>16-11</td>
</tr>
<tr>
<td>16.3.7 OracleAS Portal Actions and CMEF Events</td>
<td>16-13</td>
</tr>
<tr>
<td>16.3.7.1 Page and Page Group Actions</td>
<td>16-13</td>
</tr>
<tr>
<td>16.3.7.1.1 Creating a Page</td>
<td>16-13</td>
</tr>
<tr>
<td>16.3.7.1.2 Updating the Access Control List of a Page</td>
<td>16-14</td>
</tr>
<tr>
<td>16.3.7.1.3 Updating the Access Control List of a Page Group</td>
<td>16-14</td>
</tr>
<tr>
<td>16.3.7.1.4 Deleting a Page</td>
<td>16-15</td>
</tr>
<tr>
<td>16.3.7.2 Item Actions</td>
<td>16-15</td>
</tr>
<tr>
<td>16.3.7.2.1 Creating an Item and Publishing it at the Same Time</td>
<td>16-15</td>
</tr>
<tr>
<td>16.3.7.2.2 Adding an Item That Requires Approval</td>
<td>16-16</td>
</tr>
<tr>
<td>16.3.7.2.3 Approving an Item</td>
<td>16-16</td>
</tr>
<tr>
<td>16.3.7.2.4 Applying a Category or Perspective to an Item</td>
<td>16-17</td>
</tr>
<tr>
<td>16.3.7.2.5 Deleting an Item</td>
<td>16-18</td>
</tr>
<tr>
<td>16.4 Installing the Examples</td>
<td>16-18</td>
</tr>
<tr>
<td>16.5 Example: Portal Object Event Logging</td>
<td>16-19</td>
</tr>
<tr>
<td>16.6 Example: Item Notification</td>
<td>16-22</td>
</tr>
<tr>
<td>16.7 Example: Item Validation</td>
<td>16-27</td>
</tr>
<tr>
<td>16.8 Example: Integrating External Workflow</td>
<td>16-30</td>
</tr>
<tr>
<td>16.8.1 Integrating Workflow with OracleAS Portal</td>
<td>16-30</td>
</tr>
<tr>
<td>16.8.2 Example Overview</td>
<td>16-31</td>
</tr>
<tr>
<td>16.8.3 Detailed Example Description</td>
<td>16-32</td>
</tr>
<tr>
<td>16.8.3.1 Enable Approvals and Notifications in OracleAS Portal</td>
<td>16-32</td>
</tr>
<tr>
<td>16.8.3.2 Grant Users the Manage Items With Approval Privileges</td>
<td>16-33</td>
</tr>
<tr>
<td>16.8.3.3 Run Scripts Required for the CMEF Workflow Integration Example</td>
<td>16-34</td>
</tr>
<tr>
<td>16.8.3.4 Create Subscriber and Check Procedures</td>
<td>16-34</td>
</tr>
<tr>
<td>16.8.3.5 Register the WF_CHECKURL Process with Oracle Workflow</td>
<td>16-38</td>
</tr>
<tr>
<td>16.8.3.6 Add the CMEF_WORKFLOW Subscriber to the WWSBR_EVENT_Q Queue</td>
<td>16-39</td>
</tr>
</tbody>
</table>

**Part IV Appendixes**

**A Creating Portlets with the Portlet Builder**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1 Using a Wizard to Build a Portlet</td>
<td>A-1</td>
</tr>
<tr>
<td>A.1.1 Creating a Schema in OracleAS Portal</td>
<td>A-2</td>
</tr>
<tr>
<td>A.1.1.1 Creating a Schema</td>
<td>A-2</td>
</tr>
<tr>
<td>A.1.1.2 Granting and Revoking Privileges on Database Objects</td>
<td>A-4</td>
</tr>
<tr>
<td>A.1.1.3 Enrolling the Schema in One or More Roles</td>
<td>A-6</td>
</tr>
<tr>
<td>A.1.2 Creating a Provider for Locally Built Portlets</td>
<td>A-7</td>
</tr>
<tr>
<td>A.1.3 Exposing a Provider</td>
<td>A-8</td>
</tr>
<tr>
<td>A.1.4 Creating Portlets Using OracleAS Portal Wizards</td>
<td>A-10</td>
</tr>
<tr>
<td>A.1.4.1 Building Portlets Declaratively</td>
<td>A-12</td>
</tr>
<tr>
<td>A.1.4.2 Building Forms Declaratively</td>
<td>A-14</td>
</tr>
<tr>
<td>A.1.4.3 Building Reports Declaratively</td>
<td>A-27</td>
</tr>
</tbody>
</table>
B Troubleshooting Portlets and Providers

B.1 Diagnosing General Portlet Problems................................................................. B-1
  B.1.1 Portlet Refresh Failure.................................................................................. B-1
  B.1.2 HTML Tags Appearing in Portlet ............................................................... B-2
B.2 Diagnosing Java Portlet Problems...................................................................... B-2
  B.2.1 Portlet Logging............................................................................................ B-3
  B.2.2 Installation and Deployment Problems...................................................... B-3
    B.2.2.1 Cannot Find a Java Class Object............................................................ B-3
    B.2.2.2 Cannot Deploy the template.ear File.................................................... B-4
    B.2.2.3 Error When Attempting to Register Provider.......................................... B-4
    B.2.2.4 Error Adding a Portlet to a Provider....................................................... B-7
    B.2.2.5 Portlet Does Not Exist........................................................................ B-8
    B.2.2.6 File Not Found.................................................................................... B-8
    B.2.2.7 XML Parser Error............................................................................... B-8
    B.2.2.8 Error Adding Portlets........................................................................ B-9
    B.2.2.9 Content Request Timed Out ................................................................. B-9
    B.2.2.10 Message 500 Returned....................................................................... B-10
    B.2.2.11 JPS Portlets with the get Method not Working..................................... B-10
    B.2.2.12 Portlet Displays Session Expired Message After Redeployment............ B-11
  B.2.3 Java Portlet Wizard Not Available............................................................... B-11
  B.2.4 Portlet Code Does Not Compile ................................................................. B-11
  B.2.5 Application Server Connection Test Fails................................................ B-12
  B.2.6 Provider Test Page Shows Error............................................................... B-12
  B.2.7 Web Provider Not Appearing in Portlet Repository................................... B-12
  B.2.8 Portlet Does Not Display on Page............................................................. B-13
  B.2.9 After Initial Successful Display, Portlet Does Not Display on Page........ B-13
  B.2.10 Provider Group Not Created................................................................... B-14
  B.2.11 URL-Based Portlet Does Not Work.......................................................... B-14
B.3 Diagnosing OmniPortlet Problems................................................................. B-15
  B.3.1 OmniPortlet Cannot Access the Specified URL........................................ B-16
  B.3.2 Portlet Content Is Not Refreshed............................................................... B-17
  B.3.3 Edit Defaults Changes are Not Reflected in the Personalized Portlet........ B-17
B.4 Diagnosing Web Clipping Problems............................................................... B-18
  B.4.1 Setting Logging Levels............................................................................... B-18
  B.4.2 Reviewing Error Messages ..................................................................... B-18
  B.4.3 Checking the Status of the Provider with the Test Page............................. B-18
  B.4.4 Problem Connecting to the Web Site for Clipping .................................... B-19
  B.4.5 HTTP Error Code 407 When Clipping Outside Firewall......................... B-20
## G  Content Management Event Framework Events

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.1  Actions and Events for Items</td>
<td>G-1</td>
</tr>
<tr>
<td>G.2  Actions and Events for Pages</td>
<td>G-5</td>
</tr>
<tr>
<td>G.3  Actions and Events for Tabs</td>
<td>G-7</td>
</tr>
<tr>
<td>G.4  Actions and Events for Page Groups</td>
<td>G-8</td>
</tr>
<tr>
<td>G.5  Actions and Events for Attributes</td>
<td>G-9</td>
</tr>
<tr>
<td>G.6  Actions and Events for Item Types</td>
<td>G-9</td>
</tr>
<tr>
<td>G.7  Actions and Events for Page Types</td>
<td>G-10</td>
</tr>
<tr>
<td>G.8  Actions and Events for Categories</td>
<td>G-11</td>
</tr>
<tr>
<td>G.9  Actions and Events for Perspectives</td>
<td>G-11</td>
</tr>
<tr>
<td>G.10 Actions and Events for Templates</td>
<td>G-12</td>
</tr>
</tbody>
</table>

## Glossary

## Index
Preface

This manual describes how to build portlets for Oracle Application Server Portal (OracleAS Portal) using a variety of tools and technologies. This manual includes information that helps you understand the various technology choices open to you, choose the technology that best meets your requirements, and use the appropriate tools to build and deploy your portlets.

Intended Audience

This manual is intended primarily for portal developers, but page designers may also find it useful. This manual guides you through the process of first understanding and choosing a portlet technology, and then building your portlets with that technology.

What Is a Portal Developer? A portal developer is a user who writes code to help make a portal meet the specific requirements of an organization. For example, a portal developer may build portlets and make them available to page designers and other users for inclusion on their pages. This type of portal developer is also referred to as a portlet developer. A portal developer may also use the public APIs provided with OracleAS Portal to perform certain portal tasks programmatically, rather than through the product’s user interface. A portal developer will generally, although not always, be someone with at least some programming knowledge. The privileges assigned to a portal developer depend on the type of tasks that developer performs.

Note: Examples of how to use APIs to perform content management tasks will be provided soon on Portal Center (http://portalcenter.oracle.com) and will be incorporated into a later release of this guide.

What Is a Portlet Developer? A portlet developer is a user with the following global privileges: Create All Portal DB Providers and Manage All Shared Components. Since OracleAS Portal offers such a wide spectrum of tools and technologies for building portlets, a portlet developer may or may not have substantial programming background.

What Is a Page Designer? A page designer, also known as a page manager, is a user with the Manage privilege on a page. A user with this privilege can perform any action on the page and can create sub-pages under the page. The page designer is often responsible for designing the layout (or region configuration) of the page and assigning privileges on the page to other users (for example, to determine who can add content to the page).
The scope of a page designer’s control over a page may be limited if the page is based on a template.

For information about the different privileges in OracleAS Portal and how these affect the tasks you can perform, see the Oracle Application Server Portal User’s Guide on the OracleAS Portal Documentation page on OTN (http://www.oracle.com/technology/products/ias/portal/documentation.html).

---

**Note:** For the portable document format (PDF) version of this manual, when a URL breaks onto two lines, the full URL data is not sent to the browser when you click it. To get to the correct target of any URL included in the PDF, copy and paste the URL into your browser’s address field. In the HTML version of this manual, you can click a link to directly display its target in your browser.

---

**Documentation Accessibility**

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

http://www.oracle.com/accessibility/

**Accessibility of Code Examples in Documentation**

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

**Accessibility of Links to External Web Sites in Documentation**

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

**TTY Access to Oracle Support Services**

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

**Related Documents**

For more information, see the following manuals in the OracleAS Portal documentation set:

- Oracle Application Server Portal Release Notes
- Oracle Application Server Portal User’s Guide
Oracle Application Server Portal Configuration Guide
Oracle Application Server Portal Error Messages Guide

Note: You can find all documentation related to OracleAS Portal on the OracleAS Portal Documentation page on OTN (http://www.oracle.com/technology/products/ias/portal/documentation.html)

You may also find the following manuals in the Oracle Application Server documentation set useful:

Oracle Application Server Concepts
Oracle HTTP Server Administrator’s Guide
Oracle Application Server Web Cache Administrator’s Guide
Oracle Application Server Web Toolkit Reference

Note: You can find documentation related to Oracle Application Server on OTN (http://www.oracle.com/technology/documentation/index.html).

Conventions

The following text conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface</strong></td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td><em>italic</em></td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td><strong>monospace</strong></td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Angle brackets enclose user-supplied names.</td>
</tr>
<tr>
<td>[]</td>
<td>Brackets enclose optional clauses from which you can choose one or none.</td>
</tr>
<tr>
<td>...</td>
<td>Ellipsis points in an example mean that information not directly related to the example has been omitted.</td>
</tr>
<tr>
<td>ORACLE_HOME</td>
<td>Represents the full path of the Oracle home, and is used where it is easy to determine which Oracle home is referenced.</td>
</tr>
<tr>
<td>MID_TIER_ ORACLE_HOME</td>
<td>Represents the full path of the middle-tier Oracle home, and is used where it is necessary to distinguish between the middle tier, Oracle Application Server Infrastructure, or Oracle Application Server Metadata Repository.</td>
</tr>
<tr>
<td>INFRA_ORACLE_ HOME</td>
<td>Represents the full path of the Oracle Application Server Infrastructure Oracle home, and is used where it is necessary to distinguish between the middle tier, Oracle Application Server Infrastructure, or Oracle Application Server Metadata Repository.</td>
</tr>
</tbody>
</table>
Browser Recommendations

Use one of the following Web browsers with OracleAS Portal:

- Microsoft Internet Explorer 6.0
- Netscape 7.2
- Mozilla 1.7
- Firefox 1.0
- Safari 1.2

You may encounter JavaScript errors if you use a browser older than the recommended minimum.

Cache Settings

Your browser's default settings are sufficient for displaying valid portal content. If those defaults have been altered, you should check that their values do not inadvertently disable valuable cache capabilities. To check your browser's cache settings, perform the following steps:

**In Internet Explorer:**
1. From the Tools menu, choose **Internet Options**.
2. Click the **General** tab to bring it forward.
3. In the Temporary Internet File section, click the **Settings** button.
4. In the **Check for newer versions of stored pages** radio group, select **Automatically**.

   This is the default option. You can also select **Every visit to the page** or **Every time you start Internet Explorer**.

**Note:** Do not select **Never**. Selecting Never means the browser will never check for newer versions of stored pages, and the cache will never be updated.

5. Click **OK**.
6. Click **OK**.

**In Netscape/Mozilla:**
1. From the Edit menu, choose **Preferences**.
2. Expand the **Advanced** node.
3. Click **Cache**.
4. In the Document in cache is compared to document on network radio group, select When page is out of date.

This is the default setting. You can also select Every time I view the page or Once per session.

---

**Note:** Do not select Never. Selecting Never means the browser will never check for newer versions of stored pages, and the cache will never be updated.

---

5. Click OK.

**Image Settings**

Sometimes the browser setting that controls the automatic loading of images is disabled to increase performance on low bandwidth connections. A problem that commonly occurs when images are not loaded automatically is that, once logged out, you cannot log in again without closing and re-invoking the browser. To avoid this problem, make sure images are loaded automatically.

To ensure that images are loaded automatically:

**In Internet Explorer:**
1. From the Tools menu, choose Internet Options.
2. Click the Advanced tab to bring it forward.
3. Scroll through the list of options to the Multimedia node, and select Show Pictures.
4. Click OK.

**In Netscape:**
1. From the Edit menu, choose Preferences.
2. Click Advanced.
3. Select the Automatically load images check box.
4. Click OK.
Part I

Portlet Overview

Part I contains the following chapters:

■ Chapter 1, "Understanding Portlets"
■ Chapter 2, "Portlet Technologies Matrix"
This chapter introduces you to portlets, the anatomy of a portlet, and provides an overview of the various portlet-building tools available in Oracle Application Server Portal:

- Introduction to Portal Development
- Understanding Portlets
- Portlet Anatomy
- Portlet Resources

1.1 Introduction to Portal Development

OracleAS Portal enables you to present information from multiple, unrelated data sources in one, organized view. This view, a portal page, can contain one or more components—called portlets—that can each get their content from different data sources.

OracleAS Portal has all the tools you need for developing portlets and adding them to your portal pages. OracleAS Portal’s tools support a wide range of development skills: from the novice business developer to the experienced IT programmer. You can develop portlets either declaratively, through the OracleAS Portal user interface, or programmatically, through OracleAS Portal’s collection of application programming interfaces (APIs), known as the Oracle Application Server Portal Developer Kit (PDK). Additionally, you can develop portlets through other development tools, external to OracleAS Portal, and integrate them through the PDK and an OracleAS Portal entity called a provider. To learn more about providers, refer to Chapter 2, "Portlet Technologies Matrix".

This chapter defines portlets, lists and describes some sources for pre-built portlets and resources for building portlets, and suggests the best resource for the job.

1.2 Understanding Portlets

A portlet is a reusable, pluggable Web component that can draw content from many different sources. A typical portlet is one that displays summaries of Web content as shown in Figure 1–1.
Figure 1–1  Portlets on the My Oracle Home Page

For example, in your portal you may have a news feed portlet that supplies linked news article headlines that are each accompanied by a sentence describing the content of the article (Figure 1–2).

Figure 1–2  The Oracle News Portlet on the My Oracle Home Page

Users click the linked headlines to get to the full text of the article, which is hosted on an external news service (Figure 1–3). The portlet has a somewhat dynamic nature in that headlines change automatically as news stories are added and removed at the source.
Portlets provide a seamless, single view of data that originates from multiple sources. Since different portlets can be placed on a common page, the user receives a single-source experience. In reality, the content is derived from multiple sources.

Portlets display excerpts of other Web sites, generate summaries of key information, perform searches, and access assembled collections of information from a variety of data sources.

1.3 Portlet Anatomy

A portlet on a page is rendered in an HTML table cell. A portlet can display various types of content, such as HTML, formatted text, images, or elements of an HTML form.

Figure 1–4 shows the anatomy of a portlet, which includes a header that contains the portlet title. You can create a hyperlink in the portlet title, so that when a user clicks the title, the portlet displays in a full browser page. A portlet can also include a border, to distinguish the layout from other portlets on the page.

Portlets typically contain a Personalize link. The portlet developer can expose this link to the page designer, who can then turn these on or off. Clicking the Personalize link displays a number of options where the end user can personalize various attributes of the portlet.

Each portlet also contains a refresh icon, which the end user can click to refresh the content of the portlet (without refreshing the entire portal page), as well as the standard collapse icon, which the end user can click to collapse or expand the portlet on the page.
1.4 Portlet Resources

Portlet resources include the many pre-built portlets available out of the box from many sources, including OracleAS Portal, Oracle E-Business Suite, and third-party sources. Portlet resources also include portlet-building tools available through the OracleAS Portal user interface as well as from the PDK and other Oracle tools. Each of these tools offers different product features that are targeted toward different developer roles.

This section describes different portlet resources, suggests the level of expertise required to use them, and provides examples of when they might best be used. It includes the following subsections:

- **Out-of-the-Box Portlets**
- **Other Sources of Pre-Built Portlets**
- **Web Clipping**
- **OmniPortlet**
- **Portlet Builder**
- **Programmatic Portlets**

This section introduces you to the various portlet resources. For specific information on each tool and its benefits, refer to Chapter 2, "Portlet Technologies Matrix".

### 1.4.1 Out-of-the-Box Portlets

**What Are They?**

Out-of-the-box portlets are fully developed, registered portlets that are immediately available from the Portlet Repository when you install OracleAS Portal (Figure 1–5). They include such portlets as Search, Saved Searches, Favorites, and My Notifications.

Who Is the Intended User?
Out-of-the-box portlets are best suited for use by end users and page designers, though they are available to users at all levels of expertise.

When Should They Be Used?
Use out-of-the-box portlets when your needs are satisfied by the functions the portlets offer, and the level of personalization readily available is sufficient to complete the desired task.

Consider alternatives when you need to extend or personalize the portlet, such as when you need a different user interface, or when the functionality you require is not available out of the box.

For more information on when you should use each technology, refer to Chapter 2, "Portlet Technologies Matrix".

1.4.2 Other Sources of Pre-Built Portlets

What Are They?
Other sources of pre-built portlets include partner portlets, integration solutions, and the Portal Knowledge Exchange.

Partner portlets are available through Oracle’s partnerships with different types of leading system integrators, software vendors, and content providers. You can access these portlets through the Portal Catalog, available on Portal Center (http://portalcenter.oracle.com). The following are some examples of these portlets:

- Generating point-to-point driving directions
Portal Center also provides integration solutions that are useful for customers who require basic functionality for popular applications such as Microsoft Exchange, Lotus Notes, SAP, IMAP, SMTP, and the like.

The Portal Knowledge Exchange, also accessible on Portal Center, is an offering from Portal Developer Services. Community members exchange portal expertise that includes portlet samples, tips, white papers, sample code, and so on. Members receive a personal folder on Portal Center, which they can use to upload portlet code and portal development insights and download and rate contributions from other developers.

Who Is the Intended User?
Fully developed, downloadable portlets are best suited for use by end users and page designers who understand how to download, install, and register Web and database providers in OracleAS Portal. They are available for use by all levels of experience.

When Should They Be Used?
As with out-of-the-box portlets, use pre-built portlets from other sources when your needs are satisfied by the functions the portlets offer, and the level of personalization readily available is sufficient to complete the desired task.

Consider alternatives when you need to extend or personalize the portlet, for example, when you need a different user interface or when the functionality you require is not available out of the box.

1.4.3 Web Clipping

What Is It?
Web Clipping is a browser-based declarative tool that enables you to integrate any Web application with OracleAS Portal. It is designed to give you quick integration by leveraging the Web application’s existing user interface. Web Clipping has been implemented as a Web provider using the PDK-Java, which is a component of OracleAS Portal.

To create a Web Clipping portlet, the portal page designer uses a Web browser to navigate to the Web page that contains the desired content. Through the Web Clipping Studio, the page designer can drill down through a visual rendering of the target page to choose the desired the content (Figure 1–6 and Figure 1–7).
Web Clipping supports the following:

- **Navigation through various styles of login mechanisms**, including form- and JavaScript-based submission and HTTP Basic and Digest Authentication with cookie-based session management.

- **Fuzzy matching of clippings**. If a Web clipping gets reordered within the source page or if its character font, size, or style changes, it will still be identified correctly by the Web Clipping engine and delivered as the portlet content.

- **Reuse of a wide range of Web content**, including basic support of pages written with HTML 4.0.1, JavaScript, applets, and plug-in enabled content, retrieved through HTTP GET and POST (form submission).

- **Personalization**, allowing a page designer to expose input parameters that page viewers can modify when they personalize the portlet. These parameters can be exposed as public parameters that a page designer can map as OracleAS Portal page parameters. This feature enables end users to obtain personalized clippings.

- **Integrated authenticated Web content through Single Sign-On**, including integration with external applications, which enables you to leverage Oracle
Application Server Single Sign-On and to clip content from authenticated external Web sites.

- **Inline rendering**, enabling you to set up Web Clipping portlets to display links within the context of the portlet. As a result, when a user clicks a link in the Web Clipping portlet, the results display within the same portlet. You can use this feature with internal and external Web sites.

- **Proxy Authentication**, including support for global proxy authentication and per-user authentication. You can specify the realm of the proxy server and whether all users will automatically log in using a user name and password you provide, each user will log in using an individual user name and password, or all users will log in using a specified user name and password.

- **Migration from URL-based portlets**, enabling you to migrate your URL-based portlets to Web Clipping.

**Who Is the Intended User?**

Web Clipping is best suited for use by page designers and portlet developers who want to leverage an existing Web page for rapid portlet development. The Web Clipping portlet is accessible out of the box, and is available in the Portlet Repository of OracleAS Portal. This portlet can be added to a page by any user with the appropriate privileges.

**When Should It Be Used?**

Use Web Clipping when you want to copy content and functionality from an existing Web page and expose it in your portal as a portlet. Consider alternatives if you want to change the way information is presented in the clipped portlet. That is, you don't need to control the UI or application flow, and you are accessing Web-based applications. For a greater level of control, use the OracleAS Portal OmniPortlet's Web page data source instead of Web Clipping.

The following are some examples of when you may consider using the Web Clipping portlet:

- **Stock chart portlet.** Suppose you want to create a portlet that displays the stock market's daily performance chart from your financial advisor's Web site. You could clip this information from an external Web site, even if your company is using a proxy.

- **Personalized weather portlet.** Suppose you want to create a portlet that displays weather information from a major Internet weather site, and you want your users to be able to personalize the portlet by providing the desired zip code.

- **Web mail portlet.** Suppose your users want to access their confidential Web mail accounts through a portlet and their inboxes to display in the portlet.

For more information on using Web Clipping, refer to Chapter 5, “Creating Content-Based Portlets with Web Clipping”.

---

**Note:** To use OmniPortlet, the Simple Parameter Form, or the Web Clipping portlet, you must use Netscape 7.0 or higher, or Microsoft Internet Explorer 5.5 or higher for Windows 2000.
1.4.4 OmniPortlet

What Is It?
The OracleAS Portal OmniPortlet is a declarative portlet-building tool that enables you to build portlets against a variety of data sources, including XML files, character-separated value files (for example, spreadsheets), Web Services, databases, Web pages, and SAP data sources. OmniPortlet users can also choose a pre-built layout for the data. Pre-built layouts include tabular, news, bullet, form, chart, and HTML (Figure 1–8).

Figure 1–8 An OmniPortlet Using Tabular Format

<table>
<thead>
<tr>
<th>employee_id</th>
<th>Name</th>
<th>Gender</th>
<th>Job</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>Kathleen Bayyat</td>
<td>F</td>
<td>President - Manufacturing</td>
<td><a href="mailto:kbayyat@oracle.com">kbayyat@oracle.com</a></td>
</tr>
<tr>
<td>39</td>
<td>Robert Rodriguez</td>
<td>M</td>
<td>President - Sales</td>
<td><a href="mailto:mrodriguez@oracle.com">mrodriguez@oracle.com</a></td>
</tr>
<tr>
<td>49</td>
<td>Edward Shields</td>
<td>M</td>
<td>Chief Financial Officer</td>
<td><a href="mailto:eshields@oracle.com">eshields@oracle.com</a></td>
</tr>
<tr>
<td>110</td>
<td>Jan Francesco</td>
<td>M</td>
<td>Graphic Artist</td>
<td><a href="mailto:janfranco@oracle.com">janfranco@oracle.com</a></td>
</tr>
<tr>
<td>100</td>
<td>Lisa Williams</td>
<td>F</td>
<td>Graphic Artist</td>
<td><a href="mailto:lwilliams@oracle.com">lwilliams@oracle.com</a></td>
</tr>
<tr>
<td>500</td>
<td>Sandie Kiley</td>
<td>F</td>
<td>Course Developer</td>
<td><a href="mailto:skiley@oracle.com">skiley@oracle.com</a></td>
</tr>
<tr>
<td>773</td>
<td>Eddy You</td>
<td>F</td>
<td>Customer Sales Representative</td>
<td><a href="mailto:eddy@oracle.com">eddy@oracle.com</a></td>
</tr>
</tbody>
</table>

Note: The SAP data source is not included with OracleAS Portal. To learn more about using the SAP data source, visit the Oracle Portal Integration Solutions page on OTN (http://www.oracle.com/technology/products/ias/portal/point.html).

Like Web Clipping, OmniPortlet supports proxy authentication, including support for global proxy authentication and per-user authentication. You can specify whether all users will automatically log in using a user name and password you provide, each user will log in using an individual user name and password, or all users will log in using a specified user name and password.

You’ll find information about OmniPortlet on Portal Center (http://portalcenter.oracle.com), then click Portlet Development.

Who Is the Intended User?
OmniPortlet is best suited for use by page designers and developers.

When Should It Be Used?
Use OmniPortlet when you want to build portlets rapidly against a variety of data sources with a variety of layouts. Consider alternatives when you want complete control of the design and functionality of the portlet.

The following are some examples of when you may consider using OmniPortlet:

- RSS news feed portlet. Suppose you want to create a portlet that displays live, scrolling news information to your users. The data comes from a Really Simple Syndication news feed, such as the Oracle Technology Network Headlines. You also want the portlet to contain hyperlinks to the news source.

- Sales chart portlet. Suppose you want to present up-to-date information on your company’s sales results. You also want to display data in the form of a pie chart, and your company stores its sales information in a remote relational database.
SAP portlet. Suppose you want to display information from a company’s SAP system. To minimize the load on the company’s SAP Business Suite, the information retrieved from the system must be cached for each user for the entire day.

For more information on OmniPortlet, refer to Chapter 3, "Creating Portlets with OmniPortlet" and Chapter 4, "Building Example Portlets with OmniPortlet"

1.4.5 Portlet Builder

What Is It?
OracleAS Portal includes a number of portlet-building wizards that are accessible through the Provider tab in the Portal Navigator. These wizards can be used to build charts, reports, forms, calendars, and lists of values (Figure 1–9).

Figure 1–9 Sample Chart from the Portlet Builder

![Sample Chart from the Portlet Builder](image)

When Should It Be Used?
It is recommended that you use OmniPortlet as an alternative to Portlet Builder whenever possible. OmniPortlet provides more flexibility and a separation of data and layout which enables you to change from a report to chart without re-creating the entire portlet (as is required with Portlet Builder). OmniPortlet also provides more options for deployment to many different portals simultaneously. OracleAS Portal will continue to support Portlet Builder as a portlet building option. However, new features and enhancements will be directed toward the OmniPortlet tool.

Note: You can find information on using Portlet Builder in Appendix A “Creating Portlets with the Portlet Builder” of the Oracle Application Server Portal Developer’s Guide Release 2 (10.1.4) in the Oracle Application Server Release 2 (10.1.2.0.2) library located on the Oracle Technology Network (http://www.oracle.com/technology/documentation/appserver1012.html).

1.4.6 Programmatic Portlets

What Are They?
The OracleAS PDK contains a set of portlet-building APIs that you can use to create programmatic portlets.
Understanding Portlets

You’ll find more information about these APIs on Portal Center (http://portalcenter.oracle.com). Also, the PDK-PL/SQL is not described in detail in this manual. For specific information on the PDK-PL/SQL, refer to the Developer Services area on Portal Center (http://portalcenter.oracle.com).

Who Is the Intended User?

These tools are best used by experienced and knowledgeable IT developers.

When Should They Be Used?

Use the PDK when you have very specialized business rules or logic or when you require personalized authentication, granular processing of dynamic results, and complete user interface control. Additionally, use the PDK in the following scenarios, that is, when:

- You are building a portlet from the start and need complete control over all of its functionality.
- You know Java or PL/SQL.
- You are comfortable with the PDK and the configuration of OracleAS Portal Providers.

Consider using this approach when the out-of-the-box declarative tools do not address your needs.

The following are some examples of when you may consider using Java portlets created with the Oracle Application Server Portal Developer Kit:

- Discussion forum portlet. Suppose you want to create a portlet that integrates your company’s JSP-based discussion forum application with OracleAS Portal. The discussion forum posts are stored in a relational database. The portlet must also follow the strict look and feel of your company’s Internet Web site.

- Email portlet. Suppose you want to create a portlet that enables users to send email from the company’s intranet portal. You must integrate the email portlet with the company’s LDAP server so that the users can use the address book on the LDAP server.

For more information on using the PDK-Java and PDK-PL/SQL, refer to Chapter 6, "Creating Java Portlets", Chapter 7, "Enhancing Java Portlets", and Chapter 8, "Creating PL/SQL Portlets".

1.4.7 Deciding Which Tool to Use

Figure 1–10 illustrates the spectrum of portlet resources described in the previous section. Notice how one end of the spectrum is geared toward a page designer while the other end speaks to the portlet developer. You can choose your tool depending on which type of user most closely approximates your expertise.

For more information on deciding which tool to use, refer to Chapter 2, "Portlet Technologies Matrix".
1.5 Summary

In this chapter, you learned what portlets are, as well as the various technologies available in OracleAS Portal. For more information on these tools and technologies, refer to Chapter 2, "Portlet Technologies Matrix".
This chapter describes portlet features, characteristics, technologies, and tools to help you decide which portlet building technology best suits your needs. It includes the following sections:

- The Portlet Technologies Matrix
- General Suitability
- Expertise Required
- Deployment Type
- Caching Style
- Development Tool
- Portlet Creation Style
- User Interface Flexibility
- Ability to Capture Content from Web Sites
- Ability to Render Content Inline
- Charting Capability
- Public Portlet Parameters Support
- Private Portlet Parameter Support
- Event Support
- Ability to Hide and Show Portlets Based on User Privileges
- Multilingual Support
- Pagination Support
- Single Sign-On and External Application Integration

2.1 The Portlet Technologies Matrix

Table 2–1, "Portlet Building Technologies Comparison Matrix" summarizes the technologies and tools you can use with OracleAS Portal on one axis, and the features and characteristics on the other. The matrix describes those tools and technologies that are covered in further detail in this guide: OmniPortlet, Web Clipping, the Java Portlets (PDK-Java) including Standards, Portlet Builder as an appendix, and PL/SQL Portlets (PDK-PL/SQL) (in the matrix only).
The Portlet Technologies Matrix

Note: While these are the primary tools for building portlets, additional tools and technologies exist, such as other Oracle products, including Oracle Reports (http://www.oracle.com/technology/products/reports/index.html) and Oracle Discoverer (http://www.oracle.com/technology/products/discoverer/index.html). These other tools are not covered in this guide.

The other sections in this chapter provide further detail on the characteristics listed in Table 2–1. Use the table to quickly scan all the features and characteristics, then refer to the subsequent sections for more in-depth information. You can also find more information on Portal Center (http://portalcenter.oracle.com).

<table>
<thead>
<tr>
<th>Table 2–1</th>
<th>Portlet Building Technologies Comparison Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Clipping</td>
<td>OmniPortlet</td>
</tr>
<tr>
<td>General Suitability</td>
<td>A simple wizard-based tool that helps you retrieve and present Web content, originating from other Web sites, in your portal.</td>
</tr>
<tr>
<td>Expertise Required</td>
<td>No expertise required.</td>
</tr>
<tr>
<td>Supported Data Sources (for details, see Section 2.3, “Expertise Required”)</td>
<td>Any Web site accessible on the network over HTTP or HTTPS.</td>
</tr>
<tr>
<td>Deployment Type</td>
<td>Web provider</td>
</tr>
<tr>
<td>Development Tool</td>
<td></td>
</tr>
</tbody>
</table>
### Table 2–1 (Cont.) Portlet Building Technologies Comparison Matrix

<table>
<thead>
<tr>
<th>Web Clipping</th>
<th>OmniPortlet</th>
<th>PDK-Java</th>
<th>Standards</th>
<th>Portlet Builder</th>
<th>PDK-PL/SQL</th>
</tr>
</thead>
</table>

#### Portlet Creation Style

| Portlet Creation Style | Develop in place. | Develop in place. | No in-place portlet building experience. Add portlet to page, edit defaults, and personalize. | No in-place portlet building experience. Add portlet to page, edit defaults, and personalize. | Develop first, then add portlet to page and develop in place. | No in-place portlet building experience. Add portlet to page, edit defaults, and personalize. |

#### User Interface Flexibility


#### Ability to Capture Content from Web Sites

| Ability to Capture Content from Web Sites | Yes, by its nature. | Yes, by using the Web Page Data Source. | Yes, by using the java.net package. | Yes, by using the java.net package. | No | Yes, by using the UTIL_HTTP package. |

#### Ability to Render Content Inline

| Ability to Render Content Inline | Yes | No URL rewriting support, but can be achieved by using public portlet parameters and events. | By using private portlet parameters. | Include servlets and JSPs (using the PortletContext.getRequestDispatcher() method). | Pagination in reports and charts is rendered inline. | By using private portlet parameters. |

#### Charting Capability


#### Public Portlet Parameters Support

| Public Portlet Parameters Support | Yes | Yes | Yes | No | Yes | Yes |

#### Private Portlet Parameter Support

| Private Portlet Parameter Support | N/A | N/A | Yes | Yes | No | Yes |

#### Event Support

| Event Support | Yes | Yes | Yes | Portlet private events (actions). | No | No |

#### Ability to Hide and Show Portlets Based on User Privileges
2.2 General Suitability

This section describes each portlet-building technology in terms of its usage characteristics (for example, wizard-based or programmatic).

2.2.1 Web Clipping

Web Clipping is a simple wizard-based tool that helps you retrieve and present Web content, originating from other Web sites, in your portal. Web Clipping does not require you to have any technical background.

Examples of portlets you can build using Web Clipping

- Stock chart portlet
- Personalized weather portlet
- Web mail portlet

2.2.2 OmniPortlet

If you are looking for an easy-to-use, wizard-based tool to present information from a wide variety of data sources, you should consider OmniPortlet. OmniPortlet runs completely in the browser. All you need to do is drop your OmniPortlet on a portal page, and select your data source from a list of available data sources, which includes the following:

- Spreadsheet
- SQL
Although OmniPortlet does not require you to use an additional development tool or have a strong technical background, you can build reusable and high-performing portlets with it.

**Examples of portlets you can create with OmniPortlet**
- RSS news feed portlet
- Sales chart portlet
- SAP Business Suite portlet

### 2.2.3 Java Portlets

If the wizard-based portlet building tools do not satisfy your needs, you can build your portlets programmatically using Java. The Java Community Process standardized the Java portlet APIs in 2003. Portlets built against the Java Specification Request (JSR) 168 standard are interoperable across different portal platforms. The Java Portlet Wizard, an Oracle JDeveloper plug-in, helps you get started with your Java portlets.

---

**Note:** When building portlets in Java, you have full control over your portlet's functionality. For example, you can control what it looks like and how it behaves.

---

**Examples of portlets you can build using Java**
- Discussion forum portlet
- Email portlet

### 2.2.4 Portlet Builder

Portlet Builder is a wizard-based tool to create data-driven portlets, where the data resides in an Oracle database. You can build interactive forms to insert, update, and delete database records. You can create flexible reports and HTML bar charts to display information from the database. Portlet Builder also enables you to pass parameters and navigate between your data-driven portlets by using dynamic links.

**Examples of portlets you can build using the Portlet Builder**
- Data entry portlet
- Dynamic list of partners portlet
- Sales results portlet

### 2.2.5 PL/SQL Portlets

Similar to Java portlets, PL/SQL portlets provide a flexible approach to build Web applications that cannot be satisfied by built-in portlets. For example, your application may require implementation of special business rules or logic or meet custom-designed authorization requirements. PL/SQL portlets are commonly used when you need to perform data intensive operations by using SQL and PL/SQL. OracleAS Portal offers a rich set of PL/SQL APIs, such as programmatic provider
registration, object level privilege management, user interface control, or multilingual support.

For example, any information provider can create custom portlets to display an application to users through OracleAS Portal. Developers simply build their portlets according to OracleAS Portal Developer Kit (PDK) specifications and register the provider with OracleAS Portal. Developers can use the OracleAS PDK to develop portlets to suit their needs.

**Examples of portlets you can build using PL/SQL**
- Content upload portlet
- Site map portlet
- Sophisticated data entry and report portlet

### 2.3 Expertise Required

While some of the portlet building tools do not require portlet development skills, others assume a strong technical background. This section describes each tool in terms of the level of knowledge required to use it effectively.

#### 2.3.1 Web Clipping

Web Clipping is a tool that does not require any technical background at all. However, if you want to parameterize the Web page content that you clipped, you need to have an understanding of public portlet parameters and page parameters.

#### 2.3.2 OmniPortlet

OmniPortlet requires you to have basic knowledge of the data source you want to leverage in your portlet.

<table>
<thead>
<tr>
<th>Data source</th>
<th>What you need to know about the data source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spreadsheet</td>
<td>The URL that points to the spreadsheet containing the data that you want to display in the portlet.</td>
</tr>
<tr>
<td>SQL</td>
<td>The connection information to the data source and the SQL query that retrieves the data from the database.</td>
</tr>
<tr>
<td>XML</td>
<td>The location of the XML source and optionally the address of the XSL filter and the XML schema.</td>
</tr>
<tr>
<td>Web service</td>
<td>The WSDL URL, the method of the Web service, and optionally the XSL filter URL and the XML schema URL.</td>
</tr>
<tr>
<td>Web page</td>
<td>The Web page data source uses the same environment as Web Clipping. No technical background is required.</td>
</tr>
<tr>
<td>J2EE Connector Architecture</td>
<td>Although not displayed on the Type page of the OmniPortlet Wizard, a J2Eetm Connector Architecture (JCA) 1.0 adapter is also available. JCA provides a mechanism to store and retrieve enterprise data such as that held in ERP systems (Oracle Financials, SAP, PeopleSoft, and so on).</td>
</tr>
</tbody>
</table>

#### 2.3.3 Java Portlets

To build Java portlets, you must know at least a subset of J2EE. Knowing HTML, Java servlets, and XML is a must, and JSP experience is recommended. Additional Java
knowledge is optional, depending on the task you want to perform. Using Java portlets you can access any data source (supported by the Java language).

### 2.3.4 Portlet Builder

If you want to use Portlet Builder, you must have a good understanding of relational database concepts. Depending on what you want to achieve, SQL and PL/SQL knowledge may be required, as well. Using Portlet Builder, you can consume data from the local (Oracle Application Server infrastructure) database or remote databases using database links.

### 2.3.5 PL/SQL Portlets

To build PL/SQL portlets, you must know how to write SQL statements, code and debug PL/SQL program units using SQL*Plus or similar development tool that enables you to connect to Oracle database. You should also know HTML and PL/SQL Web Toolkit to generate the portlet content. Experience of coding the PL/SQL Server Pages (PSP) is optional.

### 2.4 Deployment Type

As shown in Figure 2–1, portlets can be deployed to OracleAS Portal through three provider types: Web providers, WSRP producers, and database providers. Web providers are deployed to a J2EE application server, which is often remote and communicates with OracleAS Portal through Simple Object Access Protocol (SOAP) over HTTP. Web Services for Remote Portlets (WSRP) is an OASIS standard that can be used to communicate between portlets and OracleAS Portal. Database providers are implemented in PL/SQL and deployed in the Oracle database where OracleAS Portal is installed.

![Portlet Provider Overview](image)

**Figure 2–1 Portlet Provider Overview**

**2.4.1 Web Providers**

Web providers are the most commonly used and flexible type of provider. They may reside on the same application server as OracleAS Portal, on a remote application...
server, or anywhere on the network. A Web provider could be implemented using virtually any Web technology. However, the Oracle Application Server Portal Developer Kit provides a Java framework that simplifies the task of building Web providers.

Web providers use open standards, such as XML, SOAP, HTTP, or J2EE for deployment, definition, and communication with OracleAS Portal (Figure 2–2). Also, because Web providers can be deployed to a J2EE container, they do not put an additional load on the OracleAS Portal Repository database.

**Figure 2–2 Web Providers**

You can do the following when developing portlets and exposing them as Web providers:

- Deploy portlets remotely.
- Leverage existing Web application code to create portlets.
- Specify providers declaratively.
- Take advantage of more functionality than that with database providers.
- Use standard Java technologies (for example, servlets and JSPs) to develop portlets of Web providers.

To expose your portlets using a Web provider, you must create a provider that manages your portlets and can communicate with OracleAS Portal using SOAP. To learn how to expose your portlets using a Web provider, refer to Section 6.4, "Building JPS-Compliant Portlets with Oracle JDeveloper".

### 2.4.2 WSRP Producers

Web Services for Remote Portlets (WSRP). Web services standard that enables the plug-and-play of visual, user-facing Web services with portals or other intermediary Web applications. Being a standard, WSRP enables interoperability between a standards-enabled container based on a particular language (such as JSR 168, .NET, Perl) and any WSRP portal. So, a portlet (regardless of language) deployed to a WSRP-enabled container can be rendered on any portal that supports this standard.
From an architecture perspective, WSRP producers are very similar to Web providers. For more information on the WSRP portal architecture, see The Relationship Between WSRP and JPS.

To expose your portlets as a WSRP producer, you must create a producer that manages your portlets. To learn more about WSRP, see the WSRP and JSR 168 Standards page on the Oracle Technology Network. To learn how to expose your portlets as a WSRP producer, see Section 6.4.2.3, "Deploying Your Portlet to an Application Server". You can also test your WSRP producers online using the OracleAS Portal Verification Service (http://portalstandards.oracle.com).

### 2.4.3 Database Providers

You can also create a database provider that owns one or more PL/SQL portlets (Figure 2–3). Database providers and their PL/SQL portlets reside in the Oracle Application Server Metadata Repository database and are implemented as PL/SQL packages. To access database providers on remote servers, you can use the Federated Portal Adapter. For more information, see the "Understanding the Federated Portal Adapter" article on Portal Center (http://www.oracle.com/technology/products/ias/portal/portlet_development_10gr2.html).

**Figure 2–3  Database Providers**

Database providers are ideal when you need to perform data-intensive operations using PL/SQL. An example of this is when you are building forms or charts with the OracleAS Portal user interface or the PL/SQL APIs provided in the PDK.

To learn how to expose your PL/SQL portlets using a database provider, refer to Section 8.13, "Registering Providers Programmatically".
2.4.4 Provider Architecture

Figure 2–4 illustrates the basic architecture of portlet providers.

Figure 2–4 Provider Architecture

When users display the portal page in their Web browsers, the flow of the request works like this:

1. The user requests a portal page from the Web browser by entering a URL in the browser’s address field.
2. The Parallel Page Engine (PPE), which resides in the Oracle Application Server’s middle tier, retrieves the portal page layout, portlet, and provider information (also called the page metadata) from the OracleAS Portal Repository.
3. The PPE contacts all the providers for the portlet content.
4. The providers make the necessary calls to their portlets so that the portlets generate the portlet content in the form of HTML or XML code.
5. The providers return the portlet content back to the PPE.
6. The PPE assembles the portal page, and the Oracle Application Server returns the page to the Web browser.

Note: The PPE is responsible for constructing the requested portal page based on the page metadata.

Note: For more information about the portlet and provider architecture, visit the Portlet Development page on Portal Center (http://portalcenter.oracle.com).
Web Clipping, OmniPortlet, and Java portlets communicate with OracleAS Portal through Web providers. After you install OracleAS Portal, Web Clipping and OmniPortlet are ready to use; their providers are registered with OracleAS Portal out of the box. You have to register the provider of your Java portlets explicitly.

**Note:** Web Clipping and OmniPortlet are developing very rapidly. The most recent versions of these portlets are available for download on OTN. If you decide to go with the downloaded version of these tools, you must deploy them to OC4J and register them with OracleAS Portal as Web providers. For more information, refer to the Oracle Application Server Portal Configuration Guide available on the Documentation on OTN (http://www.oracle.com/technology/products/ias/portal/documentation.html).

Data-driven portlets, built with Portlet Builder, communicate with OracleAS Portal through database providers. You do not need to register the Portlet Builder providers with OracleAS Portal explicitly; they are automatically registered for you.

PL/SQL portlets communicate with OracleAS Portal through a database provider. You have to register the database provider explicitly.

### 2.4.5 Provider Registration

OracleAS Portal includes a provider registration wizard, accessible from the Providers tab in the Navigator. The registration screen contains the following sections:

- **Provider Information**: Contains the provider name, timeout details, and the implementation style.
- **User/Session Information**: Contains information on how session information is communicated to the providers.
- **Database Providers**: Contains information specific to database providers, such as the implementation owner and name.
- **Web Providers**: Contains information specific to Web providers, such as the URL of the provider, the user’s identity communicated to the provider, and proxy information.
- **WSRP Producers**: Contains information specific to WSRP producers, such as the WSDL URL and the session handling information supplied by the producer.

The sections that impact session handling are the User/Session Information section and the cookie domain check box on the Web provider registration page of the wizard. For more information on using the same cookie domain, refer to the "Sharing Session Cookies Not Allowed in PDK-Java Release 2" article, which can be accessed from the Portlet Development page on Portal Center (http://www.oracle.com/technology/products/ias/portal/portlet_development_10gr2.html).

**User/Session Information**

In the User/Session Information section, you can choose one of the following two options, depending on the session-related information you want the providers to receive from OracleAS Portal:

- **Public**: Choosing this option sets the name of the user to Public. The providers will not receive any session-related information like the session ID or the time the
user logged in. This option is the equivalent of the LOGIN_FREQUENCY_PUBLIC in the provider registration API (see Section 8.13, "Registering Providers Programmatically").

- **User**: Choosing this option sends the name of the OracleAS Portal user to the providers. This section contains the following two options:
  - **Login Frequency**: Here, you can select one of three options (always, once per user session, and never) to determine how often the session information must be sent to the provider, and thus how often the user needs to log in.
  - **Require Portal user-specific session information**: Here, you can specify whether the session information will be sent in the provider calls.

### 2.5 Caching Style

Caching plays an essential role in ensuring that your portal is highly performant. OracleAS Portal supports caching on various levels, such as caching pages, portlets, styles, and page metadata. Caching portlets is key to delivering accurate information in a timely manner to your users. All portlet building technologies, available with OracleAS Portal, support caching.

As OracleAS Portal supports user personalization of pages and portlets, the view of a page can vary from user to user. OracleAS Portal's caching is designed to allow content to vary for each user. Therefore, portal objects, including portlets, can be cached at the following two levels: user level and system level.

- **User-level caching** is for a specific user; the cache entries stored are unique for that user and cannot be accessed by other users. Good candidates for user-level caching are portlets supporting personalization, such as e-mail or stock ticker portlets.

- **System-level caching** enables users to share a single cache entry and, therefore, there is no need to cache a copy of the object for every user. Examples of content that might be suitable for system-level caching are news portlets that are not personalizable, or custom-built navigation portlets.

When not using caching, you may find accessing various data sources with Web Clipping, OmniPortlet, and Portlet Builder to be time consuming. When you enable caching, you instruct OracleAS Portal or OracleAS Web Cache to maintain a copy of the portlet content. If the portlet is requested and the content was cached previously, the portlet does not have to spend time contacting the data source and regenerating its content again. Simply, the previously cached portlet content is returned.

- **Expiry-based caching**: You can use expiry-based caching when the portlet content is static or when it is not critical that the most up-to-date content be displayed. When using expiry-based caching, you must specify the caching period.

- **Validation-based caching**: You can use validation-based caching for portlets with dynamic content that changes frequently or unpredictably. The portlet associates its content with a caching key and returns the key value along with the content. When the portlet content is requested, the portlet decides, based on the caching key, if the current content is valid. If the portlet content is valid, then it returns a response indicating that the cached content can be used (that is, the content is valid) or generates the new portlet content and returns it along with a new caching key for that content.

- **Invalidation-based caching**: Invalidation-based caching is the most complex, but also the most flexible, form of caching. It combines the efficiency of expiry-based
caching with the ability to invalidate the cache content any time. Objects in OracleAS Web Cache are considered valid until they are invalidated explicitly.

2.5.1 Web Clipping, OmniPortlet, and Portlet Builder
For portlets built with Web Clipping, OmniPortlet, and Portlet Builder you can specify a period of time for which they are cached (expiry-based caching). In addition to this, portlets built with Web Clipping and OmniPortlet are refreshed automatically when the end user personalizes them.

2.5.2 Java Portlets
Java portlets support three types of caching: expiry-, validation-, and invalidation-based caching. With Java portlets, you can combine invalidation-based caching with either expiry-based or validation-based caching.

In addition to caching all your portlet's content, you can also cache fragments of your portlets by using Edge Side Includes (ESI).

2.5.3 PL/SQL Portlets
Similar to Java portlets, PL/SQL portlets also support three types of caching: expiry-, validation-, and invalidation-based caching.

2.6 Development Tool
This section describes the type of development tool you may use to build the portlet. For example, OmniPortlet is built in a browser-based wizard while Java portlets may be built in a tool like Oracle JDeveloper.

2.6.1 Web Clipping, OmniPortlet, and Portlet Builder
OmniPortlet, Web Clipping, and Portlet Builder use a browser-based wizard as the development tool.

2.6.2 Java Portlets
To build Java portlets, the only requirement is the PDK-Java. It is highly recommended, though, that you use Oracle JDeveloper, a professional, integrated development environment (IDE). While you can consider other IDEs, the PDK contains an Oracle JDeveloper plug-in that includes the Java Portlet Wizard, to minimize your Java portlet development efforts.

The Java Portlet Wizard generates a starting skeleton and file structure for both JSR 168 and PDK-Java portlets. You need to add only your own business logic to the skeleton. JDeveloper can also package and deploy your applications to your J2EE container, such as OracleAS Containers for J2EE (OC4J). Also, Oracle JDeveloper helps you test your portlet provider. You can use the integrated standalone OC4J that is shipped with Oracle JDeveloper as your development Java portlet runtime environment, if the version matches that of the platform on which you plan to deploy.

2.6.3 PL/SQL Portlets
When developing a PL/SQL portlet, you create PL/SQL program units that access OracleAS Portal by calling OracleAS Portal PL/SQL APIs. To enable this access, you create a schema, the provider schema, to store the provider and portlet PL/SQL
packages in the same database in which OracleAS Portal is installed. The provider schema must be granted execute privileges on the OracleAS Portal PL/SQL APIs.

To facilitate the development of database providers and PL/SQL portlets, you can use the PL/SQL Generator, a hosted utility that creates installable PL/SQL code for a database provider and its PL/SQL portlets. The PL/SQL Generator is a Web application that receives the provider and portlet definitions in the form of an XML file. The syntax of the XML tags that are used for the provider and portlet definition is a subset of the XML tags that are used for defining Web providers with the PDK-Java. The output of the PL/SQL Generator is a SQL script that can be run from SQL*Plus. The script contains SQL commands for installing the provider and portlet packages.


2.7 Portlet Creation Style

OracleAS Portal supports the following two types of portlet creation:

- "Develop in-place"
- "Develop first, add later"

The Figure 2–5 indicates that the "Develop first, add later" portlet creation is usually the task of the portlet developer, while the "Develop in-place" portlet creation is the page designer’s responsibility.

Figure 2–5 Portlet Creation Style
2.7.1 OmniPortlet and Web Clipping

OmniPortlet and Web Clipping offer the same approach to creating portlets. First you add the portlets to a portal page and then you define them in place on the page.

2.7.2 Java Portlets

Java portlets do not tend to provide a develop-in-place experience. You can easily add edit defaults and personalization to your Java portlets.

Note: With extensive coding, you can create develop in place Java portlets. For example, Web Clipping and OmniPortlet are both Java portlets.

2.7.3 Portlet Builder

With Portlet Builder you define the portlets first. The previously defined portlets are then made available to you in the portlet repository so you can add them to your pages. For simple portlets, though, Portlet Builder offers you the develop in place experience, similar to OmniPortlet and Web Clipping.

Note: Portlets built with Portlet Builder's develop-in-place technology are somewhat limited as compared to those built using the Navigator.

2.7.4 PL/SQL Portlets

Similar to the Java portlets, PL/SQL portlets typically follow the “Develop first, add later” creation path. Extensive coding is required to develop-in-place PL/SQL portlets. For example, simple in-place portlets that are offered by Portlet Builder are written in PL/SQL.

2.8 User Interface Flexibility

This section describes the portlet building tools in terms of the control you have over the user interface.

2.8.1 Web Clipping

Because of its nature, Web Clipping always displays the remote Web site content, therefore UI flexibility is not a requirement for this portlet.

2.8.2 OmniPortlet

OmniPortlet enables you to use a number of different pre-built layouts, such as scrolling news, tabular, and chart. You can also use the built-in HTML layout to personalize the look and feel of your portlet using HTML and JavaScript.

2.8.3 Java Portlets and PL/SQL Portlets

In Java portlets and PL/SQL portlets, you have full control over your portlet's user interface. Your portlet is free to generate any HTML content that conforms the rendering rules for OracleAS Portal pages.
2.9 Ability to Capture Content from Web Sites

This section describes the portlet building tools in terms of their ability to include content from other sources.

2.9.1 Web Clipping

In the event that you have to create a portlet that displays the content from a remote Web site as it is presented at the source location, the best tool to use is Web Clipping. Web Clipping can tolerate the changes of the source HTML page to some extent. If a clipped table moves from one place to another in the source page, the Web Clipping engine can find the table again using the internal "fuzzy match" algorithm. Portlets built with Web Clipping can also maintain sessions to the remote Web sites. Web Clipping also supports the end user personalization of HTML form values.

2.9.2 OmniPortlet

Another possible scenario is that you are interested in the data only, not in the way it is presented on the remote Web site. You want to retrieve the data, process the data (format, filter, and so on), and present it in a portlet in a tabular, chart, or news format. For this purpose, OmniPortlet is the best choice. OmniPortlet is a powerful tool that extracts data from Web pages by using its Web data source.

2.9.3 Java Portlets

In your Java portlets, similarly to other Java applications, you can always take advantage of the low-level Java networking APIs to retrieve and process content from remote Web sites. To avoid unnecessary development efforts, before choosing Java always make sure that Web Clipping or OmniPortlet are not viable options for you.

2.9.4 PL/SQL Portlets

PL/SQL portlets can communicate with Web servers to access data on the Internet by using procedures and functions from the UTL_HTTP package. The package makes HTTP callouts from SQL and PL/SQL. The package also supports HTTP over the Secured Socket Layer protocol (SSL), also known as HTTPS, directly or through an HTTP proxy. Other Internet-related data-access protocols (such as the File Transfer Protocol (FTP) or the Gopher protocol) are also supported using an HTTP proxy server that supports those protocols.

2.10 Ability to Render Content Inline

Active elements in your portlets, such as links or form buttons, enable your users to navigate to remote URLs. In a News portlet, for example, you can click a hyperlink to navigate to a news site with detailed information about the news you are interested in. You leave the portal page; the News site replaces it in your browser.

However, you may be required to keep your users within the context of the portal page by rendering the requested content within the same portlet container. In the News portlet, for example, you have to display the detailed news on the portal page within the boundaries of the same portlet.
2.10.1 Web Clipping

Web Clipping has URL rewriting support to achieve this functionality: it can process the links, originating from the source Web site, and modify (rewrite) them to achieve the desired functionality.

You can choose from the following three options:

- You can select not to rewrite the URLs within the portlet, in which case clicking the links takes the users out of Portal to the Web site providing the clipping. If the Web Clipping provider is registered with an External Application, this may require that the user enter login information before navigating the Web site.

- If the Web Clipping provider is registered with an External Application and the clipping requires authentication, you can instruct Web Clipping to rewrite all URLs within the portlet to point to the Login Server. In this case, navigation will cause the user to leave OracleAS Portal, while also using the Login Server to log the browser into the External Application.

- You can select to rewrite all URLs within the portlet (inline rendering) to point back to the portal page so that all browsing within the Web Clipping portlet remains within OracleAS Portal. If the Web Clipping provider is registered with an External Application, this will cause the Web Clipping provider to log itself into the External Application. In this case, the navigation within Portal through the Web Clipping provider is authenticated in the External Application.

2.10.2 OmniPortlet

OmniPortlet does not offer URL rewriting directly, but you can achieve the inline rendering functionality by using public portlet parameters and events. Then you have to map the events to the same portal page where your OmniPortlet resides.

2.10.3 Java Portlets

Since you have full control over the links and buttons in Java portlets, you can easily implement the inline rendering functionality. You have to append the private portlet parameters to the page URL.

If you use Struts in your portlet, the PDK-Struts integration framework renders your content always in the same portlet container.

If your portlet consists of multiple JSPs (for example, several steps in a survey, or in a Wizard), your portlet can make use of a special parameter to specify at runtime the JSP to use to render the content.

2.10.4 Portlet Builder

Portlets built with Portlet Builder do not have inherent inline rendering support. You can, however, construct your links in SQL-based reports and charts so that they point to specific portal pages. If required, you can also pass parameters to portal pages, which in turn can be mapped to portlet parameters.

2.10.5 PL/SQL Portlets

Similar to Java portlets, you have full control over the active elements in PL/SQL portlets and, therefore, you can achieve the inline rendering functionality programmatically by implementing private portlet parameters.
2.11 Charting Capability

This section describes the portlet building tools in terms of their charting functionality.

2.11.1 Web Clipping

Because of its nature, Web Clipping can retrieve and present HTML content containing charts, but it does not support the creation of charts.

2.11.2 OmniPortlet

OmniPortlet supports the following three chart types: bar, line, and pie. Charts in OmniPortlet are dynamically generated images with, optionally, event-enabled hyperlinks.

2.11.3 Java Portlets

You can create more sophisticated chart portlets programmatically in your Java portlets using Oracle’s Business Intelligence (BI) Beans.

---

**Note:** Oracle Reports and Oracle Discoverer portlets use BI Beans to create professional graphs.

---

2.11.4 Portlet Builder

With Portlet Builder, you can build HTML-based bar chart portlets. Among other features, you can specify the color and orientation of the bars.

2.11.5 PL/SQL Portlets

In PL/SQL portlets, HTML-based charting can be achieved by extensive coding.

2.12 Public Portlet Parameters Support

There are three types of parameters in OracleAS Portal: page parameters, public portlet parameters, and private portlet parameters.

- **Page parameters:** You can use a page parameter to pass a value to a page. Using page parameters, the information that is displayed on a page can vary depending on where the page is called from and who is viewing the page. Using page parameters, page designers can synchronize the portlets on a page by passing them the same values. This provides the ability to reuse and tailor portlets on pages by merely integrating them with page parameters. Without this functionality, you would have to code portlets individually to use different parameter values.

- **Public portlet parameters:** You can use a public portlet parameter to pass a value to a portlet. Using portlet parameters, the information that is displayed in a portlet can be specific to a particular page or a user. Portlet parameters are created by the portlet developer and are exposed to the page designer, through the user interface. After adding a portlet to a page, page designers can assign values to the public portlet parameters to make the information displayed in the portlet specific to the page.

  Page designers can assign values to public portlet parameters by providing a specific value (constant), a system variable (for example, the portal user name), or
a page parameter. At run time, the portlet receives the values from the sources specified. In this way, page designers have complete control over the source of the parameter, whereas you have complete control over how the data is used after it is transmitted to the portlet.

- **Private portlet parameters:** You can use private portlet parameters to implement internal navigation in your portlet. You can pass parameters to your portlets every time the page is requested. Private portlet parameters can be passed exclusively from the portlet instance to the same portlet instance.

Private portlet parameters do not require a full page refresh. You can create a link in a portlet that can be used to refresh the portlet only, without triggering a full portal page refresh. By doing this, only the content of the affected portlet is updated and the rest of the page is not. Refer to Partial Page Refresh for details about setting this programmatically.

Portlets supporting public portlet parameters enable page designers to tailor the portlets’ data input for each portlet instance. In this case, the portlet developer can focus on the portlet logic, while page designers can easily reuse portlets and address the interaction between the page and the portlets.

All five portlet building technologies discussed in this chapter (OmniPortlet, Web Clipping, Java portlets, Portlet Builder, and PL/SQL portlets) support public portlet parameters. OmniPortlet, Web Clipping, and Portlet Builder provide complete support through their wizard interface. You can add public portlet parameter support to your Java portlets programmatically or with the Java Portlet Wizard. PL/SQL portlets support public parameters only programmatically.

---

**Note:** The JSR 168 standard does not cover the notion of public portlet parameters. If you want to utilize public portlet parameters in your Java portlets, you have to use PDK-Java.

---

### 2.13 Private Portlet Parameter Support

This section describes the portlet building tools in terms of their support for private parameters.

#### 2.13.1 OmniPortlet, Web Clipping, and Portlet Builder

OmniPortlet, Web Clipping, and Portlet Builder do not provide access to the portlet developer to private portlet parameters.

#### 2.13.2 Java Portlets and PL/SQL Portlets

In your Java portlets and PL/SQL portlets, you can implement internal navigation by using private portlet parameters.

---

**Note:** PL/SQL portlets do not support private and public parameters simultaneously. You need to decide which parameter type to support before coding your PL/SQL portlet.

---

### 2.14 Event Support

An event is a user action that you define to display a Portal page. User actions include clicking a link or a button in a portlet. Page designers specify what to do when an
event occurs in a portlet on a page. When an event occurs, page designers can either redisplay the current page or navigate the user to another portal page, optionally passing values to that page’s parameters.

2.14.1 Web Clipping, OmniPortlet, and Java Portlets
Web Clipping, OmniPortlet, and Java portlets support events.

2.14.2 Portlet Builder and PL/SQL Portlets
Portlet Builder and PL/SQL portlets do not support events.

2.15 Ability to Hide and Show Portlets Based on User Privileges
This section describes the portlet building tools in terms of their support for authorization functionality.

2.15.1 Web Clipping and OmniPortlet
You can hide and show portlets built with Web Clipping and OmniPortlet on portal pages dynamically by using security managers. Although Web Clipping and OmniPortlet do not expose security managers through the user interface, you can apply them by editing their XML provider definition file.

2.15.2 Java Portlets
The PDK provides a number of security managers for Java portlets. Two examples are as follows:

- **Group security manager**: The group security manager makes the portlet appear to users who are members of a specified group, while hiding it from those who are not members.

- **Authentication level security manager**: You can use the authentication level security manager to control access to the portlets based on the user’s authentication level. For example, you may hide the portlet from public users but display it to authenticated users.

JSR 168 portlets support the standard servlet mechanisms.

2.15.3 Portlet Builder
Portlet Builder provides a declarative user interface to control access to portlets.

2.15.4 PL/SQL Portlets
The PDK provides the Security APIs to implement hiding and showing content in PL/SQL portlets.

2.16 Multilingual Support
This section describes the portlet building tools in terms of their support for other languages.
2.16.1 Web Clipping, OmniPortlet, Java Portlets, and PL/SQL Portlets

Web Clipping, OmniPortlet, Java portlets, and PL/SQL portlets display textual information in the language selected by the portal user.

2.16.2 Portlet Builder

Portlets built with Portlet Builder support English only.

2.17 Pagination Support

Support for pagination is useful when you are required to display a relatively large set of records in your portlets.

2.17.1 Web Clipping

Pagination support is not applicable to Web Clipping.

2.17.2 OmniPortlet

OmniPortlet does not support pagination.

2.17.3 Java Portlets and PL/SQL Portlets

You can implement pagination in your Java portlets and PL/SQL portlets programmatically.

2.17.4 Portlet Builder

Portlet Builder has built-in support for pagination.

2.18 Single Sign-On and External Application Integration

This section describes the portlet building tools in terms of authentication for external application.

2.18.1 Web Clipping

Web Clipping’s integration with the external application framework provides a fully automated mechanism to store passwords to external Web sites. All you have to do is to associate an External Application ID to the Web Clipping provider when registering the provider.

2.18.2 OmniPortlet

OmniPortlet enables you to store connection information when the data source is password protected. The credentials to access the data source can either be shared across all users, or saved individually for each user. OmniPortlet is capable of storing database credentials, as well as HTTP basic authentication user name-password pairs. The credentials are stored in the secured data repository of OmniPortlet, in an Oracle database.
2.18.3 Java Portlets

Java portlets can integrate with the external application framework as well as any LDAP server, such as Oracle Internet Directory, programmatically.

2.18.4 PL/SQL Portlets

You can build PL/SQL portlets that enable single sign-on by using mod_osso, an authentication module on the Oracle HTTP Server. mod_osso is a simple alternative to the single sign-on SDK, used in earlier releases to integrate partner applications. mod_osso simplifies the authentication process by serving as the sole partner application to the Single Sign-On server.

PL/SQL portlets can integrate with the external application framework programmatically.

2.19 Summary

In this chapter, you learned about the tools and technologies available in OracleAS Portal, the benefits of each, as well as the different methods of deploying portlets. For specific information on each of these tools and technologies, refer to the individual chapters in this manual:

■ Chapter 3, "Creating Portlets with OmniPortlet"
■ Chapter 5, "Creating Content-Based Portlets with Web Clipping"
■ Chapter 6, "Creating Java Portlets"
■ Chapter 8, "Creating PL/SQL Portlets"
■ Appendix A, "Creating Portlets with the Portlet Builder"
Part II
Creating Portlets

Part II contains the following chapters:

- Chapter 3, "Creating Portlets with OmniPortlet"
- Chapter 4, "Building Example Portlets with OmniPortlet"
- Chapter 5, "Creating Content-Based Portlets with Web Clipping"
- Chapter 6, "Creating Java Portlets"
- Chapter 7, "Enhancing Java Portlets"
- Chapter 8, "Creating PL/SQL Portlets"
This chapter provides an overview of OmniPortlet and explains the user interface elements associated with OmniPortlet. This chapter contains the following sections:

- **What is OmniPortlet?**
- **The OmniPortlet Wizard**
- **Parameters and Events**

For information on using OmniPortlet to build example portlets, refer to Chapter 4, "Building Example Portlets with OmniPortlet". For troubleshooting information regarding OmniPortlet, refer to Section B.3, "Diagnosing OmniPortlet Problems". For information on registering and configuring OmniPortlet with Oracle Application Server Portal, refer to Appendix I, "Configuring the Portal Tools Providers" in the Oracle Application Server Portal Configuration Guide.

### 3.1 What is OmniPortlet?

OmniPortlet is a subcomponent of Oracle Application Server Portal that enables page designers and developers to easily publish data from various data sources using a variety of layouts. You can base an OmniPortlet on almost any kind of data source, such as a spreadsheet (character-separated values), XML, and even application data from an existing Web page.

---

**Note:** You can find more information about developing different types of portlets in Chapter 1, "Understanding Portlets", and information about providers and otherportlet technologies in Chapter 2, "Portlet Technologies Matrix".

---

OmniPortlet enables page designers and content contributors to perform the following actions:

- Display data from multiple sources (CSV, XML, SQL, and so on)
- Sort the data to display
- Format data using a variety of layouts (bulleted list, chart, HTML, and so on)
- Use portlet parameters
- Raise portlet events
- Expose personalizable settings to page viewers

To display personalized data, you can refine the retrieved data by filtering the results returned from a data source, and parameterize the credential information used to
access secure data. Out of the box, OmniPortlet provides the most common layout for portlets: tabular, chart, HTML, news, bulleted list, and form.

---

**Note:** To use OmniPortlet, the Simple Parameter Form, or the Web Clipping portlet, you must use Netscape 7.0 or higher, or Microsoft Internet Explorer 5.5 or higher for Windows 2000.

---

OmniPortlet is provided with Oracle Application Server 10g; you can add an OmniPortlet from the OracleAS Portal Repository in the Portlet Builders folder. If you’ve downloaded OmniPortlet as part of the Oracle Application Server Portal Developer Kit, you must register it before you can use it. To learn more about registering a Web provider, refer to the Oracle Application Server Portal Configuration Guide, located on the OracleAS Portal Documentation page on OTN (http://www.oracle.com/technology/products/ias/portal/documentation.html). You can then add an OmniPortlet to any portal page.

You can find more information about building portal pages in the Oracle Application Server Portal User’s Guide on the OracleAS Portal Documentation page on OTN. You can also upgrade to later releases of OmniPortlet from the Portlet Development page on OTN (http://www.oracle.com/technology/products/ias/portal/) as part of the Oracle Application Server Portal Developer Kit (PDK).

Instructions for installing, configuring, and registering the OmniPortlet provider are provided within the pdksoftware.zip file containing the PDK-Java and Portal Tools. For specific information on configuring OmniPortlet, refer to Appendix I, "Configuring the Portal Tools Providers" in the Oracle Application Server Portal Configuration Guide.

### 3.2 The OmniPortlet Wizard

The OmniPortlet Wizard initially contains six steps (Table 3–1). When you first define your OmniPortlet, you set the data source type, data source options, filter options, view options, and layout. When you’ve completed these steps of the wizard, you can re-enter the wizard by clicking Edit Defaults for the portlet. When you re-enter the wizard, you can change the definitions on the Source, Filter, View, and Layout tabs, as well as set up the event parameters on the Events tab.

This section provides a high-level overview of the six tabs (see the following table).

You can also find information in the online Help (accessible by clicking the Help link in the product), which describes the options on each tab.

<table>
<thead>
<tr>
<th>Table 3–1</th>
<th>OmniPortlet Wizard and Edit Defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step/Tab</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Type</td>
<td>Provides your data source options. Displays only in the initial definition of the portlet, and is not available when editing the defaults of the portlet.</td>
</tr>
<tr>
<td>Source</td>
<td>Provides the options for the selected data source, such as the URL of the Web Service you wish to use. You can change these options later when editing the defaults of the portlet.</td>
</tr>
<tr>
<td>Filter</td>
<td>Provides sorting options at the OracleAS Portal level to enable you to refine your results. You can change these options later when editing the defaults of the portlet.</td>
</tr>
</tbody>
</table>
3.2.1 Type

When you first launch OmniPortlet, the Type step displays (Figure 3–1), which enables you to choose your data source.

Figure 3–1 Type Tab of the OmniPortlet Wizard

Note: If you’ve downloaded and installed an additional data source, the data source will display on the Type tab.

Out of the box, OmniPortlet supports the following data sources (Table 3–2):

<table>
<thead>
<tr>
<th>Data Source Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spreadsheet</td>
<td>Displays data from a text file containing character-separated values (CSV).</td>
</tr>
<tr>
<td>SQL</td>
<td>Displays data from a database using SQL.</td>
</tr>
<tr>
<td>XML</td>
<td>Displays data from an XML file.</td>
</tr>
<tr>
<td>Web Service</td>
<td>Displays data from a discrete business service that can be accessed over the Internet using standard protocols.</td>
</tr>
<tr>
<td>Web Page</td>
<td>Displays data based on existing Web content.</td>
</tr>
</tbody>
</table>
After you complete the OmniPortlet Wizard and edit the defaults of the portlet, you cannot change the data source type.

### 3.2.2 Source

After you’ve chosen your data source type, the Source step of the OmniPortlet Wizard displays. This step adapts to the data source you’ve chosen, to enable you to specify the options offered by that data source. The Source tab contains a Proxy Authentication section if the OmniPortlet provider has been configured to use a proxy server requiring authentication, and a Connection section where you can provide the necessary information for connecting to the data source.

This section contains information about the following two areas on the Source tab:

- **Proxy Authentication**
- **Connection Information**

Later, this section also describes the portion of the Source tab specific to each of the following data sources:

- Spreadsheet
- SQL
- XML
- Web Service
- Web Page

**Note:** For more information on the Source tab options, click Help in the upper right corner of the page.

#### 3.2.2.1 Proxy Authentication

OmniPortlet supports proxy authentication, including support for global proxy authentication and per-user authentication. You can specify whether all users will automatically log in using a user name and password you provide, each user will log in using an individual user name and password, or all users will log in using a specified user name and password. If the OmniPortlet provider has been set up to use proxy authentication that requires your login, a Proxy Authentication section displays on the Source tab where you can enter this information.

The Proxy Authentication section only displays for the data sources that may require you to use a proxy server to access them: CSV (character-separated values), XML, Web Service, and Web Page. For more information on configuring the OmniPortlet provider to use proxy authentication, see the online Help topic that displays when you click Help on the Edit Providers: OmniPortlet Provider page. If the OmniPortlet provider is configured to “Require login for all users,” each user must set his or her own login information:

---

**Table 3–2 (Cont.) Supported Data Source Types**

<table>
<thead>
<tr>
<th>Data Source Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>J2EE Connector Architecture*</td>
<td>(Displays only if the Sample Provider is registered with OracleAS Portal). A J2EE Connector Architecture (JCA) 1.0 adapter is also available. JCA provides a mechanism to store and retrieve enterprise data such as that held in ERP systems (Oracle Financials, SAP, PeopleSoft, and so on).</td>
</tr>
</tbody>
</table>
- For page designers, set this in Edit Defaults: Source tab.
- For page viewers, set this on the Personalize screen.

You can also find more information on configuring OmniPortlet in Appendix I, "Configuring the Portal Tools Providers" in the Oracle Application Server Portal Configuration Guide, available on the OracleAS Portal Documentation page on OTN.

---

**Note:** If you are using the Web Page data source, the Proxy Authentication section displays in the Web Clipping Studio, after you have clicked the Select Web Page button on the Source tab.

---

### 3.2.2.2 Connection Information

For each data source except the Web Page data source, the Source step contains a Connection section, where you can define the connection information to access secured data. The Source step for all data sources include a Portlet Parameters section, where you can define the parameters for the portlet (Figure 3–2). You can then map the portlet parameters to the page-level parameters.

**Figure 3–2 Source Tab: Connection and Portlet Parameters Section**

**Connection**
To access secured data, you will need to provide connection information to the data source.

- Use this connection information for all users
- User must re-enter connection information

**Portlet Parameters**
Parameters are passed to the portlet from the page when the portlet is displayed. These parameters can be mapped to page level parameters.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Default Value</th>
<th>Customizable</th>
<th>Customize Page Label</th>
<th>Customize F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Param1</td>
<td></td>
<td></td>
<td>Param1</td>
<td>Description</td>
</tr>
<tr>
<td>Param2</td>
<td></td>
<td></td>
<td>Param2</td>
<td>Description</td>
</tr>
<tr>
<td>Param3</td>
<td></td>
<td></td>
<td>Param3</td>
<td>Description</td>
</tr>
<tr>
<td>Param4</td>
<td></td>
<td></td>
<td>Param4</td>
<td>Description</td>
</tr>
<tr>
<td>Param5</td>
<td></td>
<td></td>
<td>Param5</td>
<td>Description</td>
</tr>
</tbody>
</table>

To edit the connection information, click the Edit Connection button and fill out the information on the page shown in Figure 3–3. On this page, you can enter a name for the connection information, as well as the user name and password. For the SQL data source, you can enter more information to specify the driver you wish to use to connect to the data source. For more information, refer to Section 3.2.2.4, "SQL".
3.2.2.3 Spreadsheet

Spreadsheets are a common method of storing small data sets. OmniPortlet enables you to share spreadsheets by supporting character-separated values (CSV) as a data source. On the Source tab, you specify the location of the CSV file (Figure 3–4). If the file is located on a secure server, you can specify the connection information in the Connection section described in Figure 3–2. You can also select the character set to use when OracleAS Portal reads the file, as well as the delimiter and text qualifier.

Since the OmniPortlet provider exists and executes in a tier different from the OracleAS Portal application and does not have access to the OracleAS Portal session information, you must expose CSV files that are uploaded to OracleAS Portal as PUBLIC in order for OmniPortlet to access them.

Note: For more information on using the CSV data source, refer to Section 4.3, "Building an OmniPortlet Based on a Spreadsheet (CSV)."

3.2.2.4 SQL

The relational database is the most common place to store data. OmniPortlet enables you to use standard JDBC drivers and provides out-of-the-box access to Oracle and
any JDBC database. You can specify the driver type when you configure the connection information.

3.2.2.4.1 SQL Connection Information  You can also use the DataDirect JDBC drivers to access other relational databases. To configure OmniPortlet to use these drivers, refer to Appendix I, "Configuring the Portal Tools Providers" in the Oracle Application Server Portal Configuration Guide, available on the OracleAS Portal Documentation page on OTN (http://www.oracle.com/technology/products/ias/portal/documentati on.html). After the driver is installed, you'll notice it listed in the Driver Name drop-down list on the Connection dialog box on the Source tab, as shown in Figure 3–5.

Figure 3–5  Connection Information on the SQL Source Tab

![Connection Information on the SQL Source Tab]

Note:  For more information on using DataDirect drivers with OmniPortlet, refer to the "How to Use DataDirect JDBC Drivers with OmniPortlet" technote available on the Portlet Development page of Portal Center (http://portalcenter.oracle.com).

For more information about DataDirect drivers in general, refer to the Certification Matrix for Oracle Application Server and DataDirect JDBC (http://www.oracle.com/technology/tech/java/oc4j/htdocs/datadirect-jdbc-certification.html) and the OC4J page on OTN (http://www.oracle.com/technology/software/product s/ias/htdocs/utilsoft.html).

When you want to use one of the DataDirect drivers, you must use a unique connection string format: hostname:port, where hostname is the name of the server where the database is running, and port is the listening port of the database. You can see an example in Figure 3–6.
3.2.2.4.2 Using Stored Procedures

You can also make a call to Stored Procedures instead of SQL statement to add business logic to your data. You can create your package and stored procedure in your database and refer the stored procedure in OmniPortlet.

For example, you could do the following using the SCOTT sample schema:

1. Create a package and declare a ref cursor:
   
   ```
   create or replace package emp_pack is
     type empcurr is ref cursor;
   end;
   ```

2. Define a stored procedure, for example following procedure accepts JOB as parameter and returns a ref cursor, where JOB Column in the `scott.Emp` table, its value can be CLERK, MANAGER, and so on.

   ```
   create or replace procedure emp_proc(eset OUT emp_pack.empcurr,
                                        jname IN VARCHAR2) 
   is
     sql_statement varchar2(200);
   begin
     sql_statement := 'select empno,ename,hiredate 
                      from emp 
                      where job = '''||jname||''' 
                      order by EMPNO,hiredate';
     open eset for sql_statement;
   end;
   ```

3. Add the PL/SQL statements from steps 1 and 2 to an SQL file (for example, `proc.sql`) and save it to a directory.

4. Connect to the database using the following command:
   
   ```
   sqlplus scott/tiger@Connection_String
   ```
   
   Replace `Connection_String` with the connection string to your database. You can find the connection string in the `tnsnames.ora` file within your `ORACLE_HOME/network/admin` directory.

5. Run the procedure:
   
   ```
   @proc
   ```

6. Finally, create an OmniPortlet based on the SQL data source, enter the appropriate database connection information. In the SQL Statement box, enter the following code:

   ```
   call emp_proc('CLERK')
   ```
3.2.2.5 XML

Although still a relatively new method of storing data, XML is increasingly used to control access to data sets over the intranet, and even the Internet. On the Source tab, you can specify the URL of the XML file that contains your data (Figure 3–7).

**Note:** For more information on using the XML data source, refer to Section 4.4, "Building an OmniPortlet Based on an XML Data Source".

**Figure 3–7 Source Tab: XML**

Next to the XML URL and the XSL Filter URL fields are Test buttons which you can use to validate your XML data source and the XSL filter.

The specified XML file can either be in tabular (ROWSET/ROW) structure, or you can provide an XML Style Sheet (XSL) to transform the data into the ROWSET/ROW structure. Figure 3–8 shows an example of the ROWSET/ROW structure of an XML data source.

**Figure 3–8 ROWSET/ROW Structure of an XML Data Source**

```xml
<Team>
  <Employee>
    <DEPTNO>10</DEPTNO>
    <ENAME>KING</ENAME>
    <JOB>PRESIDENT</JOB>
    <SAL>5000</SAL>
  </Employee>
  <Employee>
    <DEPTNO>20</DEPTNO>
    <ENAME>SCOTT</ENAME>
    <JOB>ANALYST</JOB>
    <SAL>3000</SAL>
  </Employee>
</Team>
```

In the previous example, the <TEAM> tags delineate the rowset, and the <EMPLOYEE> tags delineate the rows.

Regardless of the format of the XML file, OmniPortlet automatically inspects the XML to determine the column names, which will then be used to define the layout. If you want to specify this information yourself, you can supply a URL to an XML schema that describes the data.

Similar to the other data sources, you can also specify the connection information for this data source, if the XML file is located on a secured server protected by HTTP Basic Authentication.
3.2.2.6 Web Service

A Web Service is a discrete business service that can be programmatically accessed over the Internet using standard protocols, such as SOAP and HTTP. Web Services are non-platform and non-language specific, and are typically registered with a Web Service broker. When you find a Web Service you wish to use, you must obtain the URL to the WSDL (Web Service Description Language) file that describes the Web Service and specifies the methods that can be called, the expected parameters, and a description of the returned data.

OmniPortlet supports both types of Web Services: Document and RPC (Remote Procedure Calls). After a WSDL document/file is supplied, it is parsed, and the available methods that can be called display on the Source tab (Figure 3–9).

Similar to the XML data source, OmniPortlet expects the Web Service data in ROWSET/ROW format, though you can also use an XSL file to transform the data. OmniPortlet inspects the WSDL document/file to determine the column names, though you may also specify an XML schema to describe the returned data set.

If you are new to Web Services, you may want to first review the New to Web Services guide on the Web Services Technology Center page on OTN (http://otn.oracle.com/tech/webservices/learner.html).

**Note:** Since the OmniPortlet provider exists and executes in a different tier from OracleAS Portal and does not have access to the OracleAS Portal session information, you must expose XML files that are uploaded to OracleAS Portal as PUBLIC in order for OmniPortlet to access them.

**Note:** For more information on using the Web Service data source, refer to Section 4.2, "Building an OmniPortlet Based on a Web Service".

---

**Figure 3–9 Source Tab: Web Service**

![Source Tab: Web Service](image)
3.2.2.7 Web Page

OmniPortlet enables you to use existing Web content as a source of data to publish information to your portal. It provides and renders clipped Web content as a data source.

The Web Page data source extends the scope offered by the Web Clipping Portlet to include scraping functionality. It also supports the following features:

- **Navigation through various login mechanisms**, including form- and JavaScript-based submission, and HTTP Basic and Digest Authentication with cookie-based session management.

- **Fuzzy matching of clippings**. If a Web clipping gets reordered within the source page or if its character font, size, or style changes, it will still be identified correctly by the Web page data source and delivered as the portlet content.

- **Reuse of a wide range of Web content**, including basic support of pages written with HTML 4.0.1 and JavaScript, retrieved through HTTP GET and POST (form submission).

All Web clipping definitions are stored persistently in the Oracle Application Server infrastructure database or on another Oracle database. Any secure information, such as passwords, is stored in encrypted form, according to the DES (Data Encryption Standard), using Oracle Database encryption technology.

The Source tab of the OmniPortlet Wizard enables you to launch the Web Clipping Studio by clicking the Select Web Page button (Figure 3–10). Once you launch the Web Clipping Studio, you can refer to the Oracle Application Server Web Clipping online Help.

---

**Note:** For more information on using the Web Page data source, refer to Section 4.5, "Building an OmniPortlet Based on a Web Page Data Source".

---

Figure 3–10  **Source Tab: Web Page**

You can use a web page as a data source. When the portlet is displayed, data is extracted from the web page. Any changes to the data on the web page are automatically reflected in the portlet.

Select the web page and identify an area (clipping) of the page to use as data.

**Portlet Parameters**

Parameters are passed to the portlet from the page when the portlet is displayed. These parameters can be mapped to page level parameters by editing the Page Properties.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Default Value</th>
<th>Customizable</th>
<th>Customize Page Label</th>
<th>Customize Page Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Param1</td>
<td></td>
<td></td>
<td>Param1</td>
<td>Description for Param1</td>
</tr>
<tr>
<td>Param2</td>
<td></td>
<td></td>
<td>Param2</td>
<td>Description for Param2</td>
</tr>
<tr>
<td>Param3</td>
<td></td>
<td></td>
<td>Param3</td>
<td>Description for Param3</td>
</tr>
<tr>
<td>Param4</td>
<td></td>
<td></td>
<td>Param4</td>
<td>Description for Param4</td>
</tr>
<tr>
<td>Param5</td>
<td></td>
<td></td>
<td>Param5</td>
<td>Description for Param5</td>
</tr>
</tbody>
</table>

3.2.3 Filter

After you’ve selected the data source and specified the data source options, you can further refine your data by using OmniPortlet’s filtering options. To use filtering efficiently, it is better to refine the data as much as possible at the data source level on the Source tab, then use the options on the Filter tab to streamline the data. For example, if you are using a SQL data source, you could use a WHERE clause to return only specific data from the specified columns. In this case, you could skip the Filter tab and continue to the View page of the wizard. However, if there are no filtering options
at the data source level, you can use the options on the Filter tab to sort your data (Figure 3–11).

**Figure 3–11 Filter Tab**

**Filter**
Use this page to filter and order the data that appears in your portlet.

**Conditions**
Specify the conditions that the data must meet in order to appear in your portlet. Click the plus sign to specify conditions for additional columns.

<table>
<thead>
<tr>
<th>Column</th>
<th>Operator</th>
<th>Value</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;None&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Order**
Determine how the data should be ordered.

<table>
<thead>
<tr>
<th>Column</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;None&gt;</td>
<td></td>
</tr>
</tbody>
</table>

**Limit**
If desired, limit the number of rows that appear in the portlet.

- Do not limit results
- Limit to [ ] results

### 3.2.4 View

Once you’ve specified the data and sorted it, you can choose the view options and layout for your OmniPortlet. The View tab enables you to add Header and Footer text, choose a Layout style that you can later refine on the Layout tab, and enable caching (Figure 3–12). You can choose from the following layouts:

- tabular
- chart
- news
- bullet
- form
- HTML
3.2.5 Layout

The Layout tab changes depending on the Layout Style you chose on the View tab, and enables you to further personalize the appearance of your portlet. For example, OmniPortlet supports drill-down hyperlinks in the chart layout. That is, you can set up the chart so that when a user clicks on a specific part of the chart, an action occurs (for example, jump to another URL).

For the other layout styles, you can define each column to display in a specific format, such as plain text, HTML, an image, button, or field. For example, suppose you selected a data source that includes a URL to an image. To see this image, you can select Image for the display of this column. Each column can also be mapped to an action, similar to the behavior of chart hyperlinks.

The following layout styles are available with OmniPortlet:

- Tabular Layout
- Chart Layout
- News Layout
- Bullet Layout
- Form Layout
- HTML Layout

3.2.5.1 Tabular Layout

Once you've chosen the tabular style on the View tab, you can refine the layout on the Layout tab (Figure 3–13). Typically, you use the tabular layout if you have one or more
columns of data that you want to display in a table. You can choose Plain to display all rows in the table without any background color, or Alternating to display a background color for every other row in the table.

**Figure 3–13  Layout Tab: Tabular Style**

![Image of Tabular Style](image)

In the Column Layout section, you can choose which data columns to display in the portlet, then select a display format for the data (Figure 3–14). Here, you can set a column to display a hyperlink, so that a secondary Web page displays when the user clicks that column in the table. You can also specify whether the secondary Web page displays in a new window.

**Figure 3–14  Example of an OmniPortlet Using a Tabular Layout**

![Image of Tabular Layout](image)

**Note:** You can control the background color of a portlet through the portal page style. For more information on using portal page styles, refer to the *Oracle Application Server Portal User’s Guide*.

In the Column Layout section, you can choose which data columns to display in the portlet, then select a display format for the data (Figure 3–14). Here, you can set a column to display a hyperlink, so that a secondary Web page displays when the user clicks that column in the table. You can also specify whether the secondary Web page displays in a new window.

**Note:** For more information on using the OmniPortlet Wizard, click the Help link in the upper right corner of the Layout tab.

### 3.2.5.2 Chart Layout

You can use the chart layout to display your data graphically, as a bar, pie, or line chart (Figure 3–16). On the Layout tab, you select the chart style and the column layout (Figure 3–15). When you choose the column layout, you can choose the groups, or columns on which the labels will be based. The category defines the values that will be used to create the chart legend, and the value determines the relative size of the bars,
lines, or slices in the chart. You can also select whether the sections of the chart should point to a hyperlink, and whether the targeted information should display in a new window.

**Note:** To group the information in the chart, you must group the information at the data level (for example, in your SQL query statement). Also, if numeric values in a data source contain formatted strings, commas, or currency (for example, $32,789.00), they are considered to be text and ignored when the chart is generated. You should remove these formatting characters if you want them to be correctly read as numerical values.

**Figure 3–15  Layout Tab: Chart**

- **Column Layout:** Select the columns to display in the chart.
- **Chart Drilldown:** Select the action that will occur when the user clicks on the bar, pie slice or line.
You can also define chart hyperlinks so that each bar, pie section, or line links to another Web page. For example, you can display a chart portlet and a report portlet on your portal page, then set up the chart hyperlink to display a row in the report that displays more detailed information about the selected data.

In the Figure 3–17, you can see the results of the options selected on the Layout tab in the previous image. Following the chart, you can see that the category, which was Department on the Layout tab, is used for the legend.

Figure 3–17 Example of an OmniPortlet Using a Pie Chart Layout
3.2.5.3 News Layout

You can use the news layout to display links to articles with brief descriptions for each (Figure 3–18 and Figure 3–19). You can use this layout to publish information in standard XML formats, such as RDF (Resource Description Framework) or RSS (RDF Site Summary) to your portal page. In the Column Layout section, you can add a heading that displays at the top of the portlet. You can also add a logo, or use the scrolling layout so that the user can view all the information in the portlet as it moves vertically. Here, also, you can enter a URL so that another Web page displays when the user clicks on specific data in the portlet. You can also specify whether the secondary Web page displays in a new window.

Figure 3–18 Layout Tab: News

<table>
<thead>
<tr>
<th>Name</th>
<th>Column</th>
<th>Display As</th>
<th>Action</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field1</td>
<td>employee_id</td>
<td>Text</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Select the columns to display in the news Heading.

<table>
<thead>
<tr>
<th>Name</th>
<th>Column</th>
<th>Display As</th>
<th>Action</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field2</td>
<td>Name</td>
<td>Text</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Field3</td>
<td>Gender</td>
<td>Text</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Field4</td>
<td>Job</td>
<td>Text</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Field5</td>
<td>Email</td>
<td>Text</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Select the columns to display in the news Column.

Note: The News Layout Scroll type in OmniPortlet is supported on Microsoft Internet Explorer and Netscape 7.0.

Figure 3–19 Example of an OmniPortlet Using a News Layout

OmniPortlet

Kahlleen Bayyat
F
President - Manufacturing
kabayyat@oracle.com

Robert Rodriguez
M
President - Sales
rrodriguez@oracle.com

Edward Shields
M
Chief Financial Officer
eshields@oracle.com
3.2.5.4 Bullet Layout

You can use the bullet layout to display your data in a bulleted list (Figure 3–21). The Layout tab provides a variety of different bullet and numbered bullet styles (Figure 3–20). In the Column Layout section, you can choose how the columns will display in the portlet, as well as whether a second Web page will display when the user clicks that column. You can also specify whether the second Web page displays in a new window.

Figure 3–20 Layout Tab: Bullet

Column Layout

Select the columns to display in the bullet list.

<table>
<thead>
<tr>
<th>Name</th>
<th>Column</th>
<th>Display As</th>
<th>Action</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field1</td>
<td>employee_id</td>
<td>Text</td>
<td>&lt;None&gt;</td>
<td></td>
</tr>
<tr>
<td>Field2</td>
<td>Name</td>
<td>Text</td>
<td>&lt;None&gt;</td>
<td></td>
</tr>
<tr>
<td>Field3</td>
<td>Gender</td>
<td>Text</td>
<td>&lt;None&gt;</td>
<td></td>
</tr>
<tr>
<td>Field4</td>
<td>Job</td>
<td>Text</td>
<td>&lt;None&gt;</td>
<td></td>
</tr>
<tr>
<td>Field5</td>
<td>Email</td>
<td>Text</td>
<td>&lt;None&gt;</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3–21 Example of an OmniPortlet Using a Bullet Layout

- 20 Kathleen Bayyat F President - Manufacturing kbayyat@oracle.com
- 30 Robert Rodriguez M President - Sales rrodriguez@oracle.com
- 40 Edward Shields M Chief Financial Officer eshields@oracle.com
- 110 Jan Francois Stewart M Graphic Artist jfrancois@oracle.com
- 100 Lisa Williams F Graphic Artist lwilliams@oracle.com
- 430 Sandra Kyle F Course Developer skyle@oracle.com
- 770 Eaulo Yau F Customer Sales Representative eyau@oracle.com

Note: For more information on using the OmniPortlet Wizard, click the Help link in the upper right corner of the Layout tab.

3.2.5.5 Form Layout

You can use the form layout (Figure 3–22) if you have data you want to display as labels or default values in a form, such as Name: <name>. You can then use portlet parameters and events to pass data to the selected row.
You can also specify whether to display the target of a URL in a new window, as shown in Figure 3–23.

Figure 3–23  Open In New Window Check Box

<table>
<thead>
<tr>
<th>Action</th>
<th>URL</th>
<th>Open In New Window</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;None&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;None&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;None&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;None&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;None&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3–24 shows an OmniPortlet that uses a Form Layout.

Figure 3–24  Example of an OmniPortlet Using a Form Layout

<table>
<thead>
<tr>
<th>OmniPortlet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee ID 20</td>
</tr>
<tr>
<td>Name Kathleen Bayyat</td>
</tr>
<tr>
<td>Gender F</td>
</tr>
<tr>
<td>Job President - Manufacturing</td>
</tr>
<tr>
<td>Email <a href="mailto:kbayyat@oracle.com">kbayyat@oracle.com</a></td>
</tr>
</tbody>
</table>

| Employee ID 30 |
| Name Robert Rodriguez |
| Gender M |
| Job President - Sales |
| Email rrodrigue@oracle.com |

Note: For more information on using the OmniPortlet Wizard, click the Help link in the upper right corner of the Layout tab.
3.2.5.6 HTML Layout
You can use the HTML layout (Figure 3–25) to create a customized look and feel for your portlet by choosing from either a built-in HTML layout and modifying the code, or by creating a new layout from scratch. You can hand-code your own HTML or JavaScript based on data columns that OmniPortlet has retrieved based on the selected data source. By coding your own HTML and JavaScript, you have full control over the appearance and develop a rich interface for your portlet. For an example of using JavaScript in the HTML layout, choose the Sortable Table layout from the Quick Start list on this tab.

Figure 3–25  Layout Tab: HTML

Quick Start
Use the Quick Start to populate the sections below with starting HTML code.

Tip Clicking the Apply button permanently deletes any content currently in the below sections and replaces it with the template.

Non-Repeating Heading Section
Insert data label or system variables into text area: currency

Repeating Section
Insert data column or system variable into text area: currency

Non-Repeating Footer Section
Insert data label or system variable into text area: currency

Tip HTML code entered in the above section displays once for each data record.

Figure 3–26 shows an OmniPortlet that uses the HTML Layout.

Figure 3–26 Example of an OmniPortlet Using the HTML Layout

Currency Exchange

1 USD = 0.833719 EUR
3.2.6 Edit Defaults mode

After you have created your OmniPortlet and returned to your portal page, you can click the Edit Defaults icon to change the portlet options if required. You will notice that, in the Edit Defaults mode, there are tabs that correspond to the different steps in the OmniPortlet Wizard (except for the Type step) to directly access the different options. There is also one extra tab, the Events tab, which is explained in the next section.

When you edit an OmniPortlet using the Edit Defaults mode, keep in mind the following notes:

- A new mode, “none,” is the default setting for the Locale Personalization Level of OmniPortlet and the Simple Parameter form. This mode indicates that, when you edit the portlet defaults using the Edit Defaults mode, the changes apply to all users, regardless of the current OracleAS Portal session language and the locale of your browser. For more information about these settings, refer to Appendix I “Setting the LocalePersonalizationLevel” in the Oracle Application Server Portal Configuration Guide on the OracleAS Portal Documentation page on OTN.

- You can personalize the portlet at runtime by clicking the Personalize link on the portlet. When you personalize the portlet, a complete copy of the personalization object is created. Since all properties are duplicated, subsequently modifying the portlet through Edit Defaults will not be reflected in the personalized version of the portlet. To ensure the latest changes are made to the portlet, you must click Personalize again (after the modifications from the Edit Defaults wizard are made), then select the Reset to Defaults option.

- By default, the OmniPortlet provider uses the file-based Preference Store to store the personalization object, which stores the object in a file system in the middle-tier. If you decide to deploy OmniPortlet in a multiple middle-tier environment, you must use a shared Preference Store, such as the database Preference Store (DBPreferenceStore). To do so, you can choose to do one of the following:
  - Use a file-based Preference Store now, then migrate to the database Preference Store later using the PDK Preference Store Migration Utility.
  - Configure OmniPortlet to use the DBPreferenceStore, and follow the steps in Section “5.3.6 Step 6: Configure Portal Tools and Web Providers (Optional)” of the Oracle Application Server Portal Configuration Guide.

3.2.7 Events

On the Events tab (Figure 3–27) in the Edit Defaults mode of the OmniPortlet Wizard, you can identify event parameters based on the portlet parameters you selected on the Source tab.
3.3 Parameters and Events

Out of the box, OmniPortlet can receive up to five parameters and raise up to three events. Each of the events can send one or more parameters. For example, you can set up a chart that displays the employees in a department. When the user clicks one piece of the chart (for example, a department name), an event is raised that sends a parameter to the page. The page may then pass a parameter to all the portlets on that page that display information about the employees. Then, all the portlets on the page display information about the employees in the selected department.

**Note:** To learn how to use parameters and events with OmniPortlet, follow the steps in Chapter 4, "Building Example Portlets with OmniPortlet". If you are comfortable with the provider.xml file, you can add more parameters and events by editing the file.

To set up parameters and events, you must first enable the page group to accept parameters and events. In Oracle Application Server 10g, parameters and events are enabled by default. Then, you set up each portlet to accept the necessary parameters, and raise the required events. After you've set up the portlet parameters, you can link the portlets together by setting up the page-level parameters and events.

### 3.3.1 Portlet Parameters and Events

Out of the box, you can define up to five portlet parameters (Figure 3–28) for an OmniPortlet. You can define them on the Source tab of the wizard while you do the following:

- Define the OmniPortlet
- Select Edit Defaults for the OmniPortlet
Parameter values determine what data is displayed in the portlet. You can also use a parameter to pass a value in a URL or to embed a value in the portlet text.

**Note:** You can learn more about portlet parameters in the online Help, which you can access by clicking the Help link on the Source tab in the OmniPortlet Wizard. The online Help describes portlet parameters in detail, and how to set them up for your OmniPortlet. You can also refer to the Oracle Application Server Portal User’s Guide.

You can set up each OmniPortlet to raise up to three events. Each event can pass up to three parameters. Each parameter can be a portlet parameter, such as Param1, or a data source column, such as Department_No. You set up events on the Events tab (Figure 3–29) in the Edit Defaults mode of OmniPortlet.

### Figure 3–29 Events Tab

<table>
<thead>
<tr>
<th>Event Outputs</th>
<th>Event Outputs can be passed to a target page as page parameters.</th>
</tr>
</thead>
<tbody>
<tr>
<td>When Event 1 is raised, pass:</td>
<td>When Event 1 is raised, pass:</td>
</tr>
<tr>
<td>Event1Param1</td>
<td>DeptNo</td>
</tr>
<tr>
<td>Event1Param2</td>
<td>Department</td>
</tr>
<tr>
<td>Event1Param3</td>
<td>&lt;Null&gt;</td>
</tr>
<tr>
<td>When Event 2 is raised, pass:</td>
<td>When Event 2 is raised, pass:</td>
</tr>
<tr>
<td>Event2Param1</td>
<td>&lt;Null&gt;</td>
</tr>
<tr>
<td>Event2Param2</td>
<td>&lt;Null&gt;</td>
</tr>
<tr>
<td>Event2Param3</td>
<td>&lt;Null&gt;</td>
</tr>
<tr>
<td>When Event 3 is raised, pass:</td>
<td>When Event 3 is raised, pass:</td>
</tr>
<tr>
<td>Event3Param1</td>
<td>&lt;Null&gt;</td>
</tr>
<tr>
<td>Event3Param2</td>
<td>&lt;Null&gt;</td>
</tr>
<tr>
<td>Event3Param3</td>
<td>&lt;Null&gt;</td>
</tr>
</tbody>
</table>

### 3.3.2 Page Parameters and Events

After you’ve set up the parameters and events for each OmniPortlet on a portal page, you can map the portlet parameters and events to other portlets on the same page. For more information on using page parameters and events, refer to the OracleAS Portal online Help and the Oracle Application Server Portal User’s Guide.

### 3.4 Summary

In this chapter, you learned about OmniPortlet and its capabilities. You also learned about using parameters and events to integrate portlets on a page and create a portal application.

You can find more information about using the various tools in OmniPortlet by clicking the Help link on each of the pages in the wizard. For information on using
OmniPortlet, refer to Chapter 4, "Building Example Portlets with OmniPortlet". For more information on using Web Clipping, refer to Chapter 5, "Creating Content-Based Portlets with Web Clipping".
This chapter shows you how to use OmniPortlet to create four portlets based on different data sources: a Web service, a spreadsheet (CSV), XML, and an existing Web page. You will build these portlets based on the various built-in layouts, and will create a separate portlet based on an HTML layout. You will learn how to create and modify these portlets, as well as use page parameters and events to add interactivity to your portal page.

This chapter includes the following sections:

- Adding an OmniPortlet Instance to a Portal Page
- Building an OmniPortlet Based on a Web Service
- Building an OmniPortlet Based on a Spreadsheet (CSV)
- Building an OmniPortlet Based on an XML Data Source
- Building an OmniPortlet Based on a Web Page Data Source
- Setting Up Portlet Parameters and Events
- Building an OmniPortlet Using the HTML Layout

---

**Note:** To learn more about specific pages and tabs in OmniPortlet, click the Help link in the top right corner of the wizard. The online Help describes the contents of the selected page or tab. You can also find more information about using OmniPortlet in Chapter 3, "Creating Portlets with OmniPortlet", or about registering and configuring OmniPortlet with Oracle Application Server Portal in Appendix I, "Configuring the Portal Tools Providers" in the Oracle Application Server Portal Configuration Guide.

---

At the end of this chapter, you will create a page that contains four portlets, as shown in Figure 4–1.
Figure 4–1  Portal Page with Four OmniPortlet Examples

All four of these example portlets require you to be able to connect to the Internet. If you must use a proxy server to connect to the Internet, you will need to configure the HTTP proxy settings on the OmniPortlet Provider Test page to use proxy authentication. If the OmniPortlet provider has been set up to use proxy authentication that requires your login, you can enter your user information in the Proxy Authentication section of the Source tab in the OmniPortlet wizard. For more information on proxy authentication, refer to Section 3.2.2, "Source". For specific information on configuring the provider's proxy settings, refer to the Oracle Application Server Portal Configuration Guide, available on the OracleAS Portal Documentation page on OTN.

OmniPortlet is available with OracleAS Portal. However, if you installed the Oracle Application Server Portal Developer Kit separately, instructions on registering and configuring OmniPortlet with Oracle Application Server Portal are located in Appendix I, "Configuring the Portal Tools Providers" in the Oracle Application Server Portal Configuration Guide.

Note: The steps in this chapter assume that you are using the OmniPortlet provider that is available with OracleAS Portal. If you installed the Oracle Application Server Portal Developer Kit separately, you may need to slightly modify the instructions when adding an OmniPortlet instance to the page.
4.1 Adding an OmniPortlet Instance to a Portal Page

In this section, you will learn how to add an OmniPortlet instance to your portal page.

To add an OmniPortlet instance to a page:

1. In the Edit mode of the page where you want to add the OmniPortlet, click the Add Portlets icon.
2. On the Add Portlets page, in the Available Portlets list, click the Portlet Builders link.
3. Click the OmniPortlet link. You will see OmniPortlet listed in the Selected Portlets list.
4. Click OK. The new instance of OmniPortlet now displays on your portal page, as shown in Figure 4–2.

Figure 4–2 OmniPortlet Instance on a Portal Page

4.2 Building an OmniPortlet Based on a Web Service

The steps in this section will show you how to create a portlet that displays weather forecast information for a particular zip code. You will base this portlet on the Web Service available on the Oracle Technology Network. You will need Internet access to be able to complete this section.

To create an OmniPortlet based on a Web Service:

1. In the new OmniPortlet instance on your portal page, click the Define link to launch the OmniPortlet Wizard.
2. On the Type page, select the Web Service radio button, then click Next.
3. On the Source page, in the WSDL URL field, enter the following URL:

**Note:** This Web Service has one method (WeatherWS.giveMeSomeWeatherInfo) and accepts one parameter (param0). If you use a method that has parameters, the parameters will display in this section of the tab. You can enter a sample value for the parameter, then click Test to view the sample XML data, the SOAP response, and the SOAP Request.

4. Click Show Methods.
5. In the Web Service Methods section, in the param0 parameter field, enter a sample Zip code (for example, 94065), then click Test. The Web Service: Test Result window displays, where you can verify the XML data returned by the Web Service, as shown in Figure 4–3.
Building an OmniPortlet Based on a Web Service

**Figure 4–3  OmniPortlet: Web Service Test Results Page**

**Web Service: Test Result**

**XML Data**

This is the resulting XML data, extracted from the SOAP response body, which will be used to render the portlet. If the extracted XML Data is not of `<ROWSET>` format, you need to apply an XSL filter to convert it to this format.

```xml
  <item xsi:type="xsd:single">
    <param1>Value1</param1>
  </item>
</rowset>
```

6. Close the window.

The Source page should look like the image shown in **Figure 4–4**.

**Figure 4–4  OmniPortlet: Web Service Source Tab**

**Note:** If you do not have access to the Internet, you can use the another Web Service, but keep in mind that your results will not match the example in this chapter.

7. Click Next.

8. On the Filter page, click Next.


10. In the Header Text field, enter Forecast per Zip Code.

11. Make sure the Show Header Text check box is selected, and clear the Show Footer Text check box, as shown in **Figure 4–5**.
12. Make sure the Tabular radio button is selected, then click Next.

13. On the Layout page, select the Plain radio button.

14. Specify the Column Label, Column, and Display As properties for your data according to Table 4–1:

<table>
<thead>
<tr>
<th>Column Label</th>
<th>Column</th>
<th>Display As</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day</td>
<td>dayOfWeek</td>
<td>Text</td>
</tr>
<tr>
<td>High</td>
<td>hiTemp</td>
<td>Text</td>
</tr>
<tr>
<td>Low</td>
<td>lowTemp</td>
<td>Text</td>
</tr>
<tr>
<td>Precipitation</td>
<td>precip</td>
<td>Text</td>
</tr>
<tr>
<td>(Blank)</td>
<td>img</td>
<td>Image</td>
</tr>
</tbody>
</table>

15. Now that you’ve completed defining the portlet, click Finish. Your portlet should look like Figure 4–6.

4.3 Building an OmniPortlet Based on a Spreadsheet (CSV)

The steps in this section show you how to use OmniPortlet to define a portlet that displays regional information based on a spreadsheet (CSV) data source. This portlet will display the population of major U.S. urban areas in a pie chart.
To create an OmniPortlet based on a spreadsheet:

1. Create a region below the existing region on your portal page.
2. Add an OmniPortlet to this new region.
3. Launch the OmniPortlet Wizard by clicking the Define link.
4. On the Type page, select the Spreadsheet radio button, then click Next.
5. On the Source page, in the CSV URL field, replace the existing text with the following CSV:

   http://webservices.oracle.com/WeatherWS/city_population.csv

   **Note:** If you do not have access to the Internet, you can use the default URL, but keep in mind that your results will not match the example in this chapter.

6. Click Next.
7. On the Filter page, click Next.
9. Clear the Show Header Text check box.
10. In the Footer Text field, enter Click a pie section to view the population of the specified urban area.

   **Note:** The text you just entered in the Footer Text field instructs your end users to click a pie section to view more details about the selected population. To enable this feature, you will need to complete the steps in Section 4.6, "Setting Up Portlet Parameters and Events".

11. Ensure that the Show Footer Text check box is selected.
12. Under Layout Style, select the Chart radio button, then click Next.
13. On the Layout page, select the Pie radio button.
14. In the Width field, enter 300.
15. In the Height field, enter 300.
16. From the Legend list, choose Top.
17. Select the 3D Effect check box.
18. Under Column Layout, from the Group list, choose <None>.
19. From the Category list, choose City.
20. From the Value list, choose Population. The Layout tab should now look like the image shown in Figure 4–7.
21. Click Finish.

Your portlet now displays on your portal page below the Weather Forecast portlet, and should look like Figure 4–8.
4.4 Building an OmniPortlet Based on an XML Data Source

The steps in this section will show you how to use OmniPortlet to define a portlet that displays news information in a scrolling layout, based on an XML data source.

To create an OmniPortlet based on an XML data source:

1. Create a region next to the Web Services portlet (Weather Forecast) on your portal page.
2. Add an OmniPortlet to this new region.
3. Launch the OmniPortlet Wizard by clicking the Define link.
4. On the Type page, select the XML radio button, then click Next.
5. On the Source page, in the XML URL field, enter the following URL:
   
   http://www.nytimes.com/services/xml/rss/nyt/Travel.xml

   **Note:** If you do not have access to the Internet, you can use a different RSS feed, but keep in mind that your results will not match the images in this example.

6. In the XSL Filter URL field, enter the following URL for the XSL file:

   http://www.oracle.com/technology/products/ias/portal/viewlets/omniportletnews.xsl

   **Note:** This XSL filter transforms the RSS news feed into the ROWSET/ROW structure that OmniPortlet can consume.
7. Click Next.
8. On the Filter page, click Next.
9. On the View page, in the Title field, enter Travel News.
10. Clear the Show Header Text check box.
12. Ensure that the Show Footer Text check box is selected.
13. Under Layout Style, select the News radio button, then click Next.
14. On the Layout page, under News Style, select the Scrolling radio button.
15. In the Width field, enter 475.
16. In the Height field, enter 50.

The News Style section of the Layout tab should now look like the image shown in Figure 4–9.

![Figure 4–9 OmniPortlet: XML Layout Tab - News Style](image)

17. Under Column Layout, next to Field 1 choose title from the Column list, Hyperlink from the Action list, and enter ##link## in the URL field, as shown in Figure 4–10.

![Figure 4–10 OmniPortlet: News Style Column Layout](image)

18. Next to Field2, choose description from the Column list and leave the default values for the other settings. Ensure this section of the Layout tab looks like Figure 4–11.

![Figure 4–11 Column Layout Section of the Layout Tab](image)
19. Click Finish.

Your portlet now displays on your portal page, as shown in Figure 4–12.

**Figure 4–12 OmniPortlet: XML Scrolling News OmniPortlet on the Page**

<table>
<thead>
<tr>
<th>Travel News</th>
<th>Customize</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upscale Flavors on Chicago's North Side</td>
<td></td>
</tr>
<tr>
<td>Restaurants with laid-back style arrive on the North Side of Chicago.</td>
<td></td>
</tr>
<tr>
<td>Source: The New York Times</td>
<td></td>
</tr>
</tbody>
</table>

### 4.5 Building an OmniPortlet Based on a Web Page Data Source

The steps in this section will show you how to use OmniPortlet to create a portlet that displays weather information in a text format based on an existing Web Page. In this section, you will use OmniPortlet to clip and scrape content from Web sites.

To create an OmniPortlet based on a Web page data source:

1. Create a region on your portal page.
2. Add an OmniPortlet to this new region.
3. Launch the OmniPortlet Wizard by clicking the Define link.
4. On the Type tab of the OmniPortlet Wizard, select the Web Page radio button, then click Next.
5. On the Source tab, click Select Web Page.
6. On the Web Clipping Studio page that displays, in the URL Location field, enter the URL of the page you want to clip, as shown in Figure 4–13:

   ```url
   http://www.wunderground.com
   ```

   **Note:** You can use a different Web page, but keep in mind that your results will not match the example in this chapter.

   **Figure 4–13 URL Location Field**

   ![URL Location Field](image)

   **Note:** In this example, we use a third party Web site owned by The Weather Underground, Inc. Because this Web site is continually updated based on current weather forecasts, the images and steps included in this section may not reflect exactly what you see when you create this example. As you go further into this example, some of the steps may not work, as the owners may change the technology of this third party Web site.

7. Click Start to display the Web Clipping Studio. You should see the Web page display in the Web Clipping Studio, as shown in Figure 4–14.
8. On the home page, enter a zip code (for example, 94065) in the Weather field, as shown in Figure 4–15, then click the magnifying glass icon next to the field.

Figure 4–15 Entering the Zip Code 94065 into the Weather Field

9. The weather information for Redwood City, California displays in the Web Clipping Studio. A segment of the page is shown in Figure 4–16.

Figure 4–16 Weather Information for Redwood City, California
10. In the top right corner of the Web Clipping Studio, click the **Section** button, as shown in **Figure 4–17**.

**Figure 4–17  The Section Button**

![Section Button](image.png)

11. After you click the Section button, you’ll notice that the elements of the Web page are broken down, and that a new icon displays called Choose at the top of each section. **Figure 4–18** shows a snapshot of the Web Clipping Studio.

**Figure 4–18  Web Clipping Studio Displaying the Web Page Sections**

![Web Clipping Studio](image.png)

12. Find the section shown in **Figure 4–19**, which shows the weather information for the city you chose (in this example, San Mateo County).
Figure 4–19  Weather Information for San Mateo County with the Choose Icon

Click the Choose icon for this section, located directly above the "Forecast for San Mateo County" title bar, as shown in Figure 4–19. The section displays in the Web Clipping Studio, as shown in Figure 4–20.

Figure 4–20 Weather Information Section for San Mateo County in the Web Clipping Studio

13. After you have chosen the clipping, you can refine your data further by scraping the data, that is, selecting specific cells you wish to display in your portlet. Click the Scrape button (Figure 4–21).

Figure 4–21 The Scrape Button
15. While in Scraping mode, you can identify the text pieces in the Web clipping by selecting the check boxes next to each item. You can repeat these items at the column level, row level, or table level.

In this example, we want to show the title and the description of each resulting article from our search. We can repeat the title and description at the row level, so that each result returned by the search displays only the title and the description of every result. In general, you choose the text items in the first row that contains all the pieces you wish to repeat for each row.

After you click the Scrape button, you'll notice that check boxes display next to each item on the Web page, as shown in Figure 4–22.

Figure 4–22  Scraping Check Boxes

16. Select the output you want by selecting the check boxes next to the items. Figure 4–23 shows an example of a check box.

Figure 4–23  Check Box for San Mateo County Title

Note: Depending on the color of the section you want to select, the check boxes may not be prominently displayed.

17. After you select the check box for "Forecast for San Mateo County," notice that a corresponding label displays in the Data section of the Web Clipping Studio at the bottom of the screen, as shown in Figure 4–24.

Figure 4–24  San Mateo County Label in Data Section

18. In the Name field, enter a more meaningful name, such as "SanMateoCountyForecast." Do not include spaces in the name.

19. Select the fields you wish to display and add the corresponding labels as you select them. The following is a list of sample information we chose for this example, but you can choose your own examples depending on the current weather. You can see the selected weather information in Figure 4–25.

- Forecast for San Mateo County title
- Today
- Today’s Weather
- Thursday
- Thursday’s Weather

**Figure 4–25  Selected Weather Information**

<table>
<thead>
<tr>
<th>Data</th>
<th>Use to identify a cell of data. Then use the &quot;More&quot; and &quot;Less&quot; buttons to expand your selection to include more or less data within the clipping. When clipping toolbar.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>VALUE</td>
</tr>
<tr>
<td>- SanMateoCountyForecast</td>
<td>Forecast for San Mateo County</td>
</tr>
<tr>
<td>- ThursdayWeather</td>
<td>Partly to mostly cloudy with a chance of rain. Highs in the 60s to upper 70s. Northwest winds 5 to 15 mph increasing to 15 to 25 mph in the afternoon.</td>
</tr>
<tr>
<td>- TodayWeather</td>
<td>Occasional light rain. Highs in the 60s. South winds 10 to 25 mph with gusts to 35 mph at the coast.</td>
</tr>
<tr>
<td>- Today</td>
<td>Today’s Weather</td>
</tr>
<tr>
<td>- Thursday</td>
<td>Thursday’s Weather</td>
</tr>
</tbody>
</table>

**Note:** With the Web page data source, you can select URLs as part of your Web clipping. In Oracle Application Server Portal 10.1.2 and later, the context of the application is maintained. So, for example, any images that display on the hyperlinked page will be maintained.

20. Now that you’ve selected the data you want to display, click the Continue button, as shown in **Figure 4–26**.

**Figure 4–26  The Continue Button**

[Icons: Home, Close, Checkmark, Question Mark]

Cancel  Unselect  Continue  Help

21. On the page that displays, verify that the information in the Clipping Attributes section includes the title: "Weather Underground: Redwood City, California Forecast," as shown in **Figure 4–27**.
22. Verify that the information in the Clipping Parameters section includes the parameters as shown in Figure 4–28.

23. Select the Clipping Parameter? check box for the second parameter with the name "query."

24. Click OK.

25. On the Source tab of the OmniPortlet Wizard, the new title and description now display. To edit the Web clipping in the Web Clipping Studio, you can click the Select Web Page button again, as shown in Figure 4–29.

26. Under the Clipping Parameters heading, you should see the clipping parameter you set on the previous page, as shown in Figure 4–30:

27. Under the Portlet Parameters heading, next to Param1, set the Default Value to the zip code, 94065, as shown in Figure 4–31.
28. Click Next.
29. On the Filter tab, click Next.
30. On the View tab, in the Title field, enter Weather Information.
31. In the Footer Text field, enter Source: Weather Underground and make sure the Show Footer Text check box is selected.
32. Under Layout Style, select the News radio button. The View tab should look like Figure 4–32.

Figure 4–32 View Tab with the Options Selected

33. Click Next.
34. Verify that the Layout tab looks like Figure 4–33.

Figure 4–33 Layout Tab

35. Click Finish.
36. Your new Web Page portlet displays on the portal page, and should look like Figure 4–34.
Figure 4–34  Weather Information Portlet on the Portal Page

Now that you have completed building all four example OmniPortlets, your page should look like Figure 4–35.

4.6 Setting Up Portlet Parameters and Events

The steps in this section show you how to set up the portlets and page you created to use parameters. Then, when a user clicks a slice of the pie chart (generating a portlet event) that corresponds to a region (for example, New York), the Weather Forecast per Zip Code portlet (based on a Web Service) and the Weather Information portlet (based on a Web page) will display the corresponding weather information for that region (New York).

To set up portlet parameters and events, you need to do the following:

■ Configure Portlets to Accept Parameters
■ Map the Page Parameter to the Portlet Parameters
4.6.1 Configure Portlets to Accept Parameters

The steps in this section will show you how to configure the two portlets on your page that accept parameters (the Web Service portlet that displays Weather Forecast per Zip Code information and the Web page portlet that displays Weather Information).

To configure the two portlets to accept parameters:

1. In Edit mode of the page, click the Edit Defaults icon in the top left corner of the Web Services portlet, as shown in Figure 4–36.

2. On the Source tab, replace the value of \texttt{param0}\ (94065) with \texttt{##Param1##}, as shown in Figure 4–37.

3. Under Portlet Parameters, set the default value of Param1 to 94065.

4. In the Personalize Page Label field, enter Zip.

5. In the Personalize Page Description field, enter Enter zip code, as shown in Figure 4–38.

6. Click OK.

7. On the portal page, in Edit mode, click the Edit Defaults icon for the Web page portlet, as shown in Figure 4–39.
8. On the Source tab, since you have already mapped a portlet parameter to the clipping parameter, you can simply add a label and description to the portlet parameter Param1, as shown in Figure 4–40.

![Figure 4–39](image) **Edit Defaults Icon for the Web Page Portlet**

![Weather Information](image)

Forecast for San Mateo County
Mostly sunny. Highs from

9. Click **OK**.

You have created two portlet parameters in the Web Services and Web page portlets to accept page parameters. Next, you will map a page parameter to these two portlet parameters.

### 4.6.2 Map the Page Parameter to the Portlet Parameters

The steps in this section will show you how to map the page parameters to the two portlets you configured in the previous section.

To map the page parameter to the portlet parameters:

1. On the page, in Edit mode, click the Page: **Properties** link at the top of the screen, as shown in Figure 4–41.

![Figure 4–41](image) **Page: Properties Link**

2. Click the **Parameters** tab.

3. Under New Page Parameter, in the **Parameter Name** field, enter **zip**, then click **Add**, as shown in Figure 4–42.

![Figure 4–42](image) **New Page Parameter**
4. Under Page Parameter Properties, the new page parameter displays. In the **Default Value** field, enter 94065, as shown in *Figure 4–43*.

*Figure 4–43 Default Value for the “zip” Page Parameter*

<table>
<thead>
<tr>
<th>Name</th>
<th>Display Name</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>zip</td>
<td>zip</td>
<td>94065</td>
</tr>
</tbody>
</table>

5. Under Portlet Parameter Values, you will see a list of portlets listed. If you have followed all the steps in this chapter, you will see four instances of OmniPortlet listed. The Web Services portlet is the first in the list, and the Web Page data source is the fourth (or the bottom) instance of OmniPortlet in the list.

6. Click the arrow next to the first instance of OmniPortlet to expand the Portlet Parameters list, as shown in *Figure 4–44*.

*Figure 4–44 Portlet Parameter Values Section*

7. Next to Param1, from the list, choose **Page Parameter**.

8. From the list that displays, ensure that **zip** is selected.

9. For the fourth OmniPortlet in the list, follow the same steps to set **Param1** to the Page Parameter of zip, as shown in *Figure 4–45*. 
4.6.3 Configure the Chart Portlet to Use Events

The steps in this section will show you how to configure the chart portlet to use events. That is, based on an event in the portlet (such as clicking a slice in a pie chart), an event will occur. In this case, you will configure the portlet so that when a slice is clicked by an end user, the zip code associated with that slice will be sent to the page. Then, the data in the two portlets you configured in Section 4.6.1, “Configure Portlets to Accept Parameters” will refresh depending on the selected zip code.

To configure the chart portlet:

1. In the Edit mode of the page, click the **Edit Defaults** icon for the chart portlet, as shown in Figure 4–46.

2. Click the **Layout** tab.
3. Under Chart Drilldown, choose **Event1** from the Action list.

4. Notice that in Edit Defaults mode, a new tab displays in the wizard called Events. Click the **Events** tab.

5. On this tab, you will configure Event1 (which you set on the Layout tab) to pass the zip code from the chart to the page as the event output.

   Set Event1Param1 to **zipcode**, as shown in Figure 4-47, then click **OK**.

**Figure 4–47 Events Tab for the Chart Portlet**

When Event1 is raised, pass:

<table>
<thead>
<tr>
<th>Event Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event1Param1</td>
<td>zipcode</td>
</tr>
<tr>
<td>Event1Param2</td>
<td>&lt;Null&gt;</td>
</tr>
<tr>
<td>Event1Param3</td>
<td>&lt;Null&gt;</td>
</tr>
</tbody>
</table>

You have configured the chart portlet so that a user can click a slice of the pie chart, and set up an event so that the zip code selected in the pie chart will be sent to the page. Next, you will set up the page to accept this event parameter.

### 4.6.4 Map the Chart Event to the Page

The steps in this section will show you how to map the chart event you created in the previous section to the page, so that when a user clicks on a slice of the pie chart, the zip code selected in the pie chart will be accepted by the page as the page input. The page will then display the data that corresponds to the selected zip code in the Web Service and Web page portlets.

To map the chart event to the page:

1. On the page, in Edit mode, click the Page: **Properties** link, then click the **Events** tab.

2. Expand the second OmniPortlet in the list to display the events, and select **Event1**, as shown in Figure 4-48.

**Figure 4–48 Portlet Events Section of the Page Events Tab**

3. Select the **Go to page** radio button, then, next to the field, click the **Browse Pages** icon to search for the name of your page (in this case, **OmniPortlet Examples**). Next to your page name, click **Return Object**.
4. Set the Page Input as shown in Figure 4–49.

**Figure 4–49  Page Input on the Events Tab**

5. Click OK.

6. Now, when you drag your mouse over the pie chart in the Chart portlet, you should notice that you can click one of the sections. Try clicking on the largest slice (New York). You will notice that the page refreshes.

   In the URL of the page, you should see a parameter value set after the page name, for example: `OmniPortlet%20Examples?zip=10001`. The Weather Forecast information changes, and looks something like Figure 4–50.

**Figure 4–50  Weather Forecast (Web Service) Portlet for New York**

   The Weather Information portlet changes, and looks something like Figure 4–51.
4.7 Building an OmniPortlet Using the HTML Layout

As you can see from the examples in this chapter, you can use any data source with any of the built-in layouts in OmniPortlet. This section will show you how to use the same Web Service data source you used in Section 4.2, "Building an OmniPortlet Based on a Web Service" to create a portlet using the HTML layout in OmniPortlet. At the end of this section, your portlet will look like Figure 4–52:

To create a portlet using the HTML layout:

1. Follow steps 1-8 in Section 4.2, "Building an OmniPortlet Based on a Web Service" to use the Web Service data source.

2. On the View tab, under Layout Style, select HTML, then click Next.

3. On the Layout tab, the default layout is a Simple Table.
   
   You can switch the layout using the Quick Start drop-down list, or clear the fields to create your own layout. For this example, use the default Simple Table layout. In the Repeating Section, you can see how OmniPortlet automatically populates the columns from the data source into your layout. You can modify the HTML in any of the sections as you wish.

4. Click Finish. Your portlet looks like Figure 4–53:
5. You’ll notice that under “img,” the URL displays instead of an actual image. To change the HTML source, click the Edit Defaults (pencil) icon to edit your OmniPortlet.

6. Click the Layout tab.

7. In the Repeating Section, look for the row with the code: `<TD>##img##</TD>`.

8. Replace this code with: `<TD><img src="##img##"></TD>`.

9. You can also change the column headers in the Non-Repeating Heading Section. For example, replace:

   `<TH CLASS='PortletHeading1'>img</TH>`

with

   `<TH CLASS='PortletHeading1'>Weather Graphic</TH>`

10. In the Non-Repeating Heading section, delete the line:

    `<TH CLASS='PortletHeading1'>Zip Code</TH>`

11. In the Repeating Section, delete the line:

    `<TD>##zipCode##</TD>`

12. Perform the same action for any other headings you wish to change, then click Finish. Your portlet now looks like Figure 4–54:

---

**Figure 4–54  Final View of an OmniPortlet Using the HTML Layout**

---

4.8 Summary

In this chapter, you learned how to use OmniPortlet to build various types of portlets on a page. You also learned how to use parameters and events to integrate portlets on
a page and create a portal application. Finally, you learned how to use the HTML layout to modify the appearance of a portlet based on a Web Service data source.

You can find more information about using the various tools in OmniPortlet by clicking the Help link on each of the pages in the wizard. To learn more about parameters and events, refer to the Oracle Application Server Portal User’s Guide available on OracleAS Portal Documentation page on OTN (http://www.oracle.com/technology/products/ias/portal/documentation.html)
Creating Content-Based Portlets with Web Clipping

Web Clipping is a browser-based declarative tool that enables you to integrate any Web application with OracleAS Portal. It is designed to give you quick integration by leveraging the Web application's existing user interface. Web Clipping has been implemented as a Web provider using the Oracle Application Server Portal Developer Kit, which is a component of OracleAS Portal.

With Web Clipping, you can collect Web content into portlets in a single centralized Web page. You can use Web Clipping to consolidate content from Web sites scattered throughout a large organization.

This chapter contains the following sections:

- What Is Web Clipping?
- Adding a Web Clipping Portlet to a Page
- Integrating Authenticated Web Content Using Single Sign-On
- Adding a Web Clipping That Users Can Personalize
- Migrating from URL-Based Portlets
- Current Limitations for Web Clipping

5.1 What Is Web Clipping?

Web Clipping allows clipping of an entire Web page or a portion of it and reusing it as a portlet. Basic and HTML-form-based sites may be clipped. Use Web Clipping when you want to copy content from an existing Web page and expose it in your portal as a portlet. The Web Clipping portlet supports the following:

- **Navigation through various styles of login mechanisms**, including form- and JavaScript-based submission and HTTP Basic and Digest Authentication with cookie-based session management.

- **Fuzzy matching of clippings**, meaning that if a Web clipping gets reordered within the source page or if its character font, size, or style changes, it will still be identified correctly by the Web Clipping engine and delivered as the portlet content.

- **Reuse of a wide range of Web content**, including basic support of pages written with HTML 4.0.1, JavaScript, applets, and plug-in enabled content, retrieved through HTTP GET and POST (form submission).
Adding Web Page Content to a Portal Page

- **Personalization**, enabling page designers to expose input parameters that page viewers can modify when they personalize the portlet. These parameters can be exposed as public parameters that a page designer can map as OracleAS Portal page parameters. This feature enables end users to obtain personalized clippings.

- **Integrated authenticated Web content through Single Sign-On**, including integration with external applications, which enables you to leverage Oracle Application Server Single Sign-On and to clip content from authenticated external Web sites.

- **Inline rendering**, enabling you to set up Web Clipping portlets to display links within the context of the portlet. As a result, when a user clicks a link in the Web Clipping portlet, the results display within the same portlet. You can use this feature with internal and external Web sites.

- **Proxy authentication**, including support for global proxy authentication and per-user authentication. You can specify proxy server authentication details including type (Basic or Digest) and realm, through the Web Clipping Provider Test page. In addition, you can specify one of the following schemes for entering user credentials.
  - All users automatically log in using a user name and password you provide.
  - All users will need to log in using a user name and password they provide.
  - All public users (not authenticated into Portal) automatically log in using a user name and password you provide, while valid users (authenticated into Portal) will need to log in using a user name and password they provide.

  See the Oracle Application Server Portal Configuration Guide for more information.

- **Migration from URL-based portlets**, enabling you to migrate your URL-based portlets to Web Clipping. See Section 5.5, "Migrating from URL-Based Portlets" for more information.

- **Navigation and clipping of HTTPS-based external Web sites**, if appropriate server certificates are acquired.

- **Clipping of page content from HTML 4.0.1 pages**, which includes the following:
  - Clipping of <applet>, <body>, <div>, <embed>, <img>, <object>, <ol>, <span>, <table>, and <ul> tagged content
  - Preservation of <head> styles and fonts, and Cascading Style Sheets (CSS)
  - UTF-8 compliant character sets
  - Navigation through hyperlinks (HTTP GET), form submissions (HTTP POST), frames, and URL redirection

- **National Language Sets (NLS)** in URLs and URL parameters. See Section 5.6, "Current Limitations for Web Clipping" for information about how Web Clipping determines the character set of clipped content.

By default, all Web clipping definitions are stored persistently in the Oracle Application Server infrastructure database. Any secure information, such as passwords, are stored in encrypted form, according to the Data Encryption Standard (DES), using Oracle encryption technology.

### 5.2 Adding Web Page Content to a Portal Page

To add Web page content to a portal page, follow the steps described in the following sections:
1. Section 5.2.1, "Adding a Web Clipping Portlet to a Page"
2. Section 5.2.2, "Selecting a Section of a Web Page to Display in the Web Clipping Portlet"
3. Section 5.2.3, "Setting Web Clipping Portlet Properties"

5.2.1 Adding a Web Clipping Portlet to a Page

To add a Web Clipping portlet to an OracleAS Portal page, perform the following:

1. Navigate to the Page Groups portlet. By default, the Page Groups portlet is located on the Build tab of the Portal Builder page.
2. In the Layout & Appearance section, for Pages, click Browse.
   Figure 5–1 shows the Page Groups portlet.

   **Figure 5–1 Viewing the Page Group**

   ![Page Groups portlet](image)

<table>
<thead>
<tr>
<th>Work In</th>
<th>Sample</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Layout &amp; Appearance</th>
<th>Create</th>
<th>Browse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pages</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sample root page</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Styles</td>
<td>Create</td>
<td>Browse</td>
</tr>
<tr>
<td>Sample style</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Portal Templates</td>
<td>Create</td>
<td>Browse</td>
</tr>
<tr>
<td>There are no portal templates in this page group.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Navigation Pages</td>
<td>Create</td>
<td>Browse</td>
</tr>
<tr>
<td>Sample navbar Sample banner</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTML Templates</td>
<td>Create</td>
<td>Browse</td>
</tr>
<tr>
<td>There are no HTML templates in this page group.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. Figure 5–2 shows the Page Groups tab with the list of pages. For the page to which you want to add the Web Clipping portlet, click Edit in the Actions column.
4. The page is displayed. In the region in which you want to add the Web Clipping portlet, click the Add Portlet icon.
   Figure 5–3 shows a portion of the page.

5. In the Add Portlets page, navigate to the Web Clipping Portlet link and click it. The Web Clipping Portlet moves to the Selected Portlets box. By default, the Web Clipping portlet is located in the Portal Builder page of the Portlet Repository. If you cannot find this page, use the Search field to find the portlet.
6. Click OK to add a Web Clipping portlet to your page.
   Figure 5–4 shows the Web Clipping portlet added to your page.
5.2.2 Selecting a Section of a Web Page to Display in the Web Clipping Portlet

To select a section of a Web page to display in the Web Clipping portlet, you use the Web Clipping Studio. Using the Web Clipping Studio, you can perform the following tasks:

- Browse for Web content
- Section the chosen target page
- Choose the exact portion of the Web content to clip
- Preview the clipped content as a portlet
- Save the clipped content as a portlet
- Set portlet properties and save the updated portlet information

To select a section of a Web page to display in the Web Clipping portlet:

1. Above the Web Clipping portlet, click the Edit Defaults icon, as shown in Figure 5–5.

The Find a Web Clipping page is displayed.

2. In the URL Location field, enter the location of the starting Web page that links to the content you want to clip, as shown in Figure 5–6.
3. Click **Start**.

The Web Clipping Studio displays the page you specified, as shown in Figure 5–7.

Figure 5–7  Browsing to a Page Containing Content for a Web Clipping

---

Note that the URL in the browser bar changes from:

http://host:port/portal/page?_dad=portal&_schema=PORTAL...

To:

http://host:port/portalTools/webClipping...

4. Browse to the page that contains the content you want to clip.

As you click hyperlinks in the Web page, your navigation links are recorded.
5. Once you display the page that contains the content you want to clip, in the Web Clipping Studio banner, click Section, as shown in Figure 5–8.

**Figure 5–8  Sectioning the Target Web Page**

Sectioning divides the target Web page into its clippable sections, as shown in Figure 5–9. After you click Section, you are no longer able to browse links in the displayed page. If you want to continue navigation, click Unsection in the Web Clipping Studio banner.

**Figure 5–9  Sectioned Target Web Page**

6. At the top left of the section of the Web content you want to clip, click Choose. You can choose only one section as a clipping at a time.
7. Web Clipping Studio displays a preview of your chosen section. If it is the section you want, click **Select** in the Web Clipping Studio banner. The Web Clipping Studio displays the Find a Web Clipping page, with the properties of the clipping.

If you do not want to use the section you clipped in your portlet, click **Unselect** to return to the page containing the section. You can choose another section on the page, or click **Unsection** to navigate to another page.

Some sections may contain no data, only whitespace. For example, a Web page may contain an HTML `<DIV>` tag that contains no text or images. If you click **Choose** on a section that contains no data, Web Clipping displays a preview, but the preview correctly shows only whitespace. In this case, click **Unselect** in the preview page to return to the sectioned page. Then, select a section containing data.

8. In the Find a Web Clipping page, click **OK** to display the selected Web clipping in the Web Clipping portlet on your page. (You can edit default properties in the page. See Section 5.2.3, "Setting Web Clipping Portlet Properties" for more information.)

Figure 5–10 shows the content added to the Web Clipping portlet.

---

**Note:** To increase the number of sections available from which to choose, click **Section Smaller** in the Web Clipping Studio banner. For example, you would click **Section Smaller** to drill down one level of nested tables. To decrease the number of sections available from which to choose, click **Section Larger**.

---

Note that the **Refresh** link in the Web Clipping Portlet retrieves fresh data from the originating Web site. The **Portlet Refresh** link reloads the portlet, but it may retrieve the data from the cache, depending upon the settings for expiration.
5.2.3 Setting Web Clipping Portlet Properties

You can edit various portlet settings to change the appearance of the Web Clipping portlet and to specify how end users can interact with the portlet.

To set Web Clipping portlet properties:

1. At the top left of the Web Clipping portlet, click the Edit Defaults icon. Web Clipping Studio displays the Find a Web Clipping page with a Properties section, as shown in Figure 5–11.

Figure 5–11 Properties Section of Find a Web Clipping Page

Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL Rewriting</td>
<td>None</td>
</tr>
<tr>
<td>Title</td>
<td>Oracle Application Server</td>
</tr>
<tr>
<td>Description</td>
<td>New Base</td>
</tr>
<tr>
<td>Time Out (seconds)</td>
<td>60</td>
</tr>
<tr>
<td>Expires (minutes)</td>
<td>30</td>
</tr>
</tbody>
</table>

2. From the URL Rewriting list in the Properties section, choose Inline if you want link targets to be displayed inside the portlet, or choose None if you want link targets to replace the current Portal page in the browser.

3. In the Title field, enter a title to display in the portlet banner.

4. In the Description field, enter a description of the portlet.

5. In the Time Out (seconds) field, enter the amount of time (in seconds) for the Web Clipping provider to attempt to contact the Web page from which the content was clipped.

6. In the Expires (minutes) field, enter the amount of time (in minutes) that cached content is valid. Any requests for portlet content that occur within the time period you specify will be satisfied from the cache.

Once the cache period is exceeded, requests for portlet content will be satisfied by retrieving content from the portlet’s Web Clipping data source. The cache will also be refreshed with this content.

7. If you entered any information in a form while clipping content for the Web Clipping portlet, the Parameterize Inputs section is available. Select the Click to start parameterizing check box to customize parameters associated with the Web Clipping portlet content. Then:

   a. From the Parameters list, choose the parameters that you want to customize.

   b. From the Personalizable list, select a parameter if you want to allow end users to provide their own values for the parameters when they personalize the portlet. Select None if you do not want to allow this.

   c. In the Display Name field, enter a name to be displayed for the parameter.

   d. In the Default Value field, enter a value to use by default for the parameter.

Section 5.4.3, "Personalizing a Web Clipping Portlet" provides an example of personalizing parameters.

8. Click OK.
5.3 Integrating Authenticated Web Content Using Single Sign-On

This section walks you through an example that demonstrates how you can leverage OracleAS Single Sign-On to integrate content from external Web sites that require authentication into a Web Clipping portlet.

The example incorporates a secured page from Oracle MetaLink (an external application) into a Web Clipping portlet.

To integrate an external application:

1. Set up the external application in OracleAS Portal, specifying the authentication information. Refer to the Oracle Application Server Portal Configuration Guide for more detail.
   a. Log in to OracleAS Portal as a user who has SSO Administration privileges, for example, the orcladmin user.
   b. Navigate to the Administer External Applications portlet. (Click the Administer tab, then click the Portal subtab. In the SSO Server Administration section, which is in the middle column, select Administer External Applications.)
   c. Click Add External Application.
   d. In the Create External Application page, in the Application Name field, enter a name for the application, for example, Metalink.
   e. For Login URL, enter the URL to log in to the application, for example, http://metalink.oracle.com/metalink/plsql/sit_main.showSitemap?p_showTitle=0. To determine the URL, navigate to the desired application in a browser and note the URL.

   For Form-based Authorization, view the source of the login page for the external application and note the URL to be accessed during the login action.
   f. For User Name/ID Field Name, enter the field name that the external application uses for the user name. Determine the field name by viewing the source for the desired page. If the Authentication method uses Basic Authentication, you do not need to enter a field name. For Metalink, you do not need to enter anything in this field.
   g. For Password Field Name, enter the field name that the external application uses for the password. Determine the field name by viewing the source for the desired page. If the Authentication method uses Basic Authentication, you do not need to enter a field name. For Metalink, you do not need to enter anything in this field.
   h. Select Basic Authentication as the authentication method.

   Figure 5–12 shows a portion of the Create External Application page:
In the **Additional Fields** section, you can enter names and values of any additional fields that are submitted with the login form of the external application. To specify a field name that is used to indicate a redirection URL, enter `redirectFieldName` for **Field Name**. For this example, you do not need to enter additional fields. Figure 5–13 shows the Additional Fields section.

**j.** Click **OK**.

**k.** To test your credentials with Oracle Metalink, in the Administer External Applications page, click the name of the application you just created. Then, in the External Application Login page, log in to the application Oracle Metalink using your Oracle Metalink user name and password.
In the Administer External Applications page, click Close.

For more information about OracleAS Single Sign-On and external applications, see the Oracle Application Server Single Sign-On Administrator's Guide.

2. For the Web Clipping portlet, create a new Web Clipping provider:
   a. Click the Administer tab, then click the Portlets tab.
   b. Select Register a Provider.
   c. In the Register a Provider page, enter webClippingMetalink for the Name and webClipping Metalink for the Display Name. Enter values for Timeout and Timeout Message. Choose Web for the Implementation Style.
   d. Click Next.
   e. In the General Properties section of the Define Connection page, for the URL, enter:

```
http://host:port/portalTools/webClipping/providers/webClipping
```

Note that host:port refers to the host and port where the providers are located. This corresponds to the URL for OracleAS Portal.

f. For the user's identity, select The user's identity needs to be mapped to a different name in the Web provider's application, and the Web provider requires an external application login for establishment of a browser session. If selecting this option, specify the external application ID below.

g. For External Application ID, click the List of Values icon and select the external application you added.

Figure 5–14 shows the top part of the Define Connections page.

Figure 5–14 Specifying an External Application for a Web Clipping Provider

![Define Connection](image)

- [Previous] [Next] [Finish] [Cancel]

**General Properties**

Specify the URL for the Web provider and configure its communication settings.

- **URL**

```
/com/ussp?port=port&app=webClipping/providers/webClipping
```

- Web provider in same cookie domain as the portal

Enter the Service Id, if applicable. Multiple providers can now be accessed via the same URL. The Service Id identifies a specific provider that can be accessed via the specified URL.

- **Service Id**

(examples: "urn:ADAPTER_PROVIDER" or "urn:webProvider")

Specify how the user's identity will be set by the Portal when communicating with the Web provider:

- The user has the same identity in the Web provider as in the Single Sign-On identity.
- The user's identity needs to be mapped to a different name in the Web provider's application, and for the Web provider requires an external application login for establishment of a browser session. If selecting this option, specify the external application ID below.

- **External Application ID**

h. In the User/Session Information section, select User to send user specific information to the provider. For Login Frequency, select Once Per User Session.
i. Check that the proxy settings are correct. If you use a proxy server to contact the Web providers from the middle tier, enter the proxy server for **Middle Tier**.

   If you use a proxy server to contact the Web providers from the portal repository, enter the proxy server for **Portal Repository**.

   Usually, because the OracleAS Portal and Web Clipping URLs point to the same middle tier, this step is not necessary.

j. Click **Finish**.

k. In the Registration Confirmation page, if the registration was successful, click **OK**.

3. Add a portlet to a page, using the Web Clipping Metalink provider that you just created:

   a. In OracleAS Portal, navigate to the page in which you want to add the portlet.

   b. In the region in which you want to add the Web Clipping portlet, click the **Add Portlet** icon.

   c. In the Add Portlets to Region page, search for **Web Clipping Metalink**. It is located in the Portlet Staging Area. Click it to move it to the **Selected Portlets** box.

   d. Click **OK**.

   **Section 5.2.1, "Adding a Web Clipping Portlet to a Page"** describes in detail how to add a portlet.

4. If you have not entered your credentials for the External Application representing Metalink, the portlet will contain an **Update login information** link. Click the link and enter your credentials. Then, click **OK**.

5. Select a section of a page to display in the Web Clipping portlet:

   a. In the Web Clipping portlet, click the **Edit Defaults** icon.

      The Find a Web Clipping page is displayed.

   b. In the **URL Location** field, the default URL for the External Application is displayed. Change it to the location of the starting Web page that links to the content you want to clip. In this case, enter http://www.metalink.oracle.com.

   c. Click **Start**. The Web Clipping Studio displays the page you specified. Log in to Oracle Metalink.

   d. Browse to the page that contains the content you want to clip. After you display the page that contains the content you want to clip, click **Section** in the Web Clipping Studio banner. **Figure 5–15** shows the external application displayed in Web Clipping Studio.
Figure 5–15  External Application in Web Clipping Studio

e. At the top left of the section of the Web content you want to clip, click Choose.

f. Web Clipping Studio displays a preview of your chosen section. If it is the section you want, click Select in the Web Clipping Studio banner.

The Web Clipping Studio displays the Find a Web Clipping page, with the properties of the clipping, as shown in Figure 5–16.

Figure 5–16  Properties of the External Application

Find a Web clipping.

Congratulations on successfully choosing a Web clipping for this Portlet. You can click "Change" if you want to change the Web clipping, or you can set the following properties for your Web clipping. Or, you can remove your current choice and revert to the previous Web clipping by clicking "Cancel".

URL Location: http://www.metaLink.oracle.com

Properties

You can set some properties of the Web clipping.

URL Rewriting: Inline
Title: Oracle MetaLink
Description: Oracle MetaLink
Time Out (seconds): 18
Expires (minutes): 30
g. In the Find a Web Clipping page, from the **URL Rewriting** list, choose **Inline** to specify that you want link targets displayed inside the portlet, rather than in a new browser window. **OK** to display the selected Web clipping in the Web Clipping portlet on your page, as shown in **Figure 5–17**.

**Figure 5–17  External Application Displayed in Portlet**

Now, the Web clipping, even though it is from a page requiring authentication, is available in your portlet.

Note that you can associate only one external application with a provider. For each external application, you must register a new provider. Each portal user accesses the authenticated content using their user name and password for that system, not the page designer’s credentials.

### 5.4 Adding a Web Clipping That Users Can Personalize

This section walks you through an example that demonstrates how you can enable end users to personalize their own view of the content in a Web Clipping portlet.

In the example, you perform the following tasks:

- **Adding a Web Clipping Portlet to a Personal Page**
- **Selecting a Clipping in OTN**
- **Personalizing a Web Clipping Portlet**

#### 5.4.1 Adding a Web Clipping Portlet to a Personal Page

Administrators can set up personal pages for all users. This task assumes that the administrator has enabled this functionality. In this task, you add the Web Clipping portlet to your personal page.

1. In the **Work In** field of the Page Groups portlet, select the page group that contains personal pages.
   
   By default the Page Groups portlet is located on the Build tab of the Portal Builder page.

2. In the **Pages** section, select Personal Pages. Expand the node for the first letter of your user name. **Figure 5–18** shows the Personal Pages.
3. Click your user name. Your personal page is displayed.

4. In any portlet region, click the Add Portlets icon.

5. In the Add Portlets page, click the Web Clipping Portlet link.
   By default, the Web Clipping portlet is located in the Portal Builder page of the Portlet Repository. If you cannot find this page, use the Search field to find the portlet.

6. The Web Clipping portlet is added to the Selected Portlets list. Click OK.

5.4.2 Selecting a Clipping in OTN

In this task, you navigate to the Oracle Technology Network (OTN) and search for specific information, then select the results as the clipping for your portlet.

1. In the Web Clipping portlet, click the Edit Defaults icon.

2. In the Web Clipping Studio’s Find a Web Clipping page, in the URL Location field, enter:


   Click Start. OTN displays the Portal Center page.

3. Enter a search string in the Search field at the top of the page. For this example, enter "web clipping portlet" (including the quotation marks), then click Search.

   The Search result is displayed in the Web Clipping Studio, as shown in Figure 5–19.
4. Click Section. Web Clipping Studio divides the target Web page into its clippable sections, as shown in Figure 5–20.

5. At the top left corner of the search result, click Choose.

A preview of the search result section displays.

Some sections may contain no data, only whitespace. For example, a Web page may contain an HTML <DIV> tag that contains no text or images. If you click Choose on a section that contains no data, Web Clipping displays a preview, but the preview correctly shows only whitespace. In this case, click Unselect in the preview page to return to the sectioned page. Then, select a section containing data.

6. Click Select to confirm that the search result section is the one you want to clip.

7. In the Find a Web Clipping page, click OK to display the selected Web Clipping in the Web Clipping portlet on your page. Figure 5–21 shows the Web Clipping displayed in the page.
5.4.3 Personalizing a Web Clipping Portlet

In this task, you edit the properties of the Web Clipping portlet to allow end users to display different search results in the portlet:

1. Above the Web Clipping portlet you just added, click the Edit Defaults icon, as shown in Figure 5–22.

2. In the Find a Web Clipping page, modify the following items in the Properties section:
   - From the URL Rewriting list, choose Inline to specify that you want link targets displayed inside the portlet, rather than in a new browser window.
   - In the Title field, enter OTN Search. This title displays in the header of your Web Clipping portlet, as well as the pages where users can personalize parameters for the Web clipping.

Figure 5–23 shows the Properties and Parameterize Inputs sections of the Find a Web Clipping page.
Adding a Web Clipping That Users Can Personalize

Creating Content-Based Portlets with Web Clipping 5-19

Figure 5–23  Setting Properties for a Web Clipping

Properties
You can set some properties of the Web clipping.

<table>
<thead>
<tr>
<th>URL Rewriting</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>OTN Search</td>
</tr>
<tr>
<td>Description</td>
<td>New Doc</td>
</tr>
<tr>
<td>Time Out (seconds)</td>
<td>34</td>
</tr>
<tr>
<td>Expires (minutes)</td>
<td>30</td>
</tr>
</tbody>
</table>

Parameterize Inputs
The Web clipping can be made parameterizable. Click the check box to start choosing which parameters of which URLs you have visited in the studio to be parameterizable, so that page viewers can personalize their own views of this Web clipping. You can also fill in some default values for these parameters.

3. Because the content displayed in the portlet was reached by entering information in the Search field on OTN, you can customize the parameters used by the search to allow end users to specify their own search string.

In the Parameterize Inputs section, select the Click to start parameterizing check box.

4. In the parameters table, make the following changes:
   - In the Parameters column, choose p_Query from the list.
   - In the Personalizable column, choose Param1 from the list. You can manually add more parameters in the provider.xml file if you need to.
   - In the Display Name column, enter OTN Search.
   - Make sure that Default Value displays "web clipping portlet" to be sure you have selected the right parameter.

Figure 5–24 shows the parameters table.

Figure 5–24  Specifying Parameters for User Input

5. Click OK to display the default search results in the Web Clipping portlet on your page.
6. In the Editing Views section, click View Page.
7. In the Web Clipping portlet header, click Personalize, as shown in Figure 5–25.

Figure 5–25  Clicking Personalize in the Web Clipping Portlet Header

8. In the page that displays, scroll down to the Inputs section. Notice that the parameter field for the search string is labeled OTN Search, as you specified for the Display Name for this parameter. In the OTN Search field, enter a different search string. For example, enter OmniPortlet 2004, as shown in Figure 5–26.
Figure 5–26  Specifying Input for Parameters

OTN Search

Find a Web clipping.
A Web clipping has already been chosen for this Portlet instance.

Properties
The following properties were set by the creator of this Web clipping:

<table>
<thead>
<tr>
<th>Title</th>
<th>OTN Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>New Desc</td>
</tr>
<tr>
<td>Time Out (seconds)</td>
<td>34</td>
</tr>
<tr>
<td>Expires (minutes)</td>
<td>30</td>
</tr>
</tbody>
</table>

Inputs
The Web clipping has been made personalizable. You can fill in your initial values for these parameters.

OTN Search:  OmniPortlet 2004

9. Click OK.

The Web Clipping portlet now displays the results of performing a search on OTN for OmniPortlet 2004 information, as shown in Figure 5–27.

Figure 5–27  New Web Clipping Result Based on Customer Input Parameter

5.5 Migrating from URL-Based Portlets

You can migrate URL-based portlets that are stored in a provider.xml file to Web Clipping portlets. These Web Clipping portlets will exist in the Web Clipping Repository, but you can access the portlets through pointers defined in the provider.xml file.

Note that, during the migration process, no modifications will be done on the original files used by these URL-based portlets.

The following sections describe:

- Preparing for Migration
- Performing the Migration
5.5.1 Preparing for Migration

Before you begin the migration, take the following steps:

1. Verify that the Web provider that contains the URL-based portlets you want to migrate is functional. The migration process will not succeed with non-functional URL-based portlets. To verify, make sure that the URL-based portlets are used by your Portal pages and that they appear correctly.

2. Find existing URL-based portlets.

   Before you run the migration tool, make sure that you know the file path to the service deployment properties file, which holds pointers to everything about the service, including the path to the provider.xml file that holds the URL services definitions.

   You must run the migration tool once for each service. If you have multiple services, find the entire list of service deployment properties files that will be used during migration.

   The location of these service deployment properties files may vary depending on individual deployment scenarios. Typically, with JPDK samples, they are located in:

   `INSTALL_HOME/j2ee/home/applications/jpdk/jpdk/WEB-INF/deployment`

   JPDK samples that are shipped with the OracleAS Portal are located in the same directory path.

   The list of sample service deployment properties files includes the following:

   - `urlbasicauth.properties`
   - `urlexternalauth.properties`
   - `urlnls.properties`
   - `urllparams.properties`
   - `urlsample.properties`
   - `urlssl.properties`

   In the migration process, the service deployment properties file as well as the provider.xml file used to contain old URL-based portlets will be copied and imported into the Web Clipping application, together with all the previous customizations based on the File Preference store. Because no modifications will be done on the original files used by these URL-based portlets, you do not need to back up any of these files. They can continue to be used as URL-based portlets.

3. Depending on the types of URL-based portlets you have, you may need to rearrange them because of the following reasons:

   - Web Clipping does not offer parallel functionality for the XML Filter capability that URL-based portlets offer, isolate all of the URL-based portlets that use the XML Filter into a separate service deployment (and `provider.xml` file) that will not be migrated.
You may face limitations when migrating URL-based portlets that use external applications, you may decide that the migration is too costly for that subset of your URL-based portlets. In this case, you can isolate them into a separate service deployment (and provider.xml file) that will not be migrated.

4. Make sure that the machine where you are planning to run the migration tool has local access to the service deployment files and files that the deployment files point to, for both the URL-based portlets provider and the Web Clipping provider.

5. Make sure that both the Web Clipping Repository and the HTTP Proxy are configured, and that the proxy settings concur with those used by your deployment of URL-based portlets. If you have a proxy server that requires authentication, the Web Clipping provider must be configured with Requires Authentication enabled. Set the Login scheme to use a global log in (Use Login below for all users).

Check the Web Clipping Provider Test page at:
http://host:port/portalTools/webClipping/providers/webClipping

See the Oracle Application Server Portal Configuration Guide for more information about setting proxy authentication.

6. Shut down the OC4J instance where your Portal Tools application is running. This step ensures that no modifications are made to the files that will be migrated. Do not restart the OC4J instance until after the migration is completed.

To shut down the OC4J instance in an Oracle Application Server instance, use the following command (from the Oracle_home/opmn/bin directory):

```
opmnctl stopproc gid="OC4J_Instance"
```

In this example, OC4J_Instance refers to the OC4J instance where your Portal Tools application was deployed. Typically, it is OC4J_Portal.

To shut down a standalone OC4J instance, change to the OC4J_HOME/j2ee/home directory, then use the following command:

```
java -jar admin.jar ormi://host:port/ admin admin_password -shutdown
```

### 5.5.2 Performing the Migration

The migration tool parses the service deployment properties file to find the provider.xml file that contains the XML definition of the URL-based portlets to be migrated.

The tool changes the Provider Definition class from URLProviderDefinition to WcProviderDefinition. For each of the URL-based portlets defined in the provider.xml file, an equivalent clipping definition is created and inserted into the Web Clipping Repository. Then, the PortletDefinition XML snippet corresponding to the portlet being migrated is replaced with a new one that uses WcPortletDefinition as its class. The replacement XML snippet contains a Clip Id that is a pointer to the newly inserted clipping definition.

The new PortletDefinition XML snippet will no longer contain the same information as it did when it was a URLPortletDefinition. Namely, the migration tool uses the URL to visit as well as the headerTrimTag and footerTrimTag to generate an equivalent Web Clipping definition that will be stored within the Web Clipping Repository and only accessible from the provider.xml file through the Clip Id. The only information remaining will be the title and description. Refer to Section 5.2.3, "Setting Web Clipping Portlet Properties" for instructions on modifying your clipping.
To migrate portlets, take the following steps:

1. Set environment variables based on the following:
   - On UNIX, set LD_LIBRARY_PATH to either the `Install_home/lib` or `Install_home/lib32`, depending on whether the operating system is 32-bit or 64-bit.
   - On Windows, set PATH to `Install_home/bin`.

2. Change to the `ORACLE_HOME/portal/jlib` directory.

3. Use the following command to import the portlet to Web Clipping:
   ```
   java -Doracle.ons.oraclehome=ORACLE_HOME -jar wcwebdb.jar -import -from urlservices webclip_depprop urlsvc_depprop
   ```
   In the example, the parameters have the following meanings:
   - `ORACLE_HOME` refers to the directory where Oracle Application Server is installed.
   - `webclip_depprop` refers to the path to your webClipping service deployment properties file. The path is typically:
     ```
     Inst_Home/j2ee/OC4J_Instance/applications/portalTools/webClipping/WEB-INF/deployment/webClipping.properties
     ```
     `Inst_Home` refers to the location where OC4J is installed. `OC4J_Instance` refers to the OC4J instance where your PortalTools application is deployed. In an Oracle Application Server instance, it is usually `OC4J_Portal`; in a standalone OC4J instance, it may be `home`.
   - `urlsvc_depprop` refers to the path for the URL-based portlets service deployment properties file that points to URL-based portlets that you wish to migrate.
   For example, assume your URL-based portlets are deployed in the following directory on UNIX:
   ```
   $Inst_Home/j2ee/home/applications/jpdk/
   ```
   Assume that Web Clipping is deployed in the following directory and that `Inst_Home` is the same directory where Oracle Application Server is installed, `ORACLE_HOME`:
   ```
   $Inst_Home/j2ee/home/applications/portalTools/
   ```
   In this case, use the following command:
   ```
   cd $Inst_Home/portal/jlib;
   java -Doracle.ons.oraclehome=$ORACLE_HOME -jar wcwebdb.jar -import -from urlservices \ $Inst_Home/j2ee/home/applications/portalTools/webClipping/WEB-INF/deployment/webClipping.properties \ $Inst_Home/j2ee/home/applications/jpdk/jpdk/WEB-INF/deployment/urlsample.properties
   ```

4. Repeat Step 3 for each service you want to migrate.
5.5.3 Post-Migration Configuration

After you migrate all of the URL-based portlets, perform the following steps:

1. Start the OC4J instance that hosts your provider.

   To start the OC4J instance in an Oracle Application Server instance, use the following command:

   ```bash
   opmnctl startproc gid="OC4J_Instance"
   ```

   In this example, `OC4J_Instance` refers to the OC4J instance where your Portal Tools application was deployed. Typically, it is `OC4J_Portal`.

   To start a standalone OC4J instance, change directory to `$OC4J_HOME/j2ee/home`, then use the following command:

   ```bash
   java -jar oc4j.jar -err errors.log
   ```

2. Update all Portal instances where URL-Based portlets were previously registered. Go to Portal Navigator to find your previously registered URL-based Portlet provider. Click Edit Registration and then click the Connection tab to edit the connection information to the newly migrated provider.

   - If you previously registered your URL-based Portlet provider using the Service Id mechanism, your provider URL looked similar to the following:
     ```text
     http://host:port/jpdk/jpdk/providers
     ```
     In this case, because the Service Id field already contains `urn:someservice`, you now need only to update the provider URL value to the following:
     ```text
     http://host:port/portalTools/webClipping/providers
     ```
   
   - If you registered your URL-based Portlet provider using a provider URL that contained the Service Id, your provider URL looked similar to the following:
     ```text
     http://host:port/jpdk/jpdk/providers/service_id
     ```
     In this case, you need to update the provider to the following:
     ```text
     http://host:port/portalTools/webClipping/providers/service_id
     ```

   For the most part, the rest of the registration details can remain as they are. However, if you plan on using the in-line rendering functionality of the migrated portlets, you must select **Once Per User Session** for Login Frequency.

3. Restore HTTP proxy authentication setting that you altered in preparation for the migration. During the migration process, URL-based portlets are converted to Web Clipping portlets. If the Web Clipping provider has originally been configured with **Requires Authentication** enabled, your migrated URL-based portlets may not function properly.

   Check the test page at:

   ```text
   http://host:port/portalTools/webClipping/providers/webClipping
   ```

   If the Login Scheme is not set to **Use login below for all users**, then authentication is accomplished by mapping the current portal user to a proxy server username and password. A portal user’s proxy server credentials can only be entered through the **Personalize** link of the Web Clipping portlet. However this cannot be accomplished through the **Personalize** link of the migrated URL-based portlets. The workaround is to temporarily add a Web Clipping portlet to the page and...
personalize it to provide the security credentials. After this is complete, you can remove the temporary portlet.

4. Review each Portal page where you had used URL-based portlets and verify that the page still contains the content. Occasionally, you may find that, after migration, what used to be a "trimmed" Web page is now the entire Web page. This may be due to some of the limitations of the migration tool, which are discussed in Section 5.5.5, "Limitations in Migrating URL-Based Portlets". If this happens, edit the portlet by altering the Web Clipping associated with it.

5.5.4 Maintaining Migrated Portlets

After you have migrated the portlets, you no longer use the provider.xml file to edit the portlets. Instead, you use Web Clipping to edit the portlets, as described in Section 5.2.3, "Setting Web Clipping Portlet Properties".

Note the following:

- Whatever modifications you make to the clipping will affect it at the portlet definition level and therefore will affect all the instances of that portlet which may exist across Portal pages.

- When you modify the clipping and you have made the portlet rendering to be Inline, and have registered the provider with a Login Frequency of Once per User Session, then page viewers who have an instance of the particular portlet on their page, will need to log out of OracleAS Portal and log in again to see the changes implemented. This feature avoids disruption of the page viewer's usage of a particular portlet when it is being modified.

- For portlets with Web Clippings that require authentication to an external application prior to accessing the page containing the clip, the "off-line" editing mechanism detailed in the previous item does not provide such authentication information. Consequently, there is no way to modify the migrated clipping if the original URL-based portlet uses external applications. There is currently no workaround for this limitation.

5.5.5 Limitations in Migrating URL-Based Portlets

This sections describes limitations in migrating URL-based portlets to Web Clipping and describes some differences between the two methods.

- There is a fundamental difference between URL-based portlets and Web Clipping in the way that clippings are defined.

URL-based portlets use the headerTrimTags and footerTrimTags tags to define the begin and end of a particular section in an HTML page. On the other hand, Web Clippings use the HTML tag path (for example, html/body/table[2]/tr[2]/td[1]) to denote the path to use when looking for an HTML tag that would contain the clipping within the page.

While the header and footer trim tags for URL-based portlets offer greater flexibility (that is, the clipping does not need to reside within one single HTML tag), it is not as robust. When a clipping can no longer be found in the same location within a page, there is no way to apply fuzzy matching logic to find the clipping again. On the other hand, Web Clipping stores key pieces of information to locate the clipping within the page. If the clipping is not in the same location, the fuzzy match feature looks for other candidates on the same HTML page to account for the possibility that the clip may have moved.
The migration tool tries to find the clipping (HTML Tag) that encapsulates the headerTrimTags and footerTrimTags. Often, this will not be the exact HTML that was extracted by the URL-based portlet. In these cases, the entire Web page will be returned as the content of the migrated Web Clipping Portlet. To amend this, simply edit the portlet.

URL-based portlets that have been migrated to Web Clipping do not support proxy authentication by default. This is because URL-based portlets inherently do not support proxy authentication and Web Clipping preserves the edit mode of the portlets. Their edit mode does not provide an opportunity to enter authentication information. To work around this restriction, add an empty Web Clipping portlet to the same portal and use the Web Clipping portlet Personalize link to enter the user name and password for proxy authentication.

URL-based portlets that used external applications are migrated as full-page Web Clipping portlets. This means that the URL previously specified by the URL-based portlet definition will be the one used to display the full page after the log in process. Whatever headerTrimTags and footerTrimTags you have previously specified will be lost, because external application integration occurs at the Portal side while the migration tool only handles the provider side. Because you are not connected to any Portal instances at the time of the migration, and the external applications framework resides only on the Portal side, the migration tool cannot fetch the authentication information necessary to log in to parse the HTML and use the headerTrimTags and footerTrimTags that you have previously specified to compute an HTML Tag Path to store in the Web Clipping Definition.

In maintaining migrated portlets, you can edit the clipping through links in the test page. However, this does not include URL-based portlets that used external applications to authenticate themselves to the external sites. There is currently no workaround for this issue.

URL-based portlets that use the XML Filter cannot be migrated to Web Clipping portlets because there is a lack of such a functionality in the Web Clipping Portlet. Before proceeding with migration, make sure that you have backed up the URL-based portlets that use the XML Filter into a separate URL-based portlet provider. You must do this because the migration occurs at the provider level. That is, upon interpreting that a particular provider.xml file has URLProviderDefinition for its provider definition class, all the portlet definitions contained within that provider definition are migrated.

5.6 Current Limitations for Web Clipping

This section describes current limitations for Web Clipping. For information about the latest limitations in a release, be sure to read the Oracle Application Server Release Notes.

If the site to which you are connecting uses a large amount of JavaScript to manipulate cookies or uses the JavaScript method document.write to modify the HTML document being written, you may not be able to clip content from the site.

When you integrate with partner applications (through the use of mod_osso), you cannot clip directly through those partner applications in an authenticated manner. However, you can use the partner applications through the external application framework.

You cannot use the Web Clipping portlet to clip OracleAS Portal pages. As a workaround, examine the portlet that is supplying the data and take the appropriate action based on the following:
For database provider portlets, use export/import to copy pages across portals.

For Web provider portlets, re-register the same provider in the destination portal and edit the portal manually.

- Note the following about Web Clipping and the use of cascading style sheets (CSS):
  - If a Web page contains more than one portlet that uses a CSS, they should not conflict if the CSS uses distinct style names, such as OraRef, to specify a style within an HTML tag, rather than using an HTML tag name, such as <A>, as the name of the style.
  - If one portlet uses a CSS, and that CSS overwrites the behavior of HTML tags by using the name of the tag, such as <A>, as the name of the style, and a second portlet on the same page does not use a CSS, the second portlet will be affected by the style instructions of the CSS of the first portlet.
  - If two portlets on the same page use a different CSS and each CSS overwrites the behavior of HTML tags by using the name of an HTML tag, such as <A>, as the name of the style, the style that will be displayed depends on the browser.

- Web Clipping checks for NLS settings in the following way:
  1. Web Clipping checks the Content-Type in the HTTP header for the charset attribute. If this is present, it assumes that this is the character encoding of the HTML page.
  2. If the charset attribute is not present, it checks the HTML META tag on the page to determine the character encoding.
  3. If the HTML META tag is not found, Web Clipping uses the charset in the previous browsed page. If this is the first page, it defaults to the ISO-8859-1 character encoding.
  4. If the value of the charset for Content-Type or META tag is not supported (for example, if the charset was specified as NONE), Web clipping uses the default character set, ISO-8859-1, not the charset in the previously browsed page.

- To use the Web Clipping portlet, you must use Netscape 7.0 or higher, or Microsoft Internet Explorer 5.5 or higher for Windows 2000, or Microsoft Internet Explorer 6.0 or higher for Windows XP.

  If you use browser versions older than these, you may encounter JavaScript errors.

For troubleshooting information, see Appendix B, "Troubleshooting Portlets and Providers".

### 5.7 Summary

In this chapter, you learned how to use Web Clipping to add Web content to a Portal page, including adding authenticated content and personalized content to a page. In addition, you learned how to migrate URL-based portlets to Web Clipping. For more information about using Web Clipping, click the Help link on any of the Web Clipping pages.
Creating Java Portlets

This chapter explains how to create Java portlets based on the Java Portlet Specification (JSR 168) or the Oracle Application Server Portal Developer Kit-Java (PDK-Java) using the Portal Extension for Oracle JDeveloper. This chapter includes the following sections:

- Guidelines for Creating Java Portlets
- Introduction to Java Portlet Specification (JPS) and WSRP
  - The Relationship Between WSRP and JPS
- Configuring Your Application Server to Run JPS-Compliant Portlets
- Building JPS-Compliant Portlets with Oracle JDeveloper
  - Installing the Portal Extension for Oracle JDeveloper
  - Building JPS-compliant Portlets
- Introduction to PDK-Java
- Building PDK-Java Portlets with Oracle JDeveloper
  - Installing the Portal Extension for Oracle JDeveloper
  - Building PDK-Java Portlets

The source code for many of the examples referenced in this chapter is available as part of PDK-Java. You can download PDK-Java from the OracleAS Portal Developer Kit (PDK) page on OTN:


When you unzip PDK-Java, you will find the examples in:

../pdk/jpdk/v2/src/oracle/portal/sample/v2/devguide

You can find the JavaDoc reference for PDK-Java in:

../pdk/jpdk/v2/apidoc
6.1 Guidelines for Creating Java Portlets

When you write your portlets in Java for either the Java Portlet Specification (JPS) or PDK-Java, you should follow the best practices described in this section.

- Guidelines for Show Modes
- Guidelines for Navigation within a Portlet
- Guidelines for Mobile Portlets

6.1.1 Guidelines for Show Modes

Show mode exhibits the runtime portlet functionality seen by users. JPS offers some modes not offered by OracleAS Portal and vice versa. If you are coding portlets to JPS, you can declare custom portlet modes that map to the extra modes offered by OracleAS Portal.

An OracleAS Portal portlet may have the following Show modes, each with its own visualization and behavior. JPS portlets can define custom portlet modes in portlet.xml. Defining custom modes is especially useful if the portlet must interoperate with portal implementations from other vendors.

- Shared Screen Mode (View Mode for JPS)
- Edit Mode (JPS and OracleAS Portal)
- Edit Defaults Mode (JPS and OracleAS Portal)
- Preview Mode (JPS and OracleAS Portal)
- Full Screen Mode (OracleAS Portal)
- Help Mode (JPS and OracleAS Portal)
- About Mode (JPS and OracleAS Portal)
- Link Mode (OracleAS Portal)

6.1.1.1 Shared Screen Mode (View Mode for JPS)

A portlet uses Shared Screen mode (known as View mode in JPS) to appear on a page with other portlets. This is the mode most people think about when they envision a portlet. Portlets are rendered inside HTML table cells when in Shared Screen mode.

---

**Note:** Throughout this chapter, you will see references to ORACLE_HOME. ORACLE_HOME represents the full path of the Oracle home, and is used in cases where it is easy to determine which Oracle home is referenced. The following conventions are used in procedures where it is necessary to distinguish between the middle tier, OracleAS Infrastructure, or Oracle Application Server Metadata Repository Oracle home:

- MID_TIER_ORACLE_HOME, represents the full path of the middle-tier Oracle home.
- INFRA_ORACLE_HOME, represents the full path of the Oracle Application Server Infrastructure Oracle home.
- METADATA_REP_ORACLE_HOME, represents the full path of the OracleAS Infrastructure home containing the Oracle Application Server Metadata Repository.
This means a portlet can display any content that can be rendered within a table cell, including, among other technologies, HTML, plug-ins, and Java applets. The actual size of the table cell is variable depending on user settings, the browser width, and the amount and style of content in the portlet. When developing portlets, remember that your portlet will share a page with others and you cannot completely control its dimensions and placement.

6.1.1.1  **HTML Guidelines for Rendering Portlets** Plain HTML is the most basic way to render portlets and provides a great deal of flexibility to portlet developers. You can use almost any standard HTML paradigm, such as links, forms, images, tables, as long as it can display within an HTML table cell. Improperly written HTML may appear inconsistently across different browsers and, in the worst case, could cause parts of your page not to appear at all. Ensure that you adhere to the following rules:

- **Use standard HTML.** The official HTML specification is available from the W3C (more information available at: [http://www.w3.org/MarkUp/](http://www.w3.org/MarkUp/)).
- **Avoid unterminated and extraneous tags.** The behavior of pages with improperly terminated tags is unpredictable because it depends on what the browser chooses to do. Tools like weblint ([http://www.weblint.org/](http://www.weblint.org/)) and HTML Tidy ([http://www.w3.org/People/Raggett/tidy/](http://www.w3.org/People/Raggett/tidy/)) can help detect and fix hanging and unnecessary tags.
- **Avoid elements that cannot be rendered properly in an HTML table cell.** Some constructs cannot be used simply because they do not display correctly in a table cell. Frames, for example, do not appear when inserted in a table.
- **Keep portlet content concise.** Do not try to take full screen content and expose it through a small portlet. You will only end up with portlet content too small or cramped for smaller monitors. Full screen content is best viewed in Full Screen mode of OracleAS Portal.
- **Do not create fixed-width HTML tables in portlets.** You have no way to tell how wide a column your portlet will have on a user’s page. If your portlet requires more room than given, it might overlap with another portlet in certain browsers.
- **Avoid long, unbroken lines of text.** The result is similar to what happens with wide fixed-width tables. Your portlet might overlap other portlets in certain browsers.
- **Check behavior when resizing the page.** Test your portlet’s behavior when the browser window is resized to ensure that it works in different browser window sizes.
- **Check behavior when the default browser font changes.** People may choose whatever font size they wish and they can change it at any time. Your portlet should handle these situations gracefully.

The HTML you use also impacts the perceived performance of your site. Users judge performance based on how long it takes for them to see the page they requested, and browsers require time to interpret and display HTML. Given that, you should avoid the following:

- **Deeply nested tables:** Deeply nested tables slow performance dramatically in some older browser versions. OracleAS Portal draws several levels of tables to render portlets. If your portlets use tables within tables, your users may have to wait quite a while for those pages to render.
- **Lengthy, complex HTML:** Portlets share a page with other portlets. Thus, portlet generation times can significantly effect the overall performance of the page. If
portlets must render complex HTML or wait for external resources, such as third party applications, it can greatly slow the rendering of the page.

6.1.1.2 Cascading Style Sheet Guidelines for Rendering Portlets

The fonts and colors of every portlet on a page should match the style settings chosen by the user. To accomplish this goal, these style selections are embedded automatically using a Cascading Style Sheet (CSS) on each OracleAS Portal page. The portlets access these settings for their fonts and colors, either directly or using the API.

While different browsers have implemented varying levels of the full CSS specification, OracleAS Portal uses a very basic subset of this specification to allow for consistent fonts and colors. CSS implementation levels should not affect the consistency of your pages across browsers. Follow these guidelines for using CSS:

- **Use CSS instead of hard coding.** Hard coding fonts and colors is extremely dangerous. If you hard code fonts and colors, your portlet may look out of place when the user changes the page style settings. Since you have no way of knowing the user’s font and color preference choices, you might also choose to hard code a font color that turns out to be the same as the user’s chosen background color, in which case your portlet appears to be invisible to that user.

- **Use the CSS APIs to format your text.** The stylesheet definition is available at the top of OracleAS Portal pages, but you should not call it directly. Instead, use the APIs provided to format your text appropriately. This method ensures that your portlets work even if the stylesheet changes in the future.

- **Avoid using CSS for absolute positioning.** Since users can personalize their portal pages, you cannot guarantee that your portlet can appear in a particular spot.

6.1.1.2 Edit Mode (JPS and OracleAS Portal)

A portlet uses Edit mode to allow users to personalize the behavior of the portlet. Edit mode provides a list of settings that the user can change. These settings may include the title, type of content, formatting, amount of information, defaults for form elements, and anything that affects the appearance or content of the portlet.

Portal users typically access a portlet’s Edit mode by clicking **Personalize** on the portlet banner. When you click **Personalize**, a new page appears in the same browser window. The portlet typically creates a Web page representing a dialog box to choose the portlet’s settings. After applying the settings, users automatically return to the original page.

6.1.1.2.1 Guidelines for Edit Mode Operations

The following guidelines should govern what you expose to users in Edit mode:

- **Allow users to personalize the title of the portlet.** The same portlet may be added to the same portal page several times. Allowing the user to personalize the title helps alleviate confusion.

- **If using caching, invalidate the content.** If personalizations cause a change in portlet display or content, you must ensure that the portlet content is regenerated and not returned from the cache. Otherwise, the user may see incorrect content.

- **Do not use Edit mode as an administrative tool.** Edit mode is meant to give users a way of changing the behavior of their portlets. If you need to change provider settings or do other administrative tasks, you should create secured portlets specifically for those tasks.
Only show mobile options when applicable. The portlet can interrogate whether an OracleAS Portal instance is enabled for mobile devices. If the instance is not mobile-enabled, then you need not show any mobile-specific options. Furthermore, if the page might serve both mobile and desktop users, you should consider delineating between mobile options and desktop options. Refer to Section 6.1.3.6, "Tailor Personalization Pages" for additional tips.

6.1.1.2.2 Guidelines for Buttons in Edit Mode  For consistency and user convenience, Edit mode should implement the following buttons in the following order:

- **OK** saves the user personalizations and redirects the browser back to the calling portal page.
- **Apply** saves the user personalizations and reloads the current page.
- **Cancel** redirects the browser to the calling portal page without saving changes.

6.1.1.2.3 Guidelines for Rendering Personalization Values  When you show the forms used to change personalization settings, you should default the values such that the user does not have to constantly re-enter settings. When rendering the personalization values, use the following sequence to provide consistent behavior:

1. **User preference**: Query and display this user's personalizations, if available.
2. **Instance defaults**: If no user personalizations are found, query and display system defaults for the portlet instance. These are set in Edit Defaults mode and apply only to this portlet instance.
3. **Portlet defaults**: If no system default personalizations are found, display general portlet defaults, which may be blank. General portlet defaults are sometimes hard coded into the portlet but should be overridden if either of the two previous conditions apply.

This logic enables the personalizations to be presented in a predictable way, consistent with the other portlets in the portal. PDK-Java makes this type of logic easy to implement.

6.1.1.3 Edit Defaults Mode (JPS and OracleAS Portal)

A portlet uses the Edit Defaults mode to allow page designers to personalize the default behavior of a particular portlet instance. Edit Defaults mode provides a list of settings that the page designer can change. These settings may include the title, type of content, formatting, amount of information, defaults for form elements, and anything that affects the appearance or content of the portlet.

These default personalization settings can change the appearance and content of that individual portlet for all users. Because Edit Defaults mode defines the system level defaults for what a portlet displays and how it displays it, this mode should not be used as an administrative tool or for managing other portlets.

Page designers access Edit Defaults mode, when editing a page, by clicking the Edit Defaults icon just above the portlet banner.

When users click the Edit Defaults icon, a new page appears in the same browser window. The portlet typically creates a Web page representing a dialog box to personalize the portlet instance settings. After applying the settings, users are automatically returned to the original page.

6.1.1.3.1 Guidelines for Edit Defaults Mode Options  The following guidelines should govern what you expose to page designers in Edit Defaults mode:
- **If using caching, invalidate the cache.** If personalizations cause a change in portlet display or content, you must ensure that the portlet content is regenerated and not returned from the cache. Otherwise, the user may see incorrect content.

- **Do not use Edit Defaults mode as an administrative tool.** Edit Defaults mode gives users a way of changing the behavior of their portlets. If you need to change provider settings or do other administrative tasks, you should create secured portlets specifically for those tasks.

- **Only show mobile options when applicable.** The portlet can interrogate whether an OracleAS Portal instance is enabled for mobile devices. If the instance is not mobile-enabled, then you need not show any mobile-specific options. Furthermore, if the page might serve both mobile and desktop users, you should consider delineating between mobile options and desktop options. Refer to Section 6.1.3.6, "Tailor Personalization Pages" for additional tips.

### 6.1.1.3.2 Guidelines for Buttons in Edit Defaults Mode

For consistency and user convenience, Edit Defaults mode should implement the following buttons in the following order:

- **OK** saves the user personalizations and redirects the browser back to the calling portal page.
- **Apply** saves the user personalizations and reloads the current page.
- **Cancel** redirects the browser to the calling portal page without saving changes.

### 6.1.1.3.3 Guidelines for Rendering Personalization Values

When you show the forms used to change personalization settings, you should default the values so that the page designer does not have to constantly re-enter settings. When rendering personalization values, use the following sequence to provide consistent behavior:

1. **Instance preferences:** Query and display system defaults for the portlet instance.
2. **Portlet defaults:** If no system default personalizations are found, display general portlet defaults, which may be blank. General portlet defaults are sometimes hard coded into the portlet but should be overridden by system defaults.

This logic enables the personalizations to be presented in a predictable way, consistent with the other portlets in the portal.

### 6.1.1.4 Preview Mode (JPS and OracleAS Portal)

A portlet uses Preview mode to show the user how the portlet looks before adding it to a page. Preview mode visually represents what the portlet can do.

Portal users typically access a portlet's Preview mode by clicking its Preview icon from the Add Portlet page. A window then displays the preview of the chosen portlet. The user has the option to add that portlet to the page. Portal administrators may access Preview mode from the Portlet Repository.

Note that the portal does not draw the portlet banner when rendering the portlet in this mode.

**Guidelines for Preview Mode**

The following guidelines should govern what you expose to users in Preview mode:

- **Provide an idea of what the portlet does.** Preview mode should generate enough content for the user to get an idea of the actual content and functionality of the portlet.
Guidelines for Creating Java Portlets

- **Keep your portlet previews small.** The amount of data produced in this mode should not exceed a few lines of HTML or a screen shot. Preview mode appears in a small area, and exceeding the window's size looks unprofessional and forces users to scroll.

- **Do not use live hyperlinks.** Links may not behave as expected when rendered on the Add Portlet page or the Portlet Repository. Hyperlinks can be simulated using the underline font.

- **Do not use active form buttons.** Forms may not behave as you expect them to when rendered on the Add Portlet page or the Portlet Repository. If you decide to render form elements, do not link them to anything.

6.1.1.5 Full Screen Mode (OracleAS Portal)
Portlets use Full Screen mode to provide a larger version of the portlet for displaying additional details. Full Screen mode lets a portlet have the entire window to itself.

For example, if a portlet displays expense information, it could show a summary of the top ten spenders in Shared Screen mode and the spending totals for everyone in Full Screen mode. Portlets can also provide a shortcut to Web applications. If a portlet provided an interface to submitting receipts for expenses in Shared Screen mode, it could link to the entire expense application from Full Screen mode.

Portal users access a portlet's Full Screen mode by clicking the title of the portlet. Technically, JPS portlets do not have Full Screen mode. However, you can implement the equivalent of Full Screen mode for a JPS portlet with View mode (Shared Screen mode) and a maximized state for the window.

6.1.1.6 Help Mode (JPS and OracleAS Portal)
A portlet uses Help mode to display information about the functionality of the portlet and how to use it. The user should be able to find useful information about the portlet, its content, and its capabilities with this mode.

Portal users access a portlet's Help mode by clicking **Help** in the portlet header.

**Guidelines for Help Mode**
The following guidelines should govern what you expose to users in Help mode:

- **Describe how to use the portlet.** Users may not know all the features your portlet provides just from its interface. Describe the features and how to get the most out of them.

6.1.1.7 About Mode (JPS and OracleAS Portal)
Users should be able to see what version of the portlet is currently running, its publication and copyright information, and how to contact the author. Portlets that require registration may link to Web-based applications or contact information from this mode, as well.

Portal users access a portlet's About mode by clicking **About** on the portlet header. A new page appears in the same browser window. The portlet can either generate the content for this new page or take the user to an existing page or application.

**Guidelines for About Mode**
The following guidelines should govern what you expose to users in About mode:
Guidelines for Creating Java Portlets

- Display relevant copyright, version, and author information. Users want to know what portlet they are using and where they can get more information. The about page may become important when supporting your portlets.

6.1.1.8 Link Mode (OracleAS Portal)
A portlet uses Link mode to render a link to itself that displays on a mobile page. When the user clicks the link, the portlet is called in Show mode. The portlet then renders itself in the mobile View/Shared Screen mode.

For JPS portlets that declare support of the Oracle Mobile XML content type, OracleAS Portal renders the link in one of the following two ways:

- Call the portlet’s View mode with the MINIMIZED window state, if the portlet declares support for it.
- Otherwise, render a link using the portlet’s title.

Guidelines for Link Mode
The following guidelines should govern what you expose to users in Link mode:

- Limit content. The purpose of Link mode is to render a link without extraneous material. Link mode should simply render the short title and possibly some relevant summary information (usually just a word or two).

6.1.2 Guidelines for Navigation within a Portlet
In some ways, navigation between different sections or pages of a single portlet is identical to navigation between standard Web pages. Users can submit forms and click links. In the case of typical, simple Web pages, both of these actions involve sending a message directly to the server responsible for rendering the new content, which is then returned to the client. In the case of portlets, which comprise only part of a page, the form submission or link rendered within the portlet does not directly target the portlet. It passes information to the portlet through the portal. If a link or form within a portlet does not refer back to the portal, then following that link takes the user away from the Portal, which is not typically the desired behavior.

The portlet developer does not need to know the detailed mechanics of how the parameters of a form or link get passed around between the user, portal, and portlet. However, they must understand that they cannot write links in a portlet the same way they do for typical, simple Web pages.

6.1.2.1 Types of Links for Portlets
A portlet may render links of the following four classes:

- Intraportlet links require the portlet to be aware of the address of OracleAS Portal because they actually refer to it in some way.
- Portal links, like intraportlet links, must be aware of the address of OracleAS Portal for the same reason.
- External links make no reference to OracleAS Portal and behave in portlets as they would do in a normal Web page.
- Internal/Resource links, like external links, also make no reference to OracleAS Portal.

Figure 6–1 contains a summary of these link types. The arrows indicate how the links reference the resources to which they logically refer.
6.1.2.1.1 Intraportlet Links  Intraportlet links go to different sections or pages within a given portlet. Strictly speaking, they refer to the OracleAS Portal page containing the portlet, but they contain parameters that cause the portlet to render a different section or page within that OracleAS Portal page when it is requested by the user.

As a direct consequence, a portlet cannot expect to render links to different sections or pages of itself using relative links or absolute links based on its own server context. Intraportlet link are useful for intraportlet navigation, either as links or form submission targets.

6.1.2.1.2 Portal Links  Portal links refer to significant pages within OracleAS Portal, such as the user’s home page.

6.1.2.1.3 External Links  External links refer neither to the portlet (through an OracleAS Portal page) nor to any part of the portal. If selected, these links take the user away from the OracleAS Portal, for example, www.oracle.com.

6.1.2.1.4 Internal/Resource Links  Internal/Resource links refer to internal (to the portlet) resources. Sometimes they are exclusively used internally during portlet rendering, for example as a server side include. On other occasions, they may be used externally to reference portlet resources like images. In this latter case, you can use the PDK-Java constructResourceURL method in the UrlUtils class to retrieve images from behind a firewall through resource proxy. For example, lottery.jsp of the lottery sample, which is available with PDK-Java, contains resource proxy requests for images.

```jsp
<%@ page contentType="text/html;charset=UTF-8" %>
<%@ page session="false" import="oracle.portal.provider.v2.render.*" %>
<%@ page import="oracle.portal.provider.v2.render.http.HttpPortletRendererUtil" %>
<%@ page import="oracle.portal.provider.v2.url.UrlUtils" %>
<%@ page import="oracle.portal.sample.v2.devguide.lottery.*" %>
<%
    LottoPicker picker = new LottoPicker();
    picker.setIdentity(request.getRemoteAddr());
    %>
<% PortletRenderRequest portletRequest = (PortletRenderRequest)
request.getAttribute("oracle.portal.PortletRenderRequest"); ">
<br>
<br>

These calls cause the Parallel Page Engine to make the request to the resource and return it to the browser. For session-based providers, any cookies returned from the original initSession call to the provider are sent with the request back to the provider to maintain the right session context.

### 6.1.3 Guidelines for Mobile Portlets

OracleAS Portal is capable of rendering its pages for both HTML and non-HTML (mobile) devices. When rendering for a mobile device, OracleAS Portal requires portlets to generate content in a universal markup language called OracleAS Wireless XML.

Many portlets, known as desktop portlets, generate only HTML responses and as such can only render themselves in standard HTML browsers. Some portlets, known as mobile portlets, generate only OracleAS Wireless XML responses. These portlets can render themselves on any device, including standard HTML browsers. Many portlets, though, take a hybrid approach that renders either HTML or OracleAS Wireless XML depending on the environment. These hybrid portlets can render themselves on any device, but they render best on standard HTML browsers. Although OracleAS Wireless XML is sufficient for HTML responses, it is not as expressive as HTML. Since portlets running in both a desktop and mobile environment are typically accessed using the desktop, developers commonly choose to create hybrid portlets that can provide the best possible rendition in the desktop environment.

When building mobile portlets, you should adhere to the following guidelines:

- **Declare Capabilities**
- **Declare a Short Title**
- **Implement Personalization of the Short Title**
- **Implement Link Mode**
- **Heed Device Information**
- **Tailor Personalization Pages**
For information on how to build mobile-enabled portlets, refer to Section 7.2.9, "Enhancing Portlets for Mobile Devices".

6.1.3.1 Declare Capabilities
To properly manage portlets, OracleAS Portal must know the set of content types a portlet generates. OracleAS Portal uses this information in the following ways:

- To restrict the Portlet Repository view in the Add Portlet dialog. Only those portlets capable of being rendered on the targeted page will appear in the Add Portlet dialog. For example, when a user invokes the Add Portlet dialog from a mobile design page, only portlets that indicate they can generate OracleAS Wireless XML responses are displayed.

- To display an icon in the Portlet Repository view in the Add Portlet dialog that identifies portlets capable of being rendered on many devices. For example, when a user invokes the Add Portlet dialog from a standard design page, those portlets that are mobile capable are listed with the mobile-enabled icon (Figure 6–2) to indicate they will also render on mobile devices.

Figure 6–2 Mobile-enabled Icon

- To display only those portlets registered with the capability of generating OracleAS Wireless XML when rendering a standard page on a mobile device.

- To include only those portlets registered with the capability of generating OracleAS Wireless XML when creating a new mobile page based on an existing standard page.

6.1.3.2 Declare a Short Title
The small screen size of the typical mobile device limits the number of characters it can display in a single line without scrolling. Portlet titles, which appear as the menu item label when OracleAS Portal renders the mobile page in a menu structure, are often too long for mobile displays. Hence, you can define a short title for your portlet. The short title replaces the standard title where display space is limited.

6.1.3.3 Implement Personalization of the Short Title
The standard portlet title represents the default portlet instance name when rendered in the header of a portlet on a standard page. The portlet's short title represents the default portlet instance name when rendered as a menu item in a mobile page. Just as we recommend that portlets support personalizing the standard title, we also recommend that your portlets support personalizing the short title. This functionality enables the page designer or end user to give the instance a meaningful name.

6.1.3.4 Implement Link Mode
When OracleAS Portal renders a standard page to the desktop, it assembles portlets on the page in the tabular layout defined by the page designer. Thus, OracleAS Portal aggregates the content of many portlets on a single page. Because of their small displays, mobile devices cannot effectively display the content of multiple portlets on a single page. Instead, the page's portlets appear as links (menu items). Users view portlet content by navigating the menu one portlet at a time. The menu item links typically use the portlet's short name. Since well behaved portlets allow personalization of the short name and the portlet manages its own personalization
data, the portlet must participate in rendering the menu item link. To enable this functionality, you can implement the Link mode for portlets. In response to a request to render a portlet in Link mode, a portlet generates a link to itself in the appropriate content type. For example, if the render requests HTML, the portlet returns an anchor tag. If the render requests OracleAS Wireless XML, the portlet returns a SimpleHref tag.

6.1.3.5 Heed Device Information
All requests, whether from mobile devices or the desktop, pass general device information. For example, one passed attribute identifies the device class, such as pcbrowser, pdabrowser, or microbrowser (cell phones). A portlet developer may use this attribute to adjust the response's layout or quantity of data.

6.1.3.6 Tailor Personalization Pages
A single portlet instance must maintain a single set of user personalizations spanning all devices, mobile and desktop. Therefore, the same personalization page appears even if the instance is shared between a standard and mobile page, and some fields apply only to one environment, desktop or mobile. In this situation, the portlet should identify these fields that pertain to only one environment. For example, a portlet might display a mobile-only section on its personalization page. Furthermore, because the mobile capability is configurable, a portlet could remove mobile-only references from its personalization page when it detects that the mobile functionality is disabled.

6.2 Introduction to Java Portlet Specification (JPS) and WSRP
Organizations engaged in enterprise portal projects have found application integration to be a major issue. Until now, users developed portlets using proprietary APIs for a single portal platform and often faced a shortage of available portlets from a particular portal vendor. All this changes with the introduction of the following standards:

- Web Services for Remote Portlets (WSRP)
- Java Portlet Specification (JPS)\(^1\) based on JSR 168

These two standards enable the development of portlets that interoperate with different portal products, and therefore widen the availability of portlets within an organization. This wider availability can, in turn, dramatically increase an organization’s productivity when building enterprise portals.

WSRP is a Web services standard that enables the plug-and-play of visual, user-facing Web services with portals or other intermediary Web applications. Being a standard, WSRP enables interoperability between a standards-enabled container and any WSRP portal. WSRP defines the following:

- Web Services Definition Language (WSDL) interface for the invocation of WSRP services
- Markup fragment rules for markup emitted by WSRP services
- The method to publish, find, and bind WSRP services and metadata

JPS is a specification that defines a set of APIs to enable interoperability between portlets and portals, addressing the areas of aggregation, personalization, presentation, and security. JPS defines container services which provide the following:

- A portlet API for coding portlet functionality

\(^1\) The Java Portlet Specification 1.0 arose from Java Specification Request 168 and the JSR168 Expert Group.
- The URL-rewriting mechanism for creating user interaction within a portlet container
- The security and personalization of portlets

Oracle actively participates in the WSRP committee and is also a member of the expert group for JPS.

**Note:** HTML forms can be submitted using either the `get` or `post` method, but the WSRP standard only requires the consumer (portal) to use the `post` method. Support of the `get` method is optional according to the standard. Since portal consumers are not required to support the `get` method, we highly recommended that you use the `post` method when developing your portlets.

### The Relationship Between WSRP and JPS

WSRP is a communication protocol between portal servers and portlet containers, while JPS describes the Java Portlet API for building portlets. Combining these standards enables developers to integrate their applications from any internal or external source as portlets with WSRP portals. Building portal pages becomes as simple as selecting portlets from the OracleAS Portal repository. Figure 6–3 shows the architecture of the WSRP specification.

**Note:** Figure 6–3 illustrates the use of JPS portlets with WSRP, but it should be noted that WSRP can also work with non-JPS portlets.

#### Figure 6–3  WSRP Specification Architecture

Since OracleAS Portal's existing architecture is so similar to the one specified by the WSRP committee, OracleAS Portal is able to support communication between our portal and both the new Java Portlet APIs as well as our existing APIs (PDK-Java). Figure 6–4 shows the architecture of the WSRP portal. Notice that the JPS-compliant portlet container uses the WSRP protocol for communication and the PDK-Java portlet container uses Oracle's proprietary SOAP protocol for communication.
6.3 Configuring Your Application Server to Run JPS-Compliant Portlets

You can run JPS portlets from any number of configurations, each with its own set of requirements. This section describes the configuration steps for two of the more common scenarios for JPS portlets:

- Configuring OC4J in Oracle Application Server
- Configuring OC4J Standalone

Before you start the procedures in this section, you need to download the latest Java Portlet Container from OTN and unzip the contents to a local directory on the system you plan to configure for WSRP deployment. You can find the Java Portlet Container, as well as information regarding the most recent patches, on the PDK Download page on OTN:


6.3.1 Configuring OC4J in Oracle Application Server

When you install the Oracle Application Server, you may or may not have chosen the Portal and Wireless option and the configuration steps you must follow vary between these cases:

- Configuring with the Portal and Wireless Option
- Configuring without the Portal and Wireless Option

6.3.1.1 Configuring with the Portal and Wireless Option

In order to deploy your JPS portlets to the Oracle Application Server with the Portal and Wireless option, you need to first perform the following steps:

1. Create a new OC4J instance into which you can deploy your JPS portlets:
   a. Log in to the Application Server Control Console for the Oracle Application Server middle tier instance to which you wish to deploy JPS portlets.
   b. The Application Server Control Console most likely displays multiple instances that you can manage (see Figure 6–5). Click the Oracle Application Server middle tier instance to which you plan to deploy JPS portlets.
c. Under System Components (see Figure 6–6), click **Create OC4J Instance**.

Figure 6–6  System Components in Application Server Control Console

<table>
<thead>
<tr>
<th>Select Name</th>
<th>Status</th>
<th>Start Time</th>
<th>CPU Usage (%)</th>
<th>Memory Usage (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discover</td>
<td></td>
<td></td>
<td>0.00</td>
<td>5.60</td>
</tr>
<tr>
<td>Forms</td>
<td></td>
<td></td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>home</td>
<td></td>
<td></td>
<td>0.03</td>
<td>14.36</td>
</tr>
<tr>
<td>HTTP Server</td>
<td></td>
<td></td>
<td>0.03</td>
<td>14.36</td>
</tr>
<tr>
<td>OC4J RI Forms</td>
<td></td>
<td></td>
<td>0.10</td>
<td>261.95</td>
</tr>
<tr>
<td>OC4J Portal</td>
<td></td>
<td></td>
<td>0.14</td>
<td>286.30</td>
</tr>
<tr>
<td>OC4J Wireless</td>
<td></td>
<td></td>
<td>0.10</td>
<td>194.40</td>
</tr>
</tbody>
</table>

d. Enter the name of the instance and click **Create**. For example, you might name the instance wsrp. Note that it may take a few seconds for the instance to be created.

e. A confirmation message appears indicating that the instance was created. Note that the instance is not started upon creation and it should not be started until you have completed the other configuration steps that follow.

f. Click **OK** to return to the Home tab.

2. Copy *wsrp-install.jar* from your portlet container directory (where you unzipped the Java Portlet Container) to a location named for the OC4J instance. For example:

   ```
   MID_TIER_ORACLE_HOME/j2ee/wsrp
   ```

   In this example, the OC4J instance was named wsrp.

3. Run the auto-installer from your directory:

   ```
   cd MID_TIER_ORACLE_HOME/j2ee/wsrp
   ./wsrp-install.jar
   ```
This installation utility installs the portlet container and makes the necessary configuration changes for the container to run.

4. Start the OC4J instance from the Application Server Control Console:
   1. From the System Components area, select the component (for example, wsrp).
   2. Click Start.

5. You should now test your WSRP configuration to confirm it is working correctly. You can use OracleAS Portal's sample WSRP producer, \texttt{wsrp-sample.ear}, for this purpose.
   a. Return to the Application Server Control Console for the Oracle Application Server middle tier instance you configured for JPS portlets.
   b. Click the Oracle Application Server middle tier instance to which you plan to deploy JPS portlets.
   c. Click the OC4J container you created earlier, \texttt{wsrp}.
   d. Click the Applications tab (see Figure 6–7).

\textbf{Figure 6–7 Applications Tab of OC4J Container}

```
<table>
<thead>
<tr>
<th>Select Name</th>
<th>Path</th>
<th>Parent Application</th>
<th>Active Requests</th>
<th>Request Processing Time (seconds)</th>
<th>Active EJB Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>guestLogin</td>
<td>Applications/guestLogin.ear</td>
<td>default</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>FrankWebAppOC4J</td>
<td>Applications/FrankWebAppOC4J.ear</td>
<td>default</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>GroupPortlet</td>
<td>Applications/GroupPortlet.ear</td>
<td>default</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>ResPortlet</td>
<td>Applications/ResPortlet.ear</td>
<td>default</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>sampleportlets</td>
<td>Applications/sampleportlets.ear</td>
<td>default</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>webapp1</td>
<td>Applications/webapp1.ear</td>
<td>default</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>wsrp samples</td>
<td>Applications/wsrp-samples.ear</td>
<td>default</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
</tr>
</tbody>
</table>
```

5. Click Deploy EAR file.

f. Fill out the Deploy Application page (see Figure 6–8) as described in Table 6–1.

\textbf{Table 6–1 Deployment Settings for EAR File, Portal and Wireless}

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>J2EE Application</td>
<td>\texttt{PORTLET_CONTAINER/wsrp-samples.ear} where \texttt{PORTLET_CONTAINER} is your portlet container directory</td>
</tr>
<tr>
<td>Application Name</td>
<td>sampleportlets</td>
</tr>
<tr>
<td>Parent Application</td>
<td>default</td>
</tr>
</tbody>
</table>
g. Click Continue.

h. On the URL Mapping for Web Modules page (see Figure 6–9), change \portletapp\ to /sampleportlets.

i. You do not need to make any changes on the subsequent pages of the wizard, so click Finish.

j. The Review page should now appear (Figure 6–10). Carefully check the settings to ensure that they are correct.

k. Click Deploy. This displays the Confirmation page, as shown in Figure 6–11.
l. Click OK to dismiss the confirmation page.

6. Now that you have deployed the sample EAR file, you need to test its WSDL URL by entering it into a browser. The WSDL URL is of the form:

   http://host:port/sampleportlets/portlets?WSDL

   You should see a page similar to the one shown in Figure 6–12.

7. Using the WSDL URL, register the sample producer as you would any other producer by following the instructions in Section 6.4.2.4, "Registering and Viewing Your Portlet". Further test the configuration by adding some of the sample portlets to a page and displaying the page. If the registration fails for any reason or you cannot add portlets to a page, then please refer to the troubleshooting information in Section B.2, "Diagnosing Java Portlet Problems".

8. Once the sample producer registers successfully, you are ready to begin building your own portlets as described in Section 6.4.2, "Building JPS-compliant Portlets".

6.3.1.2 Configuring without the Portal and Wireless Option

In some cases, you may choose to install the Oracle Application Server without the Portal and Wireless option. For example, you might instead choose the J2EE and Web Cache, or Business Intelligence and Forms option. If you did choose something other than Portal and Wireless, you must perform the following steps to prepare for deploying JPS portlets:

1. Create a new OC4J instance into which you can deploy your JPS portlets:
a. Log in to the Application Server Control Console for the Oracle Application Server middle tier instance to which you wish to deploy JPS portlets.

b. The Application Server Control Console most likely displays multiple instances that you can manage (see Figure 6–5). Click the instance to which you plan to deploy JPS portlets.

c. Under System Components (see Figure 6–6), click Create OC4J Instance.

d. Enter the name of the instance and click Create. For example, you might name the instance wsrp. Note that it may take a few seconds for the instance to be created.

e. A confirmation message appears indicating that the instance was created. Note that the instance is not started upon creation and it should not be started until you have completed the remainder of the configuration steps.

f. Click OK to return to the Home tab.

2. Copy wsrp-install.jar from your portlet container directory to a location named for the OC4J instance. For example:

   MID_TIER_ORACLE_HOME/j2ee/wsrp

   In this example, the OC4J instance was named wsrp.

3. Run the auto-installer from your directory:

   cd MID_TIER_ORACLE_HOME/j2ee/wsrp
   MID_TIER_ORACLE_HOME/jdk/bin/java -jar wsrp-install.jar

   This installation utility installs the portlet container and makes the necessary configuration changes for the container to run.

4. JPS portlets store their registration and preference information in a database. For a non-Portal and Wireless installation, you must create a database schema with the relevant tables to store this information. You can use the Oracle Application Server infrastructure database or any other Oracle database for this purpose. To create this store, you must first connect to the database as a user with SYSDBA privileges:

   sqlplus "sys/<sys_password>@<service_name> AS SYSDBA"

5. To create the preference store, run the ptlwsrp_data.sql script using the following command. You will find ptlwsrp_data.sql in the root directory where you unzipped the portlet container.

   Note: ptlwsrp_data.sql creates a database user named portlet_prefs with a password of portlet_prefs, and populates the schema with the required database objects.

   @ptlwsrp_data portlet_prefs portal

   where portlet_prefs is the name of a user account that stores the data and portal is the name of a default table space to use for that account.

6. Return to the Application Server Control Console. Log in again if necessary and navigate back to the Home tab for the instance to which you plan to deploy JPS portlets (see Figure 6–6).

7. Click the component (for example, wsrp).
8. Click the Administration tab (see Figure 6–13).

**Figure 6–13  Administration Tab for the WSRP Container**

9. Under Application Defaults (see Figure 6–14), click Data Sources.

**Figure 6–14  Application Defaults on Administration Tab**

10. Click Create on the Data Sources page (see Figure 6–15).

**Figure 6–15  Data Sources Page**

11. Fill out the fields on the Create Data Source page as shown in Table 6–2.

**Table 6–2  Create Data Source Settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>OracleDS</td>
</tr>
<tr>
<td>Data Source Class</td>
<td>com.evermind.sql.DriverManagerDataSource</td>
</tr>
<tr>
<td>JDBC URL</td>
<td>For example:</td>
</tr>
<tr>
<td></td>
<td>jdbc:oracle:thin:@(description=(address=(host=myhost.mydomain.com) (protocol=tcp) (port=1521)) (connect_data=(service_name=myservice.mydomain.com)))</td>
</tr>
<tr>
<td>JDBC Driver</td>
<td>oracle.jdbc.driver.OracleDriver</td>
</tr>
<tr>
<td>Username</td>
<td>portlet_prefs</td>
</tr>
<tr>
<td></td>
<td>You need to fill in the appropriate user account name for your system.</td>
</tr>
<tr>
<td>Password</td>
<td>portlet_prefs</td>
</tr>
<tr>
<td></td>
<td>You need to fill in the password for the appropriate user account for your system.</td>
</tr>
</tbody>
</table>
Configuring Your Application Server to Run JPS-Compliant Portlets

Create Java Portlets

12. Click Create.

13. Click Yes when prompted to restart the OC4J instance.

14. You should now test your WSRP configuration to confirm it is working correctly. You can use OracleAS Portal’s sample WSRP producer, wsrp-sample.ear, for this purpose.

   a. Return to the Application Server Control Console for the Oracle Application Server middle tier instance to which you wish to deploy JPS portlets.

   b. Click the Oracle Application Server middle tier instance to which you plan to deploy JPS portlets.

   c. Click the OC4J container you created earlier, wsrp.

   d. Click the Applications tab (see Figure 6–7).

   e. Click Deploy EAR file.

   f. Fill out the Deploy Application page (see Figure 6–8) as described in Table 6–3.

   g. Click Continue.

   h. On the URL Mapping for Web Modules page (see Figure 6–9), change /portletapp to /sampleportlets.

   i. You do not need to make any changes on the subsequent pages of the wizard, so click Finish.

   j. The Review page should now appear (Figure 6–10). Carefully check the settings to ensure that they are correct.

   k. Click Deploy.

   l. Click OK to dismiss the confirmation page.

15. Now that you have deployed the sample EAR file, you need to test its WSDL URL by entering it into a browser. The WSDL URL is of the form:

   http://host:port/sampleportlets/portlets?WSDL

   You should see a page similar to the one shown in Figure 6–12.
16. Using the WSDL URL, register the sample producer as you would any other producer by following the instructions in Section 6.4.2.4, "Registering and Viewing Your Portlet". Further test the configuration by adding some of the sample portlets to a page and displaying the page. If the registration fails for any reason or you cannot add portlets to a page, then please refer to the troubleshooting information in Section B.2, "Diagnosing Java Portlet Problems".

17. Once the sample producer registers successfully, you are ready to begin building your own portlets as described in Section 6.4.2, "Building JPS-compliant Portlets".

6.3.2 Configuring OC4J Standalone

In order to deploy your JPS portlets to OC4J standalone, you need to first perform the following steps:

1. If you do not already have OC4J standalone, download it from OTN and install it. You can obtain the download from:
   

   If you already have a running OC4J standalone instance, you can skip this step.

2. Stop the OC4J standalone instance with the following command line:

   \java -jar admin.jar ormi://localhost admin $admin_password -shutdown

3. Copy wsrp-install.jar from your portlet container directory to:

   OC4J_HOME/j2ee/home

4. Run the auto-installer from your directory:

   \cd OC4J_HOME/j2ee/home
   \java -jar wsrp-install.jar

   This installation utility installs the portlet container and makes the necessary configuration changes for the container to run.

5. JPS portlets store their registration and preference information in a database. For an OC4J standalone installation, you must create a database schema with the relevant tables to store this information. You can use the Oracle Application Server infrastructure database or any other Oracle database for this purpose. To create this store, you must first connect to the database as a user with SYSDBA privileges:

   sqlplus "sys/<sys_password>@<service_name> AS SYSDBA"

6. To create the preference store, run the ptlwsrp_data.sql script using the following command. You will find ptlwsrp_data.sql in the root directory where you unzipped the portlet container.

   \@ptlwsrp_data portlet_prefs portal

   where portlet_prefs is the name of a user account that stores the data and portal is the name of a default table space to use for that account.
7. Edit the file `data-sources.xml` located in `OC4J_HOME/j2ee/home/config`, where `OC4J_HOME` is the location in which you unzipped OC4J. Add a new `data-source` entry that maps the connection details for the preference store schema to a JDBC data source with an `ejb-location` of `jdbc/portletPrefs`. For example:

```xml
<data-source
class="com.evermind.sql.DriverManagerDataSource"
name="OracleDS"
location="jdbc/emulatedPortletPrefs"
xa-location="jdbc/xa/portletPrefs"
ejb-location="jdbc/portletPrefs"
connection-driver="oracle.jdbc.driver.OracleDriver"
username="portlet_prefs"
password="portlet_prefs"
url="jdbc:oracle:thin:@(description=(address=
(host=myhost.domain.com)(protocol=tcp)
(port=1521))(connect_data=(
 service_name=myservice.mydomain.com)))"
/>```

Note that the values for `username`, `password`, and `url` are just examples. You will need to use the appropriate values for your system.

8. Start the OC4J instance with one of the following command lines to ensure that it is working properly:

   On Windows:
   
   start java -jar oc4j.jar

   On UNIX:
   
   java -jar oc4j.jar &

9. You should now test your WSRP configuration to confirm it is working correctly. You can use OracleAS Portal's sample WSRP producer, `wsrp-sample.ear`, for this purpose. Before proceeding, you need to deploy the sample portlets EAR file. Use `sampleportlets` as the context root of the Web application.

   Refer to Section 1.6, "Deploying Applications" of the Oracle Application Server Containers for J2EE Standalone User's Guide for complete information about how to deploy an EAR file.

10. Once you have deployed the sample EAR file, you need to test its WSDL URL by entering it into a browser. The WSDL URL is of the form:

    ```text
    http://host:port/sampleportlets/portlets?WSDL
    ```

    You should see a page similar to the one shown in Figure 6–12.

11. Using the WSDL URL, register the sample producer as you would any other producer by following the instructions in Section 6.4.2.4, "Registering and Viewing Your Portlet". Further test the configuration by adding some of the sample portlets to a page and displaying the page. If the registration fails for any reason or you cannot add portlets to a page, then please refer to the troubleshooting information in Section B.2, "Diagnosing Java Portlet Problems".

12. Once the sample producer registers successfully, you are ready to begin building your own portlets as described in Section 6.4.2, "Building JPS-compliant Portlets".
6.4 Building JPS-Compliant Portlets with Oracle JDeveloper

Using a convenient download from OTN, you can add extensions to Oracle JDeveloper for building portlets. To use this extension, perform the steps in the following procedures:

- Installing the Portal Extension for Oracle JDeveloper
- Building JPS-compliant Portlets

This section assumes the following:

- You are familiar with portlet terminology such as portlet Show modes. Refer to Chapter 1, "Understanding Portlets" and Section 6.1, "Guidelines for Creating Java Portlets".
- You have configured your Oracle Application Server to run JPS portlets in one of the configurations described in Section 6.3, "Configuring Your Application Server to Run JPS-Compliant Portlets".
- You are already familiar with Oracle JDeveloper and know how to build and deploy Java components using it. You can download Oracle JDeveloper from OTN. Visit the Oracle JDeveloper page on OTN:


6.4.1 Installing the Portal Extension for Oracle JDeveloper

The OracleAS Portal Developer Kit (PDK) provides you with the necessary libraries to install an extension for Oracle JDeveloper that dramatically increases your flexibility and productivity when developing portlets. This extension includes two wizards, one for building PDK-Java portlets and one for building JPS-compliant portlets. Both wizards guide you through the steps of creating the portlet skeleton and all you need do then is implement your own business logic.

To obtain the extension:

1. Visit the PDK Download page on OTN:

2. Under Portal Extension for JDeveloper, click Portal Extension for Oracle JDeveloper to download the extension.

3. Click Portal Extension for Oracle JDeveloper Installation Guide for installation instructions.

6.4.2 Building JPS-compliant Portlets

Once you have successfully installed the Portlet Wizard for Java, you can begin your interoperability portlet development quickly and easily with Oracle JDeveloper:

- Creating a Portlet: Use the wizard to create your basic portlet code and the necessary configuration files for the framework.
- Adding Portlet Logic: Extend the example code with your own business logic.
- Deploying Your Portlet to an Application Server: Use Oracle JDeveloper to deploy your application to your application server.
- Registering and Viewing Your Portlet: Register and view your portlet with OracleAS Portal.
6.4.2.1 Creating a Portlet

This section walks you through the Portlet Wizard. You can choose which portlet Show modes you want to implement and the implementation method (JSP, HTTP servlet, Java class, or HTML). The wizard then creates a simple sample implementation for each of the selected modes.

1. After you open Oracle JDeveloper, click the workspace where you want to create this project. If you do not have a workspace, you can create one as follows:
   a. Right-click the Applications node in the Application Navigator and choose New.
   b. Choose Workspace from the Items list in the New Gallery dialog box.
   c. Click OK.
   d. Enter a Workspace Name and Directory Name, and clear Add a New Empty Project in the Create Workspace dialog box.
   e. Click OK.

2. Right-click the name of the workspace in the Application Navigator and choose New Project.

3. Click Empty Project in the Items list in the New Gallery dialog box.

4. Click OK.

5. Enter the Project Name and Directory Name in the Create Project dialog box.

6. Click OK.

7. Right-click your project and select New.

8. In the Categories list, expand the Web Tier category and click Portlets.

   **Note:** If you cannot find Portlets, refer to Section 6.4.1, "Installing the Portal Extension for Oracle JDeveloper" to ensure that you have installed the extension correctly.

9. In the Items list, click Java Portlet. This displays the New Gallery dialog box, as shown in Figure 6–16.

   **Note:** Clicking Java Portlet will open the Portlet Wizard for creating JPS-compliant portlets. Clicking Oracle PDK Java Portlet will open the Portlet Wizard for creating PDK-Java portlets.
10. Click **OK**. The Portlet Wizard opens.

11. If you are on the Welcome page (see **Figure 6–17**) of the wizard, click **Next**.

**Figure 6–17  Welcome Page**

12. On the General Portlet Properties page, enter the values as described in **Table 6–4** and shown in **Figure 6–18**.

**Table 6–4  General Portlet Properties Values**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Enter the name of the class the wizard will create. The field is primed with a default class name that you may accept or change.</td>
</tr>
</tbody>
</table>
13. Click Next.

**Note:** Finish is not enabled until after the second step of the wizard has been completed.

14. On the Name and Attribution page, enter the values as described in Table 6–5 and shown in Figure 6–19.

**Table 6–5  Name and Attribution Values**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Name</td>
<td>Enter the name that will be displayed in the OracleAS Portal catalog or repository. Enter My Java Portlet for this example.</td>
</tr>
<tr>
<td>Portlet Title</td>
<td>Enter the title that will appear on the portlet header (on a portal page). Enter Welcome to my company for this example.</td>
</tr>
<tr>
<td>Short Title</td>
<td>Enter the title that will appear on the portlet header for mobile devices. Enter Welcome for this example.</td>
</tr>
<tr>
<td>Description</td>
<td>Enter a description of your portlet. Enter This Java portlet displays a welcome message to users. for this example.</td>
</tr>
</tbody>
</table>
15. Click Next to continue specifying the portlet's properties or click Finish. If you click Finish at this point, the wizard chooses the default values for all remaining settings.

16. On the Content Types and Portlet Modes page shown in Figure 6–20, select the content types for your portlet and map the portlet modes to an implementation. By default, your portlet will display text/html as the content type, and a portlet mode of View. If Customizable was selected on the General Portlet Properties page, edit also appears as a portlet mode for text/html.

**Figure 6–19 Name and Attribution Page**

**Table 6–5 (Cont.) Name and Attribution Values**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keywords</td>
<td>Enter keywords to help users find the portlet in the portal. Enter welcome, new users, company information for this example.</td>
</tr>
</tbody>
</table>

**Figure 6–20 Content Types and Portlet Modes Page**
a. If you need to add content types other than text/html, click text/html and then click Add. For the purposes of this example, you do not need to add any other content types.

b. If you need to add additional portlet modes, click an existing portlet mode (for example, view) and click Add. The list of available portlet modes displays, as shown in Figure 6–21, and you can add portlet modes by moving the desired portlet modes from the Available Modes list to the Selected Modes list. For this example, you do not need to select any other portlet modes. When you are finished with the Portlet Modes dialog box, click OK.

Figure 6–21 Portlet Modes Dialog Box

![Portlet Modes Dialog Box](image)

c. Once you have added all of the desired portlet modes, you need to choose the function to be performed for each mode. For each portlet mode, click the portlet mode and select a radio button on the right. For the purposes of this example, choose Generate JSP for both the edit and view portlet modes, and leave the default path and file name of the generated JSP. For more information on portlet modes, refer to Section 6.1, "Guidelines for Creating Java Portlets".

Note: The Generate JSP and Custom Code implementation methods generate code for you in the specified location. The Map to Path implementation method routes the request through to an existing Web resource that you need to create separately.

17. Click Next.

18. Because you selected Customizable on the General Portlet Properties page earlier in the wizard, the Customization Preferences page, shown in Figure 6–22, now displays, and you can declare preferences for your portlet. If you had not selected Customizable earlier, then this page would have been skipped. On the Customization Preferences page, you specify a preference name, default value, and whether the preference value should be translated:
a. To add a preference, click Add and fill in the Add New Preference dialog box, show in Figure 6–23, (Name, Default Value(s), and whether it should be translated). For this example, you do not need to add any preferences.

**Note:** The Name is always translated, but there is not always a need to translate the Default Value. For example, if the value is an integer, no translation is needed.

b. To delete a preference, choose it in the Portlet Preferences list and click Remove. For this example, you do not need to delete any preferences.

19. Click Next.

20. On the Security Roles page, shown in Figure 6–24, you can add security roles to your portlet. The wizard has no predefined security roles. It parses the web.xml for security roles and enables them to be referenced by your portlet. You will not need to do anything for this step as a new project has no security roles defined. You can manually create the security roles according to JPS later.
21. Click Next.

22. On the Caching page, you specify whether to enable caching of your portlet by default. The portlet itself may choose to cache content for any given response. The settings on this page apply only when the portlet itself does not specify a caching condition for a response. For this example, fill out this page as described in the following steps and shown in Figure 6–25:

   a. Click Cache Portlet.

   b. Click Cache Content Expires After.

   c. Accept the default duration of the cached copy (60 seconds).

   **Note:** If you do not want any default caching for this portlet, choose Do Not Cache By Default. In this case, the wizard actually sets a cache duration of 0 seconds. As stated earlier, this cache setting only comes into play when the portlet itself does not specify a caching condition for a response.

   If you choose no caching here and you later decide that you want default caching for the portlet, you can easily go back and change the cache duration value in the portlet.xml file, which is generated by the wizard, to a number greater than zero.
23. Click Next.

24. On the Initialization Parameters page, shown in Figure 6–26, you can add any required initialization parameters for the portlet. Initialization parameters provide the Web application developer, who decides what goes into the .war file, an alternative to JNDI variables for configuring the behavior of all of the different components of the Web application (for example, servlets and portlets) in a compatible way. For more information on initialization parameters, refer to the Java Portlet Specification. For this example, no initialization parameters are needed.

25. Click Next.

26. Click Finish to generate the files for your portlet. In this case, the following files should be generated for your project node in the Application Navigator (see Figure 6–27):
Generated code for each View mode, assuming that you selected Generate JSP on the Content Types and Portlet Modes page. If you selected Custom Code instead, that code will reside in the portlet's Java class.

Two Java classes
- portlet.xml
- web.xml

Figure 6–27 Application Navigator

6.4.2.2 Adding Portlet Logic
After you create the default implementation, you can extend the sample code with your own business logic to implement the desired functionality and features. Refer to the JavaDoc or JPS for more information on adding functionality and features. For this example, you do not need to perform this step and can proceed directly to the deployment procedure.

6.4.2.3 Deploying Your Portlet to an Application Server
After you finish the wizard and successfully generate your portlet, you are ready to deploy it to an application server. Because you chose to create a JPS-compliant portlet, you can deploy it using the wizard for any vendor's JPS-compliant container. The following sections describe how to deploy a JPS-compliant portlet to Oracle’s WSRP container running on Oracle Application Server Containers for J2EE:

- Creating a Connection to Your Application Server
- Deploying the WAR File

6.4.2.3.1 Creating a Connection to Your Application Server Before you deploy your WAR file, you must establish a connection to whatever application server you are using.

- Connecting to OC4J in the Oracle Application Server
- Connecting to OC4J Standalone

Connecting to OC4J in the Oracle Application Server
To establish a connection to OC4J in the Oracle Application Server:

1. In Oracle JDeveloper, open the project you created in the previous sections when you built a portlet.
2. In the navigator, click the **Connections** tab.

3. Right-click the **Connections** node and choose **New Application Server Connection**. Complete the wizard that displays as follows:
   
   a. If the Welcome page appears, click **Next**. If you do not want the Welcome page to appear in future, select **Skip this Page Next Time**.
   
   b. Enter a meaningful name for the connection, for example, `PDKStandardsOC4JAS`, and choose **Oracle Application Server 10g** as the connection type.
   
   c. Click **Next**.
   
   d. Enter the Application Server Control Console administrator's user name (for example, `ias_admin`) and password for the relevant middle tier.
   
   e. Click **Next**.
   
   f. Enter the information in Table 6–6.
   
   g. Click **Next**.
   
   h. Enter the RMI details for the EJB client connection (server URL, username, and password). All of these values are configured in the Application Server Control Console. Note that you only need to fill out this page of the wizard if you are using EJBs in your application. Otherwise, you can skip this step.
   
   i. Verify the connection details by clicking **Test Connection**. A success message should appear if everything is correct. If the test fails, you may need to revise your connection information.
   
   j. Click **Finish**.

### Connecting to OC4J Standalone

To establish a connection to your OC4J standalone, perform the following steps:

1. In Oracle JDeveloper, open the project you created in the previous sections when you built a portlet.

2. In the navigator, click the **Connections** tab.

3. Right-click the **Connections** node and choose **New Application Server Connection**. Complete the wizard that displays as follows:
   
   a. If the Welcome page appears, click **Next**. If you do not want the Welcome page to appear in future, select **Skip this Page Next Time**.

---

**Table 6–6 Settings for New OC4J Oracle Application Server Connection**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise Manager OC4J Host Name</td>
<td>The host name for the Application Server Control Console as specified in <code>MID_TIER_ORACLE_HOME/install/setupifo.txt</code>.</td>
</tr>
<tr>
<td>Enterprise Manager OC4J HTTP Port</td>
<td>The port number for the Application Server Control Console as specified in <code>MID_TIER_ORACLE_HOME/install/setupifo.txt</code>.</td>
</tr>
<tr>
<td>OC4J Instance Name (optional)</td>
<td>The name you gave to the OC4J instance (for example, home or OC4J_Portal).</td>
</tr>
</tbody>
</table>
b. Enter a meaningful name for the connection, for example, PDKStandardsOC4J, and choose Standalone OC4J as the connection type.

c. Click Next.

d. Enter the administrator's user name and password. This password was set during the installation of the PDK Standards OC4J.

e. Click Next.

f. Enter the information in Table 6–7.

g. Click Next.

h. Verify the connection details by clicking Test Connection. A success message should appear if everything is correct. If the test fails, you may need to revise your connection information.

i. Click Finish.

6.4.2.3.2 Deploying the WAR File To create and deploy a WAR file, perform the following steps:

1. Go to the Application Navigator.

2. In your current project, right-click web.xml and choose Create WAR Deployment Profile.

3. In the Save Deployment Profile dialog box, change the name to something meaningful (for example, jsrportlet1.deploy).

4. Click OK.

5. In the Deployment Profile Properties dialog box, perform the following steps:

   a. Click Specify J2EE Web Context Root and enter my-portlet in the adjacent field.

   b. Click OK.

6. Right-click the deployment profile (for example, jsrportlet1.deploy) and choose Deploy to > the application server connection (for example, PDKStandardsOC4J).

7. When the Deployment Finished message displays in the Deployment Log at the bottom of Oracle JDeveloper, verify that no errors occurred.

Table 6–7 Settings for New OC4J Standalone Connection

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>Enter the full RMI URL for this setting. For example:</td>
</tr>
<tr>
<td></td>
<td>ormi://my.machine.com:23791/</td>
</tr>
<tr>
<td></td>
<td>The RMI port number may be found in:</td>
</tr>
<tr>
<td></td>
<td>OC4J_HOME/j2ee/home/config/rmi.xml</td>
</tr>
<tr>
<td>Target Web Site</td>
<td>Enter the name of the target Web site containing your deployed J2EE application files. For this example, you can accept the default value, http-web-site.</td>
</tr>
<tr>
<td>Local Directory Where admin.jar for OC4J Is Installed</td>
<td>Enter the path to the local admin.jar. For example: OC4J_HOME\j2ee\home</td>
</tr>
</tbody>
</table>
8. If you are using an OC4J standalone instance, you can take the URL provided in the log (for example, http://myserver.com:8888/my-portlet) and append /portlets?WSDL to construct the URL you use to register your JPS-compliant portlet with OracleAS Portal. For example:


---

**Note:** In some cases, you may get a message in the log stating that Oracle JDeveloper was unable to determine the HTTP port number of the remote server. Typically, you can determine the port number yourself by looking at the URL that takes you to your Oracle Application Server Containers for J2EE home page (for example, http://myserver.com:8888).

If you are using OC4J in the Oracle Application Server, you must construct the registration URL yourself as follows:

http://host:port/context-root/portlets?WSDL

where *host* is the server to which your producer has been deployed.

*port* is the OracleAS Web Cache HTTP Listener port from the Ports tab of the Application Server Control Console main page.

*context-root* is the Web Application’s Context Root, which is found in the WAR Deployment Profile Properties under General.

9. You should now register your producer and view your portlet. Refer to Section 6.4.2.4, "Registering and Viewing Your Portlet".

When you redeploy your portlets to the portlet container, all existing sessions between the producer and all of its consumers are lost. If a consumer tries to reuse an existing producer session, it may receive an error message the first time it tries to contact the producer after redeployment.

Error: Could not get markup. The cookie or session is invalid or there is a runtime exception.

To reestablish the producer's session, refresh the portal page. You won't see this error message if you are re-accessing the portlet from a new browser session because it automatically establishes a new producer session.

### 6.4.2.4 Registering and Viewing Your Portlet

After you’ve created and deployed the provider and its portlet(s), you must register the provider with OracleAS Portal. Registering your provider gives OracleAS Portal the information it needs to locate and communicate with your provider. After you register a provider, the provider and its portlets become available in the Portlet Repository. They are also listed in the OracleAS Portal Navigator.

---

2 The examples in this chapter typically use a port of 8888, which is the default port. Of course, you may choose to use different port numbers in your own installation.
To register providers for your standards-based portlets, do the following:

1. Open OracleAS Portal and log in. Note that to register your provider, you need to have Manage or Edit privileges on providers. If you do not have these privileges, you need to request them from your administrator.

2. If you are not already on the Portal Builder page, click **Builder** in the upper right corner.

3. Click the **Administer** tab.

4. Click the **Portlets** subtab.

5. In the Remote Providers portlet, click **Register a Provider**.

6. On the Register Provider page, shown in Figure 6–28, enter the values as described in Table 6–8.

```
Note: When you build portlets and providers with built-in tools, such as the Portlet Builder, OracleAS Portal automatically registers the provider for you. Once you’ve created your portlet, it automatically displays in the Portlet Repository. OracleAS Portal also offers built-in portlets that are contained in a preconfigured provider. For example, OmniPortlet and Web Clipping are portlets that you can use out of the box, and are already registered with OracleAS Portal. You can view these portlets in the Add Portlets list. However, if you build the portlets and providers programmatically, you must then register these providers to make them available to the portal user.
```

```
Note: If you do not want to install OracleAS Portal but you still need to test your portlets, you may want to use the OracleAS Portal Verification Service on OTN to validate your WSRP producers: http://portalstandards.oracle.com/
```

```
Table 6–8  Register Provider Page Values

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>your_nameProvider</td>
</tr>
<tr>
<td>Display Name</td>
<td>your_name Provider</td>
</tr>
<tr>
<td>Timeout</td>
<td>100</td>
</tr>
<tr>
<td>Timeout Message</td>
<td>The your_name Provider has timed out!</td>
</tr>
<tr>
<td>Implementation Style</td>
<td>WSRP</td>
</tr>
</tbody>
</table>
```
7. Click Next.

8. On the Define Connection page, shown in Figure 6–29, enter the WSDL URL for your provider in the WSDL URL field. This URL is the one that you created in step 8 of Section 6.4.2.3.2, "Deploying the WAR File". For example:


**Note:** If a provider is unavailable for some reason, it can slow down the rendering of pages that contain portlets from that provider. By default, OracleAS Portal waits until all portlets are returned before completing the page assembly. To avoid delays, you can set a time limit using the Page Assembly Timeout on the Main tab of the page properties. If OracleAS Portal cannot retrieve the portlet from the provider within the specified timeout period, it will render the page without the portlet. If, at a later time, the provider becomes available, OracleAS Portal refreshes the page, adding the missing portlets. In this way, page rendering is never halted due to the unavailability of a particular provider. For more information on the Page Assembly Timeout, refer to the Oracle Application Server Portal User’s Guide.
9. Click Next.

10. On the Portal Registration Property Values page, shown in Figure 6–30, you fill in any registration properties required by the provider. If there are none, you can proceed to the next step.
11. Click **Finish**. You should see a Registration Confirmation page similar to the one in Figure 6–31.

**Figure 6–31  Registration Confirmation Page**
Adding Your Portlet

Your portlet should now be available for adding to pages like any other portlet in the Portlet Repository. To add your portlet to a page, follow the instructions in the Oracle Application Server Portal User’s Guide.

6.5 Introduction to PDK-Java

PDK-Java gives you a framework to simplify the development of Java portlets by providing commonly required utilities and allowing you to leverage existing development skills and application components such as JSPs, servlets, and static HTML pages. PDK-Java also enables you to create portlets without having to deal directly with the complexity of communications between OracleAS Portal and providers.

The PDK-Java framework is divided into the following areas:

- The **Provider Adapter** insulates the developer from the HTTP syntax defined by OracleAS Portal for communication with Web providers. It translates the information passed between OracleAS Portal and your Java Web provider. Without an adapter, your provider would not only manage portlets, but it would also have to communicate this information directly to OracleAS Portal in the expected language. The adapter eliminates the need for your Web provider to understand the portal language and vice-versa.

- The **Provider Interface** defines the APIs (functions) required by your Java implementation to integrate with the Provider Adapter. The Provider Adapter receives messages from the portal, translates them into calls to the Provider Interface, and translates the provider's response into a format that the portal can understand. The Provider Interface contains a set of Java classes that define the methods your provider needs to implement and, in many cases, provides a standard implementation. Some of the primary classes are as follows:
  - ProviderDefinition (oracle.portal.provider.v2.ProviderDefinition)
  - ProviderInstance (oracle.portal.provider.v2.ProviderInstance)
  - PortletDefinition (oracle.portal.provider.v2.PortletDefinition)
  - PortletInstance (oracle.portal.provider.v2.PortletInstance)
  - ParameterDefinition (oracle.portal.provider.v2.ParameterDefinition)
  - EventDefinition (oracle.portal.provider.v2.EventDefinition)

- The **Provider Runtime** provides a base implementation that follows the specification of the Provider Interface. The Provider Runtime includes a set of default classes that implement each one of the Provider Interfaces and lets you leverage the rendering, personalization, and security frameworks provided with PDK-Java. These classes and the associated frameworks simplify the development of a provider by implementing common functions for OracleAS Portal requests and providing a declarative mechanism for configuring the provider. Using the Provider Runtime, you can focus your development efforts on the portlets themselves rather than the infrastructure needed to communicate with the portal. If the standard behavior of the Provider Runtime does not meet your requirements, you can easily extend or override specific behaviors. Some of the primary classes are as follows:
  - DefaultProviderDefinition (oracle.portal.provider.v2.DefaultProviderDefinition)
  - DefaultProviderInstance (oracle.portal.provider.v2.DefaultProviderInstance)
The Provider Utilities provide methods for simplifying the rendering of portlets. The utilities include methods for constructing valid links (hrefs), rendering the portlet's container (including the header), rendering HTML forms that work within a portal page, and supporting portlet caching.

6.6 Building PDK-Java Portlets with Oracle JDeveloper

Using a convenient download from OTN, you can add extensions to Oracle JDeveloper for building portlets. To use this extension, perform the steps in the following procedures:

- Installing the Portal Extension for Oracle JDeveloper
- Building PDK-Java Portlets

This section assumes the following:

- You are familiar with portlet terminology such as portlet Show modes. Refer to Chapter 1, "Understanding Portlets" and Section 6.1, "Guidelines for Creating Java Portlets".
- You are already familiar with Oracle JDeveloper and know how to build and deploy Java components using it. You can download Oracle JDeveloper from OTN. Visit the Oracle JDeveloper page on OTN:

6.6.1 Installing the Portal Extension for Oracle JDeveloper

For instructions on how to install the Portal Extension for Oracle JDeveloper, refer to Section 6.4.1, "Installing the Portal Extension for Oracle JDeveloper".

6.6.2 Building PDK-Java Portlets

Once you have successfully installed the Portal Extension for Oracle JDeveloper, you can begin your portlet development quickly and easily with Oracle JDeveloper:

- Creating a Portlet and Provider: Use the Portlet Wizard to create your basic portlet code and the necessary configuration files for the provider framework.
- Adding Portlet Logic: Extend the example code with your own business logic.
- Validating Your Portlet and Provider: Using the built-in J2EE server of Oracle JDeveloper, you can validate the configuration of your provider and its portlets.
- Deploying to an Application Server: Use Oracle JDeveloper to deploy your application to your application server.
- Registering and Viewing Your Portlet: Register and view your portlet with your local OracleAS Portal instance.
**6.6.2.1 Creating a Portlet and Provider**

This section walks you through the Portlet Wizard. You can choose which portlet Show modes you want to implement and the implementation method (JSP, HTTP servlet, Java class, or HTML). The wizard then creates a simple sample implementation for each of the selected modes.

1. After you open Oracle JDeveloper, click the workspace where you want to create this project. If you do not have a workspace, you can create one as follows:
   a. Right-click the **Applications** node in the Application Navigator and choose **New**.
   b. Choose **Workspace** from the Items list in the New Gallery dialog box.
   c. Click **OK**.
   d. Enter a **Workspace Name** and **Directory Name**, and clear **Add a New Empty Project** in the Create Workspace dialog box.
   e. Click **OK**.

2. Right-click the name of the workspace in the Application Navigator and choose **New Project**.

3. Click **Empty Project** in the Items list in the New Gallery dialog box.

4. Click **OK**.

5. Enter the **Project Name** and **Directory Name** in the Create Project dialog box.

6. Click **OK**.

7. Right-click your project and select **New**.

8. In the Categories list, expand the Web Tier category and click **Portlets**.

   **Note:** If you cannot find **Portlets**, refer to **Section 6.4.1, "Installing the Portal Extension for Oracle JDeveloper"** to ensure that you have installed the extension correctly.

9. In the Items list, click **Oracle PDK Java Portlet** as shown in **Figure 6–32**.

   **Note:** Clicking **Java Portlet** will open the Portlet Wizard for creating JPS-compliant portlets. Clicking **Oracle PDK Java Portlet** will open the Portlet Wizard for creating PDK-Java portlets.
10. Click **OK**. The Portlet Wizard displays.

11. If you are on the Welcome page of the wizard (see Figure 6–33), click **Next**.

12. On the Portlet Description page, shown in Figure 6–34, enter the names, description, and timeout settings. For this example, you can accept the default values on this page.
13. Click Next.

14. The next few pages of the wizard enable you to define the portlet modes for this portlet. The Show Modes page, shown in Figure 6–35, enables you to choose the implementation style for the Show page and whether you want to implement Show details page:

a. **Show page** is selected by default. Choose an **Implementation style** from the list. For the purposes of this example, choose JSP. Enter the **File name** for the JSP to be generated by the Portlet Wizard. For this example, you may accept the default file name.

b. If your portlet requires a details page, select Show details page and enter the **Implementation style** and **File name** as appropriate. In this example, you do not need Show details.

15. Click Next.
16. On the Customize Modes page, shown in Figure 6–36, Edit page should already be selected. For this example, choose an Implementation style of JSP and accept the default File name.

17. Select Edit Defaults page and, for this example, choose an Implementation style of JSP and accept the default File name.

**Figure 6–36 Customize Modes Page**

18. Click Next.

19. On the Additional Modes page, shown in Figure 6–37, nothing is selected by default. For this example, select Help page, choose an Implementation style of JSP, and accept the default File name.

20. Select About page, choose an Implementation style of JSP, and accept the default File name.

**Figure 6–37 Additional Modes Page**

21. Click Next.
22. On the Public Portlet Parameters page, shown in Figure 6–38, click Add.

23. Enter the values as described in Table 6–9 and shown in Figure 6–38.

<table>
<thead>
<tr>
<th>Name</th>
<th>Display Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyParam</td>
<td>My Portlet Parameter</td>
<td>This parameter displays a value.</td>
</tr>
</tbody>
</table>

24. Click Next.

25. On the Public Portlet Events page, shown in Figure 6–39, you can map parameters to events. For this example, leave this page empty and click Next.

26. On the Provider Description page, shown in Figure 6–40, enter MyJPWProvider as the Provider name. Ensure that all of the check boxes are selected.
27. Click Finish to generate the files for your portlet. In this case, the following files should be generated for your project in the Application Navigator (see Figure 6–41):

- Files for each portlet mode you selected.
- provider.xml
- web.xml
- index.jsp
- _default.properties
- myjpwprompt.properties

All these files are required to deploy and run the portlet successfully, except for index.jsp, which is used by Oracle JDeveloper for testing purposes.
6.6.2.2 Adding Portlet Logic
After you create the default implementation, you can extend the sample code with your business logic to implement the desired functionality and features. For this example, you do not need to perform this step and can proceed directly to the testing and registration procedures.

6.6.2.3 Validating Your Portlet and Provider
After you have built your portlet, you need to check the configuration to ensure that the portlet and its provider operate correctly.

Note: This procedure is for testing purposes only. After this procedure, you still need to register your provider as described in Section 6.6.2.5, "Registering and Viewing Your Portlet". For development and production, you should always deploy your portlet to an application server as described in Section 6.6.2.4, "Deploying to an Application Server".

1. In Oracle JDeveloper, open the project you created in the previous sections.
2. Find the index.jsp file for your portlet in the Navigator, right-click it, choose Run. Your browser should open with a page similar to the one shown in Figure 6–42.

Figure 6–42 Portlet Application Test Page

![Portlet Application Test Page](image)

OracleAS Portlet Application Test Page

This page lists the portlet providers available in this web application. Use the registration URL in combination with the provider's service name to register the provider. Click on the service name link to view the provider's test page.

Registration URL
[
http://130.35.95.80:8988/Workspace2-Project1-context-root/providers/
]

Service Name
mxiprovider

3. Click the link underneath Service Name. Your browser should open with a page similar to the one shown in Figure 6–43. Note that you need the URL from this page to register your provider, which is the next task.
6.6.2.4 Deploying to an Application Server

After you finish the wizard and successfully generate your portlet, you are ready to deploy it to an application server, Oracle Application Server Containers for J2EE. The following sections describe how to deploy a portlet to Oracle Application Server Containers for J2EE:

- Creating a Connection to Oracle Application Server Containers for J2EE
- Deploying the WAR File

6.6.2.4.1 Creating a Connection to Oracle Application Server Containers for J2EE

To establish a connection to your application server, perform the following steps:

**Note:** The steps that follow describe the procedure for deploying to a standalone instance of Oracle Application Server Containers for J2EE. For information about deploying to a full Oracle Application Server instance, please refer to the Oracle JDeveloper online Help system.

1. If it is not still open, start Oracle JDeveloper and open the project you created in Section 6.6.2.1, "Creating a Portlet and Provider".

2. In the Navigator, right-click Connections and choose New Application Server Connection. Complete the wizard as follows:
   a. If the Welcome page appears, click Next. If you do not want the Welcome page to appear in future, be sure to select Skip this Page Next Time.
b. Enter a meaningful name for the connection, for example, PDKJavaOC4J, and choose Standalone OC4J as the connection type.

c. Click Next.

d. Enter the administrator's user name and password. This password was set during the installation of the Oracle Application Server Containers for J2EE.

e. Click Next.

f. Enter the information in Table 6–10.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>Enter the full RMI URL for this setting. For example:</td>
</tr>
<tr>
<td></td>
<td>ormi://my.machine.com:23791/</td>
</tr>
<tr>
<td></td>
<td>The RMI port number may be found in:</td>
</tr>
<tr>
<td></td>
<td>OC4J_HOME/j2ee/home/config/rmi.xml</td>
</tr>
<tr>
<td>Target Web Site</td>
<td>Enter the name of the target Web site containing your deployed</td>
</tr>
<tr>
<td></td>
<td>J2EE application files. For the purposes of this example, you can</td>
</tr>
<tr>
<td></td>
<td>accept the default value, http-web-site.</td>
</tr>
<tr>
<td>Local Directory Where</td>
<td>Enter the path to the local admin.jar that is</td>
</tr>
<tr>
<td>admin.jar for OC4J Is Installed</td>
<td>version-compatible with the remote servers specified in the URL</td>
</tr>
<tr>
<td></td>
<td>setting. For portlets you plan to deploy to Oracle Application</td>
</tr>
<tr>
<td></td>
<td>Server Containers for J2EE 9.0.3, you can use the default admin.jar</td>
</tr>
<tr>
<td></td>
<td>included with Oracle JDeveloper 9.0.3. For portlets you plan to</td>
</tr>
<tr>
<td></td>
<td>deploy to Oracle Application Server Containers for J2EE 9.0.4, you</td>
</tr>
<tr>
<td></td>
<td>need to change the default path in this setting to point to an</td>
</tr>
<tr>
<td></td>
<td>instance of Oracle Application Server Containers for J2EE Release</td>
</tr>
<tr>
<td></td>
<td>9.0.4. For example:</td>
</tr>
<tr>
<td></td>
<td>OC4J_HOME\j2ee\home</td>
</tr>
</tbody>
</table>

g. Click Next.

h. Verify the connection details by clicking Test Connection. A success message should display if everything is correct. If the test fails, you may need to revise your connection information.

i. Click Finish.

6.6.2.4.2 Deploying the WAR File To create and deploy a WAR file, perform the following steps:

1. Right-click your portlet project and choose New.

2. In the New dialog box, under Categories, choose General > Deployment Profiles and, under Items, choose WAR File.

3. In the Create Deployment Profile dialog box, change the name to something meaningful (for example, myj2eeportlet1.deploy).

4. Click OK.

5. In the Profile Settings dialog box, perform the following steps:

a. Click Specify J2EE Web Context Root and enter myj2eeportlet1.

b. In the pane on the left, choose File Groups > WEB-INF/lib > Contributors.

c. Select Portlet Development.
d. Click OK.

6. Choose File > Save All.

7. Right-click the deployment profile (for example, myj2eeporlet1.deploy) and choose Deploy to > the application server connection (for example, PDKJavaOC4J).

8. Await the Deployment Finished message in the Deployment Log at the bottom of Oracle JDeveloper and verify that no errors occurred.

9. Take the URL provided in the Deployment Log (for example, http://myserver.com:8888/myj2eeporlet1) and append /providers to construct the URL you use to test and register your J2EE portlet with OracleAS Portal. For example:

   http://myserver.com:8888/myj2eeporlet1/providers

---

**Note:** In some cases, you may get a message in the log stating that Oracle JDeveloper was unable to determine the HTTP port number of the remote server. Typically, you can determine the port number yourself by looking at the URL that takes you to your Oracle Application Server Containers for J2EE home page (for example, http://myserver.com:8888).

---

10. Enter the URL you constructed in the preceding step in your browser. You should see a page similar to the one in Figure 6–44.

**Figure 6–44  PDK - Java Test Page for Portlets**

---

[Image: PDK Test Page - Microsoft Internet Explorer]

**Congratulations! You have successfully reached your Provider's Test Page.**

Recognizing Portlets...

MyPortlet

Recognizing component versions...

pdkjava.jar version: 9.0.4.1.0
### 6.6.2.5 Registering and Viewing Your Portlet

After you’ve created and deployed the provider and its portlet(s), you must register the provider with OracleAS Portal. Registering your provider gives OracleAS Portal the information it needs to locate and communicate with your provider. After you register a provider, the provider and its portlets become available in the Portlet Repository. They are also listed in the OracleAS Portal Navigator.

---

**Note:** When you build portlets and providers with built-in tools, such as the Portlet Builder, OracleAS Portal automatically registers the provider for you. Once you’ve created your portlet, it automatically displays in the Portlet Repository. OracleAS Portal also offers built-in portlets that are contained in a preconfigured provider. For example, OmniPortlet and Web Clipping are portlets that you can use out of the box, and are already registered with OracleAS Portal. You can view these portlets in the Add Portlets list. However, if you build the portlets and providers programmatically, you must then register these providers in order to make them available to the portal user.

---

This following steps describe how to register your provider and add your portlet to pages:

1. Open OracleAS Portal and log in. Note that to register your provider, you need to have Manage or Edit privileges on providers. If you do not have these privileges, you need to request them from your administrator.

2. If you are not already on the Builder page, click **Builder** in the upper right corner.

3. Click the **Administer** tab.

4. Click the **Portlets** sub tab.

5. In the Remote Providers portlet, click **Register a Provider**.

6. On the Provider Information page, shown in **Figure 6–45**, enter the values as described in **Table 6–11**.

---

**Table 6–11  Provider Information Page Values**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>your_namePDKJProvider</td>
</tr>
<tr>
<td>Display Name</td>
<td>your_name PDKJProvider</td>
</tr>
<tr>
<td>Timeout</td>
<td>100</td>
</tr>
<tr>
<td>Timeout Message</td>
<td>The your_name PDKJProvider has timed out!</td>
</tr>
<tr>
<td>Implementation Style</td>
<td>Web</td>
</tr>
</tbody>
</table>
7. Click Next.

8. On the Define Connection page, shown in Figure 6–46, enter the URL for your provider in the URL field. This URL is the one that you noted at the end of Section 6.6.2.4.2, "Deploying the WAR File". For example:

   http://myserver.com:8888/myj2eeportlet1/providers

**Note:** PDK-Java enables you to deploy multiple providers under a single adapter servlet. The providers are identified by the Service ID field. When you deploy a new provider, you must assign a service identifier to the provider and use that service identifier when creating your provider WAR file. The service identifier is used to look up a file called service_id.properties, which defines the characteristics of the provider, such as whether to display its test page.

For more information about service identifiers, refer to Section D.1.2, "Service Identifiers".
9. Click Finish. You should see a Registration Confirmation page similar to the one in Figure 6–47.

Figure 6–46  Define Connection Page

Figure 6–47  Registration Confirmation Page
10. Your portlet should now be available for adding to pages just as any other portlet in the Portlet Repository.

**Adding Your Portlet**

Your portlet should now be available for adding to pages like any other portlet in the Portlet Repository. To add your portlet to a page, follow the instructions in the *Oracle Application Server Portal User’s Guide.*
This chapter explains how to enhance Java portlets you created with the Oracle JDeveloper Portal Add-In, and how to make a portlet out of your struts application:

- **Enhancing JPS Portlets**
  - Adding Personalization
- **Enhancing PDK-Java Portlets**
  - Adding Show Modes
  - Adding Personalization
  - Passing Parameters and Submitting Events
  - Accessing Session Information
  - Implementing Portlet Security
  - Controlling the Export/Import of Portlet Personalizations
  - Enhancing Portlet Performance with Caching
  - Enhancing Portlets for Mobile Devices
  - Writing Multilingual Portlets
- **Building Struts Portlets with Oracle JDeveloper**
  - OracleAS Portal and the Apache Struts Framework
  - Creating a Struts Portlet

The source code for many of the examples referenced in this chapter is available as part of PDK-Java. You can download PDK-Java from the OracleAS Portal Developer Kit (PDK) page on OTN:


When you unzip PDK-Java, you will find the examples in:

../pdk/jpdk/v2/src/oracle/portal/sample/v2/devguide

You can find the JavaDoc reference for PDK-Java in:

../pdk/jpdk/v2/apidoc

### 7.1 Enhancing JPS Portlets

Once you have built your initial portlet in the Portlet Wizard as described in Section 6.4.2, "Building JPS-compliant Portlets", you will want to enhance it. Because
JPS portlets adhere to the Java standards, you can find substantial information about enhancing them from many different sources, such as third party books and Web pages. One of the more important enhancements that you might wish to perform is Adding Personalization, which is described in this section.

7.1.1 Adding Personalization

In this section, you enhance the portlet you created in Section 6.4.2, “Building JPS-compliant Portlets” with some code that allows a user in Edit or Edit Defaults mode to paste HTML into a field for the portlet to render. You will also see how easily you can redeploy a portlet.

7.1.1.1 Assumptions

You built a portlet using the wizard and successfully added it to a page.

7.1.1.2 Implementing Personalization

In this section, you add some code to My Java Portlet, redeploy the portlet, and then test it in OracleAS Portal.

1. In Oracle JDeveloper, double-click the view.jsp file for your JPS-Standard portlet in the Application Navigator.

2. Add the following code that is indicated in bold:

```jsp
<%@ page contentType="text/html"
   import="javax.portlet.*,java.util.*,mypackage1.Portlet1,
mypackage1.resource.Portlet1Bundle"%>
<%@ taglib uri="http://java.sun.com/portlet" prefix="portlet"%>
<portlet:defineObjects/>

PortletPreferences prefs = renderRequest.getPreferences();

<%= prefs.getValue("portletContent", "Portlet Content") %>
```

3. Open edit.jsp in the visual designer and click the Design tab. Notice that the JSP consists of a form field, a form input field, and two form button fields, as shown in Figure 7–1.
4. Add the following code that is indicated in bold to implement a form field called `Content`:

```html
<FORM ACTION="<portlet:actionURL/>" METHOD="POST">
<TABLE BORDER="0">
<tr><td WIDTH="20%">
  <P CLASS="portlet-form-field" ALIGN="right">
   <%= res.getString(Portlet1Bundle.PORTLETTITLE) %>
  </P></td><td WIDTH="80%">
   <INPUT CLASS="portlet-form-input-field" TYPE="TEXT"
   NAME="<%= Portlet1.PORTLETTITLE_KEY %>">
   VALUE="<%= prefs.getValue(Portlet1.PORTLETTITLE_KEY,
   res.getString("javax.portlet.title")) %>
   SIZE="20">
  </td></tr>
<tr><td width="20%">
  <p class="portlet-form-field" align="right">
   Content
  </p>
 </td><td width="80%">
  <textarea rows="10" cols="60" class="portlet-form-input-field"
   name="portletContent"><%= prefs.getValue("portletContent","Portlet Content"
   %></textarea>
 </td></tr>
<tr><td COLSPAN="2" ALIGN="CENTER">
  <INPUT CLASS="portlet-form-button" TYPE="SUBMIT" NAME="<%= Portlet1.OK_ACTION %>
  VALUE="<%= res.getString(Portlet1Bundle.OK_LABEL) %>">
  <INPUT CLASS="portlet-form-button" TYPE="SUBMIT" NAME="<%=Portlet1.APPLY_ACTION %>
  VALUE="<%= res.getString(Portlet1Bundle.APPLY_LABEL) %>">
 </td></tr>
</TABLE>
```

5. Click the Design tab to see the new form field that you just added.

6. Open `WelcomePortlet.java` in the visual editor and insert the following two lines of code (indicated in bold):

```java
<% res.getString("javax.portlet.title") %>
```
Enhancing PDK-Java Portlets

Once you have built your initial portlet in the Portlet Wizard as described in Section 6.6.2, "Building PDK-Java Portlets", you may perform the following tasks to enhance it:

- Adding Show Modes
- Adding Personalization
- Passing Parameters and Submitting Events
- Using JNDI Variables
- Accessing Session Information
- Implementing Portlet Security
- Controlling the Export/Import of Portlet Personalizations
- Enhancing Portlet Performance with Caching
- Writing Multilingual Portlets

This section assumes the following:

- You are familiar with portlet terminology such as portlet Show modes. Refer to Chapter 1, "Understanding Portlets" and Section 6.1, "Guidelines for Creating Java Portlets".
- You have already downloaded and installed the Java Portlet Container and have an Oracle Application Server Containers for J2EE 9.0.4 container to which you may deploy your portlets.

7.2 Enhancing PDK-Java Portlets

Once you have built your initial portlet in the Portlet Wizard as described in Section 6.6.2, "Building PDK-Java Portlets", you may perform the following tasks to enhance it:

- Adding Show Modes
- Adding Personalization
- Passing Parameters and Submitting Events
- Using JNDI Variables
- Accessing Session Information
- Implementing Portlet Security
- Controlling the Export/Import of Portlet Personalizations
- Enhancing Portlet Performance with Caching
- Writing Multilingual Portlets

This section assumes the following:

- You are familiar with portlet terminology such as portlet Show modes. Refer to Chapter 1, "Understanding Portlets" and Section 6.1, "Guidelines for Creating Java Portlets".
- You have already downloaded and installed the Java Portlet Container and have an Oracle Application Server Containers for J2EE 9.0.4 container to which you may deploy your portlets.
You are already familiar with Oracle JDeveloper and know how to build and deploy Java components using it. You can download Oracle JDeveloper from OTN. Visit the Oracle JDeveloper page on OTN:


7.2.1 Adding Show Modes

In the Portlet Wizard, you add Show modes by checking boxes on the wizard pages. Refer to Section 6.6.2, "Building PDK-Java Portlets" for more information about using the wizard. For each Show mode that you select in the wizard, a basic HelloWorld skeleton is created. If you need to add a Show mode after creating the portlet or you are adding one of the modes (preview or link) not available through the wizard, you can do that manually by updating provider.xml and HTML or JSPs in Oracle JDeveloper. The following sections explain how to add Show modes to a PDK-Java portlet:

- Implementing Extra Show Modes
- Updating the XML Provider Definition
- Viewing the Portlet

Once you have completed this section, you will be able to implement any Show mode using RenderManager because the principles are the same for all modes. For example, even though this section does not describe how to implement the Help mode in detail, you will understand how to do it, as the process is the same as for Preview mode, which is described here.

For more detailed information on the PDK runtime classes used in this section, refer to the JavaDoc on OTN by clicking Java Doc API on the Portlet Development page available at

http://www.oracle.com/technology/products/ias/portal/portlet_development_10g1014.html

For more information on the syntax of provider.xml, refer to the provider JavaDoc:


7.2.1.1 Assumptions

You built a portlet using the wizard and successfully added it to a page.

7.2.1.2 Implementing Extra Show Modes

Your first task when creating Show modes manually is to create an HTML file or JSP for each mode. For example, if you want to implement Preview mode, you need to create an HTML file to provide preview content.

To create an HTML file to preview content, perform the following steps:

1. In Oracle JDeveloper, open the project that contains your portlets and select the portlet in the Application Navigator to ensure the HTML page is created in the appropriate place.

2. Open Oracle JDeveloper’s visual editor and create your HTML page. For example, the following HTML could serve as a preview page:

   <p>This is the <i>preview</i> mode of your portlet!</p>
Once you have created the HTML file for previewing content, you are ready to update the XML provider definition.

### 7.2.1.3 Updating the XML Provider Definition

When you want to expose additional Show modes you must update your XML provider definition as follows:

- Set a boolean flag that indicates to the PDK Framework that a link or icon to that mode should be rendered.
- Point to the HTML file or JSP that you created for that mode.

For more information on the syntax of `provider.xml`, refer to the provider JavaDoc on OTN:


For example, if you want to render Preview mode, perform the following steps:

1. Edit the provider definition file, `provider.xml` and add the tag to activate Preview mode:
   ```xml
   <showPreview>true</showPreview>
   ```

2. Specify the preview page to be the HTML page that you created in Section 7.2.1.2, "Implementing Extra Show Modes":
   ```xml
   <previewPage>/htdocs/myportlet/MyPortletPreviewPage.html</previewPage>
   ```

3. Save the updates to `provider.xml`.

4. Redeploy your portlet. Refer to step 6 in Section 6.6.2.4.2, "Deploying the WAR File".
   
   When you redeploy, Oracle JDeveloper automatically saves and compiles the code before deploying the portlet.

### 7.2.1.4 Viewing the Portlet

To view the new Show modes, you must ensure that your updated XML provider definition is re-parsed. To do this, perform the following steps:

1. Copy the HTML file you created in Section 7.2.1.2, "Implementing Extra Show Modes" and `provider.xml` to the Oracle Application Server Containers for J2EE instance where you plan to deploy the portlet.

2. Refresh the provider.

3. Refresh the portal page containing your portlet.

To view Preview mode, do the following:

1. Edit a page or create a new page and choose **Add Portlet**.

2. Navigate to the location of your provider in the Portlet Repository (for example, Portlet Staging Area) and find your portlet. Note the magnifying glass icon next to the portlet shown in **Figure 7–2**
3. Click the magnifying glass icon next to the portlet and a preview window similar to the one in Figure 7–3 displays.
7.2.2 Adding Personalization

In Section 7.2.1, "Adding Show Modes" you learned how to use the PDK Provider Framework to activate and render additional Show modes that were either not activated when creating the portlet with the wizard or not available through the wizard (such as Link and Preview modes). This section describes the two Personalization modes (Edit and Edit Defaults) in more detail. When selected in the Java Portlet Wizard, Edit page and Edit Defaults page cause the generation of skeleton code for the two Personalization modes. The skeleton code enables you to access the personalization framework with a few lines of code rather than completely hand coding a personalization framework and a data store to hold the values.

To add personalization to your portlet, you need to do the following:

- Update the Edit page of your portlet to set and retrieve personalization changes.
- Update the Edit Defaults page of your portlet to set and retrieve personalization changes.
- Update the Show page of your portlets to use the personalization set by the user.

The Edit and Edit Defaults modes enable portlet users to change a set of customizable parameters supported by the portlet, which typically drive the way the portlet is rendered in other modes. For a particular instance of a portlet on an OracleAS Portal page, the personalizations made in the Edit and Edit Defaults modes apply only to that instance of the portlet.

- **Edit** mode personalizations are specific to the individual user making the personalizations. This mode is activated by clicking the **Personalize** link on the portlet header in show mode.
- **Edit defaults** mode personalizations apply to all users in the same locale who have not yet made specific personalizations to that portlet instance. This mode is generally only available to page designers, and can be activated by following the **Edit** icon on the page.

When rendering Edit and Edit Defaults modes, a **PortletRenderer** can carry out either of these tasks to support the personalization process:
Enhancing PDK-Java Portlets

- **Render the Edit Form**: For each of the portlet's customizable parameters, PortletRenderer uses a PortletPersonalizationManager to retrieve the current value and renders a control in an HTML form so the current value can be edited.

- **Handle Edit Form actions**: When an OK or Apply button is clicked on the standard edit form header, PortletRenderer uses a PortletPersonalizationManager to store the personalized parameters submitted by the edit form and redirects the browser to the appropriate portal page.

Therefore, the purpose of the PortletPersonalizationManager controller is to enable a PortletRenderer to store and retrieve the current values of customizable parameters that apply to a particular portlet instance and user. The PDK Framework uses the abstraction of a PersonalizationObject as a container for a set of personalized parameters and a PortletReference as the key under which a set of personalizations are stored. Thus, a PortletPersonalizationManager is simply a mechanism that allows the storage and retrieval of persisted PersonalizationObjects under a given PortletReference.

A preference store is a mechanism for storing information like user preference data, portlet/provider settings, or even portlet data, while using OracleAS Portal. The information stored in the preference store is persistent in the sense that, even if you log out and log back in later, you can still access previously saved preferences. The preference store maintains the user preference information and invokes the user preferences whenever the user logs in again. PDK-Java provides the PrefStorePersonalizationManager, which uses a PreferenceStore implementation to persist personalized data. Currently, PDK-Java has two PreferenceStore implementations: DBPreferenceStore and FilePreferenceStore. The DBPreferenceStore persists data using a JDBC compatible relational database and FilePreferenceStore persists data using the file system.

For more details of these implementations, refer to the JavaDoc on OTN by clicking Java Doc API on the Portlet Development page available at

http://www.oracle.com/technology/products/ias/portal/portlet_development_10g1014.html

---

**Note:** PDK-Java provides the Preference Store Migration/Upgrade Utility to help migrate the preference store from a file system to a database and upgrade personalizations from earlier releases. This utility is described more fully on OTN.


---

To add personalization functionality to your portlet you use PrefStorePersonalizationManager in conjunction with NameValuePersonalizationObject, that is, the default PersonalizationObject implementation. By default, the wizard generates a simple edit form for both the Edit and Edit Defaults modes to enable users to personalize the portlet title. This section describes how to update the existing code to enable portal users to personalize the portlet greeting.

**7.2.2.1 Assumptions**

1. You have followed through and understood these sections:
2. You built a portlet using the wizard, with Edit page and Edit Defaults page selected, and successfully added it to a page.

7.2.2.2 Implementing Personalization for Edit and Edit Defaults Pages

The Edit page of your portlet is called when a user personalizes the portlet. By default, the JSP generated by the wizard includes all of the required code to provide personalization of the portlet title. You just need to insert a few lines of code into the Edit page for additional personalization.

7.2.2.2.1 Reviewing the Generated Code

The wizard creates the following code for you by default:

```jsp
<%@page contentType="text/html; charset=windows-1252"
    import="oracle.portal.provider.v2.render.PortletRenderRequest"
    import="oracle.portal.provider.v2.personalize.NameValuePersonalizationObject"
    import="oracle.portal.provider.v2.render.PortletRendererUtil"%>
<%
    PortletRenderRequest pReq = (PortletRenderRequest)
    request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
%>

<P>Hello <%=pReq.getUser().getName() %>.</P>
<P>This is the <b><i>Edit</i></b> render mode!</P>
<-- This page both displays the personalization form and processes it. Display the form if there is no action parameter, process it otherwise --%>
<%
    String actionParam = PortletRendererUtil.getEditFormParameter(pReq);
    String action = request.getParameter(actionParam);
    String title = request.getParameter("my2portlet_title");
    NameValuePersonalizationObject data = (NameValuePersonalizationObject)
    PortletRendererUtil.getEditData(pReq);
    // Cancel automatically redirects to the page, so
    // will only receive OK or APPLY
    if (action !=null)
    {
        data.setPortletTitle(title);
        PortletRendererUtil.submitEditData(pReq, data);
        return;
    }
    // Otherwise just render the form.
    title = data.getPortletTitle();
%>
<table border="0">
    <td width="20%">
        <p align="right">Title:</p>
    </td>
    <td width="80%">
        <input type="TEXT" name="my2portlet_title" value="<%= title %>">
    </td>
</table>
```

---
Modifying the Generated Code  The JSP contains an input field for the portlet title. This field represents the Personalize page of the portlet where users can update the portlet title.

1. Following the table in the generated code, add a second table containing a text field and a prompt, allowing users to enter a new greeting for the portlet:

```
<table border="0">
  <tr>
    <td width="20%">
      <p align="right">Greeting:</p>
    </td>
    <td width="80%">
      <input type="TEXT" name="myportlet_greeting" value="<%= greeting %>">
    </td>
  </tr>
</table>
```

2. This HTML code simply specifies a field to enter a new greeting on the Edit page. This new greeting is displayed in the portlet's Shared Screen mode. Next, you add a string below `String title` that retrieves the value of the greeting:

```java
String title = request.getParameter("my2portlet_title");
String greeting = request.getParameter("myportlet_greeting");
```

3. Generating an Edit page from the wizard automatically includes access to the personalization framework in the page code. At the top of the Edit page, you see the `NameValuePersonalizationObject` declared. This form of personalization in OracleAS Portal allows easy storage of name/value pairs.

The Edit page handles two cases: viewing the page or applying changes to it. The changes we have made so far affect the code for viewing the page. Applying changes to the Edit page is handled in the block of code beginning with `if (action !=null)`.

In this block of code, you must store the new portlet greeting. You must also account for the case where the user decides to make no changes and you simply retrieve the existing greeting:

```java
if (action !=null) {
    data.setPortletTitle(title);
    //Put the new greeting.
    data.putString("myportlet_greeting", greeting);
    PortletRendererUtil.submitEditData(pReq, data);
    return;
}
//Otherwise just render the form.
```

You are now done updating the Edit page.

You can simply duplicate these changes for the Edit Defaults page. The Edit Defaults page is called when a page designer or portal administrator clicks Edit on the page and then clicks the Edit Defaults icon for the portlet. This page sets the default personalization for this instance of the portlet. Even though the code in the JSP is
identical, the PDK Framework and OracleAS Portal automatically handle the personalization differently depending on the Show mode (Edit or Edit Defaults).

### 7.2.2.3 Implementing Personalization for Show Pages

To have access to the personalization data in the portlet’s Shared Screen mode, you need to add a few lines of code to the Show page. These lines include the following:

- Adding import statements.
- Declaring the `NameValuePersonalizationObject`.
- Retrieving the personalization data.

1. Edit your Show page and import `NameValuePersonalizationObject` and `PortletRendererUtil`. You can copy these from the Edit page if necessary.

   ```html
   <%@page contentType="text/html; charset=windows-1252"
   import="oracle.portal.provider.v2.render.PortletRenderRequest"
   import="oracle.portal.provider.v2.personalize.NameValuePersonalizationObject"
   import="oracle.portal.provider.v2.render.PortletRendererUtil"%>
   ```

2. Declare the `NameValuePersonalizationObject` and retrieve the edit data from the portlet render request. You can copy this from the portlet’s Edit page.

   ```java
   <%
   PortletRenderRequest pReq = (PortletRenderRequest) request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
   NameValuePersonalizationObject data = (NameValuePersonalizationObject) PortletRendererUtil.getEditData(pReq);
   %>
   ```

3. Get the string information from the personalization framework:

   ```java
   String greeting = data.getString("myportlet_greeting");
   ```

4. Add some text to the Show page that displays the greeting in the Shared Screen mode of the portlet.

   ```html
   <P>Hello <%= pReq.getUser().getName() %>.</P>
   <P>This is the <b><i>show</i></b>, render mode!</P>
   <P>Greeting: <%= greeting %></P>
   ```

You have now completed updating the Show page of the portlet.

### 7.2.2.4 Preference Information Within the XML Provider Definition

The Portlet Wizard generates all of the necessary tags for accessing the PreferenceStore in the XML provider definition file (`provider.xml`). By default, at the provider level, the wizard uses the `FilePreferenceStore` class to store preferences:

```xml
<provider class="oracle.portal.provider.v2.DefaultProviderDefinition">
  <session>false</session>
  <passAllUrlParams>false</passAllUrlParams>
  <preferenceStore class="oracle.portal.provider.v2.preference.FilePreferenceStore">
    <name>prefStore1</name>
    <useHashing>true</useHashing>
  </preferenceStore>
</provider>
```
At the portlet level, tags are added to use `PrefStorePersonalizationManager` as the `personalizationManager` class and `NameValuePersonalizationObject` as the data class:

```xml
<personalizationManager class="oracle.portal.provider.v2.personalize.PrefStorePersonalizationManager">
    <dataClass>oracle.portal.provider.v2.NameValuePersonalizationObject</dataClass>
</personalizationManager>
```

You need not make any changes or updates to the XML Provider Definition if you choose to continue to use the `FilePreferenceStore` class. However, if you have a global environment for OracleAS Portal (for example, you are running in a load balanced, multi-node cluster of Oracle Application Server Containers for J2EE instances) or would prefer to store preferences in the database, you can change the class from `FilePreferenceStore` to `DBPreferenceStore`.

For more information on using `DBPreferenceStore`, refer to the Oracle Application Server Portal Configuration Guide.

For more information on the syntax of `provider.xml`, refer to the provider JavaDoc on OTN:


### 7.2.2.5 Viewing the Portlet

To view the personalization changes you made in the preceding sections, you need to deploy the portlet to your application server or Oracle Application Server Containers for J2EE and refresh the page containing your portlet. For more information on deploying your portlet, refer to Section 6.6.2.4, "Deploying to an Application Server".

You should now see that the portlet contains a null greeting. Click Personalize in the portlet title bar and update the greeting. When you return to the page, you should see your changes.

You can also test Edit Defaults by clicking Edit on the page and then clicking the Edit Defaults icon. Since you have already modified the portlet, the changes will not appear to you in Shared Screen mode unless you view the page as a public user or a different user.

### 7.2.3 Passing Parameters and Submitting Events

OracleAS Portal and the PDK provide page parameters, public and private portlet parameters, and events to enable portlet developers to easily write reusable, complex portlets. The Portlet Wizard in Oracle JDeveloper creates portlets that are already set up to use parameters and events. This feature enables you to focus solely on adding business logic to your portlets and does not require any changes to `provider.xml`.

For an overview of parameters and events, refer to the following:

- Section 2.12, "Public Portlet Parameters Support"
- Section 2.13, "Private Portlet Parameter Support"
- Section 2.14, "Event Support"

#### 7.2.3.1 Assumptions

1. You have followed through and understood Section 6.6.2, "Building PDK-Java Portlets".
2. You built a portlet using the wizard and successfully added it to a page.
7.2.3.2 Adding Parameters to Your Portlets

Using the wizard in Section 6.6.2, "Building PDK-Java Portlets", you built a basic portlet and specified a parameter called MyParam. If you did not create a parameter, you can create a new portlet now by right clicking on provider.xml in the Applications - Navigator of Oracle JDeveloper, selecting Add Portlet, and following the steps in Section 6.6.2, "Building PDK-Java Portlets".

By default, the wizard creates a portlet to which you can easily map page parameters without updating any code or files. In this section, you will use the default parameter created for you by the wizard.

To use the default parameter, you need only register the provider and add the portlet to a page. After that, you perform the following tasks:

- Create a page parameter.
- Wire the page parameter to your Java portlet.
- Enter parameter values in the URL or another portlet that passes this page parameter.

1. Go to the Parameter tab of the page properties. Note that parameters should be enabled by default, but, if not, you must enable them before proceeding.

2. Create a page parameter called MyParameter with a default value of My Default Value.

3. Expand your Java portlet and map the page parameter you just created to the portlet parameter. The portlet's parameter should map to the page parameter called MyParameter.

4. Go back to the page. Notice that, in the portlet, a value of My Default Value appears.

5. View the page and enter the parameter and a value at the end of the URL:

   &MyParameter=This%20portlet%20works

   Figure 7–4 is an example of Parameter Portlet.

---

**Note:** Each portlet is limited to 4K of data. The lengths of parameter and event names, display names, and descriptions all contribute toward this 4K limit. Hence, you should not use an excessive number of parameters and events in a portlet, or give them lengthy names and descriptions.
If you have a portlet, such as the Simple Parameter Form included with OmniPortlet, that can pass parameters, you can map parameters from that portlet to your Java portlet using the Events tab.

If you now take a look at the code and tags generated by the wizard, you see that very little code was needed to enable parameters in the Java portlet.

Review provider.xml. Note that the wizard added one tag group called inputParameter, which includes the name of the parameter for which the portlet listens.

```
<inputParameter class="oracle.portal.provider.v2.DefaultParameterDefinition">
  <name>MyParam</name>
  <displayName>My Portlet Parameter</displayName>
</inputParameter>
```

For more information on the syntax of provider.xml, refer to the provider JavaDoc on OTN:


The wizard also generated code in the JSP for your Show page that receives this parameter, and displays the parameter name and its value.

```
<% ParameterDefinition params[] = pReq.getPortletDefinition().getInputParameters(); %>
<p>This portlets input parameters are ...</p>
<table align="left" width="50%"><tr><td><span class="PortletHeading1">Value</span></td></tr>
<% String name = null; String value = null; String[] values = null; for (int i = 0; i < params.length; i++) {
  name = params[i].getName();
  values = pReq.getParameterValues(name);
  if (values != null) {
    StringBuffer temp = new StringBuffer();
    for (int j = 0; j < params.length; j++) {
      temp.append(values[j]);
      if (j + 1 != values.length) {
        temp.append(", ");
      }
    }
    value = temp.toString();
  } else {
    value = "No values submitted yet.";
  }
  %>
  <tr>
    <td><span class="PortletText2" <%= name %></span></td>
    <td><span class="PortletText2" <%= value %></span></td>
  </tr>
<%}%>
```
7.2.3.3 Passing Portlet URL Parameters

Intraportlet links refer to the OracleAS Portal page on which the portlet resides, and that portlet is most likely running remotely from OracleAS Portal. Hence, you must consider how the portlet can render a link to the correct page without some knowledge of the OracleAS Portal page's URL. For more information about the types of links used by portlets, refer to Section 6.1.2, "Guidelines for Navigation within a Portlet".

When OracleAS Portal requests that a portlet render itself, OracleAS Portal passes it various URLs, which the portlet can then use to render links, including any intraportlet links it requires. You can fetch and manipulate these URLs to simplify the task of creating links among portlets and pages in OracleAS Portal.

7.2.3.3.1 Portlet URL Types

OracleAS Portal provides the following URLs to its portlets:

- **PAGE_LINK** is a URL to the page upon which the portlet instance resides. You use this URL as the basis for all intraportlet links. If the portlet renders a link that navigates the user to another section of the same portlet, then this navigation must be encoded as a set of parameters using the PAGE_LINK. This URL is useful to both desktop and mobile portlets.

- **DESIGN_LINK** is a URL to an OracleAS Portal page that represents the portlet's personalization page. In OracleAS Portal, a portlet's Edit and Customize modes are not rendered on the same page as the portlet. The Edit and Customize modes take over the entire browser window. OracleAS Portal's portlet edit/customize page is not accessible to every user. It represents a minimal, static framework in which the portlet is free to render its personalization or edit options. This URL is only of use when rendering edit and customize links, which themselves are only supported in desktop clients.

- **LOGIN_LINK** is a URL to OracleAS Single Sign-On, should the portlet need to prompt the user (if PUBLIC) to login. This link is rarely used and only applicable to the desktop rendering of portlets.

- **BACK_LINK** is a URL to a page that OracleAS Portal considers a useful return point from the current page where the portlet renders itself. For example, when the portlet is rendering its Edit page, this link refers to the page on which the portlet resides and from which the user navigated to the Edit page. Consequently, it is the link you would encode in the buttons that accept or cancel the pending action. This URL is only useful for the desktop rendering of portlets (usually in Edit or Customize mode). Mobile portlets render a Back link automatically leaving the portlet to render just its own content.

- **EVENT_LINK** is a URL that raises an event rather than explicitly navigate to some page. This link points to the OracleAS Portal entry point to the event manager. This URL is useful to both desktop and mobile portlets.

7.2.3.3.2 URL Parameters

URL parameters are used in classic Web applications to pass information from links or forms in the browser back to the server. The server in turn takes actions and returns the appropriate content. For example, if the user of a dictionary Web site asks for information about hedgehogs, the URL submitted to the server might look something like the following:
http://dictionary.reference.com/search?q=Hedgehog

If the server is responsible for rendering the whole page and the client communicates directly with the server, this form of URL works well. In the OracleAS Portal case, the client does not communicate directly with portlets. Instead, OracleAS Portal mediates between the client and the portlet. Moreover, because most pages have multiple portlets, OracleAS Portal communicates with multiple portlets.

For example, suppose a page contains two portlets, a thesaurus portlet and a dictionary portlet. Both portlets use q as a parameter to record the search queries made by the user. If the user queries the thesaurus portlet, the URL used to re-request the page with the updated thesaurus portlet must contain the thesaurus portlet's parameter, q. The thesaurus parameter must also be distinguished from dictionary portlet parameter 1, which performs the same function for that portlet. An example URL with the properly qualified thesaurus parameter might look something like the following:

http://host/portal/page?_pageid=33,1&_dad=portal&_schema=PORTAL

   &_piref33_38279_33_1_1.q=Hedgehog

Notice the fully qualified parameter name, _piref33_38279_33_1_1.q. It identifies the parameter and distinguishes it from other parameters on the page. Further, notice that the URL contains some parameters unrelated to any portlet. These parameters are untouched by the portlet because it does not own them.

The portlet developer must ensure that the portlet:

- Properly qualifies its own parameters when they are built into links and forms.
- Leaves unchanged any parameters that do not belong to it.

The following API call transforms an unqualified parameter name into a qualified parameter name:

HttpPortletRendererUtil.portletParameter(HttpServletRequest request, String param);

HttpPortletRendererUtil is in the package oracle.portal.provider.v2.render.http.

For example:

qualParamQ = HttpPortletRendererUtil.portletParameter(r, "q");

To fetch the value of a portlet parameter from the incoming request, you can use the following API:

Note: The API converts the parameter name into the qualified parameter name before fetching the value from the incoming request. Hence, you need not perform this step.

PortletRenderRequest.getQualifiedParameter(String name)

PortletRenderRequest is in the package oracle.portal.provider.v2.render.

For example:

valueQ = r.getQualifiedParameter("q");
The other aspect of a portlet's responsibilities with respect to URL parameters is to not disturb the parameters on the URL that it does not own. The utilities you may use to ensure adherence to this rule are discussed in Section 7.2.3.3.3, "Building Links with the Portlet URL Types" and Section 7.2.3.3.4, "Building Forms with the Portlet URL Types".

7.2.3.3.3 Building Links with the Portlet URL Types  To build links with the URL parameters, you need to access them and use them when writing portlet rendering code. To fetch the URL for a link, you call the following APIs in the PDK:

```java
portletRenderRequest.getRenderContext().getPageURL()
portletRenderRequest.getRenderContext().getEventURL()
portletRenderRequest.getRenderContext().getDesignURL()
portletRenderRequest.getRenderContext().getLoginServerURL()
portletRenderRequest.getRenderContext().getBackURL()
```

In the case of portlet navigation, you need to add (or update) your portlet's parameters in the page URL. To perform this task, you can use the following API to build a suitable URL:

```java
UrlUtils.constructLink(
    PortletRenderRequest pr,
    int linkType, -- UrlUtils.PAGE_LINK in this case
    NameValue[] params,
    boolean encodeParams,
    boolean replaceParams)
```

**Note:** When rendering attributes in `SimpleResult` (for a mobile portlet), you must escape the attribute value if it is likely to contain invalid XML characters. Most URLs contain `&` to separate the URL's parameters. Hence, you usually need to escape attributes that contain URLs with:

```java
oracle.portal.utils.xml.v2.XMLUtil.escapeXMLAttribute
```

```java
String paramNameQ = "q";
String qualParamNameQ =
    HttpPortletRendererUtil.portletParameter(paramNameQ);
PortletRenderRequest pRequest = (PortletRenderRequest)
```
request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
String paramValueQ = pRequest.getQualifiedParameter(paramNameQ);

%>
<!-- Output the HTML content -->
<center>
Words similar to <%= paramValueQ %> 
<br>
Click the link to search for words related to that word.
<br>
<ul>
<% 
String[] relatedWords = Thesaurus.getRelatedWords(paramValueQ);
NameValuePair[] linkParams = new NameValue[1];
for (int i=0; i<=relatedWords.length; i++)
{
   linkParams[0] = new NameValue(
      qualParamNameQ, relatedWords[i]);
}%>
<li><b> <%= relatedWords[i] %> </b>
   <%= UrlUtils.constructHTMLLink(pRequest, UrlUtils.PAGE_LINK,
            "(words related to " + relatedWords[i] + ")",
            ",
            linkParams, 
            true, 
            true) %>
</li>
<%}
</ul>
</center>

7.2.3.3.4 Building Forms with the Portlet URL Types Use of portlet parameters in forms is little different from links. The following two fundamental rules continue to apply:

- Qualify the portlet's parameter names.
- Do not manipulate or remove the other parameters on the incoming URL.

In terms of markup and behavior, forms and links differ quite considerably. However, just as with links, PDK-Java contains utilities for complying with these two basic rules. The code for properly qualifying the portlet's parameter name is the same as described in Section 7.2.3.3.3, "Building Links with the Portlet URL Types". After all, a parameter name is just a string, whether it be a link on a page or the name of a form element.

Forms differ from links in the way you ensure that the other parameters in the URL remain untouched. Once you open the form in the markup, you can make use of one of the following APIs:

UrlUtils.htmlFormHiddenFields(pRequest,UrlUtils.PAGE_LINK, formName);
UrlUtils.htmlFormHiddenFields(someURL);

where formName = UrlUtils.htmlFormName(pRequest,null).
The `htmlFormHiddenFields` utility writes HTML hidden form elements into the form, one form element for each parameter on the specified URL that is not owned by the portlet.

```html
<INPUT TYPE="hidden" name="paramName" value="paramValue">
```

Thus, the developer needs only to add their portlet's parameters to the form.

The other item of which you need to be aware is how to derive the submission target of your form. In most cases, the submission target is the current page:

```java
formTarget = UrlUtils.htmlFormActionLink(pRequest,UrlUtils.PAGE_LINK)
```

The value of `formTarget` can be the action attribute in an HTML form or the target attribute in a `SimpleForm`. Even though the method name includes HTML, it actually just returns a URL and thus you can use it in mobile portlets, too.

The following example form renders the thesaurus portlet's submission form. Refer to the example in Section 7.2.3.3.3, "Building Links with the Portlet URL Types" for the portlet that results from the submission of this form.

```java
<% String paramNameSubmit = "submit";
    String paramNameQ = "q";
    String qualParamNameQ = HttpPortletRendererUtil.portletParameter(paramNameQ);
    String qualParamNameSubmit = HttpPortletRendererUtil.portletParameter(paramNameSubmit);
    PortletRenderRequest pRequest = (PortletRenderRequest) request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
    String formName = UrlUtils.htmlFormName(pRequest,"query_form");%>

<!-- Output the HTML content -->
<center>
<b>Thesaurus</b>
Enter the word you wish to search for
<form name="" method="POST">
  action="" target="">
    <input type="text" size="20" name="" value="">
  </input>
</form>
</center>
```

**Note:** Just as parameters in URLs and element names in forms require qualification to avoid clashing with other portlets on the page, form names must be fully qualified because any given page might have several forms on it.
7.2.3.3.5 Implementing Navigation within a Portlet  You can implement navigation within a portlet in one of the following three ways:

- Pass navigation information in rendered URLs using explicit portlet parameters. Branching logic within the portlet code then determines which section of the portlet to render based on the URL. This option represents a small extension to the thesaurus example presented in Section 7.2.3.3.3, "Building Links with the Portlet URL Types" and Section 7.2.3.3.4, "Building Forms with the Portlet URL Types". Basically, instead of performing thesaurus search operations using the value of parameter \( q \), the portlet branches based on the parameter value and renders different content accordingly.

- Pass navigation information as described in the previous item but use PDK-Java to interpret the parameter and thus branch on its value. This option requires some further changes to the thesaurus example and is more fully explained later.

- Use session storage to record the portlet state and URL parameters to represent actions rather than explicit navigation. This method provides the only way that you can restore the portlet to its previous state when the user navigates off the page containing the portlet. Once the user leaves the page, all portlet parameters are lost and you can only restore the state from session storage, assuming you previously stored it there. This option requires that you understand and implement session storage. Refer to Section 7.2.5.2, "Implementing Session Storage" for more information about implementing session storage.

The following portlet code comes from the multi-page example in the sample provider of PDK-Java:

```xml
<portlet>
  <id>11</id>
  <name>Multipage</name>
  <title>MultiPage Sample</title>
  <shortTitle>MultiPage</shortTitle>
  <description>
    This portlet depicts switching between two screens all in the context of a Portal page.
  </description>
  <timeout>40</timeout>
  <timeoutMessage>MultiPage Sample timed out</timeoutMessage>
  <renderer class="oracle.portal.provider.v2.render.RenderManager">
    <contentType>text/html</contentType>
    <showPage>/htdocs/multipage/first.jsp</showPage>
    <pageParameterName>next_page</pageParameterName>
  </renderer>
</portlet>
```

Notice that the value of pageParameterName is the name of a portlet parameter, next_page, that the PDK framework intercepts and interprets as an override to the value of the showPage parameter. If the PDK framework encounters the qualified version of the parameter when the multi-page portlet is requested, it will render the resource identified by next_page rather than first.jsp. Note that the PDK does not render the parameter within the portlet, that responsibility falls to the portlet.

You can modify the thesaurus example to operate with the use of this parameter. Specifically, you can use the form submission portlet to be the input for the thesaurus (the first page of the portlet), then navigate the user to the results page, which contains links to drill further into the thesaurus. The following examples illustrate these changes.
Enhancing PDK-Java Portlets

**Note:** The example that follows is most useful for relatively simple cases, such as this thesaurus example. If your requirements are more complex (for example, you want to build a wizard experience), then you should consider using an MVC framework such as Struts. For information on how to build portlets from struts applications, refer to Section 7.3, "Building Struts Portlets with Oracle JDeveloper".

**ThesaurusForm.jsp:**

```<%  
  PortletRenderRequest pRequest = (PortletRenderRequest)  
    request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);  
  String paramNameSubmit = "submit";  
  String paramNameQ = "q";  
  String qualParamNameQ =  
    HttpPortletRendererUtil.portletParameter(pRequest, paramNameQ);  
  String qualParamNameSubmit =  
    HttpPortletRendererUtil.portletParameter(pRequest, paramNameSubmit);  
  String formName = UrlUtils.htmlFormName(pRequest,"query_form");  
%>

<!-- Output the HTML content -->
<center>
<b>Thesaurus</b>
Enter the word you wish to search for
<form name="<%= formName %>"> method="POST"
  action="<%= UrlUtils.htmlFormActionLink(pRequest,UrlUtils.PAGE_LINK) %>"
  "<%= UrlUtils.htmlFormHiddenFields(pRequest,UrlUtils.PAGE_LINK, formName) %>

  <!--next_page is implicitly set to ThesaurusLink.jsp if you don't set it explicitly-->
  <input type=submit name="<%= qualParamNameSubmit %>
     Value="Search">
 </form>
</center>

Notice how `next_page` must be explicitly set to point to ThesaurusLink.jsp. If you do not explicitly set `next_page` in this way, it defaults to the resource registered in `provider.xml`, which is ThesaurusForm.jsp.

**ThesaurusLink.jsp:**

```<%  
  PortletRenderRequest pRequest = (PortletRenderRequest)  
    request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);  
  String paramNameQ = "q";  
  String paramNameNextPage = "next_page";  
  String qualParamNameQ =  
    HttpPortletRendererUtil.portletParameter(pRequest, paramNameQ);  
  String qualParamNameNextPage =  
    HttpPortletRendererUtil.portletParameter(pRequest, paramNameNextPage);```
String paramValueQ = pRequest.getQualifiedParameter(paramNameQ);

<!-- Output the HTML content -->
<center>
Words similar to <%= paramValueQ %>
<br>
Click the link to search for words related to that word.
<br>
<ul>
<% Thesaurus t = new Thesaurus();
String[] relatedWords = t.getRelatedWords(paramValueQ);
NameValue[] linkParams = new NameValue[2];
linkParams[0] = new NameValue(
    qualParamNameNextPage, "./htdocs/ThesaurusLink.jsp");
for (int i=0; i<relatedWords.length; i++)
{
    linkParams[1] = new NameValue(
        qualParamNameQ, relatedWords[i]);
}<%= UrlUtils.constructHTMLLink(pRequest, UrlUtils.PAGE_LINK, "(words related to " + relatedWords[i] + ")",
    true, true) %>
</li>
%
</ul>
<a href="<%=XMLUtil.escapeXMLAttribute(pRequest.getRenderContext().getPageURL())%>"
    Reset Portlet
</a>
</center>

Partial Page Refresh
You should also note that portlets can refresh themselves without refreshing the entire page. For example, in the multi-page portlet sample, firstpage.jsp uses the API to specifically enable the portlet to link to and display the second page without refreshing the entire portal page.
linkParams[0] = new NameValue(HttpPortletRendererUtil.portletParameter(request, "next_page"), "/htdocs/multipage/second.jsp");

>Hello, this is the first page<p>
<%=UrlUtils.constructHTMLLink(prr, UrlUtils.REFRESH_LINK, "second page", ",
linkParams, true, true)%></center>

7.2.3.4 Submitting Events

In the previous section, you created a portlet that received parameters. Now you will create a portlet that passes parameters and events to other portlets on the same page or a different page. Some portlets, like the Simple Parameter Form in OmniPortlet, provide an easy, declarative interface to create a simple form to pass parameters to other portlets. If you want complete control over the events passed and the look of your portlet, though, you can add events to your Java portlet.

The Portlet Wizard does not create all of the code needed to pass parameters to other portlets. The wizard updates the tags in provider.xml and requires that you add the necessary business logic to your JSP code. To create a portlet that uses events, you perform the following tasks:

- Create a new portlet with the Portlet Wizard.
- Add code to your JSP page.
- Map this portlet’s parameters to the portlet you created in Section 7.2.3.2, "Adding Parameters to Your Portlets".

Creating an Events Portlet

To create an events portlet, perform the following steps:

1. Create a new portlet called MyEventsPortlet in the same provider you used for the parameter portlet in Section 7.2.3.2, "Adding Parameters to Your Portlets" by invoking the Portlet Wizard. Go through the wizard as normal. In step 5 of the wizard, create a parameter. In step 6 of the wizard (see Figure 7–5), enter the information shown in Table 7–1.

<table>
<thead>
<tr>
<th>Events Area</th>
<th>Name</th>
<th>Display Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Events Exposed</td>
<td>MyEvent</td>
<td>My Event</td>
<td>This is my event.</td>
</tr>
<tr>
<td>Parameters Associated</td>
<td>MyParam</td>
<td>My Parameter</td>
<td>This is my parameter</td>
</tr>
</tbody>
</table>
The wizard generates the following code in `provider.xml`:

```xml
<showDetails>false</showDetails>
<inputParameter class="oracle.portal.provider.v2.
    DefaultParameterDefinition">
    <name>MyParam</name>
    <displayName>My Parameter</displayName>
</inputParameter>
<event class="oracle.portal.provider.v2.DefaultEventDefinition">
    <name>MyEvent</name>
    <displayName>My Event</displayName>
    <parameter class="oracle.portal.provider.v2.DefaultParameterDefinition">
        <name>MyParam</name>
        <displayName>My Parameter</displayName>
    </parameter>
</event>
<renderer class="oracle.portal.provider.v2.render.RenderManager">

For more information on the syntax of `provider.xml`, refer to the provider JavaDoc on OTN:


2. Import the following classes:
   - `oracle.portal.provider.v2.event.EventUtils`
   - `oracle.portal.utils.NameValue`
   - `oracle.portal.provider.v2.url.UrlUtils`

3. Add a link that passes the parameter value to another portlet. As shown in the following sample code, you receive the same page parameter as the previous portlet, but in addition you create a link that passes an event as well:
4. Add the portlet to a different page (in the same page group) than the previous portlet (the Parameter Portlet). Expand the portlet and wire it to receive the same parameter as the previous portlet.

My Parameter = Page Parameter MyParameter

5. Apply your changes on the Parameter tab and go to the Events tab. Expand the Event portlet and select the event. Select Go to Page and find the page to which you want to pass the event. Choose the page where the Parameter portlet is located. Configure this portlet to pass an event as the page parameter as shown in Figure 7–6.

MyParameter = Event Output MyParameter

---

**Note:** This sample code does not handle NULL values. When the portlet is initially added to the page, you may receive an error, but, after wiring the portlet to the page parameter, it should work fine.
Figure 7–6  Portlet Events in the Edit Page

Portlet Events

Portlet Events

Event: My Event

My Event Portlet

My Event

My NLS Portlet

Set Language

Save

Masterclass for PDTS

My Parameter

My Event Portlet

My Event

My2 Portlet

My1 Portlet

Go to page: Masterclass for PDTS

Reload page: Masterclass for PDTS

Pass data to input parameters of page Masterclass for PDTS:

Page Input

stock_symbol = Null

stock = Null

MyParameter = Event Output

OK

Cancel

6. Click OK to view the page. Your Event portlet (see Figure 7–7) should have a link that displays the value received from the page.

Figure 7–7  My Event Portlet Before Parameter Change

My Event Portlet

The value of the stock is My Default Value

7. You can append a parameter value to the URL and the portlet displays the value in the link.

&MyParameter=20

8. When you click the link, that value is passed to the Parameter portlet on its page (see Figure 7–8).

Figure 7–8  My Event Portlet After Parameter Change

My Event Portlet

The value of the stock is 20
7.2.4 Using JNDI Variables

When writing Java portlets, you may set deployment specific properties through the JNDI service such that their values may be retrieved from your provider code. In this way, you can specify any property in a provider deployment and then easily access it anywhere in your provider code. PDK-Java provides utilities to enable the retrieval of both provider and non-provider JNDI variables within a J2EE container. To use JNDI variables, you need to perform the following tasks:

- Declaring JNDI Variables
- Setting JNDI Variable Values
- Retrieving JNDI Variables

7.2.4.1 Declaring JNDI Variables

You declare JNDI variables in the web.xml file for your provider. The format for declaring a JNDI variable is as follows:

```xml
<env-entry>
  <env-entry-name>variableName</env-entry-name>
  <env-entry-type>variableType</env-entry-type>
  <env-entry-value>variableValue</env-entry-value>
</env-entry>
```

The `env-entry-name` element contains the name by which you want to identify the variable. `env-entry-type` contains the fully qualified Java type of the variable. `env-entry-value` contains the variable's default value.

7.2.4.1.1 Variable Types

In the `env-entry-type` element, you should supply the fully-qualified Java type of the variable, which will be expected by your Java code. The Java types you may use in your JNDI variables are as follows:

- `java.lang.Boolean`
- `java.lang.String`
- `java.lang.Integer`
- `java.lang.Double`
- `java.lang.Float`

The J2EE container uses these type declarations to automatically construct an object of the specified type and gives it the specified value when you retrieve that variable in your code.

7.2.4.1.2 Variable Naming Conventions

The PDK-Java defines a number of environment variables that can be set at the individual provider service level or at the Web application level. To avoid naming conflicts between different provider services or different application components packaged in the same Web application, we recommend you devise some naming convention.

---

**Note:** If you use the `EnvLookup` method, you must use `oracle/portal/provider/service/property`. You cannot substitute your own company name or component in this case.

---

For example:

- Provider service specific names should be of the form:
Shared names should be of the form:
{company}/{component name}/{provider name}/{variable name}

where:
- {company} is the name of the company owning the application.
- {component name} is the name of the application or component with which the provider is associated.
- {provider name} is the service name of the provider.
- {variable name} is the name of the variable itself.

As you can see, these naming conventions are similar to those used for Java packages. This approach minimizes the chance of name collisions between applications or application components. PDK-Java provides utilities that allow you to retrieve variables in this form without hard coding the service name of the provider into your servlets or JSPs. The service name need only be defined in the provider’s WAR file. Refer to Section 7.2.4.3, “Retrieving JNDI Variables” for more information on retrieving JNDI variables.

7.2.4.1.3 Examples  The following examples illustrate provider variable names:
oracle/portal/myProvider/myDeploymentProperty
oracle/portal/myprovider/myProperties/myProperty

The following example illustrates non-provider variable names:
oracle/portal/myOtherProperty

7.2.4.2 Setting JNDI Variable Values
In your provider deployment, you may want to set a new value for some or all of your JNDI variables. You can perform this task in one of the following two ways:
- If you are using Oracle Enterprise Manager 10g, you can set the variables from there.
- If you are using a standalone instance of Oracle Application Server Containers for J2EE, then you need to manually change the variable values in the PDK-Java’s orion-web.xml file.

7.2.4.2.1 Setting Values in Oracle Enterprise Manager 10g  To set variable values in Oracle Enterprise Manager 10g, do the following:
1. In Oracle Enterprise Manager 10g, click the instance where you have deployed PDK-Java.
2. Click the name representing this deployment.
3. Under the Web modules heading, find and click jpdk.
4. In the Administration section at the bottom of the page, click Environment.
5. Under the Environment Entries section, all of the environment variables are listed. The default value, if it exists, is listed under the value heading. To update a value for a particular variable, enter the new value in the text box in that variable’s row.
6. When you have updated all of the variable values that you want, click Apply.
7.2.4.2.2 Setting Values Manually  To set variable values manually, do the following:

1. Open the orion-web.xml file in a text editor. If the file does not exist, then you must create it. For a full Oracle Application Server installation, you can locate this file in:

   ORACLE_HOME\j2ee\OC4J_instance\application-deployments\jpdk\jpdk

   For a standalone instance of Oracle Application Server Containers for J2EE, you can locate it in:

   ORACLE_HOME\j2ee\home\application-deployments\jpdk\jpdk

2. For each deployment property you want to set, add the following entry:

   <env-entry-mapping name="jndi_var_name">value</env-entry-mapping>

3. Save and close the file. When complete, your file should look something like the following:

   <?xml version = '1.0'?><!DOCTYPE orion-web-app PUBLIC "-//Evermind//DTD Orion Web Application 2.3//EN" "http://xmlns.oracle.com/ias/dtds/orion-web.dtd"><orion-web-app deployment-version="9.0.3.0.0">
   <env-entry-mapping name="oracle/portal/sample/rootDirectory">
     D:\prefs</env-entry-mapping>
   <env-entry-mapping name="oracle/portal/sample/definition">
     D:\definitions\def.xml</env-entry-mapping>
   </orion-web-app>

7.2.4.3 Retrieving JNDI Variables

JNDI is a standard J2EE technology. As such, you can access JNDI variables through J2EE APIs. For example:

```java
String myVarName = "oracle/portal/myProvider/myVar"
String myVar = null;
try {
  InitialContext ic = new InitialContext();
  myVar = (String)ic.lookup("java:env/" + myVarName);
}
```

In addition to the basic J2EE APIs, PDK-Java includes a simple utility class for retrieving the values of variables defined and used by the PDK itself. These variables all conform to the naming convention described in Section 7.2.4.1.2, "Variable Naming Conventions" and are of the form:

```
oracle/portal/provider_service_name/variable_name
```

To use these APIs, you need only provide the provider_service_name and the variable_name. The utilities construct the full JNDI variable name, based on the information you provide, and look up the variable using code similar to that shown here and return the value of the variable.
The `EnvLookup` class (`oracle.portal.utils.EnvLookup`) provides two `lookup()` methods. One retrieves provider variables and the other retrieves non-provider variables. Both methods return a `java.lang.Object`, which can be cast to the Java type you are expecting.

The following code example illustrates the retrieval of a provider variable:

```java
EnvLookup el = new EnvLookup();
String s = (String)el.lookup(myProviderName, myVariableName);
```

`myProviderName` represents the service name for your provider, which makes up part of the variable name. `myVariableName` represents the portion of the variable name that would come after the provider’s service name. The example assumes the variable being retrieved is of type `java.lang.String`.

To retrieve a non-provider variable, you use the same code, you pass only one parameter, the variable name, to the `lookup()`, again excluding the `oracle/portal` prefix.

```java
EnvLookup el = new EnvLookup();
Object o = el.lookup(myVariableName);
```

Table 7–2 shows the JNDI variables provided by default with PDK-Java. If you do not declare these variables, PDK-Java looks for their values in their original locations (`web.xml` and the deployment properties file).

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>oracle/portal/provider/provider_name/autoReload</code></td>
<td>Boolean auto reload flag. Defaults to true.</td>
</tr>
<tr>
<td><code>oracle/portal/provider/provider_name/definition</code></td>
<td>Location of provider’s definition file.</td>
</tr>
<tr>
<td><code>oracle/portal/provider/global/log/logLevel</code></td>
<td>Log setting (0 through 8). 0 being no logging and 8 the most possible logging.</td>
</tr>
<tr>
<td><code>oracle/portal/provider/provider_name/maxTimeDifference</code></td>
<td>Provider’s HMAC time difference.</td>
</tr>
<tr>
<td><code>oracle/portal/provider/&lt;service_name&gt;/resourceUrlKey</code></td>
<td>Authentication key for resource proxying through the Parallel Page Engine. Refer to Oracle Application Server Portal Configuration Guide for more information.</td>
</tr>
<tr>
<td><code>oracle/portal/provider/provider_name/rootDirectory</code></td>
<td>Location for provider personalizations. No default value.</td>
</tr>
<tr>
<td><code>oracle/portal/provider/provider_name/sharedKey</code></td>
<td>HMAC shared key. No default value.</td>
</tr>
<tr>
<td><code>oracle/portal/provider/provider_name/showTestPage</code></td>
<td>(non-provider) A boolean flag that determines if a provider’s test page is accessible. Defaults to true.</td>
</tr>
</tbody>
</table>
7.2.5 Accessing Session Information

When a user accesses any portal page, OracleAS Portal initiates a public unauthenticated session and maintains a cookie to track information about the session across requests. If the user logs in to OracleAS Portal, this session becomes an authenticated session of the logged-in user. This portal session terminates when:

- The browser session terminates (that is, the user closes all the browser windows).
- The user explicitly logs out.
- The session times out because the user’s idle time exceeds the configured limit.

As part of the metadata generation, OracleAS Portal contacts all of the providers that contribute portlets to the page, if they specify during registration that they get called for some special processing. This call allows providers to do processing based on the user session, log the user in the provider’s application if needed, and establish provider sessions in OracleAS Portal. For Database providers, this call is referred to as do_login and for Web providers it is initSession. Since most Web-enabled applications track sessions using cookies, this API call allows the provider of the application to return cookies.

You can utilize the session store to save and retrieve information that persists during the portal session. This information is only available, and useful, to you during the life of the session. You should store only temporary information in the session store. Application developers may use the session store to save information related to the current user session. Data in the session store can be shared across portlets.

If the information you want to store must persist across sessions, you may want to store it in the preference store instead. Some common applications of the session store are:

- to cache data that is expensive to load or calculate (for example, search results).
- to cache the current state of a portlet (for example, the current range, or page, of search results displayed in the portlet, or sequence of events performed by user).

**Note:** If you need to replicate session state across middle tiers, you must mark the Web application as distributable and create a cluster island for your OC4J servers. For more information, refer to Oracle Application Server Containers for J2EE Servlet Developer’s Guide.

Before you implement session storage, you should carefully consider the performance costs. Because portlets and providers are remote, it can be a relatively expensive operation to create and maintain even a small amount of information in the session.
store. For this reason, you may want to avoid altogether any session storage for public pages that are accessed frequently by many users.

Furthermore, while using the session store with Web providers, you create a stateful application that needs to track state information in memory. Similarly, you create a stateful application if you use the file-system implementation of preference store.

If scalability is an important concern for you, a stateful application may cause you problems. Stateful applications can impact the load-balancing and failover mechanism for your OracleAS Portal configuration. Even though you may deploy multiple middle-tiers accessing the same OracleAS Portal instance, you must implement sticky routing (where the same node handles subsequent requests in the same session) to track state. Sticky routing may result in lopsided load-balancing or loss of session data in case a node crashes, impacting failover. This issue is one reason why many developers prefer to build stateless applications. However, if scalability is not a concern, then a stateful application should present no problems for you.

In the example in this section, session storage is used to count the number of times your portlet has rendered in Shared Screen mode.

7.2.5.1 Assumptions
1. You have followed through and understood Section 6.6.2, "Building PDK-Java Portlets".
2. You built a portlet using the wizard and successfully added it to a page.

7.2.5.2 Implementing Session Storage

The PDK Framework represents the session with a ProviderSession object, which is established during the call to the Provider Instance’s initSession method. This object is associated with the ProviderUser. To make data persistent between requests from OracleAS Portal, you need to write data into the session object using the setAttribute method on the ProviderSession object. This method maps a java.lang.Object to a java.lang.String and stores that mapping inside the session object. The String can then be used to retrieve the Object during a subsequent request, provided the session is still valid.

A provider session may become invalid for the following reasons:

- The session times out.
- The invalidate method on ProviderSession is called.
- The JVM process running the servlet container is terminated.

All portlets contained by the same ProviderInstance share the same session for a particular ProviderUser. Therefore, data unique to a particular portlet instance must be mapped to a unique String in the session. This is accomplished using the portletParameter method in the PortletRendererUtil class. This method makes a supplied String parameter or attribute name unique to a PortletInstance, by prefixing it with a generated identifier for that instance. You can use the returned instance-specific name to write portlet instance data into the session.

For more detailed information on the PDK Framework classes, refer to the JavaDoc on OTN by clicking Java Doc API on the Portlet Development page available at http://www.oracle.com/technology/products/ias/portal/portlet_development_10g1014.html

To implement session storage, you need to perform the following tasks:
Enhancing PDK-Java Portlets

- Import ProviderSession, PortletRendererUtil, and HttpPortletRendererUtil.
- Retrieve the provider session.
- Read and write the session by accessing it from within your Java portlet.
- Set the session to true in provider.xml.
- Register the provider for session storage and set the Login Frequency.

The steps that follow describe how to add a session count to your portlet that displays how many times the portlet has been rendered for the current session.

1. After using the wizard to create a portlet, you can edit the JSP for the Show page in Oracle JDeveloper. You need to import the following classes:

```jsp
<%@page contentType="text/html; charset=windows-1252"
import="oracle.portal.provider.v2.render.PortletRenderRequest"
import="oracle.portal.provider.v2.ProviderSession"
import="oracle.portal.provider.v2.render.http.HttpPortletRendererUtil"
%
```

2. Insert code that checks for a valid session first and then increments the count and displays it. If the session is valid and a previously stored value exists, you display the value, increment the count, and store the new value. If the session is valid but no previously stored value exists, you initialize a new count starting with 1, and display and store the value. You also want to obtain the unique string key for this portlet and then use an it in an array to count the session. If no session information was received, you want to provide information to the user indicating they may need to log back in.

```jsp
<% PortletRenderRequest pReq = (PortletRenderRequest) request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST); ProviderSession pSession = pReq.getSession(); if (pSession != null) {
    String key = PortletRendererUtil.portletParameter(pReq, "count");
    Integer i = (Integer) pSession.getAttribute(key);
    if (i == null) {
        i = new Integer(0);
    } else {
        i = new Integer(i.intValue()+1);
    }
    pSession.setAttribute(key, i);
%>

<p>Render count in this session: <%=i%> </p>
```

```jsp
else {
%>

<p>The session has become invalid</p>
<br>
Please log out and log in again.
<%
```
3. By default, the wizard does not set session to true in provider.xml. You need to update this flag in order for the provider to receive session information from the portal. You should only set this tag to true if you are using session information in your provider or portlets. By setting this flag to true, extra load is added to the provider calls.

```xml
<provider class="oracle.portal.provider.v2.DefaultProviderDefinition">
  <session>true</session>
</provider>
```

For more information on the syntax of provider.xml, refer to the provider JavaDoc on OTN:


4. Register the provider in OracleAS Portal. Ensure that you select the User radio button and choose a Login Frequency of Once Per Session on the Define Connections page of the wizard. For a reminder on how to register your portlet, refer to Section 6.6.2.5, "Registering and Viewing Your Portlet".

7.2.5.3 Viewing the Portlet

If you have not already added your Java portlet to a page, do so now. Ensure that you perform the following tasks:

- Set your provider to Once per User Session for the login frequency value.
- Refresh the provider to accept the new changes.
- Re-login in case your session is no longer valid.

7.2.6 Implementing Portlet Security

This section describes the available security services for your Java portlet.

For more detailed information about the PDK classes referred to in this section, refer to the JavaDoc on OTN by clicking Java Doc API on the Portlet Development page available at

http://www.oracle.com/technology/products/ias/portal/portlet_development_10g1014.html

7.2.6.1 Assumptions

1. You have followed through and understood Section 6.6.2, "Building PDK-Java Portlets".

2. You built a portlet using the wizard and successfully added it to a page.

7.2.6.2 Introduction to Portlet Security Features

This section introduces the major features that are available to secure your portlet providers.

7.2.6.2.1 Authentication When a user first logs in to an OracleAS Portal instance, they must enter their password to verify their identity and obtain access. This authentication is performed by OracleAS Single Sign-On server. Refer to Section 7.2.6.3, "Single Sign-On" for more information.
Once the user is authenticated to OracleAS Portal, the provider code has access to the authenticated user's identity from the PortletRenderRequest that is available from the HttpServletRequest object as follows:

```java
PortletRenderRequest pr = (PortletRenderRequest)request.getAttribute
    (HttpCommonConstants.PORTLET_RENDER_REQUEST);
String userName = pr.getUser().getName();
```

When using this user identity for sensitive operations, it is important to ensure that you have configured your OracleAS Portal and this provider to use basic message authentication to secure the integrity of the identity assertion.

You also have the option of configuring the OC4J containing the JSDK provider to use JAZN-LDAP and then use J2EE security, in which case you can obtain the user's identity with the following method, again through the HttpServletRequest object:

```java
String userName = request.getRemoteUser();
```

And if you need the user Principal, you can also use:

```java
Principal userPrincipal = request.getUserPrincipal();
```

When configuring the provider's OC4J container in this manner, make sure to configure your OracleAS Portal and this provider to use enhanced message authentication to secure the integrity of this form of identity assertion.

Refer to Oracle Application Server Portal Configuration Guide for more information.

### 7.2.6.2.2 Authorization

Authorization determines if a particular user may view or interact with a portlet. OracleAS Portal provides the following three types of authorization checking:

- **Portal Access Control Lists (ACLs):** After a user is authenticated by OracleAS Single Sign-On, OracleAS Portal uses ACLs to determine what privileges that user has to perform actions on portal objects, such as folders and portlets. The actions available to a user can range from simply viewing an object to performing administrative functions on it. If a user does not belong to a group that has been granted a specific privilege, OracleAS Portal prevents that user from performing the actions associated with that privilege. Refer to Section 7.2.6.4, "OracleAS Portal Access Control Lists (ACLs)" for more information.

- **Programmatic Portlet Security:** You can also implement your own security manager programmatically. Refer to Section 7.2.6.5, "Portlet Security Managers" for more information.

- **J2EE Security Roles:** You can use J2EE programmatic security in your provider code. To leverage this capability, you must configure your provider for enhanced authentication to protect the integrity of the asserted identity. From within your portlet code, you can use `request.isUserInRole("securityrole")`, where `request` is the HttpServletRequest request and `securityrole` is a declared J2EE security role. Refer to the Oracle Application Server Portal Configuration Guide for more information.

### 7.2.6.2.3 Communication Security

To this point, we have covered user authentication and authorization, which do not check the authenticity of messages received by a provider. To completely secure your providers, secure the communication between OracleAS Portal and a Web provider. (These methods do not apply to database providers, which execute within the OracleAS Portal database.) If the communication is not secured, it is possible for someone to imitate an OracleAS Portal instance and
fool the Web provider into returning sensitive information. There are three types of communication security:

- **OracleAS Portal Server Authentication** restricts access to a provider to a small number of recognized machines. This method compares the IP address or the host name of an incoming HTTP message with a list of trusted hosts. If the IP address or host name is in the list, the message is passed to the provider. If not, it is rejected before reaching the provider. Refer to Section 7.2.6.6, "OracleAS Portal Server Security" for more information.

- **Message Authentication** appends a checksum based on a shared key to provider messages. When a message is received by the provider, the authenticity of the message is confirmed by calculating the expected value of the checksum and comparing it with the actual value received. If the values are the same, the message is accepted. If they are different, the message is rejected without further processing. The checksum includes a time stamp to reduce the chance of a message being illegally recorded in transit and resent later. Refer to Section 7.2.6.7, "Message Authentication" for more information.

- **Message Encryption** relies on the use of the HTTPS protocol for communication between OracleAS Portal and providers. Messages are strongly encrypted to protect the data therein. Encryption provides a high level of security, but it incurs a performance penalty due to the additional processing required for each message. Refer to Section 7.2.6.8, "HTTPS Communication" for more information.

For more information about communication security, refer to the Oracle Application Server Portal Configuration Guide.

### 7.2.6.3 Single Sign-On

Portlets act as windows into an application. They display summary information and provide a way to access the full functionality of the application. Portlets expose application functionality directly in the portal or provide deep links that take you to the application itself to perform a task.

An application may need to authenticate the user accessing the application through the portlet. The following are the possible application authentication methods:

- **Partner Application.** In this case, the application user is the same authenticated user used by OracleAS Portal.

- **External Application.** In this case, the OracleAS Portal user is different from the application user, but the application user name and password are managed by the OracleAS Portal user.

- **No Application Authentication.** In this case, the communication between provider and OracleAS Portal is not protected at all.

For more information about Single Sign-On, refer to the Oracle Application Server Portal Configuration Guide.

### 7.2.6.3.1 Partner Application

A partner application is an application that shares the same OracleAS Single Sign-On as OracleAS Portal for its authentication. Thus, when a user is already logged in to OracleAS Portal, their identity can be asserted to the partner application without them having to log in again.

Partner applications are tightly integrated with OracleAS Single Sign-On. When a user attempts to access a partner application, the partner application delegates the authentication of the user to OracleAS Single Sign-On. Once a user is authenticated (that is, has provided a valid user name and password) for one partner application, the user does not need to provide a user name or password when accessing other partner
applications that share the same OracleAS Single Sign-On instance. OracleAS Single Sign-On determines that the user was successfully authenticated and indicates successful authentication to the new partner application.

The advantages of a partner application implementation are as follows:

- Provides the tightest integration with OracleAS Portal and OracleAS Single Sign-On Server.
- Provides the best single sign-on experience to users.
- Provides the most secure form of integration because user names and passwords are not transmitted between OracleAS Portal and the provider.

The disadvantages of a partner application implementation are as follows:

- The application must share the same user repository as OracleAS Portal even though the application’s user community may be a subset of the OracleAS Portal user community. While worth some consideration, this issue is a minor one because the portal pages that expose the application can be easily restricted to the application’s user community.
- The application can only be tightly integrated to one or more OracleAS Single Sign-On instances if they share the same user repository.
- The application must be written such that it delegates authentication to OracleAS Single Sign-On.
- You must have access to the application source code.

7.2.6.3.2 External Application An external application uses a different authentication server than OracleAS Portal. The application may use a different instance of OracleAS Single Sign-On than that used by OracleAS Portal or some other authentication method. However OracleAS Single Sign-On does store the user name and password of the external application for that user. This means that when a user is already logged into OracleAS Portal, they will be logged into the external application without having to type in their user name or password.

Applications that manage the authentication of users can be loosely integrated with OracleAS Single Sign-On if the administrator registers them as external applications. When a user who was previously authenticated by OracleAS Single Sign-On accesses an external application for the first time, OracleAS Single Sign-On attempts to authenticate the user with the external application. The authentication process submits an HTTP request that combines the registration information and the user’s user name and password for the application. If the user has not yet registered their user name and password for the external application, OracleAS Single Sign-On prompts the user for the required information before making the authentication request. When a user supplies a user name and password for an external application, OracleAS Single Sign-On maps the new user name and password to the user’s OracleAS Portal user name and stores them. They will be used the next time the user needs authentication with the external application.

The advantages of an external application implementation are as follows:

- Allows integration with many portals. If, however, one of the portals is preferred over the others, the application could be integrated as a partner application of that preferred portal and an external application of the others.
- Provides a single sign-on experience for users. However, users still must maintain different user names and passwords. In addition, the external application user name mapping must be maintained.
Enhancing PDK-Java Portlets

- Allows integration with multiple portals independent of their user repositories and OracleAS Single Sign-On.
- Avoids the requirement of having access to the application source code.

The disadvantages of an external application implementation are as follows:
- Does not share the same user repository as the portal, which requires additional maintenance of user information by the end user.
- Transmits the user name and password to the provider in plain text, unless you implement SSL.

7.2.6.3.3 No Application Authentication

The provider trusts the OracleAS Portal instance sending the request completely. The provider can determine if the user is logged in and the portal user name, but the application has not authenticated the user.

The advantages of no application authentication are as follows:
- Provides the easiest form of integration and the fastest to implement.

The disadvantages of no application authentication are as follows:
- Provides the least security.
- Provides the weakest integration with OracleAS Portal.

7.2.6.4 OracleAS Portal Access Control Lists (ACLs)

When users log in to an OracleAS Portal instance, they are authenticated by an OracleAS Single Sign-On instance. Having verified their identity, OracleAS Portal uses ACLs to determine whether they are authorized to access particular portlets and add them to pages from the Portlet Repository.

OracleAS Portal ACLs operate according to the following security characteristics:
- **Privileges** define the actions that can be performed on the object to which they are granted. Privileges include actions such as Manage and Execute.
- **OracleAS Portal users and their privileges** are granted from the Administer tab of the Builder.
- **OracleAS Portal user groups** are administered from the Administer tab of OracleAS Portal Builder. Membership in the groups and privileges granted to the groups are all defined and maintained here. A privilege granted to a user group is inherited by all the users of that group.
- **Provider privileges** apply to the provider and all of its portlets. Provider ACLs are administered on the Provider tab of the OracleAS Portal Navigator.
- **Portlet privileges** can override the privileges set for the provider of the portlet. Portlet ACLs are administered from the Provider tab of the OracleAS Portal Navigator. Clicking **Open** for a provider takes you to a page that manages the portlets of the provider.

For more information on the available privileges for objects, users, and user groups in OracleAS Portal, refer to the *Oracle Application Server Portal Configuration Guide*.

The advantages of ACLs are as follows:
- ACLs offer a simple, yet powerful, mechanism to secure OracleAS Portal objects.
- Central management of user group membership simplifies the management of ACLs because it negates the necessity of modifying the ACLs associated with each object.
The disadvantages of ACLs are as follows:

- ACLs are applied at the provider or portlet level. You cannot vary the security rules for a portlet depending on the page where you place it.

### 7.2.6.5 Portlet Security Managers

Portlet security managers are implemented within a provider to verify that a given user may view an instance of the portlet. When a user views a page with a portlet instance on it, security managers determine whether the user has the appropriate privileges to see the portlet. Implementing access control methods in the provider restricts the retrieval of content from a portlet (that is, hides the portlet) from users without the appropriate privileges. Only if the specified characteristics, such as user details and preferences, pass the authorization logic will the content be retrieved for the user. If no portlet security methods are implemented in the provider, then any user name may be passed in, even fictitious, unauthenticated ones.

A provider can implement two portlet security methods:

- Get a list of portlets.
- Verify the accessibility of the portlet.

Portlets have access to the OracleAS Portal user privileges and groups of which the user is a member. The following information can be used by the security methods:

- The default group of the user
- The privileges of a user or group
- The highest available privilege of a user across all groups
- The objects the user can access (only in database providers)

AuthLevelSecurityManager has access to the following information about authorization level:

- Strongly authenticated.
  The user has been authenticated by OracleAS Single Sign-On in the current OracleAS Portal session, that is, the user logged in with a valid user name and password, and requested the portlet in the context of that session.

- Weakly authenticated.
  A user who was previously strongly authenticated returns to view a page without an active OracleAS Portal session. A persistent cookie (maintained by the user’s browser) indicates that in some previous session the user logged on with a valid user name and password.

- Public or not authenticated.
  The user has not logged in within the context of the current OracleAS Portal session, and does not have a persistent cookie to indicate that such a state previously existed.

To incorporate these security services into your Java portlet, you simply need to update provider.xml and set the security level to strong, weak, or public. Place the following XML right above the </portlet> tag in provider.xml:

```xml
<securityManager class="oracle.portal.provider.v2.security.AuthLevelSecurityManager">
  <securityLevel>strong</securityLevel>
</securityManager>
```

After you make this change to provider.xml, refresh the provider.
For more information on the syntax of provider.xml, refer to the provider JavaDoc on OTN:


The advantages of security methods are as follows:

- You can enable a portlet to produce different output depending on the level of authorization.

The disadvantages of security methods are as follows:

- Most security manager implementations will use the authorization level or some other user specific element in an incoming message. A check of this type could be bypassed by an entity imitating an OracleAS Portal instance.

7.2.6.5.1 Viewing the Portlet After you add a security manager to a portlet, you can validate it by following these steps:

1. Ensure you are logged in to an OracleAS Portal instance with privileges to create pages and add portlets to a page.
2. Create a new portal page, ensuring it is visible to PUBLIC.
3. Add your Java portlet to the page.
4. Make a note of the direct URL to your new Portal page.
5. Now log out of the Portal instance by clicking the Logout link.
6. Directly access the Portal page by entering the URL noted in Step 4 into your browser's address bar.

You will see the page created in Step 2 but not the portlet added in Step 3. When you added the portlet to the page, you were logged in and hence strongly authenticated. The PDK runtime detected this and allowed you to add the portlet. When you logged out and viewed the page, you were no longer strongly authenticated and hence the PDK Framework did not allow rendering of the portlet's contents.

If you log in again and view the page, you will see that the portlet is still there.

7.2.6.5.2 Implementing Your Own Security Manager If your portlet requires special security arrangements which are not provided by the implementations shipped with the PDK, you will need to supply your own custom PortletSecurityManager controller class. To do this, extend the oracle.portal.provider.v2.security.PortletSecurityManager class and supply implementations for the two methods specified by the interface. Then replace the class attribute of the securityManager controller element in the XML provider definition with you new class name and configure child elements appropriately.

7.2.6.6 OracleAS Portal Server Security

One way to prevent unauthorized access to providers is to restrict access to the provider to known client machines at the server level. Because only the identified clients may access the provider, this method defends against denial of service attacks.

In Oracle Application Server, you use the allow and deny directives in the httpd.conf file to control access to client machines based on their host names or IP addresses. If host names are used as discriminators, the server needs to look them up on its Domain Name Server (DNS), which adds extra overhead to the processing of
each request. Using the IP address circumvents this problem, but the IP address of a remote client may change without warning.

The advantages of server security are as follows:

- It limits access to the provider to trusted hosts only.
- It simplifies configuration.

The disadvantages of server security are as follows:

- OracleAS Web Cache does not have IP address checking capability. If OracleAS Web Cache sits in front of a provider, you have no protection from a client on any host sending show requests to OracleAS Web Cache.
- Restricting access to certain IP addresses and host names may be circumvented by sending messages to a provider containing fake IP addresses and host names. This trick is difficult to perform effectively since return messages go to the machine whose IP address was copied, but it can still cause problems.

For more information on this topic, refer to the Oracle Application Server Portal Configuration Guide.

### 7.2.6.7 Message Authentication

PDK-Java supports message authentication so that access may be limited to a specified provider instance or group of provider instances. A provider is registered with a secret shared key known only to OracleAS Portal and provider administrators.

OracleAS Portal sends a digital signature, calculated using a Hashed Message Authentication Code (HMAC) algorithm, with each message to a provider. A provider may authenticate the message by checking the signature using its own copy of the shared key. This technique may be used in Secure Socket Layer (SSL) communication with a provider instead of client certificates.

OracleAS Portal calculates a signature based on user information, a shared key and a time stamp. The signature and time stamp are then sent as part of the SOAP message. The time stamp is based on UTC (coordinated universal time, the scientific name for Greenwich Mean Time) so that timestamps can be used in messages between computers in different time zones.

When the provider receives this message it generates its own copy of the signature. If the signatures agree, it will then compare the message time stamp with the current time. If the difference between the two is within an acceptable value, the message is considered authentic and is processed accordingly.

A single provider instance cannot support more than one shared key because it could cause security and administration problems. For instance, if one copy of the shared key is compromised in some way, the provider administrator has to create a new key and distribute it to all of the OracleAS Portal clients, who then must update their provider definitions. The way around this problem is to deploy different provider services, specifying a unique shared key for each service. Each provider service has its own deployment properties file so that each service is configured independently of the others. The overhead of deploying multiple provider services within the same provider adapter is relatively small.

In a provider without OracleAS Web Cache in front of it, this use of the same signature cookie over the lifetime of a provider session implies a tradeoff between performance and the security provided by authenticating the requests. The signature cookie value is only calculated once after the initial SOAP request establishes the session with the provider. The shorter the provider session timeout, the more often a signature will be
calculated providing greater security against a show request being resent illegally. However, the SOAP request required to establish a session incurs a time penalty.

In a provider using OracleAS Web Cache to cache show request responses, you have a similar tradeoff. Cached content is secured in the sense that incoming requests must include the signature cookie to retrieve it, but caching content for an extended period of time leaves the provider open to illegal show requests.

While the signature element provides protection against interception and resending of messages, it does nothing to prevent interception and reading of message contents. Messages are still transmitted in plain text. If you are concerned about the content of messages being read by unauthorized people, you should use message authentication in conjunction with SSL.

The advantages of message authentication are as follows:
- Ensures that the message received by a provider comes from a legitimate OracleAS Portal instance.

The disadvantages of message authentication are as follows:
- Causes administration problems if a provider serves more than one portal.
- Entails performance implications if made very secure by having a short session timeout.

For more information on this topic, refer to the Oracle Application Server Portal Configuration Guide.

7.2.6.8 HTTPS Communication

Normal communication between OracleAS Portal and a provider uses HTTP, a network protocol that transmits data as plain text using TCP as the transport layer. HTTPS uses an extra secured layer (SSL) on top of TCP to secure communication between a client and a server, making it difficult to intercept and read messages.

Each entity (for example, an OracleAS Web Cache instance) receiving a communication using SSL has a freely available public key and a private key known only to the entity itself. Any messages sent to an entity are encrypted with its public key. A message encrypted by the public key may only be decrypted by the private key so that, even if a message is intercepted by a felonious third party, it cannot be decrypted.

Certificates used to sign communications ensure that the public key does in fact belong to the correct entity. These are issued by trusted third parties, known as Certification Authorities (CA). They contain an entity’s name, public key, and other security credentials and are installed on the server end of an SSL communication to verify the identity of the server. Client certificates may also be installed on the client to verify the identity of a client.

Oracle Wallet Manager manages public key security credentials. It generates public and private key pairs, creates a certificate request to a CA, and installs the certificate on a server.

For more information on this topic, refer to the Oracle Application Server Portal Configuration Guide.

Configuration of SSL

When a provider is registered from an OracleAS Portal instance, only one URL is entered, which means either HTTP or HTTPS may be used but not both.
Each port on each server that may be used to receive SSL messages must have a server-side certificate installed (that is, an OracleAS Web Cache instance) in front of the Web provider and the server that hosts the provider. The certificate installed on a server port ensures that communication between two points is encrypted but does not authenticate the source of a message. Message authentication should be used as well to fully secure communication between a trusted OracleAS Portal instance and a provider.

For more information about SSL configuration for OracleAS Portal, refer to the Oracle Application Server Portal Configuration Guide.

### 7.2.6.9 LDAP (Oracle Internet Directory) Security

PDK-Java uses Portlet Security Managers for LDAP (Oracle Internet Directory) security. PDK-Java uses Oracle Internet Directory as a repository of users, groups, and permissions. It retrieves information about the logged-in user and determines whether the user has the required permissions to view the portlet and data within the portlet. By enabling Oracle Internet Directory security, your providers can:

- Secure portlets based on groups.
- Restrict access to the administrative functions of your portlets (using your own security manager).
- Retrieve all of the user property information stored in the Oracle Internet Directory including first name, last name, title, e-mail, telephone number, groups, and photo.
- Create users and groups for OracleAS Portal.

By default, Oracle Internet Directory security is disabled. You must make a change in the deployment properties file for a specific provider to enable this feature. Enabling and using Oracle Internet Directory to secure your portlets can be done quickly and easily:

1. Enable the Oracle Internet Directory manager in the deployment properties file (`provider_name.properties`).
   
   ```
   oidManager=true
   oidAdminClass=class_that_extends_oracle.portal.provider.v2.oid.OidInfo
   ```

2. Provide the connection information for Oracle Internet Directory by extending the simple class called `OidInfo`.

3. Provide a list of groups that can view your portlet in the provider definition file.

   ```
   <group>cn=group1,cn=groups,dc=us,dc=oracle,dc=com</group>
   ```

Your provider connects to Oracle Internet Directory using the information provided to the `OidInfo` class by you. The portlet accesses Oracle Internet Directory using the credentials provided (for example, user name and password) and performs the specified tasks. We recommend that you create an Oracle Internet Directory user specifically for your provider connection with the minimum set of privileges needed to complete the tasks requested by your portlets. For example, if your portlet only checks group information, do not connect to the Oracle Internet Directory as an administrator.

### 7.2.6.9.1 Implementing Oracle Internet Directory Security

PDK-Java provides a set of default classes specifically for Oracle Internet Directory integration. These classes handle the connection from your portlets to Oracle Internet Directory, enable your portlets to be secured based on OracleAS Portal groups, and provide access to user
property information from within Oracle Internet Directory. The classes used by your Web provider for Oracle Internet Directory integration are as follows:

- `oracle.portal.provider.v2.oid.OidInfo` receives the Oracle Internet Directory connection information provided by the developer and connects to Oracle Internet Directory. When building your own portlets, you should extend this class to send secure connection details from the provider to Oracle Internet Directory.

- `oracle.portal.sample.v2.devguide.oid.UnsafeOidInfo` is an extension of `OidInfo` and provides an easy way to test portlet security. This class is used by the Oracle Internet Directory samples in PDK-Java and parses the deployment properties file for the Oracle Internet Directory connection information (seen subsequently). This class should be used only for testing and development, it is not safe to use in a production scenario.

- `oidManager` is set to false by default. It must be set to true in `provider_name.properties` to enable Oracle Internet Directory. (If you have only one provider in your Web application, ensure that `provider_name.properties` is identical to `_default.properties`.) For example:

```properties
serviceClass=oracle.webdb.provider.v2.adapter.soapV1.ProviderAdapter
loaderClass=oracle.portal.provider.v2.http.DefaultProviderLoader
showTestPage=true
definition=providers/lab_provider/provider.xml
autoReload=true
oidManager=true
oidAdminClass=oracle.portal.sample.v2.devguide.oid.UnsafeOidInfo
oidHost=myhost.mydomain.com
oidPort=oidPort
oidUser=oidUser
oidPasswd=oidPassword
```

- `oidAdminClass` is set to the class that extends `OidInfo`. PDK-Java provides `UnsafeOidInfo` by default, but as the name suggests, this class should not be used in production scenarios.

  - `oidHost` is the machine where Oracle Internet Directory is hosted.
  - `oidPort` is the port used by the Oracle Internet Directory.
  - `oidUser` is the Oracle Internet Directory account.
  - `oidPasswd` is the Oracle Internet Directory password.

For example:

```properties
serviceClass=oracle.webdb.provider.v2.adapter.soapV1.ProviderAdapter
loaderClass=oracle.portal.provider.v2.http.DefaultProviderLoader
showTestPage=true
definition=providers/lab_provider/provider.xml
autoReload=true
oidManager=true
oidAdminClass=oracle.portal.sample.v2.devguide.oid.UnsafeOidInfo
oidHost=myhost.mydomain.com
oidPort=oidPort
oidUser=oidUser
oidPasswd=oidPassword
```

- `oracle.portal.provider.v2.security.GroupSecurityManager` manages which groups have access to your provider and its portlets. It retrieves this information from the provider definition file and is portlet specific. Each
portlet in a provider may have different group settings. There is no limit on the number of groups that can be set using this tag, but, since the Web provider parses and validates each group in turn, listing many groups may degrade performance.

- `<group>` is the tag in `provider.xml` that handles group management. It lists the groups allowed to access the portlet. The group information here follows the same case sensitivity as the Oracle Internet Directory.

---

**Note:** The following example refers to your `portal_instance_id`, which is specific to your installation. To find your instance identifier, refer to your Oracle Internet Directory Administrator’s Guide.

```xml
<securityManager class="oracle.portal.provider.v2.security.GroupSecurityManager">
    <group>cn=DBA,cn=portal_instance_id,cn=groups,dc=us,dc=oracle,dc=com</group>
</securityManager>
```

For more information on the syntax of `provider.xml`, refer to the provider JavaDoc on OTN:


The advantages of Oracle Internet Directory security are as follows:

- Offers a simple, powerful way to secure your portlets.
- Secures data within your portlets based on the user’s group membership.
- Creates users and groups directly from your portlets exposed as Web providers.

The disadvantages of Oracle Internet Directory security are as follows:

- Slightly degrades performance when authorizing your portlet through Oracle Internet Directory. There is a cost associated with obtaining group information from any LDAP server, but this cost only happens the first time a user accesses a portlet in a session.
- Requires provider access to Oracle Internet Directory.
- Assumes all OracleAS Portal instances served by the provider use the same Oracle Internet Directory instance.

For more information on securing your providers using Oracle Internet Directory or to set up the sample portlets secured using Oracle Internet Directory, review the technical note, *Installing the Oracle Internet Directory Portlets* on OTN.

7.2.6.9.2  **Viewing Your Portlets**  After you secure your provider with Oracle Internet Directory, you can validate its behavior by following these steps:

1. Ensure you are logged in to an OracleAS Portal instance as a user who is a member of the group specified in the `<group>` tag in `provider.xml`.
2. Use an existing page or create a new one, ensuring it is visible to PUBLIC.
3. Add your Java portlet to the page.
4. Make a note of the direct URL to your new page.
5. Click Logout.
6. Directly access the page by entering the URL noted in Step 4 in your browser’s address bar or login to OracleAS Portal using a user that is not part of the group listed in provider.xml.

You will see the page created in Step 2 (see Figure 7–9) but not the portlet added in Step 3. When you added the portlet to the page, you were logged in as a user authorized to view the portlet. The PDK runtime detected this and allowed you to add the portlet. When you logged out and viewed the page, you were no longer part of the group allowed to view the portlet and hence the PDK Framework did not allow rendering of the portlet’s contents.

Figure 7–9 Page and Portlets for Developer

If you log in again and view the page, you will see that the portlet is still there (see Figure 7–10).

Figure 7–10 Page and Portlets for Developer/Administrator

7.2.7 Controlling the Export/Import of Portlet Personalizations

The export/import facility of OracleAS Portal is a multi-purpose tool for moving your portal objects, such as portlets, between instances of OracleAS Portal. For example, you might use export/import to move objects from a development environment to a stage environment and then, finally, to a production environment. You might also use export/import to move pages and page groups between OracleAS Portal instances, or to move Web providers from one machine to another. For more information about
export/import in general, please refer to the Oracle Application Server Portal
Configuration Guide.

Because portlet default settings can be set by the administrator and then changed by
the user, they require some special consideration when you import and export them.
To simplify the transport process, OracleAS Portal provides default functionality that
handles administrator personalization data (that is, data created using Edit Defaults
mode) for you. When a portlet is exported, the default personalization data stored
using PDK-Java’s PreferenceStore mechanism is exported with the portlet by
default. Hence, when the portlet is imported into a target instance of OracleAS Portal,
this data is imported along with it. As a result, the portlet instance’s default settings
are maintained when the portlet is moved from one portal instance to another.1

The aforementioned behavior is provided to you as a convenience and it requires no
action on your part to leverage. You might, however, want to exercise more granular
control over the export of personalization data than that provided by the default
functionality. To implement your own requirements for export/import, you can make
use of the programming interface to augment or override the default handling of
personalizations.

If you use the PDK-Java preference store mechanism, the export/import of your Edit
Default personalizations is built-in and requires no additional effort on your part. This
default export/import of administrator personalizations relies on the PDK-Java
preference store. If you have created your own preference store mechanism (for
example, a file or database preference storage system), then you also must implement
your own export/import support that performs the following functions:

■ Exports personalizations. This functionality must at least export administrator
personalizations, but it could optionally include user personalizations, too.

■ Imports personalizations. Note that this functionality must reflect whatever you
implemented for export. For example, if you allow the export of both
administrator and user personalizations, then the import functionality must
support both as well.

The export/import functionality for personalizations requires that your OracleAS
Portal instance and provider are on Release 10.1.2. Export/import of personalizations
behaves the same regardless of the location of your provider:

■ In the default Oracle Application Server Containers for J2EE of the Oracle
Application Server, where the OracleAS Portal instance is different.

■ In a separate Oracle Application Server Containers for J2EE, where the OracleAS
Portal instance may be different, and the provider is the same but is not registered
on the target OracleAS Portal instance.

7.2.7.1 Import/Export Programming Interface

The PDK-Java’s preference store mechanism allows data to be persisted by any
number of application entities. The following three entities are the ones that persist
data for the purposes of export/import:

1. The portlet instance is the portlet on a page with the default personalizations made
to it by the administrator. The API for the portlet instance is:

   ■ oracle.portal.provider.v2.PortletInstance

      – exportData

      public byte[] exportData

1 User personalization data for OracleAS Portal objects is never exported. This restriction
applies to portlets as well as other objects, such as pages.
2. The *portlet definition* is the base portlet without any personalizations applied to it. You might think of the portlet definition as the version of the portlet that exists in the Portlet Repository before it is placed on a particular page for use. The API for the portlet definition is as follows:

- **oracle.portal.provider.v2.PortletDefinition**
  - **exportData**
    ```java
    public byte[] exportData(
        ProviderInstance pi,
        boolean exportUsers,
        String[] userNames,
        TransportLogger logger
    )
    throws PortletException
    ```
  - **importData**
    ```java
    public void importData(
        ProviderInstance pi,
        byte[] data,
        TransportLogger logger
    )
    throws PortletException
    ```

3. The *provider instance* is the entity that contains and communicates with a set of portlets. The API for the provider instance is:

- **oracle.portal.provider.v2.ProviderInstance**
  - **exportData**
    ```java
    public byte[] exportData(
        boolean exportUsers,
        String[] userNames,
        TransportLogger logger
    )
    throws ProviderException
    ```
  - **importData**
    ```java
    public void importData(
    ```
By default, each of these entities employs an instance of `oracle.portal.provider.v2.transport.PrefStoreTransporter` to transform the data from an `oracle.portal.provider.v2.preference.PreferenceStore` to a byte array for transport. For the default export/import behavior, though, only the portlet instance entity's personalization data is exported and imported. If you have persisted data at the portlet definition or provider instance level, you may want to export that data as well. For example, a billing handle that you persisted at the `ProviderInstance` level may need to be exported.

To change the behavior of `PrefStoreTransporter`, you can override its default implementation. The example in Section 7.2.7.3.7, "Exporting by Reference Example" illustrates how you can override `PrefStoreTransporter`.

**Logging Interface**

To simplify troubleshooting of your export/import transactions, you can send messages to both the calling OracleAS Portal instance and the Web provider log. PDK-Java provides a transport logging class that enables you to add events to the log during export and import operations. In this way, you can better keep track of events that occur during the transport of portlet personalizations. The log can be a valuable troubleshooting tool if you encounter unexpected behavior in your portlets during or after transport. For example, you can log events when incompatibilities between PDK-Java versions are found.

You log events using the logger object, an instance of the `oracle.portal.provider.v2.transport.TransportLogger` class provided for each of the methods mentioned previously. You log events with the calling portal through the instance provided for each method. You record events in the Web provider log with the normal logging mechanism, `oracle.portal.log.LogManager`. The log levels for export/import are as follows:

- `TransportLogger.SEVERITY_INFO`
- `TransportLogger.SEVERITY_WARNING`
- `TransportLogger.SEVERITY_ERROR`

**7.2.7.2 Exporting Personalizations Example**

This example illustrates the most basic case where you build a portlet and accept the default behavior for the export of personalizations. In the examples in Section 7.2.7.3.6, "Encrypting Personalization Data Example" and Section 7.2.7.3.7, "Exporting by Reference Example", you will see how to enhance the security of your personalizations during export and import. To implement the more basic form of exporting personalizations, do the following:

1. Create a stock portlet and implement the Show mode with the following `MyStockPortletShowRenderer.java` class. Note that this class does not incorporate any special code to enable export/import.

```java
package oracle.portal.sample.v2.devguide.tx;
import java.util.StringTokenizer;
import oracle.portal.provider.v2.PortletException;
import oracle.portal.provider.v2.personalize.NameValuePersonalizationObject;
import oracle.portal.provider.v2.render.PortletRenderRequest;
```
import oracle.portal.provider.v2.render.PortletRendererUtil;
import oracle.portal.provider.v2.render.http.BaseManagedRenderer;
import java.io.PrintWriter;
import oracle.portal.sample.v2.devguide.webservices.
NetXmethodsServicesStockquoteStockQuoteServiceStub;
public class MyStockPortletShowRenderer extends BaseManagedRenderer
{
private String pid = null;
private String userdata;
private String stockList;
private String stockCode;
public void renderBody(PortletRenderRequest request) throws PortletException
{
// Use the PrintWriter from the PortletRenderRequest
PrintWriter out = null;
NetXmethodsServicesStockquoteStockQuoteServiceStub ns = new
NetXmethodsServicesStockquoteStockQuoteServiceStub();
try
{
out = request.getWriter();
NameValuePersonalizationObject data = null;
data = (NameValuePersonalizationObject)PortletRendererUtil.
getEditDefaultData(request);
stockList= data.getString("stock");
if(stockList!=null) {
StringTokenizer st = new StringTokenizer(stockList",");
out.println("<table border='0'>");
out.println("<thead>");
out.println("<tr>");
out.println("<th width='20%'>");
out.println("<p align='left'> Stock Code</p></th><th width='20%'>");
out.println("<th width='20%'>");
out.println("<p align='left'> Stock Code</p></th>" );
out.println("<tr>");
out.println("<td width='20%'>");
out.println("<p align='left'> Stock Code</p>" );
}
while(st.hasMoreElements()) {
stockCode= st.nextElement().toString();
out.println("<tr>");
out.println("<td width='20%'>");
out.println("<p align='left'>" + stockCode +
"</p><td width='20%'>");
out.println(ns.getQuote(stockCode));
out.println("<td>");
out.println("<tr>");
}
out.println("</table>");
else
{
out.println("<br> Click <b>Edit Defaults</b> to define stock codes.");
}
}
catch(Exception ioe)
{
throw new PortletException(ioe);
}
}
2. Implement the Edit Defaults mode for your stock portlet with the following class, MyStockPortletEditDefaultsRenderer.java. This class enables the administrator to make and store default personalizations, which are then exported according to the default behavior.

```java
package oracle.portal.sample.v2.devguide.tx;
import oracle.portal.provider.v2.PortletException;
import oracle.portal.provider.v2.render.PortletRenderRequest;
import oracle.portal.provider.v2.render.http.BaseManagedRenderer;
import oracle.portal.provider.v2.render.PortletRendererUtil;
import oracle.portal.provider.v2.personalize.NameValuePersonalizationObject;
import java.io.PrintWriter;
import java.io.IOException;
import oracle.portal.provider.v2.render.http.HttpPortletRendererUtil;
public class MyStockPortletEditDefaultsRenderer extends BaseManagedRenderer {
    public void renderBody(PortletRenderRequest request) throws PortletException {
        PrintWriter out = null;
        try {
            out = request.getWriter();
        } catch (IOException ioe) {
            throw new PortletException(ioe);
        }
        // Personalize the portlet title and stock
        String actionParam = PortletRendererUtil.getEditFormParameter(request);
        PortletRenderRequest prr = (PortletRenderRequest) request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
        String action = request.getParameter(actionParam);
        String title = prr.getQualifiedParameter("myportlet_title");
        String stock = prr.getQualifiedParameter("myportlet_stock");
        NameValuePersonalizationObject data = null;
        try {
            data = (NameValuePersonalizationObject) PortletRendererUtil.getEditDefaultData(request);
        } catch (IOException io) {
            throw new PortletException(io);
        }
        // Cancel automatically redirects to the page, so will only receive OK or APPLY
        if (action != null) {
            data.setPortletTitle(title);
            data.putString("stock", stock);
            try {
                PortletRendererUtil.submitEditData(request, data);
            } catch (IOException ioe) {
                throw new PortletException(ioe);
            }
        }
    }
}
```
return;
}
// Otherwise just render the form

```java
return;
```

```html
<title>Enhancing PDK-Java Portlets</title>

```java
3. Create the following class,
NetXmethodsServicesStockquoteStockQuoteServiceStub.java, for your stock portlet:
```
package oracle.portal.sample.v2.devguide.webservices;
import oracle.soap.transport.http.OracleSOAPHTTPConnection;
import org.apache.soap.encoding.SOAPMappingRegistry;
import java.net.URL;
import org.apache.soap.rpc.Call;
import org.apache.soap.Constants;
import java.util.Vector;
import org.apache.soap.rpc.Parameter;
import org.apache.soap.SOAPException;
import java.util.Properties;
public class NetXmethodsServicesStockquoteStockQuoteServiceStub
{
    public NetXmethodsServicesStockquoteStockQuoteServiceStub()
    {
        m_httpConnection = new OracleSOAPHTTPConnection();
        m_smr = new SOAPMappingRegistry();
    }
    private String _endpoint = "http://64.124.140.30:9090/soap";
    public String getEndpoint()
    {
        return _endpoint;
    }
    public void setEndpoint(String endpoint)
    {
        _endpoint = endpoint;
    }
    private OracleSOAPHTTPConnection m_httpConnection = null;
    private SOAPMappingRegistry m_smr = null;
    public Float getQuote(String symbol) throws Exception
    {
        ...
    }
```
Float returnVal = null;
URL endpointURL = new URL(_endpoint);
Call call = new Call();
call.setSOAPTransport(m_httpConnection);
call.setTargetObjectURI("urn:xmethods-delayed-quotes");
call.setMethodName("getQuote");
call.setEncodingStyleURI(OraclePortalConstants.NS_URI_SOAP_ENC);
Vector params = new Vector();
params.addElement(new Parameter("symbol", String.class, symbol, null));
call.setParams(params);
call.setSOAPMappingRegistry(m_smr);
Response response = call.invoke(endpointURL, "urn:xmethods-delayed-quotes#getQuote");
if (!response.generatedFault())
{
    Parameter result = response.getReturnValue();
    returnVal = (Float)result.getValue();
}
else
{
    Fault fault = response.getFault();
    throw new SOAPException(fault.getFaultCode(), fault.getFaultString());
}
return returnVal;

public void setMaintainSession(boolean maintainSession)
{
    m_httpConnection.setMaintainSession(maintainSession);
}
public boolean getMaintainSession()
{
    return m_httpConnection.getMaintainSession();
}
public void setTransportProperties(Properties props)
{
    m_httpConnection.setProperties(props);
}
public Properties getTransportProperties()
{
    return m_httpConnection.getProperties();
}

4. Create a Web provider through provider.xml for this portlet. Notice the use of the <preferenceStore> element to allow for the storing of personalizations:

```xml
<provider class="oracle.portal.provider.v2.DefaultProviderDefinition">
    <session>false</session>
    <passAllUrlParams>false</passAllUrlParams>
    <preferenceStore class="oracle.portal.provider.v2.preference.FilePreferenceStore">
        <name>prefStore1</name>
        <useHashing>true</useHashing>
    </preferenceStore>
    <portlet class="oracle.portal.provider.v2.DefaultPortletDefinition">
        <id>1</id>
        <name>MyStockPortlet</name>
        <title>My Stock Portlet</title>
        <description>Simple Stock Portlet to show Export and Import feature of web providers</description>
        <timeout>80</timeout>
```
For more information on the syntax of provider.xml, refer to the provider JavaDoc on OTN:


5. Register this export-enabled provider with the source OracleAS Portal instance. For more information about registering Web providers, refer to Section 6.6.2.5, "Registering and Viewing Your Portlet".

Note: If the Web provider is running in a secured environment, remember to provide the proxy host and port while starting up Oracle Application Server Containers for J2EE. For example:

java -Dhttp.proxyHost=www-proxy.us.oracle.com -Dhttp.proxyPort=80 -jar oc4j.jar

6. Create two regions on a sample page and add My Stock Portlet to the first region. For information on creating regions and pages, refer to the Oracle Application Server Portal User’s Guide.

7. Edit the page and click the Edit Defaults icon for My Stock Portlet. Choose the stock codes SUNW, IBM, ORCL. For more information on how to edit defaults for a portlet on a page, refer to the Oracle Application Server Portal User’s Guide.

8. Add My Stock Portlet to a second region and again edit the defaults. Use a different stock code this time, MSFT.

9. Export the page group containing this page. For instructions on how to export a page group, refer to Chapter 10, "Exporting and Importing Content," in the Oracle Application Server Portal Configuration Guide.
10. Import the page group into a target OracleAS Portal instance. For instructions on how to import a page group, refer to Chapter 10, "Exporting and Importing Content," in the Oracle Application Server Portal Configuration Guide.

11. View the page with My Stock Portlet in the target OracleAS Portal instance and ensure that the personalizations were maintained.

7.2.7.3 Implementing Security for Export/Import

Transporting personalizations can present a security concern if your portlet stores sensitive data and is not operating in a secured environment. At the provider and portlet level, OracleAS Portal provides several ways for you to secure the export and subsequent import of portlet personalizations. To better secure portlets and providers for exportation and importation, you can take the following actions:

- **Securing Provider Communications.** Using OracleAS Portal configuration options, you can secure the communications between providers and OracleAS Portal. This step in turn makes the export and import of portlets more secure.

- **Disabling Export/Import of Personalizations.** You can disable the export of all portlet personalization data on an each Web application basis. This method provides the greatest security but only at a significant cost in functionality because it prevents administrators from retaining their default personalizations when the portlet is moved.

- **Obfuscating Data for Transport (Automatic).** By default, OracleAS Portal obfuscates but does not encrypt personalization data before transporting it.

- **Encrypting Personalization Data for Transport.** You may want to encrypt personalization data for transport if any of the following are true:
  - Your Web provider connection is not secured through HTTPS.
  - You want to ensure the data is secured during transit.
  - You want the data to remain secure while stored in the OracleAS Portal instance.

- **Exporting by Reference.** Instead of including portlet personalization data directly in the transport set, you can include it by reference in the transport set. Because the data itself is not present in the transport set, export by reference is the most secure way of transporting personalizations.

7.2.7.3.1 Securing Provider Communications If the security of exporting/importing portlets is of concern to you, you should configure OracleAS Portal to secure communications with your portlet providers. The chief mechanisms for securing provider communications in OracleAS Portal are as follows:

- **Message authentication through a Hashed Message Authentication Code (HMAC) algorithm.** For more information on message authentication for providers, refer to Section 6.1.7.8, "Message Authentication", in the Oracle Application Server Portal Configuration Guide.

- **HTTPS between providers and OracleAS Portal.** For more information on HTTPS for provider communications, refer to Section 6.1.7.9, "HTTPS Communication", in the Oracle Application Server Portal Configuration Guide.

---

**Note:** You cannot use certificates for the HTTPS communication with providers.
7.2.7.3.2 Disabling Export/Import of Personalizations  The JNDI variable, oracle/portal/provider/global/transportEnabled, controls whether to allow the exportation and importation of personalizations. If you set the variable to true, personalizations are transported as part of export and import. If you set it to false, they are not transported. You can set JNDI variables for PDK-Java through Oracle Enterprise Manager 10g. If for some reason Oracle Enterprise Manager 10g is not available for the instance, you can set the values manually in orion-web.xml. For a full Oracle Application Server installation, this file is located in:

ORACLE_HOME/j2ee/OC4J_instance/application-deployments/jpdk/jpdk

For a standalone OC4J installation, this file is located in:

ORACLE_HOME/j2ee/home/application-deployments/jpdk/jpdk

7.2.7.3.3 Obfuscating Data for Transport (Automatic)  By default, personalization data is encoded (Base64). This encoding ensures that data is obfuscated during transport. You do not need to take any actions to leverage Base64 encoding as it is provided by default. However, if you want greater security, you can encrypt the data. Refer to Section 7.2.7.3.4, "Encrypting Personalization Data for Transport".

7.2.7.3.4 Encrypting Personalization Data for Transport  By implementing the oracle.portal.provider.v2.security.CipherManager class for your provider, you can encrypt the personalization data prior to exporting it. Upon import, the cipher manager is invoked again to decrypt the data. Refer to Section 7.2.7.3.6, "Encrypting Personalization Data Example".

Note: If you choose to encrypt your Web providers for export using the cipher manager, you must also devise your own key management strategy for the encryption algorithm.

7.2.7.3.5 Exporting by Reference  As mentioned previously, the default behavior for exporting of portlets is to include the actual personalization data in the transport set. For a more secure transport, you can code your portlet such that the personalizations are exported using pointer rather than by including the actual preference data. When the transport set is imported, the target OracleAS Portal instance sends the pointer back to the Web provider, which then has the opportunity to reassociate the actual data with the new portlet instance. Refer to Section 7.2.7.3.7, "Exporting by Reference Example".

Note: When exporting across security zones, exporting by reference may not work effectively. In general, you should only employ export by reference when transporting within the same general security environment.

7.2.7.3.6 Encrypting Personalization Data Example  To encrypt personalization data in your Web provider, you need to create your own cipher manager and associate it with your portlet provider. This example provides a simple, insecure cipher manager for illustrative purposes only. To implement a secure implementation of the cipher manager for your production system, you would need to significantly extend this sample. Some of the issues you would need to consider for a production implementation are as follows:

- Do not hold the key object in memory. Read it from a persistent store as necessary.
Use the provider's `PreferenceStore` API supported by a `DBPreferenceStore` to work in the clustered case.

On import, if the cipher manager instance obtained from `provider.xml` matches the class name returned in the SOAP message, that `CipherManager` instance is used to perform the decryption. Hence, the instance maintained in the portlet/provider definition may be configured using any applicable means (for example, tags in `provider.xml` or JNDI variable) and that configuration is reused on import.

To encrypt personalization data in your Web provider, do the following:

---

**Note:** The sample provided next is for illustrative purposes only. You would need to significantly enhance it for use in a production environment.

---

1. Create a cipher manager class, `InsecureCipherManager`. This class will be used for encryption and decryption of personalization data exported from or imported to a Web provider. A base64 encoded, hard coded secret key is used with the DES algorithm supplied by the default `javax.crypto` provider of the underlying Java Runtime Environment. As a result, this particular sample is insecure because the encoded key can be recovered by a malicious party simply by decompiling the byte code.

---

**Note:** This sample makes use of the `javax.crypto` package, which is optional in Java 1.3 and must be installed manually. In Java 1.4, though, this package is present by default.

---

code

```java
package oracle.portal.sample.v2.devguide.tx;
import java.io.IOException;
import java.security.GeneralSecurityException;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.DESKeySpec;
import oracle.portal.provider.v2.ProviderException;
import oracle.portal.provider.v2.security.CipherManager;
import sun.misc.BASE64Decoder;
public final class InsecureCipherManager implements CipherManager {
    /**
     * Base64 encoded external form of a javax.crypto.SecretKey which was
     * generated for the DES algorithm. This is completely insecure! Anyone
     * can decompile the bytecode and recostitute the key. A more secure
     * implementation would implement a key management policy in a
     * java.security.KeyStore.
     */
    private static final String sEncodedKey = "UTJds807Arw=";
    /**
     * Generated from the (insecure) encoded form in sEncodedKey.
     */
    private SecretKey mKey;
    /**
     * Transforms the input data to a more secure form, in a single operation,
     * using the DES cryptographic algorithm along with a statically defined
     * secret key.
     */
    public void transformInput
```
public final byte[] encode(byte[] toEncode) throws ProviderException {
    try {
        Cipher c = Cipher.getInstance("DES");
        c.init(Cipher.ENCRYPT_MODE, getSecretKey());
        return c.doFinal(toEncode);
    } catch (GeneralSecurityException gse) {
        throw new ProviderException(gse);
    } catch (IOException ioe) {
        throw new ProviderException(ioe);
    }
}

public final byte[] decode(byte[] toDecode) throws ProviderException {
    try {
        Cipher c = Cipher.getInstance("DES");
        c.init(Cipher.DECRYPT_MODE, getSecretKey());
        return c.doFinal(toDecode);
    } catch (GeneralSecurityException gse) {
        throw new ProviderException(gse);
    } catch (IOException ioe) {
        throw new ProviderException(ioe);
    }
}

private SecretKey getSecretKey() throws GeneralSecurityException, IOException {
    if (mKey == null) {
        DESKeySpec ks = new DESKeySpec((new BASE64Decoder()).decodeBuffer(sEncodedKey));
        SecretKeyFactory skf = SecretKeyFactory.getInstance("DES");
        mKey = skf.generateSecret(ks);
    }
}
2. Modify your provider.xml to reference the cipher manager:

```xml
<?xml version='1.0' encoding='UTF-8'?>
<providerDefinition version="3.1">
  <provider class="oracle.portal.provider.v2.DefaultProviderDefinition">
    <providerInstanceClass>net.mzyg.tx.TxProviderInstance</providerInstanceClass>
    <session>false</session>
    <passAllUrlParams>false</passAllUrlParams>
    <preferenceStore class="oracle.portal.provider.v2.preference.DBPreferenceStore">
      <name>prefStore1</name>
      <connection>java:comp/env/jdbc/PortletPrefs</connection>
    </preferenceStore>
    <cipherManager class="oracle.portal.sample.v2.devguide.tx.InsecureCipherManager"/>
  </provider>
</providerDefinition>
```

7.2.7.3.7 Exporting by Reference Example To export by reference rather than exporting the actual personalization, do the following:

1. Override the DefaultPortletInstance with the following

   ```java
   package oracle.portal.sample.v2.devguide.tx;
   import oracle.portal.provider.v2.DefaultPortletInstance;
   import oracle.portal.provider.v2.preference.PreferenceStore;
   import oracle.portal.provider.v2.transport.PrefStoreTransporter;
   public class ExportByRefDefaultPortletInstance extends DefaultPortletInstance {
     /**
      * Returns a {@link oracle.portal.provider.v2.transport.PrefStoreTransporter}
      * capable of carrying out transport operations such as export/import on
      * data applicable to {@link oracle.portal.provider.v2.PortletInstance}
      * persisted in {@link oracle.portal.provider.v2.preference.PreferenceStore}.
      * This implementation returns an {@link ExportByRefPrefStoreTransporter}.
      * @param ps the {@link oracle.portal.provider.v2.preference.PreferenceStore}
      * containing the data to be transported.
      * @return a {@link oracle.portal.provider.v2.transport.PrefStoreTransporter}
      *          containing the data to be transported.
      */
     protected PrefStoreTransporter getPrefStoreTransporter(PreferenceStore ps) {
       return new ExportByRefPrefStoreTransporter(ps);
     }
   }
   ```

2. Create the ExportByRefPrefStoreTransporter class referenced in ExportByRefDefaultPortletInstance. This class implements an alternative preference store transporter that does not send preference data during the export operation. Instead, it writes the context path of the source preference to the stream. During the export, it reads the context path and uses that path to look up the preference data and copy it to the new instance. This method of exporting by reference assumes that the source and target providers have access to the same preference store. In fact, the best case for this example would be the situation where the source and target providers are the same.

   ```java
   package oracle.portal.sample.v2.devguide.tx;
   import java.io.IOException;
   import java.io.InputStream;
   ```
import java.io.OutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import oracle.portal.provider.v2.transport.PrefStoreTransporter;
import oracle.portal.provider.v2.transport.TransportLogger;
import oracle.portal.provider.v2.preference.Preference;
import oracle.portal.provider.v2.preference.PreferenceStore;
import oracle.portal.provider.v2.preference.PreferenceStoreException;
public class ExportByRefPrefStoreTransporter extends PrefStoreTransporter {
    public ExportByRefPrefStoreTransporter(PreferenceStore prefStore) {
        super(prefStore);
    }

    /**
     * Exports the context path of the supplied @link
     * oracle.portal.provider.v2.preference.Preference from the @link
     * @param pref the source @link
     * oracle.portal.provider.v2.preference.Preference
     * @param out the <code>java.io.OutputStream</out> to which data will be
     * written.
     * @param logger
     */
    protected void exportPreference(Preference pref, OutputStream out,
    TransportLogger logger) throws PreferenceStoreException, IOException {
        // Get the context path of the preference we are exporting.
        String contextPath = pref.getContextPath();
        DataOutputStream dos = new DataOutputStream(out);
        // Write the context path in the export data. The import process
        // will use this context path to lookup this preference in the
        // preference store and copy it to the new context
        dos.writeUTF(contextPath);
    }

    /**
     * Reads a context path from the stream and copies preference data
     * from that location into the @link
     * @param pref the target @link
     * oracle.portal.provider.v2.preference.Preference
     * @param in the <code>java.io.InputStream</code> from which to read data.
     * @param logger
     */
    protected void importPreference(Preference pref, InputStream in,
    TransportLogger logger) throws PreferenceStoreException, IOException {
        // Read the context path to copy the value for this
        // preference from.
        DataInputStream dis = new DataInputStream(in);
        String contextPath = dis.readUTF();
        // Create preference object to copy from (identical to the
        // target preference but with a different context path)
        Preference sourcePref = new Preference(contextPath,
        pref.getName(), pref.getType(), (String)null);
        // Copy across the preference
        getPrefStore().copy(sourcePref, pref, true);
    }
}
3. Update provider.xml to include the following element for your portlet:

```xml
<portlet class="oracle.portal.provider.v2.DefaultPortletDefinition">
  ...
  <portletInstanceClass>oracle.portal.sample.v2.devguide.tx.
    ExportByRefDefaultPortletInstance</portletInstanceClass>
</portlet>
```

For more information on the syntax of provider.xml, refer to the provider JavaDoc on OTN:


### 7.2.8 Enhancing Portlet Performance with Caching

In the previous sections of this chapter, you learned how to write fully-functional Java portlets using the PDK Framework. Once you complete the basic functionality of your portlet, you may want to turn your attention to portlet performance.

Caching is a common technique for enhancing the performance of Web sites that include a great deal of dynamic content. The overhead involved in retrieving data and generating the output for dynamic content can be significantly reduced by proxying requests through a local agent backed by a large, low-latency data store known as a cache. The cache agent responds to a request in one of the following two ways:

- If a valid version of the requested content exists in the cache, the agent simply returns the existing cached copy, thus skipping the costly process of content retrieval and generation. This condition is called a cache hit.

- If a valid version of the requested content does not exist in the cache, the agent forwards the request to its destination and awaits the return of the content. The agent returns the content to the requestor and stores a local copy in its cache for reuse if a subsequent request for the same content arises. This condition is called a cache miss.

Web providers generate dynamic content (that is, portlets) and they often reside remotely from the OracleAS Portal instance on which they are deployed. As such, caching might improve their performance. The architecture of OracleAS Portal lends itself well to caching, since all rendering requests originate from a single page assembling agent, known as the Parallel Page Engine (PPE), which resides on the middle tier. You can make the PPE cache the portlets rendered by your Web provider and reuse the cached copies to handle subsequent requests, minimizing the overhead your Web provider imposes on page assembly.

The Web provider can use any one of three different caching methods, depending upon which one is best suited to the application. The methods differ chiefly in how they determine whether content is still valid:

1. **Expiry-based Caching**: When a provider receives a render request, it stamps its response with an expiry time. The rendered response remains in the cache and fills all subsequent requests for the same content until its expiry time passes. This caching scheme is perhaps the simplest and most performant because the test for cache validity requires very little overhead and does not involve any network round trips. Expiry-based caching suits applications where the content has a well-defined life span. For content with a less certain life span, however, expiry-based caching is less effective. Refer to Section 7.2.8.2, "Activating Caching" and Section 7.2.8.3, "Adding Expiry-Based Caching" for more information.
2. **Invalidation-based Caching**: Invalidation-based caching works essentially the same way as expiry-based caching, except that the contents of the cache can expire or become invalid at any time. Invalidation of cache content is usually triggered by an event.

For example, if you have a calendar portlet that shows your appointments for the day, the content for the portlet could be generated once, the first time you show the calendar for that day. The content remains cached until something happens to change your schedule for that day, such as the addition of an appointment, the deletion of an existing appointment, or a change of time for an appointment. Each of these change events can trigger an action in the calendar application. When such an event takes place, your calendar application can generate an invalidation request for any cached portlet content affected by the change. The next time you view a page containing your calendar portlet, the cache will not contain an entry. Your Web provider will be contacted to regenerate the new content with the modified schedule.

This method is a very efficient way to cache content because the originator of the content, that is, your Web provider, is contacted only when new content needs to be generated, but you are not bound to a fixed regeneration schedule. Refer to Section 7.2.8.2, "Activating Caching" and Section 7.2.8.4, "Adding Invalidation Based Caching" for more information.

3. **Validation-based Caching**: When a provider receives a render request, it stamps its response with a version identifier (or E Tag). The response goes into the cache, but, before the PPE can reuse the cached response, it must determine whether the cached version is still valid. It sends the provider a render request that includes the version identifier of the cached content. The provider determines whether the version identifier remains valid. If the version identifier is still valid, the provider immediately sends a lightweight response to the PPE without any content, which indicates the cached version can be used. Otherwise, the provider generates new content with a new version identifier, which replaces the previously cached version. In this form of caching, the PPE must always confirm with the provider whether the content is up to date. The validity of the cached copy is determined by some logic in the provider. The advantage of this approach is that the provider controls the use of the cached content rather than relying on a fixed period of time. Refer to Section 7.2.8.2, "Activating Caching" and Section 7.2.8.5, "Adding Validation-Based Caching" for more information.

### 7.2.8.1 Assumptions

1. You have followed through and understood Section 6.6.2, "Building PDK-Java Portlets".

2. You built a portlet using the wizard and successfully added it to a page.

### 7.2.8.2 Activating Caching

To use the caching features of OracleAS Portal in your Web providers, you must first activate the middle tier cache. This cache is known as the PL/SQL Cache because it is the same cache used by **mod_plsql**, the Oracle HTTP Server plug-in that calls database procedures, and hence database providers, over HTTP.

Usually, your OracleAS Portal administrator is responsible for the configuration details of caching.

For invalidation-based caching, you need to configure OracleAS Web Cache in front of the Web provider. Bear in mind that remote Web providers often do not have OracleAS

Once you have installed and configured OracleAS Web Cache, ensure the following in the OracleAS Web Cache Manager:

- It points to the host name and port of the Web provider.
- Caching rules do not cause the caching of provider content. Content should be cached according to the surrogate control headers generated by the provider in its response.

### 7.2.8.3 Adding Expiry-Based Caching

Expiry-based caching is one of the simplest caching schemes to implement, and can be activated declaratively in your XML provider definition. You can set an expiry time for the output of any ManagedRenderer you utilize by setting its pageExpires property to the number of minutes you want the output to be cached for. For example, suppose you want portlet output to be cached for one minute.

1. After you have used the Portlet Wizard to build a portlet as described in Section 6.6.2, “Building PDK-Java Portlets”, edit the provider.xml file and set the pageExpires property tag of showPage to 1. This will set an expires entry of 1 minute for the portlet.

   By default the wizard generates a standard and compressed tag for showPage. You need to expand the tag to include a subtag of pageExpires:

   ```xml
   <showPage class="oracle.portal.provider.v2.render.http.ResourceRenderer">
     <resourcePath>/htdocs/mycacheportlet/MyCachePortletShowPage.jsp
     <pageExpires>1</pageExpires>
   </showPage>
   ```

   For more information on the syntax of provider.xml, refer to the provider JavaDoc on OTN:


2. Test that the portlet is cached for 1 minute by adding some JSP code to your show page. You can simply add the current time to your JSP:

   ```jsp
   <%@page contentType="text/html; charset=windows-1252"
   import="oracle.portal.provider.v2.render.PortletRenderRequest"
   import="java.util.Date"
   import="java.text.SimpleDateFormat"%>

   <PortletRenderRequest pReq = (PortletRenderRequest)
   request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST));
   SimpleDateFormat df = Dateformat.getTimeInstance(DateFormat.LONG,
   DateFormat.LONG,pReq.getLocale());
   String time = df.format(new Date());%

   <P>Hello <%=pReq.getUser().getName() %>.</P>
   <P>This is the <b><i>Edit</i></b> render mode!</P>
   <P>This information is correct as of <%=time%>.</P>
   ```
When viewing the portlet, you see that the time (including seconds) is constant for 1 minute. After the time has expired, the portlet displays the most current time and a new cache is set.

### 7.2.8.4 Adding Invalidation Based Caching

When using Oracle AS Web Cache, requests for content are sent through HTTP and content is either returned from the cache or the HTTP request is forwarded to the originator of the content. When content is returned to Oracle AS Web Cache, it is added to the cache before being returned to the requestor. Cache content is invalidated by sending invalidation requests directly to Oracle AS Web Cache. PDK-Java uses the Java API for Web Cache (JAWC) to generate invalidation requests. This section describes how to configure Oracle AS Web Cache and use the invalidation-based caching sample that comes with PDK-Java.

Not all requests sent to Oracle AS Web Cache are cached. In order for the content to be cached, the content must include directives that tell Oracle AS Web Cache to cache the content. Usually Oracle AS Web Cache uses the URL associated with the request as the cache key, but you can specify additional keys by setting special HTTP headers, known as surrogate control headers, on the response.

To configure a Java portlet to use invalidation-based caching, you do the following:

- **Configuring Oracle AS Web Cache.** Refer to Oracle Application Server Web Cache Administrator’s Guide for more information.
- **Configuring the Provider Servlet**
- **Defining the Oracle AS Web Cache Invalidation Port**
- **Configuring the XML Provider Definition**
- **Manually Invalidating the Cache**

#### 7.2.8.4.1 Configuring the Provider Servlet

To enable invalidation-based caching, you must switch it on at the provider servlet level. The flag is set in an initialization parameter inside the PDK-Java Web application deployment descriptor, `web.xml`. For example:

```xml
<servlet>
  ...  
  <init-param>
    <param-name>invalidation_caching</param-name>
    <param-value>true</param-value>
  </init-param>
</servlet>
```

If the flag is not defined here, then invalidation-based caching is switched off. The status of this flag can be checked by displaying the provider’s test page. If the `testPageURI` property is not set in the `sample.properties` file, then the provider code displays the test page in the old default style. The old style test page correctly picks up the invalidation caching flag from the `web.xml` file. The format of the test page URL is as follows:

```
http://provider_hostname:port/jpdk/providers/sample
```

#### 7.2.8.4.2 Defining the Oracle AS Web Cache Invalidation Port

If you are using an Oracle Application Server installation type where PDK-Java was automatically pre-installed (for example, an Oracle AS Portal and Wireless type installation), you should find that Oracle AS Web Cache invalidation settings have already been preconfigured in `MID_
TIER_ORACLE_HOME/portal/conf/cache.xml. In this case, you can safely ignore the instructions in this section and proceed to Section 7.2.8.4.3, “Configuring the XML Provider Definition”. Otherwise, you will need to configure the invalidation portlet for OracleAS Web Cache.

First, you need the user name and password for the invalidation port(s) of the OracleAS Web Cache instance(s) associated with your application server. After you obtain those, use the provided oracle.webdb.provider.v2.cache.Obfuscate Java utility to convert the user name and password into an obfuscated string of the format required by the JAWC API. In a default Oracle Application Server installation, the user name for the invalidation port is usually invalidator and the password is usually the same as the application server administrator's password. For example, suppose you installed Oracle Application Server in D:\as904, with an invalidation port user name of invalidator and a password of welcome. You would run the following command:

`D:\as904\jdk\bin\java -classpath D:\as904\portal\jlib\pdkjava.jar oracle.webdb.provider.v2.cache.Obfuscate invalidator:welcome`

If successful, the command should print a hexadecimal string like the following:

`0510198d5df8efd5779406342be2528aa0cccb179ea6b77baf49f019f5075a3a11`

Now, use this information to create a JAWC configuration file, cache.xml, which specifies one or more OracleAS Web Cache instances and their invalidation ports. For example:

```xml
<?xml version="1.0"?>
<webcache>
<invalidation
  host="cache.mycompany.com"
  port="4001"
  authorization="0510198d5df8efd5779406342be2528aa0cccb179ea6b77baf49f019f5075a3a11"/>
</webcache>
```

JAWC finds this configuration file using the system property oracle.http.configfile. To set this system property for an Oracle Application Server Containers for J2EE instance within an Oracle Application Server installation, simply add an appropriate line to the oc4j.properties file for the particular instance in which PDK-Java is installed (for example, MID_TIER_ORACLE_HOME/j2ee/OC4Jinstance/config/oc4j.properties) and then restart that instance. For example:

`oracle.http.configfile=fully_qualified_filename`

If you are running Oracle Application Server Containers for J2EE standalone, you can specify the option in the command line you use to start it.

`java -Doracle.http.configfile=fully_qualified_filename -jar oc4j.jar`
### 7.2.8.4.3 Configuring the XML Provider Definition

Using a combination of tags in provider.xml, you can activate automatic invalidation-based caching for an entire portlet or some of its pages. To turn on automatic invalidation-based caching, you need to declare it as follows either at the level of individual pages or the renderer, to indicate that invalidation-based caching should be activated for all pages:

```xml
<useInvalidationCaching>true</useInvalidationCaching>
```

If your portlet supports personalization (using the Edit or Edit Defaults modes), you may also want to declare `<autoInvalidate>true</autoInvalidate>` for your renderer. This declaration causes invalidation of all the cached content for the portlet when you click OK or Apply in Edit mode, thus causing new markup to be generated based on the personalized settings.

The maximum time for holding the page in the cache is the minimum of the following:

- Maximum expiry time from OracleAS Portal defined in the Cache tab of the Global Settings page.
- Web Provider default (24 hours) if no maximum expiry time is specified by OracleAS Portal.
- The time in minutes of the `<pageExpires>` tag, if present.

The following excerpt from provider.xml specifies that this portlet shall be cached for up to 5 minutes and shall be automatically invalidated upon personalization:

```xml
<renderer class="oracle.portal.provider.v2.render.RenderManager">
  <contentType>text/html</contentType>
  <renderContainer>true</renderContainer>
  <autoRedirect>true</autoRedirect>
  <autoInvalidate>true</autoInvalidate>
  <showPage class="oracle.portal.provider.v2.render.http.ResourceRenderer">
    <resourcePath>/htdocs/invalidation/invalidation1.jsp</resourcePath>
    <useInvalidationCaching>true</useInvalidationCaching>
    <pageExpires>5</pageExpires>
  </showPage>
  <editPage class="oracle.portal.sample.v2.devguide.invalidation.InvalidationEditRenderer"/>
</renderer>
```

**Note:** The `pageExpires` tag is also used for normal expiry based caching. These two forms of caching are mutually exclusive. Invalidation-based caching takes place in an OracleAS Web Cache instance located in the same place as the Web provider. Pages stored using expiry based caching are cached in the middle tier of OracleAS Portal.

For more information on the syntax of provider.xml, refer to the provider JavaDoc on OTN:


---

**Note:** Since the location of the cache configuration file is defined as a system property, only one cache may be defined for each server instance. It is not possible to have two different Web Provider instances behind separate OracleAS Web Cache configurations.

---

**Note:** The `pageExpires` tag is also used for normal expiry based caching. These two forms of caching are mutually exclusive.
7.2.8.4 Manually Invalidating the Cache You may want the cached version of the portlet invalidated when a request is processed or information somewhere has been updated. In these cases, you may want to manually invalidate the cache.

The invalidation-based caching portlet sample included with PDK-Java contains a single portlet that displays the time the content was cached and a link to trigger an invalidation request. The first time a page request is made to the Web provider through the cache, the response is cached. Subsequent requests for the portlet content are fulfilled by returning content from OracleAS Web Cache. When you click the link at the bottom of the portlet an invalidation request is generated by the provider that removes the portlet from the cache. The next request for the portlet is forwarded to the provider and the provider generates a new portlet with the current time.

To perform invalidation calls to OracleAS Web Cache, first you need to have a handle to a ServletInvalidationContext object. You can get this handle by calling the static getServletInvalidationContext() method of the ServletInvalidationContextFactory class.

Once you have the handle, you need to specify the cache key. In the cache key, you need to specify whether you want to invalidate a particular portlet instance, all the instances of a portlet, or all the portlets managed by a provider. To perform this task, you use the methods of the ServletInvalidationContext class or the methods of its super class, InvalidationContext. This is where you can specify whether the portlet should be cached for this particular user (USER level). If there is no user specified in the cache key, then the cached content is returned to all users, which means you are using SYSTEM level caching.

The following example invalidates a portlet instance and calls the setFullProviderPath() and setPortletReference() methods. When the caching key is set, you call the invalidate() method on the InvalidationContext object that sends the invalidation message to OracleAS Web Cache.

ServletInvalidationContext inv = ServletInvalidationContextFactory.getServletInvalidationContext();
inv.setFullProviderPath(((ServletPortletRenderRequest)pReq);
inv.setPortletReference(pReq.getPortletReference());
int num = inv.invalidate();

Review the Web cache sample provider for more information.

7.2.8.5 Adding Validation-Based Caching

Adding validation-based caching requires slightly more effort, but gives you explicit control over exactly which requests to your provider are cache hits. As an example, you may want to update the cache only when data within the portlet has changed. To implement this algorithm, you need to override the prepareResponse method. The signature of the BaseManagedRenderer.prepareResponse method is:

public boolean prepareResponse(PortletRenderRequest pr)
    throws PortletException, PortletNotFoundException

In your version of prepareResponse(), you need to do the following:

- Retrieve the cached version identifier set by the PPE in the render request by calling the HttpPortletRendererUtil.getCachedVersion() method:

  public static java.lang.String getCachedVersion
If the portlet finds the previously cached version valid, the appropriate header has to be set by calling the `HttpPortletRendererUtil.useCachedVersion()` method. It also instructs the `RenderManager` that it won’t be necessary to call `renderBody()` to render the portlet body.

```java
public static void useCachedVersion(PortletRenderRequest request)
```

Otherwise, use `HttpPortletRendererUtil.setCachedVersion()` to generate a new version of the portlet, which will be cached. It also indicates to the PPE that the `renderBody()` method has to be called to regenerate the portlet content.

```java
public static void setCachedVersion(PortletRenderRequest request, java.lang.String version, int level)
    throws java.lang.IllegalArgumentException
```

For validation-based caching, you need not update `provider.xml`. You can view the portlet by refreshing the page or adding the portlet to a page and updating the content. If the content has changed, the portlet shows the new content. If the content has not changed, a cached version of the portlet is displayed.

### 7.2.9 Enhancing Portlets for Mobile Devices

This section explains how to go about enhancing a PDK-Java portlet for a mobile device. Before proceeding with this section, you should familiarize yourself with the guidelines for building mobile-enabled portlets, Section 6.1.3, "Guidelines for Mobile Portlets", and the methods of building portlets with PDK-Java, Section 7.2, "Enhancing PDK-Java Portlets".

To properly build a portlet for a mobile device, do the following:

1. Create a JSP to generate the OracleAS Wireless XML in response to the show request for the portlet. The portlet’s JSPs are managed and controlled by the built-in renderer, `oracle.portal.provider.v2.render.http.ResourceRenderer`.

   With this renderer, you can define distinct resources (JSPs) for each Show mode as well as which JSP to call for each Show mode. Hence, you can define one JSP to handle the Show mode when the requested content is HTML and another when the requested content is OracleAS Wireless XML. For example, this capability enables you to put the lottery portlet’s mobile rendition in a separate JSP. In this sample, `mlotto.jsp` generates the mobile rendition.

   ```java
   <%@ page session="false" contentType="text/vnd.oracle.mobilexml" %>
   <%@ page import="oracle.portal.sample.v2.devguide.lottery.LottoPicker" %>
   <LottoPicker picker = new LottoPicker();
   picker.setIdentity(request.getRemoteAddr() ); %>
   <SimpleText>
   <SimpleTextItem />
   <SimpleTextItem>Your numbers:</SimpleTextItem>
   <SimpleTextItem />
   <SimpleTextItem HALIGN="CENTER" >
   ```
<%  
int [] picks = picker.getPicks();  
for (int i = 0; i < picks.length; i++) {  
    out.print(picks[i] + " ");  
}  
out.println(" ");  
%>

2. Modify provider.xml by adding the acceptContentType tag in the portlet definition to indicate that the portlet can generate OracleAS Wireless XML responses. Each acceptContentType tag defines a distinct content type that the portlet can render. The value is the MIME type of the content. In this case the lottery portlet declares that it can generate HTML (text/html) and OracleAS Wireless XML (text/vnd.oracle.mobilexml).

<acceptContentType>text/html</acceptContentType>
<acceptContentType>text/vnd.oracle.mobilexml</acceptContentType>

For more information on the syntax of provider.xml, refer to the provider JavaDoc on OTN:

3. Declare the mapping of JSP renderers to Show modes in provider.xml.

You need to define a render handler for the pertinent Show modes by content type. One showPage declaration identifies a JSP that renders the HTML response and another one identifies the JSP that renders the OracleAS Wireless XML response. Note that the class attribute is now required, whereas, in a simpler case, it could allow the class to default. You express the association of a particular JSP to a particular content type through the resourcePath and contentType tags inside the showPage declaration:

- resourcePath defines the JSP used to render this mode.
- contentType defines the response type of this JSP.

When the ResourceRenderer receives a show request for the lottery portlet it invokes lotto.jsp to render the HTML response and mlotto.jsp for OracleAS Wireless XML. The following code illustrates how you express this relationship in provider.xml:

<renderer class="oracle.portal.provider.v2.render.RenderManager">  
<contentType>text/html</contentType>  
<renderContainer>true</renderContainer>  
<showPage  
class="oracle.portal.provider.v2.render.http.ResourceRenderer">  
<resourcePath>/htdocs/lottery/lotto.jsp</resourcePath>  
<contentType>text/html</contentType>  
</showPage>  
<showPage  
class="oracle.portal.provider.v2.render.http.ResourceRenderer">  
<resourcePath>/htdocs/lottery/mlotto.jsp</resourcePath>  
<contentType>text/vnd.oracle.mobilexml</contentType>  
</showPage>  
...  
</renderer>
Enhancing Java Portlets

4. Declare a short title. A short title is the short form of a portlet's name and is for use where display space is limited. Specifically, OracleAS Portal uses it when rendering portlets as menu items in the page menu generated in response to a mobile request. If the portlet has registered a short title, OracleAS Portal uses that as the menu item label. Otherwise, it uses the portlet's standard title. To declare a short title, you include the shortTitle tag in the portlet's metadata section in provider.xml:

```xml
<portlet class="oracle.portal.provider.v2.DefaultPortletDefinition">
  <id>2</id>
  <name>Lottery</name>
  <title>Lottery Portlet</title>
  <shortTitle>Lottery</shortTitle>
  ...
</portlet>
```

5. Support short title personalization. Because the portlet's short title is presented to the user as the menu item label that references the portlet instance on the page, we recommend that all portlets allow users to personalize the short title. PDK-Java provides a base data class that manages both the short and standard title:

```java
oracle.portal.provider.v2.personalize.NameValuePersonalizationObject
```

---

**Note:** The ResourceRenderer (and any portlet) determines the type of request it has received by the request's Accept header. The Accept header is a standard HTTP header that defines the acceptable response types for a given request. It may be a list with multiple values if multiple content types are acceptable. When there are multiple values, they are listed in order from most to least preferred. OracleAS Portal controls the values passed to the portlet in the Accept header. In the case of an HTML request, the Accept header is:

```
text/html, text/xml, text/vnd.oracle.mobilexml
```

These values indicate that OracleAS Portal prefers an HTML response but also accepts XML and OracleAS Wireless XML.

For mobile requests, the Accept header is:

```
text/vnd.oracle.mobilexml, text/xml
```

These values indicate that OracleAS Portal prefers OracleAS Wireless XML but also accepts general XML that contains a stylesheet reference that transforms to OracleAS Wireless XML.

The ResourceRenderer maps requested content type to the specific resource renderer by working through the Accept list in preference order. Once it finds a match between an acceptable response type and a registered renderer, it dispatches the request to the resource. For example, for HTML requests the ResourceRenderer will match the first entry, text/html, to the declaration that defines lotto.jsp as the text/html handler. Likewise, when a mobile request is received, the ResourceRenderer matches the first entry, text/vnd.oracle.mobilexml, to the declaration that defines mlotto.jsp as the text/vnd.oracle.mobilexml handler.
The NameValuePersonalizationObject manages both the title and short title. It contains methods for getting and setting the values. The personalization code is split into two parts, form display and form submit. The logic first acquires the current data personalization object. Next, it checks to see if this is a form submit. If so, it updates the values in the data personalization object, writes them back to the repository, and returns. PDK-Java subsequently redirects back to this same form but without the parameters that indicate it is a form submit causing the form to redisplay with the new values. In this situation, the JSP responds with the personalization page including the current personalization values.

We recommend that portlets use this class or subclass to inherit this base support. The only personalization the lottery sample supports is these two titles. Hence, it uses the NameValuePersonalizationObject class directly in custom.jsp:

```jsp
<%@ page session="false" import="oracle.portal.provider.v2.*" %>
<%@ page import="oracle.portal.provider.v2.http.*" %>
<%@ page import="oracle.portal.provider.v2.render.*" %>
<%@ page import="oracle.portal.provider.v2.personalize.NameValuePersonalizationObject" %>

<%-- This page both displays the personalization form and processes it. We display the form if there isn’t a submitted title parameter, else we apply the changes --%>
<%
PortletRenderRequest portletRequest = (PortletRenderRequest) request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
String title = request.getParameter("lotto_title");
String sTitle = request.getParameter("lotto_short_title");
String actionParam = PortletRendererUtil.getEditFormParameter(portletRequest);
String action = request.getParameter(actionParam);
NameValuePersonalizationObject data = (NameValuePersonalizationObject) PortletRendererUtil.getEditData(portletRequest);
// Cancel automatically redirects to the page
// -- so will only receive 'OK' or 'APPLY'
if (action != null) {
    data.setPortletTitle(title);
    data.setPortletShortTitle(sTitle);
    PortletRendererUtil.submitEditData(portletRequest, data);
    return;
}
// otherwise just render the form
String title = data.getPortletTitle();
String sTitle = data.getPortletShortTitle();
%>
<TABLE BORDER="0">
    <TR>
        <TD WIDTH="20%">
            <P ALIGN="RIGHT">Title:</P>
        </TD>
        <TD WIDTH="80%">
            <INPUT TYPE="TEXT" NAME="lotto_title" VALUE="<%= title %>"></TD>
    </TR>
    <TR>
        <TD WIDTH="20%">
            <P ALIGN="RIGHT">Short Title:</P>
        </TD>
        <TD WIDTH="80%">
    </TABLE>
```
6. Support the rendering of personalized short titles. Once you add short title personalization, a portlet must take responsibility for rendering the portlet as a menu item in the mobile page response. Because of the small screen displays on mobile devices, portlets aren’t rendered together. Users are presented with a menu of links to portlets on a given page. The user navigates to each portlet using this menu to see the content. To better support this model, OracleAS Portal provides Link mode, where portlets generate Link references to themselves.

For HTML requests, Link mode generates an anchor tag, `<a href>` For mobile requests, Link mode generates a `<simpleHref>` tag. Currently, OracleAS Portal only sends a Link mode request when assembling a mobile page response. Hence, you only need to support rendering the `text/vnd.oracle.mobilexml` content type for Link mode. If your portlet also has an HTML rendition, we recommend you also support HTML Link mode.

The lottery portlet example implements each Link rendition in a distinct JSP:

- `milotto.jsp` contains the code to generate a `text/vnd.oracle.mobilexml` Link response.
- `anchorlotto.jsp` contains the code to generate a `text/html` Link response.

The code for each is almost identical. Since the purpose of supporting Link mode is to render the personalized short title, the code first acquires the short title from the repository. It then generates the appropriate link tag for the requested markup. The acquired short title is rendered as the link’s label. Since OracleAS Portal is responsible for creating the URL that references any particular usage of a portlet on a page, OracleAS Portal creates the URL used in the link and passes it to the portlet, which then uses the URL to create the link. The URL for the link is retrieved from the request’s `PageURL` parameter. The code for `milotto.jsp` is as follows:

```jsp
<%@ page session="false" contentType="text/vnd.oracle.mobilexml" %>
<%@ page import="oracle.portal.provider.v2.http.HttpCommonConstants" %>
<%@ page import="oracle.portal.provider.v2.render.*" %>
<%@ page import="oracle.portal.provider.v2.personalize.NameValuePersonalizationObject" %>
<%@ page import="oracle.portal.utils.xml.v2.XMLUtil" %>

```PortletRenderRequest` portletRequest = (PortletRenderRequest)
        request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
NameValuePersonalizationObject data = (NameValuePersonalizationObject)
        PortletRendererUtil.getEditData(portletRequest);
String title = data.getPortletShortTitle();
// if short title is empty then use the title
if (title == null || title.length() == 0)
    title = data.getPortletTitle();
```
7. Declare support for Link mode. Once you have implemented Link mode, you must update provider.xml to indicate the presence of Link mode. You declare Link mode by adding code similar to the following to provider.xml:

```xml
<portlet class="oracle.portal.provider.v2.DefaultPortletDefinition">
  <id>2</id>
  <name>Lottery</name>
  ...
  <showLink>true</showLink>
  <showEdit>true</showEdit>
  <showEditToPublic>false</showEditToPublic>
  ...
</portlet>
```

8. Bind the JSPs to the Link mode. The portlet must declare the mapping between the Show modes and the JSP renderers. The syntax for Link mode is similar to that for Show mode.

```xml
<portlet class="oracle.portal.provider.v2.DefaultPortletDefinition">
  <id>2</id>
  <name>Lottery</name>
  ...
  <renderer class="oracle.portal.provider.v2.render.RenderManager">
    <contentType>text/html</contentType>
    <renderContainer>true</renderContainer>
    <linkPage
      class="oracle.portal.provider.v2.render.http.ResourceRenderer">
      <resourcePath>/htdocs/lottery/anchorlotto.jsp</resourcePath>
      <contentType>text/html</contentType>
    </linkPage>
    <linkPage
      class="oracle.portal.provider.v2.render.http.ResourceRenderer">
      <resourcePath>/htdocs/lottery/milotto.jsp</resourcePath>
      <contentType>text/vnd.oracle.mobilexml</contentType>
    </linkPage>
    ...
  </renderer>
  ...
</portlet>
```

7.2.9.1 Accessing Configuration, User, and Device Information

To better support mobile devices, OracleAS Portal passes extra information to the portlet for use in generating its response. This information falls into the following three major categories:

- Configuration Data
Enhancing PDK-Java Portlets

- User Data
- Device Information

7.2.9.1.1 Configuration Data OracleAS Portal sends a flag that indicates whether the mobile function is enabled when requesting that the portlet render its personalization or edit pages. Portlets can use this information to exclude or include mobile specific attributes in their personalization pages as appropriate.

The portlet accesses this information using the `PortletRenderRequest` object:

```java
if (portletRequest.getPortalConfig().isMobileEnabled()) {
    ...
}
```

7.2.9.1.2 User Data OracleAS Wireless adds the user location to the requests it forwards to OracleAS Portal for response. This user location information is determined by the user's actual location, if the user's device has this support, or a profiled location. The user location is exposed by the `ProviderUser` object, which you can access from the `PortletRenderRequest` object. Portlets that are location aware can use this information to adjust the content they generate.

```java
UserLocation location = portletRequest.getUser().getLocation();
if (location != null) {
    ...
}
```

7.2.9.1.3 Device Information On each request, OracleAS Portal sends characteristics about the requesting device to the portlet. This information is sent for all portlet requests, not just mobile requests. The information classifies the type of device making the request, its layout orientation, the maximum response size the device can handle, and an indication of whether the connection with the device is secure. Table 7–3 describes the available device types.

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>voice</td>
<td>Indicates a voice-only device, such as a normal telephone calling a voice access number.</td>
</tr>
<tr>
<td>micromessenger</td>
<td>Indicates text messaging devices, such as SMS phones or pagers.</td>
</tr>
<tr>
<td>messenger</td>
<td>Indicates general messaging devices, such as e-mail.</td>
</tr>
<tr>
<td>microbrowser</td>
<td>Indicates a small size display device, which supports a markup browser, such as a WAP phone.</td>
</tr>
<tr>
<td>pdabrowser</td>
<td>Indicates a medium size display device, such as a Palm or PocketPC.</td>
</tr>
<tr>
<td>pcbrowser</td>
<td>Indicates a large size display device used with desktop browsers.</td>
</tr>
</tbody>
</table>

Portlets may choose to alter the layout or representation of the content in their response based on the type of device making the request. For example, a portlet may break a wide table into a series of screens that link column groups together for small screen devices.
The maximum response size is a hint that a portlet can use to constrain the amount of data it returns. The size is expressed in bytes. A maximum response size of 0 means the size is unknown.

The portlet accesses this information using the PortletRenderRequest object.

```
...  
DeviceInfo deviceInfo = portletRequest.getDeviceInfo();
switch (deviceInfo.getDeviceClass()) {
    case DeviceInfo.DeviceClass.MICROBROWSER:
        renderMicroBrowser(portletRequest);
        break;
    default:
        renderDefault(portletRequest);
        break;
}
...  
```

### 7.2.9.2 Modifying Navigation for Mobile Portlets

Much of the information in Section 7.2.3.3.5, "Implementing Navigation within a Portlet" is also relevant to the development of mobile portlets, but some of the utilities referenced are specific to portlets that render HTML. For portlets that render SimpleResult, the equivalent utilities shown in Table 7–4 are available.

<table>
<thead>
<tr>
<th>HTML Utilities</th>
<th>SimpleResult Utilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>UrlUtils.constructHTMLLink</td>
<td>UrlUtils.constructMXMLLink</td>
</tr>
<tr>
<td>UrlUtils.htmlFormHiddenFields</td>
<td>UrlUtils.mxmlFormHiddenFields</td>
</tr>
<tr>
<td>UrlUtils.emitHiddenField</td>
<td>UrlUtils.emitMxmlHiddenField</td>
</tr>
</tbody>
</table>

The following example illustrates how to adapt the thesaurus sample for a mobile-enabled portlet. Note that, when deployed as a mobile portlet, both sets of JSPs (HTML and SimpleResult) are deployed. These JSPs complement their HTML counterparts, they do not replace them. Notice also that the JSPs use SimpleResult as their markup and the value of the navigation parameter has changed such that it points to the next mobile JSP rather than the next desktop JSP.

```
mThesaurusForm.jsp:
<%
    PortletRenderRequest pRequest = (PortletRenderRequest)
        request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
    String paramNameQ = "q";
    String qualParamNameQ =
        HttpPortletRendererUtil.portletParameter(pRequest, paramNameQ);
%>
<!-- Output the MXML content -->
<SimpleText>
    <SimpleTitle>Thesaurus</SimpleTitle>
    <SimpleFormItem>Enter the word you wish to search for:</SimpleFormItem>
    <SimpleForm
        method="POST"
        target="<%= UrlUtils.htmlFormActionLink(pRequest,UrlUtils.PAGE_LINK) %>">
        <%= UrlUtils.mxmlFormHiddenFields
            (pRequest.getRenderContext().getPageURL()) %>
        <%= UrlUtils.emitMxmlHiddenField
            HttpPortletRendererUtil.portletParameter(request, "next_page"),
```

7-76   Oracle Application Server Portal Developer's Guide
"htdocs/path/mThesaurusLink.jsp" } %>
 <SimpleFormItem type="text" size="20" name="<%= qualParamNameQ %>">
     value="" />
 </SimpleForm>
</SimpleText>

**mThesaurusLink.jsp:**

<%
PortletRenderRequest pRequest = (PortletRenderRequest)
    request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
String paramNameQ = "q";
String paramNameNextPage = "next_page";
String qualParamNameQ = HttpPortletRendererUtil.portletParameter(pRequest, paramNameQ);
String qualParamNameNextPage = HttpPortletRendererUtil.portletParameter(pRequest, paramNameNextPage);
String paramValueQ = pRequest.getQualifiedParameter(paramNameQ);
%>

<!-- Output the MXML content -->

<SimpleText>

<SimpleTitle>Words similar to <%= paramValueQ %></SimpleTitle>

<%
    Thesaurus t = new Thesaurus();
    String[] relatedWords = t.getRelatedWords(paramValueQ);
    NameValue[] linkParams = new NameValue[2];
    linkParams[0] = new NameValue(
        qualParamNameNextPage, "htdocs/path/mThesaurusLink.jsp");
    for (int i=0; i<relatedWords.length; i++)
    {
        linkParams[1] = new NameValue(
            qualParamNameQ, relatedWords[i]);
    }
%>

<SimpleTextItem>
<%= relatedWords[i] %>
</SimpleTextItem>

<SimpleTextItem>
<%= UrlUtils.constructMXMLLink(  
    pRequest,  
    pRequest.getRenderContext().getPageURL(),  
    "(words related to " + relatedWords[i] + ")",  
    ",",  
    linkParams,  
    true,  
    true) %>
</SimpleTextItem>

<SimpleTextItem>
<%= relatedWords[i] %>
</SimpleTextItem>

<SimpleTextItem>
<SimpleHref target="" portletURL=""/>
</SimpleTextItem>

</SimpleText>
7.2.10 Writing Multilingual Portlets

This section shows you how to build a Java portlet that can be rendered in different languages. The language used in your portlet will depend upon on the language setting that has been chosen in the portal that is displaying it.

Once you have completed this section you will be able to write portlets that support as many or as few languages as you wish. You will also be able to convert your existing portlets to support multiple languages. Once a portlet is written to support multiple languages, it is easy to plug in new languages. The basic model for multilingual Java portlets is similar to the standard Java Internationalization model. If you already know about Java Internationalization, you should find this process very familiar.

7.2.10.1 Assumptions

1. You have followed through and understood Section 6.6.2, "Building PDK-Java Portlets".
2. You built a portlet using the wizard and successfully added it to a page.

7.2.10.2 Internationalizing Your Portlet

- Providing Translations for Portlet Content
- Providing Translation for Portlet Attributes

7.2.10.2.1 Providing Translations for Portlet Content

In Section 6.6.2, "Building PDK-Java Portlets", you created a portlet using the Java Portlet Wizard. The basic message created by the wizard is only available in one language and the text displayed is hard-coded in to the portlet’s renderer class. To make your portlets available in multiple languages, you have to store such language dependent elements in their own resource bundles.

Creating Resource Bundles

For each language you want your portlet to be available in, you will need a resource bundle. You will also need to create a resource bundle to use when there is no resource bundle corresponding to the language setting chosen in the portal.

- Create a Default Resource Bundle

1. In Oracle JDeveloper, create a Java class called MyProviderBundle that extends ListResourceBundle from the java.util.package. The class should contain a multi-dimensional array of objects that holds key-value pairs representing each of the language dependent elements from your JSP show page. This implementation is demonstrated in the following code:

```java
package mypackage2;
import java.util.ListResourceBundle;
public class MyProviderBundle extends ListResourceBundle {
    public static String HELLO_MSG = "MyPortletHelloMessage";
    public static String INFO_MSG = "MyPortletInfoMessage";
    public Object[][] getContents()
    {
        return contents;
    }
    static private final Object[][] contents = {
        {HELLO_MSG, "Hello"},
        {INFO_MSG, "This is the show page of the portlet and it is being generated"}
    }
}
```
Enhancing PDK-Java Portlets

In the default language!

1. Save MyProviderBundle.

Creating Resource Bundles for Other Supported Languages

Now you must create a resource bundle class for each language you want your portlet to support. Each of these classes must be named the same as your default resource bundle class, but with a language code appended to the end. For example, if you want to support the French language, create a Java class named MyProviderBundle_fr. The language code fr is the same as the code that will be used by the locale object in the portal if the language setting is set to French.


When you change the language setting in OracleAS Portal, you change the value of the current locale object and therefore the locale object's language code. These language codes adhere to the ISO:639 codes for representation for names of languages.

1. To create a French resource bundle, create a Java class named MyProviderBundle_fr, as described earlier.

2. Using your default resource bundle as a template, replace the English language strings with their French equivalents. See the following example:

   ```java
   package mypackage2;

   import java.util.ListResourceBundle;
   public class MyProviderBundle_fr extends ListResourceBundle
   {
   public Object[][] getContents()
   {
   return contents;
   }
   static private final Object[][] contents =
   {
   {MyProviderBundle.HELLO_MSG, "Bonjour"},
   {MyProviderBundle.INFO_MSG, "Cette page est le 'show mode' de la portlet et est generee dans la langue par defaut."}
   };
   }
   

   4. Repeat steps 1 through 3 for every language that you wish to create a resource bundle for, updating the class name with the appropriate language code and the message strings with their equivalent in the appropriate language.

Updating Your Renderer

To make use of the resource bundles you just created, you need to edit the JSP show page and replace the hard-coded messages with references that will pickup the
messages at run time from the resource bundle that corresponds most closely with the locale object of the portal.

1. Open the JSP that represents your show page and change the following:

```jsp
<%@page contentType="text/html; charset=windows-1252"
    import="oracle.portal.provider.v2.render.PortletRenderRequest"
    import="java.util.ResourceBundle"
%>

<% PortletRenderRequest pReq = (PortletRenderRequest)
    request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
    ResourceBundle b =
    ResourceBundle.getBundle("mypackage2.MyProviderBundle",pReq.getLocale()); %>

<!-- Pull the message from the appropriate resource bundle. -->
<P> <%= b.getString(mypackage2.MyProviderBundle.HELLO_MSG) %>
P> <%= pReq.getUser().getName() %>.</P>

2. Save your JSP page.

Now you can refresh your portlet and view the changes (see Figure 7–11).

Figure 7–11 Portlet in English

To view the French greeting (see Figure 7–12), you set the language in the Set Language portlet to French instead of English.

Figure 7–12 Portlet in French

Notice that the text inside the portlet has changed, but the portlet title remains in the default language, English. You can also have the portlet set the appropriate portlet attributes (such as portlet name, portlet title, and portlet description) by pointing to a resource bundle from provider.xml, as described in the next section.
Providing Translation for Portlet Attributes

In your provider's definition file, `provider.xml`, a number of attributes describing your portlet are defined such as the portlet's name and description, these are used in places, for example in your portlet's title bar in Show mode and so should be translated, too. There are two different ways of providing these translations, which one you choose is up to you. Both of these methods are outlined as follows:

- **Method 1: Using Resource Bundles at the Provider Level**
- **Method 2: Creating Resource Bundles at Portlet Level**

### Method 1: Using Resource Bundles at the Provider Level

You can provide translations for your portlet attributes in your resource bundle(s), then specify that you want to use these resource bundles in `provider.xml`, specifying the keys you have used in your resource bundles. Using this method you can use the keys you want to, and as long as you use different keys for each corresponding attribute in your provider's various portlets you can have just one set of resource bundles that all of your provider's portlets can use.

#### Updating Your Resource Bundles

1. Open your default resource bundle, `MyProviderBundle.java`.
2. Add additional strings to your resource bundle that represent your portlet attributes and then add text for those strings:

   ```java
   package mypackage2;

   import java.util.ListResourceBundle;
   public class MyProviderBundle extends ListResourceBundle {
     public static String HELLO_MSG = "MyPortletHelloMessage";
     public static String INFO_MSG = "MyPortletInfoMessage";
     public static String PORTLET_NAME = "FirstPortletName";
     public static String PORTLET_TITLE = "FirstPortletTitle";
     public static String PORTLET_SHORT_TITLE = "FirstPortletShortTitle";
     public static String PORTLET_DESCRIPTION = "FirstPortletDescription";
     public static String TIMEOUT_MESSAGE = "FirstPortletTimeoutMessage";

     public Object[][] getContents() {
       return contents;
     }

     static private final Object[][] contents = {
       {HELLO_MSG, "Hi"},
       {INFO_MSG, "This is the show page of the portlet and it is being generated in the default language!"},
       {PORTLET_NAME, "MyNLSportlet"},
       {PORTLET_TITLE, "My NLS Portlet"},
       {PORTLET_SHORT_TITLE, "MyNLSportlet"},
       {PORTLET_DESCRIPTION, "My first ever NLS portlet, using my MyPortletShowPage.jsp"},
       {TIMEOUT_MESSAGE, "Timed out waiting for MyNLSportlet"}
     };
   }
   ```

3. Save `MyProviderBundle.java`.
4. Open `MyProviderBundle_fr.java`. Change it so that it contains the French strings that match the strings declared in `MyProviderBundle`.
package mypackage2;

import java.util.ListResourceBundle;
public class MyProviderBundle_fr extends ListResourceBundle {
    public Object[][] getContents()
    {
        return contents;
    }
    static private final Object[][] contents =
    {
        {MyProviderBundle.HELLO_MSG, "Bonjour"},
        {MyProviderBundle.INFO_MSG, "Cette page est le 'show mode' de la portlet et est generée en français!"},
        {MyProviderBundle.PORTLET_NAME, "MaPremierePortlet"},
        {MyProviderBundle.PORTLET_TITLE, "Ma Portlet Multi-Langue"},
        {MyProviderBundle.PORTLET_SHORT_TITLE, "Ma NLS Portlet"},
        {MyProviderBundle.PORTLET_DESCRIPTION, "Ma premiere portlet multi-langue, utilisant mon renderer"},
        {MyProviderBundle.TIMEOUT_MESSAGE, "Temps d'accès à la portlet demandée expire"}
    };
}

5. Save MyProviderBundle_fr.java.

- Updating provider.xml

  1. Open the XML provider definition file and update it to point to the resource bundle instead of using the hard-coded portlet attribute values.

```xml
<portlet class="oracle.portal.provider.v2.DefaultPortletDefinition">
    <id>3</id>
    <resource>mypackage2.MyProviderBundle</resource>
    <nameKey>FirstPortletName</nameKey>
    <titleKey>FirstPortletTitle</titleKey>
    <ShortTitleKey>FirstPortletShortTitle</ShortTitleKey>
    <descriptionKey>FirstPortletDescription</descriptionKey>
    <timeout>10</timeout>
    <timeoutMessageKey>FirstPortletTimeoutMessage</timeoutMessageKey>
    <showEditToPublic>false</showEditToPublic>
    <hasAbout>true</hasAbout>
</portlet>
```

For more information on the syntax of provider.xml, refer to the provider JavaDoc on OTN:


**Method 2: Creating Resource Bundles at Portlet Level**

PDK-Java defines a set of resource bundle keys that you can use for providing translations for your portlet attributes. Making use of these keys means that you don't have to specify the resource bundle keys in your provider.xml file, as we did in Method 1: Using Resource Bundles at the Provider Level. However, you do have to provide a separate set of resource bundles for each portlet in your provider as the keys you use for each portlet need to be the same, but their values will differ.

- Updating Your Resource Bundles

  1. Open your default resource bundle, MyProviderBundle.java.
2. Remove any changes you made from the previous section, and import oracle.portal.provider.v2.PortletConstants. You can then reference the following constants instead of the strings. You do not have to declare static strings when using PortletConstants:

{PortletConstants.NAME, "MyNLSPortlet"},
{PortletConstants.TITLE, "My NLS portlet"},
{PortletConstants.SHORTTITLE, "MyNLSPortlet"},
{PortletConstants.DESCRIPTION, "My first ever NLS portlet"},
{PortletConstants.TIMEOUTMSG, "Timed out waiting for MyNLSPortlet"}

3. Save MyProviderBundle.java.

4. Open MyProviderBundle_fr.java. Remove the portlet attributes added in the previous section, import oracle.portal.provider.v2.PortletConstants, and reference the constants instead of the strings.

{PortletConstants.NAME, "MaPremierePortlet"},
{PortletConstants.TITLE, "Ma Portlet Multi-Langue"},
{PortletConstants.SHORTTITLE, "Ma NLS Portlet"},
{PortletConstants.DESCRIPTION, "Ma premiere portlet multi-langue, utilisant mon renderer"},
{PortletConstants.TIMEOUTMSG, "Temps d'accès à la portlet demandée expire"}

5. Save MyProviderBundle_fr.java.

■ Updating provider.xml

1. In provider.xml, you need to use only one tag instead of one tag for each string as you did in Method 1: Using Resource Bundles at the Provider Level. Delete all translated strings in the file, for example, the <nameKey>, <titleKey>, <ShortTitleKey>, and <descriptionKey> tags. Then add the following tag between the portlet id and the timeout number value:

<resource>mypackage2.MyProviderBundle</resource>

For more information on the syntax of provider.xml, refer to the provider JavaDoc on OTN:

c/xml_tag_reference_v2.html

For more information on Java Internationalization see the Internationalization trail of the Java Tutorial.

7.2.10.3 Viewing the Portlet

Once you have updated your provider and deployed it to Oracle Application Server Containers for J2EE, refresh the provider and portal page containing your portlet. To see your resource bundles working, add the "Set Language" portlet to your page and try changing the language setting to French. Remember that the default resource bundle is English, and that selecting any other language that doesn't have a corresponding resource bundle will result in the portlet being displayed in English.

7.3 Building Struts Portlets with Oracle JDeveloper

This section describes the framework for building Struts portlets with Oracle JDeveloper for use in OracleAS Portal. You will learn how to build a Struts portlet
from an existing application by adding the Struts Tag Library from the Oracle Application Server Portal Developer Kit (version 9.0.4.0.2 or higher) to Oracle JDeveloper, then use the Oracle PDK Java Portlet wizard to create the Java portlet itself.

- OracleAS Portal and the Apache Struts Framework
- Creating a Struts Portlet

### 7.3.1 OracleAS Portal and the Apache Struts Framework

This section discusses the use of the Apache Struts with OracleAS Portal. Struts is an implementation of the Model-View-Controller (MVC) design pattern.

- Model View Controller Overview
- Apache Struts Overview
- OracleAS Portal Integration with Struts
- Summary

#### 7.3.1.1 Model View Controller Overview

Enterprise applications undertake the following distinct tasks:

- Data access
- Business logic implementation
- User interface display
- User interaction
- Application (page) Flow

The MVC (Model View Controller) architecture provides a way of compartmentalizing these tasks, based on the premise that activities, such as data presentation, should be separate from data access. This architecture enables you to easily plug a data source into the application without having to rewrite the user interface. MVC allows the logical separation of an application into three distinct layers: the Model, the View, and the Controller.

**The Model**

The Model is the repository for the application data and business logic. One facet of the Model’s purpose is to retrieve data from and persist data to the database. It is also responsible for exposing the data in such a way that the View can access it, and for implementing a business logic layer to validate and consume the data entered through the View. At the application level, the Model acts as the validation and abstraction layer between the user interface and the business data that is displayed. The database server itself is simply a persistence layer for the Model.

**The View**

The View is responsible for rendering the Model data using JSPs. The View code does not include a hardcoded application or navigation logic, but may contain some logic to carry out tasks like displaying conditional data based on a user role. When an end user executes an action within the HTML page that the View renders, an event is submitted to the Controller. The Controller then determines the next step.
The Controller

The Controller is the linchpin of the MVC pattern. Every user action carried out in the View is submitted through the Controller. The Controller then performs the next action, based on the content of the request from the browser.

The Controller can be driven in several different ways. For example, you can use URL arguments to route the requests to the correct code. The MVC pattern itself determines the function of the Controller, not how it should work.

Benefits

The MVC architecture provides a clear and modular view of the application and its design. By separating the different components and roles of the application logic, it allows developers to design applications as a series of simple and different components: the Model, the View, and the Controller. This pattern should help to create applications that are easier to maintain and evolve. For example, once you create one view, you can easily create another view using the same business logic. Because of the ease and reusability, the MVC pattern is the most widely used pattern in the context of Web-based application development.

Figure 7–13 shows how the MVC pattern applies to a conventional thin-client Web application:

Figure 7–13  The MVC Pattern

7.3.1.2 Apache Struts Overview

The Apache Struts framework (http://struts.apache.org) is one of the most popular frameworks for building Web applications, and provides an architecture based on the JSP Model 2 approach of the MVC design paradigm. In the Model 2 approach, end user requests are managed by a servlet that controls the flow, and uses components such as JavaBeans or EJBs to access and manipulate the data. It then uses JSPs to render the application content in a Web browser. This model differs from JSP Model 1, where the JSPs managed the browser request and data access.

The Struts framework provides its own HTTP Servlet as a controller component. The Struts framework is driven by an XML configuration file that contains the page flow of the application. Struts does not provide the Model, but allows developers to integrate it to any data access mechanism, for example EJBs, TopLink, or JDBC. The most common technology for writing View components is JSP and Struts provides various tag libraries to help in writing these, although some of these tags have now been superseded by the Java Standard Tag Library (JSTL), which may also be used.
7.3.1.3 OracleAS Portal Integration with Struts

The Oracle Application Server Portal Developer Kit contains numerous examples and documents regarding the usage of the OracleAS Portal APIs, such as personalization and caching. The integration of the application flow and business logic is not part of the portlet APIs. By using the Struts framework, however, you can leverage the MVC architecture to create and publish applications within your enterprise portal.

Oracle Struts Portlet

To create a portlet using the Struts framework (see Figure 7–14), or to generate a portlet from an existing Struts application, you must deploy all the components in the J2EE container. In the context of OracleAS Portal, the Struts application is called by the PPE, and not by the browser as compared to a standalone Struts application. When a portlet show call is made, the page engine sends a request to the Struts portlet renderer, which then forwards the request to the Apache Struts Controller servlet.

Figure 7–14 Integrating Struts Applications with OracleAS Portal

The following code shows a portion of the provider definition file (provider.xml):

```xml
...<renderContainer>true</renderContainer>
    <renderCustomize>true</renderCustomize>
    <autoRedirect>true</autoRedirect>
    <contentType>text/html</contentType>
    <showPage class="oracle.portal.provider.v2.render.http.StrutsRenderer">
        <defaultAction>showCustomer.do</defaultAction>
    </showPage>
</renderer>
...```

Note: For more information about JSTL and JSF, see the FAQ on the Apache Software Foundation Web site (http://jakarta.apache.org/kickstart.html).
For more information on the syntax of provider.xml, refer to the provider JavaDoc on OTN:


The showPage tag defines the business logic that will be executed in the Show mode of the portlet. The showPage of the Struts portlet contains two important components:

1. The renderer class
   (oracle.portal.provider.v2.render.http.StrutsRenderer), which receives the portlet request from the PPE and acts as a proxy to forward the request to the Struts Action Servlet.

2. The defaultAction tag, which defines the Struts action that will be used by default when the portlet is called for the first time.

The PDK-Java enables you to easily develop a view (Portal View) of your Struts application. This view enforces a consistent look and feel of your Struts portlet using portal styles, and allows the end user to use the application within the portal.

To create a Struts portlet, you must use the OracleAS Portal JSP tags, which are extensions of the default Struts JSP tags. This development process is similar to that of creating a standalone Struts application. To learn how to build a Struts portlet, refer to Section 7.3.2.1, "Creating a Struts Portlet". Also, since the portlet and struts application must also be in the same Servlet Context, you must create a single Web application that contains both elements. To learn how to easily create this Web application in Oracle JDeveloper, refer to the next section, Section 7.3.2.1, "Creating a Struts Portlet".

### 7.3.2 Creating a Struts Portlet

OracleAS PDK contains new extensions to integrate Apache Struts applications. This section explains how to build a portlet from an existing struts application. You can also follow these steps to create a portlet that uses the Model View Controller paradigm. To learn more about the Apache Struts framework, refer to Section 7.3.1, "OracleAS Portal and the Apache Struts Framework". The PDK-Java extensions described in this section rely on Apache Struts 1.1.

This section contains the following steps:

- Creating a Struts Portlet
- Registering the Provider
- Summary
7.3.2.1 Creating a Struts Portlet
To publish a part of an existing Struts application as portlet, we recommend that you first create a new view to serve as the Portal View of your application. This view uses existing objects (Actions, ActionForm, and so on) with a new mapping and new JSPs.

---

**Note:** Although we recommend that you create a Portal View of your application, you could alternatively replace your application's struts tags with PDK-Java struts tags. This approach enables your application to run both as a standalone struts application and a portlet.

---

In this example (see Figure 7–15 and Figure 7–16), you will create a portlet that enables you to add a new entry to a Web Logger (Blog).

*Figure 7–15  Submitting a Blog*

```
<prepareNewBlog
  success
    enterNewBlog.jsp
```

*Figure 7–16  Saving a Blog Entry*

```
/saveNewBlog
  success
    view/newBlogConfirmation.jsp
```

prepareNewBlog is a simple empty action that redirects the request to the `enterNewBlog.jsp` page. This page shows a form for submitting a new blog.

The corresponding entry in the `struts-config.xml` is:

```
<action path="/prepareNewBlog" scope="request"
  type="view.PrepareNewBlogAction">
  <forward name="success" path="/view/enterNewBlog.jsp"/>
</action>

<action path="/saveNewBlog" name="blogForm" scope="request"
  type="view.SaveNewBlogAction" input="/view/enterNewBlog.jsp"/>
```
7.3.2.1 Create a new flow and view to host the portlet actions

To create a new view, first create a new set of ActionMappings (page flow) that will redirect the various actions and requests to Portal-specific JSPs.

```xml
<action path="/portal/prepareNewBlog" scope="request"
  type="view.PrepareNewBlogAction" >
  <forward name="success" path="/view/portal/enterNewBlog.jsp"/>
</action>

<action path="/portal/saveNewBlog" name="blogForm" scope="request"
  type="view.SaveNewBlogAction" input="/view/enterNewBlog.jsp">
  <forward name="success" path="/view/portal/newBlogConfirmation.jsp"/>
</action>
```

As you can see, only the path attributes are modified. The FormBean Action responsible for the application business logic remains unchanged.

7.3.2.1.2 Creating the new JSPs

As specified in the previous step, the actions forward the request to new JSPs, which are responsible for rendering the portlet content. Your new portlet view JSPs can share the HTML with the standalone view, but be sure that the portlet:

- Uses Portal styles that enforce a consistent look and feel with the rest of the portal page.
- Contains HTML code that is allowed in HTML table cells (that is, no `<html>`, `<body>`, and `<frame>` tags).
- Renders portal-aware links and forms. This is necessary to ensure that your Struts portlet renders its content inline, thus keeping your users within the context of the portal page by rendering the requested content within the same portlet container.

To achieve inline rendering in your Struts portlet, you must use OracleAS PDK tags:

```xml
<pdk-struts-html:form action="/portal/saveNewBlog.do">
  ...
  ...
</pdk-struts-html:form>
```

During the rendering of the portlet, one of the JSP tags (for example, the `pdk-struts-html:form` tag), submits the form to the Parallel Page Engine (PPE), which then sends the parameters to the Struts portlet. The Struts controller executes the logic behind these actions and returns the next JSP to the portlet within the portal page.

The PDK contains all the Struts tags, and extends all the tags that are related to URLs. The following is a list of the PDK extended tags:

- `form`: creates an HTML form and embeds the portal page context in the form to ensure inline rendering
- `text`: renders fields on the form.
- `link` and `rewrite`: create a link to the portal page, and are required for inline rendering
- `img`: creates an absolute link that points to the Web provider. If you want to use this tag in the context of Internet Web sites that have firewalls, you must make sure the provider is directly accessible from the Internet. If it is not possible, you
can deploy the images to the OracleAS Portal middle tier and use the Apache
Struts image link to generate a relative link (relative to the portal, not to the
application).

Note: You can register the OracleAS PDK with Oracle JDeveloper so
that you can drop the tags from the Oracle JDeveloper Components
Palette. For more information, see the Registering a Custom Tag Library
in JDeveloper section in the Oracle JDeveloper online help.

7.3.2.1.3 Creating a Portlet  You can create your Struts portlet either manually or by
using the Oracle PDK Java Portlet option in Oracle JDeveloper. Although the Oracle
PDK Java Portlet option does not explicitly offer Struts support, you can utilize this to
build your Struts portlet.

To create a portlet:

1. In Oracle JDeveloper, open the Java Portlet Wizard to create an Oracle PDK Java
Portlet.

Note: The Java Portlet and Oracle PDK Java Portlet options are used
to create JPS-compliant portlets and PDK-Java portlets respectively.
Clicking Java Portlet or Oracle PDK Java Portlet opens the Java
Portlet Wizard. For more information on opening the wizard, see
Section 6.6.2.1, "Creating a Portlet and Provider".

2. For the Implementation Style of the show page, select Java Class.

3. For the Package Name, enter oracle.portal.provider.v2.render.http

4. For the Class Name, enter StrutsRenderer. This generates the skeleton of the
portlet renderer class, StrutsRenderer.

5. Since the StrutsRenderer is part of the PDK, you do not need this generated
file. So, when you finish the wizard, you must delete the file generated by the
wizard. To do so, click the file in the System Navigator window, then choose File >
Erase from Disk in Oracle JDeveloper.

6. Edit the provider.xml and change the following properties:

At the provider level:

- If you want users to always return to the same portlet state as when they left
  the portal page, you can configure the struts renderer to save the struts action
  in the struts context:

  `<actionInSession>true</actionInSession>`

  If you prefer that users always start from the beginning of the portlet when
  they return from outside the portal page, then you should not save the struts
  action:

  `<actionInSession>false</actionInSession>`

- If the Struts application uses sessions (for example, the form synchonizer
token mechanism is used or `<actionInSession>` is set to true), enable
  session handling:

  `<session>true</session>`
At theportlet level:

- Specify the first action to raise when theportlet is called. Use the following code:

```xml
<showPage class="oracle.portal.provider.v2.render.http.StrutsRenderer">
<defaultAction>/portal/prepareNewBlog.do</defaultAction>
</showPage>
```

For more information on the syntax of provider.xml, refer to the provider JavaDoc on OTN:


7.3.2.1.4 Extending the portlet to add Portal Business Logic In your application, you should add code specific to your portal, such as the user's information, personalization, and localization. To do so, you can create a new Action class that is only called in the Portal context, and handles all Portal-specific business logic.

7.3.2.2 Registering the Provider

Now that your portlet is ready to be used by OracleAS Portal, you must make it accessible to OracleAS Portal by registering it. For information on how to register your PDK-Java portlet, refer to Section 6.6.2.5, "Registering and Viewing Your Portlet". If you chose to save the struts action in the session context, `<actionInSession>true</actionInSession>`, then you must specify the provider login frequency as Once per user session during registration. Setting the login frequency this way ensures the session information is passed to your struts portlet.

7.3.2.3 Summary

Oracle Application Server enables you to easily create Struts portlets using Oracle JDeveloper and publish existing Struts applications to OracleAS Portal. For more information on using the Oracle JDeveloper Java Portlet wizards, refer to the beginning of this chapter. For more information on using OracleAS Portal, refer to the Oracle Application Server Portal User’s Guide and the OracleAS Portal Online Help.

7.3.3 Creating an Oracle Application Development Framework (ADF) Portlet

Similarly to Struts, Oracle ADF relies on the MVC design pattern. Oracle ADF applications leveraging the Struts controller can be turned into portlets and deployed to OracleAS Portal the same way as Struts applications. Refer to Section 7.3.2, "Creating a Struts Portlet".

---

**Note:** After creating the Oracle ADF portlet, you may find that the JSP page does not display correctly. This is because the Parallel Page Engine request is sent to a provider through a SOAP request (oracle.webdb.provider.v2.adapter.SOAPServlet), which implies that the portal does not serve the page as a standard .JSP page. To resolve this, create the ADFBindingFilter filter.

---

To create the ADFBindingFilter filter and filter mappings, include the following in your web.xml file:

```xml
<filter>
  <filter-name>ADFBindingFilter</filter-name>
</filter>
```
<filter-class>oracle.adf.model.servlet.ADFBindingFilter</filter-class>
<init-param>
  <param-name>encoding</param-name>
  <param-value>windows-1252</param-value>
</init-param>
</filter>
<filter-mapping>
  <filter-name>ADFBindingFilter</filter-name>
  <url-pattern>*.jsp</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>ADFBindingFilter</filter-name>
  <url-pattern>*.jspx</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>ADFBindingFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>ADFBindingFilter</filter-name>
  <url-pattern>*/action</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>ADFBindingFilter</filter-name>
  <url-pattern>*/jsp</url-pattern>
</filter-mapping>
Creating PL/SQL Portlets

The OracleAS Portal PL/SQL APIs are implemented as a set of PL/SQL packages and objects. Database providers and portlets are deployed to a database schema as PL/SQL packages. This chapter explains how to create PL/SQL portlets based on the Oracle Application Server Portal Developer Kit-PL/SQL (PDK-PL/SQL). To make effective use of this chapter, you should already know PL/SQL and have some familiarity with the PL/SQL Web Toolkit.

This chapter contains the following sections:

- Guidelines for Creating PL/SQL Portlets
- Building PL/SQL Portlets with the PL/SQL Generator
- Building PL/SQL Portlets Manually
- Implementing Information Storage
- Using Parameters
- Accessing Context Information
- Implementing Portlet Security
- Improving Portlet Performance with Caching
- Implementing Error Handling
- Implementing Event Logging
- Writing Multilingual Portlets
- Enhancing Portlets for Mobile Devices
- Registering Providers Programmatically

The source code for many of the examples referenced in this chapter is available as part of PDK-PL/SQL. You can download PDK-PL/SQL from the OracleAS Portal Developer Kit (PDK) page on OTN:


When you unzip PDK-PL/SQL, you will find the examples in:
Guidelines for Creating PL/SQL Portlets

When you write your portlets in PL/SQL, you should follow the best practices described in this section:

- Portlet Show Modes
- Recommended Portlet Procedures and Functions
- Guidelines for Mobile Portlets

8.1 Guidelines for Creating PL/SQL Portlets

Just like a Java portlet, a PL/SQL portlet has a variety of Show modes available to it. A Show mode is an area of functionality provided by a portlet. The available Show modes are described fully in Chapter 6, "Creating Java Portlets":

-Shared Screen mode is described in Section 6.1.1.1, "Shared Screen Mode (View Mode for JPS)"
-Edit mode is described in Section 6.1.1.2, "Edit Mode (JPS and OracleAS Portal)"
-Edit Defaults mode is described in Section 6.1.1.3, "Edit Defaults Mode (JPS and OracleAS Portal)"
-Preview mode is described in Section 6.1.1.4, "Preview Mode (JPS and OracleAS Portal)"
-Full Screen mode is described in Section 6.1.1.5, "Full Screen Mode (OracleAS Portal)"
-Help mode is described in Section 6.1.1.6, "Help Mode (JPS and OracleAS Portal)"
-About mode is described in Section 6.1.1.7, "About Mode (JPS and OracleAS Portal)"
-Link mode is described in Section 6.1.1.8, "Link Mode (OracleAS Portal)"

To check for the selected Show mode, you can use the constants in the `wwpro_api_provider` package. These constants are listed with their corresponding Show mode in Table 8–1.

<table>
<thead>
<tr>
<th>Show mode</th>
<th>Constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared Screen</td>
<td>MODE_SHOW</td>
</tr>
<tr>
<td>Edit</td>
<td>MODE_SHOW_EDIT</td>
</tr>
<tr>
<td>Edit Defaults</td>
<td>MODE_SHOW_EDIT_DEFAULTS</td>
</tr>
<tr>
<td>Preview</td>
<td>MODE_SHOW_PREVIEW</td>
</tr>
<tr>
<td>Full Screen</td>
<td>MODE_SHOW_DETAILS</td>
</tr>
</tbody>
</table>
8.1.2 Recommended Portlet Procedures and Functions

The primary goal of the portlet's code is to generate the HTML output that displays on a page for all of the Show modes required by OracleAS Portal. Although it is possible to implement the portlet as a set of separate PL/SQL stored program units, organizing the portlet's code into a PL/SQL package is the best way of encapsulating related portlet code and data as a single unit in the database. You also achieve better database performance and ease of portlet maintenance.

As you may recall from Section 2.4, "Deployment Type", requests from OracleAS Portal for a particular portlet go through the portlet's provider. To communicate with its portlets, the provider contains a set of required methods that make calls to the portlet code.

When implementing a portlet as a PL/SQL package, it is a good idea to organize the portlet code in parallel with the provider code. For example, when the provider needs to retrieve information about one of its portlets, it uses its get_portlet function. Hence, it makes sense for the portlet to contain a get_portlet_info function that returns the requested information when called by the provider's get_portlet function. Similarly, it is logical for the provider's show_portlet procedure to call the portlet's show procedure, which produces the HTML output for a requested Show mode and returns it to the provider.

Table 8–2 describes the recommended procedures and functions for a PL/SQL portlet to communicate effectively with the database provider.

**Table 8–2 Recommended Functions and Procedures for PL/SQL Portlets**

<table>
<thead>
<tr>
<th>Procedure/Function Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_portlet_info</td>
<td>Returns the portlet record to the provider.</td>
</tr>
<tr>
<td>show</td>
<td>Produces HTML output for a requested Show mode and returns it to the provider.</td>
</tr>
<tr>
<td>register</td>
<td>Initializes the portlet at the instance level. The register procedure should not contain any transaction closing statements, such as COMMIT, ROLLBACK, or SAVEPOINT. OracleAS Portal handles the closing of the transactions itself.</td>
</tr>
<tr>
<td>deregister</td>
<td>Enables cleanups at the instance level. The deregister procedure should not contain any transaction closing statements, such as COMMIT, ROLLBACK, or SAVEPOINT. OracleAS Portal handles the closing of the transactions itself.</td>
</tr>
<tr>
<td>isRunnable</td>
<td>Determines whether the portlet can be run. Security checks can be performed in this function.</td>
</tr>
<tr>
<td>copy</td>
<td>Copies the personalized and default values of portlet preferences from one portlet instance to a new portlet instance when OracleAS Portal makes a copy of the page.</td>
</tr>
<tr>
<td>describe_parameters</td>
<td>Returns a list of public portlet parameters.</td>
</tr>
</tbody>
</table>

---

### Table 8–1 (Cont.) Show Mode Constants in wwpro_api_provider

<table>
<thead>
<tr>
<th>Show mode</th>
<th>Constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help</td>
<td>MODE_SHOW_HELP</td>
</tr>
<tr>
<td>About</td>
<td>MODE_SHOW_ABOUT</td>
</tr>
<tr>
<td>Link</td>
<td>MODE_SHOW_LINK</td>
</tr>
</tbody>
</table>
8.1.3 Guidelines for Mobile Portlets

OracleAS Portal is capable of rendering its pages for both HTML and non-HTML (mobile) devices. When designing a portlet for a mobile device, you must consider some additional guidelines. The guidelines for mobile portlets are described fully in Section 6.1.3, "Guidelines for Mobile Portlets".

For information on how to build mobile-enabled portlets, refer to Section 8.12, "Enhancing Portlets for Mobile Devices".

8.2 Building PL/SQL Portlets with the PL/SQL Generator

To facilitate the development of database providers and PL/SQL portlets, you can use the PL/SQL Generator, a utility that creates installable PL/SQL code for a database provider and its portlets. The PL/SQL Generator is a standalone Web application that receives the provider and portlet definitions in the form of an XML file (similar in format to the provider.xml file). The XML tags used for the provider and portlet definition are a subset of the XML tags used for defining Web providers with PDK-Java. The output of the PL/SQL Generator is a SQL script that can be run from SQL*Plus. The script contains SQL commands for installing the provider and portlet packages in the correct order.

You can download the PL/SQL Generator along with its installation instructions from:

The general model for working with the PL/SQL Generator is as follows:

1. Create an XML file that defines the provider and portlets that you want to build, as described in Section 8.2.1, "Creating the Input XML File".

2. Run the PL/SQL Generator using the XML file as input, as described in Section 8.2.2, "Running the PL/SQL Generator".

3. Publish the generated PL/SQL portlet, which includes the following steps:
   - Install the provider generated by the PL/SQL Generator into the database, as described in Section 8.2.3.1, "Installing the Packages in the Database".
   - Register the database provider with the Oracle Application Server, as described in Section 8.2.3.2, "Registering the Database Provider".
   - Add the generated portlet to a page, as described in Section 8.2.3.3, "Adding Your Portlet to a Page".

8.2.1 Creating the Input XML File

The source XML file starts and ends with the <provider> and </provider> tags, and can include one or more portlet definitions. Each portlet definition is bracketed by the <portlet> and </portlet> tags. A portlet definition includes the XML tags that specify values for the portlet record attributes and enable the links in the portlet header. For example, the <name> tag specifies the portlet name in the provider domain and the <title> tag specifies the portlet display name or title. When set to true, the <showEdit> tag enables the Edit mode for the portlet and the corresponding link in the portlet header. Table 8–3 lists the available XML tags for PL/SQL Generator input.
<table>
<thead>
<tr>
<th>XML Tag</th>
<th>Definition</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>provider</td>
<td>Encloses provider definition tags.</td>
<td>Not applicable</td>
</tr>
<tr>
<td>portlet</td>
<td>Encloses portlet definition tags.</td>
<td>Not applicable</td>
</tr>
<tr>
<td>id</td>
<td>Specifies the portlet ID in the provider. This value must be unique within the provider.</td>
<td>string</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the portlet name. The name should not contain any spaces. The generator uses the information provided in the name tag for the portlet package name.</td>
<td>string</td>
</tr>
<tr>
<td>title</td>
<td>Specifies the portlet display name.</td>
<td>string</td>
</tr>
<tr>
<td>shortTitle</td>
<td>Specifies the portlet short display name. This tag is useful for mobile portlets.</td>
<td>string</td>
</tr>
<tr>
<td>description</td>
<td>Specifies the portlet description.</td>
<td>string</td>
</tr>
<tr>
<td>defaultLocale</td>
<td>Specifies the language the portlet renders by default. The value is the two letter ISO language and country codes expressed as language.country.</td>
<td>string</td>
</tr>
<tr>
<td>timeout</td>
<td>Specifies the portlet’s timeout interval in seconds.</td>
<td>number</td>
</tr>
<tr>
<td>timeoutMsg</td>
<td>Specifies the message to display when the portlet times out.</td>
<td>string</td>
</tr>
<tr>
<td>showEdit</td>
<td>Indicates whether the portlet supports Edit mode, which enables the user to personalize the portlet’s properties.</td>
<td>Boolean</td>
</tr>
<tr>
<td>showEditDefault</td>
<td>Indicates whether the portlet supports the Edit Defaults mode, which enables page administrators to personalize the default values of the portlet’s properties.</td>
<td>Boolean</td>
</tr>
<tr>
<td>showDetails</td>
<td>Indicates whether the portlet can be viewed in Full Screen mode. In this mode, the entire browser window is dedicated to the portlet. Full screen mode enables the portlet to show more details than when it shares the page with other portlets.</td>
<td>Boolean</td>
</tr>
<tr>
<td>showPreview</td>
<td>Indicates whether the portlet supports the Preview mode.</td>
<td>Boolean</td>
</tr>
<tr>
<td>hasHelp</td>
<td>Indicates whether the portlet supports the Help mode.</td>
<td>Boolean</td>
</tr>
<tr>
<td>hasAbout</td>
<td>Indicates whether the portlet supports the About mode.</td>
<td>Boolean</td>
</tr>
<tr>
<td>language</td>
<td>Defines the portlet’s default language (for example, en).</td>
<td>string</td>
</tr>
<tr>
<td>contentType</td>
<td>Indicates the default content type supported by the portlet. The tag can take one of the following values:</td>
<td>string</td>
</tr>
<tr>
<td>apiVersion</td>
<td>Specifies the version of the OracleAS Portal PL/SQL API to which the portlet conforms. The value should be wwpro_api_provider.API_VERSION_1.</td>
<td>string</td>
</tr>
<tr>
<td>XML Tag</td>
<td>Definition</td>
<td>Value Type</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>callIsRunnable</td>
<td>Indicates whether OracleAS Portal must check for the user’s credentials before displaying the portlet. The default value is true.</td>
<td>Boolean</td>
</tr>
<tr>
<td>callGetPortlet</td>
<td>Indicates whether the portal can use the portlet record data stored in the Portlet Metadata Repository (PMR) instead of contacting the provider for the portlet record. If the portlet record (specified by provider id, portlet id, and language) returned by a provider does not change, then the provider should set the value for call_get_portlet to false. This tells the portal to use the PMR instead of making calls to the provider’s get_portlet and get_portlet_list functions. An example of when a provider would not want the portal to useportlet metadata from the PMR is when the value of the portlet records is different for logged on users. The default value is true.</td>
<td>Boolean</td>
</tr>
<tr>
<td>acceptContentType</td>
<td>Specifies a comma-delimited list of content types that the portlet can produce. For example, if a portlet can produce content of both HTML and MOBILEXML type, then the tag value is: text/html,text/vnd.oracle.mobilexml</td>
<td>string</td>
</tr>
<tr>
<td>hasShowLinkMode</td>
<td>Indicates whether the portlet implements the Link mode. If the value is false, the portlet uses its short or full title to display a link label that references the portlet content in a mobile device. Otherwise, a personalized link can be generated in the portlet code. The default value is false.</td>
<td>Boolean</td>
</tr>
<tr>
<td>mobileOnly</td>
<td>Indicates whether the portlet is available only to mobile devices. The default value is false.</td>
<td>Boolean</td>
</tr>
<tr>
<td>preferenceStorePath</td>
<td>Specifies the base preference store path where the provider has stored the portlet personalization information. This path is used when exporting portlets.</td>
<td>string</td>
</tr>
<tr>
<td>createdOn</td>
<td>Defines the portlet creation date. The default value is sysdate.</td>
<td>date</td>
</tr>
<tr>
<td>createdBy</td>
<td>Identifies the user who created the portlet record.</td>
<td>string</td>
</tr>
<tr>
<td>lastUpdatedOn</td>
<td>Defines the most recent date on which the portlet record was changed. The default value is sysdate.</td>
<td>date</td>
</tr>
<tr>
<td>lastUpdatedBy</td>
<td>Identifies the user who most recently changed the portlet record.</td>
<td>string</td>
</tr>
<tr>
<td>passAllUrlParams</td>
<td>Indicates parameter passing behavior in the portlet. If the tag value is true, then OracleAS Portal passes all parameters in the URL to the portlet. If the tag value is false, then the portlet receives only those parameters that are intended for the portlet. The default value is true.</td>
<td>Boolean</td>
</tr>
<tr>
<td>cacheLevel</td>
<td>Indicates a portlet’s cache level. It can take one of the following values: wwpro_api_provider.CACHE_LEVEL_SYSTEM, wwpro_api_provider.CACHE_LEVEL_USER, wwpro_api_provider.CACHE_LEVEL_PTL_SESSION</td>
<td>string</td>
</tr>
</tbody>
</table>
Following is a sample of the input XML for the PL/SQL Generator. Mandatory information is shown in bold.

```xml
<!-- This is a sample provider.xml file for the PLSQL Generator 1.2 -->
<provider>
    <portlet>
        <id>1</id>
        <name>Test_Portlet</name>
        <title>Test Portlet Title</title>
        <shortTitle>Short portlet title</shortTitle>
        <description>This is a Test portlet</description>
        <timeout>30</timeout>
        <timeoutMsg>Test Portlet Timed Out</timeoutMsg>
        <showEdit>true</showEdit>
        <showEditDefault>true</showEditDefault>
        <showDetails>true</showDetails>
        <showPreview>true</showPreview>
        <hasHelp>true</hasHelp>
        <hasAbout>true</hasAbout>
        <language>en</language>
        <contentType>wwpro_api_provider.CONTENT_TYPE_HTML</contentType>
        <apiVersion>wwpro_api_provider.API_VERSION_1</apiVersion>
        <callIsRunnable>true</callIsRunnable>
        <callGetPortlet>true</callGetPortlet>
        <acceptContentType>'text/html'</acceptContentType>
        <hasShowLinkMode>false</hasShowLinkMode>
        <mobileOnly>false</mobileOnly>
        <passAllUrlParams>true</passAllUrlParams>
        <cacheLevel>wwpro_api_provider.CACHE_LEVEL_USER</cacheLevel>
        <rewriteUrls>true</rewriteUrls>
    </portlet>
</provider>
```

### 8.2.2 Running the PL/SQL Generator

After you have created a valid XML input file, you can run the PL/SQL Generator to generate the provider and portlet packages in the form of a SQL file:

1. If you have not already done so, install the PL/SQL Generator according to the instructions that came with the download.

2. From your browser, go to the URL for the PL/SQL Generator. It should look something like the page shown in Figure 8–1.
Click **Browse** and select the source XML file for the **Source XML File** field. Refer to Section 8.2.1, "Creating the Input XML File" for more information on creating the XML file.

In the **Provider Name** field, enter the name of the provider. The provider name must not contain any spaces. The generator uses the value entered in this field for the provider package name.

Click **Generate** to generate the SQL file that contains the installable PL/SQL code for the provider and portlet packages. When the browser prompts you to save or open the file, choose **Save**.

In the Save dialog box, change the file extension to `.sql` and revise the file name as you wish.

Save the file.

### 8.2.3 Publishing the Generated PL/SQL Portlet

After you have run the PL/SQL Generator and obtained a SQL file, you still need to perform the following tasks to make the provider and portlets available to OracleAS Portal:

- Installing the Packages in the Database
- Registering the Database Provider
- Adding Your Portlet to a Page

#### 8.2.3.1 Installing the Packages in the Database

To install the generated provider and portlet packages into the database where you installed OracleAS Portal, perform the following steps:

1. Start a SQL*Plus session and log in to the PORTAL schema.
2. Create a new database schema, the provider schema, to store the generated provider and portlet packages by entering the following commands in SQL*Plus:
3. Grant the EXECUTE privilege for the OracleAS Portal APIs to the provider schema by running the `provsyns.sql` script that is located in the MID_TIER_ORACLE_HOME/portal/admin/plsql/wwc directory as follows:

   ```sql
   @provsyns.sql
   ```

4. Log in to the provider schema and run the generated SQL file. It will create the provider and portlet packages in the database.

### 8.2.3.2 Registering the Database Provider

After creating the provider and portlet packages in the database, you must register the provider with OracleAS Portal before adding the PL/SQL portlet to a portal page:

1. Log in to OracleAS Portal as an administrator.
2. From the Portal Builder, click the **Administer** tab then the **Portlets** tab.
3. In the **Remote Providers** portlet, click **Register a Provider**.
4. Fill in the **Name**, **Display Name**, **Timeout**, and **Timeout Message** as desired.
5. From the **Implementation Style**, list choose **Database**.
6. Click **Next** and complete the remainder of the wizard.
7. When you complete the wizard, click **Finish**.
8. From the Portlet Repository portlet, click **Display Portlet Repository**.
9. Browse the repository and find the provider that you just registered. Typically, new providers appear in the Portlet Staging Area of the repository.
10. Once you find the provider, confirm that it contains all of the portlets you created in the provider. If the provider or its portlets do not appear, then retrace the steps in this section and the preceding sections (Section 8.2.3.1, "Installing the Packages in the Database", Section 8.2.1, "Creating the Input XML File", and Section 8.2.2, "Running the PL/SQL Generator") to ensure that you correctly created and registered your provider and portlet.

### 8.2.3.3 Adding Your Portlet to a Page

Once your provider and its portlets appear in the repository, you can add it to a page. To add your portlet to a page, follow the instructions in the Oracle Application Server Portal User’s Guide.

### 8.3 Building PL/SQL Portlets Manually

This section describes how to build a basic PL/SQL portlet using the **hello world** sample contained in the **starter** provider sample. The **starter** provider sample, located in `..\pdkplsql\pdk\plsql\starter` in PDK-PL/SQL (`pdkplsql.zip`), consists of the following files:

- `starter_provider.pks` is the package specification of the **starter** provider.
- `starter_provider.pkb` is the package body of the **starter** provider.
- `helloworld_portlet.pks` is the package specification of the **hello world** portlet.
- `helloworld_portlet.pkb` is the package body of the **hello world** portlet.
Building PL/SQL Portlets Manually

- `snoop_portlet.pks` is the package specification of the `snoop` portlet.
- `snoop_portlet.pkb` is the package body of the `snoop` portlet.
- `insintpr.sql` is the installation script for the `starter` provider.

The general model for building PL/SQL portlets manually is as follows:

1. Modify the `hello world` portlet package specification and body to create your own portlet package, as described in Section 8.3.1, "Implementing the Portlet Package".
2. Modify the `starter` provider package specification and body to add your new portlet to a provider, as described in Section 8.3.2, "Implementing the Provider Package".
3. Add your portlet to a page, as described in Section 8.3.3, "Adding Your Portlet to a Page".

### 8.3.1 Implementing the Portlet Package

To modify `helloworld_portlet.pks` and `helloworld_portlet.pkb` to create your own portlet package, perform the following steps:

1. Make copies of the package specification, `helloworld_portlet.pks`, and body, `helloworld_portlet.pkb`.
2. Rename the copies to `my_first_portlet.pks` and `my_first_portlet.pkb`, respectively.
3. Open `my_first_portlet.pks` in an editor and change the name of the package to `my_first_portlet`:

   ```sql
   CREATE OR REPLACE
   package my_first_portlet
   is
   ...
   end my_first_portlet;
   ```

4. Open `my_first_portlet.pkb` in an editor and repeat the change that you made in the previous step; that is, change the name of the package to `my_first_portlet`.
5. In `my_first_portlet.pkb`, find the function named `get_portlet_info` and modify it as follows:

   ```sql
   function get_portlet_info
   (p_provider_id in integer,
   p_language in varchar2)
   return wwpro_api_provider.portlet_record
   is
   l_portlet wwpro_api_provider.portlet_record;
   begin
   l_portlet.id := starter_provider.PORTLET_FIRST;
   l_portlet.provider_id := p_provider_id;
   l_portlet.title := 'My First Portlet';
   l_portlet.name := 'My_First_Portlet';
   ```
6. In my_first_portlet.pkb, find the procedure named show and modify it as follows:

```plsql
procedure show
(
    p_portlet_record        wwpro_api_provider.portlet_runtime_record
) is
    l_portlet      wwpro_api_provider.portlet_record;
    l_text_name in varchar2(100);
    l_text in varchar2(200);
begin
    ...
    /*
    Display the content of the portlet in the show mode.
    Use the wwui_api_portlet.portlet_text() API when generating the content of the portlet so that the output uses the portlet CSS.
    */
    htp.p(wwui_api_portlet.portlet_text(
        p_string   =>  'Hello World - Mode Show'
        ,p_level    =>  1
    ));
    /*
    Add the functionality you want here. In this case we are adding a welcome message addressed to the current user.
    */
    l_text_name := 'Welcome to my first portlet ' || wwctx_api.get_user;
    l_text := wwui_api_portlet.portlet_text(
        p_string   =>  l_text_name
        ,p_level    =>  1
    );
    htp.p(l_text);        htp.para;
    if (p_portlet_record.has_border) then
        wwui_api_portlet.close_portlet;
    end if;
    ...

7. Save my_first_portlet.pkb.

8.3.2 Implementing the Provider Package

After you implement the portlet package, you must add your portlet to a provider. To modify starter_provider.pks and starter_provider.pkb to add your new portlet to a provider, perform the following steps:

1. Make copies of the package specification, starter_provider.pks, and body, starter_provider.pkb.
2. Rename the copies to starter_provider2.pks and starter_provider2.pkb, respectively.

   **Note:** If you want to create a new, empty provider, remove all references to the hello world and snoop portlets from starter_provider2.pks and starter_provider2.pkb before performing the steps that follow.

3. Open starter_provider2.pks in an editor.
4. Add a constant called PORTLET_FIRST. This constant is used as the identifier for the portlet within the provider. Hence, the constant’s value must be unique within the provider.

```sql
CREATE OR REPLACE
package STARTER_PROVIDER
is
  /**
   * This package is used as an example to show how providers can be created
   * in the portal system.
   *
   * This provider contains the following portlets:
   *
   * Hello World (PORTLET_HELLOWORLD)
   * Snoop (PORTLET_SNOOP)
   *
   */
  PORTLET_HELLOWORLD constant integer := 1;
  PORTLET_SNOOP      constant integer := 2;
  PORTLET_FIRST      constant integer := 3;
```

5. Save starter_provider2.pks.

6. Open starter_provider2.pkb in an editor.

7. In starter_provider2.pkb, add a call for the new portlet’s get_portlet_info function in the get_portlet function of the provider package. This step entails adding the call `my_first_portlet.get_portlet_info` in the get_portlet function. The get_portlet function allows the portal to retrieve information for the portlet when necessary.

```sql
function get_portlet
  p_provider_id in integer,
  p_portlet_id in integer,
  p_language in varchar2
) return wwpro_api_provider.portlet_record
is
  begin
    if (p_portlet_id = PORTLET_HELLOWORLD) then
      return helloworld_portlet.get_portlet_info(
        p_provider_id => p_provider_id,
        p_language => p_language);
    elsif (p_portlet_id = PORTLET_SNOOP) then
      return snoop_portlet.get_portlet_info(
        p_provider_id => p_provider_id,
        p_language => p_language);
    elsif (p_portlet_id = PORTLET_FIRST) then
      return my_first_portlet.get_portlet_info(
        p_provider_id => p_provider_id,
        p_language => p_language);
    else
      raise wwpro_api_provider.PORTLET_NOT_FOUND_EXCEPTION;
    end if;
  end get_portlet;
```

8. In starter_provider2.pkb, add the new portlet to the list of portlets returned by the provider. This step entails adding the new portlet to the get_portlet_
list function of the provider. The get_portlet_list function tells the portal which portlets the provider implements.

function get_portlet_list
...
begin
  l_cnt := 0;
  if (p_security_level = false) then
    l_cnt := l_cnt + 1;
    lportlet_list(l_cnt) := getportlet(
      p_provider_id => p_provider_id,
      p_portlet_id  => PORTLET_HELLOWORLD,
      p_language    => p_language
    );
    l_cnt := l_cnt + 1;
    lportlet_list(l_cnt) := getportlet(
      p_provider_id => p_provider_id,
      p_portlet_id  => PORTLET_SNOOP,
      p_language    => p_language
    );
    l_cnt := l_cnt + 1;
    lportlet_list(l_cnt) := getportlet(
      p_provider_id => p_provider_id,
      p_portlet_id  => PORTLET_FIRST,
      p_language    => p_language
    );
  else
    if (helloworld_portlet.isRunnable(
      p_provider_id => p_provider_id,
      p_reference_path => null
    ) then
      l_cnt := l_cnt + 1;
      lportlet_list(l_cnt) := getportlet(
        p_provider_id => p_provider_id,
        p_portlet_id  => PORTLET_HELLOWORLD,
        p_language    => p_language
      );
    end if;
    if (snoop_portlet.isRunnable(
      p_provider_id => p_provider_id,
      p_reference_path => null
    ) then
      l_cnt := l_cnt + 1;
      lportlet_list(l_cnt) := getportlet(
        p_provider_id => p_provider_id,
        p_portlet_id  => PORTLET_SNOOP,
        p_language    => p_language
      );
    end if;
    if (my_first_portlet.isRunnable(
      p_provider_id => p_provider_id,
      p_reference_path => null
    ) then
      l_cnt := l_cnt + 1;
      lportlet_list(l_cnt) := getportlet(
        p_provider_id => p_provider_id,
        p_portlet_id  => PORTLET_FIRST,
        p_language    => p_language
      );
    end if;
  end if;
return l_portlet_list;
end get_portlet_list;

9. In starter_provider2.pkb, modify the is_portlet_runnable function to add a call to the is_runnable function of the new portlet.

```plsql
function is_portlet_runnable
  p_portlet_instance in wwpro_api_provider.portlet_instance_record
return boolean
is
begin
  if (p_portlet_instance.portlet_id = PORTLET_HELLOWORLD) then
    return helloworld_portlet.is_runnable(
      p_provider_id => p_portlet_instance.provider_id,
      p_reference_path => p_portlet_instance.reference_path
    );
  elsif (p_portlet_instance.portlet_id = PORTLET_SNOOP) then
    return snoop_portlet.is_runnable(
      p_provider_id => p_portlet_instance.provider_id,
      p_reference_path => p_portlet_instance.reference_path
    );
  elsif (p_portlet_instance.portlet_id = PORTLET_FIRST) then
    return my_first_portlet.is_runnable(
      p_provider_id => p_portlet_instance.provider_id,
      p_reference_path => p_portlet_instance.reference_path
    );
  else
    raise wwpro_api_provider.PORTLET_NOT_FOUND_EXCEPTION;
  end if;
end is_portlet_runnable;
```

10. Repeat step 9 according to the information in Table 8–4.

<table>
<thead>
<tr>
<th>Procedure/Function</th>
<th>Addition</th>
</tr>
</thead>
<tbody>
<tr>
<td>procedure register_portlet</td>
<td>elsif (p_portlet_instance.portlet_id = PORTLET_FIRST) then my_first_portlet.register(p_portlet_instance)</td>
</tr>
<tr>
<td>procedure deregister_portlet</td>
<td>elsif (p_portlet_instance.portlet_id = PORTLET_FIRST) then my_first_portlet.deregister(p_portlet_instance)</td>
</tr>
<tr>
<td>function describe_portlet_</td>
<td>elsif (p_portlet_id = PORTLET_FIRST) then return my_first_portlet.describe_parameters(p_provider_id, p_language);</td>
</tr>
<tr>
<td>parameters</td>
<td></td>
</tr>
<tr>
<td>procedure show_portlet</td>
<td>elsif (p_portlet_record.portlet_id = PORTLET_FIRST) then my_first_portlet.show(p_portlet_record)</td>
</tr>
<tr>
<td>procedure copy_portlet</td>
<td>elsif (p_copy_portlet_info.portlet_id = PORTLET_FIRST) then my_first_portlet.copy(p_portlet_record)</td>
</tr>
</tbody>
</table>

11. Save and close starter_provider2.pkb.
12. Log in to OracleAS Portal as you normally would.

13. From the Portal Builder, click the **Administer** tab then the **Portlets** tab.

14. From the Portlet Repository portlet, click **Display Portlet Repository**.

15. Browse the repository and find the starter provider (typically it will appear in the Portlet Staging Area of the repository). It should contain its two original portlets: **hello world** and **snoop**.

16. From a command line prompt, start SQL*Plus and connect as the owner of the starter provider schema.

17. Compile the new and modified PL/SQL packages in the following order:
   - starter_provider2.pks
   - my_firstportlet.pks
   - starter_provider2.pkb
   - my_firstportlet.pkb

18. If any compilation errors occur, fix and recompile them until all of the packages compile successfully.

19. From the Portlet Repository portlet, click **Display Portlet Repository**.

20. Browse the repository and find the starter provider again. It should now contain your new portlet, **my_firstportlet**, in addition to its original portlets.

---

**Note:** If you make changes to an existing provider or the portlet record, you need to refresh your provider before seeing the changes reflected in your OracleAS Portal instance.

---

### 8.3.3 Adding Your Portlet to a Page

Your portlet should now be available for adding to pages like any other portlet in the Portlet Repository. To add your portlet to a page, follow the instructions in *Oracle Application Server Portal User’s Guide*.

### 8.4 Implementing Information Storage

OracleAS Portal provides APIs for storing and retrieving individual portlet preferences, and storing and manipulating temporary data for the current session:

- Implementing a Preference Store
- Implementing a Session Store

#### 8.4.1 Implementing a Preference Store

OracleAS Portal provides a set of APIs for storing and retrieving individual preferences for each unique portlet instance in a persistent manner. It provides a unique identifier for each individual, a preference store automatically mapped by user, and access mechanisms for storing and retrieving personalization information in your PL/SQL portlets.

By default, when you enable end-user personalization, **Personalize** appears on the title bar of your portlet. This link displays a form where users can choose settings for that portlet.
Implementing Information Storage

End-user personalization options are available through the wwpre_api_name and
wwpre_api_value packages.

8.4.1.1 Using a Preference Store
In general, you can set up preference storage as follows:
1.

Create the preference path with wwpre_api_name.create_path.

2.

Create the preference with wwpre_api_name.create_name.

3.

Set the preference values by providing the preference name and scoping level for
which you want to set the value. Use wwpre_api_value.set_value_as_
varchar2, set_value_as_number, or set_value_as_date for this purpose.

4.

Get preference values by providing the preference name and path whenever you
want to retrieve the preference value. Use wwpre_api_value.get_value_as_
varchar2, get_value_as_number, or get_value_as_date for this purpose.

8.4.1.2 Creating and Accessing a Preference Store
The services example, located in ..\pdkplsql\pdk\plsql\svcex in PDK-PL/SQL
(pdkplsql.zip), illustrates how you can implement preference storage. The objective
is to achieve the following functionality:
■

■

■

■

When a user clicks Personalize, they can enter text in two fields.
The first field prompts for personalized text. The second prompts for a
personalized portlet title.
The values the user enters for these two fields are stored in the preference store.
The personalized text and portlet titles are retrieved whenever that user invokes
the portlet instance.

You can browse through this example as follows to see how to create the preference
store, store values in it, and retrieve values from it:
1.

Open the services_portlet.pkb file in an editor.
The portlet path and preference names are provided with aliases in the constants
part of your portlet definition.
DOMAIN
SUBDOMAIN
PORTLET_PATH
PREFNAME_STRING
PREFNAME_TITLE

2.

constant
constant
constant
constant
constant

varchar2(30) :=
varchar2(32) :=
varchar2(256):=
varchar2(30) :=
varchar2(30) :=

'provider';
'services';
'oracle.portal.pdk.servicesportlet';
'services_string';
'services_title';

Find the register procedure. Your portlet needs to create a path for storing
preferences. To do so, it calls wwpre_api_name.create_path for creating the
preference path. It then calls wwpre_api_name.create_name for creating the
preference name, taking the portlet path, name, and description as input
parameters. Another input parameter is the p_type_name that indicates special
value types. The NLSID type indicates that the value stored is an NLS id. The
functions for setting and retrieving this type treat it as a number value. Apart from
that, when a preference store value of this type is exported or copied, so are its
associated strings. The last input parameter, the language, is obtained from a
context API.
procedure register
(
p_portlet_instance in wwpro_api_provider.portlet_instance_record

8-16 Oracle Application Server Portal Developer's Guide


Creating PL/SQL Portlets

3. The deregister procedure must eliminate the preference store with a call to `wwpre_api_name.delete_name`.

   ```sql
   procedure deregister
   (    
     p_portlet_instance in wwpro_api_provider.portlet_instance_record
   )
   is
   begin
     -- Delete the path used by the portlet instance. This will delete 
     -- all the names and all the values associated with the path.
     --
     wwpre_api_name.delete_path(
       p_path => PORTLET_PATH || p_portlet_instance.reference_path
     );
     exception
     when others then
       raise;
     end deregister;
   
4. The portlet must also get and set the values in the preference store using `wwpre_api_value.set_value` and `wwpre_api_value.get_value`. Find the `get_default_preference` function. Notice how this function loads the system level default values from the preference store. The default preferences are associated with an instance. The language strings are set in the database.
function get_default_preference
...
begin
--
-- Try to find a previously entered portlet instance string preference,
-- if any.
-- A portlet instance string preference is stored in the preference
-- store and has a level of SYSTEM_LEVEL_TYPE.
--
-- p_path => PORTLET_PATH || p_reference_path,
l_prefs.string_id := to_char(wwpre_api_value.get_value_as_number
  p_name => PREFNAME_STRING,
  p_level_type => wwpre_api_value.SYSTEM_LEVEL_TYPE
));
--
-- If the value returned here is null it is an indication that there
-- is no default string yet. Initialize the string id to 0 to indicate
-- this and load the default string value.
--
if (l_prefs.string_id is null or to_number(l_prefs.string_id) = 0) then
  wwpre_api_value.set_value_as_number(
    p_path => PORTLET_PATH || p_reference_path,
    p_name => PREFNAME_STRING,
    p_level_type => wwpre_api_value.SYSTEM_LEVEL_TYPE,
    p_level_name => null,
    p_value => 0
  );
... end get_default_preference;

5. Find the show procedure. Notice the behavior when the portlet is in Edit Defaults
or Edit mode. Note also how p_action is populated when the user clicks APPLY,
CANCEL, or OK. Once the form is submitted, the show procedure of the portlet is
called again and, if the p_action parameter is not null, then the save_prefs
procedure is called to save the personalizations and redirect to the relevant page.

procedure show
{
  p_portlet_record wwpro_api_provider.portlet_runtime_record
}
is
l_str varchar2(32000);
l_pref_record preference_record;
l_action varchar2(10);
l_names owa.vc_arr;
l_values owa.vc_arr;
begin
...
elsif (p_portlet_record.exec_mode =
  wwpro_api_provider.MODE_SHOW_EDIT)
  or (p_portlet_record.exec_mode =
    wwpro_api_provider.MODE_SHOW_EDIT_DEFAULTS)
then
  wwpro_api_parameters.retrieve(l_names, l_values);
  for i in 1..l_names.count loop
    if (upper(l_names(i)) = upper('p_string')) then
      l_pref_record.string := l_values(i);
    elsif l_names(i) = 'p_title' then
      l_pref_record.title_string := l_values(i);
    elsif l_names(i) = 'p_action' then
      l_pref_record.title_string := l_values(i);
    else
      l_names(i) = 'p_action' then
      l_pref_record.title_string := l_values(i);
    end if;
  end loop;
...
l_action := l_values(i);
end if;
end loop;
if (l_action in (ACTION_OK, ACTION_APPLY, ACTION_CANCEL)) then
  if (p_portlet_record.exec_mode = wwpro_api_provider.MODE_SHOW_EDIT) then
    save_prefs(p_string => l_pref_record.string,
               p_title  => l_pref_record.title_string,
               p_action => l_action,
               p_level  => wwpre_api_value.USER_LEVEL_TYPE,
               p_portlet_record  => p_portlet_record);
  else
    save_prefs(p_string => l_pref_record.string,
               p_title  => l_pref_record.title_string,
               p_action => l_action,
               p_level  => wwpre_api_value.SYSTEM_LEVEL_TYPE,
               p_portlet_record  => p_portlet_record);
  end if;
else
  show_edit(p_portlet_record  => p_portlet_record);
end if;
...
end show;

6. The show_edit procedure renders the page for Edit or Edit Defaults mode. It renders two text fields that allow the user to change the personalizable values in a form with three buttons (Apply, OK, and Cancel). Note that this function uses the wwpro_api_adapter.open_form to create the HTML form with the correct action attribute for the <FORM> tag and with the correct hidden fields. It is important to use this procedure to create the <FORM> tag if you want to use the portlet with the Federated Portal Adapter from remote OracleAS Portal instances.

procedure show_edit
  (p_portlet_record in wwpro_api_provider.portlet_runtime_record)
is
  l_prefs    preference_record;
  l_text_prompt_string  varchar2(30);
  l_title_prompt_string varchar2(30);
begin
  ...
7. Review the following procedures and functions, which are related to the preference storage implementation in this example:
   - `get_user_preference` retrieves the user personalized string and title for the portlet.
   - `save_prefs` is invoked to save the preferences to the preference store when the user clicks OK or Apply after making personalization changes.
   - `entered_text_is_valid` checks to see if the text entered in the personalizable text fields is valid.

8. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 8.3.2, "Implementing the Provider Package" for information on how to add it.

9. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in Oracle Application Server Portal User’s Guide.

8.4.2 Implementing a Session Store

The services example, located in ..\pdkplsql\pdk\plsql\svcex in PDK-PL/SQL (pdkplsql.zip), illustrates how you can implement a session store. The objective is to achieve the following functionality:

- When a user invokes this portlet, it displays text that reads: "This portlet has rendered x times in this session." x is the number of times the portlet has been rendered.
- Every time the user invokes the portlet, the counter increases by 1.
- Clicking Details in the portlet enables the user to reset the counter using Clear. After clearing the counter, the counter starts again from zero.

Creating and Accessing a Session Store

You can browse through this example as follows to see how to create the session store, store values in it, and retrieve values from it:

1. Open the services_portlet.pkb file in an editor.

   The domain and subdomain definitions for your session object are provided with aliases in the constants part of your portlet definition.

   ```sql
   DOMAIN           constant varchar2(30) := 'provider';
   SUBDOMAIN        constant varchar2(32) := 'services';
   PORTLET_PATH     constant varchar2(256):= 'oracle.portal.pdk.servicesportlet';
   PREFNAME_STRING  constant varchar2(30) := 'services_string';
   PREFNAME_TITLE   constant varchar2(30) := 'services_title';
   ```

2. Find the clear_count procedure. clear_count is called from the show procedure when the user clicks Clear to reset the counter. clear_count calls wwsto_api_session.load_session to load the session object. Then, it calls wwsto_api_session.set_attribute to set the counter to zero. Lastly, it saves the session object by calling save_session.

   ```sql
   procedure clear_count
   (p_action          in varchar2,
   p_back_url        in varchar2,
   p_reference_path  in varchar2)
   ```
implementing information storage

creating pl/sql portlets

3. Find the show_contents procedure. show_contents is called from the show procedure to retrieve the counter, increment it by one, and save the value in the session store. Notice how it retrieves the session object to display the number of times the user has rendered the portlet. It also retrieves the counter value with get_attribute_as_number and increments the counter for every invocation of this procedure.

procedure show_contents

begin
    -- In this mode a session counter is used to indicate
    -- the number of invocations of this portlet during the
    -- current session. The counter is stored in the session
    -- store.
    --
    session_parms :=
        &&1..wwsto_api_session.load_session (DOMAIN, SUBDOMAIN);
    ex_counter :=
        session_parms.get_attribute_as_number('ex_counter' || p_reference_path);
    --
    -- Reset the display counter.
    --
    ex_counter := 0;
    session_parms.set_attribute('ex_counter' || p_reference_path, ex_counter);
    --
    -- Save the changes to the database immediately to avoid any
    -- data consistency problems with the data stored in the
    -- session object.
    --
    session_parms.save_session;
end if;
owa_util.redirect_url(curl=>p_back_url);
end clear_count;

3. Find the show_contents procedure. show_contents is called from the show procedure to retrieve the counter, increment it by one, and save the value in the session store. Notice how it retrieves the session object to display the number of times the user has rendered the portlet. It also retrieves the counter value with get_attribute_as_number and increments the counter for every invocation of this procedure.

procedure show_contents

begin
    -- In this mode a session counter is used to indicate
    -- the number of invocations of this portlet during the
    -- current session. The counter is stored in the session
    -- store.
    --
    session_parms :=
        &&1..wwsto_api_session.load_session (DOMAIN, SUBDOMAIN);
    ex_counter :=
        session_parms.get_attribute_as_number('ex_counter' || p_reference_path);
    --
    -- Reset the display counter.
    --
    ex_counter := 0;
    session_parms.set_attribute('ex_counter' || p_reference_path, ex_counter);
    --
    -- Save the changes to the database immediately to avoid any
    -- data consistency problems with the data stored in the
    -- session object.
    --
    session_parms.save_session;
end if;
owa_util.redirect_url(curl=>p_back_url);
end clear_count;
8.5 Using Parameters

The functionality of portlets can be extended with the help of parameters. The business logic implemented by portlets may produce different HTML output depending on the parameters passed to the page. By using portlet parameters, you can navigate within the portlet in Shared Screen mode without changing the current page. Portlets can also communicate with each other through parameters.

Portlet parameters are structured as name-value pairs. These pairs map directly to the URL parameter passing format by using the GET submission method or can use the HTTP message body by using the POST submission method. Portlets can also expose their parameters to OracleAS Portal. When added to a page, these portlets can accept values in the form of page parameters created by the page designer.

---

**Note:** Portlet parameter names should not start with an underscore (_) because those parameters are reserved for internal use by OracleAS Portal and are not passed to the portlet.

---

Portlets do not have direct access to the URL, the HTTP message body, or the page parameters. To retrieve the parameter values, portlets must call the OracleAS Portal PL/SQL parameter APIs provided in the `wwpro_api_parameters` package.

OracleAS Portal offers the following types of parameters:

---

**Caution:** You cannot mix the usage of public and private parameters in a portlet. To enable public parameters for your portlet, you must take steps that preclude the usage of private parameters and vice versa.

---

- **Private portlet parameters** enable the implementation of internal navigation in your portlet.
Using Parameters

- **Public portlet parameters** let you pass control over the data flow of your portlet to the page designer. The page designer can map the public portlet parameters to their page parameters, provide default values, and allow users to personalize those values.

- **Page parameters** are defined in a simple user interface by page designers. These page parameters can be mapped to public portlet parameters in order for the page designer to pass parameter values from the page to the portlets on it.

For more information about parameters, refer to Section 2.12, "Public Portlet Parameters Support" and Section 2.13, "Private Portlet Parameter Support".

### 8.5.1 Passing Private Parameters

You can use either **GET** or **POST** HTML submission methods when passing private portlet parameters. The **GET** method uses the URL to pass the parameters, whereas the **POST** method places the parameters in an HTTP message body. For both methods, you must specify the portlet instance on the portal page, how the parameter is called, and the value of the parameter.

There are the following two types of private portlet parameters:

- **Qualified parameters** ensure that a private portlet parameter is not read by any other portlet on the page. The reference path, which is assigned when the portlet is added to a page, is the unique prefix of the parameter. For example, http://page_url?277_MAP_368673.region=Europe. The qualified parameter's reference path is 277_MAP_368673, the name is region, and the value is Europe. For private parameters, we strongly recommend that you always use qualified parameters.

- **Unqualified parameters** have no information about the portlet instance and can be read by any portlet on the page. For example, http://page_url?region=Europe. The unqualified parameter's name is region and its value is Europe. For private parameters, we strongly recommend that you avoid unqualified parameters.

### 8.5.2 Passing Page Parameters and Mapping Public Portlet Parameters

Public portlet parameters enhance the flexibility of your portlets by enabling page designers to reuse your portlets on multiple pages. As a result, page designers do not have to ask you to make changes to the portlet code when adding the portlet to different pages. By using public portlet parameters, any portlet on a page can easily receive its value from the mapped page parameter, regardless of the portlet parameter name.

For example, suppose you have a page parameter named `dept_id`. Three portlets need this value, but one portlet calls it `dept`, another calls it `deptno`, and still another `department_id`. Mapping the page parameter enables all three portlets to receive the value from the `dept_id` parameter and place it in the appropriate portlet parameter. Furthermore, the page designer may set a default value (for example, department 20) that can be personalized by users (for example, department 30) and applied to all three portlets.

The general model for passing public and page parameters is as follows:

1. Enable public parameters in the portlet record by setting `pass_all_url_params` to `false`. This ensures that the portlet is only passed parameters intended for that portlet.
2. Declare the public parameters in the provider's `describe_portlet_parameters` function. For each of the portlets that belong to the provider, this procedure should return a list of the parameters that the portlet accepts in the form of a PL/SQL table of records of the type:

```plsql
type portlet_parameter_table is table of portlet_parameter_record index by binary_integer;
```

3. Provide descriptive information for the parameters in the portlet's `describe_parameters` function. For example:

```plsql
function describe_parameters
(p_provider_id in integer, p_language in varchar2)
return wwpro_api_provider.portlet_parameter_table
is
    l_params wwpro_api_provider.portlet_parameter_table;
    begin
        l_params(1).name := 'dept_id';
        l_params(1).datatype := wwpro_api_provider.STRING_TYPE;
        l_params(1).description := 'Defines a department ID';
        l_params(1).display_name := 'Department ID';
        return l_params;
    end describe_parameters;
```

4. Assign values to the public parameters. Public parameters typically get their values through page parameters. Page parameters are usually assigned default values by the page designer and the user can then personalize the value at runtime. Alternatively, page parameter values can be assigned in the calling URL. For more information about how page designers can use page parameters, refer to the [Oracle Application Server Portal User's Guide](#).

### 8.5.3 Retrieving Parameter Values

Regardless of whether you are using private or public parameters, you use the same APIs to retrieve their values. Portlets obtain their parameters by calling the PL/SQL parameter APIs in the `wwpro_api_parameters` package:

- `wwpro_api_parameters.get_value` returns the parameter value that is specified by a given parameter name. Parameter names are not case sensitive, whereas parameter values are case sensitive. For example:

  ```plsql
  l_region := wwpro_api_parameters.get_value
  (p_name => 'region',
   p_reference_path => p_portlet_record.reference_path);
  ```

- `wwpro_api_parameters.get_values` returns an array of parameter values. This function returns all the values that are associated with a single parameter name or an empty list if no matches are found. Some business logic may require multiple selections, when multiple values are passed to the portlet by using the same parameter name. Portlets can take one or more values of the same parameter. For example:

  ```plsql
  l_region_values owa.vc_arr;
  ...
  l_region_values := wwpro_api_parameters.get_values
  (p_name = 'region',
   p_reference_path => p_portlet_record.reference_path);
  ```
8.6 Accessing Context Information

Whenever a user accesses a page in OracleAS Portal, a public session is established. When the user logs in to OracleAS Portal, the public session becomes an authenticated session. This session contains several pieces of context information about the user, such as user name, current session ID, IP address, and language preference. It also includes supporting information such as the OracleAS Portal schema currently in use.

Session context services return information about a user’s session and are available through the `wwctx_api` package.

8.6.1 Using Context Information

The general model for working with the session context is as follows:

1. Identify the piece of information you require for your functionality.
2. Use the appropriate method from `wwctx_api` to get and optionally set this value.

Table 8–5 lists the function calls used to obtain the various pieces of session information.
Note: For more information on the context APIs, see the PL/SQL API Reference. The API Reference can be found on Portal Center (http://portalcenter.oracle.com) or, if you downloaded PDK-PL/SQL (pdkplsql.zip), in ..\pdkplsql\pdk\plsql\doc.

<table>
<thead>
<tr>
<th>Session Information</th>
<th>Function Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current user</td>
<td>wwcxt_api.get_user</td>
</tr>
<tr>
<td>Login status of user</td>
<td>wwcxt_api.is_logged_on</td>
</tr>
<tr>
<td>Login time</td>
<td>wwcxt_api.get_login_time</td>
</tr>
<tr>
<td>Language</td>
<td>wwcxt_api.get_nls_language</td>
</tr>
<tr>
<td>Current session id</td>
<td>wwcxt_api.get_sessionid</td>
</tr>
<tr>
<td>IP address of user client</td>
<td>wwcxt_api.get_ip_address</td>
</tr>
<tr>
<td>User schema</td>
<td>wwcxt_api.get_db_user</td>
</tr>
<tr>
<td>OracleAS Portal schema</td>
<td>wwcxt_api.get_product_schema</td>
</tr>
<tr>
<td>OracleAS Portal version</td>
<td>wwcxt_api.get_product_version</td>
</tr>
</tbody>
</table>

8.6.2 Using wwcxt_api to Obtain Context Information

The services example, located in ..\pdkplsql\pdk\plsql\svcex in PDK-PL/SQL (pdkplsql.zip), illustrates how you can obtain session information using the wwcxt_api package. You can browse through this example as follows to see how the function calls are implemented in a portlet:

1. Open the services_portlet.pkb file in an editor.
2. Find the get_portlet_info function.
3. Notice the usage of wwcxt_api.get_user to derive the user information and set that value in the portlet information record:

   ...  
   l_portlet.timeout := null;  
   l_portlet.timeout_msg := null;  
   l_portlet.created_on := to_date('10/19/2000', 'MM/DD/YYYY');  
   l_portlet.created_by := wwcxt_api.get_user;  
   l_portlet.last_updated_on := to_date('10/19/2000', 'MM/DD/YYYY');  
   l_portlet.last_updated_by := wwcxt_api.get_user;  
   l_portlet.has_show_edit_defaults := true;  
   l_portlet.has_show_preview := true;  
   l_portlet.preference_store_path := PORTLET_PATH;  
   ...  

4. wwcxt_api.get_user is used similarly in various places throughout services_portlet.pkb. Search the code for other occurrences of wwcxt_api.get_user.

5. Another example of getting context information occurs in the isRunnable function:

   function isRunnable
   
   p_provider_id in integer
   ,p_reference_path in varchar2

8-26 Oracle Application Server Portal Developer's Guide
Implementing Portlet Security

Portlet security refers to the techniques and methods used by portlets to control their access by end users. The portlets leave authentication to OracleAS Portal and trust that the portal will return them to the correct, validated user upon request.

OracleAS Portal strictly controls access to information and applications by assigning specific privileges to users and groups. Portal security services allow you to specify access control programmatically and check for the appropriate privileges at runtime. Security mechanisms used by portlets ensure that only authorized users gain access to these portlets. These security services are available through the \texttt{wwsec_api} package.
Portlet security is invoked when a portlet is displayed on a portal page and when a portlet is returned in a portlet list by the getportlet_list function for database providers. Security services in the Portal framework have the following key features:

- **Portlet Display:** Before a portlet is displayed on a page, the provider checks for the portlet's access privileges. The provider needs to define the isportletRunnable function which calls the portlet's isRunnable function to check access privileges.

- **User Group:** You can find which default group a user belongs to by using the wwsec_api.getdefaultgroup function.

- **Check Privileges:** You can find whether a user or group has the required privileges to personalize a portlet by using the wwsec_api.has_privilege function.

- **Highest Privilege:** You can find the highest available privilege of a user across all groups by using the wwsec_api.get_privilege_level function.

- **Accessible Objects:** You can find all the objects to which a user has access, given a privilege level, by using the wwsec_api.accessible_objects function. You can find other similar associated functions in the API documentation. The API Reference can be found on Portal Center (http://portalcenter.oracle.com) or, if you downloaded PDK-PL/SQL (pdkplsql.zip), in ..\pdkplsql\pdk\plsql\doc.

### 8.7.1 Using Security

To implement PL/SQL portlet security, the portal requires the function isportletRunnable be implemented by database providers. The actual implementation of this function is up to the application; that is, the security scheme that determines whether the current user has enough privileges to access the portlet is defined by the individual portlet implementation. The portal also requires the function getportlet_list for database providers to return the set of portlets that are accessible by the current user.

#### Guidelines for Using the Security APIs

The portlet security mechanism may use the context and security subsystem APIs and infrastructure. The context APIs can be used to retrieve information about the current user. The security subsystem can be used to check the privileges of the current user.

---

**Note:** For more information on the context and security subsystem APIs, see the PL/SQL API Reference. The API Reference can be found on Portal Center (http://portalcenter.oracle.com) or, if you downloaded PDK-PL/SQL (pdkplsql.zip), in ..\pdkplsql\pdk\plsql\doc.

---

While using these APIs, keep in mind the following:

- **Only authorized users should be able to see your portlet in the Add Portlet dialog.** This objective can be accomplished by implementing the isportletRunnable function in the provider. You can also allow public access to your portlet.

- **If a portlet does not want to render itself to a user, it should return no HTML or return an exception that the page engine will ignore.** It should not return an error message. Doing so adds unnecessarily to the error stack, which has its limits. Refer to Section 8.9, “Implementing Error Handling” for more information.
Portlet security allows the portlet to perform a runtime security check to ensure that the current user has the necessary authorization to access the portlet.

When a portlet is rendered in Show mode, it may call the isRunnable function for database providers to determine whether the portlet should be displayed for the currently logged on user. The portal does not make the call to this function directly. It is not a requirement, however, for the portlet to make this call. The portlet should make this call in its Show mode only if it implements portlet security.

The result of the call to isRunnable determines whether the portlet is actually displayed. If the result is true, the portlet displays; otherwise it does not display. The portlet is rendered in Show mode when it is displayed in a portal page.

When a portlet is returned in a portlet list by a call to the provider function getPortletList, the value of the p_security_level parameter determines the purpose of the function call. When the call is made from the Portlet Repository refresh operation in order to retrieve the master list of portlets that the provider implements, the parameter p_security_level has a value of false. This setting indicates to the provider that no portlet security check should be made and a master list of all the portlets that the provider implements must be returned. The master list of portlets returned in this case is used to populate the Portlet Repository for that provider.

If the value of p_security_level is true, then it is up to the provider implementation to decide whether portlet security should be performed. If portlet security is implemented, the provider may return a different list of portlets depending on the current user.

When the Portlet Repository is displayed, OracleAS Portal calls the isPortletRunnable function for database providers for each of the portlets that exist in the Portlet Repository. This step is done to display only the portlets that the currently logged on user is authorized to see. One example where the Portlet Repository is displayed is in the Add Portlets dialog.

### 8.7.2 Coding Security

The services example, located in ..\pdk\pdk\pdkplsql\plsql\svcex in PDK-PL/SQL (pdkplsql.zip), illustrates how you can implement security. You can browse through this example as follows to see how the security functions are implemented in a portlet:

1. Open the services_provider.pkb file in an editor.
2. Find the isPortletRunnable function. This function calls the security implementation through the portlet’s isRunnable function to check portlet access privileges.

```plsql
function is_portlet_runnable
{
    p_portlet_instance in wwpro_api_provider.portlet_instance_record
} return boolean
is begin
    if (p_portlet_instance.portlet_id = SERVICES_PORTLET_ID) then
    return services_portlet.is_runnable(
        p_provider_id => p_portlet_instance.provider_id
        ,p_reference_path => p_portlet_instance.reference_path
    );
```
else
    raise wwpro_api_provider.PORTLET_NOT_FOUND_EXCEPTION;
end if;
end is_portletRunnable;

3. Find the get_portlet_list procedure. get_portlet_list allows the portlet to be included in the list of portlets implemented by this provider. get_portlet_list first checks the security flag (p_security_level) to find out whether security is enabled. If the flag is set to true, get_portlet_list uses is_runnable to check whether the portlet is accessible. The value of the p_security_level parameter indicates whether to perform security checks before returning a portlet in the list. When a portlet repository refresh operation retrieves the master list of portlets implemented by the provider, p_security_level has a value of false. A value of false means the provider does not need to perform a security check and that a master list of all of the portlets implemented by the provider must be returned. The master list of portlets returned is used to populate the portlet repository for that provider. If the value of p_security_level is true, then the provider implementation decides whether to perform portlet security checks. If portlet security is implemented, the provider may return a different list of portlets depending on the currently logged on user.

function get_portlet_list
...
    if (p_security_level = false) then
        l_cnt := l_cnt + 1;
        l_portlet_list(l_cnt) := get_portlet(
            p_provider_id => p_provider_id,
            p_portlet_id => SERVICES_PORTLET_ID,
            p_language => p_language
        );
    else
        if (services_portlet.is_runnable(
            p_provider_id => p_provider_id,
            p_reference_path => null)
        ) then
            l_cnt := l_cnt + 1;
            l_portlet_list(l_cnt) := get_portlet(
                p_provider_id => p_provider_id,
                p_portlet_id => SERVICES_PORTLET_ID,
                p_language => p_language
            );
        end if;
    end if;
end get_portlet_list;

4. Open the services_portlet.pkb file in an editor.

5. Find the show procedure. Before displaying a portlet, the show procedure runs a security check to determine whether the current user is allowed to see the portlet.

procedure show
...
    -- Perform a security check
    if (not is_runnable(
        p_provider_id => p_portlet_record.provider_id,
        p_reference_path => p_portlet_record.reference_path)
    ) then
        wwerr_api_error.add(
            DOMAIN, SUBDOMAIN,
            'securityerr', 'services_portlet.show');
raise wwpro_api_provider.PORTLET_SECURITY_EXCEPTION;
end if;
...
end show;

6. Find the `is_runnable` function. `is_runnable` is the place where you implement your security checks. In this example, the security check is quite simple. If the user is logged on (that is, not in a public session), then the function returns `true` and the portlet is displayed to the user. For your own purposes, you could, of course, code much more complex security checks in the `is_runnable` function.

```plsql
function is_runnable
(
    p_provider_id in integer,
    p_reference_path in varchar2
) return boolean is
begin
    -- Portlet security check. It allows the portlet to be visible
    -- if the user is logged on, that is, the current session is not a
    -- public session.
    return wwctx_api.is_logged_on;
end is_runnable;
```

7. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 8.3.2, "Implementing the Provider Package" for information on how to add it.

8. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in Oracle Application Server Portal User’s Guide.

### 8.8 Improving Portlet Performance with Caching

OracleAS Portal provides for the caching of PL/SQL portlets. This functionality permits PL/SQL portlets to cache their Web content on the middle tier. Subsequent requests for the content may be retrieved from the cache, with or without validation from the database, decreasing the database workload.

OracleAS Portal provides the following three types of caching for your PL/SQL portlets:

- **Validation-based caching** compares a key value to check whether the contents of the cache are still valid. If the key value does not change, it uses the cached content. Otherwise, it makes a round trip to the portal node to fetch the portlet content.

- **Expiry-based caching** uses a given expiration period for the contents of the cache when rendering the portlet. This form of caching is useful for content that changes infrequently or at very regular intervals (for example, every day at the close of business).

- **Invalidation-based caching** is the most complex form of caching but also the most flexible. The objects in OracleAS Web Cache are considered valid as long as they are not invalidated explicitly. You can also combine invalidation-based caching with either expiry-based or validation-based caching.
Because OracleAS Portal supports user personalization of pages and portlets, the view of a page can vary from one user to another. OracleAS Portal's caching is designed to allow content to vary for each user, even if the URL is the same across all users. Therefore, portal objects can be cached at either the user level or the system level, as follows:

- **User-level caching** is for a specific user. The cache entries are unique for that user and cannot be accessed by other users.
- **System-level caching** is for all users. One cache entry is used for all users. Examples of content that might be suitable for system-level caching are page banners and news portlets.

When a database provider issues a request for a portlet, the request is sent to the portlet's `show` procedure. This procedure accepts the `portlet_runtime_record` as a parameter. This record structure contains fields that can be examined and set by the portlet to enable caching. The caching control fields of this record are as follows:

- **caching_key**: This value is communicated in the `ETAG` header for this request and returned back to the portlet provider in subsequent requests. Setting this field enables validation-based caching.
- **caching_period**: This field enables expiry-based caching. The value is the number of minutes the content should be held in the cache. This mode overrides validation-based caching. If a value is set for this field, then the `caching_key` field is ignored.
- **caching_level**: This field defines whether the content is meant for general use or for a specific user. The valid values are `SYSTEM` and `USER`.

### 8.8.1 Using Caching

The general model for working with portlet caching varies according to the type of caching you choose. To a great extent, the type of caching you choose depends on the portlet content. If the portlet content changes at fairly regular intervals (for example, at the close of business every day), then it probably makes sense to use expiry-based caching. If the portlet content changes at irregular intervals, then validation- or invalidation-based caching is probably best.

#### 8.8.1.1 Validation-Based Caching

If you choose validation-based caching, the general model is as follows:

1. Set the `caching_key` field of the `portlet_runtime_record` parameter. Add a check to compare the value of the current key with the value of the `caching_key` field of the `portlet_runtime_record` parameter. Note that the first time the `show` procedure is called, the key is null and its value must be set.

2. Determine whether you want to use system or user level caching. Set the `caching_level` field of the `portlet_runtime_record` parameter accordingly.

#### 8.8.1.2 Expiry-Based Caching

If you choose expiry-based caching, the general model is as follows:

1. Set the `caching_period` field of the `portlet_runtime_record` parameter to the desired interval for the cache (in minutes).
2. Determine whether you want to use system or user level caching. Set the `caching_level` field of the `portlet_runtime_record` parameter accordingly.

### 8.8.1.3 Invalidation-Based Caching

If you choose invalidation-based caching, the general model is as follows:

1. Indicate to OracleAS Portal that it must generate specific headers for OracleAS Web Cache by calling `wwpro_api_provider.USE_INVALIDATION`.

2. Determine whether you want to use system or user level caching. Set the `caching_level` field of the `portlet_runtime_record` parameter accordingly.

3. Optionally, set up validation- or expiry-based caching as well.

4. Add invalidation logic to your portlet where needed (for example, when the portlet is personalized) and make appropriate calls to `wwpro_api_invalidation`.

### 8.8.2 Configuring and Monitoring the Cache

The *Oracle Application Server Portal Configuration Guide* describes how to configure caching as well as how to monitor and tune performance.

### 8.8.3 Implementing Validation-Based Caching

The caching example, located in `..\pdkplsql\pdk\pls\cache` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can implement validation and expiry-based caching. You can browse through this example as follows to see how the validation-based functions are implemented in a portlet:

1. Open the `validcache_portlet.pkb` file in an editor.

2. At the very top of the file, notice the aliases for the caching level constants.

   ```sql
   CREATE OR REPLACE
   package body VALIDCACHE_PORTLET
   is
   -- Caching Constants
   CACHE_LEVEL_SYSTEM constant varchar2(10) := 'SYSTEM';
   CACHE_LEVEL_USER   constant varchar2(10) := 'USER';
   ```

3. Find the `show` procedure. Notice first that the `p_portlet_record` is an `in` and `out` parameter for this procedure.

   ```sql
   procedure show
   (p_portlet_record in out
   wwpro_api_provider.portlet_runtime_record
   )
   ```

4. In the procedure’s security check, the caching fields of `p_portlet_record` are set to null if the security check fails.

   ```sql
   begin
   if (not is_runnable(
       p_provider_id   => p_portlet_record.provider_id,
       p_reference_path => p_portlet_record.reference_path)
   ) then
       -- Set it to null so that cache does not get used even if exists
       p_portlet_record.caching_level := null;
   ```
5. After that, the procedure calls the `get_cache_key` function to get the cache key’s value and assign it to a temporary value:

```sql
--
-- CACHE IS VALID?
--
l_cache_key := get_cache_key();
```

6. Find the `get_cache_key` function, which is referenced from the `show` procedure. This function generates a key for the portlet. You can implement your own logic here based upon your portlet’s requirements.

```sql
function get_cache_key
return varchar2
is
  l_date date;
begin
  select sysdate into l_date from dual;
  return trim(substr(to_char(l_date, 'YYYY:MM:DD:HH:MI:SS'),1,18));
exception
  when others then
    null;
end get_cache_key;
```

7. Now return to the `show` procedure. Notice how the code checks your `portlet_runtime_record` parameter for the current values of the `caching_key` and the `caching_level`. This same piece of code can compare your `caching_key` values.

```sql
if p_portlet_record.caching_level = CACHE_LEVEL_SYSTEM then
  if l_cache_key is not null then
    -- Cache exists for the user, overwrite it
    p_portlet_record.caching_level := CACHE_LEVEL_USER;
    p_portlet_record.caching_key := l_cache_key;
  else
    return; -- System cache is still valid.
  end if;
elsif p_portlet_record.caching_level = CACHE_LEVEL_USER then
  if p_portlet_record.caching_key != l_cache_key then
    -- cache has expired. reset it
    p_portlet_record.caching_key := l_cache_key;
  else
    return; -- User cache is good as gold
  end if;
elsif p_portlet_record.caching_level is null then
  if p_portlet_record.caching_key is not null then
    -- Cache does not exists for the user, create it
    p_portlet_record.caching_level := CACHE_LEVEL_USER;
    p_portlet_record.caching_key := l_cache_key;
  else
    -- Define a sytem cache. This can happen only once!
    -- the first time the portlet is rendered.
    p_portlet_record.caching_level := CACHE_LEVEL_SYSTEM;
    p_portlet_record.caching_key := 'MY_INITIAL_CACHE_KEY';
  end if;
else
  p_portlet_record.caching_level := CACHE_LEVEL_SYSTEM;
```
Improving Portlet Performance with Caching

Creating PL/SQL Portlets

8. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 8.3.2, "Implementing the Provider Package" for information on how to add it.

9. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in Oracle Application Server Portal User’s Guide.

8.8.4 Implementing Expiry-Based Caching

The caching example, located in ..\pdkplsql\pdk\plsql\cache in PDK-PL/SQL (pdkplsql.zip), illustrates how you can implement expiry-based caching. You can browse through this example as follows to see how the expiry-based functions are implemented in a portlet:

1. Open the expirycache_portlet.pkb file in an editor.

2. At the very top of the file, notice the aliases for the caching level constants.

   ```sql
   CREATE OR REPLACE
   package body VALIDCACHE_PORTLET
   is
     -- Caching Constants
     CACHE_LEVEL_SYSTEM constant varchar2(10) := 'SYSTEM';
     CACHE_LEVEL_USER   constant varchar2(10) := 'USER';
   end if;
   
   8. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 8.3.2, "Implementing the Provider Package" for information on how to add it.
   
   9. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in Oracle Application Server Portal User’s Guide.

   8.8.4 Implementing Expiry-Based Caching

   The caching example, located in ..\pdkplsql\pdk\plsql\cache in PDK-PL/SQL (pdkplsql.zip), illustrates how you can implement expiry-based caching. You can browse through this example as follows to see how the expiry-based functions are implemented in a portlet:

   1. Open the expirycache_portlet.pkb file in an editor.

   2. At the very top of the file, notice the aliases for the caching level constants.

      ```sql
      CREATE OR REPLACE
      package body VALIDCACHE_PORTLET
      is
        -- Caching Constants
        CACHE_LEVEL_SYSTEM constant varchar2(10) := 'SYSTEM';
        CACHE_LEVEL_USER   constant varchar2(10) := 'USER';
      end if;
      
      3. Find the show procedure. Notice first that the p_portlet_record is an in and out parameter for this procedure.

         ```sql
         procedure show
         (p_portlet_record in out wwpro_api_provider.portlet_runtime_record)
         end if;
         
         5. After that, the procedure sets the value of the caching period in minutes in a temporary variable:

            ```sql
            -- Set the Caching Period to one minute
            l_cache_period := 1;
            
            6. Next, notice how the code checks your portlet_runtime_record parameter for the current values of the caching_period and sets the caching_period accordingly. This same piece of code can compare your caching_period values.

               ```sql
               if p_portlet_record.caching_level = CACHE_LEVEL_SYSTEM then
-- Cache does not exists for the user, create it
p_portlet_record.caching_level := CACHE_LEVEL_USER;
p_portlet_record.caching_period := l_cache_period;
elsif p_portlet_record.caching_level = CACHE_LEVEL_USER then
-- Cache exists for the user, overwrite it
p_portlet_record.caching_period := l_cache_period;
elsif p_portlet_record.caching_level is null then
if p_portlet_record.caching_period is not null then
-- Cache does not exists for the user, create it
p_portlet_record.caching_level := CACHE_LEVEL_USER;
p_portlet_record.caching_period := l_cache_period;
else
-- Define a system cache. This can happen only once!
p_portlet_record.caching_level := CACHE_LEVEL_SYSTEM;
p_portlet_record.caching_period := l_cache_period;
end if;
else -- p_portlet_record.caching_level value is messed up!
p_portlet_record.caching_level := CACHE_LEVEL_SYSTEM;
p_portlet_record.caching_period := l_cache_period;
end if;

7. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 8.3.2, "Implementing the Provider Package" for information on how to add it.

8. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in Oracle Application Server Portal User’s Guide.

8.8.5 Implementing Invalidation-Based Caching

Suppose you have a portlet that displays a map of the world, map_portlet.pkb and map_portlet.pks. You would go about adding invalidation-based functions to it as follows:

1. In the show procedure, you need to add a call to wwpro_api_provider.use_invalidation. This call indicates to OracleAS Portal that the portlet content should be cached by OracleAS Web Cache. Note that we have also specified that the content be cached at the user level and that expiry-based caching be used as well (that is, an expiration interval of one minute has been set).

procedure show
... if (p_portlet_record.exec_mode = wwpro_api_provider.MODE_SHOW) then
    p_portlet_record.caching_invalidation :=
    wwpro_api_provider.use_invalidation;
    p_portlet_record.caching_level := 'USER';
    p_portlet_record.caching_period := 1;
...  

2. Create a procedure in your map_portlet.pkb file that invalidates the cache. For example:

procedure map_invalidation
{ p_provider_id in number,
  p_portlet_id in number,
  p_instance_id in varchar2,
  p_page_url in varchar2
}
Implementing Error Handling

OracleAS Portal provides the capability for you to trap erroneous input and return meaningful error messages. It manages the internal error stack by tracking the raised exceptions and retaining information about them. OracleAS Portal also includes a set of APIs for presenting errors in a standardized way.

Error handling services are available through the `wwerr_api_error` and `wwerr_api_error_ui` packages. These error handling services include the following key features:

- **Error stack.** OracleAS Portal uses an error stack to keep track of the error messages. When an error occurs, the error message is pushed onto an error stack. Whenever procedures or function calls are nested, the error stack keeps track of all the error messages. You can choose to retrieve only the top error (or most recent error) by using the `wwerr_api_error.get_top` method. Alternatively, you can get all the error messages on the stack using the `wwerr_api_error.get_errors` function. The stack can also be checked for the presence of any errors by calling the `wwerr_api_error.is_empty` function.

- **Error messages.** Error handling services provide a way to define meaningful error messages. To define your own error messages, you need to define their name space. The name space consists of the following:
  - **Name** is the error name.
Implementing Error Handling

- **Domain** is the area of the product where the error occurred.
- **Subdomain** is the subsystem where the error occurred.
- **Context** is the name of the function where the error occurred.

The name space uniquely identifies your error message. If it does not do so, a wwc-0000 error message is generated.

The default domains include the portal (WWC), application (WWV), and page groups (WWS). Each domain is further classified into subdomains, which define the object types. The portal domain includes the portlet, page, and document object types. The application domain includes object types such as forms, menus, reports, and charts. The page group domain includes object types such as pages, items, categories, and perspectives. If you need to define an error that does not fall within these classifications, you can define your own domain with subdomains for your errors.

- **Message parameters.** The other language strings that you create for your errors can take substitution parameters for your messages. The $p1, p2, p3...$ parameters can be used to pass substitution parameters to the error messages. For example, for this string:

  (domain='yahoo', subdomain='provider', name='generalerror', string='Error: %1')

an error can be added as follows:

```
wwerr_api_error.add(p_domain=>'yahoo', p_sub_domain=>'provider',
  p_name=>'generalerror', p_context=>'yahoo.show', p1=> sqlerrm);
```

- **Error display.** The wwerr_api_error_ui package provides a means to generate a standard user interface for displaying the errors in OracleAS Portal. The error messages can be displayed in the following two different ways:

  - **Full screen user interface:** These error messages are displayed in a full screen mode. You may want to display full screen errors when the system encounters fatal or show-stopper errors.
  - **Inline user interface:** These error messages are displayed within the current page itself. You may use inline errors for minor errors or warnings.

Additionally, you can choose the output format of the display (HTML, XML, or ASCII text).

### 8.9.1 Using Error Handling

In general, you set up error handling as follows:

1. On detecting error conditions, add the error message, with an appropriate domain and sub-domain combination, to the stack using the wwerr_api_error.add procedure.
2. When necessary (for example, at the end of a routine), expose the error messages using the wwerr_api_error_ui procedures. To display full screen messages, use the procedures show_html, show_xml, or show_text depending on your preferred output type. To display inline messages, use the procedures show_inline_html, show_inline_xml, or show_inline_text, depending on the output type you desire.

**Guidelines for Error Handling**

While implementing error handling, keep in mind the following:
While defining your own error messages, use your own error domain for these messages. Never use the WWC, WWV, or WWS domain for your error messages. You will need to write a small loader script to load these into the other language tables.

Avoid unnecessary error messages. If you do not want to do anything in a function, just return null rather than an error. For example, suppose you are coding a copyportlet procedure for your portlet because the provider calls it for all of its other portlets. If you do not wish the copyportlet procedure for this particular portlet to do anything, then simply have it return null. If you return errors, it will unnecessarily disrupt the portlet functionality.

A maximum of ten error messages is kept on the stack. Beyond ten, messages are ignored when a call to wwerr_api_error.add is made.

Use the API as a programmatic way of finding the problem. You can use the non-user-interface format for this purpose. For example, when you programmatically register a provider, the exception block can use get_text_stack to get the error messages and print them. This approach helps when debugging calls to public APIs since all of them add errors to the stack for exceptions.

Remember to seed the other language strings for your error messages. For more information, refer to Section 8.11, "Writing Multilingual Portlets".

The standard user interface for error messages provides a navigation link back to the previous page. It also includes a Help icon for the specified help URL.

8.9.2 Adding Error Handling

The services example, located in ..\pdkplsql\pdk\plsql\svcex in PDK-PL/SQL (pdkplsql.zip), illustrates how you can implement error handling. You can browse through the following example to see how the error handling functions are implemented in a portlet:

1. Open the servicesportlet.pkb file in an editor.
2. The domain and subdomain definitions for your error messages are provided with aliases in the constants part of your portlet definition.

```sql
DOMAIN           constant varchar2(30) := 'provider';
SUBDOMAIN        constant varchar2(32) := 'services';
PORTLET_PATH     constant varchar2(256) := 'oracle.portal.pdk.servicesportlet';
PREFNAME_STRING  constant varchar2(30) := 'services_string';
PREFNAME_TITLE   constant varchar2(30) := 'services_title';
```
3. Find the show procedure. This procedure performs a security check and, if an error condition arises, it calls wwerr_api_error.add to push the securityerr error message onto the stack.

```sql
procedure show
{
  p_portlet_record        wwpro_api_provider.portlet_runtime_record
}
```

```sql
begin
  -- Perform a security check
  if (not is_runnable(
    p_provider_id     =>  p_portlet_record.provider_id
    ,p_reference_path  =>  p_portlet_record.reference_path)
  ) then
```
wwerr_api_error.add(
  DOMAIN, SUBDOMAIN,
  'securityerr', 'services_portlet.show');
raise wwpro_api_provider.PORTLET_SECURITY_EXCEPTION;
end if;

4. The show procedure also checks for any other kind of execution mode and generates an appropriate error message for an invalid display mode.

if (p_portlet_record.exec_mode = wwpro_api_provider.MODE_SHOW) then ...
elsif (p_portlet_record.exec_mode = wwpro_api_provider.MODE_SHOW_EDIT) ...
else
  wwerr_api_error.add(DOMAIN, SUBDOMAIN,
    'invaliddispmode', 'services_portlet.show');
  raise wwpro_api_provider.PORTLET_EXECUTION_EXCEPTION;
end if;

5. Lastly, the show procedure implements a general error message in the exception handler to catch any errors not trapped by the preceding conditions.

exception when others then
  wwerr_api_error.add(
    DOMAIN, SUBDOMAIN,
    'generalerr', 'services_portlet.show');
  raise wwpro_api_provider.PORTLET_EXECUTION_EXCEPTION;
end show;

6. Error handling is also implemented in the save_prefs and save_default_prefs procedures. They check whether the error stack is empty and, if it is not, the portlet makes a call to wwerr_api_error.show_html to display the error in full screen mode.

exception when INVALID_TEXT_EXCEPTION then
  l_information := l_user||'%'||l_time
  ||'INVALID_TEXT_EXCEPTION%'||p_string;
  l_action := LOG_FAILED;
  wwlog_api.log (p_domain => DOMAIN,
    p_subdomain => SUBDOMAIN,
    p_name => l_user,
    p_action => l_action,
    p_information => l_information,
    p_url => l_url,
    p_row_count => 0,
    p_elapsed_time=> l_elapsed_time);
  wwerr_api_error.add(DOMAIN, SUBDOMAIN,
    'invalid_text', 'services_portlet.save_prefs');
  if (not wwerr_api_error.is_empty) then
    wwerr_api_error_ui.show_html;
  end if;
end save_prefs;

7. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 8.3.2, "Implementing the Provider Package" for information on how to add it.
8. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in Oracle Application Server Portal User’s Guide.

8.10 Implementing Event Logging

OracleAS Portal can log events that occur during transactions with its objects. It stores these logs in the database, which makes them available through standard SQL calls and reporting tools.

You can choose the events you would like to log and organize them categorically based on user-defined domains and subdomains. For the logged events, you can view information about the event, the time the event started and stopped, the host or IP address of the remote user, the browser type, and the language.

Event logging services are available through the wwlog_api and wwlog_api_admin packages. These services include the following key features:

- Event logs are useful for tracking specific usage of the portal. To track such information, you create a log event. Log events require a name space that consists of the following:
  - **Name**, which is the event name.
  - **Domain**, which is the area of the product where the event occurred.
  - **Subdomain**, which is the subsystem that generated the event.

The default domains include the portal (WWC), application (WWV), and page group (WWS). Each domain is further classified into subdomains that define the object types. The portal domain includes the portlet, page, and document object types. The application domain includes object types such as forms, menus, reports, and charts. The page group domain includes object types such as folders, items, categories, and perspectives. Events themselves could be of types such as add, delete, personalize, hide, copy, execute, and export. If you need to define an event that does not fall within these classifications, you can define your own domain with subdomains for your events.

- Logs can track information in the following two different ways:
  - **Interval logging** calculates the elapsed time for the action performed (for example, the time taken to render a portlet).
  - **Event logging** logs the occurrence of a single step event you care about (for example, whenever a user personalizes a portlet).

- Log switching enables you to set a switch interval that defines how long you want to maintain your existing log records. The log information stored in the database uses two different tables. The log records are purged based on the value entered for the Activity Log Interval in the Configuration tab of Global Settings (accessible from the Services portlet in the Portal subtab of the Administer tab). When the log interval (in days) is reached, the logging switches between the two logging tables in the database (for example, A and B). Logs first go into A. When the log interval is reached the first time, the logs are written to B. When the log interval is reached again, the logs go back to A. A is emptied in preparation to store the new log records. If you set your log interval to 14 (the default setting), the logs will switch every 14 days, thus preserving for you, at any point in time, records dated between 14 and 28 days old.
8.10.1 Using Event Logging

In general, you can set up event logging as follows:

1. Add the event object, with an appropriate domain and subdomain combination, using \texttt{wwlog_api_admin.add_log_event}. Adding the event ensures that lists of values and other user interface components invoked when the user is monitoring the events show this new event in their lists.

2. Register the log event record by using \texttt{wwlog_api_admin.add_log_registry}. The log registry record represents the events you want to log in the future and provides a means to filter the events that need to be logged.

3. Use \texttt{start_log} and \texttt{stop_log} to mark the events you want to log in your code. Alternatively, for entering single step event log information, just call the log method to mark that event.

Guidelines for Event Logging

While implementing event logging, keep in mind the following:

- Log only what you really care about to improve performance. You don't want to flood the system with log messages that are irrelevant to you. If events are logged in Show mode, then multiple instances of these portlets mean additional hits to the database.

- Choose your domain, subdomain, and log events carefully. While using the log APIs, do not use the OracleAS Portal domains such as \texttt{WWC}, \texttt{WWV}, or \texttt{WWS} for your log messages. Organize your domains and subdomains hierarchically ensuring that they are unique across portlets. If other portlets happen to use the same domains or subdomains, you will see those log messages interspersed with your own.

- Create log events that show up in the pop-up lists of values monitoring the logs. You can simply create log registry records that filter the events that would actually be logged, either by specifying particular events or using the generic filters with wild cards (\texttt{%}). Apart from creating log registry records, we recommend that you create log events for events that you want to monitor. This way the lists of values in the user interface show these records for additional functions such as monitoring.

- Provide required privileges to users or user groups who need to monitor the logs. Any logs created by a user can be viewed by that user, the Portal Administrator, and any user with the Edit privilege on the \texttt{ANY_LOGS} object type.

8.10.2 Adding Event Logging

The services example, located in \texttt{..\pdkplsql\pdk\plsql\svcex} in PDK-PL/SQL (\texttt{pdkplsql.zip}), illustrates how you can implement event logging. You can browse through this example as follows to see how the event logging functions are implemented in a portlet:

1. Open the \texttt{services_portlet.pkb} file in an editor.

The domain and subdomain definitions for your log messages are provided with aliases in the constants part of your portlet definition.

\begin{verbatim}
DOMAIN constant varchar2(30) := 'provider';
SUBDOMAIN constant varchar2(32) := 'services';
PORTLET_PATH constant varchar2(256):= 'oracle.portal.pdk.servicesportlet';
prefname_string constant varchar2(30) := 'services_string';
prefname_title constant varchar2(30) := 'services_title';
\end{verbatim}
2. Find the `save_prefs` procedure. This procedure provides personalizable functionality where you can personalize text and the portlet title in Edit mode. `save_prefs` stores these personalizations in the database. While saving the changes, you should also log them. Hence, this procedure provides an ideal example of implementing the logging service. A single step event is logged using `wwlog_api.log`. The first instance of `wwlog_api.log` logs the event of personalizing text. The second instance logs the event of personalizing the portlet title.

```plsql
procedure save_prefs
begin

  -- Log this transaction
  l_information := l_user||'%'||l_time||'%completed%'||p_string;
  wwlog_api.log (p_domain => DOMAIN,
                  p_subdomain => SUBDOMAIN,
                  p_name      => l_user,
                  p_action    => l_action,
                  p_information => l_information,
                  p_url       => l_url,
                  p_row_count => l_row_count,
                  p_elapsed_time => l_elapsed_time);

  if (l_prefs.string_id is null or to_number(l_prefs.string_id) = 0) then
    l_action := LOG_INSERT;
  ... else -- string exists in at least one language so update it
    l_action := LOG_UPDATE;
    ... end if;
    -- Log this transaction
    l_information := l_user||'%'||l_time||'%completed%'||p_string;
    wwlog_api.log (p_domain => DOMAIN,
                   p_subdomain => SUBDOMAIN,
                   p_name      => l_user,
                   p_action    => l_action,
                   p_information => l_information,
                   p_url       => l_url,
                   p_row_count => l_row_count,
                   p_elapsed_time => l_elapsed_time);

  if (l_prefs.title_id is null or to_number(l_prefs.title_id) = 0) then
    l_action := LOG_INSERT;
  ... else
    l_action := LOG_UPDATE;
    ... end save_prefs;
```

3. The `save_prefs` procedure also logs an event with `wwlog_api.log` when an exception occurs.

```plsql
exception
  when INVALID_TEXT_EXCEPTION then
    l_information := l_user||'%'||l_time||'%'INVALID_TEXT_EXCEPTION%'||p_string;
    wwlog_api.log (p_domain => DOMAIN,
                   p_subdomain => SUBDOMAIN,
                   p_name      => l_user,
                   p_action    => l_action,
                   p_information => l_information,
                   p_url       => l_url,
                   p_row_count => l_row_count,
                   p_elapsed_time => l_elapsed_time);
```

Creating PL/SQL Portlets 8-43
Writing Multilingual Portlets

OracleAS Portal has a robust set of APIs for interacting with OracleAS Portal multilingual storage facility. This storage facility provides a mechanism for storing and retrieving strings in different languages. These APIs abstract the native multilingual functionality and provide developers with a powerful storage mechanism for developing providers that support different language environments.

Multilingual services are available through the **wwnls_api** package. These services include the following key features:

- The multilingual APIs enable the provider to load several translations for the strings displayed in their portlets. Once the strings have been loaded, the provider can call the APIs to retrieve the strings from the multilingual table as needed.
- Context APIs retrieve the user’s language and the appropriate translation for that language. The Context APIs determine the user's language environment from the language setting in the browser. When a requested translation does not exist, the APIs return the base language translation.

For example, assume that the provider's `register` procedure loads US and French translations for the portlet title. When the portlet is rendered, the provider implementation retrieves the portlet title string from the table and displays the following results:

- A request for a French string causes the portlet title to appear in French.
- A request for a US string causes the portlet title to appear in US English.
- A request for a Chinese string causes the portlet title to appear in US English because we did not load a translation for the Chinese language.

### 8.11.1 Using Multilingual Support

In general, you can set up multilingual support as follows:

1. Load your string definitions into the database using the string equivalents for each language you intend to use. For this purpose, call the `wwnls_api.add_string` or `wwnls_api.set_string` with an appropriate domain, subdomain, error message name, and error text combination.
2. Retrieve the strings you require with \texttt{wwnls_api.get_string} for the language that you desire.

### 8.11.2 Adding Multilingual Support

To add multilingual support, you need to perform the following tasks:

- Loading Language Strings
- Retrieving Language Strings

#### 8.11.2.1 Loading Language Strings

Language strings can be loaded by a script that is part of the provider installation. This script calls \texttt{add_string} and \texttt{set_string} to create equivalent strings for different languages.

OracleAS Portal uniquely identifies language strings using a combination of domain, subdomain, and name. The domain and subdomain provide a way to categorize the strings. The domain and subdomain should be unique enough to reasonably preclude conflicts with other users of the APIs.

- A domain is a particular area of the product. An example of a domain could be provider or page group.

- A subdomain is a subsystem of the domain. For example, the subdomain could be the provider name (for example, HelloProvider) or subpage name (for example, HelloPage).

The services example, located in ..\pdkplsql\pdk\plsql\svcex in PDK-PL/SQL (pdkplsql.zip), illustrates how you can implement multilingual support. You can browse through this example as follows to see how to load strings for multilingual support:

1. Open the \texttt{services_seed.sql} file in an editor.

2. Notice the \texttt{add_string} call with the parameters for domain name, subdomain name, string name, language, and the actual string text. It returns the String ID for the language string. For setting equivalent strings in other languages, \texttt{set_string} is called with the same parameters.

```sql
set serveroutput on size 1000000
set define off
declare
    l_string_id  integer;
    l_person_id  integer;
    l_group_id   integer;
begin
    ...-- strings for portlet record fields
    l_string_id := wwnls_api.add_string(
        'provider','services','ptldefname','us','DatabaseServicesPortlet');
    wwnls_api.set_string(
        'provider','services','ptldeftitle','d','DatenbankServicesPortlet-d');
    l_string_id := wwnls_api.add_string(
        'provider','services','ptldefdesc','us','Database Services Portlet');
    wwnls_api.set_string(
        'provider','services','ptldefdesc','d','Datenbank Services Portlet - d');
    l_string_id := wwnls_api.add_string(
        'provider','services','ptldefdesc','us','This is the database services portlet implemented in PL/SQL. It displays 6 show modes.');
```
8.11.2.2 Retrieving Language Strings

The services example, located in ..\pdkplsql\pdk\plsql\svcex in PDK-PL/SQL (pdkplsql.zip), illustrates how you can implement multilingual support. You can browse through this example as follows to see how to retrieve strings for multilingual support:

1. Open the services_portlet.pkb file in an editor.

2. The domain and subdomain definitions for your language strings are provided with aliases in the constants part of your portlet definition.

   ```
   DOMAIN     constant varchar2(30) := 'provider';
   SUBDOMAIN  constant varchar2(32) := 'services';
   PORTLET_PATH constant varchar2(256):= 'oracle.portal.pdk.servicesportlet';
   PREFNAME_STRING constant varchar2(30) := 'services_string';
   PREFNAME_TITLE constant varchar2(30) := 'services_title';
   ```

3. Find the get_portlet_info function. Notice the calls to wwnls_api.get_string to populate the portlet title, name, and description.

   ```
   function get_portlet_info
   (   p_provider_id in integer,
      p_language in varchar2)
   return wwpro_api_provider.portlet_record
   is
      l_portlet wwpro_api_provider.portlet_record;
      begin
         l_portlet.id := services_provider.SERVICES_PORTLET_ID;
         l_portlet.provider_id := p_provider_id;
         l_portlet.language := p_language;
         l_portlet.title :=
            wwnls_api.get_string(
               p_domain       =>  DOMAIN,
               p_sub_domain   =>  SUBDOMAIN,
               p_name         =>  'ptldeftitle',
               p_language     =>  p_language);
         l_portlet.description :=
            wwnls_api.get_string(
               p_domain       =>  DOMAIN,
               p_sub_domain   =>  SUBDOMAIN,
               p_name         =>  'ptldefdesc',
               p_language     =>  p_language);
         l_portlet.name :=
            wwnls_api.get_string(
               p_domain       =>  DOMAIN,
               p_sub_domain   =>  SUBDOMAIN,
               p_name         =>  'ptldevtmmmsg',
               p_language     =>  'us',
               'Web Services Portlet Timed Out.');
      end;
   ```
8.12 Enhancing Portlets for Mobile Devices

This section explains how to go about enhancing a portlet with PDK-PL/SQL for a mobile device. Before proceeding with this section, you should familiarize yourself with the guidelines for building mobile-enabled portlets, Section 8.1.3, "Guidelines for Mobile Portlets", and the methods of building portlets with PDK-PL/SQL, Section 8.2, "Building PL/SQL Portlets with the PL/SQL Generator" and Section 8.3, "Building PL/SQL Portlets Manually".

To properly build a portlet for a mobile device, do the following:

1. Set the portlet record attributes to support mobile output. Requests arriving from mobile devices through the mobile gateway need to be answered with OracleAS Wireless XML using the `text/vnd.oracle.mobilexml` content type in the header. A mobile-enabled portlet must specify that it can handle these types of requests and produce the mobile content accordingly.

   Table 8–6 lists the portlet record attributes pertinent to mobile portlets and explains how you should specify them.

### Table 8–6 Portlet Record Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>accept_content_type</td>
<td>Specify a comma-delimited list of the content types that the portlet produces. If the portlet can produce both HTML and OracleAS Wireless XML, the string would be:</td>
</tr>
<tr>
<td></td>
<td><code>text/html, text/vnd.oracle.mobilexml</code></td>
</tr>
<tr>
<td></td>
<td>If the portlet can produce only OracleAS Wireless XML, the string would be: <code>text/vnd.oracle.mobilexml</code></td>
</tr>
<tr>
<td>mobile_only</td>
<td>Indicate whether the portlet is available only to mobile pages. TRUE declares the portlet as mobile only, and it will not appear in the Add Portlets page for a standard page. FALSE declares the portlet as mobile and standard, and it will appear in the Add Portlets page for both standard and mobile pages. If the portlet only produces OracleAS Wireless XML and you set this flag to FALSE, the OracleAS Wireless XML is automatically transformed into HTML for desktop devices (such as a normal PC Web browser). This functionality enables you to develop portlets that output only OracleAS Wireless XML but can be viewed on standard pages for desktop access as well as for mobile access.</td>
</tr>
</tbody>
</table>

4. Browse the rest of the file to examine other usage examples of `wwnls_api.get_string`, which is used in several other places in `services_portlet.pkb`.

5. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 8.3.2, "Implementing the Provider Package" for information on how to add it.

6. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in Oracle Application Server Portal User’s Guide.
Table 8–6 (Cont.) Portlet Record Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>short_title</td>
<td>Enter a shorter version of the portlet title. This title is used on mobile devices because it is more likely to fit the smaller screen. If you do not set a short title, then the portlet title is used instead.</td>
</tr>
<tr>
<td>has_show_link_mode</td>
<td>Indicate whether you have implemented Link mode for the portlet. We recommend that all mobile portlets enable Link mode. The rationale for using Link mode is more fully explained when you reach step 3.</td>
</tr>
</tbody>
</table>

2. If the portlet can produce both HTML and OracleAS Wireless XML, then, during execution, it must determine whether the request is for a mobile or a desktop device. If it is a desktop request, then the portlet must produce HTML output. If it is a mobile request, then it must produce OracleAS Wireless XML output. You can determine the request type with the `wwctx_api.get_http_accept()` function. It fetches the HTTP accept header, which indicates the request type. In the portlet response, you must set the MIME header to the appropriate value in the HTTP header before the portlet content is produced. If not, then the portlet response is rejected when the resulting page is built by OracleAS Portal. If the call to `wwctx_api.get_http_accept()` returns a string starting with `text/vnd.oracle.mobilexml`, then you can assume it is a mobile request. Otherwise, it is a desktop request. In the case of a mobile request, you should set the MIME header to `text/vnd.oracle.mobile.xml`. In the case of a desktop request, you can explicitly set the MIME header to `text/html`, but it is not required that you do so because it's the default setting.

If you want to produce HTML for desktop requests and OracleAS Wireless XML for mobile requests, the `show` procedure for your portlet should look similar to the following:

```plaintext
procedure show
{
    p_portlet_record in out wwpro_api_provider.portlet_runtime_record
}
begin
-- Does the portal want us to render the portlet contents?
--
if (p_portlet_record.exec_mode = wwpro_api_provider.MODE_SHOW) then
--
-- Is this a mobile request?
--
if owa_util.get_cgi_env('HTTP_ACCEPT') like
    wwpro_login.CONTENT_TYPE_MOBILEXML || '%' then
--
-- This is a mobile request for the portlet contents
-- call the mobile show contents procedure
--
    render_mobile_show_contents(p_portlet_record);
else
--
-- This is a desktop request for the portlet contents,
-- call the desktop show contents procedure
--
    render_desktop_show_contents(p_portlet_record);
end if;
```
elsif -- check for other show modes, and handle them
...
end show;

If you want to produce only OracleAS Wireless XML for all requests, the show procedure for your portlet should look similar to the following:

procedure show
  (p_portlet_record in out wwpro_api_provider.portlet_runtime_record)
is
begin
  -- Does the portal want us to render the portlet contents?
  --
  if (p_portlet_record.exec_mode = wwpro_api_provider.MODE_SHOW) then
  --
  -- This is either a desktop or mobile request for the portlet
  -- contents call the show contents procedure to render the
  -- OracleAS Wireless XML output
  --
  render_show_contents(p_portlet_record);
  elsif -- check for other modes
  ...
end show;

In order to generate OracleAS Wireless XML, the render_show_contents or render_show_mobile_contents procedure would need to contain something similar to the following:

... (p_portlet_record in out wwpro_api_provider.portlet_runtime_record)
is
begin
  -- Set the MIME type to be Oracle9iAS Wireless XML
  --
  owa_util.mime_header('text/vnd.oracle.mobilexml');
  --
  -- Output the Oracle9iAS Wireless XML markup
  --
  htp.p('<SimpleText><SimpleTextItem>');
  htp.p('Hello World!');
  htp.p('</SimpleTextItem></SimpleText>');
...

---

Note: As this is a mobile request the MIME header is set to text/vnd.oracle.mobilexml. In the OracleAS Wireless XML markup, you only need to render the actual content. OracleAS Portal fits it into the complete OracleAS Wireless XML document. You do not need <SimpleResult> or <SimpleContainer> tags. These tags are rendered by OracleAS Portal along with a back link, which, by default, is assigned to one of the buttons on the mobile device.

---
3. If you want your portlet to allow personalization of titles for mobile rendering, you need to implement Link mode. Link mode is only called for mobile requests and it renders a link to the portlet content. This link appears in the menu of page contents when the user first navigates to the page. If a Link mode is not present for a portlet, then OracleAS Portal renders a default link mode using the short title from the portlet record. You can also use Link mode to render more than just a title. For example, in a stock portlet, you could render the stock price of a user's favorite stock. Thus, they could see the current stock price without drilling down further.

The link for Link mode rendition is provided by OracleAS Portal and passed to the portlet through the page_url parameter on the portlet record. The portlet can extend or completely replace this behavior. If this link is rewritten as a URL that takes the user away from OracleAS Portal (it does not point to the OracleAS Portal middle tier server), then the portlet should set the show Behaviour_style field to wwpro_api_provider.EXTERNAL_PORTLET on the portlet record.

The following example illustrates how to code your show procedure to include Link mode:

```cpp
procedure show
    (p_portlet_record in out wwpro_api_provider.portlet_runtime_record
is
begin
  -- Does the portal want us to render the portlet contents?
  -- if (p_portlet_record.exec_mode = wwpro_api_provider.MODE_SHOW) then
    -- Is this a mobile request?
    -- if owa_util.get_cgi_env('HTTP_ACCEPT') like
      wwpro_login.CONTENT_TYPE_MOBILEXML || '% then
        -- This is a mobile request for the portlet contents,
        -- call the mobile show contents procedure
        -- render_mobile_show_contents(p_portlet_record);
      else
        -- This is a desktop request for the portlet contents,
        -- call the desktop show contents procedure
        -- render_desktop_show_contents(p_portlet_record);
      end if;
  -- Does the portal want us to render the LINK mode?
  elsif (p_portlet_record.exec_mode = wwpro_api_provider.MODE_LINK) then
    -- This is a mobile request for the portlet link mode,
    -- call the mobile link procedure
    -- render_mobile_link(p_portlet_record);
  elsif -- check for other show modes, and handle them ...
  end show;
```
The following example illustrates what you must implement in the `render_mobile_link` procedure for Link mode. In particular, notice the setting of the MIME type, the usage of `get_custom_short_title`, and the OracleAS Wireless XML output.

```plsql
procedure render_mobile_link
  (p_portlet_record in out wwpro_api_provider.portlet_runtime_record)
is
  l_short_title varchar2(80);
begin
  -- Set the MIME type to be Oracle9iAS Wireless XML
  owa_util.mime_header('text/vnd.oracle.mobilexml');
  -- Get the personalized short title for this portlet instance
  -- using the language and reference path held in the portlet
  -- record
  l_short_title := get_custom_short_title(p_portlet_record);
  -- Output the OracleAS Wireless XML markup
  if l_short_title is not null then
    htp.p('<SimpleHref target="'
             || htf.escape_sc(p_portlet_record.page_url)
             || '">'
             || l_short_title
             || '</SimpleHref>');
  else
    htp.p('<SimpleHref target="'
             || htf.escape_sc(p_portlet_record.page_url)
             || '">'
             || get_custom_title(p_portlet_record)
             || '</SimpleHref>');
  end if;
end render_mobile_link;
```

**Accessing the DeviceClass Header**

To further facilitate the creation of mobile-enabled portlets, you can access information in the `DeviceClass` header, which is sent to all portlets through `owa_util.get_cgi_env('x-oracle-device.class')`. You can use the values of this header to determine the complexity of the OracleAS Wireless XML rendition of the portlet. For example, the PDA rendition could contain considerably richer content than the microbrowser rendition. The voice rendition could contain extra tags for voice commands and links to sound files, which play instead of the text-to-speech system.

Table 8–7 describes the values available in the header.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>voice</td>
<td>Indicates a voice-only device, such as a normal telephone calling a voice access number.</td>
</tr>
<tr>
<td>microbrowser</td>
<td>Indicates a small size display device, which supports a markup browser, such as a WAP phone.</td>
</tr>
</tbody>
</table>
8.13 Registering Providers Programmatically

In most cases, you use the OracleAS Portal user interface to register providers as described in Section 8.2.3.2, "Registering the Database Provider". In some instances, though, you may wish to register a provider programmatically rather than through the user interface. This section describes how to use `wwpro_api_provider_registry.register_provider` to register your providers:

- **Registration Prerequisites**
- **Provider Record Input**
- **Registration Example**

### 8.13.1 Registration Prerequisites

In order to register a provider programmatically with `wwpro_api_provider_registry.register_provider`, the following requirements must be met:

- You must first install the provider as described in Section 8.2.3.1, "Installing the Packages in the Database".
- You must have the necessary privileges to execute `wwpro_api_provider_registry.register_provider`. As it is a secure API, a security check determines whether the user calling it has the necessary privileges. At a minimum, you must have `wwsec_api.ANYPROVIDER_PUBLISH`. If you have the privileges to execute the API, you also have sufficient privileges to set the OracleAS Portal session through `wwctx_api.set_context`.

#### Note:
If the database user of the SQL*Plus session is the OracleAS Portal schema owner, then, by default, the user is the OracleAS Portal schema owner (unless `wwctx_api.set_context` is called with a different user). The schema owner already has the necessary privileges to execute `wwpro_api_provider_registry.register_provider`.

### 8.13.2 Provider Record Input

The `wwpro_api_provider_registry.register_provider` API requires the `provider_record` as input. When you pass the `provider_record` to `register_provider`, all of the fields are not required. Table 8-8 indicates which fields are required for Web and database providers. If a field is not applicable for a particular type of provider, it is shown as NA.

#### Table 8-8 Required provider_record Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Required for database providers</th>
<th>Required for Web providers</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Yes</td>
<td>Yes</td>
<td>None</td>
</tr>
</tbody>
</table>
Creating PL/SQL Portlets 8-53

8.13.3 Registration Example

The following sample SQL script illustrates the usage of `wwpro_api_provider_registry.register_provider`.

```
set serveroutput on size 1000000
set define on
declare
  l_prov_rec     wwpro_api_provider_registry.provider_record;
  l_prov_id      integer;
begin
  l_prov_rec.name                 := 'CacheProvider';
  l_prov_rec.display_name         := 'Cache Provider';
  l_prov_rec.timeout              := 10;
  l_prov_rec.timeout_msg          := 'The provider timed out';
  l_prov_rec.implementation_style := wwpro_api_provider_registry.DATABASE_IMPL;
  l_prov_rec.implementation_owner := '&&2';
  l_prov_rec.implementation_name  := 'cache_provider';
  l_prov_rec.language             := wwctx_api.get_nls_language;
  l_prov_rec.enable_distribution  := true;
  l_prov_rec.login_frequency      := wwpro_api_provider_registry.LOGIN_FREQUEN
```

Table 8–8 (Cont.) Required provider_record Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Required for database providers</th>
<th>Required for Web providers</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>implementation_style</td>
<td>Yes</td>
<td>Yes</td>
<td>This field is always the following for PL/SQL portlets: <code>wwpro_api_provider_registry.DATABASE_IMPL</code></td>
</tr>
<tr>
<td>implementation_owner</td>
<td>NA</td>
<td>Yes</td>
<td>This field is the schema where the provider/portlet is located.</td>
</tr>
<tr>
<td>implementation_name</td>
<td>NA</td>
<td>Yes</td>
<td>When registering a database provider the implementation package of the provider must be valid for the registration to be successful.</td>
</tr>
<tr>
<td>login_frequency</td>
<td>Yes</td>
<td>Yes</td>
<td>This field specifies how often to grab the session information.</td>
</tr>
<tr>
<td>http_app_type</td>
<td>Yes</td>
<td>NA</td>
<td>For external application Web providers: <code>wwpro_api_provider_registry.HTTP_APP_TYPE_EXTERNAL</code> For regular Web providers: <code>wwpro_api_provider_registry.HTTP_APP_TYPE_PORTAL</code></td>
</tr>
<tr>
<td>http_url</td>
<td>Yes</td>
<td>NA</td>
<td>None</td>
</tr>
<tr>
<td>require_url</td>
<td>Yes</td>
<td>NA</td>
<td>None</td>
</tr>
<tr>
<td>encryption_key</td>
<td>If provider_key is not null, yes.</td>
<td>If provider_key is not null, yes.</td>
<td>If provider_key is null, no.</td>
</tr>
<tr>
<td></td>
<td>If provider_key is null, no.</td>
<td>If provider_key is null, no.</td>
<td>None</td>
</tr>
</tbody>
</table>

8.13.3 Registration Example

The following sample SQL script illustrates the usage of `wwpro_api_provider_registry.register_provider`.

```
set serveroutput on size 1000000
set define on
declare
  l_prov_rec     wwpro_api_provider_registry.provider_record;
  l_prov_id      integer;
begin
  l_prov_rec.name                 := 'CacheProvider';
  l_prov_rec.display_name         := 'Cache Provider';
  l_prov_rec.timeout              := 10;
  l_prov_rec.timeout_msg          := 'The provider timed out';
  l_prov_rec.implementation_style := wwpro_api_provider_registry.DATABASE_IMPL;
  l_prov_rec.implementation_owner := '&&2';
  l_prov_rec.implementation_name  := 'cache_provider';
  l_prov_rec.language             := wwctx_api.get_nls_language;
  l_prov_rec.enable_distribution  := true;
  l_prov_rec.login_frequency      := wwpro_api_provider_registry.LOGIN_ FREQUENCY_NEVER;
```

Creating PL/SQL Portlets 8-53

8.13.3 Registration Example

The following sample SQL script illustrates the usage of `wwpro_api_provider_registry.register_provider`.

```
set serveroutput on size 1000000
set define on
```
l_prov_rec.created_on := sysdate;
l_prov_rec.created_by := '&&1';
l_prov_rec.last_updated_on := sysdate;
l_prov_rec.last_updated_by := '&&1';

l_prov_id := wwpro_api_provider_registry.register_provider(l_prov_rec);
commit;
dbms_output.put_line('Cache Provider successfully registered');

exception
  when others then
    dbms_output.put_line('ERROR: Could not register Cache Provider');
    dbms_output.put_line('SQLERRM: ' || SQLERRM);
    rollback;
end;
/

**Note:** After you have registered your provider, you may also refresh the Portlet Repository programmatically using wwpro_api_provider_registry.refresh_portlet_repository or through the OracleAS Portal user interface.
Part III contains the following chapters:

- Chapter 9, "Content Management API Introduction"
- Chapter 10, "Getting Started with Content Management APIs"
- Chapter 11, "Performing Simple Content Management Tasks"
- Chapter 12, "Extending Your Portal"
- Chapter 13, "Searching Portal Content"
- Chapter 14, "Creating Multi-Lingual Content"
- Chapter 15, "Implementing Content Security"
- Chapter 16, "Using the Content Management Event Framework"
This chapter provides an overview of a selection of the APIs provided with Oracle Application Server Portal. You can use these particular APIs to write code for performing content management tasks. It contains the following sections:

- Overview
- Content Management APIs
- Providing Access to the APIs and Secure Views
- Guidelines for Using the APIs
- Guidelines for Using the Secure Views
- Code Samples

9.1 Overview

OracleAS Portal uses a schema within the Oracle Application Server Metadata Repository (Figure 9-1) to store the content and metadata associated with the portal instance. This schema is sometimes referred to as the content repository. For example, when a contributor adds a file item to a portal page, the file is uploaded to a table in the portal schema of the OracleAS Metadata Repository along with the metadata supplied.
For more information about the OracleAS Metadata Repository and OracleAS Portal architecture in general, refer to the *Oracle Application Server Portal Configuration Guide*. OracleAS Portal provides many built-in tools to help you manage the content in the portal schema of the OracleAS Metadata Repository right from your Web browser. However, sometimes you may find that you need to work with your content in an environment outside of the OracleAS Portal browser-based user interface. The following are example scenarios of when you might want to do this:

- Building an alternative user interface when the product user interface does not quite meet your requirements, for instance:
  - to change the look and feel of the wizards to meet your own corporate design
  - to add custom validations
  - to provide a custom search form or search results page

- Bulk loading content with metadata
- Integrating with external applications and content management systems
- Archiving items (for example, moving them to an archive page or offline storage)
- Managing versions (for example, deleting or purging noncurrent versions, limiting the number of versions, and so on)
- Integrating with external workflow systems

The content management APIs enable you to interact with the portal schema of the OracleAS Metadata Repository programmatically, rather than by using the OracleAS Portal user interface. For more information, refer to Section 9.2, "Content Management APIs".

You can also query the data in the content repository using a set of secure views. For more information, refer to Section 9.2.1, "Secure Content Repository Views".
9.2 Content Management APIs

There are multiple public APIs that enable you to programmatically perform many content management tasks, such as adding items and creating pages.

The majority of the APIs for content management are contained within the WWSBR_API package. You may also find the following API packages useful when writing code to perform content management tasks:

- The WWSRC_API package contains APIs for performing searches on content in the portal schema of the OracleAS Metadata Repository.
- The WWSEC_API package contains APIs for controlling access to content in the portal schema of the OracleAS Metadata Repository.
- The WWCTX_API package contains APIs for managing a session context for a specific user.
- The WWPRO_API_INVALIDATION package contains APIs for invalidating content in Oracle Application Server Web Cache.

Note: Only use public APIs in your code. The use of non-public APIs is not supported and may cause your code to break when upgrading to new releases.

For more information about these packages, refer to Section F.1, "Supported APIs". For a full list of the supported PL/SQL APIs, refer to the Oracle Application Server Portal PL/SQL API Reference on Portal Center:

http://portalcenter.oracle.com

In the Portal Focus Areas section, click Portlet Development, then in the APIs and References section, click PL/SQL API Reference.

9.2.1 Secure Content Repository Views

Held within the portal schema are a number of content repository views. You can use these views to query back data on the documents and items stored in the content repository. For a list of the secure views, refer to Section F.2, "Secure Views".

Many of the content management APIs that are discussed in this manual require you to pass object IDs as parameters. To do this, you need to know the IDs of the objects with which you want to work. You can use the secure content repository views to find the IDs of portal objects. For more information, refer to Section 10.3, "Finding an Object ID".

Note: Direct access to other tables and views in the portal schema of the OracleAS Metadata Repository is not supported, as the definition of those tables and views may change between releases.

9.2.2 Terminology

To maintain backward compatibility with the Oracle9iAS Portal Release 1 (3.0.9) content area APIs and views, many of the API procedure and parameter names and the view and column names continue to use Release 1 terminology. You may find Table 9–1 useful to map the Release 1 terminology to the current terminology.
9.3 Providing Access to the APIs and Secure Views

The portal schema automatically has the appropriate access to the public APIs and secure views. To enable another schema to access the public APIs and secure views, use the following script:

MID_TIER_ORACLE_HOME/portal/admin/plsql/wwc/provsyns.sql

To provide access to the APIs:

1. Change to the directory containing the provsyns.sql script:
   
   Windows: cd <MID_TIER_ORACLE_HOME>/portal/admin/plsql/wwc
   Linux/Unix: cd <MID_TIER_ORACLE_HOME>/portal/admin/plsql/wwc

2. Log in to SQL*Plus as the portal schema owner. For example:

   sqlplus portal/oracle1

   You must run provsyns.sql as the portal schema owner. By default the portal schema is called PORTAL. An administrator can use Oracle Internet Directory to obtain the portal schema password as follows:

   a. Navigate to:

      – Entry Management
      – cn=OracleContext
      – cn=Products
      – cn=IAS
      – cn=Infrastructure Databases
      – OrclReferenceName=Infrastructure Database (for example, iasdb.server.domain.com)
      – OrclResourceName=Schema Name (for example, PORTAL)

   b. Click this entry.

   c. Look for the orclpasswordattribute value in the right panel. This is the schema password.

3. Run the provsyns.sql script:

   SQL>@provsyns.sql <schema>

   Where schema is the name of the schema to which you want to grant access.

   Tip: Ignore any messages related to a missing file when running provsyns.sql.
9.4 Guidelines for Using the APIs

When using the APIs described in this manual, you should follow the best practice guidelines described in the following sections.

9.4.1 Using a Separate Schema

When creating procedures and packages that use the content management APIs, create them in a separate schema of the database in which OracleAS Portal is installed. Do not create them in the portal schema. Creating additional procedures and packages in the portal schema is not supported and they may be lost when upgrading to a new release.

Once you have created the schema for your procedures and packages, you must grant it access to the APIs and secure content repository views. For information about how to do this, refer to Section 9.3, "Providing Access to the APIs and Secure Views".

9.4.2 Using Constants

All the API packages include predefined constants that you can use to easily refer to OracleAS Portal objects and metadata. For example, the WWSBR_API package includes constants for base attributes (such as ATTRIBUTE_AUTHOR and ATTRIBUTE_TEXT), seeded item types (such as ITEM_TYPE_FILE and ITEM_TYPE_URL), and image alignment options (such as ALIGN_BOTTOM and ALIGN_RIGHT). To make your code robust, you should use these constants wherever possible for object IDs and attribute values. That way, if the actual ID or values change, your code will still work because the name of the constant will stay the same.

For a full list of the constants available in each API package, refer to the Oracle Application Server Portal PL/SQL API Reference on Portal Center:
http://portalcenter.oracle.com

In the Portal Focus Areas section, click Portlet Development, then in the APIs and References section, click PL/SQL API Reference.

9.4.3 Invalidating the Cache

Many of the APIs automatically generate cache invalidation messages for the pages that are affected by the changes. Therefore, you should always call `wwpro_api_invalidation.execute_cache_invalidation` at the end of your routine to process these messages (see Example 11–3 for an example of how to do this). If you do not call this procedure, your changes may not be visible until the affected pages are invalidated through other means.

There is no need to call `execute_cache_invalidation` more than once. For example, if you are adding or updating multiple items in a loop, call `execute_cache_invalidation` when the loop is complete.

9.4.4 Issuing Commits

The APIs do not issue commits. However, if you are calling the APIs from a browser session through Portal Services, Portal Services will perform an automatic commit before returning control to the browser.

The `wwpro_api_invalidation.execute_cache_invalidation` API also issues a commit. If you are calling the APIs from an external environment (for example, SQL*Plus or a Web provider), and you need to commit before processing the cache
invalidations, you must explicitly issue a commit statement in your code. Similarly, if you need to rollback, do so before calling execute_cache_invalidation.

9.4.5 Resetting CMEF Global Variables

If you are using the content management APIs in conjunction with the Content Management Event Framework (CMEF) you need to make sure that you reset the CMEF global variables before or after each API call.

When a user performs an action using the user interface, OracleAS Portal uses various global variables to determine which CMEF messages get logged. After each action, these global variables are automatically reset ready for the next action. When you use the APIs to perform portal actions, these global variables are not automatically reset. Therefore, to ensure that CMEF messages are logged correctly, you must reset the CMEF variables explicitly by calling the wwsbr_api.clear_cmef_context API before or after each API call.

For example, if you use the APIs to create several items on a page, you need to call the clear_cmef_context API between each of the add_item calls (see Example 9–1).

Example 9–1 Calling the clear_cmef_context API

```sql
... l_new_item_master_id1 := wwsbr_api.add_item(
    p_caid              => l_caid,
    p_folder_id         => l_folder_id,
    ...
); wwsbr_api.clear_cmef_context;

l_new_item_master_id2 := wwsbr_api.add_item(
    p_caid              => l_caid,
    p_folder_id         => l_folder_id,
    ...
); wwsbr_api.clear_cmef_context;
...
```

For more information about CMEF, refer to Chapter 16, "Using the Content Management Event Framework".

9.4.6 Using Predefined Exceptions

The API packages contain many predefined exceptions. When coding with the content management APIs, it is good practice to include the appropriate predefined exceptions rather than relying on the WHEN OTHERS exception to pick up all errors. This also has the advantage that any error messages generated by the code can be more specific to the actual problem. Example 9–2 shows some of the exceptions you might include when calling the wwsbr_api.set_attribute API.

Example 9–2 Using Predefined Exceptions

```sql
begin
    wwsbr_api.set_attribute(
        p_site_id           => 37,
        p_thing_id          => 8056,
        p_attribute_site_id => wwsbr_api.SHARED_OBJECTS,
        p_attribute_id      => wwsbr_apiATTRIBUTE_TITLE,
        p_attribute_value   => 'New Display Name'
    );
```

For more information about CMEF, refer to Chapter 16, "Using the Content Management Event Framework".
-- Process cache invalidation messages.
wwpro_api_invalidation.execute_cache_invalidation;
exception
  when wwsbr_api.ITEM_NOT_FOUND_ERROR then
    dbms_output.put_line('Item does not exist');
  when wwsbr_api.ATTRIBUTE_NOT_FOUND then
    dbms_output.put_line('Attribute does not exist');
  when wwsbr_api.ITEM_NOT_FOR_UPDATE then
    dbms_output.put_line('Cannot update an item with a status of Rejected, Deleted or Marked for Delete');
  when wwsbr_api.NOT_AUTHORIZED_USER then
    dbms_output.put_line('User trying to update the item is not the current user or the user who checked the item out');
  when wwsbr_api.EDIT_CUSTOM_ATTR then
    dbms_output.put_line('Error while trying to update a custom attribute');
  when OTHERS then
    dbms_output.put_line('Error '||to_char(sqlcode)||': '||sqlerrm);
end;
/

Tip: When using predefined exceptions, remember to include the name of the package that owns it, for example wwsbr_api.PAGE_NOT_FOUND.

For a full list of the exceptions available in each API package, refer to the Oracle Application Server Portal PL/SQL API Reference on Portal Center:
http://portalcenter.oracle.com

In the Portal Focus Areas section, click Portlet Development, then in the APIs and References section, click PL/SQL API Reference.

9.4.7 Naming Objects

When creating page groups and page group objects, you specify a unique internal name for the object (p_name). Internal names must:

- be no more than 60 characters in length.
- not contain spaces or special characters other than the underscore character (_).

9.5 Guidelines for Using the Secure Views

When using the secure views described in this manual, you should follow the best practice guidelines described in the following sections.

Note: In this release, there is no supported view of the document table in the repository. A secure view of the document table is planned for a future release.

9.5.1 Identifying Primary Keys

The primary key for most objects contains the object id, the page group id (caid), and the language. When joining between views, always use these columns in the JOIN clause. Example 9–3 shows the JOIN when joining an item to its page.
Example 9–3  Joining an Item to Its Page

```sql
select ... 
from wwsbr_all_items i, 
     wwsbr_all_folders p 
where i.folder_id = p.id 
  and i.caid = p.caid 
  and i.language = p.language
```

9.5.2 Querying Translatable Objects

If an object is translatable (that is, it resides in a page group for which translations are enabled and it includes languages in its key), you must observe the following rules:

- If the object (or its current version) is translated, a row will exist for the translation. To select the row for the current session language, compare the value of the language column to the function `wwctx_api.get_nls_language()`.
- If the object is not translated, select the row for the page group’s default language.

Example 9–4 selects the translated page display name for all pages in a given page group.

Example 9–4  Selecting a Translated Page Display Name

```sql
select p.display_name title 
from wwsbr_all_folders p 
where p.caid = 53 
and (p.language = wwctx_api.get_nls_language -- The current language. 
  or (exists -- A row for the page in the page group default language. 
    (select pg.id 
     from wwsbr_all_content_areas pg 
     where pg.id = p.caid 
       and pg.default_language = p.language 
    ) 
    and not exists -- A row for the page in the current language. 
    (select p2.id 
     from wwsbr_all_folders p2 
     where p2.id = p.id 
       and p2.language = wwctx_api.get_nls_language 
    ) 
  )
)

Tip:  When writing PL/SQL routines that refer to the current session language, call `wwctx_api.get_nls_language` once and store it in a variable. Referring to the variable in any following SQL statements, rather than calling the API multiple times, results in better performing code.

9.5.3 Selecting Data for the Current User

To select data for the current user, use the function `wwctx_api.get_user`. Example 9–5 selects items created by the current user.

Example 9–5  Selecting Items Created by the Current User

```sql
declare 
  l_user varchar2(60);
```
... 
begin 
... 
l_user := wwctx_api.get_user;
select ...
from wwsbr_all_items
where creator = l_user
...
end;
/

9.6 Code Samples

The code samples provided in the next few chapters are intended to provide some examples of how you can use the content management and associated APIs. Not all of the APIs are used in these samples, and the samples provided do not utilize all the parameters, constants, exceptions, and so on available to the APIs.

For a full list of the public APIs available with OracleAS Portal and their parameters, constants, exceptions and so on, refer to the Oracle Application Server Portal PL/SQL API Reference on Portal Center:

http://portalcenter.oracle.com

In the Portal Focus Areas section, click Portlet Development, then in the APIs and References section, click PL/SQL API Reference.

The examples are written to be run from SQL*Plus. If running from other environments, for example Portal Services, you may need to make changes to the code given the context in which it is executing.

The examples, with the exception of Chapter 14, "Creating Multi-Lingual Content", which deals with translations, assume that the language context is set to the default language of the page group being manipulated. This simplifies the WHERE clause on any SQL statements. Objects always have a translation in the default language therefore the queries will return a single row as appropriate.

If this assumption on the language code does not hold true for your code, refer back to Section 9.5.2, "Querying Translatable Objects" for an example of how to extend the WHERE clause to deal with multiple translations or the non-existence of a translation in the given language context.
Chapter 10

Getting Started with Content Management APIs

This chapter introduces some tasks and concepts that you may find useful before you start using the content management APIs provided with OracleAS Portal. It contains the following sections:

- Setting the Session Context
- API Parameters
- Finding an Object ID

10.1 Setting the Session Context

All of the content management APIs and the secure views assume that the context has been set for the functions to execute as a specific portal user. If you call an API from within a browser session (for example, a database provider or PL/SQL item), the context is set automatically to the user who is currently logged in to the portal. If you call an API from a Web provider or an external application (for example, SQL*Plus or Portal Services), you must set the context to a specific user by using the `wwctx_api.set_context` API.

This API allows an application to assert an identity, by providing a user name and password that the portal can verify with the Oracle Application Server Single Sign-On Server before establishing a session for the asserted user. If the assertion fails, due to an invalid password, for example, an exception is raised.

Example 10–1 sets the context to user JOE.BLOGGS with the password welcome1.

```sql
Example 10–1 Setting the Session Context (set_context API)

declare
    p_user_name varchar2(60) := 'JOE.BLOGGS';
    p_password  varchar2(60) := 'welcome1';
    p_company   varchar2(60) := null;
begin
    wwctx_api.set_context (p_user_name, p_password, p_company);
end;
/
```

10.2 API Parameters

- The following parameters all refer to the page group ID:
Finding an Object ID

- `p_site_id`
- `p_siteid`
- `p_caid`

For backward compatibility purposes these parameter names have not been changed, and will not be changed, to use the most recent terminology. For more information, refer to Section 9.2.2, "Terminology".

- The following parameters both refer to the page ID:
  - `p_folder_id`
  - `p_id`

For backward compatibility purposes these parameter names have not been changed, and will not be changed, to use the most recent terminology. For more information, refer to Section 9.2.2, "Terminology".

- An item has two IDs. All versions of the item have the same master item ID (GUID). However, when you create a new version of an item it is assigned its own ID that uniquely identifies that particular version of the item. Some APIs simply need the master item ID to identify an item (`p_master_item_id`). However, some APIs (for example, `modify_item`) also need to know which particular version of the item you want to work with. In this case you need to pass the unique item ID (`p_item_id`). In some APIs, you will find that the item ID is referred to as `p_thing_id`.

- The primary key for most objects contains the ID of the actual object and the ID of the page group to which the object belongs. When referencing an existing object in a parameter, you must reference both the object ID and its page group ID. The referenced object's page group must either be the same as the page group of the referencing object or the Shared Objects page group. For example, when adding an item, you need to specify the item type (for example, file, text, image, and so on). To do this you must pass the item type ID (`p_type_id`) and the ID of the page group which contains the item type (`p_type_caid`).

10.3 Finding an Object ID

Many of the content management APIs require you to pass object IDs to their parameters. To do this, you need to know the IDs of the objects with which you want to work. You can obtain these IDs by using the secured content repository views.

10.3.1 Finding a Page Group ID

Many of the content management APIs include the ID of a page group as one of the parameters. For example, when creating a page using the `add_folder` API, you need to specify the ID of the page group in which you want to create the page using the `p_caid` parameter. Example 10–2 shows a query that you could use to find out this ID.

**Example 10–2 Finding the ID of a page group**

```sql
select id "Page Group ID"
from wwsbr_all_content_areas
where name = '&pagegroup'
  and language = '&language'
/
```

Example 10–3 shows how you might use this query in a function.
Example 10–3  Function to Find the ID of a Page Group

create or replace function get_page_group_id (p_name in varchar2) return number is
    l_page_group number;
    l_language varchar2(30);
begin
    l_language := wwctx_api.get_nls_language;
    select id
    into l_page_group
    from wwsbr_all_content_areas
    where name = p_name
    and language = l_language;
    return l_page_group;
exception
    ...
end;
/

10.3.2 Finding a Page ID

When working with pages and items using the content management APIs, you need to know the ID of the page in which you want to work. For example, if you want to delete a page using the delete_page API, you need to identify the page that you want to delete by its ID. Example 10–4 shows how to find out the ID of a page if you know the ID of the page group to which it belongs.

Example 10–4  Finding the ID of Page in a Known Page Group

select id "Page ID"
from wwsbr_all_folders
where name = '&page'
    and caid = 53
    and language = '&language'
/

Sometimes you may not know the ID of the page group to which a particular page belongs. Example 10–5 shows how to find out the ID of a page and its owning page group.

Example 10–5  Finding the IDs for a Page and Its Page Group Given Page and Page Group Names

column 'Page Group Name' format a25;
column 'Page Name' format a25;
select f.caid "Page Group ID",
    c.name "Page Group Name",
    f.id   "Page ID",
    f.name "Page Name"
from wwsbr_all_content_areas c,
    wwsbr_all_folders f
where f.name = '&page'
    and c.id = f.caid
    and c.name = '&pagegroup'
    and c.language = '&language'
    and c.language = f.language
/
10.3.3 Finding Region IDs

When adding an item to a page using the `add_item` API, you need to know the ID of the region in which you want to place the item. You can add items only to item regions that have been set up to allow content to be added to them. Example 10–6 shows a query that finds out the IDs and types of the insertable regions on a particular page. Given this information, you can choose the region ID of an appropriate region on the page in which to place your new item.

**Example 10–6 Finding the IDs and Types of Insertable Regions on a Given Page**

```sql
select distinct r.id "Region ID",
        r.type "Type"
from wwsbr_all_content_areas c,
     wwsbr_all_folders f,
     wwsbr_all_folder_regions r
where f.name = '&page'
    and c.name = '&pagegroup'
    and c.id = f.caid
    and f.id = r.folder.id
    and f.caid = r.folder.caid
    and r.allow_content = 1
    and c.language = '&language'
    and c.language = f.language
    and (f.language = r.language
         or r.language is null)
/```

10.3.4 Finding an Item ID

Example 10–7 illustrates how to find the ID of an item by querying the WWSBR_ALL_ITEMS view.

**Example 10–7 Finding the ID of the Current Version of an Item Given its Master Item ID**

```sql
select id
from wwsbr_all_items
where masterid = 513
    and caid = 53
    and active = 1
    and is_current_version = 1
    and language = '&language'
/```

To avoid duplicate rows when querying for currently published items, always include the LANGUAGE, ACTIVE (active=1), and IS_CURRENT_VERSION (is_current_version=1) columns. Example 10–8 shows how to select all items on a given page (folder_id=1) and a given page group (caid=75).

**Example 10–8 Finding Item IDs**

```sql
select i.display_name title, i.id latestversion
from wwsbr_all_items i
where i.folder_id = 1
    and i.caid = 75
    and i.active = 1
    and i.is_current_version = 1
    and (i.language = '&language' -- The current session language.
         or ( exists -- A row for the item in the page group default language.
             (select pg.id
      /```
from wwsbr_all_content_areas pg
where pg.id = i.caid
    and pg.default_language = i.language
)
and not exists -- A row for the item in the current language.
(select i2.id
from wwsbr_all_items i2
where i2.id = i.id
    and i2.language = i.language
    and i2.active = i.active
    and i2.is_current_version = i.is_current_version
)
)
}
order by i.id
/

This chapter describes how to use the APIs provided with OracleAS Portal to perform simple content management tasks. It contains the following sections:

- Editing Page Properties
- Editing Content
- Reorganizing Content
- Copying Content
- Deleting Content

For more information about any of the APIs mentioned in this chapter, refer to the Oracle Application Server Portal PL/SQL API Reference on Portal Center:

http://portalcenter.oracle.com

In the Portal Focus Areas section, click Portlet Development, then in the APIs and References section, click PL/SQL API Reference.

Tip: Remember, if you are calling the APIs from a Web provider or external application, you need to set the session context first. For more information, refer to Section 10.1, "Setting the Session Context".

### 11.1 Editing Page Properties

If you want to edit page properties, you need to:

1. Query the WWSBR_USER_PAGES view to populate a page record. You may need to create this view first (refer to Section 11.1.1, "Creating the WWSBR_USER_PAGES View").
2. Modify the properties you want to update.
3. Pass the updated page record to the modify_folder API.

You can update the properties shown in Table 11–1 in the page record.

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>VARCHAR2(60)</td>
<td>Name of the page. This name is used in path based URLs.</td>
</tr>
</tbody>
</table>
Table 11–1 (Cont.) Editable Page Record Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLE</td>
<td>VARCHAR(256)</td>
<td>Display name, or title, of the page.</td>
</tr>
<tr>
<td>SETTINGSSETID</td>
<td>NUMBER</td>
<td>ID of the style used by the page.</td>
</tr>
<tr>
<td>SETTINGSSETSITEID</td>
<td>NUMBER</td>
<td>Page group ID of the style used by the page.</td>
</tr>
<tr>
<td>ISPUBLIC</td>
<td>NUMBER(1)</td>
<td>Indicates that the page is viewable by public users. Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 0 - not public</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 1 - is public</td>
</tr>
<tr>
<td>IMAGE</td>
<td>VARCHAR2(350)</td>
<td>Unique document name of the image associated with the page, for example 6001.JPG.</td>
</tr>
<tr>
<td>ROLLOVERIMAGE</td>
<td>VARCHAR2(350)</td>
<td>Unique document name of the rollover image associated with the page, or the inactive tab image for the tab, for example 6001.JPG.</td>
</tr>
<tr>
<td>TITLEIMAGE</td>
<td>VARCHAR2(350)</td>
<td>Unique document name of the active tab image for the tab, for example 6001.JPG.</td>
</tr>
<tr>
<td>LEADER</td>
<td>VARCHAR2(256)</td>
<td>E-mail address of the page contact.</td>
</tr>
<tr>
<td>CREATEDATE</td>
<td>DATE</td>
<td>Date the page was created.</td>
</tr>
<tr>
<td>CREATOR</td>
<td>VARCHAR2(256)</td>
<td>User name of the person who created the page.</td>
</tr>
<tr>
<td>HAVEITEMSECURITY</td>
<td>NUMBER(1)</td>
<td>Indicates that item level security is enabled for the page. Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 0 - ILS disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 1 - ILS enabled</td>
</tr>
<tr>
<td>ITEMVERSIONING</td>
<td>VARCHAR2(30)</td>
<td>Indicates the level of item versioning for the page. Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ versionnone - no versioning</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ versionsimple - simple versioning</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ versionaudit - audit versioning</td>
</tr>
<tr>
<td>TOPICID</td>
<td>NUMBER</td>
<td>ID of the category assigned to the page.</td>
</tr>
<tr>
<td>TOPIC_SITEID</td>
<td>NUMBER</td>
<td>Page group ID of the category assigned to the page.</td>
</tr>
<tr>
<td>Property</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>VALUE</td>
<td>VARCHAR2(2000)</td>
<td>For PL/SQL pages, the PL/SQL code. For JSP pages, the JSP source document name. Do not change this value for JSP pages.</td>
</tr>
<tr>
<td>IS_PORTLET</td>
<td>NUMBER(1)</td>
<td>Indicates if the page is published as a portlet. Valid values: 0 - not a portlet, 1 - is a portlet</td>
</tr>
<tr>
<td>PLSQL_EXECUTOR</td>
<td>VARCHAR2(30)</td>
<td>For PL/SQL type pages, the database schema used to execute the PL/SQL code. Valid values: $PUBLIC$, $CREATOR$, &lt;database user name&gt;</td>
</tr>
<tr>
<td>KEYWORDS</td>
<td>VARCHAR2(2000)</td>
<td>Keywords for the page.</td>
</tr>
<tr>
<td>IS_READY</td>
<td>NUMBER(1)</td>
<td>Indicates that page creation is complete. Valid values: 1 - page creation is complete</td>
</tr>
<tr>
<td>INHERIT_PRIV</td>
<td>VARCHAR2(200)</td>
<td>The page from which this page inherits its privileges. Use the following format: &lt;page group id&gt;/&lt;page id&gt;</td>
</tr>
<tr>
<td>CACHE_MODE</td>
<td>NUMBER(1)</td>
<td>Caching mode for the page. Valid values: 2 - no caching, 1 - cache page definition only, 0 - cache page definition and content for x minutes</td>
</tr>
<tr>
<td>CACHE_EXPIRES</td>
<td>NUMBER(38)</td>
<td>Cache period in minutes.</td>
</tr>
<tr>
<td>ALLOW_PAGE_STYLE</td>
<td>NUMBER(1)</td>
<td>For templates, indicates if pages can use a different style. Valid values: 1 - allow pages to use different style.</td>
</tr>
<tr>
<td>Property</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ALLOW_PAGE_ACL</td>
<td>NUMBER(!)</td>
<td>For templates, indicates if pages have different access settings. Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 1 - allow pages to have different access settings</td>
</tr>
<tr>
<td>INIT_JSPFILE</td>
<td>VARCHAR2(256)</td>
<td>For JSP pages, initial JSP file if the JSP source of the page is a JAR or WAR file.</td>
</tr>
<tr>
<td>UI_TEMPLATE_ID</td>
<td>NUMBER(38)</td>
<td>ID of HTML page skin.</td>
</tr>
<tr>
<td>TEMPLATE_ISPUBLIC</td>
<td>NUMBER(1)</td>
<td>Indicates if the template is ready to use. Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 1 - template is ready to use</td>
</tr>
<tr>
<td>CONTAINER_ID</td>
<td>NUMBER</td>
<td>ID of the container page.</td>
</tr>
<tr>
<td>DEFAULT_ITEM_REGION_ID</td>
<td>NUMBER</td>
<td>ID of the default item region for the page.</td>
</tr>
<tr>
<td>DEFAULT_PORTLET_REGION_ID</td>
<td>NUMBER</td>
<td>ID of the default portlet region for the page.</td>
</tr>
<tr>
<td>ITEMTYPE_INHERIT_FLAGS</td>
<td>NUMBER(1)</td>
<td>For WebDAV, indicates if default item types are inherited from parent page. Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 7 - inherit all item types from parent page</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 0 - specify all types on this page</td>
</tr>
<tr>
<td>REGFILE_ITEMTYPE</td>
<td>RAW(32)</td>
<td>For WebDAV, GUID of default item type for regular files.</td>
</tr>
<tr>
<td>ZIPFILE_ITEMTYPE</td>
<td>RAW(32)</td>
<td>For WebDAV, GUID of default item type for zip files.</td>
</tr>
<tr>
<td>IMAGEFILE_ITEMTYPE</td>
<td>RAW(32)</td>
<td>For WebDAV, GUID of default item type for image files.</td>
</tr>
<tr>
<td>DISPLAYINPARENT</td>
<td>NUMBER(1)</td>
<td>Indicates if the page is displayed in its parent page’s sub page region. Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 1 - display page in parent’s sub page region</td>
</tr>
<tr>
<td>SEQ</td>
<td>NUMBER</td>
<td>The sequence (order) of the page in its parent page’s sub page region.</td>
</tr>
<tr>
<td>ALPHABETICAL_SORT</td>
<td>NUMBER(1)</td>
<td>Indicates that sub pages are displayed in alphabetical order. Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 1 - display sub pages in alphabetical order</td>
</tr>
<tr>
<td>ITEMTYPE_ID</td>
<td>NUMBER(38)</td>
<td>ID of the item template.</td>
</tr>
<tr>
<td>ITEMTYPE_SITE_ID</td>
<td>NUMBER(28)</td>
<td>Page group ID ID of the item template.</td>
</tr>
</tbody>
</table>
Users need to have Manage privileges on a page to edit its properties.

When updating translatable page attributes, modify_folder updates the translation determined by the session language setting. Therefore to make sure you update the translation with the correct values for all the translatable attributes, you must query the page attributes of the same language first.

Example 11–1 shows how to use the modify_folder API to edit the English translation of the display name of a page.

Example 11–1 Editing Page Properties

declare
   l_page     wwsbr_api.page_record;
begin
   wwctx_api.set_nls_language(
      p_nls_language => wwnls_api.AMERICAN
   );
   select *
   into l_page
   from   <schema>.wwsbr_user_pages
   where  siteid = 33
         and id = 1
         and language = wwnls_api.AMERICAN;
   l_page.title := 'New Page Display Name';
   wwsbr_api.modify_folder(
      p_page => l_page
   );
   -- Process cache invalidation messages.
   wwpro_api_invalidation.execute_cache_invalidation;

<table>
<thead>
<tr>
<th>Property</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITEM_PAGE_TABSTRING</td>
<td>VARCHAR2(512)</td>
<td>Tab strings of the item template. Use the following format:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;tab name&gt;:&lt;sub tab name&gt;:...:&lt;sub tab name&gt;</td>
</tr>
<tr>
<td>INHERIT_ITEM_PAGE</td>
<td>NUMBER(1)</td>
<td>For item template only, indicates that items inherit the parent page's item</td>
</tr>
<tr>
<td></td>
<td></td>
<td>template. Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 1 - inherit parent page's item template</td>
</tr>
<tr>
<td>ALLOW_ITEM_PAGE_OVERRIDE</td>
<td>NUMBER(1)</td>
<td>For item template only, indicates that items can have their own item template. Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 1 - allow items on the page to have their own item template</td>
</tr>
<tr>
<td>HAS_INPLACE_ITEM</td>
<td>NUMBER(1)</td>
<td>Indicates if the page or tab has a placeholder item. Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ 1 - has a placeholder item</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>NUMBER</td>
<td>Limit time, in seconds, used to fetch portlets.</td>
</tr>
</tbody>
</table>
exception
...
end;
/

p_page is the page record for the page you want to modify.
schema is the portal schema (that is, the schema in which OracleAS Portal is installed).

### 11.1.1 Creating the WWSBR_USER_PAGES View

The modify_folder API takes a page record as a parameter. To populate the page record, you need to query the data from the WWSBR_USER_PAGES view. This view lists all the pages on which the current user has Manage privileges. If the WWSBR_USER_PAGES view does not already exist in your portal schema, you will need to create it.

To create the WWSBR_USER_PAGES view:

1. Log in to SQL*Plus as the portal schema owner, for example:
   
   ```sql
   sqlplus portal/oracle1
   ```

2. Create the WWSBR_USER_PAGES view:
   
   ```sql
   create or replace view WWSBR_USER_PAGES as
   select *
   from WWV_USER_OWNED_CORNERS
   /
   ```

3. If the procedures and packages that reference this view exist in a schema other than the portal schema, you need to grant privileges on the view to that schema:
   
   ```sql
   grant select on WWSBR_USER_PAGES to <schema>
   /
   ```

### 11.2 Editing Content

This section contains the following examples:

- Setting Item Attributes
- Editing an Item
- Checking Items Out and In
- Using Version Control

---

**Note:** When editing an item, it is good practice to check the item out first to ensure that any changes made by your code are not overwritten by other users. Don’t forget to check the item back in after you have completed your edits. For information about checking items out and in, refer to Section 11.2.3, “Checking Items Out and In”.

### 11.2.1 Setting Item Attributes

If you want to set a particular attribute for an item, use the set_attribute API (Example 11–2). This API updates the value of an attribute for a particular version of an item. The values provided are validated against the data type of the attribute. You
can use this API to set the value of a base attribute (see Table 11–2) or a custom attribute. You cannot use this API to edit rejected items, items marked for deletion, and rejected items that have been marked for deletion.

**Tip:** Use the set_attribute API when you want to set a single item attribute. If you want to set multiple item attributes, use the modify_item API (described in Section 11.2.2, “Editing an Item”).

**Example 11–2 Setting the Display Name of an Item (set_attribute API)**

```sql
begin
  wwsbr_api.set_attribute(
    p_site_id => 37,
    p_thing_id => 8056,
    p_attribute_site_id => wwsbr_api.SHARED_OBJECTS,
    p_attribute_id => wwsbr_api.ATTRIBUTE_TITLE,
    p_attribute_value => 'New Display Name'
  );

  -- Process cache invalidation messages.
  wwpio_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
```

- **p_site_id** is the ID of the page group to which the item belongs.
- **p_thing_id** is the unique item ID. The item ID identifies the particular version of the item that you want to edit.
- **p_attribute_site_id** is the ID of the page group to which the attribute belongs. For base attributes, use the wwsbr_api.SHARED_OBJECTS constant.
- **p_attribute_id** is the ID of the attribute. For base attributes, use the constants from the WWSBR_API package as shown in Table 11–2.
- **p_attribute_value** is the new value that you want to assign to the attribute. For base attributes, see Table 11–2.

For information about how to obtain an object ID, for example a page group ID or item ID, refer to Section 10.3, “Finding an Object ID”.

Table 11–2 provides guidelines for setting the values of base attributes.

**Table 11–2 Attribute Values for Base Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Constant</th>
<th>Value (p_attribute_value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>wwsbr_api.ATTRIBUTE_AUTHOR</td>
<td>Any varchar2(50) value</td>
</tr>
<tr>
<td>Category</td>
<td>wwsbr_api.ATTRIBUTECATEGORY&lt;pagegroupid&gt;_&lt;categoryid&gt;</td>
<td>CATEGORY</td>
</tr>
</tbody>
</table>
### Table 11–2 (Cont.) Attribute Values for Base Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Constant</th>
<th>Value (p_attribute_value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character Set</td>
<td>wwsbr_api.ATTRIBUTE_CHARSET</td>
<td>■ wwsbr_api.VALUE_BIG5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_EUC_JP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_GBK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_ISO_8859</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_SHIFT_JIS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_US_ASCII</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_UTF8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_WINDOWS_1252</td>
</tr>
<tr>
<td>Description</td>
<td>wwsbr_api.ATTRIBUTE_DESCRIPTION</td>
<td>Any varchar2(2000) value</td>
</tr>
<tr>
<td>Display Name</td>
<td>wwsbr_api.ATTRIBUTE_TITLE</td>
<td>Any varchar2(256) value</td>
</tr>
<tr>
<td>Display Option</td>
<td>wwsbr_api.ATTRIBUTE_DISPLAYOPTION</td>
<td>■ wwsbr_api.IN_PLACE to display the item directly in place</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.FULL_SCREEN to display the item in the full browser window</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.NEW_WINDOW to display the item in a new browser window</td>
</tr>
<tr>
<td>Enable Item Check-out</td>
<td>wwsbr_api.ATTRIBUTE_ITEMCHECKOUT</td>
<td>■ wwsbr_api.ENABLE_ITEM_FOR_CHECK_OUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.DISABLE_ITEM_FOR_CHECK_OUT</td>
</tr>
<tr>
<td>Expiration Period</td>
<td>wwsbr_api.ATTRIBUTE_EXPIRATIONPER</td>
<td>■ wwsbr_api.PERMANENT to un-expire the item</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Any numeric value to set the expire mode to NUMBER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Any date value greater than the publish date to set the expire date to the value passed and the expire mode to DATE</td>
</tr>
<tr>
<td>File</td>
<td>wwsbr_api.ATTRIBUTE_FILE</td>
<td>The path and file name of the file that you want to upload</td>
</tr>
<tr>
<td>Image</td>
<td>wwsbr_api.ATTRIBUTE_IMAGE</td>
<td>The path and file name of the image that you want to upload</td>
</tr>
<tr>
<td>ImageMap</td>
<td>wwsbr_api.ATTRIBUTE_IMAGEMAP</td>
<td>Any varchar2(4000) value</td>
</tr>
</tbody>
</table>
### Table 11–2 (Cont.) Attribute Values for Base Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Constant</th>
<th>Value (p_attribute_value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Alignment</td>
<td>wwsbr_api.IMAGEALIGN</td>
<td>- wwsbr_api.ALIGN_TEXT_TOP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- wwsbr_api.ALIGN_ABSOLUTE_BOTTOM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- wwsbr_api.ALIGN_ABSOLUTE_MIDDLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- wwsbr_api.ALIGN_BOTTOM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- wwsbr_api.ALIGN_RIGHT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- wwsbr_api.ALIGN_TOP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- wwsbr_api.ALIGN_LEFT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- wwsbr_api.ALIGN_MIDDLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- wwsbr_api.ALIGN_BASELINE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- wwsbr_api.ALIGN_IMAGE_ABOVE_LINK</td>
</tr>
<tr>
<td>Item Link</td>
<td>wwsbr_api.ITEM_LINK</td>
<td>&lt;pagegroupid&gt;_&lt;itemid&gt;</td>
</tr>
<tr>
<td>Item Template</td>
<td>wwsbr_api.ITEM_TEMPLATE</td>
<td>&lt;pagegroupid&gt;_&lt;itemtemplateid&gt;</td>
</tr>
<tr>
<td>Keywords</td>
<td>wwsbr_api.KEYWORDS</td>
<td>Any varchar2(2000) value</td>
</tr>
<tr>
<td>Mime Type</td>
<td>wwsbr_api.MIME_TYPE</td>
<td>Any of the mime type constants in the WWSBR_API package. For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- wwsbr_api.VALUE_TEXT_HTML</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- wwsbr_api.VALUE_TEXT_PLAIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- wwsbr_api.VALUE_IMAGE_GIF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For a list of the available mime type constants, refer to the Oracle Application Server Portal PL/SQL API Reference on Portal Center:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://portalcenter.oracle.com">http://portalcenter.oracle.com</a></td>
</tr>
<tr>
<td>Name</td>
<td>wwsbr_api.NAME</td>
<td>Any varchar2(256) value</td>
</tr>
<tr>
<td>Page Link</td>
<td>wwsbr_api.PAGE_LINK</td>
<td>&lt;pagegroupid&gt;_&lt;pageid&gt;</td>
</tr>
<tr>
<td>Perspectives</td>
<td>wwsbr_api.PERSPECTIVES</td>
<td>&lt;pagegroupid&gt;_&lt;perspectiveid&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Separate multiple perspectives with commas.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Non-existing perspective IDs are identified by the API. Only those perspectives that are available to the page group are added to the item.</td>
</tr>
<tr>
<td>PL/SQL</td>
<td>wwsbr_api.PLSQL</td>
<td>Any varchar2(2000) value</td>
</tr>
</tbody>
</table>
Table 11–2 (Cont.) Attribute Values for Base Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Constant</th>
<th>Value (p_attribute_value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publish Date</td>
<td>wwsbr_api.ATTRIBUTE_PUBLISHDATE</td>
<td>Any date value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This is a required attribute.</td>
</tr>
<tr>
<td>Rollover Image</td>
<td>wwsbr_api.ATTRIBUTE_ROLLOVERIMAGE</td>
<td>The path and file name of the image file that you want to upload</td>
</tr>
<tr>
<td>Smart Link</td>
<td>wwsbr_api.ATTRIBUTE_SMARTLINK</td>
<td>■ wwsbr_api.VALUE_ACCOUNT_INFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_ADVANCED_SEARCH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_BUILDER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_COMMUNITY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_CONTACT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_EDIT_PAGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_FAVOURITES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_HELP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_PORTAL_HOME</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_CUST_MOBILE_HOME_PAGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_NAVIGATOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_PAGE_GROUP_HOME</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_PERSONAL_PAGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_CUSTOMIZE_PAGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_PORTLET_REPOS_REF_STATUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_PORTLET_REPOS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_PROPERTY_SHEET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_REF_PORTLET_REPOS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_REFRESH_PAGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_SUBSCRIBE</td>
</tr>
<tr>
<td>Smart Text</td>
<td>wwsbr_api.ATTRIBUTE_SMARTTEXT</td>
<td>■ wwsbr_api.VALUE_SMART_TEXT_CURRENT_DATE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_SMART_TEXT_CURRENT_PAGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ wwsbr_api.VALUE_SMART_TEXT_CURRENT_USER</td>
</tr>
<tr>
<td>Text</td>
<td>wwsbr_api.ATTRIBUTE_TEXT</td>
<td>Any varchar2(4000) value</td>
</tr>
<tr>
<td>URL</td>
<td>wwsbr_api.ATTRIBUTE_URL</td>
<td>Any URL value up to varchar2(2000)</td>
</tr>
</tbody>
</table>
11.2.2 Editing an Item

To edit an existing item, use the `modify_item` API. As well as setting the properties of the item, this API also enables you to:

- Add new versions of the item (if item versioning is enabled for the page). See Section 11.2.4, "Using Version Control".
- Enable item level security for the item (if item level security (ILS) is enabled for the page). See Section 15.3, "Setting Item Level Privileges" for an alternative way of doing this.
- Upload files that are associated with the item. If the file that you want to associate with the item has already been uploaded, use the `modify_item_post_upload` API instead.

You should use this API to edit only those items with an ACTIVE value of 1. That is, do not use it to edit pending, rejected, or deleted items.

You must pass both the item's master ID to identify the item and its ID to identify the specific version of the item that you want to update. The specified ID does not have to be the current version of the item. For information about obtaining an object ID, refer to Section 10.3, "Finding an Object ID".

The `modify_item` API modifies the value of every attribute, even if the attribute is not passed in the parameter list. If an attribute is not passed, its value will revert to the default value of the parameter. If you want to preserve the current value of attributes, you must retrieve those values for the item and pass them to the API. The value of attributes can be retrieved from the following views and APIs:

- WWSBR_ALL_ITEMS for built-in attributes
- WWSBR_ITEM_ATTRIBUTES for custom attributes
- WWSBR_ITEM_TYPE_ATTRIBUTES for a list of attributes for an item type
- WWSBR_ITEM_PERSPECTIVES for perspectives
- `wwsec_api.grantee.list` for a list of privileges on the item type. See Section 15.3, "Setting Item Level Privileges" for more information on using this API.

Example 11–3 updates the display name and text of a text item.

**Example 11–3 Editing an Item (modify_item API)**

declare
l_item_master_id number;
beginn
l_item_master_id := wwsbr_api.modify_item(
p_master_item_id => 453,
p_item_id => 454,
p_caid => 33,
p_folder_id => 45,
p_display_name => 'Movie Review',
p_text => 'This is the text of the review.'
);
-- Process cache invalidation messages.
wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;

■ p_master_item_id is the master ID of the item. You can look up this value in the
  MASTERID column of the WWSBR_ALL_ITEMS view.
■ p_item_id is the ID of the item. This identifies the version of the item and does not
  have to be the item's current version.
  
  Tip: To identify the item’s current version, use the following query:

      select id
      from wwsbr_all_items
      where master_id = 453
          and is_current_version = 1;

■ p_caid is the ID of the page group to which the item belongs.
■ p_folder_id is the ID of the page on which the item appears.
■ p_display_name is the display name (title) of the item.
■ p_text is the text for the item (the item in this example is a text item).

11.2.3 Checking Items Out and In

If an item is enabled for check-out, before editing it, you should check the item out so
that other users cannot make changes at the same time (Example 11–4).

Example 11–4 Checking an Item Out and In (check_out_item and check_in_item)
begin
  -- Check out the item.
  wwsbr_api.check_out_item(
    p_master_item_id => 12345, -- Master ID is the same for all versions.
    p_caid => 33
  );
  -- Update the display name of the item.
  wwsbr_api.set_attribute(
    p_site_id => 33,
    p_thing_id => 8056, -- Unique item ID.
    p_attribute_site_id => wwsbr_api.SHARED_OBJECTS,
    p_attribute_id => wwsbr_api.ATTRIBUTE_TITLE,
    p_attribute_value => 'New Display Name'
  );
  -- Check the item back in.
  wwsbr_api.check_in_item(
    p_master_item_id => 12345, -- Master ID is the same for all versions.
    p_caid => 33
  );
11.2.4 Using Version Control

When you edit an item using the modify_item API, if you set $p_{\text{addnewversion}} = \text{true}$, a new version is created for the item (Example 11–5). The new version is given a new ID (the master ID stays the same). If you do not want the new version of the item to be immediately published as the current version, set $p_{\text{add\_as\_current\_version}} = \text{false}$.

**Example 11–5 Creating a New Version of an Item (modify_item API)**

```plsql
declare
    l_item_master_id number;
begin
    l_item_master_id := wwsbr_api.modify_item(
        p_master_item_id => 453,
        p_item_id        => 454,
        p_caid           => 33,
        p_folder_id      => 45,
        p_display_name   => 'Movie Review',
        p_text           => 'This is the text of the review.',
        p_addnewversion  => true
    );
    -- Process cache invalidation messages.
    wwpro_api_invalidation.execute_cache_invalidation;
except
    ...
end;
/
```

For a description of the parameters used here, see Example 11–3.

11.3 Reorganizing Content

Content is very rarely static, so you may find that you need to move content around within your portal.

This section contains the following examples:

- Moving an Item to a Different Page
- Moving Pages
11.3.1 Moving an Item to a Different Page

You can use the move_item API to move an item to a different page in the same page group or to a page in a different page group. Moving an item preserves the item’s metadata. To move an item to a different page group, the item must not be:

- based on a local item type.
- associated with a local category.
- associated with any local perspectives.

Example 11–6 moves an item from one page group to a page in a different page group.

Example 11–6  Moving an Item to a Different Page Group (move_item API)

begin
  wwsbr_api.move_item(
    p_caid           => 33,
    p_master_item_id => 12345,
    p_dest_caid      => 53,
    p_dest_page_id   => 1,
    p_dest_region_id => 5
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/

- **p_caid** is the ID of the page group to which the item currently belongs.
- **p_master_item_id** is the master ID of the item that you want to move.
- **p_dest_caid** is the ID of the page group to which you want to move the item.
- **p_dest_page_id** is the ID of the page to which you want to move the item.
- **p_dest_region_id** is the ID of the item region of the page in which you want to move the item. If you do not specify a region ID, the item is moved to the page’s default item region, which may not be desirable.

11.3.2 Moving Pages

You can use the move_folder API to move a page within the same page group. You cannot move a page to a different page group.

To move a page, users must have Manage privileges on the page being moved and the page under which the page is being moved.

Example 11–7 shows how to move a page.

Example 11–7  Moving a Page (move_folder API)

begin
  wwsbr_api.move_folder(
    p_id        => 12345,
    p_siteid    => 33,
    p_parent_id => 10000
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
11.3.3 Moving Categories and Perspectives

You can, if necessary, move categories and perspectives around within a page group. You cannot move a category or perspective to a different page group.

To move a category or perspective, a user must have Manage Classifications privileges or higher on the owning page group.

To move a category, use the move_category API (Example 11–8).

**Example 11–8 Moving a Category (move_category API)**

```ruby
begin
  wwsbr_api.move_category(
    p_src_id => 2000,
    p_dest_id => 3000,
    p_siteid => 33
  );
exception
  ...
end;
```

To move a perspective, use the move_perspective API (Example 11–9).

**Example 11–9 Moving a Perspective (move_perspective API)**

```ruby
begin
  wwsbr_api.move_perspective(
    p_src_id => 2000,
    p_dest_id => 3000,
    p_siteid => 33
  );
exception
  ...
end;
```

- **p_src_id** is the ID of the category or perspective that you want to move.
- **p_dest_id** is the ID of the category or perspective under which you want to move the category or perspective.
- **p_siteid** is the ID of the page group to which the category or perspective belongs.

11.4 Copying Content

If you want to add content that is the same as, or similar to, some content that already exists, instead of creating it all over again, you can copy the existing content.

This section contains the following examples:
11.4.1 Copying Items

You can use the copy_item API to copy an item to a different page in the same page group or to a page in a different page group. To copy an item to a different page group, the item must not be:

- based on a local item type
- associated with a local category
- associated with any local perspectives

Copying an item preserves the item’s metadata.

Example 11–10 copies an item to a page in a different page group.

**Example 11–10 Creating a Copy of an Item in a Different Page Group (copy_item API)**

```sql
begin
  wwsbr_api.copy_item(
    p_caid           => 33,
    p_master_item_id => 12345,
    p_dest_caid      => 53,
    p_dest_page_id   => 1,
    p_dest_region_id => 5
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

- **p_caid** is the ID of the page group to which the item currently belongs.
- **p_master_item_id** is the master ID of the item that you want to copy.
- **p_dest_caid** is the ID of the page group to which you want to copy the item.
- **p_dest_page_id** is the ID of the page to which you want to copy the item.
- **p_dest_region_id** is the ID of the item region of the page in which you want to copy the item. If you do not specify a region ID, the item is copied to the page’s default item region, which may not be desirable.

11.4.2 Copying Pages

You can use the copy_folder API to copy a page within the same page group. You cannot copy a page to a different page group.

To copy a page, users must have View privileges on the page being copied and Manage privileges on the page under which the page is being copied.

Example 11–11 shows how to copy a page.

**Example 11–11 Creating a Copy of a Page (copy_folder API)**

```sql
declare
  l_new_pageid number;
begin
```

---

---
l_new_pageid := wwsbr_api.copy_folder(
    p_id => 12345,
    p_siteid => 33,
    p_parent_id => 10000,
    p_name => 'pagel',
    p_title => 'page1'
);
-- Process cache invalidation messages.
wwpro_api_invalidation.execute_cache_invalidation;

exception...
end;
/

- **p_id** is the ID of the page that you want to copy.
- **p_siteid** is the ID of the page group to which the page belongs.
- **p_parent_id** is the ID of the page under which you want to copy the page.
- **p_name** is the name for the new page created by the copy operation.
- **p_title** is the display name for the new page created by the copy operation.

### 11.5 Deleting Content

When content is no longer required, you can use APIs to remove it from the portal and free up space.

This section contains the following examples:

- Deleting Items
- Deleting Pages

#### 11.5.1 Deleting Items

You can use the delete_item API to delete all versions of an item or just a single specific version. This API also deletes all sub items associated with the versions being deleted. When you delete an item using this API, you can choose to accept the default delete mode for the page group or override this setting. Example 11–12 deletes version 1 of an item permanently from the page group. The item version is permanently removed, even if the page group is set up to retain deleted items.

**Example 11–12  Deleting an Item (delete_item API)**

```sql
begin
    procedure delete_item(
        p_master_item_id => 48037,
        p_caid => 54,
        p_version_number => 1,
        p_mode => wwsbr_api.DELETE_ITEM_PURGE
    );
    -- Process cache invalidation messages.
    wwpro_api_invalidation.execute_cache_invalidation;
    exception...
end;
/```
Deleting Content

- **p_master_item_id** is the master ID of the item that you want to delete.
- **p_caid** is the ID of the page group to which the item belongs.
- **p_version_number** is the version of the item that you want to delete. If you attempt to delete the current version of the item, the ITEM_ACTIVE_VERSION exception is raised. If you do not specify an item version, all versions of the item are deleted.

  **Tip:** If you want to delete the current version of the item, you must either delete all versions, or use the OracleAS Portal user interface to revert to a previous version.

- **p_mode** is the mode to use when deleting the item:
  - wwsbr_api.DELETE_ITEM_DEFAULT uses the page group setting.
  - wwsbr_api.DELETE_ITEM_PURGE deletes the item immediately, regardless of the page group setting.
  - wwsbr_api.DELETE_ITEM_LAZYDELETE marks the item as deleted, regardless of the page group setting.

To restore a previously deleted item, use the **undelete_item** API (Example 11–13). If the item had any sub-items, those sub-items are also restored.

```
Note: You can restore a deleted item only if the page group that owns the item is set up to retain deleted items, or the item was deleted using the wwsbr_api.DELETE_ITEM_LAZYDELETE mode, and the item has not been purged from the content repository.
```

**Example 11–13  Restoring a Previously Deleted Item (undelete_item API)**
```
begin
  wwsbr_api.undelete_item(
    p_thing_id => 12345,
    p_caid     => 33
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

- **p_thing_id** is the unique ID of the deleted item that you want to restore.
- **p_caid** is the ID of the page group to which the item belongs.

11.5.2 Deleting Pages

Use the **delete_folder** API to delete a page. Users need to have Manage privileges on the page to be able to delete it.

```
Note: Deleted pages cannot be restored by another API call.
```

**Example 11–14** shows how to delete a page.
Example 11–14  Deleting a Page (delete_folder API)

begin
    wwsbr_api.delete_folder(
        p_id   => 12345,
        p_siteid => 33
    );
    -- Process cache invalidation messages.
    wwpro_api_invalidation.execute_cache_invalidation;
except
    ...
end;
/

- **p_id** is the ID of the page that you want to delete.
- **p_siteid** is the ID of the page group to which the page belongs.
This chapter describes how to use the APIs provided with OracleAS Portal to extend your portal. It contains the following sections:

- Creating a Page Group
- Creating Pages
- Creating Categories and Perspectives
- Creating Items
- Setting Perspectives Attributes of Pages and Items
- Approving and Rejecting Items

For more information about any of these APIs, refer to the *Oracle Application Server Portal PL/SQL API Reference* on Portal Center:

http://portalcenter.oracle.com

In the Portal Focus Areas section, click *Portlet Development*, then in the APIs and References section, click *PL/SQL API Reference*.

**Tip:** Remember, if you are calling the APIs from a Web provider or external application, you need to set the session context first. For more information, refer to Section 10.1, "Setting the Session Context".

### 12.1 Creating a Page Group

To create a new page group, use the `add_content_area` API as shown in Example 12–1.

**Example 12–1 Creating a Page Group (add_content_area API)**

```sql
declare
    l_new_page_group_id number;
begin
    -- Create the page group.
    l_new_page_group_id := wwsbr_api.add_content_area(
        p_name => 'ENTERTAINMENT',
        p_display_name => 'Entertainment Page Group',
        p_versioning => wwsbr_api.VERSIONING_AUDIT,
        p_default_language => 'us'
    );
    -- process cache invalidation messages.
    wwpro_api_invalidation.execute_cache_invalidation;
exception
```

... 
end; 
/

- **p_name** is the internal name for the new page group. This name is used in path based URLs and must be unique.
- **p_display_name** is the display name for the new page group.
- **p_versioning** is the versioning level for items in the page group:
  - wwsbr_api.VERSIONING_NONE
  - wwsbr_api.VERSIONING_SIMPLE
  - wwsbr_api.VERSIONING_AUDIT
- **p_default_language** is the default language for the page group.

## 12.2 Creating Pages

If you want to create a new page, use the add_folder API (Example 12–2). By default, the new page has two regions: a portlet region containing the default navigation page of the page group, and an item region. This means that pages created programmatically, rather than through the OracleAS Portal user interface, may be quite limited in their layout.

However, you can programmatically create pages with a more sophisticated layout by basing the parent page of the new page on a Portal Template with the desired layout. Then if you configure the page group to automatically copy parent page properties, your new page will use the same template as its parent. Using this method means that your pages can have any layout that you require.

You cannot use this API to create JSP pages or navigation pages. To create these types of pages, use the Create Page Wizard in OracleAS Portal.

If the page type on which you base the page has default values set for any page properties, these values override any values set using this API.

To define privileges for your new page, use the APIs in the WWSEC_API package. For more information, refer to Section 15.2, "Setting Page Level Privileges".

### Example 12–2  Creating a Page (add_folder API)

```sql
declare
  l_new_page_id       number;
  l_caid              number := 33;
begin
  -- create the page.
  l_new_page_id := wwsbr_api.add_folder(
    p_caid             => l_caid,
    p_name             => 'ENTERTAINMENT',
    p_display_name     => 'Entertainment Page',
    p_type_id          => wwsbr_api.FOLDER_TYPE_CONTAINER,
    p_type_caid        => wwsbr_api.SHARED_OBJECTS,
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ... 
end; 
/```
p_caid is the ID of the page group to which you want to add the page.

p_name is the internal name for the new page. This name is used in path based URLs.

p_display_name is the display name (title) for the new page.

p_type_id is the ID of the page type on which you want to base the page. The page type must be available to the page group in which you are creating the page. Seeded page types have the following constants defined:

- wwsbr_api.FOLDER_TYPE_CONTAINER
- wwsbr_api.FOLDER_TYPE_URL
- wwsbr_api.FOLDER_TYPE_PLSQL
- A value from the ID column of the WWSBR_FOLDER_TYPES view

You can find the IDs for custom page types by querying the ID column of the WWSBR_FOLDER_TYPES view.

p_type_caid is the ID of the page group to which the page type used for the page belongs. This value must be the same as the page group in which you are creating the page (p_caid) or the Shared Objects page group (use the wwsbr_api.SHARED_OBJECTS constant).

12.3 Creating Categories and Perspectives

If you have already defined a quite large taxonomy for your portal, you might prefer to set it up programmatically, rather than use the OracleAS Portal user interface.

Use the add_category API to create a new category (Example 12–3). Use the add_perspective API to create a new perspective (Example 12–4). To create a category or perspective, users must have Manage Classifications privileges or higher on the page group.

Example 12–3  Creating a Category (add_category API)

declare
  l_new_category_id number;
  l_caid number := 33;
begin
  l_new_category_id := wwsbr_api.add_category( 
    p_caid => l_caid, 
    p_parent_id => 0, 
    p_name => 'newcategory1', 
    p_display_name => 'New Category 1'
  );
  -- Process cache invalidation messages.
  wwpwo_api_invalidation.execute_cache_invalidation;
exception
  ... 
end;
/

Example 12–4  Creating a Perspective (add_perspective API)

declare
  l_new_perspective_id number;
l_caid               number := 33;
begin
l_new_perspective_id := wwsbr_api.add_perspective(
    p_caid         => 33,
    p_parent_id    => 0,
    p_name         => 'newperspective1',
    p_display_name => 'New Perspective 1';
);
-- Process cache invalidation messages.
wwpro_api_invalidation.execute_cache_invalidation;
exception
... end;
/

■ p_caid is the ID of the page group in which you want to create the category or perspective.

■ p_parent_id is the ID of the category or perspective under which you want to create the new category or perspective. If this is a top-level category or perspective, use 0 (zero).

■ p_name is the internal name for the new category or perspective.

■ p_display_name is the display name for the new category or perspective.

12.4 Creating Items

To create an item on a page, use the add_item API (Example 12–5).

This API returns the master ID of the item, which is not the item's unique ID. To look up the ID of the item after it is created, query the WWSBR_ALL_ITEMS view as shown in Example 10–7. For more information about the difference between an item's unique ID and its master item ID, refer to Section 10.2, “API Parameters”.

Example 12–5  Creating a Text Item (add_item API)

declare
    l_new_item_master_id number;
begin
    l_new_item_master_id := wwsbr_api.add_item(
        p_caid              => 33,
        p_folder_id         => 13923,
        p_display_name      => 'Movie Review',
        p_type_id           => wwsbr_api.ITEM_TYPE_TEXT,
        p_type_caid         => wwsbr_api.SHARED_OBEJCTS,
        p_region_id         => 5,
        p_text              => 'This is the text of the review.',
    );
    -- Process cache invalidation messages.
    wwpro_api_invalidation.execute_cache_invalidation;
exception
    ... end;
/

■ p_caid is the ID of the page group that owns the page in which you want to create the item.

■ p_folder_id is the ID of the page in which you want to create the item.
Creating Items

- **p_display_name** is the display name (title) of the new item.
- **p_type_id** is the ID of the item type on which the item is based. Use the constants defined in the WWSBR_API package, as shown in the following examples:
  - wwsbr_api.ITEM_TYPE_FILE
  - wwsbr_api.ITEM_TYPE_TEXT
  - wwsbr_api.ITEM_TYPE_URL

For a full list of constants for the seeded item types, refer to the Oracle Application Server Portal PL/SQL API Reference on Portal Center:

http://portalcenter.oracle.com

In the Portal Focus Areas section, click **Portlet Development**, then in the APIs and References section, click **PL/SQL API Reference**.

You can find the IDs for custom item types by querying the ID column of the WWSBR_ITEM_TYPES view.

The item type must be available in the page group on which you are creating the item, and therefore listed in the WWSBR_CONTENT_AREA_ITEM_TYPES view.

- **p_type_caid** is the ID of the page group to which the item type used for the item belongs. This value must be the same as the page group to which you are adding the item (p_caid), or the Shared Objects page group (use the wwsbr_api.SHARED_OBJECTS constant).

- **p_region_id** is the ID of the region in which you want to create the item. If you do not specify a region, or if the region ID is invalid, the item is placed in the default item region for the page. Use the WWSBR_ALL_FOLDER_REGIONS view to look up the region ID.

- **p_text** is the text for a text item.

### Creating File Items

If an object is associated with one or more files (for example, images, file items, and so on), its APIs allow you to upload these files when creating or modifying the object. You can pass a file location consisting of a directory path and file name to the file and image parameters for these APIs. The directory in which you place the files must be visible to your database (that is, local to the database server) and the files themselves must have proper permissions. If you see strange exceptions resulting from API calls in which you are trying to upload files, double check the permissions on the directories and files.

When the file that you want to upload resides on the same server as the database in which OracleAS Portal is installed, you can upload the file at the same time as you create or update the item. **Example 12–6** shows how you might do this.

**Example 12–6 Creating a File Item (add_item API)**

```sql
declare
  l_item_masterthing_id number;
begin
  -- Add an item that resides on the same server as the OracleAS Portal content repository database.
  l_item_masterthing_id := wwsbr_api.add_item(
    p_caid => 53, -- A known page group ID.
    p_folder_id => 1, -- A known page ID.
    p_display_name => 'My File',
```

p_type_id       => wwsbr_api.ITEM_TYPE_FILE,
p_type_caid    => wwsbr_api.SHARED_OBJECTS,
p_region_id    => 513, -- A known region on the page.
p_description  => 'Description of my file',
p_file_filename => '/tmp/myfile.txt');
-- Process cache invalidation messages.
wwpro_api_invalidation.execute_cache_invalidation;
exception
...
end;
/

■ p_caid is the ID of the page group that owns the page in which you want to create
the item.

■ p_folder_id is the ID of the page in which you want to create the item.

■ p_display_name is the display name (title) of the new item.

■ p_type_id is the ID of the item type on which the item is based.

■ p_type_caid is the ID of the page group to which the item type used for the item
belongs. This value must be the same as the page group to which you are adding
the item (p_caid), or the Shared Objects page group (use the wwsbr_api.SHARED_
OBJECTS constant).

■ p_region_id is the ID of the region in which you want to create the item. If you do
not specify a region, or if the region ID is invalid, the item is placed in the default
item region for the page. Use the WWSBR_ALL_FOLDER_REGIONS view to look
up the region ID.

■ p_description is a description of the item.

■ p_file_filename is the directory path and file name of the file associated with this
item. The file must be located on the same server as database in which OracleAS
Portal is installed.

If the file does not reside on the same server as the database in which OracleAS Portal
is installed, you must first load the file into the content repository document tables
using the upload_blob API. The upload_blob API returns a unique document
name. Use the add_item_post_upload API to create a new item that claims the file
specified by the returned document name. Example 12–7 shows how these APIs can be
used.

---

**Note:** Your calling application must declare a BLOB and assign it to
an object compatible with the database BLOB data type (refer to
*Application Developer’s Guide - Large Objects* in your Oracle Database
documentation set). You can then pass the BLOB to the p_blob
parameter of the upload_blob API.

---

**Example 12–7  Uploading a File to the Content Repository and Creating a File Item Using
the Uploaded File (upload_blob API and add_item_post_upload API)**

declare
    l_blob                blob;
    l_blob_filename       varchar2(250);
    l_mime_type           varchar2(100) := 'text/html';
    l_doc_name            varchar2(500);
    l_display_name        varchar2(250);
Creating Items
Extending Your Portal

l_region_id number := 2013;
l_file_name varchar2(100);
l_image_name varchar2(100);
l_site_id number := 73;
l_page_id number := 1;
l_item_type_id number := wwsbr_api.ITEM_FILE_TYPE;
l_item_type_siteid number := wwsbr_api.SHARED_OBJECTS;
l_description varchar2(1000);
l_item_masterthing_id number;
begin
  -- Your calling application must define the my_get_blob() function and retrieve
  -- your document into a blob so that it can be uploaded in the subsequent step.
l_blob := my_get_blob('8001.HTML');
  -- Upload the BLOB to the OracleAS Portal document table.
l_blob_filename := 'index2.html';
l_doc_name := wwsbr_api.upload_blob(
    p_file_name => l_blob_filename,
    p_blob      => l_blob,
    p_mime_type => l_mime_type
  );
l_display_name := l_blob_filename;
l_file_name := l_doc_name;
l_description := 'File uploaded to portal = ' || l_doc_name;
  -- Use add_item_post_upload() to claim the document and add the item to a page.
l_item_masterthing_id := wwsbr_api.add_item_post_upload(
    p_caid          => l_site_id,
    p_folder_id     => l_page_id,
    p_display_name  => l_display_name,
    p_file_name     => l_file_name,
    p_type_id       => l_item_type_id,
    p_type_caid     => l_item_type_siteid,
    p_region_id     => l_region_id,
    p_description   => l_description,
    p_display_option => wwsbr_api.FULL_SCREEN
  );
  -- Process cache invalidation messages.
  wwpas_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/

Parameters for upload_blob

- **p_file_name** is the file name that you want to assign to the BLOB. This is not the
  value returned by the function. Usually this is the file name by which the file is
  known on the source file system.

- **p_blob** is the BLOB containing the content.

- **p_mime_type** is the MIME type for the BLOB. Use the predefined constants
  provided in the WWSBR_API package.

Parameters for add_item_post_upload

- **p_caid** is the ID of the page group that owns the page in which you want to create
  the item.

- **p_folder_id** is the ID of the page in which you want to create the item.

- **p_display_name** is the display name (title) of the new item.
- **p_file_name** is the internal document name for the file item, matching a document in the document table. Pass the value returned by `upload_blob` to this parameter to reference the uploaded document as the file for this item.

- **p_type_id** is the ID of the item type on which the item is based.

- **p_type_caid** is the ID of the page group to which the item type used for the item belongs. This value must be the same as the page group to which you are adding the item (p_caid), or the Shared Objects page group (use the wwsbr_api.SHARED_OBJECTS constant).

- **p_region_id** is the ID of the region in which you want to create the item. If you do not specify a region, or if the region ID is invalid, the item is placed in the default item region for the page. Use the WWSBR_ALL_FOLDER_REGIONS view to look up the region ID.

- **p_description** is a description of the item.

- **p_file_filename** is the directory path and file name of the file associated with this item. The file must be located on the same server as database in which OracleAS Portal is installed.

- **p_display_option** is how the item should be displayed. Use the following predefined constants:
  - `wwsbr_api.FULL_SCREEN` to display a link that, when clicked, displays the item in the same browser window.
  - `wwsbr_api.NEW_WINDOW` to display a link that, when click, displays the item in a new browser window.
  - `wwsbr_api.IN_PLACE` to display the item in the region (this applies to text and PL/SQL items only).

If you have already created the item that you want to use to claim the uploaded document, use the `modify_item_post_upload` API.

### 12.5 Setting Perspectives Attributes of Pages and Items

When creating a page or item you can, optionally, specify the perspectives that apply to that page or item.

Example 12–8 shows how you specify the perspectives that apply to a page when creating the page. Example 12–9 shows how you specify the perspectives that apply to an item when creating the item.

**Example 12–8  Specifying Perspectives When Creating a Page**

```sql
declare
  l_new_page_id       number;
  l_perspective_ids   wwsbr_api.g_perspectiveidarray;
  l_perspective_caids wwsbr_api.g_caid_array;
begin
  select id, caid
    bulk collect into l_perspective_ids, l_perspective_caids
    from wwsbr_all_perspectives
    where caid in (l_caid, wwsbr_api.shared_objects);
  l_new_page_id := wwsbr_api.add_folder(
    p_caid => 33,
    p_name => 'ENTERTAINMENT',
    p_display_name => 'Entertainment Page',
    p_type_id => wwsbr_api.FOLDER_TYPE_CONTAINER,
```
Example 12–9 Specifying Perspectives When Creating an Item
declare
l_new_item_master_id number;
l_perspective_ids wwsbr_api.q_perspectiveidarray;
l_perspective_caids wwsbr_api.g_caid_array;
begin
-- Select perspectives with the name prefix = 'ENTERTAINMENT'.
select id, caid
bulk collect into l_perspective_ids, l_perspective_caids
from wwsbr_all_perspectives
where caid in (l_caid, wwsbr_api.shared_objects) and
display_name like 'ENTERTAINMENT%';
l_new_item_master_id := wwsbr_api.add_item(
p_caid => 33,
p_folder_id => 13923,
p_display_name => 'Movie Review',
p_type_id => wwsbr_api.ITEM_TYPE_TEXT,
p_type_caid => wwsbr_api.SHARED_OBJECTS,
p_region_id => 5,
p_text => 'This is the text of the review.',
p_perspectives => l_perspective_ids,
p_perspectives_caid => l_perspective_caids
);
-- Process cache invalidation messages.
wwpro_api_invalidation.execute_cache_invalidation;
exception
...
end;
/

- **p_perspectives** is an array of perspective IDs.
- **p_perspective_caid** is an array of page group IDs for the perspectives identified in **p_perspectives**. The position of each element in this array must match the position of the corresponding perspective ID in **p_perspectives**. The values in **p_perspective_caid** must be the same as the page group in which you are creating the page or item (p_caid) or the Shared Objects page group (use the wwsbr_api.SHARED_OBJECTS constant).

For descriptions of the other parameters in these examples, refer to Example 12–2 and Example 12–5.

12.6 Approving and Rejecting Items

The WWSBR_API package includes two APIs to help with managing approvals. To see how you might use these APIs in conjunction with the Content Management Event Framework, refer to Section 16.7, "Example: Item Validation" and Section 16.8, "Example: Integrating External Workflow"
Example 12–10 shows how to approve a pending item. Example 12–11 shows how to reject a pending item.

**Example 12–10 Approving an Item**

```plsql
begin
    wwsbr_api.approve(
        p_item_id => 8056,
        p_site_id => 53,
        p_comment => 'Item approved'
    );
    -- Process cache invalidation messages.
    wwpro_api_invalidation.execute_cache_invalidation;
end;
/
```

**Example 12–11 Rejecting an Item**

```plsql
begin
    wwsbr_api.reject(
        p_item_id => 8056,
        p_site_id => 53,
        p_comment => 'Item rejected'
    );
    -- Process cache invalidation messages.
    wwpro_api_invalidation.execute_cache_invalidation;
end;
/
```

- **p_item_id** is the unique ID of the item being approved or rejected.
- **p_site_id** is the ID of the page group to which the item belongs.
- **p_comment** provides additional information about the approval or rejection of the item.
This chapter describes how to use the public search APIs provided with Oracle AS Portal. The search APIs are available in the WWSRC_API package.

The public search APIs enable you to search your portal programmatically and return search results as content records or an XML document. Using these APIs, you can design completely custom search forms that seamlessly integrate with Oracle AS Portal pages. You can also customize how portal search results get displayed on a page; you are no longer limited to the search results portlet provided with Oracle AS Portal. As the public search APIs can return the results in XML, you may process or format the search results however you require.

You can also use the search APIs to display portal content in any custom application. For example, if you have a Web-based application you could easily provide a search form in the application and display the search results from Oracle AS Portal there as well.

The WWSRC_API package provides the following two types of API:

- Search submission/execution APIs
  - Item search
  - Page search
  - Category search
  - Perspective search

- Search results APIs
  - Get item results as XML document
  - Get page results as XML document

To determine the structure of the search results returned by the search APIs, refer to the appropriate secure content repository view, as shown in Table 13–1.

<table>
<thead>
<tr>
<th>Search API</th>
<th>Secure Content Repository View</th>
</tr>
</thead>
<tbody>
<tr>
<td>wwsrc_api.item_search</td>
<td>WWSBR_ALL_ITEMS</td>
</tr>
<tr>
<td>wwsrc_api.page_search</td>
<td>WWSBR_ALL_FOLDERS</td>
</tr>
<tr>
<td>wwsrc_api.category_search</td>
<td>WWSBR_ALL_CATEGORIES</td>
</tr>
<tr>
<td>wwsrc_api.perspective_search</td>
<td>WWSBR_ALL_PERSPECTIVES</td>
</tr>
</tbody>
</table>
For example, the results returned by the `item_search` API have the same structure as the WWSBR_ALL_ITEMS view. For information about the structure of the secure views, refer to Chapter F, "Content Management APIs and Views".

This chapter contains the following sections:

- Searching For Items Across All Page Groups
- Searching For Pages in Specific Page Groups
- Searching For Items By Attribute
- Transforming Search Results into XML
- Displaying Search Results

For the examples in the following sections, you need to enable output in SQL*Plus. To do this log on to SQL*Plus as the portal schema owner and enter the following command:

```sql
SQL> SET SERVEROUTPUT ON
```

For more information about the public search APIs, refer to the Oracle Application Server Portal PL/SQL API Reference on Portal Center:

http://portalcenter.oracle.com

In the Portal Focus Areas section, click Portlet Development, and then click PL/SQL API Reference (APIs and References section).

**Tip:** Remember, if you are calling the APIs from a Web provider or external application, you need to set the session context first. For more information, refer to Section 10.1, "Setting the Session Context".

### 13.1 Searching For Items Across All Page Groups

Example 13–1 uses the `item_search` API to search all the page groups in your portal for items that contain the term 'portal'.

**Example 13–1 Searching Across All Page Groups (item_search API)**

```sql
declare
    l_results    wwsrc_api.items_result_array_type;
    l_count      number;
    l_scores     wwsrc_api.number_list_type;
begin
    l_results := wwsrc_api.item_search(
        p_mainsearch => 'portal',
        p_out_count  => l_count,
        p_out_scores => l_scores
    );
    dbms_output.put_line('Number of results: ' || l_count);
exception
    ...
end;
```

- **p_mainsearch** is the keyword or term for which you want to search. This may be any value, including an Oracle Text query expression.
- **p_out_count** is the number of search hits.
- **p_out_scores** is an array of search results scores. This is the Oracle Text relevancy score, rating how well each result matches the search term (or any other textual search criteria). The index of the array matches the index of the results array returned by the function.

In Example 13–1, the maximum number of search results that may be returned is determined by the **wwsrc_api** constant **MAX_ROWS** (default is 1000). This avoids the possibility of a search query with many hits taking too long to run if a specific number of rows is not specified. For an example of how to specify the number of rows to return see Example 13–2.

### 13.2 Searching For Pages in Specific Page Groups

**Example 13–2** uses the **page_search** API to search in two specific page groups (MyPageGroup and Shared) for pages that contain the term 'Oracle'. In this example, only the first ten results are returned.

**Example 13–2  Searching in Specific Page Groups (page_search API)**

```sql
declare
    l_results   wwsrc_api.pages_result_array_type;
    l_count     number;
    l_scores    wwsrc_api.number_list_type;
    l_pggroups  wwsrc_api.number_list_type;
begin
    l_pggroups(1) := 0;  -- Page group 0 (shared).
    l_pggroups(2) := 33; -- Page group 33 (mypagegroup).
    l_results := wwsrc_api.page_search(
        p_mainsearch  => 'Oracle',
        p_page_groups => l_pggroups,
        p_rows        => 10,
        p_out_count   => l_count,
        p_out_scores  => l_scores
    );
    dbms_output.put_line('Number of results: ' || l_count);
    exception
        ...;
end;
/
```

- **p_mainsearch** is the keyword or term for which you want to search. This may be any value, including an Oracle Text query expression.

- **p_page_groups** is an array of the IDs of the page groups that you want to search. If you do not include this parameter, the value defaults to **wwsrc_api.EMPTY_NUMBER_LIST** (that is, all page groups).

- **p_rows** is the maximum number of search results to return. This defaults to the **wwsrc_api** constant value **MAX_ROWS**.

- **p_out_count** is the number of search hits.

- **p_out_scores** is an array of search result scores. This is the Oracle Text relevancy score, rating how well each result matches the search term (or any other textual search criteria). The index of the array matches the index of the results array returned by the function.
13.3 Searching For Items By Attribute

Example 13–3 uses the `specify_attributes` and `item_search` APIs to search for all file items created by Joe Bloggs since 01-Jan-2004 that contain the term 'Oracle'.

Example 13–3  Searching Specific Attributes (specify_attributes API)

declare
  l_results       wwsrsrc_api.items_result_array_type;
  l_count         number;
  l_scores        wwsrsrc_api.number_list_type;
  l_attributes    wwsrsrc_runtime_attr_varray;
  l_createdate_attrid wwsbr_api.attributes.id%type;
  l_createdate_caid wwsbr_api.attributes.caid%type;
  l_createdate_type wwsbr_api.attributes.data_type%type;
begin
  -- Build up attribute object with author criteria.
  wwsrsrc_api.specify_attributes(
    p_id => wwsbr_api.ATTRIBUTE_AUTHOR,
    p_siteid => wwsbr_api.SHARED_OBJECTS,
    p_value => 'Joe Bloggs',
    p_operator => wwsrsrc_api.CONTAINS_ALL,
    p_datatype => wwsrsrc_api.DATA_TYPE_TEXT,
    p_in_out_attr_varray => l_attributes
  );
  -- Build up attribute object with create date criteria
  -- using wwsbr_attributes view.
  select id, caid, data_type
  into l_createdate_attrid, l_createdate_caid, l_createdate_type
  from wwsbr_attributes
  where name = 'createdate'
  and rownum = 1; -- Ignore translations.
  wwsrsrc_api.specify_attributes(
    p_id => l_createdate_attrid,
    p_siteid => l_createdate_caid,
    p_value => '01-JAN-2004',
    p_operator => wwsrsrc_api.GREATER_THAN,
    p_datatype => l_createdate_type,
    p_in_out_attr_varray => l_attributes
  );
  -- Perform the search.
  l_results := wwsrsrc_api.item_search(
    p_mainsearch => 'Oracle',
    p_itemtypeid => wwsbr_api.ITEM_TYPE_FILE,
    p_itemtypesiteid => wwsbr_api.SHARED_OBJECTS,
    p_attributes => l_attributes,
    p_out_count => l_count,
    p_out_scores => l_scores
  );
  dbms_output.put_line('Number of results: ' || l_count);
exception
  ...
end;
/

Parameters for specify_attributes

- **p_id** is the ID of the attribute. Use the wwsbr_api constants or query the WWSBR_ATTRIBUTES view to find this ID.
- **p_siteid** is the ID of the page group to which the attribute belongs.
- **p_value** is the attribute value to include as search criteria. This must be a text value.
- **p_operator** is the search operator that you want to use.
- **p_datatype** is the datatype of the attribute value.
- **p_in_out_attr_varray** is the existing varray of attributes (if any).

**Parameters for item_search**

- **p_mainsearch** is the keyword or term for which you want to search. This may be any value, including an Oracle Text query expression.
- **p_itemtypeid** is the ID of the type of item for which you want to search. Use the wwsbr_api constants or query the WWSBR_ITEM_TYPES view to find this ID. Note that this is the actual subtype of the item, not the base type.
- **p_itemtypesiteid** is the ID of the page group to which the item type belongs.
- **p_attributes** is an array of attributes to search together with their operators and values. You can build up the values for this parameter by calling the specify_attributes API.
- **p_out_count** is the number of search hits.
- **p_out_scores** is an array of search result scores. This is the Oracle Text relevancy score, rating how well each result matches the search term (or any other textual search criteria). The index of this array matches the index of the results array returned by the function.

### 13.4 Transforming Search Results into XML

The APIs mentioned so far return search results in an array. To provide you with more flexibility about how to display your search results, OracleAS Portal also provides several APIs that return search results as XML. These APIs return the XML results as a CLOB.

The following sections describe how you can generate a physical XML file from a CLOB produced by the search APIs and store the XML file in a directory on the OracleAS Portal middle tier:

1. **Creating a Directory for the XML File.**
2. **Creating an XML File from a CLOB**
3. **Generating Search Results in XML**

These examples assume that the database is on the same machine as the OracleAS Portal middle tier.

#### 13.4.1 Creating a Directory for the XML File

If you decide to transform your search results into an XML file, you must first perform some set up tasks to define the physical directory in which to write the file.

To create a directory on the OracleAS Portal middle tier:

1. In SQL*Plus, connect as SYSTEM or SYS and enter the following command:
   ```sql
   create directory <dirname> as '<physical_location>';
   ```
   For example:
   ```sql
   create directory RESULTDIR as '/u02/ora/OraHome_101202_
   ```
In this example, we use the htdocs directory of the OracleAS Portal middle tier. This is a good location to use as you can then access the XML results file through HTTP, in the format:


For example:

http://my.portal.com:7778/searchapi/results.xml

You can then use this to specify the XML file as the data source for OmniPortlet. For an example of how to do this, refer to Section 13.5.1, "Displaying XML Search Results in OmniPortlet".

2. Still in SQL*Plus, grant write privileges to the directory from PL/SQL programs. In our example, we would do this by entering the following command:

grant write on directory RESULTDIR to public;

3. Check that you can write files to the new directory by running the following procedure:

```plsql
declare
    v_output_file1 utl_file.file_type;
begin
    v_output_file1 := utl_file.fopen('RESULTDIR', 'NEW.txt', 'a');
    utl_file.put_line(v_output_file1, 'NATURE and Beauty');
    utl_file.fclose_all;
end;
```

PL/SQL procedure successfully completed.

Now, when you go to the directory on the file system, you should see the NEW.txt file.

### 13.4.2 Creating an XML File from a CLOB

When you have specified the location for the XML output, you can write the procedure that creates it. Example 13–4 shows a generic procedure that takes the CLOB produced by the `get_all_items_xml` API and writes an XML file to a specific location. It takes two input parameters: the file name and the CLOB.

**Example 13–4 Writing an XML File to a Specific File System Location**

```plsql
create or replace procedure results_xml_to_file(xml_filename varchar2, result_xml clob) as
    l_amount binary_integer;
    l_offset number := 1;
    l_text varchar2(32767);
    l_file utl_file.file_type;
    l_clob clob;
begin
    l_clob := result_xml;
    l_file := utl_file.fopen(
        location => 'RESULTDIR', -- Directory name is case-sensitive.
        filename => xml_filename,
        open_mode => 'w'
    );
```
l_amount := 32767;
l_offset := 1;
begin
  dbms_lob.open(l_clob, dbms_lob.file_readonly);
  loop
    dbms_lob.read(l_clob, l_amount, l_offset, l_text);
    utl_file.put(
      file => l_file,
      buffer => l_text
    );
    l_offset := l_offset + l_amount;
    exit when l_amount < 32767;
  end loop;
  utl_file.new_line(file => l_file);
  dbms_lob.close(l_clob);
end;
utl_file.fclose(file => l_file);
exception...
end;
/

13.4.3 Generating Search Results in XML

Now, when you use the APIs to produce search results as XML, you can call the procedure in Example 13–4 to write the resulting CLOB to an XML file on the file system. Example 13–5 shows how you might do this.

Example 13–5 Generating Search Results in XML (get_all_items_xml API)

declare
  l_results   wwsrsrc_api.items_result_array_type;
  l_scores    wwsrsrc_api.number_list_type;
  l_count     number;
  l_clob      clob;
begin
  l_results := wwsrsrc_api.item_search(
    p_mainsearch => 'portal',
    p_out_count  => l_count,
    p_out_scores => l_scores
  );
  l_clob := wwsrsrc_api.get_all_items_xml(l_results);
  results_xml_to_file('results12.xml', l_clob);
exception...
end;
/

For information about the item_search API used in Example 13–5, refer to Section 13.1, "Searching For Items Across All Page Groups".

You can use the XML file generated by this API as a data source for OmniPortlet. For an example of how you might do this, refer to Section 13.5.1, "Displaying XML Search Results in OmniPortlet".

13.4.4 Workaround for get_item_xml

When you use get_item_xml to transform search results to XML, the API produces a <xml version="1.0" ?> element for each search result. This produces invalid
XML. Example 13–6 shows a workaround that removes these tags and also adds opening and closing <ResultSet> tags to produce an XML file with the ROWSET/ROW structure. If you want to use the XML file as a data source for OmniPortlet, you should do something similar.

---

**Note:** Example 13–6 loops through the search results and writes them all to XML (producing the same results as using get_all_items_xml). Typically, you would use get_item_xml to filter the search results somehow.

---

**Example 13–6  Using get_item_xml to Write Search Results to a File**

```plsql
declare
  l_amount    binary_integer;
  l_offset    number       := 1;
  l_text      varchar2(32767);
  l_file      utl_file.file_type;
  l_clob      clob;
  l_results   wwsrc_api.items_result_array_type;
  l_scores    wwsrc_api.number_list_type;
  l_count     number;
begin
  l_results := wwsrc_api.item_search(
    p_mainsearch => 'portal',
    p_out_count  => l_count,
    p_out_scores => l_scores
  );
  l_file := utl_file.fopen(
    location  => 'RESULTDIR', -- Directory name is case-sensitive.
    filename  => 'results14.xml',
    open_mode => 'w'
  );
  utl_file.put(
    file => l_file,
    buffer => '<ResultSet>
  );
  for i in 1..l_results.count loop
    l_amount := 32767;
    l_offset := 1;
    begin
      l_clob := wwsrc_api.get_item_xml(l_results(i));
      dbms_lob.open(l_clob, dbms_lob.file_readonly);
      loop
        dbms_lob.read(l_clob, l_amount, l_offset, l_text);
        -- Workaround for XML generated with get_item_xml.
        if instr(l_text, '?>') != 0 then
          l_text := substr(l_text, instr(l_text, '?>') + 2);
        end if;
        -- End of workaround.
        utl_file.put(
          file => l_file,
          buffer => l_text
        );
        l_offset := l_offset + l_amount;
      exit when l_amount < 32767;
    end loop;
    utl_file.new_line(file => l_file);
    dbms_lob.close(l_clob);
  end loop;
end;
```
end loop;
util_file.put(
    file => l_file,
    buffer => '</ResultSet>'
);
util_file.fclose(file => l_file);
except
...
end;
/

13.5 Displaying Search Results

For your search query to be useful, you need to display the results somewhere. You can choose to display the results of the search directly from the array, for example, in a dynamic page. For more flexibility, you can transform the search results into XML first and then display the XML, for example in OmniPortlet.

13.5.1 Displaying XML Search Results in OmniPortlet

The following steps provide an example of how to use the XML generated by the search APIs as the data source for OmniPortlet. For example, you could schedule a job to regularly execute an API to produce search results as XML and then automatically display those results in an OmniPortlet.

For more information about OmniPortlet, refer to the Oracle Application Server Portal Developer’s Guide.

---

**Note:** The XML produced by the search APIs is returned in a CLOB. Therefore, if you want to use the XML as a data source for OmniPortlet, you must first write the CLOB to a file. For an example of how to do this, refer to Section 13.4, "Transforming Search Results into XML".

---

1. First you need to add a new OmniPortlet instance to a page.

   **Tip:** OmniPortlet is usually available in the Portlet Builders area of the Portlet Repository.

2. Click the **Define** link to launch the OmniPortlet Wizard.

3. On the Type page of the wizard, select **XML**, then click **Next**.

4. On the Source page, in the **XML URL** field, enter the URL of your XML file, for example:

   http://my.portal.com:7778/searchapi/results.xml

   Because the XML data produced by the search APIs uses a ROWSET/ROW structure, you do not need to specify an XSL filter.

5. Click **Next**.

6. On the View page, select **Tabular** for the **Layout Style**, then click **Next**.

7. On the Layout page, choose the columns you want to display in the portlet and how you want them to appear.
8. Click **Finish** to create the OmniPortlet instance.

### 13.5.2 Displaying Search Results in a Dynamic Page

The following two examples show how to display search results in a dynamic page portlet. Users can then add this portlet to any page within your portal.

1. Create a procedure that the dynamic page can call to perform the search.

   The procedure in Example 13–7 uses the `item_search` API to search for a specified term (searchterm) and displays only those results with a score higher than a specified value (score).

   **Note:** You must create the procedure in the database in which your portal resides.

   **Example 13–7  Procedure to Perform the Search**

   ```sql
   create or replace procedure search_results(searchterm varchar2, score number) as
   x varchar2(100);
   y number;
   l_results wwsrc_api.items_result_array_type;
   l_count number;
   l_scores wwsrc_api.number_list_type;
   begin
   x := searchterm;
   y := score;
   l_results := wwsrc_api.item_search(
       p_mainsearch => x,
       p_out_count  => l_count,
       p_out_scores => l_scores );
   htp.p('Number of total hits: ' || l_count);
   htp.p('<br>');
   htp.p('<br>');
   for i in 1..l_results.count loop
     if (l_scores(i) > y) then
       htp.p('<b>' || i || '</b> - <a href="" || l_results(i).url || '">' || l_results(i).display_name || '</a>);
       htp.p('<br>');
       htp.p('score = ' || l_scores(i));
       htp.p('<br>');
     end if;
   end loop;
   exception
     ...;
   end;
   / 
   grant execute on search_results to public;
   ```

   For information about the `item_search` API used in Example 13–7, refer to Section 13.1, "Searching For Items Across All Page Groups".

2. Create the dynamic page.

   A dynamic page is one of the portlets you can build using the OracleAS Portal Portlet Builder. For more information about Portlet Builder, refer to the Oracle Application Server Portal Developer’s Guide. For more information about building dynamic pages, refer to the OracleAS Portal online Help.
Example 13–8 shows the code to use for the dynamic page.

Tip: Replace $<\text{schema}>$ with the name of the schema in which you created your procedure.

**Example 13–8  Code for Dynamic Page to Display Results**

```html
<html>
<head>
<title>Example for Search API UI</title>
</head>
<body>
<h2>My Search Form</h2>
<oracle>
declare
  x varchar2(100);
  y number;
begin
  x := :searchterm;
  y := :score;
  $<\text{schema}>$.search_results(x,y);
end;
</oracle>
</body>
</html>
```

3. Provide default values for the two bind variables defined in the HTML code so that the dynamic page displays some results (Figure 13–1).

**Figure 13–1  Providing Default Values for Bind Variables**

4. Make sure the dynamic page is available as a portlet.

5. You can now add this dynamic page portlet to any page in your portal.

Tip: You should find the portlet in the Staging Area of the Portlet Repository under the name of the provider in which you created it.

The dynamic page portlet displays search results based on the default values provided when you created the dynamic page (Figure 13–2).
6. Users can personalize this portlet to provide their own search term and minimum score (Figure 13–3).

In Figure 13–4 you can see the changes in the search results when a user personalizes the portlet and changes the minimum score to 70.
This chapter describes how to use the APIs provided with OracleAS Portal to create content in different languages. It contains the following sections:

- Introduction to Multi-Lingual Support
- Querying the Default Language
- Setting the Session Language
- Modifying an Existing Translation
- Creating a Translation for an Item
- Translations and Item Versioning

For more information about any of these APIs, refer to the Oracle Application Server Portal PL/SQL API Reference on Portal Center:

http://portalcenter.oracle.com

In the Portal Focus Areas section, click Portlet Development, then in the APIs and References section, click PL/SQL API Reference.

Tip: Remember, if you are calling the APIs from a Web provider or external application, you need to set the session context first. For more information, refer to Section 10.1, "Setting the Session Context".

14.1 Introduction to Multi-Lingual Support

The page group administrator must have configured the page group to support multiple languages before you can use the APIs to create translations of portal objects. Before using the APIs, it is recommended that you read Chapter 20 "Translating Portal Content" of the Oracle Application Server Portal User’s Guide to understand how to configure a page group and how objects with translations behave.

The basic approach to creating or modifying a translation of a portal object is to set the language context and then use the appropriate content management API to create or modify the item.

14.2 Querying the Default Language

To determine the default language of a page group, run the following query:

```sql
select distinct wnnls_api.get_language(default_language) "Page Group Default Language"
from wwsbr_all_content_areas pg
where name = 'MyPageGroup';
```
To determine the language context of a SQL*Plus session, run the following query:

```sql
select wwctx_api.get_nls_language "Session Context Language Code",
       wwnls_api.get_language(wwctx_api.get_nls_language) "Language Description"
from dual;
```

**Note:** When you log in to SQL*Plus and set your user context, you are initially in the language specified by your NLS_LANG environment variable.

### 14.3 Setting the Session Language

Before creating content in a different language, use the `wwctx_api.set_nls_language` API to set the session language, as shown in Example 14–1.

**Example 14–1 Setting the Session Language (set_nls_language API)**

```sql
begin
    wwctx_api.set_nls_language(
        p_nls_language => wwnls_api.GREEK,
        p_nls_territory => wwnls_api.TER_GREECE
    );
exception
    ...
end;
/
```

- **p_nls_language** is the Globalization Support language abbreviation of the language that you want to set for the session.
- **p_nls_territory** is the Globalization Support territory abbreviation of the territory that you want to set for the session.

For a list of available language and territory constants, refer to the documentation for the WWNLS_API package in the *Oracle Application Server Portal PL/SQL API Reference* on Portal Center:

[http://portalcenter.oracle.com](http://portalcenter.oracle.com)

In the Portal Focus Areas section, click Portlet Development, then in the APIs and References section, click PL/SQL API Reference.

### 14.4 Modifying an Existing Translation

Modifying an existing translation is simply a matter of setting the language context and using the appropriate API to modify the object. Example 14–2 modifies the text of an existing text item that has a translation in French.

**Example 14–2 Modifying a Translation**

```sql
declare
    l_item_master_id number;
begin
    -- Set the language context.
    wwctx_api.set_nls_language(
        p_nls_language => wwnls_api.FRENCH
    );
    -- Edit the item (this edits the French translation).
```
Creating a Translation for an Item

If the session is using a different language from the default language for the page group and translations for the session language are enabled for the page group, the add_item API creates a translation for an item when the item is created. For example, if the default language for the page group is English, the session language is French, and French translations are enabled for the page group, then the API adds two rows to the item table, as follows:

- One with a value of 'us' (English) for the default language.
- One with a value of 'f' (French) for the session language.

Other than the language, both rows will have identical values for all attributes, including the item ID. To avoid this, it is recommended that you always add an item in the default language first before creating any of its translations.

In similar circumstances, the modify_item API updates the translation for the session language (if it exists). If the translation does not exist, as long as translations are enabled for the session language, the API creates a new translation. Therefore, if you want to add a translation for an existing item, use the modify_item (or modify_item_post_upload) API. Example 14–3 creates a file item then changes the language context and adds a translation for the item.

Example 14–3  Creating a Translation

```sql
declare
  l_item_master_id  number;
  l_item_masterthing_id2 number;
  l_item_id              number;
begin
  -- Set the language context.
  wwctx_api.set_nls_language(
    p_nls_language => wwnls_api.AMERICAN
  );
  -- Add an item that resides on the same server as the portal content repository
  -- database.
  l_item_masterthing_id := wwsbr_api.add_item(
    p_caid     => 53,  -- A known page group ID.
    p_folder_id => 1,  -- A known page ID.
    p_display_name => 'My File',
    p_type_id    => wwsbr_api.ITEM_TYPE_FILE,
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

For information about the modify_item API used in Example 14–2, refer to Section 11.2.2, "Editing an Item".

14.5 Creating a Translation for an Item

For information about the modify_item API used in Example 14–2, refer to Section 11.2.2, "Editing an Item".

```sql
l_item_master_id := wwsbr_api.modify_item(
  p_master_item_id => 453,
  p_item_id        => 454,
  p_caid           => 33,
  p_folder_id      => 45,
  p_display_name   => 'Revue de Film');
  p_text           => 'C’est le texte de la revue');
);
-- Process cache invalidation messages.
wwpro_api_invalidation.execute_cache_invalidation;
```
14.6 Translations and Item Versioning

If you attempt to create a new version of an item in a nondefault language, by setting `p_addnewversion` to TRUE, a new version is not created; a new translation is added instead.

When using versioning in conjunction with translations, we recommend that you create new versions of items in the default language and then translate the new version of the item.

For a more detailed discussion of the effect of item versioning on translations, refer to Section 20.3 “Creating Translatable Content” in the Oracle Application Server Portal User’s Guide.
This chapter describes how to use the APIs provided with OracleAS Portal to ensure that your content is secure. It contains the following sections:

- Retrieving Object Privileges
- Setting Page Level Privileges
- Setting Item Level Privileges

For more information about any of these APIs, refer to the Oracle Application Server Portal PL/SQL API Reference on Portal Center:

http://portalcenter.oracle.com

In the Portal Focus Areas section, click Portlet Development, then in the APIs and References section, click PL/SQL API Reference.

**Tip**: Remember, if you are calling the APIs from a Web provider or external application, you need to set the session context first. For more information, refer to Section 10.1, "Setting the Session Context".

### 15.1 Retrieving Object Privileges

To retrieve a list of privileges that are currently defined for an object, use the `wwsec_api.grantee_list` API. Example 15–1 prints the values of the grantee array for a page with an ID of 17623 in page group with an ID of 33. Example 15–2 prints the values of the grantee array for an item with a master ID of 32919 in a page group with an ID of 53.

**Example 15–1  Retrieving the List of Privileges for a Page (grantee_list API)**

```sql
declare
  l_grantees         wwsec_api.grantee_array;
  l_object_type_name varchar2(5)  := wwsec_api.PAGE_OBJ;
  l_name             varchar2(60) := '33/17623';
begin
  -- Call the function.
  l_grantees := wwsec_api.grantee_list(
    p_object_typ_name => l_object_type_name,
    p_name            => l_name
  );
  -- Output the results
  if l_grantees is not null then
    if l_grantees.count > 0 then
      for i in l_grantees.first..l_grantees.last loop
        if l_grantees.exists(i) then
```
Example 15–2 Retrieving the List of Privileges for an Item

declare
  l_grantees         wwsec_api.grantee_array;
  p_object_type_name varchar2(5)  := wwsec_api.ITEM_OBJ;
  p_name             varchar2(60) := '53/32919';
beginn
-- Call the function.
l_grantees := wwsec_api.grantee_list(p_object_type_name, p_name);
-- Output the results.
if l_grantees is not null then
  if l_grantees.count > 0 then
    for i in l_grantees.first..l_grantees.last loop
      if l_grantees.exists(i) then
        dbms_output.put.line('GRANTEE_TYPE '||to_char(i)||'= '||l_grantees(i).GRANTEE_TYPE);
        dbms_output.put.line('GRANTEE_ID '||to_char(i)||'= '||l_grantees(i).GRANTEE_ID);
        dbms_output.put.line('GRANTEE_NAME '||to_char(i)||'= '||l_grantees(i).GRANTEE_NAME);
        dbms_output.put.line('PRIVILEGE '||to_char(i)||'= '||l_grantees(i).PRIVILEGE);
      end if;
    end loop;
  end if;
end if;
end if;
exception
  ...
end; /

The grantee_list API takes the following three parameters:

- **p_object_type_name** is the type of the object. Use the predefined constants in the WWSEC_API package to specify the value of this parameter, for example wwsec_api.PAGE_OBJ or wwsec_api.ITEM_OBJ.
- **p_name** is the reference to the object. Use the format '<page group ID>/<object ID>'. So for items, use '<page group ID>/<master item ID>', for example '53/32919'.
- **p_owner** is the name of the schema that owns the object. For items, do not pass a value to this parameter as it defaults to the portal schema owner.
The API returns an array (WWSEC_API.GRANTEE_ARRAY) with the following columns:

- **grantee_type** is either USER or GROUP
- **grantee_id** is the unique ID of the user or group
- **grantee_name** is the user name or group name
- **privilege** is the privilege granted to the user or group

### 15.2 Setting Page Level Privileges

This section shows how to use APIs in the WWSEC_API package to set page level privileges.

---

**Note:** You can also use the APIs listed in the following sections to set tab level access by using the following format for the *p_name* parameter:

<p>page group ID>/tab ID</p>

You do not need to specify the ID of the container page, as the tab ID is enough to uniquely identify the tab within the page group.

---

### 15.2.1 Granting Page Level Privileges

**Example 15–3** shows how to use the *set_group_acl* API to grant privileges to a group. **Example 15–4** shows how to use the *set_user_acl* API to grant privileges to a user.

**Example 15–3  Granting Page Privileges to a Group (set_group_acl API)**

```sql
declare
  l_group_id number       := wwsec_api.group_id('MYGROUP');
  l_name     varchar2(60) := '33/17623';
BEGIN
  wwsec_api.set_group_acl(
    p_group_id         => l_group_id,
    p_object_type_name => wwsec_api.PAGE_OBJ,
    p_name             => l_name,
    p_privilege        => wwsec_api.VIEW_PRIV
  );
end;
/
```

**Example 15–4  Granting Page Privileges to a User (set_user_acl API)**

```sql
declare
  l_person_id number      := wwsec_api.id('JOHN.SMITH');
  l_name     varchar2(60) := '33/17623';
begin
  wwsec_api.set_user_acl(
    p_person_id        => l_person_id,
    p_object_type_name => wwsec_api.PAGE_OBJ,
    p_name             => l_name,
    p_privilege        => wwsec_api.VIEW_PRIV
  );
end;
/
These two APIs take the following parameters:

- **p_group_id** is the ID of the group to which you want to grant privileges (set_group_acl only)
- **p_person_id** is the ID of the user to whom you want to grant privileges (set_user_acl only)
- **p_object_type_name** is type of the object on which you want to grant privileges. Use the predefined constants in the WWSEC_API package to specify the value of this parameter, for example wwsec_api.PAGE_OBJ.
- **p_name** is the reference to the object. Use the format '<page group ID>/<page ID>', for example '33/17623'.
- **p_privilege** is the level of privilege you want to grant to the user or group. Use the predefined constants in the WWSEC_API package to specify the value of this parameter, for example wwsec_api.VIEW_PRIV.

### 15.2.2 Removing Page Level Privileges

At some point, it may become necessary to remove a user or group’s privileges from a page. Example 15–5 shows how to use the remove_group_acl API to remove a group’s privileges. Example 15–6 shows how to use the remove_user_acl API to remove a user's privileges.

**Example 15–5  Removing Page Privileges from a Group (remove_group_acl API)**

```sql
declare
  l_group_id number       := wwsec_api.group_id('MYGROUP');
  l_name     varchar2(60) := '33/17623';
BEGIN
  wwsec_api.remove_group_acl(
    p_object_type_name => wwsec_api.PAGE_OBJ,
    p_name             => l_name,
    p_group_id        => l_group_id,
    p_privilege        => wwsec_api.MANAGE_PRIV
  );
end;
/
```

**Example 15–6  Removing Page Privileges from a User (remove_user_acl API)**

```sql
declare
  l_person_id number       := wwsec_api.id('JOHN.SMITH');
  l_name     varchar2(60)  := '33/17623';
BEGIN
  wwsec_api.remove_user_acl(
    p_object_type_name => wwsec_api.GROUP_OBJ,
    p_name             => l_name,
    p_person_id        => l_person_id,
    p_privilege        => wwsec_api.MANAGE_PRIV
  );
end;
/
```

These two APIs take the following parameters:
Setting Item Level Privileges

- **p_object_type_name** is type of the object from which you want to remove privileges. Use the predefined constants in the WWSEC_API package to specify the value of this parameter, for example wwsec_api.PAGE_OBJ.

- **p_name** is the reference to the object. Use the format '<page group ID>/<page ID>', for example '33/17623'.

- **p_group_id** is the ID of the group whose privileges you want to remove (remove_group_acl only). Set this parameter to NULL if you want to remove the specified privilege on this page from all groups.

- **p_person_id** is the ID of the user whose privileges you want to remove (remove_user_acl only). Set this parameter to NULL if you want to remove the specified privilege on this page from all users.

- **p_privilege** is the level of privilege you want to remove from the user or group. Use the predefined constants in the WWSEC_API package to specify the value of this parameter, for example wwsec_api.VIEW_PRIV. Set this parameter to NULL if you want to remove all privileges on the page from the user or group.

15.3 Setting Item Level Privileges

If item level security (ILS) is enabled for a page, you can specify access privileges for individual items on the page.

**Example 15–7** shows how to use the modify_folder API to enable ILS for a page.

**Example 15–7  Enabling Item Level Security for a Page**

```sql
declare
    l_page     wwsbr_api.page_record;
begin
    select *
    into l_page
    from <schema>.wwsbr_user_pages
    where siteid = 33
        and id = 1
        and rownum = 1;
    l_page.haveitemsecurity := 1;
    wwsbr_api.modify_folder(
        p_page => l_page
    );
    -- Process cache invalidation messages.
    wwpro_api_invalidation.execute_cache_invalidation;
exception
    ...
end;
/
```

For information about the modify_folder API and WWSBR_USER_PAGES view used in **Example 15–7**, refer to Section 11.1, "Editing Page Properties".

**Example 15–8** shows how to enable ILS for an individual item (this is the same as selecting Define Item Level Access Privileges in the OracleAS Portal user interface):

**Example 15–8  Enabling Item Level Security for an Item (enable_ils_for_item API)**

```sql
begin
    wwsbr_api.enable_ils_for_item(
        p_master_item_id => 453,
        p_caid           => 33,
        ...
    );
end;
```

For information about the enable_ils_for_item API, refer to Section 11.1, "Editing Page Properties".
### 15.3.1 Granting Item Level Privileges

After enabling ILS for the item, you can define access privileges for one or more users or groups.

When setting item level privileges, the type of privileges that are granted is dependent on which of the following parameters are passed rather than the parameter values:

- Pass an array of user IDs to `p_itemown_user` to grant the *Manage* privilege to a list of users.
- Pass an array of user IDs to `p_itemmanage_user` to grant the *Edit* privilege to a list of users.
- Pass an array of user IDs to `p_itemview_user` to grant the *View* privilege to a list of users.
- Pass an array of group IDs to `p_itemown_group` to grant the *Manage* privilege to a list of groups.
- Pass an array of group IDs to `p_itemmanage_group` to grant the *Edit* privilege to a list of groups.
- Pass an array of group IDs to `p_itemview_group` to grant the *View* privilege to a list of groups.

You can pass values to any combination of these parameters in the same procedure call to set a range of privileges across different users and groups. Example 15–9 shows an example for granting item level privileges to users.

**Example 15–9  Granting Item Level Privileges to Users (add_item_ils_privileges API)**

```sql
declare
  l_itemown_username_array  wwsbr_type.array;
  l_itemown_userid_array    wwsbr_type.array;
begin
  l_itemown_username_array(1) := 'jsmith';
  l_itemown_username_array(2) := 'janesmith';
  l_itemown_username_array(3) := 'joedoe';
  for i in 1 .. l_itemown_username_array.count loop
```
-- Get the user ID from the wwsec_api.id_sso API.
l_itemown_userid_array(i) := wwsec_api.id_sso(
    p_username => l_itemown_username_array(i)
);
end loop;
wwsbr_api.add_item_ils_privileges(
    p_master_item_id => 453,
    p_caid           => 33,
    p_folder_id      => 45,
    p_itemown_user   => l_itemown_userid_array
);
-- Process cache invalidation messages.
wwpro_api_invalidation.execute_cache_invalidation;
exception
...
end;
/

■ **p_master_item_id** is the master ID of the item. You can find this value in the
MASTERID column of the WWSBR_ALL_ITEMS view.

■ **p_caid** is the ID of the page group to which the item belongs.

■ **p_folder_id** is the ID of the page on which the item appears.

■ **p_itemown_user** is an array of user IDs to which you want to grant *Manage*
privileges.

If you pass the same user or group in more than one of the privilege arrays, the user or
group is granted the highest privilege level specified.

**Tip:** You can also update the access privileges for an item using the
add_item_ils_privileges API.

### 15.3.2 Removing Item Level Privileges

If for some reason, you need to remove a user or group’s privileges on an item, use the
delete_ils_privilege API (Example 15–10).

**Example 15–10  Removing Item Level Privileges (delete_ils_privilege API)**

```sql
declare
    l_user_id       number := 334;
    l_page_group_id number := 75;
    l_page_id       number := 1;
    l_item_id       number := 74637;
begin
    wwsbr_api.delete_ils_privilege(
        p_user_or_group_id => l_user_id,
        p_caid             => l_page_group_id,
        p_folder_id        => l_page_id,
        p_master_item_id   => l_item_id
    );
    -- Process cache invalidation messages.
    wwpro_api_invalidation.execute_cache_invalidation;
exception
...
end;
/
```

Tip: You can also update the access privileges for an item using the
add_item_ils_privileges API.
Setting Item Level Privileges

- **p_user_or_group_id** is the ID of the user or group for which you want to remove privileges.
- **p_caid** is the ID of the page group to which the item belongs.
- **p_folder_id** is the ID of the page on which the item appears.
- **p_master_item_id** is the master ID of the item. You can find this value in the MASTERID column of the WWSBR_ALL_ITEMS view.

### 15.3.3 Inheriting Item Level Privileges from the Page

If you decide that an item that has its own privileges defined should, in fact, simply inherit its privileges from the page instead, use the `inherit_folder_privileges` API to disable ILS (this is the same as selecting *Inherit Parent Page Access Privileges* in the OracleAS Portal user interface) as shown in Example 15–11.

**Example 15–11 Inheriting Item Privileges from the Parent Page (inherit_folder_privileges API)**

```sql
begin
  wwsbr_api.inherit_folder_privileges(
    p_master_item_id => 453,
    p_caid           => 33,
    p_folder_id      => 45
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
```

- **p_master_item_id** is the master ID of the item. You can find this value in the MASTERID column of the WWSBR_ALL_ITEMS view.
- **p_caid** is the ID of the page group to which the item belongs.
- **p_folder_id** is the ID of the page on which the item appears.

**Tip:** If you want to edit other attributes for the item as well as the ILS setting, you can use the `modify_item` or `modify_item_post_upload` APIs instead. To enable ILS set the `p_access_level` parameter to `wwsbr_api.ITEM_ACCESS`, to disable ILS set the parameter to `wwsbr_api.FOLDER_ACCESS`. 
This chapter describes the OracleAS Portal Content Management Event Framework (CMEF). It contains the following sections:

- What Is the Content Management Event Framework?
- How Does the Content Management Event Framework Work?
- Using the Content Management Event Framework
- Installing the Examples
- Example: Portal Object Event Logging
- Example: Item Notification
- Example: Item Validation
- Example: Integrating External Workflow

16.1 What Is the Content Management Event Framework?

CMEF enables you to extend OracleAS Portal’s content management functionality by adding programmatic hooks to certain pre-defined portal events. The framework publishes these events to an Oracle database queue. This allows third party programs to subscribe to these events and to use the APIs to extend your portal. In this way, you can use page and item related events within a portal to trigger actions within one or more external applications.

16.2 How Does the Content Management Event Framework Work?

CMEF uses Oracle Streams Advanced Queuing (AQ) technology. Oracle Streams AQ is an Oracle database component that provides a message-queue system with a rich and industry standard feature set. Oracle Streams AQ offers:

- Multiple ways for applications (producers) to place messages in a queue (enqueue).
- Multiple ways for applications (consumers) to get messages from a queue (dequeue).
- A publish/subscribe model that enables the producer application to be independent of the consumer applications.
- Propagation of messages between queues on different machines and databases.
16-2
Oracle Application Server Portal Developer's Guide

Guaranteed delivery of messages along with exception handling in case messages cannot be delivered.

Persistent storage of messages.

Message prioritization.

Time properties for messages such as expiration and delays.

For more information, refer to the Oracle Streams Advanced Queuing page on the Oracle Technology Network (OTN):
http://www.oracle.com/technology/products/aq

Actions within a portal (through the OracleAS Portal user interface, the Web-based Distributed Authoring and Versioning (WebDAV) protocol, or PL/SQL APIs) trigger CMEF events. These events cause OracleAS Portal to publish CMEF messages to a queue. You can create subscribers to consume the messages on the queue and perform actions based on them. For example, you could create a subscriber to verify that when a user adds an item to a portal page, its display name is less than 80 characters. The subscriber may process events as they occur or process them based on some time interval.

There are three major types of CMEF events: ADD, UPDATE, and DELETE. Every content management action falls under one of these events. An event can be in one of several different states. For example, when a user adds an item to a page, the state of the ADD event indicates whether the item is available immediately (PUBLISHED) or at some later date (NOT_PUBLISHED). Refer to Section 16.3, "Using the Content Management Event Framework" to learn more about how subscribers can use this information to retrieve messages of interest to them.

Using the events published by CMEF involves the following five basic queuing operations:

- **Enqueuing Messages**: OracleAS Portal publishes messages (known as enqueuing) to the multiconsumer queue named WWSBR_EVENT_Q. Messages to this queue never expire. For more information, refer to Section 16.2.1, "Enqueuing Messages".

- **Dequeuing Messages**: Subscribing applications pick up messages (known as dequeuing) from the queue. For more information, refer to Section 16.2.2, "Subscribers and Dequeuing Messages".

- **Exception Handling**: A message is said to be processed normally if it is consumed within the specified time interval and within the specified number of attempts. Messages not consumed normally are placed in a separate queue called the exception queue. For more information, refer to Section 16.2.3, "Exception Handling".

- **Listening for Messages**: An application can use LISTEN to wait for messages for multiple subscriptions without having to repeatedly poll the queue. For more information, refer to Section 16.2.4, "Listening for Messages".

- **Notifications**: This Oracle Streams AQ feature enables users or clients to receive notification of a message of interest. For more information, refer to the Oracle Streams AQ documentation.

You use the DBMS_AQ package to perform queuing operations.

### 16.2.1 Enqueuing Messages

CMEF enqueues messages to the WWSBR_EVENT_Q queue, specifying AQ message properties that subscriber applications can then use at dequeue time (Table 16–1).
How Does the Content Management Event Framework Work?

The WWSBR_EVENT_Q queue is a multiconsumer queue. This enables more than one subscriber to consume a single message, allowing for multiple subscribers without having to make multiple copies of each message. Since all messages in the queue have the same priority, this queue works on a first-in first-out basis.

Each message that CMEF enqueues to the WWSBR_EVENT_Q queue contains a payload that contains information about the portal object to which the event relates, such as page ID or page group ID. Subscribers can use this payload information to perform actions on the portal object. The contents of the CMEF message payload are described in more detail in Section 16.3.6, "CMEF Message Payload".

CMEF enqueues messages for immediate consumption, that is, a message is posted on the WWSBR_EVENT_Q queue as soon as an action occurs in the portal; there is no delay. Messages created by CMEF do not have an expiration time.

### 16.2.2 Subscribers and Dequeuing Messages

On the consuming end, subscribers have various mechanisms to consume the messages produced by CMEF. Subscribers can process messages as they arrive, and thus must wait for the messages to arrive. Alternatively, subscribers can choose to be notified when the messages arrive. These notifications can be Oracle Call Interface (OCI) callback functions, PL/SQL functions, or even e-mails.

#### 16.2.2.1 Adding a Subscriber to the WWSBR_EVENT_Q Queue

For a subscriber to be able to consume messages produced by CMEF, you need to add it to the WWSBR_EVENT_Q queue using the DBMS_AQADM.ADD_SUBSCRIBER procedure. Example 16–1 adds the JAY subscriber to the WWSBR_EVENT_Q queue.

**Example 16–1 Adding a Subscriber to WWSBR_EVENT_Q**

```sql
subscriber := sys.aq$_agent('JAY', null, null);
dbms_aqadm.add_subscriber(  
    queue_name => 'portal.wwsbr_event_q',  
    subscriber => subscriber  
);
```

You can remove a subscriber using the DBMS_AQADM.REMOVE_SUBSCRIBER procedure.

<table>
<thead>
<tr>
<th>Message Property</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIORITY</td>
<td>BINARY_INTEGER</td>
<td>1</td>
</tr>
<tr>
<td>DELAY</td>
<td>BINARY_INTEGER</td>
<td>NO_DELAY</td>
</tr>
<tr>
<td>EXPIRATION</td>
<td>BINARY_INTEGER</td>
<td>NEVER</td>
</tr>
<tr>
<td>CORRELATION</td>
<td>VARCHAR2(128)</td>
<td>NULL</td>
</tr>
<tr>
<td>RECIPIENT_LIST</td>
<td>AQ$_RECIPIENT_LIST_T</td>
<td>NULL</td>
</tr>
<tr>
<td>EXCEPTION_QUEUE</td>
<td>VARCHAR2(51)</td>
<td>WWSBR_EVENT_ERR_Q</td>
</tr>
<tr>
<td>ORIGINAL_MSGID</td>
<td>RAW(16)</td>
<td>NULL</td>
</tr>
</tbody>
</table>

The WWSBR_EVENT_Q queue is a multiconsumer queue. This enables more than one subscriber to consume a single message, allowing for multiple subscribers without having to make multiple copies of each message. Since all messages in the queue have the same priority, this queue works on a first-in first-out basis.

Each message that CMEF enqueues to the WWSBR_EVENT_Q queue contains a payload that contains information about the portal object to which the event relates, such as page ID or page group ID. Subscribers can use this payload information to perform actions on the portal object. The contents of the CMEF message payload are described in more detail in Section 16.3.6, "CMEF Message Payload".

CMEF enqueues messages for immediate consumption, that is, a message is posted on the WWSBR_EVENT_Q queue as soon as an action occurs in the portal; there is no delay. Messages created by CMEF do not have an expiration time.

### 16.2.2 Subscribers and Dequeuing Messages

On the consuming end, subscribers have various mechanisms to consume the messages produced by CMEF. Subscribers can process messages as they arrive, and thus must wait for the messages to arrive. Alternatively, subscribers can choose to be notified when the messages arrive. These notifications can be Oracle Call Interface (OCI) callback functions, PL/SQL functions, or even e-mails.

#### 16.2.2.1 Adding a Subscriber to the WWSBR_EVENT_Q Queue

For a subscriber to be able to consume messages produced by CMEF, you need to add it to the WWSBR_EVENT_Q queue using the DBMS_AQADM.ADD_SUBSCRIBER procedure. Example 16–1 adds the JAY subscriber to the WWSBR_EVENT_Q queue.

**Example 16–1 Adding a Subscriber to WWSBR_EVENT_Q**

```sql
subscriber := sys.aq$_agent('JAY', null, null);
dbms_aqadm.add_subscriber(  
    queue_name => 'portal.wwsbr_event_q',  
    subscriber => subscriber  
);
```

Since Oracle Streams AQ supports a maximum of 1024 subscribers for each multiconsumer queue, you can add up to a maximum of 1024 subscribers to the WWSBR_EVENT_Q queue. All consumers that are added as subscribers to this queue must have unique values for the AQ$_AGENT parameter.

You can remove a subscriber using the DBMS_AQADM.REMOVE_SUBSCRIBER procedure.
For an example of a simple subscriber, refer to Section 16.3.1, "Creating Subscriber Code”.

16.2.2.2 Subscriber Queue Management
Oracle Enterprise Manager DBA Studio enables you to manage Oracle Streams AQ. You can use DBA Studio to create queue tables, create queues, browse AQ messages, archive or purge AQ messages, add AQ subscribers, and manage propagation. DBA Studio also shows the topology for the propagation of messages between queues at database level and queue level.

The Oracle Diagnostics and Tuning pack supports alerts and monitoring for AQ queues. You can set up alerts for when the number of messages for a particular subscriber exceeds a threshold, or when there is an error in propagation. In addition, you can monitor queues for the number of messages in a ready state or the number of messages for each subscriber, and so on.

You can also manage the subscriber queue using the standard AQ APIs. For more information, refer to your Oracle Enterprise Manager documentation.

16.2.2.3 Dequeuing Messages
The operation of retrieving messages from a queue is known as dequeuing (Figure 16–1).

![Figure 16–1 The Dequeuing Process](image)

You use the DBMS_AQ.DEQUEUE procedure for dequeuing messages from the WWSBR_EVENT_Q queue. Example 16–2 illustrates dequeuing for the subscriber JAY.

**Example 16–2 Dequeuing Messages**

```sql
...
dequeue_options.wait := dbms_aq.NO_WAIT;
dequeue_options.consumer_name := 'JAY';
dequeue_options.navigation := dbms_aq.FIRST_MESSAGE;
dequeue_options.dequeue_mode := dbms_aq.BROWSE;
```
dbms_aq.dequeue(
    queue_name => 'WWSBR_EVENT_Q',
    dequeue_options => dequeue_options,
    message_properties => message_properties,
    payload => message,
    msgid => message_handle
);
...

- **NAVIGATION:** Use the NAVIGATION parameter of the DBMS_AQ.DEQUEUE operation to determine the sequence in which you want to dequeue the messages. The default NAVIGATION parameter for the dequeue request is NEXT_MESSAGE. This means that the subsequent dequeue operation will retrieve the messages from the queue based on the snapshot obtained in the first dequeue. In particular, a message that is enqueued after the dequeue command will be processed only after processing all the messages already enqueued before in the queue. This is sufficient for messages enqueued for the WWSBR_EVENT_Q queue since it does not have priority-based ordering.

    __Note__: NEXT_MESSAGE with some delay is the optimal way of processing AQ messages. When the first message in the queue needs to be processed by every dequeue command, subscribers must explicitly use the FIRST_MESSAGE navigation option.

- **DEQUEUE_MODE:** A dequeue request can either view a message or delete a message. To view a message, the subscriber can use the BROWSE or LOCK modes. To consume a message, the subscriber can use the REMOVE or REMOVE_NODATA modes. If a subscriber browses a message, the message remains available for further processing. Similarly a locked message remains available for further processing after the subscriber releases the lock by performing a transaction commit or rollback. To prevent a viewed message from being dequeued by a concurrent user, you should view the message in locked mode. After a subscriber consumes a message using either of the REMOVE modes, the message is no longer available for dequeue requests.

    When a subscriber dequeues a message using REMOVE_NODATA mode, the request does not retrieve the payload of the message. This mode is useful when the user has already examined the message payload, possibly by means of a previous BROWSE dequeue. In this way, you can avoid the overhead of payload retrieval that can be substantial for large payloads.

    __Note__: One event is enqueued for each subscriber. Thus removing an event from one subscriber's queue does not remove it from the queues of other subscribers.

- **CONSUMER_NAME:** A subscriber can dequeue a message from the WWSBR_EVENT_Q queue by supplying the following queue name:

    - In PL/SQL you supply the consumer name using the CONSUMER_NAME field in the DEQUEUE_OPTIONS_T record.
    - In OCI you supply the consumer name using the OCISetAttr procedure to specify a text string as the OCI_ATTR_CONSUMER_NAME of an OCI_DTYPE_AQDEQ_OPTIONS descriptor.
– In Visual Basic (OO4O) you supply the consumer name by setting the consumer property of the OraAQ object.

Multiple processes or operating system threads can use the same to dequeue concurrently from a queue. Unless the message ID of a specific message is specified during dequeue, the consumers can dequeue messages that are in the READY state.

---

**Note:** You should not need to use the Search parameters to dequeue CMEF events.

---

### 16.2.3 Exception Handling

A message is considered processed when all intended consumers have successfully dequeued the message. If a message cannot be processed for some reason, it moves to an exception queue.

A message is considered expired if one or more consumers does not dequeue it before the expiration time. Expired messages also move to an exception queue. An exception queue is a repository for all expired or unserviceable messages.

Applications cannot directly enqueue into exception queues. Also, an exception queue cannot have subscribers associated with it. However, an application that intends to handle these expired or unserviceable messages must dequeue from the exception queue.

CMEF exceptions are sent to the WWSBR_EVENT_ERR_Q exception queue. Expired messages from the WWSBR_EVENT_Q multiconsumer queue cannot be dequeued by the intended recipients of the message. However, they can be dequeued in REMOVE mode once by specifying a NULL consumer name in the dequeue options. The queue monitor removes expired messages from multiconsumer queues. This allows dequeuers to complete the dequeue operation by not locking the message in the queue table.

Since the queue monitor removes messages that have been processed by all consumers from multiconsumer queues at regular intervals, users may see a delay between when the messages have been completely processed and when they are physically removed from the queue.

---

**Note:** The WWSBR_EVENT_ERR_Q exception queue, like all exception queues, is a single-consumer queue.

---

### 16.2.4 Listening for Messages

Oracle Streams AQ can monitor multiple queues for messages with a single LISTEN call. A subscriber can use LISTEN to wait for messages for multiple subscriptions. It can also be used by gateway applications to monitor multiple queues. If the LISTEN call returns successfully, a dequeue must be used to retrieve the message. Without the LISTEN call, an application which sought to dequeue from a set of queues would have to continuously poll the WWSBR_EVENT_Q queue to determine if there is a message.

Alternatively, you could design your subscriber to have a separate dequeue process for each queue. However, if there are long periods with no traffic in any of the queues, including WWSBR_EVENT_Q, these approaches will create unacceptable overhead. The LISTEN call is well suited for such subscribers. When there are messages for multiple agents in the agent list, LISTEN returns with the first agent for whom there is a message.
You can use the LISTEN call to monitor receipt of messages on one or more queues on behalf of a list of agents, as shown in Example 16–3. The call takes a list of agents as an argument. You specify the queue to be monitored in the address field of each agent listed. You also must specify the name of the agent when monitoring multiconsumer queues.

Example 16–3 Listening to Messages on Multiple Queues

```sql
declare
    agent_w_msg   aq$agent;
    qlist         dbms_aq.agent_list_t;
begin
    -- MYQ1, MYQ2, MYQ3 are multi-consumer queues in the SCOTT schema.
    qlist(1) := aq$agent('agent1', 'scott.MYQ1', null);
    qlist(2) := aq$agent(null, 'scott.MYQ2', null);
    qlist(3) := aq$agent('agent3', 'scott.MYQ3', null);
    -- Listen with a timeout of 100 seconds.
    dbms_aq.listen(
        agent_list => qlist,
        wait       => 100,
        agent      => agent_w_msg
    );
    dbms_output.put_line('MSG in Q:'||agent_w_msg.address||'for '||agent_w_msg.name);
    dbms_output.put_line('');
end;
/
```

This is a blocking call that returns when there is a message ready for consumption for an agent in the list. If there are messages for more than one agent, only the first agent listed is returned. If there are no messages found when the wait time expires, an error is raised.

A successful return from the call is only an indication that there is a message for one of the listed agents in one of the specified queues. The interested agent should dequeue the relevant message. Example 16–4 illustrates the dequeue process combined with listening. Here, we dequeue the messages for the subscriber, JAY, for a certain time period.

Example 16–4 Listening and Dequeuing Messages

```sql
begin
    agent_list(1) := sys.aq$_agent('JAY', 'WWSBR_EVENT_Q', null);
    wait_time integer := 100;
    loop
        -- Wait for order status messages.
        dbms_aq.listen(
            agent_list => agent_list,
            wait       => wait_time,
            agent      => agent_w_message
        );
        -- If there are messages for JAY, dequeue them.
        if (agent_w_message.name = 'JAY') then
            dequeue_options.wait          := dbms_aq.NO_WAIT;
            dequeue_options.consumer_name := 'JAY';
            dequeue_options.navigation    := dbms_aq.FIRST_MESSAGE;
            dequeue_options.dequeue_mode  := dbms_aq.BROWSE;
            dbms_aq.dequeue(
                queue_name => 'item_queue',
                dequeue_options => dequeue_options,
            );
        end if;
    end loop;
end;
/
```
message_properties => message_properties,
payload            => message,
msgid              => message_handle
);
end if;
end loop;
exception
when NO_MESSAGES then
    dbms_output.put_line('No more messages for Jay');
end;
/

16.3 Using the Content Management Event Framework

Every portal item and page action generates a CMEF message that is enqueued to the
WWSBR_EVENT_Q queue. A subscriber can use the information contained within this
message to perform various actions using the OracleAS Portal PL/SQL APIs.

There are three basic steps in handling CMEF events, each of which is described and
illustrated later in this section:

1. Creating Subscriber Code
2. Adding a Subscriber to WWSBR_EVENT_Q
3. Enabling CMEF Events at the Page Group Level

This section also provides a description of the message payload, followed by several
examples of common portal actions and the events they generate.

16.3.1 Creating Subscriber Code

Oracle Streams AQ offers a content-based subscription model. Subscriber applications
can specify interest based on message content. You should execute CMEF event
subscriber code in the OracleAS Portal schema. Example 16–5 shows an example
subscriber.

Example 16–5 An Example Subscriber

create or replace <procedure name> as
    agent_list         dbms_aq.aq$_agent_list_t;
    wait_time          integer       := <time in seconds>;
    agent_w_message    sys.aq$_agent;
    dequeue_options    dbms_aq.dequeue_options_t;
    message_properties dbms_aq.message_properties_t;
    message_handle     raw(16);
    message            <portal schema>.wwsbr_event;
    l_subscriber       varchar2(30)  := '<subscriber name>';
    l_queue            varchar2(30)  := 'PORTAL.WWSBR_EVENT_Q';
    l_mode             binary_integer := dbms_aq.[BROWSE|LOCK|REMOVE|REMOVE_NODATA];
...
    <additional parameters>
...
BEGIN
    agent_list(1) := sys.aq$_agent(l_subscriber, l_queue, null);
    loop
        -- Listen for messages.
        dbms_aq.listen(
            agent_list => agent_list,
            wait       => wait_time,
Using the Content Management Event Framework

16.3.2 Adding a Subscriber to WWSBR_EVENT_Q

A subscriber subscribes to a queue from where it consumes messages. You have to add a subscriber to the WWSBR_EVENT_Q queue in order to process CMEF event messages, as shown in Example 16–6.

Example 16–6 Adding a Subscriber to WWSBR_EVENT_Q

declare
  subscriber sys.aq$_agent;
begin
  subscriber := sys.aq$_agent('<subscriber>', null, null);
  dbms_aqadm.add_subscriber(
    queue_name => '<portal schema>.wwsbr_event_q',
    subscriber => subscriber
  );
end;
/

16.3.3 Enabling CMEF Events at the Page Group Level

In OracleAS Portal, CMEF is enabled or disabled at the page group level. By default, CMEF is enabled when a user creates a page group, and thus, events are triggered whenever changes occur within the OracleAS Portal user interface or WebDAV.

To enable or disable CMEF for a page group:

1. Go to any page of the page group and switch to Edit mode.
2. In the toolbar at the top of the page, click the Properties link next to Page Group.
3. Click the Configure tab to bring it forward.

4. In the Content Management Event Framework section you can see whether CMEF is enabled or disabled. If you want to change this setting, click the Edit link (Figure 16–3).

5. To enable CMEF, select the Enable Content Management Event Framework check box. To disable CMEF, clear the check box (Figure 16–4).

6. Click OK to save your changes.

7. Click Close to return to the page.

Note: Make sure you click the link next to Page Group and not the one next to Page (Figure 16–2).
16.3.4 Examining CMEF Events

Use the CMEF Events page (Figure 16–5) to examine the subscribers that have been added to the WWSBR_EVENTS_Q queue.

To access the CMEF Events page, perform the following steps:

1. Login to your portal as the portal schema owner.
2. Enter the following URL in the browser Address field:
   
   http://<host>:<port>/portal/pls/<dad>/<schema>.wwsbr_event_dbg.show

   - host is the machine on which your portal is installed
   - port is the port used by your portal
   - dad is the Database Access Descriptor (DAD) for your OracleAS Portal installation.
   - schema is the schema in which OracleAS Portal is installed.

---

**Note:** The CMEF Events page is not supported by Oracle, but it is included within OracleAS Portal for debugging purposes. It is only accessible by the portal schema owner.

---

**Figure 16–5  CMEF Events Page**

---

16.3.5 Running a CMEF Subscriber

To run a subscriber, issue the command shown in Example 16–7 at a SQL prompt.

**Example 16–7  Running a Subscriber**

begin
  <subscriber procedure>();
end;
/

16.3.6 CMEF Message Payload

Every OracleAS Portal user interface or PL/SQL content management action falls under one of the three main CMEF events: INSERT, UPDATE, DELETE. States provide
more meaning to events. For example, the following types of ADD/INSERT ITEM events:

- Add an item and publish it immediately.
- Add an item and publish it at a later date.
- Add an item that requires approval, because the user has Manage Items With Approval privileges.

For a detailed list of the actions and related events and states, refer to Appendix G, "Content Management Event Framework Events".

Each CMEF event has an associated CMEF message payload as shown in Table 16–2.

<table>
<thead>
<tr>
<th>Message Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTION</td>
<td>VARCHAR2(30)</td>
<td>The portal action that triggered the event.</td>
</tr>
<tr>
<td>RAW_EVENT</td>
<td>VARCHAR2(30)</td>
<td>The event produced as a result of the portal action.</td>
</tr>
<tr>
<td>STATE</td>
<td>VARCHAR2(30)</td>
<td>Provides additional information related to the event.</td>
</tr>
<tr>
<td>OBJECT_ID</td>
<td>NUMBER</td>
<td>The ID of the object to which the event relates (for example, item, page, category, and so on).</td>
</tr>
<tr>
<td>OBJECT_SITE_ID</td>
<td>NUMBER</td>
<td>The ID of the page group to which the object belongs.</td>
</tr>
<tr>
<td>OBJECT_LANGUAGE</td>
<td>VARCHAR2(30)</td>
<td>The session language. NULL if the action is independent of language.</td>
</tr>
<tr>
<td>PAGE_ID</td>
<td>NUMBER</td>
<td>For items, the ID of the page on which the item appears.</td>
</tr>
<tr>
<td>PAGE_SITE_ID</td>
<td>NUMBER</td>
<td>For items, the ID of the page group to which the page identified in PAGE_ID belongs.</td>
</tr>
<tr>
<td>OBJECT_CLASS</td>
<td>VARCHAR2(30)</td>
<td>Records the class of object about which an event has been raised.</td>
</tr>
<tr>
<td>EVENTS_USER</td>
<td>VARCHAR2(256)</td>
<td>The name of the user who performed the portal action.</td>
</tr>
<tr>
<td>EVENTS_DATE</td>
<td>VARCHAR2(60)</td>
<td>The date on which the event occurred. Format: dd-mon-yyyy HH12:MI PM</td>
</tr>
<tr>
<td>ID1 (OVERLOADED)</td>
<td>NUMBER</td>
<td>For items, the item type ID.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For pages, the page type ID.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For item and page types, the base type ID.</td>
</tr>
</tbody>
</table>
This section describes some of the most common portal actions and shows how to include code in your subscriber to detect these actions. These actions may occur through the OracleAS Portal user interface, the OracleAS Portal PL/SQL APIs, or WebDAV.

For a more detailed list of portal actions and the events and message payloads that they generate, refer to Appendix G, "Content Management Event Framework Events".

### 16.3.7.1 Page and Page Group Actions

In OracleAS Portal, a portal is a collection of one or more page groups. A page group is a hierarchical collection of pages for which common attributes and mechanisms can be established.

#### 16.3.7.1.1 Creating a Page

Creating a page produces the following CMEF message payload:

<table>
<thead>
<tr>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD_PAGE</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>PAGE</td>
</tr>
<tr>
<td>ADD_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
</tbody>
</table>

The first message is for the page itself, and the second is for the portlet instance that displays the default navigation page on the page.

If you want your subscriber to respond to the creation of a page, perform the following check:

```sql
if ((message.object_class = 'PAGE') and
    (message.raw_event    = wwsbr_event_q_access.EVENT_INSERT) then
  . . .
end if;
```
16.3.7.1.2 Updating the Access Control List of a Page  Changing the ACL of a page so that it does not inherit from that of its page group, then clicking **Apply** or **OK** produces the following message payload:

<table>
<thead>
<tr>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPECIFY_PAGE_ACL</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
</tbody>
</table>

Now, adding a user to the ACL of a page, then clicking **Add** produces the following message payload:

<table>
<thead>
<tr>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD_PAGE_ACL</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
</tbody>
</table>

Changing the ACL of a page so that it does not inherit from that of its page group, immediately adding a user or group to the ACL, and then clicking **Apply** or **OK** produces the following message payload:

<table>
<thead>
<tr>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPECIFY_AND_ADD_PAGE_ACL</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
</tbody>
</table>

Changing the ACL of a page so that it inherits that of the page group then clicking **Apply** or **OK** produces the following message payload:

<table>
<thead>
<tr>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>INHERIT_PAGE_ACL</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
</tbody>
</table>

Clicking **Apply** or **OK** on the Page Properties page, produces an additional message:

<table>
<thead>
<tr>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPDATE_PAGE_ACL</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
</tbody>
</table>

16.3.7.1.3 Updating the Access Control List of a Page Group  Updating the access control list (ACL) of a page group by adding a user or group, or deleting a user or group produces the following message payload:

<table>
<thead>
<tr>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD_PAGEGROUP_ACL</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE_GROUP</td>
</tr>
<tr>
<td>or DELETE_PAGEGROUP_ACL</td>
<td></td>
<td></td>
<td>PAGE_GROUP</td>
</tr>
<tr>
<td>UPDATE_PAGEGROUP_ACL</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE_GROUP</td>
</tr>
</tbody>
</table>
If you want your subscriber to respond to general ACL updates on a page group, perform the following check:

```plaintext
if ((message.action       = 'UPDATE_PAGE_GROUP_ACL') and
    (message.object_class = 'PAGE_GROUP') and
    (message.raw.event    = wwsbr_event_q_access.EVENT_UPDATE)) then
  . . .
end if;
```

However, if you are more interested in filtering for actual update and delete events on the page group’s ACL, then your subscriber should perform the following checks:

```plaintext
if ((message.object_class = 'PAGE_GROUP') and
    (message.raw_event    = wwsbr_event_q_access.EVENT_UPDATE)) then
  if (message.action = 'ADD_PAGE_GROUP_ACL') then
    . . .
  end if;
  if (message.action = 'DELETE_PAGE_GROUP_ACL') then
    . . .
  end if;
end if;
```

**Note:** The `ADD_PAGE_GROUP_ACL` and `DELETE_PAGE_GROUP_ACL` events are triggered when the user clicks **Add** for each user that is added to/deleted from the ACL. The `UPDATE_PAGE_GROUP_ACL` is generated when the user clicks **Apply** or **OK**.

### 16.3.7.1.4 Deleting a Page
Deleting a page produces the following message payload:

<table>
<thead>
<tr>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE_PAGE</td>
<td>DELETE</td>
<td>PURGED</td>
<td>PAGE</td>
</tr>
</tbody>
</table>

**Note:** Events are not generated for the sub-pages or items that are on the page.

### 16.3.7.2 Item Actions
Items are one of the basic components of a portal page. Items in a portal are based on item types. An action on an item triggers a CMEF event irrespective of the item type.

For example, an `ADD_ITEM` action occurs whenever a user adds an item of a base, extended, or custom item type. This ensures that there is consistent CMEF messaging behavior when an item action occurs.

#### 16.3.7.2.1 Creating an Item and Publishing it at the Same Time
If a user has **Manage** privileges, creating an item of any type on a page produces the following message payload:

<table>
<thead>
<tr>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
</tbody>
</table>

If you want you subscriber to respond to this action, perform the following check:
if ((message.action = 'ADD_ITEM') and
(message.object_class = 'ITEM') and
(message.raw_event = 'INSERT') and
(message.state = 'PUBLISHED')) then
   ... 
end if;

16.3.7.2.2  Adding an Item That Requires Approval  If approvals and notifications are
enabled for a page (or page group), and a user with Manage Items With Approval
privileges, adding an item to a page produces the following message payload:

<table>
<thead>
<tr>
<th>Action Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBMIT_ITEM_FOR_ APPROVAL INSERT</td>
<td>NOT_PUBLISHED</td>
<td>ITEM</td>
</tr>
</tbody>
</table>

If you want your subscriber to respond to this action, perform the following check:

if ((message.action = 'SUBMIT_ITEM_FOR_APPROVAL') and
(message.object_class = 'ITEM') and
(message.raw_event = 'INSERT') and
(message.state = 'NOT_PUBLISHED')) then
   ... 
end if;

16.3.7.2.3  Approving an Item  Approving an item triggers either an INSERT or UPDATE
event, which may also be followed by either a DELETE or UPDATE event depending
upon whether or not versioning is enabled.

*Item approved; versioning disabled*

This is the simplest case. If the item still has approval steps to pass through, then an
UPDATE event is triggered:

<table>
<thead>
<tr>
<th>Action Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPROVE_ITEM_STEP UPDATE GENERAL</td>
<td>ITEM</td>
<td></td>
</tr>
</tbody>
</table>

If the item has completed all approval steps, then an INSERT event is triggered:

<table>
<thead>
<tr>
<th>Action Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPROVE_ITEM INSERT PUBLISHED ITEM</td>
<td>ITEM</td>
<td></td>
</tr>
</tbody>
</table>

*Item approved; versioning enabled; current version overwritten*

If item versioning is set to Simple at the page or page group level, and a user selects
Overwrite Current Version when editing an item, on approval of the item, two events
are generated. The first event is for the item that is approved, and a second DELETE
event for the item that is overwritten as a result of the item being approved:

<table>
<thead>
<tr>
<th>Action Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPROVE_ITEM INSERT PUBLISHED ITEM</td>
<td>ITEM</td>
<td></td>
</tr>
<tr>
<td>APPROVE_ITEM DELETE PURGED ITEM</td>
<td>ITEM</td>
<td></td>
</tr>
</tbody>
</table>
**Item approved; versioning enabled; new and current version**

If item versioning is set to **Audit** at the page or page group level, or it is set to **Simple** and a user selects **Add Item As New And Current Version**, on approval of the item an **INSERT** event is generated. An **UPDATE** event is also generated that is related to marking the previous version of the item as UNPUBLISHED:

<table>
<thead>
<tr>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPROVE_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>APPROVE_ITEM</td>
<td>UPDATE</td>
<td>UNPUBLISHED</td>
<td>ITEM</td>
</tr>
</tbody>
</table>

If you are writing a subscriber that sends a notification when an item is approved or has passed through a stage of being approved, you should perform the following check:

```plaintext
-- If an item is approved.
if ((message.action = 'APPROVE_ITEM') and
    (message.object_class = 'ITEM') and
    (message.raw_event = 'INSERT') and
    (message.state = 'PUBLISHED')) then
  ...
-- If an item has passed an approval step.
elsif ((message.action = 'APPROVE_ITEM_STEP') and
       (message.object_class = 'ITEM') and
       (message.raw_event = 'UPDATE') and
       (message.state = 'PUBLISHED')) then
  ...
end if;
```

**Item approved, versioning enabled; new but not current version**

An **INSERT** event occurs for the item that is added, but the state of the item is marked as **NOT_PUBLISHED** to indicate that it is not published as the current version:

<table>
<thead>
<tr>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPROVE_ITEM</td>
<td>INSERT</td>
<td>NOT_PUBLISHED</td>
<td>ITEM</td>
</tr>
</tbody>
</table>

16.3.7.2.4  **Applying a Category or Perspective to an Item** Applying a different category or perspective to an item produces the same message payload as editing an item:

<table>
<thead>
<tr>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDIT_ITEM</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>ITEM</td>
</tr>
</tbody>
</table>

**Note:** No specific event is generated when the category or perspectives applied to an item are changed, and no additional information is provided.

If you want your subscriber to respond to this action, perform the following check:

```plaintext
if ((message.action = 'EDIT_ITEM') and
    (message.object_class = 'ITEM') and
    (message.raw_event = 'UPDATE') and
    (message.state = 'GENERAL')) then
```

Using the Content Management Event Framework  16-17
16.3.7.2.5 Deleting an Item Deleting an item from a page group that retains deleted items (that is, items are marked for deletion, but not actually deleted) produces the following message payload:

<table>
<thead>
<tr>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE_ITEM</td>
<td>DELETE</td>
<td>MARKED_FOR_DELETE</td>
<td>ITEM</td>
</tr>
</tbody>
</table>

Deleting an item from a page group that does not retain deleted items (that is, deleted items are immediately and permanently removed) produces the following message payload:

<table>
<thead>
<tr>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE_ITEM</td>
<td>DELETE</td>
<td>PURGED</td>
<td>ITEM</td>
</tr>
</tbody>
</table>

Your subscriber can use the state value within the message payload to determine what type of delete action occurred:

```sql
if ((message.action       = 'DELETE_ITEM') and
    (message.object_class = 'ITEM') and
    (message.raw_event    = 'DELETE')) then
    . . .
    -- If item is in a page group that does not actually delete items.
    if (message.state = 'MARKED_FOR_DELETE') then
        . . .
        -- If item is in a page group that actually deletes items.
    elsif (message.state = 'PURGED') then
        . . .
    end if;
end if;
```

16.4 Installing the Examples

If you would like to deploy and use the examples within the next few sections, we recommend that you create them in a separate schema, called CMEFSAMPLES. To create this schema, use the following steps:

1. Create the database schema CMEFSAMPLES. You need to do this as the SYS user, since you need to grant permissions packages to which only SYS has access. This database schema must be in the same database as that in which the OracleAS Portal repository resides. For example:

   ```sql
   @connect "/ as sysdba"
   drop user cmefsamples cascade;
   create user cmefsamples identified by oracle1
   default tablespace users temporary tablespace temp;
   grant connect, resource to cmefsamples;
   ```

2. As the SYS schema, grant the following privileges to CMEFSAMPLES:

   ```sql
   @connect "/ as sysdba"
   grant create table                 to cmefsamples;
   grant create sequence             to cmefsamples;
   ```
3. Grant the CMEFSAMPLES schema the permission to call the OracleAS Portal PL/SQL APIs. For information about how to do this, refer to Section 9.3, "Providing Access to the APIs and Secure Views".

4. Log into the portal schema and grant permissions on the following:

   EXECUTE DBMS_AQADM.GRANT_QUEUE_PRIVILEGE('DEQUEUE', 'WWSBR_EVENT_Q', 'cmefsamples', FALSE);
   grant execute on wwsbr_event_q_access to cmefsamples;
   grant execute on wwpob_page_util to cmefsamples;
   grant select on wwsbr_all_folders to cmefsamples;
   grant execute on wwsbr_thing_types to cmefsamples;
   grant execute on wwv_thingdb to cmefsamples;
   grant execute on wwsbr_event to cmefsamples;

5. Log into the CMEFSAMPLES schema and run the following:

   sqlplus cmefsamples/<password>
   create synonym aq$_agent for sys.aq$_agent;

You can download the code for the following examples from OTN:

http://www.oracle.com/technology/products/ias/portal/files/cm_overview_10g1014_cmef_samples.zip

### 16.5 Example: Portal Object Event Logging

The LOG_PORTAL_EVENT subscriber in Example 16–9 listens to CMEF events and then writes them to a log database table called CMEF_LOG_TABLE (Example 16–8).

#### Example 16–8 The CMEF_LOG_TABLE

```sql
create table cmef_log_table(
    action varchar2(30),
```
Example: Portal Object Event Logging

```sql
event           varchar2(30),
state           varchar2(30),
object_type     varchar2(30),
object_id       number,
object_site_id  number,
object_language varchar2(30),
page_id         number,
page_site_id    number,
performed_by    varchar2(30)
)

Example 16–9 The LOG_PORTAL_EVENT Subscriber

create or replace procedure log_portal_event as
agent_list         dbms_aq.aq$_agent_list_t;
wait_time          integer        := 5;
agent_w_message    sys.aq$_agent;
dequeue_options    dbms_aq.dequeue_options_t;
message_properties dbms_aq.message_properties_t;
message_handle     raw(16);
message            portal.wwsbr_event;
l_subscriber       varchar2(30)   := 'CMEF_LOG_PORTAL_EVENT';
l_queue            varchar2(30)   := 'PORTAL.WWSBR_EVENT_Q';
l_mode             binary_integer := dbms_aq.REMOVE;
begin
agent_list(1) := sys.aq$_agent(l_subscriber, l_queue, null);
loop
  -- Wait for messages.
  dbms_aq.listen(
    agent_list => agent_list,
    wait       => wait_time,
    agent      => agent_w_message
  );
  if (agent_w_message.name = l_subscriber) then
    dequeue_options.wait          := dbms_aq.NO_WAIT;
    dequeue_options.consumer_name := l_subscriber;
    dequeue_options.navigation    := dbms_aq.FIRST_MESSAGE;
    dequeue_options.dequeue_mode  := l_mode;
    dbms_aq.dequeue(
      queue_name     => l_queue,
      dequeue_options => dequeue_options,
      message_properties => message_properties,
      payload         => message,
      msgid           => message_handle
    );
    insert into cmef_log_table values(
      message.action,
      message.raw_event,
      message.state,
      message.object_class,
      message.object_id,
      message.object_site_id,
      message.object_language,
      message.page_id,
      message.page_site_id,
      message.events_user
    );
    commit;
  end if;
end loop;
```
The LOG_PORTAL_EVENT subscriber continuously listens for events on the WWSBR_EVENT_Q queue until the wait time of 5 seconds (as specified in the wait_time variable) is reached. It then dequeues CMEF events in REMOVE mode and then inserts the message payload values into the log table.

You could use the code in Example 16–9 to build an HTML page that displays the results of the log table. For example, the table in Figure 16–6 shows the CMEF message payload for an edit item action:

**Figure 16–6  HTML Table Displaying CMEF Message Payload Values**

<table>
<thead>
<tr>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
<th>Object ID</th>
<th>Object Site ID</th>
<th>Object Language</th>
<th>Page ID</th>
<th>Page Site ID</th>
<th>Performed by</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDIT_ITEM</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>ITEM</td>
<td>4259</td>
<td>55</td>
<td>us</td>
<td>1</td>
<td>55</td>
<td>JAY</td>
</tr>
</tbody>
</table>

The properties of the CMEF message payload (for example, message.raw_event) are described in Section 16.3.6, "CMEF Message Payload".

If you want the LOG_PORTAL_EVENT subscriber to continually remove messages off the WWSBR_EVENT_Q queue, then you need to remove the following from Example 16–9:

```sql
dbms_aq.listen(
  agent_list => agent_list,
  wait       => wait_time,
  agent      => agent_w_message
);
if (agent_w_message.name = l_subscriber) then
  dequeue_options.wait          := dbms_aq.NO_WAIT;
  dequeue_options.consumer_name := l_subscriber;
  dequeue_options.navigation    := dbms_aq.FIRST_MESSAGE;
  dequeue_options.dequeue_mode  := l_mode;
END IF;
```

Example 16–10 shows how to add the LOG_PORTAL_EVENT subscriber created in the previous section to the WWSBR EVENT_Q queue.

**Example 16–10  Adding the LOG_PORTAL_EVENT Subscriber to WWSBR_EVENT_Q**

```sql
declare
  subscriber sys.aq$_agent;
begin
  subscriber := sys.aq$_agent('CMEF_LOG_PORTAL_EVENT', null, null);
  dbms_aqadm.add_subscriber(
    queue_name => 'portal.wwsbr_event_q',
    subscriber => subscriber
  );
  end;
/
```

To run the LOG_PORTAL_EVENT subscriber, issue the command shown in Example 16–11.

**Example 16–11  Running the LOG_PORTAL_EVENT CMEF Subscriber**

```sql
begin
```
16.6 Example: Item Notification

The item_notify subscriber in Example 16–12 sends an e-mail notification whenever a user adds, updates, or deletes an item on a specified page:

Example 16–12 The CMEF_ITEM_NOTIFY Subscriber

create or replace procedure item_notify as
  MIME_TYPE_TEXT      constant varchar(30)  := 'text/plain';
  CUSTOM_ATTRIBUTE_ID constant number := 1020;
  ADDED               constant varchar2(20) := 'added';
  UPDATED             constant varchar2(20) := 'updated';
  DELETED             constant varchar2(20) := 'deleted';
  ITEM                constant varchar2(10) := 'ITEM';
  agent_list          dbms_aq.aq$_agent_list_t;
  wait_time           integer               := 5;
  begin_time          pls_integer;
  end_time            pls_integer;
  agent_w_message     sys.aq$_agent;
  dequeue_options     dbms_aq.dequeue_options_t;
  message_properties  dbms_aq.message_properties_t;
  message_handle      raw(16);
  message             portal.wwsbr_event;
  l_subscriber        varchar2(30)          := 'CMEF_ITEM_NOTIFY';
  l_queue             varchar2(30)          := 'PORTAL.WWSBR_EVENT_Q';
  l_page_id           number                := 33; -- Page ID of page.
  l_mode              binary_integer        := dbms_aq.REMOVE;
  l_rec               varchar2(256);
  l_body              varchar2(4000)        := null;
  l_from_user         varchar2(30)          := '<from-email-address>'; 
  l_to_user           varchar2(30)          := '<to-email-address>'; 
  l_event             varchar2(30)          := ''; 
  l_portal_user_name  varchar2(30)          := 'portal';
  l_portal_password   varchar2(30)          := '<portal password>';
  l_display_name      varchar2(256)         := '';
begin
  begin_time := dbms_utility.get_time;
  agent_list(l) := sys.aq$agent(l_subscriber, l_queue, null);
  loop
    -- Wait for messages.
    dbms_aq.listen(
      agent_list => agent_list,
      wait       => wait_time,
      agent      => agent_w_message
    );
    if (agent_w_message.name = l_subscriber) then
      dequeue_options.wait := DBMS_AQ.NO_WAIT;
      dequeue_options.consumer_name := l_subscriber;
      dequeue_options.navigation := dbms_aq.FIRST_MESSAGE;
      dequeue_options.dequeue_mode := l_mode;
      dbms_aq.dequeue(
        queue_name => l_queue,
        dequeue_options => dequeue_options,
        message_properties => message_properties,
      );
    end if;
  end loop;
end;
Example: Item Notification

```sql
payload => message,
msgid => message_handle
);
if ((message.object_class = ITEM)) then
    -- Determine the type of event that occurred.
    if (message.raw_event = portal.wwsbr_event_q_access.EVENT_INSERT) then
        l_event := ADDED;
    elsif (message.raw_event = portal.wwsbr_event_q_access.EVENT_UPDATE) then
        l_event := UPDATED;
    elsif (message.raw_event = portal.wwsbr_event_q_access.EVENT_DELETE) then
        l_event := DELETED;
    end if;
    if ((l_event = ADDED) or
        (l_event = UPDATED) or
        (l_event = DELETED)) then
        -- Set the Portal Context.
        portal.wwctx_api.set_context(l_portal_user_name, l_portal_password);
        begin
            -- Get the Item Display name from the all items view.
            select display_name
            into l_display_name
            from portal.wwsbr_all_items
            where id=message.object_id
            and caid=message.object_site_id;
            exception
            when NO_DATA_FOUND then
                dbms_output.put_line(sqlerrm);
                exit;
            end;
            -- Only send an e-mail if the item has a display name.
            if (l_display_name is not null) then
                -- Construct the body of the message.
                l_body := l_body || '<br>An item titled ' ||
                    CSBR_UTIL.file_viewer_hyperlink(
                        message.object_id, message.page_site_id, l_display_name)
                    || ' was ' || l_event || ' on ' ||
                    to_char(message.events_date,'dd-mon-yyyy hh12:mi pm')
                    || ' by ' || message.events_user || '. '; 
                -- Send the message.
                html_email(
                    p_from => l_from_user,
                    p_to => l_to_user,
                    p_subject => 'Item ' || l_event || ' Notification',
                    p_text => 'Text',
                    p_html => l_body
                );
            end if;
        end if;
    end if;
end if;
end if;
end loop;
end;
```
The ITEM_NOTIFY CMEF subscriber removes the actual events from the queue so that multiple e-mails are not sent to the user.

---

**Note:** For descriptions of the file_viewer_hyperlink function, which gets the URL of the item, and the html_email procedure, which handles the actual sending of the message, refer to Additional Code.

---

For further information on the performance and timing of AQ events, refer to the Streams Advanced Queuing page on OTN:

http://www.oracle.com/technology/products/aq

For an example of how to add a subscriber to the WWSBR_EVENT_Q queue, refer to Example 16–10.

For an example of how to run a subscriber, refer to Example 16–11.

The ITEM_NOTIFY CMEF subscriber runs for 3 seconds. This ensures that the subscriber does not continuously run and consume system resources. Alternatively, you could modify the wait_time value used in the subscriber to a higher value, say 60 seconds, so that the subscriber listens for new events every minute. You could also use a DBMS_JOB to have the subscriber run at a specified interval, for example, 30 minutes, as shown in Example 16–13:

**Example 16–13 Using DMBS_JOB**

```sql
declare
  v_job number;
begin
  dbms_job.submit(
    job => v_job,
    what => 'item_notify()',
    interval => 'SYSDATE + (30/(24*60))'
  );
end;
/
```

**Additional Code**

Example 16–14 shows the CSBR_UTIL package.

**Example 16–14 CSBR_UTIL Package**

```sql
create or replace package CSBR_UTIL as
  function file_viewer_hyperlink(
    p_item_id in number,
    p_caid in number,
    p_text in varchar2) return varchar2;
end;

create or replace package body CSBR_UTIL as

  function file_viewer_url(
    p_item_id in number,
    p_caid in number) return varchar2 is

    l_return varchar2(10000);
```
Example 16–15 shows the html_email procedure.

Note: For this procedure to work you will need to deploy the Oracle Database Sendmail package on your database.

Example 16–15 The HTML E-mail Procedure

create or replace procedure html_email(
  p_to     in varchar2,
  p_from   in varchar2 default '',
  p_subject in varchar2,
  p_text   in varchar2 default null,
  p_html   in varchar2 default null
)
Example: Item Notification

```sql
is

  l_boundary   varchar2(255) default 'a1b2c3d4e3f2g1';
  l_connection utl_smtp.connection;
  l_body_html clob := empty_clob; -- This LOB will be the e-mail message.
  l_offset     number;
  l_ammount    number;
  l_temp       varchar2(32767) default null;

begin

  l_connection := utl_smtp.open_connection('<Mail Server here>');
  utl_smtp.helo( l_connection, '<Mail Server here>' );
  utl_smtp.mail( l_connection, p_from );
  utl_smtp.rcpt( l_connection, p_to );

  l_temp := l_temp || 'MIME-Version: 1.0' || chr(13) || chr(10);
  l_temp := l_temp || 'To: ' || p_to || chr(13) || chr(10);
  l_temp := l_temp || 'From: ' || p_from || chr(13) || chr(10);
  l_temp := l_temp || 'Subject: ' || p_subject || chr(13) || chr(10);
  l_temp := l_temp || 'Reply-To: ' || p_from || chr(13) || chr(10);
  l_temp := l_temp || 'Content-Type: multipart/alternative; boundary=' || l_boundary || chr(13) || chr(10);

  dbms_lob.createtemporary( l_body_html, false, 10 );
  dbms_lob.write(l_body_html, length(l_temp), 1,l_temp);

  l_offset := dbms_lob.getlength(l_body_html) + 1;
  l_temp   := '--' || l_boundary || chr(13) || chr(10);
  l_temp   := l_temp || 'content-type: text/plain; charset=us-ascii' || chr(13) || chr(10);
  dbms_lob.write(l_body_html, length(l_temp), l_offset, l_temp);

  dbms_lob.write(l_body_html, length(p_text), l_offset, p_text);

  l_temp := chr(13) || chr(10) || 'content-type: text/html;' || chr(13) || chr(10);
  dbms_lob.write(l_body_html, length(l_temp), l_offset, l_temp);

  dbms_lob.write(l_body_html, length(p_html), l_offset, p_html);

  l_temp := chr(13) || chr(10) || chr(13) || 'Content-Type: text/html;' || chr(13) || chr(10);
  dbms_lob.write(l_body_html, length(l_temp), l_offset, l_temp);

  dbms_lob.write(l_body_html, length(p_html), l_offset, p_html);

end;
```

-- Send the e-mail in 1900 byte chunks to UTL_SMTP.
1.6.7 Example: Item Validation

The item_validation CMEF subscriber in Example 16–16 validates that a URL item added to a specified portal page includes a URL. If the item includes a URL, then the item is approved using the wwsbr_api.approve API. Otherwise the item is rejected using the wwsbr_api.reject API.

Example 16–16  The ITEM_VALIDATION CMEF Subscriber

create or replace procedure item_validation as
MIME_TYPE_TEXT constant varchar2(30) := 'text/plain';
agent_list dbms_aq.aq$_agent_list_t;
wait_time integer := 5;
begin_time pls_integer;
end_time pls_integer;
agent_w_message sys.aq$_agent;
dequeue_options dbms_aq.dequeue_options_t;
message_properties dbms_aq.message_properties_t;
message_handle raw(16);
message portal.wwsbr_event;
1_subscriber varchar2(30) := 'CMEF_ITEM_VALIDATION';
1_queue varchar2(30) := 'PORTAL.WWSBR_EVENT_Q';
1_mode binary_integer := dbms_aq.REMOVE;
1_attribute_site_id number := 0;
1_event varchar2(30);
1_doc wwdoc_api.document_record;
1_desc varchar2(4000);
1_portal_user_name varchar2(30) := 'portal';
1_portal_password varchar2(30) := '<portal password>';
1_itemtype varchar2(30) := '';
1_url varchar2(4000) := '';
begin
begin_time := dbms_utility.get_time;
agent_list(1) := sys.aq$_agent(1_subscriber, 1_queue, null);
loop
   -- Wait for messages.
   dbms_aq.listen(
      agent_list => agent_list,
      wait => wait_time,
      agent => agent_w_message
   );
   if (agent_w_message.name = 1_subscriber) then
      dequeue_options.wait := DBMS_AQ.NO_WAIT;
      dequeue_options.consumer_name := 1_subscriber;
end loop;
dequeue_options.navigation := dbms_aq.FIRST_MESSAGE;
dequeue_options.dequeue_mode := l_mode;
dbms_aq.dequeue(
    queue_name => l_queue,
    dequeue_options => dequeue_options,
    message_properties => message_properties,
    payload => message,
   msgid => message_handle);

-- If the event is an ITEM INSERT event and marked for later publication.
if ((message.action = 'SUBMIT_ITEM_FOR_APPROVAL') and
    (message.object_class = 'ITEM') and
    (message.raw_event = 'INSERT') and
    (message.state = 'NOT_PUBLISHED')) then

  -- Set the Portal Context.
  portal.wwctx_api.set_context(l_portal_user_name, l_portal_password);
  -- Get the Object Information.
  begin
    select itemtype, url
    into l_itemtype, l_url
    from portal.wwsbr_all_items
    where id=message.object_id
    and caid=message.object_site_id;
  exception
    when NO_DATA_FOUND then
      dbms_output.put_line(sqlerrm);
      exit;
  end;

  -- If the item type is a URL and no URL is specified.
  if (l_itemtype = portal.wwsbr_thing_types.BASE_ITEM_TYPE_URL) and
      (l_url is null) then
    begin
      -- Update item to indicate that a URL needs to be specified.
      l_desc := '<font color=RED>NO URL SPECIFIED</font>;
      -- Reset the CMEF global variables.
      wwsbr_api.clear_cmef_context;
      wwsbr_api.set_attribute(
        p_site_id => message.object_site_id,
        p_thing_id => message.object_id,
        p_attribute_site_id => l_attribute_site_id,
        p_attribute_id => wwsbr_api.ATTRIBUTE_DESCRIPTION,
        p_attribute_value => l_desc,
        p_log_cmef_message => FALSE)
    end;  

  -- Reject the item.
  wwsbr_api.reject(
    p_item_id => message.object_id,
    p_site_id => message.object_site_id,
    p_comment => 'Rejected'
  );

  -- Reset the CMEF global variables.
  wwsbr_api.clear_cmef_context;
  exception
    when OTHERS then
      dbms_output.put_line(sqlerrm);
  end;
else
  -- Approve the item.
  begin
Example: Item Validation

Example: Item Validation

```sql
wwsbr_api.approve(
    p_item_id => message.object_id,
    p_site_id => message.object_site_id,
    p_comment => 'Approved'
);
-- Reset the CMEF global variables.
wwsbr_api.clear_cmef_context;
exception
    when OTHERS then
        dbms_output.put_line(sqlerrm);
end;
commit;
end if;
end if;
end if;
end_time := dbms_utility.get_time;
if (end_time - begin_time) > 3000 then
    exit;
end if;
end loop;
-- Process cache invalidation messages.
wwpro_api_invalidation.execute_cache_invalidation;
end;/
```

The following check in the item_validation CMEF subscriber determines whether the item requires approval:

```
-- If the event is an ITEM INSERT event and marked for later publication.
if ((message.action = 'SUBMIT_ITEM_FOR_APPROVAL') and
    (message.object_class = 'ITEM') and
    (message.raw_event = 'INSERT') and
    (message.state = 'NOT_PUBLISHED')) then
    . . .
end if;
```

The item_validation CMEF subscriber uses the `wwsbr_api.set_attribute` API to set the item description to indicate that the user did not specify a URL. The subscriber also chooses not to log CMEF messages when calling this API (`p_log_cmef_messages = FALSE`).

```
-- Update item to indicate that a URL needs to be specified.
l_desc := '<font color=RED>No FILE NAME SPECIFIED</font>';
wwsbr_api.set_attribute(
    p_site_id => l_thing.siteid,
    p_thing_id => l_thing.id,
    p_attribute_site_id => l_attribute_site_id,
    p_attribute_id => wwsbr_api.ATTRIBUTE_DESCRIPTION,
    p_attribute_value => l_desc
    p_log_cmef_message => FALSE
);
```

Approving or rejecting an item with the `approve` or `reject` APIs triggers an event just the same as if the item were approved or rejected through the OracleAS Portal UI. For example, calling the `wwsbr_api.approve` API triggers an INSERT event with an action of APPROVE_ITEM.

```
wwsbr_api.approve(
    p_item_id => l_thing.id,
    p_site_id => l_thing.siteid,
    p_comment => 'Approved'

Using the Content Management Event Framework   16-29
```
16.8 Example: Integrating External Workflow

Organizations often find that they want to integrate their business processes into their enterprise portal. Using the capabilities of Oracle Workflow, you can implement portals to route information of any type according to the compliance rules defined for your organization. With CMEF, you can integrate both traditional, applications-based workflow and e-business integration workflow with your OracleAS Portal content management system.

This section provides an example of how to integrate external workflow with the OracleAS Portal content management system.

16.8.1 Integrating Workflow with OracleAS Portal

Oracle Workflow is a component of Oracle Application Server and Oracle E-Business Suite that enables you to design internal business processes and store them in a central repository. You can use Oracle Workflow to support a wide variety of compliance mandates, designing processes that are both auditable and repeatable, and enforce pre-set approvals and limits. Oracle's newest compliance solution, Oracle Internal Controls Manager (Oracle ICM) works in conjunction with Oracle Workflow to monitor internal business processes and ensure they are performed as designed.

In general, there are five steps involved in integrating Oracle Workflow with the OracleAS Portal content management system for content approval:

1. Enable approvals and notifications in OracleAS Portal.
2. Grant users the Manage Items With Approval privileges.
3. Create a portal user that will be used by the Oracle Workflow process to either approve or reject items and add that user to the portal approval process.
4. Register the workflow process with Oracle Workflow. This workflow process calls your compliance process, or it may perform your actual compliance process.
5. Create subscriber code and add the subscriber to WWSBR_EVENT_Q to process CMEF events. This subscriber initializes the workflow engine and then calls the workflow process.

We can see how this works by modifying the item validation example described in Section 16.7, “Example: Item Validation” to call a workflow process to perform the item validation (Figure 16–7).
Let's take a look at how this process would work, based on the diagram in Figure 16–7.

### 16.8.2 Example Overview

Figure 16–7 shows the process flow diagram for using CMEF to integrate Oracle Workflow with the OracleAS Portal content management system. When a user with Manage Items With Approval privileges adds an item to the portal, that item is marked as pending. Just as with the ITEM_VALIDATION subscriber in Example 16–16, an Oracle Streams Advanced Queuing subscriber is required to listen for events on the WWSBR_EVENT_Q queue (see Example 16–17). If the event is of type SUBMIT_ITEM_FOR_APPROVAL, the subscriber launches the workflow engine process, WF_CHECKURL, and passes the parameters for the portal item to the workflow process. The workflow process calls an external PL/SQL procedure (see Example 16–18) to perform its business logic, which in this case includes either approving or rejecting the portal item.

If the user specifies a URL, then Oracle Workflow approves the item using the wwsbr_api.approve API. Otherwise, Oracle Workflow rejects the item using the wwsbr_api.reject API.

This example assumes the following prerequisites:

- The CMEFSAMPLES schema exists in the same database as the portal schema.
- The Oracle OWF_MGR workflow schema is installed in the same database as the portal schema.
- The AQ_TM_PROCESSES parameter should be set to at least five. To check this, log in to the database as the SYS user and execute the following query:

```sql
SELECT value
FROM v$parameter
WHERE name = 'aq_tm_processes';
```
If the value is less than five, then set the value as follows:

```sql
alter system set aq_tm_processes=5;
```

- OracleAS Portal 10.1.4 has been installed.

### 16.8.3 Detailed Example Description

When a portal user with `Manage Items With Approval` privileges adds an item to a page, a `SUBMIT_ITEM_FOR_APPROVAL` event is added to the `WWSBR_EVENT_Q` queue. The item is marked as pending on the portal page. This user's items must be approved before they are made visible on the page.

The Streams Advanced Queuing subscriber `CMEF_WORKFLOW` listens for events on `WWSBR_EVENT_Q`. If the event is of type `SUBMIT_ITEM_FOR_APPROVAL`, then it launches the workflow engine process and passes the parameters for the portal item to the workflow process.

The workflow process `WF_CHECKURL` performs its business logic, which in this case includes either approving or rejecting portal items. `WF_CHECKURL` uses the views and APIs to approve or reject the item. If it approves the item, then an `APPROVE_ITEM` event is added to `WWSBR_EVENT_Q`. If it rejects the item, then a `REJECT_ITEM` event is added to `WWSBR_EVENT_Q`.

When the user refreshes the page, he or she will either see the item published on the page (if it was approved), or removed (if it was rejected).

The following six steps occur within this example:

1. **Enable Approvals and Notifications in OracleAS Portal**
2. **Grant Users the Manage Items With Approval Privileges**
3. **Run Scripts Required for the CMEF Workflow Integration Example**
4. **Create Subscriber and Check Procedures**
5. **Register the WF_CHECKURL Process with Oracle Workflow**
6. **Add the CMEF_WORKFLOW Subscriber to the WWSBR_EVENT_Q Queue**

Each of these steps is described in more detail in the following sections.

#### 16.8.3.1 Enable Approvals and Notifications in OracleAS Portal

The first thing you need to do is enable approvals and notifications in the page group.

To enable approvals and notifications:

1. Go to any page of the page group and switch to Edit mode.
2. In the toolbar at the top of the page, click the **Properties** link next to Page Group.
3. Click the **Configure** tab to bring it forward.
4. In the Approvals and Notifications section you can see whether approvals and notifications are enabled or disabled. If you want to change this setting, click the **Edit** link (Figure 16–8).

---

**Note:** Make sure you click the link next to Page Group and not the one next to Page (Figure 16–2).
5. To enable approvals and notifications, select the **Enable Approvals and Notifications** check box (Figure 16–9).

6. Click **OK** to save your changes.

7. Click **Close** to return to the page.

### 16.8.3.2 Grant Users the Manage Items With Approval Privileges

**Note:** You can skip this step if your external workflow process does not require the approval or rejection of portal items.

The next step is to grant *Manage Items With Approval* privileges to all users who need approval for their items. For the purposes of this example, let's specify that all users require approval for their items.

To specify that all users require approval for their items:

1. Go to any page in the page group and switch to Edit mode.
2. In the toolbar at the top of the page, click the **Properties** link next to Page Group.

**Note:** Make sure you click the link next to Page Group and not the one next to Page (Figure 16–2).

3. Click the **Approval** tab to bring it forward.
4. Select the **Require Approval for All Users** check box (Figure 16–10).
5. Click OK.

16.8.3.3 Run Scripts Required for the CMEF Workflow Integration Example

Before you proceed to the next step, there are several tasks that you need to perform. These are described in the following section.

1. Log in to the SYS schema and run the following command:

   \[\text{GRANT EXECUTE ON owf_mgr.wf_engine to cmefsamples;}\]

2. Log in to the portal schema and run \text{provsyns.sql}. When prompted for the schema/provider name enter \text{owf_mgr}.

3. Log in to the CMEFSAMPLES schema and run the following:

   \begin{verbatim}
   drop sequence WF_SEQ
   create sequence WF_SEQ
   increment by 1
   start with 1
   maxvalue 1000000000
   minvalue 1
   cache 20
   
   grant select on wf_seq to owf_mgr;
   \end{verbatim}

   This creates the sequence required by workflow, and allows the OWF_MGR workflow schema SELECT privileges on the sequence.

4. Create the synonym WF_ENGINE to the workflow schema:

   \[\text{create synonym wf_engine for owf_mgr.wf_engine;}\]

16.8.3.4 Create Subscriber and Check Procedures

You need to create a subscriber that listens to the WWSBR_EVENT_Q queue, waiting until a user adds an item that requires approval. The subscriber calls the WF_CHECKURL workflow process to perform the item validation and to send the e-mail notification. This is shown in Example 16–17.

\textbf{Example 16–17 The WORKFLOW_APPROVAL Subscriber}

\begin{verbatim}
create or replace procedure workflow_approval is
    MIME_TYPE_TEXT constant varchar2(30) := 'text/plain';
    agent_list          dbms_aq.aq$_agent_list_t;
    wait_time           integer               := 30;
    begin_time          pls_integer;
    end_time            pls_integer;
    agent_w_message     aq$_agent;
    dequeue_options     dbms_aq.dequeue_options_t;
    message_properties  dbms_aq.message_properties_t;
\end{verbatim}
message_handle raw(16);
message portal.wwsbr_event;
l_subscriber varchar2(30) := 'CMEF_WORKFLOW';
l_queue varchar2(30) := 'PORTAL.WWSBR_EVENT_Q';
l_mode binary_integer := dbms_aq.REMOVE;
l_attribute_site_id number := 0;
l_event varchar2(30);
l_doc wwdoc_api.document_record;
l_desc varchar2(4000);
l_itemkey varchar2(100);
l_job_nr number;
l_itemtype varchar2(100) := 'WF';
l_wf_process varchar2(100) := 'WF_CHECKURL';
l_is_indirect boolean := TRUE;
l_portal_user_name varchar2(30) := 'portal';
l_portal_password varchar2(30) := '<portal password>';
l_url varchar2(4000); := '';

begin
  begin_time := dbms_utility.get_time;
  agent_list(1) := aq$_agent(l_subscriber, l_queue, null);
  loop
    -- Wait for messages.
    dbms_aq.listen(
      agent_list => agent_list,
      wait => wait_time,
      agent => agent_w_message
    );
    if (agent_w_message.name = l_subscriber) then
      dequeue_options.wait := dbms_aq.NO_WAIT;
      dequeue_options.consumer_name := l_subscriber;
      dequeue_options.navigation := dbms_aq.FIRST_MESSAGE;
      dequeue_options.dequeue_mode := l_mode;
      dbms_aq.dequeue(
        queue_name => l_queue,
        dequeue_options => dequeue_options,
        message_properties => message_properties,
        payload => message,
        msgid => message_handle
      );
    end if;
  end loop;
  -- If the event is an ITEM INSERT event and marked for later publication.
  if ((message.action = 'SUBMIT_ITEM_FOR_APPROVAL') and
      (message.object_class = 'ITEM') and
      (message.raw_event = 'INSERT') and
      (message.state = 'NOT_PUBLISHED')) then
    portal.wwctx_api.set_context(l_portal_user_name, l_portal_password);
  -- Get the URL property for the object.
    select url
    into l_url
    from portal.wwsbr_all_items
    where id = message.object_id
    and caid = message.object_site_id;
  -- Get the nextval of the workflow sequence for the itemkey.
    select wf_seq.nextval
    into l_job_nr
    from dual;
    l_itemkey := lpad(to_char(l_job_nr), 5, '0');
  -- Launch the workflow engine process and pass the parameters for
  -- the portal item to the workflow process.
    wf_engine.createprocess(l_itemtype, l_itemkey, l_wf_process);
  end if;
end;
Example: Integrating External Workflow

-- Set the value for the portal URL.
  wf_engine.setitemattrtext(l_itemtype, l_itemkey, 'PORTAL_URL', l__url);
-- Set the value for the portal item ID.
  wf_engine.setitemattrnumber(l_itemtype, l_itemkey, 'PORTAL_ITEM_ID',
message.object_id);
-- Set the value for the portal page group ID.
  wf_engine.setitemattrnumber(l_itemtype, l_itemkey, 'PORTAL_SITE_ID',
message.object_site_id);
-- Start the workflow process.
  wf_engine.startprocess(l_itemtype, l_itemkey);
end if;
end if;
end loop;
exception
  when OTHERS then
    dbms_output.put_line(sqlerrm);
end workflow_approval;
/

You also need to create a procedure to check whether the user specified a file name for
the item to determine whether or not it can be approved. The workflow process
degrelates the actual validation and approval of the item to this CHECK_URL
procedure (Example 16–18).

**Example 16–18 The CHECK_URL Procedure**

create or replace procedure check_url(
  itemtype  in  varchar2,
  itemkey   in  varchar2,
  actid     in  number,
  funcmode  in  varchar2,
  resultout out varchar2) is

  MIME_TYPE_TEXT      constant varchar2(30) := 'text/plain';
  agent_list          dbms_aq.aq$_agent_list_t;
  wait_time           integer               := 30;
  begin_time          pls_integer;
  end_time            pls_integer;
  agent_w_message     aq$_agent;
  dequeue_options     dbms_aq.dequeue_options_t;
  message_properties  dbms_aq.message_properties_t;
  message_handle      RAW(16);
  message             portal.wwsbr_event;
  l_subscriber        varchar2(30)      := 'CMEF_WORKFLOW';
  l_queue             varchar2(30)      := 'PORTAL.WWSBR_EVENT_Q';
  l_mode              BINARY_INTEGER    := dbms_aq.REMOVE;
  l_attribute_site_id number            := 0;
  l_event             varchar2(30);
  l_doc               wwdoc_api.document_record;
  l_desc              varchar2(4000);
  l_itemtype          varchar2(100)    := 'WF';
  l_itemkey           varchar2(100);
  l_url               varchar2(100);
  l_siteid            number;
  l_itemid            number;
  l_ignore            boolean          := FALSE;
  l_is_indirect       boolean          := TRUE;
l_portal_user_name  varchar2(30)     := 'portal';
l_portal_password   varchar2(30)     := '<portal password>');

begin
    -- Get the values for the attributes stored in the procedure.
    l_url := wf_engine.getitemattrtext(itemtype,itemkey,'PORTAL_URL',l_ignore);
l_itemid := wf_engine.getitemattrnumber(itemtype,itemkey,'PORTAL_ITEM_ID',l_ignore);
l_siteid := wf_engine.getitemattrnumber(itemtype,itemkey,'PORTAL_SITE_ID',l_ignore);
    -- Set the portal context.
    portal.wwctx_api.set_context(l_portal_user_name, l_portal_password);
    -- If the item type is a URL item and no URL is specified.
    if l_url is null then
        begin
            -- Update the item to indicate that the URL needs to be specified.
            l_desc := '<font color=RED>NO URL SPECIFIED</font>';
            -- Reset the CMEF global variables.
            wwsbr_api.clear_cmef_context;
            wwsbr_api.set_attribute(
                p_site_id   => l_siteid,
                p_thing_id  => l_itemid,
                p_attribute_site_id => l_attribute_site_id,
                p_attribute_id      => wwsbr_api.attribute_description,
                p_attribute_value   => l_desc
            );
            -- Reject the item.
            wwsbr_api.reject(
                p_item_id  => l_itemid,
                p_site_id  => l_siteid,
                p_is_indirect => l_is_indirect,
                p_comment => 'Rejected'
            );
            -- Reset the CMEF global variables.
            wwsbr_api.clear_cmef_context;
            end;
        else
            -- Approve the item.
            begin
                wwsbr_api.approve(
                    p_item_id => l_itemid,
                    p_site_id => l_siteid,
                    p_comment => 'Approved'
                );
                -- Reset the CMEF global variables.
                wwsbr_api.clear_cmef_context;
                end;
            end if;
        commit;
    -- Process cache invalidation messages.
    wwpro_api_invalidation.execute_cache_invalidation;
exception
    when OTHERS then
        dbms_output.put_line(sqlerrm);
end check_url;
/
16.8.3.5 Register the WF_CHECKURL Process with Oracle Workflow

The WF_CHECKURL process actually approves or rejects the portal item and sends the e-mail notification. You need to register this process with Oracle Workflow using the Oracle Workflow Builder. Figure 16–11 shows the CHECK_FILE Oracle Workflow process.

Figure 16–11 The CHECK_FILE Oracle Workflow Process

Use the following steps to install the WF_CHECKURL process into the OWF_MGR schema:

1. Open the Oracle Workflow Builder.

   **Tip:** After installing the Workflow Builder, you may have to run it from the command line because you will have to set the ORACLE_HOME environment variable to that of the Workflow Builder. For example:
   ```
   set ORACLE_HOME=c:\oraclewf
   ```

2. Start a new workflow project.

3. Save the workflow project as `wf_checkurl.wft`.


5. Open `wf_checkurl.wft` in a text editor, for example, Notepad.

6. Copy and paste the WF_CHECKURL code into the file. You can download this code from Portal Center:
   ```
   http://www.oracle.com/technology/products/ias/portal/files/cm_overview_10g1014_cmef_samples.zip
   ```

7. Save the file and close the text editor.

8. Open `wf_checkurl.wft` in Oracle Workflow Builder.

9. Save `wf_checkurl.wft` into your portal database using the OWF_MGR schema.
**Tip:** If you do not know the OWF_MGR password, see MetaLink Note 198800.1.

### 16.8.3.6 Add the CMEF_WORKFLOW Subscriber to the WWSBR_EVENT_Q Queue

Next, you need to add the CMEF_WORKFLOW subscriber to the WWSBR_EVENT_Q queue so that it can process the CMEF events and trigger the WF_CHECKURL Oracle Workflow process, as shown in Example 16–19.

**Example 16–19 Adding the CMEF_WORKFLOW Subscriber to WWSBR_EVENT_Q**

```sql
declare
  subscriber sys.aq$_agent;
begin
  subscriber := sys.aq$_agent('CMEF_WORKFLOW', null, null);
  dbms_aqadm.add_subscriber(
    queue_name => 'portal.wwsbr_event_q',
    subscriber => subscriber
  );
end;
/
```

Now, when a user with *Manage Items With Approval* privileges add an item to a page, it is marked as Pending and a single INSERT event occurs, with a state of NOT_PUBLISHED:

<table>
<thead>
<tr>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBMIT_ITEM_FOR_APPROVAL</td>
<td>INSERT</td>
<td>NOT_PUBLISHED</td>
<td>ITEM</td>
</tr>
</tbody>
</table>

The CMEF_WORKFLOW_APPROVAL subscriber responds to this action and invokes the PORTAL_ITEM_WORKFLOW Oracle Workflow process.

The PORTAL_ITEM_APPROVAL Oracle Workflow process performs its business logic and calls the OracleAS Portal approve or reject API to approve or reject the item. It also sends an e-mail notification to the specified e-mail ID indicating the approval status.
Part IV contains the following appendixes:

- Appendix A, "Creating Portlets with the Portlet Builder"
- Appendix B, "Troubleshooting Portlets and Providers"
- Appendix D, "Manually Packaging and Deploying PDK-Java Providers"
- Appendix E, "OracleAS Portal Provider Test Suite"
- Appendix F, "Content Management APIs and Views"
- Appendix G, "Content Management Event Framework Events"
For developers who want quick solutions for building components, OracleAS Portal provides the Portlet Builder. The Portlet Builder wizards guide you step-by-step through the process of creating portlets. The wizards enable even the novice business developer to begin creating portlets right away.

After answering a few basic questions, you can decide whether to continue with the wizard or to create the portlet based on information already collected. A Finish button appears as soon as the wizard has collected enough information, allowing you to create the portlet without having to traverse all the wizard’s pages. When you’re ready for other users to run the portlet, you need only assign portlet access privileges, publish the portlet, and add it to a page.

This appendix describes the process of creating a portlet through a wizard and steps you through creating, editing, managing, and running different types of portlets. It contains the following sections:

- Using a Wizard to Build a Portlet
- Editing a Portlet Builder Component
- Managing Portlets
- Managing Versions
- Managing Portlet Security
- Performing Test Runs on a Portlet
- Referencing the OracleAS Portal Schema
- Coding Additional Functionality
- Using Shared Components to Create a Look and Feel
- Example: Building Charts and Reports

A.1 Using a Wizard to Build a Portlet

Portlets must be built under the ownership of a provider. This means that before you can build a portlet, you must first create a provider—or select an existing provider—to host (own or contain) the portlet. If you’re creating a new provider, you may first want to create a schema against which to build the provider.

Although every portlet you create using OracleAS Portal requires that you build it under the ownership of a provider, it is not necessary that you build a provider each time you build a portlet. In fact, you can use an existing provider, or create one dedicated to the ownership of your locally built portlets, and then never have to create another provider again.
One provider can be used as a container that manages multiple portlets. A good rule is to use one provider for each logical application, for example, a Human Resources application that consists of reports, forms, and charts.

This section steps you through the process of creating a schema, creating a provider, and building a portlet under the ownership of that provider in OracleAS Portal. It includes the following subsections:

- Creating a Schema in OracleAS Portal
- Creating a Provider for Locally Built Portlets
- Creating Portlets Using OracleAS Portal Wizards

### A.1.1 Creating a Schema in OracleAS Portal

A schema is an Oracle database user account under the ownership of which you can store database objects, applications, and components and control the schema’s database privileges. Use a Portal schema to enforce password access to the schema, to set aside tablespace specifically for your locally built portlets, and to identify the temporary tablespace the schema should use.

Creating a schema involves three main tasks:

- Creating a Schema
- Granting and Revoking Privileges on Database Objects
- Enrolling the Schema in One or More Roles

This section steps you through each of these tasks.

#### A.1.1.1 Creating a Schema

To create a schema in OracleAS Portal, perform the following steps:

1. Log in to OracleAS Portal.
2. Click the **Navigator** link at the top of the page.
3. In the Navigator, click the **Database Objects** tab to bring it forward.
4. Click the **Schema** link next to **Create New…** (Figure A–1)

![Figure A–1 The Schema Link on the Database Objects Tab in the Portal Navigator](image)

5. On the Create Schema page (Figure A–2), enter a name to identify the schema in the **Schema** field.

   The name must be unique within the portal. Blank and special characters are not allowed in the name. Type an underscore character to add a space in a name. For example, you can name a schema **PORTLETS_SCHEMA**, but not **PORTLETS SCHEMA**. Additionally, you cannot name a schema **PORTLETS_SCHEMA** nor **PORTLETS%SCHEMA**.
6. In the **Password** field, enter the password you will assign to the schema.

   Asterisks appear for each character you enter. Users will enter this password when logging into the Oracle database as this schema.

7. Reenter the password in the **Confirm Password** field.

8. Select a default tablespace from the **Default Tablespace** list.

   This will be used for storing any database objects created by the schema. We recommend that you select a tablespace created specifically for storing locally build portlet applications. If necessary, talk about this with your database administrator.

9. Select a temporary tablespace from the **Temporary Tablespace** list.

   This will be used by the schema to create temporary storage for operations such as sorting table rows.

10. Select an Oracle resource profile for the new schema from the **ORACLE Profile** list.

    The default profile is DEFAULT. Feel free to use this. If you plan to create a new profile, you must use Oracle SQL commands to do so. Refer to the Oracle database documentation for more information. You’ll find this on the Oracle Technology Network at [http://www.oracle.com/technology/index.html](http://www.oracle.com/technology/index.html).

11. Select the **Use this Schema for Portal Users** check box to add the schema you are creating to the list of database schemas to which portal users can map for administrative purposes.

    Every OracleAS Portal user must be associated with a database schema. By default, the name of this schema is `<portal>_public`, where `<portal>` is the name of the schema in which OracleAS Portal is installed. If you select this check box, a portal user can be mapped instead to the schema you are creating here.

12. Click **Create** to save your changes and return to the Portal Navigator.

    Alternatively, click the link to your new schema that now appears at the top of the page to begin editing the new schema’s properties. See Section A.1.1.2, “Granting and Revoking Privileges on Database Objects” and Section A.1.1.3, “Enrolling the Schema in One or More Roles” for more information.
A.1.1.2 Granting and Revoking Privileges on Database Objects

You may need to increase or limit the access your users will have on some database objects associated with this schema. Once they are able to log in to this schema, users’ access privileges are influenced by the roles of which this schema is a member (see Section A.1.1.3, “Enrolling the Schema in One or More Roles”). You can use grants to select specific database objects and specify which of those access privileges should be further limited or increased.

For example, imagine that you have a BASIC_USER role. This role allows users to SELECT from all tables in the database. You may want to restrict selections on some tables. You can use grants to select one or more database objects (such as specific tables, views, and the like) and remove the SELECT privilege. Conversely, if a role does not include all the access privileges you want a schema to have on one or more database objects, you can use grants to increase access privileges.

---

**Note:** By default, OracleAS Portal allows only SELECT on database objects. Other privileges must be granted explicitly.

---

This section describes how to further control privileges on objects associated with your schema.

**Note:** If you have arrived at this task by clicking a link to edit the schema that appeared at the top of the Create Schema page, skip to step 5.

To grant privileges to database objects associated with a schema, perform the following steps:

1. Log in to OracleAS Portal.
2. Click the **Navigator** link at the top of the page.
3. Click the **Database Objects** tab to bring it forward.
4. On the Database Objects tab (Figure A–3), go to the **Name** column to locate the schema you will work with, and click the **Edit** link that is next to it.

*Figure A–3  The Edit Link Next to a Schema in the Portal Navigator*

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Actions</th>
<th>Creator</th>
<th>Last Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema</td>
<td>ANONYMOUS</td>
<td>Edit, Drop, Grant Access</td>
<td>ANONYMOUS</td>
<td>15-SEP-2003 02:26 PM</td>
</tr>
<tr>
<td>Schema</td>
<td>CTXSYS</td>
<td>Edit, Drop, Grant Access</td>
<td>CTXSYS</td>
<td>15-SEP-2003 02:26 PM</td>
</tr>
<tr>
<td>Schema</td>
<td>DESNMP</td>
<td>Edit, Drop, Grant Access</td>
<td>DESNMP</td>
<td>15-SEP-2003 02:26 PM</td>
</tr>
<tr>
<td>Schema</td>
<td>HR</td>
<td>Edit, Drop, Grant Access</td>
<td>HR</td>
<td>15-SEP-2003 02:26 PM</td>
</tr>
<tr>
<td>Schema</td>
<td>MDSYS</td>
<td>Edit, Drop, Grant Access</td>
<td>MDSYS</td>
<td>15-SEP-2003 02:26 PM</td>
</tr>
</tbody>
</table>

5. On the resulting page, click the **Grants** tab (Figure A–4).
6. In the **Object** field, enter the name of a database object on which you want to control privileges.

   Click the Browse icon to select from a list of objects. Prefix the name of the object with the schema that owns it. For example, add SCOTT to EMP (SCOTT.EMP) to indicate the EMP table in the SCOTT schema.

7. Click **Add** to add the object to the list at the bottom of the page.

   Objects are grouped by object type. That is, tables are listed with tables, views with views, and so on.

8. Select or clear check boxes next to the object to increase or limit the access privileges on the object (**Figure A–5**).
A.1.1.3 Enrolling the Schema in One or More Roles

You will want to enroll your schema in one or more roles to provide the desired level of overall database access privileges. If you think of a schema as a super-user, created from a population of users who have password access to the schema, you enroll a schema in one or more roles to provide access privileges to the database to all users who in turn have login access to the schema.

The access privileges provided through the schema’s membership in one or more roles can be overridden on specific database objects through grants (see “Granting and Revoking Privileges on Database Objects”).

Note: If you have arrived at this task by clicking a link to edit the schema that appeared at the top of the Create Schema page or if you are continuing from editing the schema’s Main or Grants tab, skip to step 5.

To enroll a schema in one or more roles, perform the following steps:

1. Log in to OracleAS Portal.
2. Click the Navigator link at the top of the page.
3. Click the Database Objects tab to bring it forward.
4. On the Database Objects tab, go to the Name column to locate the schema you will work with and click the Edit link that is next to it.
5. On the resulting page, click the Roles tab to bring it forward (Figure A–6).
By default, the schema is enrolled in the CONNECT role, and may be enrolled in others.

6. Click the Browse icon next to the **Role** field.

7. Select a role to which you will enroll the schema.

8. Click the **Add** button next to the **Role** field to add the selected role to the **Is Member Of** list.

   Continue selecting roles and clicking the **Add** button until you have enrolled the schema in all desired roles.

9. Click **Apply** to save your changes, or click **OK** to save your changes and return to the Portal Navigator.

**A.1.2 Creating a Provider for Locally Built Portlets**

Once you have created or identified an existing schema that will own your provider, you are ready to create a provider to host (own or contain) your locally built portlets.

To create a provider for locally build portlets, perform the following steps:

1. Log in to OracleAS Portal.

2. Click the **Navigator** link at the top of the page.

3. In the Portal Navigator, click the **Providers** tab to bring it forward.

4. On the Providers tab, click the **Locally Built Providers** link.

5. On the resulting page, click the **Database Provider** link next to **Create New...**
Note: If you do not see the Create New link on the Locally Built Providers page, this means that the user name you used to log in does not have privileges in any schema. The capability to create a new provider is available only when the current user has some privilege in a database schema. In this case, you must have the Portal Administrator edit your user profile to grant you the privilege to create a schema. To do this, perform the following steps:

1. Go to the Portal User Profile portlet on the Administer tab of the Portal Builder page.
2. Enter the user name and click Edit.
3. On the Edit Portal User Profile page, click the Privileges tab to bring it forward.
4. Under Administration Privileges, set All Schemas to Create.
5. Under All Portal DB Providers, set Create.

When these steps are complete, you can log in again with your user name and create a schema on the Database Objects tab in the Portal Navigator. Once you have created a schema, you will see the Create New link on the Locally Built Providers page.

6. In the Portal DB Provider Name field, enter an internal name to identify this provider to the portal.
   Blank characters and special characters are not allowed. Type an underscore character to add a space, for example, portal_provider.
7. In the Portal DB Provider Display Name field, enter a display name to identify this provider to users.
8. Select a schema, for example, the one you just created, from the Portal DB Provider Schema drop-down list.
   This list of schemas includes only those on which you have the Manage privilege.
9. Click OK to save your changes and return to the Portal Navigator.

A.1.3 Exposing a Provider

Before you can create and publish portlets under the auspices of a provider, you must ensure that the provider has been identified as a provider to OracleAS Portal. To expose a provider, perform the following steps:

1. Create a provider, for example, as described in Section A.1.2, "Creating a Provider for Locally Built Portlets".
2. Locate the listing for your new provider in the Portal Navigator.
3. Click the Grant Access link next to your new provider (Figure A–7).
4. On the Grant Access page, verify that Expose as Provider is selected (Figure A–8).

5. Click OK to save your change and exit the Grant Access page.

Selecting this check box enables users to publish portlets to the portlet repository under the auspices of this provider. Provided you have selected this box, the portlets you create using the Portlet Builder will be published to the portlet repository automatically (under the Portlet Staging Area node) when you finish them (Figure A–9).
A.1.4 Creating Portlets Using OracleAS Portal Wizards

This section lists and describes the types of portlets you can build using the portlet-building wizards available in OracleAS Portal. It provides general information on how to build a portlet and specific information on how to build forms, reports, charts, and lists of values.

Beginning with OracleAS Portal 10g (9.0.4), you can define portlets “in place” on a page. In other words, you can add a portlet to a page and define the complete portlet using the Portlet Builder without leaving the page. To add and create portlets in this way, you must have the required privileges: you must belong to the PORTAL_ADMINISTRATORS group and have the MANAGE CONTENT privilege on the page. This section describes how to define portlets outside of the page; then, you can modify and manage portlets independently, and later add the portlet to the page, as described in Section A.6.4, “Running the Component as a Portlet through the Portlet Personalization Form”.

Note: If you are planning to build portlets in an OracleAS Portal database provider and accept the defaults in the wizards, you must ensure that the appropriate privileges have been granted on the objects of the SCOTT schema to the Portal database provider schema. Otherwise, you will receive errors when going through the wizards.

This section contains the following subsections:

- Building Portlets Declaratively
- Building Forms Declaratively
- Building Reports Declaratively
You can build additional types of portlets through the portlet-building wizards. Table A–1 lists and describes the types of portlets you can build.

**Table A–1 Types of Portlets Available Through Portlet-Building Wizards**

<table>
<thead>
<tr>
<th>Portlet Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Form**      | Displays a personalized form that can be used as an interface for updating tables, executing stored procedures, and generating other personalized forms. Through the OracleAS Portal portlet-building wizards, you can build three types of forms.  
  - A form based on a table or view enables users to insert, update, and delete data in a database table or view.  
  - A master-detail form displays a master row and multiple detail rows within a single HTML page. The form contains fields for updating values in two database tables or views.  
  - A form based on a procedure enables users to insert, update, and delete data over a database stored procedure. |
| **Report**    | Displays the data that you select from a database table or view in report format. The report can have tabular, form, or custom layout. You can build three types of reports:  
  - Query By Example (QBE) Report allows users to query, insert, update and delete data in tables and views. In the QBE report build wizard, you choose which data to display in the report. Or, you can allow end users to make their own queries in the QBE report's personalization form.  
  - Reports from Query Wizard guides you through building a report using a wizard to construct your SELECT statement.  
  - Reports from SQL Query builds a report from your manually-constructed SQL query. |
| **Chart**     | Displays data that you select from a database table or view as a bar chart. |
| **Calendar**  | Displays the data that you select from a database table or view in calendar format. |
| **Dynamic Page** | Displays dynamically-generated HTML content on a Web page. |
| **XML Component** | Displays an XML page. |
| **Hierarchy** | Displays the data that you select from a database table or view as a graphical hierarchy of items containing up to three levels. |
| **Menu**      | Displays an HTML-based menu that contains options that are linked to other menus, OracleAS Portal database portlets, or URLs. |
| **URL**       | Renders the content of a URL target in a portlet. |
| **Frame Driver** | Displays a Web page with two frames. The queries entered in one frame control the content of the other. |
| **Link**      | Displays a link that provides a hypertext jump between OracleAS Portal database portlets and other database portlets, database portlet personalization forms, or any HTML page. |
To use a wizard to build a portlet, perform the following steps:

1. Log in to OracleAS Portal.
2. Click the Navigator link at the top of the page.
3. Click the Providers tab to bring it forward.
4. Click the Locally Built Providers link (Figure A–10).

Figure A–10  The Locally Built Providers Link in the Portal Navigator

5. Click the name of the OracleAS Portal database provider you want to host the new portlet.

The Name column displays the names of all the providers on which you have privileges.

6. Next to Create New..., click the link for the type of portlet that you want to build, for example, Chart or Form (Figure A–11).

<table>
<thead>
<tr>
<th>Portlet Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Values</td>
<td>Use LOVs when creating database portlets to preselect the possible values in an entry field. Users select a value from the list rather than enter it manually. You can build LOVs based on other LOVs.</td>
</tr>
<tr>
<td>Data Component</td>
<td>Displays data in a spreadsheet format.</td>
</tr>
</tbody>
</table>

A.1.4.1 Building Portlets Declaratively

**Note:** To build a portlet, you must have at least the Edit privilege on the provider that will own the portlet. For more information, see “Creating a Provider for Locally Built Portlets”.

<table>
<thead>
<tr>
<th>Portlet Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Values</td>
<td>Use LOVs when creating database portlets to preselect the possible values in an entry field. Users select a value from the list rather than enter it manually. You can build LOVs based on other LOVs.</td>
</tr>
<tr>
<td>Data Component</td>
<td>Displays data in a spreadsheet format.</td>
</tr>
</tbody>
</table>

**Table A–1 (Cont.) Types of Portlets Available Through Portlet-Building Wizards**

**Note:** To build a portlet, you must have at least the Edit privilege on the provider that will own the portlet. For more information, see “Creating a Provider for Locally Built Portlets”.

To use a wizard to build a portlet, perform the following steps:

1. Log in to OracleAS Portal.
2. Click the Navigator link at the top of the page.
3. Click the Providers tab to bring it forward.
4. Click the Locally Built Providers link (Figure A–10).

5. Click the name of the OracleAS Portal database provider you want to host the new portlet.

The Name column displays the names of all the providers on which you have privileges.

6. Next to Create New..., click the link for the type of portlet that you want to build, for example, Chart or Form (Figure A–11).
See Table A–1 for a list and description of the types of portlets you can build.

Depending on the type of portlet you are building, OracleAS Portal displays either the first step of the portlet build wizard or a menu of additional choices. If you click Form, for example, you have the option of creating a form based on a table or view, a master-detail form, or a form based on a procedure (Figure A–12).

7. When the first step of the portlet build wizard displays, follow the instructions shown on each step.

A progress indicator displays how many wizard steps are left to complete. The Finish button becomes available when you have provided the wizard with sufficient information to build the portlet. The fields you don’t complete will populate with default values or will remain null.

Note: The progress indicator jumps from Step 1 of n to Step 3 of n. This can be ignored because Step 2 of n displays only when you define a portlet in place on a page. In an in place portlet definition, the wizard starts from Step 2 of n.

If you are unsure of how to complete the fields on a particular step of the wizard, click the online Help icon in OracleAS Portal for more information.

The next sections guide you through the specific steps required to build a particular type of portlet. For more information, see:

- Building Forms Declaratively
A.1.4.2 Building Forms Declaratively

Forms provide a means of inserting, updating, and deleting content from your database through a user-friendly interface. You control form layout through options available in the Forms build wizard. Wizard options enable you to control the look and feel of the overall form as well as the form’s individual entry fields and buttons.

You can perform JavaScript validation on user-specified values entered in any text field on the form. Additionally, you can add PL/SQL event handlers that run when a user clicks a button on the form. After successful submission of form content, you can specify a PL/SQL block or procedure that will execute.

The portlet-building wizards in OracleAS Portal provide three construction methods for building forms:

- Forms based on tables or views
- Master-detail forms
- Forms based on a procedure

All of these forms are described in Table A–1. This section describes how to create a form based on a table or view (Figure A–13).

**Figure A–13  Sample Form Based on a Table**

To create a form based on a table or view, perform the following steps:

1. Follow the instructions detailed in "Building Portlets Declaratively". Return to this section once you complete step 6.
2. On the Forms page, click **Form based on table or view** (Figure A–14).
3. In the **Name** field, enter an internal name for the form (Figure A–15).

4. In the **Display Name** field, enter the name by which users will identify this form (Figure A–15).

   The display name is used at runtime. It displays as the title of the browser window as well as in the banner of the form. When the form is published as a portlet, the display name is used as the title of the portlet and in the List of Available Portlets in the Portlet Repository.

5. In the **Description** field, enter a description of this form (Figure A–15).

   The description displays below the portlet in the Portlet Repository. It can be a summary of the portlet's purpose, a classification of its type, or any other descriptive information. It may be useful within your organization to have a standard approach to what is included in the description.

6. From the list of **Portal DB Providers**, select the provider that will own this form (Figure A–15).
We recommend that you use a provider specifically created for owning, or hosting, your declaratively-built portlets. This list displays only those providers in which you have been granted privileges to build portlets.

7. Click Next.

8. Click the Browse icon next to the Table or View field, and choose a table or view on which to base your form (Figure A–16).

You can use a constant in this field (for example, #APP_SCHEMA#.EMP, #PORTAL_SCHEMA#.DEPT) in lieu of a fixed value. For more information, see "Referencing the OracleAS Portal Schema".

Figure A–16  Selecting a Table or View

The tables and views that you can choose in this step are those that are accessible from the Portal DB Provider you selected in the previous step. You can also enter the table name directly into the field.

Note: If you are using the Example Application (based on the PORTAL_DEMO schema) to build a form, the default table on which to base the form is SCOTT.EMP. As the SCOTT table does not have PUBLIC EXEC privileges, the administrator must manually grant public access to the SCOTT schema to create a form based on the default table. If this is not done, and you choose the default table, the following error is generated:

Error: This table does not exist or you do not have the require privileges (WWV-13020).

9. Click Next.

10. Select a form layout.

Choose between Tabular or Custom. Tabular layouts are created automatically. Custom layouts are based on HTML code that you supply in a later step.
11. Click Next.
12. Set Formatting and Validation Options (Figure A–17) as follows:

**Figure A–17  Form Formatting and Validation Options**

![Form Formatting and Validation Options](image)

- Click **Form** in the left column to set up the formatting and validation options for the entire form (Figure A–17).

  The formatting and validation options that appear in the right column relate to the object you have selected in the left column.

  **Table A–2** lists and describes form-level formatting and validation options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form Background Color</td>
<td>Choose the background color of the form.</td>
</tr>
<tr>
<td>Form Background Image</td>
<td>Choose an image that will appear in the background of the form.</td>
</tr>
<tr>
<td>Form Border</td>
<td>Choose a style for the border around the form background.</td>
</tr>
<tr>
<td>Log Activity Check Box</td>
<td>Select to record activity in the OracleAS Portal log. Log information includes performance statistics as well as the names of users who request the form.</td>
</tr>
<tr>
<td>Show Timing Check Box</td>
<td>Select to display the time elapsed, starting from when the server receives the request to generate HTML for the form. Timing displays at the bottom of the form.</td>
</tr>
<tr>
<td>Order by</td>
<td>Choose the column by which you will order the data returned by the SELECT statement. Select the percent sign (%) if you do not wish to specify a column for ordering.</td>
</tr>
<tr>
<td>Then by</td>
<td>Choose the column(s) for subsequent ordering of the data returned by the SELECT statement. Select the percent sign (%) if you do not wish to specify a column for ordering.</td>
</tr>
<tr>
<td>Ascending</td>
<td>Choose Ascending (from a to z/0 to 9) or Descending (from z to a/9 to 0), depending on how you want the column values ordered.</td>
</tr>
</tbody>
</table>
**Table A–2** (Cont.) Form-Level Formatting and Validation Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>On Successful Submission of a Form Execute</td>
<td>Enter optional PL/SQL code that will execute after a user clicks a button on the form. The button must cause an operation, such as INSERT, to be performed on the table or view on which the form is based. For example, you might enter code that causes a message to display to the user when a table row is successfully updated.</td>
</tr>
</tbody>
</table>

b. Select a TOP_SECTION item in the left column (that is, INSERT_TOP, UPDATE_TOP, and the like) to set formatting and validation options in the right column for INSERT, UPDATE, and other like items that will display at the top of the form (Figure A–18).

**Figure A–18 Button Formatting and Validation Options**

By default, insert, update, delete, query, and reset buttons appear at the top and bottom of each form. To prevent these buttons from displaying, click the Delete icon that appears next to the item in the left column.

Table A–3 lists and describes the formatting and validation options for buttons.
### Table A–3 Formatting and Validation Options for Buttons

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Item Type</strong></td>
<td>Select the method for displaying the selected item type on your form. Note that the default for items, such as INSERT, UPDATE, DELETE, QUERY, and RESET, you will likely most often want to use the default. If you choose another item type, you will need to add code to invoke the item’s functionality (for example, an insert) through a JavaScript or PL/SQL Event Handler. Item types include:</td>
</tr>
<tr>
<td>■ Blank:</td>
<td>Inserts a line break in between fields.</td>
</tr>
<tr>
<td>■ Button:</td>
<td>Inserts a button. INSERT, UPDATE, DELETE, QUERY, and RESET items require the selection of an event handler further down on this page.</td>
</tr>
<tr>
<td>■ CheckBox:</td>
<td>For INSERT, UPDATE, and so on, you must write a JavaScript handler for this to be meaningful. For columns, the underlying column must be at least VARCHAR2(5) to accommodate entries of TRUE (selected) and FALSE (cleared).</td>
</tr>
<tr>
<td>■ ComboBox:</td>
<td>Inserts a pop-list of values with an extra blank space that allows for manual entry of a value. You must already have defined a list of values to make this selection. You can enter a constant in the Default Value field, and define the default value as a constant in the Default Value Type field. For more information about the constants you can use, see &quot;Referencing the OracleAS Portal Schema&quot;.</td>
</tr>
<tr>
<td>■ File Upload (Binary):</td>
<td>Used to insert a BLOB column.</td>
</tr>
<tr>
<td>■ File Upload (interMedia):</td>
<td>Select this type for interMedia rich content, such as images, audio clips, and video clips. The table must have columns of the type ORDIMAGE, ORDAUDIO, or ORDVIDEO. For more information, see &quot;Building Forms and Reports against interMedia Rich Content&quot;.</td>
</tr>
<tr>
<td>■ Hidden:</td>
<td>Creates a hidden form field.</td>
</tr>
<tr>
<td>■ Horizontal Rule:</td>
<td>Draws a horizontal line, similar to &lt;hr&gt;.</td>
</tr>
<tr>
<td>■ Image:</td>
<td>Used to show an image.</td>
</tr>
<tr>
<td>■ Label Only:</td>
<td>This shows only a prompt.</td>
</tr>
<tr>
<td>■ Password:</td>
<td>This will have the behavior of a password field, where input will be hidden.</td>
</tr>
<tr>
<td>■ TextArea:</td>
<td>Provides a large text area for the entry of multiple lines of text.</td>
</tr>
<tr>
<td>■ TextBox:</td>
<td>Provides a text box.</td>
</tr>
<tr>
<td><strong>Display Options</strong></td>
<td><strong>Label:</strong> Enter text that will display next to the button.</td>
</tr>
<tr>
<td><strong>Layout Options</strong></td>
<td>This section displays only if you selected Tabular in the Form Layout step. If you selected Custom, you can specify your own sophisticated layout in the next step using HTML code.</td>
</tr>
<tr>
<td><strong>Begin on New Line:</strong></td>
<td>Select this option to display the item on a new line on the form. Leaving this cleared will display the item on the same line as the previous item or column field.</td>
</tr>
<tr>
<td><strong>JavaScript Event Handlers</strong></td>
<td>Choose an event and enter the JavaScript for the action you want to occur when that event happens. Refer to your JavaScript documentation for descriptions of the events on this list.</td>
</tr>
<tr>
<td><strong>PL/SQL Button Event Handler</strong></td>
<td>Choose a button event and enter the PL/SQL code for the action you want to occur when the event happens. For advanced personalization, choose Custom from the drop-down list, and enter the PL/SQL code for the custom event.</td>
</tr>
</tbody>
</table>
c. Select a column name in the left column to set formatting and validation options in the right column for table or view columns that will display in the form (Figure A–19).

**Figure A–19  Column Formatting and Validation Options**

To prevent the display of a column, click the **Delete** icon next to it.

Table A–4 lists and describes the formatting and validation options available for table or view columns.

**Table A–4  Formatting and Validation Options for Table or View Columns**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item Type</td>
<td>Defines how the column you've selected on the left will display on your form. Options are listed and described in Table A–3.</td>
</tr>
</tbody>
</table>
Display Options

**Label**: Enter text to label this column on the form. Text will be display-only and will appear next to the field that represents this table or view column on the form. For example, you can add a label next to the field for the EMPNO column called Employee Identification Number.

**Link**: Specify a link to another OracleAS Portal portlet or URL. If you specify this option, the Label appears on the form as a hypertext link. If you want to link the form to another URL, type the URL location in the Link text box. If you want to link to an OracleAS Portal portlet, you can type the name of the package containing the portlet, for example:

    SCHEMA.portlet.SHOW

SCHEMA is the name of the schema that owns the portlet; portlet is the portlet name; and SHOW is the procedure used to display the portlet. You can also specify SHOW_PARMS to display the personalization form for the portlet.

**Font Face, Color, Size**: Specify the font characteristics for displaying text associated with the selected column on the form.

**Input Width**: Enter the character width of the field associated with the selected column.

**Input Max Length**: Enter the maximum length of the data that can be entered into the form field associated with the selected column.
Validation Options

**Mandatory:** Select to require that the user specify a value in the field associated with the selected column before submitting the form.

**Updatable:** Select to enable the user to update the column. Leaving this blank prevents updates. This feature is useful when you want to allow users to update some table columns in the form, but only view others. For example, by clearing **Updatable** for employee names and ID numbers, and selecting **Updatable** for the employee's department, you can create a form that enables users to update an employee’s department (for example, when the employee is transferred), but disables users from updating the employee name and ID number.

When users display views, updatable columns display in black, non-updatable columns display in blue, and mandatory columns display in red.

**Insertable:** Select to store the value of the column in the database when the user creates a new record. Leaving this blank prevents the value from being stored in the database (that is, it removes the column from the INSERT statement).

**Default Value:** Enter a default value for the field associated with the selected column. Users can accept this value or specify their own. Specify a constant, function, expression, or SQL query.

**Default Value Type:** Specify a type for the default value entered in the previous field (**Default Value**).

**Format Mask:** Enter an Oracle display format for columns containing numeric and date data types. For example, you could enter **DD/MM/YYYY** to display dates according to this pattern; or you could enter **999,999,999.99** to place commas and decimals according to this pattern.

**Note:** Refer to the Oracle database documentation ([http://www.oracle.com/technology/index.html](http://www.oracle.com/technology/index.html)) for information about date and numeric formatting options.

**Field Level Validation:** Choose a JavaScript validation routine that verifies whether the user enters a valid value in the field. For example, you could choose a JavaScript routine called **IsNumber** that verifies that a number has been typed in a SALARY field. Field validation providers are implemented in JavaScript and run when the OnChange condition occurs; for example, when the user presses the Return key after entering a value in the field.

**Form Level Validation:** Choose a JavaScript validation provider that verifies whether the user enters a valid value in the field. Form validation providers run when the user submits the information, for example, after the user clicks the Insert button on the form.
Select a BOTTOM_SECTION item in the left column (that is, PREVIOUS, NEXT, INSERT_BOTTOM, UPDATE_BOTTOM, and the like) to set formatting and validation options in the right column for PREVIOUS, NEXT, INSERT, UPDATE, and other like items that will display at the bottom of the form. See step b of this task for more information.

13. Click Next.

14. If you selected a Custom layout:

Enter the HTML code that will control the layout of your form. The text boxes on this page contain sample HTML code that creates a table. Using this model, each column you selected in the Tables or Views step of this wizard is formatted to display in a table row on the form.

Column labels are specified by the suffix .LABEL in the sample code. Column values are specified by the suffix .ITEM in the sample code.

You can update the sample code with any HTML of your own, provided you do not change the column values (.ITEM suffix) or labels (.LABEL suffix). You can, however, delete column values from this code.

If you selected a Tabular layout:

Enter descriptive text that you want to appear on the top or bottom of the form (Figure A–20). You can add a form title and help text for the form. You can also choose a template that controls the look and feel of the page on which the form appears.
Figure A–20  Entering Header, Footer, and Other Text for a Form

Table A–5 lists and describes the options available on this page.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template</td>
<td>Choose a template to set the look and feel of form elements such as background colors and images and the image that appears in the upper left corner of the page. Templates are used only when your form is displayed in full-page view, and not when it is displayed as a portlet. When a form is displayed as a portlet, the host page's style controls the look and feel.</td>
</tr>
<tr>
<td>Preview Template</td>
<td>Click to view the appearance of the template currently selected in the Template drop-down list.</td>
</tr>
<tr>
<td>Display Name</td>
<td>Edit the display name of the form. You can specify HTML in this field.</td>
</tr>
<tr>
<td>Header Text</td>
<td>Enter any introductory text that you want to display at the top of the form, just below the title, but above any buttons. You can specify HTML in this field.</td>
</tr>
<tr>
<td>Footer Text</td>
<td>Enter any text that you want to display at the bottom of the form. You can specify HTML in this field.</td>
</tr>
</tbody>
</table>
15. Click Next.

16. Optionally, enter the PL/SQL that will run at different points during the execution of the HTML code that creates this form (Figure A–21).

In the PL/SQL you enter into these fields, you can use constants for the OracleAS Portal schema, the application schema, and the application name. For more information, see "Referencing the OracleAS Portal Schema".

**Figure A–21** Optional Fields for Entering Additional PL/SQL Code

Table A–6 lists and describes the options on this page.

**Table A–6** Additional PL/SQL Code Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>… before displaying the page.</td>
<td>Enter a PL/SQL procedure that will execute before the HTML for the form is generated (that is, before the &lt;FORM&gt; tag is generated). In spite of the name of this field, the PL/SQL procedure actually executes after the page itself is displayed.</td>
</tr>
</tbody>
</table>
Table A–6  (Cont.)  Additional PL/SQL Code Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>… before displaying the form.</td>
<td>Enter a PL/SQL procedure that will execute after the &lt;FORM&gt; tag but before displaying any form elements.</td>
</tr>
<tr>
<td>… after displaying the form.</td>
<td>Enter a PL/SQL procedure that will execute after all form elements are displayed but before the ending &lt;/FORM&gt; tag.</td>
</tr>
<tr>
<td>… after displaying the page.</td>
<td>Enter a PL/SQL procedure that will execute after the ending &lt;/FORM&gt; tag.</td>
</tr>
<tr>
<td>… before processing the form.</td>
<td>Enter a PL/SQL procedure that will execute before the form is processed.</td>
</tr>
<tr>
<td>… after processing the form.</td>
<td>Enter a PL/SQL procedure that will execute after the form is processed.</td>
</tr>
</tbody>
</table>

**Note:** If you want to display output from a PL/SQL procedure, you must enter the PL/SQL in one of these four fields:
- before displaying the page
- before displaying the form
- after displaying the form
- after displaying the page

You cannot display output from a PL/SQL procedure in the before/after processing the form fields; the output will not display. Additionally, if you want to display output from your PL/SQL procedure, you must specify the schema name with the procedure. This is required even if the form and the procedure are in the same schema.

17. Click **Finish** to save your changes.

Once you click **Finish**, Portal jumps to a summary page from which you can manage your form (**Figure A–22**). These tasks are discussed in later sections in this appendix.
18. Click Close to return to the Portal Navigator.

A.1.4.3 Building Reports Declaratively

Reports display data from tables and views (Figure A–23). You can base a report on multiple tables or views using a JOIN condition. You can highlight data that satisfies conditions you specify. For example, you can display in bold red text the records of all employees who have been employed less than two years. You can create hypertext links from values displayed in the report to other OracleAS Portal database portlets or URLs.

In addition, the wizards enable you to specify other options to achieve the following results:

- Limit data displayed in the report
Using a Wizard to Build a Portlet

- Sum column values
- Use logical operators to select data within the columns
- Format data in the report
- Format the report's personalization form
- Specify PL/SQL code that executes at various points in the report.

The OracleAS Portal Portlet Builder wizards provide three construction methods for building reports:

- Query By Example (QBE) Reports
- Reports From Query Wizard
- Reports From SQL Query

These reports are described in Table A–1. This section describes how to build a report using the Query Wizard.

To build a report using the Query Wizard, perform the following steps:

1. Follow the instructions detailed in "Building Portlets Declaratively". Return to this section once you complete step 6.

2. Click the Report link next to Create New… (Figure A–24).

![Figure A–24 The Report Link on the Providers tab in the Portal Navigator](image)

3. On the Reports page, click Reports From Query Wizard (Figure A–25).

![Figure A–25 The Reports From Query Wizard Link in the Portlet Builder](image)

4. In the Name field, enter an internal name for the report (Figure A–26).

A-28 Oracle Application Server Portal Developer's Guide
5. In the **Display Name** field, enter a display name for the report (Figure A–26).

The display name is used at runtime. It displays as the title of the browser window as well as in the banner of the report. When the report is published as a portlet, the display name is used as the title of the portlet and in the List of Available Portlets in the Portlet Repository.

6. From the list of **Portal DB Providers**, select the provider that will own this report (Figure A–26).

   We recommend that you use a provider specifically created for owning, or hosting, your declaratively-built portlets. This list displays only those providers in which you have been granted privileges to build portlets.

7. Click **Next**.

8. From the **Tables and Views** list, choose one or more tables or views on which to base this report (Figure A–27).

   **Note:** In the **Tables and Views** field, you can use constants for the OracleAS Portal schema and the application schema (for example, enter #APP_SCHEMA#.EMP). For more information, see "Referencing the OracleAS Portal Schema".
9. Click **Add** after each selection (Figure A–28).

10. Click **Next**.

11. If you are basing your report on multiple tables or views, identify the columns that will be joined to one another.

   This page displays only when you have selected multiple columns or views in the previous step. Default join conditions are automatically generated. You can accept or modify these defaults.

12. Click **Next**.

13. Select the columns you want to display in your report (Figure A–29).
Figure A–29  Selecting the Columns That Will Display in the Report

Click the arrows between the two columns to move individual selections or the entire list from one column to the other. Use the rearrange icons to the right of the Selected Columns list to set the order of appearance of your selected columns. Select no more than 255 columns.

14. Click Next.

15. Optionally, specify conditions that limit the data displayed in the report (Figure A–30).

Figure A–30  Setting Column Conditions

Table A–7 lists and describes the options on this page.
16. Click Next.

17. Select from three report layout types: Tabular, Form, or Custom.

- **Tabular** layouts are based on options you choose in the Report Building wizard. Results are displayed in a table format, where each record represents a row in the underlying table or view, and each data column displays as a column in the table.

- **Form** layouts are based on options you choose in the Report Building wizard. Results are displayed in rows, where each data column is displayed in a separate row.

- **Custom** layouts are based on HTML code that you supply in a later wizard step. Because you specify your own HTML code, you will have a greater level of control over the appearance of your report than you would using a structured (tabular or form) layout.

This description will proceed as if you have selected a tabular layout. Some of the wizard steps you would take with other types of layouts are not included in the construction of a tabular report. For example, with a custom layout, you would be given an opportunity to specify your custom HTML code. That step does not display in the wizard when you choose a tabular layout.
18. Click Next.

19. Specify column formatting (Figure A–31).

**Figure A–31  The Column Formatting Page**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column</td>
<td>Displays the names of the table or view columns you selected in a previous wizard step.</td>
</tr>
<tr>
<td>Column Heading</td>
<td>Enter the name you will use to identify this column in your report. For example, if your table has an EMPNO column, you can enter Employee ID Number, in lieu of the column’s actual name.</td>
</tr>
<tr>
<td>Sum</td>
<td>Select to add values together and display the result in the report.</td>
</tr>
<tr>
<td>Data Type</td>
<td>Displays the data type of the data in the associated column.</td>
</tr>
<tr>
<td>Align</td>
<td>Choose whether to align data to the left, center, or right margin of a report column. By default, numeric data aligns to the right, and alphabetical data aligns to the left.</td>
</tr>
<tr>
<td>Display as</td>
<td>Choose from:</td>
</tr>
<tr>
<td></td>
<td>- Text—Displays the value in the report without interpreting any associated HTML tags. For example, the value &lt;i&gt;Name&lt;/i&gt; displays as Name.</td>
</tr>
<tr>
<td></td>
<td>- HTML—Displays the value in the report using associated HTML tags. For example, the value &lt;i&gt;Name&lt;/i&gt; displays as Name.</td>
</tr>
<tr>
<td></td>
<td>- Hidden—Hides the value from view in the report. For example, choose Hidden for columns whose values are included for link parameters that you do not want displayed in the report.</td>
</tr>
<tr>
<td>Format Mask</td>
<td>Enter an Oracle display format for columns that contain date and numeric data types. See Table A–9 and Table A–10 for examples of date and number format masks. You cannot use format masks in reports generated in Excel format.</td>
</tr>
<tr>
<td>Link</td>
<td>Create a link from the column to another OracleAS Portal portlet. The column value will display as a hypertext link. You can specify a link only if one has already been created for this column and has been stored in the database.</td>
</tr>
</tbody>
</table>

Table A–8 lists and describes the options presented on this page.

**Table A–8  Column Formatting Options for a Tabular Report**
Using a Wizard to Build a Portlet

Edit Link Click to display the Set Link Parameters window. In this window, you can set the values of the link parameters to pass to the target portlet. Enter a static value, such as 10 or KING, or select from the list of columns you identified in the table or view columns step earlier in this wizard. If you need the value of a column for a link parameter, but do not want that column displayed in the report, you can set the Display as value to Hidden.

Width Type
Specify one of the following:

- **Char**—Displays the output in the specified number of characters in each line. For example, if you enter 20 in **Width**, the report displays 20 characters of the column data in each line. If the number of characters exceeds the specified number, the remaining characters are wrapped to the next line. Note, however, that if the output is ASCII format, Char is based on the column width.

- **Pixel**—Displays the output in the specified number of pixels in each row. For example, if you enter 10 in **Width**, the column data displays 10 pixels of data in each line.

- **Percent**—Displays the output in a column that takes up the specified percentage of the table. For example, if you enter 25 in **Width**, the column data displays in a column that takes up 25 percent of the overall table width.

Width
Enter the numeric value for the width type.

---

### Table A–8 (Cont.) Column Formatting Options for a Tabular Report

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Link</td>
<td>Click to display the Set Link Parameters window. In this window, you can set the values of the link parameters to pass to the target portlet. Enter a static value, such as 10 or KING, or select from the list of columns you identified in the table or view columns step earlier in this wizard. If you need the value of a column for a link parameter, but do not want that column displayed in the report, you can set the Display as value to Hidden.</td>
</tr>
</tbody>
</table>

---

### Table A–9 Examples of Date Format Masks

<table>
<thead>
<tr>
<th>Sample Date Format</th>
<th>Date Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>MM/DD/RRR</td>
<td>03/04/85</td>
</tr>
<tr>
<td>Mon. DD, RRRR</td>
<td>Mar. 4, 1985</td>
</tr>
<tr>
<td>Day Month DD fmHH:MI AM</td>
<td>Monday March 4 11:35 PM</td>
</tr>
<tr>
<td>Dy Mon ddth fmHH24:MI:SS</td>
<td>Mon Mar 4th 23:35:22</td>
</tr>
<tr>
<td>Day &quot;the&quot; ddthsp &quot;of&quot; Month</td>
<td>Monday the fourth of March</td>
</tr>
</tbody>
</table>

### Table A–10 Examples of Number Format Masks

<table>
<thead>
<tr>
<th>Sample Number Format</th>
<th>Number</th>
<th>Number Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0000</td>
<td>7934</td>
<td>&quot;7934&quot;</td>
</tr>
<tr>
<td>-00000</td>
<td>7934</td>
<td>&quot;07934&quot;</td>
</tr>
<tr>
<td>-NNNN</td>
<td>7639</td>
<td>&quot;7639&quot;</td>
</tr>
<tr>
<td>-NNNN</td>
<td>535</td>
<td>&quot;535&quot;</td>
</tr>
<tr>
<td>+NNNN</td>
<td>100</td>
<td>&quot;100&quot;</td>
</tr>
<tr>
<td>+NNNN</td>
<td>.99</td>
<td>&quot;0.99&quot;</td>
</tr>
<tr>
<td>-%NN</td>
<td>10</td>
<td>&quot;%10&quot;</td>
</tr>
<tr>
<td>$&lt;NNNNNN.00&gt;</td>
<td>120014</td>
<td>&quot;$1200.14&quot;</td>
</tr>
<tr>
<td>+KKNNNNNN.00</td>
<td>1950</td>
<td>&quot;+ 1950.00&quot;</td>
</tr>
</tbody>
</table>

---

A-34 Oracle Application Server Portal Developer's Guide
The quotation marks in Table A–10 will not appear in your output. They are used here to make it clear where there are leading spaces. See the Oracle Application Server documentation for more information on format masks. You’ll find it on the Oracle Technology Network, http://www.oracle.com/technology/index.html.

20. Click Next.

21. Optionally, specify conditions and define special formatting for displayed report data that will be used whenever those conditions are met.

For example, specify that the background color of a particular row should become yellow whenever the JOB column equals PRESIDENT (Figure A–32).

Table A–10 (Cont.) Examples of Number Format Masks

<table>
<thead>
<tr>
<th>Sample Number Format</th>
<th>Number</th>
<th>Number Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ &lt;NNNNN.00&gt;</td>
<td>1200</td>
<td>&quot;$1200.00&quot;</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>&quot;$400.00&quot;</td>
</tr>
</tbody>
</table>

Table A–11 lists and describes the options on this page.

Table A–11 Options on the Formatting Conditions Page of the Report Wizard

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column</td>
<td>Choose a column on which the condition will be applied. The column must be one of those selected for display in an earlier wizard step.</td>
</tr>
</tbody>
</table>
22. Click Next.

23. Define display options for the various possible views of the report:

### Table A–11 (Cont.) Options on the Formatting Conditions Page of the Report Wizard

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition</td>
<td>Choose conditions that will define Column’s relationship to Value. Choose from:</td>
</tr>
<tr>
<td></td>
<td>■ equal to (=)</td>
</tr>
<tr>
<td></td>
<td>■ greater than (&gt;)</td>
</tr>
<tr>
<td></td>
<td>■ greater than or equal to (&gt;=)</td>
</tr>
<tr>
<td></td>
<td>■ less than (&lt;)</td>
</tr>
<tr>
<td></td>
<td>■ less than or equal to (&lt;=)</td>
</tr>
<tr>
<td></td>
<td>■ like</td>
</tr>
<tr>
<td></td>
<td>■ not equal to (!=)</td>
</tr>
<tr>
<td></td>
<td>■ null</td>
</tr>
<tr>
<td></td>
<td>■ not null</td>
</tr>
<tr>
<td></td>
<td>■ in</td>
</tr>
<tr>
<td></td>
<td>■ not in</td>
</tr>
<tr>
<td>Value</td>
<td>Enter a column value to trigger the condition. There is no need to apply formatting masks to numeric values. For example, in a table with an EMPNO column, if you want to apply special formatting to EMPNO values greater than 3000, select &gt; for the Condition, and enter 3000 in this field. To specify multiple values after an IN or NOT IN condition, type a colon (:) between each value, for example, 10:20:30.</td>
</tr>
<tr>
<td>Row/Col</td>
<td>Choose whether to apply the conditional formatting to an entire row or to just the affected column data. If you specify a condition, you must choose Row or a column name here.</td>
</tr>
<tr>
<td>Color</td>
<td>Choose a text color for displaying data that meets the specified condition.</td>
</tr>
<tr>
<td>Background Color</td>
<td>Choose a background color for the column or row that contains data that meets the specified condition. In some cases, the background color does not change as expected. This is a known issue with reports.</td>
</tr>
<tr>
<td>Font Face</td>
<td>Choose a font in which to display data that meets the specified condition.</td>
</tr>
<tr>
<td><strong>A</strong></td>
<td>Select to display results that meet the specified condition in bold.</td>
</tr>
<tr>
<td><strong>I</strong></td>
<td>Select to display results that meet the specified condition in italic.</td>
</tr>
<tr>
<td><strong>U</strong></td>
<td>Select to underscore results that meet the specified condition.</td>
</tr>
<tr>
<td>Blink</td>
<td>Select to cause results that meet the specified condition to blink on and off.</td>
</tr>
<tr>
<td>Seq</td>
<td>Enter a sequential number to establish an order of precedence for execution of the formatting. This is useful, for example, when a row contains data that meets more than one condition, and the formatting rules differ for each. The condition with the highest position in the order (that is, 1), takes precedence over all the others.</td>
</tr>
</tbody>
</table>
Creating Portlets with the Portlet Builder

Table A–12 describes display options common to all views (Figure A–33).

Table A–13 describes display options for full-page views (Figure A–34).

Options for portlet views are the same as those for full-page views, with a few noted exceptions. See Table A–13 and Figure A–34.

Table A–14 describes display options for column breaks (Figure A–35).

Table A–15 describes display options for row order (Figure A–36).

Table A–16 describes display options for mobile devices (Figure A–37).

Figure A–33  Display Options Common to All Views

Common Options
Choose common options that affect the appearance of the report displayed in portlet mode and in a full Web page.

- Show Total Row Count
- Embed interMedia rich content in the report

Expire After (minutes) 0
Default Format HTML

Table A–12  Report Display Options Common to All Views

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Total Row Count</td>
<td>Select to display the total number of rows in the report.</td>
</tr>
<tr>
<td>Show Null Values as</td>
<td>Enter the text string you want to display for all null values in the report, for example, (null).</td>
</tr>
<tr>
<td>Embed interMedia rich content</td>
<td>Select to generate report output that shows any interMedia data values, such as audio, video, or images.</td>
</tr>
<tr>
<td>content in the report</td>
<td>For more information, see 'Building Forms and Reports against interMedia Rich Content'.</td>
</tr>
<tr>
<td>Expire After (minutes)</td>
<td>Enter the number of minutes after the initial display to keep the report in the cache. When the time expires, the report data is refreshed from the database.</td>
</tr>
<tr>
<td>Default Format</td>
<td>Choose a display format for the report:</td>
</tr>
<tr>
<td></td>
<td><strong>HTML</strong>—Formats the report using HTML tables, and displays output on a new page in the Web browser. Portlets that contain large amounts of data may take longer to display in this format.</td>
</tr>
<tr>
<td></td>
<td><strong>Excel</strong>—Formats the report for display in Microsoft Excel. When running the finished report, your local Excel settings may make it necessary for you to first save the report locally before it can be opened and viewed.</td>
</tr>
<tr>
<td></td>
<td><strong>ASCII</strong>—Formats the report using the HTML PRE tag to display heading and values in the report as ASCII text. This option is useful for displaying large amounts of data.</td>
</tr>
</tbody>
</table>
**Figure A–34  Report Display Options for Full Web-Page and Portlet Views**

**Full Page Options**
Choose options that affect the appearance of the report displayed in a full Web page.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Font Face</td>
<td>Specify a font for report text.</td>
</tr>
<tr>
<td>Font Color</td>
<td>Specify a font color for report text.</td>
</tr>
<tr>
<td>Size</td>
<td>Specify a font size for report text. Use relative values, such as +1, +2, and so on.</td>
</tr>
</tbody>
</table>

Relative font size is the last font size specified in the HTML code for the page plus the relative value. For example, the last font size specified in HTML code is 14 points; a new heading is coded at +2 font size; the heading will display in 16-point type.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Border</td>
<td>Specify the thickness of border to display around the report.</td>
</tr>
<tr>
<td></td>
<td>Choose <strong>No Border</strong>, <strong>Thin Border</strong>, or <strong>Thick Border</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heading Background Color</td>
<td>Specify a background color for report column headings.</td>
</tr>
<tr>
<td>Table Row Color(s)</td>
<td>Specify a background color for report rows. To choose more than one color use CTRL-Click (Windows). Rows will display in alternating colors.</td>
</tr>
<tr>
<td>Maximum Rows Per Page</td>
<td>Enter the maximum number of rows you want to display in the report.</td>
</tr>
<tr>
<td>Draw Lines Between Rows</td>
<td>Select to display lines between report rows.</td>
</tr>
<tr>
<td>Log Activity</td>
<td>(Full page only.) Check to enter performance information and the names of users who request the report in the OracleAS Portal activity log.</td>
</tr>
<tr>
<td>Show Timing</td>
<td>(Full page only.) Check to display report timing at the bottom of the report. Timing runs from when the server receives the request to generate the report to when the HTML for the report is generated.</td>
</tr>
</tbody>
</table>

**Table A–13  Report Display Options for Full Web-Page and Portlet Views**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Font Face</td>
<td>Specify a font for report text.</td>
</tr>
<tr>
<td>Font Size</td>
<td>Specify a font size for report text. Use relative values, such as +1, +2, and so on.</td>
</tr>
</tbody>
</table>

Relative font size is the last font size specified in the HTML code for the page plus the relative value. For example, the last font size specified in HTML code is 14 points; a new heading is coded at +2 font size; the heading will display in 16-point type.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Border</td>
<td>Specify the thickness of border to display around the report. Choose <strong>No Border</strong>, <strong>Thin Border</strong>, or <strong>Thick Border</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heading Background Color</td>
<td>Specify a background color for report column headings.</td>
</tr>
<tr>
<td>Table Row Color(s)</td>
<td>Specify a background color for report rows. To choose more than one color use CTRL-Click (Windows). Rows will display in alternating colors.</td>
</tr>
<tr>
<td>Maximum Rows Per Page</td>
<td>Enter the maximum number of rows you want to display in the report.</td>
</tr>
<tr>
<td>Draw Lines Between Rows</td>
<td>Select to display lines between report rows.</td>
</tr>
<tr>
<td>Log Activity</td>
<td>(Full page only.) Check to enter performance information and the names of users who request the report in the OracleAS Portal activity log.</td>
</tr>
<tr>
<td>Show Timing</td>
<td>(Full page only.) Check to display report timing at the bottom of the report. Timing runs from when the server receives the request to generate the report to when the HTML for the report is generated.</td>
</tr>
</tbody>
</table>
Table A–13  (Cont.) Report Display Options for Full Web-Page and Portlet Views

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Query Conditions</td>
<td>(Full page only.) Check to display user-specified parameters that are passed to the query that created the report and the time when the report was created.</td>
</tr>
</tbody>
</table>

Figure A–35  Report Display Options for Column Breaks

Break Options
Choose whether to break the report on values in one, two or three column values:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Break Style</td>
<td>Left Break</td>
</tr>
<tr>
<td>First Break Column</td>
<td>%</td>
</tr>
<tr>
<td>Second Break Column</td>
<td>%</td>
</tr>
<tr>
<td>Third Break Column</td>
<td>%</td>
</tr>
</tbody>
</table>

Table A–14  Report Display Options for Column Breaks

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Break Style</td>
<td>Choose a break style.</td>
</tr>
<tr>
<td>First Break Column</td>
<td>Choose the first column on which to break. For example, if you choose DEPTNO, the report will show all the rows associated with the first department number, followed by all the rows associated with the next department number, and so on. If you specify break columns, then you must also specify Order by values for the columns in the same order (see Table A–15).</td>
</tr>
<tr>
<td>Second Break Column</td>
<td>Choose the second column on which to break. If you do not require a second level break, choose %.</td>
</tr>
<tr>
<td>Third Break Column</td>
<td>Choose the third column on which to break. If you do not require a third level break, choose %.</td>
</tr>
</tbody>
</table>

Figure A–36  Report Display Options for Row Order

Row Order Options
Choose the columns whose values will be used to sort rows in the report:

<table>
<thead>
<tr>
<th>Order by</th>
<th>%</th>
<th>Ascending</th>
</tr>
</thead>
<tbody>
<tr>
<td>then by</td>
<td>%</td>
<td>Ascending</td>
</tr>
<tr>
<td>then by</td>
<td>%</td>
<td>Ascending</td>
</tr>
<tr>
<td>then by</td>
<td>%</td>
<td>Ascending</td>
</tr>
<tr>
<td>then by</td>
<td>%</td>
<td>Ascending</td>
</tr>
<tr>
<td>then by</td>
<td>%</td>
<td>Ascending</td>
</tr>
</tbody>
</table>

Table A–15  Report Display Options for Row Order

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order by</td>
<td>Choose the table or view column whose values will be used to sort rows in the report. Choosing this option is equivalent to specifying a SQL ORDER BY clause. Choose Ascending to sort A to Z or 0 to 9. Choose Descending to sort Z to A or 9 to 0.</td>
</tr>
<tr>
<td>then by</td>
<td>Choose additional columns whose values will be used to sort report rows. For example, if you choose Order by Department ID, then by Employee Name, Oracle Portal sorts report rows numerically using department IDs. Rows containing the same Department ID are then sorted alphabetically using employee names.</td>
</tr>
</tbody>
</table>
Figure A–37 Report Display Options for Mobile Devices

**Mobile Display Options**

Choose the columns whose values will be displayed when the report is viewed on a mobile device.

<table>
<thead>
<tr>
<th>Columns to Display</th>
<th>Maximum Rows Per Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMP.DEPTNO</td>
<td></td>
</tr>
<tr>
<td>EMP.MGR</td>
<td></td>
</tr>
<tr>
<td>EMP.ENAME</td>
<td></td>
</tr>
<tr>
<td>EMP.JOB</td>
<td></td>
</tr>
</tbody>
</table>

For each parameter you choose, an entry field appears on the personalization form that enables users to choose their own conditions for displaying data in the report. Check **Value Required** if you want to require the user to enter a value in the parameter entry field.

<table>
<thead>
<tr>
<th>Value Required</th>
<th>Column Name</th>
<th>Prompt</th>
<th>LOV</th>
<th>Display LOV As</th>
<th>Make Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>☑</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>☑</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For each parameter you choose, an entry field appears on the personalization form that enables users to choose their own conditions for displaying data in the report. Check **Value Required** if you want to require the user to enter a value in the parameter entry field. If you wish, you can use bind variables to allow users to specify their own amounts. See "Using Bind Variables" for more information.

Table A–17 lists and describes the display options for the report personalization form.

Table A–16 Report Display Options for Mobile Devices

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns to Display</td>
<td>Choose the columns you want to display on a mobile device. Use this feature to reduce the size of the report to better fit onto the smaller screens available on mobile devices. For example, you would probably want to limit the number of columns displayed to 2 or 3 for most mobile devices.</td>
</tr>
<tr>
<td>Maximum Rows Per Page</td>
<td>Enter the maximum number of rows you want to display on a single screen. For example, you probably ought to set this option to a relatively small value, such as 5, for most mobile devices.</td>
</tr>
</tbody>
</table>

24. Click Next.

25. In the **Personalization Form Display Options** section, choose the parameters that you want to display on the report’s personalization form (Figure A–38).

Table A–17 Display Options for the Personalization Form Display Options Page

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Required</td>
<td>Select to require that the user specify a value for the column’s entry field on the report personalization form. If you do not check this box, the user will not be required to specify a value.</td>
</tr>
</tbody>
</table>
26. In the **Formatting options** section, choose formatting options to include on the report personalization form (Figure A–39).

Including one or more of these selections enables users to have a greater level of control over the final appearance of the report.
27. In the **Public Formatting Options** section, choose formatting options to include on the parameter tab for the page where the report is displayed (Figure A–40).

**Figure A–40 The Public Formatting Options Section of the Personalization Form Display Options Page**

Public Formatting Options

Choose formatting options that you want to display on the page’s parameter tab. End users can choose these options to format the report.

28. In the **Button Options** section, select the buttons that will display on the report personalization form (Figure A–41).

Table A–18 lists and describes button display options for the report personalization form.

---

**Note:** Break options appear in the personalization form of a report with tabular layout only.

If the **Break Columns** formatting option is moved to the Selected Columns list, then it will be ignored if the report layout is not **Tabular**.
29. Click Next.

30. Optionally, select a template to set the look and feel of the report and the report personalization form when they are displayed on a full Web page.

The template you choose here applies to both the report and the report personalization form when they are displayed on a full Web page. The template is not used when the report and report personalization form are displayed as portlets; in this case, the style of the page is used.

The template controls the look and feel of the page on which the report appears. In contrast, the display options you have specified on other pages in this wizard control the look and feel of the report itself.

Click the Preview button to see how the template elements will display (Figure A–42).

**Figure A–41 The Button Options Section of the Personalization Form Display Options Page**

**Button Options**
Choose the buttons that you want to display on the report's customization form. End users can click the buttons to perform various actions on the report.

<table>
<thead>
<tr>
<th>Button</th>
<th>Name</th>
<th>Location</th>
<th>Alignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>Run Report</td>
<td>Top</td>
<td>Center</td>
</tr>
<tr>
<td>Save</td>
<td>Save</td>
<td>Top</td>
<td>Center</td>
</tr>
<tr>
<td>Batch</td>
<td>Batch</td>
<td>Top</td>
<td>Center</td>
</tr>
<tr>
<td>Reset</td>
<td>Reset</td>
<td>Top</td>
<td>Center</td>
</tr>
</tbody>
</table>

**Table A–18 Button Display Options for the Personalization Form Display Options Page**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button</td>
<td>Select one or more buttons to display on the report personalization form:</td>
</tr>
<tr>
<td>Run</td>
<td>Displays the URL portlet with the options the user has specified in the personalization form.</td>
</tr>
<tr>
<td>Save</td>
<td>Saves users' current selections to enable them to rerun a report using the same parameters without having to reenter them into the report personalization form. When you include this button, the report personalization form also displays a Reset to Defaults button that resets all values in the report personalization form to their default values.</td>
</tr>
<tr>
<td>Batch</td>
<td>Runs the URL portlet in batch mode and saves the results in the database.</td>
</tr>
<tr>
<td>Reset</td>
<td>Undoes any changes made to the report personalization form since the last Save, Reset, or Reset to Defaults event.</td>
</tr>
<tr>
<td>Name</td>
<td>Enter the label that will display on the selected button. Keep the name short to avoid displaying large buttons. You cannot change the label on the Reset to Defaults button.</td>
</tr>
<tr>
<td>Location</td>
<td>Specify the location of the button on the report personalization form. Choose from Top, Top and Bottom, or Bottom.</td>
</tr>
<tr>
<td>Alignment</td>
<td>Specify the alignment of the button on the report personalization form. Choose from Left, Center, or Right.</td>
</tr>
</tbody>
</table>
31. Optionally, enter text for display on the top or bottom of the report or the report personalization form (Figure A–42).

Table A–19 lists and describes the options for including text. Enter text for the report in the left column. Enter text for the report personalization form in the right column.

| **Table A–19** Options for Including Text on a Report or a Report Personalization Form |
|-------------------------|--------------------------------------------------|
| **Option**              | **Description**                                 |
| Display Name            | Edit the display name for the report and the report personalization form. You can specify HTML in this field. |
| Description             | (Report only.) Provide a description of the purpose of this report. This is an attribute that may or may not display, depending on which attributes have been selected for display in the region where the report or report portlet is placed. |
| Header Text             | Enter any introductory text for display at the top of the report or report personalization form. Header Text displays below the display name or title. You can specify HTML in this field. |
| Footer Text             | Enter any text for display at the bottom of the report or report personalization form. You can specify HTML in this field. |
32. Click Next.

33. Optionally, enter PL/SQL code that runs during the execution of the report or report personalization form HTML (Figure A–43).

Table A–20 lists and describes the options available for running PL/SQL code at different points during the execution of the HTML code that creates the report or the report personalization form.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help Text</td>
<td>Enter any text for inclusion on a help page for the report or report personalization form. OracleAS Portal automatically includes a help button with the report and the report personalization form when they are displayed on a full page. When they are displayed as a portlet, whether the help button displays is controlled by the way the host portlet region has been configured. Users click the help button to display the text you enter here. You can specify HTML in this field.</td>
</tr>
<tr>
<td>About Text</td>
<td>Enter any text for inclusion in an About box. OracleAS Portal automatically includes an About button with the report and the report personalization form when they are displayed on a full page. When they are displayed as a portlet, whether the about button displays is controlled by the way the host portlet region has been configured. Users click the about button to display the text you enter here.</td>
</tr>
</tbody>
</table>

Note: In the PL/SQL you enter into these fields, you can use constants for the OracleAS Portal schema, the application schema, and the application name. For more information, see "Referencing the OracleAS Portal Schema".

Figure A–43  Additional PL/SQL Code Page

Additional PL/SQL Code
Enter PL/SQL code that runs at different points during the execution of the report and personalization form.
34. Click Finish.

This saves the report and takes you to a summary page where you can manage and test run the report (Figure A–44). Subsequent sections in this appendix provide more detail about these tasks.

Figure A–44  The Report Summary Page

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>... before displaying the page</td>
<td>Enter a PL/SQL procedure that will execute before the page containing the report or the report personalization form displays.</td>
</tr>
<tr>
<td>... after displaying the header</td>
<td>Enter a PL/SQL procedure that will execute after the report or the report personalization form header displays.</td>
</tr>
<tr>
<td>... before displaying the footer</td>
<td>Enter a PL/SQL procedure that will execute before the report or the report personalization form footer displays.</td>
</tr>
<tr>
<td>... after displaying the page</td>
<td>Enter a PL/SQL procedure that will execute after the page containing the report or the report personalization form displays.</td>
</tr>
</tbody>
</table>

A.1.4.4 Building Forms and Reports against interMedia Rich Content

You can integrate interMedia rich content, such as images, audio, and video, into the reports and forms you build with OracleAS Portal. These objects are stored in the Oracle Database 10g.

The database column types for the interMedia rich content you can display include:

- ORDIMAGE: This object type supports the storage, management, and manipulation of image data, such as GIF, JPEG, and BMP.
- ORDAUDIO: This object type supports the storage and management of audio data, such as MP3, AU, WAV, and MPEG.
ORDVIDEO: This object type supports the storage and management of video data such as REAL, QuickTime 3/4, AVI, and MPEG.

In addition to allowing the use of these interMedia rich content type columns in an OracleAS Portal report or form, object attributes can be displayed in join conditions, formatting, column conditions, and so on. For example, display a video clip’s size or duration in a report, or set a report condition that specifies that only objects modified after a certain date will display.

Keep in mind that, although interMedia supports a variety of content types and formats, the browser being used to view this material must natively support the relevant MIME type or have an installed plug-in that displays rich content that is not typically supported on the Web. For example, most browsers can natively display GIF and JPEG images, but TIFF images are not displayed without an installed plug-in.

**interMedia and Reports**

Two OracleAS Portal Reports Wizards offer the opportunity to include interMedia rich content:

- Reports from Query Wizard
- Query By Example Report (QBE)

With QBE reports, object-type attributes will not display. Consequently, trying to provide a value for an interMedia-based column in a report personalization form results in an error. This means that you cannot use a report personalization form for a QBE report that runs against interMedia object-type columns.

You can display interMedia rich content in a report in one of two ways:

- Embedded (inline)
  The image, audio, or video runs in place within the report.

- As a linked icon
  When a user clicks the link, the content is displayed in a new browser window, or it may be handled by the associated source application. For example, it may display in a RealPlayer window.

These display options apply to all columns that contain interMedia content in the OracleAS Portal report. They can be selected on the Display Options tab in the OracleAS Portal Report Wizard. By default, interMedia rich content is represented by icons in a report. Users click an icon to view the actual content. If you plan to rely on the default, you do not need to do anything further. If you plan to embed interMedia rich content, select the Embed interMedia rich content in the report check box on the Display Options page in the Report Wizard. Refer to Table A–12.

For more information about the Report Wizard, see "Building Reports Declaratively".

**interMedia and Forms**

Two OracleAS Portal Form Wizards offer the opportunity to include interMedia rich content in your form:

- Form based on table or view
- Master-detail form

With either of these methods, you must make sure that the underlying database object you are changing through your form is based on columns of type ORDIMAGE, ORDAUDIO, or ORDVIDEO.
To include *inter*Media rich content in your form, on the **Formatting and Validation Options** page of the OracleAS Portal Form Wizard, select the column name in the left frame, and specify an **Item Type** of **File Upload (interMedia)** in the right frame.

---

**Note:** The assigning of column types is done outside of OracleAS Portal. Consult your database administrator if necessary to create a table with *inter*Media column types.

---

One advantage of uploading images, audio, and video clips into *inter*Media-based columns over uploading into BLOB columns is that the data is automatically parsed to extract several attributes, such as MIME type, length, and any user-defined metadata that might be included in the original media file. For example, a QuickTime file might contain close-captioning, or a RealVideo file might have copyright information. With *inter*Media-based columns, these attributes are automatically extracted and stored in the *inter*Media object for indexing and querying.

### A.1.4.5 Building Charts Declaratively

Charts display data from database tables or views as bar charts. They are based on at least two table or view columns:

- Values in the **Label** column provide a display name for the bars on the chart.
- Values in the **Values** column calculate the size of the bars on the chart relative to one another. Value columns must always contain numeric data.

You can also specify a **Link** column. Values in this column create hypertext links from the chart’s labels to other OracleAS Portal database portlets or URLs. For example, with a link in place, users can click the label in a chart and jump to a form that enables them to update the data on which the chart is based.

Charts are also an effective way to display aggregate data. You can use **Group By** and **Summary Options** in the wizard to sum the column values returned by your query, for example, the minimum, maximum, or average value, or the number of values in a column.

**Figure A–45** shows a chart built with the Portlet Builder Chart wizard.

**Figure A–45  A Chart Built with the Portlet Builder Chart Wizard**

The portlet-building wizards in OracleAS Portal provide two construction methods for charts:

- **Charts From Query Wizard**
- **Charts From SQL Query**

The Query Wizard provides a greater level of assistance in constructing a chart. The SQL Query method provides you with a greater level of control over the query that will be used to fetch data for the chart.
This section describes how to build a chart using the Query Wizard.

To create a chart using the Query Wizard, perform the following steps:

1. Follow the instructions detailed in "Building Portlets Declaratively".
   Return to this section once you complete step 6.

2. Click the Chart link next to Create New… (Figure A–46).

3. On the Charts page, click Charts From Query Wizard (Figure A–47).

4. On the resulting page enter an internal name for the chart in the Name field (Figure A–48).
   The internal name is not published to users.
5. In the **Display Name** field, enter the name by which users will identify this chart (Figure A–48).

The display name is used at runtime. It displays as the title of the browser window as well as in the banner of the chart. When the chart is published as a portlet, the display name is used as the title of the portlet and in the List of Available Portlets in the Portlet Repository.

6. In the **Description** field, enter a description of this chart (Figure A–48).

The description displays below the portlet in the Portlet Repository. It can be a summary of the portlet's purpose, a classification of its type, or any other descriptive information. It may be useful within your organization to have a standard approach to what is included in the description.

7. From the list of **Portal DB Providers**, select the provider that will own this chart (Figure A–48).

This list displays only those providers in which you have been granted privileges to build portlets. We recommend that you use a provider specifically created for owning, or hosting, your declaratively-built portlets.

8. Click **Next**.

9. Click the Browse icon next to the **Tables/Views** field, and choose a table or view on which to base your chart (Figure A–49).

---

**Note:** In the **Tables/Views** field, you can use constants for the OracleAS Portal schema and the application schema (for example, enter `#APP_SCHEMA#.EMP`). For more information, see "Referencing the OracleAS Portal Schema".

---

**Figure A–49  Clicking the Browse Icon to Select a Table or View**

---

Copyright© 2003, Oracle. All Rights Reserved.
The tables and views that you can choose in this step are those that are accessible from the Portal DB Provider you selected in the previous step. Additionally, the list is limited to those tables and views on which you have the following privileges: all table views in the provider schema, granted select to public, granted select to the provider schema.

If clicking the Browse icon produces a "No response from Server" message, and you know the table you want to select, try entering the table name directly into the field.

**Note:** You can choose only one table or view using this wizard. If you want to base the chart on more than one table or view, you must use the Charts from SQL Query wizard to write your own SQL.

10. Click Next.

11. Choose the column(s) that contain the values you will display as Labels, Links, and Values in the chart (Figure A–50).

**Figure A–50  Columns to Be Used for Labels, Links, and Values in the Chart**

Table A–21 lists and describes the options on this page.

**Table A–21  Data Selection Options for Charts**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Choose the column that contains the values you will use as labels for different bars on the chart. For example, if you were going to create a chart that compared salaries of each job title in the department, you might choose the JOB column as the <strong>Label</strong>, SAL as the <strong>Value</strong>, and AVG as the <strong>Group Function</strong>.</td>
</tr>
<tr>
<td>Link</td>
<td>Links must be preconstructed and stored in the database before you can use them here. Use this field to choose a preconstructed link that jumps from a chart bar's label to another OracleAS Portal portlet or URL. Click the Edit icon next to the <strong>Link</strong> field to set link parameters. If you do not set link parameters, the default values of the link are used.</td>
</tr>
</tbody>
</table>
12. Click Next.

13. Optionally, specify conditions that limit the data displayed in the chart.

For example, to display data from all employees in Department 10, choose DEPTNO from Column Name, choose = or like from Condition, and enter 10 in the Value field (Figure A–51).

**Figure A–51  Defining Column Conditions for a Chart**

Table A–21 (Cont.) Data Selection Options for Charts

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| Value      | Choose the column that contains the values to be used to calculate the relative size of the bars in the chart. Value columns always contain numeric data.  

The default for Value is 1. Specifying a value of 1 is useful if you also choose a group function. For example, you can choose the JOB column from SCOTT.EMP as the Label, 1 as the Value, and COUNT as the Group Function. This creates a chart that displays the number of employees in each job classification. |
| Group Function | A group function calculates a single summary value (% , sum, count, average, maximum, minimum, standard deviation, or variance) from groups of numeric values in the Value column. OracleAS Portal uses values in the Label column to determine these groupings.  

For example, you can choose the JOB column from SCOTT.EMP as the Label, SAL as the Value, and AVG as the Group Function. This creates a chart that displays the average salary for each job classification. |

Table A–22 lists and describes the options that appear on the Column Conditions page.
14. Click Next.

15. Optionally, specify conditions and define special formatting for chart data that will be used whenever those conditions are met.

   For example, you can specify that a chart bar should be red if its related column value exceeds a particular amount (Figure A–52).

   ![Figure A–52 Defining Formatting Conditions](image)

   Table A–23 lists and describes the formatting condition options on this page.
### Options on the Formatting Conditions Page of the Chart Wizard

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column</td>
<td>Choose a column on which the condition will be applied.</td>
</tr>
<tr>
<td>Condition</td>
<td>Choose conditions that will define Column’s relationship to Value. Choose from:</td>
</tr>
<tr>
<td></td>
<td>■ equal to (=)</td>
</tr>
<tr>
<td></td>
<td>■ greater than (&gt;)</td>
</tr>
<tr>
<td></td>
<td>■ greater than or equal to (&gt;=)</td>
</tr>
<tr>
<td></td>
<td>■ less than (&lt;)</td>
</tr>
<tr>
<td></td>
<td>■ less than or equal to (&lt;=)</td>
</tr>
<tr>
<td></td>
<td>■ like</td>
</tr>
<tr>
<td></td>
<td>■ not equal to (!=)</td>
</tr>
<tr>
<td></td>
<td>■ null</td>
</tr>
<tr>
<td></td>
<td>■ not null</td>
</tr>
<tr>
<td></td>
<td>■ in</td>
</tr>
<tr>
<td></td>
<td>■ not in</td>
</tr>
<tr>
<td></td>
<td>■ between</td>
</tr>
<tr>
<td>Value</td>
<td>Enter a column value to trigger the condition. There is no need to apply formatting masks to numeric values. For example, in a table with an EMPNO column, if you want to apply special formatting to EMPNO values greater than 3000, select &gt; for the Condition, and enter 3000 in this field. Enter date values in nls_date_format. To specify multiple values after an IN or NOT IN condition, type a colon (:) between each value, for example, 10:20:30. To specify a BETWEEN condition, separate the two values by either a comma, a space, or the word and.</td>
</tr>
<tr>
<td>Col/Bar</td>
<td>Choose whether you want the conditional formatting to be applied to a displayed column name or to the &lt;BAR&gt; associated with value that meets the condition. For example, imagine that your chart uses the SAL and ENAME columns. ENAME is used to label each bar in the chart, and SAL is used to provide the value that drives the size of the bar. If you want a chart bar to turn red and blink if a salary is under 40,000, select &lt;BAR&gt;. If you want the ENAME to turn red and blink when the condition is met, select ENAME. If you want the salary, which is included in the chart along with the label and the bar, to turn red and blink when the condition is met, select SAL.</td>
</tr>
<tr>
<td>Color</td>
<td>Choose a color for displaying data that meets the specified condition. The color will be applied to text or to the bar, depending on the choice you made in the Col/Bar field.</td>
</tr>
<tr>
<td>Type Face</td>
<td>Choose a font in which to display data that meets the specified condition.</td>
</tr>
<tr>
<td></td>
<td>A Select to display results that meet the specified condition in bold.</td>
</tr>
<tr>
<td></td>
<td>A Select to display results that meet the specified condition in italic.</td>
</tr>
<tr>
<td></td>
<td>A Select to underscore results that meet the specified condition.</td>
</tr>
<tr>
<td>Blink</td>
<td>Select to cause results that meet the specified condition to blink on and off.</td>
</tr>
</tbody>
</table>
16. Click Next.

17. Define display options for the various possible views of the chart:

- **Table A–24** describes display options common to all views of a chart (Figure A–53).
- **Table A–25** describes display options for full-page views of a chart (Figure A–54).
- **Table A–26** describes display options for charts displayed as portlets (Figure A–55).
- **Table A–27** describes display options for charts displayed on mobile devices (Figure A–56).

### Figure A–53  Display Options Common to All Chart Views

| Seq | Enter a sequential number to establish an order of precedence for execution of the formatting. This is useful, for example, when a row contains data that meets more than one condition, and the formatting rules differ for each. The condition with the highest position in the order (that is, 1), takes precedence over all the others. |

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Order By</strong></td>
<td>Choose a method for ordering the chart’s data:</td>
</tr>
<tr>
<td></td>
<td>- <strong>ORDER BY VALUE</strong>—The bars in the chart are shown in the same numeric order as values in the table column that you specified in the <strong>Value</strong> entry field.</td>
</tr>
<tr>
<td></td>
<td>- <strong>ORDER BY VALUE DESC</strong>—The bars in the chart are shown in reverse order of values in the table column that you specified in the <strong>Value</strong> entry field.</td>
</tr>
<tr>
<td></td>
<td>- <strong>ORDER BY LABEL</strong>—The bars in the chart are shown in the same order as values in the table column that you specified in the <strong>Label</strong> entry field.</td>
</tr>
<tr>
<td></td>
<td>- <strong>ORDER BY LABEL DESC</strong>—The bars in the chart are shown in the reverse order of values in the table column that you specified in the <strong>Label</strong> entry field.</td>
</tr>
<tr>
<td>Include Null Values</td>
<td>Select to display null values in the chart.</td>
</tr>
<tr>
<td>Treat Null Values as</td>
<td>Enter the text string you want to be displayed for any null values in the chart, for example: <em>(this field is null).</em></td>
</tr>
</tbody>
</table>
### Table A–24 (Cont.) Display Options Common to All Chart Views

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expire After (minutes)</td>
<td>Enter the number of minutes after the initial display to keep the chart in the cache. When the time expires, the chart data is refreshed from the database.</td>
</tr>
</tbody>
</table>

### Figure A–54 Display Options for Full Web Page Chart Views

<table>
<thead>
<tr>
<th>Full Page Options</th>
<th>Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Face</td>
<td>Tahoma</td>
<td>Font Color</td>
</tr>
<tr>
<td>Font Size</td>
<td>12pt</td>
<td>Chart Type</td>
</tr>
<tr>
<td>Axis</td>
<td>100%</td>
<td>Bar Image</td>
</tr>
<tr>
<td>Chart Scale</td>
<td>200%</td>
<td>Bar Width</td>
</tr>
<tr>
<td>Bar Height</td>
<td>20</td>
<td>Value Format Mask</td>
</tr>
<tr>
<td>Maximum Rows Per Page</td>
<td>20</td>
<td>Summary Options</td>
</tr>
<tr>
<td>Log Activity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Show Timing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Show Query Conditions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table A–25 Display Options Applying to Full Web-Page Chart Views

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type Face</td>
<td>Choose the font for displaying chart text.</td>
</tr>
<tr>
<td>Font Color</td>
<td>Choose the font color for displaying chart text. User-defined colors are not available for selection.</td>
</tr>
<tr>
<td>Font Size</td>
<td>Choose the font size for displaying chart text.</td>
</tr>
<tr>
<td>Chart Type</td>
<td>Choose whether to display the chart bars in a horizontal or vertical orientation.</td>
</tr>
<tr>
<td>Axis</td>
<td>Choose a method for displaying chart bars relative to the value of the chart's axis. For example, if you choose Zero, the value of the axis is set at 0. If you choose Average Value, the axis is set at the average of all values in the table or view column on which the Value section of the chart is based.</td>
</tr>
<tr>
<td>Bar Image</td>
<td>Choose an image that will be used to fill in the bars on the chart. Choose MULTI to display bars in different colors.</td>
</tr>
<tr>
<td>Chart Scale</td>
<td>Choose a percent (%) value to set the scale of chart bars relative to the Web page that will host the chart. Higher percentages display larger bars.</td>
</tr>
<tr>
<td>Bar Width</td>
<td>Choose a width in pixels for chart bars. This option applies to bars in both horizontally-and vertically-oriented charts.</td>
</tr>
<tr>
<td>Bar Height</td>
<td>Enter a height in pixels for chart bars. This option applies to bars in both horizontally-and vertically-oriented charts.</td>
</tr>
<tr>
<td>Value Format Mask</td>
<td>Enter a format for numeric values or dates that appear on the chart.</td>
</tr>
<tr>
<td>Maximum Rows Per Page</td>
<td>Enter the maximum number of bars you want to display on each page.</td>
</tr>
<tr>
<td>Summary Options</td>
<td>Choose one or more options to display summary information about the chart. Each option you choose is included in the summary information box at the bottom of the chart. Windows users can choose more than one option by pressing the Ctrl key while clicking each option.</td>
</tr>
</tbody>
</table>
Table A–25  (Cont.) Display Options Applying to Full Web-Page Chart Views

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Activity</td>
<td>Select to enter information in the OracleAS Portal activity log. Information includes performance statistics and the names of users who request the chart.</td>
</tr>
<tr>
<td>Show Timing</td>
<td>Select to display the time when the server received the request to generate HTML for the chart. This information is displayed at the bottom of the chart.</td>
</tr>
<tr>
<td>Show Query Conditions</td>
<td>Select to display all user-specified parameters passed to the SQL query that created the chart and the time the chart was created. This information is displayed at the bottom of the chart.</td>
</tr>
</tbody>
</table>

Table A–26  Display Options for Charts Viewed as Portlets

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Style</td>
<td>Choose the page style element to use for displaying chart text.</td>
</tr>
<tr>
<td>Type Face</td>
<td>Choose a font to override the font of the chosen page style element.</td>
</tr>
<tr>
<td>Font Color</td>
<td>Choose a font color to override the font color of the chosen page style element.</td>
</tr>
<tr>
<td>Font Size</td>
<td>Choose a font size to override the font size of the chosen page style element.</td>
</tr>
<tr>
<td>Bar Width</td>
<td>Choose a width in pixels for bars on the chart. This option applies to bars in both horizontally- and vertically-oriented charts.</td>
</tr>
<tr>
<td>Bar Height</td>
<td>Choose a height in pixels for bars on the chart. This option applies to bars in both horizontally- and vertically-oriented charts.</td>
</tr>
<tr>
<td>Maximum Rows Per Page</td>
<td>Enter the maximum number of bars you want to display in the chart.</td>
</tr>
</tbody>
</table>

Figure A–55  Display Options for Charts Viewed As Portlets

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Style</td>
<td>Choose the page style element to use for displaying chart text.</td>
</tr>
<tr>
<td>Type Face</td>
<td>Choose a font to override the font of the chosen page style element.</td>
</tr>
<tr>
<td>Font Color</td>
<td>Choose a font color to override the font color of the chosen page style element.</td>
</tr>
<tr>
<td>Font Size</td>
<td>Choose a font size to override the font size of the chosen page style element.</td>
</tr>
<tr>
<td>Bar Width</td>
<td>Choose a width in pixels for bars on the chart. This option applies to bars in both horizontally- and vertically-oriented charts.</td>
</tr>
<tr>
<td>Bar Height</td>
<td>Choose a height in pixels for bars on the chart. This option applies to bars in both horizontally- and vertically-oriented charts.</td>
</tr>
<tr>
<td>Maximum Rows Per Page</td>
<td>Enter the maximum number of bars you want to display in the chart.</td>
</tr>
</tbody>
</table>

Table A–27  Display Options for Charts Viewed on Mobile Devices

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Rows Per Page</td>
<td>Enter the maximum number of bars you want to display in the chart. For example, you probably ought to set this option to a relatively small value, such as 5, for most mobile devices.</td>
</tr>
</tbody>
</table>
18. Click Next.

19. Define the content for display on the chart personalization form.

Use the options on this page to give your users control over what data will appear in their view of the chart.

For example, if you choose the DEPTNO column from the SCOTT.EMP table as a Column Name in this step, OracleAS Portal adds an entry field for DEPTNO to the chart's personalization form. Users can type a department number in the field to limit the display to data relevant to that department.

Optionally, you can add a list of values to the entry fields. Instead of requiring users to type a numeric value for DEPTNO, you could add a list of values that enables them to choose, for example, 10, 20, or 30.

In portlet implementations, the chart personalization form displays when users click the Personalize link at the top of the portlet. In full-page implementations, you can provide users with a link to run the chart; the link can target the chart personalization form, which in turn can have a Run button. Users click this button to run the chart once they’ve set their personalization options.

If you wish, you can use bind variables to allow users to specify their own values in the personalization form. See "Using Bind Variables" for more information.

- Table A–28 describes chart personalization form display options (Figure A–57).
- Table A–29 describes chart personalization form formatting options (Figure A–58).
- Table A–30 describes chart personalization form public formatting options (Figure A–59).
- Table A–31 describes chart personalization form button options (Figure A–60).

**Figure A–57 Chart Personalization Form Display Options**

**Personalization Form Display Options**

Choose the parameters that you want to display on the chart's personalization form. For each parameter you choose, an entry field appears on the personalization form that enables users to select their own conditions for displaying data in the chart. Check Value Required if you want to require the end user to enter a value in the personalization form.

<table>
<thead>
<tr>
<th>Value Required</th>
<th>Column Name</th>
<th>Prompt</th>
<th>LOV</th>
<th>Display LOV As</th>
<th>Make Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑</td>
<td>DEPTNO</td>
<td>Limit data to this dept</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>☑</td>
<td>%</td>
<td>%</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>☑</td>
<td>%</td>
<td>%</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
</tbody>
</table>

**Table A–28 Chart Personalization Form Display Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Required</td>
<td>Select to require the user to specify a value for the column's entry field on the chart's personalization form.</td>
</tr>
<tr>
<td>Column Name</td>
<td>Choose a table or view column for inclusion on the chart personalization form. An entry field will display on the form that enables users to specify values that limit the data that will be displayed in the chart. If you do not choose a table or view column, no entry appears on the personalization form.</td>
</tr>
</tbody>
</table>
Figure A–58 Chart Personalization Form Formatting Options

Table A–28 (Cont.) Chart Personalization Form Display Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prompt</td>
<td>Enter prompt text to display next to the entry field. The prompt text tells users what to enter in the field, for example: Choose the department number you want to display.</td>
</tr>
<tr>
<td>LOV</td>
<td>Select a list of values (LOV) for the column’s entry field. Users of the personalization form can choose values from this list to limit the data displayed in the report. Click the Browse icon next to this field to locate an LOV. Note that the LOV must already be built and stored in the Portal Repository. For more information, see “Building Lists of Values Declaratively”. In the LOV field, you can use a constant for the application name (for example, enter #APP_NAME#.dept_lov). For more information, see “Referencing the OracleAS Portal Schema”.</td>
</tr>
</tbody>
</table>

Display LOV As

- **<Blank>:** No option is selected.
- **Check box:** Each selection displays with a check box next to it. Users select or clear the box to indicate their preferred value.
- **Combo box:** Combines a pop-up list and a manual entry field. Users can either select a provided value or enter their own.
- **Pop up:** Provides a pop-up list from which users select a value.
- **Radio group:** Lists each selection with a radio button next to it. Users select one of the buttons to indicate their preferred value.
- **Multiple Select:** Allow users to choose more than one value.

Make Public

Check this field if you want this element to display on the chart personalization form.

Be sure to make public any element that will be a variable in the list of portlet parameters associated with a page parameter.

More Parameters

Click to add more entry fields on the personalization form.

**Figure A–58 Chart Personalization Form Formatting Options**

**Formatting Options**

Choose formatting options that you want to display on the chart’s customization form. End users can select these options to format the chart.
### Table A–29  Chart Personalization Form Formatting Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis</td>
<td>Select to enable users of the chart personalization form to choose a method for displaying chart bars relative to the value of the chart's axis. For example, if users choose <strong>Zero</strong>, the value of the axis is set at 0. If users choose <strong>Average Value</strong>, the axis is set at the average of all values in the table or view column on which the <strong>Value</strong> section of the chart is based.</td>
</tr>
<tr>
<td>Include Nulls</td>
<td>Select to enable users of the chart personalization form to specify whether to display null values in the chart.</td>
</tr>
<tr>
<td>Maximum Rows/Page</td>
<td>Select to enable users of the chart personalization form to specify the maximum number of bars to display in the chart.</td>
</tr>
<tr>
<td>Summary</td>
<td>Select to enable users of the chart personalization form to choose one or more options that display summary information about the chart. Each option that users choose is included in the summary information box at the bottom of the chart.</td>
</tr>
<tr>
<td>Type</td>
<td>Select to enable users of the chart personalization form to choose a font for displaying chart text.</td>
</tr>
<tr>
<td>Display Name</td>
<td>Select to allow users of the chart personalization form to edit the display name for the chart.</td>
</tr>
<tr>
<td>Order By</td>
<td>Select to enable users of the chart personalization form to set the display order for chart data.</td>
</tr>
</tbody>
</table>

### Table A–30  Chart Personalization Form Public Formatting Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Select to enable unauthenticated users of the chart personalization form to choose a font for displaying chart text.</td>
</tr>
<tr>
<td>Display Name</td>
<td>Select to enable unauthenticated users of the chart personalization form to edit the display name of the chart.</td>
</tr>
<tr>
<td>Maximum Rows</td>
<td>Select to enable unauthenticated users of the chart personalization form to specify the maximum number of bars to display in the chart.</td>
</tr>
<tr>
<td>Order By</td>
<td>Select to enable unauthenticated users of the chart personalization form to set the display order for chart data.</td>
</tr>
</tbody>
</table>
Figure A–60  Chart Personalization Form Button Options

**Button Options**

Choose the buttons that you want to display on the chart’s customization form. End users can click the buttons to perform various actions on the chart.

<table>
<thead>
<tr>
<th>Button</th>
<th>Name</th>
<th>Location</th>
<th>Alignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>Run</td>
<td>Top</td>
<td>Center</td>
</tr>
<tr>
<td>Save</td>
<td>Save</td>
<td>Top</td>
<td>Center</td>
</tr>
<tr>
<td>Batch</td>
<td>Batch</td>
<td>Top</td>
<td>Center</td>
</tr>
<tr>
<td>Reset</td>
<td>Reset</td>
<td>Top</td>
<td>Center</td>
</tr>
</tbody>
</table>

**Table A–31  Chart Personalization Form Button Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button</td>
<td><em>(Full page only)</em> Select one or more buttons to display on the chart personalization form:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Run</strong>—Displays the URL portlet with the options the user has specified in the chart personalization form.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Save</strong>—Saves users’ current selections to enable them to rerun a chart using the same parameters without having to reenter them into the chart personalization form. When you include this button, the chart personalization form also displays a <strong>Reset to Defaults</strong> button that resets all values in the chart personalization form to their default values.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Batch</strong>—Runs the URL portlet in batch mode and saves the results in the database.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Reset</strong>—Undoes any changes made to the chart personalization form since the last Save, Reset, or Reset to Defaults event.</td>
</tr>
<tr>
<td>Name</td>
<td>Enter the label that will display on the selected button. Keep the name short to avoid displaying large buttons. You cannot change the label on the <strong>Reset to Defaults</strong> button.</td>
</tr>
<tr>
<td>Location</td>
<td>Specify the location of the button on the chart personalization form. Choose from <strong>Top</strong>, <strong>Top and Bottom</strong>, or <strong>Bottom</strong>.</td>
</tr>
<tr>
<td>Alignment</td>
<td>Specify the alignment of the button on the chart personalization form. Choose from <strong>Left</strong>, <strong>Center</strong>, or <strong>Right</strong>.</td>
</tr>
</tbody>
</table>

20. Click **Next**.

21. Optionally, select a template to define the look and feel of the page that will host the chart, and enter text for display at the top and bottom (optional) of the chart and the chart personalization form.

The template is applied only when the chart is displayed on a full Web page. If the chart is displayed as a portlet, the template you select here will be ignored and the style of the host page will be used.

**Table A–32** lists and describes the options available on this page (Figure A–61).
Figure A–61  Options Available on the Chart and Personalization Form Text Page

**Chart and Personalization Form Text**
Enter descriptive text that you want to appear on the top or bottom of the chart and its personalization form. You can add a chart title, welcome text, footer text that appears at the bottom of the page and help text for the chart and personalization form. You can also choose a template that controls the look and feel of the page on which the chart and personalization form appear.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template</td>
<td>Choose a template to set the look and feel of the page that will host the chart. The template is applied only when the chart is displayed on a full Web page. If the chart is displayed as a portlet, the template you select here will be ignored and the style of the host page will be used. Display options, which you specified on previous wizard pages, control the look and feel of the chart itself.</td>
</tr>
<tr>
<td>Preview Template</td>
<td>Click to view the appearance of the template that is currently selected in the Template list.</td>
</tr>
<tr>
<td>Display Name</td>
<td>Edit the chart's display name or the title for the chart personalization form. You can specify HTML in this field.</td>
</tr>
<tr>
<td>Header Text</td>
<td>Enter any introductory text that you want to display at the top of the chart or the chart personalization form. This text displays below the display name or title. You can specify HTML in this field.</td>
</tr>
</tbody>
</table>
22. Click Next.

23. Optionally, enter PL/SQL code to run at different points during the runtime creation of the chart or the chart personalization form.

   Table A–33 lists and describes the options available on this page (Figure A–62).

**Figure A–62 Additional PL/SQL Code Options for Charts**

<table>
<thead>
<tr>
<th>Option Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter a PL/SQL procedure that will execute before the</td>
</tr>
<tr>
<td>page containing the chart or the chart personalization</td>
</tr>
<tr>
<td>form displays.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Enter a PL/SQL procedure that will execute after the</td>
</tr>
<tr>
<td>chart or chart personalization form header displays.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>If you want to display output in the chart from a PL/SQL</td>
</tr>
<tr>
<td>procedure, you must enter the PL/SQL in the ... after</td>
</tr>
<tr>
<td>displaying the header field.</td>
</tr>
</tbody>
</table>

---

**Table A–32 (Cont.) Options Available on the Chart and Personalization Form Text Page**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Footer Text</td>
<td>Enter any text that you want to display at the bottom of the chart or the</td>
</tr>
<tr>
<td></td>
<td>chart personalization form. You can specify HTML in this field.</td>
</tr>
<tr>
<td>Help Text</td>
<td>Enter any text that you want to display in a help page for the chart or</td>
</tr>
<tr>
<td></td>
<td>chart personalization form. OracleAS Portal automatically includes a help</td>
</tr>
<tr>
<td></td>
<td>button with the chart and the chart personalization form when they are</td>
</tr>
<tr>
<td></td>
<td>displayed on a full page. When they are displayed as a portlet, whether the</td>
</tr>
<tr>
<td></td>
<td>help button displays is controlled by the way the host portlet region has</td>
</tr>
<tr>
<td></td>
<td>been configured. Users click the help button to display the text you enter.</td>
</tr>
<tr>
<td></td>
<td>You can specify HTML in this field.</td>
</tr>
</tbody>
</table>

---

**Table A–33 Options Available on the Additional PL/SQL Code Page**

<table>
<thead>
<tr>
<th>Option Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>... before displaying the page.</td>
</tr>
<tr>
<td>... after displaying the header.</td>
</tr>
<tr>
<td>... before displaying the footer.</td>
</tr>
<tr>
<td>... after displaying the page.</td>
</tr>
</tbody>
</table>
Using a Wizard to Build a Portlet

Table A–33 (Cont.) Options Available on the Additional PL/SQL Code Page

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>... before displaying the footer.</td>
<td>Enter a PL/SQL procedure that will execute before the chart or personalization form footer displays.</td>
</tr>
<tr>
<td>... after displaying the page.</td>
<td>Enter a PL/SQL procedure that will run after the page containing the chart or personalization form displays.</td>
</tr>
</tbody>
</table>

24. Click Finish.

Clicking Finish saves your changes and takes you to a summary page where you can test your results (Figure A–63). See “Performing Test Runs on a Portlet” for more information on testing your results.

Figure A–63 Chart Summary Page

A.1.4.6 Building Lists of Values Declaratively

When you build a list of values using the OracleAS Portal List of Values Wizard, you have the option of creating two types of lists:

- Dynamic list of values
- Static list of values

A dynamic list of values takes its content from a SELECT statement you build through the wizard. The content of the list may change, depending on the data that is available for selection from the source database. A static list of values takes its content from values you enter in the wizard. These values do not change unless you edit the list and specifically alter them.

Figure A–64 shows an example list of values page with examples of the types of lists of values you can create.
This section describes how to create a dynamic list of values.

To create a dynamic list of values, perform the following steps:

1. Follow the instructions detailed in "Building Portlets Declaratively". Return to this section once you complete step 6.

2. Click the List of Values link next to Create New… (Figure A–65).

3. On the List of Values page, click Dynamic List of Values (Figure A–66).

4. On the resulting page, select the Portal DB Provider that will own this list from the Owner drop-down list (Figure A–67).
5. In the **Name** field, enter a name for this list of values (Figure A–67).
This name appears on the selection list when you insert a list of values.

6. From the **Default Format** drop-down list, choose a default format for the list of values (Figure A–67).
Choose from:
- **Check box**: Each selection displays with a check box next to it. Users select or clear the box to indicate their preferred value.
- **Combo box**: Combines a pop-up list and a manual entry field. Users can either select a provided value or enter their own.
- **Pop up**: Provides a pop-up list from which users select a value.
- **Radio group**: Lists each selection with a radio button next to it. Users select one of the buttons to indicate their preferred value.
- **Multiple Select**: Allow users to choose more than one value.

Note that portlet developers who add the list of values to their portlet can override this default and display the list in a different format.

7. Specify whether to show null values by selecting either **Yes, No**, or % from the **Show Null Value** drop-down list (Figure A–67).
   - When you select **Yes**, null values display as *Null* in the list of values.
   - When you select **No**, null values display as blank spaces in the list of values.
   - When you select %, blank spaces are also displayed. The difference between % and **No** is that, with %, when you place the list of values on a form, the null...
value (the wildcard %) is available for the set up of dynamic transactions in the form's code.

8. Enter a SQL SELECT statement in the SQL Query text box (Figure A–67).

Select two values from two table or view columns (though you can select each value from the same column):

- The first column specifies the values displayed in the list of values.
- The second column specifies the actual values that are passed to the portlet.

For example, the query `select ename, empno from scott.emp` creates a list of values that displays employee names from the ENAME column. It passes the employee's associated ID number (EMPNO) to the portlet when a user chooses a name from the list of values.

Do not end your query with a semicolon.

---

**Notes:**

- You can enter bind variables by prefixing them with colons (:), but only if the list of values is to be used in a form portlet. Lists of values with bind variables will not work in other kinds of OracleAS Portal portlets.

- In the SQL query, you can use constants for the OracleAS Portal schema and the application schema. For more information, see "Referencing the OracleAS Portal Schema".

- To create an LOV from a remote database, create a database link to the remote database and then use this database link name in the SQL query. Database links can be created in different ways. An example for creating a database link is as follows:

```sql
create database link <database_link_name> connect to <remote_portal_schema> identified by <remote_portal_password> using <database_connect_info>;
```

Refer to the Oracle Database Administrator's Guide for more information about creating a database link.

Use the following query to check if you can connect to the remote database using the database link:

```sql
select version from wwc_version$@<database_link_name>;
```

---

9. Click OK to save your changes.

Clicking OK takes you to a summary page where you can manage the list of values (Figure A–68). For example, from this page, you can run the list to see the different ways it can display.
Once you complete a list of values, it becomes available. It displays automatically on selection lists wherever you can add a list of values. For example, both the form and report personalization forms you construct with Portlet Builder wizards allow for the addition of lists of values. As you go through these wizards, you’ll note that the list you created here now appears as a selection option.

A.2 Editing a Portlet Builder Component

After you build a portlet using a wizard, you can use options in the Navigator to edit it. You can edit the portlet and overwrite the current version with your changes, or you can edit the portlet as new, and create a new version of it (consequently leaving the old, original version unchanged).

Where the wizards step you through the creation of a portlet, the editor provides tabs that contain the options that were available in the wizard. Each tab corresponds to a step in the wizard that created the portlet. Entry fields on each tab contain the values that were specified during creation of the portlet, or by the user who last edited the portlet.

A portlet is locked while you edit it, preventing other users from making changes to it. The portlet remains locked until you click OK in the Edit page. For more information, see “Managing Locks on Portlets”.

Note: To edit a portlet, you must have at least the Edit privilege on the portlet or the provider that owns it.

To edit a portlet, perform the following steps:

1. At the top of the Portal Builder page, click the Navigator link.

2. Click the Providers tab to bring it forward.

3. Click the Locally Built Providers link.
   The Name column displays the names of all the providers on which you have privileges.

4. Click the name of the OracleAS Portal database provider that contains the portlet that you want to edit.

5. Click the name of the portlet.
6. On the resulting page, the Develop tab should be selected. Click the Edit link at the bottom of the tab.

To leave the original version of a portlet unchanged, click the Edit as New link instead of Edit.

7. Click one or more tabs to bring them forward and edit their associated options.

The tabs contain the same options that are available in the wizards. For more information about these options, see the previous sections that cover building portlets:

- Building Forms Declaratively
- Building Reports Declaratively
- Building Charts Declaratively
- Building Lists of Values Declaratively

As you switch from tab to tab, OracleAS Portal keeps track of any changes you make. Click OK only after you have made all the changes on all the tabs that you want to make. If you decide you don’t want to save your edits, click Cancel.

Note that it is important that you click either OK or Cancel. Simply closing the window will cause the component to remain locked. Special measures must be taken to unlock inappropriately locked portlets. For more information, see “Managing Locks on Portlets”.

A.3 Managing Portlets

For each locally-built portlet, OracleAS Portal provides a central location from which you can perform management tasks: the management page. This section describes how to locate any locally built portlet's management page and how to perform various management tasks. It includes the following subsections:

- Navigating to the Component Management Page
- Renaming a Portlet
- Deleting a Portlet
- Copying a Portlet
- Generating the PL/SQL Package for a Portlet

**Note:** To rename, delete, or copy a portlet, you must have at least the Edit privilege on the provider that owns the portlet.

To generate a portlet, you must have at least the View Source privilege on the portlet or on the provider that owns it.

A.3.1 Navigating to the Component Management Page

To navigate to the component management page, perform the following steps:

1. Click the Navigator link at the top of the Portal Builder page.
2. Click the Providers tab to bring it forward.
3. On the Providers tab, click the Locally Built Providers link.
4. Click the name of the Database Provider that owns the portlet you want to manage, then click the Manage link next to the component you will manage.
Note the three tabs on this page:

- **Develop**—Run, test, revise, delete and manage the various versions of a component.
- **Manage**—Export, copy, rename (both internal and display names), generate, and monitor the performance of the component.
- **Access**—Define security for the component within the OracleAS Portal framework. This includes publication of the component as a portlet, privilege inheritance from the provider, and cache invalidation.

### A.3.2 Renaming a Portlet

To rename a portlet, perform the following steps:

1. Navigate to the component management page for the portlet you want to rename.
   
   For information on how to navigate to this page, see "Navigating to the Component Management Page".

2. Click the **Manage** tab to bring it forward.

3. Click the **Rename** icon.

4. Rename the internal name and the display name (optional) for the portlet by performing the following steps:
   
   a. For the internal name, in the **New Component Name** field enter a new name for the portlet.
   
   b. For the display name, in the **New Component Display Name** field enter a new display name for the portlet.

5. Click **OK**.

   The portlet is renamed with the name you provided.

### A.3.3 Deleting a Portlet

To delete a portlet from the database, perform the following steps:

1. Navigate to the component management page for the portlet you want to delete.
   
   For information on how to navigate to this page, see "Navigating to the Component Management Page".

2. Click the **Develop** tab to bring it forward.

3. Click the **Delete** icon.

   The Delete Portlet page displays the versions of the portlet that you are authorized to delete.

4. Select the check box next to each version of the portlet that you want to delete.

5. Click **Yes**.

### A.3.4 Copying a Portlet

To copy a portlet, perform the following steps:

1. Navigate to the component management page for the portlet you want to copy.
   
   For information on how to navigate to this page, see "Navigating to the Component Management Page".
2. Click the Manage tab to bring it forward.

3. Click the Copy icon.

   The Copy Portlet page displays.

4. From the New Owner list, choose the provider that will own the new copy of the portlet.

   You can choose the same provider.

5. Provide new internal and display names for the portlet by performing the following steps:
   a. For the internal name, in the New Component Name field enter a new name for the portlet.
   b. For the display name, in the New Component Display Name field enter a new display name for the portlet.

6. Click OK.

A.3.5 Generating the PL/SQL Package for a Portlet

Use this feature to make sure that the portlet code compiles without errors and performs within a reasonable time. This is particularly useful if you’ve written more complex code than is currently supported in the portlet builder wizards. For example, you could create a package; base a form or report on the package; then, using this feature, verify that the form or report will compile.

To generate the PL/SQL package for a portlet, perform the following steps:

1. Navigate to the component management page for the portlet with which you want to work.
   
   For information on how to navigate to this page, see "Navigating to the Component Management Page".

2. Click the Manage tab to bring it forward.

3. Click the Generate icon.

   OracleAS Portal generates the PL/SQL package using the most recent OracleAS Portal routines.

A.3.6 Viewing Source Code

OracleAS Portal stores portlets in the database as PL/SQL packaged procedures. You can view the package spec and body for the portlet as well as its call interface.

The portlet call interface displays the arguments that can be set during run time. For example, if you created a report and selected HTML as a Default Format in the Display Options step of the Report build wizard, the call interface displays HTML as the default value for the _format_out argument.

When you run the package containing the portlet in PL/SQL or by calling it from a URL, you can edit the call interface to accept different arguments. You could change the _format_out argument to another report output format, such as ASCII.

This section describes how to view the package spec, body, and call interface for a portlet. It includes the following subsections:

- Viewing the Package Spec and Body for a Portlet
- Viewing the Call Interface for a Portlet
A.3.6.1 Viewing the Package Spec and Body for a Portlet
To view the package spec and body for a portlet, perform the following steps:

1. Navigate to the component management page for the portlet with which you want to work. For information on how to navigate to this page, see "Navigating to the Component Management Page".
2. Click the Develop tab to bring it forward.
3. Click the Package Spec link next to PL/SQL Source to view the portlet's spec; click the Package Body link next to PL/SQL Source to view the body.

A.3.6.2 Viewing the Call Interface for a Portlet
To view the call interface for a portlet, perform the following steps:

1. Navigate to the component management page for the portlet with which you want to work. For information on how to navigate to this page, see "Navigating to the Component Management Page".
2. Click the Develop tab to bring it forward.
3. Click the Show link next to Call Interface.

A.3.7 Hiding the SQL Query and Bind Information
When building a report using a report portlet or dynamic report portlet, you may find that, along with the output report, the SQL query and bind variable information are displayed. The information displayed is as shown in the following example:

```
Query:
declare A_subject varchar2(200) := :M_subject; begin htp.print('');
htp.print(''); htp.print(''); htp.print(''||A_subject||''); htp.print('');
htp.print('Test - Passing a parameter to a Dynamic Page'); htp.print('');
end;

Binding:
:M_subject binding with "My Report Portlet"
```

You can choose to hide the SQL query and bind variable information in the output. To do this, perform the following steps:

1. Navigate to the Component Management page. Refer to Section A.3.1, "Navigating to the Component Management Page" for details.
2. Click the Manage tab.
3. Select the Hide SQL Query Info option.
4. Click OK.
A.3.8 Managing Locks on Portlets

A portlet lock prevents other developers from overwriting your changes while you are editing a portlet. No other developer can edit the portlet when it is locked. Portlets are locked automatically when they are being edited. You can determine who is doing the editing by taking the steps outlined in this section.

---

**Note:** To view a portlet lock or to unlock a portlet, you must have at least the Edit privilege on the portlet or the provider that owns it.

---

To view the name of a user who has locked a portlet, perform the following steps:

1. Navigate to the component management page for the portlet with which you want to work.
   
   For information on how to navigate to this page, see "Navigating to the Component Management Page".

2. Click the Manage tab to bring it forward.

3. Click the Show Locks on this Component link.

Locked portlets show a Status of EDIT on the resulting page. In addition to status, this page also lists the user responsible for the lock, the length of time the version has been locked, and other information. It also provides an Unlock link you can click to unlock the portlet.

The Unlock link is useful if a developer has edited a component, then closed the browser window without clicking OK or Cancel. In such case, the component remains locked. Use the Unlock link to unlock the component.

A.4 Managing Versions

OracleAS Portal enables you to store multiple versions of the same portlet in the database. Use the Edit as New link on the Develop tab (see Section A.3.1) to create a new version of the portlet that is based on the current production version. When you save your edits, you will be saving them to a new version, rather than overwriting any existing versions. The next time you edit the portlet, you can open the most current version or an earlier version of the portlet. This section describes how to access various versions of an OracleAS Portal component.

To access various versions of an OracleAS Portal component, perform the following steps:

1. Navigate to the component management page for the portlet with which you want to work.
   
   For information on how to navigate to this page, see "Navigating to the Component Management Page".

2. Click the Develop tab to bring it forward.

3. To access the production version:
   
   - Click the Edit link to make changes to the existing production version.
   
   - Click the Edit as New link to make changes to a new version that is based on the production version.
The production version is listed next to **Production Version Status** on the **Develop** tab. This field lists the production version, and indicates whether this version’s code package is valid, that is, that it will run.

4. To access an archived version of the component, perform the following steps:
   
a. Next to **Archive Version(s)**, click the \(x\) **(Archive)** version with which you want to work.
   
The variable value \(x\) stands for the archive number.

b. On the resulting page, click **Yes** to make the selected version the production version.
   
   Note that the version number listed next to **Production Version Status** changes to your selection.

   c. On the **Develop** tab, click the **Edit** or the **Edit as New** link to edit the newly selected production version of the component.

You can delete portlet versions from the database at any time. Follow the steps outlined in "Deleting a Portlet", then check the box next to the version(s) you want to delete, and click **Yes**. If you delete the production version and leave an archive version intact, you may want to select the archive version on the **Develop** tab and make it the production version.

The most recent version of a portlet is called the **production version**. The production version can be valid or invalid, depending on whether the database package containing the portlet will run without errors. If a portlet has a version status of **(PRODUCTION with INVALID package)**, you must either generate the portlet (see Section A.3.5) or edit the portlet (see Section A.2) to fix the errors before it will run.

### A.5 Managing Portlet Security

The **Actions** column of the Navigator (Figure A–69) displays the actions you may perform on a portlet. The actions listed here depend on the access privileges you have been granted on the portlet.

**Figure A–69  Portlet Actions**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Actions</th>
<th>Creator</th>
<th>Last Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report from SQL Query</td>
<td>Detail Report</td>
<td>Manage, Run, Edit, Delete, Grant Access, Export</td>
<td>PEOPLE_APP</td>
<td>31-MAR-2003</td>
</tr>
<tr>
<td>Hierarchies</td>
<td>Organization Chart</td>
<td>Manage, Run, Edit, Delete, Grant Access, Export</td>
<td>PEOPLE_APP</td>
<td>31-MAR-2003</td>
</tr>
<tr>
<td>Dynamic Pages</td>
<td>People Search</td>
<td>Manage, Run, Edit, Delete, Grant Access, Export</td>
<td>PEOPLE_APP</td>
<td>31-MAR-2003</td>
</tr>
<tr>
<td>Report from SQL Query</td>
<td>People Search Report</td>
<td>Manage, Run, Edit, Delete, Grant Access, Export</td>
<td>PEOPLE_APP</td>
<td>31-MAR-2003</td>
</tr>
<tr>
<td>Report from SQL Query</td>
<td>Person Details</td>
<td>Manage, Run, Edit, Delete, Grant Access, Export</td>
<td>PEOPLE_APP</td>
<td>31-MAR-2003</td>
</tr>
<tr>
<td>Link from Wizard</td>
<td>OPRP_LINK</td>
<td>Manage, Run, Edit, Grant Access, Export</td>
<td>PEOPLE_APP</td>
<td>31-MAR-2003</td>
</tr>
</tbody>
</table>

By default, you have the **Manage** privilege on any portlet you create. The Manage privilege provides the highest level of access. This means you can run, edit, delete, or export the portlet. You can also grant access privileges to another user on any portlet that you create.

To build a portlet within OracleAS Portal, you must have at least the **Edit** privilege on the provider that will own the finished portlet. By default, portlets inherit whatever privileges are assigned to the provider that owns them. For example, after a component is created in the MY_APP provider, all developers with at least the **Execute**
privilege on MY_APP can run the component after it is published as a portlet. Portlet owners can override these privileges and set access on a user-by-user level, rather than at the provider level.

Table A–34 lists the actions available for portlets and indicates which privileges enable a user to perform those actions.

<table>
<thead>
<tr>
<th>Table A–34 Portlet Actions Associated with Portlet Privilege Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>Grant portlet privileges to other users</td>
</tr>
<tr>
<td>Manage portlet locks</td>
</tr>
<tr>
<td>Edit any portlet version</td>
</tr>
<tr>
<td>Delete the portlet from the database</td>
</tr>
<tr>
<td>Rename the portlet</td>
</tr>
<tr>
<td>Export the portlet to another database</td>
</tr>
<tr>
<td>Copy the portlet</td>
</tr>
<tr>
<td>Generate the portlet PL/SQL package</td>
</tr>
<tr>
<td>Monitor portlet usage</td>
</tr>
<tr>
<td>View portlet's call interface, package spec, and body</td>
</tr>
<tr>
<td>Personalize the portlet</td>
</tr>
<tr>
<td>Run the portlet</td>
</tr>
<tr>
<td>Add the portlet to the Favorites list</td>
</tr>
</tbody>
</table>

**A.5.1 Granting Portlet Access Privileges**

*Note:* To grant access privileges on a portlet to other users or groups, you must have at least the Manage privilege on the portlet or the provider that owns it.

You can define access to your portlet at the provider level and at the portlet level. This section describes how to inherit provider access privileges or override them by setting privileges at the portlet level. It contains the following subsections:

- **Inheriting Portlet Access Privileges from a Provider**
- **Granting Access Privileges to Individual Users**

**A.5.1.1 Inheriting Portlet Access Privileges from a Provider**

By default, access to your portlet is inherited from the provider that owns it. For example, all users who have the *Execute* privilege on the owning provider are automatically able to execute your portlet. Users with the *Edit provider* privilege can edit all the portlets owned by the provider, and so forth. You can set provider access
Managing Portlet Security

through the Grants tab (for more information, see "Granting and Revoking Privileges on Database Objects"). On the Grants tab, you can grant different privileges to different groups of users. Once privileges on the provider are established, you can specify that any portlets created under the provider must inherit the same set of privileges. This section describes how to specify that portlets should inherit privileges granted to their host providers.

Note: Before you can publish Portlet Builder components as portlets, the host provider must be exposed to OracleAS Portal as a provider. For information on how to enable this setting, see "Exposing a Provider".

To inherit portlet access privileges from a provider, perform the following steps:

1. Click the Navigator link at the top of the Portal Builder page.
2. Click the Providers tab to bring it forward.
3. Click the Locally Built Providers link.
4. Click the name of the Database Provider that contains the portlet on which you want to grant access privileges.
5. Click the Manage link next to the relevant portlet.
   The Manage page displays.
6. Click the Access tab.
7. Select the Inherit Privileges from Provider check box.
   Portlet access will be defined by the portlet’s host provider.

A.5.1.2 Granting Access Privileges to Individual Users
You can override the default access privileges (that is, those granted through the provider) by setting them at the portlet level. This section describes how to set access privileges at the portlet level, allowing you to more specifically target privileges to individual users.

To grant access privileges to individual users, perform the following steps:

1. Click the Navigator link at the top of the Portal Builder page.
2. Click the Providers tab to bring it forward.
3. Click the Locally Built Providers link.
4. Click the name of the Database Provider that contains the portlet on which you want to grant access privileges.
5. Click the Manage link next to the relevant portlet.
   The Manage page displays.
6. Click the Access tab.
7. Clear the Inherit Privileges from Provider check box, then click Apply.
   The Access page displays two new sections: Grant Access and Change Access.
8. In the Grantee field, enter the name of the user or group to whom you want to grant privileges.
If you are not sure of the name of the user or group click the Browse Users or Browse Groups icon and select from the list provided.

9. From the privilege list, choose an access privilege to grant to the user or group.

See Table A–34 for information on the actions allowed with different access privileges.

10. Click Add.

The user or group you entered in the Grantee field displays along with the access privilege you granted in the Change Access section at the bottom of the page.

11. (Optional) To modify a user or group’s access privilege, choose a new privilege next to the user or group in the Change Access section.

12. (Optional) To remove all privileges, click the Delete icon next to the user or group in the Change Access section.

13. Click Apply.

14. Click Close to exit the Manage page and return to the Portal Navigator.

### A.6 Performing Test Runs on a Portlet

After you create a component under a Portal DB Provider, you will likely want to test whether it runs to your satisfaction and make edits to it if it does not. All of this functionality is available to you through the Develop tab on the Manage page. From this tab, you can run a component as a full page, as a portlet, and through the component’s personalization form.

You also have the option of running the component as a portlet through the portlet personalization form. But this option is not available on the Develop tab. To do this, you must first publish the component as a portlet, place the portlet on a page, then click the Personalize link in the portlet header.

This section discusses how to run a component using these various methods. It includes the following subsections:

- Running a Component as a Full Page
- Running a Component as a Portlet
- Running a Component through the Personalization Form

---

**Note:** OracleAS Portal uses the Oracle Internet Directory for identity management, serving as the repository for users and groups. In Oracle Internet Directory, groups are uniquely identified by their distinguished name (DN). Each group has a unique DN, though many groups can share a common name, in the same way that two people can share a common name, yet have completely different lineage (that is, John Smith and John Doe). When working within the portal, groups created from within that portal are displayed simply with their common names. However, when the portal references a group from some other location in the Oracle Internet Directory—such as a group from some other portal associated with the same Identity Management Infrastructure—the DN of the group is displayed to distinguish it from the portal’s locally defined groups.
Performing Test Runs on a Portlet

- Running the Component as a Portlet through the Portlet Personalization Form

**Note:** To run a portlet, you must have at least the Execute privilege on the portlet or the provider that owns it. You must also ensure that the Publish as Portlet check box is selected on the Access tab of the Manage page.

You can run a portlet only if there is a valid portlet version. See "Managing Versions" for more information.

A.6.1 Running a Component as a Full Page

To run a component as a full page, perform the following steps:

1. Click the Navigator link at the top of the Portal Builder page.
2. Click the Providers tab to bring it forward.
3. Click the Locally Built Providers link.
4. Click the name of the Database Provider that owns the portlet.
5. Click the Manage link next to the component you want to run.
   The Develop tab of the Manage page displays.
6. Click Run.
   The component displays in a separate window on a full Web page.

A.6.2 Running a Component as a Portlet

To run a component as a portlet, perform the following steps:

1. Click the Navigator link at the top of the Portal Builder page.
2. Click the Providers tab to bring it forward.
3. Click the Locally Built Providers link.
4. Click the name of the Database Provider that owns the portlet.
5. Click the Manage link next to the component you want to run.
   The Develop tab of the Manage page displays.
6. Click Run as Portlet.
   The portlet displays in a separate browser window in a smaller-than-full-page format.

A.6.3 Running a Component through the Personalization Form

To run a component through the full page personalization form, perform the following steps:

1. Click the Navigator link at the top of the Portal Builder page.
2. Click the Providers tab to bring it forward.
3. Click the Locally Built Providers link.
4. Click the name of the Database Provider that owns the portlet.
5. Click the Manage link next to the component you want to run.
The Develop tab of the Manage page displays.

6. Click Personalize.

The personalization form displays.

7. Personalize your settings and click the Run button.

The component displays in a separate browser window on a full Web page.

### A.6.4 Running the Component as a Portlet through the Portlet Personalization Form

To run a component as a portlet through the portlet personalization form, perform the following steps:

1. Publish the component as a portlet by performing the following steps:
   a. Click the Navigator link at the top of the Portal Builder page.
   b. Click the Providers tab to bring it forward.
   c. Click the Locally Built Providers link.
   d. Click the Database Provider that owns the component with which you want to work.
   e. Click the Manage link next to the portlet with which you want to work.
   f. On the Manage page, click the Access tab to bring it forward.
   g. Under the Portal Access heading, verify that Publish as Portlet is selected.

   Note: The Publish As Portlet check box is available only when the provider has been configured appropriately. For more information, see "Exposing a Provider".

   h. Click Close to save your change and return to the Portal Navigator.

2. Add the portlet to a page by performing the following steps:
   a. Go to the page where you want to place the portlet.
   b. Click the Edit link at the top of the page.
   c. Go to a portlet region on the page.
   d. Click the Add Portlet icon.
   e. Enter the portlet’s display name in the Portlet Repository’s Search field, and click Go.

      If you do not remember the exact name, you can drill to the portlet’s location within the Portlet Repository. Typically, you’ll find newly-created portlets under the Portlet Staging Area node. Click the node, then click the portlet’s host provider name.

   f. Click the portlet to add it to the Selected Portlets list.
   g. Click OK to add the portlet and return the page where you have placed it.

3. Click the Personalize link in the portlet header.

   If this link does not display, edit the host region’s properties to include it (in Page Edit mode, click the Edit Region icon at the top of the region).

4. Personalize your settings and click OK.
The portlet displays with your personalizations.

### A.6.5 Running in Batch Mode

Developers can add a Batch button to a personalization form that enables end users to run the component in batch mode. Batch mode is asynchronous, allowing users to run a transaction in the background, freeing their systems for other tasks. You can place a Batch button on the personalization forms for charts and reports. Batch processing is useful if the component is based on a large amount of data, or if you anticipate that the portlet will display many rows of data.

When you execute a job in batch mode, OracleAS Portal displays a page indicating that the job was submitted to the batch queue, and provides a number for identifying the job.

This section provides information on how to add the Batch button to an existing component, and the parameters that should be pre-set by the database administrator in the `init.ora` file to enable batch processing. It includes the following subsections:

- Setting `init.ora` Parameters for Batch Jobs
- Adding a Batch Button to an Existing Component

#### A.6.5.1 Setting `init.ora` Parameters for Batch Jobs

To enable end users to execute jobs in batch mode, the database administrator should review the parameters in the `init.ora` file for the Oracle database where OracleAS Portal is installed. If these parameters are not set correctly, even though batch jobs may be sent to the batch queue, they may not run.

Table A–35 provides some suggested settings.

<table>
<thead>
<tr>
<th><code>init.ora</code> Parameter and Setting</th>
<th>Specifies</th>
</tr>
</thead>
<tbody>
<tr>
<td>job_queue_processes=2</td>
<td>Two background processes.</td>
</tr>
<tr>
<td>job_queue_intervals=60</td>
<td>The processes wake up every 60 seconds.</td>
</tr>
<tr>
<td>job_queue_keep_connections=TRUE</td>
<td>Sleep, don't disconnect.</td>
</tr>
</tbody>
</table>

#### A.6.5.2 Adding a Batch Button to an Existing Component

To add a Batch button to an existing component, perform the following steps:

1. Navigate to the component management page for the portlet you will work with. For information on how to navigate to this page, see "Navigating to the Component Management Page".

2. If necessary, click the Develop tab to bring it forward.

3. Click the Edit link at the bottom of the tab.

4. Click the Personalization Form Display Options tab to bring it forward.

5. Scroll to the Button Options section at the bottom of the tab.

6. Select the check box next to Batch.

7. Optionally, set the Batch button's display options by performing the following steps:
   a. Enter a display name for the Batch button.
The default is Batch.

b. Specify the location for the button.

Choose from Top, Top and Bottom, and Bottom. Top is the default.

c. Specify button alignment.

Choose from Center, Left, or Right. Center is the default.

8. Click OK to save your changes and return to the Manage page.

9. Click Close to return to the Portal Navigator.

A.7 Referencing the OracleAS Portal Schema

To make the import and export of applications more robust, wherever you can use PL/SQL code, you can use constants in place of the names of the OracleAS Portal Schema, the application schema, and the application name. Table A–36 lists the components that can use these constants and the places where they can be used within the Portlet Builder.

<table>
<thead>
<tr>
<th>Component</th>
<th>Constants</th>
<th>Places to Use in the Portlet Builder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reports</td>
<td>#APP_SCHEMA#</td>
<td>SQL query</td>
</tr>
<tr>
<td></td>
<td>#PORTAL_SCHEMA#</td>
<td>Table Name (#APP_SCHEMA#, #PORTAL_SCHEMA#)</td>
</tr>
<tr>
<td></td>
<td>#APP_NAME#</td>
<td>Advanced PL/SQL section</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOV Name (#APP_NAME#)</td>
</tr>
<tr>
<td>Charts</td>
<td>#APP_SCHEMA#</td>
<td>SQL query</td>
</tr>
<tr>
<td></td>
<td>#PORTAL_SCHEMA#</td>
<td>Table Name (#APP_SCHEMA#, #PORTAL_SCHEMA#)</td>
</tr>
<tr>
<td></td>
<td>#APP_NAME#</td>
<td>LOV Name (#APP_NAME#)</td>
</tr>
<tr>
<td>Calendars</td>
<td>#APP_SCHEMA#</td>
<td>SQL query</td>
</tr>
<tr>
<td></td>
<td>#PORTAL_SCHEMA#</td>
<td>Advanced PL/SQL section</td>
</tr>
<tr>
<td></td>
<td>#APP_NAME#</td>
<td>LOV Name (#APP_NAME#)</td>
</tr>
<tr>
<td>Forms</td>
<td>#APP_SCHEMA#</td>
<td>Table Name (#APP_SCHEMA#, #PORTAL_SCHEMA#)</td>
</tr>
<tr>
<td></td>
<td>#PORTAL_SCHEMA#</td>
<td>Advanced PL/SQL section</td>
</tr>
<tr>
<td></td>
<td>#APP_NAME#</td>
<td>LOV Name (#APP_NAME#)</td>
</tr>
<tr>
<td>Dynamic Page</td>
<td>#APP_SCHEMA#</td>
<td>SQL query</td>
</tr>
<tr>
<td></td>
<td>#PORTAL_SCHEMA#</td>
<td>Advanced PL/SQL section</td>
</tr>
<tr>
<td></td>
<td>#APP_NAME#</td>
<td>LOV Name (#APP_NAME#)</td>
</tr>
<tr>
<td>XML</td>
<td>#APP_SCHEMA#</td>
<td>SQL query</td>
</tr>
<tr>
<td></td>
<td>#PORTAL_SCHEMA#</td>
<td>Advanced PL/SQL section</td>
</tr>
<tr>
<td></td>
<td>#APP_NAME#</td>
<td>LOV Name (#APP_NAME#)</td>
</tr>
<tr>
<td>Hierarchy</td>
<td>#APP_SCHEMA#</td>
<td>Table Name (#APP_SCHEMA#, #PORTAL_SCHEMA#)</td>
</tr>
<tr>
<td></td>
<td>#PORTAL_SCHEMA#</td>
<td>Advanced PL/SQL section</td>
</tr>
<tr>
<td></td>
<td>#APP_NAME#</td>
<td>LOV Name (#APP_NAME#)</td>
</tr>
</tbody>
</table>
These constants should be replaced by the caller at runtime, that is, passed as a parameter to the component when the component is run. When you edit a component in the Portlet Builder, any constants you have used should still appear "as is," that is, as #APP_SCHEMA# or whatever constant you used.

Table A–37 provides some examples of how you can use these constants.

### Table A–37 Examples of the Use of Constants

<table>
<thead>
<tr>
<th>Location in Portlet Builder</th>
<th>Examples</th>
</tr>
</thead>
</table>
| SQL Query                   | Select * from #app_schema#.emp;  
                          | Select * from #portal_schema#.emp;  
                          | Select #app_name# from #portal_schema#.emp; |
| Advanced PL/SQL Section     | Htp.p('#APP_NAME#');  
                          | Htp.p('#APP_SCHEMA#');  
                          | Htp.p('#PORTAL_SCHEMA#');  
                          | Select * from #app_schema#.emp;  
                          | Select * from #portal_schema#.emp;  
                          | Select #app_name# from #portal_schema#.emp; |
| LOV Name                    | #APP_NAME#.dept_lov |
| Table Name                  | #APP_SCHEMA#.EMP  
                          | #PORTAL_SCHEMA#.DEPT |

### A.8 Coding Additional Functionality

The data for OracleAS Portal components is fetched from a database through the use of a SQL query. When you build charts and reports, you can write your own SQL query or use a build wizard to generate it for you. When you build a calendar or frame driver, you must write your own SQL query.

Charts, calendars, and hierarchies all require OracleAS Portal-specific syntax in the SQL statement that creates these portlets. For example, you must identify in the chart SQL statement a table or view column that supplies labels for the chart. You also identify a column that contains numeric data to determine the size of chart bars. The wizards for building charts and other types of components guide you through the specification of any special syntax.

This section explores some of the ways you can build additional functionality into your locally built components. It includes the following subsections:

- Using Bind Variables
A.8.1 Using Bind Variables

You can add bind variables to the SQL query that enables the locally built component to accept user input from a personalization form. Each bind variable corresponds to a column in the table or view on which the component is based. They each create an entry field in the personalization form for the portlet. The user can then choose which data to display in the component.

**Note:** You can also set up bind variables declaratively, through wizards. For an example of how to do this as well as for additional information on how to map the variable to a page parameter, see "Defining Values through Page Parameters".

A bind variable appears in a SQL query as an alphanumeric string preceded by a colon (:var1, :var2, :var3, and so on). For example, the following SQL query creates entry fields for the \textit{SALARY} and \textit{DEPT} columns of the \textit{SCOTT.EMP} table:

\begin{verbatim}
select ename, sal, dept
from scott.emp
where sal = :salary and deptno like :dept
\end{verbatim}

Entering this SQL query creates a personalization form with two text entry fields. The first enables the user to select a salary. The second field selects a department number. OracleAS Portal uses the values that the user types in the personalization form entry fields to create the output.

You do not need to know SQL to specify bind variables. You can identify columns that will accept parameters in the \textit{Column} field in the \textit{Personalization Form Display Options} step of several portlet build wizards.

A.8.2 Specifying Hyperlinks in the SQL Query

You can specify hyperlinks in the SQL query. These links jump from column values that are displayed in the component to other components or Web pages. For example, you can link employee names on a department chart to individual reports containing information about each employee on the chart.

Following is an example of SQL-based report query with a link to another database portlet:

The following query creates a report that displays all employees from the \textit{SCOTT.EMP} table. Values in the employee ID number column (\textit{EMPNO}) link to \textit{Example_Report} in the \textit{SCOTT} schema. Note that the link is specified within an HTML anchor and that the anchor precedes the \textit{EMPNO} column in the SQL query.

**Tip:** You can create a link using the OracleAS Portal link wizard and copy it to your SQL query, as shown in the following example.

\begin{verbatim}
select
\end{verbatim}
In the resulting report, the employee number link will open the new report in the same window by replacing the existing report. If you want to open the report in a new window, then you can add a target attribute to the HTML anchor tag as shown in the following query:

```sql
select 'empno, ename, deptno, mgr, sal
from SCOTT.EMP
```

In the resulting report, the employee number link will open the new report in the same window by replacing the existing report. If you want to open the report in a new window, then you can add a target attribute to the HTML anchor tag as shown in the following query:

```sql
select 'empno, ename, deptno, mgr, sal
from SCOTT.EMP
```

Figure A–70 shows the resulting report.

### A.8.3 Writing Event Handlers for Items on Forms

You can code JavaScript and PL/SQL event handlers to personalize the layout and operation of items on forms. An in-depth knowledge of JavaScript and PL/SQL is required to successfully include event handlers in your forms. JavaScript and PL/SQL event handlers can be used as follows:

- You can use JavaScript event handlers to personalize the behavior of form items, such as buttons, check boxes, lists, text boxes, and the like. Your browser must support the JavaScript version you use.

- You can use PL/SQL event handlers to personalize the behavior of buttons.

If you code both JavaScript and PL/SQL event handlers for the same button, be careful to avoid potential conflicts between the execution of the two scripts.

Refer to your JavaScript or PL/SQL documentation for descriptions of the events that are available to you in OracleAS Portal.

This section describes how to add event handlers through the portlet build wizards. It includes the following subsections:

- Writing a JavaScript Event Handler for an Item on a Form
A.8.3.1 Writing a JavaScript Event Handler for an Item on a Form

To write a JavaScript event handler for an item on a form, perform the following steps:

1. Edit the form that includes the item for which you want to write an event handler by performing the following steps:
   a. Go to the Manage page for the relevant form component.
      For information on navigating to this page, see "Navigating to the Component Management Page".
   b. If necessary, click the Develop tab to bring it forward.
   c. Click the Edit link toward the bottom of the tab.
2. Click the Formatting and Validation Options icon to bring that tab forward.
3. In the left frame, click the name of the item for which you want to write a JavaScript event handler.
4. In the right frame, scroll to the JavaScript Event Handlers section.
5. From the JavaScript Event Handlers list, choose the required event, and enter the JavaScript to be executed when the event occurs, as shown in Example A–1, Example A–2, and Example A–3.

Example A–1 Converting a Value to Uppercase

To convert a value in a text box or text area input field to uppercase whenever the field loses focus, choose the onBlur event and enter the following JavaScript:

```javascript
this.value = this.value.toUpperCase();
```

Example A–2 Confirming Submission of a Form

To add a confirmation when the end user clicks a button before submitting a form, choose the onClick event and enter the following JavaScript:

```javascript
return confirm('Are you sure you want to submit the form?');
```

This displays a prompt dialog where the end user can click OK or Cancel; clicking OK submits the form.

Example A–3 Checking the Data Type of a Value

To verify that a value entered by the end user is a number only, choose the onChange event and enter the following JavaScript:

```javascript
if (isNaN(this.value)) {
alert('Please enter a valid number.');
}
```

A.8.3.2 Writing a PL/SQL Event Handler for a Button on a Form

To write a PL/SQL event handler for a button on a form, perform the following steps:

1. Edit the form that includes the button for which you want to write an event handler by performing the following steps:
   a. Go to the Manage page for the relevant form component.
For information on navigating to this page, see "Navigating to the Component Management Page".

b. If necessary, click the Develop tab to bring it forward.

c. Click the Edit link toward the bottom of the tab.

2. Click the Formatting and Validation Options icon to bring that tab forward.

3. In the left frame, click the name of the button for which you want to write a PL/SQL event handler.

4. In the right frame, scroll to the PL/SQL Button Event Handler section.

5. Define either of the following:
   - A pre-defined event: Choose an event for the button from the PL/SQL Button Event Handler list.
   - A custom event: Choose Custom from the PL/SQL Button Event Handler list, and type the associated PL/SQL code.

**A.8.4 Using PL/SQL to Get and Set Values in a Form**

You can access and modify form field values using the methods of the form's session storage object contained in the variable p_session. To get a value, you must use type-specific get methods on p_session. This p_session has the following get functions for the data types NUMBER, VARCHAR2, and DATE:

- get_value_as_NUMBER
- get_value_as_VARCHAR2
- get_value_as_DATE

You must provide the block name and attribute name as arguments to these functions. For a single-block form, the block name is DEFAULT. For a master-detail form, the block name is MASTER_BLOCK or DETAIL_BLOCK.

For the detail block, you must also provide the row index. The attribute name is the column name prefixed by A_.

To get the value of DEPTNO in a single block form, you would code:

```plsql
declare
    my_deptno number;
begin
    my_deptno := p_session.get_value_as_NUMBER(
        p_block_name => 'DEFAULT',
        p_attribute_name => 'A_DEPTNO');
end;
```

For example, if you're doing this in the third record of a detail block, use:

```plsql
my_deptno := p_session.get_value_as_NUMBER(
    p_block_name => 'DETAIL_BLOCK',
    p_attribute_name => 'A_DEPTNO',
    p_index => 3);
```

To set a field value, use p_session.set_value. This p_session.set_value function takes the same arguments as the get_value functions, with the addition of p_value for specifying the value (p_value is overloaded for any data type). Here, set_value is a procedure, and does not return a value. For example, to set the value of DEPTNO, use:
Creating Portlets with the Portlet Builder

You can use the session storage object (p_session) in any button event handler and in the following places in the Advanced PL/SQL step of the Forms wizard (see also Table A–6):

- Before page
- After page
- Before processing
- After processing

Note that the Before form and After form handlers do not have access to the session storage object.

If you define custom code for a button mapped to an Insert, Update, or Delete event, you must call the procedure doInsert, doUpdate, or doDelete at an appropriate place in your code to perform the button’s default function. For example, if you programmatically set a field value in an Insert button, call doInsert after p_session.set_value to perform the insert into the table.

A.8.5 Using PL/SQL to Get or Set Cookies in a Form or Report

When you use PL/SQL in OracleAS Portal, you have access to a variety of packages that are part of the PL/SQL Gateway. The gateway is delivered as part of OracleAS Portal. One of these is the package owa_cookie.

This package contains data types, procedures, and functions that enable you to send HTTP cookies to and get them from the client’s browser. HTTP cookies are opaque strings sent to the browser to maintain state between HTTP calls. State can be maintained throughout the client’s session—longer if an expiration date is included.

Maintaining a state means that the client can be identified throughout a session, allowing for the execution of transactions. So, for example, shopping applications can store information about currently-selected items; for-fee services can send back registration information, freeing the client from retyping a user ID on the next connection; and sites can store users preferences on the client, and in turn, have those clients supply those preferences every time it reconnects to that site.

The owa_cookie package contains subprograms and data types that you can use to set and get cookie values. Table A–38 lists some owa_cookie subprograms and data types, and describes how each is used.

<table>
<thead>
<tr>
<th>Subprogram or Data Type</th>
<th>Used to</th>
</tr>
</thead>
<tbody>
<tr>
<td>owa_cookie.cookie data type</td>
<td>Contain cookie name-value pairs</td>
</tr>
<tr>
<td>owa_cookie.get function</td>
<td>Get the value of the specified cookie</td>
</tr>
<tr>
<td>owa_cookie.get_all procedure</td>
<td>Get all cookie name-value pairs</td>
</tr>
<tr>
<td>owa_cookie.remove procedure</td>
<td>Remove the specified cookie</td>
</tr>
<tr>
<td>owa_cookie.send procedure</td>
<td>Generate a “Set-Cookie” line in the HTTP header</td>
</tr>
</tbody>
</table>
Sessions are used by the Web Request Broker to maintain persistent states within gateways through multiple accesses over a period of time. Since the PL/SQL Gateway is unique in connecting to the database and all the states are maintained within the database, the concept of sessions does not apply to the PL/SQL Gateway. Instead, cookies can be used to maintain persistent state variables from the client browser.

A.8.6 Defining Values through Page Parameters

For many components, you can set values for columns using page parameters. To do this, you need to do the following:

- Indicate that a column is customizable in the component’s Personalization Form Display Options page.
- Enable parameters for the page group that will host the component.
- Add a page parameter to the page that will host the component.
- Map the page parameter to the portlet parameter.

This section provides a brief overview of this process. We'll add a customizable value—a parameter or bind variable—to a report and map that parameter to the page that will host the portlet component.

To set up a parameter and use it on a portal page, perform the following steps:

1. Create a report as described in "Building Reports Declaratively".
2. When you reach the step covering Personalization Form Display Options, select the Value Required check box, choose a column, assign a prompt to it, and make it public (Figure A–71).

3. Complete the creation of your report.
4. Go to the Page Groups portlet (typically, on the Build tab of the Portal Builder.)
5. In the **Page Group** field, select the page group that will host the report and click the **Edit** button to edit the page group’s properties.

6. Click the **Configure** tab to bring it forward, and scroll down to the **Parameters and Events** section of the tab.

7. Click the **Edit** link.

8. Select the **Enable Parameters and Events** check box.

9. Click **OK**, then click **Close**.

10. In the Page Groups portlet, click the Browse icon next to the **Name** field under **Edit a Page**.

11. In the resulting secondary window, drill to the page on which you want to place the portlet, and click the **Return Object** link next to it.

12. Click the **Edit** button next to the now-populated **Name** field.

13. Add the new report portlet to the page by performing the following steps:
   a. Click the Add Portlet icon over a region.
   b. In the Portlet Repository, click the **Portlet Staging Area** node.
   c. Click the name of the provider that owns the portlet.
   d. Click the portlet to add it to the **Selected Portlets** list.
   e. Click **OK** to return to the host page.

   The portlet will display with the following error message: *Error: Required field not set yet - emp.deptno (WWV-14900)*. Once you define the page parameter and wire it to the portlet parameter you can provide a value, and this message will no longer display.

14. In the page tool bar at the top of the page, click the **Page: Properties** link.

15. On the resulting page, click the **Parameters** tab to bring it forward.

   This tab displays only when parameters and events are enabled for the page’s page group.

16. In the **Parameter Name** field, enter a name for the page parameter you will wire to the parameter (bind variable) you created for your component (Figure A–72).

![Figure A–72 The Parameter Name Field on the Parameters tab](image)

17. Click **Add**.

18. Optionally, go to the **Page Parameter Properties** section and configure the parameter (Figure A–73) by performing the following steps:
a. Enter a display name, which identifies the parameter to other users.

b. Enter a default value.

c. Select whether to allow users to change the value of the parameter when they personalize the page.

d. Enter a description of the parameter.

**Figure A–73  Defining Page Parameter Properties**

**Page Parameter Properties**
Enter a display name, default value, and a description for each parameter. Decide whether to allow users to change the value of each parameter when they personalize the page. You may also delete parameters, or change the order in which they are displayed when users personalize the page.

<table>
<thead>
<tr>
<th>Name</th>
<th>Display Name</th>
<th>Default Value</th>
<th>Personalizable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>department_no</td>
<td>department_no</td>
<td>[0]</td>
<td>✔️</td>
<td></td>
</tr>
</tbody>
</table>

19. Go to the **Portlet Parameter Values** section, and expand the node next to the portlet you added to the page in Step 13 (Figure A–74).

**Figure A–74  Expanding a Portlet Node under Portlet Parameter Values**

**Portlet Parameter Values**
Expand a portlet to view its parameters and specify how to set the values of those parameters. You can map portlet parameters to page parameters, system variables, or constant values.

20. From the drop-down list next to your portlet parameter, select **Page Parameter** (Figure A–75).

**Figure A–75  Selecting the Type of Mapping to Use with a Portlet Parameter**

**Portlet Parameter Values**
Expand a portlet to view its parameters and specify how to set the values of those parameters. You can map portlet parameters to page parameters, system variables, or constant values.

21. By default, the next drop-down list should display the relevant page parameter display name. If it doesn’t, select the relevant name from the list (in our example, the relevant display name is **Department Number**).

   This maps the parameter you defined for your portlet (on the Personalization Form Display Options page) to the parameter you have defined for the page (on the Parameters tab).

22. Click **OK**.
23. If you provided a default value for the page parameter in Step 18b, the report portlet will display with results relevant to the parameter, for example, with results for the specified department number (Figure A–76).

Figure A–76  Report Results for Department 20

<table>
<thead>
<tr>
<th>Department Number</th>
<th>Manager</th>
<th>Employee Job</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>JONES</td>
<td>MANAGER</td>
</tr>
<tr>
<td>20</td>
<td>SCOTT</td>
<td>ANALYST</td>
</tr>
<tr>
<td>21</td>
<td>ADAMS</td>
<td>CLERK</td>
</tr>
<tr>
<td>20</td>
<td>FORD</td>
<td>ANALYST</td>
</tr>
</tbody>
</table>

All rows for company Presidents are highlighted in yellow.

If you did not provide a default value, you can provide one by clicking the Personalize link in the portlet header (if the parameter was made customizable in Step 18c) and providing a value under Query Options (Figure A–77).

Figure A–77  Personalizing Page and Portlet Parameter Values

A.9 Using Shared Components to Create a Look and Feel

Each portlet build wizard contains options that enable you to define a look and feel for the portlet's content. For example, you can choose the colors of the rows in a report; the font type, color, and size for chart labels; and the height, width, and order of entry fields in a form. In addition to its built-in look-and-feel capabilities, OracleAS Portal offers advanced features through the Shared Components provider for the creation of a more personalized look and feel.

Through the Shared Components provider, you can build JavaScripts for use in your wizard-built forms; define custom colors, fonts, and images; and apply User Interface Templates to portlets.

The Shared Components provider is located in the Portal Navigator on the Providers tab under Locally Built Providers.

OracleAS Portal includes a default set of shared components. These are system type components. The components you develop yourself under the Shared Components provider are user type components. You can edit, export, and delete a user-type shared component, but not a system-type. To edit a system type, you must first copy it, then edit the copy.
This section explores the capabilities of the Shared Components provider and describes how to put them to use. It includes the following subsections:

- Granting Access to Shared Components
- Using JavaScript to Create Field- and Form-Level Validation
- Creating Color Definitions
- Creating Image Definitions
- Creating Font Definitions
- Using User Interface Templates

### A.9.1 Granting Access to Shared Components

Access privileges to the Shared Components provider define the actions you can perform on shared components. Table A–39 lists and describes the access privileges that are relevant to the Shared Components provider.

<table>
<thead>
<tr>
<th>Access Privilege</th>
<th>Enables You to</th>
</tr>
</thead>
</table>
| Manage           | - Grant shared component access privileges to other users or groups.  
                  | - Create a new shared component.  
                  | - Copy a System type shared component to create a new User type.  
                  | - Edit any User type shared component.  
                  | - Delete any User type shared component.  
                  | - Export any User type shared component to another schema or database.  
| Create           | - Create a new shared component.  
                  | - Copy a System type shared component to create a new User type.  

By default, all users who are members of the DBA or PORTAL_DEVELOPERS groups have Manage shared component access privileges.

Access privileges are granted for all shared components. Access privileges cannot be granted on a shared component type, such as all JavaScripts, nor on an individual shared component, such as a particular JavaScript. Access must be granted to all shared components or none of them.

To grant access privileges to the Shared Components provider to a user or group, perform the following steps:

1. In the Navigator, click the Providers tab to bring it forward.
2. At the root level of the Providers tab, click the Locally Built Providers link.
3. In the Actions column for the Shared Components provider, click the Grant Access link.
4. In the Grantee field, enter the name of the user or group that you want to allow to access the shared components.

   Optionally, click the Browse Users or Browse Groups icon and select from the list provided.
5. Choose the level of access to grant to the user or group from the list of available privileges.
   See Table A–39 for a list of relevant privileges.

6. Click Add.
   The user or group you specified now appears in the Change Access section at the bottom of the page.

7. (Optional) To modify an access privilege, choose a new privilege next to the user or group in the Change Access section.

8. (Optional) To remove a privileges, click the Delete icon next to the user or group in the Change Access section.

9. Click OK.

A.9.2 Using JavaScript to Create Field- and Form-Level Validation

OracleAS Portal provides tools for you to create JavaScripts that perform field- and form-level validation on entry fields in forms. Field-level validation is performed when the end user causes the onBlur condition to occur after entering a value in an entry field, for example, when tabbing to another entry field. Form-level validation occurs after the user enters a value in an entry field and submits all values on the page, for example, when clicking an OK button.

This section provides a few guidelines for using JavaScript to build field- or form-level validation in a form and describes how to create and add JavaScript to your form. It contains the following subsections:

■ Guidelines for Writing Field- or Form-Level Validation JavaScript
■ Creating JavaScript under the Shared Components Provider
■ Adding JavaScript to a Form

A.9.2.1 Guidelines for Writing Field- or Form-Level Validation JavaScript

Consider the following guidelines when writing a field- or form-level validation JavaScript:

Note: You must have at least the Create shared component access privilege to create a JavaScript.

You must have the Manage shared component access privilege to edit a JavaScript.

■ All validation routines should be written as functions and return either TRUE or FALSE values.
■ The routine should display an alert message to users if the element (entry field) being validated contains an invalid value.
■ The routine should bring focus (position the cursor) to the entry field where the user entered the incorrect value flagged by the JavaScript.

Example A–4 demonstrates the use of JavaScript to perform field-level validation.

Example A–4  Example JavaScript

```javascript
1-> function isNumber(theElement)
```
The JavaScript presented in Example A–4 performs field-level validation tasks in the following sequence:

1. Identifies the name of the function and the entry field being validated.
2. Checks whether the absolute value of the entry field is a number. A relevant JavaScript function that signifies that a value is not a number is: isNaN
3. If the value in the entry field is not a number, the user is alerted with the message, “Value must be a number.”
4. The routine brings focus to the entry field.

A.9.2.2 Creating JavaScript under the Shared Components Provider

To create JavaScript under the Shared Components Provider, perform the following steps:

1. In the Navigator, click the Providers tab to bring it forward.
2. At the root level of the Providers tab, click the Locally Built Providers link.
3. Click the link for the Shared Components provider.
4. Click the Javascript link next to Create New…
5. In the JavaScript Name field, enter a descriptive name for the JavaScript. For example, enter NotNull for a JavaScript that ensures there are no null values in an entry field.
6. In the Language field, enter the language in which the JavaScript will be written. For example, enter JavaScript1.1 or JavaScript1.2.
7. Click Next.
8. Enter or copy your JavaScript into the field provided.
9. Click Finish.

To edit the JavaScript you just created, drill to the relevant JavaScript (in the Portal Navigator: Providers tab: Locally Built Providers link: Shared Components: JavaScripts: your JavaScript), and click the Edit link next to the relevant JavaScript.

A.9.2.3 Adding JavaScript to a Form

Once you have created JavaScript under the Shared Components provider, it is automatically added to the selection list that is available in the Build Form wizard. For example, when you create a form, you can select a JavaScript that you created once you reach the Formatting and Validation Options page. The list of available JavaScripts is located on this page under the Validation Options section (for more information, see “Building Forms Declaratively”).

```javascript
{  
  if (isNaN(Math.abs(theElement.value)))
  |
  | alert("Value must be a number.");
  | theElement.focus();
  | return false;
  |
  return true;
}
```
A.9.3 Creating Color Definitions

Creating a color definition is an opportunity for you to provide a meaningful name to a color you plan to use in your portal. A color definition is an association between a color name and its hexadecimal value. For example, you might use a standard red as your corporate color. Using your ability to define colors, you could select that red and give it a meaningful name within OracleAS Portal, such as `standard_red`, `company_red`, or even `<your company name>_red`.

You associate a color value in the form `#XXXXXX`, where X is a value in the range 0-9 or A-F, with any name you choose. The color names you define are used in fonts, page backgrounds, and other elements of OracleAS Portal portlets.

```
Note: You must have at least the Create shared component access privilege to create a color definition.
You must have at least the Manage shared component access privilege to edit a color definition.
```

To create a color definition, perform the following steps:

1. In the Navigator, click the **Providers** tab to bring it forward.
2. At the root level of the Providers tab, click the **Locally Built Providers** link.
3. Click the link for the **Shared Components** provider.
4. Click the **Color** link next to **Create New…**
5. In the **Color Name** field, enter the name you want to give the color.
   You can identify a color by any name you choose; for example, `My_Blue_Color`.
6. In the **Color Value** field, enter the hexadecimal value for the color, for example, `#FF0000` for a shade of red.
   Hexadecimal values must be prefaced by the `#` character. You can click a color in the palette to automatically enter its hexadecimal value in the **Color Value** field.
7. (Optional) To preview a color value, click **Preview**.
8. When you are satisfied with your color definition, click **Create**.
   The page updates with a link, which you can click to edit the color definition. If you do not want to edit the color definition at this time, click **Close**.

A.9.4 Creating Image Definitions

Creating an image definition is an opportunity for you to provide a meaningful name and type to an image you plan to use in your portal. An image definition is an association between an image name and the name of the file containing the image. You can specify any name you choose.

```
Note: You must have at least the Create shared component access privilege to create an image definition.
You must have at least the Manage shared component access privilege to edit an image definition.
```

To create an image definition, perform the following steps:
1. In the Navigator, click the Providers tab to bring it forward.
2. At the root level of the Providers tab, click the Locally Built Providers link.
3. Click the link for the Shared Components provider.
4. Click the Image link next to Create New…
5. In the Image Name field, enter the name you want to give the image.
   You can identify an image by any name you choose; for example, SiteLogo.
6. In the Image Filename field, enter the name and extension of the file containing
   the image.
   For example, enter logo.gif. The image must be located in a directory mapped
   to the OracleAS Portal virtual directory /images/.

   Note: The /images/ virtual directory path is set in the Oracle
   HTTP Listener psql.conf file.

7. From the Image Type list, choose an image type, for example Icon 24x24.
   The type you choose will display next to the image in the Type column of the
   Portal Navigator.
8. Click Create.
   The page updates with a link, which you can click to edit the image definition. If
   you do not want to edit the image definition at this time, click Close.

A.9.5 Creating Font Definitions

Creating a font definition is an opportunity for you to provide a meaningful name to a
font you plan to use in your portal. A font definition is an association between the
name of a font face and any descriptive name you choose to give it. For example, your
company may have identified a font to be used in all public documents. Using your
ability to create a font definition, you could identify that font in OracleAS Portal with a
custom name, such as <your company name>_font. The fonts you define are used
for text that appears in portlets.

   Note: You must have at least the Create shared component access
   privilege to create a font definition.
   You must have at least the Manage shared component access
   privilege to edit a font definition.

To create a font definition, perform the following steps:
1. In the Navigator, click the Providers tab to bring it forward.
2. At the root level of the Providers tab, click the Locally Built Providers link.
3. Click the link for the Shared Components provider.
4. Click the Font link next to Create New…
5. In the Font Name field, enter the name you want to give the font.
   You can identify a font by any name you choose; for example, web_banner_font.
6. In the Font Value field, enter the name of a font, for example Arial.

You can specify the name of any font that is supported by a Web browser. If you specify a font that is not supported, the Web browser will use its own default font.

You can specify alternative fonts by separating them with commas in the Font Value field; for example, Times New Roman, Times. In this example, if the user’s Web browser does not support the Times New Roman font, it will use the Times font instead.

7. Click Create.

The page updates with a link, which you can click to edit the font definition. If you do not want to edit the font definition at this time, click Close.

A.9.6 Using User Interface Templates

Use User Interface (UI) templates to provide a header and footer to an application. UI templates can be applied to Portlet Builder components. By applying a template, you can automatically specify a title, a title background, links to home and help pages, and background colors and images.

UI templates are good for standardizing the overall look and feel of many Portlet Builder components. For example, you can design a UI template for a provider that includes the company logo in the heading, the name of the company in the title, and a common background image. By ensuring every component in the application uses the same UI template, you impose a standard appearance.

Note: In earlier releases you could apply UI templates to pages as well as components within OracleAS Portal. This functionality of UI templates has been replaced by HTML Skins. Refer to Oracle Application Server Portal User’s Guide.

This section provides information about the two types of user interface template—structured and unstructured—and describes generally how to use them to create a look and feel. It contains the following subsections:

- Building a Structured User Interface Template
- Building an Unstructured User Interface Template

A.9.6.1 Building a Structured User Interface Template

You can use structured UI templates only with portlets. You create them with a wizard. In the wizard, you specify images, text, and layout elements that you want to apply to every portlet that uses the template.

You cannot use your own HTML code to extend the template beyond the pre-identified attributes. To achieve greater flexibility, you may want to build an unstructured user interface template (see "Building an Unstructured User Interface Template").

To build a structured user interface template, perform the following steps:

1. In the Navigator, click the Providers tab to bring it forward.
2. At the root level of the Providers tab, click the Locally Built Providers link.
3. Click the link for the Shared Components provider.
4. Click the User Interface Template link next to Create New...
5. Click the **Structured UI Template** link.

6. In the **Template Name** field, enter a name for the template.
   Make the name as descriptive as possible. This is the name users will see when applying a UI template to a portlet during the build process. For example, if the template will be applied to all portlets created for a scheduling provider, you could name it **Schedule_Template**.

7. Select other options to refine the look and feel of the template.
   For example, you can choose an image to appear in the upper left corner of the template and a background image to display behind the portlet, as shown in **Figure A–78**.

   **Figure A–78  Structured UI Template with Cloud Image**

   ![My Template - Notescape](image)

   If you have a question about an option, click the help icon. Leave an option blank if you do not want to include it in your template.

8. (Optional) Click **Preview** to open a new browser window that displays the UI template.
   You can reselect options then click **Preview** again to see how the changes affect the look of the template.

9. When you are satisfied with your template, click **Create**.
   The page updates with a link, which you can click to edit the template. If you do not want to edit the template at this time, click **Close**.

### A.9.6.2 Building an Unstructured User Interface Template

Unstructured user interface templates are based on HTML code that you supply. Because you are writing your own HTML code, you can create a more elaborate and sophisticated unstructured UI template than you can a structured UI template.

To create an unstructured user interface template, you first write HTML code to create a Web page. You can also copy this code into OracleAS Portal from another source, such as a Web page editor. Once entered into OracleAS Portal, you edit the HTML code to add substitution tags. When the HTML code executes, the substitution tags embed portlets, titles, and other elements into the Web page. For example, you can add a **#BODY#** tag that adds a portlet such as a chart or report to the original Web page background. For a list of all the substitution tags you can include in your HTML code, see **Table A–40**.
In an unstructured template, you can use `<ORACLE>` tags to include SQL statements or PL/SQL blocks. You can also include HEAD elements such as custom JavaScript, cascading style sheet references, and META tags. You can use special substitution tags for integrating page metadata to embed PL/SQL scripting.

To build an unstructured user interface template, perform the following steps:

1. In the Navigator, click the Providers tab to bring it forward.
2. At the root level of the Providers tab, click the Locally Built Providers link.
3. Click the link for the Shared Components provider.
4. Click the User Interface Template link next to Create New…
5. Click the Unstructured UI Template link.
6. In the Template Name field, enter a name for the template.
   Because this is the name users will see when applying a UI template to a page or portlet during the build process, you should make the name as descriptive as possible. For example, if the template will be applied to all portlets created for a calendar, you could name it Calendar_Template.
7. In the Template Definition field, enter or paste the HTML code you want to use as the basis for your unstructured user interface template.
   The HTML code you supply should create a Web page.
8. Embed substitution tags in the HTML code in the location where you want the items associated with the tags to appear in the finished template.
   Table A–40 lists and describes the tags you can use. For example, you may want to embed a #BODY# substitution tag in the code. #BODY# adds the main body of the page, such as page content or a portlet, to the Web page when the HTML code executes. If the HTML source code divides a page into two frames, you can embed the #BODY# tag in different places in the code, causing the portlet to display in the left frame or the right frame.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Value set by</th>
</tr>
</thead>
<tbody>
<tr>
<td>#BODY#</td>
<td>The portlet itself or the page's portlets.</td>
</tr>
<tr>
<td>#IMAGE_PREFIX#</td>
<td>The OracleAS Portal images directory as specified in the plsql.conf configuration file.</td>
</tr>
<tr>
<td>#USER#</td>
<td>The user name of the user who is currently logged on.</td>
</tr>
</tbody>
</table>
| #USER.FULLNAME# | The full name of the user who is currently logged on.  
The full name includes the user's first, middle, and last names. |
| #VERSION#    | The version of this installation of OracleAS Portal.                        |
| #HELPSCRIPT# | The JavaScript function used to open a window to display the online help. If you include #HELPLINK# in your template, you should also include this tag. |
| #HELPLINK#   | Used to display the help text for the component and its personalization form that you specified in the component creation wizard. |
| #DIRECTION#  | The direction of character layout (left-to-right or right-to-left). This value is set based on the language. |
| #ALIGN_LEFT# | Align text to the left.                                                      |
### Table A–40 (Cont.) Unstructured UI Template Substitution Tags

<table>
<thead>
<tr>
<th>Tag</th>
<th>Value set by</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#ALIGN_RIGHT#</code></td>
<td>Align text to the right</td>
</tr>
<tr>
<td><code>#PAGE.STYLE#</code></td>
<td>The HTML LINK element that references the page’s cascading style sheet.</td>
</tr>
<tr>
<td></td>
<td>This tag is allowed only in the document HEAD.</td>
</tr>
<tr>
<td><code>#PAGE.STYLE.URL#</code></td>
<td>The URL of the page’s cascading style sheet.</td>
</tr>
<tr>
<td><code>#PAGE.BASE#</code></td>
<td>The HTML base element for the base URL of the document.</td>
</tr>
<tr>
<td></td>
<td>This tag is allowed only in the document HEAD.</td>
</tr>
<tr>
<td><code>#PAGE.BASE.URL#</code></td>
<td>The base URL.</td>
</tr>
<tr>
<td><code>#PAGE.BGIMAGE#</code></td>
<td>The HTML source for the whole page background image.</td>
</tr>
<tr>
<td><code>#PAGE.BGCOLOR#</code></td>
<td>The HTML source for the whole page background color.</td>
</tr>
<tr>
<td><code>#PAGE.SUBPAGELINK#</code></td>
<td>The URL for a sub-page link.</td>
</tr>
<tr>
<td><code>#PORTAL.HOME#</code></td>
<td>The HTML image hyperlink to the portal home page.</td>
</tr>
<tr>
<td><code>#PORTAL.HOME.URL#</code></td>
<td>The URL of the portal home page.</td>
</tr>
<tr>
<td><code>#PORTAL.HOME.IMAGE#</code></td>
<td>The image used for the portal home page link.</td>
</tr>
<tr>
<td><code>#PORTAL.HOME.LABEL#</code></td>
<td>The text used for the portal home page link.</td>
</tr>
<tr>
<td><code>#PORTAL.NAVIGATOR#</code></td>
<td>The HTML image hyperlink to the Navigator.</td>
</tr>
<tr>
<td><code>#PORTAL.NAVIGATOR.URL#</code></td>
<td>The URL of the Navigator.</td>
</tr>
<tr>
<td><code>#PORTAL.NAVIGATOR.IMAGE#</code></td>
<td>The image used for the Navigator link.</td>
</tr>
<tr>
<td><code>#PORTAL.NAVIGATOR.LABEL#</code></td>
<td>The text used for the Navigator link.</td>
</tr>
<tr>
<td><code>#PORTALHELP#</code></td>
<td>The HTML image hyperlink to the online help.</td>
</tr>
<tr>
<td><code>#PORTALHELP.URL#</code></td>
<td>The URL of the online help.</td>
</tr>
<tr>
<td><code>#PORTALHELP.IMAGE#</code></td>
<td>The image used for the online help link.</td>
</tr>
<tr>
<td><code>#PORTALHELP.LABEL#</code></td>
<td>The text used for the online help link.</td>
</tr>
<tr>
<td><code>#PORTAL.LOGOUT#</code></td>
<td>The HTML text hyperlink to the logout URL.</td>
</tr>
<tr>
<td><code>#PORTAL.LOGOUT.URL#</code></td>
<td>The logout URL.</td>
</tr>
<tr>
<td><code>#PORTAL.LOGOUT.LABEL#</code></td>
<td>The text used for the logout link.</td>
</tr>
<tr>
<td><code>#PORTALACCOUNTINFO#</code></td>
<td>The HTML text hyperlink to the account information dialog.</td>
</tr>
<tr>
<td><code>#PORTALACCOUNTINFO.URL#</code></td>
<td>The URL of the account information page.</td>
</tr>
<tr>
<td><code>#PORTALACCOUNTINFO.LABEL#</code></td>
<td>The text used for the account information link.</td>
</tr>
<tr>
<td><code>#PORTALCOMMUNITY#</code></td>
<td>The name of the OracleAS Portal Community Web site.</td>
</tr>
<tr>
<td><code>#PORTALCOMMUNITY.URL#</code></td>
<td>The URL of the OracleAS Portal Community Web site.</td>
</tr>
<tr>
<td><code>#PORTALCOMMUNITY.IMAGE#</code></td>
<td>The image of the OracleAS Portal Community Web site.</td>
</tr>
<tr>
<td><code>#PORTALCOMMUNITY.LABEL#</code></td>
<td>The label of the OracleAS Portal Community Web site.</td>
</tr>
<tr>
<td><code>#PAGE.CUSTOMIZEPAGE#</code></td>
<td>The HTML text hyperlink to the personalize page.</td>
</tr>
</tbody>
</table>
9. (Optional) Click **Preview** to open a new browser window that displays the UI template.

   You can update the code then click **Preview** again to see how the changes affect the look of the template.

10. When you are satisfied with your template, click **Create**.

    The page updates with a link, which you can click to edit the template. If you do not want to edit the template at this time, click **Close**.

### A.10 Example: Building Charts and Reports

This example assumes that you have access to an OracleAS Portal database provider in the SCOTT schema called myCompany_DB_Provider. If you have the appropriate privileges, you can create this provider yourself (for information about how to do this, see "Creating a Provider for Locally Built Portlets"). If you do not have the appropriate privileges to create OracleAS Portal database providers, ask your portal administrator to create the provider for you.

This example includes the following exercises:

- Exercise: Building the Team Details Report
- Exercise: Building the Average Salaries Chart
- Exercise: Building the Team Bonuses Report
- Exercise: Building the Job Profiles Report

#### A.10.1 Exercise: Building the Team Details Report

The Team Details report, shown in Figure A–79, displays a list of employees in the Sales department (department 30).
To build the Team Details report, perform the following steps:

1. In the Navigator, click the Providers tab to bring it forward.
2. At the root level of the Providers tab, click the Locally Built Providers link.
3. Click the link for the provider myCompany_DB_Provider.
   If you do not see this provider, you can create it. For more information, see “Creating a Provider for Locally Built Portlets”.
4. Click the Report link next to Create New…
5. Click the Reports From Query Wizard link.
6. In the Name field, enter <YourName>_team_details.
7. In the Display Name field, enter Team Details.
8. In the Description field, enter team details exercise.
9. In the Portal DB Provider list, MYCOMPANY_DB_PROVIDER should already be selected.
10. Click Next.
11. In the Tables and Views field, enter SCOTT.EMP if necessary.
12. Click Add.
13. Click Next.
14. From the Columns list, select the following:
   ■ EMP.ENAME
   ■ EMP.EMPNO
   ■ EMP JOB
   Click the right arrow button after each selection to move it from Columns to Selected Columns.
15. If necessary, use the up and down arrows to the right of Selected Columns to arrange the columns in the order specified earlier.
16. Click Next.
17. From the Column Name list, choose EMP.DEPTNO.
18. From the Condition list, choose =.
19. In the Value field, enter 30.
20. Click Next.
21. Select Tabular.

22. Click Next.

23. Next to each column, enter the Column Heading Text as indicated in Table A–41.

<table>
<thead>
<tr>
<th>Column</th>
<th>Column Heading Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENAME</td>
<td>Name</td>
</tr>
<tr>
<td>EMPNO</td>
<td>Employee No</td>
</tr>
<tr>
<td>JOB</td>
<td>Job</td>
</tr>
</tbody>
</table>

This will add descriptive labels above columns that appear in your report.

24. Click Next twice, or until you see the Display Options step of the wizard.

25. In both the Full Page Options and Portlet Options sections, select the values shown in Table A–42.

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heading Font Face</td>
<td>Arial</td>
</tr>
<tr>
<td>Heading Font Color</td>
<td>White</td>
</tr>
<tr>
<td>Heading Size</td>
<td>10pt</td>
</tr>
<tr>
<td>Row Text Font Face</td>
<td>Arial</td>
</tr>
<tr>
<td>Row Text Font Color</td>
<td>Black</td>
</tr>
<tr>
<td>Row Text Size</td>
<td>10pt</td>
</tr>
<tr>
<td>Heading Background Color</td>
<td>Slate Gray</td>
</tr>
<tr>
<td>Border</td>
<td>Thin Border</td>
</tr>
</tbody>
</table>

26. Click Finish.

27. Click Run as Portlet to see what your report looks like.

28. Close the browser window where the report is displayed, and click Close to close the component manager and return to the Portal Navigator.

A.10.2 Exercise: Building the Average Salaries Chart

The Average Salaries chart, shown in Figure A–80, displays the average salary for each job title.

Figure A–80  Average Salaries Chart

<table>
<thead>
<tr>
<th>Position</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALYST</td>
<td>3,000</td>
</tr>
<tr>
<td>CLERK</td>
<td>1,187</td>
</tr>
<tr>
<td>MANAGER</td>
<td>3,133</td>
</tr>
<tr>
<td>PRESIDENT</td>
<td>7,295</td>
</tr>
<tr>
<td>SALESMAN</td>
<td>1,400</td>
</tr>
</tbody>
</table>

To build the Average Salaries chart, perform the following steps:
1. In the Navigator, click the Providers tab to bring it forward.
2. At the root level of the Providers tab, click the Locally Built Providers link.
3. Click the link for the provider myCompany_DB_Provider.
4. Click the Chart link next to Create New…
5. Click the Charts From SQL Query link.
6. In the Name field, enter <YourName>_team_average_salary.
7. In the Display Name field, enter Average Salaries.
8. In the Description field, enter average salaries exercise.
9. In the Portal DB Provider list, MYCOMPANY_DB_PROVIDER should already be selected.
10. Click Next.
11. In the SQL Query field, enter the following code:
    
    ```sql
    select
        null     the_link,
        job      the_name,
        avg(sal) the_data
    from emp
    group by job
    ```

    This SQL query will work only if you have SELECT privileges on the EMP table in the SCOTT schema.
12. Click Next twice, or until you see the Display Options step of the wizard.
13. In both the Full Page Options and Portlet Options sections, select the values shown in Table A–43.
14. Click Finish.
15. Click Run as Portlet to see what your chart looks like.
16. Close the browser window where the chart is displayed, and click Close to close the component manager and return to the Portal Navigator.

A.10.3 Exercise: Building the Team Bonuses Report

The Team Bonuses report, shown in Figure A–81, displays the average bonus paid to employees in each department.
Figure A–81 Team Bonuses Report

To build the Team Bonuses report, perform the following steps:

1. In the Navigator, click the Providers tab to bring it forward.
2. At the root level of the Providers tab, click the Locally Built Providers link.
3. Click the link for the myCompany_DB_Provider provider.
4. Click the Report link next to Create New…
5. Click the Reports From SQL Query link.
6. In the Name field, enter <YourName>_team_bonuses.
7. In the Display Name field, enter Team Bonuses.
8. In the Description field, enter team bonuses example.
9. In the Portal DB Provider list, MYCOMPANY_DB_PROVIDER should already be selected.
10. Click Next.
11. In the SQL Query field, enter the following code:

   ```sql
   select dept.deptno, dept.dname||','||dept.loc, sum(emp.comm)
   from dept, emp
   where dept.deptno = emp.deptno
   group by dept.deptno, dept.dname||','||dept.loc
   ```

   This query summarizes data from selected columns contained in the DEPT and EMP tables. Although we used a SQL query to build the report, we just as easily could have built it using the Reports Query Wizard.

   This SQL query will work only if you have SELECT privileges on the DEPT and EMP tables in the SCOTT schema.
12. Click Next.
13. Select Tabular.
14. Click Next.
15. Next to each column, enter the Column Heading Text as indicated in Table A–44.

<table>
<thead>
<tr>
<th>Column Column Heading Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPTNO Department No</td>
</tr>
<tr>
<td>DEPT.DNAME</td>
</tr>
<tr>
<td>SUM(EMP.COMM) Bonus Paid</td>
</tr>
</tbody>
</table>

   This will add descriptive labels above the columns that appear in your report.

16. Click Next twice, or until you see the Display Options step of the wizard.
17. For both the Full Page Options and Portlet Options sections, select the values shown in Table A–45.

### Table A–45 Team Bonuses Report Full Page and Portlet Display Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heading Font Face</td>
<td>Arial</td>
</tr>
<tr>
<td>Heading Font Color</td>
<td>White</td>
</tr>
<tr>
<td>Heading Size</td>
<td>10pt</td>
</tr>
<tr>
<td>Row Text Font Face</td>
<td>Arial</td>
</tr>
<tr>
<td>Row Text Font Color</td>
<td>Black</td>
</tr>
<tr>
<td>Row Text Size</td>
<td>10pt</td>
</tr>
<tr>
<td>Heading Background Color</td>
<td>Slate Gray</td>
</tr>
<tr>
<td>Border</td>
<td>Thin Border</td>
</tr>
</tbody>
</table>

18. Click **Finish**.

19. Click **Run as Portlet** to see what your report looks like.

20. Close the browser window where the report is displayed, and click **Close** to close the component manager and return to the Portal Navigator.

### A.10.4 Exercise: Building the Job Profiles Report

The Job Profiles report, shown in Figure A–82, displays a count of the employees having a specific work profile in each department. In this example, data is grouped by the department number column, and in the resulting report, the department number is displayed only once for every distinct occurrence.

#### Figure A–82 Job Profiles Report

<table>
<thead>
<tr>
<th>Department No</th>
<th>Employee Name</th>
<th>Hire Date</th>
<th>Bonus Paid</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Paul</td>
<td>02-FEB-06</td>
<td>2500</td>
</tr>
<tr>
<td></td>
<td>Peter</td>
<td>26-FEB-06</td>
<td>1950</td>
</tr>
<tr>
<td>20</td>
<td>Stewart</td>
<td>07-APR-06</td>
<td>2210</td>
</tr>
<tr>
<td>30</td>
<td>Mark</td>
<td>15-MAR-06</td>
<td>3000</td>
</tr>
<tr>
<td></td>
<td>Blake</td>
<td>07-APR-06</td>
<td>2000</td>
</tr>
<tr>
<td></td>
<td>James</td>
<td>30-MAY-06</td>
<td>2210</td>
</tr>
</tbody>
</table>

To build the Job Profiles report, perform the following steps:

1. In the Navigator, click the **Providers** tab to bring it forward.
2. At the root level of the Providers tab, click the **Locally Built Providers** link.
3. Click the link for the **myCompany_DB_Provider** provider.
4. Click the **Report** link next to **Create New**....
5. Click the **Reports From SQL Query** link.
6. In the **Name** field, enter `<YourName>_job_profiles`.
7. In the **Display Name** field, enter **Job Profiles**.
8. In the Description field, enter job profiles example.

9. In the Portal DB Provider list, MYCOMPANY_DB_PROVIDER should already be selected.

10. Click Next.

11. In the SQL Query field, enter the following code:

```sql
select deptno, job, count(empno)
from scott.emp
group by deptno, job
```

This query provides data from selected columns contained in the EMP table. This SQL query will work only if you have SELECT privileges on the EMP table in the SCOTT schema.

12. Click Next.

13. Select Tabular.

14. Click Next.

15. Next to each column, enter the Column Heading Text as indicated in Table A–46.

16. Click Next twice, or until you see the Display Options step of the wizard.

17. For both the Full Page Options and Portlet Options sections, select the values shown in Table A–47.

18. To display the department number only once for every distinct occurrence, in the Display Options step, under Break Options, and set the First Break Column to DEPTNO.

This will group data by the department number and display its value once for every distinct occurrence.

---

**Table A–46** New Hires Bonuses Report Column Heading Text

<table>
<thead>
<tr>
<th>Column</th>
<th>Column Heading Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPTNO</td>
<td>Department No</td>
</tr>
<tr>
<td>JOB</td>
<td>Job Profile</td>
</tr>
<tr>
<td>COUNT(EMPNO)</td>
<td>No. of Employees</td>
</tr>
</tbody>
</table>

This will add descriptive labels above the columns that appear in your report.

**Table A–47** New Hires Bonuses Report Full Page and Portlet Display Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heading Font Face</td>
<td>Arial</td>
</tr>
<tr>
<td>Heading Font Color</td>
<td>White</td>
</tr>
<tr>
<td>Heading Size</td>
<td>10pt</td>
</tr>
<tr>
<td>Row Text Font Face</td>
<td>Arial</td>
</tr>
<tr>
<td>Row Text Font Color</td>
<td>Black</td>
</tr>
<tr>
<td>Row Text Size</td>
<td>10pt</td>
</tr>
<tr>
<td>Heading Background Color</td>
<td>Slate Gray</td>
</tr>
<tr>
<td>Border</td>
<td>Thin Border</td>
</tr>
</tbody>
</table>
Notes:

- The Row Order Options must have the same order as Break Options. For example, if you have set First Break Column to DEPTNO, and Second Break Column to JOB, then, under Row Order Options, Order by 1. must be DEPTNO and 2. must be JOB.

- You can set break options for a maximum of three columns. If you specify a column in the Second Break Column field, then the data under that column will be grouped within the groups formed by the first break column.

19. Click Finish.

20. Click Run as Portlet to see what your report looks like.

21. Close the browser window where the report is displayed, and click Close to close the component manager and return to the Portal Navigator.
This appendix describes common problems that you might encounter when using OracleAS Portal and explains how to solve them. It contains the following topics:

- Diagnosing General Portlet Problems
- Diagnosing Java Portlet Problems
- Diagnosing OmniPortlet Problems
- Diagnosing Web Clipping Problems
- Need More Help?

Note: Throughout this chapter, you will see references to ORACLE_HOME. ORACLE_HOME represents the full path of the Oracle home, and is used in cases where it is easy to determine which Oracle home is referenced. The following conventions are used in procedures where it is necessary to distinguish between the middle tier, OracleAS Infrastructure, or Oracle Application Server Metadata Repository Oracle home:

- MID_TIER_ORACLE_HOME, represents the full path of the middle-tier Oracle home.
- INFRA_ORACLE_HOME, represents the full path of the Oracle Application Server Infrastructure Oracle home.
- METADATA_REP_ORACLE_HOME, represents the full path of the OracleAS Infrastructure home containing the Oracle Application Server Metadata Repository.

B.1 Diagnosing General Portlet Problems

This section describes common problems and solutions for all portlets. It contains the following topics:

- Portlet Refresh Failure
- HTML Tags Appearing in Portlet

B.1.1 Portlet Refresh Failure

When using the JavaScript `document.write()` method in conjunction with a Mozilla browser, you may find that portlet refresh does not work.
Problem
When using a Mozilla browser, such as Firefox, the `document.write()` method causes portlet refresh to no longer work as expected.

Solution
Avoid using `document.write()` in your output if you also want portlet refresh to work on Firefox. If this is not possible, you can choose to have the portlet not display the Refresh and Restore icons during rendering. You can also turn off portlet refresh and restore at the region level.

B.1.2 HTML Tags Appearing in Portlet
After upgrading from an earlier release to Release 10.1.4, you may find unformatted HTML appearing in your portlets, for example, in the portlet title.

Problem
When viewing a portlet, you see raw HTML. For example, you might see the following in a portlet title:

```
<b>Oracle Application Server 10<i>g</i> Collateral</b>
```

when what you expected to see was:

**Oracle Application Server 10g Collateral**

To protect against cross site scripting (XSS) attacks, OracleAS Portal Release 10.1.4 escapes all user input by default. Thus, some HTML that was formatted in earlier releases now appears unformatted because it has been escaped to plain text.

Solution
You can resolve this issue in one of two ways as follows:

- Tell your users to stop entering HTML tags as part of their input and have them enter only plain text.
- Turn on compatibility mode to make your OracleAS Portal instance behave the way it did in releases prior to Release 10.1.4, that is, stop the default escaping of HTML to plain text. Note that turning on compatibility mode makes your portal instance less secure.

For more information on these options, refer to Oracle Application Server Portal Installation and Upgrade Guide.

B.2 Diagnosing Java Portlet Problems
This section describes common problems and solutions for Java portlets. It contains the following topics:

- Portlet Logging
- Installation and Deployment Problems
- Java Portlet Wizard Not Available
- Portlet Code Does Not Compile
- Application Server Connection Test Fails
- Provider Test Page Shows Error
B.2.1 Portlet Logging

In addition to specific issues listed in the other sections of this appendix, the following general techniques can help you to troubleshoot portlet problems.

Problem 1
When accessing a portal page, the portlet does not display or displays an error message.

Solution 1
Check the application log file look for errors.
For OracleAS Portal:
MID_TIER_ORACLE_HOME/j2ee/OC4J_Portal/application-deployments/portal/OC4J_Portal_default_island_1/application.log
For a custom Web provider:
MID_TIER_ORACLE_HOME/j2ee/OC4J_instance_name/application-deployments/web_application_name/OC4J_Portal_default_island_1/application.log

Problem 2
Pertinent information that may help solve the problem is not being recorded in the log file.

Solution 2
Increase the provider's log level to produce more detailed logging information by adding the following entry to the web.xml file:

```xml
<env-entry>
  <env-entry-name>oracle/portal/provider/global/log/logLevel</env-entry-name>
  <env-entry-value>6</env-entry-value>
  <env-entry-type>java.lang.Integer</env-entry-type>
</env-entry>
```
You may also change this value for runtime by editing orion-web.xml or updating it in the Grid Control Console. You may have to restart OC4J to ensure that the changes made to the configuration file take effect.

B.2.2 Installation and Deployment Problems

This section describes problems that you may encounter when installing and deploying the PDK-Java framework and samples.

B.2.2.1 Cannot Find a Java Class Object
You receive an error message about a Java class that does not exist. For example:

An unexpected error occurred-29540 : class oracle/webdb/provider/web/HttpProviderDispatcher does not exist (WWC-43000)
Problem
The referenced Java class object is not in the database or it is invalid. OracleAS Portal, not PDK-Java, generates the error message when it cannot execute the class.

Solution
Log into the database as your main OracleAS Portal schema. Execute a SELECT statement to verify that the object is in the schema. For example:

```
SELECT object_name, object_type, status FROM user_objects
  WHERE object_name like '%HttpProvider%'
```

Note that the name between quotes is case sensitive.

You should receive the following back:

```
/3334f18_HttpProviderDispatcher
```

Check for invalid or missing JAVA CLASS objects. Something could be wrong with the OracleAS Portal installation. If so, then recompiling invalid objects might help resolve it.

B.2.2.2 Cannot Deploy the template.ear File
You receive the following error message when deploying your EAR file based on template.ear:

```
Invalid J2EE application file specified - Base Exception:
Cannot get xml document by parsing WEB-INF/web.xml in template.war: <Line 88, Column 14> : '--' is not allowed in comments.
```

Resolution:

Problem
The `web.xml` file contains a syntax error.

Solution
Verify that the syntax of `web.xml` is correct. Some versions of `web.xml` shipped in `template.war` have a misplaced comment tag. Make sure that all opening comment tags, `<!--`, have a matching closing tag, `-->`, before starting a new comment tag.

B.2.2.3 Error When Attempting to Register Provider
A number of errors may occur when you attempt to register your provider. This section explains what can go wrong and how you can correct it. If your scenario is not covered in this section, then check the `application.log` of your provider or, if that does not help, use `logcfg.sql` for Portal Repository logging. For more information `logcfg.sql` and `application.log`, refer to the Oracle Application Server Portal Configuration Guide.

Problem 1
You receive the following error message:

```
Internal error (WWC-00006)
The provider URL specified may be wrong or the provider is not running. (WWC-43176)
```

When OracleAS Portal attempts to register your provider, it must contact the listener that serves your Web portlets. If you have already accessed the provider’s test page,
Troubleshooting Portlets and Providers

http://host.domain:port/context/providers/servicename, according to the installation instructions, OracleAS Portal may not recognize your machine.

**Solution 1**

Make sure the machine where OracleAS Portal resides can access your Web provider listener. The easiest way to test this setup is to start a browser on the host of the OracleAS Portal repository database and try to access the provider’s test page. If the browser needs a proxy setting to reach the provider, then you should set the same proxy for OracleAS Portal on the Administer tab. You should also use this proxy when registering the provider.

If your Web provider server is not part of the DNS, then add an entry in the OracleAS Portal server’s host file.

If a firewall exists between the machine with the OracleAS Portal database and the machine with the provider, then make sure that the necessary port is open.

**Problem 2**

You receive the following error:

```
Internal error (WWC-00006)
The provider URL specified may be wrong or the provider is not running. (WWC-43176)
The following error occurred during the call to Web provider: Can't read deployment properties for service: _default (WWC-43147)
```

When you register a provider, you have the option of specifying a Service Id. If you do not specify anything in the Service Id field, the registration URL will not contain the service name part, `http://host.domain:port/context/providers/`. Therefore, OracleAS Portal is looking for a default service, which is stored in a deployment file called `_default.properties`. When this file is not found, you receive this error.

**Solution 2**

Make sure that you bundle a file called `_default.properties` into your WAR file with the following path:

```
/Web-inf/deployment/
```

The `_default.properties` file should define the default service to use when none is specified for the provider. The file should look similar to the following:

```
serviceClass=oracle.webdb.provider.v2.adapter.soapV1.ProviderAdapter
loaderClass=oracle.portal.provider.v2.http.DefaultProviderLoader
showTestPage=true
definition=providers/test/provider.xml
autoReload=true
testPageURI=/htdocs/testpage/TestPage.jsp
```

**Problem 3**

You receive the following error message:

```
Internal error (WWC-00006)
The provider URL specified may be wrong or the provider is not running. (WWC-43176)
The following error occurred during the call to Web provider: Can't read deployment properties for service: test (WWC-43147)
```
OracleAS Portal cannot find the deployment properties file for the service you specified in the Service Id field or the service name part of the registration URL, http://host:domain:port/context/providers/servicename. For example, suppose you received this error in response to your attempt to register your provider with the following URL:

http://host:domain:port/context/providers/test

In this case, you probably do not have a file called test.properties in your WAR file in /Web-inf/deployment/.

**Solution 3**
Make sure that you bundle a deployment properties file (for example, test.properties) into your WAR file with the following path:

/Web-inf/deployment/

The file should look similar to the following:

```properties
serviceClass=oracle.webdb.provider.v2.adapter.soapV1.ProviderAdapter
loaderClass=oracle.portal.provider.v2.http.DefaultProviderLoader
showTestPage=true
definition=providers/test/provider.xml
autoReload=true
testPageURL=/htdocs/testpage/TestPage.jsp
```

**Problem 4**
You receive the following error message:

An error occurred when attempting to call the providers register function.  
(WWC-43134)
The provider URL specified may be wrong or the provider is not running.  
(WWC-43176)
The following error occurred during the call to Web provider: Class oracle.portal.provider.v2.render.RenderManager has no set or add method for tag "createdOn"  
(WWC-43147)

**Solution 4**
If you are using DefaultProvider, provider.xml includes the element <createdOn>. When the XML is parsed, the initialization process looks for a method called setCreatedOn in the class representing the containing element. For example:

```xml
<renderer class="my.local.Renderer">
<createdOn>12-Mar-2001</createdOn>
...
</renderer>
```

In this example, the initialization process looks for my.local.Renderer.setCreatedOn(). In your case, the appropriate method does not exist, causing this error.

**Problem 5**
You receive an error like the following:

(WWC-00006)
An unexpected error occurred: User-Defined Exception  
(WWC-43000)
wwpro_api_provider_registry.register_provider
An unexpected error occurred: ava.sql.SQLException: Inserted value too large for
This error indicates that your portlet has exceeded the data limit for a portlet. Each portlet is limited to 4K of data. The lengths of all of the following contribute toward this data limit:

- provider name, display name, and description
- parameter name, display name, and description
- event name, display name, and description

**Solution 5**
You can take two actions as follows to reduce the amount of data used for the portlet:

- Reduce the length of the portlet’s parameter and event names, display names, and descriptions.
- Reduce the number of parameters and events in your portlet

**Problem 6**
You receive an error similar to the following when trying to register your WSRP provider:

An error occurred when attempting to call the providers register function.
(WWC-43134)
An error occurred during the call to the WSRP Provider: Java stack trace from root exception:
java.rmi.ServerException: Internal Server Error (caught exception while handling request: oracle.webdb.wsrp.server.ContainerRuntimeException: An internal error has occurred in method <init>)
at com.sun.xml.rpc.client.StreamingSender._raiseFault(StreamingSender.java:384)
at com.sun.xml.rpc.client.StreamingSender._send(StreamingSender.java:245)
at oracle.webdb.wsrp.WSRP_v1_Registration_PortType_Stub.register(WSRP_v1Registration_PortType_Stub.java:183)
at oracle.webdb.wsrp.client.design.v1.OraWSRP_v1_Registration_PortType.register(Unknown Source). (WWC-43273)

When you configure OC4J standalone for WSRP, you must edit `data-sources.xml`. Typically, this message indicates a problem in your `data-sources.xml`.

**Solution 6**
Review the entry you added to `data-sources.xml` as described in Section 6.3.2, "Configuring OC4J Standalone". Ensure that the syntax is correct and that line breaks are occurring in the proper places. Also confirm that you can connect to the database with the specified user name, password, port, host, and service name.

**B.2.2.4 Error Adding a Portlet to a Provider**
You added a portlet to your provider, but, when you view the provider in OracleAS Portal, you do not see the new portlet.

**Problem**
The most common problem is a syntactical error in `provider.xml`, but it could also be that the updated `provider.xml` file is not being found for some reason.

**Solution**
Try the following:
Diagnosing Java Portlet Problems

- Examine provider.xml. Ensure that you have two syntactically correct opening and closing portlet tags within the provider tag. For example:

  ```
  <provider class="oracle.portal.provider.v2.DefaultProviderDefinition" session="true">
    <portlet class="oracle.portal.provider.v2.DefaultPortletDefinition" version="1">
      <id>1</id>
      ...
    </portlet>
    <portlet class="oracle.portal.provider.v2.DefaultPortletDefinition" version="1">
      <id>2</id>
      ...
    </portlet>
  </provider>
  ```

- Check the test page for errors. If no errors are found, then refresh the provider.
- Check the application.log of the provider for errors.

### B.2.2.5 Portlet Does Not Exist

You received the following error message:

Error: The listener returned the following Message: 500 Portlet with id $1 doesn't exist

**Problem**

You removed a portlet from provider.xml without first removing the portlet from the page. The portlet no longer appears on the Customization page and therefore you cannot remove it from the page.

**Solution**

Delete the region that contains the portlet. Remember, when deleting a portlet from provider.xml, delete the portlet from the page first.

### B.2.2.6 File Not Found

You receive the following error message:

Request URL:/jpdk/snoop/snoopcustom.jsp Exception: javax.servlet.ServletException: java.io.FileNotFoundException: C:\IAS\Apache\Apache\htdocs\jpdk\snoop\snoopcustom.jsp (The system cannot find the path specified)

**Problem**

You added a JSP portlet to your page. OracleAS Portal cannot locate the JSP in the file system using provider.xml.

**Solution**

Confirm the file name and location and verify the information within provider.xml.

### B.2.2.7 XML Parser Error

The XML Parser may fail at times when parsing your provider.xml. This section explains what can go wrong and how you can correct it.
**Problem 1**
You receive the following error message:

Error: The XML parser encountered an Error, and could not complete the conversion for portlet id=150, it returned the following message: XML Parsing Error

You attempted to register your provider or display portlets on a page, and received this error message. When altering `provider.xml`, one or more of the tags were corrupt and the file cannot be parsed.

**Solution 1**
Review `provider.xml` for errors.

**Problem 2**
You receive the following error message:

Error: The XML parser encountered an Error, and could not complete the conversion for portlet id=146, it returned the following message: XSL reference value missing in XML document.

You attempt to add your portlet to a page or register your provider, and you receive this error. The portlet information within `provider.xml` may be incorrect. Another possibility is that your classes cannot be located.

**Solution 2**
Check that no tags are missing and that they appear in the correct sequence. Verify that your class files exist and are specified correctly within `provider.xml`.

**B.2.2.8 Error Adding Portlets**
You receive the following error message when trying to add portlets to a page:

Error: An unexpected error occurred: ORA-29532: Java call terminated by uncaught Java exception: Attribute value should start with quote. (WWC-43000) An unexpected error occurred: Attribute value should start with quote. at oracle.xml.parser.v2.XMLError.flushErrors(XMLError.java:205)

**Problem**
OracleAS Portal has an issue with `provider.xml`. Most likely, an attribute within the file is corrupt.

**Solution**
Review `provider.xml` for errors.

**B.2.2.9 Content Request Timed Out**
You receive the following error message:

ERROR TIMEOUT FOR CONTENT=timeout

**Problem**
You successfully registered your Web provider. When adding portlets to a page or refreshing a page with existing portlets, you receive this error message. This error indicates that OracleAS Portal timed out trying to contact your Java portlet listener.

**Solution**
Verify that you can still access the sample page:
You set the timeout period by adding the following init parameters to the page servlet of OracleAS Portal's web.xml at MID_TIER_ORACLE_HOME/j2ee/OC4J_Portal/applications/portal/portal/WEB-INF/web.xml. Note that this web.xml file is for the Parallel Page Engine. For example:

```xml
<init-param>
  <param-name>requesttime</param-name>
  <param-value>1000</param-value>
</init-param>
<init-param>
  <param-name>minTimeout</param-name>
  <param-value>100</param-value>
</init-param>
<init-param>
  <param-name>stall</param-name>
  <param-value>500</param-value>
</init-param>
```

For more information about web.xml and its configuration parameters, refer to Oracle Application Server Portal Configuration Guide.

You may also set a timeout value for individual providers when you register them with OracleAS Portal. Note that the provider timeout value must match or exceed the minTimeout parameter value in web.xml in order to have any effect. For information about registering PDK-Java providers, refer to Section 6.6.2.5, "Registering and Viewing Your Portlet". For information about registering WSRP providers, refer to Adding Your Portlet.

**B.2.2.10 Message 500 Returned**

You receive the following error message:

`Error: The listener returned the following Message: 500`

**Problem**

This generic message displays when OracleAS Portal receives an Internal Server Error while attempting to display your portlet. This problem might have any one of several causes.

**Solution**

Review the application.log file to discover the cause of the error.

For OracleAS Portal:

```
MID_TIER_ORACLE_HOME/j2ee/OC4J_Portal/application-deployments/portal/OC4J_Portal_default_island_1/application.log
```

For a custom Web provider:

```
MID_TIER_ORACLE_HOME/j2ee/OC4J_instance_name/application-deployments/web_application_name/OC4J_Portal_default_island_1/application.log
```

**B.2.2.11 JPS Portlets with the get Method not Working**

HTML forms can be submitted using either the get or post method, but the WSRP standard only requires the consumer (portal) to use the post method. Support of the get method is optional according to the standard. Since portal consumers are not
required to support the `get` method, we highly recommended that you use the `post` method when developing your portlets.

**Problem**
You have JPS portlets that use the `get` method and they are not working correctly in OracleAS Portal. Forms with `get` lose the query string in the action URL when the form is submitted. The query string is needed for the portlets to return back to the portal and for the WSRP state.

**Solution**
Replace the `get` method with the `post` method.

**B.2.2.12 Portlet Displays Session Expired Message After Redeployment**
When you redeploy your portlets to the portlet container, all existing sessions between the producer and all of its consumers are lost. If a consumer tries to reuse an existing producer session, it may receive an error message the first time it tries to contact the producer after redeployment.

**Problem**
You receive an error message in your portlet the first time you access it after redeployment.

Error: Could not get markup. The cookie or session is invalid or there is a runtime exception.

**Solution**
To reestablish the producer's session, refresh the portal page. You won't see this error message if you are re-accessing the portlet from a new browser session because it automatically establishes a new producer session.

**B.2.3 Java Portlet Wizard Not Available**
In Oracle JDeveloper, when you try to open the Java Portlet Wizard by choosing **File > New > Web tier**, the Portlets menu selection is not present.

**Problem**
The Java Portlet Wizard is not installed.

**Solution**
Install the Portal Extension to Oracle JDeveloper, as described in Section 6.4.1, "Installing the Portal Extension for Oracle JDeveloper".

**B.2.4 Portlet Code Does Not Compile**
When you try to compile the code for your portlet, you receive a compilation error.

**Problem**
The Portlet Development library is not selected.

**Solution**
Make sure that Portlet Development library is selected for the project. Edit your project's properties and select the Profiles > Development > Libraries entry in the...
pane on the right. Make sure that the Portlet Development library is listed under Selected Libraries.

B.2.5 Application Server Connection Test Fails

In Oracle JDeveloper, the Application Server Connection Test fails with the message Connection refused: connect.

**Problem 1**
OC4J is not running.

**Solution 1**
Make sure OC4J is up and running by typing the following URL in your browser:
http://yourhostyourdomain:8888

**Problem 2**
The connection information is incorrect.

**Solution 2**
Verify the connection information that you provided in the Connection Setup wizard.

B.2.6 Provider Test Page Shows Error

When accessing the provider test page with your browser, an error is shown.

**Problem 1**
The provider.xml syntax is incorrect.

**Solution 1**
Correct the provider.xml syntax. Refer to the PDK-Java XML Provider Definition Tag Reference document on Portal Studio:
http://portalstudio.oracle.com/pls/ops/docs/folder/community/pdk/jpdk/v2/xml_tag_reference_v2.html

**Problem 2**
Needed JAR files are missing from the deployment environment.

**Solution 2**
Search the JAR files for the missing class. When the missing JAR file is identified, you can add to the lib directory of the application from the Oracle JDeveloper Deployment Profile. Refer to the Oracle JDeveloper Online Help System for more information about the deployment profile.

B.2.7 Web Provider Not Appearing in Portlet Repository

Users cannot see a Web provider, such as Omniportlet, in the Portlet Repository even though it has been registered successfully and is visible in the navigator.

**Problem**
The Web provider uses DBPreferenceStore and the database information stored in data-sources.xml is incorrect.
Solution
Correct the information in data-sources.xml. Refresh the provider and invalidate the Portlet Repository cache.

B.2.8 Portlet Does Not Display on Page

When you access a portal page, the portlet does not display on the page.

Problem 1
The provider is not running.

Solution 1
Make sure that the provider is up and running by entering the provider registration URL in your browser’s address bar.

Problem 2
The security manager for the provider is preventing the portlet from displaying.

Solution 2
In the provider definition file, provider.xml, delete or comment out the security manager, if any.

Problem 3
The user may not have the Execute privilege for the portlet as defined on the portlet’s Access page.

Solution 3
Check the privileges for the portlet and, if the user or a group the user belongs to does not have the Execute privilege, grant them access. Portlets sometimes inherit their access privileges from the provider. You can use the Navigator in OracleAS Portal to find the portlet’s provider and check its access privileges, then drill down to the portlet to see its access privileges.

Problem 4
You get an error trying to display the portlet after you redeployed it.

Solution 4
Refer to Section B.2.2.12, "Portlet Displays Session Expired Message After Redeployment".

B.2.9 After Initial Successful Display, Portlet Does Not Display on Page

When you access a portal page, the portlet initially displays on the page but returns error messages in subsequent display attempts.

Problem
The provider session information is incorrect.

Solution
Check whether or not the provider uses sessions. If it does, edit the provider registration information to make sure that you registered it accordingly:
Login Frequency: Once per User Sessions

You should also confirm that <session> is set to true in provider.xml.

B.2.10 Provider Group Not Created

When you create a new provider group, the provider group is not created.

Insufficient Privilege. Please contact the administrator for privilege in Default Provider Builder Instance.

You might also receive some mod_osso errors.

Problem 1
If you get mod_osso errors during authentication, the most likely problem is that mod_osso is not correctly configured on the middle tier or you have a mod_osso registration problem.

Solution 1
For information on reregistering mod_osso, refer to the Oracle Application Server Single Sign-On Administrator’s Guide.

Problem 2
You have a problem in the jpdk application classpath.

Solution 2
Check the jpdk application log for errors.

Problem 3
The user does not have the necessary privileges to access these pages.

Solution 3
Modify provideruiacls.xml to grant the user the necessary privileges. provideruiacls.xml has two instances, one for PDK-Java providers and one for OracleAS Portal tools (such as Omniportlet) providers:

MID_TIER_ORACLE_HOME/j2ee/OC4J_Portal/applications/jpdk/jpdk/WEB-INF/deployment_providerui

HOME/j2ee/OC4J_Portal/applications/portalTools/providerBuilder/WEB-INF/deployment_providerui

The following code sample illustrates how you can add users in provideruiacls.xml:

<object name="ANY_PROVIDER" owner="providerui">
    <user name="orcladmin" privilege="500"/>
    <user name="portal" privilege="500"/>
</object>

B.2.11 URL-Based Portlet Does Not Work

Attempting to display your portlet through URL generates the following error message:

500 INTERNAL SERVER ERROR
Problem
The httpProxyHost is not defined correctly.

Solution
In general, you should replace your URL-based portlets (formerly URL Services) with Web Clipping. Refer to Chapter 5, "Creating Content-Based Portlets with Web Clipping" for more information about Web Clipping.

If for some reason you must continue to use a URL-based portlet, check the default proxy information in your provider.xml file. For example:

```xml
<proxyInfo class="oracle.portal.provider.v1.http.ProxyInformation">
  <httpProxyHost>www-proxy.us.oracle.com</httpProxyHost>
  <httpProxyPort>80</httpProxyPort>
</proxyInfo>
```

Change the httpProxyHost and httpProxyPort proxy settings or remove these settings if you do not use proxy. Restart the OC4J instance to test if this resolves the problem.

B.3 Diagnosing OmniPortlet Problems

This section provides information to help you troubleshoot problems you may encounter while using OmniPortlet.

To view errors that occur during the execution of OmniPortlet, do the following:

- Open the application log file:
  
  ```
  MID_TIER_ORACLE_HOME/j2ee/OC4J_Portal/application-deployments/portalTools/OC4J_Portal_default_island_1/application.log
  ```
  or, if you are using a standalone OC4J instance:
  
  ```
  OC4J_HOME/j2ee/home/application-deployments/portalTools/application.log
  ```

- Display the HTML source (for example, in Microsoft Internet Explorer browser, choose View > Source), and locate the errors embedded in the output HTML as comments.

To alter the logging level of the OmniPortlet Provider, do the following:

- Open the web.xml file and modify the context-param value of oracle.portal.log.LogLevel to the possible values ranging from 1 to 8 (where 8 means debug). The web.xml file is located at:
  
  ```
  MID_TIER_ORACLE_HOME/j2ee/OC4J_Portal/applications/portalTools/omniPortlet/WEB-INF/web.xml
  ```
  or, if you are using a standalone OC4J instance:
  
  ```
  OC4J_HOME/j2ee/home/applications/portalTools/omniPortlet/WEB-INF/web.xml
  ```

The OmniPortlet errors that you are most likely to encounter, and possible solutions, are as follows:

- **OmniPortlet Cannot Access the Specified URL**
- **Portlet Content Is Not Refreshed**
B.3.1 OmniPortlet Cannot Access the Specified URL

Your OmniPortlet displays errors or does not display the correct content.

**Problem 1**
The URL is not active.

**Solution 1**
Type the URL directly in your browser address field to test its validity.

**Problem 2**
If a proxy server is required to reach the site, the proxy settings are not valid. The following messages may display:

- Failed to open specified URL.
- Cannot open the URL specified because of connection timeout.

**Solution 2**
Check that your proxy settings are valid by clicking Edit for the HTTP Proxy Setting in the OmniPortlet Provider Test Page.

**Problem 3**
The message OmniPortlet timed out displays in your OmniPortlet.

**Solution 3**
1. If a proxy server is required to reach the site, see Solution 2.
2. If the URL request takes a long time to process (for example, it executes a long running query), try increasing the timeout value (in seconds) in the OmniPortlet provider.xml file in $MID_TIER_ORACLE_HOME/j2ee/OC4J_Portal/applications/portalTools/omniPortlet/WEB-INF/providers/omniPortlet. If you change the timeout value, you also need to do the following:
   - Bounce your middle-tier.
   - Increase the provider registration Timeout value of the OracleAS Portal instances with which this provider is registered.

**Problem 4**
If HTTP authentication is required and the user name and password are missing or not valid, the following error message displays:

Authorization failed when connecting to the URL specified.
Provide correct user name and password to connect.

**Solution 4**
To configure the Secured Data (Web Clipping) Repository Setting, click Edit on the OmniPortlet Provider Test Page. On the Source tab, click Edit Connection and enter a valid user name and password on the Connection Information page. To enter the connection information, the Secured Data Repository must be already be configured. To configure its settings, click Edit on the OmniPortlet Provider Test Page.
Problem 5
If opening a URL to an HTTPS site with a certificate and the certificate is identified as not valid, the following error message displays in the portlet:

SSL handshake failed for HTTPS connection to the specified URL. The certificate file needs to be augmented.

Solution 5
See the Oracle Application Server Portal Configuration Guide.

Problem 6
If the proxy server requires authentication and the user name and password are missing or not valid, the following error message displays:

Invalid or missing user proxy login information.

Solution 6
Check that your proxy server user name and password are valid. Refer to Section B.4.5, "HTTP Error Code 407 When Clipping Outside Firewall" for more information.

B.3.2 Portlet Content Is Not Refreshed

After changing portlet properties in the Edit Defaults page, your portlet content is not refreshed.

Problem 1
OracleAS Web Cache invalidation is not configured properly.

Solution 1
See Section I.2.1.3 "Configuring Caching (PDK Only)" in the Oracle Application Server Portal Configuration Guide.

Problem 2
You have personalized the portlet.

Solution 2
See Section B.3.3, "Edit Defaults Changes are Not Reflected in the Personalized Portlet".

B.3.3 Edit Defaults Changes are Not Reflected in the Personalized Portlet

When you personalize the portlet at runtime using the Personalize link, the new property values are not reflected in the personalized version of the portlet.

Problem
When you personalize the portlet, a complete copy of the personalization object file is created. Since all properties are duplicated, subsequent modifications through Edit Defaults are not reflected in the personalized version.

Solution
To ensure the latest changes are made to the portlet, click Personalize again (after the modifications using the Edit Defaults option), then select the Reset to Defaults option.
B.4 Diagnosing Web Clipping Problems

This section provides information to help you troubleshoot problems you may encounter while using the Web Clipping provider or Web Clipping Studio.

This section includes the following topics:

- Setting Logging Levels
- Reviewing Error Messages
- Checking the Status of the Provider with the Test Page
- Problem Connecting to the Web Site for Clipping
- HTTP Error Code 407 When Clipping Outside Firewall
- Cannot Clip a Page
- Images Not Retrieved with Clipping
- Resolving Problems with Migration of URL-based Portlets

B.4.1 Setting Logging Levels

By default, the logging level of Web Clipping is set to level 3, which provides information about configuration, severe errors, and warnings. This level is reasonable for day-to-day operations. To view information useful for debugging, you should set the logging level to 8.

To set the logging level, edit the `web.xml` file, which is located at:

OracleHome/j2ee/OC4J_Portal/applications/portalTools/webClipping/WEB-INF

To set the level to 8 and display debugging information, set the value of the parameter `oracle.portal.log.LogLevel` as follows:

```
<context-param>
  <param-name>oracle.portal.log.LogLevel</param-name>
  <param-value>8</param-value>
</context-param>
```

After you make the change, restart the Web application.

B.4.2 Reviewing Error Messages

Errors that occur when accessing the Test Page or executing of the Web Clipping portlet are written to one of the following files:

OC4J_HOME/j2ee/home/application-deployments/portalTools/application.log
IAS_HOME/j2ee/OC4J_instance/application-deployments/portalTools/OC4J_instance_default_island_1/application.log

B.4.3 Checking the Status of the Provider with the Test Page

You can use the Web Clipping Provider Test Page to determine if the provider is functioning properly. To access the Test Page, click Web Clipping Provider from the Portal Tools Application Welcome Page, which is located at:

http://host:port/portalTools

The Provider Test Page: Web Clipping is displayed. It provides the following information:
Portlet information about the Web Clipping portlet. The Web Clipping provider contains only one portlet.

Provider initialization parameters and values.

Provider status, with links to pages for editing the configuration.

For more information about using the Test Page, see the "Administering Web Clipping" appendix in the Oracle Application Server Portal Configuration Guide.

B.4.4 Problem Connecting to the Web Site for Clipping

You encounter difficulties making or maintaining connections to the Web site containing the clipping.

Problem 1
If you cannot access the Web site using Web Clipping Studio, you may be using an incorrect URL.

Solution 1
To be sure a URL is correct, test the URL that you want to clip in a browser before you attempt to clip it. Also test the URL to be sure that it is accessible from the provider middle tier.

Problem 2
The connection times out when attempting to browse to any Web site and your environment uses a proxy server to connect to HTTP servers outside a firewall.

Solution 2
Make sure that the proxy servers are configured correctly.

To configure the proxy servers, go to the Web Clipping Provider Test Page, as described in Section B.4.3, "Checking the Status of the Provider with the Test Page". In the Web Clipping Provider Test Page, click Edit in the Actions column of the HTTP Proxy row. In the Edit Provider page, specify the HTTP Proxy Host and the HTTP Proxy Port for the HTTP Proxy.

For access to servers inside the firewall, you can specify a list of domain names that need not go through the firewall by selecting No Proxy for Domains beginning with and entering the URL. You do not need to restart OC4J for the new settings to take effect.

For more information about configuring proxy servers, see the "Administering Web Clipping" appendix of the Oracle Application Server Portal Configuration Guide.

Problem 3
Your configuration includes a load balancer and you experience difficulty making or maintaining connections while attempting to add a clip to a Web Clipping portlet.

Solution 3
The configuration was not set up correctly. To rectify this error, you must ensure the following:

- If multiple OC4J instances are set up behind a load balancer, the Web Clipping Repository and HTTP proxy must be configured identically on all OC4J instances before you join them to the load balancer.
Web clippings have definitions that must be stored persistently in the Web Clipping Repository hosted by an Oracle Database server. In a multiple middle-tier environment, all instances of OC4J must store definitions in the same repository.

In addition, all instances of OC4J must have identical HTTP proxy configurations.

- The Load Balancer must be session-enabled. If it is not, the first request connects, but subsequent requests, which may be routed to a different instance, fail.

For more information about configuring with a load balancer, see the "Performing Advanced Configuration" chapter of the Oracle Application Server Portal Configuration Guide.

**Problem 4**
You use a reverse proxy, but cannot make a connection.

**Solution 4**
Make sure that the reverse proxy server is configured correctly. See the "Performing Advanced Configuration" chapter of the Oracle Application Server Portal Configuration Guide for information about configuring a reverse proxy.

**Problem 5**
You cannot connect and the error log contains a message about denying logon to the database. (See Section B.4.1, "Setting Logging Levels" for information about the error log.)

**Solution 5**
The PORTAL schema password for the infrastructure database may have been modified manually and it may not match the password stored in Oracle Internet Directory. Refer to the Oracle Internet Directory Administrator’s Guide for more information about setting the password.

**B.4.5 HTTP Error Code 407 When Clipping Outside Firewall**

The proxy servers are configured for proxy authentication and you receive HTTP error code 407 when you attempt to clip a page outside the firewall.

**Problem**
The proxy servers are configured for proxy authentication, but you have not configured proxy authentication.

**Solution**
You must manually configure proxy authentication. Web Clipping supports both global proxy and per-user authentication. You can specify the realm of the proxy server and whether all users login automatically with a provided user name and password, each user logs in with an individual user name and password, or all users log in with a specified user name and password. For more information about configuring proxy authentication, see the "Administering Web Clipping" appendix of the Oracle Application Server Portal Configuration Guide.

**B.4.6 Cannot Clip a Page**

You can connect to the Web site, but you cannot clip the page.
Problem
The page may be overpopulated with IFrames.

Solution
View the page in a browser and look at the page source. If it contains IFrames, start browsing in Web Clipping Studio using the URL pointed to by the IFrame src attribute.

B.4.7 Images Not Retrieved with Clipping
Images in a Web clipping are not retrieved with the rest of the clipping.

Problem
Because images are treated as links (using the src attribute of the IMG tag), images from clipped sites are served directly from the original sites. If the images require that you enable proxy settings during creation of the clipping, then disabling the browser proxy setting disables the viewing of the images in a clipping.

Solution
Enable proxy settings in the browser.

B.4.8 Resolving Problems with Migration of URL-based Portlets
You can migrate URL-based portlets that are stored in a provider.xml file to Web Clipping portlets. This section explains how to troubleshoot issues that arise as you migrate your portlets to Web Clipping portlets. For more information on the migration process itself, refer to Section 5.5, “Migrating from URL-Based Portlets”.

B.4.8.1 File Not Found Exception When Running Migration Tool
When you run the migration tool, you receive the following exception:

```
Exception in thread "main" java.io.FileNotFoundException: /wrongpath/somservice.properties (No such file or directory)
at java.io.FileInputStream.open(Native Method)
at java.io.FileInputStream.<init>(FileInputStream.java:103)
at oracle.webdb.wcs.client.urlservices.UrlServicesMigrator.importService(UrlServicesMigrator.java:631)
at oracle.webdb.wcs.client.urlservices.UrlServicesMigrator.main(UrlServicesMigrator.java:724)
at oracle.webdb.wcs.WcWebdbMain.main(WcWebdbMain.java:60)
```

Problem
The files you specified for the deployment properties file for either the URL-based portlets service or the Web Clipping Portlet service do not exist.

Solution
Verify the path to both service deployment properties files in the argument list to see if they point to the correct location.

B.4.8.2 Null Pointer Exception When Running Migration Tool
When you run the migration tool, you receive the following exception:

```
Exception in thread "main" java.lang.NullPointerException
at java.util.Hashtable.put(Hashtable.java:375)
```
at java.util.Properties.setProperty(Properties.java:97)
at oracle.webdb.wcs.client.urlservices.UrlServicesMigrator.setRepository(UrlServicesMigrator.java:415)
at oracle.webdb.wcs.client.urlservices.UrlServicesMigrator.main(UrlServicesMigrator.java:733)
at oracle.webdb.wcs.WcWebdbMain.main(WcWebdbMain.java:58)

Problem
The Repository settings are not configured for the Web Clipping Provider.

Solution
See Section 5.5.1, "Preparing for Migration" for information about steps you must take before running the migration tool.

B.4.8.3 Target provider.xml is Already Migrated Error
When you run the migration tool, you receive the following error:

Error: Target provider.xml is already migrated.

Problem
The provider.xml file pointed to by the URL-based portlets service deployment properties file has already been migrated. The provider definition class specified in the provider.xml file is already in WcProviderDefinition.

Solution
If you want to redo the migration, rename your service in both the deployment properties file and the entire directory name under WEB-INF/providers and then run the migration again.

B.4.8.4 Cannot Migrate provider.xml with Class Error
When you run the migration tool, you receive the following error:

Error: Can't migrate provider.xml with class <someclass>

Problem
The provider.xml file pointed to by the URL-based portlets service deployment properties file contains a provider class name that does not match the one used by URL-based portlets. Specifically, the Provider class name specified in the file is not oracle.portal.provider.v2.http.URLProviderDefinition. The migration tool is intended to migrate URL-based Portlet provider definitions, not any other kind of provider definitions. Even those classes which sub-class from URL-based portlets provider definition will not be correctly migrated.

Solution
You cannot migrate this service.

B.5 Need More Help?
You can find more solutions on Oracle MetaLink, http://metalink.oracle.com. If you do not find a solution for your problem, log a service request.
See Also: Oracle Application Server Release Notes, available on the Oracle Technology Network:
http://www.oracle.com/technology/documentation/ias.html
When developing portlet functionality that requires access to a user’s identity, you often need to map between attributes in OracleAS Portal and Oracle Internet Directory, and `userProfileItems` in WSRP. Note that for OracleAS Portal Release 2 (10.1.4), these attributes are static and cannot be configured. This appendix provides a table that shows the mapping between various attributes.

### C.1 Mapping `userProfileItems` to Attributes

Table C–1 provides a mapping of `userProfileItems` to Oracle Internet Directory and OracleAS Portal attributes.

**Note:** In cases where the Oracle Internet Directory attribute holds a complex string, like `homePostalAddress`, it is parsed into the needed component strings. The delimiter is assumed to be the dollar sign ($). For example, suppose you had the following value for `homePostalAddress` in Oracle Internet Directory:

```
123 Main St$$$Smallville$NY$10001
```

In the OracleAS Portal profile attributes, this string would be parsed as follows:

- 123 Main St goes into `HOME_ADDR1`.
- Smallville goes into `HOME_CITY`.
- NY goes into `HOME_STATE`.
- 10001 goes into `HOME_ZIP`.

<table>
<thead>
<tr>
<th><strong>Table C–1 userProfileItems Mapping</strong></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>OracleAS Portal Profile Attributes</strong></th>
<th><strong>Oracle Internet Directory Attributes</strong></th>
<th><strong>WSRP userProfileItems</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRST_NAME</td>
<td><code>givenName</code></td>
<td>name/given</td>
</tr>
<tr>
<td>LAST_NAME</td>
<td><code>sn</code></td>
<td>name/family</td>
</tr>
<tr>
<td>MIDDLE_NAME</td>
<td><code>middleName</code></td>
<td>name/middle</td>
</tr>
<tr>
<td>KNOWN_AS</td>
<td><code>displayName</code></td>
<td>name/nickname</td>
</tr>
<tr>
<td>DATE_OF_BIRTH</td>
<td><code>orclDateOfBirth</code></td>
<td>bDate</td>
</tr>
<tr>
<td>OracleAS Portal Profile Attributes</td>
<td>Oracle Internet Directory Attributes</td>
<td>WSRP userProfileItems</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>ORGANIZATION</td>
<td>o</td>
<td>employerInfo/employer</td>
</tr>
<tr>
<td>TITLE</td>
<td>title</td>
<td>employerInfo/jobtitle</td>
</tr>
<tr>
<td>HOME_ADDR1</td>
<td>homePostalAddress</td>
<td>homeInfo/postal/name</td>
</tr>
<tr>
<td>HOME_ADDR2</td>
<td></td>
<td>homeInfo/postal/street</td>
</tr>
<tr>
<td>HOME_ADDR3</td>
<td></td>
<td>homeInfo/postal/organization</td>
</tr>
<tr>
<td>HOME_CITY</td>
<td></td>
<td>homeInfo/postal/city</td>
</tr>
<tr>
<td>HOME_STATE</td>
<td></td>
<td>homeInfo/postal/stateprov</td>
</tr>
<tr>
<td>HOME_ZIP</td>
<td></td>
<td>homeInfo/postal/postalcode</td>
</tr>
<tr>
<td>HOME_COUNTRY</td>
<td></td>
<td>homeInfo/postal/country</td>
</tr>
<tr>
<td>HOME_PHONE</td>
<td>homePhone</td>
<td>homeInfo/telecom/telephone/number</td>
</tr>
<tr>
<td>OFFICE_ADDR1</td>
<td>street</td>
<td>businessInfo/postal/name</td>
</tr>
<tr>
<td>OFFICE_ADDR2</td>
<td></td>
<td>businessInfo/postal/street</td>
</tr>
<tr>
<td>OFFICE_ADDR3</td>
<td></td>
<td>businessInfo/postal/organization</td>
</tr>
<tr>
<td>OFFICE_CITY</td>
<td>1</td>
<td>businessInfo/postal/city</td>
</tr>
<tr>
<td>OFFICE_STATE</td>
<td>st</td>
<td>businessInfo/postal/stateprov</td>
</tr>
<tr>
<td>OFFICE_ZIP</td>
<td>postalCode</td>
<td>businessInfo/postal/postalcode</td>
</tr>
<tr>
<td>OFFICE_COUNTRY</td>
<td>c</td>
<td>businessInfo/postal/country</td>
</tr>
<tr>
<td>WORK_PHONE</td>
<td>telephoneNumber</td>
<td>businessInfo/telecom/telephone/number</td>
</tr>
<tr>
<td>FAX</td>
<td>facsimileTelephoneNumber</td>
<td>businessInfo/telecom/fax/number</td>
</tr>
<tr>
<td>MOBILE_PHONE</td>
<td>mobile</td>
<td>businessInfo/telecom/mobile/number</td>
</tr>
<tr>
<td>EMAIL</td>
<td>mail</td>
<td>businessInfo/online/email</td>
</tr>
</tbody>
</table>
This appendix explains how to manually package your PDK-Java provider implementation into a portable format suitable for deployment on the Oracle Application Server or another J2EE application server. It then explains how to deploy the resulting EAR file in an Oracle Application Server environment and subsequently register it with one or more OracleAS Portal instances.

**Note:** In general, we recommend that you package and deploy your providers using the tools available in Oracle JDeveloper. However, if you find that you must package and deploy your providers manually for some reason, we provide the information in this appendix for your reference.

This chapter includes the following sections:

- **Introduction**
- **Packaging and Deploying Your Providers**

**Note:** Throughout this chapter, you will see references to ORACLE_HOME. ORACLE_HOME represents the full path of the Oracle home, and is used in cases where it is easy to determine which Oracle home is referenced. The following conventions are used in procedures where it is necessary to distinguish between the middle tier, OracleAS Infrastructure, or Oracle Application Server Metadata Repository Oracle home:

- MID_TIER_ORACLE_HOME, represents the full path of the middle-tier Oracle home.
- INFRA_ORACLE_HOME, represents the full path of the Oracle Application Server Infrastructure Oracle home.
- METADATA_REP_ORACLE_HOME, represents the full path of the OracleAS Infrastructure home containing the Oracle Application Server Metadata Repository.

**D.1 Introduction**

Before preceding with packaging and deploying your provider, you must understand a couple of basic concepts about the following:
D.1.1 WAR and EAR files

WAR and EAR files are used to deploy applications on a J2EE application server, such as Oracle Application Server. The WAR and EAR files encapsulate all of the components necessary to run an application in a single file. These files make the deployment of an application very easy and consistent, reducing the possibility of errors when moving an application from development to test, and test to production. These files can be described as follows:

- **WAR files** represent a Web application and include all the components of that Web application, including Java libraries or classes, servlet definitions and parameter settings, JSP files, static HTML files, and any other required resources.

- **EAR files** represent an enterprise application.

D.1.2 Service Identifiers

PDK-Java enables you to deploy multiple providers under a single adapter servlet. The providers are identified by a service identifier. When you deploy a new provider, you must assign a service identifier to the provider and use that service identifier when creating your provider WAR file. The service name is used to look up a file called `service_id.properties`, which defines the characteristics of the provider, such as whether to display its test page.

For example, you can register the PDK-Java samples provider using the following URL and a service identifier of `urn:sample`:

http://mycompany.com/jpdk/providers

Alternatively, you can use a URL of the form:

http://mycompany.com/jpdk/providers/sample

where the provider name (sample) is appended to the URL of the PDK-Java samples provider. In this case, you should leave the Service Id field blank when registering the provider.

You can specify the service identifier separately in cases where multiple portals are sharing the same provider. By registering each portal with a different service identifier, you can specify the provider properties for each consumer independently.

Once your provider has been deployed, you must use the correct service identifier to register your provider with OracleAS Portal, which ensures that requests are routed to the correct provider. If the adapter servlet receives a request without a service identifier, the request goes to the default provider.

**Note:** If you do not know the service identifier, check the provider test page or contact the administrator of the provider. If you are using the Federated Portal Adapter, the URL points to the adapter, not the provider, thus you must enter a value for this field. In this case, the service identifier would be `urn:` followed by the name of the database provider.
D.2 Packaging and Deploying Your Providers

The following sections show the steps you must perform to package and deploy a provider manually:

- Packaging Your Provider
- Deploying Your EAR File
- Testing Deployment
- Setting Deployment Properties
- Securing Your Provider
- Registering Your Provider

D.2.1 Packaging Your Provider

The following steps explain how to manually package a WAR file. If you are familiar with one of the various utilities for assembling WAR files, you are free to assemble your WAR file that way.

- Preparing Your Directories
- Specifying Your Default Service
- Creating Your WAR File
- Creating Your EAR File

D.2.1.1 Preparing Your Directories

In preparation for creating your WAR file, you need to perform the following steps:

1. Create a working directory where you can collect the necessary files.
2. Extract the `template.war` file from `/pdk/jpdk/v2/template.war` into your working directory. Make sure that you extract the file paths, too.
3. If your provider needs any additional JAR files, add them to the `WEB-INF/lib` directory.
4. If your provider needs any additional Java classes not contained in a JAR file, add them to the `WEB-INF/classes` directory. Make sure that you save the class file in a directory structure that corresponds to their Java package names.
5. Add any static HTML files, JSPs and images to your working directory. Create subdirectories as needed to organize the files. Note that the subdirectories will become part of the path necessary to access the HTML or JSP files.
6. Create a subdirectory, `provider_new`, for your provider under the `providers` directory.
7. Copy the `_default.properties` file as `provider_new.properties` under the `WEB-INF/deployment` folder, and edit it to reflect your provider's configuration.
8. Set the `definition` parameter in the `provider_new.properties` file as follows:

   ```plaintext
   definition=providers/provider_new/provider.xml
   ```

9. Place your provider definition file, `provider.xml`, in the `provider_new` subdirectory that you created.
10. Edit _default.properties to reflect the configuration settings of your default provider. The default provider is accessed if a service identifier is not specified in a request. Refer to Section D.2.1.2, "Specifying Your Default Service" for more information on this step.

11. If you use servlets to render content, edit WEB-INF/web.xml to add your servlets to the list of pre-defined servlets. Be careful not to remove the entries for servlets required by PDK-Java.

D.2.1.2 Specifying Your Default Service

The default service is the provider that receives any request without a service name. You specify a default provider by editing the _default.properties file in the deployment directory of your WAR file.

Edit the definition entry to point to the provider definition file that represents your default provider. Paths should be relative to the WEB-INF directory within your WAR file, not the physical location of the file in the file system.

The _default.properties file looks similar to the following:

```
serviceClass=oracle.webdb.provider.v2.adapter.soapV1.ProviderAdapter
loaderClass=oracle.portal.provider.v2.http.DefaultProviderLoader
showTestPage=true
definition=providers/sample/provider.xml
autoReload=true
```

D.2.1.3 Creating Your WAR File

Once you have specified the contents of your WAR file, you are ready to create the WAR file itself. To create the WAR file, do the following:

1. Zip the contents of the working directory you created in Section D.2.1.1, "Preparing Your Directories", including the subdirectory paths but not the working directory path itself.

2. Rename the resulting file to give it a meaningful name and change the extension to .war.

D.2.1.4 Creating Your EAR File

The following steps represent manual configuration of an EAR file. To create the EAR file, perform the following steps:

1. Create another working directory for the creation of your EAR file.

2. Extract the template.ear file from /pdk/jpdk/v2/template.ear into your working directory. Make sure that you extract the file paths, too.

3. Open the META-INF/application.xml file that was contained in the template EAR file. It should look something like the following:

```xml
<?xml version "1.0">
<!DOCTYPE application PUBLIC "-//Sun Microsystems, Inc. //DTD J2EE Application 1.3//EN" "http://java.sun.com/j2ee/dtds/application_1_3.dtd">
<application>
  <display-name>Display Name of the Application</display-name>
  <description>Description of the application</description>
  <module>
    <web>
      <web-uri>yourwarfile.war</web-uri>
      <context-root></context-root>
    </web>
  </module>
</application>
```
Table D–1 describes the elements of application.xml.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;display-name&gt;</td>
<td>Is the name of the application.</td>
</tr>
<tr>
<td>&lt;description&gt;</td>
<td>Is a description of the application and its functions.</td>
</tr>
<tr>
<td>&lt;web-uri&gt;</td>
<td>Is the name of your WAR file.</td>
</tr>
<tr>
<td>&lt;context-root&gt;</td>
<td>Is the prefix you would like to map to your application by default (for example, /myapp).</td>
</tr>
</tbody>
</table>

4. Save application.xml back to the same location without changing the name of the file.

5. Copy the WAR file you created earlier into your working directory. Put it in the working directory itself, not a subdirectory.

6. Zip the contents of the working directory, including the subdirectory paths but not the working directory path itself.

7. Rename the resulting file to give it a meaningful name and change the extension to .ear.

D.2.2 Deploying Your EAR File

You can deploy your EAR file in any one the following ways depending upon your requirements:

- Deploying with the Grid Control Console
- Deploying Manually with dcmctl
- Deploying Manually to Standalone OC4J

D.2.2.1 Deploying with the Grid Control Console

To deploy your EAR file using the Grid Control Console, you must have PDK-Java installed in your target OC4J. Perform the following steps:

1. Start the Grid Control Console and navigate to the home page of the OC4J instance in which you configured PDK-Java (for example, OC4J_Portal).
2. On the Applications tab, click Deploy EAR file.
3. Enter the information in Table D–2.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>J2EE Application</td>
<td>Browse to the location of your EAR file in your local file system</td>
</tr>
<tr>
<td>Application Name</td>
<td>Enter the unique name you want to associate with your provider application.</td>
</tr>
<tr>
<td>Parent Application</td>
<td>Use default.</td>
</tr>
</tbody>
</table>
4. **Click Continue.** The URL mapping for Web Modules displays. The mappings will default to the context roots specified in `application.xml` (for example, `/myapp`), but you can change them to avoid clashing with the context roots of other deployed applications.

5. **Click Finish.** A summary appears with all of the information you entered.

6. **Click Deploy.**

7. **Click OK.**

Your provider application is now deployed to your Oracle Application Server instance. You should see the newly deployed application in the list of applications for the selected OC4J instance. Once you have successfully deployed your EAR file, you need to test the deployment. Refer to Section D.2.3, "Testing Deployment".

**D.2.2.2 Deploying Manually with dcmctl**

To deploy your EAR file using `dcmctl`, you must have PDK-Java installed in your target OC4J.

---

**Note:** Before using `dcmctl` to manage an Oracle Application Server instance, make sure you have no Grid Control Console processes managing that same instance. If multiple processes manage the same instance, you run the risk of inconsistencies or corruption in the data used to manage the instance.

---

You deploy your EAR file using the command-line deployment utility `dcmctl`:

(Arguments in brackets are optional.)

```
  cd MID_TIER_ORACLE_HOME/dcm/bin/
  ./dcmctl deployApplication -f file -a app_name
   [-co comp_name] [-enableIIOP] [-rc rootcontext] [-pa parent_name]
```

where:

- `file` is the name of the EAR or WAR file you want to deploy.
- `app_name` is the name of the application specified by the user in the original deployment.
- `comp_name` is the name of the OC4J instance to which the application will be deployed. The default is the home instance. (Optional)
- `enable IIOP` enables the Internet Inter-Orb Protocol. (Optional)
- `rootcontext` is the base path used in the URL to access the Web module (for example, `http://hostname:port/context root`). This option applies only to the deployment of WAR files. (Optional)
- `parent_name` is the parent application name. The parent application contains common classes used by child applications. (Optional)

Your provider application is now deployed to your Oracle Application Server instance. Once you have successfully deployed your EAR file, you need to test the deployment. Refer to Section D.2.3, "Testing Deployment".
D.2.2.3 Deploying Manually to Standalone OC4J

To deploy your EAR file to a standalone instance of OC4J, it must be a compatible version and have PDK-Java installed. Perform the following steps to deploy your EAR file to a standalone instance of OC4J:

1. If OC4J is not already running, start it as a background process with the following commands:

   On Microsoft Windows:
   
   ```
   cd OC4j_HOME\j2ee\home
   start java -server -Xmx256m -jar oc4j.jar
   ```

   On UNIX/Linux (Bourne shell):
   
   ```
   cd OC4J_HOME/j2ee/home
   java -server -Xmx256m -jar oc4j.jar &
   ```

   where:

   OC4J_HOME is your OC4J installation root directory (for example, D:\oc4j904).

   **Note:** The -Xmx256m option specifies a maximum heap size of 256 Mb for the OC4J process, which is the recommended setting. You may raise or lower this setting to suit your application. If you encounter java.lang.OutOfMemoryError exceptions, you should raise this setting.

2. Deploy your EAR file using the following command:

   ```
   java -jar admin.jar ormi://localhost admin admin_password -deploy
   -deploymentName application_name -file ear_file_path
   ```

   where:

   admin_password is the OC4J administration password you set on installation.

   application_name is the unique name given to the application for administrative purposes.

   ear_file_path is the full path to your EAR file on the file system.

3. For each of the WAR files in your EAR file (as listed in OC4J_HOME/j2ee/home/applications/application_name/application.xml), bind the corresponding Web applications to a URI path on your Web site with the following command:

   ```
   java -jar admin.jar ormi://localhost admin admin_password
   -bindWebApp application_name web_app_name
   file:OC4J_HOME/j2ee/home/config/default-web-site.xml context_root
   ```

   where:

   web_app_name is WAR file name without the .war extension (for example, jpdk).

   context_root is the URI path prefix you would like to be mapped to that Web application (for example, /myapp).

   Your provider application is now deployed to your Oracle Application Server instance. Once you have successfully deployed your EAR file, you need to test the deployment. Refer to Section D.2.3, "Testing Deployment".
D.2.3 Testing Deployment

To test your provider deployment, you access the provider test page with a URL of the following form:

http://host:port/context_root/providers

where:

*host* and *port* are the host name and port number of the HTTP listener for your target OC4J instance. In an Oracle Application Server installation with OracleAS Web Cache installed, *port* should be the OracleAS Web Cache listener port (for example, 7777). In a standalone OC4J installation, the default HTTP port number is 8888.

*context_root* is the URI path prefix you mapped to the provider Web application on deployment (for example, /myapp) or the default one specified in application.xml in the case of a manual deployment with dcmctl.

For example:

http://my.host.com:7777/newProvider/providers

If your .properties file specifies showTestPage=true, you should see the familiar test page for your default provider. To view the test page for a specific provider service, you can append the service name to the URL. For example:

http://my.host.com:7777/newProvider/providers/myService

D.2.4 Setting Deployment Properties

In PDK-Java, you can specify a number of deployment properties using JNDI variables. Table D–3 provides a list of these variables with descriptions.

### Table D–3 JNDI Variables for Provider Deployment

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oracle/portal/provider/global/log/logLevel</td>
<td>Is the logging level (0-8) used by PDK-Java and applies to all providers.</td>
</tr>
<tr>
<td>oracle/portal/service_name/showTestPage</td>
<td>Is a Boolean flag that specifies whether a provider’s test page is accessible. The default value is true.</td>
</tr>
<tr>
<td>oracle/portal/service_name/maxTimeDifference</td>
<td>Is the provider’s HMAC time difference.</td>
</tr>
<tr>
<td>oracle/portal/service_name/definition</td>
<td>Is the location of the provider’s definition file, provider.xml.</td>
</tr>
<tr>
<td>oracle/portal/service_name/autoReload</td>
<td>Is a Boolean auto reload flag. The default value is true.</td>
</tr>
<tr>
<td>oracle/portal/service_name/sharedKey</td>
<td>Is the HMAC shared key. It has no default value.</td>
</tr>
<tr>
<td>oracle/portal/service_name/rootDirectory</td>
<td>Is the location for provider customizations. It has no default value.</td>
</tr>
</tbody>
</table>

Setting the Variables

You can set the values of the variables in Table D–3 as you would any other JNDI variables. Refer to Section 7.2.4.2, “Setting JNDI Variable Values” for information on how to set JNDI variables.
D.2.5 Securing Your Provider

When using the PDK-Java framework in a production environment, you should secure your providers. For more information on securing providers, refer to Section 7.2.6, "Implementing Portlet Security" and the Oracle Application Server Portal Configuration Guide.

D.2.6 Registering Your Provider

Once you have successfully deployed and verified your provider, you can register it as you would any other provider. Refer to Section 6.6.2.5, "Registering and Viewing Your Portlet" for more information about registering your provider.
The OracleAS Portal Provider Test Suite performs sanity, performance, and unit tests on a Web provider without an installation of OracleAS Portal or Internet access from the provider machine. The following two types of utilities are available to the user in the test suite:

- **Provider Test Page**
- **Test Harness**

You can download the OracleAS Portal Provider Test Suite from the Oracle Application Server Portal Developer Kit (PDK) page on OTN:


Click **Download the JPDK Test Suite**.

**E.1 Provider Test Page**

The provider test page provides a basic sanity test for the provider. It contains a list of portlets, servlet initialization arguments, and the version numbers of the `ptlshare` and `pdkjava` libraries. The provider test page is the simplest utility available for testing any Web provider. You access the test page by a URL after deploying the enterprise application or Web application on OC4J. You test the Web provider by accessing this URL from a browser:

http://server:port/application_name/providers/provider_name

For example, the PDK-Java comes with a sample application and portlets. The application is encapsulated in a WAR file, which in turn is encapsulated in an EAR file. When you deploy it, OC4J extracts the files and creates a directory structure with the sample portlets under:

`ORACLE_HOME/j2ee/home/applications/jpdk`

To view the test page for this provider, you would use this URL:

http://server:port/jpdk/providers/sample

When you access the test page, the SOAP servlet validates the XML provider definition, `provider.xml`, ensuring that the corresponding provider is well formed. This validation is useful for debugging deployment issues with your provider before attempting to register it with a OracleAS Portal. If you successfully deploy your Web application on OC4J, you receive a success message on the test page.

You can turn the test page on and off by using the JNDI variable `oracle/portal/provider/provider_name/showTestPage` to true or false.
Once a provider is tested and in service you might want to restrict access to the test page. For more information about retrieving and setting JNDI variables, refer to Section 7.2.4, "Using JNDI Variables".

### E.2 Test Harness

The test harness is a command line utility for unit- and performance-testing your providers without accessing an OracleAS Portal instance. The test harness sends HTTP requests to the target Web provider and records the responses for further analysis. The responses are logged into an XML file. Performance statistics are logged into another file for analysis.

The test harness provides considerable flexibility in the following ways:

- You create your own test definition. Based on the information in the test definition, the harness sends requests to the provider. The test definition file is in XML format and lists request instances to send to the target Web provider.

- The information returned by the provider is stored in a standard XML file, which makes it easier to understand and analyze.

- You can perform load testing of the Web provider.

#### E.2.1 Test Definition File

The test definition file is an XML file that lists the request instances for a particular test. You can optionally subdivide the request instances into request groups within the test definition file. You can include the details of the request instances in the test definition file or refer to a request library XML file that defines the instance details.

In addition to the request instances, the test definition file also includes information about the host and port of the target Web provider and any pre-processors to which it needs to be sent. A pre-processor enables you to include application-specific logic in the request instances at runtime in the test harness. For example, a pre-processor might check the validity of XML in the SOAP messages being sent to the target Web provider. The test harness provides the following three built-in pre-processors:

- OracleAS Portal pre-processor
- Validation-based caching pre-processor
- HMAC (Hashed Message Authentication Checksum) pre-processor

The following sample test definition file illustrates the format and content of the file:

```xml
<xml version="1.0" encoding="ISO-8859-1">
<testDefinition version="0.1">
<testDefinition>
<defaultHost>machine.name.com</defaultHost>
<defaultPort>80</defaultPort>
<defaultPath>/jpdk/providers/sample</defaultPath>
<property name="portletId" value="1"/>
<preProcessorDefinitions>
<preProcessor class="oracle.webdb.testharness.preprocessor.PortalPreProcessor">
 <name>portal</name>
</preProcessor>
<preProcessor class="oracle.webdb.testharness.preprocessor.HMACPreProcessor">
 <name>hmac</name>
 <sharedKey>1234567890aBcDeFgHiJ</sharedKey>
</preProcessor>
</preProcessorDefinitions>
</testDefinition>
</testDefinition>
```
E.2.2 runTest Command

The runTest command invokes the test harness with the specified options using the specified test definition.

```
runTest options test_definition
```

Table E–1 describes the command options for runTest.

### Table E–1  runTest Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-n testname</code></td>
<td>Assigns the specified name to this run. Defaults to a date-time name.</td>
</tr>
<tr>
<td><code>-g groupId</code></td>
<td>Is a comma separated list of the identifiers of the request groups in the test definition file to be run for this test. To run all groups, specify <code>--all-groups</code>.</td>
</tr>
<tr>
<td><code>-p perfLogFile</code></td>
<td>Is the name of the performance log file. Defaults to the test name. To disable performance testing, use <code>--no-perf</code>.</td>
</tr>
<tr>
<td><code>-l resLogFile</code></td>
<td>Is the name of the response log file. Defaults to the test name.</td>
</tr>
<tr>
<td><code>-v resLogLevel</code></td>
<td>Is the level of response logging (MIN</td>
</tr>
<tr>
<td><code>-c csvFile</code></td>
<td>Is the name of the response data CSV file. Default is to not generate it.</td>
</tr>
<tr>
<td><code>-s ctlFile</code></td>
<td>Is the name of the response data ctl file. Default is to not generate it.</td>
</tr>
<tr>
<td><code>--no-perf</code></td>
<td>Switches off all logging of performance data.</td>
</tr>
<tr>
<td><code>--all-groups</code></td>
<td>Runs all groups in the test definition.</td>
</tr>
<tr>
<td><code>-verbose</code></td>
<td>Produces verbose informational logging.</td>
</tr>
</tbody>
</table>
E.2.3 Running a Test with Test Harness

In order to test Web providers using the test harness, do the following:

1. Extract the pdktestsuite.zip file to a convenient location. This will create a directory called pdktest to which all the test harness files are extracted.

2. Set an environment variable called PDKTEST_HOME to point to the location created in the previous step. For example, if you extracted to your D drive, you would now have a directory called D:/pdktest. Set the value of PDKTEST_HOME, to D:/pdktest. The Test Harness provides executable scripts in the bin directory.

3. Create a test definition file similar to the one in Section E.2.1, "Test Definition File".

4. Invoke the runTest command as described in Section E.2.2, "runTest Command".
Content Management APIs and Views

This appendix lists the supported OracleAS Portal content management APIs and views. It contains the following sections:

- Supported APIs
- Secure Views

F.1 Supported APIs

This section provides an overview of the APIs available to you for performing content management tasks.

For more information about the contents of the different packages, refer to the Oracle Application Server Portal PL/SQL API Reference on Portal Center:

http://portalcenter.oracle.com

In the Portal Focus Areas section, click Portlet Development, then in the APIs and References section, click PL/SQL API Reference.

F.1.1 The WWSBR_API Package

This package contains APIs for accessing and manipulating the contents in the portal schema of the OracleAS Metadata Repository. Use these APIs to do the following:

- Create, edit, and delete portal objects, such as pages, categories, and items.
- Copy and move items and pages.
- Check items in and out.
- Approve or reject pending items.

F.1.2 The WWSRC_API Package

This package contains APIs for searching content in the OracleAS Metadata Repository. Use these APIs to do the following:

- Search for items, pages, categories, and perspectives.
- Filter search results based on specific attribute values.
- Return search results as an XML document.
F.1.3 The WWSEC_API Package

This package contains APIs for administering OracleAS Portal security. Use these APIs for the following:

- User maintenance (create user profile entries, activate portal access, update user properties, delete user profile entries, and other associated tasks).
- Group maintenance (create, activate, update, delete groups, and other associated tasks).
- Access control (grant, check, copy, update, remove user/group privileges, and other associated tasks).

F.1.4 The WWCTX_API Package

This package contains APIs for enabling access to a session context. These APIs provide the methods necessary to manage a session context for a specific user.

A session is established when a user accesses the portal. A public session is created upon initial access. It is converted to an authenticated session after the user logs in and exists until the user logs out. Old sessions are also periodically cleaned up by the portal.

F.1.5 The WWPRO_API_INVALIDATION Package

This package contains APIs for invalidating content cached in the Oracle Application Server Web Cache. Use these APIs when your code causes content to change to make sure that those changes are immediately visible to users. For example, you may need to call an API to invalidate the cache in code that adds an item to a page.

F.2 Secure Views

The views in wwsbr_api_views enable you to safely build custom queries against the portal schema of the OracleAS Metadata Repository, without having to worry about the definitions changing between releases. Table F–1 lists the currently supported views.

<table>
<thead>
<tr>
<th>View</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WWSBR_ALL_CATEGORIES</td>
<td>Describes all categories.</td>
</tr>
<tr>
<td>WWSBR_ALL_CONTENT AREAS</td>
<td>Describes all page groups.</td>
</tr>
<tr>
<td>WWSBR_ALL_FOLDER_REGIONS</td>
<td>Describes all page regions on pages that the current user is able to view.</td>
</tr>
<tr>
<td>WWSBR_ALL_FOLDERS</td>
<td>Describes pages on which the current user has View privileges or higher.</td>
</tr>
<tr>
<td>WWSBR_ALL_ITEMS</td>
<td>Describes the items that the current user is able to view.</td>
</tr>
<tr>
<td>WWSBR_ALL_NAVIGATION BARS</td>
<td>Describes all navigation pages that the current user is able to view.</td>
</tr>
<tr>
<td>WWSBR_ALL_PERSPECTIVES</td>
<td>Describes all perspectives.</td>
</tr>
<tr>
<td>WWSBR_ALL_STYLES</td>
<td>Describes all styles.</td>
</tr>
<tr>
<td>WWSBR_APPROVER</td>
<td>Describes the approvers for every page group approval step.</td>
</tr>
</tbody>
</table>
Table F–1  (Cont.) Secure Content Repository Views

<table>
<thead>
<tr>
<th>View</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WWSBR_ATTRIBUTES</td>
<td>Describes all attributes.</td>
</tr>
<tr>
<td>WWSBRCONTENT_AREA_APPROVAL</td>
<td>Describes the approval definition for a page group.</td>
</tr>
<tr>
<td>WWSBRCONTENT_AREA_ITEM_TYPES</td>
<td>Describes which item types are available to which page groups.</td>
</tr>
<tr>
<td>WWSBR_FOLDER_ATTRIBUTES</td>
<td>Describes the attributes associated with pages that the current user is able to view.</td>
</tr>
<tr>
<td>WWSBR_FOLDER_PERSPECTIVES</td>
<td>Describes the perspectives associated with pages that the current user is able to view.</td>
</tr>
<tr>
<td>WWSBR_FOLDER_TYPE_ATTRIBUTES</td>
<td>Describes the attributes that are available within each page type.</td>
</tr>
<tr>
<td>WWSBR_FOLDER_TYPES</td>
<td>Describes all page types.</td>
</tr>
<tr>
<td>WWSBR_ITEM_APPROVAL</td>
<td>Describes item approval information for every item that goes through approval.</td>
</tr>
<tr>
<td>WWSBR_ITEM_ATTRIBUTES</td>
<td>Describes the attributes and attribute values associated with items that the current user is able to view.</td>
</tr>
<tr>
<td>WWSBR_ITEM_PERSPECTIVES</td>
<td>Describes the perspectives associated with items that the current user is able to view.</td>
</tr>
<tr>
<td>WWSBR_ITEM_TYPE_ATTRIBUTES</td>
<td>Describes the attributes that are available within each item type.</td>
</tr>
<tr>
<td>WWSBR_ITEM_TYPES</td>
<td>Describes all item types.</td>
</tr>
<tr>
<td>WWSBR_SUBSCRIPTION</td>
<td>Describes user subscription to portal pages and items.</td>
</tr>
<tr>
<td>WWSBR_USER_FOLDERS</td>
<td>Describes pages on which the current user has Manage privileges or higher.</td>
</tr>
<tr>
<td>WWSBR_USER_PAGES</td>
<td>Describes the pages on which the current user has Manage privileges or higher.</td>
</tr>
</tbody>
</table>

Note: In the following tables, an asterisk (*) indicates that the column is part of the view’s primary key.

F.2.1 WWSBR_ALL_CATEGORIES

Table F–2 lists the columns in the WWSBR_ALL_CATEGORIES view, which describes all categories.

Table F–2  WWSBR_ALL_CATEGORIES

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the category.</td>
<td></td>
</tr>
<tr>
<td>CAID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the category.</td>
<td>Foreign key to WWSBR_ALL_CONTENT AREAS.ID</td>
</tr>
<tr>
<td>PARENTID</td>
<td></td>
<td>NUMBER</td>
<td>ID of the parent category.</td>
<td>0 (zero) if the category has no parent.</td>
</tr>
</tbody>
</table>
Table F–2 (Cont.) WWSBR_ALL_CATEGORIES

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>LANGUAGE*</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Language code.</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td></td>
<td>VARCHAR2(60)</td>
<td>Internal name of the category. Unique within parent category.</td>
<td></td>
</tr>
<tr>
<td>DISPLAY_NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(256)</td>
<td>Display name, or title, of the category.</td>
<td></td>
</tr>
<tr>
<td>DEFAULT_LANGUAGE</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Language code for the default language of the page group.</td>
<td></td>
</tr>
</tbody>
</table>

F.2.2 WWSBR_ALL_CONTENT AREAS

Table F–3 lists the columns in the WWSBR_ALL_CONTENT AREAS view, which describes all page groups. This view shows all page groups in the repository, regardless of the current user's privileges.

Note: An alternative way of uniquely identifying a row in this view is to use the NAME and LANGUAGE columns.

Table F–3 WWSBR_ALL_CONTENT AREAS

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the page group.</td>
<td></td>
</tr>
<tr>
<td>LANGUAGE*</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Language code.</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(60)</td>
<td>Internal name of the page group.</td>
<td></td>
</tr>
<tr>
<td>DISPLAY_NAME</td>
<td></td>
<td>VARCHAR2(256)</td>
<td>Display name, or title, of the page group.</td>
<td></td>
</tr>
<tr>
<td>DEFAULT_LANGUAGE</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Language code for the default language of the page group.</td>
<td></td>
</tr>
</tbody>
</table>

F.2.3 WWSBR_ALL_FOLDER_REGIONS

Table F–4 lists the columns in the WWSBR_ALL_FOLDER_REGIONS view, which describes all page regions on pages viewable by the current user.

Table F–4 WWSBR_ALL_FOLDER_REGIONS

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID*</td>
<td></td>
<td>NUMBER(38)</td>
<td>ID of the region.</td>
<td></td>
</tr>
<tr>
<td>DISPLAY_NAME*</td>
<td></td>
<td>VARCHAR2(4000)</td>
<td>Display name, or title, of the region.</td>
<td></td>
</tr>
<tr>
<td>LANGUAGE</td>
<td></td>
<td>VARCHARAR(7)</td>
<td>Language code.</td>
<td></td>
</tr>
<tr>
<td>FOLDER_ID</td>
<td></td>
<td>NUMBER(38)</td>
<td>ID of the page containing the region.</td>
<td></td>
</tr>
</tbody>
</table>

Foreign key to:
- WWSBR_ALL_FOLDERS.ID
- WSBR_USER_FOLDERS.ID
Table F–5 lists the columns in the WWSBR_ALL_FOLDERS view, which describes the pages on which the current user has View privileges or higher.

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Unique identifier of the page within the page group.</td>
<td>Foreign key to WWSBR_ALL_CONTENT AREAS.ID</td>
</tr>
<tr>
<td>CAID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the page group.</td>
<td></td>
</tr>
<tr>
<td>LANGUAGE*</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Language code.</td>
<td></td>
</tr>
<tr>
<td>PARENT_ID</td>
<td>NUMBER</td>
<td></td>
<td>ID of the parent page.</td>
<td>0 (zero) if root page of the page group.</td>
</tr>
<tr>
<td>NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(60)</td>
<td>Internal name of the page.</td>
<td>Must be unique within the parent page.</td>
</tr>
<tr>
<td>DISPLAY_NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(256)</td>
<td>Display name, or title, of the page.</td>
<td></td>
</tr>
<tr>
<td>CATEGORY_ID</td>
<td>NUMBER</td>
<td></td>
<td>ID of the category assigned to the page.</td>
<td>Foreign key to WWSBR_ALL_CATEGORIES.ID</td>
</tr>
<tr>
<td>Column</td>
<td>Null?</td>
<td>Data Type</td>
<td>Description</td>
<td>Notes</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>CATEGORY_SITEID</td>
<td></td>
<td>NUMBER</td>
<td>Page group ID for the category assigned to the page.</td>
<td>Foreign key to WWSBR_ALL_CATEGORIES.CAID</td>
</tr>
<tr>
<td>TYPE_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the page type.</td>
<td>Foreign key to WWSBR_FOLDER_TYPES.ID</td>
</tr>
<tr>
<td>TYPE_SITEID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the page type.</td>
<td>Foreign key to WWSBR_FOLDER_TYPES.CAID</td>
</tr>
<tr>
<td>BASE_TYPE_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>The ID of the page base type</td>
<td>Foreign key to WWSBR_FOLDER_TYPES.ID where WWSBR_FOLDER_TYPES.IS_BASE_FOLDER_TYPE=1</td>
</tr>
<tr>
<td>IS_PORTLET</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates if the page is published as a portlet.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 0 - not a portlet</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 1 - is a portlet</td>
</tr>
<tr>
<td>IS_CACHING_ON</td>
<td></td>
<td>VARCHAR2</td>
<td>DEPRECATED. This column is retained for backward compatibility only.</td>
<td></td>
</tr>
<tr>
<td>CACHE_MODE</td>
<td></td>
<td>NUMBER(1)</td>
<td>The caching mode for the page.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 2 - no caching</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 1 - cache page definition only</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 0 - cache page definition and content for x minutes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 4 - cache page definition only at system level</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 3 - cache page definition and content at system level for x minutes</td>
</tr>
<tr>
<td>SUB_FOLDER_SEQUENCE</td>
<td></td>
<td>NUMBER</td>
<td>The sequence (order) of the page in its parent page’s sub-page region.</td>
<td></td>
</tr>
<tr>
<td>DISPLAY_IN_PARENT_FOLDER</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates if the page is displayed in its parent page’s sub page region.</td>
<td></td>
</tr>
<tr>
<td>ITEMVERSIONING</td>
<td></td>
<td>VARCHAR2(30)</td>
<td>Determines the level of item versioning for the page.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ versionnone - no versioning</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ versionsimple - simple versioning</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ versionaudit - audit versioning</td>
</tr>
<tr>
<td>Column</td>
<td>Null?</td>
<td>Data Type</td>
<td>Description</td>
<td>Notes</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>STYLE_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID for the style used by the page.</td>
<td>Foreign key to WWSBR_ALL_STYLES.ID</td>
</tr>
<tr>
<td>STYLE_CAID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the style used by the page.</td>
<td>Foreign key to WWSBR_ALL_STYLES.CAID</td>
</tr>
<tr>
<td>URL_VALUE</td>
<td></td>
<td>VARCHAR2(4000)</td>
<td>URL value for a URL type page.</td>
<td></td>
</tr>
<tr>
<td>SEARCH_VALUE</td>
<td></td>
<td>VARCHAR2(2000)</td>
<td>DEPRECATED. This column is retained for backward compatibility only.</td>
<td></td>
</tr>
<tr>
<td>IMAGE</td>
<td></td>
<td>VARCHAR2(350)</td>
<td>Unique name of the image associated with the page.</td>
<td>Matches the NAME column in the document table.</td>
</tr>
<tr>
<td>TITLE_IMAGE_NAME</td>
<td></td>
<td>VARCHAR2</td>
<td>DEPRECATED. This column is retained for backward compatibility only.</td>
<td></td>
</tr>
<tr>
<td>ROLLOVER_IMAGE_NAME</td>
<td></td>
<td>VARCHAR2(350)</td>
<td>The unique name of the rollover image associated with the page.</td>
<td>Matches the NAME column in the document table.</td>
</tr>
<tr>
<td>BANNER_IMAGE_NAME</td>
<td></td>
<td>VARCHAR2</td>
<td>DEPRECATED. This column is retained for backward compatibility only.</td>
<td></td>
</tr>
<tr>
<td>NAVIGATION_BAR_ID</td>
<td></td>
<td>VARCHAR2</td>
<td>DEPRECATED. This column is retained for backward compatibility only.</td>
<td></td>
</tr>
<tr>
<td>NAVIGATION_BAR_CAID</td>
<td></td>
<td>VARCHAR2</td>
<td>DEPRECATED. This column is retained for backward compatibility only.</td>
<td></td>
</tr>
<tr>
<td>IS_PUBLIC</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates that the page is viewable by public users.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td>ITEM_LEVEL_SECURITY</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates that item level security is enabled for the page.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td>DISPLAY_FULL_SCREEN</td>
<td></td>
<td>VARCHAR2</td>
<td>DEPRECATED. This column is retained for backward compatibility only.</td>
<td></td>
</tr>
<tr>
<td>PLSQL_FOLDER_EXECUTOR</td>
<td></td>
<td>VARCHAR2(30)</td>
<td>For PL/SQL type pages, the database schema used to execute the PL/SQL code.</td>
<td></td>
</tr>
</tbody>
</table>
Secure Views

Table F–6 lists the columns in the WWSBR_ALL_ITEMS view, which describes the items viewable by the current user.

For an example of how to use this view, refer to Section 10.3.4, "Finding an Item ID".

Table F–6  WWSBR_ALL_ITEMS

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATEDATE</td>
<td></td>
<td>DATE</td>
<td>Date the page was created.</td>
<td></td>
</tr>
<tr>
<td>CREATOR</td>
<td></td>
<td>VARCHAR2(256)</td>
<td>User name of the person who created the page.</td>
<td></td>
</tr>
<tr>
<td>UPDATEDATE</td>
<td></td>
<td>DATE</td>
<td>Date the page was last updated.</td>
<td></td>
</tr>
<tr>
<td>UPDATOR</td>
<td></td>
<td>VARCHAR2(256)</td>
<td>User name of the person who last updated the page.</td>
<td></td>
</tr>
</tbody>
</table>

F.2.5  WWSBR_ALL_ITEMS

Table F–6 lists the columns in the WWSBR_ALL_ITEMS view, which describes the items viewable by the current user.

For an example of how to use this view, refer to Section 10.3.4, "Finding an Item ID".

Table F–6  WWSBR_ALL_ITEMS

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MASTERID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Master ID of the item.</td>
<td>All versions and translations of an item have the same master ID.</td>
</tr>
<tr>
<td>ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the item.</td>
<td></td>
</tr>
<tr>
<td>CAID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the item.</td>
<td>Foreign key to:</td>
</tr>
<tr>
<td>LANGUAGE*</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Language code.</td>
<td></td>
</tr>
<tr>
<td>IS_CURRENT_VERSION*</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates that the item is the current version.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td>FOLDER_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the page containing the item.</td>
<td>Foreign key to:</td>
</tr>
</tbody>
</table>

F-8   Oracle Application Server Portal Developer's Guide
<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOLDER_REGION_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the region containing the item.</td>
<td>Foreign key to WWSBR_ALL_FOLDER_REGIONS.ID</td>
</tr>
<tr>
<td>NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(256)</td>
<td>Internal name of the item.</td>
<td>The name is a system generated GUID value that can be used as an alternate ID. The name is also used in path based URLs.</td>
</tr>
<tr>
<td>DISPLAY_NAME</td>
<td></td>
<td>VARCHAR2(256)</td>
<td>Display name, or title, of the item.</td>
<td></td>
</tr>
<tr>
<td>ITEMTYPE</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>The base item type, for example, BASEFILE, BASETEXT, BASEURL, and so on.</td>
<td></td>
</tr>
<tr>
<td>SUBTYPE</td>
<td></td>
<td>VARCHAR2(40)</td>
<td>ID of the item type.</td>
<td>Foreign key to WWSBR_ITEM_TYPES.ID</td>
</tr>
<tr>
<td>SUBTYPE_CAID</td>
<td>NUMBER</td>
<td></td>
<td>Page group ID of the item type.</td>
<td>Foreign key to WWSBR_ITEM_TYPES.CAID</td>
</tr>
<tr>
<td>PARENT_ITEM_ID</td>
<td>NUMBER</td>
<td></td>
<td>ID of the parent item, if the item is a sub item.</td>
<td>0 (zero) if the item is not a sub item.</td>
</tr>
<tr>
<td>CATEGORY_ID</td>
<td>NUMBER</td>
<td></td>
<td>ID for the category assigned to the item.</td>
<td>Foreign key to WWSBR_ALL_CATEGORIES.ID</td>
</tr>
<tr>
<td>CATEGORY_CAID</td>
<td>NUMBER</td>
<td></td>
<td>page group ID for the category assigned to the item.</td>
<td>Foreign key to WWSBR_ALL_CATEGORIES.CAID</td>
</tr>
<tr>
<td>AUTHOR</td>
<td></td>
<td>VARCHAR2(50)</td>
<td>Author of the item.</td>
<td></td>
</tr>
<tr>
<td>PUBLISH_DATE</td>
<td>NOT NULL</td>
<td>DATE</td>
<td>Date the item will be published.</td>
<td></td>
</tr>
<tr>
<td>EXPIREMODE</td>
<td></td>
<td>VARCHAR2(90)</td>
<td>Expiration mode for the item.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 'PERMANENT' - item never expires</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 'NUMBER' - item expires in EXPIRENUMBER days from CREATEDATE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 'DATE' - item expires on EXPIREDATE</td>
</tr>
<tr>
<td>EXPIRENUMBER</td>
<td>NUMBER</td>
<td></td>
<td>Number of days after CREATEDATE when item will expire.</td>
<td>Only valid when EXPIREMODE = 'NUMBER'.</td>
</tr>
<tr>
<td>EXPIREDATE</td>
<td>DATE</td>
<td></td>
<td>Date when item will expire.</td>
<td>Only valid when EXPIREMODE = 'DATE'.</td>
</tr>
</tbody>
</table>
Table F–6 (Cont.) WWSBR_ALL_ITEMS

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMAGE</td>
<td></td>
<td>VARCHAR2(350)</td>
<td>Image associated with the item.</td>
<td>This value may be a reference (for example, a path to an image stored on the file system or in the content repository) or a unique document name that matches the NAME column in the document table.</td>
</tr>
<tr>
<td>KEYWORDS</td>
<td></td>
<td>VARCHAR2(2000)</td>
<td>Keywords for the item.</td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td></td>
<td>VARCHAR2(4000)</td>
<td>URL for a URL type item.</td>
<td></td>
</tr>
<tr>
<td>FILENAME</td>
<td></td>
<td>VARCHAR2(350)</td>
<td>Unique name of the file associated with a file type item.</td>
<td>Matches the NAME column in the document table.</td>
</tr>
<tr>
<td>TEXT</td>
<td></td>
<td>CLOB</td>
<td>Text for a text type item.</td>
<td></td>
</tr>
<tr>
<td>FOLDER_LINK_ID</td>
<td></td>
<td>NUMBER</td>
<td>Page ID for a page link type item</td>
<td>Foreign key to:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ WWSBR_ALL_FOLDERS.ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ WWSBR_USER_FOLDERS.ID</td>
</tr>
<tr>
<td>FOLDER_LINK_CAID</td>
<td></td>
<td>NUMBER</td>
<td>Page group ID for a page link type item.</td>
<td>Foreign key to:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ WWSBR_ALL_FOLDERS.CAID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ WWSBR_USER_FOLDERS.CAID</td>
</tr>
<tr>
<td>ACTIVE*</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates the active status of the item.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 1 - active</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 2 - pending approval as new and current version</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 3 - pending approval as new but not current version</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 0 - pending approval</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ -1 - deleted</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ -7 - rejected</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ -9 - reject deleted</td>
</tr>
<tr>
<td>CAN_BE_CHECKEDOUT</td>
<td></td>
<td>NUMBER(1)</td>
<td>Indicates if the item can be checked out.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 0 - cannot be checked out</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 1 - can be checked out</td>
</tr>
<tr>
<td>Column</td>
<td>Null?</td>
<td>Data Type</td>
<td>Description</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
<td>-----------------</td>
<td>------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>IS_ITEM_</td>
<td></td>
<td>NUMBER(1)</td>
<td>Indicates if the item is</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td>CHECKEDOUT</td>
<td></td>
<td></td>
<td>checked out.</td>
<td>■ 0 - item is not checked out</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 1 - item is checked out</td>
</tr>
<tr>
<td>CHECKER_</td>
<td></td>
<td>VARCHAR2(256)</td>
<td>User name of the user who</td>
<td></td>
</tr>
<tr>
<td>USERNAME</td>
<td></td>
<td></td>
<td>checked out the item.</td>
<td></td>
</tr>
<tr>
<td>CHECKOUT_DATE</td>
<td></td>
<td>DATE</td>
<td>Date the item was checked out.</td>
<td></td>
</tr>
<tr>
<td>FULLSCREEN</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates the rendering</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>behavior for an item shown as a</td>
<td>■ 0 - clicking the link displays the item in a new browser window</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>link.</td>
<td>■ 1 - clicking the link displays the item in the same browser window</td>
</tr>
<tr>
<td>INPLACE</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates the rendering behavior</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>for a text or PL/SQL item.</td>
<td>■ 0 - a link to the item is displayed; the rendering behavior of</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ the link depends on the value of the FULLSCREEN column</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 1 - the item is rendered in-place in the region</td>
</tr>
<tr>
<td>CREATEDATE</td>
<td>NOT NULL</td>
<td>DATE</td>
<td>Date the item was created.</td>
<td></td>
</tr>
<tr>
<td>CREATOR</td>
<td>NOT NULL</td>
<td>VARCHAR2(256)</td>
<td>User name of the person who</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>created the item.</td>
<td></td>
</tr>
<tr>
<td>UPDATEDATE</td>
<td>NOT NULL</td>
<td>DATE</td>
<td>Date the item was last updated.</td>
<td></td>
</tr>
<tr>
<td>UPDATOR</td>
<td>VARCHAR2(256)</td>
<td>DATE</td>
<td>Date the item was last updated. Date the item was last updated.</td>
<td></td>
</tr>
<tr>
<td>SECURITY</td>
<td>VARCHAR2(25)</td>
<td>DATE</td>
<td>Date the item was last updated. Date the item was last updated.</td>
<td></td>
</tr>
<tr>
<td>VISIBLE</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates whether the item is</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>hidden.</td>
<td>■ 0 - item is hidden</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 1 - item is shown</td>
</tr>
</tbody>
</table>
### Secure Views

#### F.2.6 WWSBR_ALL_NAVIGATION_BARS

Table F–7 lists the columns in the WWSBR_ALL_NAVIGATION_BARS view, which describes all navigation pages viewable by the current user.

**Note:** An alternative way of uniquely identifying a row in this view is to use the NAME, CAID, and LANGUAGE columns.

#### Table F–7 WWSBR_ALL_NAVIGATION_BARS

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the navigation page.</td>
<td>Foreign key to WWSBR_ALL_CONTENT AREAS.ID</td>
</tr>
<tr>
<td>CAID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the navigation page.</td>
<td></td>
</tr>
<tr>
<td>LANGUAGE*</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Language code.</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(60)</td>
<td>Internal code of the navigation page.</td>
<td>Unique within page group.</td>
</tr>
<tr>
<td>DISPLAY_NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(256)</td>
<td>Display name, or title, of the navigation page.</td>
<td></td>
</tr>
<tr>
<td>STYLE_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID for the style used by the navigation page.</td>
<td>Foreign key to WWSBR_ALL_STYLES.ID</td>
</tr>
<tr>
<td>STYLE_CAID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the style used by the navigation page.</td>
<td>Foreign key to WWSBR_ALL_STYLES.CAID</td>
</tr>
<tr>
<td>IS_PORTLET</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates if the page is published as a portlet.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td>ALIGNMENT</td>
<td></td>
<td>VARCHAR2</td>
<td>DEPRECATED. This column is retained for backward compatibility.</td>
<td></td>
</tr>
</tbody>
</table>
F.2.7 WWSBR_ALL_PERSPECTIVES

Table F–8 lists the columns in the WWSBR_ALL_PERSPECTIVES view, which describes all perspectives.

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the perspective.</td>
<td></td>
</tr>
<tr>
<td>CAID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the perspective.</td>
<td>Foreign key to WWSBR_ALL_CONTENT AREAS.ID</td>
</tr>
<tr>
<td>PARENTID</td>
<td></td>
<td>NUMBER</td>
<td>ID of the parent perspective.</td>
<td>0 (zero) if the perspective has no parent</td>
</tr>
<tr>
<td>LANGUAGE*</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Language code.</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td></td>
<td>VARCHAR2(60)</td>
<td>Internal name of the perspective.</td>
<td>Unique within parent perspective.</td>
</tr>
<tr>
<td>DISPLAY_NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(240)</td>
<td>Display name, or title, of the perspective.</td>
<td></td>
</tr>
</tbody>
</table>

Note: An alternative way of uniquely identifying a row in this view is to use the NAME, CAID, and LANGUAGE columns.

F.2.8 WWSBR_ALL_STYLES

Table F–9 lists the columns in the WWSBR_ALL_STYLES view, which describes all styles.

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the style.</td>
<td></td>
</tr>
<tr>
<td>CAID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the style.</td>
<td>Foreign key to WWSBR_ALL_CONTENT AREAS.ID</td>
</tr>
<tr>
<td>LANGUAGE*</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Language code.</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(65)</td>
<td>Internal name of the style.</td>
<td>Unique within the page group.</td>
</tr>
<tr>
<td>DISPLAY_NAME</td>
<td></td>
<td>VARCHAR2(350)</td>
<td>Display name, or title, of the style.</td>
<td></td>
</tr>
</tbody>
</table>

F.2.9 WWSBR_APPROVER

Table F–10 lists the columns in the WWSBR_APPROVER view, which describes the approvers for every page group approval step. The STEP_ID column corresponds to the STEP_ID column in the WWSBR_CONTENT_AREA_APPROVAL view. For every step, there could be one or more approvers. An approver can be a user or group and this information is stored in the APPROVER_TYPE column.
Table F–10  WWSBR_APPROVER

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Internal ID, unique to each row.</td>
<td>Foreign key to WWSBR_CONTENT_AREA_APPROVAL.ID</td>
</tr>
<tr>
<td>STEP_ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Sequential step ID of an approval process.</td>
<td></td>
</tr>
<tr>
<td>APPROVER_ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>User ID of the approver.</td>
<td></td>
</tr>
<tr>
<td>APPROVER_TYPE*</td>
<td>NOT NULL</td>
<td>VARCHAR2(10)</td>
<td>The type of the approver.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ USER - approver is a user</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ GROUP - approver is a group</td>
</tr>
</tbody>
</table>

Table F–11  WWSBR_ATTRIBUTES

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the attribute.</td>
<td></td>
</tr>
<tr>
<td>CAID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the attribute.</td>
<td>Foreign key to WWSBR_ALL_CONTENT AREAS.ID</td>
</tr>
<tr>
<td>LANGUAGE*</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Language code.</td>
<td>Functions as alternate key (unique within page group).</td>
</tr>
<tr>
<td>NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(256)</td>
<td>Internal name of the attribute.</td>
<td></td>
</tr>
<tr>
<td>DISPLAY_NAME</td>
<td></td>
<td>VARCHAR2(256)</td>
<td>Display name, or title, of the attribute.</td>
<td></td>
</tr>
<tr>
<td>IS_BASE_ATTRIBUTE</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates that the attribute is a base attribute.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 0 - not a base attribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 1 - is a base attribute</td>
</tr>
<tr>
<td>DATA_TYPE</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Data type of the attribute.</td>
<td></td>
</tr>
<tr>
<td>LENGTH</td>
<td>NOT NULL</td>
<td>NUMBER(5)</td>
<td>Length of the attribute.</td>
<td></td>
</tr>
<tr>
<td>IS_TRANSLATABLE</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates that the attribute is translatable, meaning that a different value can be stored for each translation. If the attribute is not translatable, the same value is stored for all translations.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 0 - not translatable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 1 - is translatable</td>
</tr>
</tbody>
</table>
F.2.11 WWSBR_CONTENT_AREA_APPROVAL

Table F–12 lists the columns in the WWSBR_CONTENT_AREA_APPROVAL view, which describes the approval definition for a page group or a page. For every approval definition there are x rows corresponding to the x steps defined in the approval process. To determine the approvers for each step, see the WWSBR_APPROVER view.

Table F–12 WWSBR_CONTENT_AREA_APPROVAL

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the approval process</td>
<td></td>
</tr>
<tr>
<td>STEP_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the step in the approval process</td>
<td></td>
</tr>
<tr>
<td>CAID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the page group against which the approval</td>
<td></td>
</tr>
<tr>
<td>FOLDER_ID</td>
<td></td>
<td>NUMBER</td>
<td>process is defined.</td>
<td></td>
</tr>
<tr>
<td>ALL_OR_ANY</td>
<td>VARCHAR2(10)</td>
<td></td>
<td>Indicates if all the approvers need to approve the</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>item for it to be published, or just one.</td>
<td>■ ALL - all approvers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>must approve the item</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ ANY - only one approver</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>needs to approve the item</td>
</tr>
<tr>
<td>PARALLEL_OR_</td>
<td>VARCHAR2(10)</td>
<td></td>
<td>Indicates if all the approvers are notified at</td>
<td>Valid values are as</td>
</tr>
<tr>
<td>SERIAL</td>
<td></td>
<td></td>
<td>the same time, or one after the other.</td>
<td>follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ P - all approvers are</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>notified at the same time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ S - approvers are</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>notified one after the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>other</td>
</tr>
</tbody>
</table>

F.2.12 WWSBR_CONTENT_AREA_ITEM_TYPES

Table F–13 lists the columns in the WWSBR_CONTENT_AREA_ITEM_TYPES view, which describes which item types are available to which page groups. An item type cannot be assigned to an item unless the item type is available within the item's page group.

Table F–13 WWSBR_CONTENT_AREA_ITEM_TYPES

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the page group.</td>
<td>Foreign key to WWSBR_ALL_CONTENT_AREAS.ID</td>
</tr>
<tr>
<td>ITEM_TYPE_ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the item type.</td>
<td>Foreign key to WWSBR_ITEM_TYPES.ID</td>
</tr>
<tr>
<td>ITEM_TYPE_CAID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the item type.</td>
<td>Foreign key to WWSBR_ITEM_TYPES.CAID</td>
</tr>
</tbody>
</table>
**F.2.13 WWSBR_FOLDER_ATTRIBUTES**

Table F–14 lists the columns in the WWSBR_FOLDER_ATTRIBUTES view, which describes the attributes associated with pages viewable by the current user.

**Table F–14 WWSBR_FOLDER_ATTRIBUTES**

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOLDER_CAID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID of the page.</td>
<td>Foreign key to:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ WWSBR_ALL_FOLDERS.CAID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ WWSBR_USER_FOLDERS.CAID</td>
</tr>
<tr>
<td>FOLDER_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the page.</td>
<td>Foreign key to:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ WWSBR_ALL_FOLDERS.ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ WWSBR_USER_FOLDERS.ID</td>
</tr>
<tr>
<td>ATTRIBUTE_ID</td>
<td>NUMBER</td>
<td>NUMBER</td>
<td>ID of the attribute.</td>
<td>Foreign key to WWSBR_ATTRIBUTES.ID</td>
</tr>
<tr>
<td>ATTRIBUTE_CAID</td>
<td>NUMBER</td>
<td>NUMBER</td>
<td>Page group ID for the attribute.</td>
<td>Foreign key to WWSBR_ATTRIBUTES.CAID</td>
</tr>
<tr>
<td>LANGUAGE</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Language code.</td>
<td>Foreign key to:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ WWSBR_ALL_FOLDERS.LANGUAGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ WWSBR_USER_FOLDERS.LANGUAGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ WWSBR_ATTRIBUTES.LANGUAGE</td>
</tr>
<tr>
<td>VALUETYPE</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Data type of the attribute.</td>
<td></td>
</tr>
</tbody>
</table>

**F.2.14 WWSBR_FOLDER_PERSPECTIVES**

Table F–15 lists the columns in the WWSBR_FOLDER_PERSPECTIVES view, which describes the perspectives associated with pages viewable by the current user.
Secure Views

Content Management APIs and Views

F.2.15 WWSBR_FOLDER_TYPE_ATTRIBUTES

Table F–16 lists the columns in the WWSBR_FOLDER_TYPE_ATTRIBUTES view, which describes the attributes that are available within each page type.

**Note:** An alternative way of uniquely identifying a row in this view is to use the FOLDER_TYPE_ID, FOLDER_TYPE_CAID, ATTRIBUTE_ID, and ATTRIBUTE_CAID columns.

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the page type attribute row.</td>
<td>Foreign key to WWSBR_ALL_CONTENT AREAS.ID</td>
</tr>
<tr>
<td>CAID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID.</td>
<td></td>
</tr>
<tr>
<td>LANGUAGE*</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Language code.</td>
<td></td>
</tr>
<tr>
<td>FOLDER_TYPE_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the folder type.</td>
<td>Foreign key to WWSBR_FOLDER_TYPES.ID</td>
</tr>
<tr>
<td>FOLDER_TYPE_CAID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the page type.</td>
<td>Foreign key to WWSBR_FOLDER_TYPES.CAID</td>
</tr>
</tbody>
</table>
**F.2.16 WWSBR_FOLDER_TYPES**

Table F–17 lists the columns in the WWSBR_FOLDER_TYPES view, which describes all page types.

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the page type.</td>
<td>Foreign key to WWSBR_ALL_CONTENT_AREAS.ID</td>
</tr>
<tr>
<td>CAID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the page type.</td>
<td></td>
</tr>
<tr>
<td>LANGUAGE*</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Language code.</td>
<td>Functions as alternate key (unique within page group).</td>
</tr>
<tr>
<td>NAME</td>
<td>NULL</td>
<td>VARCHAR2(30)</td>
<td>Internal name of the page type.</td>
<td></td>
</tr>
<tr>
<td>DISPLAY_NAME</td>
<td>NULL</td>
<td>VARCHAR2(256)</td>
<td>Display name, or title, of the page type.</td>
<td></td>
</tr>
<tr>
<td>IS_BASE_FOLDER_TYPE</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Indicates that the item is a base page type.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 0 - not a base page type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 1 - is a base page type</td>
</tr>
<tr>
<td>BASE_FOLDER_TYPE_ID</td>
<td>NULL</td>
<td>NUMBER</td>
<td>ID of the base page type on which this page type is based.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** An alternative way of uniquely identifying a row in this view is to use the NAME, CAID, and LANGUAGE columns.

**Table F–17 WWSBR_FOLDER_TYPES**

**F.2.17 WWSBR_ITEM_APPROVAL**

Table F–18 lists the columns in the WWSBR_ITEM_APPROVAL view, which describes item approval information for every item that goes through approval. You can use this view to find information, such as the following:

- Location of item x in the approval chain
■ Reviewer for item x
■ Approver who rejected item x

In addition to the information about pending items, you can also use this view to query details on approvals, including approval date, comments, approver, and so on.

Table F–18  WWSBR_ITEM_APPROVAL

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Internal ID.</td>
<td></td>
</tr>
<tr>
<td>APPROVAL_INSTANCE_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Internal ID.</td>
<td></td>
</tr>
<tr>
<td>ITEM_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the item pending approval.</td>
<td></td>
</tr>
<tr>
<td>CAID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the page group that contains the item pending approval.</td>
<td></td>
</tr>
<tr>
<td>LANGUAGE</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Language code.</td>
<td></td>
</tr>
<tr>
<td>FOLDER_ID</td>
<td>NUMBER</td>
<td></td>
<td>ID of the page on which the item pending approval appears.</td>
<td></td>
</tr>
<tr>
<td>STEP_ID</td>
<td>NUMBER</td>
<td></td>
<td>ID of the step for which the item requires approval.</td>
<td></td>
</tr>
<tr>
<td>APPROVAL_ID</td>
<td>NUMBER</td>
<td></td>
<td>ID of the approval process for the item.</td>
<td>Foreign key to WWSBR_CONTENT_AREA_APPROVAL.ID</td>
</tr>
<tr>
<td>CURRENT_APPROVER_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the approver of the approval step.</td>
<td>For an ANY approval step this value will be the same for all rows of the approval step. For an ALL approval step this value will be the same as APPROVER_ID.</td>
</tr>
<tr>
<td>CURRENT_APPROVER_TYPE</td>
<td>VARCHAR2(10)</td>
<td></td>
<td>The type of the current approver.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ USER - approver is a user</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ GROUP - approver is a group</td>
</tr>
<tr>
<td>APPROVAL_DATE</td>
<td>DATE</td>
<td></td>
<td>Date when the item was approved.</td>
<td></td>
</tr>
<tr>
<td>APPROVER_ID</td>
<td>NUMBER</td>
<td></td>
<td>ID of the approver.</td>
<td>Use wwsec_api.user_name to get the user name of the approver.</td>
</tr>
<tr>
<td>APPROVER_COMMENT</td>
<td>VARCHAR2(4000)</td>
<td></td>
<td>Comment specified at the time of approval or rejection.</td>
<td></td>
</tr>
</tbody>
</table>
Table F–18  (Cont.) WWSBR_ITEM_APPROVAL

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATUS</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Status of the item.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 1 - active</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 2 - pending approval as new and current version</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 3 - pending approval as new but not current version</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 9 - draft</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 0 - pending approval</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ -1 - deleted</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ -7 - rejected</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ -9 - reject deleted</td>
</tr>
<tr>
<td>SUBMITTER</td>
<td>VARCHAR2(256)</td>
<td>ID of the user who submitted the item for approval.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VALUE1</td>
<td>VARCHAR2(256)</td>
<td>Not used.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

F.2.18  WWSBR_ITEM_ATTRIBUTES

Table F–19 lists the columns in the WWSBR_ITEM_ATTRIBUTES view, which describes the attributes and attribute values associated with items viewable by the current user.

Table F–19  WWSBR_ITEM_ATTRIBUTES

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITEM_CAID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the item.</td>
<td>Foreign key to WWSBR_ALL_ITEMS.CAID</td>
</tr>
<tr>
<td>ITEM_MASTERID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Master ID of the item.</td>
<td>Foreign key to WWSBR_ALL_ITEMS.MASTERID</td>
</tr>
<tr>
<td>ATTRIBUTE_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the attribute.</td>
<td>Foreign key to WWSBR_ATTRIBUTES.ID</td>
</tr>
<tr>
<td>ATTRIBUTE_CAID</td>
<td>NUMBER</td>
<td>Page group ID for the attribute.</td>
<td>Foreign key to WWSBR_ATTRIBUTES.CAID</td>
<td></td>
</tr>
<tr>
<td>LANGUAGE</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Language code.</td>
<td>Foreign key to WWSBR_ALL_ITEMS.LANGUAGE</td>
</tr>
<tr>
<td>VALUETYPE</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Date type of the attribute.</td>
<td></td>
</tr>
<tr>
<td>VALUE</td>
<td>VARCHAR2(4000)</td>
<td>Value of the attribute.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

F.2.19  WWSBR_ITEM_PERSPECTIVES

Table F–20 lists the columns in the WWSBR_ITEM_PERSPECTIVES view, which describes the perspectives associated with items viewable by the current user.
Table F–20  WWSBR_ITEM_PERSPECTIVES

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITEM_CAID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the item.</td>
<td>Foreign key to WWSBR_ALL_ITEMS.CAID</td>
</tr>
<tr>
<td>ITEM_ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the item.</td>
<td>Foreign key to WWSBR_ALL_ITEMS.ID</td>
</tr>
<tr>
<td>ITEM_MASTERID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Master ID of the item.</td>
<td>Foreign key to WWSBR_ALL_ITEMS.MASTERID</td>
</tr>
<tr>
<td>PERSPECTIVE_ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the perspective.</td>
<td>Foreign key to WWSBR_ALL_PERSPECTIVES.ID</td>
</tr>
<tr>
<td>PERSPECTIVE_CAID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the perspective.</td>
<td>Foreign key to WWSBR_ALL_PERSPECTIVES.CAID</td>
</tr>
<tr>
<td>PERSPECTIVE_NAME</td>
<td></td>
<td>VARCHAR2(60)</td>
<td>Internal name of the perspective.</td>
<td>Foreign key to WWSBR_ALL_PERSPECTIVES.NAME</td>
</tr>
</tbody>
</table>

F.2.20  WWSBR_ITEM_TYPE_ATTRIBUTES

Table F–21 lists the columns in the WWSBR_ITEM_TYPE_ATTRIBUTES view, which describes the attributes that are available within each item type.

Note: An alternative way of uniquely identifying a row in this view is to use the ITEM_TYPE_ID, ITEM_TYPE_CAID, ATTRIBUTE_ID, and ATTRIBUTE_CAID columns.

Table F–21  WWSBR_ITEM_TYPE_ATTRIBUTES

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the item type attribute row.</td>
<td>Foreign key to WWSBR_ALL_CONTENTAREAS.ID</td>
</tr>
<tr>
<td>CAID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID.</td>
<td>Foreign key to WWSBR_ALL_ITEM_TYPES.ID</td>
</tr>
<tr>
<td>LANGUAGE*</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Language code.</td>
<td>Foreign key to WWSBR_ALL_CONTENTAREAS.ID</td>
</tr>
<tr>
<td>ITEM_TYPE_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the item type.</td>
<td>Foreign key to WWSBR_ALL_ITEM_TYPES.ID</td>
</tr>
<tr>
<td>ITEM_TYPE_CAID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the item type.</td>
<td>Foreign key to WWSBR_ALL_ITEM_TYPES.CAID</td>
</tr>
<tr>
<td>ATTRIBUTE_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the attribute.</td>
<td>Foreign key to WWSBR_ALL_ATTRIBUTES.ID</td>
</tr>
</tbody>
</table>
Table F–21 (Cont.) WWSBR_ITEM_TYPE_ATTRIBUTES

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTRIBUTE_CAID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the attribute.</td>
<td>Foreign key to WWSBR_ATTRIBUTES.CAID</td>
</tr>
<tr>
<td>HIDDEN_ATTRIBUTE</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates if the attribute should be hidden.</td>
<td>This column is used for certain built-in item types and is not intended for customer use.</td>
</tr>
<tr>
<td>DEFAULT_VALUE</td>
<td></td>
<td>VARCHAR2(2000)</td>
<td>Default value of the attribute.</td>
<td>The default value is always stored as VARCHAR2, regardless of the attribute's base type.</td>
</tr>
</tbody>
</table>

Table F–22 WWSBR_ITEM_TYPES

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the item type.</td>
<td></td>
</tr>
<tr>
<td>CAID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the item type.</td>
<td>Foreign key to WWSBR_ALL_CONTENT AREAS.ID</td>
</tr>
<tr>
<td>LANGUAGE*</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Language code.</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td></td>
<td>VARCHAR2(30)</td>
<td>Internal name of the item type.</td>
<td>Functions as alternate key (unique within page group).</td>
</tr>
<tr>
<td>DISPLAY_NAME</td>
<td></td>
<td>VARCHAR2(256)</td>
<td>Display name, or title, of the item type.</td>
<td></td>
</tr>
</tbody>
</table>

Note: An alternative way of uniquely identifying a row in this view is to use the NAME, CAID, and LANGUAGE columns.
Table F–22 (Cont.) WWSBR_ITEM_TYPES

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS_BASE_ITEM_TYPE</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Indicates that the item is a base item type.</td>
<td>Only non-base item types can be assigned to items. Valid values are as follows: 0 - not a base item type, 1 - is a base item type</td>
</tr>
<tr>
<td>BASE_ITEM_TYPE_ID</td>
<td>NUMBER</td>
<td>ID of the base item type on which this item type is based.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

F.2.22 WWSBR_SUBSCRIPTION

Table F–23 lists the columns in the WWSBR_SUBSCRIPTION view, which describes user subscription to portal pages and items.

Table F–23 WWSBR_SUBSCRIPTION

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Internal ID.</td>
<td></td>
</tr>
<tr>
<td>OBJECT_CONTEXT_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the page group to which the subscribed object belongs.</td>
<td></td>
</tr>
<tr>
<td>FOLDER_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the page to which the subscribed object belongs.</td>
<td>In the case of a page subscription, this value is NULL.</td>
</tr>
<tr>
<td>OBJECT_ID</td>
<td>NUMBER</td>
<td>ID of the subscribed item.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GROUP_ID</td>
<td>NUMBER</td>
<td>Not used.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECIPIENT_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the user or group who has subscribed to the object.</td>
<td></td>
</tr>
<tr>
<td>RECIPIENT_TYPE</td>
<td>VARCHAR2(10)</td>
<td>Not used.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

F.2.23 WWSBR_USER_FOLDERS

Table F–24 lists the columns in the WWSBR_USER_FOLDERS view, which describes the pages on which the current user has Manage privileges or higher.
<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Unique identifier of the page within the page group.</td>
<td>Foreign key to WWSBR_ALL_CONTENT AREAS.ID</td>
</tr>
<tr>
<td>CAID*</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the page group.</td>
<td></td>
</tr>
<tr>
<td>LANGUAGE*</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Language code.</td>
<td></td>
</tr>
<tr>
<td>PARENT_ID</td>
<td>NUMBER</td>
<td></td>
<td>ID of the parent page.</td>
<td>0 (zero) if root page of the page group.</td>
</tr>
<tr>
<td>NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(60)</td>
<td>Internal name of the page.</td>
<td>Must be unique within parent page.</td>
</tr>
<tr>
<td>DISPLAY_NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(256)</td>
<td>Display name, or title, of the page.</td>
<td></td>
</tr>
<tr>
<td>CATEGORY_ID</td>
<td>NUMBER</td>
<td></td>
<td>ID of the category assigned to the page.</td>
<td>Foreign key to WWSBR_ALL_CATEGORIES.ID</td>
</tr>
<tr>
<td>CATEGORY_CAID</td>
<td>NUMBER</td>
<td></td>
<td>Page group ID for the category assigned to the page.</td>
<td>Foreign key to WWSBR_ALL_CATEGORIES.CAID</td>
</tr>
<tr>
<td>TYPE_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the page type.</td>
<td>Foreign key to WWSBR_FOLDER_TYPES.ID</td>
</tr>
<tr>
<td>TYPE_CAID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the page type.</td>
<td>Foreign key to WWSBR_FOLDER_TYPES.CAID</td>
</tr>
<tr>
<td>BASE_TYPE_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>The ID of the page base type</td>
<td>Foreign key to WWSBR_FOLDER_TYPES.ID where WWSBR_FOLDER_TYPES.IS_BASE_FOLDER_TYPE=1</td>
</tr>
<tr>
<td>IS_PORTLET</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates if the page is published as a portlet.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 0 - not a portlet</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 1- is a portlet</td>
</tr>
<tr>
<td>IS_CACHING_ON</td>
<td></td>
<td>VARCHAR2</td>
<td>DEPRECATED. This column is retained for backward compatibility only.</td>
<td></td>
</tr>
<tr>
<td>Column</td>
<td>Null?</td>
<td>Data Type</td>
<td>Description</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CACHE_MODE</td>
<td></td>
<td>NUMBER(1)</td>
<td>The caching mode for the page.</td>
<td>Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 2 - no caching</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 1 - cache page definition only</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 0 - cache page definition and content for x minutes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 4 - cache page definition only at system level</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 3 - cache page definition and content at system level for x minutes</td>
</tr>
<tr>
<td>SUB_FOLDER_SEQUENCE</td>
<td></td>
<td>NUMBER</td>
<td>The sequence (order) of the page in its parent page's sub-page region.</td>
<td></td>
</tr>
<tr>
<td>DISPLAY_IN_PARENT_FOLDER</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates if the page is displayed in its parent page's sub page region.</td>
<td></td>
</tr>
<tr>
<td>ITEMVERSIONING</td>
<td></td>
<td>VARCHAR2(30)</td>
<td>Determines the level of item versioning for the page.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ versionnone - no versioning</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ versionsimple - simple versioning</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ versionaudit - audit versioning</td>
</tr>
<tr>
<td>STYLE_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID for the style used by the page.</td>
<td>Foreign key to WWSBR_ALL_STYLES.ID</td>
</tr>
<tr>
<td>STYLE_CAID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID for the style used by the page.</td>
<td>Foreign key to WWSBR_ALL_STYLES.CAID</td>
</tr>
<tr>
<td>URL_VALUE</td>
<td></td>
<td>VARCHAR2(4000)</td>
<td>URL value for a URL type page.</td>
<td></td>
</tr>
<tr>
<td>SEARCH_VALUE</td>
<td></td>
<td>VARCHAR2(2000)</td>
<td>DEPRECATED. This column is retained for backward compatibility only.</td>
<td></td>
</tr>
<tr>
<td>IMAGE</td>
<td></td>
<td>VARCHAR2(350)</td>
<td>Unique name of the image associated with the page.</td>
<td>Matches the NAME column in the document table.</td>
</tr>
<tr>
<td>TITLE_IMAGE_NAME</td>
<td></td>
<td>VARCHAR2</td>
<td>DEPRECATED. This column is retained for backward compatibility only.</td>
<td></td>
</tr>
<tr>
<td>ROLLOVER_IMAGE_NAME</td>
<td></td>
<td>VARCHAR2(350)</td>
<td>The unique name of the rollover image associated with the page.</td>
<td>Matches the NAME column in the document table.</td>
</tr>
</tbody>
</table>
Table F–25 lists the columns in the WWSBR_USER_PAGES view, which describes the pages on which the current user has Manage privileges or higher. This view is for use with the wwsbr_api.modify folder API and contains more columns than the WWSBR_USER_FOLDERS view.

If the WWSBR_USER_PAGES view does not exist in your portal schema, refer to the instructions on how to create it in Section 11.1.1, "Creating the WWSBR_USER_PAGES View".
<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUID</td>
<td>NOT NULL</td>
<td>RAW(32)</td>
<td>Globally unique ID of the page.</td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the page.</td>
<td></td>
</tr>
<tr>
<td>SITEID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID of the page.</td>
<td></td>
</tr>
<tr>
<td>LANGUAGE</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
<td>Language code.</td>
<td></td>
</tr>
<tr>
<td>PARENTID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the parent page.</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(60)</td>
<td>Name of the page. This name is used in path based URLs.</td>
<td>This name is used in path based URLs.</td>
</tr>
<tr>
<td>TITLE</td>
<td>NOT NULL</td>
<td>VARCHAR(256)</td>
<td>Display name, or title, of the page.</td>
<td></td>
</tr>
<tr>
<td>SETTINGSSETID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the style used by the page.</td>
<td></td>
</tr>
<tr>
<td>SETTINGSSETSITEID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID of the style used by the page.</td>
<td></td>
</tr>
<tr>
<td>ISPUBLIC</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates that the page is viewable by public users.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 - not public</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 - is public</td>
</tr>
<tr>
<td>IMAGE</td>
<td></td>
<td>VARCHAR2(350)</td>
<td>Unique document name of the image associated with the page, for example 6001.JPG.</td>
<td></td>
</tr>
<tr>
<td>ROLLOVERIMAGE</td>
<td></td>
<td>VARCHAR2(350)</td>
<td>Unique document name of the rollover image associated with the page, or the inactive tab image for the tab, for example 6001.JPG.</td>
<td></td>
</tr>
<tr>
<td>TITLEIMAGE</td>
<td></td>
<td>VARCHAR2(350)</td>
<td>Unique document name of the active tab image for the tab, for example 6001.JPG.</td>
<td></td>
</tr>
<tr>
<td>LEADER</td>
<td></td>
<td>VARCHAR2(256)</td>
<td>E-mail address of the page contact.</td>
<td></td>
</tr>
<tr>
<td>PRODUCTION</td>
<td></td>
<td>NUMBER(1)</td>
<td>Not used.</td>
<td></td>
</tr>
<tr>
<td>CREATEDATE</td>
<td></td>
<td>DATE</td>
<td>Date the page was created.</td>
<td></td>
</tr>
<tr>
<td>CREATOR</td>
<td></td>
<td>VARCHAR2(256)</td>
<td>User name of the person who created the page.</td>
<td></td>
</tr>
<tr>
<td>UPDATEDATE</td>
<td></td>
<td>DATE</td>
<td>Date the page was last updated.</td>
<td></td>
</tr>
<tr>
<td>UPDATOR</td>
<td></td>
<td>VARCHAR2(256)</td>
<td>User name of the person who last updated the page.</td>
<td></td>
</tr>
<tr>
<td>PUBLISHDATE</td>
<td></td>
<td>DATE</td>
<td>Not used.</td>
<td></td>
</tr>
<tr>
<td>ICON</td>
<td></td>
<td>VARCHAR2(256)</td>
<td>Not used.</td>
<td></td>
</tr>
<tr>
<td>CTXTXT</td>
<td></td>
<td>VARCHAR2(1)</td>
<td>Used internally by search.</td>
<td></td>
</tr>
<tr>
<td>Column</td>
<td>Null?</td>
<td>Data Type</td>
<td>Description</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------</td>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>HAVEITEMSECURITY</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates that item level security is enabled for the page.</td>
<td>Valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■  0 - ILS disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■  1 - ILS enabled</td>
</tr>
<tr>
<td>ITEMVERSIONING</td>
<td></td>
<td>VARCHAR2(30)</td>
<td>Indicates the level of item versioning for the page.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■  versionnone - no versioning</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■  versionsimple - simple versioning</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■  versionaudit - audit versioning</td>
</tr>
<tr>
<td>TOPICID</td>
<td></td>
<td>NUMBER</td>
<td>ID of the category assigned to the page.</td>
<td></td>
</tr>
<tr>
<td>TOPIC_SITEID</td>
<td></td>
<td>NUMBER</td>
<td>Page group ID of the category assigned to the page.</td>
<td></td>
</tr>
<tr>
<td>VALUE</td>
<td></td>
<td>VARCHAR2(2000)</td>
<td>For PL/SQL pages, the PL/SQL code.</td>
<td>For JSP pages, the JSP source document name. Do not change this value for JSP pages.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TYPE</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the page type for the page.</td>
<td></td>
</tr>
<tr>
<td>TYPE_SITEID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>Page group ID of the page type for the page.</td>
<td></td>
</tr>
<tr>
<td>BASE_TYPE</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the base page type.</td>
<td></td>
</tr>
<tr>
<td>IS_PORTLET</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates if the page is published as a portlet.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■  0 - not a portlet</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■  1 - is a portlet</td>
</tr>
<tr>
<td>QUOTA</td>
<td></td>
<td>VARCHAR2(5)</td>
<td>Not used.</td>
<td></td>
</tr>
<tr>
<td>SYSPRI_NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(200)</td>
<td>The root page from which the page inherits access settings.</td>
<td></td>
</tr>
<tr>
<td>PLSQL_EXECUTOR</td>
<td></td>
<td>VARCHAR2(30)</td>
<td>For PL/SQL type pages, the database schema used to execute the PL/SQL code.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■  $PUBLIC$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■  $CREATOR$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■  &lt;database user name&gt;</td>
</tr>
<tr>
<td>KEYWORDS</td>
<td></td>
<td>VARCHAR2(2000)</td>
<td>Keywords for the page.</td>
<td></td>
</tr>
<tr>
<td>IS_READY</td>
<td></td>
<td>NUMBER(1)</td>
<td>Indicates that page creation is complete.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■  1 - page creation is complete</td>
</tr>
</tbody>
</table>
### Table F–25 (Cont.)  WWSBR_USER_PAGES

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>INHERIT_PRIV</td>
<td></td>
<td>VARCHAR2(200)</td>
<td>The page from which this page inherits its privileges.</td>
<td>Use the following format: &lt;page group id&gt;/&lt;page id&gt;</td>
</tr>
<tr>
<td>CACHE_MODE</td>
<td></td>
<td>NUMBER(1)</td>
<td>Caching mode for the page.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ 2 - no caching</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ 1 - cache page definition only</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ 0 - cache page definition and content for x minutes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ 4 - cache page definition only at system level</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ 3 - cache page definition and content at system level for x minutes</td>
<td></td>
</tr>
<tr>
<td>CACHE_EXPIRES</td>
<td></td>
<td>NUMBER(38)</td>
<td>Cache period in minutes.</td>
<td></td>
</tr>
<tr>
<td>TEMPLATE_ID</td>
<td></td>
<td>NUMBER(38)</td>
<td>ID of the template on which the page is based.</td>
<td></td>
</tr>
<tr>
<td>TEMPLATE_SITEID</td>
<td></td>
<td>NUMBER(38)</td>
<td>Page group ID of the template on which the page is based.</td>
<td></td>
</tr>
<tr>
<td>ALLOW_PAGE_STYLE</td>
<td></td>
<td>NUMBER(1)</td>
<td>For templates, indicates if pages can use a different style.</td>
<td>Valid value is as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ 1 - allow pages to use different style.</td>
<td></td>
</tr>
<tr>
<td>ALLOW_PAGE_ACL</td>
<td></td>
<td>NUMBER(!)</td>
<td>For templates, indicates if pages have different access settings.</td>
<td>Valid value is as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ 1 - allow pages to have different access settings.</td>
<td></td>
</tr>
<tr>
<td>DAV_ID</td>
<td>NOT NULL</td>
<td>VARCHAR2(60)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAV_LOCK_TOKEN</td>
<td></td>
<td>VARCHAR2(36)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IS_TEMPLATE</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates that the page is a template.</td>
<td>Valid value is as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ 1 - is template</td>
<td></td>
</tr>
<tr>
<td>INIT_JSPFILE</td>
<td></td>
<td>VARCHAR2(256)</td>
<td>For JSP pages, initial JSP file if the JSP source of the page is a JAR or WAR file.</td>
<td></td>
</tr>
<tr>
<td>UI_TEMPLATE_ID</td>
<td></td>
<td>NUMBER(38)</td>
<td>ID of HTML page skin.</td>
<td></td>
</tr>
<tr>
<td>TEMPLATE_ISPUBLIC</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates if the template is ready to use.</td>
<td>Valid value is as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ 1 - template is ready to use</td>
<td></td>
</tr>
<tr>
<td>CONTAINER_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the container page.</td>
<td></td>
</tr>
<tr>
<td>DEFAULT_ITEM_REGION_ID</td>
<td>NOT NULL</td>
<td>NUMBER</td>
<td>ID of the default item region for the page.</td>
<td></td>
</tr>
</tbody>
</table>
Table F–25  (Cont.) WWSBR_USER_PAGES

<table>
<thead>
<tr>
<th>Column</th>
<th>Null?</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT_PORTLET_REGION_ID</td>
<td></td>
<td>NUMBER</td>
<td>ID of the default portlet region for the page.</td>
<td></td>
</tr>
<tr>
<td>ITEMTYPE_INHERIT_FLAGS</td>
<td></td>
<td>NUMBER(1)</td>
<td>For WebDAV, indicates if default item types are inherited from parent page.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 7 - inherit all item types from parent page</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 0 - specify all types on this page</td>
</tr>
<tr>
<td>REGFILE_ITEMTYPE</td>
<td></td>
<td>RAW(32)</td>
<td>For WebDAV, GUID of default item type for regular files.</td>
<td></td>
</tr>
<tr>
<td>ZIPFILE_ITEMTYPE</td>
<td></td>
<td>RAW(32)</td>
<td>For WebDAV, GUID of default item type for zip files.</td>
<td></td>
</tr>
<tr>
<td>IMAGEFILE_ITEMTYPE</td>
<td></td>
<td>RAW(32)</td>
<td>For WebDAV, GUID of default item type for image files.</td>
<td></td>
</tr>
<tr>
<td>DISPLAYINPARENT</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates if the page is displayed in its parent page’s sub page region.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 1 - display page in parent’s sub page region</td>
</tr>
<tr>
<td>SEQ</td>
<td></td>
<td>NUMBER</td>
<td>The sequence (order) of the page in its parent page’s sub page region.</td>
<td></td>
</tr>
<tr>
<td>ALPHABETICAL_SORT</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates that sub pages are displayed in alphabetical order.</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 1 - display sub pages in alphabetical order</td>
</tr>
<tr>
<td>IS_ITEM_PAGE</td>
<td></td>
<td>NUMBER(1)</td>
<td>Indicates that the template is for items (with item placeholder item)</td>
<td>Valid values are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 1 - is item template</td>
</tr>
<tr>
<td>ITEM_PAGE_ID</td>
<td></td>
<td>NUMBER(38)</td>
<td>ID of the item template.</td>
<td></td>
</tr>
<tr>
<td>ITEM_PAGE_SITE_ID</td>
<td></td>
<td>NUMBER(28)</td>
<td>Page group ID of the item template.</td>
<td></td>
</tr>
<tr>
<td>ITEM_PAGE_TABSTRING</td>
<td></td>
<td>VARCHAR2(512)</td>
<td>Tab strings of the item template.</td>
<td>Use the following format:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&lt;tab name&gt;:&lt;sub tab name&gt;:....:&lt;sub tab name&gt;</td>
</tr>
<tr>
<td>INHERIT_ITEM_PAGE</td>
<td></td>
<td>NUMBER(1)</td>
<td>For item template only, indicates that items inherit the parent page’s item template.</td>
<td>Valid value is as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 1 - inherit parent page’s item template</td>
</tr>
<tr>
<td>ALLOW_ITEM_PAGE_OVERRIDE</td>
<td></td>
<td>NUMBER(1)</td>
<td>For item template only, indicates that items can have their own item template.</td>
<td>Valid value is as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ 1 - allow items on the page to have their own item template</td>
</tr>
<tr>
<td>Column</td>
<td>Null?</td>
<td>Data Type</td>
<td>Description</td>
<td>Notes</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------</td>
<td>-----------</td>
<td>------------------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>HAS_INPLACE_ITEM</td>
<td>NOT NULL</td>
<td>NUMBER(1)</td>
<td>Indicates if the page or tab has a placeholder item.</td>
<td>Valid value is as follows:  1 - has a placeholder item</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>NUMBER</td>
<td></td>
<td>Limit time, in seconds, used to fetch portlets.</td>
<td></td>
</tr>
<tr>
<td>LAST_ITEM_TEMPLATE_ASSIGNMENT</td>
<td>DATE</td>
<td></td>
<td>Set by trigger.</td>
<td></td>
</tr>
</tbody>
</table>
This appendix lists and describes the OracleAS Portal actions and the CMEF events that they generate. It contains the following sections:

- Actions and Events for Items
- Actions and Events for Pages
- Actions and Events for Tabs
- Actions and Events for Page Groups
- Actions and Events for Attributes
- Actions and Events for Item Types
- Actions and Events for Page Types
- Actions and Events for Categories
- Actions and Events for Perspectives
- Actions and Events for Templates

G.1 Actions and Events for Items

Table G–1 lists the actions performed when adding and updating items and the events raised by those actions.

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>An item is added and published immediately</td>
<td>ADD_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is added but set to be published some time in the future</td>
<td>ADD_ITEM</td>
<td>INSERT</td>
<td>NOT_PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>A portlet instance is added</td>
<td>ADD_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is updated but there is no change to the state of the item (that is, it stays as published or unpublished)</td>
<td>EDIT_ITEM</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>ITEM</td>
</tr>
</tbody>
</table>
Table G–2 lists the actions performed when copying and moving items and the events raised by those actions.

### Table G–2  Copying and Moving Items

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>An item is copied - the copied item is in the published state</td>
<td>COPY_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is copied - the copied item is in the unpublished state</td>
<td>COPY_ITEM</td>
<td>INSERT</td>
<td>NOT_PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is moved (within a page or to a different page or page group) - the item is in the published state</td>
<td>MOVE_ITEM</td>
<td>DELETE</td>
<td>PURGED</td>
<td>ITEM</td>
</tr>
<tr>
<td>--</td>
<td>MOVE_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is moved (within a page or to a different page or page group) - the item is in the unpublished state</td>
<td>MOVE_ITEM</td>
<td>DELETE</td>
<td>PURGED</td>
<td>ITEM</td>
</tr>
<tr>
<td>--</td>
<td>MOVE_ITEM</td>
<td>INSERT</td>
<td>NOT_PUBLISHED</td>
<td>ITEM</td>
</tr>
</tbody>
</table>

Table G–3 lists the actions performed when deleting and restoring items and the events raised by those actions.

### Table G–3  Copying and Moving Items

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>An item is hidden</td>
<td>HIDE_ITEM</td>
<td>UPDATE</td>
<td>UNPUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is shown</td>
<td>SHOW_ITEM</td>
<td>UPDATE</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is expired</td>
<td>EXPIRE_ITEM</td>
<td>UPDATE</td>
<td>UNPUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is unexpired</td>
<td>UNEXPIRE_ITEM</td>
<td>UPDATE</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is checked-out</td>
<td>CHECK_OUT_ITEM</td>
<td>UPDATE</td>
<td>CHECKED_OUT</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is checked-in</td>
<td>CHECK_IN_ITEM</td>
<td>UPDATE</td>
<td>CHECKED_IN</td>
<td>ITEM</td>
</tr>
</tbody>
</table>
### Table G–3 Deleting and Restoring Items

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>An item is deleted in a page group that is set to retain deleted items</td>
<td>DELETE_ITEM</td>
<td>DELETE</td>
<td>MARKED_FOR_DELETE</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is deleted in a page group that is not set to retain deleted items</td>
<td>DELETE_ITEM</td>
<td>DELETE</td>
<td>PURGED</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item in the unpublished state is undeleted (item is either hidden or expired or is set to be published in the future)</td>
<td>UNDELETE_ITEM</td>
<td>UPDATE</td>
<td>NOT_PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item in the published state is undeleted</td>
<td>UNDELETE_ITEM</td>
<td>UPDATE</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
</tbody>
</table>

Table G–4 lists the actions performed when defining privileges for items and the events raised by those actions.

### Table G–4 Defining Item Privileges

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify item level access privileges</td>
<td>SPECIFY_ITEM_ACL</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>ITEM</td>
</tr>
<tr>
<td>Add user to item ACL</td>
<td>ADD_ITEM_ACL</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>ITEM</td>
</tr>
<tr>
<td>User’s privilege is updated on an item</td>
<td>UPDATE_ITEM_ACL</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>ITEM</td>
</tr>
<tr>
<td>Delete user from item ACL</td>
<td>DELETE_ITEM_ACL</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>ITEM</td>
</tr>
<tr>
<td>Choose to inherit security from parent page</td>
<td>INHERIT_ITEM_ACL</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>ITEM</td>
</tr>
</tbody>
</table>

Table G–5 lists the actions performed when working with item versions and the events raised by those actions.

### Table G–5 Item Versioning

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A new version of an item is added and is set to be the current version</td>
<td>ADD_ITEM_VERSION</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>A new version of an item is added but is not set to be the current version</td>
<td>ADD_ITEM_VERSION</td>
<td>INSERT</td>
<td>NOT_PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>A version of an item is deleted in a page group that is set to retain deleted items</td>
<td>DELETE_ITEM_VERSION</td>
<td>DELETE</td>
<td>MARKED_FOR_DELETE</td>
<td>ITEM</td>
</tr>
<tr>
<td>A version of an item is deleted in a site that is not set to retain deleted items</td>
<td>DELETE_ITEM_VERSION</td>
<td>DELETE</td>
<td>PURGED</td>
<td>ITEM</td>
</tr>
</tbody>
</table>
Table G–6 lists the actions performed when working with item approvals and the events raised by those actions.

### Table G–5 (Cont.) Item Versioning

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>The current version of an item is changed (the previous current version and the new current version are in the published state)</td>
<td>SWITCH_ITEM_VERSION</td>
<td>UPDATE</td>
<td>UNPUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td></td>
<td>--</td>
<td>UPDATE</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>The current version of an item is changed (the previous current version and the new current version are in the unpublished state)</td>
<td>SWITCH_ITEM_VERSION</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>ITEM</td>
</tr>
<tr>
<td></td>
<td>--</td>
<td>UPDATE</td>
<td>NOT_PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>The current version of an item is changed (the previous current version was in the published state and the new current version is in the unpublished state)</td>
<td>SWITCH_ITEM_VERSION</td>
<td>UPDATE</td>
<td>UNPUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td></td>
<td>--</td>
<td>UPDATE</td>
<td>NOT_PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>The current version of an item is changed (the previous current version was in the unpublished state and the new current version is in the published state)</td>
<td>SWITCH_ITEM_VERSION</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>ITEM</td>
</tr>
<tr>
<td></td>
<td>--</td>
<td>UPDATE</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
</tbody>
</table>

### Table G–6 Item Approvals

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>An item is added in pending mode</td>
<td>SUBMIT_ITEM_FOR_APPROVAL</td>
<td>INSERT</td>
<td>NOT_PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is added in draft mode</td>
<td>ADD_DRAFT_ITEM</td>
<td>INSERT</td>
<td>NOT_PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>A pending item is edited by an approver</td>
<td>EDIT_ITEM_BY_APPROVER</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is approved but is pending approval from the next approver</td>
<td>APPROVE_ITEM_STEP</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is approved by all approvers (the approved item overwrites the current, published version)</td>
<td>APPROVE_ITEM</td>
<td>UPDATE</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
</tbody>
</table>
Table G–7 lists the actions performed when adding and editing pages and the events raised by those actions.

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>An item is approved by all approvers (the approved item becomes the new current version)</td>
<td>APPROVE_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is approved by all approvers (the approved item becomes a new, but not current version)</td>
<td>APPROVE_ITEM</td>
<td>UPDATE</td>
<td>UNPUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is updated and submitted for approval</td>
<td>SUBMIT_ITEMFOR_APPROVAL</td>
<td>UPDATE</td>
<td>NOT_PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is rejected</td>
<td>REJECT_ITEM</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>ITEM</td>
</tr>
<tr>
<td>A rejected item is resubmitted for approval</td>
<td>SUBMIT_ITEMFOR_APPROVAL</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>ITEM</td>
</tr>
<tr>
<td>A draft item is updated (attributes such as author, description, and so on are updated)</td>
<td>EDIT_DRAFT_ITEM</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>ITEM</td>
</tr>
<tr>
<td>A draft item is updated (the publish date is updated to publish the item some time in the future)</td>
<td>EDIT_DRAFT_ITEM</td>
<td>UPDATE</td>
<td>UNPUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>A draft item is updated (the publish date is updated to publish the item immediately)</td>
<td>EDIT_DRAFT_ITEM</td>
<td>UPDATE</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>A draft item is submitted for approval</td>
<td>SUBMIT_ITEMFOR_APPROVAL</td>
<td>UPDATE</td>
<td>NOT_PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is copied into a page where draft items are enabled</td>
<td>COPY_DRAFT_ITEM</td>
<td>INSERT</td>
<td>NOT_PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is copied to a page and is submitted for approval</td>
<td>SUBMIT_ITEMFOR_APPROVAL</td>
<td>INSERT</td>
<td>NOT_PUBLISHED</td>
<td>ITEM</td>
</tr>
</tbody>
</table>

G.2 Actions and Events for Pages

Table G–7 lists the actions performed when adding and editing pages and the events raised by those actions.

Table G–7 Adding and Editing Pages

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A page is created - there are two messages: one for the page that is added and one for the portlet instance for the default navigation page</td>
<td>ADD_PAGE</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>PAGE</td>
</tr>
<tr>
<td>A page based on a template is created. The template contains one item and one portlet</td>
<td>ADD_PAGE</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td></td>
<td>ADD_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
</tbody>
</table>

Table G–6 (Cont.) Item Approvals
Table G–8 lists the actions performed when deleting pages and the events raised by those actions.

Table G–8   Deleting Pages

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A page is deleted (if the page has sub-pages, a message is logged for each deleted sub-page, however no messages are logged for deleted items)</td>
<td>DELETE_PAGE</td>
<td>DELETE</td>
<td>PURGED</td>
<td>PAGE</td>
</tr>
</tbody>
</table>

Table G–9 lists the actions performed when defining privileges for pages and the events raised by those actions.

Table G–9 (Cont.) Adding and Editing Pages

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A page is renamed. If the root page is renamed there are two messages: one for the page and one for the page group.</td>
<td>RENAME_PAGE</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
<tr>
<td>A page property is updated (for example the description)</td>
<td>EDIT_PAGE</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
<tr>
<td>A page is copied (the page contains one portlet instance and one item)</td>
<td>ADD_PAGE</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>PAGE</td>
</tr>
<tr>
<td>A page is moved under a different parent</td>
<td>MOVE_PAGE</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>PAGE</td>
</tr>
<tr>
<td>A page is translated</td>
<td>EDIT_PAGE</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
<tr>
<td>Versioning is enabled or disabled for a page</td>
<td>EDIT_PAGE</td>
<td>INSERT</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
</tbody>
</table>

1 As the page designer steps through the wizard, a page is created that includes the default navigation page. However, as soon as it is determined that the page is based on a template, this page is deleted and a new page (based on the template) is created instead.

2 The first update is for the default language, the last is for the current translation. The message of interest is the insert for the current translation.
Table G–9  Defining Page Privileges

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILS is enabled or disabled on a page</td>
<td>UPDATE_PAGE_ACL</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
<tr>
<td>--</td>
<td>UPDATE_PAGE_</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ILS</td>
<td>ILS</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
<tr>
<td>A page is displayed to public users</td>
<td>UPDATE_PAGE_</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
<tr>
<td>--</td>
<td>PUBLIC_SETTING</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A user is added to a page ACL</td>
<td>ADD_PAGE_ACL</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
<tr>
<td>A user is deleted from a page ACL</td>
<td>DELETE_PAGE_</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
<tr>
<td>A user’s privilege is changed in a page ACL</td>
<td>UPDATE_PAGE_</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
<tr>
<td>A page is set to inherit its ACL from its parent</td>
<td>INHERIT_PAGE_</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
<tr>
<td>An ACL is specified at the page level (the page does not inherit its ACL from its parent)</td>
<td>UPDATE_PAGE_</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
<tr>
<td>--</td>
<td>SPECIFY_PAGE_</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
</tbody>
</table>

G.3 Actions and Events for Tabs

Table G–10 lists the actions performed when adding and editing tabs and the events raised by those actions.

Table G–10  Adding and Editing Tabs

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A tab is added</td>
<td>ADD_PAGE</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>PAGE</td>
</tr>
<tr>
<td>--</td>
<td>ADD_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>A tab is added in a region that contains two items</td>
<td>ADD_PAGE</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>PAGE</td>
</tr>
<tr>
<td>--</td>
<td>ADD_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>A tab is hidden</td>
<td>HIDE_ITEM</td>
<td>UPDATE</td>
<td>UNPUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>A hidden tab is shown</td>
<td>SHOW_ITEM</td>
<td>UPDATE</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>A tab is deleted</td>
<td>DELETE_PAGE</td>
<td>DELETE</td>
<td>PURGED</td>
<td>PAGE</td>
</tr>
<tr>
<td>--</td>
<td>DELETE_ITEM</td>
<td>DELETE</td>
<td>PURGED</td>
<td>ITEM</td>
</tr>
<tr>
<td>A tab is renamed</td>
<td>RENAME_PAGE</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
<tr>
<td>A tab property is updated (for example, the display name)</td>
<td>EDIT_PAGE</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
</tbody>
</table>
### G.4 Actions and Events for Page Groups

Table G–11 lists the actions performed when adding and editing page groups and the events raised by those actions.

**Table G–11 Adding and Editing Page Groups**

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A page group is added</td>
<td>ADD_PAGE_GROUP</td>
<td>INSERT</td>
<td>GENERAL</td>
<td>PAGE_GROUP</td>
</tr>
<tr>
<td>A page group is renamed</td>
<td>RENAME_PAGE_GROUP</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE_GROUP</td>
</tr>
</tbody>
</table>

Table G–12 lists the actions performed when deleting and purging page groups and the events raised by those actions.

**Table G–12 Deleting and Purging Page Groups**

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A page group is deleted. Messages are not logged for any items or sub-pages.</td>
<td>DELETE_PAGE_GROUP</td>
<td>DELETE</td>
<td>PURGED</td>
<td>PAGE_GROUP</td>
</tr>
<tr>
<td>Deleted items are purged from a page group (one message for each item and one message for the actual purge operation)</td>
<td>DELETE_ITEM</td>
<td>DELETE</td>
<td>PURGED</td>
<td>ITEM</td>
</tr>
<tr>
<td>--</td>
<td>PURGE_DELETED_ITEM</td>
<td>DELETE</td>
<td>PURGED</td>
<td>PAGE_GROUP</td>
</tr>
<tr>
<td>Expired items are purged from a page group (one message for each item and one message for the actual purge operation)</td>
<td>DELETE_ITEM</td>
<td>DELETE</td>
<td>PURGED</td>
<td>ITEM</td>
</tr>
<tr>
<td>--</td>
<td>DELETE_ITEM</td>
<td>DELETE</td>
<td>PURGED</td>
<td>ITEM</td>
</tr>
<tr>
<td>Choose to not retain deleted items at the page group level, thus purging all existing retained deleted items (one message for each item and one message for the actual purge operation)</td>
<td>DELETE_ITEM</td>
<td>DELETE</td>
<td>PURGED</td>
<td>ITEM</td>
</tr>
<tr>
<td>--</td>
<td>PURGE_DELETED_ITEM</td>
<td>DELETE</td>
<td>PURGED</td>
<td>PAGE_GROUP</td>
</tr>
</tbody>
</table>

Table G–13 lists the actions performed when defining privileges for page groups and the events raised by those actions.

---

1 Items on the original page are moved to the tab. So the item is first deleted from the original page and then added to the new tab.
### Table G–13  Defining Page Group Privileges

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A user is added to a page group ACL</td>
<td>ADD_PAGE_GROUP_ACL</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE_GROUP</td>
</tr>
<tr>
<td>An user’s privilege is changed in a page group ACL</td>
<td>UPDATE_PAGE_GROUP_ACL</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE_GROUP</td>
</tr>
<tr>
<td>A user is deleted from a page group ACL</td>
<td>DELETE_PAGE_GROUP_ACL</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE_GROUP</td>
</tr>
</tbody>
</table>

Table G–14 lists the actions performed when installing translations for page groups and the events raised by those actions.

### Table G–14  Page Group Translations

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A translation is installed</td>
<td>TRANSLATE_PAGE</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>PAGE</td>
</tr>
</tbody>
</table>

### G.5 Actions and Events for Attributes

Table G–15 lists the actions performed when working with attributes and the events raised by those actions.

### Table G–15  Attributes

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>An attribute is added. The data type of the attribute is stored in the STATE field, for example, DATA_TYPE_TEXT.</td>
<td>ADD_ATTRIBUTE</td>
<td>INSERT</td>
<td>DATA_TYPE_TEXT</td>
<td>ATTRIBUTE</td>
</tr>
<tr>
<td>An attribute is updated</td>
<td>EDIT_ATTRIBUTE</td>
<td>UPDATE</td>
<td>DATA_TYPE_TEXT</td>
<td>ATTRIBUTE</td>
</tr>
<tr>
<td>An attribute is promoted (the message is the same regardless of how many items use the attribute)</td>
<td>MOVE_ATTRIBUTE</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>ATTRIBUTE</td>
</tr>
<tr>
<td>An attribute is deleted (the message is the same even if there are items using the attribute)</td>
<td>DELETE_ATTRIBUTE</td>
<td>DELETE</td>
<td>DATA_TYPE_TEXT</td>
<td>ATTRIBUTE</td>
</tr>
</tbody>
</table>

### G.6 Actions and Events for Item Types

Table G–16 lists the actions performed when working with item types and the events raised by those actions.
### Table G–16  Item Types

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A custom item type is created</td>
<td>ADD_ITEM_TYPE</td>
<td>INSERT</td>
<td>GENERAL</td>
<td>ITEM_TYPE</td>
</tr>
<tr>
<td>An item type is edited and changes are made in the Main tab (if you edit the name, a different message is generated, see following description)</td>
<td>EDIT_ITEM_TYPE</td>
<td>UPDATE</td>
<td>MAIN_TAB</td>
<td>ITEM_TYPE</td>
</tr>
<tr>
<td>An item type is renamed</td>
<td>RENAME_ITEM_TYPE</td>
<td>UPDATE</td>
<td>MAIN_TAB</td>
<td>ITEM_TYPE</td>
</tr>
<tr>
<td>An attribute is attached to an item type</td>
<td>EDIT_ITEM_TYPE</td>
<td>UPDATE</td>
<td>ATTRIBUTES_TAB</td>
<td>ITEM_TYPE</td>
</tr>
<tr>
<td>An attribute is detached from an item type</td>
<td>DETACH_ATTRIBUTE</td>
<td>UPDATE</td>
<td>ATTRIBUTES_TAB</td>
<td>ITEM_TYPE</td>
</tr>
<tr>
<td>Attribute default values are updated</td>
<td>EDIT_ITEM_TYPE</td>
<td>UPDATE</td>
<td>ATTRIBUTES_TAB</td>
<td>ITEM_TYPE</td>
</tr>
<tr>
<td>A procedure is attached to an item type</td>
<td>EDIT_ITEM_TYPE</td>
<td>UPDATE</td>
<td>PROCEDURES_TAB</td>
<td>ITEM_TYPE</td>
</tr>
<tr>
<td>A procedure attached to an item type is edited and new parameters are added</td>
<td>EDIT_ITEM_TYPE</td>
<td>UPDATE</td>
<td>PROCEDURES_TAB</td>
<td>ITEM_TYPE</td>
</tr>
<tr>
<td>An item type is promoted</td>
<td>MOVE_ITEM_TYPE</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>ITEM_TYPE</td>
</tr>
<tr>
<td>An item type is deleted (the message is the same even if there are items based on the item type)</td>
<td>DELETE_ITEM_TYPE</td>
<td>DELETE</td>
<td>PURGED</td>
<td>ITEM_TYPE</td>
</tr>
</tbody>
</table>

### G.7 Actions and Events for Page Types

Table G–17 lists the actions performed when working with page types and the events raised by those actions.

### Table G–17  Page Types

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A custom page type is created</td>
<td>ADD_PAGE_TYPE</td>
<td>INSERT</td>
<td>GENERAL</td>
<td>PAGE_TYPE</td>
</tr>
<tr>
<td>A page type is edited and changes are made in the Main tab (if you edit the name, a different message is generated, see following description)</td>
<td>EDIT_PAGE_TYPE</td>
<td>UPDATE</td>
<td>MAIN_TAB</td>
<td>PAGE_TYPE</td>
</tr>
<tr>
<td>A page type is renamed</td>
<td>RENAME_PAGE_TYPE</td>
<td>UPDATE</td>
<td>MAIN_TAB</td>
<td>PAGE_TYPE</td>
</tr>
<tr>
<td>An attribute is attached to a page type</td>
<td>EDIT_PAGE_TYPE</td>
<td>UPDATE</td>
<td>ATTRIBUTES_TAB</td>
<td>PAGE_TYPE</td>
</tr>
<tr>
<td>Attribute default values are added</td>
<td>EDIT_PAGE_TYPE</td>
<td>UPDATE</td>
<td>ATTRIBUTES_TAB</td>
<td>PAGE_TYPE</td>
</tr>
<tr>
<td>An attribute is detached from a page type</td>
<td>DETACH_ATTRIBUTE</td>
<td>UPDATE</td>
<td>ATTRIBUTES_TAB</td>
<td>PAGE_TYPE</td>
</tr>
<tr>
<td>A procedure is attached to a page type</td>
<td>EDIT_PAGE_TYPE</td>
<td>UPDATE</td>
<td>PROCEDURES_TAB</td>
<td>PAGE_TYPE</td>
</tr>
</tbody>
</table>
### G.8 Actions and Events for Categories

Table G–18 lists the actions performed when working with categories and the events raised by those actions.

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A category is added</td>
<td>ADD_CATEGORY</td>
<td>INSERT</td>
<td>GENERAL</td>
<td>CATEGORY</td>
</tr>
<tr>
<td>A category is renamed</td>
<td>RENAME_CATEGORY</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>CATEGORY</td>
</tr>
<tr>
<td>--</td>
<td>RENAME_PAGE</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
<tr>
<td>A category is edited and changes are made in the Main tab</td>
<td>EDIT_CATEGORY</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>CATEGORY</td>
</tr>
<tr>
<td>An image is attached to a category</td>
<td>EDIT_CATEGORY_IMAGE</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>CATEGORY</td>
</tr>
<tr>
<td>An image is detached from a category</td>
<td>DELETE_CATEGORY_IMAGE</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>CATEGORY</td>
</tr>
<tr>
<td>A category is promoted</td>
<td>MOVE_CATEGORY</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>CATEGORY</td>
</tr>
<tr>
<td>A category is deleted</td>
<td>DELETE_CATEGORY</td>
<td>DELETE</td>
<td>PURGED</td>
<td>PAGE_TYPE</td>
</tr>
</tbody>
</table>

### G.9 Actions and Events for Perspectives

Table G–19 lists the actions performed when working with perspectives and the events raised by those actions.

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A perspective is added</td>
<td>ADD_PERSPECTIVE</td>
<td>INSERT</td>
<td>GENERAL</td>
<td>PERSPECTIVE</td>
</tr>
<tr>
<td>A perspective is renamed</td>
<td>RENAME_PERSPECTIVE</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PERSPECTIVE</td>
</tr>
<tr>
<td>--</td>
<td>RENAME_PAGE</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
<tr>
<td>A perspective is edited and changes are made in the Main tab</td>
<td>EDIT_PERSPECTIVE</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PERSPECTIVE</td>
</tr>
</tbody>
</table>
Table G–19  (Cont.) Perspectives

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>An image is attached to a perspective</td>
<td>EDIT_PERSPECTIVE_IMAGE</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PERSPECTIVE</td>
</tr>
<tr>
<td>An image is detached from a perspective</td>
<td>DELETE_PERSPECTIVE_IMAGE</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PERSPECTIVE</td>
</tr>
<tr>
<td>A perspective is promoted</td>
<td>MOVE_PERSPECTIVE</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PERSPECTIVE</td>
</tr>
<tr>
<td>A perspective is deleted</td>
<td>DELETE_PERSPECTIVE</td>
<td>DELETE</td>
<td>GENERAL</td>
<td>PERSPECTIVE</td>
</tr>
</tbody>
</table>

G.10 Actions and Events for Templates

Table G–20 lists the actions performed when working with templates and the events raised by those actions.

Table G–20  Templates

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A page based on a template is created¹</td>
<td>ADD_PAGE</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>PAGE</td>
</tr>
<tr>
<td>--</td>
<td>ADD_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>--</td>
<td>DELETE_PAGE</td>
<td>DELETE</td>
<td>PURGED</td>
<td>PAGE</td>
</tr>
<tr>
<td>--</td>
<td>ADD_PAGE</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>PAGE</td>
</tr>
<tr>
<td>--</td>
<td>ADD_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>--</td>
<td>ADD_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>A page is edited to be based on a template, resulting in content from one region being deleted</td>
<td>DELETE_ITEM</td>
<td>DELETE</td>
<td>PURGED</td>
<td>ITEM</td>
</tr>
<tr>
<td>--</td>
<td>EDIT_PAGE</td>
<td>UPDATE</td>
<td>GENERAL</td>
<td>PAGE</td>
</tr>
<tr>
<td>--</td>
<td>ADD_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>--</td>
<td>ADD_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is added to a template. There is a message for the item on the template itself and then additional messages for each time the item is replicated on pages based on the template</td>
<td>ADD_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>An item is deleted from a template. There is a message for the item on the template itself and then additional messages (with different page IDs and page group IDs) for each time the item is deleted from pages based on the template.</td>
<td>DELETE_ITEM</td>
<td>DELETE</td>
<td>PURGED</td>
<td>ITEM</td>
</tr>
</tbody>
</table>
**Table G–20 (Cont.) Templates**

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>State</th>
<th>Object Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A page is detached from a template. One or more messages, as shown, is</td>
<td>COPY_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>produced for each item and portlet that was on the template</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A template, on which one or more pages are based, is deleted.</td>
<td>DELETE_PAGE</td>
<td>DELETE</td>
<td>PURGED</td>
<td>PAGE</td>
</tr>
<tr>
<td>--</td>
<td>COPY_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>--</td>
<td>COPY_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
<tr>
<td>--</td>
<td>DELETE_ITEM</td>
<td>DELETE</td>
<td>PURGED</td>
<td>ITEM</td>
</tr>
<tr>
<td>--</td>
<td>COPY_ITEM</td>
<td>INSERT</td>
<td>PUBLISHED</td>
<td>ITEM</td>
</tr>
</tbody>
</table>

1 The first two inserts are for the page creation. The next delete is because that page gets deleted when you attach the template. The next insert is for the new page based on the template. The following two item inserts are for item replication from the template. There will be an insert event for every item and portlet on the template.

2 The delete event is for the deletion of the actual template. The first two inserts are when the replicated items on a page based on the template are copied over. There will be an insert message for each item on the template. The following delete and insert events are for a non-replicated item on the page based on the template. There will be a delete and corresponding insert event for every non-replicated item on the page.
Glossary

About mode
An optional portlet Show mode that displays information about the portlet's copyright, version, and author.

access control list
See ACL.

ACL
Access Control List. A list of groups and users authorized for specific access to an object.

action
In the context of CMEF, an action occurs when something happens in a portal (either through the OracleAS Portal user interface, WebDAV, or PL/SQL APIs). Actions within a portal trigger CMEF events, which cause OracleAS Portal to write messages to the WWSBR_EVENT_Q queue. A subscriber can consume these messages and perform its own actions depending on what those messages are.

add-in
See extension.

advanced search
A search engine that enables users to:
- Find content that contains any or all terms in the search string.
- Search selected page groups, or search across all page groups.
- Restrict a search to a particular page, category, perspective, item type, or attribute.

If Oracle Text is installed and enabled, all text attributes and the actual content of documents and URLs are searched. If Oracle Text is not installed, or is disabled, the following metadata is searched: item attributes (Display Name, Description, Keywords, and Author), page attributes (Display Name, Description, and Keywords) and category/perspective attributes (Display Name and Description).

If Oracle Text is installed and enabled, you can also use advanced search to perform near, soundex, and fuzzy searches.

See also search portlet. Contrast with basic search and custom search.
API
Application Programming Interface. A set of exposed data structures and functions that an application can use to invoke services on a portal object, such as a portlet, page, or page group. Note that OracleAS Portal APIs are exposed through the PDK available from the Oracle Portal Developer Kit (PDK) page on OTN:

application
INTERNAL: Obsolete terminology. DO NOT USE to refer to a provider of portlets built using the Portlet Builder (for example, forms, reports, charts). Use database provider instead.

Application Programming Interface
See API.

Application Service Provider
See ASP.

approval notification
A message in the Notification portlet indicating that a user whose content requires approval has created or updated an item. The intended recipients of approval notifications (approvers) relating to a particular page group or page are identified in an approval process. An approver may respond to the notification by approving or rejecting the item in question.

approval process
A series of one or more steps that determines which users (or groups) need to review content that requires approval before it can be published. Each step in an approval process must have one or more approvers. Routing to the approvers can be in serial (one at a time) or in parallel (all at once), and each step can be defined to require a response (either an approval or a rejection) by any one member or by all members. Once the required number of responses is received during a step, the process continues to the next step. The process ends when the item is rejected, or the final step is reached and the document is approved.

approver
A user identified in a page group or page approval process, either explicitly or implicitly through group membership, as someone who needs to review content that requires approval before it can be published. An approver may choose to approve or reject such content.

ASP
Application Service Provider. A service that provides remote hosting of applications, maintaining and operating the hardware, software, and other resources required to run the applications. A good example is Oracle Portal Online (http://portal.oracle.com), a hosted subscription service that provides the features of OracleAS Portal to smaller businesses and organizations who want to build portals but do not have the resources in-house to build and manage them.

attribute
A portal object that stores information (or metadata) about an item or page: for example, Create Date, Expire Date, or Author. A page group administrator can create custom attributes to extend the functionality of item types and page types. For
example, a base attribute on a file is Display Name; a custom attribute might be a check box to indicate whether the file is confidential. Custom attributes are useful for assigning unique, searchable identifiers to items.

**authenticated user**
A user who is logged on to a portal. By default, authenticated users can access and, based on privileges granted to the user, act on certain portal objects, such as pages.

Contrast with public users, who can access public content only.

**authorization**
The evaluation of security constraints to send a message or make a request. Authorization uses specific criteria—authentication and restriction—to determine whether the request should be permitted.

**authorized user**
See authenticated user.

**banner**
See region banner. See also navigation page.

**base attribute**
See attribute.

**base item type**
See item type.

**base page type**
See page type.

**basic search**
A search engine that enables users to find content that contains a specific search string.

If Oracle Text is installed and enabled, all text attributes and the actual content of documents and URLs are searched. If Oracle Text is not installed, or is disabled, the following metadata is searched: item attributes (Display Name, Description, Keywords, and Author), page attributes (Display Name, Description, and Keywords) and category/perspective attributes (Display Name and Description).

See also search portlet. Contrast with advanced search and custom search.

**basic search box item**
A navigation item that users can add to a page to enable other users to perform basic searches. The search box can initiate a search of all page groups or a specified subset of page groups.

**batch job**
The process of running a Portlet Builder portlet in the background using the OracleAS Portal batch job facility. An end user can run a portlet in batch mode by selecting options on the portlet’s personalization form. Batch processing is useful if the portlet is based on a large amount of data, if the portlet displays many rows of data, or if the job may take a long time to run.
bind variable
A variable in a SQL statement that must be replaced with a valid value or address of a value in order for the statement to execute successfully. Portlet developers typically use bind variables (for example, dept) to display a parameter entry field in a portlet's personalization form. The entry field enables end users to choose the data that the portlet will display.

bookmark
See favorite.

breadcrumbs
See page path item.

Builder page
See Portal Builder page.

bulk load
See zip file item.

caching
The act of storing frequently accessed information, typically Web pages or portlets in OracleAS Portal, in a location where it can be accessed quickly to avoid frequent content generation. For example, Oracle Application Server Web Cache stores dynamically-generated portlets in its memory, then serves them to the PPE when there is a request for the specified portlet. This storage reduces the total time spent handling the request by avoiding connections to the back-end database and other Web sites.

See also expiry-based caching, invalidation-based caching, system level caching, user level caching, and validation-based caching.

calendar
A portlet created with the Portlet Builder that displays the results of a SQL query in calendar format.

call interface
The call interface displays the arguments that were selected when a Portlet Builder portlet was originally created or last edited.

category
A predefined attribute used to group or classify pages, items, and portlets in a page group. A category helps users answer the question: What is this item or page? For example, in a travel page group, you might have categories for maps, snapshots, and hotel reviews. Users can assign only one category to a particular item or page.

See also perspective.

chart
A portlet created with the Portlet Builder that displays the results of a SQL query as a chart, such as a bar chart, pie chart, or line chart. Bar charts are based on at least two table or view columns: one that identifies the bars on the chart and another that calculates the size of the bars on the chart.
check out/check in
A mechanism that allows a user to lock an item, by checking it out, so that other users cannot edit that same item. This prevents users from overwriting each others changes. After editing the item, the user releases it by checking it back in, making it available again for other users to edit.

child object
An object that is part of a hierarchy. For example, sub-pages, sub-categories, and sub-perspectives are child objects of a page, category, and perspective respectively.
See also manifest.

child page
In Oracle Instant Portal, a page created beneath a top-level page. Child pages are listed along the left side of the top-level page to which they belong, in the navigation area, and usually support the top-level page's main theme. You can add, delete, re-position, or edit child pages right in the navigation area, assuming you have the appropriate privileges.

CHTML
Compact HTML. A subset of HTML recommendations, designed for small devices.

classification
Categories and perspectives are used to classify the content of a page or item so that it is easy for users to locate that content during a search.
See also category and perspective.

CLOB
Character Large Object. A Large Object (LOB) datatype whose value is composed of character data corresponding to the database character set. A CLOB can be indexed and searched by the Oracle Text search engine.

cluster
A database object used to store tables that are related to one another and that are often joined together in the same area on a disk.

CMEF
Content Management Event Framework. A feature that enables you to extend OracleAS Portal's content management event functionality by adding programmatic hooks to certain pre-defined portal events. The framework publishes these events to an Oracle Streams Advanced Queuing (AQ) queue. This allows third party programs to subscribe to these events and to use the APIs to extend the functionality of the portal. CMEF should not be confused with the portlet events feature.

CMEF event
An action within a portal triggers a corresponding CMEF event. A developer can write code that responds to CMEF events.

community
See Portal Community.

compact HTML
See CHTML.
content area
In Oracle Instant Portal, refers to the area in which a page's items (or content) appear, on the right side of the screen.

content contributor
A user who has the appropriate privileges to add items to a page. Appropriate page privileges include Manage Content and Manage Items With Approval. Appropriate item privileges include Manage, Edit, and View.

content item type
A means of classifying the actual content of an item that is being uploaded to a page, such as a document, text, or an image.

Built-in content item types are:
- file item and simple file item
- text item and simple text item
- URL item and simple URL item
- image item and simple image item
- image map item
- PL/SQL item and simple PL/SQL item
- page link item and simple page link item
- zip file item

See also item type. Contrast with navigation item type.

Content Management Event Framework
See CMEF.

content repository
The OracleAS Portal schema within the Oracle Application Server Metadata Repository that contains the content and metadata associated with a particular portal instance.

Contribute privileges
A privilege level on an Oracle Instant Portal page that allows the user to add, edit, move, or delete items on the page. A user with Contribute privileges also has all View privileges.

CSS
Cascading Style Sheet. A simple mechanism for adding style, such as fonts, colors, and spacing, to Web documents.

current version
The version of an item that is displayed on the page. The current version is not necessarily the most recent version of the item.

See also versioning.

custom attribute
See attribute.
custom item type
See item type.

custom page type
See page type.

custom provider
A type of provider that enables you to create and maintain portlets that access customer-specific content or applications. You can build custom portlets in OracleAS Portal either declaratively or programmatically.

custom search
A search engine that enables users to define a variety of searches against information stored in the OracleAS Portal schema of the Oracle Application Server Metadata Repository. By editing the defaults of the Custom Search portlet, you can define unique search forms and search results pages that meet the specific search requirements, or configure portlets that execute and return results based on predefined search criteria.

If Oracle Text is installed and enabled, all text attributes and the actual content of documents and URLs are searched. If Oracle Text is not installed, or is disabled, the following metadata is searched: item attributes (Display Name, Description, Keywords, and Author), page attributes (Display Name, Description, and Keywords) and category/perspective attributes (Display Name and Description).

See also search portlet. Contrast with advanced search and basic search.

DAD
Database Access Descriptor. A set of values that specify how an application connects to an Oracle database to fulfill an HTTP request. The information in the DAD includes the user name (which also specifies the schema and the privileges), password, connect-string, error log file, standard error message, and Globalization Support parameters such as language, date format, date language, and currency.

Database Access Descriptor
See DAD.

database administrator
See DBA.

database object
See object.

database provider
A type of provider that is written as a PL/SQL stored procedure and is used to create portlets that reside in the database. One example of a database provider is a provider built using the Portlet Builder to provide form, report, and chart portlets.

Contrast with Web provider.

data portlet
A portlet built using the Portlet Builder that displays data in a spreadsheet format.

DAV
See WebDAV.
**DBA**
Database Administrator. A user belonging to the DBA group. By default, members in the DBA group have access to all OracleAS Portal product pages, and have the Manage privilege for all pages, page groups, database providers, and administration.

**default language**
The language in which a page group is originally created. After creating the page group, you can add translations to it to enable users to add content in different languages.

**default subscriber**
The base subscriber that is installed along with the install of OracleAS Portal.

**de-militarized zone**
See DMZ.

**dequeue**
The operation of retrieving a CMEF message from an Oracle Streams AQ queue. Contrast with enqueue.

**developer**
A user who builds portlets for others to include on their pages. Relies heavily on the APIs to extend the capabilities of OracleAS Portal; may frequently consult the Portal Knowledge Exchange or the forums for advice or inspiration.

**Developer Services**
See Portal Developer Services.

**DIP**
Directory Integration Platform. The provisioning platform provided by Oracle Internet Directory to synchronize different directories and directory enabled applications.

**direct access URL**
Obsolete terminology. See path-based URL.

**Directory Information Tree**
See DIT.

**Directory Integration Platform**
See DIP.

**display name**
An object’s external name used throughout OracleAS Portal, for example, in the Navigator, on pages, and in the page editor. When the object is published as a portlet, the display name is used as the title of the portlet in the Portlet Repository.

**Distinguished Name**
See DN.
DIT
Directory Information Tree. A hierarchical tree-like structure in Oracle Internet Directory consisting of the DNs of the entries.

DN
Distinguished Name. The unique name of a directory entry in Oracle Internet Directory. It includes all the individual names of the parent entries back to the root. The DN tells you exactly where the entry resides in the directory’s hierarchy. This hierarchy is represented by a directory information tree (DIT).

DMZ
De-militarized Zone. A computer host or small network inserted as a “neutral zone” between a company’s private network and the outside public network. It prevents outside users from getting direct access to a server that has company data. A DMZ is an optional and more secure approach to a firewall and effectively acts as a proxy server as well. (The term comes from the geographic buffer zone that was set up between North Korea and South Korea following the UN police action in the early 1950s.)

document control
See check out/check in.

draft item
An item that has been added to a page but has not yet been submitted for approval. In View mode, a draft item is visible only to its author. When a draft item is ready for approval, the author can submit it, at which point the approval process is triggered. Draft items can be added to a page only if approvals are enabled for the page group.

durable URL
A URL that uses the item’s GUID (globally unique ID) to uniquely identify it. An item’s GUID does not change so its durable URL will not break when the item is edited, renamed, moved, or imported to a different portal instance.
Contrast with path-based URL.

dynamic page
A portlet created with the Portlet Builder that displays dynamic content on a page. The dynamic page wizard enables you to specify one or many PL/SQL blocks within HTML code to create a page. This code executes every time an end user requests the page.

dynamic URL
A URL that contains a query string (one or more parameters and the characters ? and &). Contrast with path-based URL.

edge side includes
See ESI.

Edit Defaults mode
An optional portlet Show mode that enables administrators to set the defaults of a portlet for all users.
Contrast with Edit mode.
Edit mode

1. Page editing: Edit mode enables an authenticated user with appropriate privileges to set page properties and to add, modify, or delete portlets and items on the page. To switch to Edit mode, the user clicks an Edit link on the page. There are three Edit mode views: Graphical view, Layout view, and List view.

   See also Mobile Preview mode and Pending Items Preview mode.

2. Portlets: An optional portlet Show mode that enables personalization of the portlet for each user, for each instance.

   Contrast with Edit Defaults mode.

3. Oracle Instant Portal: Opposite of View mode. When the Edit mode handle bar is clicked, thus displaying the edit toolbar and the Add Item and Add Page buttons, the page is said to be in Edit mode. This is the only state in which action may be taken upon the page.

Edit mode handle bar

In Oracle Instant Portal, the gray icon that one clicks to display the edit toolbar. When the edit toolbar is displayed, the page is in Edit mode. Users with View privileges on the page do not see the Edit mode handle bar.

email item

In Oracle Instant Portal, an item represented by a yellow envelope and, optionally, a title on the page. When the user clicks the envelope, an email editor is opened displaying a blank email. The email is pre-populated with the address specified by the item's creator.

tenqueue

The operation of publishing messages to an Oracle Streams AQ queue.

Contrast with dequeue.

Enterprise Manager

Oracle Enterprise Manager 10g is a component of the Oracle Application Server that enables administrators to manage Oracle Application Server services through a single environment. For example, an administrator can use Enterprise Manager to monitor the services that make up an OracleAS Portal instance, including HTTP services, the PPE, the Oracle database, providers, and Oracle Ultra Search.

enterprise portal

A common, integrated starting point that provides personalized access to relevant enterprise information sources. Enterprise portals enable site visitors to personalize their view of the resources available on the public Internet.

ESI

Edge Side Includes. A markup language to enable partial page caching (PPC) of HTML fragments.

event

An action within a portal triggers a corresponding event. Page designers can specify what should happen when these events occur, for example, by specifying that a particular event forces the reloading of the current page, and passes parameters to the newly loaded page.
**Event servlet**
The Event servlet implements the functionality of OracleAS Portal to allow for dynamic page navigation when accessing an event enabled portlet. The Event servlet runs in the same container as the PPE.

**expandable rich text item**
In Oracle Instant Portal, a type of text item designed to conserve space on the page. When in View mode, an expandable rich text item is represented by a white envelope, a title, and a summary. When the user clicks the envelope, the item expands to reveal the text associated with the item. Compare to a rich text item, in which the full text of the item is never hidden. Both expandable rich text and rich text items may contain images, hyperlinks, and tables, as well as any valid HTML code.

**expiration period**
The number of days after which, or an exact date on which, an item expires. After an item expires, it is viewable only by the item's or page's owner and the page group administrator in Edit mode. Expired items are removed from the database during a system purge of all expired items.

**expiry-based caching**
A caching method that uses a retention period to specify how long the item is valid in the cache before a refresh is required. When there is a request for the item beyond the retention period, it is refreshed in the cache.

Oracle Application Server Web Cache uses both expiry-based caching and invalidation-based caching. Any data saved in OracleAS Web Cache is considered valid until it is invalidated or it expires. For example, if expiry-based caching is specified for a fully assembled page, the page content remains valid in the cache for the specified retention period before it needs to be regenerated.

See also invalidation-based caching and validation-based caching.

**expiry notification**
A message automatically sent to a user or group indicating that an item on the page is about to expire. The notification is set up by the page group administrator.

**explicit object**
An object which is explicitly selected, from the Navigator or Bulk Actions, for export.

See also manifest and referenced object.

**export**
A method of creating a set of files (transport set) that contains page groups, pages, portlets, and other content from a single OracleAS Portal instance. You can then import this set of files into another Oracle Application Server instance.

**eXtensible Markup Language**
See XML.

**extension**
A Java class that extends the functionality of Oracle JDeveloper. For example the PDK includes an extension to aid with the development of portlets with Oracle JDeveloper.
external application
An application external to OracleAS Portal that is typically launched from the External Applications portlet. As each external application is configured by a portal administrator, users simply supply their user name and password information. The Oracle Application Server Single Sign-On will present these credentials for future authentication challenges.

external object
An object which is an external dependency of an explicit object. External objects ensure that the explicit objects perform on the target portal.
See also manifest.

favorite
A hyperlink in the Favorites portlet that provides quick access to a frequently visited URL, either inside or outside your company firewall. An authenticated user can personalize the Favorites portlet with his or her own preferred set of frequently accessed URLs.

Favorite Content area
An area on the Oracle Instant Portal home page in which a user's favorite content appears. Users select items for this area by clicking an icon that looks like a house beside the item. The Favorite Content area is different for each user.

favorite group
A collection of favorites (and favorite groups) that are usually logically related.

Federated Portal Adapter
See FPA.

file item
A type of item that a user can add to a page. When a user adds a file item to a page, the file is uploaded into the OracleAS Portal schema of the Oracle Application Server Metadata Repository and is displayed as a hyperlink on the page. When a user clicks the display name link, the file may be downloaded to the user's computer or displayed in the user's Web browser, depending on the file type and the configuration of the browser.
In Oracle Instant Portal file items are represented by a rectangle with a red asterisk at the top and, optionally, a title and summary. When the user clicks the icon, the file associated with the item opens in a second browser window.

firewall
A system (either hardware or software) that acts as an intermediary to protect a set of computers or networks from outside attack. It regulates access to computers on a local area network from outside, and regulates access to outside computers from within the local area network. A firewall can work either by acting as a proxy server that forwards requests so that the requests behave as though they were issued by the firewall machine, or by examining requests and attempting to eliminate suspect calls.

form
A portlet created with the Portlet Builder that provides a transactional interface to one or more database tables, views, or procedures. For example, you can the Portlet Builder to build a form for entering new employee information into your Human Resources database.
See also master-detail form.

**FPA**
Federated Portal Adapter. The Federated Portal Adapter is a module in the portal instance (written in both Java and PL/SQL) that receives SOAP messages for a Web provider, parses the SOAP, and then dispatches the messages to a database provider as PL/SQL procedure calls. In effect, the Federated Portal Adapter makes a database provider behave exactly the same way as a Web provider, allowing users to distribute their database providers across database servers. All remote providers can be treated as Web providers, hiding their implementation (database or Web) from the user. The most common use is to share database providers (including page groups) owned by one portal instance among other portal instances.

**frame driver**
A portlet created with the Portlet Builder consisting of a Web page divided into two frames. A driving frame contains a SQL query that drives the contents of the second (target) frame.

**Full Screen mode**
An optional portlet Show mode that provides more content than can be shown in the portlet when it is sharing a page with other portlets.

**function**
A PL/SQL subprogram that performs a specified sequence of actions and then returns a value. Functions are usually small blocks of code written to perform a specific task within the scope of a larger application.

In a page, end users execute functions by clicking the title of a PL/SQL or custom item.

**gist**
An Oracle Text summary consisting of the document paragraphs that best represent the overall subject matter. You can use such summaries to skim the main content of the text or assess your interest in the text's subject matter.

**global privilege**
A privilege that grants a certain level of access to a user or group on all objects of a particular type. For example, you could grant a Web Designer group Manage privileges on all styles.

**grantee**
A user who is given privileges on an object by another user.

**Graphical view**
A page editing view that renders page content in-place on the page. Graphical view enables you to view pages and items as they appear on the finished page as you edit. Contrast with Layout view and List view.

**group**
A collection of OracleAS Portal users who typically share a common need or interest; for example, Human Resources, Accounting, and so on. Groups make it easy to grant access to an object (such as a page or portlet) to several users at once. You can also use groups to implement user roles by assigning role-related privileges to a group, then
adding users in that role. **Oracle Internet Directory** tracks the membership of OracleAS Portal groups.

**group owner**

A user who has the privilege to add or delete members from a **group**, or to delete the group itself. Groups can have more than one owner.

**HA**

High Availability. A collection of solutions to ensure that your applications meet the required availability to achieve your business goals, eliminating single points of failure with no or minimal outage in service.

**Handheld Device Markup Language**

See **HDML**.

**HDML**

Handheld Device Markup Language. A simple language to define hypertext-like markup content and applications for handheld devices with small display.

**Help mode**

An optional **portlet Show mode** that displays usage information about the functionality of the portlet.

**hierarchy**

A **portlet** created with the **Portlet Builder** that displays data from a self-referencing table or view. At least two columns in the table must share a recursive relationship. A hierarchy can contain up to three levels and display data such as employees in an organization chart or the hierarchical relationship between menus in a Web site.

**home page**

The **page**, defined within OracleAS Portal, that typically displays when logging on or when a user clicks a Home **smart link item**. The **portal administrator** chooses this page for **public users**; **authenticated users** may choose their own. If the portal administrator enables **mobile page** design, he or she can specify a separate mobile home page to display when the portal is accessed from a mobile device.

**hosted site**

See **stripe**.

**HTML**

Hyper Text Markup Language. A format for encoding hypertext documents that may contain text, graphics, and references to programs and other hypertext documents.

**HTML content layout**

A type of **HTML Template** that uses item-level **substitution tags** to define a formatting scheme for individual **regions**. The HTML content layout repeats itself for each **item** or **portlet** in the region.

Contrast with **HTML page skin**.

**HTML page skin**

A type of **HTML Template** that uses page-level **substitution tags** to control the appearance of the area surrounding page content. You can apply HTML page skins to **pages** or **Portal Templates**.
Contrast with HTML content layout.

**HTML Template**
A portal object built using your own HTML code that wraps around your page or region content. You can use an OracleAS Portal wizard or any third party HTML editor to build HTML Templates.

See also HTML content layout and HTML page skin. Contrast with Portal Template.

**HTTP**
Hyper Text Transfer Protocol. The underlying format, or protocol, used across the Web to format and transmit messages and determine what actions Web servers and browsers should take in response to various commands. HTTP is the protocol typically used between Oracle Application Server and its clients.

**Hyper Text Markup Language**
See HTML.

**Hyper Text Transfer Protocol**
See HTTP.

**iasconfig.xml**
The Portal Dependency Settings file, in which configuration settings for an OracleAS Portal instance are established. Each change to iasconfig.xml requires an update to the OracleAS Portal schema in the Oracle Application Server Metadata Repository, using the tool ptlconfig.

**IDE**

**ILS**
Item Level Security. A mechanism that controls granular access to items on a given page. ILS authorizes item managers to grant explicit item access to users and groups that take precedence over page-level privileges.

**image item**
A type of item that a user can add to a page. You can add images in JPEG, GIF, or PNG formats.

In Oracle Instant Portal, as opposed to the images that are added as part of a rich text item or expandable rich text item, image items are designed to standalone on the page. An image item is added through the Add Item icon; an image that is part of a text item is added through the Insert Image window.

**image map item**
A type of item that a user can add to a page. An image map is a single image with hotspots that, when clicked, link to other URLs. For example, you can create an image map of the world in which each continent is hyperlinked to more information about the continent.

**import**
A method of transporting content and objects (for example, page groups, pages, and portlets) into an OracleAS Portal instance. For example, you can import a page, its associated style, and its contents from one instance of OracleAS Portal to another.
index
An optional structure associated with a table used to locate rows of the table quickly, and (optionally) to guarantee that every row is unique.

in-place editing
In Oracle Instant Portal, the ability to add items or pages in context, as opposed to having to create these objects elsewhere and then add them to the page or portal.

Integrated Development Environment
See IDE.

internal image name
The name used to identify an image that has been uploaded to the portal. Uploaded images can be reused within the portal by referencing their internal names.

internal provider
A type of provider that makes page group objects (pages, navigation pages, and so on) available to instances of OracleAS Portal.

invalidation-based caching
A caching method where an item remains in the cache until it is explicitly invalidated. For example, a user may update an item, requiring the item in the cache to be invalidated. The next time there is a request for the invalidated item, it is refreshed in the cache.

Oracle Application Server Web Cache uses both expiry-based caching and invalidation-based caching. Any data saved in OracleAS Web Cache is considered valid until it is invalidated or it expires. When the information cached in OracleAS Web Cache becomes inaccurate, it must be invalidated. For example, page metadata saved in OracleAS Web Cache is invalidated when a page designer changes the page structure or when user privileges change. Likewise, a portlet instance is invalidated whenever it is personalized by an end user.

See also expiry-based caching and validation-based caching.

item
1. An individual piece of content (text, hyperlink, image, and so on) that resides on a page in an item region. Users with an appropriate privilege level can add items to a page. Item content and metadata are stored in the OracleAS Portal schema of the Oracle Application Server Metadata Repository. Items are rendered on the page according to the layout, style, and attribute display defined for the item region.

   See also item type.

2. In Oracle Instant Portal, the means through which content is added to a page. Available item types are rich text item, expandable rich text item, file item, URL item, email item, and image item.

item ID
The local database reference to the content of an item. An item ID value is used in custom item types to pass items to PL/SQL procedures. The function uses the item ID to access the content of the item.

item level security
See ILS.
item metadata
The stored information or attributes of an item.

item placeholder item
A type of item that a user can add to a page. An item placeholder identifies where the content from items that use a Portal Template for items will display in relation to the rest of the template content.

item type
An object that defines the contents of an item and the attributes that are stored (metadata) about an item. Base item types included with OracleAS Portal are categorized as content item types and navigation item types.

Custom item types are item types created by page group administrators to extend the functionality provided by base item types and store additional attribute information about items.

item versioning
See versioning.

J2EE
Java 2 Platform, Enterprise Edition. A platform that enables application developers to develop, deploy, and manage multitier, server-centric, enterprise level applications. The J2EE platform offers a multitiered distributed application model, integrated XML-based data interchange, a unified security model, and flexible transaction control. You can build your own J2EE portlets and expose them through Web providers.

See also OC4J.

J2SE
Java 2 Platform, Standard Edition. A platform that enables application developers to develop, deploy, and manage Java applets and applications on a desktop client platform such as a personal computer or workstation. J2SE not only defines API standards, but also specifies the deployment of enterprise applications, thus enabling application server administrators to perform the deployment regardless of the vendor of the J2SE server.

See also OC4J.

Java 2 Platform, Enterprise Edition
See J2EE.

Java 2 Platform, Standard Edition
See J2SE.

JavaScript
A scripting language developed by Netscape that enables generation of portlets that introduce dynamic behavior in otherwise static HTML. The Portlet Builder enables you to use JavaScript to create routines that validate entry fields in forms and personalization forms. You can also create JavaScript event handlers for entry fields and buttons on forms.

Java Specification Request
See JSR 168.
JavaServer Page
See JSP.

JSP
JavaServer Pages. An extension to servlet functionality that provides a simple programmatic interface to Web pages. JSPs are HTML pages with special tags and embedded Java code that is executed on the Web or application server. JSPs provide dynamic functionality to HTML pages. They are actually compiled into servlets when first requested and run in the servlet container.

See also JSP tags.

JSR 168
Java Specification Request (JSR) 168. Defines a set of APIs for building standards-based portlets using Java. Portlets built to this specification can be rendered to a portal locally or deployed to a WSRP container for rendering portlets remotely. For more information, see http://jcp.org/en/jsr/detail?id=168.

JSP tags
Tags that can be embedded in JSPs to enclose Java code. These tags use the <jsp: syntax and enclose action elements in the JSP with begin and end tags similar to XML elements.

keyword
An attribute used to provide additional information about a page or item so that users can easily locate the page or item during a search.

Layout view
A page editing view that enables you to add, arrange, and remove regions on the page. You can also hide, show, delete, or move content in this view.
Contrast with Graphical view and List view.

LBR
Load-balancing router. A very fast network device that distributes Web requests to a large number of servers. It provides portal users with a single published address, without their having to send each request to a specific middle tier server.

LDAP

level
An object used to provide structure to mobile pages and as a way to limit the amount of content displayed on the smaller screens of mobile devices. Users drill down into the levels on a mobile page to view more content.

library
A collection of one or more PL/SQL or Java program units. Libraries can be referenced by several applications simultaneously.

Lightweight Directory Access Protocol
See LDAP.
Link mode
An optional portlet Show mode that enables portlets to render themselves on mobile devices, such as cellular telephones.

List view
A page editing view that displays a listing of all page content and provides options that enable you to perform actions (delete, move, copy, and so on) on multiple objects.
Contrast with Graphical view and Layout view.

list of objects item
A type of navigation item that a user can add to a page to list objects (for example, pages and perspectives) as a drop-down list or as links (with or without associated images).

list of values
See LOV.

load-balancing router
See LBR.

local provider group
The collection of providers that are defined within an instance of OracleAS Portal. Provider groups make it easier to share providers defined or registered within one instance of OracleAS Portal with other OracleAS Portal instances.
See also provider group. Contrast with remote provider group.

lock
1. A setting automatically applied to a Portlet Builder portlet when it is being edited. The setting prevents other users from editing the portlet.
2. In WebDAV the action of preventing other users from editing a file. Locking a file in a WebDAV client checks out the corresponding item in the portal itself.

login/logout link item
A type of navigation item that a user can add to a page to enable other users to log in or log out of the portal.

LOV
List of values. A portlet created with the Portlet Builder that enables developers to add selectable values to entry fields in forms. A single list of values can be displayed in different formats, such as combo boxes, radio buttons, or check boxes.

Manage privileges
A privilege level on an Oracle Instant Portal page that allows the user to edit, move, or delete a top-level page and all its child pages. A user with Manage privileges on a portal's home page is considered an administrator of that portal. A user with Manage privileges also has all Contribute and View privileges.

manifest
The list of objects in a transport set and their dependents, which provides a granular level of control over the import mode.
**master-detail form**

A portlet created with the Portlet Builder that displays a master table row and multiple detail rows within a single HTML page. Values in the master row determine which detail rows are displayed for querying, updating, inserting, and deleting.

See also form.

**master item ID**

An identifier for an item that is the same for all versions of that item.

**menu**

A portlet created with Portlet Builder that displays a Web page containing options that end users can click to navigate to other menus, other Portlet Builder portlets, or URLs.

**message payload**

Provides details about CMEF events such as the portal action that triggered the event, the type of event (INSERT, UPDATE, or DELETE), and the state of the event.

**metadata**

Data that describes data. For example item attributes store item metadata, such as display name, description, and author.

**middle tier**

Part of the Oracle Application Server architecture that handles HTTP user requests by forwarding them to the appropriate portal database or provider, assembles portal pages, and manages caching of portal content.

**MIME type**

Multipurpose Internet Mail Extension type. A message format used on the Internet to describe the contents of a message. MIME is used by HTTP servers to describe the type of content being delivered.

**mobile page**

A type of page that enables page designers to produce pages specifically for mobile devices, for example, cellular phones.

**Mobile Preview mode**

A preview mode that enables you to preview how your page will look on a mobile device.

**mobile XML**

See Oracle Application Server Wireless XML.

**Model-View-Controller**

See MVC.

**mod_oc4j**

The Oracle HTTP Server module that manages the communication between the Oracle HTTP Server and OC4J.
mod_plsql
The Oracle HTTP Server module that handles the database connections made from the Oracle HTTP Server. It enables PL/SQL database procedures to generate HTTP responses containing formatted data and HTML code that can display in a Web browser.

MVC
A classic design pattern often used by applications that need the ability to maintain multiple views of the same data. The MVC pattern hinges on a clean separation of objects into one of three categories: models for maintaining data, views for displaying all or a portion of the data, and controllers for handling events that affect the model or views. Because of this separation, multiple views and controllers can interface with the same model. Even new types of views and controllers that never existed before, such as portlets, can interface with a model without forcing a change in the model design.

navigation area
In Oracle Instant Portal, the area along the left side of the page in which the search box and a list of the page’s child pages appear.

navigation item type
A means of providing navigation and access to portal-specific functions.
Built-in navigation item types include:
- smart link item
- smart text item
- login/logout link item
- basic search box item
- list of objects item
- object map link item
- page path item
- page function item
See also item type. Contrast with content item type.

navigation page
A special purpose page within a page group that is typically embedded on other pages or Portal Templates to implement standard user interface effects such as navigation bars and banners. Often contains navigation item types for navigation within the portal.

Navigator
A feature for locating objects and interacting with OracleAS Portal. Provides access to objects to which the user has privileges, such as page groups, providers, and database objects.

New Content area
An area on the Oracle Instant Portal home page in which content added anywhere in the portal over the last 24 hours is gathered.

non-default subscriber
A subscriber that has a stripe on a hosted OracleAS Portal provided by an ASP.
object
1. Portal object: A structure such as a page group, portlet, page, or style.
2. Database object: An Oracle database structure such as a table, procedure, or trigger. These objects can be created using OracleAS Portal wizards or Oracle database commands.

object map link item
A type of navigation item that a user can add to a page to display a map of objects that are available in the portal.

OC4J
Oracle Application Server Containers for J2EE. The J2EE server component of Oracle Application Server written entirely in Java that executes on the standard Java Development Kit (JDK) Virtual machine (Java VM). It includes a JSP Translator, a Java servlet container, and an Enterprise JavaBeans (JB) container.

OEM
See Enterprise Manager.

OID
See Oracle Internet Directory.

OmniPortlet
A Web provider that provides portlets that can display spreadsheet, XML, and Web Service data as tabular, chart, news, bullet, and form layouts.

Oracle Application Server
Oracle's integrated application server:
- Is standards compliant (J2EE, Web Services, and XML)
- Delivers a comprehensive set of capabilities, including portal, caching, wireless, integration, and personalization
- Provides a single, unified platform for Java and J2EE, Web Services, XML, SQL, and PL/SQL

Oracle Application Server Metadata Repository
An Oracle database that contains schemas and business logic used by application server components (including OracleAS Portal) and other pieces of the infrastructure.

Oracle Application Server Portal
A component of Oracle Application Server used for the development, deployment, administration, and configuration of enterprise class portals. OracleAS Portal incorporates a portal building framework with self-service publishing features to enable you to create and manage information accessed within your portal.
Oracle Application Server Portal Developer Kit
See PDK.

Oracle Application Server Single Sign-On
A component of Oracle Application Server that enables users to log in to all features of the Oracle Application Server product suite, as well as to other Web applications, using a single user name and password. OracleAS Portal is integrated with Oracle Application Server Single Sign-On as a partner application and delegates authentication to it.

Oracle Application Server Web Cache
A component of Oracle Application Server that improves the performance, scalability, and availability of frequently used Web sites. By storing frequently accessed URLs in memory, Oracle Application Server Web Cache eliminates the need to repeatedly process requests for those URLs on the Web server. Oracle Application Server Web Cache uses invalidation-based caching and is integrated with OracleAS Portal for improved performance.

Oracle Application Server Wireless
A component of Oracle Application Server used to deliver information and applications to mobile devices. Using Oracle Application Server Wireless, you can create custom portal sites that use different kinds of content, including Web pages, custom Java applications, and XML-based applications. Oracle Application Server Wireless sites make this diverse information accessible to mobile devices without your having to rewrite the content for each target device platform.

Oracle Application Server Wireless XML
A device independent markup language used for communication between OracleAS Portal and Oracle Application Server Wireless.

OracleAS Portal
See Oracle Application Server Portal.

OracleAS Portal Verification Service
(Previously known as Portal Studio). A major component of Portal Center (http://www.oracle.com/technology/products/ias/portal) that is specifically tuned to the needs of the portal developer. This site provides developers a way to test and display remote portlets exposed as Web or WSRP providers without having to install their own copy of OracleAS Portal. Developer’s portlets reside on their servers and must be accessible over the Internet. You can access OracleAS Portal Verification Service directly at http://portalstandards.oracle.com.

Oracle Enterprise Manager
See Enterprise Manager.

Oracle HTTP Server
The Web server component of Oracle Application Server, built on Apache Web server technology and used to service HTTP requests. It is the part of the middle tier that handles requests between the Web and OracleAS Portal. Extensions to the Oracle HTTP Server support Java servlets, JSPs, Perl, PL/SQL, and CGI applications.
**Oracle Instant Portal**
A product built on top of OracleAS Portal, designed to provide a common place to share and exchange content for smaller groups of users.

**Oracle Instant Portal administrator**
A user who has full privileges over the entire portal. Any user with Manage privileges on the portal's home page is considered an Oracle Instant Portal administrator for that portal. The PORTAL and ORCLADMIN users, created during the installation process, are Oracle Instant Portal administrators for all portals. Only an Oracle Instant Portal administrator can change the banner or style, manage users, and create new top-level pages.

**Oracle Internet Directory**
The repository for storing OracleAS Portal user credentials and group memberships. By default, the Oracle Application Server Single Sign-On authenticates user credentials against Oracle Internet Directory information about dispersed users and network resources. Oracle Internet Directory combines LDAP version 3 with the high performance, scalability, robustness, and availability of the Oracle database.

**Oracle JDeveloper**
Oracle JDeveloper is an integrated development environment (IDE) for building applications and Web services using the latest industry standards for Java, XML, and SQL. Developers can use Oracle JDeveloper to create Java portlets.

**Oracle PartnerNetwork Solutions Catalog**
(http://solutions.oracle.com/) A collection of information on Oracle Partner joint products and services. The Solutions Catalog includes information on partners who offer OracleAS Portal related products and services.

**Oracle Streams Advanced Queuing**
Oracle Streams Advanced Queuing (AQ) provides database-integrated message queuing functionality. It is built on top of Oracle Streams and leverages the functions of the Oracle database so that messages can be stored persistently, propagated between queues on different computers and databases, and transmitted using Oracle Net Services and HTTP(S).

**Oracle Technology Network**
See OTN.

**Oracle Text**
A feature of Oracle9i and later that provides advanced search and retrieval services on content stored in an Oracle repository. It is fully integrated into OracleAS Portal to provide users with the ability to perform a full text search and retrieve content managed within the OracleAS Portal schema of the Oracle Application Server Metadata Repository. It also provides automatic grouping and classification of results by gist and theme.

**Oracle Ultra Search**
An Oracle Text-based application that supports crawling, indexing, and federated searching of multiple, heterogeneous repositories including databases, file systems, Web servers, and e-mailing list archives.
Contrast with search portlet.
**Oracle Workflow**
Oracle Workflow enables you to automate and continuously improve business processes, routing information of any type according to easily-changeable business rules to users both inside and outside a company’s enterprise.

**OTN**
Oracle Technology Network. The online Oracle technical community that provides a variety of technical resources for building Oracle-based applications. You can access OTN at [http://www.oracle.com/technology/](http://www.oracle.com/technology/).

**Overwrite mode**
See Replace on Import mode.

**package**
A database object consisting of a PL/SQL specification and a body. The specification includes the data types and subprograms that can be referenced by other program units. The body includes the actual implementation of the package.

**page**
A portal object that contains portlets and items. Each time you display a page, it is dynamically assembled and formatted according to the portlets and layout chosen for that page.

See also page type.

**page designer**
A user with the Manage privilege on a page (also known as a page manager). A user with this privilege can perform any action on the page and can create sub-pages under the page. The page designer is often responsible for designing the layout (or region configuration) of the page and assigning privileges on the page to other users (for example, to determine who can add content to the page).

The scope of a page designer’s control over a page may be limited if the page is based on a template.

**page function item**
A type of navigation item that a user can add to a page. A page function is a procedure call that a user can add to a custom page type. If there are no page functions associated with the current page, the page function item does not display.

**page group**
A portal object that groups and sets properties of related portal objects, such as pages, styles, navigation pages, and perspectives. Page groups typically contain a hierarchy of pages and sub-pages for organizing content.

**page group administrator**
A user who has full privileges over an entire page group. Page group administrators set up and maintain the page group; designate page owners; and create a taxonomy. Page group administrators can also view and manage all the pages in the page group.

**page group map**
A hierarchical representation of all page groups in a portal, which enables users to access individual pages within the page group. The page group map is tailored for each user; only the pages the user is authorized to view or edit are displayed.
page group quota
See quota.

page link item
A type of item that a user can add to a page. A page link provides a route using a hyperlink to another page within the portal. When the user clicks the display name link, the page referenced by the item is displayed in the user’s browser.

page manager
See page designer.

page metadata
Stored information or attributes about a page, which is used by OracleAS Portal to set its layout and cache.

page path item
A type of navigation item that a user can add to a page. A page path is a chain of page reference names. Page paths, often called breadcrumbs, describe the complete directory path.

page toolbar
In page Edit mode, the links at the top of the page that enable you to edit various aspects of the page, switch editing views (for example, from Graphical view to Layout view), edit page group properties, and so on.

page type
A portal object that defines the content of a page and the information that is stored about a page. Base page types included with OracleAS Portal are: standard page, mobile page, PL/SQL page, JSP, and URL page. Custom page types are page types created by page group administrators to extend the functionality provided by base page types and store additional information about pages.

Parallel Page Engine
See PPE.

parameter
A value passed between pages and portlets, or between portlets.
A page parameter is a page level parameter (created by a page designer) whose values can be mapped to portlet parameters.
A portlet parameter is declared by a provider. Page designers map page parameters to portlet parameters. When the PPE requests a portlet from a provider, only the portlet parameters that the portlet declared and mapped to page parameters are sent.

parameter entry field
A field on a personalization form that enables end users to enter values that will be passed to an OracleAS Portal portlet.

partial page caching
See PPC.
**partner application**

An application that has delegated its authentication to Oracle Application Server Single Sign-On. If registered with the OracleAS Single Sign-On, users can log in to multiple partner applications using a single log in page. In a given session, once users have been authenticated by the OracleAS Single Sign-On, they won't need to log in again to access additional partner applications.

**path aliasing**

See path-based URL.

**path-based URL**

A path-based URL identifies the path taken through the portal to get to a particular object. It is an easy-to-read URL but as it contains the names of portal objects, the URL becomes invalid if the name of any object within the path changes.

For example, the path-based URL to access a top-level page (sample_page) of the page group MyPageGroup:


Contrast with durable URL.

**PDK**

OracleAS Portal Developer Kit. The development framework used to build and integrate Web content and applications with OracleAS Portal. It includes toolkits, samples, and technical articles that help make portal development simple. You can take existing Java servlets, JSPs, URL-accessible content and Web Services and turn them into portlets. It is typically used by external developers and vendors to create portlets and services. The PDK is regularly updated on Portal Center (http://portalcenter.oracle.com/) to provide developers with the latest tools and techniques.

See also PDK-Java, PDK-PL/SQL, and PDK-URL Services.

**PDK-Java**

A toolkit for implementing portlets in Java and adding portal features. Used to declaratively turn your existing Java servlets, JSPs, and Web services into portlets.

See also PDK. Contrast with PDK-PL/SQL.

**PDK-PL/SQL**

A set of articles, samples, and services that enable PL/SQL programmers to easily create portlets and extend them by using PL/SQL APIs.

See also PDK. Contrast with PDK-Java.

**PDK-URL Services**

A utility for declaratively turning secured and public Web content into portlets. These services are capable of dynamically passing parameters to target URLs, clipping and reformattting content, and providing Oracle Application Server Single Sign-On for applications requiring form-based or basic authentication. These services also allow developers to take any application written in any language and easily create integrated portlets. PDK-URL Services takes the URL of an application, parses the content, and uses the PDK-Java framework to create a portlet.

See also PDK.
Pending Approvals Monitor
A portlet that enables you to list pending approvals in the page groups that you administer. You can list the pending approvals by approver, date, page group, or submitter.

Pending Items Preview mode
A preview mode that enables you to view items that are awaiting approval. This mode can be used by content contributors to preview items they have added before they are approved, and by approvers to preview items before they approve or reject them.

See also Edit mode.

personalization form
A page that prompts end users for values to pass to a Portlet Builder portlet. End users can view the personalization form for a portlet, if one has been created, by clicking the portlet's Personalize link.

personal page
An area within OracleAS Portal where authenticated users can store personal content and share it with other users. The portal administrator can choose to create a personal page for a user when creating a user account.

perspective
A cross-category grouping of items. Perspectives help users answer the question, Who will be interested in this item? For example, you can add links to diverse vacation spots around the world and assign perspectives like Vacations for Nordic Enthusiasts, Archeology Expeditions, and Extreme Vacations for Adventurers to items about vacation types. Users publishing content using an item type that includes a perspective attribute may specify none, one, or many values.

Contrast with category.

PL/SQL item
A type of item that a user can add to a page. A PL/SQL item contains a block of PL/SQL code. When a user clicks the item, the code is executed. The result displays in the user’s browser. PL/SQL items can also be displayed directly on the page.

PL/SQL function
See function.

PL/SQL page
A type of page. PL/SQL pages contain PL/SQL code that generates HTML when the page is rendered.

plug-in
See extension.

poll
A set of questions used to find out information from users.

Contrast with survey and test.

portal
A common interface (that is, a Web page) that provides a personalized, single point of interaction with Web-based applications and information relevant to individual users.
or class of users. Portals built using OracleAS Portal are made up of pages managed within page groups, containing portlets and items.

**portal administrator**
A user with the highest level of privileges in OracleAS Portal. Portal administrators can view and modify anything in OracleAS Portal, even pages and database providers marked private. (The only exception is groups: although portal administrators can modify the PORTAL_ADMINISTRATORS and PORTAL_PUBLISHERS groups, they cannot modify any other group unless they have been named group owner.)

**Portal Builder page**
A predefined page that contains development and administrative portlets used to build and manage portal objects and services.

**portal cache**
A mechanism for storing cache entries for objects that use validation-based caching. It also acts as a backup to the memory-based Oracle Application Server Web Cache when objects use both validation-based caching and invalidation-based caching.

**Portal Catalog**
See Oracle PartnerNetwork Solutions Catalog.

**Portal Center**
(http://portalcenter.oracle.com/) A Web site where you can find out everything you want to know about OracleAS Portal. Updated frequently, it always has the latest product information, and is home to the Portal Developer Services and OracleAS Portal Verification Service. This Web site contains all information about the product (including documentation, demonstrations, and so on), and provides access to OracleAS Portal expertise.

**Portal Community**
A network of people dedicated to creating and exchanging information about OracleAS Portal. This community includes anyone who uses OracleAS Portal; it can leverage the Portal Knowledge Exchange for sharing Portal-related information and take advantage of the Portal Developer Services.

**Portal DB provider**
See database provider.

**Portal Developer Kit**
See PDK.

**Portal Developer Services**
The network of people dedicated to creating and exchanging OracleAS Portal expertise. This program provides online testing tools through Portal Center (http://portalcenter.oracle.com/), as well as other venues, and interaction with the product team and other portal developers, using newsletters, surveys, and the Portal Knowledge Exchange.

See also Portal Community.

**Portal Knowledge Exchange**
A self-service Web site where subscribers to the Portal Developer Services can share white papers, techniques, and portlets with others in the Portal Community.
Contributions can also be rated so that the most valuable contributions can be easily located.

**portal page**
See page.

**portal repository**
See Oracle Application Server Metadata Repository.

**Portal Services**
A set of services running in the OracleAS Portal OC4J instance that are used to assemble portal pages and to access portal and page metadata. The Parallel Page Engine (PPE) is one of the Portal Services that assembles portal pages. Other services, like those previously provided by mod_plsql, are incorporated into the Portal Services as well.

**portal session**
A period of interaction between a browser and OracleAS Portal, from the initial access to log off, closure of the browser window, or expiration of the session after a period of inactivity.

**Portal smart link item**
See smart link item.

**Portal smart text item**
See smart text item.

**Portal Studio**
See OracleAS Portal Verification Service.

**Portal Template**
A portal object built declaratively using a wizard that enforces specific layouts, colors, fonts, and backgrounds for pages and items. You can use Portal Templates with navigation pages, standard pages, and pages based on custom page types that are based on the standard page type. You can also use Portal Templates with text items, PL/SQL items, URL items, and all file items with a MIME type of text/html or text/plain.

Contrast with HTML Template.

**portlet**
A reusable, pluggable Web component that typically displays portions of Web content. Portlets are the fundamental building blocks of a portal page. Using the Portlet Builder, you can easily create your own portlets. OracleAS Portal also provides several ways to build portlets programmatically and to integrate any kind of Web content. Portlets may be implemented using various technologies, such as Java, JSPs, Java servlets, PL/SQL, Perl, ASP, and so on. The PDK covers the standard-based portlet development options that OracleAS Portal provides.

**Portlet Builder**
A collection of portlet-building wizards that are accessible through the Provider tab in the Portal Navigator. You can use these wizards to build charts, forms, reports, calendars, and lists of values.
portlet instance

A portlet placed on a particular page.

portlet provider

See provider.

portlet publisher

A user who can publish portal objects (pages or otherwise) as portlets so that they can be included on pages.

portlet record

A programmatic structure that contains detailed information about a portlet, such as its implementation style and Show mode (PL/SQL).

Portlet Repository

A special page group that contains the portlets available from the local providers and any registered remote providers. When you register a provider, the provider and its portlets are added to the Portlet Repository.

PPC

Partial Page Caching. A feature that enables Oracle Application Server Web Cache to independently cache and manage fragments of HTML documents. A template page is configured with Edge Side Includes (ESI) markup tags that tell Oracle Application Server Web Cache to fetch and include the HTML fragments. The fragments themselves are HTML files containing discrete text or other objects.

PPE

Parallel Page Engine. A multithreaded servlet engine that runs in the OC4J container and services page requests. The PPE reads page metadata, calls providers for portlet content, accepts provider responses, and assembles the requested page in the specified page layout. The Parallel Page Engine is part of the Portal Services, which run on the Oracle Application Server middle tier.

pretty URL

See path-based URL.

Preview mode

An optional portlet Show mode that provides users with a preview of the portlet before they add it to a page.

primary key

One or more columns in a database table that, in combination, uniquely identify a row in a table.

privilege

In OracleAS Portal, the right to perform an action. Privileges are either global (set through in the User or Group Profile) or specific to particular objects (usually set through the object’s Access tab). When building applications, access can also be granted to database objects, shared portlets, portlets, and applications.

producer

See provider.
profile
The information stored about an OracleAS Portal user or group, such as password, user ID, and privileges.

property sheet
A built-in attribute that displays a summary of an item’s attributes or a page’s properties.

provider
The communication link between OracleAS Portal and a portlet. There are two types of providers: Web providers and database providers. Web providers may reside anywhere on the network and are addressed through SOAP. Web providers may be implemented using any Web technology. You can build your own Web providers by using the PDK-Java and the PDK-URL Services. Database providers reside within an Oracle database and manage portlets while performing data-intensive operations. Providers act as containers for portlets; each portlet communicates with OracleAS Portal through its provider. Providers also manage the portlets they contain.

provider definition
A declarative, XML-based configuration file (provider.xml) that describes a Web provider, its portlets, and the location of the content to be displayed in the portlets. This configuration file also describes the behavior of the provider and its portlets.

provider group
A logical collection of Web providers defined by a provider group service. A portal administrator can register provider groups for use with their portal. Once registered, a provider group simplifies the process of registering individual providers in the group. This enables organizations that create Web providers to publish registration details of their providers and facilitate automatic registration with any OracleAS Portal instance. The only information that must be given to the portal administrator is the name and location of the provider group.

See also local provider group and remote provider group.

provider record
The record returned by a database provider containing specified information about a portlet.

proxy server
A proxy server typically sits on a network firewall and enables clients behind the firewall to access Web resources. All requests from clients go to the proxy server rather than directly to the destination server. The proxy server forwards the request to the destination server and passes the received information back to the client. The proxy server channels all Web traffic at a site through a single, secure port; this enables an organization to create a secure firewall by preventing Internet access to internal machines, while allowing Web access.

ptlconfig
The Portal Dependency Settings Tool, used to update the OracleAS Portal schema in the Oracle Application Server Metadata Repository with settings specified in the iasconfig.xml file.
**public page**
Any page in a page group that is viewable by public users (users who are not logged onto OracleAS Portal). The page designer or page group administrator must explicitly designate a page as public.

**public user**
A user who can access, but is not logged onto, OracleAS Portal. When users first access OracleAS Portal, they do so as public users, whether or not they have the ability to log on. A public user can view any page that has been marked as public, but cannot personalize or edit any content, or view pages that have any form of access control. Contrast with authenticated user.

**purge**
See system purge.

**query**
A SQL SELECT statement that specifies which data to retrieve from one or more tables or views in a database.

**queue**
The abstract storage unit used by a messaging system, such as CMEF, to store messages.

**quota**
The amount of space provided in a page group or in the OracleAS Portal schema of the Oracle Application Server Metadata Repository to store uploaded documents.

**recent object**
A portal object, such as a page or portlet, that has recently been displayed or edited. Each authenticated user has his or her own Recent Objects portlet that provides links to the last n objects accessed.

**referenced object**
An object which is directly or indirectly referenced by an explicit object.
See also manifest.

**reference path**
The path that uniquely identifies a portlet instance on a page. A reference path can be used to target parameters to individual portlets on a given page.

**region**
A carved-out area on a standard page used to define the page layout, define page content (portlets and items), and control the style and attributes for content displayed in a region. A standard page can have one or multiple regions. Regions can be created above, beneath, or beside other regions.

You can create the following types of regions:
- Undefined regions are regions that have not been assigned a particular type.
- Item regions allow you to add items such as text, images, files, and so forth.
- Portlet regions allow you to include portlets in a region.
- **Sub-Page Links regions** allow you to display a list of the current page's sub-pages in a region.

- **Tab regions** allow you to include tabs in a region.

**Region banner**
A colored, horizontal bar with a title displayed in a region of a portal page. A banner breaks up the visual flow of a page and groups related items that appear beneath it.

**Remote database**
A database running on a separate machine that can be accessed over the network through a connect string or database link.

**Remote provider group**
The collection of providers that are defined outside of your local instance of OracleAS Portal.

See [provider group](#). Contrast with [local provider group](#).

**Replace on Import mode**
An import mode. When this option is selected, if the object exists on the target, then it is replaced. If the object does not exist then it is created. When this option is not selected, if the object exists on the target, it is referenced. If it does not exist on the target, it is created.

**Report**
A portlet created with the Portlet Builder that displays the results of a SQL query in a tabular format.

**Reuse mode**
See [Replace on Import mode](#).

**Rich text editor**
A WYSIWIG editor that enables content contributors to easily apply formatting to text items.

**Rich text item**
One of the available items in Oracle Instant Portal, designed for smaller blocks of text. A rich text item appears on the page as a title, a summary, and the full text of the item. Compare to [expandable rich text item](#), in which the text is hidden until the user clicks an icon. Both expandable rich text and rich text items may contain images, hyperlinks, and tables, as well as any valid HTML code.

**Root page**
The top level of the page hierarchy in a page group; it contains all other sub-pages in the page group. Also known as the page group’s home page.

**Routing method**
A mechanism for organizing an approval process. OracleAS Portal provides three approval routing methods:

- **All, Parallel** enables OracleAS Portal to send the approval to recipients in the step all at the same time. All of the recipients must respond to the approval before the item approval can move to the next step.
- All, Serial enables OracleAS Portal to send the approval to recipients in the step one at a time in the sequence specified. All of the recipients must respond to the approval before the item approval can move to the next step.

- Any, Parallel enables OracleAS Portal to send the approval to recipients in the step all at the same time. However, only one of the recipients must respond to the approval before the item approval can move to the next step.

**row**
A set of values in a table; for example, the values representing one employee in the SCOTT.EMP table.

**saved search**
A mechanism for saving search criteria under a single name. This feature enables you to repeat the search quickly, by choosing the saved search name rather than re-entering the criteria manually. You can save the results of a basic search, advanced search, or custom search. The Saved Searches portlet lists all the saved searches in a page group.

**schema**
A collection of database objects, including logical structures such as tables, views, sequences, stored procedures, synonyms, indexes, clusters, and database links. A schema has the name of the user who controls it.

**search portlet**
A portlet that enables users to search for pages and content within the OracleAS Portal schema of the Oracle Application Server Metadata Repository. Users can also search based on text strings, categories, perspectives, and attributes, and using operators such as CONTAINS, GREATER THAN, LESS THAN, and EQUAL TO.

Contrast with Oracle Ultra Search.

**secure view**
A view on the data in the content repository that is guaranteed to not change between releases. You should use the secure views to query data on the documents and items stored in the content repository to ensure that those queries will continue to work with later releases of Oracle Application Server Portal.

**self registration**
A mechanism that allows users to create new accounts for themselves through a link in the Login portlet.

**sequence**
A database object used to automatically generate numbers for table rows.

**servlet**
A Java program that usually runs on a Web server, extending the Web server’s functionality. HTTP servlets take client HTTP requests, generate dynamic content (such as through querying a database), and provide an HTTP response.

**session**
See portal session.

**shared object**
A portal object such as a personal page, navigation page, style, Portal Template, perspective, category, or custom type that can be shared across page groups.
shared portlet
A portlet created with the Portlet Builder that is shared between other Portlet Builder portlets. Each shared portlet can be displayed on multiple pages with the same personalization.

Shared Screen mode
A portlet Show mode that renders the body of the portlet. Every portlet must have at least a Shared Screen mode.

Show mode
The ways by which a portlet can be called to display information. These methods include:

- Shared Screen mode
- Edit mode
- Edit Defaults mode
- Preview mode
- Help mode
- About mode
- Link mode
- Full Screen mode

simple file item
Similar to a file item except with fewer attributes, which makes it quicker and easier to create.

simple image item
Similar to an image item except with fewer attributes, which makes it quicker and easier to create.

Simple Object Access Protocol
See SOAP.

simple page link item
Similar to a page link item except with fewer attributes, which makes it quicker and easier to create.

simple PL/SQL item
Similar to a PL/SQL item except with fewer attributes, which makes it quicker and easier to create.

simple text item
Similar to a text item except with fewer attributes, which makes it quicker and easier to create.

simple URL item
Similar to a URL item except with fewer attributes, which makes it quicker and easier to create.
Single Sign-On
See Oracle Application Server Single Sign-On.

smart link item
A type of navigation item that is self-configuring. For example, if you add a Home smart link to a navigation page, when a user clicks the Home link, he or she is automatically taken to his or her home page.

smart text item
A type of navigation item that is self-configuring. For example, if you add a Current Date smart text item to a navigation page, the current date is automatically pulled from the server and does not need to be specified (through complex coding) by you.

snapshot
A table that contains the results of a query on one or more tables, called master tables, in a remote database.

snapshot log
A table associated with the master table of a snapshot tracking changes to the master table.

SOAP
Simple Object Access Protocol. A lightweight, XML-based protocol for exchanging information in a decentralized, distributed environment. SOAP supports different styles of information exchange, including: Remote Procedure Call style (RPC) and Message-oriented exchange. RPC style information exchange allows for request-response processing, where an endpoint receives a procedure-oriented message and replies with a correlated response message. Message-oriented information exchange supports organizations and applications that need to exchange business or other types of documents where a message is sent but the sender may not expect or wait for an immediate response.

SSL accelerator card
A hardware device that handles traffic much faster than regular SSL software. See also LBR.

SSO
See Oracle Application Server Single Sign-On.

standard page
A type of page used to contain and manage items and portlets.

state
In the context of CMEF, the state of an event provides additional information about that event so that developers can correctly determine what a subscriber should do when a portal action triggers an event with that particular state.

stored procedure
A set of PL/SQL procedures that are stored in a database.

stripe
A secure slice on a virtual private portal that is assigned to a particular subscriber.
structured UI template
A shared portlet that controls the look and feel of Portlet Builder portlets and runs in standalone mode. Structured UI templates display the same image and text in the same location around every portlet that uses the template.

See user interface (UI) template. Contrast with unstructured UI template.

struts
A development framework for Java servlet applications based upon the MVC design paradigm.

style
A set of values and parameters that controls the colors and fonts of pages and regions. Style settings include font style, size, color, alignment, and background color. Styles can be created for a specific page group or as a shared object that is used by pages within multiple page groups.

sub-category
A category that appears hierarchically beneath another (parent) category. This provides a way of grouping closely related categories together.

sub-item
An item that appears hierarchically beneath another (parent) item. This provides a way of grouping closely related items together, for example, the spreadsheet that is used by a particular HTML file item could be added as a sub-item of the HTML file item.

sub-page
A page that appears hierarchically beneath another (parent) page. Every page in a page group (except the root page) is a sub-page.

sub-perspective
A perspective that appears hierarchically beneath another (parent) perspective. This provides a way of grouping closely related perspectives together.

subscriber
In the context of CMEF, a subscriber consumes messages from the WWSBR_EVENT_Q queue and performs actions based on those messages.

In the context of hosted portals, a subscriber is a company that signs up with an ASP and receives a stripe on a hosted OracleAS Portal.

subscription notification
A method by which end users can subscribe to a particular page or item so that they are notified (through the My Notifications portlet) when that page or item is updated. The page designer must include a Subscribe Portal smart link item for users to be able to subscribe to a page, and must display the Subscribe attribute for users to be able to subscribe to items. Additionally, the page group administrator must enable approvals and notifications for the page group.

substitution tag
A special portal tag used within HTML Templates and unstructured UI templates to dynamically embed titles, headings, and other elements into the template.
survey
A set of questions used to find out information from users. Surveys can redirect users to different sections of the survey depending on their answers to particular questions. Contrast with poll and test.

synonym
An additional name assigned to a table or view that can thereafter be used to refer to it.

system level caching
A caching method where a single copy of an item is stored in the cache (on the middle tier) for all users. Consequently, such items cannot be user specific. All users see the same content for the item. Contrast with user level caching.

system purge
A process that deletes all items in a page group from the OracleAS Portal schema of the Oracle Application Server Metadata Repository that are marked as deleted or expired. System purges are performed by the page group administrator or portal administrator.

tab
An area on a page used to increase the amount of content that the page can display by effectively doubling (or tripling, quadrupling, and so on) the amount of real estate available. Tabs also allow you to group content that is common to a subject area, organization, specific role, and so forth.

table
The basic storage structure in a relational database.

tablespace
The allocation of space in the database.

template
See HTML Template, Portal Template or user interface (UI) template.

temporary tablespace
The allocation of space in the database used for the creation of temporary table segments for operations such as sorting table rows.

test
A set of questions used to assess a user’s understanding of a particular subject or subjects. You can provide the correct answer for questions and assign each question a score. You can also hand score essay-type answers. Contrast with poll and survey.

text item
A type of item that a user can add to a page. When you create a text item, you enter text (up to 32KB) in the Item Wizard. The text block is then stored in the OracleAS Portal schema of the Oracle Application Server Metadata Repository.
theme
A snapshot generated by Oracle Text that describes a document. Rather than searching for documents that contain specific words or phrases, users can use Oracle Text to search for documents that are about a certain subject, even if that subject is not mentioned explicitly in the document.

title
See display name.

top-level page
An Oracle Instant Portal page represented by a tab in the portal’s tab set. May contain one or more child pages. Page privileges applied to a top-level page are also applied to all its child pages.

translation
A page group rendered in another language. When a page group administrator creates a translation, content contributors can add content in that language. Page group users can also view the translated content by setting their language to one of the supported languages.

transport set
A collection of portal objects for export or import. It can contain more than one object of a particular type, such as multiple page groups and multiple pages.

trigger
A database object associated with a table. It executes before or after one or more specified events.

Ultra Search
See Oracle Ultra Search.

Uniform Resource Locator
See URL.

unstructured UI template
A shared portlet that is used to insert content and HTML code to control the look and feel of Portlet Builder portlets. Unstructured UI templates are based on HTML code that, when executed, dynamically embeds titles, headings, and other elements that make up a page.

See also substitution tag and user interface (UI) template. Contrast with structured UI template.

URL
Uniform Resource Locator. A compact string representation of the location for a resource that is available through the Internet. It is also the format Web clients use to encode requests to Oracle Application Server.

URL item
A type of item that a user can add to a page. A URL item, when clicked, provides a route to another Web page. When a user clicks the URL item’s display name, the Web page referenced by the URL displays.
In Oracle Instant Portal, a URL item is represented on the page by a globe and, optionally, a title and summary. When the user clicks the globe, the location specified in the item opens in a secondary browser window.

**URL page**
A type of page that provides a route to another Web page, identified by its URL. When a user clicks the page link, the Web page referenced by the link is displayed.

**URL portlet**
A portlet created with the Portlet Builder that displays the contents of a Web page specified by a URL.

**user interface (UI) template**
A shared portlet that controls the look and feel of Portlet Builder portlets in full page display mode. Selecting a UI template when you are building a portlet automatically selects a title on the page where the portlet is displayed, a title background, links to other Web pages, and background colors and images.

See also structured UI template and unstructured UI template. Contrast with HTML Template and Portal Template.

**user level caching**
A caching method where a copy of an item is stored in the cache (on the middle tier) for each user. Consequently, users may see different content for the same item.

Contrast with system level caching.

**validation-based caching**
A caching method that uses a validation check to determine if the cached item is still valid. This is the caching method used by the portal cache. Before an item in the portal cache is used, the PPE contacts the portal repository or the provider that the content came from to determine if the cached item is still valid.

Contrast with expiry-based caching and invalidation-based caching.

**versioning**
A mechanism that allows multiple versions of an item to simultaneously exist in the OracleAS Portal schema of the Oracle Application Server Metadata Repository. This feature is useful for tracking document changes from one version to the next or for reverting to a previous version if necessary.

**view**
A virtual table whose rows do not actually exist in the database, but which is based on a table that is physically stored in the database.

**View mode**
The runtime view of a page.

Contrast with Edit mode.

**View privileges**
A privilege level on an Oracle Instant Portal page that allows the user to view content on the page. All users may navigate pages, select items for the Favorite Content area of the home page, and search for content.
**virtual private database**
See VPD.

**virtual private portal**
Features for hosting multiple companies or multiple organizations securely within the same portal instance.

**VPD**
Virtual Private Database. A feature for ASPs that want to leverage the Oracle database to host their customers. Essentially, it uses one physical database instance for all customers, but to each customer it looks as if they have their own database. Users cannot see any information that is not meant for them and complete customer isolation is achieved. It requires little to no changes in the core application to take effect as most of the work is done at the database level. Implementing VPD basically requires two key steps: adding a context column (for example, company name) to all the database tables, and implementing a policy to restrict queries on each table based on the context of the logged in user. VPD provides highly secure, full subscriber isolation using this method.

**WAP**
Wireless Application Protocol. A set of open, global protocols for developing applications and services that use wireless networks.

**Web Cache**
See Oracle Application Server Web Cache.

**Web clipping**
A feature that enables page designers to collect Web content into a single centralized portal. It can be used to consolidate content from hundreds of different Web sites scattered throughout a large organization.

**WebDAV**
Web-based Distributed Authoring and Versioning. A protocol extension to HTTP 1.1 that supports distributed authoring and versioning. With WebDAV, the Internet becomes a transparent read and write medium, where content can be checked out, edited, and checked in to a URL address.

**Web Services for Remote Portlets**
See WSRP.

**Web provider**
An entity that is called, using an HTTP request, by OracleAS Portal and returns portlet content in HTML, XML, or WSRP. A Web provider acts as a proxy for one or more portlets that are defined within a particular application environment (for example, Java, ASP, or Perl) and executed as applications external to OracleAS Portal. Web providers are particularly appropriate for Web-accessible information sources. See also provider. Contrast with database provider.

**Web server**
A program that delivers Web pages.

**Wireless Application Protocol**
See WAP.
**Wireless Markup Language**

See [WML](#).

**wireless portal**

A portal accessible from wireless devices, such as cellular telephones.

See also [Oracle Application Server Wireless](#).

**wizard**

A graphical interface that guides a user step-by-step through a process. In OracleAS Portal, wizards are used for creating [database providers](#), [portlet](#), [database objects](#), [pages](#), [page groups](#), and [items](#).

**WML**

Wireless Markup Language. An XML-based markup language used to define hypertext-like content and applications for handheld devices.

**WSRP**

Web Services for Remote Portlets (WSRP). A Web services standard that allows the plug-and-play of visual, user-facing Web services with portals or other intermediary Web applications. Being a standard, WSRP enables interoperability between a standards-enabled container based on a particular language (such as JSR 168, .NET, Perl) and any WSRP portal. So, a portlet (regardless of language) deployed to a WSRP-enabled container can be rendered on any portal that supports this standard.

**XML**

Extensible Markup Language. An open standard for describing data using a subset of the SGML syntax.

**XML portlet**

An OracleAS Portal portlet that displays the executed results of XML code. To create the portlet, you either specify XML code or a [URL](#) that points to the XML code.

**XSL**

Extensible Stylesheet Language. The language used within stylesheets to transform or render XML documents.

**zip file item**

A type of item that a user can add to a page. Zip file items enable you to upload many files in a single operation. You can use them to migrate the contents of a file system or Web site into the OracleAS Portal schema of the [Oracle Application Server Metadata Repository](#). When you upload a zip file to OracleAS Portal, then unzip the uploaded file, a page is created for each directory and an item is created for each file. The items are published in the target page.
A

About mode, 6-7
access control lists
   see ACLs
Access tab for managing portlets, A-70
ACLs, 7-39
   portlet privileges, 7-39
   privileges, 7-39
   provider privileges, 7-39
add_category API, 12-3
add_content_area API, 12-1
add_folder API, 12-2
add_item API, 12-4, 12-5
add_item_ils_privileges API, 15-6
add_item_post_upload API, 12-6
add_perspective API, 12-3
ADF, 7-91
ALIGN_LEFT substitution tag, A-99
ALIGN_RIGHT substitution tag, A-100
Apache struts
   creating a portlet, 7-87
   overview, 7-85
APIs, 1-10
   content management, xix
   providing access to, 9-4
   supported, 9-3, F-1
wwctx_api.get_nls_language, 9-8, 14-2
wwctx_api.set_context, 10-1
wwctx_api.set_nls_language, 14-2
wwpro_invalidation_api.execute_cache_invalidation, 9-5
wwsbr_api.add_category, 12-3
wwsbr_api.add_content_area, 12-1
wwsbr_api.add_folder, 12-2
wwsbr_api.add_item, 12-4, 12-5
wwsbr_api.add_item_ils_privileges, 15-6
wwsbr_api.add_item_post_upload, 12-6
wwsbr_api.add_perspective, 12-3
wwsbr_api.approve, 16-27
wwsbr_api.check_in_item, 11-12
wwsbr_api.check_out_item, 11-12
wwsbr_api.clear_cmef_context, 9-6
wwsbr_api.copy_folder, 11-16
wwsbr_api.copy_item, 11-16
wwsbr_api.delete_folder, 11-19
wwsbr_api.delete_ils_privilege, 15-7
wwsbr_api.delete_item, 11-17
wwsbr_api.enable_ils_for_item, 15-5
wwsbr_api.inherit_folder_privileges, 15-8
wwsbr_api.modify_folder, 11-5
wwsbr_api.modify_item, 11-11
   p_addnewversion, 11-13
wwsbr_api.modify_item_post_upload, 12-8
wwsbr_api.move_category, 11-15
wwsbr_api.move_folder, 11-14
wwsbr_api.move_item, 11-14
wwsbr_api.move_perspective, 11-15
wwsbr_api.reject, 16-27
wwsbr_api.set_attribute, 11-7
wwsbr_api.undo_delete_item, 11-18
wwsbr_api.upload_blob, 12-6
wwsec_api.grantee_list, 15-1
wwsec_api.remove_group_acl, 15-4
wwsec_api.remove_user_acl, 15-4
wwsec_api.set_group_acl, 15-3
wwsec_api.set_user_acl, 15-3
wwsrc_api.get_all_items_xml, 13-7
wwsrc_api.get_item_xml, 13-8
wwsrc_api.item_search, 13-2
wwsrc_api.page_search, 13-3
wwsrc_api.specify_attributes, 13-4
Application Development Framework, 7-91
application.log, B-3
approve API, 16-27
attributes
   events, G-9
   searching, 13-4
   setting for items, 11-6
   values, 11-7
audio data, A-46
authenticated content
   Single Sign-On and, 1-7, 5-2, 5-10
authentication, 7-35
   comparison of portlet builders, 2-21
   external application, 7-38
   message, 7-37, 7-42
   none, 7-39
   partner application, 7-37
   proxy
      Web Clipping and, B-20
server, 7-37

Index
single sign-on, 7-37
authorization, 7-36

B

BACK_LINK, 7-16
base attributes
values, 11-7
Batch button, adding, A-80
batch mode, for running portlets, A-80
best practices
About mode, 6-7
CSS for Shared Screen mode, 6-4
Edit Defaults mode, 6-5
Edit mode, 6-4
error handling PL/SQL, 8-38
event handling PL/SQL, 8-42
for Java portlets, 6-2
Help mode, 6-7
HTML for Shared Screen mode, 6-3
Link mode, 6-8
navigation, 6-8
PL/SQL procedures and functions, 8-3
Preview mode, 6-6
security PL/SQL, 8-28
bind variables
LOV, A-67
using, A-83
Blank item type, form portlets, A-19
BLOBs, A-19, A-48
BODY substitution tag, A-99
browsers
recommended cache settings, xxii
recommended image settings, xxii
recommended versions, xxii
browsing
Web Clipping and, 5-7
Button item type, form portlets, A-19

caching, 7-62
activation, 7-63
adding, 7-64, 7-65, 7-68
expiry-based, 2-12, 7-62
expiry-based PL/SQL, 8-35
invalidation port, 7-65
invalidation-based, 2-12, 7-63
invalidation-based PL/SQL, 8-36
manually invalidating the cache, 7-68
PL/SQL, 8-31
provider servlet for invalidation based, 7-65
provider.xml for invalidation based, 7-67
recommended browser settings, xxii
style, 2-12
system-level, 2-12
user-level, 2-12
validation-based, 2-12, 7-63
validation-based PL/SQL, 8-33
calendar portlets, A-11
categories
applying to an item, 16-17
creating, 12-3
events, G-11
character sets
Web Clipping and, 5-2, 5-27
character-separated values
building a portlet based on a spreadsheet, 4-5
chart portlets, A-11
accessible tables and views, A-51
column conditions, A-53
common display options, A-55
comparison, 2-18
customization form button options, A-61
customization form display options, A-58
customization form formatting options, A-60
definitions, A-103
formatting conditions, A-54
from query wizard, A-49
full page display options, A-56
group functions, A-52
labels, A-48, A-51
links, A-48, A-51
mobile display options, A-57
PORTLET code options, A-63
portlet display options, A-57
supplementary text options, A-62
values, A-48, A-52
check box, LOV, A-66
check_in_item API, A-11-12
check_out_item API, A-11-12
CheckBox item type, form portlets, A-19
cipher manager, 7-58
clear_cmef_context API, 9-6
clearing cache, 7-62
clipping sections, 5-7
CMEF
about, 16-1
applying a category, 16-17
applying a perspective, 16-17
approving an item, 16-16
creating a page, 16-13
creating a subscriber, 16-8
creating an item, 16-15
deleting a page, 16-15
deleting an item, 16-18
dequeuing messages, 16-4
enabling CMEF events, 16-9
enqueuing messages, 16-2
external workflow example, 16-30
installing examples, 16-18
item approval, 16-16
item notification example, 16-22
item verification example, 16-27
listening to messages, 16-6
message payload, 16-11
portal object event logging example, 16-19
registering a subscriber, 16-3, 16-9
running a subscriber, 16-11
updating a page group ACL, 16-14
updating page ACL, 16-14
Index-2
code
  adding functionality, A-82
  bind variables, A-83
  viewing body, A-72
  viewing call interface, A-72
  viewing package spec, A-72
  viewing portlet source, A-71
color, creating definitions for, A-95
column conditions, A-31
combo box, LOV, A-66
ComboBox item type, form portlets, A-19
communication
  HTTPS, 7-43
  security, 7-36
connection
  to application server in Oracle JDeveloper, 6-33, 6-50
connection information
  OmniPortlet, 3-5
connection problems
  Web Clipping, B-19
constructResourceURL, 6-9
content
  adding to page, 5-2
  rendering inline, 2-16
content management, xix
Content Management Event Framework
  see CMEF
context
  setting for session, 10-1
context information
  accessing PL/SQL, 8-25
  function calls PL/SQL, 8-25
  obtaining using wwctx_api package, 8-26
cookies, A-87
  Web Clipping and, 5-26
copy_folder API, 11-16
copy_item API, 11-16
Create shared components privilege, A-92
creating
  WSRP portlet preference store, 6-22
cSS
  guidelines for Shared Screen mode, 6-4
customization
  implementing PL/SQL session store, 8-20
  preference store for PL/SQL, 8-15
customizations
  transporting, 7-47
Customize privilege, A-75
customizing
  Show page, 7-12

D
data
  filtering, 3-11
data component portlets, A-12
data sources
  filtering data, 3-11
  using a spreadsheet, 3-6
using a Web Service, 3-10
using an existing Web page, 3-11
using SQL, 3-6
using with portlets, 3-3
using XML, 3-9
database providers, 2-9
DBA group, A-92
DBA Studio, 16-4
DBMS_AQADM.ADD_SUBSCRIBER
  procedure, 16-3
DBMS_AQADM.REMOVE_SUBSCRIBER
  procedure, 16-3
DBMS_AQ.DEQUEUE
  procedure, 16-4
  CONSUMER_NAME parameter, 16-5
  DEQUEUE_MODE parameter, 16-5
  NAVIGATION parameter, 16-5
DBMS_AQ.LISTEN
  procedure, 16-6
dcmctl, D-6
debugging information
  Web Clipping and, B-18
default profile, A-3
delete_folder API, 11-19
delete_ils_privilege API, 15-7
delete_item API, 11-17
deploying
  properties, D-8
  testing, D-8
  WAR file for JPS portlet, 6-35
  WAR file for PDK-Java portlet, 6-51
dequeuing, 16-4
DESIGN_LINK, 7-16
develop in-place portlets, 2-14
Develop tab for managing portlets, A-70
develop-in-place portlets, 2-3, A-10
DeviceClass header, 8-51
DIRECTION substitution tag, A-99
directories, virtual, A-96
doDelete, A-87
dolInsert, A-87
doUpdate, A-87
dynamic page portlets, A-11

E
EAR file, D-2
  creating manually, D-4
  deploying manually, D-5
Edit Defaults mode, 3-21, 6-5
Edit Defaults page, 7-10
Edit mode, 6-4
Edit page, 7-10
Edit privilege, A-75
enable_ils_for_item API, 15-5
encryption
  example, 7-57
  for transport, 7-57
  message, 7-37
enqueuing, 16-2
error handling
  PL/SQL, 8-37
error messages
  Web Clipping,  B-18
errors
  Web Clipping and,  B-18, B-20
event handling
  PL/SQL,  8-41
  EVENT_LINK,  7-16
  events,  7-13
  adding to a portlet,  3-21
  attributes,  G-9
  categories,  G-11
  configuring with portlet parameters,  4-18
  enabling for CMEF,  16-9
  generated code in provider.xml,  7-25
  item types,  G-9
  items,  G-1
  page groups,  G-8
  page types,  G-10
  pages,  G-5
  perspectives,  G-11
  submitting,  7-24
  tabs,  G-7
  templates,  G-12
  using with OmniPortlet,  3-22
examples
  average salaries chart,  A-103
  building an OmniPortlet on a spreadsheet (CSV),  4-5
  building an OmniPortlet on a Web Service,  4-3, 4-25
  building an OmniPortlet on an existing Web page,  4-10
  building an OmniPortlet on an XML data source,  4-8
  configuring portlet parameters and events,  4-18
  installing for CMEF,  16-18
  job profiles report,  A-106
  JSR 168, registering,  6-21
  OmniPortlet,  4-1
  team bonuses report,  A-104
  team details report,  A-101
  using Chart layout,  4-5
  using HTMLayout,  4-25
  using RSS and the news layout,  4-8
  using Tabular layout,  4-3
Execute privilege,  A-75
execute_cache_invalidation API,  9-5
expiration
  portlet content,  5-9
Expires field
  Web Clipping and,  5-9
  expiry-based caching,  2-12
export
  by reference,  7-57
  by reference example,  7-60
  customizations,  7-47
  encrypting for,  7-57
  encryption example,  7-57
  example of,  7-50
  JNDI variable,  7-57
logging interface,  7-50
programming interface,  7-48
securing communications,  7-56
security,  7-56
exportData,  7-48, 7-49
external applications
  authentication,  7-38
  Web Clipping and,  1-7, 5-2, 5-10
external links,  6-9

F
  file not found,  B-8
  file upload (binary) item types, form portlets,  A-19
  file upload (interMedia) item types
    form portlets,  A-19
    including in form,  A-48
  files
    creating file items,  12-5
    uploading,  12-6
  font definitions, creating,  A-96
  form portlets,  A-11, A-14
    adding JavaScript,  A-94
    audio data,  A-46
    doDelete,  A-87
    doInsert,  A-87
    doUpdate,  A-87
    formatting and validation options,  A-17
    getting and setting values,  A-86
    image data,  A-46
    interMedia rich content,  A-46
    JavaScript Event Handlers,  A-19, A-23
    p_session,  A-87
    p_value,  A-86
    PL/SQL code options,  A-25
    PL/SQL Event Handlers,  A-19
    set_value,  A-86
    supplementary text options,  A-24
    video data,  A-47
  format masks
    numbers,  A-34
    formatting and validation options
      buttons,  A-18
      columns,  A-20
      form level,  A-17
  formatting conditions,  A-35
  forms
    building with the Portlet Builder,  A-14
    building with URL types,  7-19
    Web Clipping and,  5-9, 5-19
  frame driver portlets,  A-11
  Full Screen mode,  6-7
  fuzzy matching
    with Web Clipping,  1-7, 5-1

G
  generator
    PL/SQL,  8-4
  get method
transforming for JPS portlets, B-10
get_all_items_xml API, 13-7
get_item_xml API, 13-8
get_nls_language API, 9-8, 14-2
grantee_list API, 15-1
grants, A-4
groups
  DBA, A-92
  PORTAL_DEVELOPERS, A-92
guidelines
  About mode, 6-7
  CSS for Shared Screen mode, 6-4
  Edit Defaults mode, 6-5
  Edit mode, 6-4
  error handling PL/SQL, 8-38
  event handling PL/SQL, 8-42
  Help mode, 6-7
  HTML for Shared Screen mode, 6-3
  Link mode, 6-8
  mobile, PL/SQL, 6-10, 8-4
  navigation, 6-8
  Preview mode, 6-6
  security PL/SQL, 8-28
  security, 7-56
import
  by reference, 7-57
  by reference example, 7-60
  customizations, 7-47
  encrypting for, 7-57
  encryption example, 7-57
  JNDI variable, 7-57
  logging interface, 7-50
  programming interface, 7-48
  securing communications, 7-56
  security, 7-56
importData, 7-49
inherit_folder_privileges API, 15-8
init.ora parameters, A-80
inline rendering, 2-16
  Web Clipping and, 1-8, 5-2
input parameters
  Web Clipping and, 5-9, 5-19
interMedia rich content, A-46, A-48
internal links, 6-9
internationalization
  Web Clipping and, 5-2
intraportlet links, 6-9
  URL parameters, 7-16
invalidation-based caching, 2-12
ISO-8859-1 character set, 5-27
item ID
  finding, 10-4
  item level security
    see ILS
item types
  events, G-9
    file upload (interMedia), A-48
item types, form portlets
  blanks, A-19
  buttons, A-19
  check boxes, A-19
  combo boxes, A-19
  file upload (binary), A-19
  file upload (interMedia), A-19
  hidden, A-19
  horizontal rule, A-19
  image, A-19
  label only, A-19
  password, A-19
  text area, A-19
  text box, A-19
item_search API, 13-2
items
  applying a category, 16-17
  applying a perspective, 16-17
  approval, 16-16
  approving, 12-10, 16-16
  associating with uploaded files, 12-6
  checking in, 11-12
  checking out, 11-12
  copying, 11-16
  creating, 12-4, 16-15
  creating file items, 12-5
  defining privileges, 15-6

H
Help mode, 6-7
HELP LINK substitution tag, A-99
HELPSCRIPT substitution tag, A-99
hidden item types, form portlets, A-19
hierarchy portlets, A-11
horizontal rule item types, form portlets, A-19
HTML
  guidelines for Shared Screen mode, 6-3
  HTML layout
    using with OmniPortlet, 4-25
htmlFormHiddenFields, 7-20
HTTP error code 407
  Web Clipping and, B-20
httpd.conf, 7-41
HTTPS, 7-43
  configuration, 7-43
I
IDs
  finding, 10-2
ILS, 15-5
  defining privileges, 15-6
  enabling for item, 15-5
  enabling for page, 15-5
image data, A-46
Image item types, form portlets, A-19
IMAGE_PREFIX substitution tag, A-99
images
  creating definitions for, A-95
  in Web Clipping clip, B-21
  recommended browser settings, xxiii
  resource proxy, 6-9
  virtual directory, A-96

Index-5
deleting, 11-17, 16-18
editing, 11-11
enabling ILS, 15-5
events, G-1
moving, 11-14
new version, 11-13
rejecting, 12-10
searching, 13-2
setting attributes, 11-6
setting perspectives, 12-8
translations, 14-3

personalizing portlets, 7-2
wizard in Oracle JDeveloper, 6-26
JSR-168, 2-2

L
Label Only item types, form portlets, A-19
language
items, 14-3
setting for session, 14-2
string loading PL/SQL, 8-45
string retrieval PL/SQL, 8-46
language support
see multi-lingual portlets
layout
bullet, 3-18
chart, 3-14
form, 3-18
news, 3-17
OmniPortlet, 3-13
tabular, 3-13
LDAP
see Oracle Internet Directory
limitations
Web Clipping and, 5-26
Link mode, 6-8
for mobile PL/SQL portlets, 8-50
link portlets, A-11
links
building with URL parameters, 7-18
portlet, 6-8
list of values
portlets, A-12
lists of values
bind variables, A-67
check box, A-66
combo box, A-66
creating, A-64
dynamic, A-64
multiple select, A-66
pop-up list, A-66
radio group, A-66
static, A-64
load balancers
Web Clipping and, B-19
locks, on portlets, A-73
logging
application.log, B-3
provider levels, B-3
Web Clipping files, B-18
Web Clipping levels, B-18
LOGIN_LINK, 7-16
logon denied message
Web Clipping and, B-20

M
Manage privilege, A-75
Manage shared components privilege, A-92
Manage tab for managing portlets, A-70

Index-6
matrix, portlet technologies, 2-1
menu portlets, A-11
message payload, 16-11
messages
500, B-8, B-10, B-14
authentication, 7-42
dequeuing, 16-4
encryption, 7-37
enqueuing, 16-2
listening, 16-6
migration
preference store, 7-9
MIME type
setting for mobile portlets, 8-48
MLS
see multi-lingual portlets
mobile
accessing configuration data, 7-74
determining device type, 8-48
DeviceClass header for PL/SQL, 8-51
Link mode for PL/SQL, 8-50
navigation, 7-76
PL/SQL guidelines, 6-10, 8-4
portlets, PDK-Java, 7-69
portlets, PL/SQL, 8-47
mobile device display options, A-40
mod_osso
Web Clipping and, 5-26
mode
Edit Defaults, 3-21
modes
About, 6-7
adding render, 7-5
Edit, 6-4
Edit Defaults, 6-5
Full Screen, 6-7
Help, 6-7
Link, 6-8
list of Show, 6-2
Preview, 6-6
Shared Screen, 6-2
modify_folder API, 11-5
modify_item API, 11-11
p_addnewversion, 11-13
modify_item_post_upload API, 12-8
move_category API, 11-15
move_folder API, 11-14
move_item API, 11-14
move_perspective API, 11-15
multi-lingual
PL/SQL portlets, 8-44
multi-lingual portlets
comparison, 2-20
provider.xml, 7-81
resource bundles, 7-78
updating renderer, 7-79
multiple select, LOV, A-66

N

navigation
implement within a portlet, 7-21
link API, 7-18
with Web Clipping, 1-7, 5-1
within a Java portlet, 6-8
NLS
see multi-lingual portlets
Web Clipping and, 5-2

O

OmniPortlet, 1-9
adding to a page, 4-3
authentication for external application, 2-21
building based on a spreadsheet, 4-5
building based on an existing Web page, 4-10
building based on an XML data source, 4-8
building based on character-separated values, 4-5
bullet layout, 3-18
caching style, 2-13
capturing content, 2-16
chart layout, 3-14
Chart layout example, 4-5
charting, 2-18
comparing to other portlet builders, 2-2
configuring portlet parameters and events, 4-18
connection information, 3-5
definition, 3-1
development tool, 2-13
event support, 2-20
Events tab, 3-21
tables, 4-1
tables, 4-1
tables, 4-1
tables, 4-1
experts required, 2-6
Filter tab, 3-11
filtering data, 3-11
form layout, 3-18
HTML layout example, 4-25
Layout tab, 3-13
multi-lingual support, 2-21
news layout, 3-17
pagination support, 2-21
parameter support, 2-19
proxy authentication, 3-4
rendering content inline, 2-17
RSS and news layout example, 4-8
security, 2-20
Source tab, 3-4
spreadsheets (CSV) example, 4-5
tabular layout, 3-13
Tabular layout example, 4-3
troubleshooting, B-15
Type tab, 3-3
usage suitability, 2-4
using data sources, 3-3
using the wizard, 3-2
View tab, 3-12
Web page example, 4-10
Web Service example, 4-3, 4-25
XML example, 4-8
Oracle Diagnostics and Tuning pack, 16-4
Oracle Enterprise Manager
  DBA Studio, 16-4
Oracle Internet Directory, 7-44
Oracle JDeveloper
  application server connection, 6-33, 6-50
deploying JPS portlet, 6-33
deploying PDK-Java portlet, 6-50
portal add-in, 6-24
  WAR file deployment, 6-35, 6-51
Oracle Streams AQ
  and CMEF, 16-1
Oracle struts portlet, 7-86
OracleAS Metadata Repository, 9-1
OracleAS Portal
  add-in for Oracle JDeveloper, 6-24
  repository, 2-10
  using data sources, 3-3
OracleAS Portal pages
  Web Clipping and, 5-26
OracleAS Web Cache
  invalidation port, 7-65, 7-67
  invalidation-based caching, 2-13
ORDAUDIO, A-46
ORDIMAGE, A-46
ORDVIDEO, A-47
owa_cookie package
  data types, A-87
  subprograms, A-87
P
  p_session, A-87
  p_value, A-86
package
  implementing portlet with PL/SQL, 8-10
  implementing provider with PL/SQL, 8-11
page
  partial refresh, 7-23
page designer, definition, xix
page groups
  creating, 12-1
  enabling CMEF events, 16-9
  events, G-8
  searching all, 13-2
  searching in specific, 13-3
  updating ACL, 16-14
page manager
  see page designer
page metadata, 2-10
page parameters, 2-18
page types
  events, G-10
PAGE_LINK, 7-16
page_search API, 13-3
PAGE.BASE substitution tag, A-100
PAGE.BASE.URL substitution tag, A-100
PAGE.BGCOLOR substitution tag, A-100
PAGE.BGIMAGE substitution tag, A-100
PAGE.CUSTOMIZEPAGE substitution tag, A-100
PAGE.CUSTOMIZEPAGE.URL substitution tag, A-101
PAGE.CUSTOMIZEPAGE.LABEL substitution tag, A-101
PAGE.EDITPAGE substitution tag, A-101
PAGE.EDITPAGE.LABEL substitution tag, A-101
PAGE.EDITPAGE.URL substitution tag, A-101
PAGE.REFRESH substitution tag, A-101
PAGE.REFRESH.LABEL substitution tag, A-101
PAGE.REFRESH.URL substitution tag, A-101
  copying, 11-16
  creating, 12-2, 16-13
  creating an item, 12-4
  creating file items, 12-5
  deleting, 11-18, 16-15
  editing, 11-1
  enabling ILS, 15-5
  events, G-5
  moving, 11-14
  searching, 13-3
  setting perspectives, 12-8
  updating ACL, 16-14
PAGE.STYLE substitution tag, A-100
PAGE.STYLE.URL substitution tag, A-100
PAGE.SUBPAGELINK substitution tag, A-100
Pagination support in portlets, comparison, 2-21
Parallel Page Engine (PPE), 2-10
parameters, 7-13
  adding, 7-14
  building links with URL, 7-18
  configuring portlet parameters and events, 4-18
  generated code in provider.xml, 7-15
  mapping public PL/SQL, 8-23
  passing page PL/SQL, 8-23
  passing private PL/SQL, 8-23
  PL/SQL, 8-22
  qualified PL/SQL, 8-23
  retrieving values PL/SQL, 8-24
  types, 2-18
  unqualified PL/SQL, 8-23
  URL, 7-16
  using with OmniPortlet, 3-22
  Web Clipping and, 5-9, 5-19
parameters and events
  using with OmniPortlet, 3-22
partner application, 7-37
Password item types, form portlets, A-19
PDK, 1-10
PDK-PL/SQL
  comparing to other portlet builders, 2-2
  deploying portlet, 6-50
  JNDI variables, 7-31
  manually packaging providers, D-1
  personalizing portlets, 7-8
  Portlet Wizard, 6-43
  render modes, 7-5
  testing portlet and provider, 6-49
PDK-PL/SQL
  comparing to other portlet builders, 2-2
pending items
approving, 12-10
rejecting, 12-10
performance
caching, 7-62
caching PL/SQL, 8-31
persistent state variables, A-88
personal pages
Web Clipping and, 5-15
personalizing
code generated by wizard, 7-10
Edit and Edit Defaults pages, 7-10
JPS portlets, 7-2
modifying generated code, 7-11
PDK-Java portlets, 7-8
personalizing content
Web Clipping, 5-15, 5-18
perspectives
applying to an item, 16-17
creating, 12-3
events, G-11
setting for items, 12-8
setting for pages, 12-8
PL/SQL
accessing context information, 8-25
building portlets, 8-9
caching, 8-31
error handling, 8-37
event handling, 8-41
gateway, A-87
generating for portlets, A-71
generator, 8-4
getting and setting values, A-86
implementing the portlet package, 8-10
implementing the provider package, 8-11
owa_cookie package, A-87
parameters, 8-22
preference store, 8-15
publishing generated portlets, 8-8
recommended procedures and functions, 8-3
running the generator, 8-7
security, 8-27
session store, 8-20
starter provider sample, 8-9
XML for generator, 8-4
PL/SQL code options
chart portlets, A-63
form portlets, A-25
report portlets, A-45
PL/SQL Event Handlers, A-84
for buttons, A-19, A-85
writing, A-85
PL/SQL portlets
authentication for external application, 2-22
building expertise required, 2-7
caching style, 2-13
capturing content, 2-16
charting, 2-18
comparing to other portlet builders, 2-2
development tool, 2-13
event support, 2-20
mobile guidelines, 6-10, 8-4
multi-lingual support, 2-21
pagination support, 2-21
parameter support, 2-19
rendering content inline, 2-17
security, 2-20
usage suitability, 2-5
user interface, 2-15
plsql.conf file, A-96
pop-up list, LOV, A-66
portal developer, definition, xix
portal links, 6-9
Portal schema
password and Web Clipping, B-20
Portal Tools
OmniPortlet, 3-1
PORTAL_DEVELOPERS group, A-92
PORTAL.ACCOUNTINFO substitution tag, A-100
PORTAL.ACCOUNTINFO.LABEL substitution tag, A-100
PORTAL.ACCOUNTINFO.URL substitution tag, A-100
PORTAL.COMMUNITY substitution tag, A-100
PORTAL.COMMUNITY.IMAGE substitution tag, A-100
PORTAL.COMMUNITY.LABEL substitution tag, A-100
PORTAL.COMMUNITY.URL substitution tag, A-100
PORTAL.COMMUNITY.LABEL substitution tag, A-100
PORTAL.HELP substitution tag, A-100
PORTAL.HELP.IMAGE substitution tag, A-100
PORTAL.HELP.LABEL substitution tag, A-100
PORTAL.HELP.URL substitution tag, A-100
PORTAL.HOME substitution tag, A-100
PORTAL.HOME.IMAGE substitution tag, A-100
PORTAL.HOME.LABEL substitution tag, A-100
PORTAL.HOME.URL substitution tag, A-100
PORTAL.LOGOUT substitution tag, A-100
PORTAL.LOGOUT.LABEL substitution tag, A-100
PORTAL.LOGOUT.URL substitution tag, A-100
PORTAL.NAVIGATOR substitution tag, A-100
PORTAL.NAVIGATOR.IMAGE substitution tag, A-100
PORTAL.NAVIGATOR.LABEL substitution tag, A-100
PORTAL.NAVIGATOR.URL substitution tag, A-100
portlets
about, 1-1
using data sources, 3-3
portlet
refresh, 7-23
Portlet Builder
building portlets with, A-1
caching style, 2-13
charting, 2-18
comparing to other portlet builders, 2-2
development tool, 2-13
event support, 2-20
expertise required, 2-7
Index-9
URL types, 7-16
URLs, A-11
verification service for WSRP, 6-37
View Source privilege, A-75
viewing call interface, A-72
viewing package spec and body, A-72
viewing source code, A-71
Web Clipping, 1-6
Web source, 1-1
XML component, A-11
preference store, 7-9
creating and accessing in PL/SQL, 8-16
migration utility, 7-9
PL/SQL, 8-15
PrefStoreTransporter, 7-60
Preview mode, 6-6
private portlet parameters, 2-18
privileges, 7-39
access to shared components, A-92
Create shared components, A-92
defining for items, 15-6
for building portlets, A-75
granting on portlets, A-75
granting to individuals, A-76
inheriting from provider, A-75
Manage shared components, A-92
portlet, 7-39
provider, 7-39
retrieving, 15-1
properties
Web Clipping portlet, 5-9, 5-18
provider_record, 8-52
ProviderInstance, 7-49
providers, A-1
architecture, 2-10
database providers, 2-9
group not created, B-14
inheriting privileges from, A-75
locally built providers, A-7
logging levels, B-3
manually packaging, D-1
package, PL/SQL, 8-11
Portal DB Provider, A-15
portlet deployment, 2-7
privileges, 7-39
registering for JPS, 6-36
registering for PDK-Java, 6-53
registering programmatically, 8-52
registration, 2-11
runTest, E-3
schemas, A-1
servert for invalidation based caching, 7-65
shared components, A-91
test harness, E-2
test page, E-1
provider.xml
activating invalidation based caching, 7-67
events, 7-25
multi-lingual, 7-81
parameters, 7-15
portlet resource bundles, 7-82
preference information, 7-12
provider resource bundles, 7-81
session, 7-35
updating for render modes, 7-6
provsyns.sql script, 9-4
proxy authentication
OmniPortlet, 3-4
URL-based portlets and, 5-26
Web Clipping and, 1-8, 5-2, B-20
proxy server setting
Web Clipping and, B-19
proxy, resource, 6-9
ptlwsrp_data.sql, 6-19, 6-22
public portlet parameters, 2-18
Q
query wizard, charts, A-49
queues
registering a subscriber, 16-3
WWSBR_EVENT_ERR_Q, 16-6
WWSBR_EVENT_Q, 16-2
R
radio group, LOV, A-66
refresh
partial page, 7-23
registering
JPS portlet, 6-36
PDK-Java portlet, 6-53
providers programmatically, 8-52
troubleshooting, B-4
reject API, 16-27
remove_group_acl API, 15-4
remove_user_acl API, 15-4
render modes, 7-5
upating provider.xml for, 7-6
report portlets, A-11
audio data, A-46
building, A-27
column break display options, A-39
column conditions, A-31
column formatting options, A-33
column common display options, A-37
customization forms, A-40, A-42
data, A-46
formatting conditions, A-35
full Web page display options, A-38
image data, A-46
interMedia rich content, A-46
layout types, A-32
mobile device display options, A-40
number format masks, A-34
PL/SQL code options, A-45
row order display options, A-39
supplementary text options, A-44
templates, A-43
video data, A-47
resource
  proxy, 6-9
resource bundles, 7-78
  updating renderer, 7-79
resource links, 6-9
reverse proxy
  Web Clipping and, B-20
roles, A-6
runTest, E-3

S

schemas, A-1, A-2
  access privileges, A-4, A-6
  creating, A-2
  default profile, A-3
  default tablespace, A-3
  enrolling in roles, A-6
  granting privileges, A-4
  naming, A-2
  temporary tablespace, A-3
script
  ptlwsrp_data.sql, 6-19, 6-22
searching
  all page groups, 13-2
  items, 13-2
  limiting results returned, 13-3
  pages, 13-3
  specific attributes, 13-4
  specific page groups, 13-3
sections
  Web Clipping and, B-20
secure content repository views, 9-3, F-2
security, 7-35
  authentication, 7-35
  authorization, 7-36
  coding in PL/SQL, 8-29
  communication, 7-36
  features, 7-35
  for export, 7-56
  for import, 7-56
  for transport communications, 7-56
  guidelines for PL/SQL, 8-28
  implementing PL/SQL, 8-27
  managers, 7-40
  message authentication, 7-42
  server, 7-41
services
  default, D-4
  identifier, D-2
  name, D-2
session
  context, 10-1
  language, 14-2
  persistent states, A-88
  session information, 7-32
  checking for valid session, 7-34
  enabling in provider.xml, 7-35
  storage implementation, 7-33
session store
  creating and accessing in PL/SQL, 8-20
  PL/SQL, 8-20
set_attribute API, 11-7
set_context API, 10-1
set_group_acl API, 15-3
set_nls_language API, 14-2
set_user_acl API, 15-3
set_value, A-86
shared components, A-86
  creating JavaScript under, A-91
  editing, A-91
  granting access, A-92
shared components provider
  creating color definitions, A-95
  creating image definitions, A-95
  font definitions, A-96
  user interface templates, A-97
Shared Screen mode, 6-2
Show modes
  list, 6-2
Show page, 7-12
Simple Object Access Protocol (SOAP), 2-7
Single Sign-On
  Web Clipping and, 1-7, 5-2, 5-10
  single sign-on, 7-37
    comparison of portlet builders, 2-21
    external application, 2-21, 7-38
    partner application, 7-37
SOAP, 2-7
specify_attributes API, 13-4
spreadsheet
  building a portlet based on a spreadsheet, 4-5
  building an OmniPortlet, 4-5
  using as a data source, 3-6
SQL
  using as a data source, 3-6
SSL, 7-43
  configuration, 7-43
starter provider sample, 8-9
status
  checking for Web Clipping, B-18
strings
  loading language PL/SQL, 8-45
  retrieving language PL/SQL, 8-46
struts
  building portlet in JDeveloper, 7-83
  creating a portlet, 7-87
  OracleAS Portal integration, 7-86
  overview, 7-85
subscribers
  creating, 16-8
  external workflow example, 16-34
  item notification example, 16-22
  item verification example, 16-27
  portal object event logging example, 16-20
  registering, 16-3, 16-9
  running, 16-11
substitution tags, A-99
  ALIGN_LEFT, A-99
  ALIGN_RIGHT, A-100

Index-12
Tables

Default, A-3
Temporary, A-3

Tabs

Events, G-7
Tags, Substitution, A-99
Technologies, Portlets, 2-1
Templates, A-43
Events, G-12

See also user interface templates, A-97

Terminology, 9-3
Test Harness, E-2
Test Page
Provider, E-1
Web Clipping, B-18
Testing
Portlet and Provider for PDK-Java, 6-49
Portlets, A-77
Text Options
Chart Portlets, A-62
Form Portlets, A-24
Report Portlets, A-44
TextArea Item Types, Form Portlets, A-19
TextBox Item Types, Form Portlets, A-19
The, 4-3
Time Out Field
Web Clipping and, 5-9
Timeout
Troubleshooting, B-9
Translating
Creating, 14-3
Transport
By Reference, 7-57
By Reference Example, 7-60
Customizations, 7-47
Encryption for, 7-57
Encryption Example, 7-57
Example, 7-50
JNDI Variable, 7-57
Logging Interface, 7-50
Programming Interface, 7-48
Securing Communications, 7-56
Security, 7-56
Troubleshooting
Adding a Portlet to Your Page, B-9
Adding a Portlet to Your Provider, B-7
Installation and Deployment, B-3
Java Portlet Wizard, B-11
Java Portlets, B-2
OmniPortlet, B-15
Portlet Does Not Compile, B-11
Portlets Not Displaying, B-13
Provider Groups, B-14
Registration, B-4
Template.EAR, B-4
Timeout, B-9
URL Portlet, B-14
Web Clipping, B-18
XML Parser, B-8

Undelete_Item API, 11-18
Upload_Blob API, 12-6
URL Rewriting Field
Web Clipping and, 5-9
URL-Based Portlets
About, A-11
Migrating to Web Clipping, 1-8, 5-2
URLs
  in forms, 7-19
  parameters, 7-16
  specifying for Web Clipping, 5-5, 5-13
types for portlets, 7-16
UrlUtils.constructLink, 7-18
user interface templates, A-97
  structured, A-97
  substitution tags, A-99
  unstructured, A-98
USER substitution tag, A-99
USER.FULLNAME substitution tag, A-99
users
  page designer, xix
  portal developer, xix
  portlet developer, xix
UTF-8 character sets
  Web Clipping and, 5-2

V
validation
  field-level, A-93
  form-level, A-93
  JavaScript guidelines, A-93
  JavaScripts, A-93
  writing field-level, A-93
  writing form-level, A-93
validation options, form portlets
  Default Value, A-22
  Default Value Type, A-22
  Field Level Validation, A-22
  Form Level Validation, A-22
  Format Mask, A-22
  Insertable, A-22
  Mandatory, A-22
  Updatable, A-22
validation-based caching, 2-12
variables
  declaring JNDI, 7-28
  JNDI, 7-28
  JNDI naming conventions, 7-28
  JNDI provided by PDK-Java, 7-31
  JNDI types, 7-28
  retrieving JNDI variables, 7-30
  setting values for JNDI, 7-29
variables, persistent state, A-88
verification service
  WSRP, 6-37
version management, A-73
VERSION substitution tag, A-99
versions
  new, 11-13
video data, A-47
View mode, 6-2
View Source privilege, A-75
views
  providing access to, 9-4
  supported, 9-3, F-2
  WWSBR_ALL_CATEGORIES, F-3
  WWSBR_ALL_CONTENT AREAS, F-4
  WWSBR_ALL_FOLDER_REGIONS, F-4
  WWSBR_ALL FOLDERS, F-5
  WWSBR_ALL_ITEMS, 10-4, F-8
  WWSBR_ALL_NAVIGATION BARS, F-12
  WWSBR_ALL_PERSPECTIVES, F-13
  WWSBR_ALL STYLES, F-13
  WWSBR_APPROVER, F-13
  WWSBR_ATTRIBUTES, F-14
  WWSBR_CONTENT_AREA_APPROVAL, F-15
  WWSBR_CONTENT_AREA_ITEM_TYPES, F-15
  WWSBR_FOLDER_ATTRIBUTES, F-16
  WWSBR_FOLDER_PERSPECTIVES, F-16
  WWSBR_FOLDER_TYPE_ATTRIBUTES, F-17
  WWSBR_FOLDER_TYPES, F-18
  WWSBR_ITEM_APPROVAL, F-18
  WWSBR_ITEM_ATTRIBUTES, F-20
  WWSBR_ITEM_PERSPECTIVES, F-20
  WWSBR_ITEM_TYPE_ATTRIBUTES, F-21
  WWSBR_ITEM_TYPES, F-22
  WWSBR_SUBSCRIPTION, F-23
  WWSBR_USER FOLDERS, F-23
  WWSBR_USER_PAGES, F-26
virtual directories, images, A-96

W
WAR file, D-2
  creating manually, D-4
  for JPS, 6-35
  for PDK-Java, 6-51
Web browsers
  recommended cache settings, xxii
  recommended image settings, xxiii
  recommended versions, xxii
Web Clipping
  authentication for external application, 2-21
  caching style, 2-13
  capturing content, 2-16
  charting, 2-18
  comparing to other portlet builders, 2-2
  connection problems, B-19
  development tool, 2-13
  event support, 2-20
  expertise required, 2-6
  firewalls and, B-19
  forms and, 5-9, 5-19
  limitations, 5-26
  load balancers and, B-19
  logging levels and, B-18
  multi-lingual support, 2-21
  parameter support, 2-19
  personal pages and, 5-15
  proxy authentication, B-20
  proxy servers and, B-19
  rendering content inline, 2-17
  reverse proxy and, B-20
  searching, 5-16
  security, 2-20
  test page, B-18
troubleshooting,  B-18
usage suitability,  2-4
user interface,  2-15
Web Clipping portlet,  1-6, 5-1
adding,  5-2
properties,  5-9, 5-18
Web Clipping provider,  5-1
Web Clipping Studio using,  5-5
Web content adding to page,  5-2
browsing for,  5-5
expiration,  5-9
reuse,  1-7, 5-1
specifying,  5-5, 5-13
timeout value,  5-9

Web page building a portlet based on an existing Web page,  4-10
building an OmniPortlet,  4-10
using as a data source,  3-11

Web providers,  2-7
Web Service building an OmniPortlet,  4-3, 4-25
using as a data source,  3-10
Web Services for Remote Portlets
see WSRP
Web source,  1-1
web.xml file
for Web Clipping,  B-20
WSRP,  2-2, 2-7, 6-12
verification service,  6-37
WSRP portlet preference store,  6-22
WSRP producers,  2-8
wwctx_api
context information,  8-26
WWCTX_API package
  get_nls_language API,  9-8, 14-2
  set_context API,  10-1
  set_nls_language API,  14-2
WWPRO_API_INVALIDATION package
  execute_cache_invalidation API,  9-5
wwpro_api_provider_registry.register_provider,  8-53
WWSBR_ALL_CATEGORIES view,  F-3
WWSBR_ALL_CONTENTAREAS view,  F-4
WWSBR_ALL_FOLDER_REGIONS view,  F-4
WWSBR_ALL_FOLDERS view,  F-5
WWSBR_ALL_ITEMS view,  10-4, F-8
WWSBR_ALL_NAVIGATION_BARS view,  F-12
WWSBR_ALL_PERSPECTIVES view,  F-13
WWSBR_ALL_STYLES view,  F-13
WWSBR_API package
  add_category API,  12-3
  add_content_area API,  12-1
  add_folder API,  12-2
  add_item API,  12-4, 12-5
  add_item_ils_privileges API,  15-6
  add_item_post_upload API,  12-6
  add_perspective API,  12-3
approve API,  16-27
check_in_item API,  11-12
check_out_item API,  11-12
clear_cmeef_context API,  9-6
copy_folder API,  11-16
copy_item API,  11-16
delete_folder API,  11-19
delete_ils_privilege API,  15-7
delete_item API,  11-17
enable_ils_for_item API,  15-5
inherit_folder_privileges API,  15-8
modify_folder API,  11-5
modify_item API,  11-11
p_addnewversion,  11-13
modify_item_post_upload API,  12-8
move_category API,  11-15
move_folder API,  11-14
move_item API,  11-14
move_perspective API,  11-15
reject API,  16-27
set_attribute API,  11-7
undelete_item API,  11-18
upload_blob API,  12-6
WWSBR_APPROVER view,  F-13
WWSBR_ATTRIBUTES view,  F-14
WWSBR_CONTENTAREA_APPROVAL view,  F-15
WWSBRCONTENTAREAITEMTYPES view,  F-15
WWSBR_EVENT_ERR_Q queue,  16-6
WWSBR_EVENT_Q queue
dequeuing,  16-4
enqueuing,  16-2
WWSBRFOLDERATTRIBUTES view,  F-16
WWSBRFOLDERPERSPECTIVES view,  F-16
WWSBRFOLDERTYPEATTRIBUTES view,  F-17
WWSBRFOLDERTYPES view,  F-18
WWSBRITEMAPPROVAL view,  F-18
WWSBRITEMATTRIBUTES view,  F-20
WWSBRITEMPERSPECTIVES view,  F-20
WWSBRITEMTYPEATTRIBUTES view,  F-21
WWSBRITEMTYPES view,  F-22
WWSBRSUBSCRIPTION view,  F-23
WWSBRUSERFOLDERS view,  F-23
WWSBRUSERPAGES view,  F-26
WWSEC_API package
  grantee_list API,  15-1
  remove_group_acl API,  15-4
  remove_user_acl API,  15-4
  set_group_acl API,  15-3
  set_user_acl API,  15-3
WWSRC_API package
  get_all_items_xml API,  13-7
  get_item_xml API,  13-8
  item_search API,  13-7
  page_search API,  13-3
  specify_attributes API,  13-4
XML
  building a portlet based on XML, 4-8
  building an OmniPortlet, 4-8
  component portlets, A-11
  for PL/SQL Generator, 8-4
  troubleshooting, B-8
  using as a data source, 3-9
XML provider definition
  see provider.xml