WebSphere. software

IBM

IBM WebSphere Everyplace Access V5 Handbook for Developers and Administrators

Volume II: Application Development

- Develop mobile applications with offline capabilities
- Develop online portlet applications for PDAs
- Adapt content to multiple markup languages



Juan R. Rodriguez
George Kroner
Gianfranco Rutigliano
Sung-Ik Son
Guillermo Villavicencio

Redbooks

ibm.com/redbooks

Petitioner has added letters to pages (a)-(c) and numbers to pages 455-456. Otherwise, it leaves the original page numbering.

SAP AMERICA, INC.
EXHIBIT 1119

(a)



International Technical Support Organization

IBM WebSphere Everyplace Access V5 Handbook for Developers and Administrators Volume II: Application Development

March 2005

SG24-6463-00

(b)

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (March 2005)

This edition applies to Version 5 of IBM WebSphere Everyplace Access for multiplatforms.

© Copyright International Business Machines Corporation 2005. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices ix
Trademarks x

Preface xi
The team that wrote this redbook. xii
Become a published author xiii
Comments welcome. xiii

Part 1. Online applications. 1

Chapter 1. My first pervasive portlet application 3
1.1 Overview 4
1.2 Create a portlet for HTML, WML, and PDA markups 4
1.3 Examine the relevant files and source code 11
1.4 Run the portlet. 21

Chapter 2. Portlet action event handling. 33
2.1 Overview 34
2.2 Setup the ActionEvent portlet framework 36
2.2.1 Create the framework with the portlet application wizard 36
2.2.2 Configure the test environment to support this project 39
2.3 Examine and modify source code 44
2.3.1 Examine the model and controller components 44
2.3.2 Examine the HTML source generated by the wizard 47
2.3.3 Modify the source for PDA markup 49
2.3.4 Modify the source for WML markup 50
2.4 Run the project 52

Chapter 3. Portlet messaging. 63
3.1 Overview 64
3.2 Setup the MessageEvent portlet framework 65
3.2.1 Create the framework with the portlet application wizard 65
3.2.2 Configure the test environment to support this project 68
3.3 Examine and modify source code 73
3.3.1 Examine and modify the model and controller components 73
3.3.2 Examine and modify the HTML source generated by the wizard . . . 75
3.3.3 Modify the source for PDA markup 76
3.3.4 Add view support for the message sender portlet for PDAs 78
3.4 Run the project 79

© Copyright IBM Corp. 2005. All rights reserved.

Chapter 4. National Language Support (NLS) 87
4.1 Resource bundles 88
4.1.1 Creating resource bundles in WebSphere Studio 90
4.1.2 Translating resource bundles 92
4.1.3 Accessing resource bundles in portlets. 94
4.1.4 Accessing resource bundles in JSPs 95
4.2 Translating whole resources 97
4.3 NLS administration 99
4.3.1 Portlet NLS administration 99
4.3.2 Portal NLS administration 102
4.3.3 Setting NLS titles. 102
4.3.4 Adjusting Portal resource bundles 103
4.4 Working with characters 104
4.5 NLS best practices 104
4.6 Sample scenario: NLS bundles 105
4.6.1 NLS bundles 106
4.6.2 Accessing NLS bundles from JSPs. 110
4.6.3 Running the NLS scenario 113
4.6.4 Accessing NLS bundles in Java portlets 120
4.6.5 Adding PDA markup support. 122
4.7 Sample scenario: translating whole resources 125
4.7.1 Adding PDA markup support. 130

Chapter 5. Developing a database access portlet 135
5.1 Create a test database and populate the table 136
5.2 Use the wizard to generate the portlet 137
5.3 Use the Portal Test Environment to test the portlet. 152
5.4 Export the portlet application. 157
5.5 Deploy to your WEA/Portal server. 159
5.6 Execute the portlet from a Pocket PC 173
5.7 Use proper driver if the database is not DB2 175

Chapter 6. Using the Credential Vault 181
6.1 Overview 182
6.2 Sample scenario 185
6.2.1 Importing a protected servlet application 187
6.2.2 Using active credentials 190
6.2.3 Updating the generated portlet 195
6.2.4 Running the portlet 198
6.2.5 Using passive credentials 204

Chapter 7. Accessing Web Services 207
7.1 Overview 208
7.2 A simple Web Service project 209

7.2.1 A sample Web Service 214
7.3 Creating a Web Services client portlet 226
7.4 Run the WSClientPortlet application 232

Part 2. Transcoding 237

Chapter 8. Transcoding Technology 239
8.1 Overview 240
8.1.1 Architecture 241
8.1.2 Preference profiles 243
8.1.3 XML stylesheets 246
8.1.4 Annotators 246
8.1.5 Transcoding plug-ins 247
8.2 XML configuration utility 248
8.3 Request Viewer 249
8.3.1 How to start Request Viewer 252
8.4 Logging and tracing 255
8.4.1 Message files 257
8.4.2 Trace files 258
8.4.3 Gather troubleshooting data 258
8.5 A simple portlet using Transcoding Technology 258
8.5.1 Enable transcoding 258
8.5.2 Use Request Viewer to monitor the process 263

Chapter 9. A sample scenario using Transcoding Technology 265
9.1 Overview 266
9.2 WebSphere Everyplace Access configuration 268
9.3 Text transcoding 268
9.4 Image transcoding 270
9.5 Fragmentation portlet 271
9.6 Annotations 282
9.7 Using external annotators 293
9.8 Using stylesheets 294

Part 3. Offline applications 305

Chapter 10. Building a portlet for offline browsing 307
10.1 Build a portlet to fetch content from a Web site 308
10.2 Using a transcoding annotator plugin for Web clipping 311
10.3 Export the portlet to your local file system 314
10.4 Deploy and configure the portlet for offline access 315
10.5 Running the portlet online 319
10.6 Configure Everyplace Client for offline access 321
10.7 Running offline browsing on a PDA 323

Chapter 11. Building a portlet for offline form access 325
11.1 Building a form based portlet 326
11.2 Adding field validations 333
11.3 Adding a script to highlight the field in error 338
11.4 Deploying the sample portlet as an offline portlet 339
11.5 Running the portlet from a Pocket PC 342
11.6 Portlet offline configuration 344
11.7 Configure the portlet client with offline option 351

Chapter 12. DB2 Everyplace applications with WebSphere Studio Device Developer 355
12.1 Overview 356
12.1.1 Introduction 356
12.1.2 Java 2 Micro Edition 356
12.1.3 WebSphere Micro Environment 358
12.2 Installing WebSphere Studio Device Developer 358
12.3 Working with WebSphere Studio Device Developer 359
12.3.1 Using the workbench 359
12.3.2 Using the Update Manager 360
12.4 Sample scenario 364
12.4.1 Define the database tables for the application 364
12.4.2 Configure the Pocket PC device for development 367
12.4.3 Create the project on WebSphere Studio Device Developer 369
12.4.4 Write the Java classes of the application 372
12.4.5 Creating the device database 401
12.4.6 Set up the build and test environment 402
12.4.7 Testing the application 409

Part 4. Appendixes 413

Appendix A. Portlet development platform installation 415
A.1 Prerequisites 416
A.2 WebSphere Studio Site Developer V5.1.1 416
A.2.1 Install WebSphere Studio Site Developer 417
A.2.2 Apply fixes to WebSphere Studio Site Developer 419
A.3 WebSphere Studio Device Developer 425
A.4 WebSphere Everyplace Toolkit V5.0.1 427
A.4.1 Begin Everyplace Toolkit installation 427
A.4.2 Install Portal Toolkit V5.0.2.1 428
A.4.3 Complete Everyplace Toolkit Installation 432
A.4.4 Complete installation of Micro Environment Tools 435

Appendix B. Additional material 443
Locating the Web material 443

Using the Web material 443
System requirements for downloading the Web material 444
How to use the Web material 444

Related publications 445
IBM Redbooks 445
Other publications 446
Online resources 446
How to get IBM Redbooks 447
Help from IBM 447

Index 449

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

© Copyright IBM Corp. 2005. All rights reserved.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

ibm.com®	Lotus Notes®	Tivoli®
DB2 Universal Database™	Lotus®	WebSphere®
DB2®	Notes®	Workplace™
Everyplace®	Redbooks™	Workplace Client Technology™
IBM®	Redbooks (logo)  ™	

The following terms are trademarks of other companies:

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel Inside (logos) and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Preface

This IBM® Redbook is part of a series of four volumes related to WebSphere® Everyplace® Access Version 5.0. These Redbooks will help you plan, install, administer and develop mobile applications to run in a WebSphere® Everyplace® Access Version 5.0 environment. The information provided in these Redbooks target business-to-employee (B2E) enterprise applications, but most of the scenarios presented will apply to business-to-consumer (B2C) applications as well. In these Redbooks, you will find step-by-step examples and scenarios showing ways to integrate your enterprise applications into a WebSphere Everyplace Access environment using the WebSphere Studio Site Developer and the Everyplace Toolkit as well as extending your online and offline application capabilities to use other advanced functions.

In this redbook you will also find step-by-step sample scenarios illustrating procedures to develop online applications using WebSphere Studio Site Developer and WebSphere Application Site Developer along with the WebSphere Everyplace Toolkit. Sample scenarios also include offline applications using Offline Portal Content and Forms, Transcoding and DB2e applications using the WebSphere Studio Device Developer.

A basic knowledge of Java™ technologies, such as servlets, JavaBeans, EJBs, JavaServer Pages (JSPs), as well as XML applications and the terminology used in Web publishing, is assumed.

The following IBM WebSphere Everyplace Access V5 Redbooks are part of this offering:

- *IBM WebSphere Everyplace Access V5 Handbook for Developers and Administrators Volume I: Installation and Administration*, SG24-6462
- *IBM WebSphere Everyplace Access V5 Handbook for Developers and Administrators Volume II: Application Development*, SG24-6463
- *IBM WebSphere Everyplace Access V5 Handbook for Developers and Administrators Volume III: E-Mail and Database Synchronization*, SG24-6676
- *IBM WebSphere Everyplace Access V5 Handbook for Developers and Administrators Volume IV: Advanced Topics*, SG24-6677

© Copyright IBM Corp. 2005. All rights reserved.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.



Juan R. Rodriguez is a Consultant at IBM ITSO, Raleigh Center. He received his M.S. degree in Computer Science from Iowa State University. He writes extensively and teaches IBM classes worldwide on such topics as networking, pervasive computing, Web technologies, and information security. Before joining the IBM ITSO, he worked at IBM Research Triangle Park, North Carolina as a designer and developer of networking products.



George Kroner is a is a Co-op IT Specialist at the IBM ITSO Center in Raleigh, North Carolina. He is currently pursuing a Bachelor of Science degree in Information Sciences and Technology at Pennsylvania State University. His interests include Web applications, wireless/mobile computing, and intelligent interfaces.



Gianfranco Rutigliano holds a degree in Systems Engineering from the University of Lima (Peru) and has been a member of IBM Global Services for four years. He has considerable experience designing and implementing pervasive and Java-based Web solutions. He is currently a Software Architect for Avatar e-Business Solutions, an IBM Business Partner, and he has participated in several ITSO residencies for pervasive computing and WebSphere technologies.



Sung-Ik Son is a Senior Software Engineer for IBM Software Services at the IBM Raleigh Lab. He has been a developer in IBM system and application development organizations, and currently is a member of the WebSphere Enablement and Consulting team. He holds a B.S. in Computer Science from the University of New Brunswick in Canada and an M.S. in Computer Science from Purdue University.



Guillermo Villavicencio holds a degree in Informatics Engineering from the Pontifical Catholic University of Peru and works as a Software Architect for Avatar e-Business Solutions, an IBM Business Partner. He has been the architect for several e-business projects including wireless and Portal solutions. His current area of expertise is centered around Web technologies and pervasive computing.

Thanks to the following people for their contributions to this project:

Margaret Ticknor
International Technical Support Organization, Raleigh Center

Mary Fisher
IBM Boca Raton, Florida, USA

Lindamay Patterson
IBM Rochester, Minnesota, USA

Gabrielle Velez
International Technical Support Organization, Rochester Center

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbook@us.ibm.com

- Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 662
P.O. Box 12195
Research Triangle Park, NC 27709-2195

Part 1

Online applications

In this part of the redbook we introduce online application development in WebSphere Everyplace Access V5. Online application development is accomplished by using the Everyplace Toolkit installed on a WebSphere Studio Site Developer or WebSphere Studio Application Developer environment.

Note: For a sample installation of the online application development environment see Appendix A, “Portlet development platform installation” on page 415.

© Copyright IBM Corp. 2005. All rights reserved.

1

My first pervasive portlet application

This chapter provides a sample scenario for developing your first pervasive portlet application. It will allow you to see how portlets function to display content in various markup languages.

The development environment used to create and run these portlets is described in Appendix A, “Portlet development platform installation” on page 415.

In addition, you should install and configure both the Palm OS V5.4 Simulator and the Nokia Mobile Internet Toolkit V4.1 with the Series 40 Developer Platform 2.0 SDK (based on the Nokia 6230 mobile phone). These simulators will allow you to test your applications for PDA and WML markup languages.

In this chapter, we will:

- Create a portlet application with support for HTML, PDA, and WML markup languages using the Everyplace Toolkit Portlet Application Wizard
- Examine the structure of a pervasive portlet application
- Run and test your application in the WebSphere Studio browser, the Palm OS Simulator, and the Nokia Series 40 Simulator

© Copyright IBM Corp. 2005. All rights reserved.

1.1 Overview

We will start by creating a portlet application using the WebSphere Studio Site Developer. This portlet is based on the Model-View-Controller (MVC) design pattern. The MVC methodology allows efficient relationships between the user interface and the underlying data model. The main components of this design pattern are:

- Model: The logical structure of data in a software application
- View: All elements in the user interface
- Controller: The elements connecting the Model and View elements

Figure 1-1 illustrates the MVC portlet application. The portlet that you will create has a HelloWorldPortletView.jsp file to act as the view component and a HelloWorldPortlet.java file to act as the controller. JavaBeans are mainly used for storing data from user input. They will be introduced in the Action Event chapter. Your portlet will not have a JavaBean. Additional JSPs are generated for the view component for portlets that support additional portlet modes (View, Edit, Help, and Configure).The sample HelloWorld portlet displays a Hello World message.

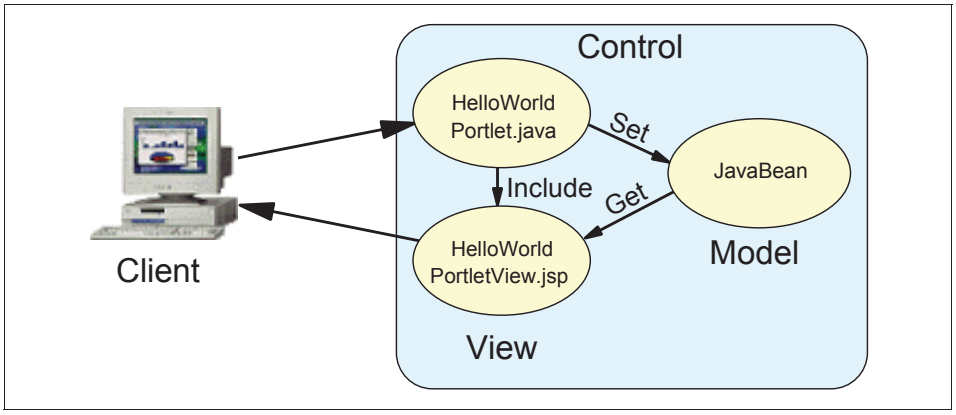


Figure 1-1 MVC architecture

1.2 Create a portlet for HTML, WML, and PDA markups

In this chapter, you will create a portlet application that can support multiple markup languages and test it in both the Palm OS Simulator and the Nokia cellular telephone simulator. The HelloWorld portlet displays a Hello World message.

Perform the following steps to create the portlet application:

1. Open the IBM WebSphere Studio Site Developer by clicking **Start -> Programs -> IBM WebSphere Studio -> Site Developer 5.1.1**.
2. If this is the first time that you are running WebSphere Studio, you will receive a dialog box that asks you to choose a workspace location. Some features of WebSphere Studio require short path names to your application development files in order to function correctly. It is recommended to choose a short path name such as C:\workspace. Optionally select **Use this workspace as the default and do not show this dialog box again**. Click **OK** to continue.

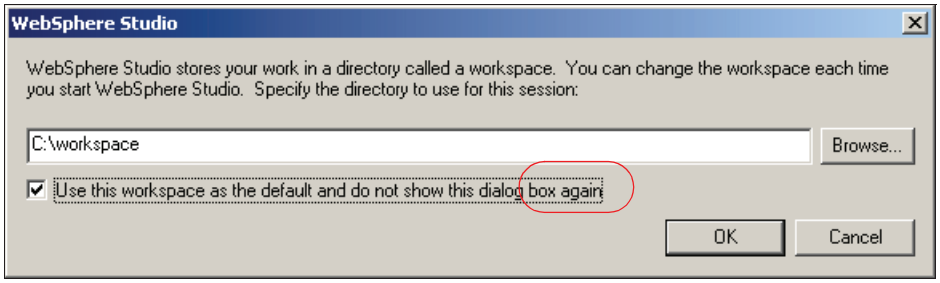


Figure 1-2 Setting your project workspace

Note: To reset the location of your workspace, run the following command from a command prompt where WebSphere Studio is installed:

```
wsappdev.exe -setworkspace
```

3. Select **File -> New -> Other**, as shown in Figure 1-3.

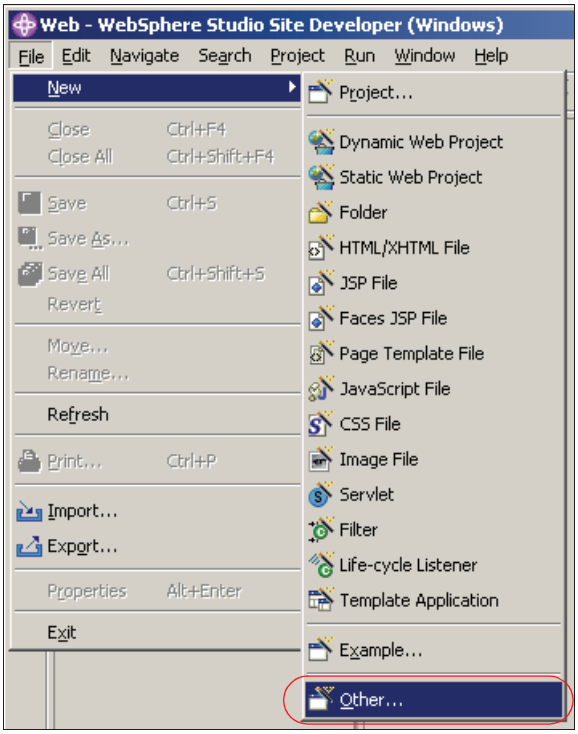


Figure 1-3 Starting your project

4. In the window shown in Figure 1-4 on page 7, select **Portlet Development** from the left pane, then **Portlet application project** from the right pane. Then click **Next**. This will start the Portlet Application Wizard project wizard.

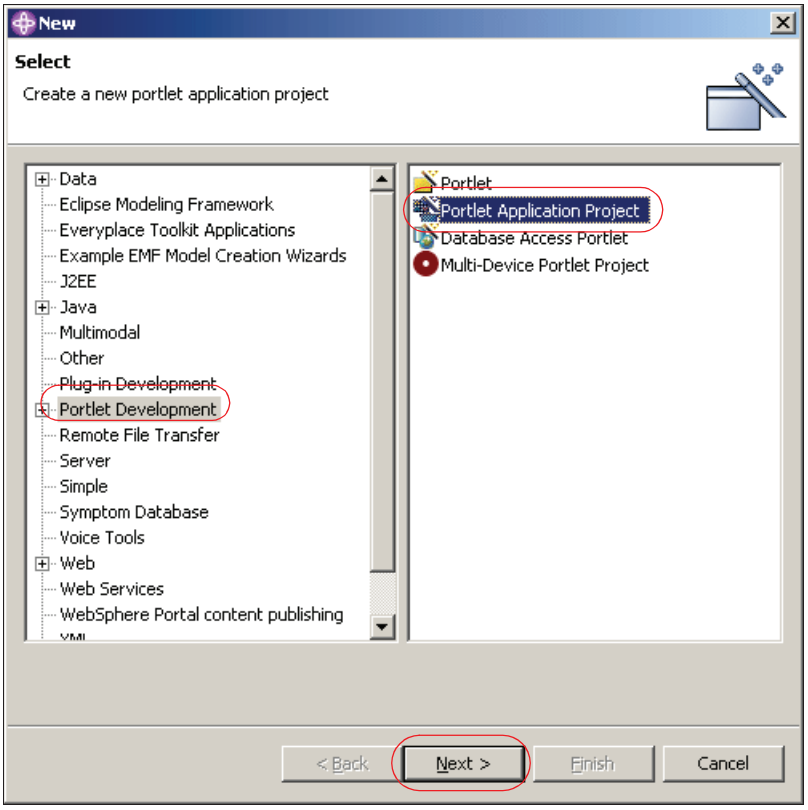


Figure 1-4 Selecting the portlet application project

5. On the Define the Portlet Project window, select **Create a basic portlet** and check **Configure advanced options**. Then click **Next**.

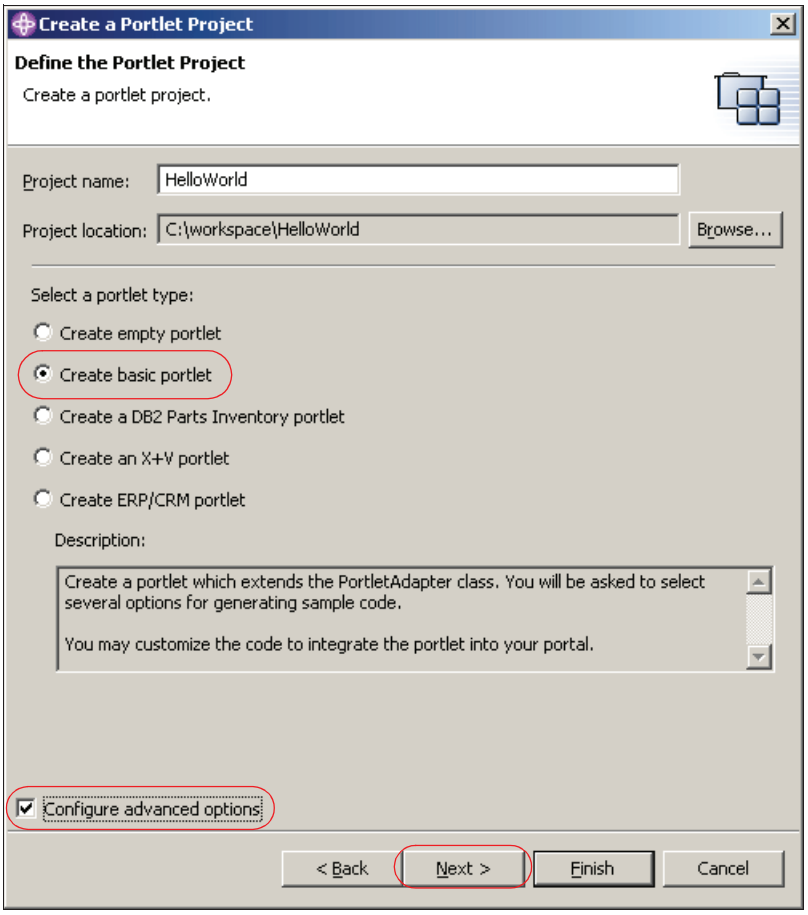


Figure 1-5 Define the portlet project

6. The J2EE level option will be set to **Level 1.3 / WebSphere Portal 5.0**. Accept this default value and click **Next** to continue.

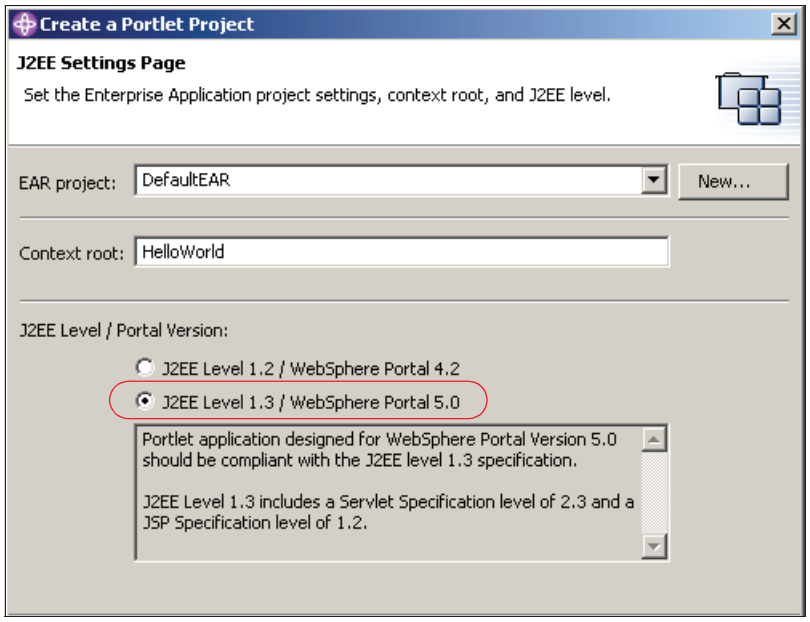


Figure 1-6 J2EE settings window

7. On the Portlet Settings window, accept the default values and click **Next**.

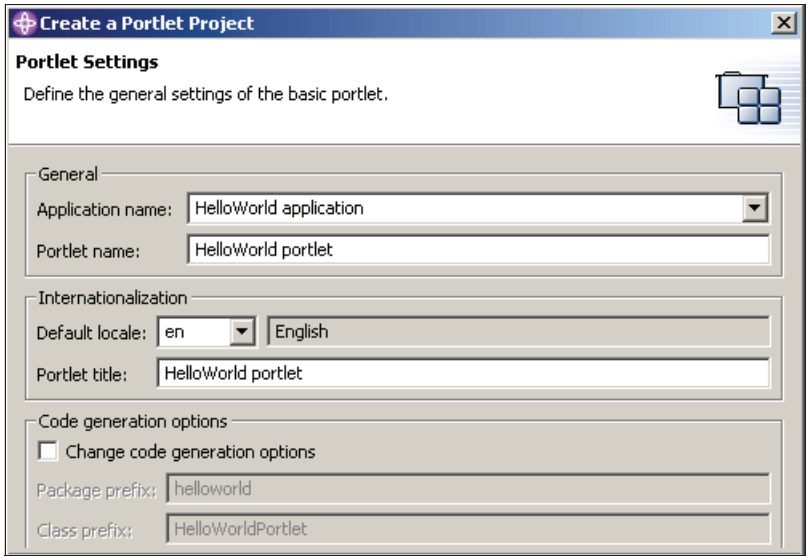


Figure 1-7 Portlet settings window

8. On the Event Handling window, uncheck **Add form sample** and **Add action listener**. Click **Next** to continue.

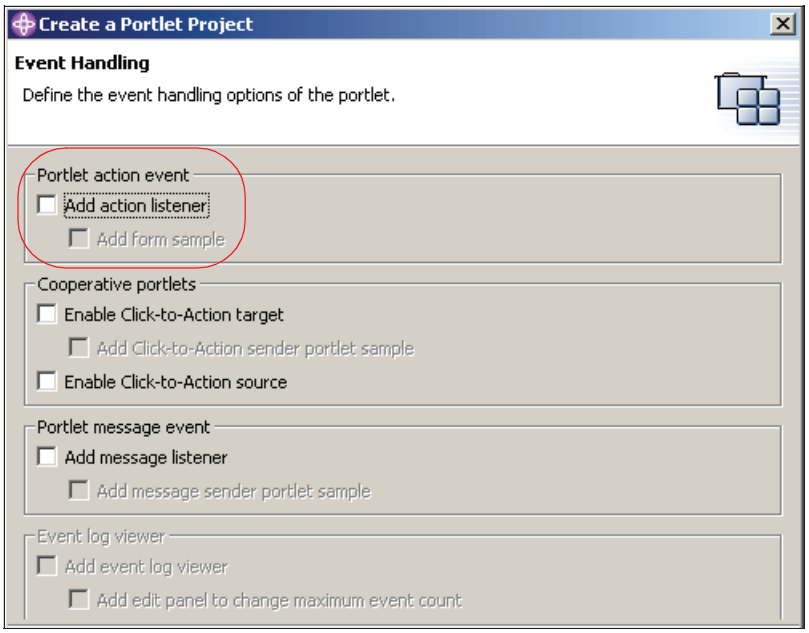


Figure 1-8 Event handling window

9. On the Single Sign-On window, accept the default values and click **Next**.

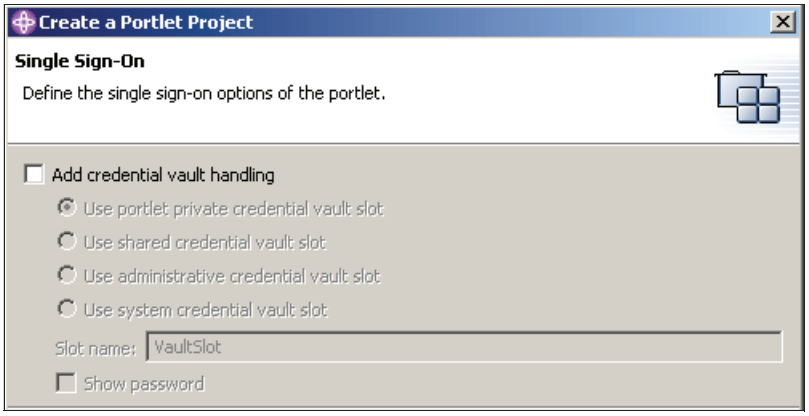


Figure 1-9 Single Sign-On window

10. On the Miscellaneous window, add **PDA** and **WML** markup support. Click **Finish** to complete the wizard.

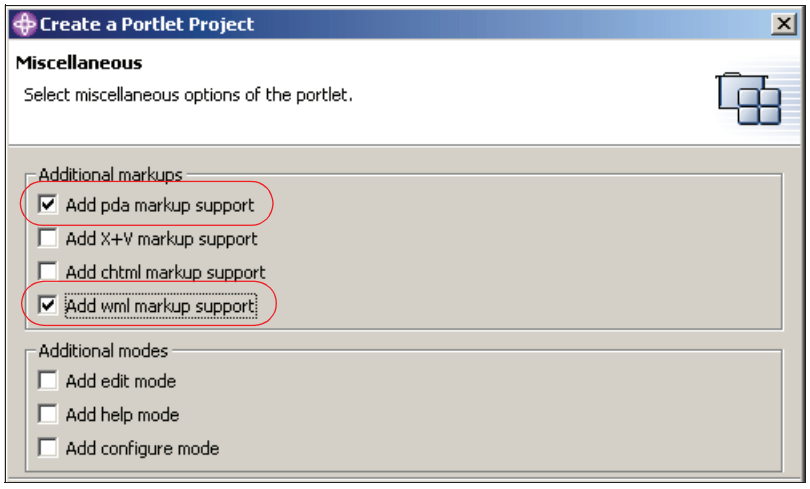


Figure 1-10 Miscellaneous window

11. You may receive a dialog box prompting you to switch to the Portlet Perspective to work on this project. Click **Yes** and optionally check **Do not show this message again** to continue.

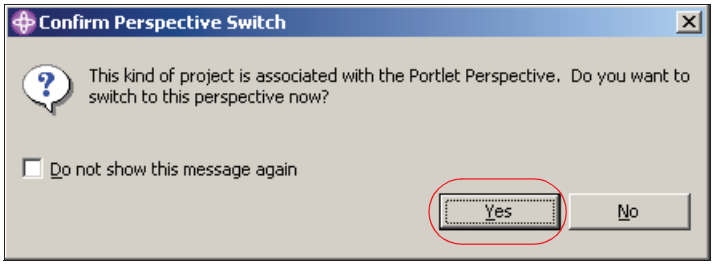


Figure 1-11 Perspective switch

1.3 Examine the relevant files and source code

Now that the framework for the project has been made, we will examine how this portlet works. Follow these instructions:

1. To begin, we will examine the files that are used to configure this portlet so that it can run on a portal server: portlet.xml and web.xml. These files are located in the Project Navigator panel in the upper left corner of the window. Double-click the portlet.xml file. This is the portlet deployment descriptor. It contains important information regarding what markups and modes the portlet supports.

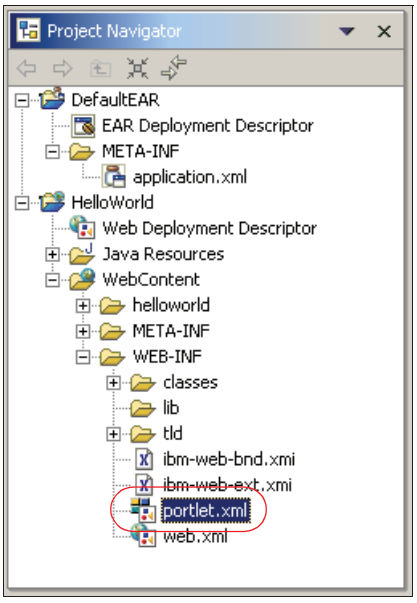


Figure 1-12 Portlet.xml in the File Navigator panel

The portlet deployment descriptor editor displays the portlet.xml values.

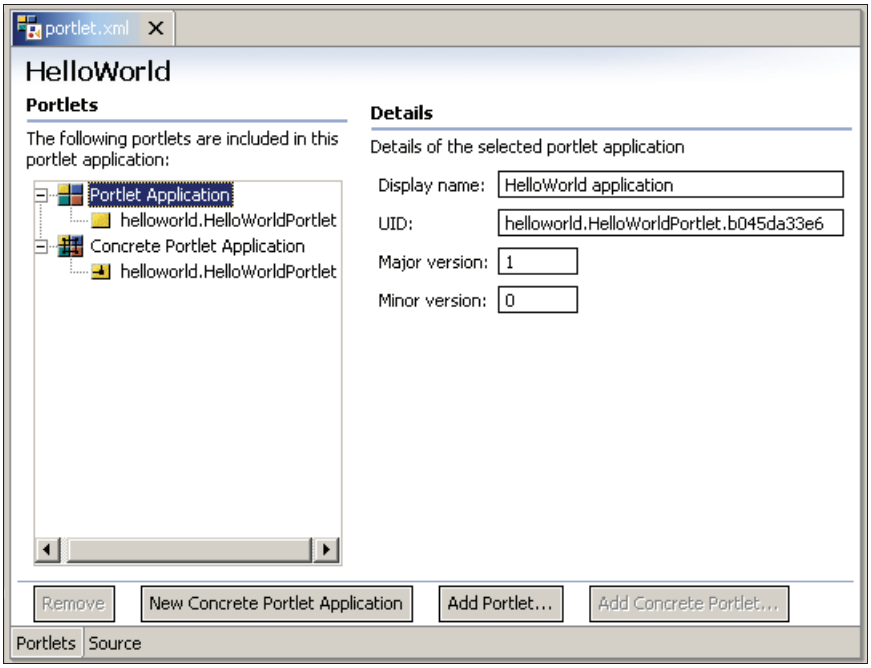


Figure 1-13 Portlet deployment descriptor editor

By selecting helloworld.HelloWorldPortlet under the Portlet Application, you can view the options related to this portlet such as supported markup languages and modes. If you were adapting an existing portlet to multiple languages and devices, this is where you would add that support.

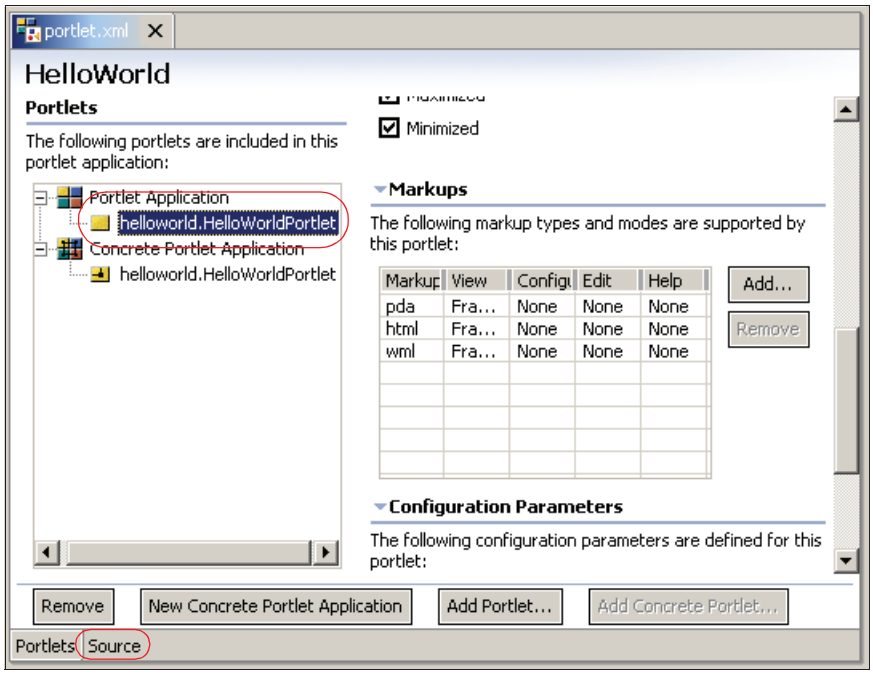


Figure 1-14 Portlet deployment descriptor editor

By selecting the source button in the bottom left corner of the portlet.xml editor, you can view the source of the portlet.xml file.

Example 1-1 Portlet.xml source

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE portlet-app-def PUBLIC "-//IBM//DTD Portlet Application 1.1//EN"
"portlet_1.1.dtd">
<portlet-app-def>
   <portlet-app uid="helloworld.HelloWorldPortlet.b045da33e6"
major-version="1" minor-version="0">
      <portlet-app-name>HelloWorld application</portlet-app-name>
      <portlet id="helloworld.HelloWorldPortlet"
href="WEB-INF/web.xml#helloworld.HelloWorldPortlet" major-version="1"
minor-version="0">
         <portlet-name>HelloWorld portlet</portlet-name>
         <cache>
            <expires>0</expires>
            <shared>no</shared>
         </cache>
         <allows>
            <maximized/>
            <minimized/>
```

```
            </allows>
            <supports>
               <markup name="pda">
                  <view />
               </markup>
               <markup name="html">
                  <view />
               </markup>
               <markup name="wml">
                  <view />
               </markup>
            </supports>
         </portlet>
      </portlet-app>
      <concrete-portlet-app uid="helloworld.HelloWorldPortlet.bO45da33e6.1">
         <portlet-app-name>HelloWorld application</portlet-app-name>
         <concrete-portlet href="#helloworld.HelloWorldPortlet">
            <portlet-name>HelloWorld portlet</portlet-name>
            <default-locale>en</default-locale>
            <language locale="en">
               <title>HelloWorld portlet</title>
               <title-short></title-short>
               <description></description>
               <keywords></keywords>
            </language>
            <config-param>
               <param-name>pda-icon</param-name>
               <param-value>/images/pda/default.gif</param-value>
            </config-param>
         </concrete-portlet>
      </concrete-portlet-app>
</portlet-app-def>
```

2. Now double-click the web.xml file in the Project Navigator panel. This is the Web deployment descriptor. It stores parameters related to how the portlet will run on an application server.

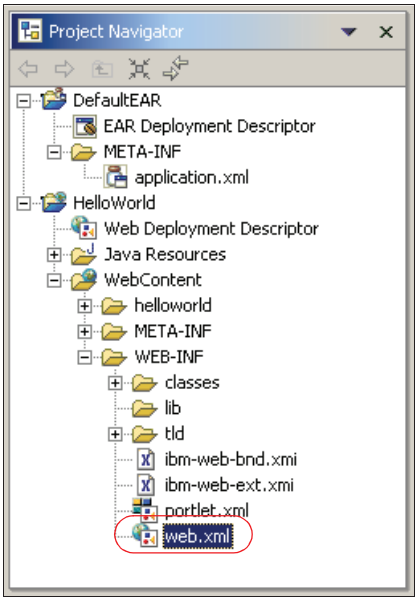


Figure 1-15 Web.xml in file navigator panel

The web.xml file will also display in its own editor.

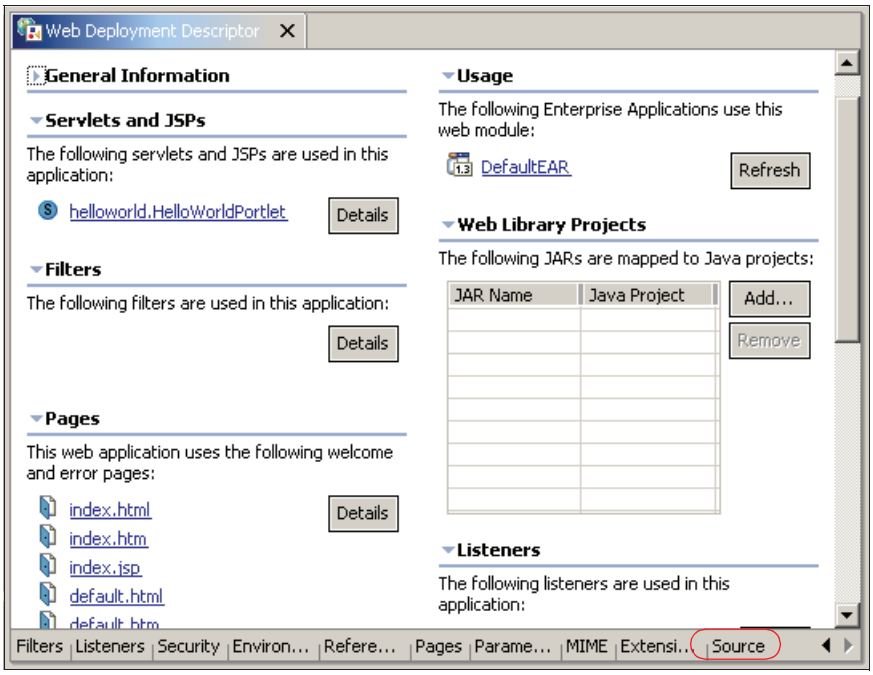


Figure 1-16 Web deployment descriptor editor

You can view the source to the web.xml file by clicking the source button in the lower right corner of the Web deployment descriptor editor.

Example 1-2 Web.xml source

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app id="WebApp">
   <display-name>HelloWorld</display-name>
   <servlet id="helloworld.HelloWorldPortlet">
      <servlet-name>helloworld.HelloWorldPortlet</servlet-name>
      <display-name>HelloWorldPortlet</display-name>
      <servlet-class>helloworld.HelloWorldPortlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>helloworld.HelloWorldPortlet</servlet-name>
      <url-pattern>/helloworld.HelloWorldPortlet/*</url-pattern>
   </servlet-mapping>
   <welcome-file-list>
      <welcome-file>index.html</welcome-file>
      <welcome-file>index.htm</welcome-file>
      <welcome-file>index.jsp</welcome-file>
```

```
        <welcome-file>default.html</welcome-file>
        <welcome-file>default.htm</welcome-file>
        <welcome-file>default.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

3. Expand the folders in the Project Navigator panel. Examine the structure of the files that are generated by the wizard. These files include a HelloWorldPortlet.java file that acts as the portlet's controller and a series of JSP files that are used to generate the portlet's output. These JSP files are located in folders named according to the markup languages they support.

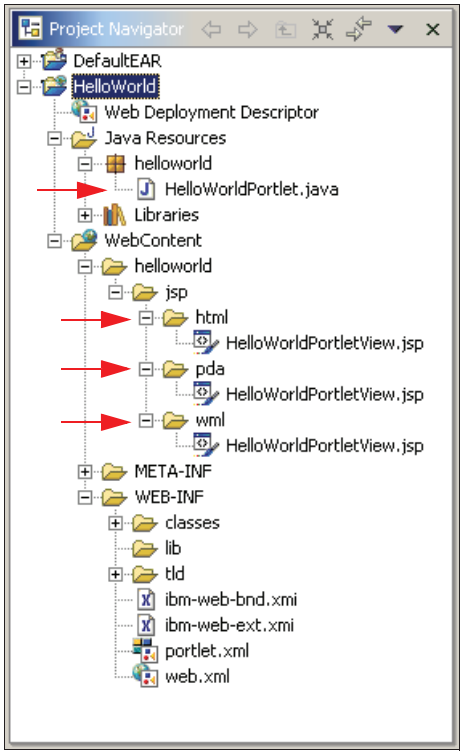


Figure 1-17 Project Navigator panel

4. Double-click the HelloWorldPortlet.java file to examine the file. HelloWorldPortlet.java acts as a controller to manage the functionality and output of the HelloWorld portlet.

   HelloWorldPortlet.java has a doView() method that is called to display the content of the portlet and a getJspExtension() method that is used to

determine which set of JSPs is used to display the content as according to the markup languages and characteristics supported by the requesting device.

Example 1-3 HelloWorldPortlet.java

```
package helloworld;
import java.io.IOException;
import org.apache.jetspeed.portlet.*;

public class HelloWorldPortlet extends PortletAdapter {

    public static final String VIEW_JSP =
"/helloworld/jsp/HelloWorldPortletView."; // JSP file name to be rendered on
the view mode

   public void init(PortletConfig portletConfig) throws UnavailableException {
      super.init(portletConfig);
   }

   public void doView(PortletRequest request, PortletResponse response) throws
PortletException, IOException {

   // Invoke the JSP to render
   getPortletConfig().getContext().include(VIEW_JSP+getJspExtension(request),
request, response);
   }

   private static String getJspExtension(PortletRequest request) {
      String markupName = request.getClient().getMarkupName();
      return "jsp";
   }
}
```

5. Now double-click the HelloWorldPortletView.jsp files that are located in the html, pda, and wml folders under the HelloWorld/WebContent/helloworld/jsp directory. These files are important when you need to adapt the same business logic to more than one markup language or device type. Each of these files opens in the page designer editor. You can click the **Source** tab at the bottom of each page as shown in Figure 1-18 to view its source.

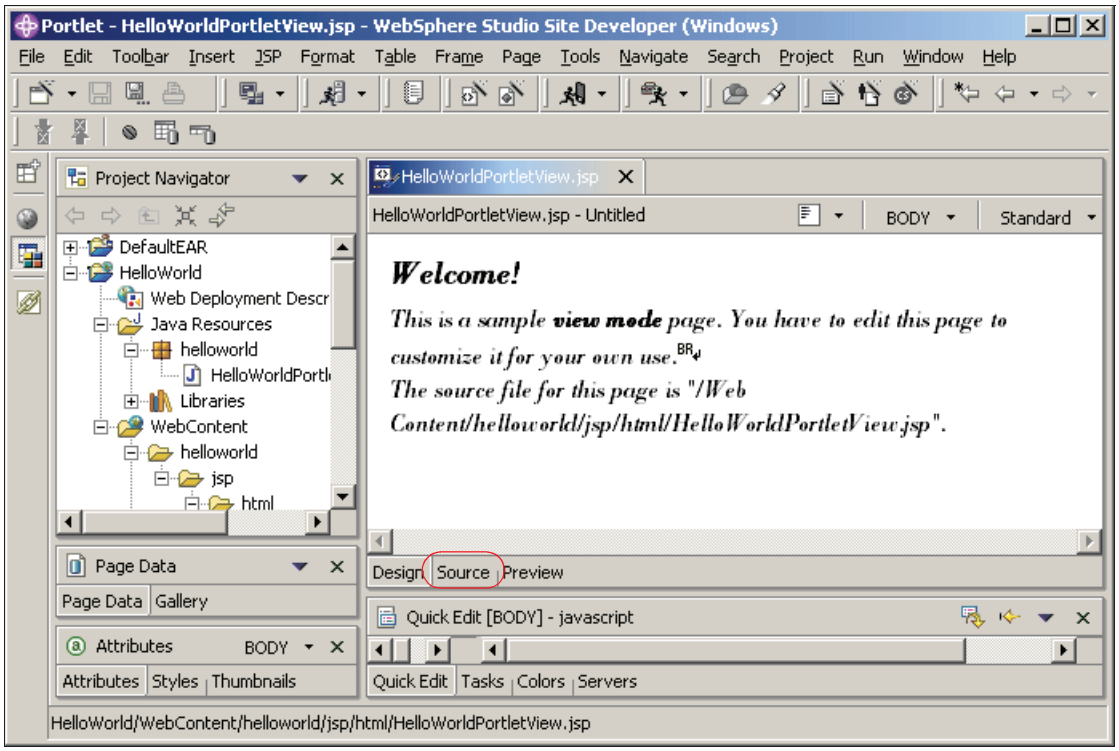


Figure 1-18 Viewing the source of the JSPs

The JSP located in the /html folder is used to generate the HTML source used when a Web browser views this portlet.

Example 1-4 HelloWorldPortletView.jsp - HTML

```
<%@ page session="false" contentType="text/html" import="java.util.*,
helloworld.*"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>

<DIV style="margin: 6px">

<H3 style="margin-bottom: 3px">Welcome!</H3>
This is a sample <B>view mode</B> page. You have to edit this page to customize
it for your own use.<BR>
The source file for this page is "/Web
Content/helloworld/jsp/html/HelloWorldPortletView.jsp".

</DIV>
```

The JSP in the /pda folder generates a simplified version of the HTML shown above. This JSP is used when PDAs access the portlet.

Example 1-5 HelloWorldPortletView.jsp - PDA

```
<%@ page session="false" contentType="text/html" %>
<P>
<B>Hello</B><BR>
This is a sample pda page
</P>
```

The JSP in the /wml folder is used to generate the content that is shown when cellular telephones are used to access the portlet. It has a different content type than the previous JSPs, and the markup used in this JSP, such as
, conforms to WML standards.

Example 1-6 HelloWorldPortletView.jsp - WML

```
<%@ page session="false" contentType="text/vnd.wap.wml" %>
<p>
<b>Hello</b><br/>
This is a sample wml page
</p>
```

It is important to know that each of these JSPs can use a JSP directive to import and use other Java packages that are bundled with the portlet and to use the WebSphere Portal tag library.

1.4 Run the portlet

1. If you have previously worked on any previous portlet projects, they may still exist in the DefaultEAR project. We suggest that you remove the other portlets from the EAR deployment descriptor before proceeding. You can do this by following these steps:
   a. Open your DefaultEAR/META-INF folder in the navigator pane.
   b. Double-click the application.xml file.
   c. Select the **Module** tab at the bottom of the Application deployment descriptor editor.
   d. Remove all WAR modules except for your portlet application, Web HelloWorld.
   e. Save any changes by clicking **File -> Save**.
   f. Click **OK** if you receive the Server Configuration window.

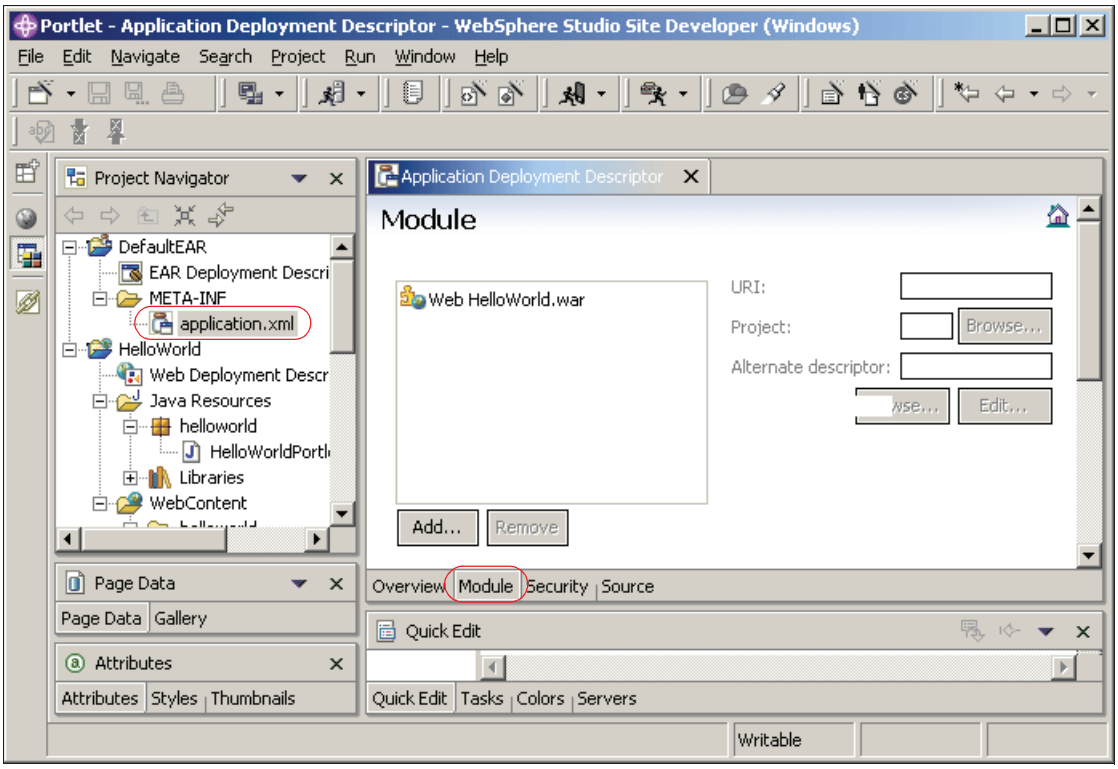


Figure 1-19 Removing WAR modules

2. You will need to create a test environment to execute your portlet application project. Additional instructions and recommendations on configuring a new test environment can be found in Appendix A, “Portlet development platform installation” on page 415.

   a. If this is your first time running a portlet application in WebSphere Studio, right-click the project name and select **Run on Server**.

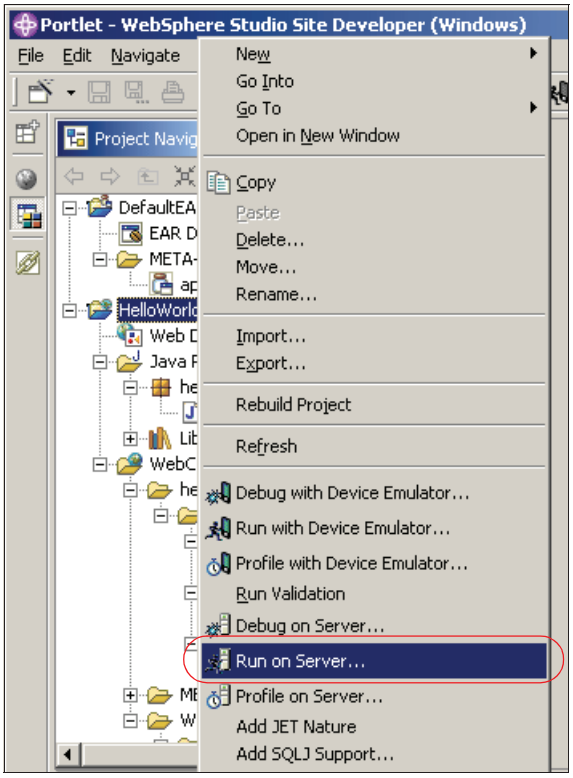


Figure 1-20 Run on server

b. Select WebSphere Portal v5.0 Test Environment and click **Next**.

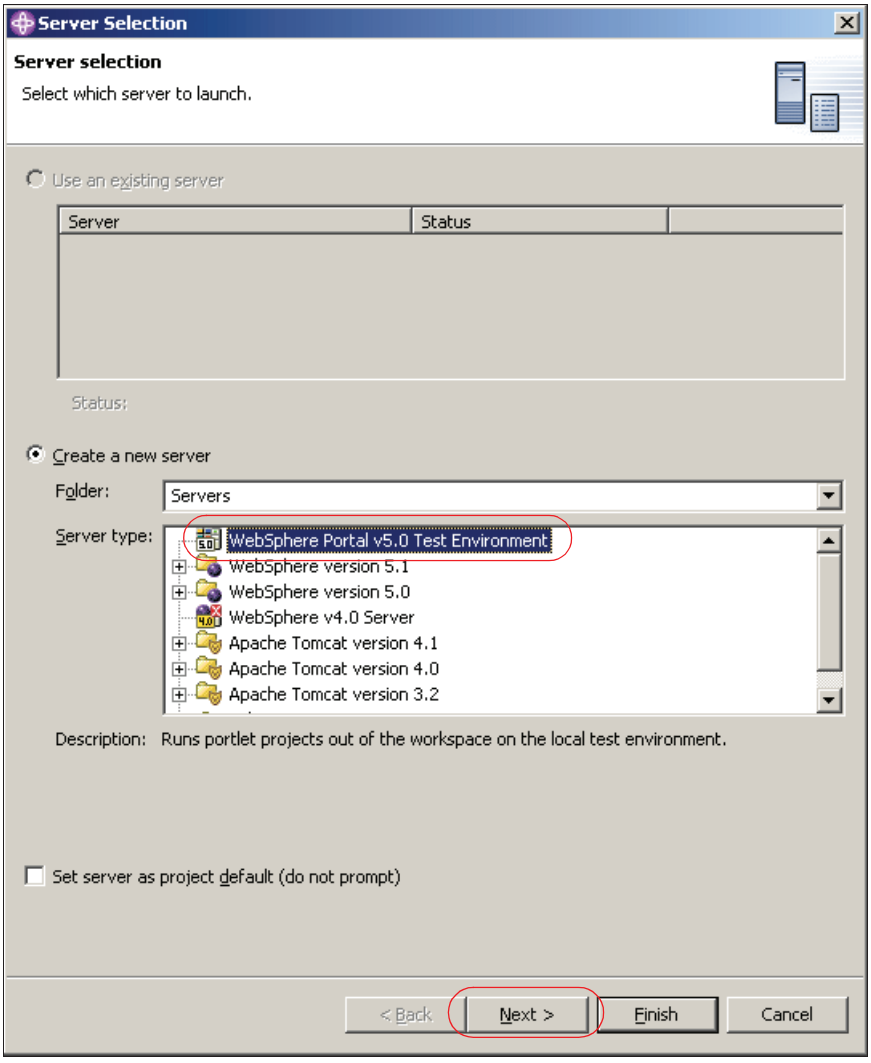


Figure 1-21 Server selection window

c. Accept the default port of 9081, and click **Finish** to run your project.

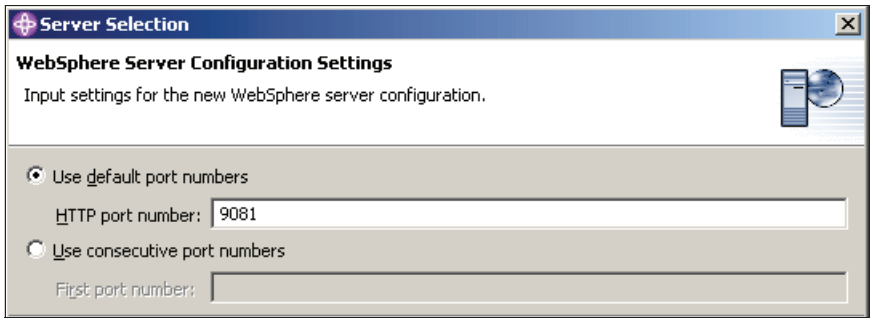


Figure 1-22 Port configuration window

3. Your project will be published, the server will start, and the portlet application will run in View mode. The portlet will display in the Web browser in WebSphere Studio Site Developer. At this time, if you want to view the portlet with an external device or simulator, follow the instructions in the next sections.

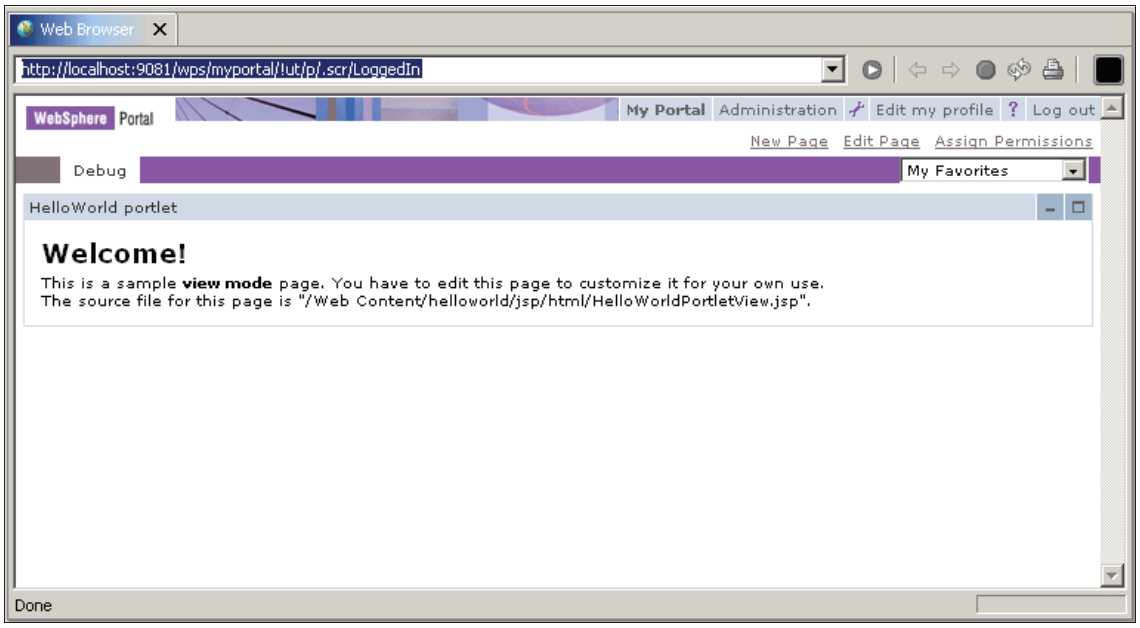


Figure 1-23 Portlet in Web browser in view mode

Displaying the portlet content on the Palm OS V5.4 Simulator

Note: You must install and configure the Palm Simulator for Internet access before you will be able to view the portlets.

1. Open the Palm Simulator and click **Web** to open the Web browser.

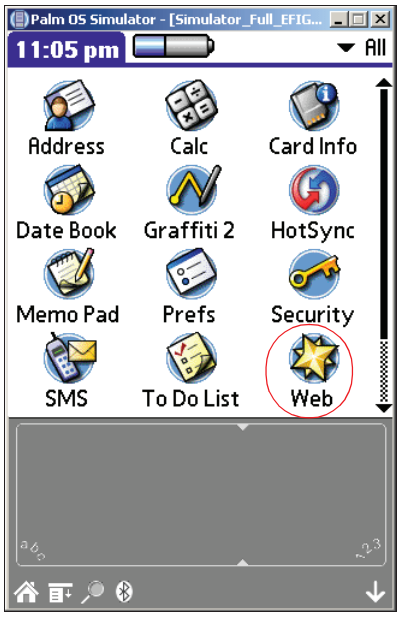


Figure 1-24 Accessing the Web browser on the Palm Simulator

2. Click the globe in the upper-right corner of the Simulator to enter the URL to navigate to. If your test environment is located on the same machine as the Palm Simulator, enter the following URL:

   ```
   http://localhost:9081/wps/portal/!ut/p/.cmd/LoginUserNoAuth?userid=wpsadmin
   &password=wpsadmin
   ```

3. Click **Go** to continue. If your test environment is located on a remote server, enter the following URL:

   ```
   http://your-server-name.your-co.com/wps/portal/!ut/p/.cmd/LoginUserNoAuth?u
   serid=wpsadmin&password=wpsadmin
   ```

4. Click **Go** to continue. When prompted to connect to the Internet, click **Connect.**

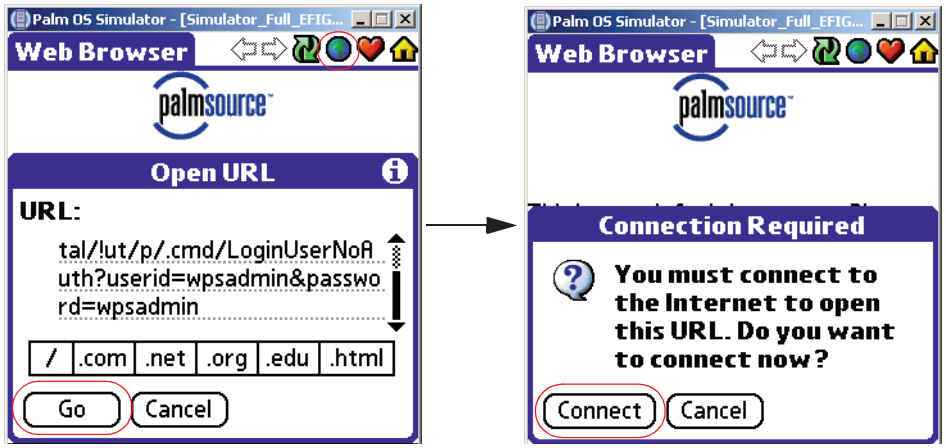


Figure 1-25 Entering a URL and connecting to the Internet

5. The portlet will be displayed as an icon in the browser. Click this icon to view the content of your portlet as generated for PDA devices.



Figure 1-26 Viewing your portlet

Displaying the portlet content on a Pocket PC device

The easiest way to test your portlets on your Pocket PC device is to connect the PDA to your computer via its cradle using the Microsoft® ActiveSync software.

1. Insert the PDA into its cradle and connect it to the computer via the manufacturer-supplied USB or serial cable.
2. Using the Microsoft ActiveSync software, establish a connection with the host computer.
3. Open a Web browser on the Pocket PC device. Enter `http://your-server-name.your-co.com/wps/myportal` as the destination URL to continue.
4. Just as in the Palm Simulator, you will be prompted to log in. Enter `wpsadmin` as the user name and password.

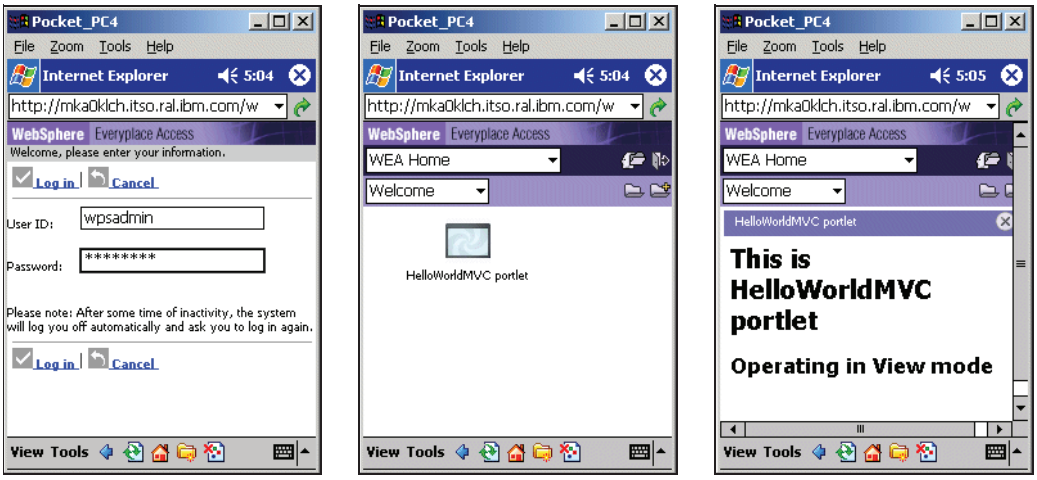


Figure 1-27 Pocket PC login and portlet

5. The portlet will load in the browser on the Pocket PC device.

Displaying the portlet content on the Nokia Simulator

Note: You must install and configure the Nokia Simulator for Internet access before you will be able to view the portlets.

1. Open the Nokia Mobile Internet Toolkit by clicking **Start -> Programs -> Nokia Developer Tools -> Nokia Mobile Internet Toolkit -> NMIT 4.1**.

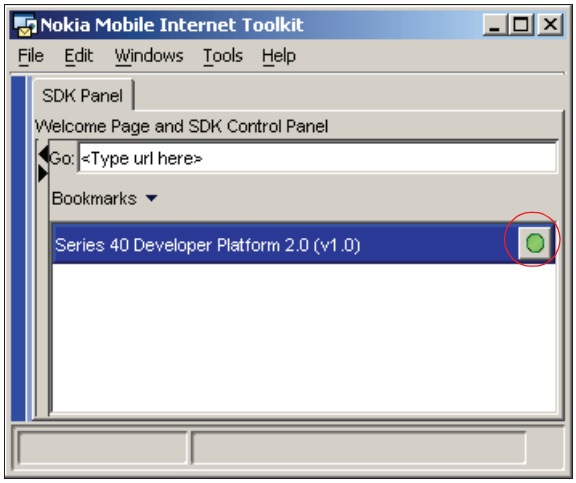


Figure 1-28 Nokia Mobile Internet Toolkit main window

2. Click the green button next to Series 40 Developer Platform to start the phone simulator.



Figure 1-29 Nokia Series 40 Simulator (based on the Nokia 6230 mobile phone)

3. In the toolkit window, type the URL:

   http://localhost:9081/wps/portal/!ut/p/.cmd/LoginUserNoAuth?userid=wpsadmin&password=wpsadmin

4. Press the **Enter** key.
5. If the portlet is being developed on another machine you can enter:

   http://your-server-name.your-co.com/wps/portal/!ut/p/.cmd/LoginUserNoAuth?userid=wpsadmin&password=wpsadmin

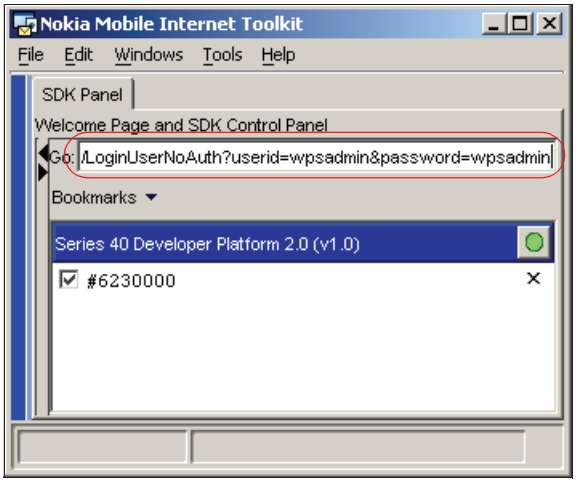


Figure 1-30 Entering a URL

6. The portlet will load in the phone simulator. Select the **HelloWorld portlet** link to view the portlet's content.

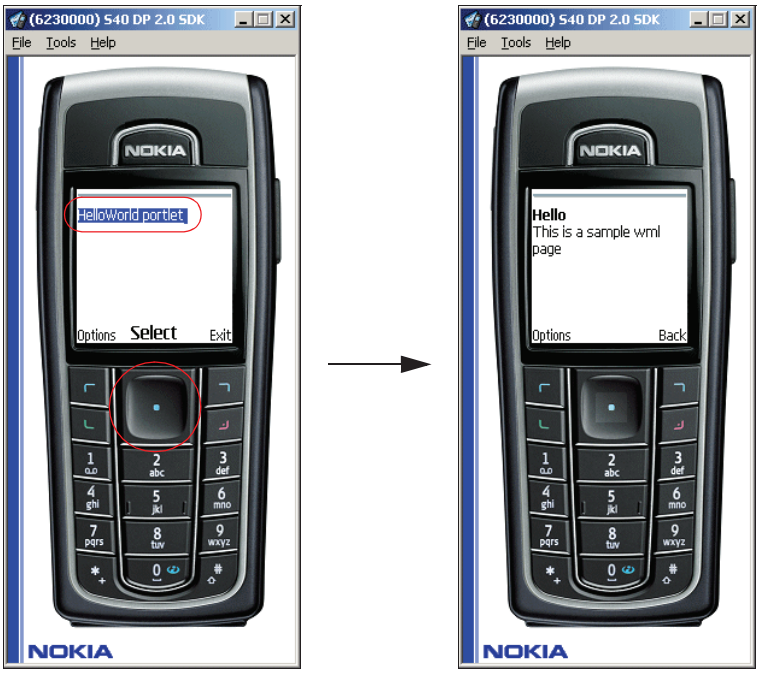


Figure 1-31 Viewing the portlet in the simulator

7. The Nokia simulator also includes a diagnostics window to view information surrounding user interaction with the portlet. This is a useful tool for debugging purposes.

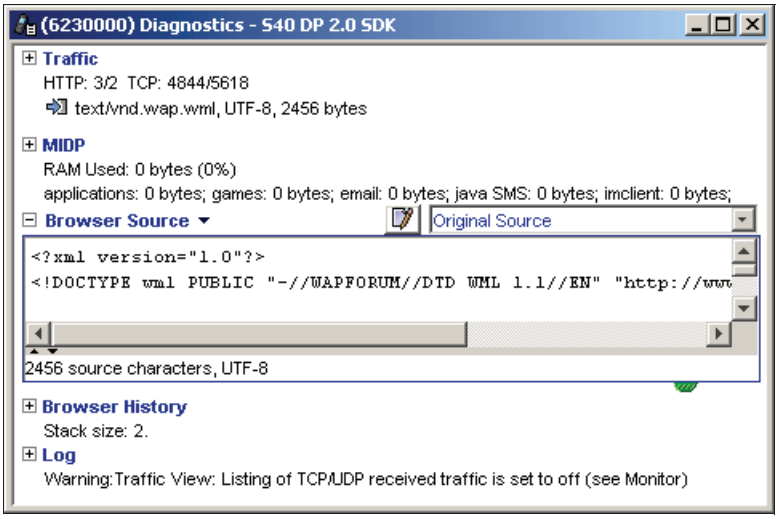


Figure 1-32 Toolkit diagnostics window

2

Portlet action event handling

This chapter introduces you to the action event handling capabilities of portlet applications. Included is a scenario that will allow you to understand the techniques used to develop portlets that process action events.

In this chapter you will find the following topics:

- How action events are processed
- Creating a basic portlet to implement an action event
- Code samples
- Running the portlet on the Palm OS V5.4 Simulator
- Running the portlet on the Nokia Series 40 browser simulator

© Copyright IBM Corp. 2005. All rights reserved.

2.1 Overview

We will start by creating a portlet application using the WebSphere Studio Site Developer. This portlet is again based on the Model-View-Controller (MVC) design pattern. In this scenario, the JavaBean is introduced as the model component of this architecture and is used to store user input.

Figure 2-1 illustrates the structure of this portlet application.

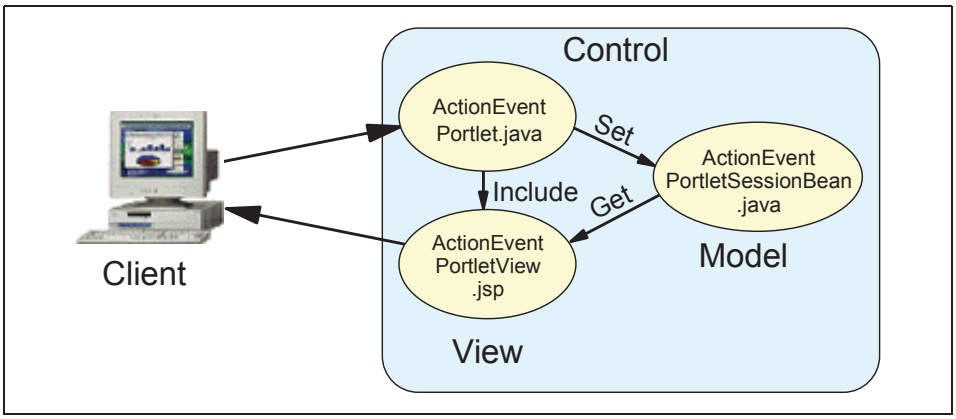


Figure 2-1 Example portlet application using MVC architecture

Portlet events contain information to which a portlet might need to respond such as resizing the portlet or processing data entered in a form submission. To receive these events, an event listener must be implemented in the portlet controller class. The three most commonly used types of events found in the Portlet API are:

- ActionEvents
- MessageEvents
- WindowEvents

There are also two additional events, PortletSettingsAttributeEvent and PortletApplicationSettingsAttributeEvent, which are used for notifications about changes to the attributes of the portlet settings of a concrete portlet or concrete portlet application.

In this scenario, you will create a portlet with support for action events using the Portlet wizard. You will then investigate and modify this source code to support multiple markup languages.

To send an ActionEvent you must associate a PortletAction with the HTTP request. The PortletAction is normally linked with URLs or buttons in an HTML form to provide a way for portlet programmers to implement different processing actions for different user input or interaction.

See Figure 2-2 for an illustration of action event handling.

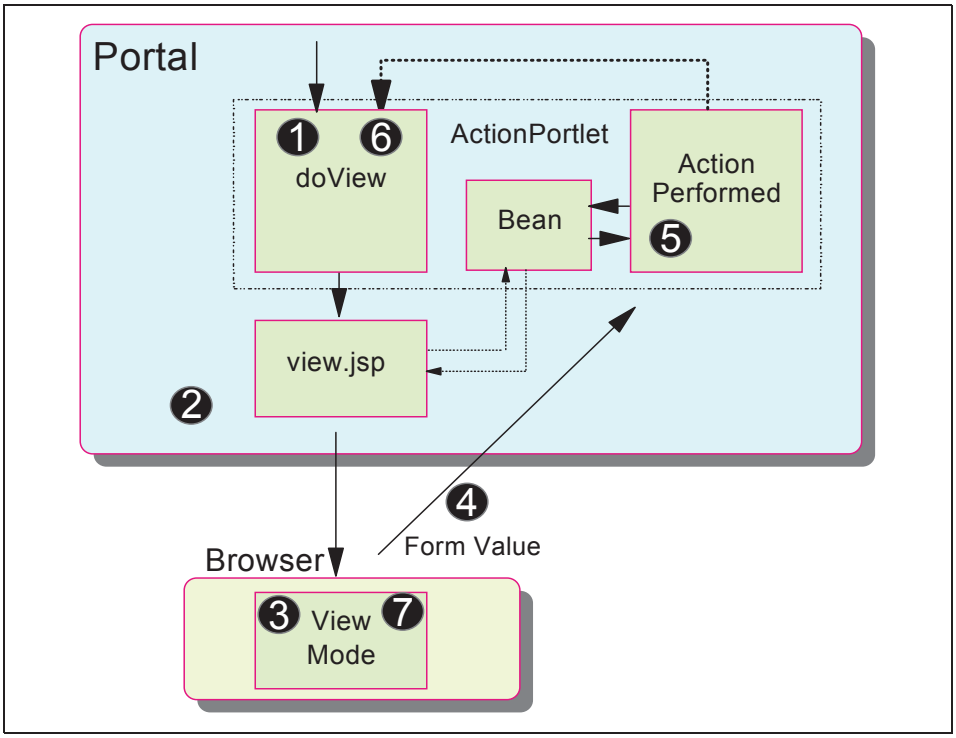


Figure 2-2 The event handling scenario

The sequence flow for this scenario is as follows:

1. The doView method is executed.
2. A JSP is invoked to render an initial screen. The resulting page will have a form in which to enter a value. Order details will not be displayed because the user has not entered an order id that has been stored in the bean yet.
3. The portlet is shown in view mode in the browser.
4. The user enters a value and clicks submit.
5. The actionPerformed method is executed to process the action. As part of the actionPerformed method, the entered value is stored in the session bean object.

6. The doView method is executed.
7. The view.jsp is again invoked to render the final screen. The JSP accesses the bean to display the value that the user entered.

2.2 Setup the ActionEvent portlet framework

In this section, you will create a portlet application that supports action events. The portlet application will be published and executed in the test environment and viewed on the Palm Simulator and on the Nokia simulator.

2.2.1 Create the framework with the portlet application wizard

1. Before you start this lab, make sure that the test environment is stopped. Click the **Server** tab in the tasks area to verify this. Also make sure all editors and browsers have been closed
2. In WebSphere Studio Site Developer, click **File->New->Other**.
3. Select **Portlet Development** in the left column and **Portlet Application Project** in the right column. The click **Next** to continue.
4. Name the project ActionEvent. Select the option **Create basic portlet**. Make sure that **Configure advanced options** is checked. Click **Next** to continue.

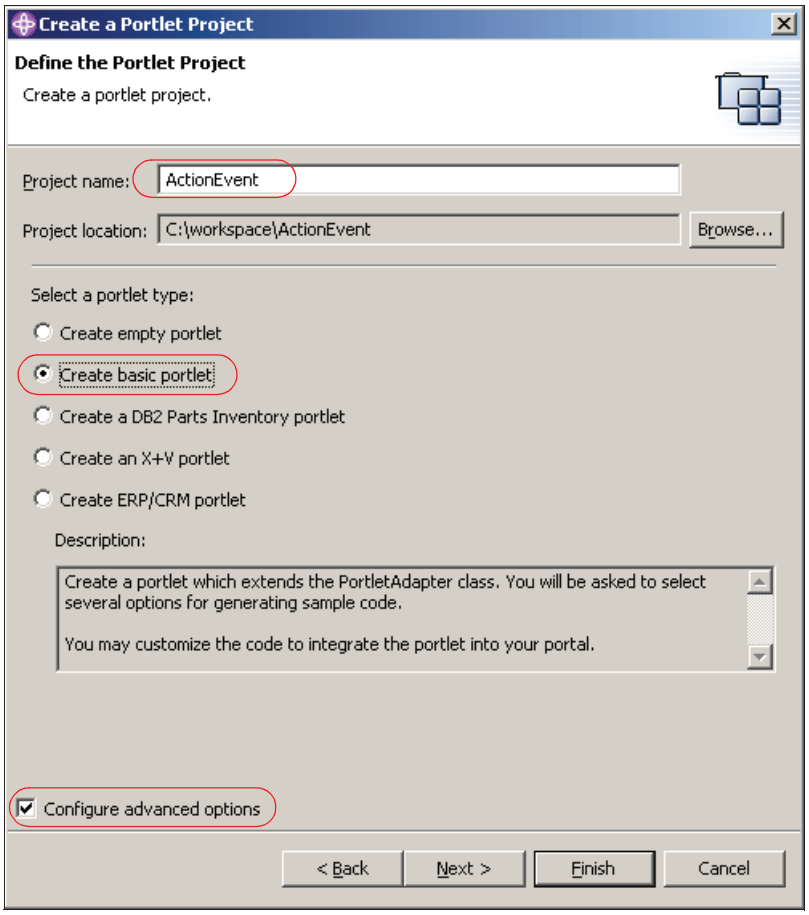


Figure 2-3 Naming the project

5. Accept the default settings on the J2EE settings window. Click **Next** to continue.
6. Click **Next** to accept the default settings on the Portlet settings window.
7. On the Event Handling window, check the options to **Add action listener** and to **Add form sample**.

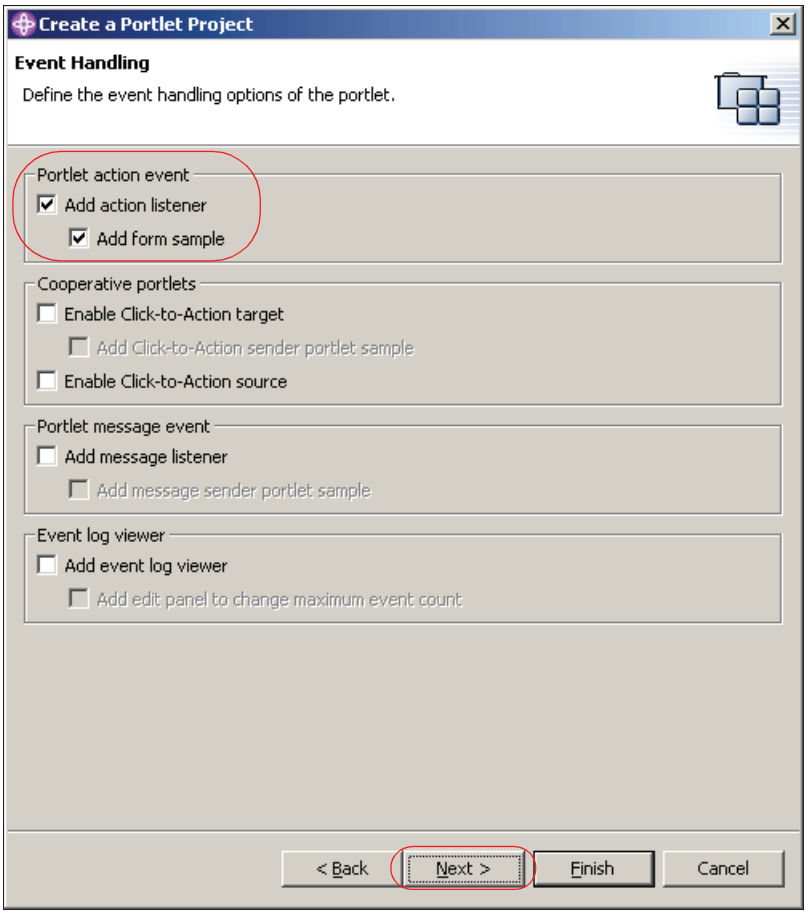


Figure 2-4 Adding support for action events

8. Accept the default values on the Single Sign-On window. Click **Next** to continue.
9. On the Miscellaneous window, select **Add PDA markup support** and **Add wml markup support**. Click **Finish** to generate the framework for your project.

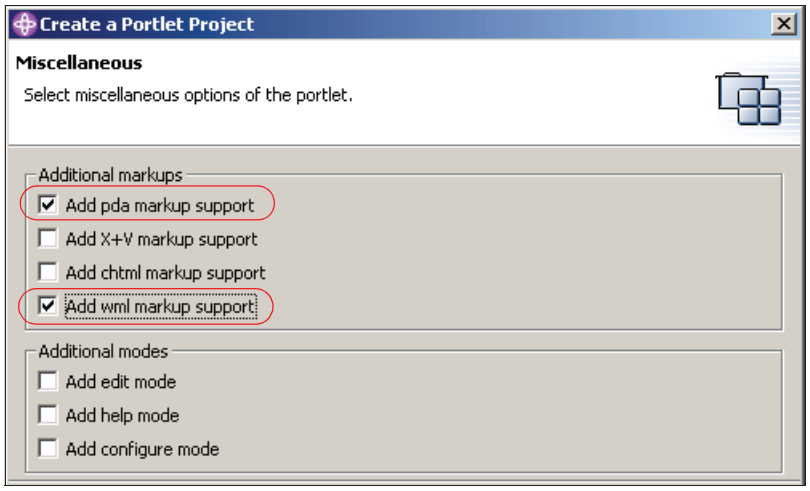


Figure 2-5 Adding support for multiple markup languages

2.2.2 Configure the test environment to support this project

Follow these steps to properly configure your project for the test environment and start working on your project:

1. In the Project Navigator panel, expand the DefaultEAR project and double-click **application.xml**. This file is the application deployment descriptor and will open in its own editor.

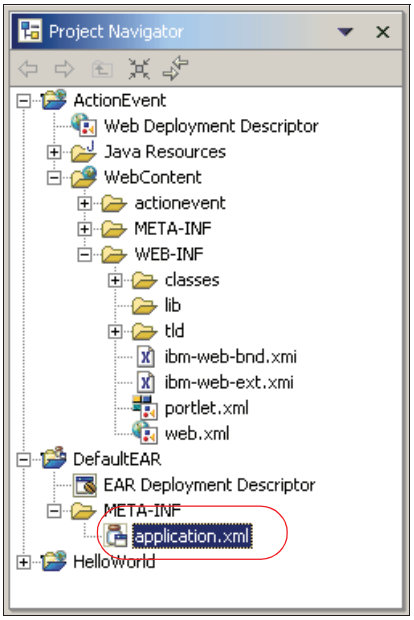


Figure 2-6 Opening the application deployment descriptor

2. Once the application deployment descriptor opens, select the Module button along the bottom of the editor.

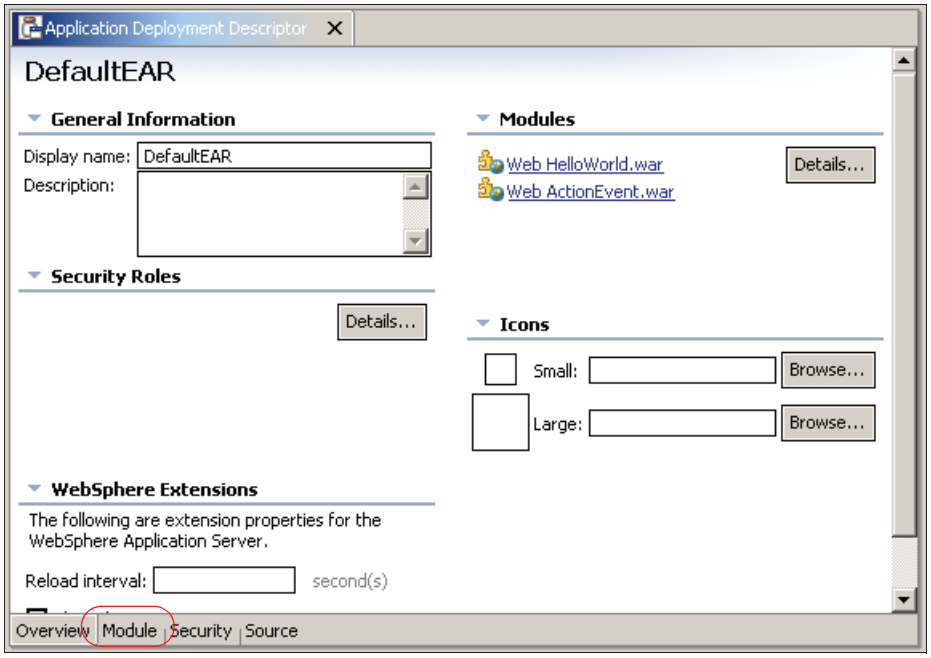


Figure 2-7 The application deployment descriptor editor

3. Select Web HelloWorld.war and click the Remove button. Click **File->Save** and close this file.

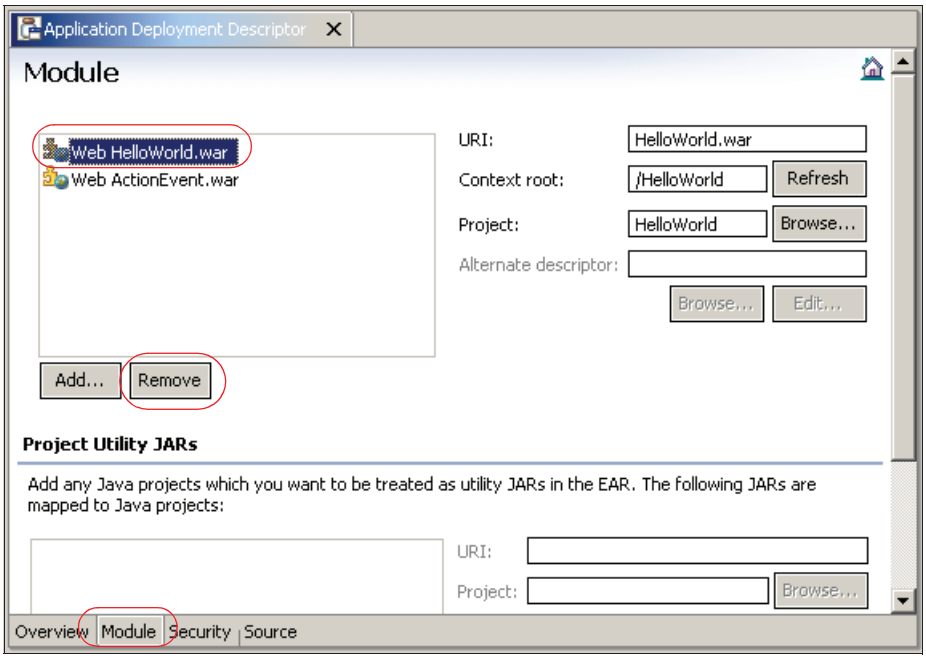


Figure 2-8 Removing the HelloWorld project

4. In the bottom panel of WebSphere Studio, click the **Servers** tab. Double-click **WebSphere Portal v5.0 Test Environment**.

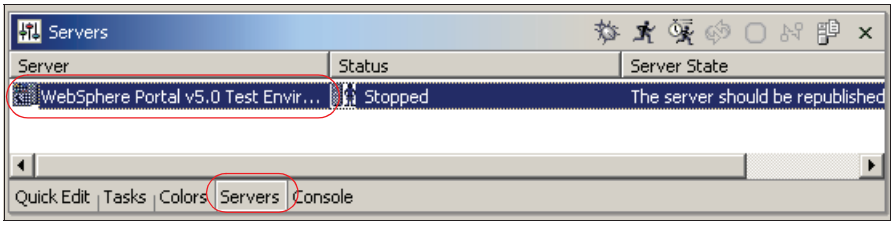


Figure 2-9 The servers panel

5. You will receive a dialog box stating that certain entries in the server configuration are not valid. Click **Yes** to repair them.

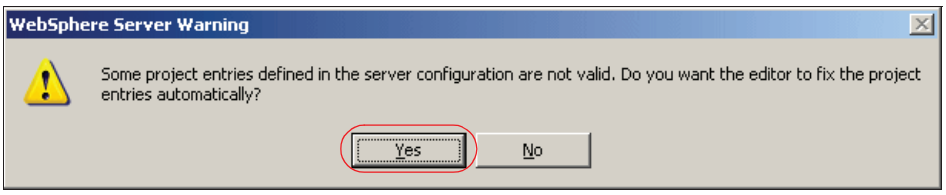


Figure 2-10 Server configuration dialog box

6. Click **File->Save** to save the server configuration.
7. You will receive a window prompt to repair the WebSphere v5.0 Server Configuration. Click **OK** to do this.

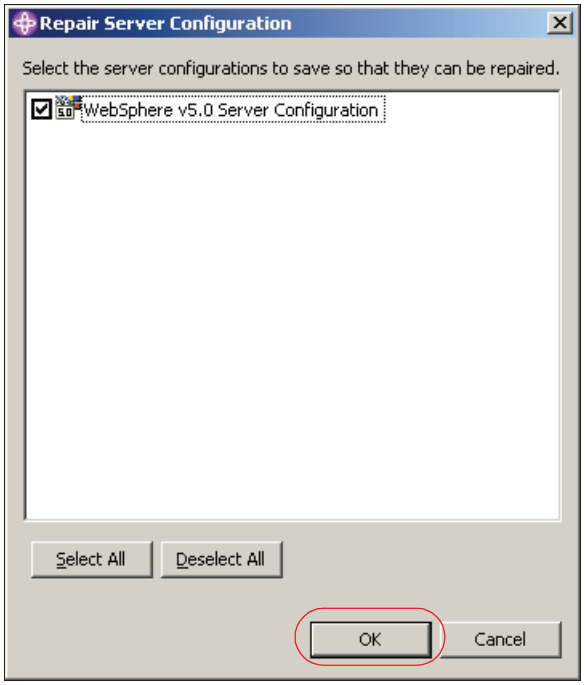


Figure 2-11 Server configuration dialog box

8. Click **OK** on the next window to remove the child project HelloWorld.war from the EAR project.

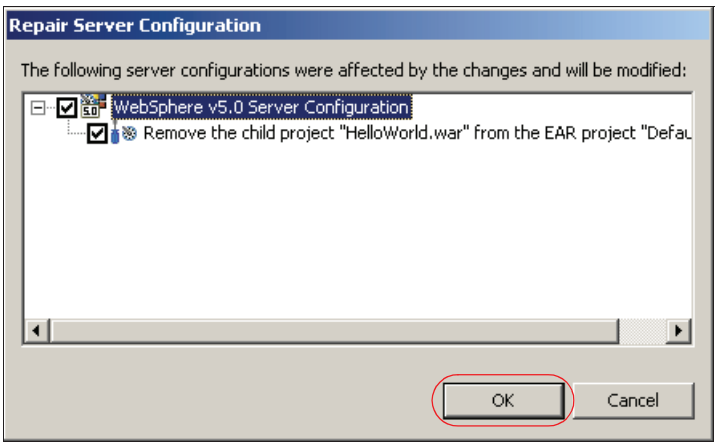


Figure 2-12 Removing HelloWorld dialog box

9. Click **File->Save** and exit the server configuration to complete configuration of the test environment for your project.

2.3 Examine and modify source code

In this section, you will review the code generated by the portlet application wizard and investigate how it works.

2.3.1 Examine the model and controller components

1. In the project navigator panel, double-click the **ActionEventPortlet.java** file.

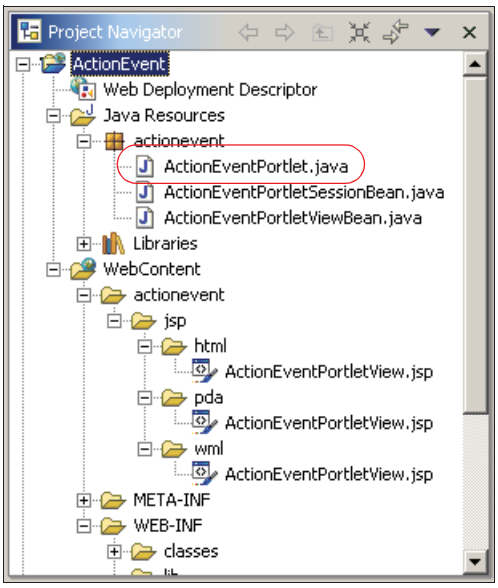


Figure 2-13 Project navigator panel

This file contains the doView() method used to render the JSP in view mode. The JSP that is used to generate the content is dependent on the markup language required by the requesting device. This is determined by the getJspExtension() method. The various JSPs used to render content are located in their respectively named folders under the /WebContent/actionevent/jsp directory.

The most important component of this file the actionPerformed() method. In this method, the ActionListener implemented by this class receives the result of the user interaction with the application. In this case, the order number that the user enters is stored as the form text in the session bean.

Example 2-1 ActionEventPortlet.java

```
package actionevent;
...
public class ActionEventPortlet extends PortletAdapter implements
ActionListener {
...
   public void init(PortletConfig portletConfig) throws UnavailableException {
      super.init(portletConfig);
   }
...
   public void doView(PortletRequest request, PortletResponse response) throws
PortletException, IOException {
   ...
```

```
   // Invoke the JSP to render

   getPortletConfig().getContext().include(VIEW_JSP+getJspExtension(request),
request, response);
   }
...
   public void actionPerformed(ActionEvent event) throws PortletException {
      ...
      // ActionEvent handler
      String actionString = event.getActionString();
      // Add action string handler here
        PortletRequest request = event.getRequest();
      ActionEventPortletSessionBean sessionBean = getSessionBean(request);

      if( FORM_ACTION.equals(actionString) ) {
         // Set form text in the session bean
         sessionBean.setFormText(request.getParameter(TEXT));
      }
   }
...
   private static String getJspExtension(PortletRequest request) {
      String markupName = request.getClient().getMarkupName();
      return "jsp";
   }

}
```

2. Now double-click the **ActionEventPortletSessionBean.java** file. This file is used as the model component of the MVC architecture and stores the text that the user enters into the form. The actionPerformed method stores the form text value in this bean so that the view mode JSP can access this value using the getFormText() method when it renders content.

Example 2-2 ActionEventPortletSessionBean.java

```
...
public class ActionEventPortletSessionBean {

   private String formText = "";

   public void setFormText(String formText) {
        this.formText = formText;
    }

    public String getFormText() {
        return this.formText;
    }
}
```

2.3.2 Examine the HTML source generated by the wizard

To examine the automatically generated source code, follow these steps:

1. In the project navigator panel, double-click the ActionEventPortletView.jsp file located under the /html directory in the /WebContent/actionevent/jsp folder.

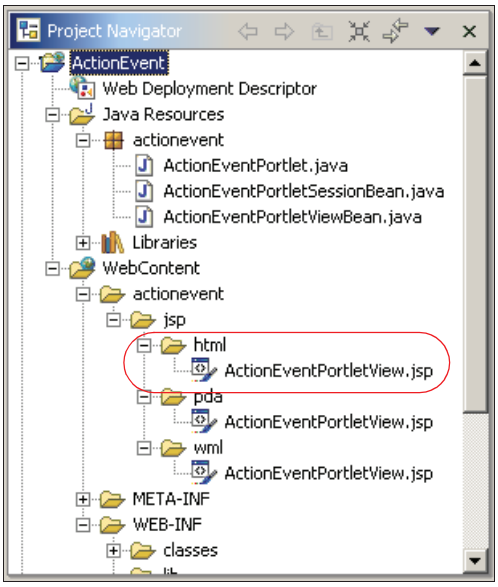


Figure 2-14 Wizard generated JSPs

The page directive at the top of the page sets the content type generated by this JSP and imports the actionevent package so all the supporting java classes can be used for exchanging data with this JSP.

```
<%@ page session="false" contentType="text/html" import="java.util.*,
actionevent.*"%>
```

The portlet container provides tags for use in portlet JSPs. You need to include this directive to make these tags available in your project and then initialize the tag library:

```
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>
```

The JSP instantiates the session bean so that it can access the form text that the user enters. If the JSP detects that the user has already entered an order number, it will display “Order details for order id 'order number' should be displayed here.”

It will always display a form for the user to enter another order id.

Example 2-3 ActionEventPortletView.jsp

```
<%@ page session="false" contentType="text/html" import="java.util.*,
actionevent.*"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>

<%
   ActionEventPortletViewBean viewBean =
(ActionEventPortletViewBean)portletRequest.getAttribute(ActionEventPortlet.VIEW
_BEAN);
   ActionEventPortletSessionBean sessionBean =
(ActionEventPortletSessionBean)portletRequest.getPortletSession().getAttribute(
ActionEventPortlet.SESSION_BEAN);
%>

<DIV style="margin: 6px">

<H3 style="margin-bottom: 3px">Welcome!</H3>
This is a sample <B>view mode</B> page. You have to edit this page to customize
it for your own use.<BR>
The source file for this page is "/Web
Content/actionevent/jsp/html/ActionEventPortletView.jsp".

<H3 style="margin-bottom: 3px">Form sample</H3>
This is a sample form to test action event handling.
<DIV style="margin: 12px; margin-bottom: 36px">
<% /******** Start of sample code ********/ %>

  <%
  String formText = sessionBean.getFormText();
  if( formText.length()>0 ) {
    %>Order details for order id '<%=formText%>' should be displayed here.<%
  }
  %>
  <FORM method="POST" action="<%=viewBean.getFormActionURI()%>">
    <LABEL class="wpsLabelText"   for="<portletAPI:encodeNamespace
value='<%=ActionEventPortlet.TEXT%>'/>">Enter order id:</LABEL><BR>
    <INPUT class="wpsEditField"  name="<portletAPI:encodeNamespace
value='<%=ActionEventPortlet.TEXT%>'/>" type="text"/>
    <INPUT class="wpsButtonText" name="<portletAPI:encodeNamespace
value='<%=ActionEventPortlet.SUBMIT%>'/>" value="Submit" type="submit"/>
  </FORM>

<% /******** End of sample code *********/ %>
</DIV>

</DIV>
```

2.3.3 Modify the source for PDA markup

You will need to modify the code in the JSP that supports PDA markup that has been generated by the portlet application wizard. The modifications will allow PDAs to support the same form as the page used to generate HTML markup.

The original source code to the JSP that supports PDA markup is:

Example 2-4 Original PDA JSP source

```
<%@ page session="false" contentType="text/html" %>
<P>
<B>Hello</B><BR>
This is a sample pda page
</P>
```

Follow these steps to allow this JSP to support a form to process order id entry:

1. Open the ActionEventPortletView.jsp file located in the /WebContent/actionevent/jsp/pda folder by double-clicking it.
2. Add the JSP directives to allow the JSP to interact with your portlet application, initialize the Portlet API tag library, and instantiate the session bean. This code is exactly the same as the one used in the JSP that supports HTML markup and can be copied from that file.

Example 2-5 Adding JSP directives

```
<%@ page session="false" contentType="text/html" import="java.util.*,
actionevent.*"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>

<%
   ActionEventPortletViewBean viewBean =
(ActionEventPortletViewBean)portletRequest.getAttribute(ActionEventPortlet.VIEW
_BEAN);
   ActionEventPortletSessionBean sessionBean =
(ActionEventPortletSessionBean)portletRequest.getPortletSession().getAttribute(
ActionEventPortlet.SESSION_BEAN);
%>
<P>
<B>Hello</B><BR>
This is a sample pda page
</P>
...
```

3. Now add the form to handle the user input of order numbers. This is the exact same form that is generated and used by the JSP that supports HTML markup and can be copied from this file.

   Because this is a simple example, no modification to the source code is required. If this form was structured in a complex table or used large graphics, it would need to be modified and/or scaled down to function correctly on a PDA due to the limitations of these devices.

Example 2-6 Adding the form

```
...
<P>
<B>Hello</B><BR>
This is a sample pda page
</P>
  <%
  String formText = sessionBean.getFormText();
  if( formText.length()>0 ) {
    %>Order details for order id '<%=formText%>' should be displayed here.<%
  }
  %>
  <FORM method="POST" action="<%=viewBean.getFormActionURI()%>">
    <LABEL class="wpsLabelText"   for="<portletAPI:encodeNamespace
value='<%=ActionEventPortlet.TEXT%>'/>">Enter order id:</LABEL><BR>
    <INPUT class="wpsEditField"  name="<portletAPI:encodeNamespace
value='<%=ActionEventPortlet.TEXT%>'/>" type="text"/>
    <INPUT class="wpsButtonText" name="<portletAPI:encodeNamespace
value='<%=ActionEventPortlet.SUBMIT%>'/>" value="Submit" type="submit"/>
  </FORM>
```

2.3.4 Modify the source for WML markup

Because of the limitations of small-screened devices such as Web-enabled cellular telephones, the approach taken to modify the JSP used to generate WML markup is different. Instead of relying on user input as used in the forms above, we will display to the user a series of links to click to receive information about different order numbers.

The original source to the JSP that supports WML markup is:

Example 2-7 Original WML JSP source

```
<%@ page session="false" contentType="text/vnd.wap.wml" %>
<p>
<b>Hello</b><br/>
This is a sample wml page
```

```
</p>
```

To modify the source code of his JSP, follow these instructions:

1. Open the ActionEventPortletView.jsp located in the /WebContent/actionevent/jsp/wml folder.
2. As you did with the PDA JSP, add the page directives, initialize the Portlet API, and instantiate the session bean so that they can be used in this project.

> **Note:** Make sure that you do not change the content type from "text/vnd.wap.wml." This content type is required for this form to work on mobile devices that only support the WML markup language.

Example 2-8 Add JSP directives, Portlet API, and session bean

```
<%@ page session="false" contentType="text/vnd.wap.wml" import="java.util.*,
actionevent.*"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>

<%
   ActionEventPortletViewBean viewBean =
(ActionEventPortletViewBean)portletRequest.getAttribute(ActionEventPortlet.VIEW
_BEAN);
   ActionEventPortletSessionBean sessionBean =
(ActionEventPortletSessionBean)portletRequest.getPortletSession().getAttribute(
ActionEventPortlet.SESSION_BEAN);
%>
<p>
<b>Hello</b><br/>
This is a sample wml page<br/>
...
```

3. Now we will modify the source code to this file and add links to view order details for order "1234" and order "5678." To use the Portlet API to make a WML link, use the following code:

   ```
   <a href="<portletAPI:createURI><portletAPI:URIAction
   name="<%=ActionEventPortlet.FORM_ACTION%>"/><portletAPI:URIParameter
   name="<%=ActionEventPortlet.TEXT%>"
   value="1234"/></portletAPI:createURI>">1234</a>
   ```

   By passing a parameter with the same name as the input box used in the forms in the previous JSPs, the same business logic in the actionPerformed() method of the ActionEventPortlet.java file can be used to process this input.

Example 2-9 Add links and support for displaying order details

```
...
<p>
<b>Hello</b><br/>
This is a sample wml page<br/>
<%
  String formText = sessionBean.getFormText();
  if( formText.length()>0 ) {
    %>Order details for order id '<%=formText%>' should be displayed
here.<br/><%
  }
  %>
  <a href="<portletAPI:createURI><portletAPI:URIAction
name="<%=ActionEventPortlet.FORM_ACTION%>"/><portletAPI:URIParameter
name="<%=ActionEventPortlet.TEXT%>"
value="1234"/></portletAPI:createURI>">1234</a><br/>
  <a href="<portletAPI:createURI><portletAPI:URIAction
name="<%=ActionEventPortlet.FORM_ACTION%>"/><portletAPI:URIParameter
name="<%=ActionEventPortlet.TEXT%>"
value="5678"/></portletAPI:createURI>">5678</a>
</p>
```

Important: Make sure to include the final </p> in the sample code above. If this tag is not included, the project may not display correctly on the mobile device.

2.4 Run the project

Follow these steps to run the project:

1. Make sure you have saved the changes to all the files you have modified.
2. Right-click the project name and select **Run on Server**.

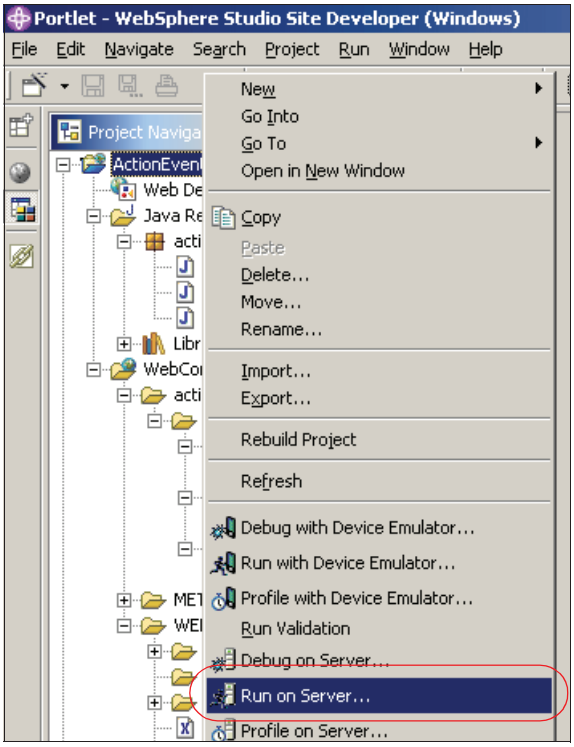


Figure 2-15 Run on Server

3. Because you have already configured your project to run on the test environment with the steps at the beginning of this chapter, you can use the existing test environment to run your project. Click **Finish** to run your project in the test environment.

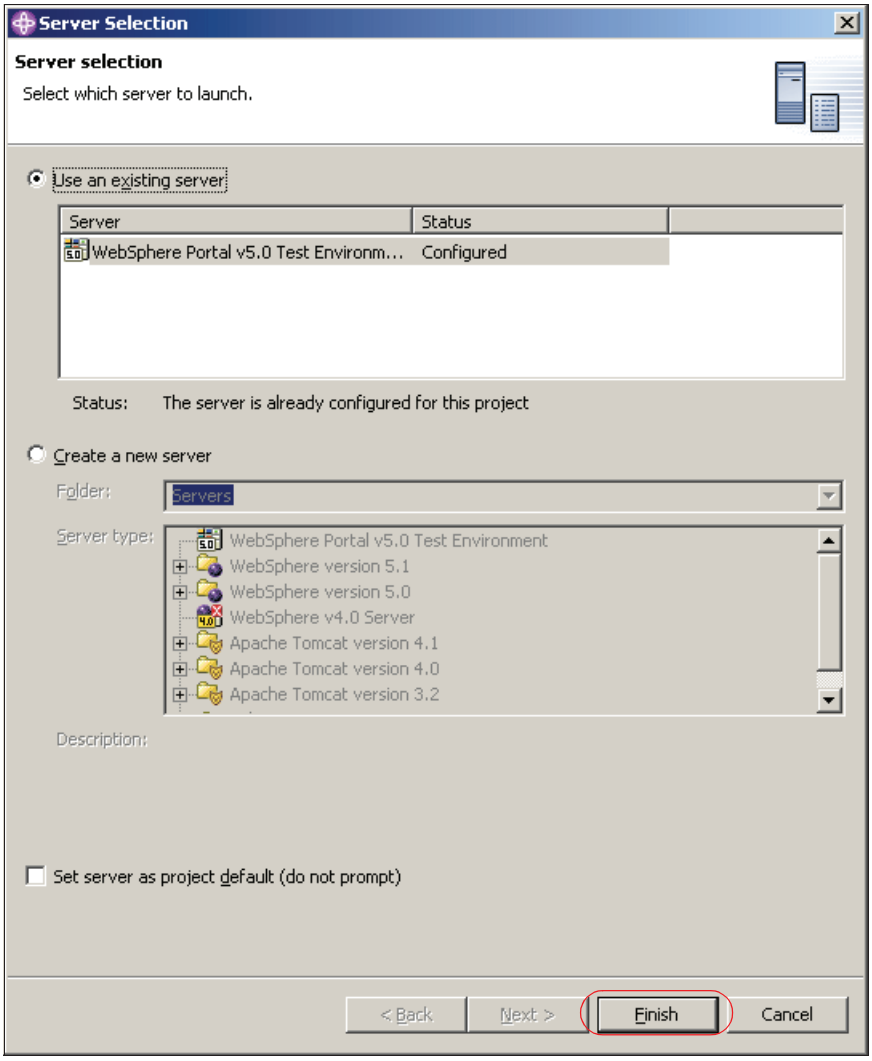


Figure 2-16 Choosing the test environment

4. The project will load in the Web browser integrated into the development environment. Enter a value in the text box and click **Submit**.

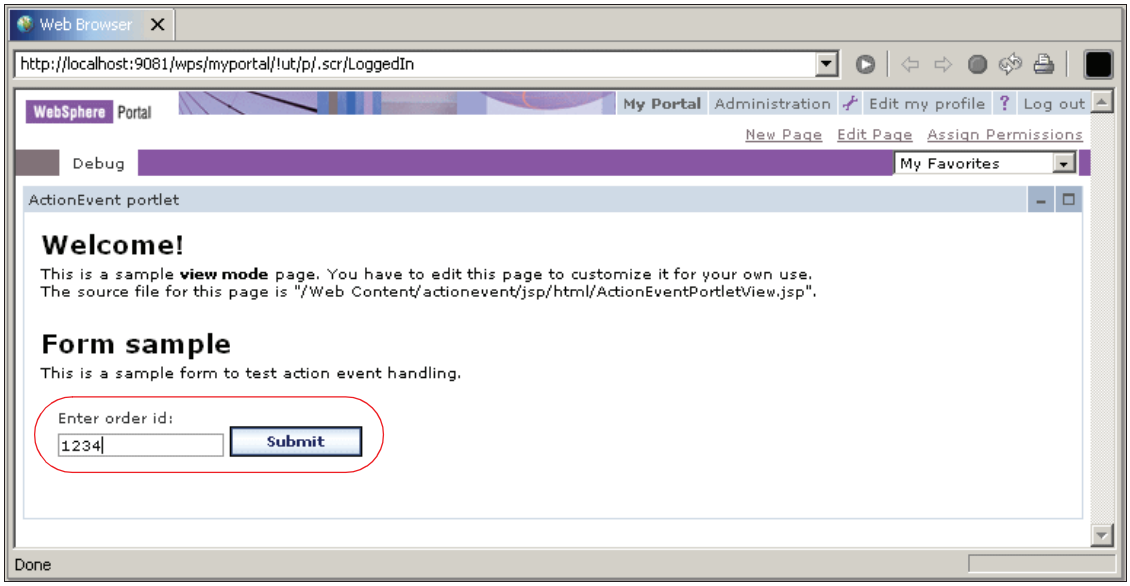


Figure 2-17 Enter an order id number

5. The window will now show the details for the order and the order id number.

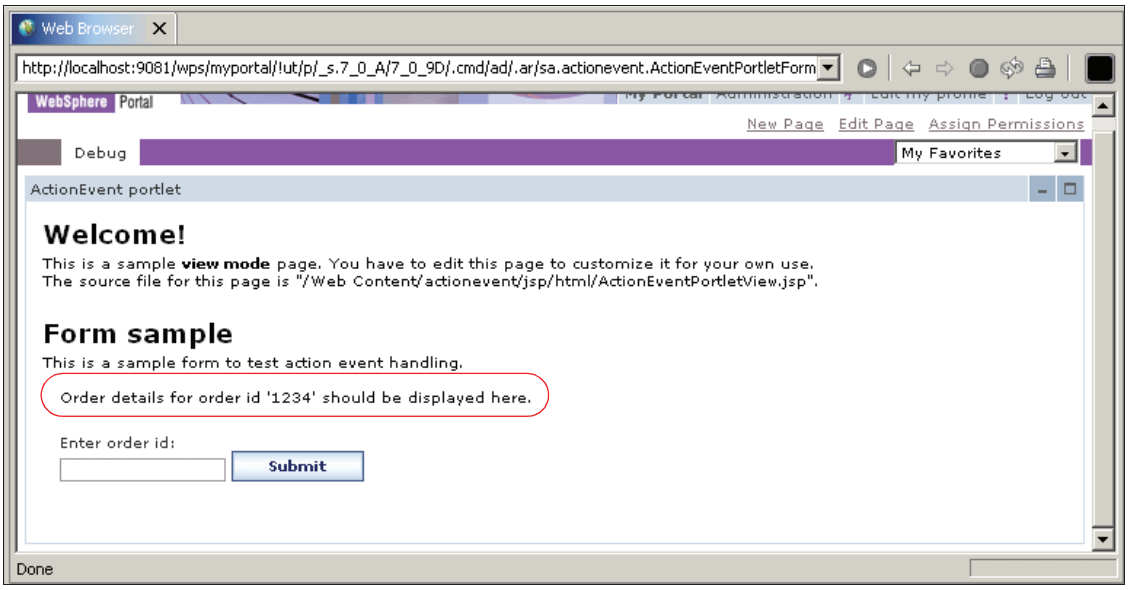


Figure 2-18 Submit results

Run your project in the Palm Simulator

Follow these steps to view your portlet project in the Palm Simulator:

1. Open the Palm Simulator and click **Web** to open a Web browser on the device.

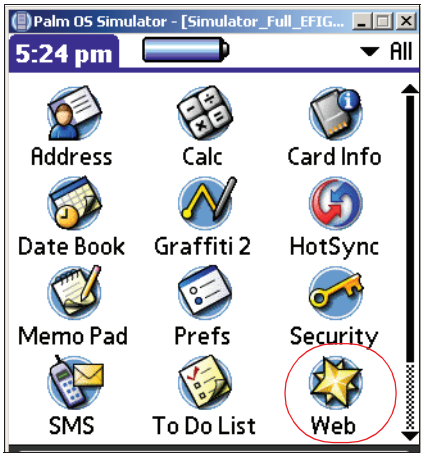


Figure 2-19 Opening the Web browser on the Palm Simulator

> **Note:** You must configure the Palm Simulator to run the Web browser and access the Internet before you are able to perform these steps.

2. Click **Connect** to access the Internet.



Figure 2-20

3. Click the **globe** in the upper right corner to enter a URL. Enter the following URL to access your project, then click **Go**:
   `http://localhost:9081/wps/portal/!ut/p/.cmd/LoginUserNoAuth?userid=wpsadmin&password=wpsadmin`

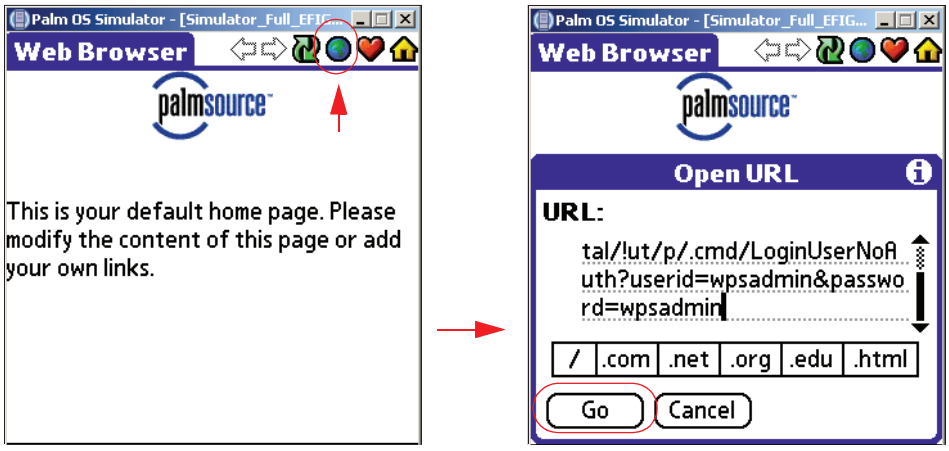


Figure 2-21 Navigating to a URL

4. Click the icon for **ActionEvent portlet**. This will display the content of your project.

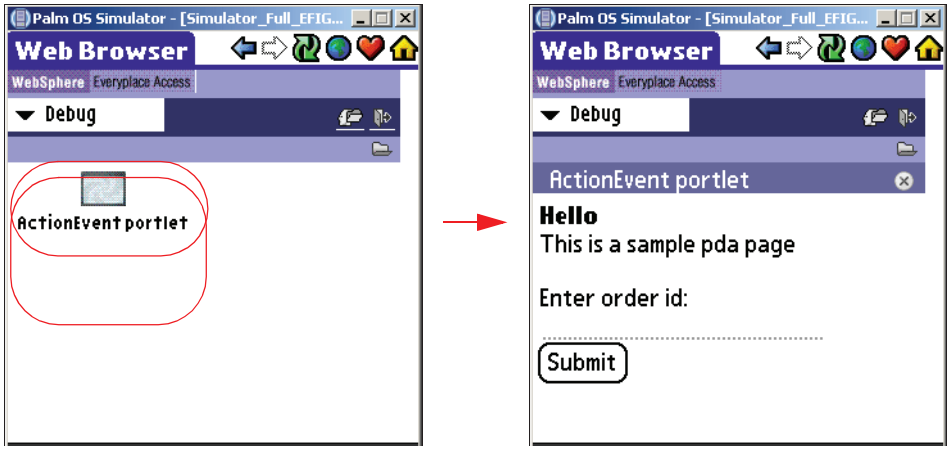


Figure 2-22 ActionEvent portlet

5. Enter an order id and click the **Submit** button.



Figure 2-23 Entering an order id

6. The window will now show the details for the order and the order id number.



Figure 2-24 Order details and number displayed

Run your project in the Nokia Simulator

Follow these steps to view your project on the Nokia Simulator:

1. Click **Start->Programs->Nokia Developer Tools->Nokia Mobile Internet Toolkit->NMIT 4.1** to start the Nokia Mobile Internet Toolkit.

> **Note:** You must install and configure the Nokia Mobile Internet Toolkit with the Series 40 Simulator before you are able to perform these steps.

2. Click the **green button** to the right of Series 40 Developer Platform 2.0 to start the phone simulator.

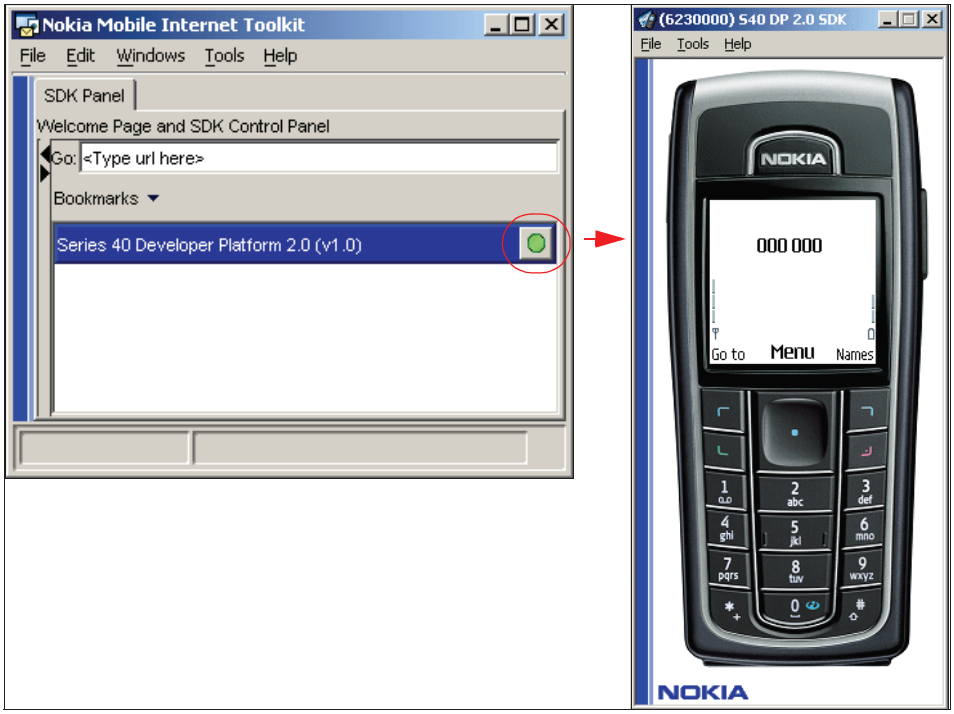


Figure 2-25 Start the Nokia phone simulator

3. Enter the following URL into the text box in the toolkit and press the **Enter** key to load your project into the simulator:

   `http://localhost:9081/wps/portal/!ut/p/.cmd/LoginUserNoAuth?userid=wpsadmin&password=wpsadmin`

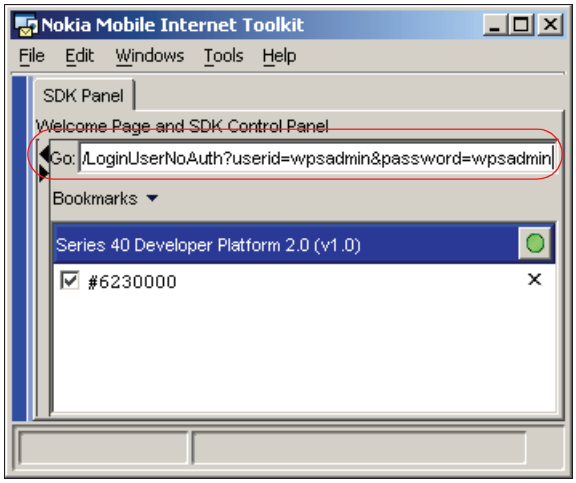


Figure 2-26 Entering a URL

4. Select **ActionEvent portlet** to view the content of your project



Figure 2-27 Selecting ActionEvent portlet

5. You now see two links corresponding to two sample order numbers. Select **1234** to view the order details and number of that project.



Figure 2-28 Select an order and display its details

3

Portlet messaging

Message events are a way for portlets to communicate messages to each other. This is accomplished through a familiar event listener model. Portlets that need to listen for message events must implement a MessageListener interface, and portlets that need to send message events do so within the handling of their own ActionEvents. Message events can be sent to named portlets or broadcast to all portlets on the same Portal page. All events are handled during the page's event-processing phase, which comes before the content generation phase.

In this chapter you will find the following topics:

- How message events are processed
- Using a portlet to send a message
- Receiving and displaying a message
- Running the message event handling portlet on the Palm Simulator

© Copyright IBM Corp. 2005. All rights reserved.

3.1 Overview

In this lab you will use the portlet application wizard to create a message sender and a message receiver portlet example for HTML markup. You will then modify the source code to send and receive messages in PDA markup using the Palm Simulator.

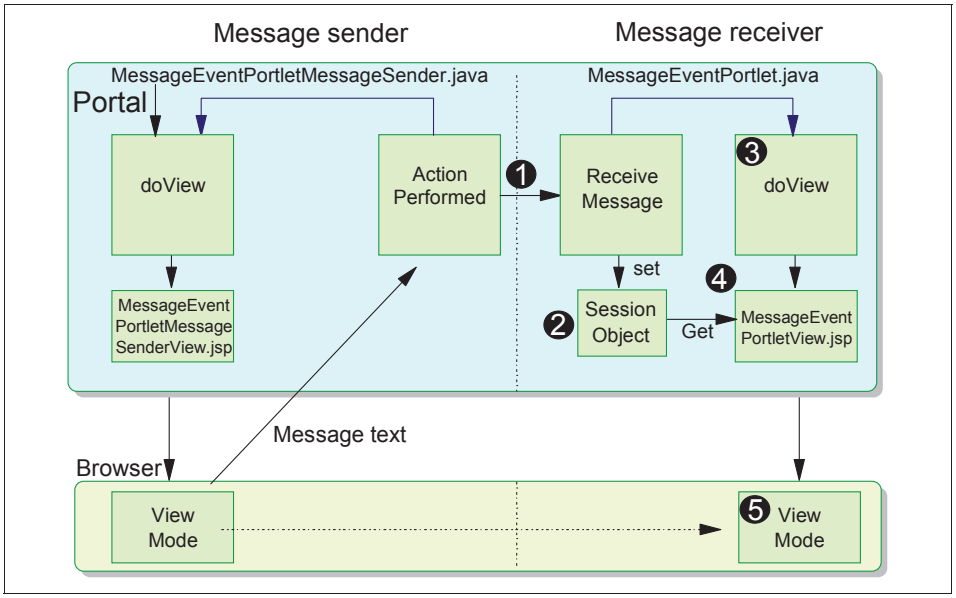


Figure 3-1 Message Event handling scenario

The flow for this scenario as follows:

1. The actionPerformed() method in the Message Sender portlet will send a broadcast message event of type DefaultPortletMessage upon arrival of action events. In this example, the message to be sent is entered by the user as text in a form.

   The MessageReceiver portlet that will implement the MessageListener interface receives the message in the messageReceived() method.

2. In this scenario, the received message is saved in the PortletSession object. Portlet data can also be used. It is not recommended to use the request object when developing applications for mobile devices. Because of the user interaction flow on PDAs and mobile phones, data passed in the request object may be lost.

3. Portal invokes the doView method, which in turn invokes the JSP.

4. The JSP retrieves the message from the session object and displays the message.

This scenario will be implemented using a broadcast style of message event rather than point-to-point messaging. In addition, the DefaultPortletMessage object will be used.

Note: While a DefaultPortletMessage object allows you to send messages to portlets in different applications, you can only send a String type message.

3.2 Setup the MessageEvent portlet framework

To construct this portlet, we will first create the framework for this portlet using the portlet application wizard and then modify the source code to suit our needs. Our goal is to demonstrate sending and receiving a message event using the Palm PDA simulator.

3.2.1 Create the framework with the portlet application wizard

Follow these steps to create the portlet application framework:

1. From the WebSphere Studio Site Developer menu bar, click **File->New->Other**.
2. Click **Portlet Development** in the left panel and **Portlet Application Project** on the right panel of the next window. Click **Next** to continue.
3. Name the project MessageEvent, select the **Create basic portlet** option, and check the **Configure advanced options** box. Click **Next** to continue.

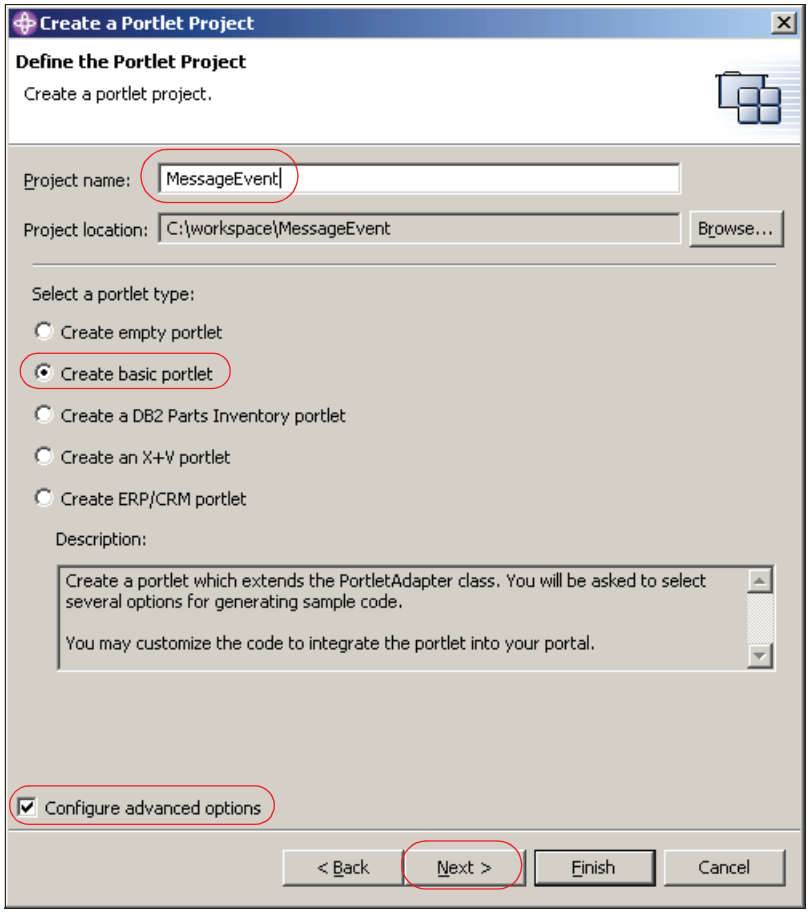


Figure 3-2 Naming the portlet project

4. On the J2EE Settings window, accept the default values and click **Next**.
5. On the Portlet Settings window, accept the default values and click **Next**.
6. On the Event Handling window, deselect **Add form sample** and **Add action listener**. Check the boxes to select **Add message listener** and **Add message sender portlet sample**.

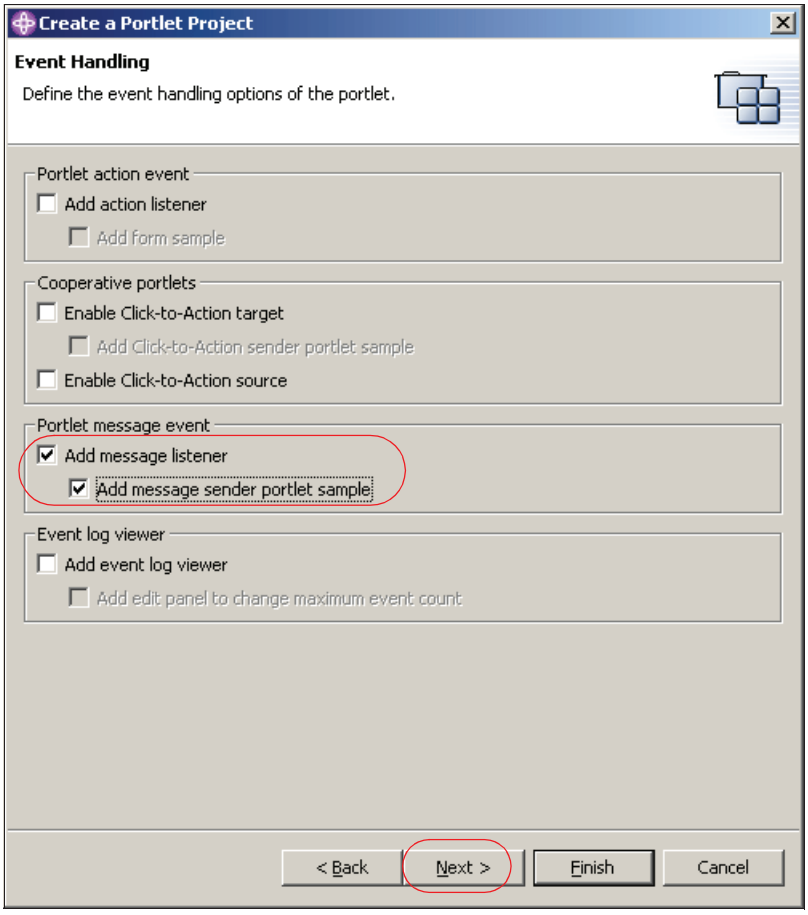


Figure 3-3 Adding the Message Event listener

7. Accept the default values on the Single Sign-On window and click **Next**.
8. On the Miscellaneous window, check the **Add pda markup support** option. Click **Finish** to generate your portlet application framework.

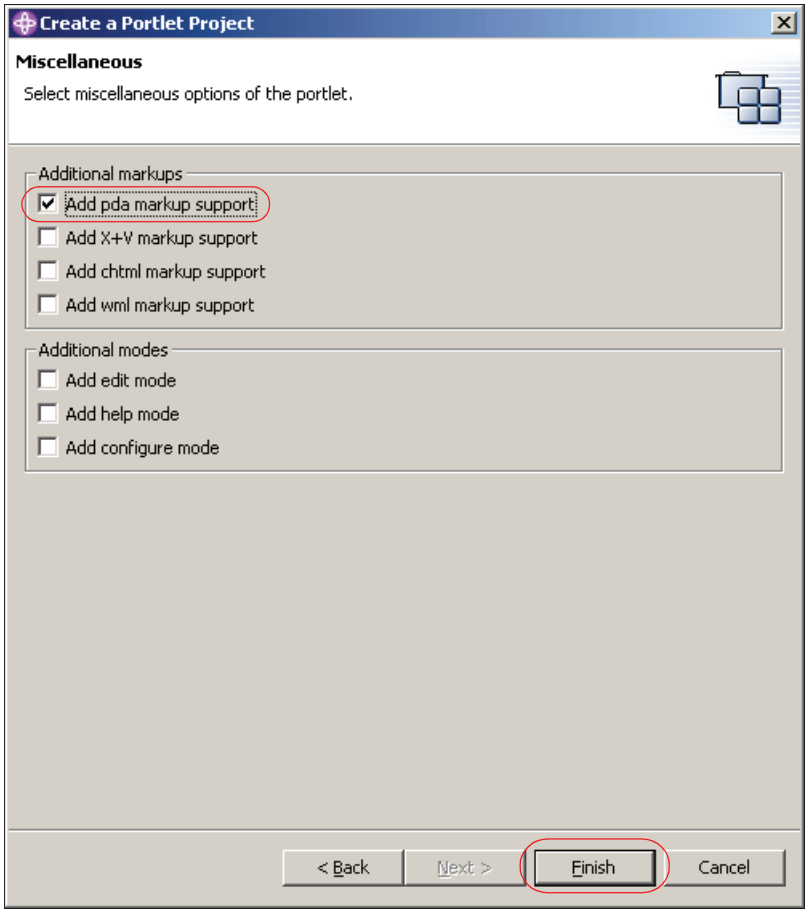


Figure 3-4 Adding PDA markup support

3.2.2 Configure the test environment to support this project

You will need to remove the ActionEvent portlet from the DefaultEAR project and add the MessageEvent portlet to the DefaultEAR project in order for it to run correctly in the test environment. Follow these steps:

1. Expand the /DefaultEAR/META-INF folder in the Project Navigator panel. Double-click **application.xml** to open the EAR Deployment Descriptor in its own editor.

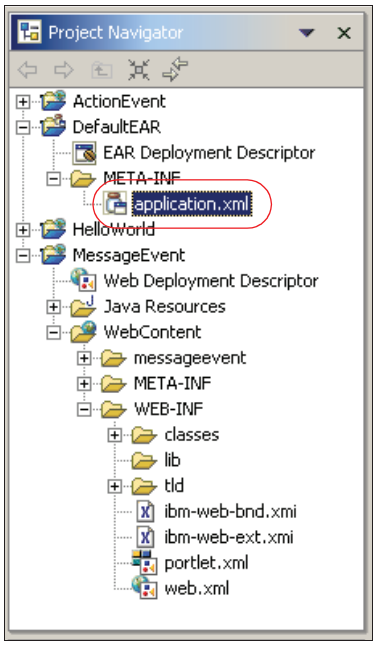


Figure 3-5 Opening application.xml

2. Click the **Module** tab at the bottom of this editor.

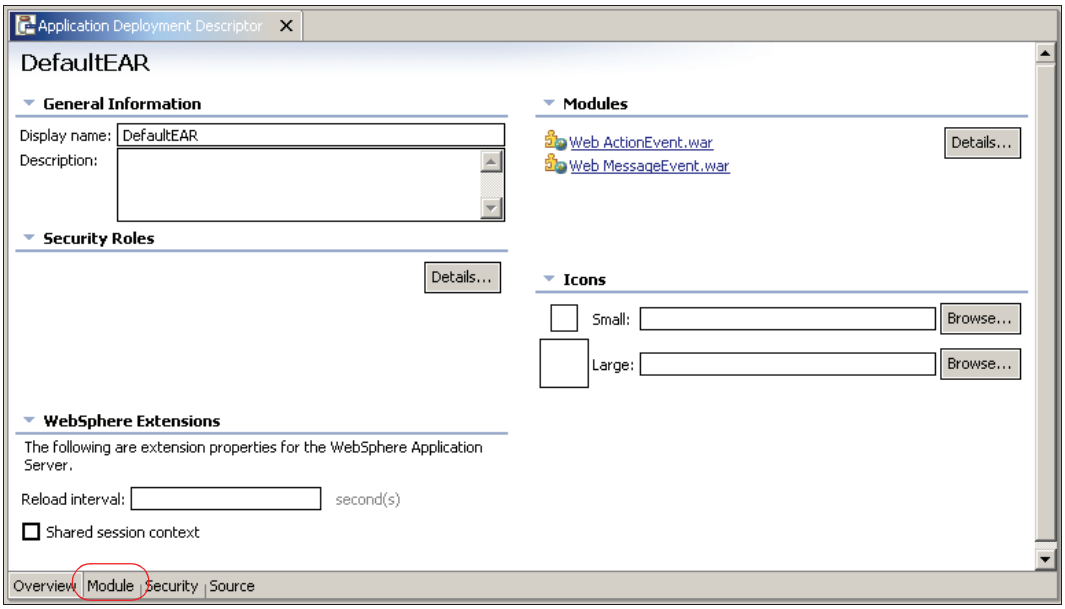


Figure 3-6 Selecting the module tab

3. Click **Web ActionEvent.war** and click the **Remove** button to remove it from this project.

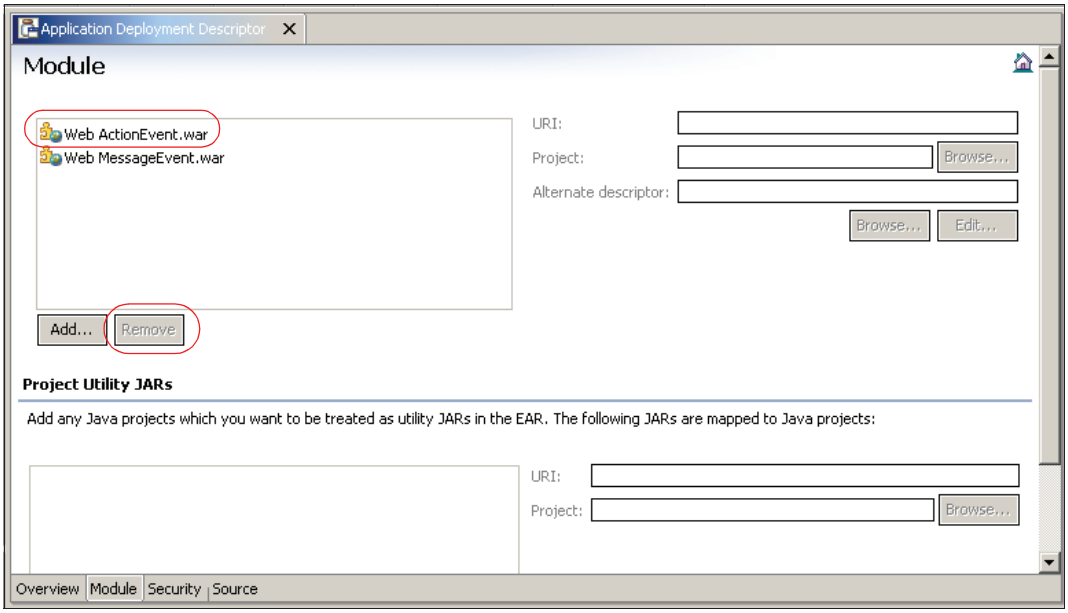


Figure 3-7 Removing the Web ActionEvent.war module

4. In the bottom panel of WebSphere Studio, select the **Servers** tab. Double-click **WebSphere Portal v5.0 Test Environment**.

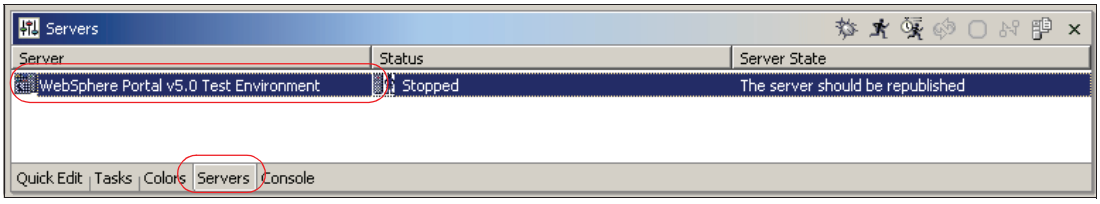


Figure 3-8 Updating the test environment server configuration

5. Click **Yes** to the warning dialog box that you receive to fix the configuration entries automatically.

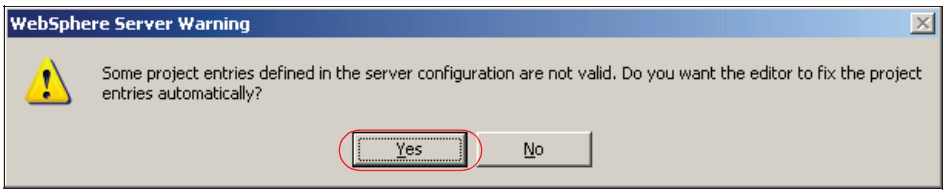


Figure 3-9 Repair server configuration dialog

6. The server configuration editor will open. Click **File->Save** to save the changes made when the configuration was changed automatically.
7. Click **OK** on the Repair Server Configuration dialog box.

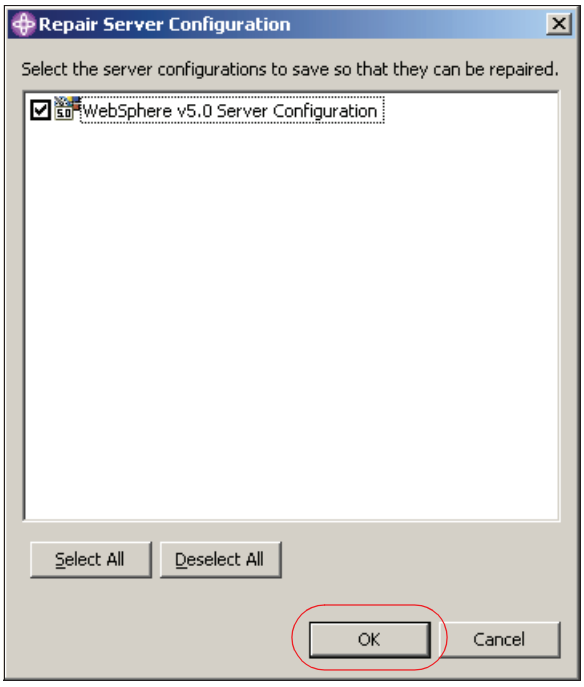


Figure 3-10 Repair server configuration dialog

8. Click **OK** to remove the ActionEvent.war project from the DefaultEAR project.

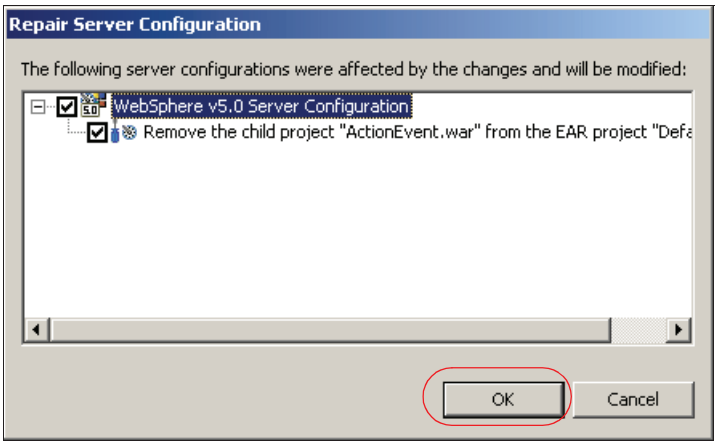


Figure 3-11 Removing the ActionEvent.war module

3.3 Examine and modify source code

Now we will examine the source code to this project and modify it to meet our needs.

3.3.1 Examine and modify the model and controller components

Follow these instructions to change the message sender and receiver portlets:

1. Open **MessageEventPortletMessageSender.java** by double-clicking it.
2. Examine the actionPerformed() method in this file. The actionPerformed() method instantiates a DefaultPortletMessage object and sends the message. The parameter **`null`** indicates that this is a broadcast message. The command **`event.getRequest().getParameter(TEXT)`** is the value of the message. This is the value entered by the user in the form of the message sender portlet.

Example 3-1 actionPerformed() method of sender portlet

```
...
   public void actionPerformed(ActionEvent event) throws PortletException {
      if (ACTION.equals(event.getActionString()))
         getPortletConfig().getContext().send(null, new
DefaultPortletMessage((String)event.getRequest().getParameter(TEXT)));
   }
...
```

3. Once the message is sent, there needs to be a way to receive it. This is done by implementing a message receiver interface and a messageReceived method. This is very similar to the way action events are processed. Double-click **MessageEventPortlet.java** to view this source code.

Example 3-2 MessageEventPortlet.java

```
...
public class MessageEventPortlet extends PortletAdapter implements
MessageListener {

   ...

   public void doView(PortletRequest request, PortletResponse response) throws
PortletException, IOException {
        // Invoke the JSP to render

getPortletConfig().getContext().include(VIEW_JSP+getJspExtension(request),
request, response);
   }

   ...

   public void messageReceived(MessageEvent event) throws PortletException {
       if( getPortletLog().isDebugEnabled() )
           getPortletLog().debug("MessageListener - messageReceived called");
       // MessageEvent handler
       PortletMessage msg = event.getMessage();
       // Add PortletMessage handler here
       if( msg instanceof DefaultPortletMessage ) {
           String messageText = ((DefaultPortletMessage)msg).getMessage();
           // Add DefaultPortletMessage handler here
       }
       else {
           // Add general PortletMessage handler here
       }
   }

   ...

}
```

4. Because we want to capture this message and display it to the user, we need to add some code to the DefaultPortletMessage handler. For this example, we will use the session object to store the message value. We will store the message value as an attribute called “MyMessage.” Using the session object will enable us to easily access this value from within the view JSP.

Example 3-3 Modifying the DefaultPortletMessage handler

```
...
      public void messageReceived(MessageEvent event) throws PortletException{
       if( getPortletLog().isDebugEnabled() )
           getPortletLog().debug("MessageListener - messageReceived called");
       // MessageEvent handler
       PortletMessage msg = event.getMessage();
       // Add PortletMessage handler here
       if( msg instanceof DefaultPortletMessage ) {
           String messageText = ((DefaultPortletMessage)msg).getMessage();
           // Add DefaultPortletMessage handler here
             PortletRequest request = event.getRequest();
             PortletSession session = request.getPortletSession();
             session.setAttribute("MyMessage", messageText);
       }
       else {
           // Add general PortletMessage handler here
       }
   }
```

5. Save all of your changes.

3.3.2 Examine and modify the HTML source generated by the wizard

Follow these steps to examine and modify the source code to the view component of this portlet application.

1. Double-click the **MessageEventPortletMessageSenderView.jsp** file. This file generates a form with which the user can enter and send a message. The value entered in this form is extracted in the actionPerformed() method of the MessageEventPortletMessageSender.java file and broadcasted as a message.

Example 3-4 Form used to enter a message in the message sender JSP

```
...
<FORM method="POST"
action="<%=portletRequest.getAttribute(MessageEventPortletMessageSender.ACTION_
URI)%>">
    <LABEL class="wpsLabelText"   for="<portletAPI:encodeNamespace
value='<%=MessageEventPortletMessageSender.TEXT%>'/>">Enter
message:</LABEL></BR>
    <INPUT class="wpsEditField"  name="<portletAPI:encodeNamespace
value='<%=MessageEventPortletMessageSender.TEXT%>'/>" type="text"/>
    <INPUT class="wpsButtonText" name="<portletAPI:encodeNamespace
value='<%=MessageEventPortletMessageSender.SUBMIT%>'/>" value="Submit"
type="submit"/>
</FORM>
```

```
...
```

2. Double-click the **MessageEventPortletView.jsp** file. This is the file that will receive the broadcasted message. Since we are storing our message in the session object, we will need to set the page session to **true** in the page directive on the first line of this file.

Example 3-5 Changing the scope of the JSP to use the session

```
<%@ page session="true" contentType="text/html" import="messageevent.*" %>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>
...
```

3. Add the following code to display the message if one is received. If the attribute “MyMessage” does not exist in the session, a message will be shown indicating this condition. If there is a value, this value will be displayed.

Example 3-6 Adding code to display a message if one is received

```
<DIV style="margin: 6px">

<H3 style="margin-bottom: 3px">Welcome!</H3>
This is a sample <B>view mode</B> page. You have to edit this page to customize
it for your own use.<BR>
The source file for this page is "/Web
Content/messageevent/jsp/html/MessageEventPortletView.jsp".

<br>
<% if(session.getAttribute("MyMessage") == null) { %>
A message has not been sent yet
<% } else { %>
The message you entered is:<br>
<%= session.getAttribute("MyMessage") %><br>
<% } %>

</DIV>
```

4. Save all of your changes.

3.3.3 Modify the source for PDA markup

Follow these steps to make your portlet support PDA markup.

1. Right-click MessageEventPortletMessageSenderView.jsp in the /WebContent/messageevent/jsp/html folder and click **Copy**. Paste this file into the /WebContent/messageevent/jsp/pda folder. Do the same for the

MessageEventPortletView.jsp file. Click **Yes** when prompted to overwrite the existing file. This will replicate an exact copy of the JSPs used to render the HTML content in the PDA directory.

2. Double-click **MessageEventPortletMessageSenderView.jsp** in the /WebContent/messageevent/jsp/pda folder. We will modify the source code of this file to display more appropriately on the smaller screen of the PDA. If this step is not completed, the user will have to scroll excessively to read the content of the screen. Change the code to match that of Example 3-7. Be sure to also erase the final </DIV> tags from the original JSP.

Example 3-7 View.jsp code for PDA message sender

```
<%@ page session="false" contentType="text/html" import="messageevent.*" %>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>

<H3>Message sender sample</H3>

<% /******** Start of sample code ********/ %>

  <FORM method="POST"
action="<%=portletRequest.getAttribute(MessageEventPortletMessageSender.ACTION_
URI)%>">
    <LABEL class="wpsLabelText"   for="<portletAPI:encodeNamespace
value='<%=MessageEventPortletMessageSender.TEXT%>'/>">Enter
message:</LABEL></BR>
    <INPUT class="wpsEditField"  name="<portletAPI:encodeNamespace
value='<%=MessageEventPortletMessageSender.TEXT%>'/>" type="text"/>
    <INPUT class="wpsButtonText" name="<portletAPI:encodeNamespace
value='<%=MessageEventPortletMessageSender.SUBMIT%>'/>" value="Submit"
type="submit"/>
  </FORM>

<% /******** End of sample code *********/ %>
```

3. Now we will perform a similar option to the MessageEventPortletView.jsp for PDAs. Double-click this file to open it and perform these actions. Again be sure to erase the final </DIV> tag from the original JSP.

Example 3-8 Modifications to the message receiver portlet for PDAs

```
<%@ page session="true" contentType="text/html" import="java.util.*,
messageevent.*"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>

<H3>Message receiver</H3>
```

```
<br>
<% if(session.getAttribute("MyMessage") == null) { %>
A message has not been sent yet
<% } else { %>
The message you entered is:<br>
<%= session.getAttribute("MyMessage") %><br>
<% } %>
```

4. Save all of your changes.

3.3.4 Add view support for the message sender portlet for PDAs

In order to send a message with PDA markup, you must first enable the message sender portlet to work in PDA markup mode. To do this, we need to modify the portlet.xml file.

1. Expand the /WebContent/WEB-INF folder and double-click **portlet.xml**.

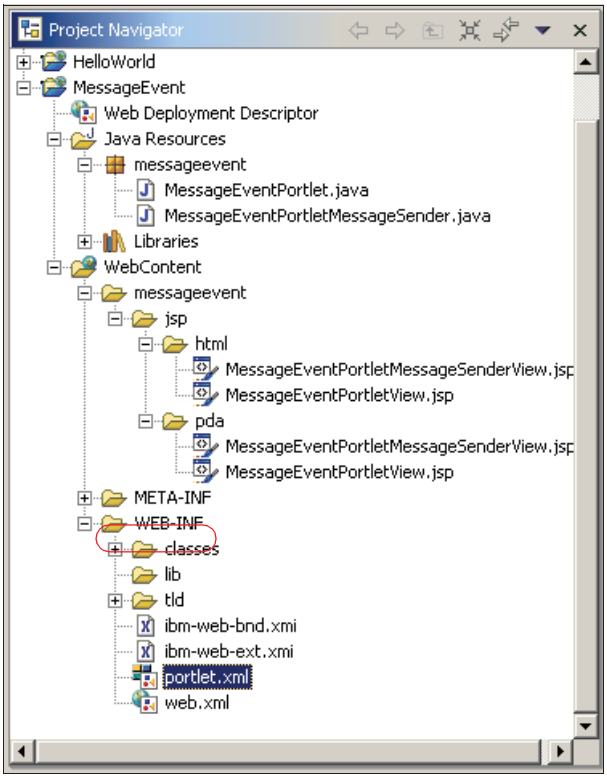


Figure 3-12 Opening portlet.xml

2. In the portlet.xml editor, expand Portlet Application and select **messageevent.MessageEventPortletMessageSender**.

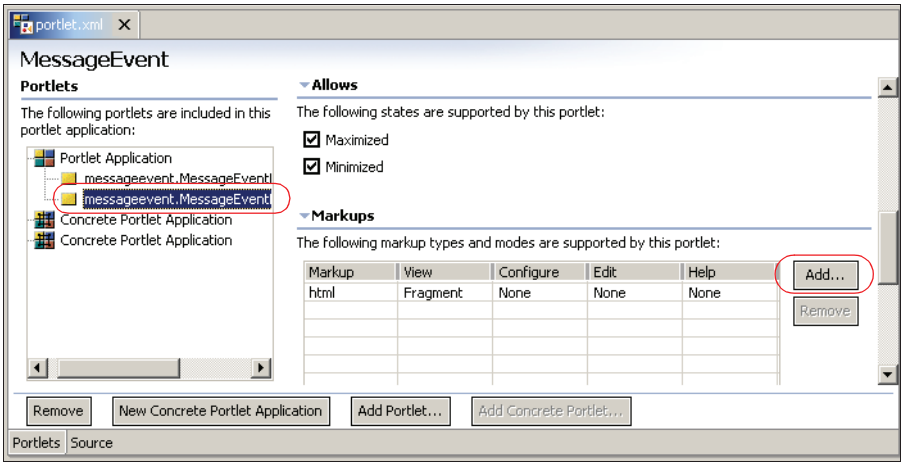


Figure 3-13 Portlet.xml editor

3. Scroll down to reach the markups section of the editor. Click the **Add** button as shown in Figure 3-13.
4. Add pda markup.

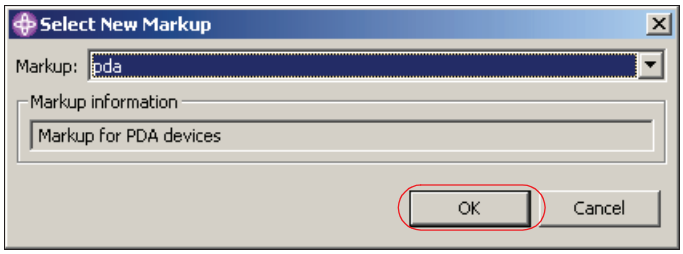


Figure 3-14 Adding portlet.xml

5. Save your changes.

3.4 Run the project

Follow these steps to run this project:

1. Make sure you have saved the changes to all the files you have modified.
2. Right-click the project name and select **Run on Server**.

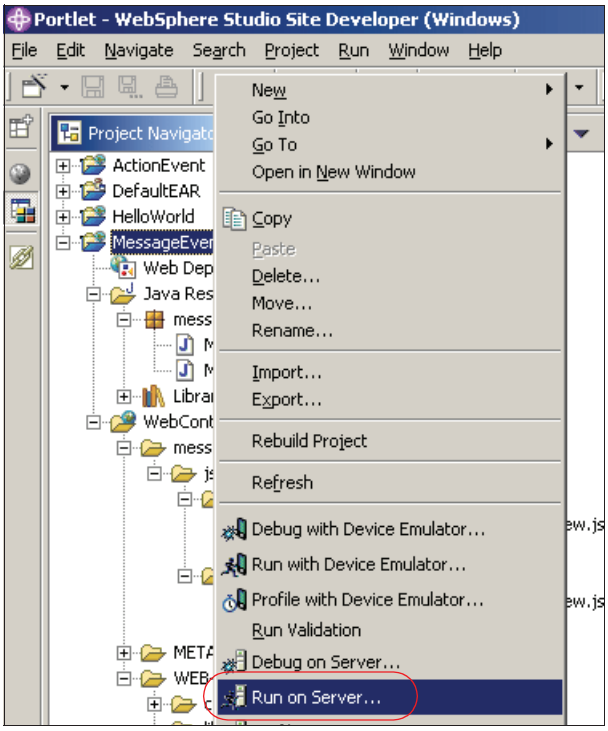


Figure 3-15 Running the portlet

3. Because you have already configured your project to run on the test environment with the steps at the beginning of this chapter, you can use the existing test environment to run your project. Click **Finish** to run your project in the test environment.
4. The project will initially load in the Web browser. When it loads, the message viewer portlet will display “A message has not been sent yet.” Enter a message in the message sender sample and click **Submit**.

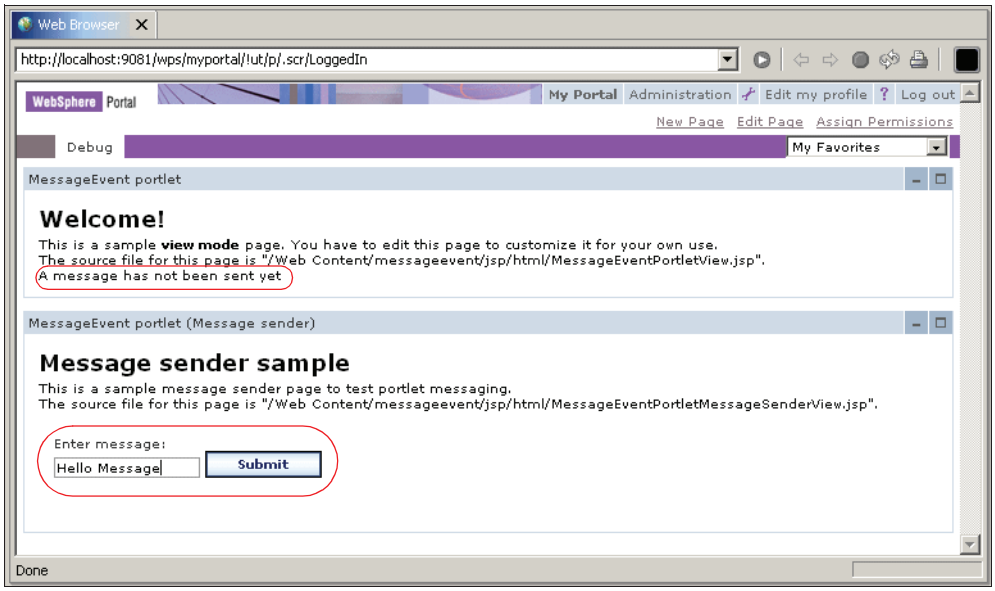


Figure 3-16 Sending a message

5. View your message in the message viewer portlet.

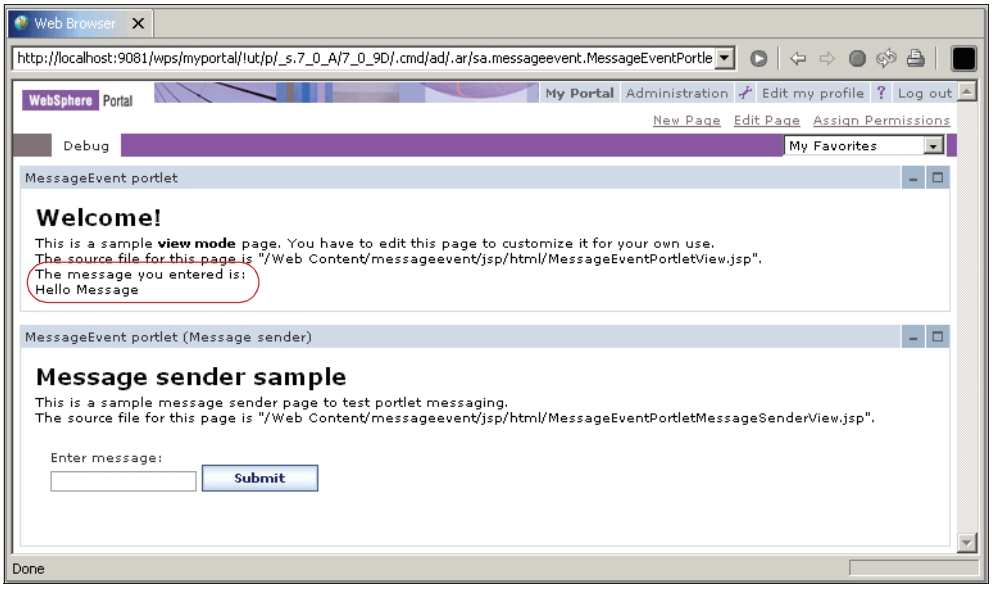


Figure 3-17 Displaying a message

Run the project in the Palm Simulator

Follow these steps to view your portlet project in the Palm Simulator:

1. Open the Palm Simulator and click **Web** to open a Web browser on the device.



Figure 3-18 Opening the Palm Web browser

> **Note:** You must configure the Palm Simulator to run the Web browser and access the Internet before you are able to perform these steps.

2. Click **Connect** to access the Internet.
3. Click the **globe** in the upper right corner to enter a URL. Enter the following URL to access your project, then click **Go**:
   `http://localhost:9081/wps/portal/!ut/p/.cmd/LoginUserNoAuth?userid=wpsadmin&password=wpsadmin`

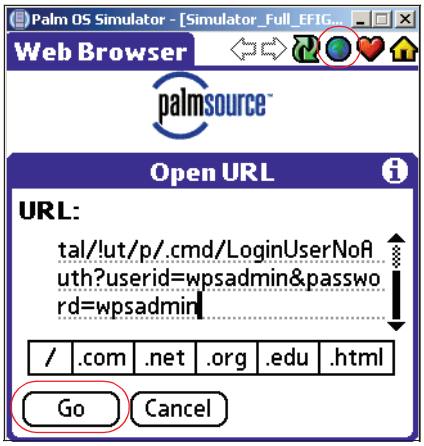


Figure 3-19 Opening the URL

4. Click the icon for **MessageEventPortlet (Message sender)**.



Figure 3-20 PDA main screen

5. Enter a message and click **Submit**.



Figure 3-21 Sending your message

6. After you submit the message, exit the message sender portlet by click the **X** in the upper right corner. This will bring you back to the main PDA screen.

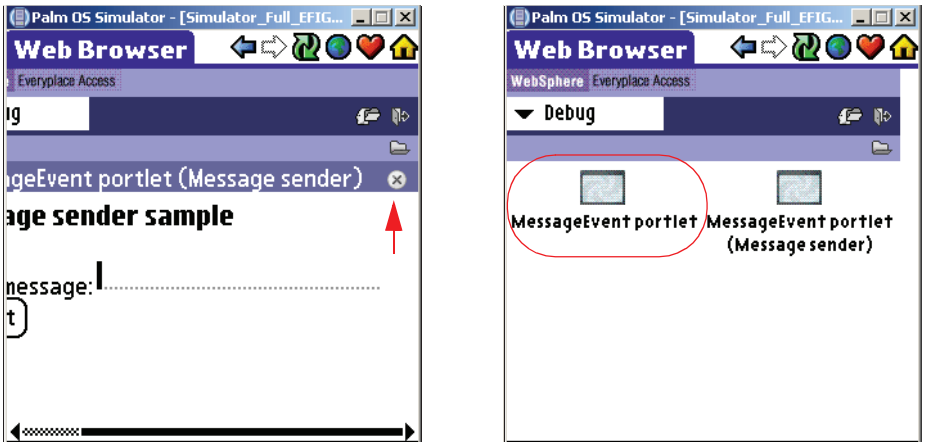


Figure 3-22 Exiting the sender portlet and returning to the main screen

7. Click the **MessageEvent portlet** icon.
8. View your message in the message receiver portlet.

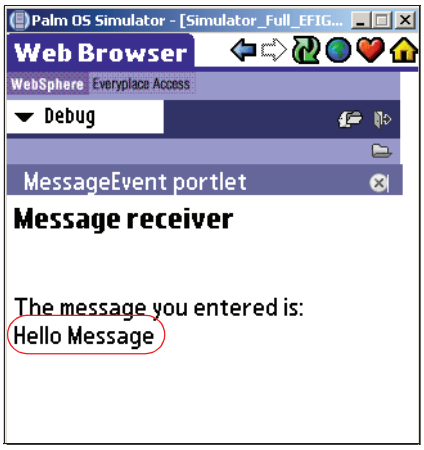


Figure 3-23 Viewing your message

4

National Language Support (NLS)

In order to make a portal accessible and attractive to a wider audience, it is necessary to provide the portal and its components in multiple languages. The WebSphere Portal architecture makes it easy and efficient to provide an NLS enabled portal. The enablement can be performed during development, deployment or runtime with the proper design decisions up front.

There are several ways to accomplish internationalization. One way is to translate and maintain separate JSP and HTML pages within the folder structure of your portlet application. Another way is to use property bundles which contain key-value pairs for specific translated strings that you can use in your portlet. With both of these approaches, the portal server handles which translated content to present to the user based on either the default locale setting for the portal or the locale preference stored in the user profile.

This chapter will guide you through several approaches to NLS enablement.

- Using resource bundles
- Translating whole resources
- NLS administration
- Working with characters
- NLS best practices

© Copyright IBM Corp. 2005. All rights reserved.

4.1 Resource bundles

A resource bundle is a simple text file that contains key-value pairs. The key is used by a Java class to retrieve a locale-specific value. To provide support for a new locale, you need only create a new resource bundle with the same key names and translated values.

Example 4-1 demonstrates a base resource bundle. Example 4-2 demonstrates the resource bundle translated for Spanish. Notice the key names do not change, only the value is translated.

Example 4-1 NLSExample.properties resource bundle

```
welcome = hello
goodbye = goodbye
message = This is the NLSExample portlet
```

Example 4-2 NLSExample_es.properties resource bundle

```
welcome = hola
goodbye = adiós
message = éste es el portlet NLSExample
```

The file name of the resource bundle is very important. The file must of type *properties*. All translated copies of the default resource bundle must include the locale in their title. This is illustrated in Figure 4-1.

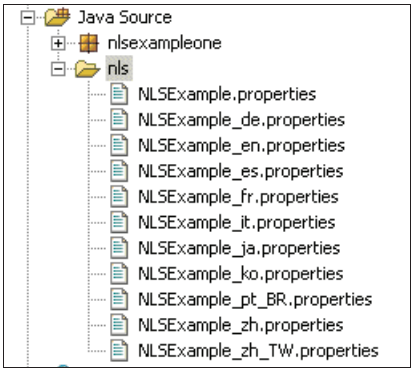


Figure 4-1 Translated resource bundles

The name is important because the Portal will locate the appropriate bundle for you based on the locale you provide. You need only provide the base name of the bundle and it will append the appropriate locale. The Group 1 locales are listed in Table 4-1.

Table 4-1 Group 1 languages

Locale Code	Language
ar	Arabic
cs	Czech
da	Danish
de	German
el	Greek
en	English
ru	Russian
sv	Swedish
es	Spanish
tr	Turkish
fi	Finnish
fr	French
zh	Simplified Chinese
zh_TW	Traditional Chinese
hu	Hungarian
it	Italian
iw	Hebrew
ja	Japanese
ko	Korean
nl	Dutch
no	Norwegian
pl	Polish
pt	Portuguese
pt_BR	Brazilian Portuguese

4.1.1 Creating resource bundles in WebSphere Studio

The resource bundles need to be created in the Java Source directory as illustrated in Figure 4-1 on page 88. Though not required, as a matter of good practice, you should place the files in a dedicated directory such as nls. To create a new resource bundle in WebSphere Studio, open the navigator view in the portlet perspective. Locate the Java Source directory of the portlet you are enabling and right-click. From the context menu, select **New -> Other**.

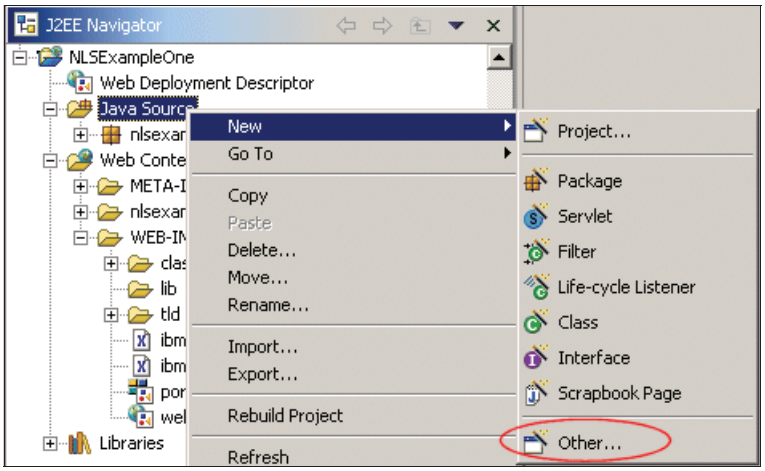


Figure 4-2 Creating new folder

Select **Simple** in the left panel and **Folder** in the right panel and click **Next**.

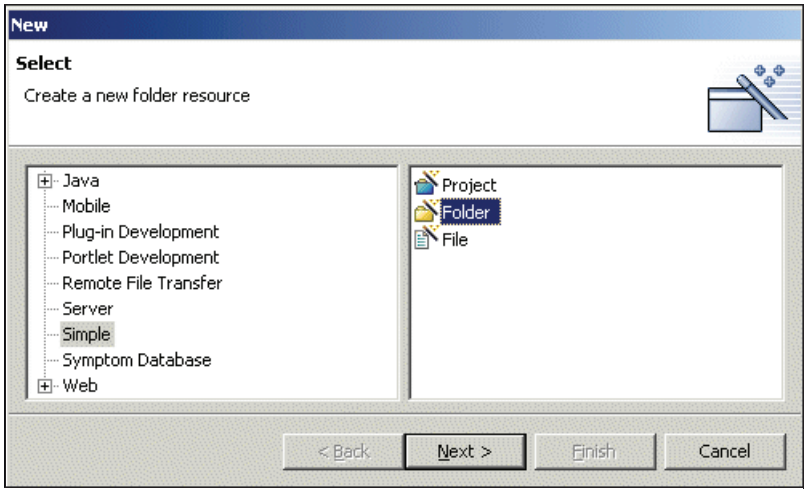


Figure 4-3 Creating nls folder

Enter the name of the new folder, typically `nls` as shown in Figure 4-4.

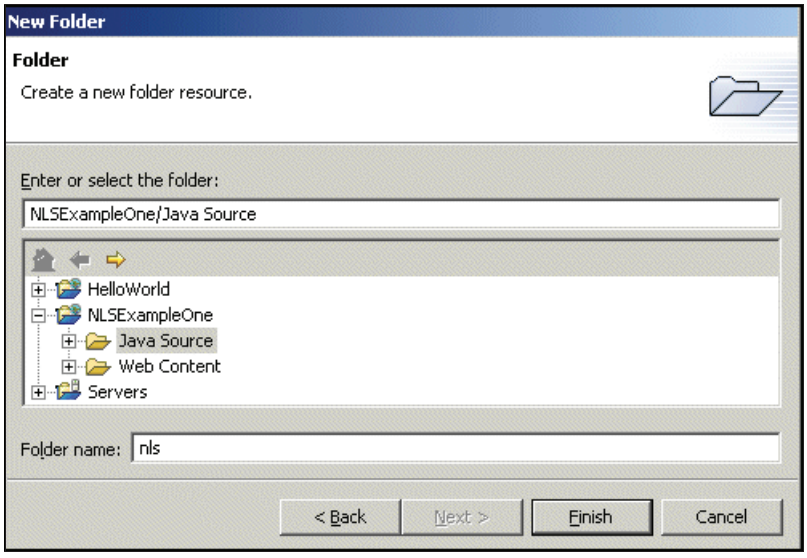


Figure 4-4 Creating the nls folder in WebSphere Studio Application Developer

In the nls folder, you need to create the default properties files. Select the **nls** folder and right-click. From the context menu, select **New -> Other -> Simple -> File**. Be sure the correct directory is selected and enter the name of the default properties file as illustrated in Figure 4-5. Do not include any language codes in the name, or include any spaces in the name of the resource bundle.

When you are done, double-click the properties file in the navigator to open the simple text editor. Using the text editor, define your keys and the default values, such as those shown in Example 4-1 on page 88. Use **CTRL-S** to save the file.

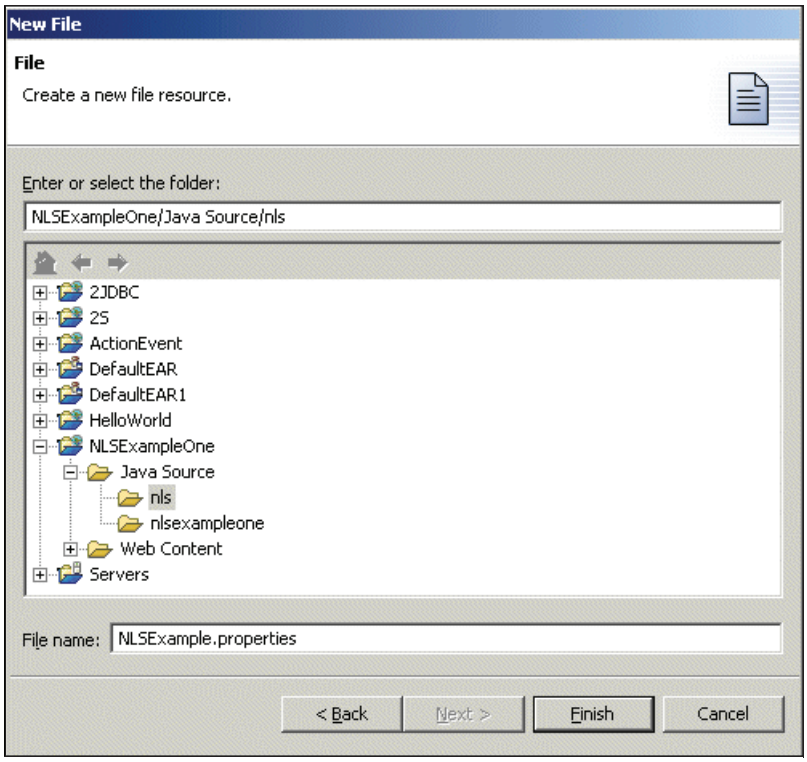


Figure 4-5 Creating the default properties bundle

Tip: If you create your nls folder directly in the Java Source subdirectory, WebSphere Studio Site Developer deletes this folder when it rebuilds the project.

4.1.2 Translating resource bundles

Once you have defined your default resource bundle with all the keys that will be used by the portlets and JSPs in your application, you must provide translations. It is possible to use the copy functionality in WebSphere Studio. However, there are several reasons you may choose not to. It is a cumbersome process in that simple CTRL-C commands are not recognized when copying whole files. Also, renaming requires a selection from the context menu. While these may seem trivial issues, when creating dozens of resources bundles, they can be frustrating.

Instead, you may find it easier to work directly with the source files on the file system. Locate the directory containing the current workspace. You can obtain

this path by right-clicking the portlet application and selecting **Properties** from the context menu. The Info option will display the file system location of the application. This is illustrated in Figure 4-6.

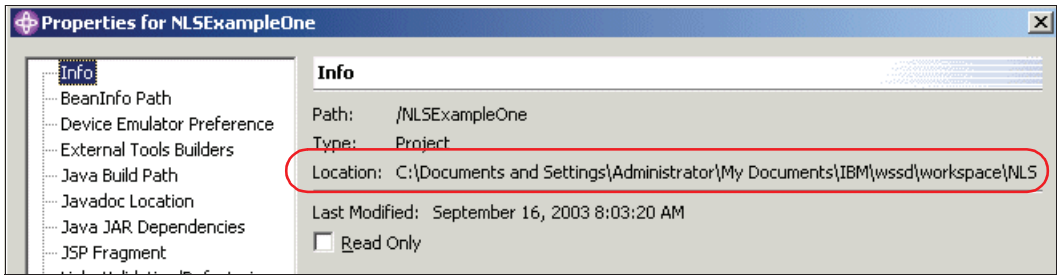


Figure 4-6 Locating the application on the file system

Open the directory containing the application and use the normal system copy/paste and rename functionality to create the new resource bundles. Each new bundle should have a unique locale appended. In practice, you may at development time only have the default and English properties files. This same approach can later be used to import translated files received from an outside source.

Once you have created the bundles you want, you need to make them available in the WebSphere Studio environment. To do this, simply select the **nls** folder, right-click and select **Refresh** as shown in Figure 4-7 on page 94.

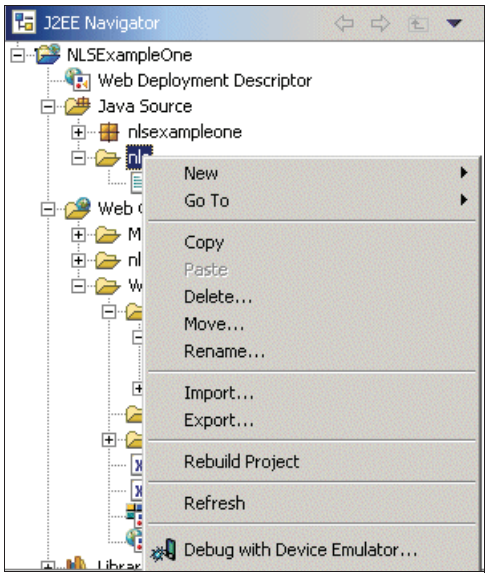


Figure 4-7 Loading resource bundles into WebSphere Studio Site Developer

When you are done, the folder should appear as in Figure 4-1 on page 88, depending of course on the number of languages you choose to support.

4.1.3 Accessing resource bundles in portlets

If you are printing out content directly from the portlet, you can use the portlet API to access the resource bundles quite easily. Most of your development will adhere to a good MVC approach; you can use this approach for setting the title, predefining parameters in a PortletURI or if you are providing some content via the beginPage or endPage methods.

The resource bundle is accessed via the PortletContext object's getText method as displayed in Example 4-3.

Example 4-3 getText API

```
PortletContext.getText("Bundle Base Name", "Key", Locale)
```

- `Bundle Base Name`: the first parameter indicates the base name of the resource bundle. The name includes the path relative to the classes directory as shown in Example 4-4 on page 95. The name does not specify the locale suffix or the properties file type. If the base file name cannot be found, or the key is not present in the properties file, a PortletException will be thrown.

- `Key`: this parameter maps to a key value in the properties file. If the key is not found, a PortletException is thrown.
- `Locale`: this is used by the Portal to create the complete resource bundle name. You are free to use any locale you like but to ensure the user's locale is returned, the code in Example 4-4 works well. The getLocale method returns the preferred locale for the user. The Portal Server determines the locale by first retrieving the user's preferred language set during registration. If the preferred language is not set, the locale is retrieved from the accept-language header supplied by the client.

Example 4-4 Accessing resource bundles via the API

```
getPortletConfig().getContext().getText("nls.NLSExample", "welcome",
request.getLocale());
```

4.1.4 Accessing resource bundles in JSPs

When you employ a well designed MVC approach to your portlet development, the vast majority of NLS enablement work will need to take place in the view space. This section will guide you through providing locale-specific strings in a JSP. Section 4.2, "Translating whole resources" on page 97 will guide you through providing a unique JSP for each locale you choose to support.

To access resource bundles in JSP, you need to include the JSP Standard Tag Library. Right-click in your portlet application, select **Properties** and then **Web Project Features**. In Available Web project features, check the **JSP Standard Tag Library** and click **OK** as shown in Figure 4-8 on page 96.

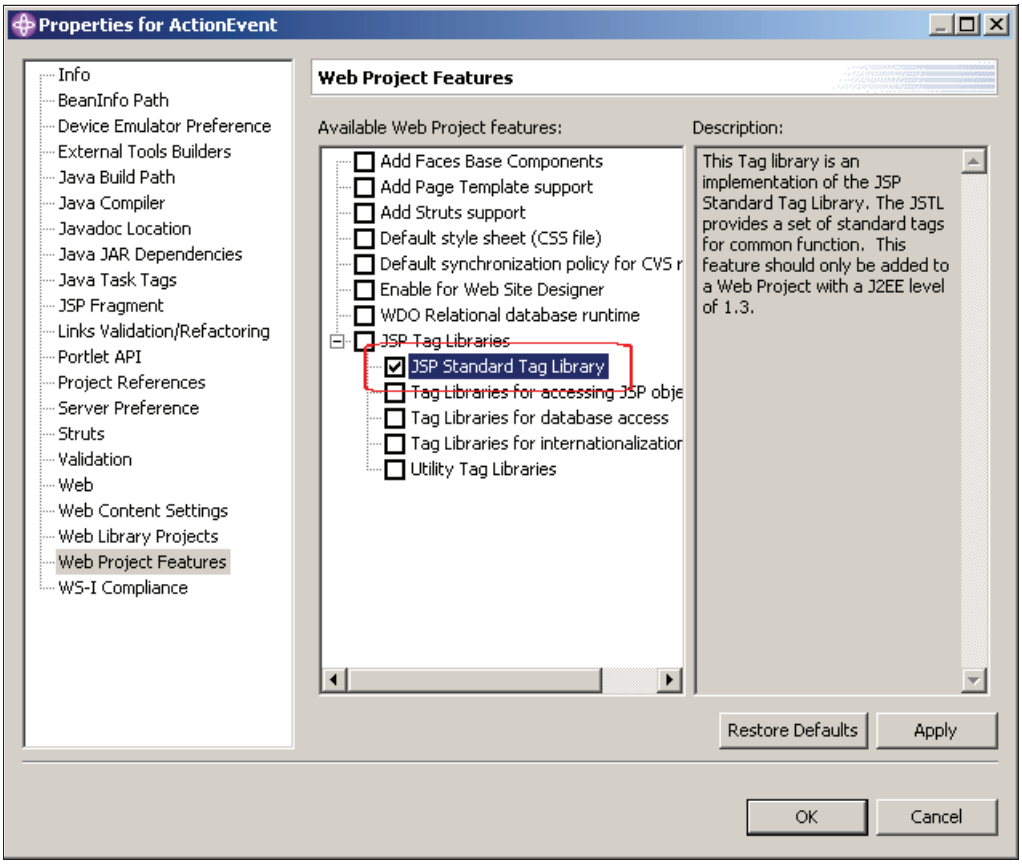


Figure 4-8 Including the JSP Standard Tag Library

Your JSP files can access resource bundles in two ways, as shown in the following examples.

Example 4-5 Accessing resource bundles in a portlet JSP

```
<%@ taglib uri="http://java.sun.com/jstl/fmt" prefix="fmt" %>
<fmt:setBundle basename="nls.NLSExample"/>
<fmt:message key="message"/>
```

Example 4-6 Accessing resource bundles in a portlet JSP

```
<%@ taglib uri="http://java.sun.com/jstl/fmt" prefix="fmt" %>
<fmt:bundle basename="nls.NLSExample">
<fmt:message key="message"/>
</fmt:bundle>
```

As with specifying bundles in portlet code, the bundle name here must include the package name relative to the classes directory.

If the key cannot be located in the properties file, you will see the key written between question marks.



Figure 4-9 Key not found in a properties file

4.2 Translating whole resources

If the entirety of your JSP requires translation, you may find programmatically accessing resource bundles impractical. In practice, you will find that help JSPs, for example, contain little or no code and as such can be completely translated without incurring the runtime expense of NLS enablement.

WebSphere Portal facilitates this approach by allowing you to organize translated files in a predictable directory structure. Portal will then take responsibility for locating the correct file at runtime. This facility is also available for multiple markup support. Your directory structure should reflect Figure 4-10 on page 98.

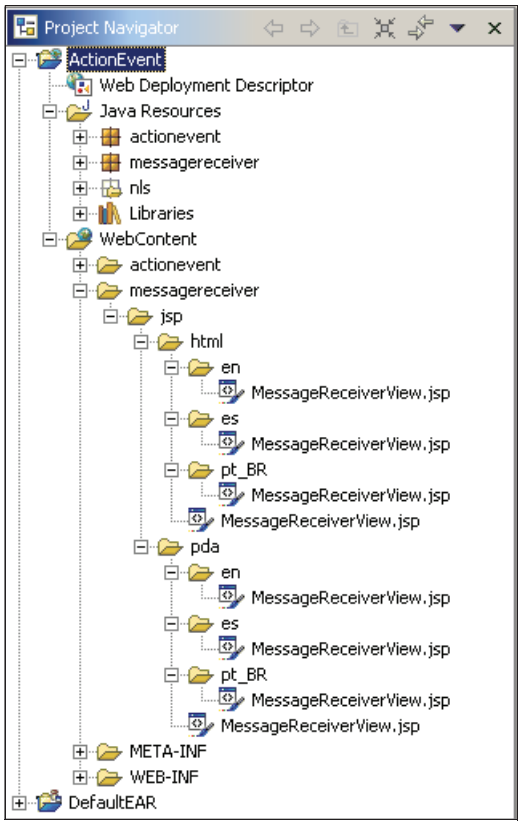


Figure 4-10 NLS directory structure

Each locale you support must have its own folder and contain whatever fully translated resources you want the portal to serve. If the portal cannot find the requested resource in the appropriate directory, it will attempt to locate the default by searching one level higher. If no default resource is located up the directory tree, an exception is thrown.

To retrieve the translated resource, simply use the include method of the PortletContext object. Do not specify the NLS directory structure. This code is illustrated in Example 4-10. You should notice that there is in fact nothing unique about calling a translated JSP and calling the simple JSPs. All the work is done by the portal.

Example 4-7 Including translated JSP files

```
getPortletConfig().getContext().include("/messagereceiver/jsp/MessageReceiverVi
ew.jsp", request, response);
```

4.3 NLS administration

Certain aspects of NLS enablement can be configured via the Administration window in WebSphere Portal. While this section will guide you through these features, bear in mind that the administration does not replace the developer's responsibility for designing and incorporating NLS enablement.

4.3.1 Portlet NLS administration

The Manage Portlets page allows you to set locale-specific titles for portlets. You cannot add support for new locales. Only locales specified by the portlet.xml deployment descriptor can be adjusted. Furthermore, you cannot change the default locale specified by the portlet.xml. To access the titles, log in as the administrator and navigate to the Manage Portlets page in the Administration window, as illustrated in Figure 4-11.

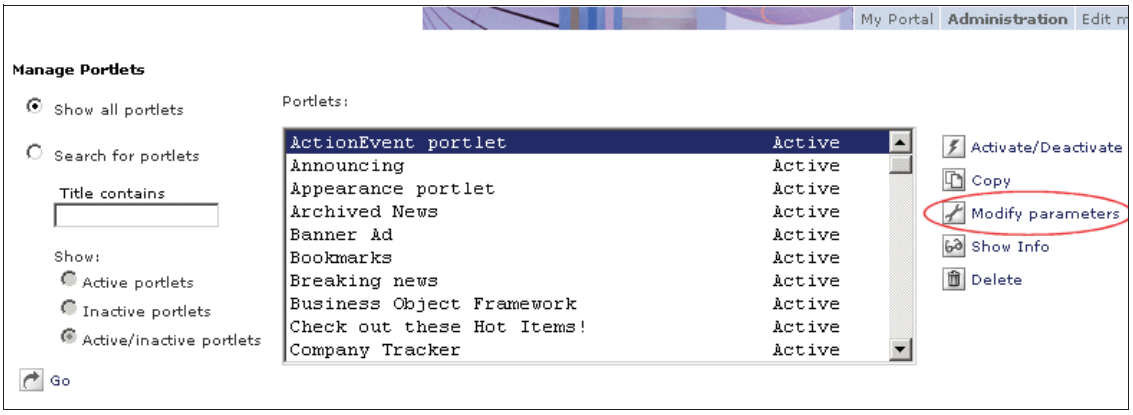


Figure 4-11 Working with portlets

Locate the portlet you want to work with and select the **Modify parameters** link. By default, the resulting window, shown in Figure 4-12 on page 100, will only display the Group 1 languages indicated for support by the portlet.xml. To add new languages in the administration space, you need to add that language to the file language.properties and to the language table in the Web Member Services database if you use the Member Subsystem. Then you have to insert resource bundles, with an appropriate name, in the directory located at /wp_root/shared/nls. The directory where you have to store the JSP will depend on how the portal locates your JSP for rendering its content.

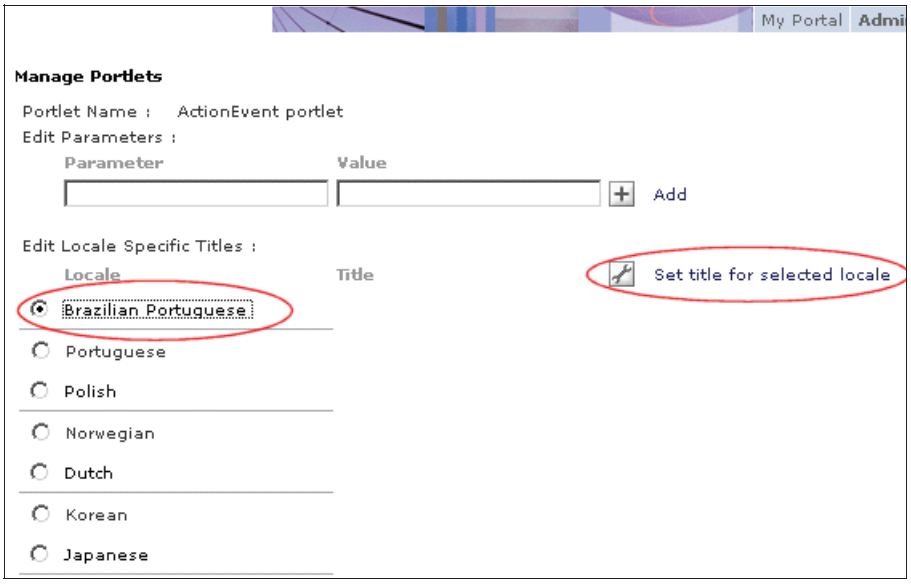


Figure 4-12 Supported languages of the selected portlet

To change the title specified by the portlet.xml, select the portlet's associated radio button and select the **Set title for selected locale** link as shown in Figure 4-12. Enter the new title in the resulting window shown in Figure 4-13.

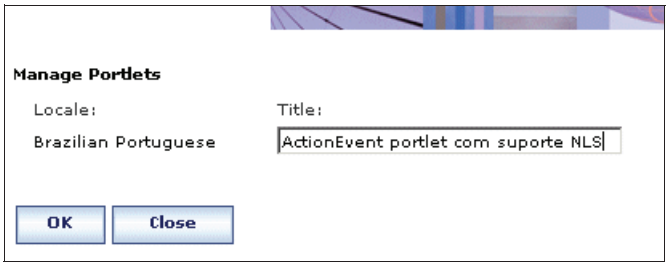


Figure 4-13 Setting the locale-specific title

You may have noticed that only the title can be adjusted via the Administration window.

Again, it is important to stress that by default, the administrator will only be able to set the title for Group 1 locales explicitly specified in the portlet.xml. To add support for a locale in the portlet.xml in WebSphere Studio, open the portlet.xml editor and select the concrete portlet you want to work with. This is illustrated in Figure 4-14 on page 101.

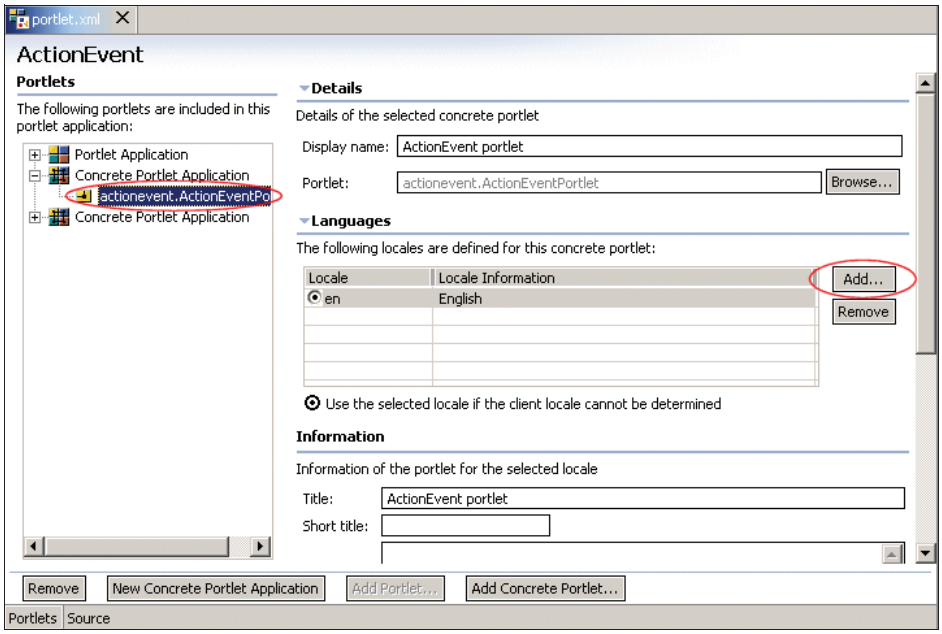


Figure 4-14 Adding locale support to a concrete portlet

To add a new locale, select the **Add** button in the locale section as shown in Figure 4-14. In the resulting dialog, you can select the locale from the drop-down list or enter the country (or region) code if you know it. This is illustrated in Figure 4-15. If the locale you choose is already defined in the portlet.xml, you will be prevented from adding it again.

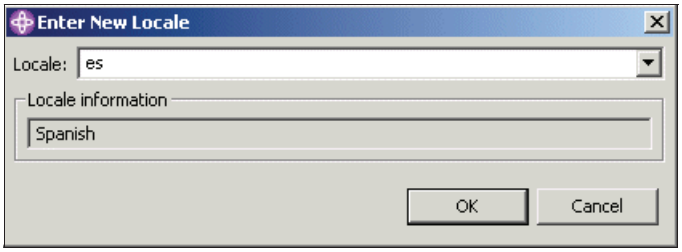


Figure 4-15 Specifying a new locale

All portlets must have a default language specified in the deployment description, otherwise the portlet cannot be installed.

4.3.2 Portal NLS administration

While it is the developer's responsibility to carefully consider the NLS support portlet applications will bring, the administrator is responsible for ensuring the Portal itself properly supports multiple languages.

Some of the basic settings for NLS enablement include setting page, theme and skin names properly, configuring or maintaining property files and incorporating support for new languages.

4.3.3 Setting NLS titles

To set locale-specific titles for a page, navigate to **Administration -> Portal User Interface -> Manage Pages**, then locate your page and click the **Edit Page Properties** icon. Display the advanced options as illustrated in Figure 4-16.

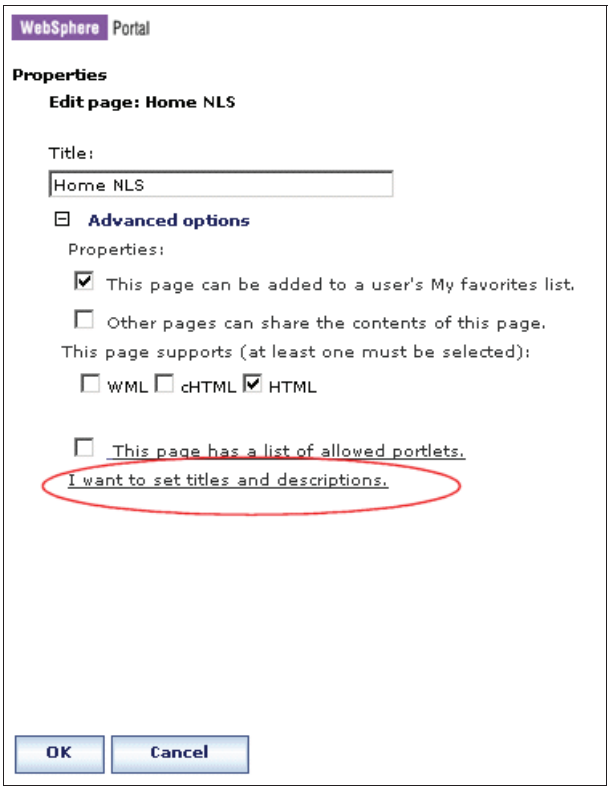


Figure 4-16 Editing page properties

In the resulting window, select the **Edit** icon of the language for which you want to set a title and/or description. Enter the new title and/or description and select **OK**.

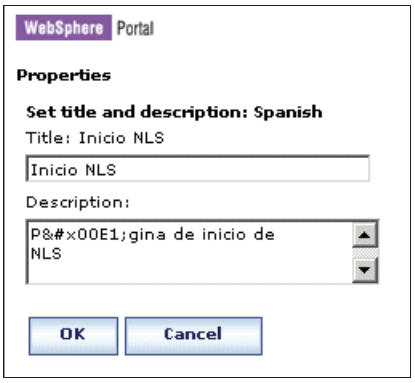


Figure 4-17 Using unicode values

Now you will see the new title and description in the list.

If you need to enter characters that you cannot generate with your keyboard, such as Japanese characters, written accents, etc., you will need to discover the unicode values and enter them using entity references as illustrated in Figure 4-17. The four character unicode values can be found at `http://www.unicode.org/charts`. The entity reference syntax is `&#x`***`0000`***`;` where ***`0000`*** represents the unicode character you want to display. If you have access to the interpreted unicode character, you can copy and paste it in the text field as well.

4.3.4 Adjusting Portal resource bundles

There are several resource bundles that are used by the portal server to present locale-specific messages. Be aware that changes to a property file are not recognized until the portal is restarted. All the properties files listed below can be found in /wp_root/shared/app/nls.

- button.properties
- commonAdmin.properties
- problem.properties
- field.properties
- engine.properties
- CSRes.properties (not available in Enable)
- titlebar.properties
- registration.properties
- LocaleNames.properties

- pbruntime.properties
- virtual_principals.properties

4.4 Working with characters

Typically, it will not be the developer's responsibility to provide the translations necessary to provide NLS enablement. Once your base resource bundles and/or static files have been created, the translation process should be completed by a language expert. However, during development, you may need to include some characters that cannot be created by the typical English keyboard. To include these characters, make use of the unicode mappings. Refer to http://www.unicode.org for the current character mappings.

Example 4-8 Using unicode in NLSExample_es.properties resource bundle

```
welcome = hola
goodbye = adi\u00F3s
message = \u00E9ste es el portlet NLSExample
```

4.5 NLS best practices

You may want to consider the following issues when implementing NLS in portlet applications:

- Make the decision to use the API or translated resources early. This decision will play a large role in the design and development of your View components.
- Do not commit entirely to one approach. For example, it may make sense to translate your View JSPs at runtime and have your Help JSPs fully translated since they are simple text.
- Plan for NLS enablement from the beginning. Though you may not have access to the translated values during development, building the default resource bundle as you iterate through the development will make future NLS enablement much easier and virtually painless.
- Be conscious of character-locale ratios. If you are developing in English, be aware that translations into other languages such as German or Spanish may require more screen real estate. A number of API facilities are available for you to determine the current locale.
- Do not rely on an administration implementation of NLS. The NLS enablement facility for portlets is limited and there is no guarantee or check system. To implement dynamic NLS titles, consider implementing the

PortletTitleListener interface and generating the title content via JSP or HTML files.

- Leave translations to language experts. With proper design, planning and construction of your portlet applications, there should be little to no effort involved in incorporating support for new languages.

4.6 Sample scenario: NLS bundles

In this sample scenario, NLS bundles will be created to support multiple languages. Once you have done this using WebSphere Studio, the JSP that delivers markup content in View mode for portlet MessageReceiver will be enhanced to access the NLS bundles. In addition, PDA support will be included.

The Portlet Messaging application sample scenario from Chapter 3, “Portlet messaging” on page 63 will be used as a base application to add NLS support. The scenario is illustrated in Figure 4-18.

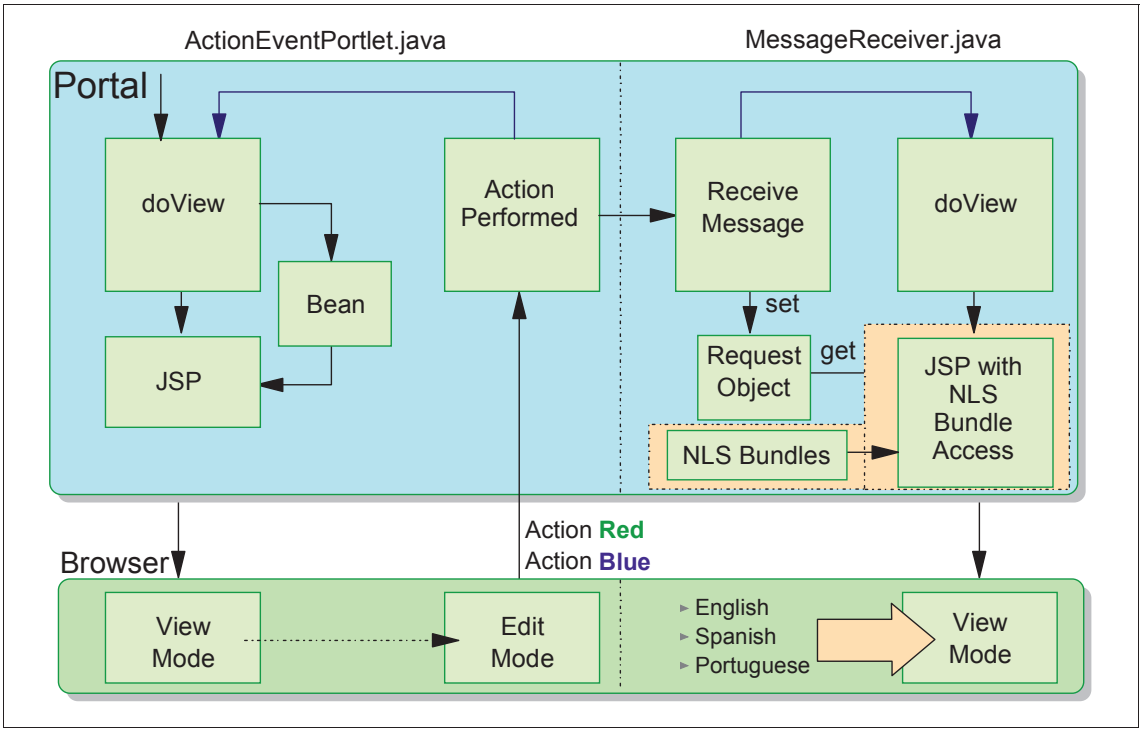


Figure 4-18 National Language Support (NLS) scenario

The resource bundle is accessed via the PortletContext object's getText method and you will need to provide the following:

- Bundle Base Name: the first parameter indicates the base name of the resource bundle. The name includes the path relative to the classes directory. The name does not specify the locale suffix or the properties file type. If the base file name cannot be found, or the key is not present in the properties file, a PortletException is thrown.
- Key: this parameter maps to a key value in the properties file. If the key is not found, a PortletException is thrown.

In addition, the locale is used by the Portal to select the proper language bundle. However, you cannot set this value when invoking NLS bundles from JSPs.

4.6.1 NLS bundles

In this section, a sample portlet application is used to add NLS support. The portlet application will be enhanced to support NLS. Follow these steps:

1. If needed, start the IBM WebSphere Studio Site Developer. Click **Start -> Programs -> IBM WebSphere Studio -> Site Developer 5.1.1**.
2. You will create a new folder with the name nls to store the resource bundles. The following resource bundles will be imported into this folder:
   - NLSLab.properties (default)
   - NLSLab_en.properties (English)
   - NLSLab_es.properties (Spanish)
   - NLSLab_pt_BR.properties (Brazilian Portuguese)
3. Select your **ActionEvent/Java Source** folder.

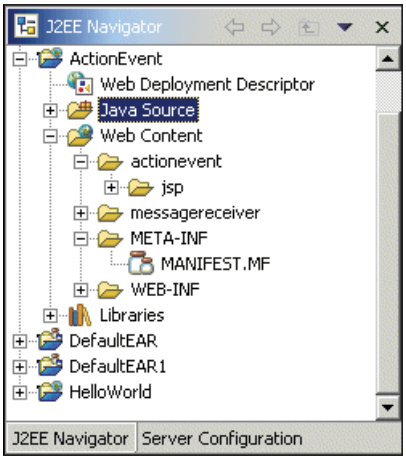


Figure 4-19 Select Java source to create an nls folder

4. Select **File -> New -> Folder**.

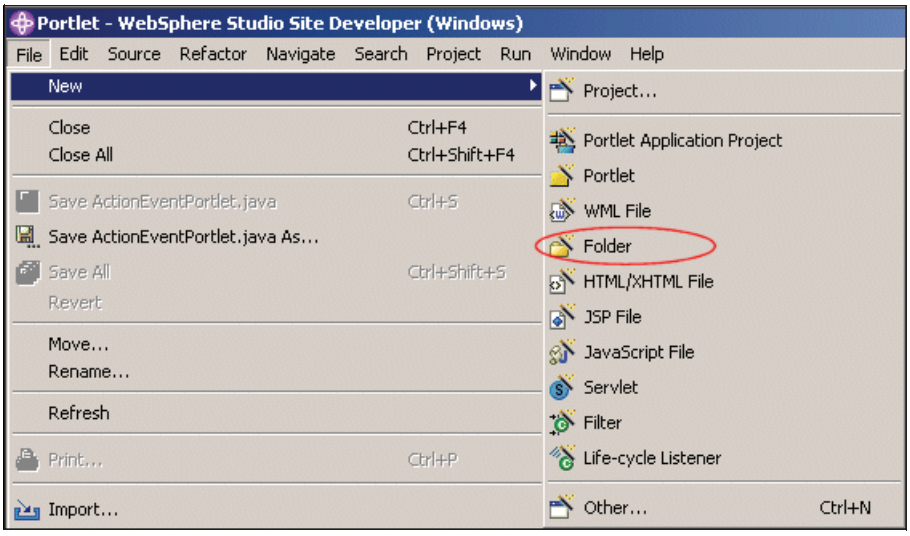


Figure 4-20 Select a new folder

5. Enter `nls` for the Folder name field then click **Finish**.

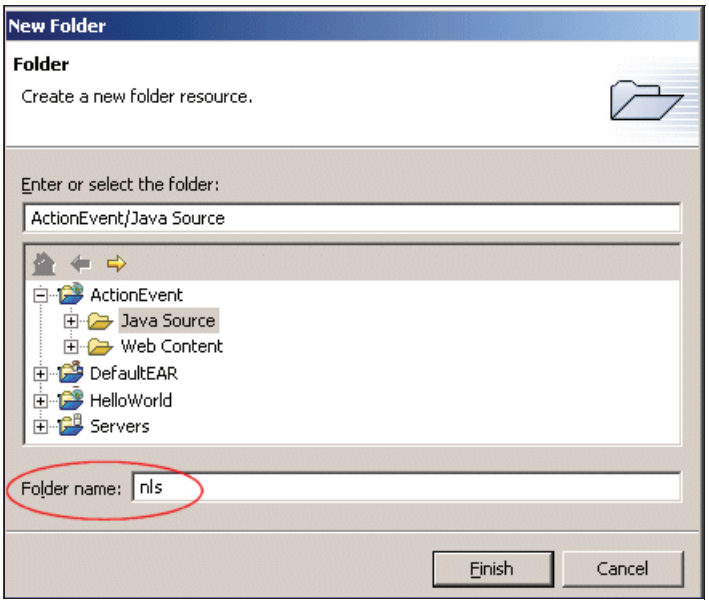


Figure 4-21 Enter a name for the new folder (nls)

6. Your directory structure should now look as shown in Figure 4-22.

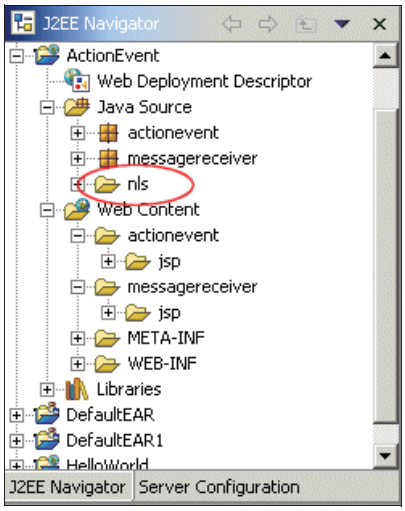


Figure 4-22 New nls folder

7. Select the new **ActionEvent/Java Source/nls** folder.
8. Click **File -> Import**.

9. Select **File System** and click **Next**. For example browse to:

   ```
   C:\LabFiles\NLSLab\Bundles
   ```

   Note: The sample scenario included in this chapter requires that you download the sample code available as additional materials. See Appendix B, “Additional material” on page 443.

10. Select all four properties files and click **Finish** to import.

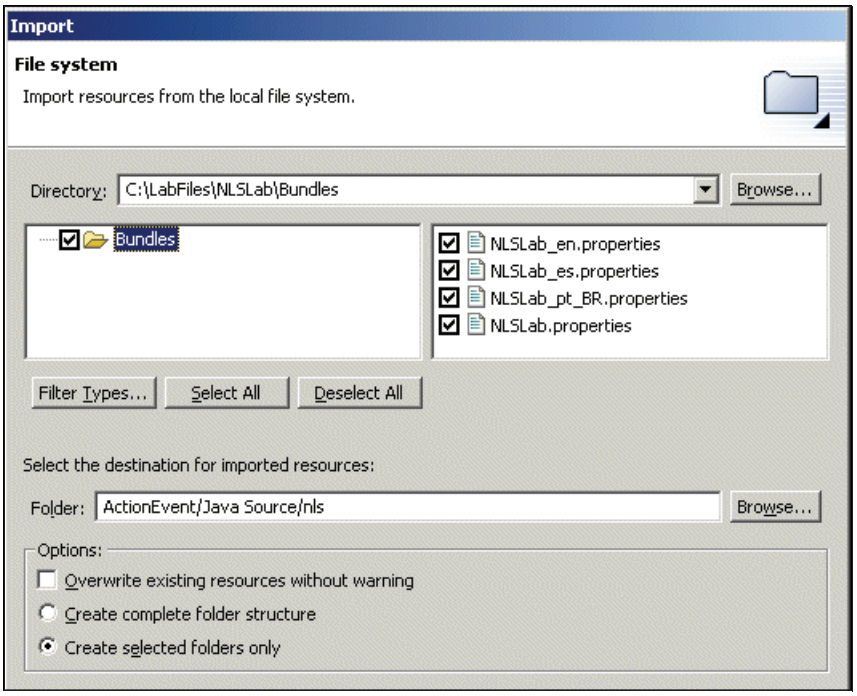


Figure 4-23 Importing the bundles

11. View the files in the nls folder by double-clicking them. Notice how they are structured.

Example 4-9 NLSLab.properties (default)

```
readystatus=Ready to receive message
receivedstatus=Received a message
viewmode=Operating in View mode
redColor=RED
blueColor=BLUE
```

Example 4-10 NLSLab_en.properties (English)

```
readystatus=Ready to receive message
```

```
receivedstatus=Received a message
viewmode=Operating in View mode
redColor=RED
blueColor=BLUE
```

Example 4-11 NLSLab_es.properties (Spanish)

```
readystatus=Listo para recibir mensaje
receivedstatus=Mensaje recibido
viewmode=Operando en modo de visualizacion
redColor=ROJO
blueColor=AZUL
```

Example 4-12 NLSLab_pt_BR.properties (Brazilian Portuguese)

```
readystatus=Pronto para receber mensagem
receivedstatus=Mensagem recebida
viewmode=Operando em modo de Visualização
redColor=VERMELHA
blueColor=AZUL
```

4.6.2 Accessing NLS bundles from JSPs

In this section, you will update the JSP MessageReceiverView.jsp to display NLS content based on the locale value (English, Spanish or Brazilian Portuguese). In general, modifications to the JSPs are necessary to allow them to display language-specific content. Follow these steps:

1. First you need to make sure the JSP Standard Tag Library is available. For example include the JSP standard tag library in your project as follows:
   a. Right-click the **ActionEvent** project.
   b. Select **Properties**.
   c. Select the **Web Project Features** option from the list.
   d. In Available Web project features, check the **Include the JSP Standard Tag library** box.
   e. Click **OK**.
   f. Your JSP files can access resource bundles in two ways, as shown in the following examples.

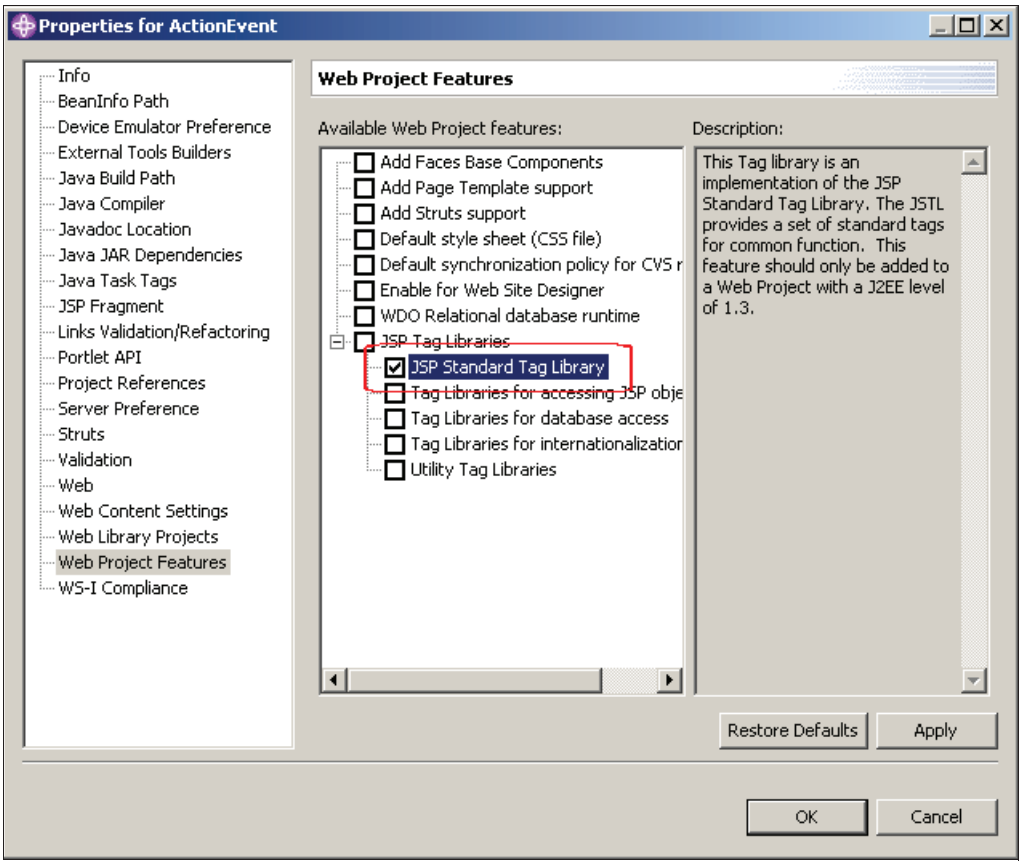


Figure 4-24 Including the JSP Standard Tag Library

2. Open the MessageReceiverView.jsp file in WebSphere Studio Site Developer. This file is located in the /Web Content/messagereceiver/jsp/html/ directory.

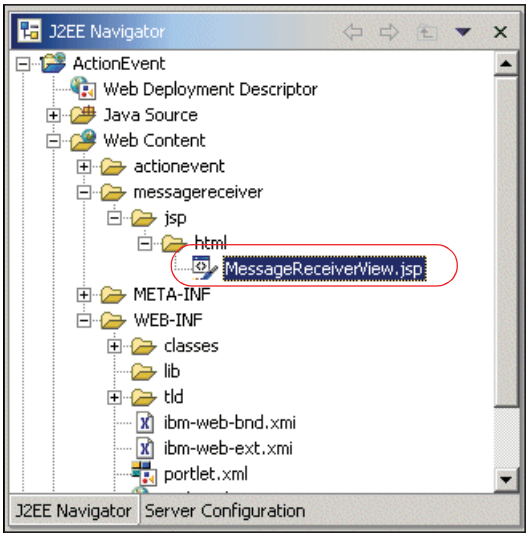


Figure 4-25 MessageReceiverView.jsp

3. In this JSP, you have the following text with static information:
   - `Ready to receive message`
   - `Received a message`
4. Add logic to display messages in the proper language. Updates to this JSP are highlighted in bold in Example 4-13.

Example 4-13 MessageReceiverView.jsp supporting NLS with bundles

```
<%@ page contentType="text/html" import="java.util.*, messagereceiver.*"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<%@ taglib uri="http://java.sun.com/jstl/fmt" prefix="fmt" %>
<portletAPI:init/>

<DIV style="margin: 6px">

<H3 style="margin-bottom: 3px">Welcome!</H3>
This is a sample <B>view mode</B> page. You have to edit this page to customize
it for your own use.<BR>
The source file for this page is "/Web
Content/messagereceiver/jsp/html/MessageReceiverView.jsp".

<fmt:setBundle basename="nls.NLSLab"/>

<br>
<% if (request.getAttribute("MyMessage") == null) { %>
```

```
   <B><fmt:message key="readystatus"/> ...</B>
<% } else { %>
   <B><fmt:message key="receivedstatus"/>:</B>
   <B><%= request.getAttribute("MyMessage") %></B>
<% } %>

</DIV>
```

5. Select **File -> Save All** to save all your changes to the project.

4.6.3 Running the NLS scenario

In this section, you will run the portlet application messaging scenario now enabled for NLS.

1. Stop the Test Environment server so that next time, the new properties files will be used.
2. Right-click **ActionEvent** and select **Run on Server**.

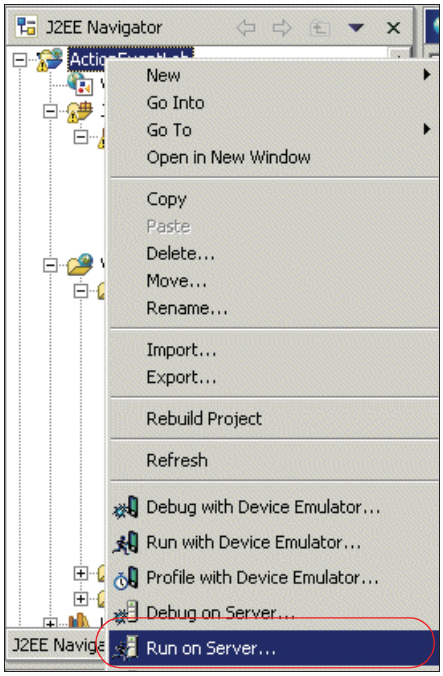


Figure 4-26 Running the NLS project

3. The message receiver portlet will now display its markup using NLS.

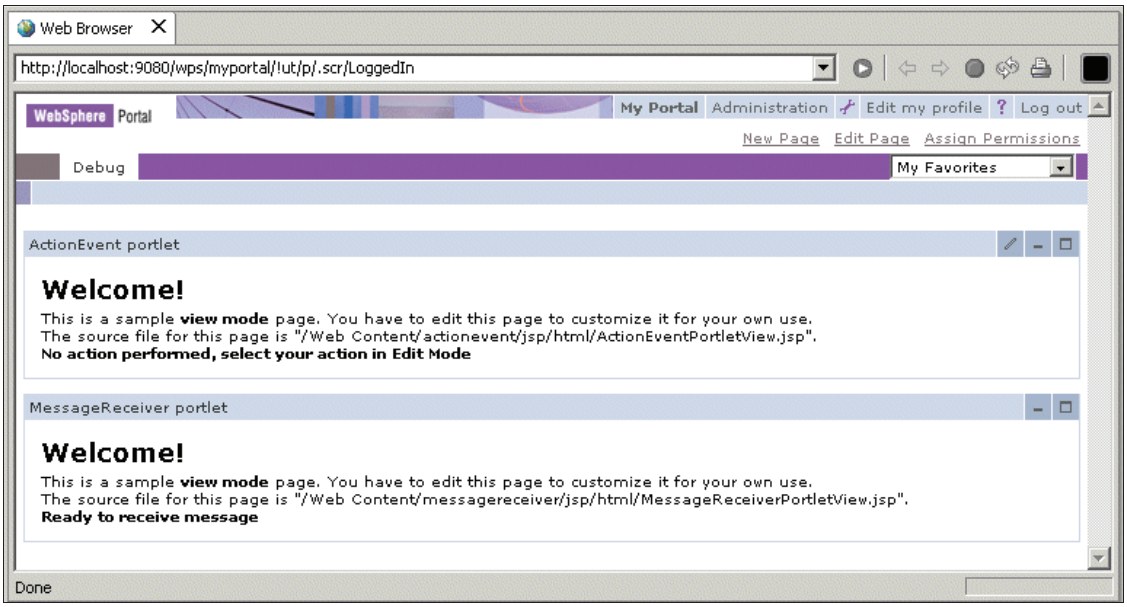


Figure 4-27 ActionEvent Portlet with no preferred language selected

4. Select a new locale value by clicking the **Edit my profile** icon to select a preferred language, as shown in Figure 4-28.

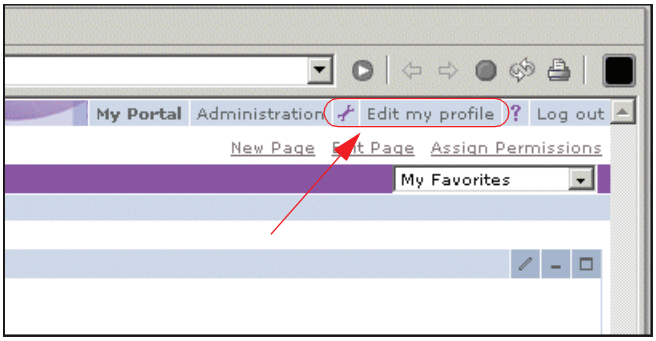


Figure 4-28 Edit my profile icon

For example, select **Brazilian Portuguese** as the preferred language for the wpsadmin user (default user for portlet development environment). It may be necessary to enter a first and last name before you can continue. Enter `wps` for both if this happens.

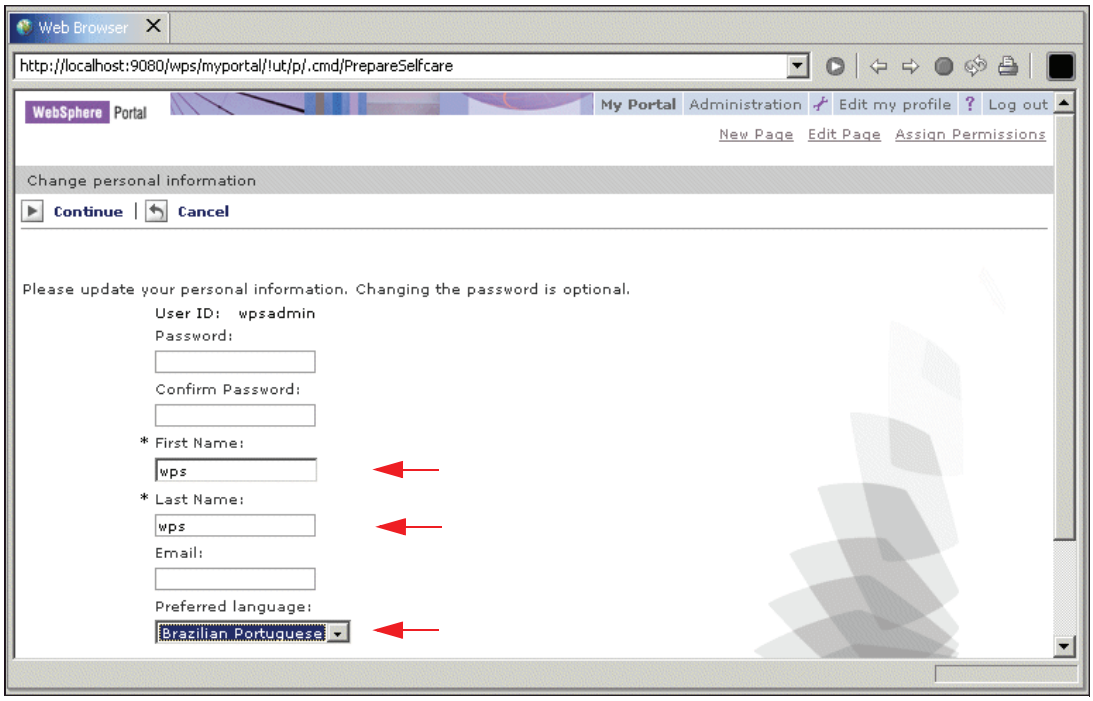


Figure 4-29 User profile

5. Click **Continue**. You will need to click **Continue** again to confirm your change and view the information displayed in the MessageReceiver portlet in Brazilian Portuguese.

   Note: You should also notice that when the language locale changes, the text of the WebSphere Portal menu at the top of the page also changes.

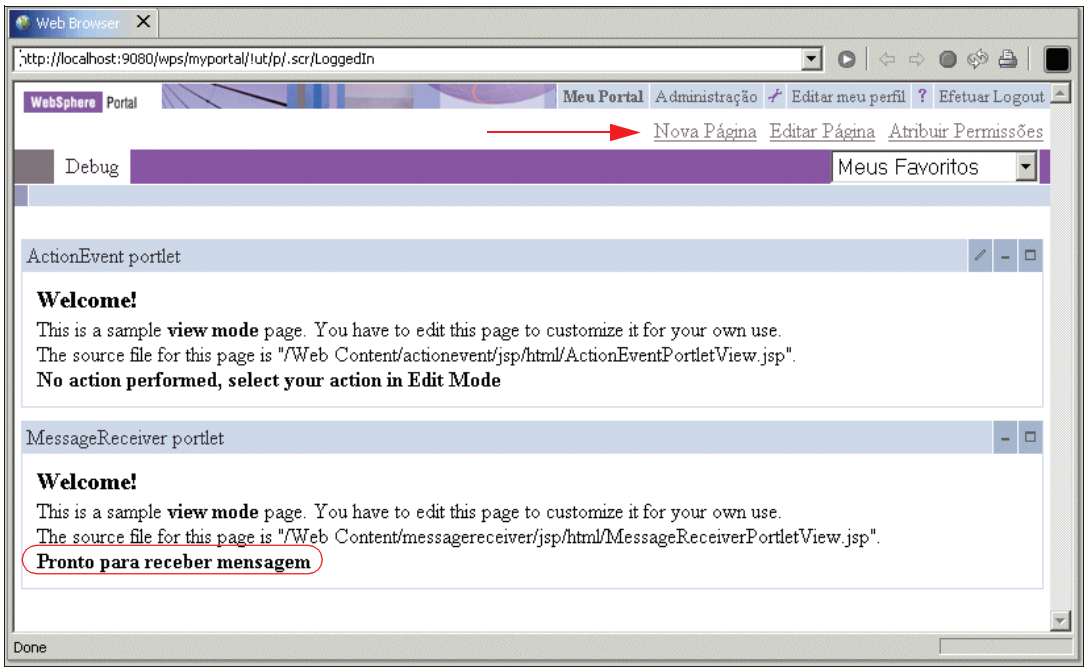


Figure 4-30 ActionEvent Portlet in View mode after action selected

6. Edit the user profile again and try **Spanish** as the new locale.

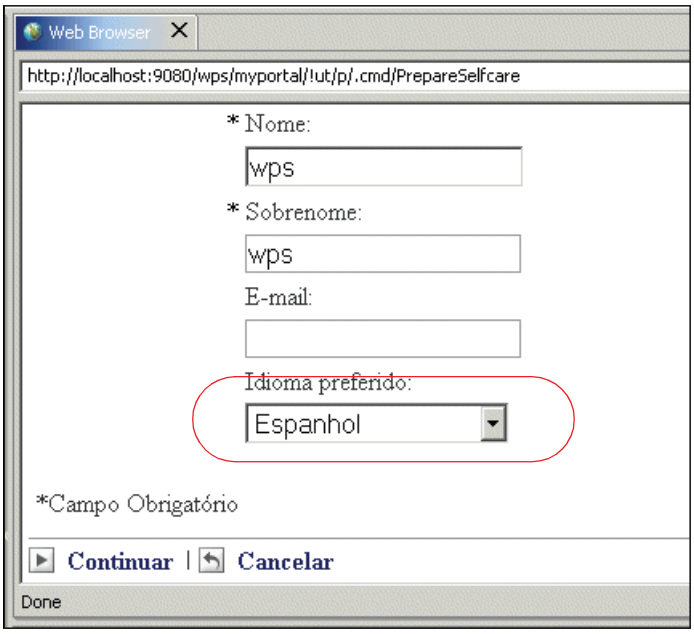


Figure 4-31 Selecting Spanish

The content you specified will display in Spanish.

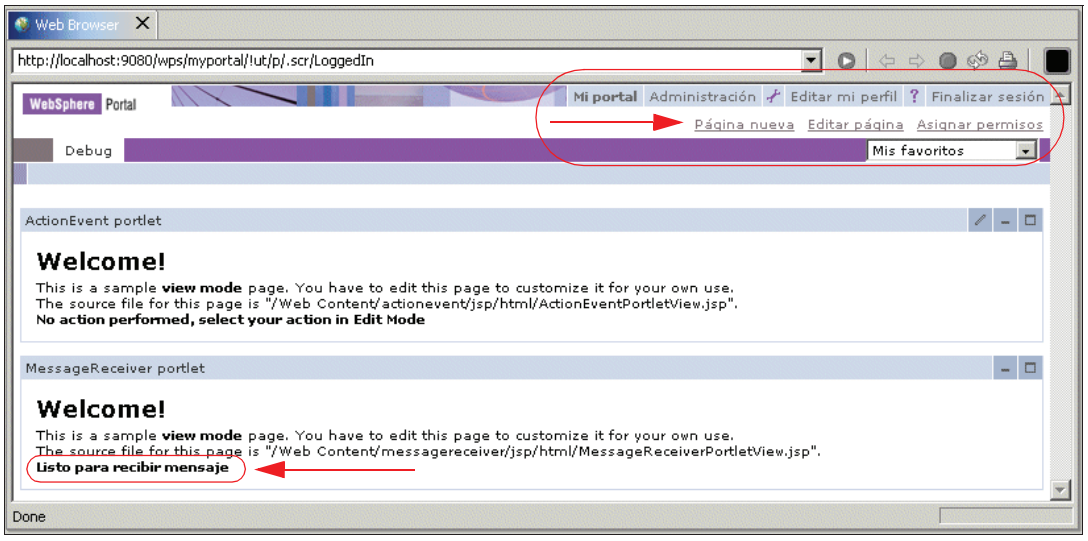


Figure 4-32 Content in Spanish

7. Edit the user profile again and try **French** (**Frances** in Spanish) as the new locale.

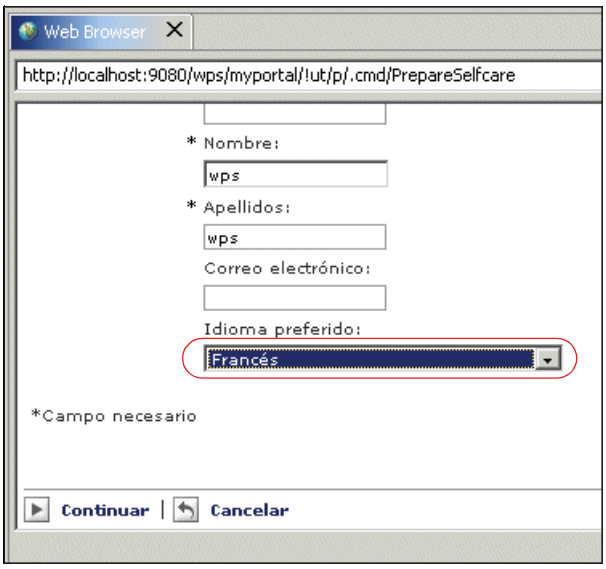


Figure 4-33 Selecting French as the language

8. Since French has not been enabled, Portal will use the default bundle. Your message will display in English (as is specified in the default bundle), but the WebSphere Portal menu at the top will display in French.

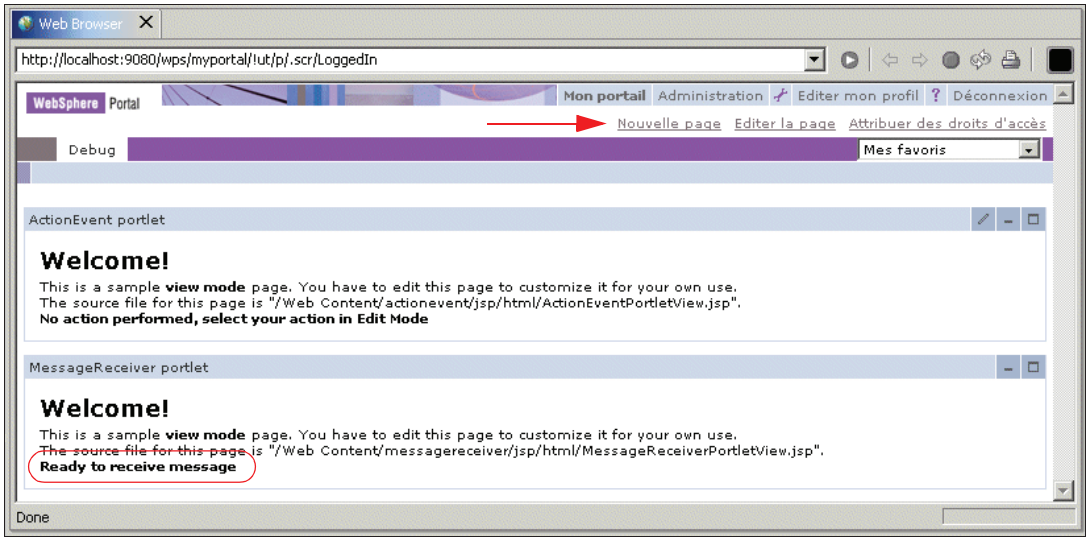


Figure 4-34 In French

9. To change your language back to English before you exit, click **Edit my profile** and select **English** (**Anglais** in French) as your language.

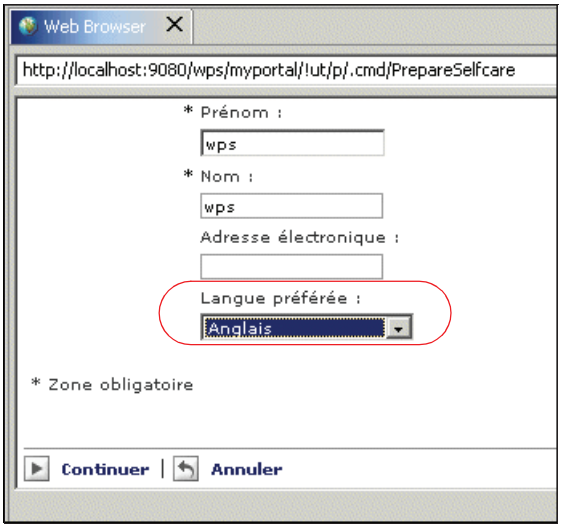


Figure 4-35 Back to English

4.6.4 Accessing NLS bundles in Java portlets

In this section, you will update the ActionEventPortlet.java file to display NLS content based on the locale value (English, Spanish or Brazilian Portuguese). You also need to add the new key-value pairs in the associated property file.

1. Open the file ActionEventPortlet.java for editing by double-clicking it. Next, you will update the code to display the color in the preferred language selected by the user. The resource bundle is accessed via the PortletContext object's getText() method. This method receives three parameters:
   a. Base name of the resource bundle, including the path relative to the classes directory and without the locale suffix or the properties file type.
   b. Key specified in the properties file.
   c. Locale.
2. Make the following highlighted updates in the actionPerformed() method.

Example 4-14 ActionEventPortlet.java

```
......
import java.io.IOException;
import java.util.Locale;

......

   public void actionPerformed(ActionEvent event) throws PortletException {
      if( getPortletLog().isDebugEnabled() )
         getPortletLog().debug("ActionListener - actionPerformed called");
      // ActionEvent handler
      String actionString = event.getActionString();

      //get the preferred locale for the user
      Locale loc = event.getRequest().getLocale();

      // Add action string handler here
      if (actionString.equalsIgnoreCase(ACTION_RED)) {
         //access the resource bundle via the PortletContext object's getText
method
         String red = getPortletConfig().getContext().getText("nls.NLSLab",
"redColor", loc);

         //create the string of HTML to be rendered
         String value = "Action <FONT color=\"#ff0000\">" + red + "</FONT>";

         //create a portlet request
         PortletRequest request = event.getRequest();
         ......................
```

```
      if (actionString.equalsIgnoreCase(ACTION_BLUE)) {
         //access the resource bundle via the PortletContext object's getText
method
         String blue = getPortletConfig().getContext().getText("nls.NLSLab",
"blueColor", loc);

         //create the string of HTML to be rendered
         String value = "Action <FONT color=\"#0000ff\">" + blue + "</FONT>";

         //create a portlet request
         PortletRequest request = event.getRequest();
.......
```

3. Open the resource bundles located in the Java Source/nls folder and make sure the following key-value pairs required for this scenario have been included in the properties files.

Example 4-15 NLSLab.properties (default)

```
redColor=RED
blueColor=BLUE
```

Example 4-16 NLSLab_en.properties (English)

```
redColor=RED
blueColor=BLUE
```

Example 4-17 NLSLab_es.properties (Spanish)

```
redColor=ROJO
blueColor=AZUL
```

Example 4-18 NLSLab_pt_BR.properties (Brazilian Portuguese)

```
redColor=VERMELHA
blueColor=AZUL
```

4. Select **File -> Save All** to save all your changes to the project.
5. Close the browser.
6. Click **Run on Server** to test your changes.
7. Click **Edit my profile** to change the preferred languages and execute the application again to check that portlets display the word `BLUE` or `RED` in the language you have selected.

   Note: For simplicity, not all text in this sample scenario has been enabled for NLS.

4.6.5 Adding PDA markup support

In this section, you will update the sample scenario to add PDA markup support. Follow these steps:

1. Create new folders with the name pda to store the JSP files for PDA browsers.

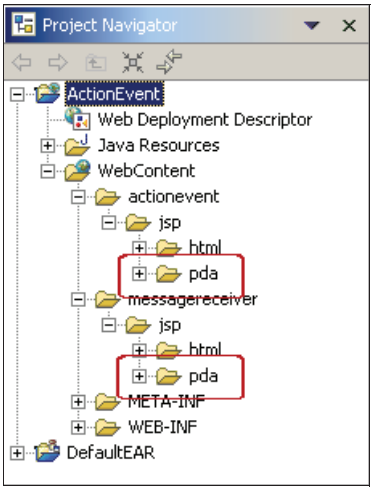


Figure 4-36 New pda folders

2. In the PDA directories, include the corresponding JSP files. You can create new files or copy from the html directories. Don't forget to include access to NLS bundles as shown in 4.6.2, "Accessing NLS bundles from JSPs" on page 110.

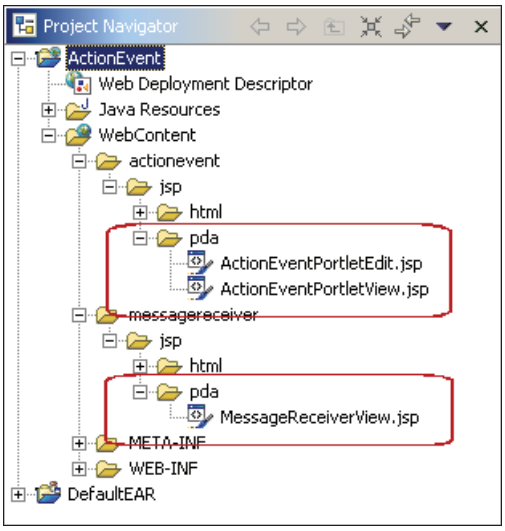


Figure 4-37 New jsp files

3. Update the portlet.xml file to indicate it supports PDA markup. For each portlet, add a markup name of pda to the support tags in the portlet.xml file.

Example 4-19 portlet.xml file

```
......
      <portlet-app-name>ActionEvent application</portlet-app-name>
      <portlet id="actionevent.ActionEventPortlet"
href="WEB-INF/web.xml#actionevent.ActionEventPortlet" major-version="1"
minor-version="0">
         <portlet-name>ActionEvent portlet</portlet-name>
         <cache>
            <expires>0</expires>
            <shared>no</shared>
         </cache>
         <allows>
            <maximized />
            <minimized />
         </allows>
         <supports>
            <markup name="pda">
               <view />
               <edit />
            </markup>
            <markup name="html">
               <view />
               <edit />
            </markup>
```

```
            </supports>
         </portlet>
         <portlet id="messagereceiver.MessageReceiver"
href="WEB-INF/web.xml#messagereceiver.MessageReceiver" major-version="1"
minor-version="0">
            <portlet-name>MessageReceiver portlet</portlet-name>
            <cache>
               <expires>0</expires>
               <shared>no</shared>
            </cache>
            <allows>
               <maximized />
               <minimized />
            </allows>
            <supports>
               <markup name="pda">
                  <view />
               </markup>
               <markup name="html">
                  <view />
               </markup>
            </supports>
         </portlet>
      </portlet-app>
......
```

4. Select **File -> Save All** to save all your changes to the project.
5. Run the NLS scenario.

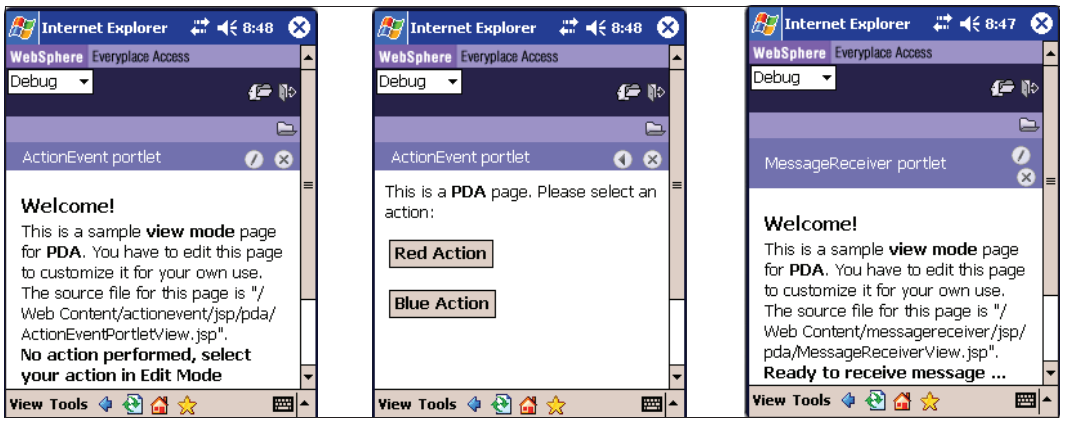


Figure 4-38 Running sample scenario in PDA

Note: For more information about adding PDA markup support, refer to the WebSphere Everyplace Access InfoCenter documentation.

4.7 Sample scenario: translating whole resources

Another way to accomplish internationalization is by translating and maintaining separate JSPs within a predictable directory structure. The Portal will take responsibility for locating the correct file at runtime, depending on the preferred language selected by the user.

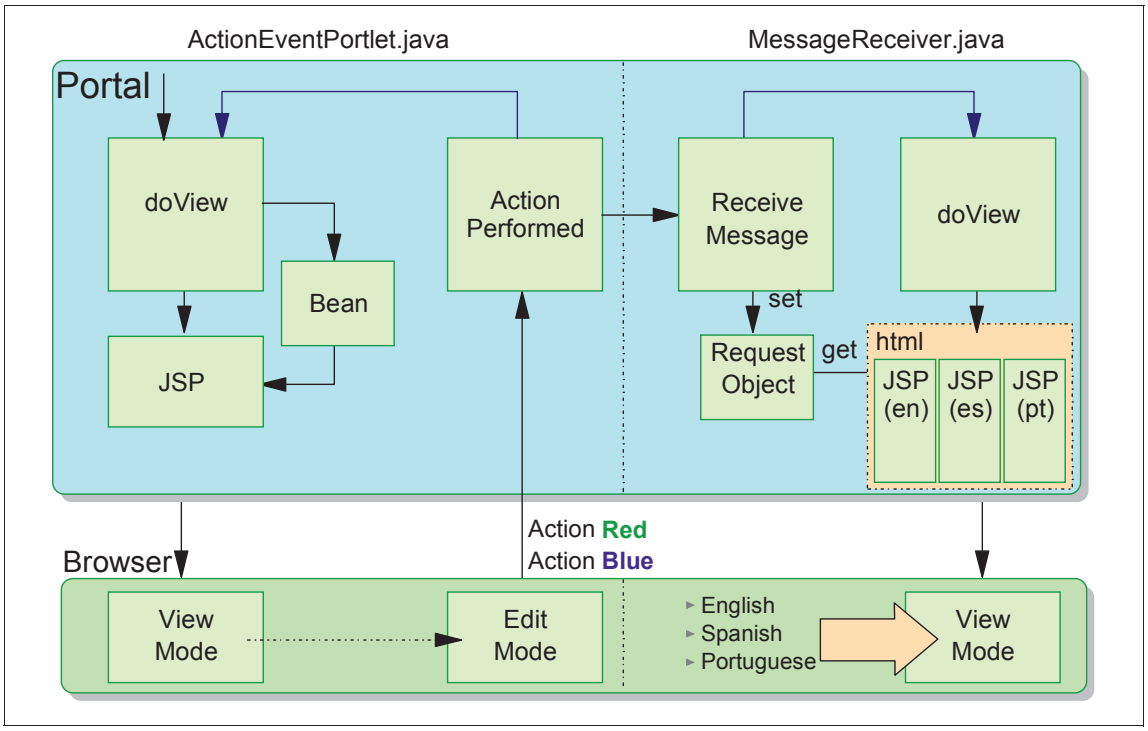


Figure 4-39 Sample scenario

1. Open the Web Content/messagereceiver/jsp/html/MessageReceiverView.jsp page to delete fmt tags and return to static information. Your code should look as shown in Example 4-20.

Example 4-20 MessageReceiverView.jsp

```
<%@ page contentType="text/html" import="java.util.*, messagereceiver.*"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>

<DIV style="margin: 6px">

<H3 style="margin-bottom: 3px">Welcome!</H3>
```

```
This is a sample <B>view mode</B> page. You have to edit this page to customize
it for your own use.<BR>
The source file for this page is "/Web
Content/messagereceiver/jsp/html/MessageReceiverView.jsp".

<br>
<% if (request.getAttribute("MyMessage") == null) { %>
   <B>Ready to receive message ... </B>
<% } else { %>
   <B>Received a message:</B>
   <B><%= request.getAttribute("MyMessage") %></B>
<% } %>

</DIV>
```

2. Select your **Web Content/messagereceiver/jsp/html** folder, right-click it and select **New -> Folder**. Type en for the Folder name field and click **Finish**.
3. Right-click **Web Content/messagereceiver/jsp/html/MessageReceiverView.jsp** and select **Copy**.

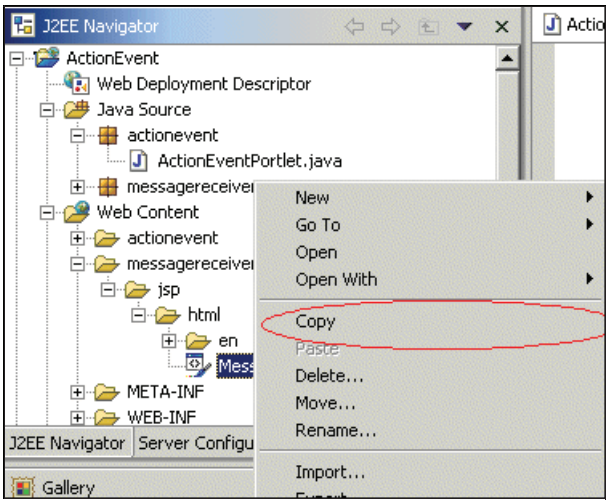


Figure 4-40 Copy a JSP page

4. Now select the **Web Content/messagereceiver/jsp/html/en** folder, right-click it and select **Paste**.

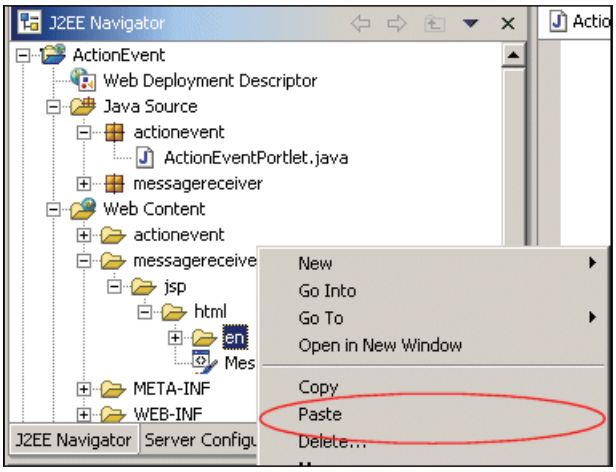


Figure 4-41 Paste a JSP page

5. Open the page Web Content/messagereceiver/jsp/html/en/MessageReceiverView.jsp to update the text indicating the location of the source file page. This is not required but it is recommended for clarity.

Example 4-21 MessageReceiverView.jsp (English)

```
<%@ page contentType="text/html" import="java.util.*, messagereceiver.*"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>

<DIV style="margin: 6px">

<H3 style="margin-bottom: 3px">Welcome!</H3>
This is a sample <B>view mode</B> page. You have to edit this page to customize
it for your own use.<BR>
The source file for this page is "/Web
Content/messagereceiver/jsp/html/en/MessageReceiverView.jsp".


<br>
<% if (request.getAttribute("MyMessage") == null) { %>
   <B>Ready to receive message ...</B>
<% } else { %>
   <B>Received a message:</B>
   <B><%= request.getAttribute("MyMessage") %></B>
<% } %>

</DIV>
```

6. Optionally, repeat the steps to create the folders and JSPs for other languages such as Spanish (es) and Brazilian Portuguese (pt_BR). Your directory structure should be as illustrated in Figure 4-42 on page 129.

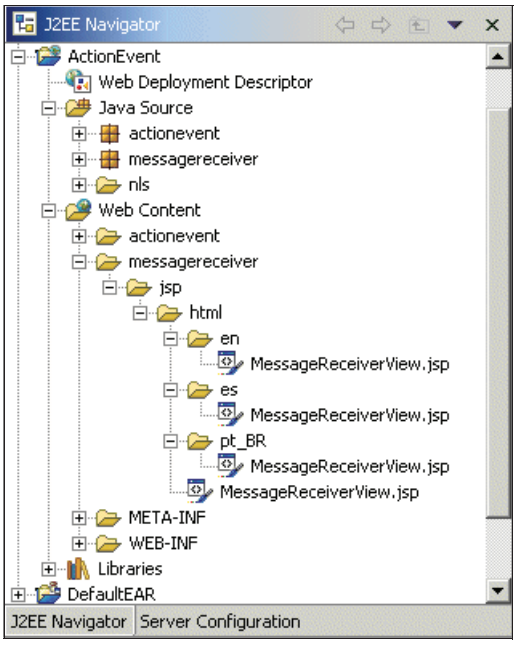


Figure 4-42 Directory structure

7. Modify the JSP pages to display a message in the proper language. Also change the directory of the source file pages. For example, create a folder with a JSP for Spanish (es).

Example 4-22 MessageReceiverView.jsp (Spanish)

```
<%@ page contentType="text/html" import="java.util.*, messagereceiver.*"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>

<DIV style="margin: 6px">

<H3 style="margin-bottom: 3px">Welcome!</H3>
This is a sample <B>view mode</B> page. You have to edit this page to customize
it for your own use.<BR>
The source file for this page is "/Web
Content/messagereceiver/jsp/html/es/MessageReceiverView.jsp".


<br>
<% if (request.getAttribute("MyMessage") == null) { %>
   <B>Listo para recibir mensajes ...</B>
<% } else { %>
   <B>Mensaje recibido:</B>
```

```
   <B><%= request.getAttribute("MyMessage") %></B>
<% } %>

</DIV>
```

Example 4-23 MessageReceiverView.jsp (Br Portuguese)

```
<%@ page contentType="text/html" import="java.util.*, messagereceiver.*"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>

<DIV style="margin: 6px">

<H3 style="margin-bottom: 3px">Welcome!</H3>
This is a sample <B>view mode</B> page. You have to edit this page to customize
it for your own use.<BR>
The source file for this page is "/Web
Content/messagereceiver/jsp/html/pt_BR/MessageReceiverView.jsp".


<br>
<% if (request.getAttribute("MyMessage") == null) { %>
   <B>Pronto para receber mensagem ...</B>
<% } else { %>
   <B>Mensagem recebida:</B>
   <B><%= request.getAttribute("MyMessage") %></B>
<% } %>

</DIV>
```

8. Run the scenario (click **Run on Server**) and verify your results in multiple languages.
9. Change the locale in the user profile as before and try other supported languages.

4.7.1 Adding PDA markup support

Follow these step to add PDA markup support by translating and maintaining separate JSPs:

1. Create a new folder with the name pda. Create folders into pda folder for each languages you want to support such as English (en), Spanish (es) and Brazilian Portuguese (pt_BR).
2. In the PDA directories, include the corresponding JSP files for each language folder. You can create new files or copy from the html directories. Don't forget to display a message in the proper language for each JSP file. See 4.7,

"Sample scenario: translating whole resources" on page 125 for more details. Your directory structure should be as illustrated in Figure 4-43.

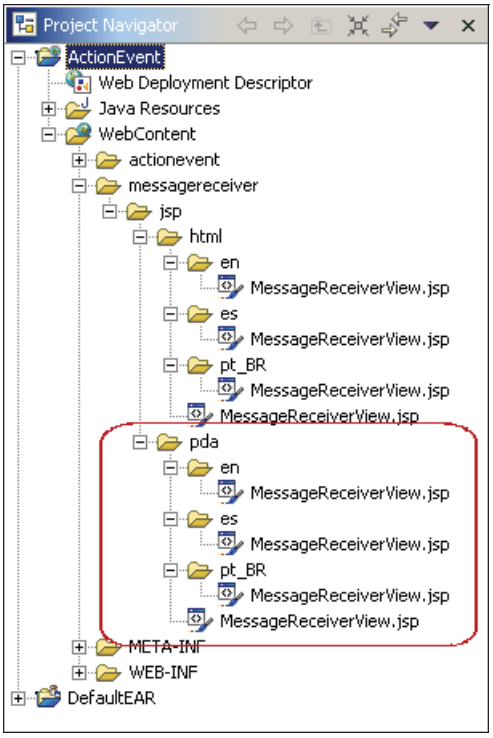


Figure 4-43 Directory structure

3. Update the portlet.xml file to indicate it supports PDA markup. For each portlet, add a markup name of pda to the support tags in the portlet.xml file.

Example 4-24 portlet.xml file

```
......
      <portlet-app-name>ActionEvent application</portlet-app-name>
      <portlet id="actionevent.ActionEventPortlet"
href="WEB-INF/web.xml#actionevent.ActionEventPortlet" major-version="1"
minor-version="0">
         <portlet-name>ActionEvent portlet</portlet-name>
         <cache>
            <expires>0</expires>
            <shared>no</shared>
         </cache>
         <allows>
            <maximized />
            <minimized />
```

```
         </allows>
         <supports>
            <markup name="pda">
               <view />
               <edit />
            </markup>
            <markup name="html">
               <view />
               <edit />
            </markup>
         </supports>
      </portlet>
      <portlet id="messagereceiver.MessageReceiver"
href="WEB-INF/web.xml#messagereceiver.MessageReceiver" major-version="1"
minor-version="0">
         <portlet-name>MessageReceiver portlet</portlet-name>
         <cache>
            <expires>0</expires>
            <shared>no</shared>
         </cache>
         <allows>
            <maximized />
            <minimized />
         </allows>
         <supports>
            <markup name="pda">
               <view />
            </markup>
            <markup name="html">
               <view />
            </markup>
         </supports>
      </portlet>
   </portlet-app>
......
```

4. Select **File -> Save All** to save all your changes to the project.
5. Run the NLS scenario.



Figure 4-44 Running sample scenario in PDA

Note: For more information about adding PDA markup support, refer to the WebSphere Everyplace Access InfoCenter documentation.

5

Developing a database access portlet

This chapter describes a sample procedure to create, test and deploy a database access portlet using the WebSphere Everyplace Toolkit. Application developers can follow these high-level steps to write and test a Database Access Portlet. In this chapter you will find a suggested procedure to build a database portlet, test it in the embedded portal test environment and finally, deploy to a portal server.

The following tasks are described in this chapter:

1. Create a test database and populate the table
2. Use the wizard to generate the portlet
3. Use the Portal test environment to test the portlet
4. Export the portlet application
5. Deploy to your WebSphere Everyplace Access server
6. Execute the portlet from a Pocket PC
7. Use proper driver if database is not DB2®

© Copyright IBM Corp. 2005. All rights reserved.

5.1 Create a test database and populate the table

The following software is required to successfully run the sample scenario described in this chapter:

- WebSphere Studio Site Developer or WebSphere Studio Application Developer Version 5.1.1
- WebSphere Everyplace Toolkit Version 5.0.1
- DB2 UDB Version 8.16
- MySQL Server Version 4.1.1

In this section you will go through a few steps to create and populate a database. For example execute the following tasks:

1. Open a DB2 command window. **Start** -> **Programs** -> **IBM DB2** -> **Command Line Tools** -> **Command Window**.
2. Execute these DB2 commands in the DB2 command window in turn
3. Create the database. This step may take a few minutes.

   ```
   db2 create db testdb
   ```

4. Now that it has been created, connect to it.

   ```
   db2 connect to testdb
   ```

5. Create the user table.

   ```
   db2 create table user(id varchar(5) not null, firstname varchar(10),
   lastname varchar(10), username varchar(20), email varchar(20), primary
   key(id))
   ```

6. Populate the user table. For example insert the following values:

   ```
   db2 insert into user values('001', 'admin', 'admin', 'admin',
   'admin@123.com')
   db2 insert into user values('002', 'company', 'company', 'company',
   'company@123')
   db2 insert into user values('003', 'anonymous', 'anonymous', 'anonymous',
   'anonymous@123.com')
   db2 insert into user values('004', 'operator', 'operator', 'operator',
   'operator'@123.com')
   ```

7. Optionally, use the Control Center (**Start** -> **Programs** -> **IBM DB2** -> **General Administration Tools** -> **Control Center**) to check if the database has been successfully created

5.2 Use the wizard to generate the portlet

In this part, you will use the Database Access Portlet wizard to generate the actual DB2 database access portlet. In this section you are shown how to use the wizard to generate a portlet development project and add a db access portlet to the project. Execute the following steps:

1. Start WebSphere Studio Site Developer.
2. Create a portlet project first. From the studio menu, choose **File** -> **New** -> **Project**

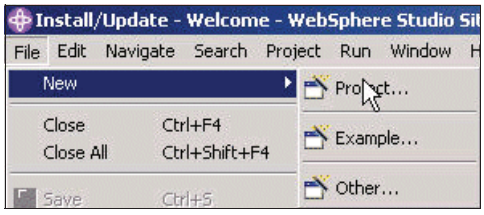


Figure 5-1 New project

3. This opens the **New Project** Wizard. The first window is split into two panes, the left pane shows families of projects that the wizard is able to generate. Select **Portlet Development** in the left pane. The right pane lists the projects that are available in the selected family. Select **Portlet Application Project** in the right pane. This will open a dialog asking for project specifications: Click **Next**,

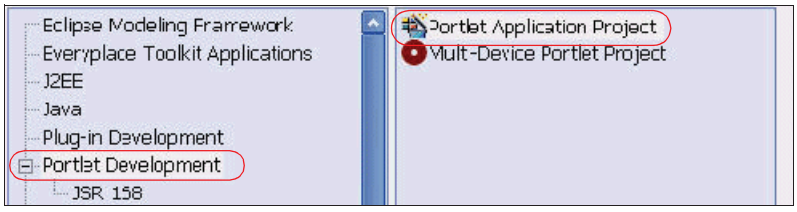


Figure 5-2 Create a new portlet application project

4. This will bring up the a new window.

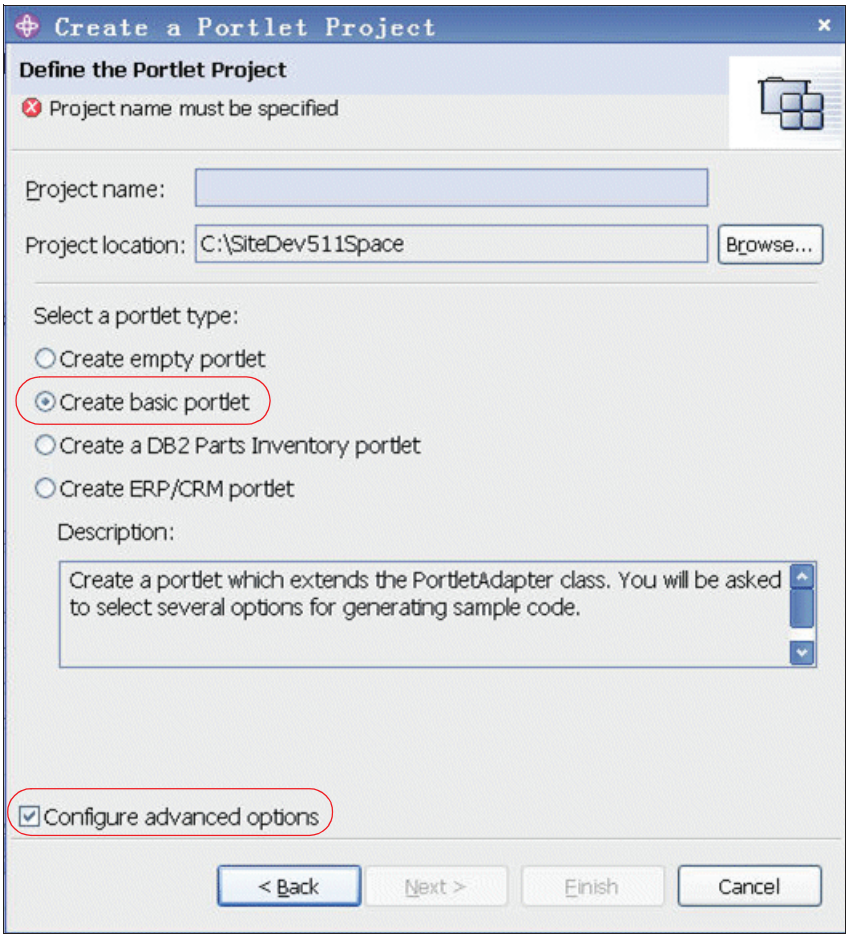


Figure 5-3 Define the portlet project

5. Specify a name for the Project name value.

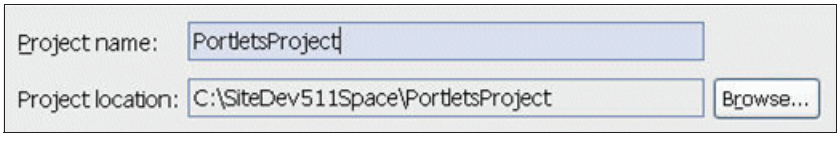


Figure 5-4 Project name and location

6. Select the option Create a basic portlet.
7. Select Configure advanced options.
8. Click Next

9. Change the EAR Project name to PortletsProjectEAR. Choose the default option J2EE Level 1.3/WebSphere Portal 5.0. Click **Next**.

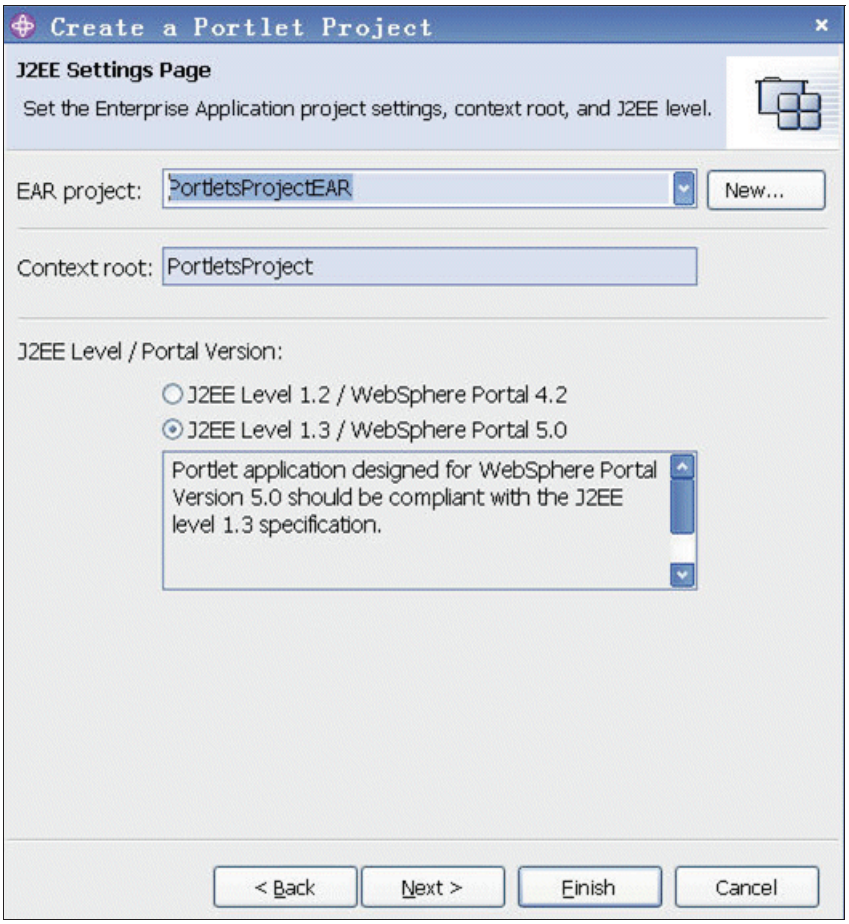


Figure 5-5 J2EE settings page

10. You will see the default settings shown in Figure 5-6. Take the default values and click **Next**.

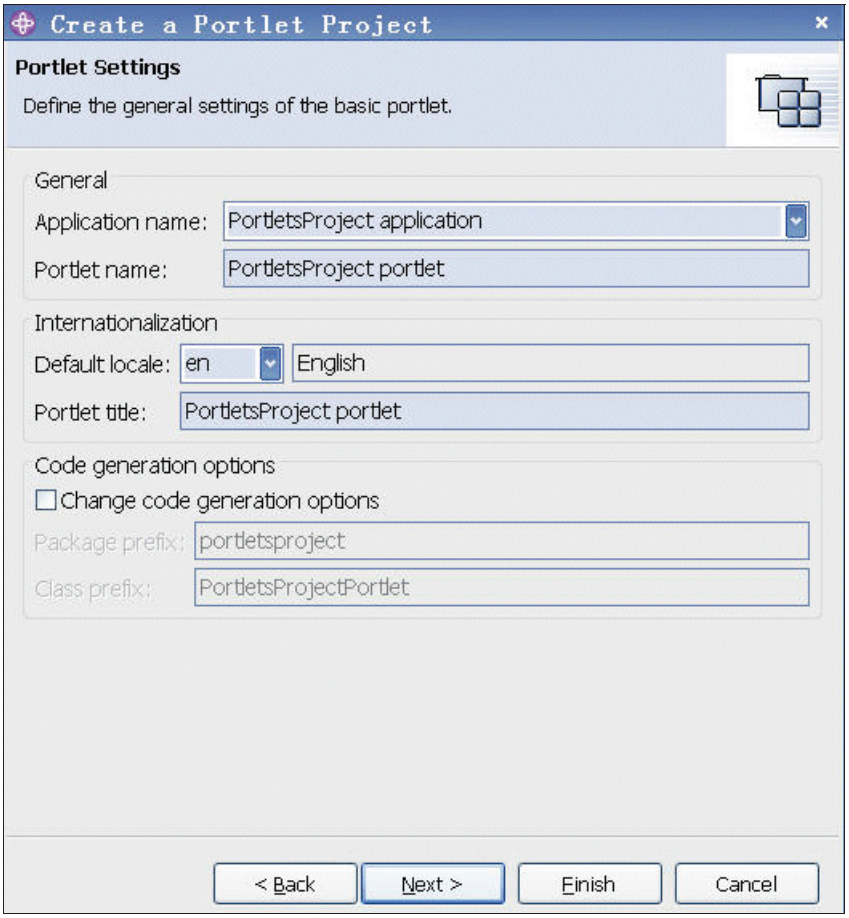


Figure 5-6 Portlet settings

11. The next window allows you to add Event handling to the generated portlet, just accept the default value and click **Next**.

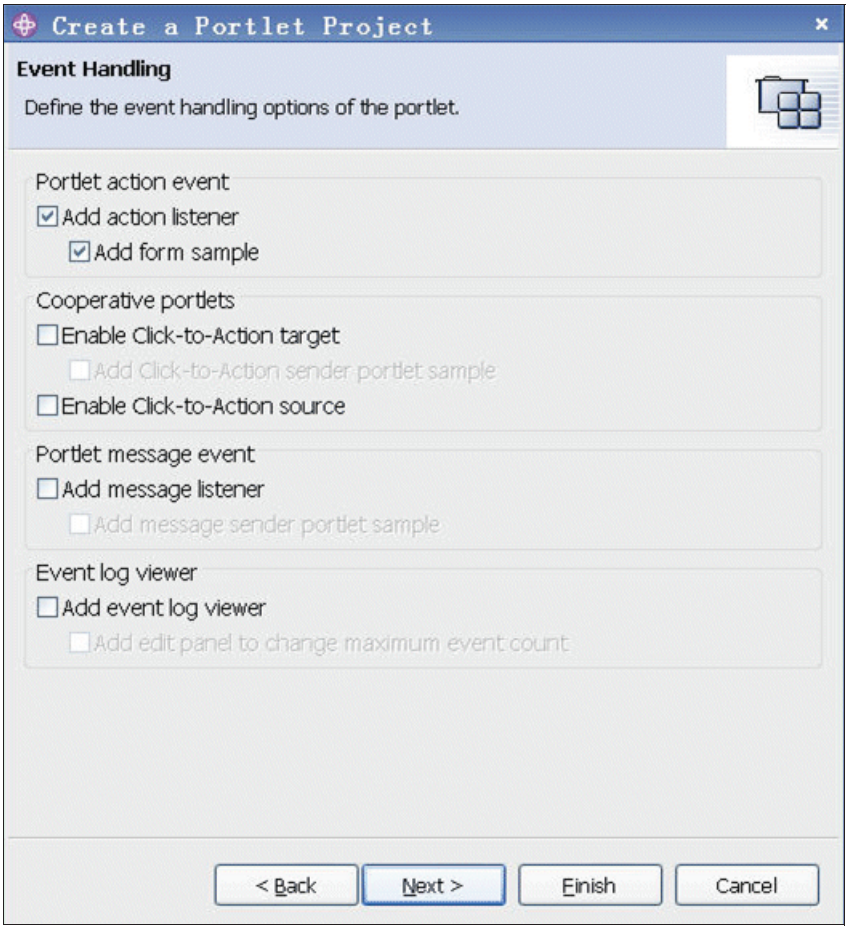


Figure 5-7 Event handling options

12. The next window allows you to add credential vault handling. The credential vault will not be used in the sample scenario, just accept the default value and click **Next**.
13. The next window will let you add some other mark-up languages for specific device such as PDA etc. We are going to test on PDA devices, so check **Add pda markup support** and click **Finish**.
14. After the code generation completes you may be prompted to switch to the portlet development perspective. If so, click **yes** to switch perspectives.
15. The project should now be ready. The next steps to generate a Database Access Portlet by using the db wizard.
16. Select your portlet project from the Project Navigator inside Studio.

17. Right-click over the project choose the context menu option **New** -> **Other,** then choose **Portlet Development** from left pane and **Database Access Portlet** on the right pane:

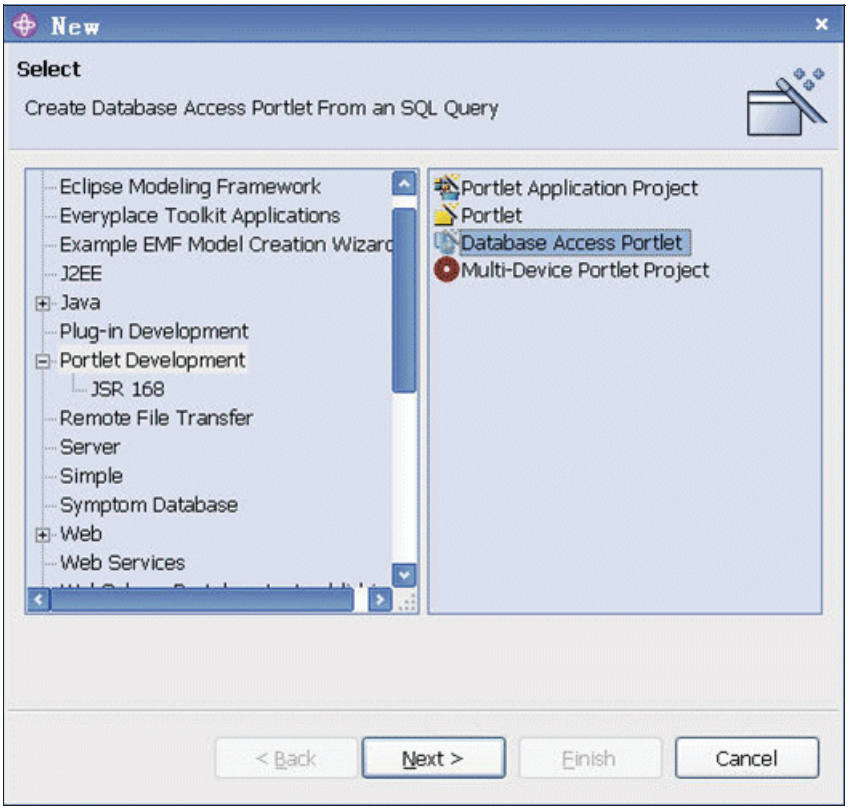


Figure 5-8 Create a database access portlet

18. The next window is complex and allows you to specify many parameters. The Portlet project and Java package fields should already be filled in.

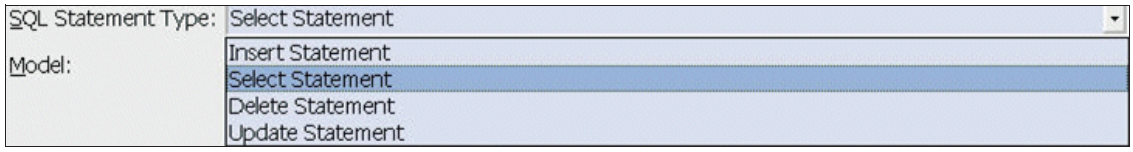


Figure 5-9 SQL statements

The following SQL statement types are available:

a. **Select Statement:** Use an SQL SELECT statement to generate the query result

b. **Insert Statement:** Use an SQL INSERT statement to insert the records to the table

c. **Update Statement:** Use an SQL UPDATE statement to update the record in the table

d. **Delete Statement:** Use an SQL DELETE statement to delete the record in the table

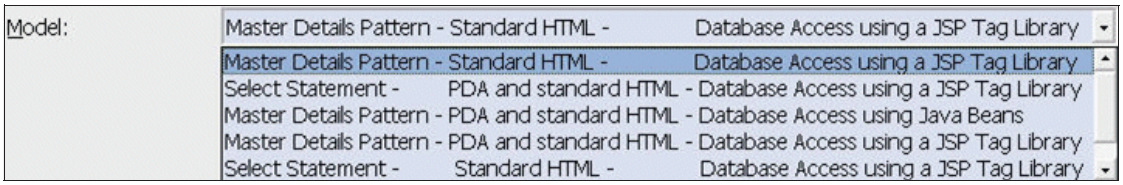


Figure 5-10 Master details pattern

For each model here, you can select to generate HTML only or HTML and PDA markups. Each page generation model generates portlet pages that access database information using a JSP Tag Library to access the data, with the exception of ***Master Details Pattern – Database Access using Java Beans***. This pattern generates portlet pages that access database information using IBM Data Access Beans.

19. For the SQL Statement choose **Select Statement**
20. For the Model choose the default **Master Details Pattern …**
21. The bottom part of the window shows which files will be generated.
22. Click Next.
23. The **Portlet Settings** page lets you specify some basic portlet settings. Insert any updates and click **Next**.



Figure 5-11 Portlet settings

24. The **Choose SQL Method** page to let you choose how to create a SQL statement. If you have already created SQL statements you can elect to use the existing statements.
25. If you need to create a new statement, check the option “Be guided through creating an SQL statement.
26. Leave the option to “Connect to a database and import a new database model” selected.
27. Click **Next**.

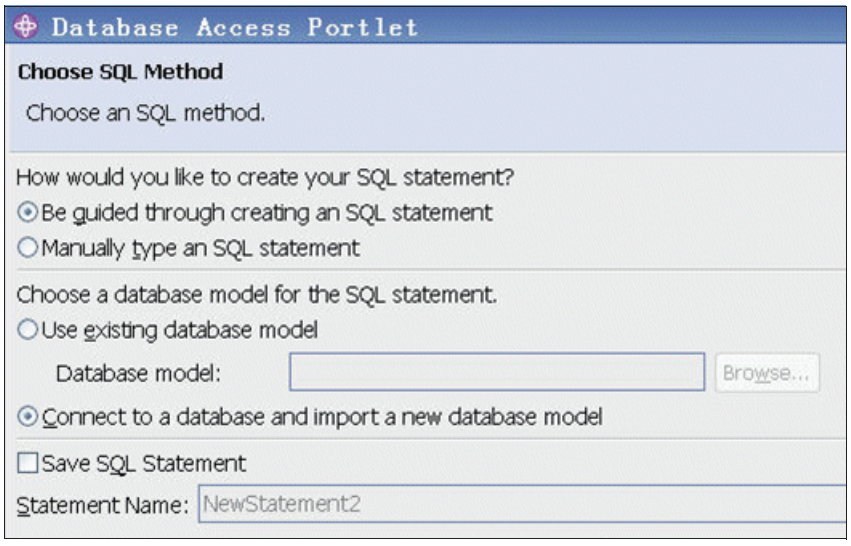


Figure 5-12 Choose SQL method

28. The **Connection Page** will show up. You must specify all the information here to let the wizard connect to the DB2 database.
29. Give the Connection a meaningful name.
30. Set the database to the test database you previously created.
31. Enter a user ID / password that has access to the database.
32. Select the database vendor type to “DB2 Universal Database™ V8.1”
33. Click **Connect to Database** to confirm the connection is good. If it is then you will see a window allowing you to construct the SQL Statement.

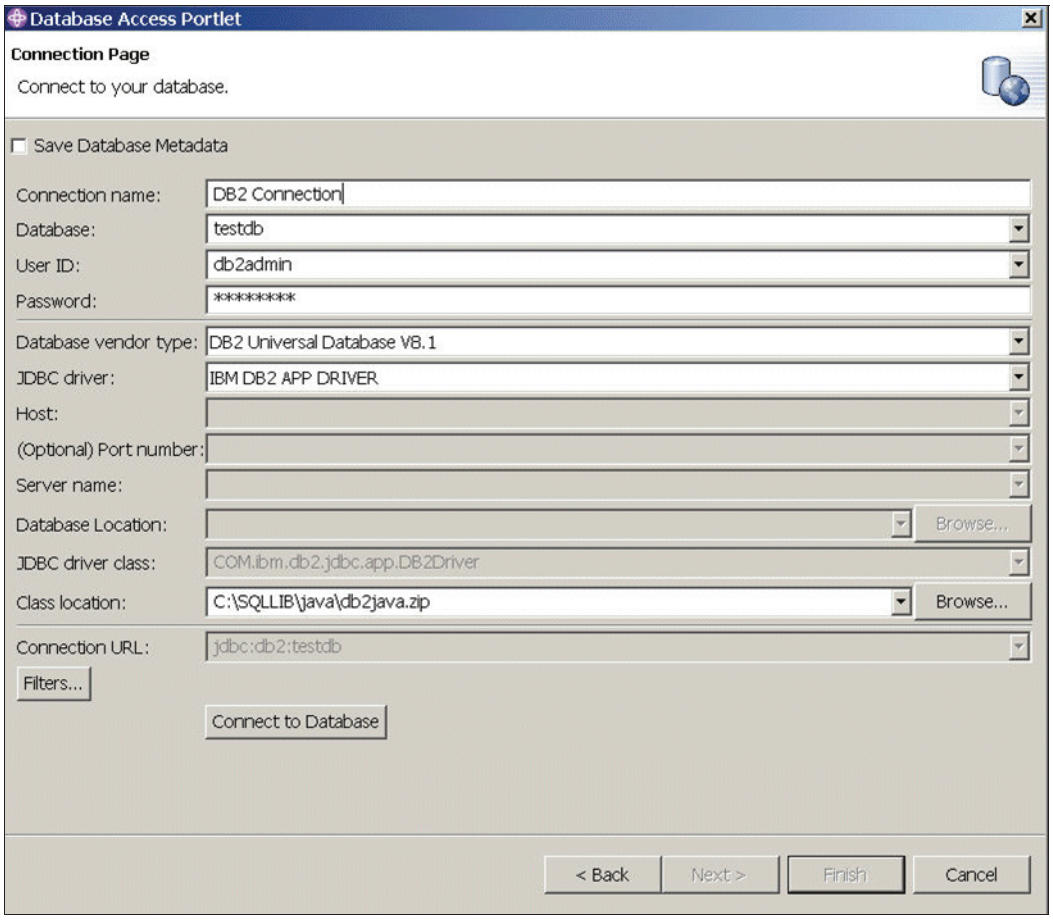


Figure 5-13 Connection page

34. The SQL wizard will show up to help you go through all the steps to complete a SQL statement. You can add **Tables**, **Columns**, specify **Joins** and **Conditions**, and make up the **Groups** or **Order**.
35. On the Tables tab, you will find your user table under the Administrator ID.
36. Select the table (Administrator. user or db2admin.user).
37. Click the “>” button to add the table to the selected list.

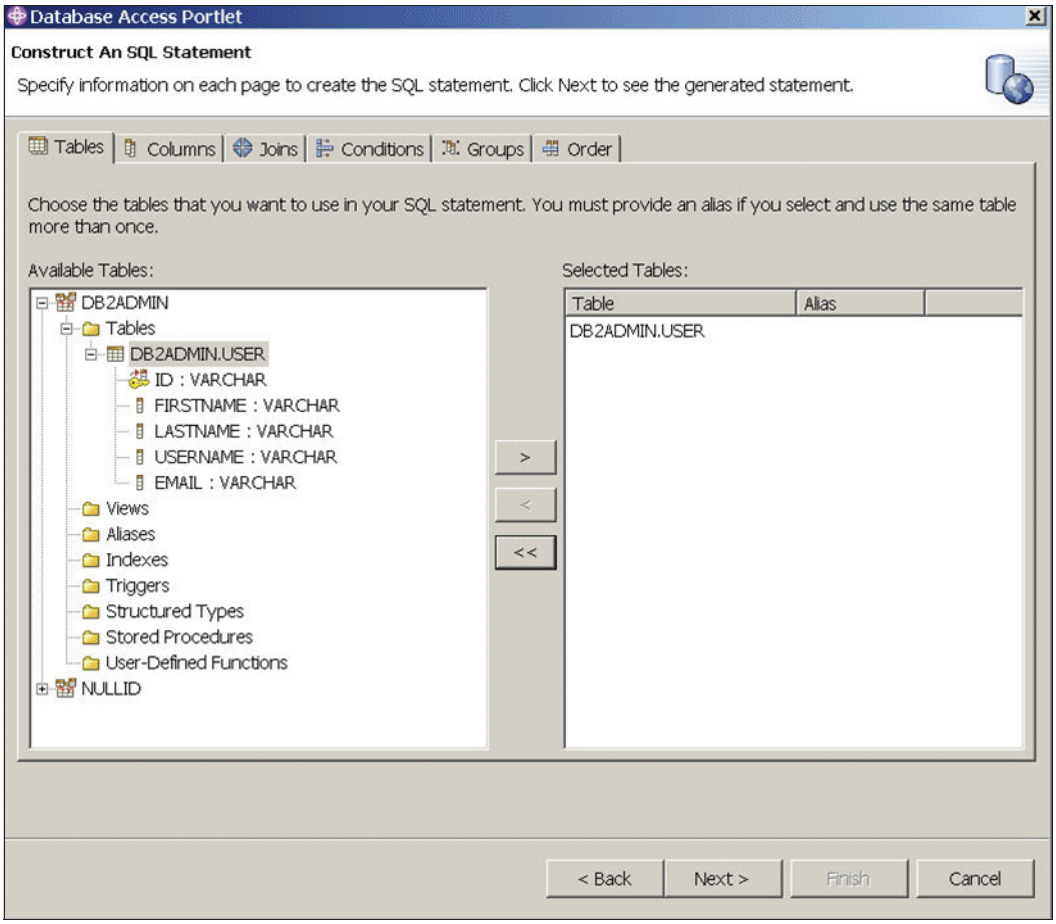


Figure 5-14 Construct the SQL statement

38. If you leave the SQL statement as it is a Select will be generated. Click the Columns tab.
39. Select the columns you want to add, and click the “>” button to add them to the selected columns list.

40. This should be all you need. Click **Next**
41. You will then be shown the SQL that will be generated. From this window you are able to make modifications to the SQL. If you make a change then you must Parse the SQL. You can also Execute the SQL. From this window to check the results.
42. Again you now have enough details to generate the portlet, Do not click finish, but if you did, the wizard will take the default options through the next set of windows. If you need to modify the jdbc driver to another database, you will need to make modifications to the Database Access Portlet. Click **Next**.
43. The **Runtime Connection Page** will let you specify whether to use the data source or driver manager to get the database connection. Click **Next**.

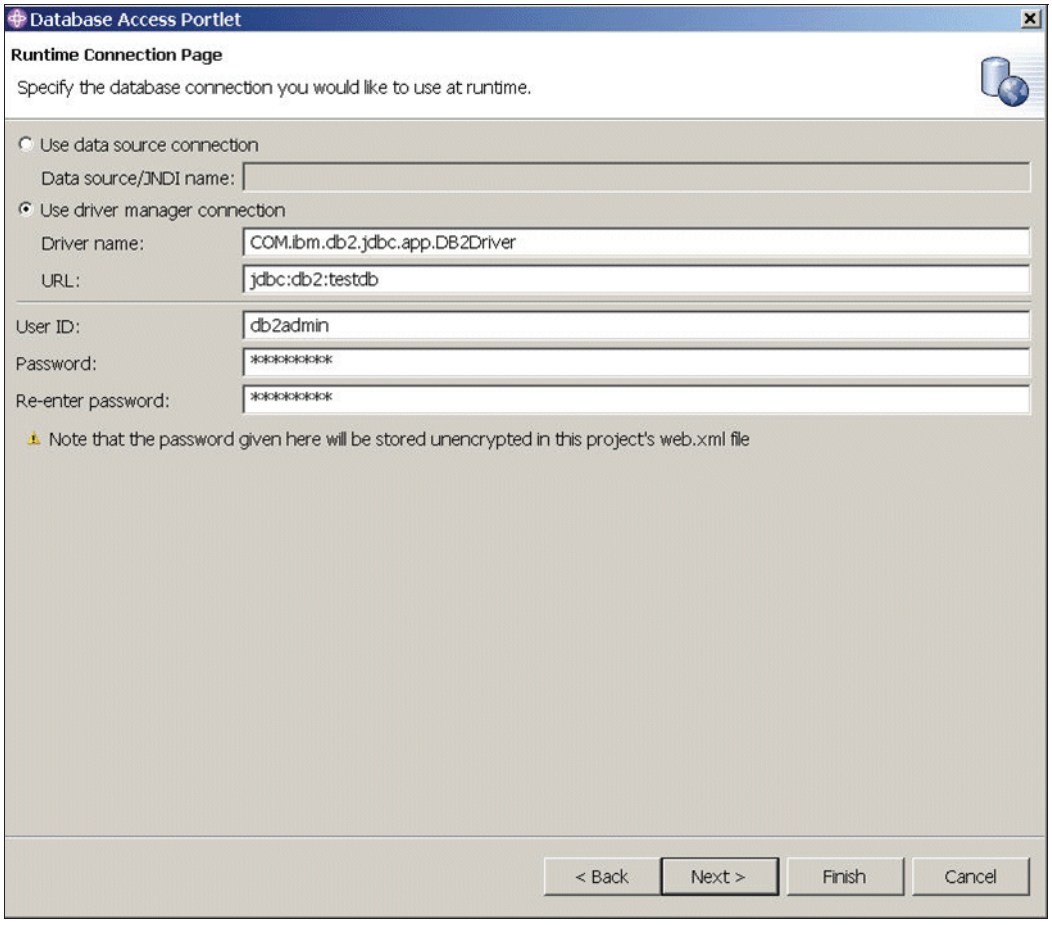


Figure 5-15 Runtime connection page

44. The **Result Page** will let you specify whether to store the query results in a request or in the session. Leave the setting as “Request” and click **Next**.

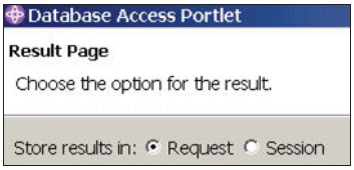


Figure 5-16 Result page

45. You will now go through several pages to design the actual view of how to display the data in the Web page. The first of these allows you to design the input form. If you did not specify a where clause in your SQL then you will not have any input fields. As a minimum you will be able to modify the page title.
46. To modify the page title select the page property tab on the bottom left panel, and overwrite the value on the page title row.

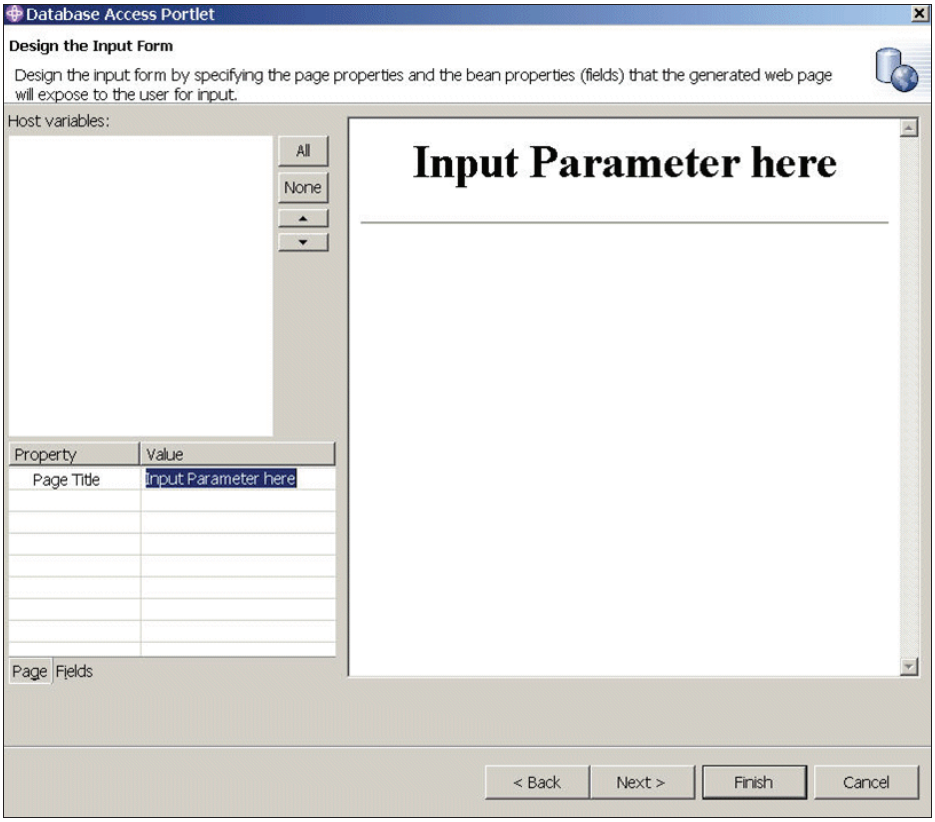


Figure 5-17 Design the input form

47. Click **Next**.
48. On the next window you are able to design the Master View.
49. To modify the page title click the page tab on the bottom left, and modify the page title row.
50. To add or remove columns, select or deselect the columns on the top left panel.
51. To modify column names, select to highlight a column in the top left panel. Click the Fields tab in the bottom left panel, and modify the label line.
52. You can modify the column width by setting either the column width or the percentage properties, in the bottom left panel.

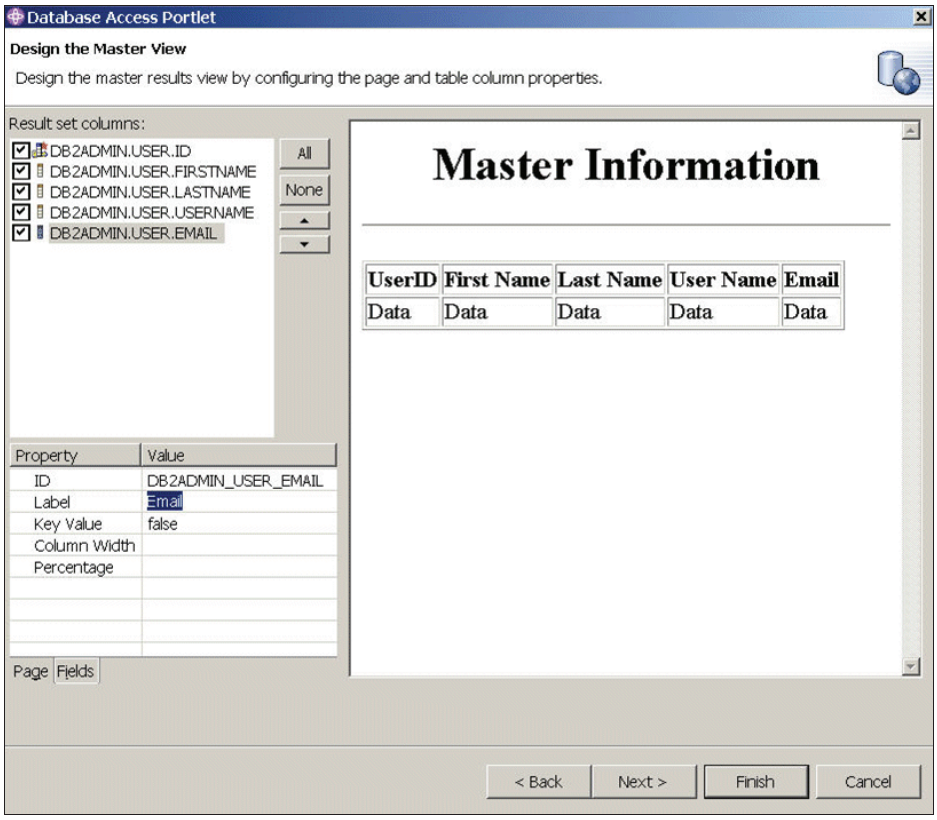


Figure 5-18 Design the master view

53. Click **Next**.
54. On the next window you are able to design the details form. This is the details view, when you select a row from the master view.
55. Use the top left panel to add / remove fields.
56. Use the bottom left panel to add entry fields to the form, allowing updates. You can also set the size of the field and the max length of the field.

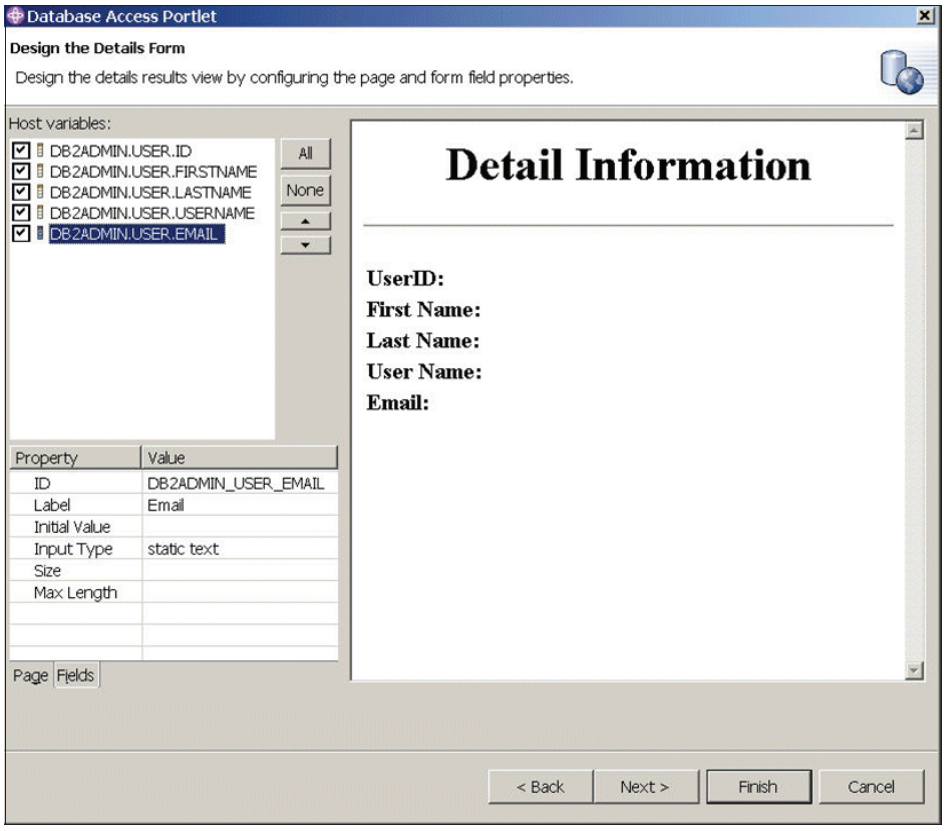


Figure 5-19 Design the details form

57. Click **Next**.
58. The last window in this series allows you to specify the prefix to resolve name clashes with other generated portlets.
59. Click **Finish** to complete the wizard.
60. Specify the prefix to use on the generated files.

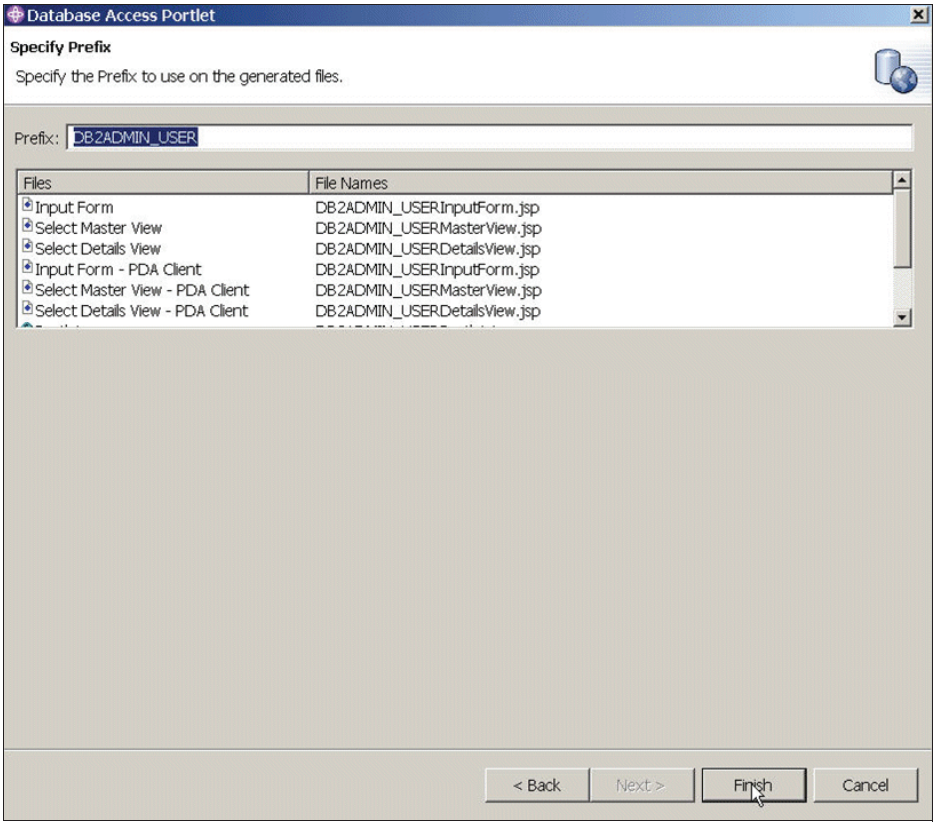


Figure 5-20 Specify the prefix

5.3 Use the Portal Test Environment to test the portlet

In this section you will go through the steps to run the database access portlet in the portal test environment which is already embedded in the WebSphere studio site developer.

Execute the following steps:

1. Switch to the Server Perspective.
2. Define a test portal server. We will name the server as “portal5”.

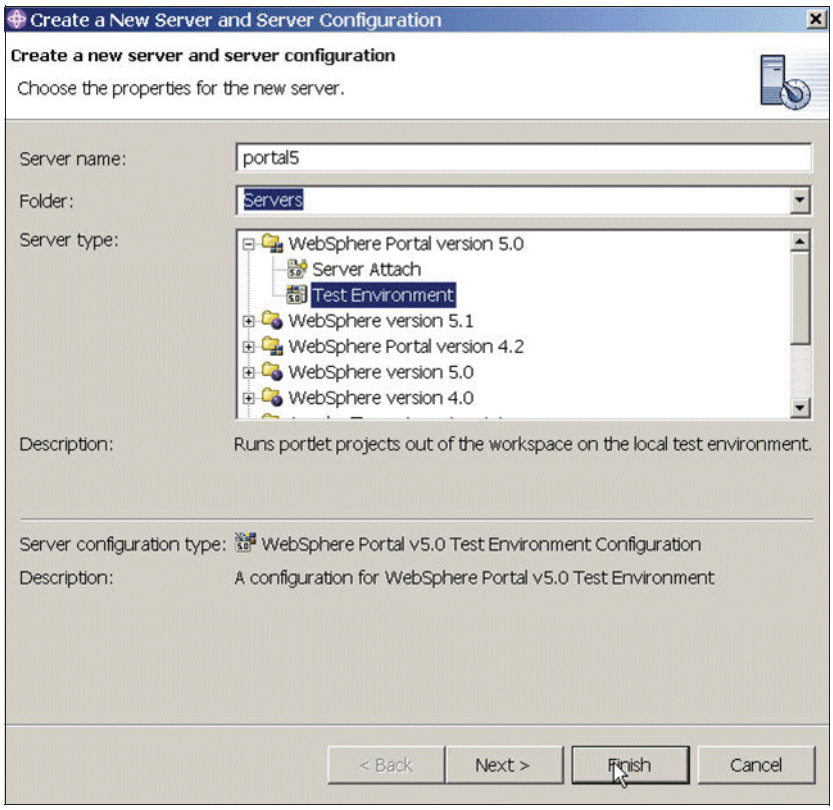


Figure 5-21 Create a new server instance

3. If needed, add your project to the server you just created.

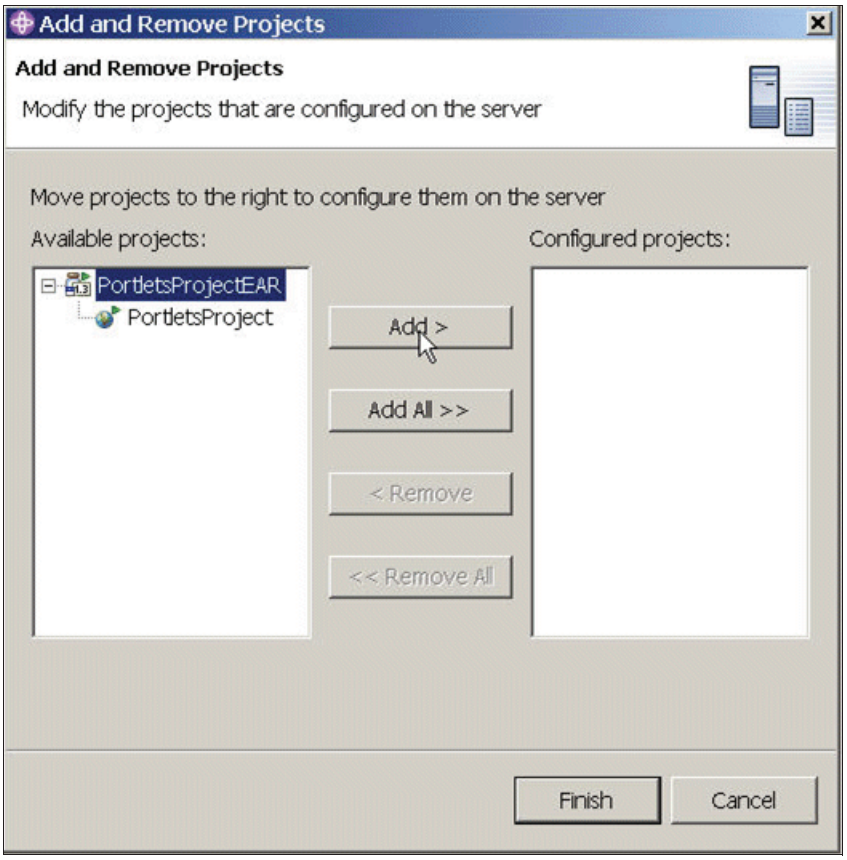


Figure 5-22 Adding the project to the server

4. Start your server and wait a few minutes to see the *WebSphere Portal is open for business* message in the console.

```
[6/3/04 14:18:27:141 EDT] 4e18eaa4 HttpTransport A SRVE0171I: Transport http is listening on port 9,081.
[6/3/04 14:18:29:484 EDT] 4e18eaa4 HttpTransport A SRVE0171I: Transport https is listening on port 9,444.
[6/3/04 14:18:29:516 EDT] 4e18eaa4 HttpTransport A SRVE0171I: Transport http is listening on port 9,091.
[6/3/04 14:18:30:031 EDT] 4e18eaa4 HttpTransport A SRVE0171I: Transport https is listening on port 9,044.
[6/3/04 14:18:30:094 EDT] 4e18eaa4 RMIConnectorC A ADMC0026I: RMI Connector available at port 9810
[6/3/04 14:18:30:156 EDT] 4e18eaa4 WsServer      A WSVR0001I: Server WebSphere Portal open for e-business
```

Figure 5-23 WebSphere Portal open for e-business message

5. Run the portlet. Right-click the *PortletsProject* Web project, choose **Run On Server**, and choose **portal5** server in the pop-up window.

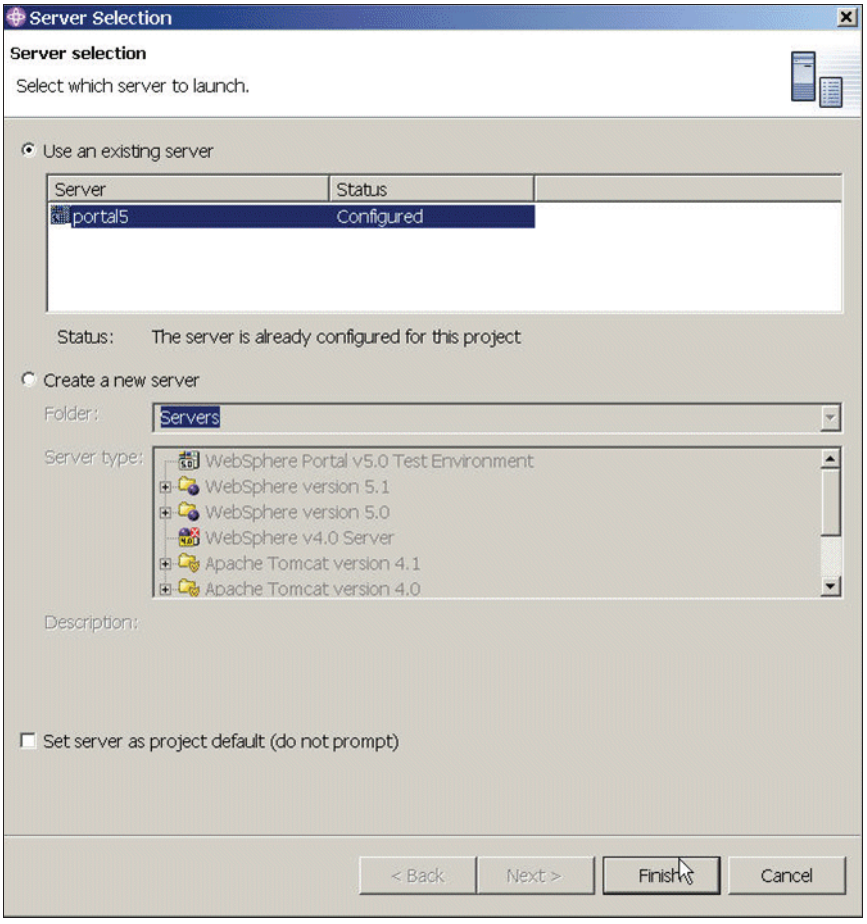


Figure 5-24 Selecting the server

6. In the browser you should get the information shown in Figure 5-25.

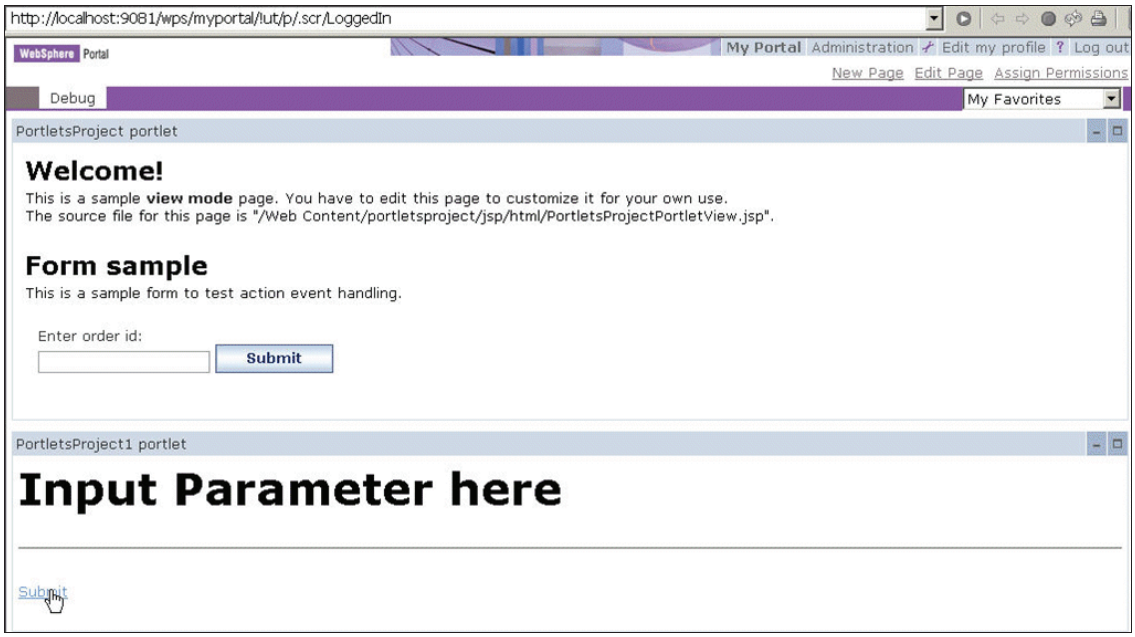


Figure 5-25 Sample portlet

7. Click the “Submit” at left bottom; you will see this “**Master Information**” if everything is fine.

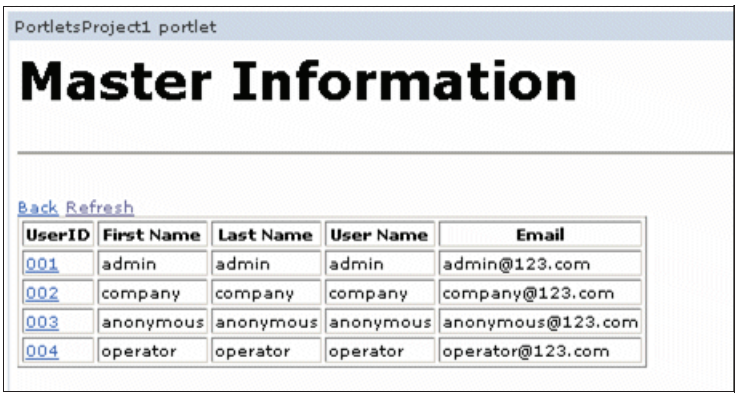


Figure 5-26 Master information

8. Click each record in “UserID” and you will get detailed information:



Figure 5-27 Sample details view

5.4 Export the portlet application

In this section you will go through the steps to export the sample portlet to the local file system. Execute the following steps:

1. Edit the portlet descriptor (portlet.xml). The following example assumes that WEA/Portal server's DB is DB2 with DB2admin id and password is db2admin. Save it.

```
<config-param>
    <param-name>username</param-name>
    <param-value>db2admin</param-value>
</config-param>
<config-param>
    <param-name>password</param-name>
    <param-value>db2admin</param-value>
</config-param>
<config-param>
    <param-name>driverName</param-name>
    <param-value>COM.ibm.db2.jdbc.app.DB2Driver</param-value>
</config-param>
<config-param>
    <param-name>url</param-name>
    <param-value>jdbc:db2:testdb</param-value>
</config-param>
```

Figure 5-28 Configuration parameters in portlet descriptor (portlet.xml)

2. Export the portlet application to a war file so that you can deploy the portlet application to the WEA/Portal server at a later time.
3. Select the project and click **Export...** .

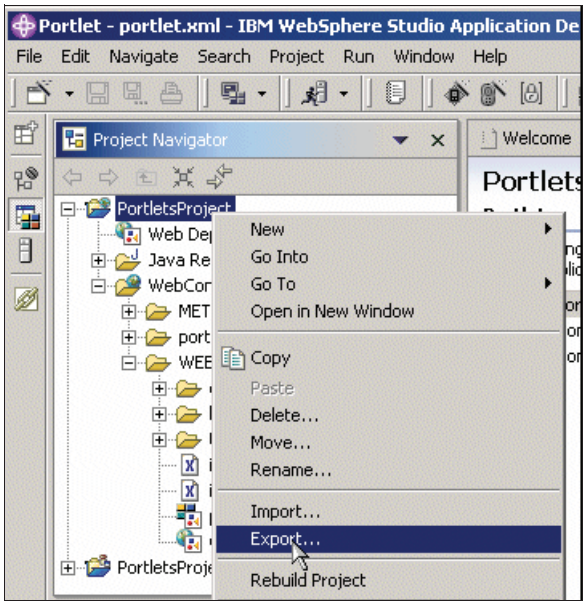


Figure 5-29 Export option

4. Select **WAR file** from the list and click **Next**.

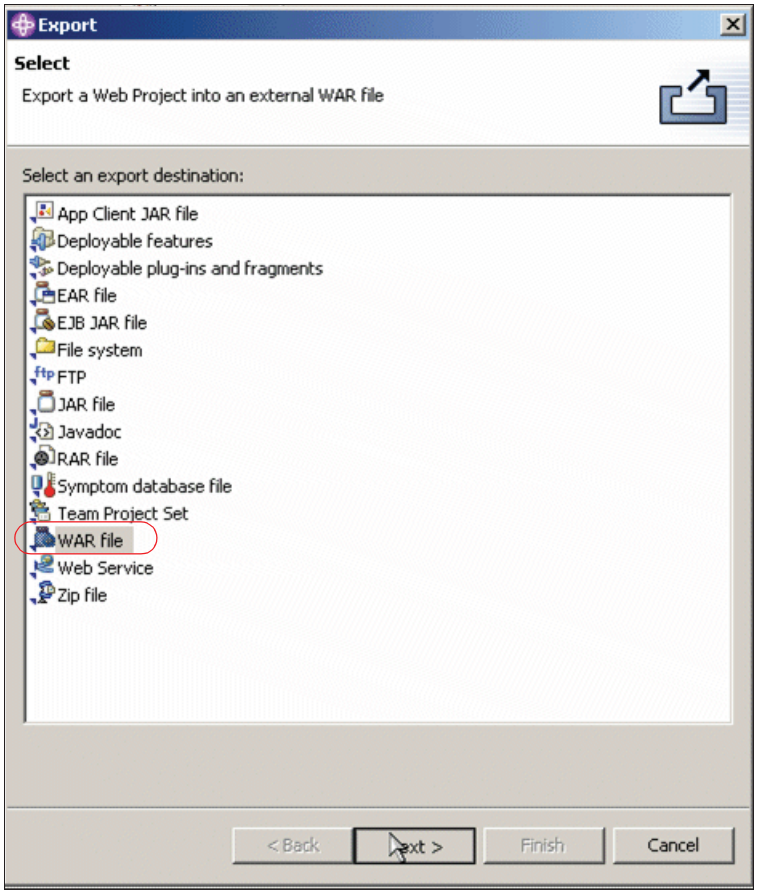


Figure 5-30 Export as external WAR file

5. Click **Browse...** button and navigate your local file system to export the war file to a proper destination.
6. Click **Finish**.

5.5 Deploy to your WEA/Portal server

In this section you will go through the steps to run the database access portlet on the WEA/Portal server.

Execute the following steps:

1. Before deploying the portlet to the WEA/Portal server, you will need to setup the database, table and some records. Open a DB2 command window to execute DB2 commands.



Figure 5-31 DB2 command window

2. For example, execute the following DB2 commands:
   a. db2 create db testdb
   b. db2 create table user(id varchar(5) not null, firstname varchar(10), lastname varchar(10), username varchar(20), email varchar(20), primary key(id))
   c. db2 insert into user values('001', 'admin', 'admin', 'admin', 'admin@123.com')
   d. db2 insert into user values('002', 'company', 'company', 'company', 'company@123')
   e. db2 insert into user values('003', 'anonymous', 'anonymous', 'anonymous', 'anonymous@123.com')
   f. db2 insert into user values('004', 'operator', 'operator', 'operator', 'operator'@123.com')

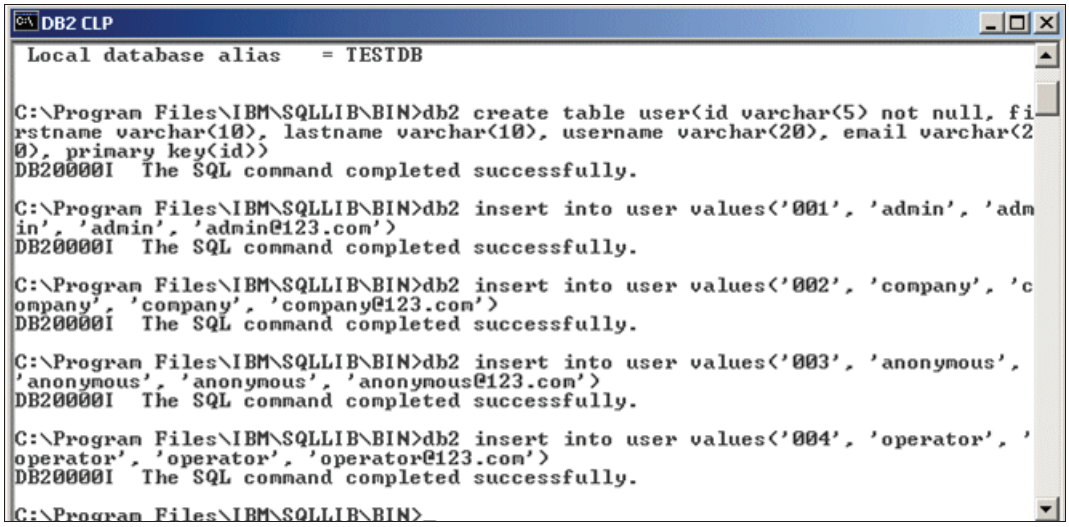


Figure 5-32 Command console

3. Open the DB2 Control Center by executing **Programs** -> **IBM DB2** -> **General Administration Tools** -> **Control Center**. Select Tables under TESTDB and double-click USER table.

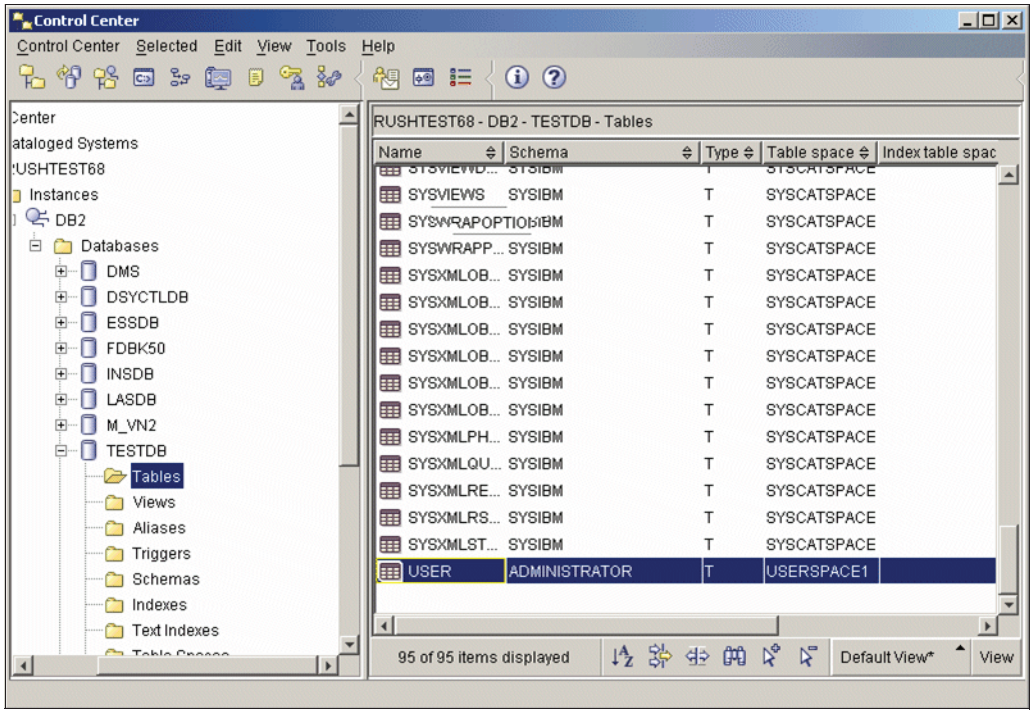


Figure 5-33 DB2 Control Center

4. Verify the table column names. Notice the ADMINISTRATOR.USER. Currently the DBAccess portlet has the table name hard coded and you will need to replace ADMINISTRATOR.USER table for the hard coded value.

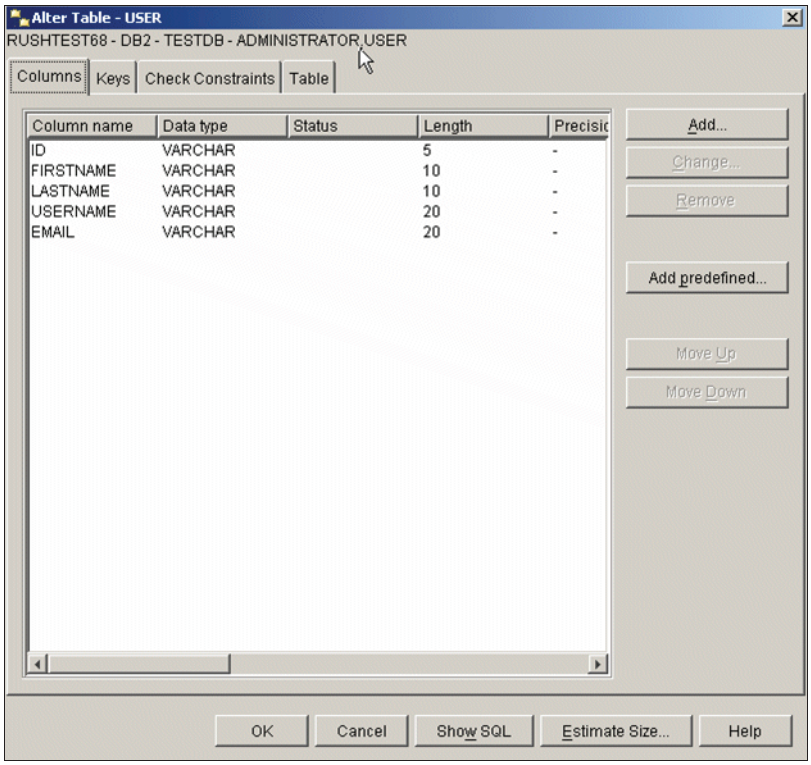


Figure 5-34 Alter table

5. Execute the following commands:
   a. db2 connect to testdb user <DB2Userid> using <DB2password>.
   b. db2 select * from ADMINISTRATOR. user

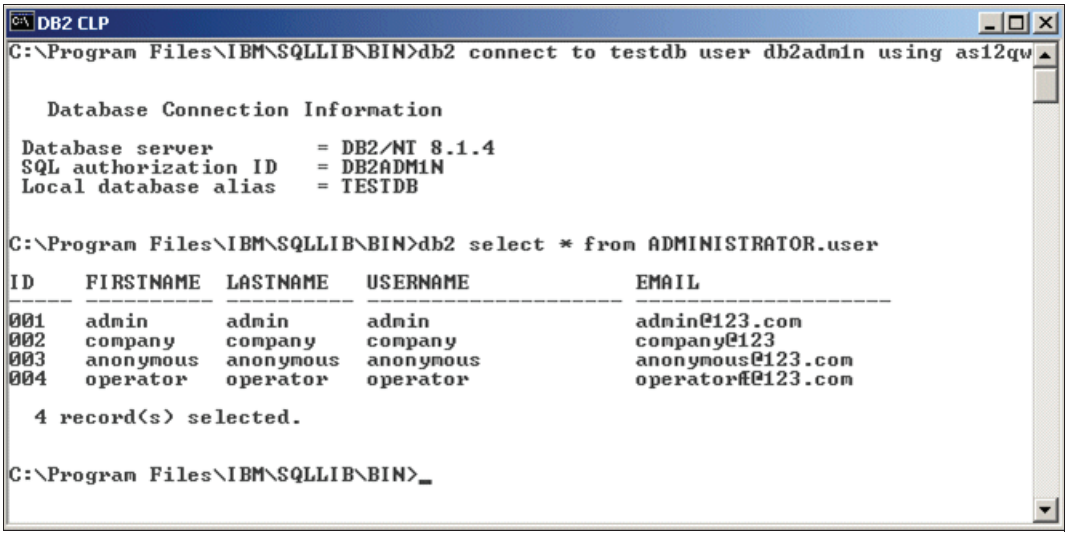


Figure 5-35 Command console

6. Fix the DBAccess portlet problem by editing <DBUserid>_USERMasterView.jsp. You will find the following hard coded statement.

```
<%--Execute the query--%>
<dab:select id="select_master" scope="request" connectionSpecRef="<%=dsSpec%>">
   <dab:sql>
      SELECT * FROM DB2ADMIN.USER
   </dab:sql>
</dab:select>
```

Figure 5-36 Select statement (master view)

7. Change SELECT * FROM DB2ADMIN.USER to SELECT * FROM ADMINISTRATOR.USER. Save.

```
<%--Execute the query--%>
<dab:select id="select_master" scope="request" connectionSpecRef="<%=dsSpec%>">
   <dab:sql>
      SELECT * FROM ADMINISTRATOR.USER
   </dab:sql>
</dab:select>
```

Figure 5-37 Modified select statement

8. Edit <DBUserid>_USERDetailsView.jsp. Change hard coded DB2ADMIN.USER to ADMINISTRATOR.USER.

```
<%--Execute the query--%>
<dab:select id="select_details" scope="request" connectionSpecRef="<%=dsSpec%>">
   <dab:sql>
      SELECT * FROM ADMINISTRATOR.USER WHERE ADMINISTRATOR.USER.ID = :DB2ADMIN_USER_USERID
   </dab:sql>
   <dab:parameter position="1" type="VARCHAR" value="<%=inputDRAGON_USER_USERID%>"/>
</dab:select>
<!--Banner-->
```

Figure 5-38 Select statement (details view)

9. Export the portlet again to your local file system as you did in Part 4.
10. Start WEA/Portal server and login to the server as an administrator.

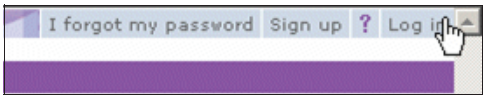


Figure 5-39 Log in

11. Click **Portlets** and then **Install**.

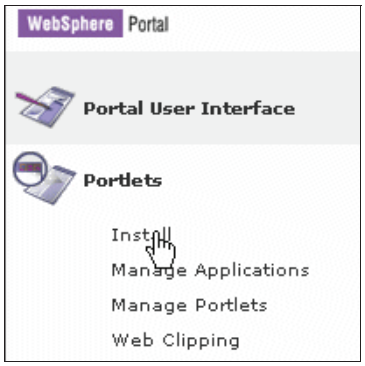


Figure 5-40 Select install portlets

12. Click the **Browse...** button to navigate the WAR file you previously exported. Click **Next**.

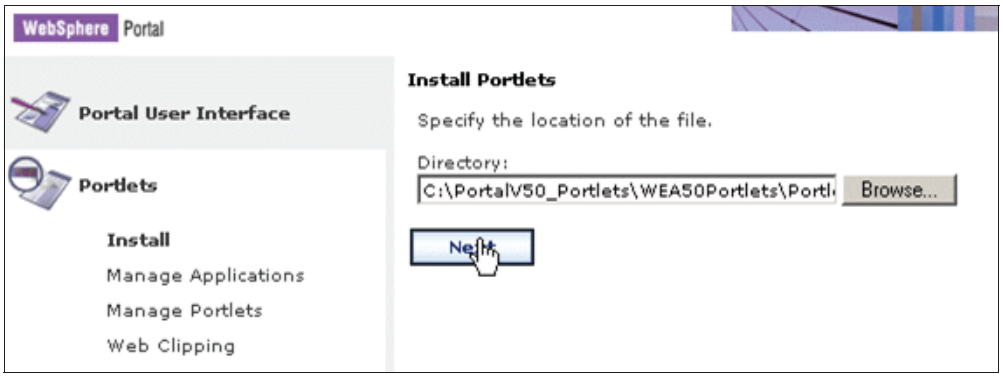


Figure 5-41 Install portlets location

13. Click **Install** button.

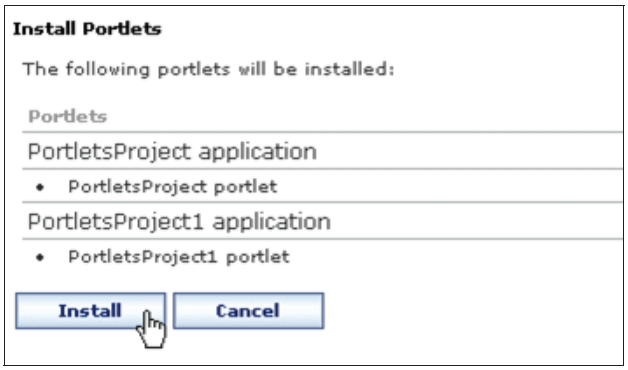


Figure 5-42 Install portlet

14. You will create a new page for this portlet to be deployed. Click **Portal User Interface**.

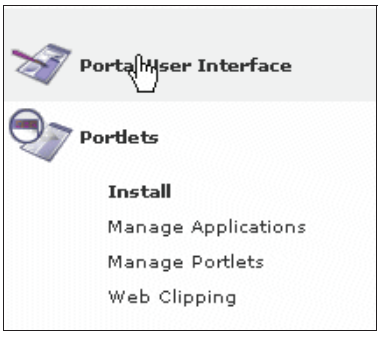


Figure 5-43 Portla user interface

15. Click **Manage Pages**.

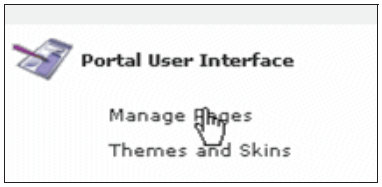


Figure 5-44 Manage pages

16. You will create a new page under My Portal. Click **My Portal**.

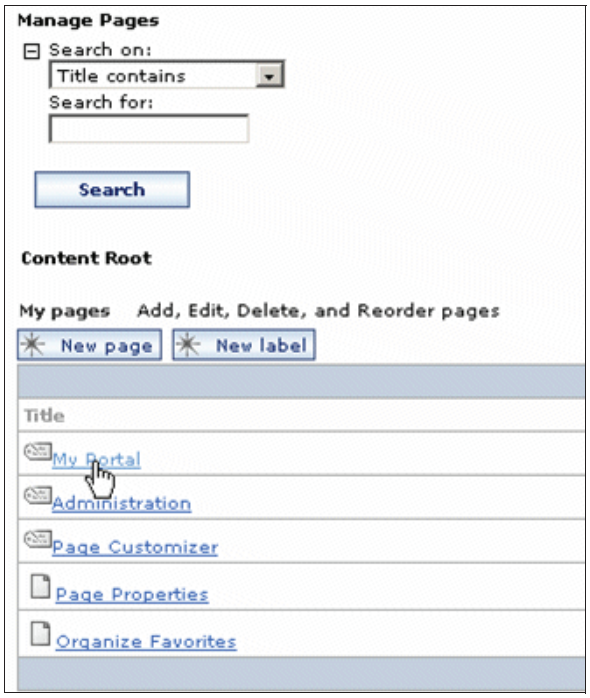


Figure 5-45 Manage pages

17. Click **New Page** tab.



Figure 5-46 New page

18. Enter **DBAccess** to the Title entry. Check **PDA** in addition to HTML. (In this lab, we will not use any other markup than HTML). Click **OK**.

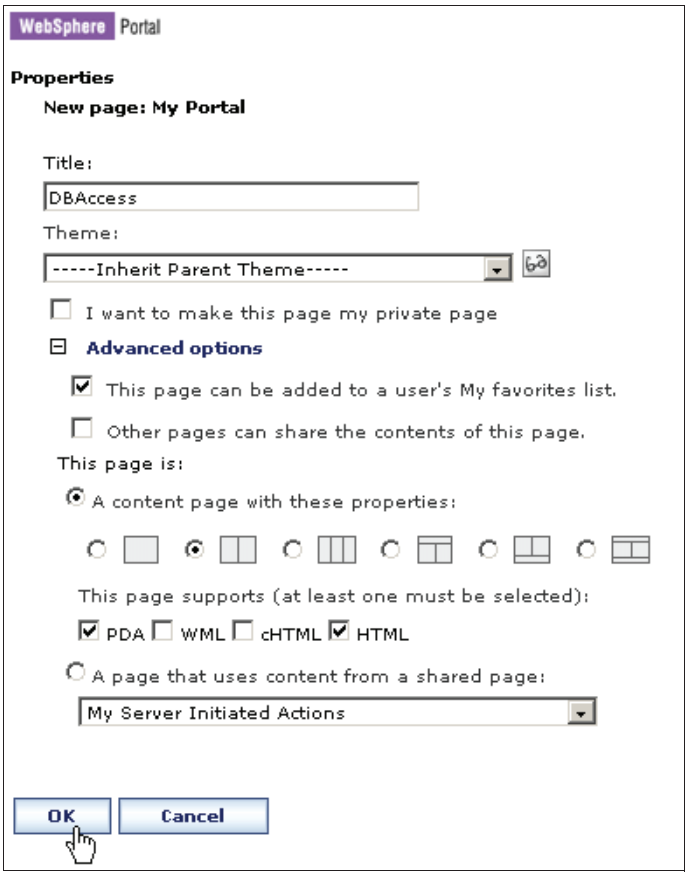


Figure 5-47 Page properties

19. Click **OK** after a successful page creation message.

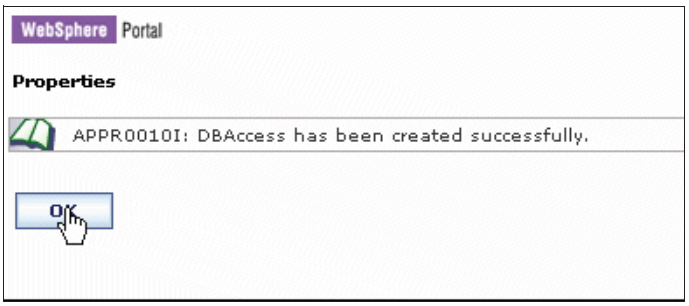


Figure 5-48 DBAccess page successfully created message

20. Click **My Portal** page.



Figure 5-49 My Portal page

21. You will see the newly created page called DBAccess. Click **DBAccess** page tab.



Figure 5-50 DBAccess page

22. Click **Edit Page** link.



Figure 5-51 Edit page

23. Click **Add portlets** button.

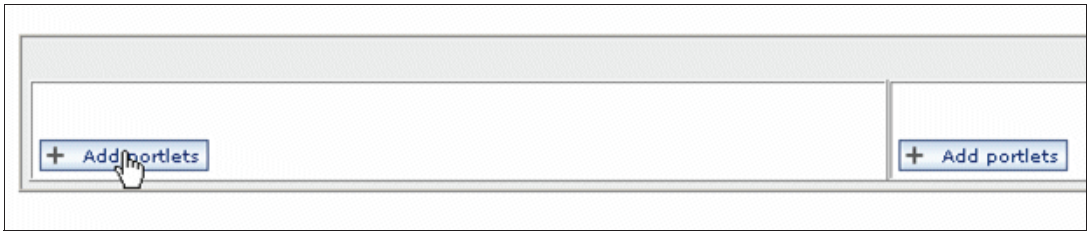


Figure 5-52 Add the portlet to the created page

24. Check **PortletsProject portlet** and click **OK**.

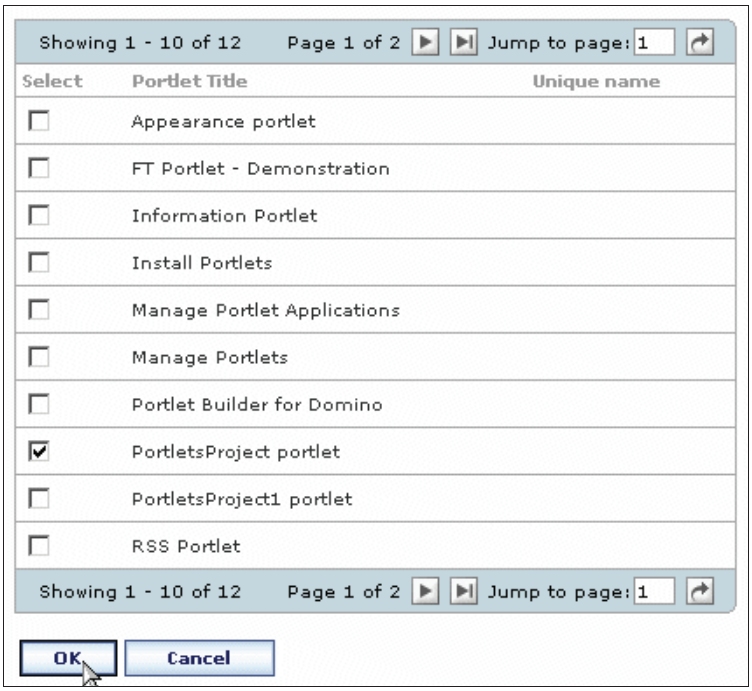


Figure 5-53 Select the portlet

25. Make sure the portlet is added.

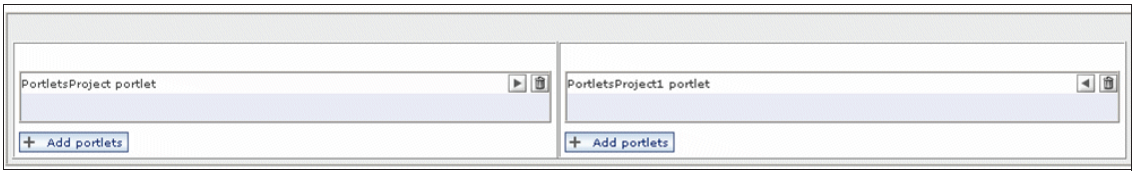


Figure 5-54 Portlet has been added

26. Click **Done** button.



Figure 5-55 Done button

27. You will see two portlets from DBAccess page. We are only interested in the right side portlet. Click **Submit** link from the PortletsProject 1 portlet.

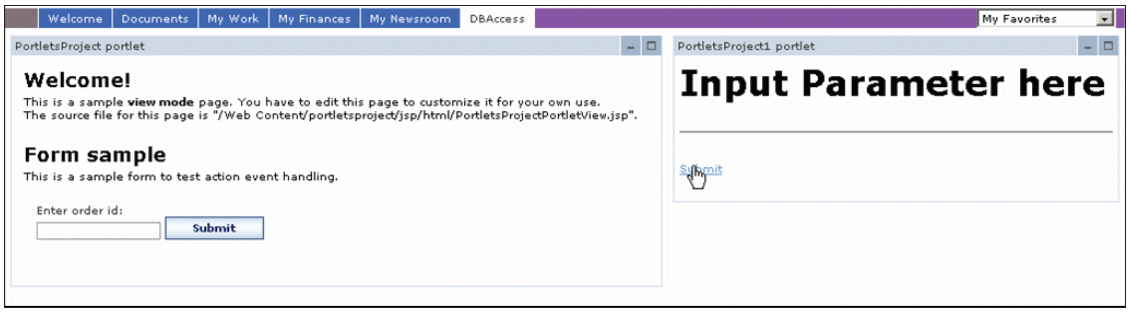


Figure 5-56 Execute the sample portlet

28. You will see the query result layout. Click any userID.

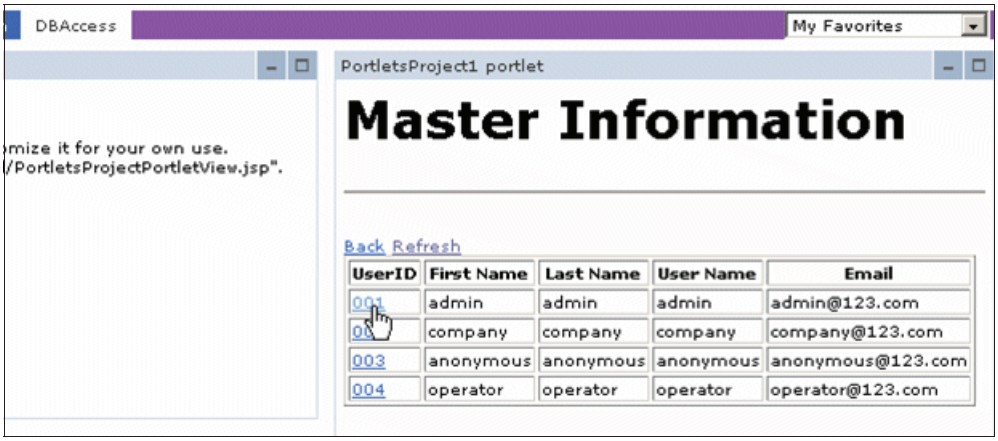


Figure 5-57 Master information

29. You will see the details view of the userID you clicked.

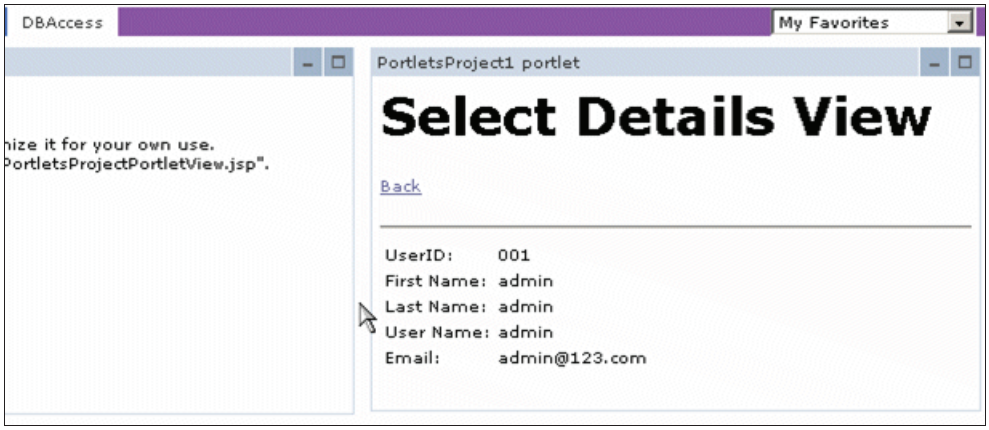


Figure 5-58 Details view

5.6 Execute the portlet from a Pocket PC

In this section you will execute the same portlet from a PocketPC. Overwrite all files from html directory under jsp to pda directory. Test the portlet using PocketPC device.

1. Create new JSP files for PDA or simply copy all files from html directory to the pda directory
2. If needed, export the portlet again and deploy it to the WEA/Portal server.
3. From your PocketPC, enter the WEA/Portal server URL.



Figure 5-59 WEA home

4. Click 
5. Login as an administrator. For example enter user ID wpsadmin and the proper password.
6. Scroll down Welcome combo list.
7. Select **DBAccess.** You will find two portlets.

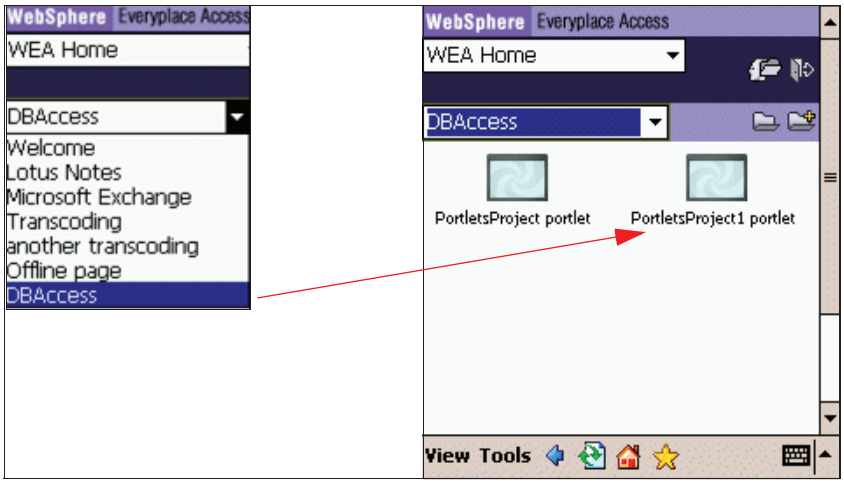


Figure 5-60 Available portlets in DBAccess page

8. Click **PortletsProject1 portlet** and click **Submit**.

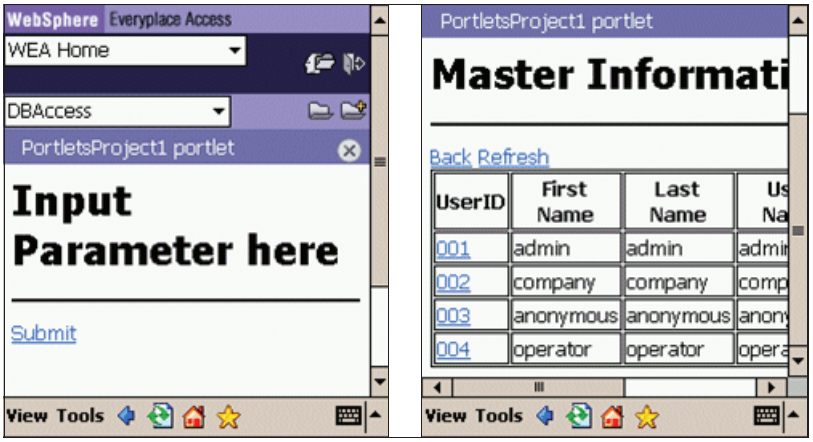


Figure 5-61 Executing the portlet submit

9. Click any userID. For example 003 in this sample scenario.

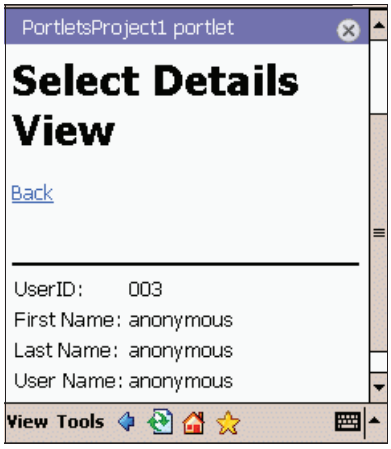


Figure 5-62 Details view

5.7 Use proper driver if the database is not DB2

In this section you will go through the steps to run the database access portlet in order to access a different database such as MySQL.

Execute the following steps:

1. Setup the MySQL server, create the database, table, and populate with some records. Refer to MySQL documentations if you have problems doing this.
2. Follow the same procedure to create the sample database portlet using the Database Access Portlet wizard but change the database driver as illustrated in Figure 5-63.

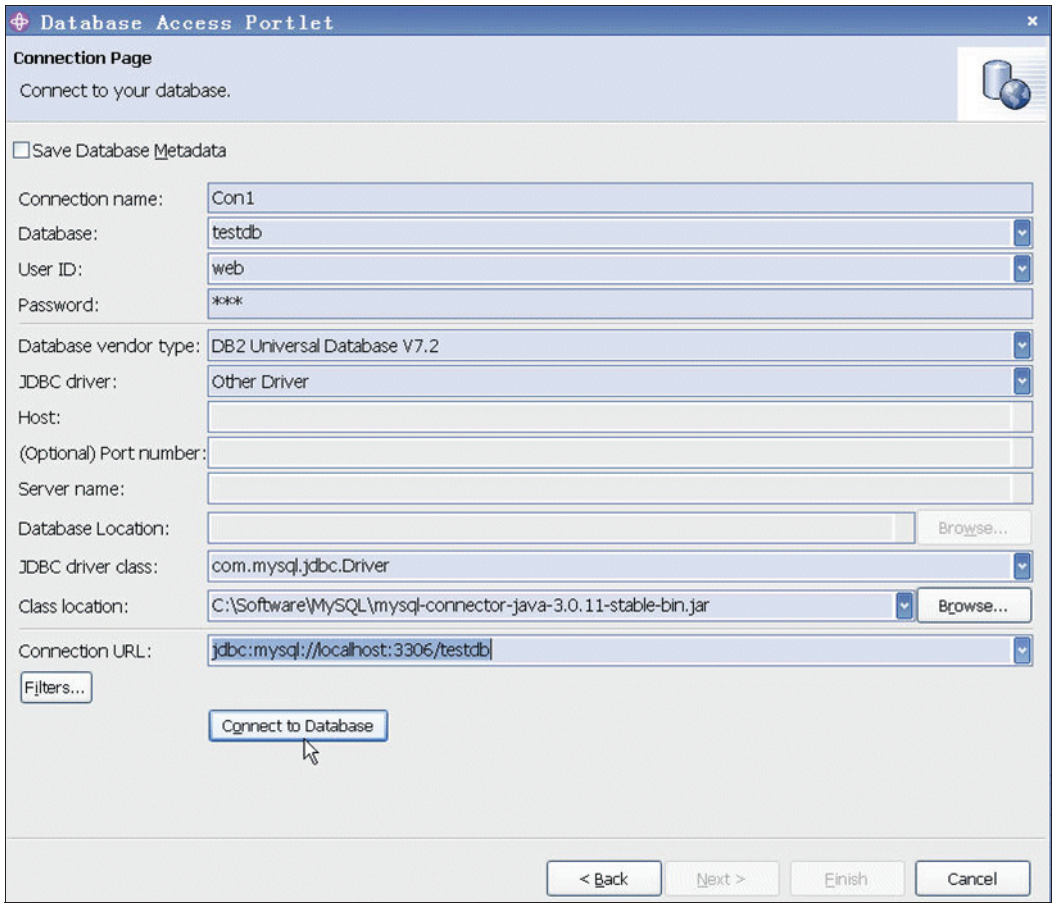


Figure 5-63 Configuring the proper JDBC driver

3. Once this is done, open the _prefix_DetailsView.jsp and the _prefix_MasterView.jsp to change the following line:

   ```
   SELECT * FROM "user" WHERE "user"."id" = :f_user___id_
   ```

   to:

   ```
   SELECT * FROM user WHERE user.id = :f_user___id_
   ```

   and:

   ```
   SELECT * FROM "user" to SELECT * FROM user
   ```

4. Add the MySQL JDBC driver to your Web project's lib directory. For example, click lib directory under WEB-INF, and import the **mysql-connector-java-3.0.11-stable-bin.jar** file to lib directory.

Note: You can download this JDBC driver from the MySQL Web site.

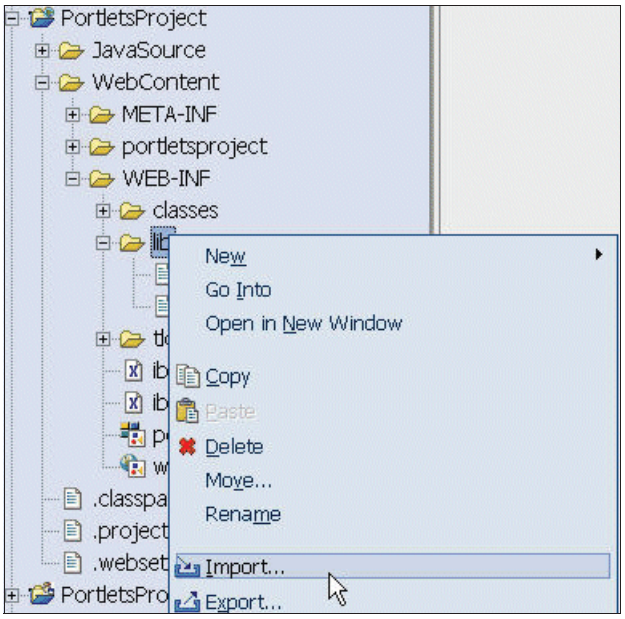


Figure 5-64 Importing the JDBC driver

5. Select the driver.

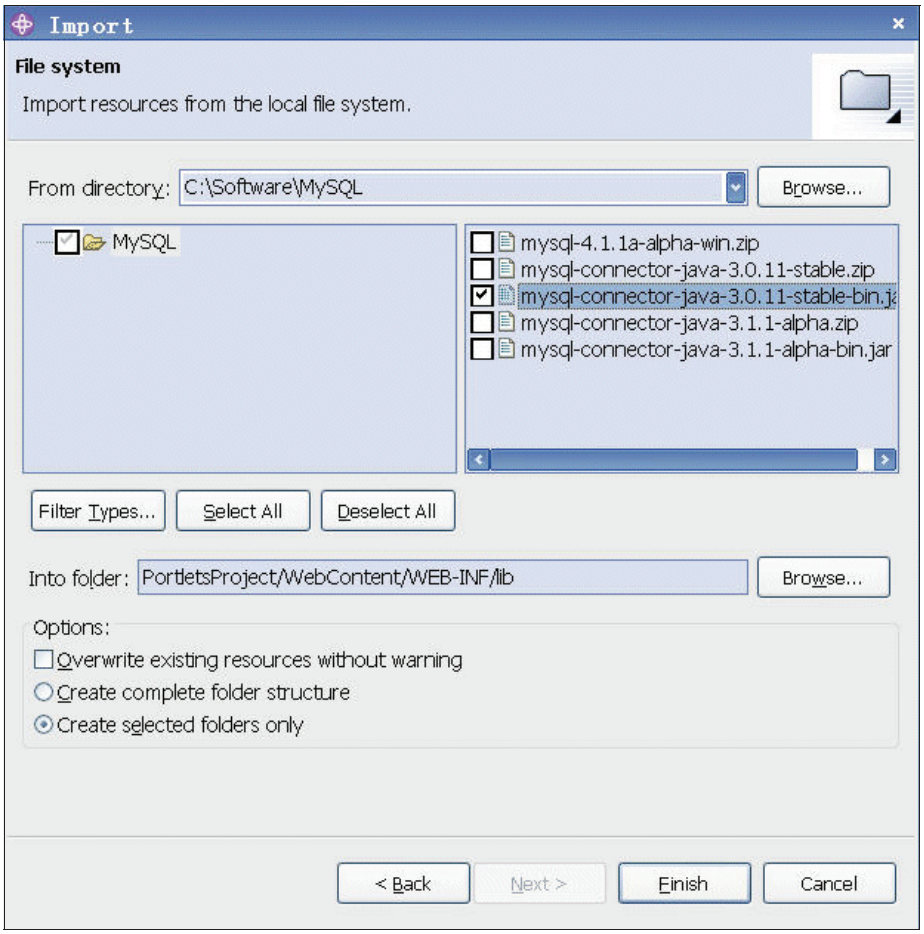


Figure 5-65 Select the driver

6. Then, try to publish your application to portal5 server, and start the portal5 server in the test environment. After the portal server is started, you can bring up your browser to test the portlet, you will see following windows if everything is fine.

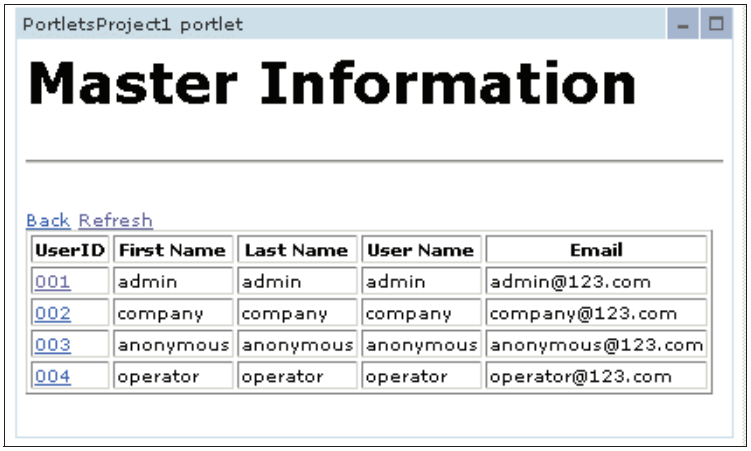


Figure 5-66 Master information

Note: Results are similar as for the DB2 scenario since the same table definition and records were used.

7. When you click user ID 002 you will get the details view for this user.

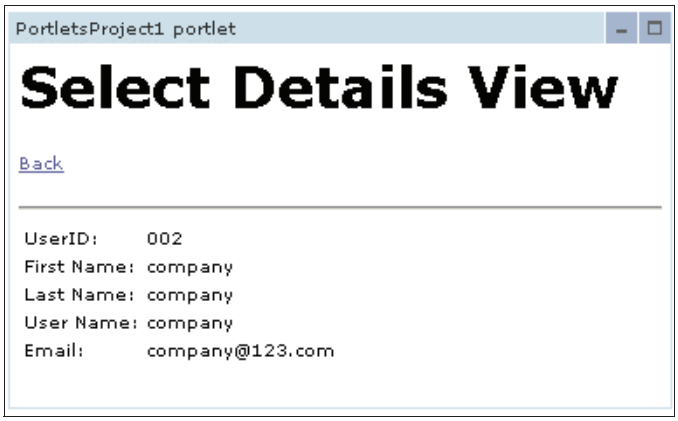


Figure 5-67 Details view

6

Using the Credential Vault

This chapter provides an overview and a sample scenario for creating a sample portlet application that uses Credential Vault to log in and interact with back-end systems. You will create, deploy and run portlet applications. The sample scenario will allow you to understand the techniques used to develop portlets using the Credential Vault provided by IBM WebSphere Portal, on which WebSphere Everyplace Access is based.

© Copyright IBM Corp. 2005. All rights reserved.

6.1 Overview

When integrating different back-end systems, portlets often need to provide some type of authentication to access these back-end systems. WebSphere Portal provides the use of a Credential Vault to store and retrieve user credentials. By using Credential Vault portlets, you can provide a single sign-on experience to the user.

After reading this chapter, you will be able to:

- Understand the value of Credential Vault for portlet development
- Identify the different components of Credential Vault
- Build portlet applications using Credential Vault technology and active and passive objects

Portlets running on WebSphere Portal may need to access remote applications that require some form of authentication by using appropriate credentials. In this section, we provide an overview of the Credential Vault components.

Credentials

Examples of credentials are user IDs and passwords, SSL client certificates and private keys. In order to provide a single sign-on user experience, portlets should not ask the user for the credentials of individual applications each time the user starts a new portal session. Instead, they must be able to store and retrieve user credentials for their particular associated application and use those credentials to log in on behalf of the user. The Portal back-end secure access is illustrated in Figure 6-1 on page 183.

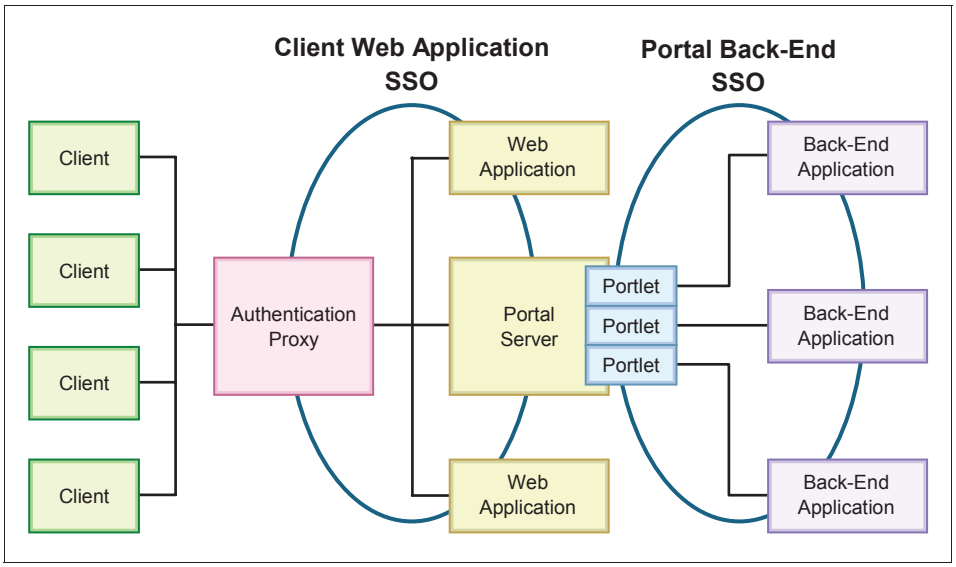


Figure 6-1 Credential Vault in action

The Credential Vault provides this functionality and portlets can use it through the Credential Vault Portlet Service.

Components of the Credential Vault organization

The organization of Credential Vault in WebSphere Portal consists of vault segments and credential slots. Figure 6-2 on page 184 shows an overview of these components.

Vault segments

The Credential Vault is partitioned into segments and a vault segment contains one or more credential slots.

There are two different types of vault segments:

- Administrator-managed segments: in this type of vault segment, the creation of new slots is restricted to the portlet administrator.
- User-managed segments: in this type of vault segment, portlets can also create new slots on behalf of the user.

Note: Setting and retrieving credentials can be performed by portlets for both types of vault segments.

Vault implementations are the actual locations where the credentials are stored. This can be for example the default database of WebSphere Portal or the Tivoli® Access Manager lock box.

Credential slots

As mentioned previously, every vault segment contains one or more credential slots. Slots are “drawers” where portlets store and retrieve a user’s credentials. Each slot holds one credential and links to a resource in a vault implementation. There are four different types of slots:

- A system slot stores system credentials where the actual secret is shared among all users and portlets.
- An administrative slot allows each user to store a secret for an administrator-defined resource (for example, Lotus® Notes®).
- A shared slot stores user credentials that are shared among the user's portlets.
- A portlet private slot stores user credentials that are not shared among portlets.

Note: In the sample scenario included in this chapter, only private slots will be used.

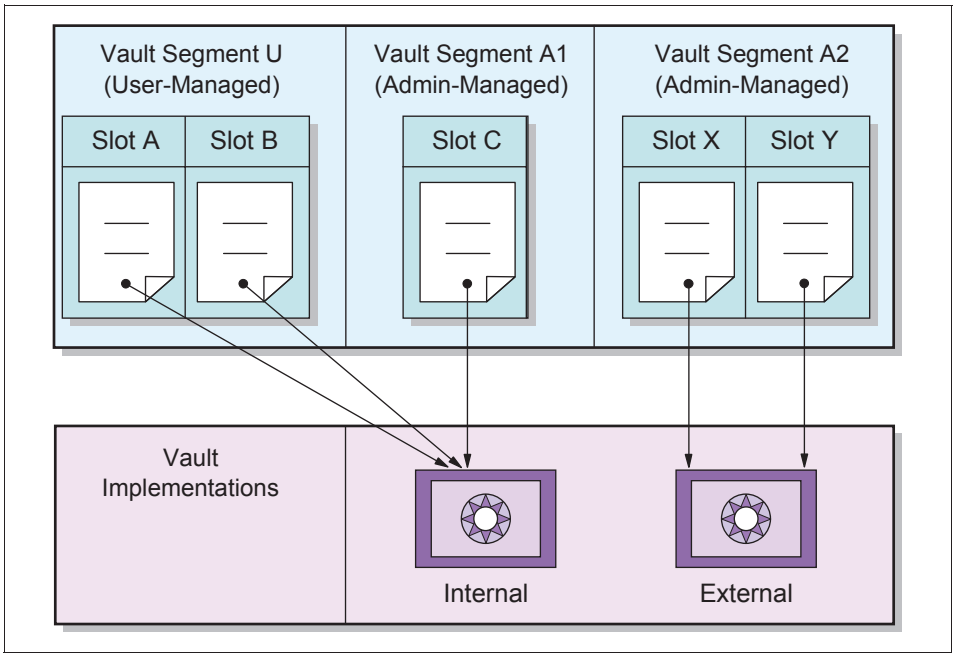


Figure 6-2 Credential Vault organization

Credentials objects

WebSphere Portal differentiates between passive and active credential objects:

- **Passive credential objects** are containers for the credential's secret. Portlets that use passive credentials need to extract the secret out of the credential and do all the authentication communication with the back-end resource. The following passive credential support is provided with WebSphere Portal:
  - UserPasswordPassive, which stores secrets in the form of user ID/password pairs
  - SimplePassive, which stores secrets in the form of serializable Java objects
  - JaasSubjectPassive (Java Authentication and Authorization Service), which stores secrets in form of javax.security.auth.Subject objects

  Currently, the vault service in WebSphere Portal only supports UserPasswordPassive.
- **Active credential objects** hide the credential's secret from the portlet; there is no way of extracting it out of the credential. In return, active credential objects offer business methods that take care of all the authentication. The following active credential support is provided with WebSphere Portal:
  - HttpBasicAuth
  - HttpFormBasedAuth
  - JavaMail
  - LtpaToken
  - SiteMinderToken
  - WebSealToken

Note: When using active credentials, portlets never get in touch with the credential secrets and thus there is no risk a portlet could violate any security rules such as, for example, storing the secret on the portlet session. While there might not always be an appropriate active credential class available, this is the preferred type of credential objects to use.

6.2 Sample scenario

In this sample scenario, you will create a sample portlet based on a Basic portlet type using the Portlet Wizard. You will also use this wizard to enable Credential Vault to interact with back-end resources.

In this scenario, the protected back-end resource is a servlet and requires a user ID and password credentials to log in to the Web application (servlet). The servlet application has been secured with HTTP Basic Authentication.

The sample scenario illustrates the following:

- How the Credential Vault with active credentials is used
- How the Credential Vault with passive credentials is used
- How to store credentials
- How to retrieve credentials
- How to log in to the Web application
- How to retrieve the Web application content in the portlet's View mode

In the first part of this scenario, active credentials are used to access a secure Web application using HTTP Basic Authentication, as shown in Figure 6-3.

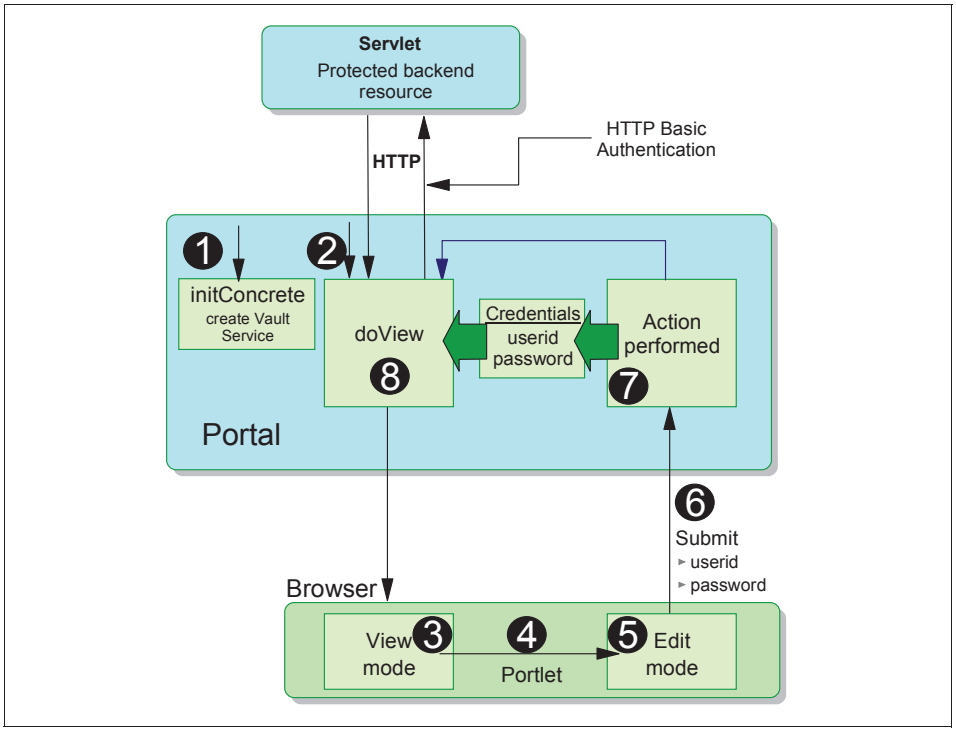


Figure 6-3 Credential Vault sample scenario

The sequence flow for this scenario is as follows:

1. The initConcrete method is used to initialize the Credential Vault Service.

2. Portal invokes the portlet doView method. Since initially, no credentials have been stored, a message is written indicating that a user ID and password must be entered in Edit mode.
3. In the portlet View mode, a message is shown directing the user to use the Edit mode to enter credentials.
4. The user clicks **Edit** to go into Edit mode.
5. The Edit mode screen is displayed, that is, the doEdit method is executed and a JSP displays a form to enter credentials and submit the action.
6. The user enters a user ID and password and selects **Submit**.
7. The actionPerformed method is executed to process the action. It creates a slot and stores the user ID/password information.
8. The doView method is executed to complete the cycle. The following tasks are executed in this mode:
   a. An HttpBasicAuth active credential object is retrieved from the credential service. Because authentication is done in this object, we never get in touch with the real credentials.
   b. The authorization header is set in the request HTTP header.
   c. The connection to the back-end resource (protected servlet in this scenario) is invoked.
   d. The user is authenticated and the servlet executes.
   e. The received content is rendered in View mode.

6.2.1 Importing a protected servlet application

In this section, you will import a previously created Treasure servlet. This servlet will be the back-end secure resource you will access using Credential Vault. The servlet displays a simple image and is only accessible via HTTP basic authentication.

Note: The sample scenario included in this chapter requires that you download the sample code available as additional materials. See Appendix B, “Additional material” on page 443.

Follow these steps to import the secure servlet:

1. If required, start WebSphere Studio Site Developer.
2. Switch to the Web perspective.
3. From the main menu, select **File -> Import...**.

4. Select **WAR file** and click **Next**.

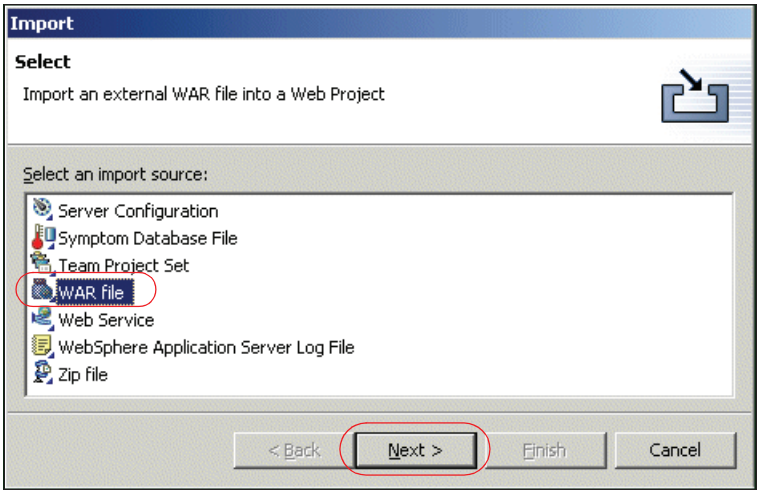


Figure 6-4 Importing a WAR file

5. Browse to the location of the TreasureWeb.war file that can be obtained from the additional materials for this redbook.
6. Click **New** to begin configuring a new project.
7. Enter TreasureWeb as the project name, then click **Next**.

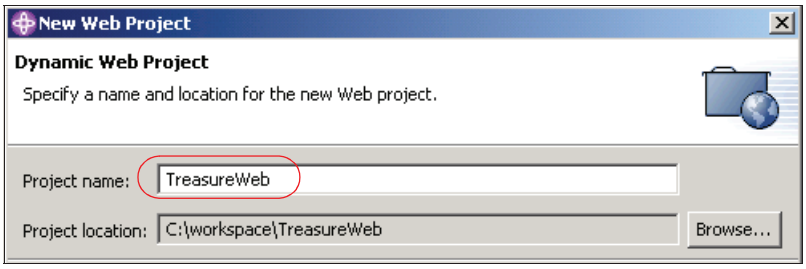


Figure 6-5 Starting a new dynamic Web project

8. On the EAR project window, rename the EAR project to TreasureEAR. Also change the J2EE level to 1.2. Then click **Next**.

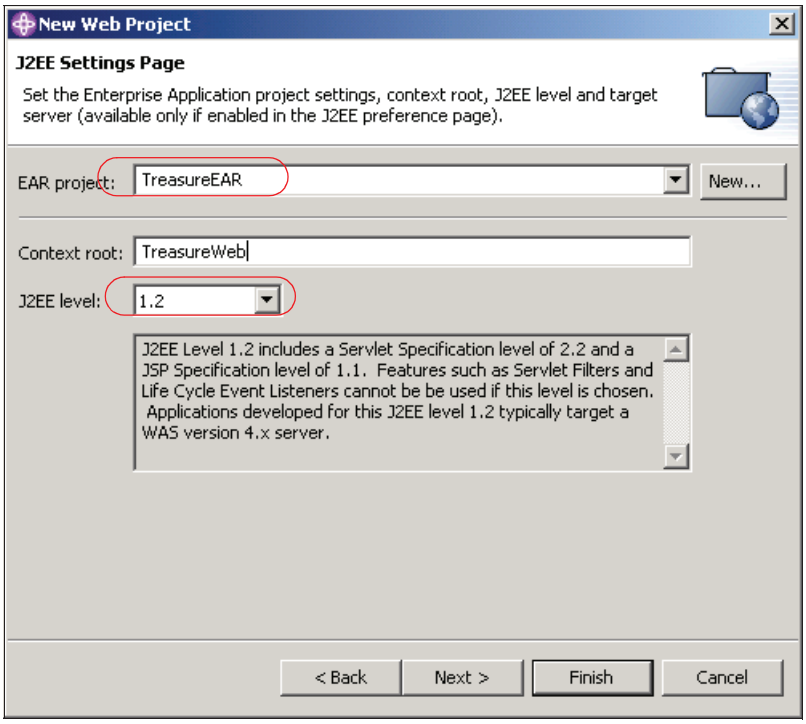


Figure 6-6 J2EE settings window

9. Uncheck Default style sheet on the following window, then click **Next**.
10. Click **Finish** to import the application (secure servlet WAR file).

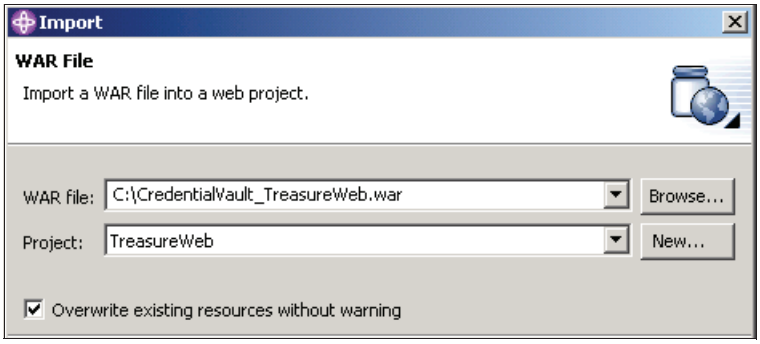


Figure 6-7 Importing the project

After importing the WAR file, a new Web and enterprise project exist in your project.

6.2.2 Using active credentials

After importing the protected servlet, you will build a portlet application accessing the TreasureWeb servlet and using active credential objects.The portlet will be created based on the Basic portlet type and will demonstrate the use of credentials. Once the project is created, you will run it in the WebSphere Portal Test Environment to view it.

Creating the Credential Vault portlet application

To create the new portlet project, follow these steps:

1. Switch to the Portlet perspective (**Window -> Open Perspective**).
2. Select **File -> New -> Other**.

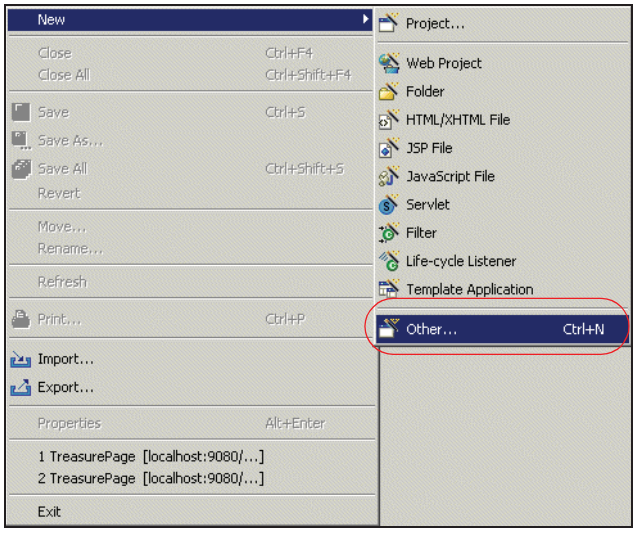


Figure 6-8 Invoking New Project wizard

3. Select **Portlet development** -> **Portlet application project.** Click **Next**.

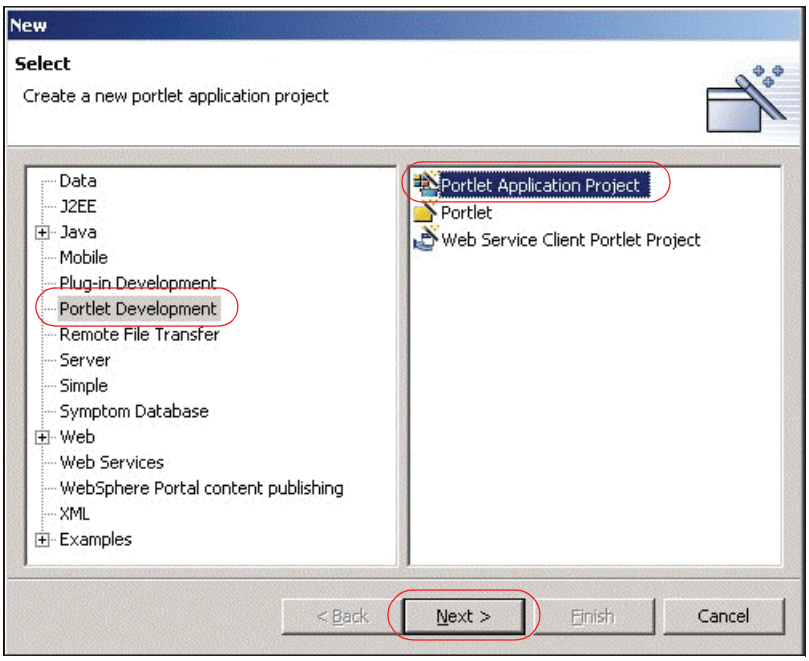


Figure 6-9 Creating a new portlet application

4. In the Define the Portlet Project page, enter a project name of `CredVaultBasicAuth` and click **Next**.

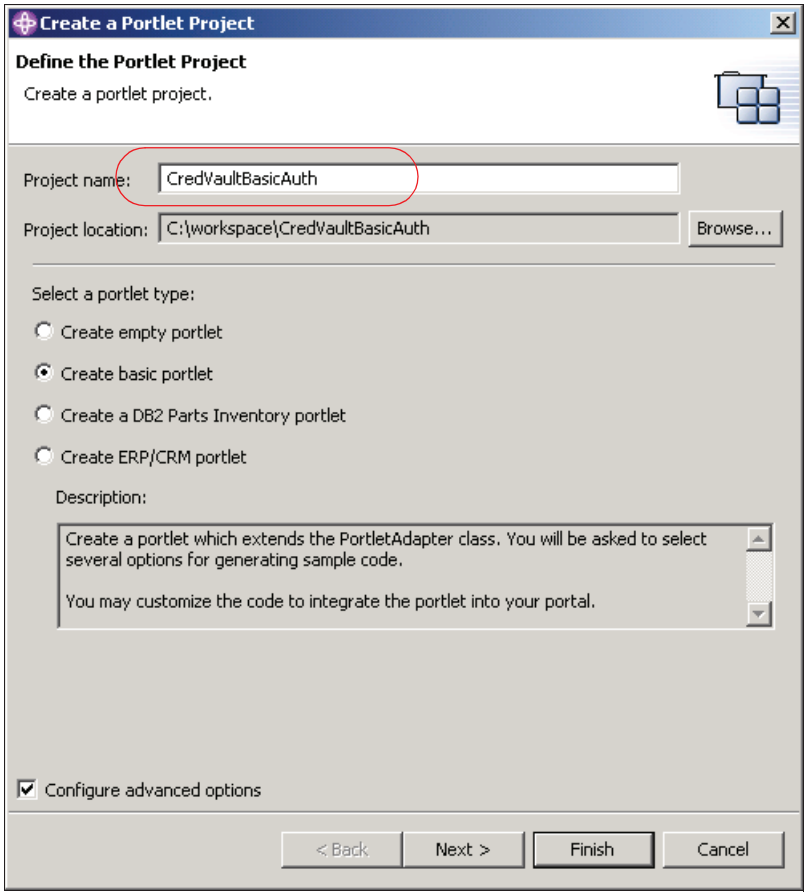


Figure 6-10 Define the Portlet Project

5. In the J2EE Settings Page, leave the defaults and click **Next**.
6. In Portlet Settings, accept all values. Click **Next**.
7. In Event Handling, uncheck **Add form sample** so that only **Add action listener** is checked. Click **Next**.

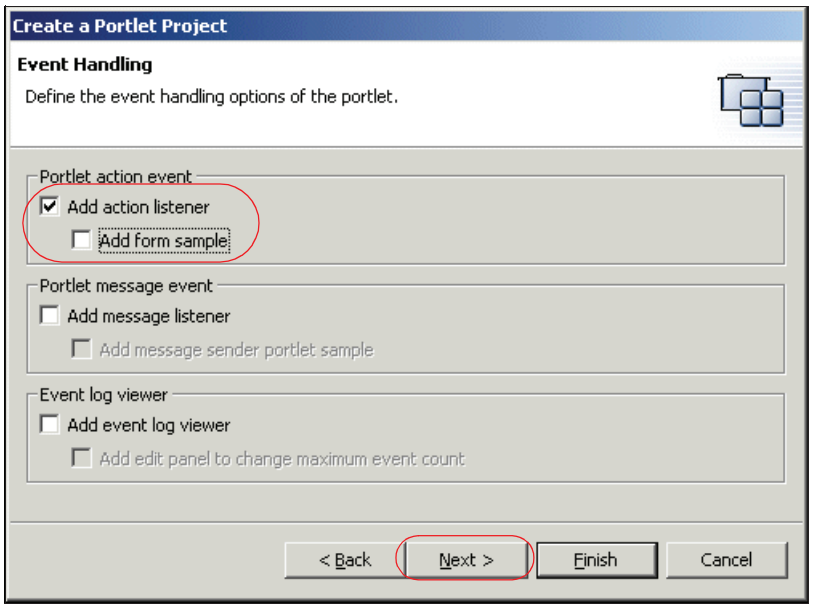


Figure 6-11 Event Handling page of New Portlet Application wizard

8. In the Single Sign-On page, check **Add credential vault handling** and enter a slot name of `TreasureCredentialSlot`. Click **Next**.

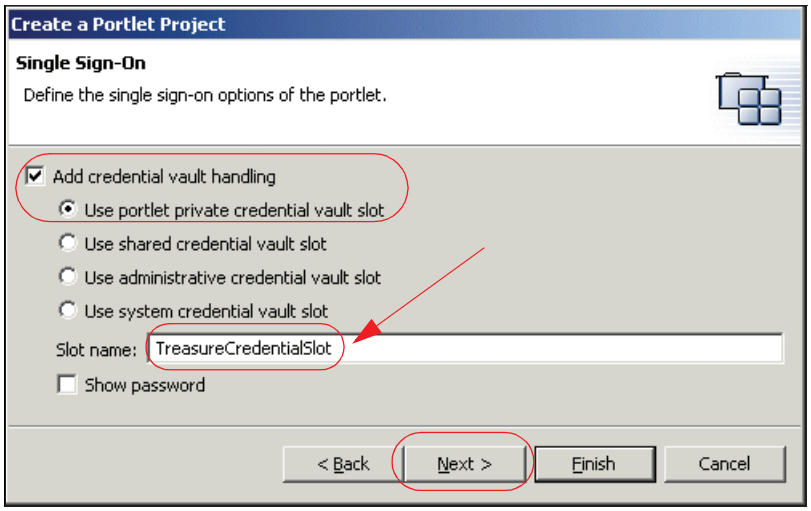


Figure 6-12 Single Sign-On page of the new portlet application wizard

9. Add PDA markup and click **Finish** to generate the portlet.

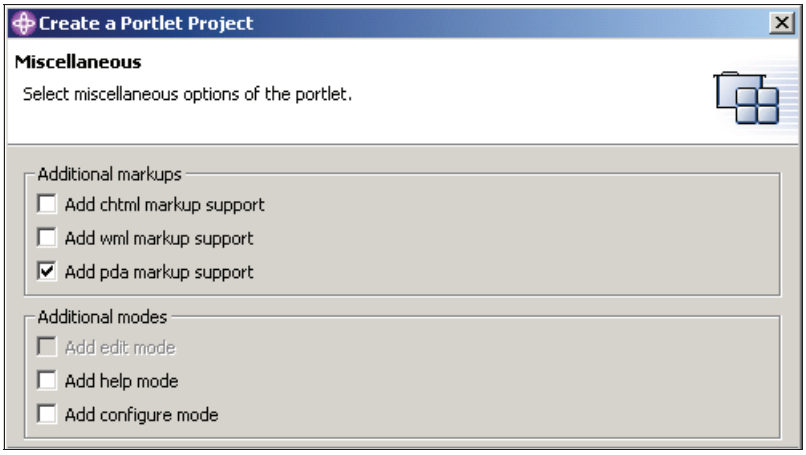


Figure 6-13 Adding PDA markup

Reviewing the generated code

Before the portlet code is modified to access the secure portlet, let's examine the wizard generated code.

If you expand the credvaultbasicauth package in the Source folder of the new project, you can see a CredVaultBasicAuthSecurityManager class in addition to the portlet and bean classes. This class is responsible for initializing the Credential Vault service and administering the credentials.

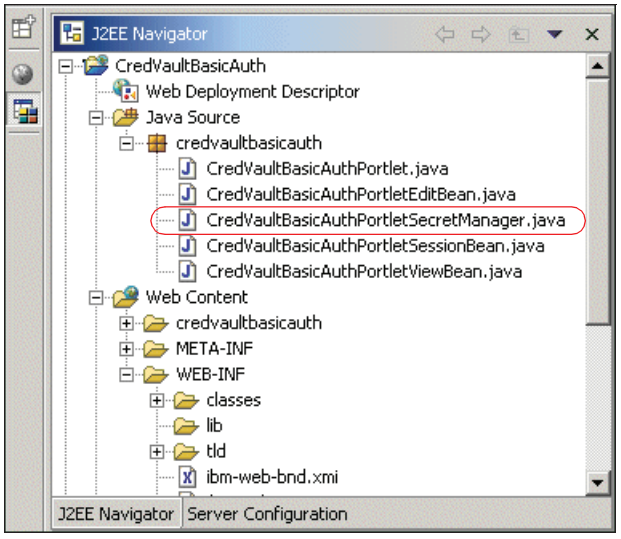


Figure 6-14 Reviewing CredVaultBasicAuthSecurityManager class

The following methods are provided in this class to handle Credential Vault issues:

- The init method of this class initializes the vaultService data member.
- getCredential returns the user name and password by using a string buffer.
- setCredential sets the user name and password.
- getSlotId returns the ID of the slot. Depending on the type of slot, this method uses PortletData or VaultService to get the ID.
- New slots are created in the createNewSlot method.
- getPrincipalFromSubject retrieves the specified Principal from the provided subject.
- isWritable checks whether the password can be saved.

The wizard has also created an input form for a user ID and password in the CredVaultBasicAuthPortletEdit.jsp. As previously described, when clicking the **Save** button, the actionPerformed() method in the portlet class is called. This method retrieves the user ID and password from the form and uses the security manager class to set the credentials.

The current version of the doView method retrieves the user credentials from the security manager and displays them in the JSP. Because we want to include the content of the secured Treasure servlet, we will replace this method in the next section of this scenario.

6.2.3 Updating the generated portlet

Modify the portlet application as follows:

1. Open CredVaultBasicAuthPortletSecretManager from the credvaultbasicauth package.
2. Using the Java editor, add the method shown in Example 6-1 to the class.

   Note: The sample scenario included in this chapter requires that you download the sample code available as additional materials. See Appendix B, “Additional material” on page 443.

   You may want to use WordPad to edit getConnectionUsingActiveObject.java and then copy and paste.

   Note: The getConnectionUsingActiveObject method returns an http connection.

Example 6-1 getConnectionUsingActiveObject method (active credentials)

```
public static HttpURLConnection getConnectionUsingActiveObject(
   PortletRequest portletRequest,
```

```
    CredVaultBasicAuthPortletSessionBean sessionBean,
    String host, String port, String path) {
    HttpURLConnection connection=null;
    try {
       URL urlSpec =
          new URL("http://" + host + ":" + port + path);
       String slotId = getSlotId(portletRequest, sessionBean, false);
       if (slotId != null) {
          HttpBasicAuthCredential credential =
             (HttpBasicAuthCredential) vaultService.getCredential(
                slotId,
                "HttpBasicAuth",
                new HashMap(),
                portletRequest);

          connection = credential.getAuthenticatedConnection(urlSpec);
       }
    } catch (Exception e) {
       e.printStackTrace();
    }
    return connection;
}
```

3. Some code errors appear because the required import statements are missing. To fix these errors, right-click the Java editor and select **Source -> Organize Imports**.
4. In the Organize Imports dialog, choose:
   a. **java.net.HttpURLConnection**
   b. **select java.net.URL**

   Click **Finish** to close the Organize Imports dialog.

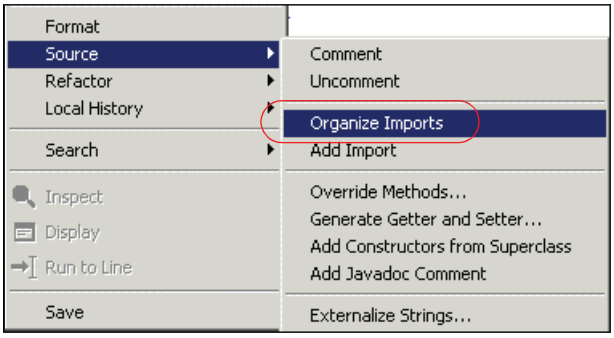


Figure 6-15 Importing missing import statements using Organize Imports tool

5. Save and close the Java file.

6. Open the class CredVaultBasicAuthPortlet from the credvaultbasicauth package.
7. Replace the doView method so it looks as shown in Example 6-2.

   Note: The sample scenario included in this chapter requires that you download the sample code available as additional materials. See Appendix B, “Additional material” on page 443.

Example 6-2 The doView method uses a Http connection from the SecretManager class

```
public void doView(PortletRequest request, PortletResponse response)
   throws PortletException, IOException {
   // Check if portlet session exists
   CredVaultBasicAuthPortletSessionBean sessionBean =
      getSessionBean(request);
   if (sessionBean == null) {
      response.getWriter().println("<b>NO PORTLET SESSION YET</b>");
      return;
   }
   // get output stream to write the results
   PrintWriter writer = response.getWriter();
   // get the CredentialVault PortletService
   PortletContext context = this.getPortletConfig().getContext();
   try {
      String host = request.getServerName();
      //String host = request.getRemoteHost();
      String port = String.valueOf(request.getServerPort());
      String path = "/TreasureWeb/TreasurePage";
      HttpURLConnection connection =

CredVaultBasicAuthPortletSecretManager.getConnectionUsingActiveObject(
               request, sessionBean, host, port, path );
         if (connection != null) {
            connection.connect();
            String responseMessage = connection.getResponseMessage();
            int responseCode = connection.getResponseCode();
            // Were we successful?
            if (HttpURLConnection.HTTP_OK == responseCode) {
               writer.println("<P>Successfully connected!</P>");
            } else {
               writer.println(
                  "<P>Unable to successfully connect to back end."
                     + ", HTTP Response Code = " + responseCode
                     + ", HTTP Response Message = \"" + responseMessage
                     + "\"</P>");
            }
            BufferedReader br =
               new BufferedReader(
                  new InputStreamReader(connection.getInputStream()));
```

```
            String line;
            while ((line = br.readLine()) != null)
               writer.println(line + "\n");
         } else {
            writer.println(
               "<h2>Credential not found. Please set it in the edit mode!
</h2>");
            return;
         }
      } catch (IOException exc) {
         writer.println(
            "<h2>Single-sign-on error, login at back-end failed! </h2>");
         return;
      }
   }
```

8. Organize the import statements as you did before.
9. Save and close the Java file.

Important: If you get a message indicating that getConnectionUsingActiveObject() is undefined, try making a small modification to the file and enabling the save option. Save the file again. This procedure should resolve any pending undefined issues.

10. Expand the CredVaultBasicAuth/WebContent/credvaultbasicauth/jsp/html folder. Copy the CredVaultBasicAuthPortletEdit.jsp and CredVaultBasicAuthPortletView.jsp files and paste them into the CredVaultBasicAuth/WebContent/credvaultbasicauth/jsp/pda folder. This ensures that the PDA markup will be able to perform the same actions as the HTML equivalent.

6.2.4 Running the portlet

In this section, you will run the portlet using active credentials to access the back-end resource, a protected servlet in this case.

1. Close any open browser viewers.
2. Switch to the Server perspective.
3. In the Server Configuration panel, right-click the **Servers** folder and choose **New -> Server and Server Configuration.**

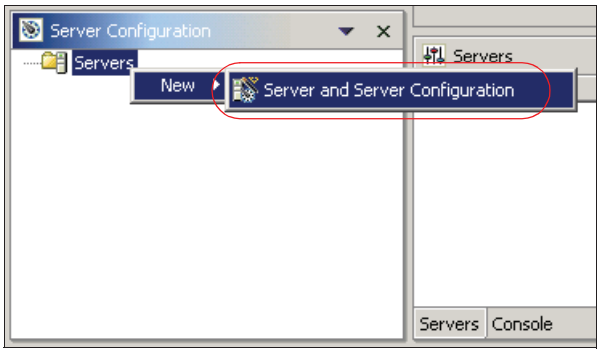


Figure 6-16 Creation of a new server

4. In the Server Selection dialog, choose a server of the WebSphere Portal V5.0 Test Environment and enter a server name of `WPS 5.0`. Click **Next** and accept the default port for the server. Finally, click **Finish** to add the new server.

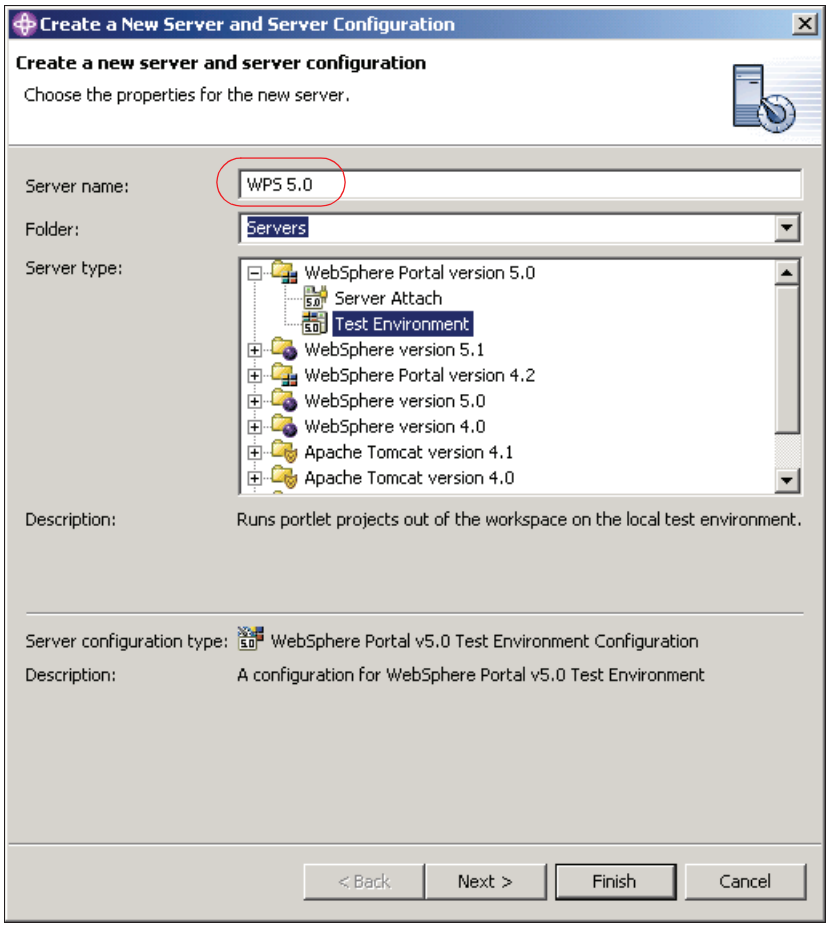


Figure 6-17 Create a new WebSphere Portal Test Environment

5. Right-click the **WPS5.0** server and choose **Add and remove projects**.

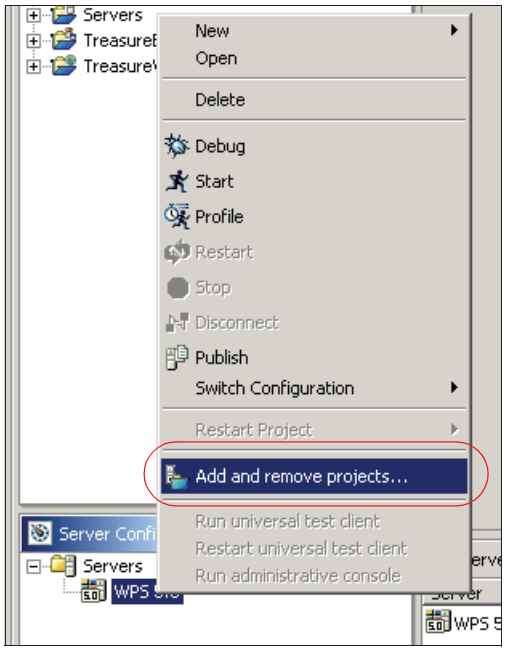


Figure 6-18 Add projects to portal test environment

6. Add the two projects to the portal test environment. Click **Finish**.

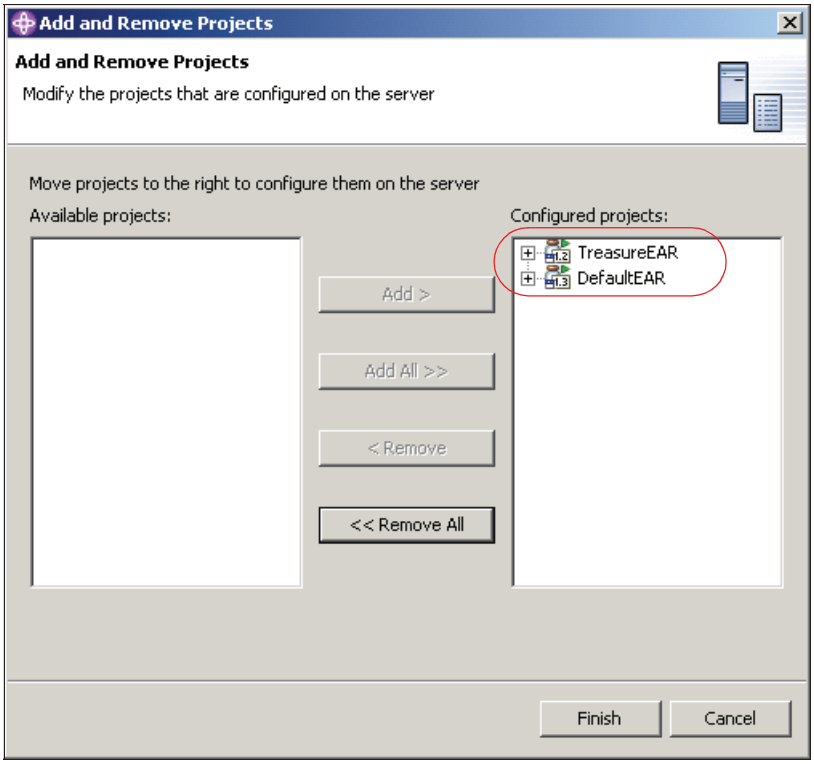


Figure 6-19

7. Switch the Portlet perspective.
8. In the J2EE Navigator view select **CredVaultBasicAuth**, choose **Run on server** and wait a few minutes for the Portal server to open for e-business. This will start the server and will also open a browser displaying the portlet.

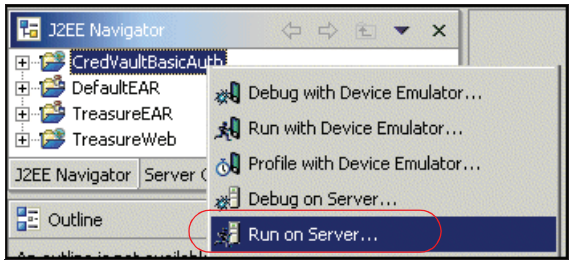


Figure 6-20 Selecting Run on Server... to test the portlet

9. Access the portlet with the Palm Simulator by entering the following URL:

   http://localhost:9081/wps/portal/!ut/p/.cmd/LoginUserNoAuth?userid=wpsadmin&password=wpsadmin

10. The portlet will execute the initConcrete method to initialize the Credential Vault Service and the doView method. Since there are no credentials yet, a message is displayed.

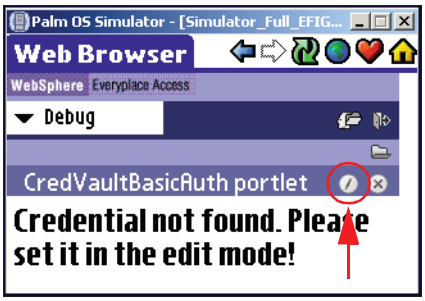


Figure 6-21 Credential not found, enter edit mode

11. Switch to the Edit mode as indicated by the arrow in the above figure and enter the following information:
   - User ID: `user1`
   - Password: `password1`

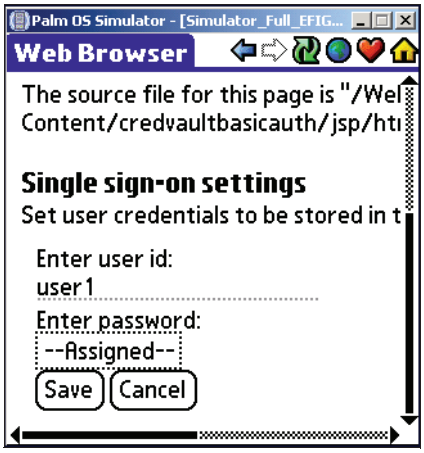


Figure 6-22 Entering credentials

12. Submit the action. This will generate an action that will be checked by the actionPerformed method in the CredVaultBasicAuthPortlet class. The portlet returns to View mode, showing the contents of the Treasure Servlet.

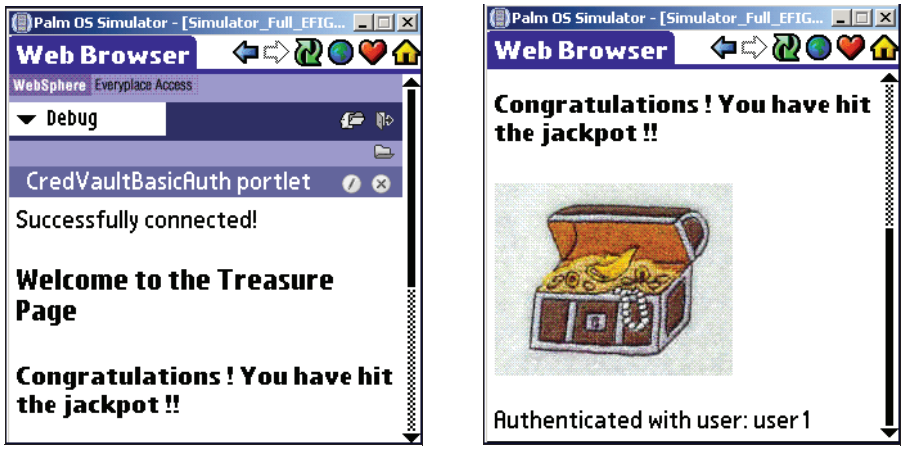


Figure 6-23 Credential portlet in action

13. If you do not enter the correct credentials, you will receive the following screen.

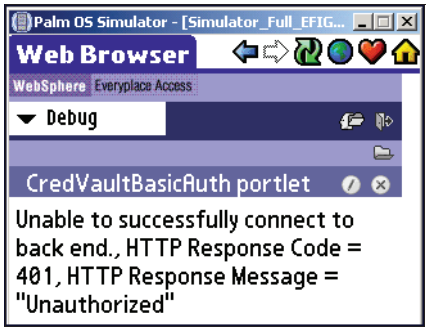


Figure 6-24 Incorrect credentials

6.2.5 Using passive credentials

In the previous section, a portlet using an active credential object was built. Although this is the preferred type of credential object, there are certain cases where you have to use passive credential objects, for example when an appropriate active credential class is not available.

In this sample scenario, the portlet will be changed to use a passive credential object.

To modify the portlet application, proceed as follows:

1. Open the class CredVaultBasicAuthPortletSecretManager from the credvaultbasicauth project.
2. In the Java editor, add the following method to the class.

Example 6-3 The getConnectionUsingPassiveObject method (passive credentials)

```
public static HttpURLConnection getConnectionUsingPassiveObject(
      PortletRequest portletRequest,
      CredVaultBasicAuthPortletSessionBean sessionBean,
      String host, String port, String path) {
      StringBuffer userid = new StringBuffer("");
      StringBuffer password = new StringBuffer("");
      HttpURLConnection connection = null;
      try {
         getCredential(portletRequest, sessionBean, userid, password);
         if (!userid.toString().equals("")) {
            String userAndPassword =
               new String(userid.toString() + ":" + password.toString());
            byte[] userAndPasswordBytes = userAndPassword.getBytes();
            BASE64Encoder encoder = new BASE64Encoder();
            String basicAuth =
               new String(encoder.encode(userAndPasswordBytes));
            basicAuth = "Basic " + basicAuth;
            URL url = new URL("http://" + host + ":" + port + path);
            connection = (HttpURLConnection) url.openConnection();
            connection.setRequestProperty("authorization", basicAuth);
         }
      } catch (Exception e) {
         e.printStackTrace();
      }
      return connection;
   }
```

3. Organize the import statements as you did in the previous scenario.
4. Save and close the Java file.
5. Open the class CredVaultBasicAuthPortlet from the credvaultbasicauth package.

   In the doView method, change the line:

   ```
   HttpURLConnection connection =
   CredVaultBasicAuthPortletSecretManager.getConnectionUsingActiveObject(reque
   st,   sessionBean,     host, port, path );
   ```

   to:

   ```
   HttpURLConnection connection =
   CredVaultBasicAuthPortletSecretManager.getConnectionUsing Passive Object(requ
   est,  sessionBean,    host, port, path );
   ```

6. Save and close the Java file.
7. Start the server and test the servlet as described in 6.2.4, “Running the portlet” on page 198. Now the portlet runs exactly the same way as it did in the previous section when it was using active credentials. This time, the getConnectionUsingPassiveObject method has access to the credentials.

Note: Having access to credentials could be a security risk, so when possible, use always active credential objects.

7

Accessing Web Services

This chapter provides an overview and a sample scenario for creating a sample portlet project that will work as a Web Service client to interact with a Web Service. The sample scenario in this chapter will allow you to understand the techniques used to develop portlets that retrieve information using Web Services.

Note: The sample scenario included in this chapter requires that Web Services be available. You can download the sample code available as additional materials. See Appendix B, “Additional material” on page 443.

© Copyright IBM Corp. 2005. All rights reserved.

7.1 Overview

In this section, we show the most important steps required to develop a Web Service client portlet accessing a local or remote Web Service. In most cases, the following tasks will be executed:

1. Develop a sample Web Service from a JavaBean. This Web Service will be used to test and run the Web Service client portlet. The following tasks are required:
   a. Create a sample Web project and import an existing JavaBean class.
   b. Using the available wizards in WebSphere Studio Site Developer, transform this JavaBean into a Web Service so it can be accessed from portlet applications.
2. Develop a Web Services client portlet to access the sample Web Service.

The scenario illustrated in Figure 7-1 shows how portlet applications can be easily integrated with existent Web Services without the need to write extra code.

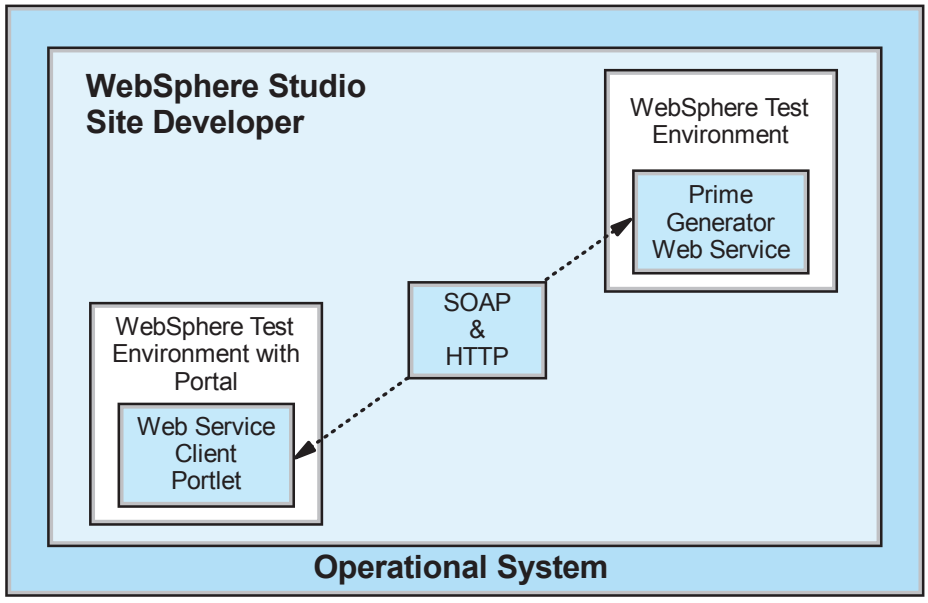


Figure 7-1 Web Services client portlet scenario

The development workstation used to create the sample application is illustrated in Figure 7-2 on page 209.

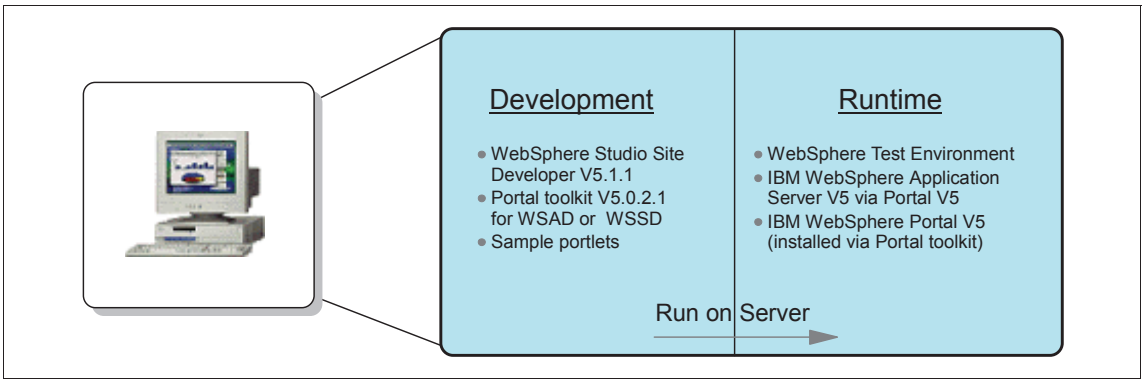


Figure 7-2 Development workstation

7.2 A simple Web Service project

The project will be created using the Dynamic Web Project wizard. In this section, you create a Web project with the name *PrimesWebService*. You will import a JavaBean that generates prime numbers into this project. The Web project will also be published and executed in a WebSphere Test Environment.

1. If not already running, start the IBM WebSphere Studio Site Developer; click **Start -> Programs -> IBM WebSphere Studio -> Site Developer 5.1.1**.
2. Select **File -> New -> Other**.

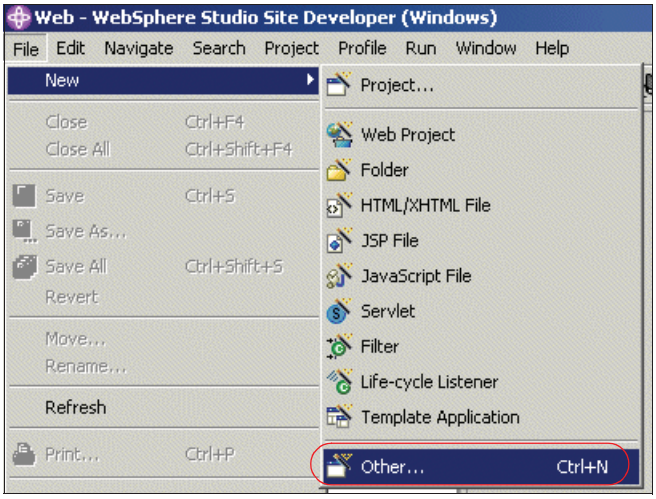


Figure 7-3 New project creation

3. Select **Web** -> **Dynamic Web Project** and click **Next**.

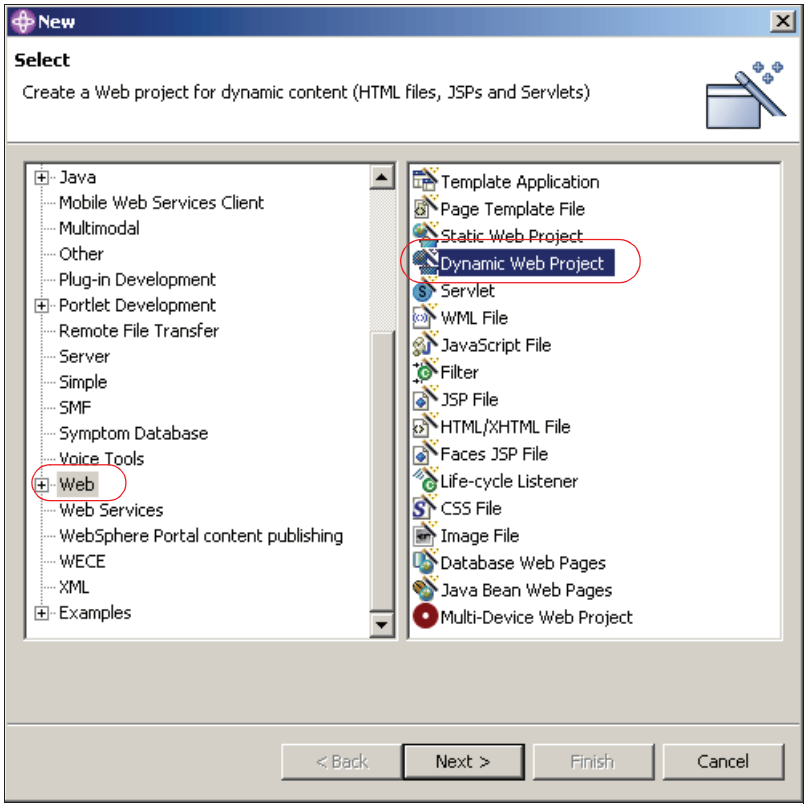


Figure 7-4 Selection of Dynamic Web Project

4. Enter `PrimesWebService` for the Project name, select **Configure advanced options** and click **Next**.

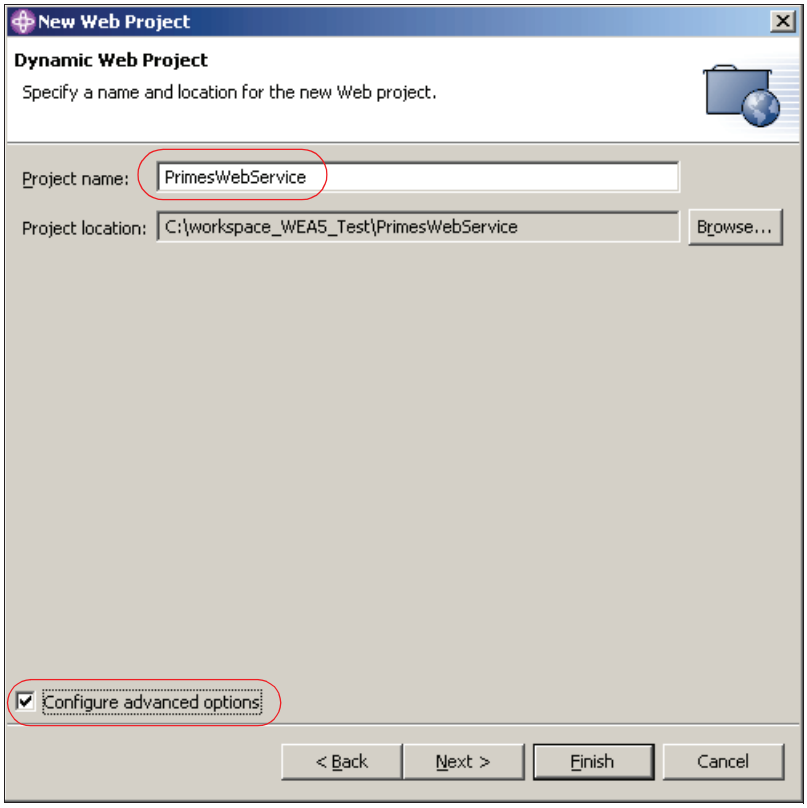


Figure 7-5 New Dynamic Web project wizard

5. Click **New** for the EAR project and enter `PrimesWebServiceEAR` (was DefaultEAR) in the New project name field. Click **Finish**.

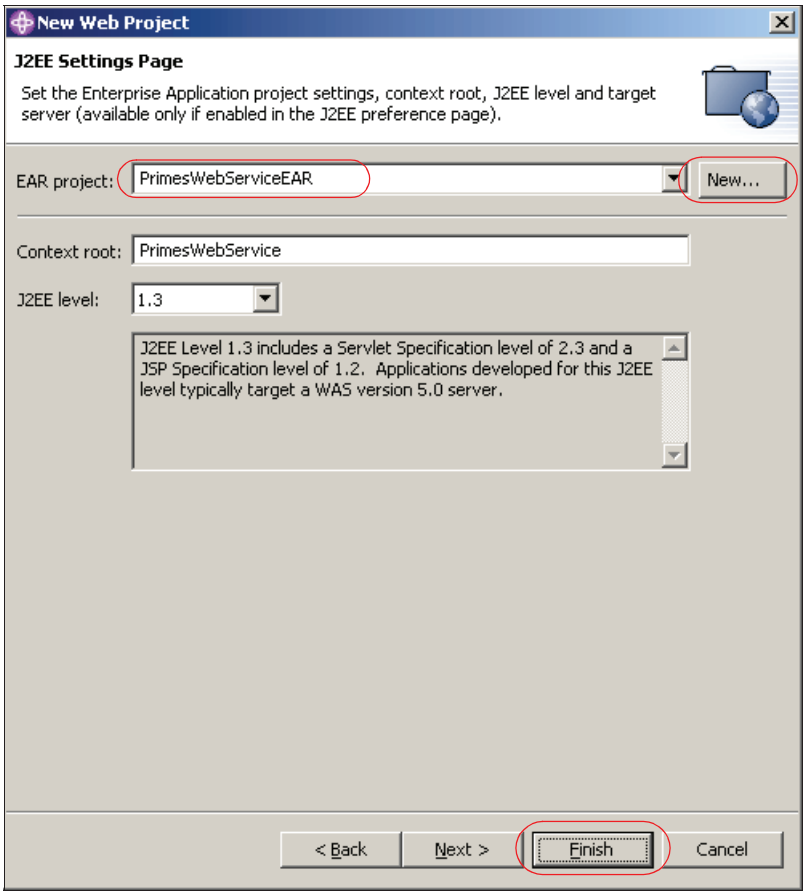


Figure 7-6 New Dynamic Web project wizard

6. A class file is provided for this sample scenario (Prime.java); follow these steps to import the Java file:

   a. Import the file by selecting **File -> Import**, select **File system** and click **Next**.

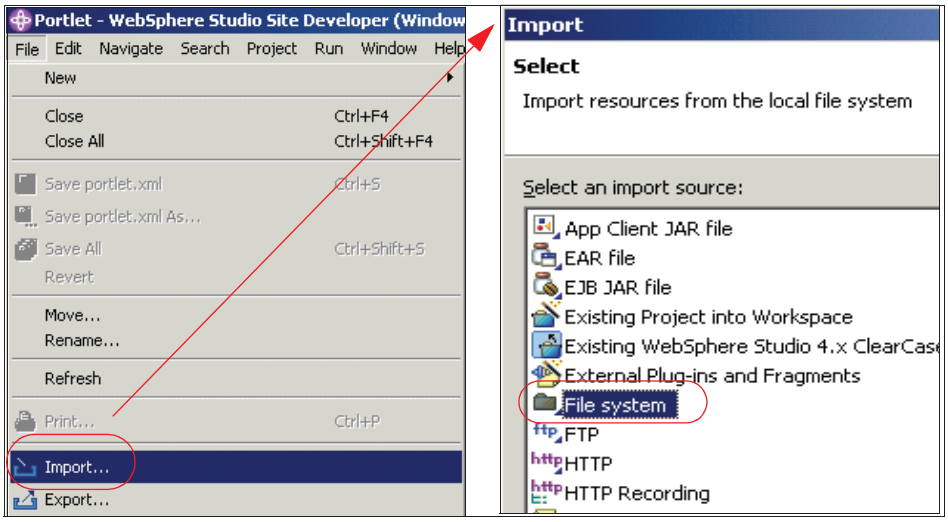


Figure 7-7 Importing a file

b. In the Import File system window, enter the following information:

   i. From Directory: browse to `C:\LabFiles\WebServicesClient\Java Source`, then select **Primes.java**.

      Note: You can also download the sample code available as additional materials. See Appendix B, “Additional material” on page 443.

   ii. Into folder: browse to the folder: `PrimesWebService/JavaSource/primes`.

   iii. Select **Create selected folders only** option.

   iv. Click **Finish**.

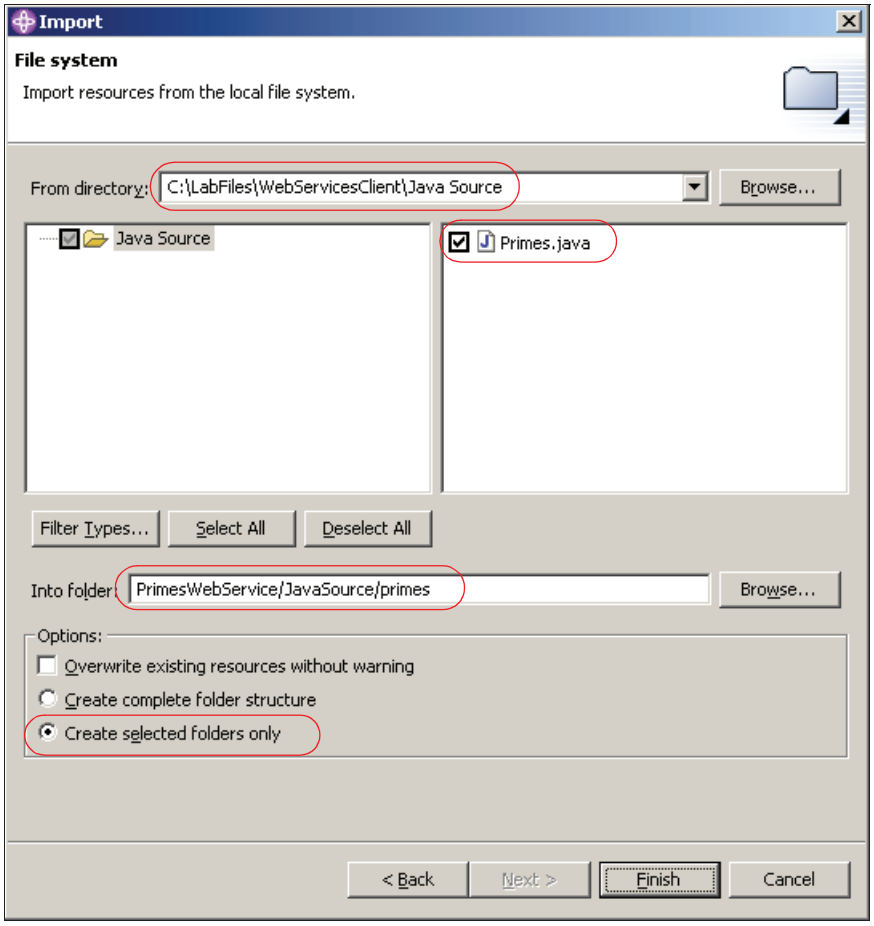


Figure 7-8 Importing the provided Java file

7. If a package error appears, include the following statement at the beginning of the Primes.java file, located in the /JavaSource/primes/ folder:

```
package primes;
```

7.2.1 A sample Web Service

In this section of the sample scenario, you will review the Primes.java code. For example, to view the source code to *Primes.java*, double-click the file name. This file is located in the /Java Source/primes/ folder.

- The only important method for you in this code is the getPrime() method. This method receive a number of digits and returns a prime number with the specified length of digits.

- When executing this method, try generating prime numbers of 20 or fewer digits to avoid long computations. A short version of the prime number generator is shown in Example 7-1.

Example 7-1 Primes.java - getPrime() method

```
package primes;

import java.math.BigInteger;
import java.util.Vector;


public class Primes {

   private static final BigInteger ZERO = new BigInteger("0");
   private static final BigInteger ONE = new BigInteger("1");
   private static final BigInteger TWO = new BigInteger("2");
   private String prime = "";

   ......
   ......
   ......

   public static String getPrime(int numDigits) {

      BigInteger start = random(numDigits);
      start = nextPrime(start);

      return start.toString();
   }
}
```

Creating a Web Service

In this section, you create a Web Service in the PrimesWebService project that you just created. You will use the wizards provided by WebSphere Studio Site Developer to do this. Once created, the Web Service will be published and invoked to be executed in a WebSphere Test Environment.

1. Select **File -> New -> Other**.

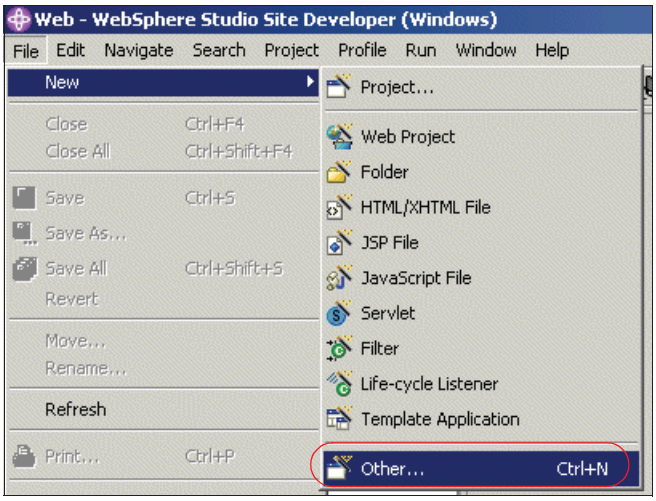


Figure 7-9 Starting creation of Portlet project

2. Select **Web Services** -> **Web Service** and click **Next**.

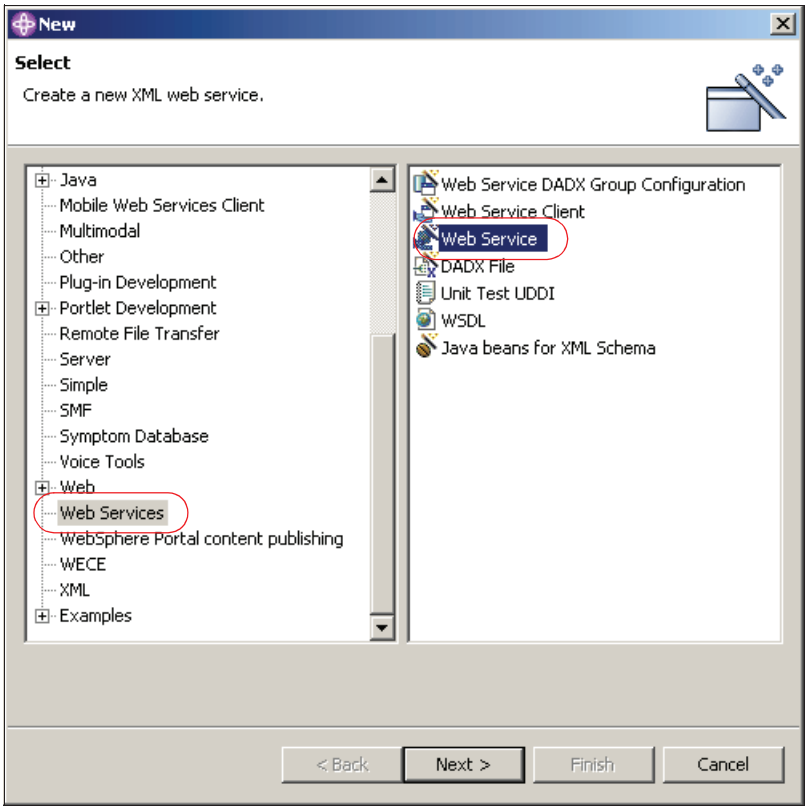


Figure 7-10 Selection of Web Service

3. Enter the following information in the Web Services window:
   a. Web Service type: `Java bean Web Service`.
   b. Check the box **Start Web service in Web project**.
   c. Check the box **Generate a proxy**.
   d. Client proxy type: `Java proxy`.
   e. Check the box **Test the generate proxy**.

   Click **Next**.

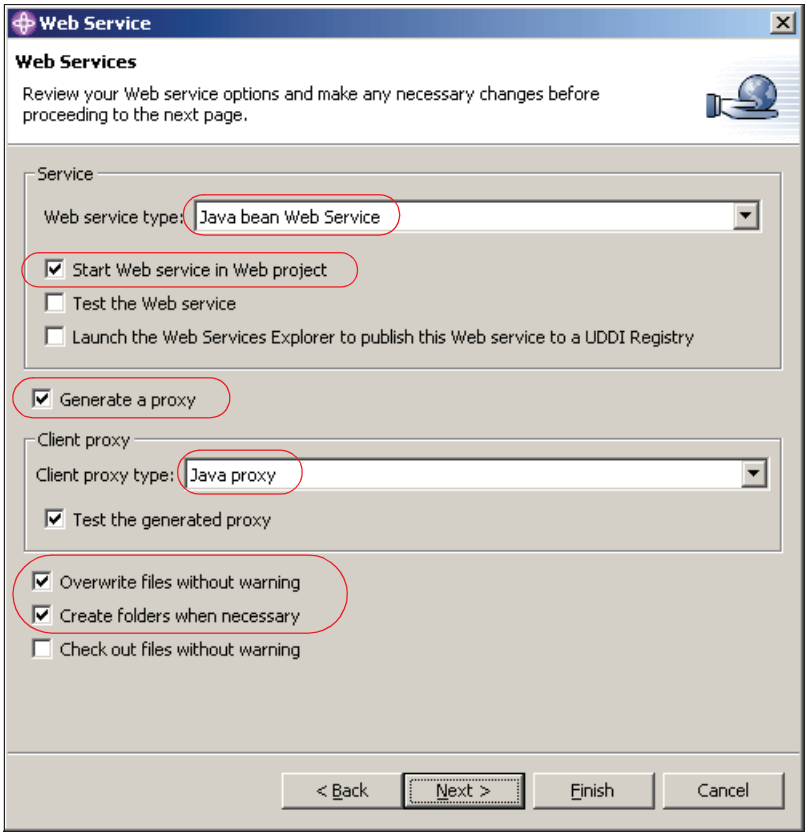


Figure 7-11 Web Service wizard

4. In Service Deployment Configuration, click Edit in Server-Side Deployment Selection section and select the following:
   - Server: Server Types → WebSphere version 5.0 → WebSphere v5.0 Test Environment
   - Web service runtime: IBM WebSphere V5.
   - Service Web Project: PrimesWebService.

   Click **OK**.

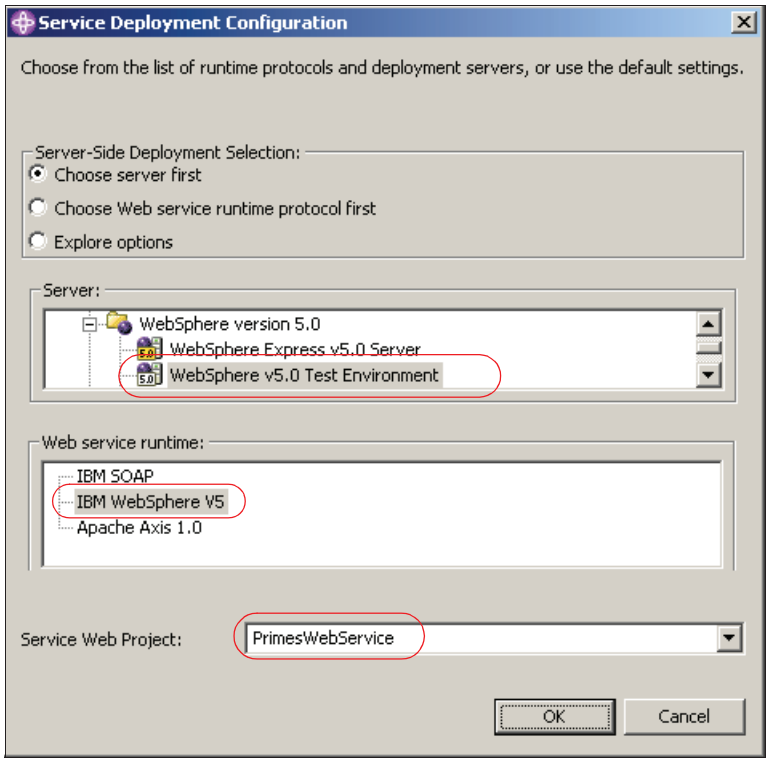


Figure 7-12 Web Service wizard

5. In Client-Side Environment Selection section, examine and accept default values. Click **Next**.

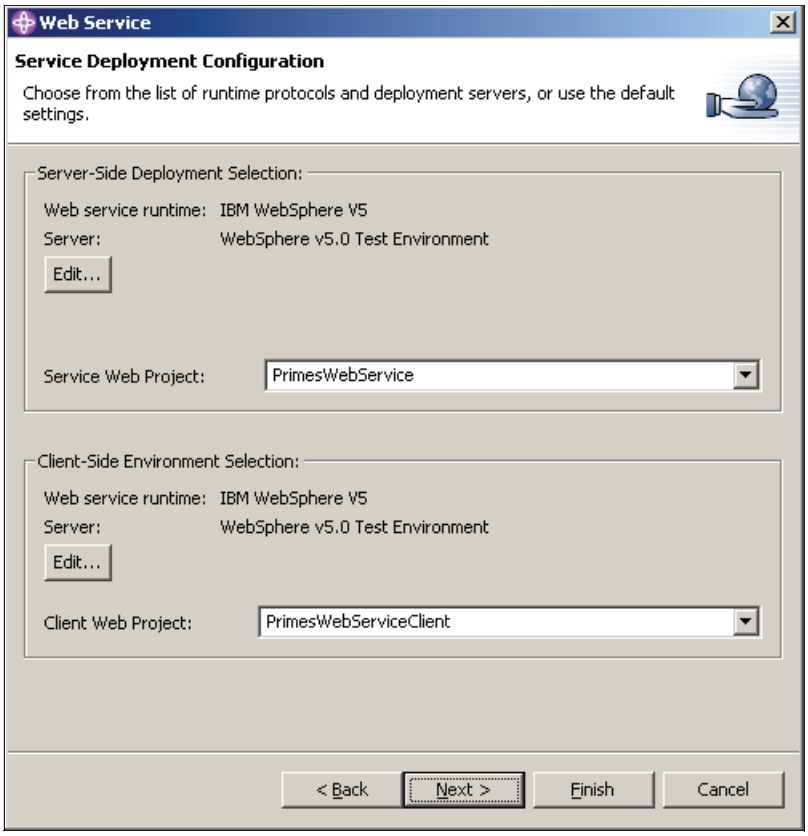


Figure 7-13 Web Service wizard

6. In the Web Service Java Bean Selection window, select primes.Primes class. Click **Next**.

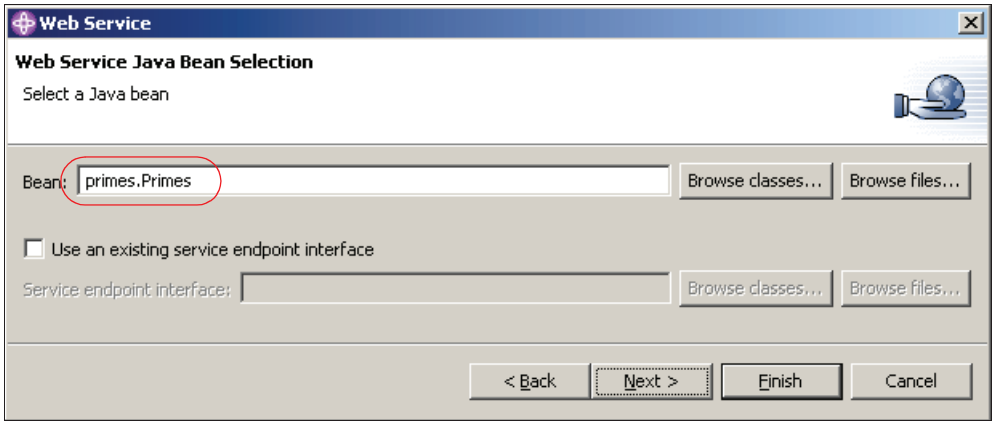


Figure 7-14 Web Service wizard

7. In the Web Service Java Bean Identity:
   a. Select the **getPrime(int)** method.
   b. Deselect other methods as shown in Figure 7-15 on page 222.
   c. Click **Next.**

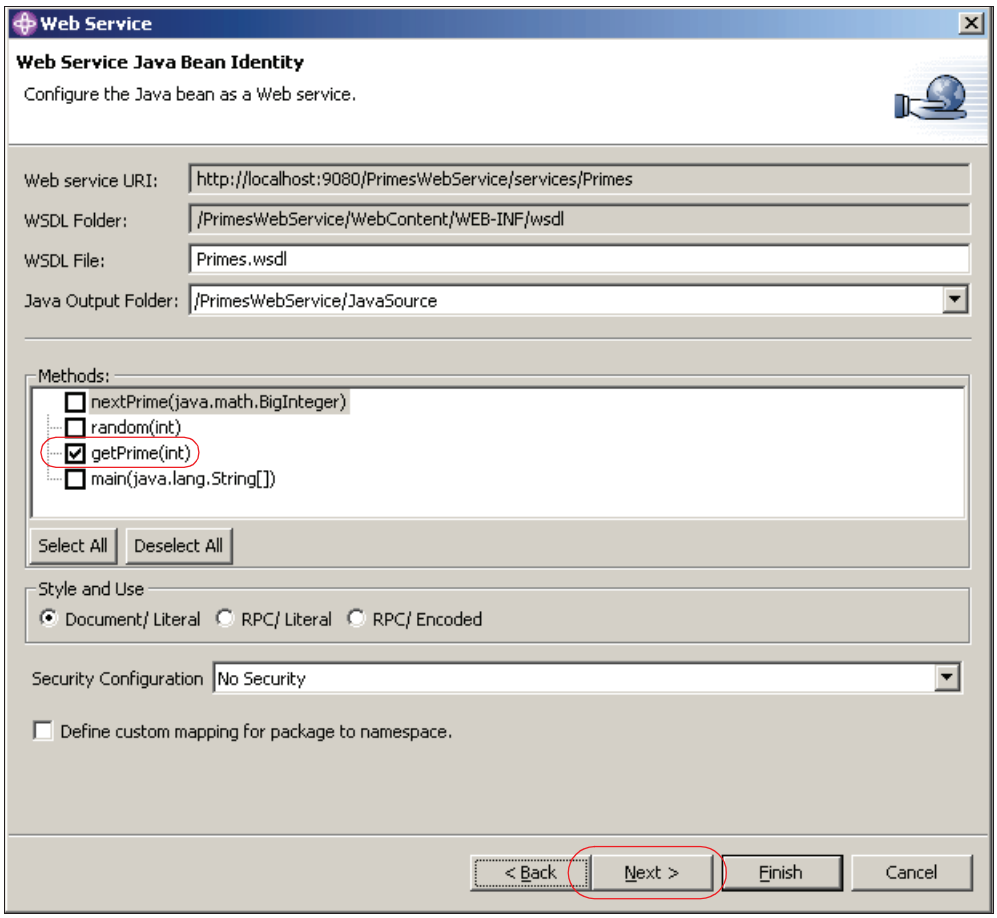


Figure 7-15 Web Service wizard

8. In the Web Service Proxy Page window, accept the defaults and click **Next**.

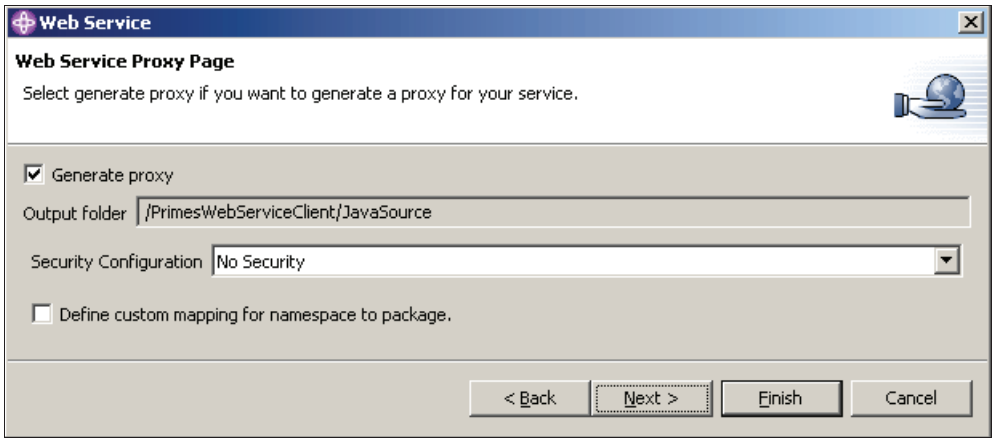


Figure 7-16 Web Service wizard

9. In the Web Service Client Test window:
   a. Select the **getPrime(int)** method.
   b. Deselect other methods.
   c. Select the **Run test on Server** option.
   d. Click **Next**.

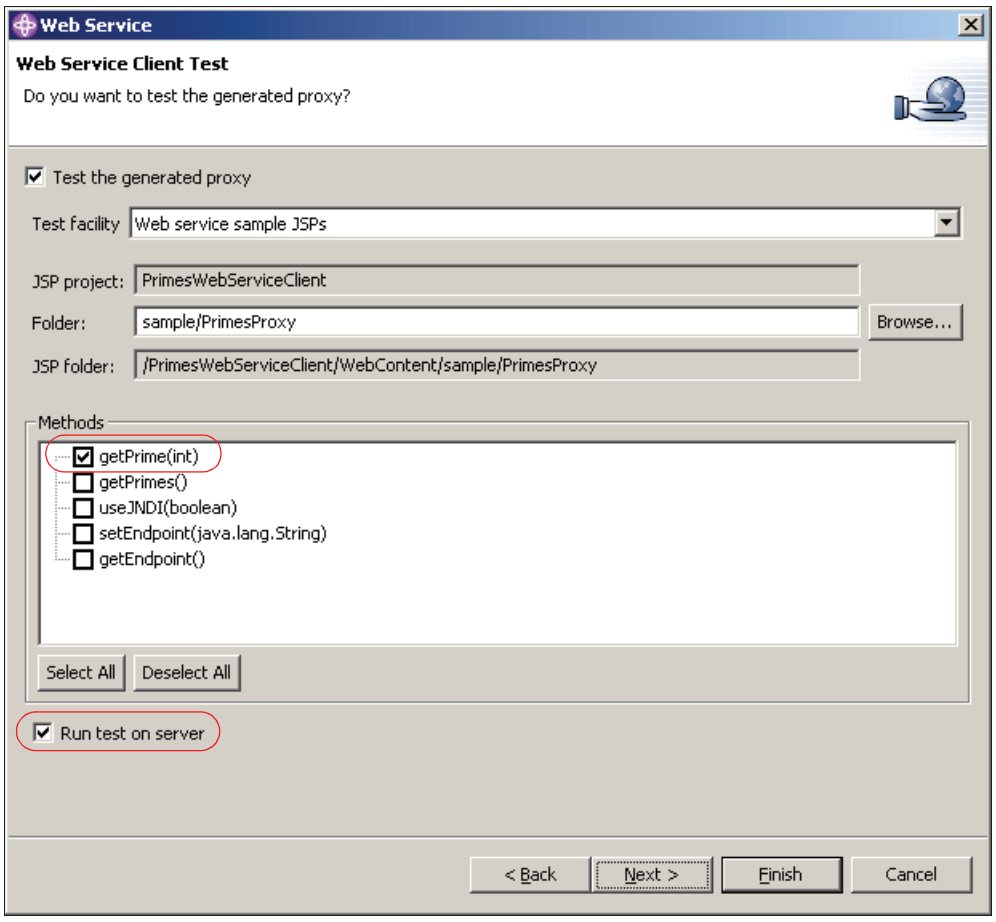


Figure 7-17 Web Service wizard

10. In the Web Server Publication window, accept the defaults and click **Finish** to publish the Web Service.

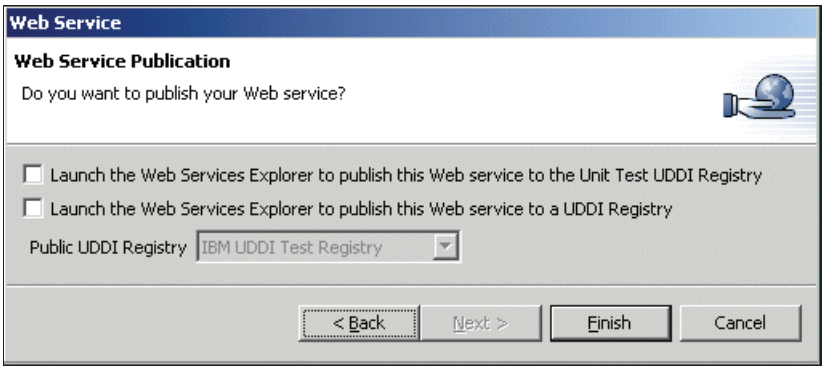


Figure 7-18 Web Service wizard

11. You will be presented with the available methods in the built-in browser. It is a Web test client interface to access and verify that the Web Service is running as expected.
12. Click the method **getPrime(int)**.

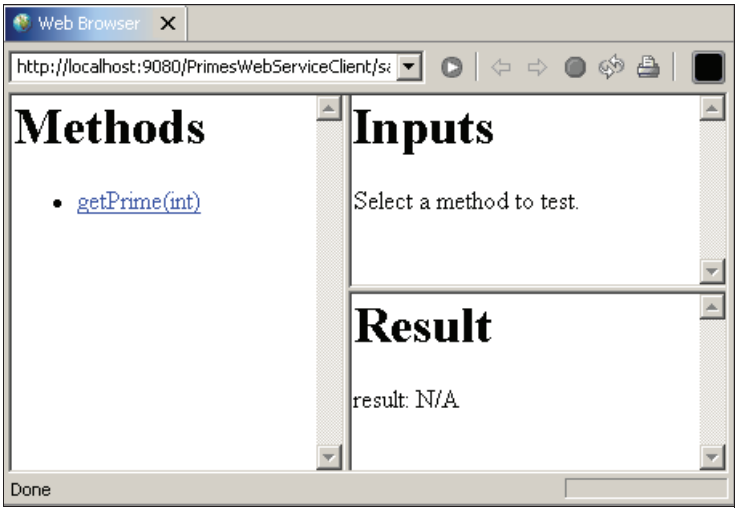


Figure 7-19 Available methods

13. Enter the number of digits you want the prime number to be.

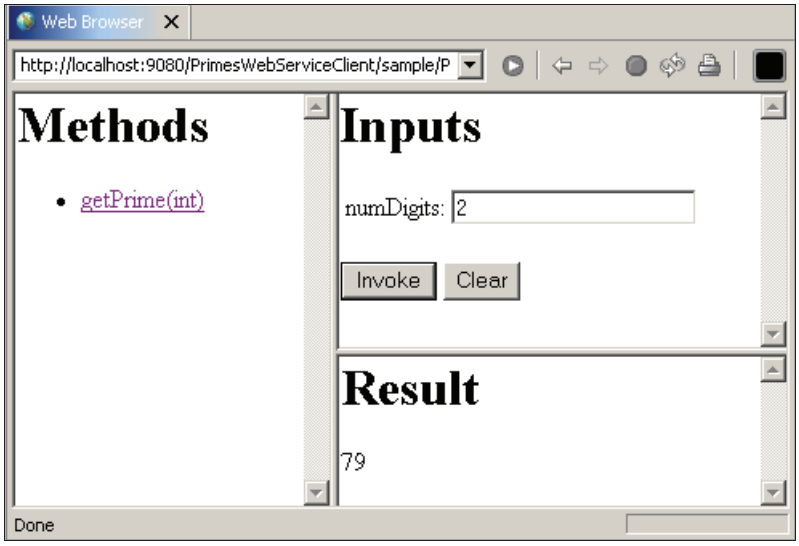


Figure 7-20 Accessing a Web Service to generate prime numbers

14. Every time you invoke this Web Service to generate a prime number, you will probably get a different result.

7.3 Creating a Web Services client portlet

In this section of the sample scenario, you will create a Web Services client portlet to access the prime number generator Web Service. This portlet uses the proxy programs generated in “Creating a Web Service” on page 215 to interact with the Web Service.

The sample Web Services client portlet consists of the following parts:

- JSP files (View)
  - Method.jsp: select available methods.
  - Input.jsp: insert input of selected method.
  - Result.jsp: shows the output of the method.
  - Configure.jsp, Edit.jsp and Help.jsp.
- Portlet file (Control)
  - MyPortlet.java: control client requests and the view components to present to the client.
- Bean file (Model)

– PrimesCommandBean.java: retrieves data from proxy.

To create a Web Services client portlet follow these steps:

1. If not already running, start the IBM WebSphere Studio Site Developer; click **Start -> Programs -> IBM WebSphere Studio -> Site Developer 5.1.1**.
2. Select **File -> New -> Other**.

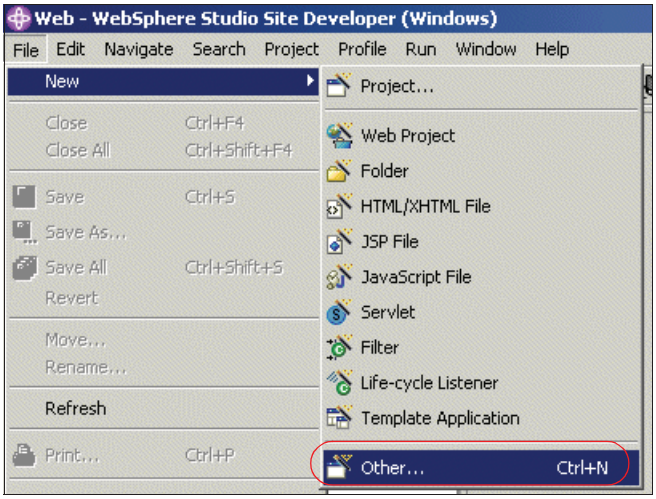


Figure 7-21 New project creation

3. Select **Portlet Development** -> **Portlet Application Project** and click **Next.**

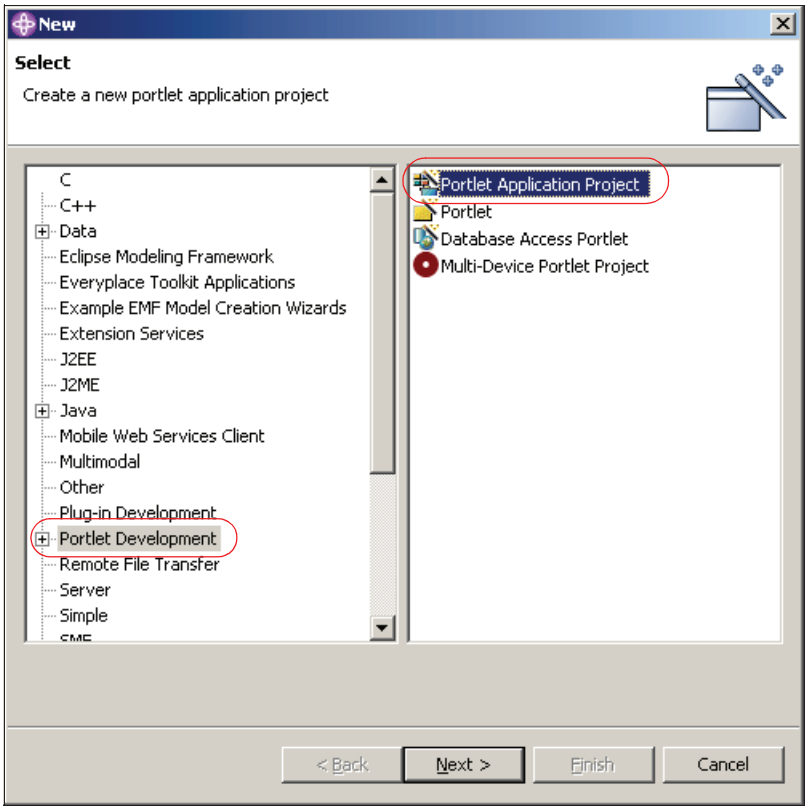


Figure 7-22 New Portlet Application Project

4. Enter the following information in the Portlet Project window:
   a. Project name: WSClientPortlet
   b. Check the box **Create basic portlet**.
   c. Check the box **Configure advanced options**.

   Click **Next**.

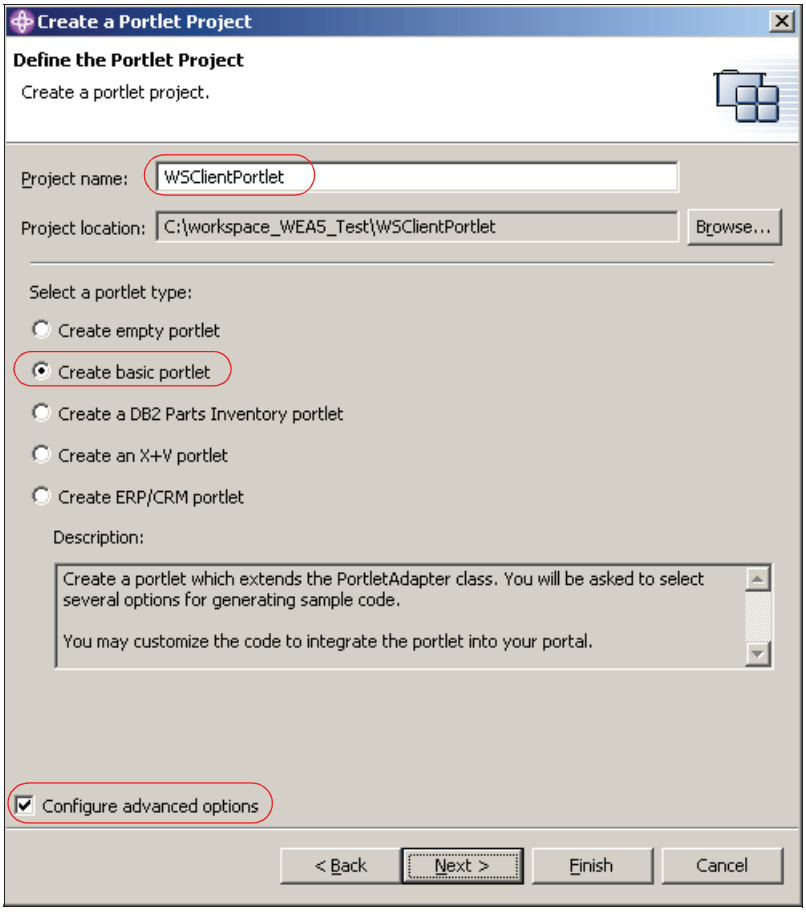


Figure 7-23 New Portlet project wizard

5. In J2EE Setting Page windows enter WSClientPortletEAR as EAR project and click **Next**.

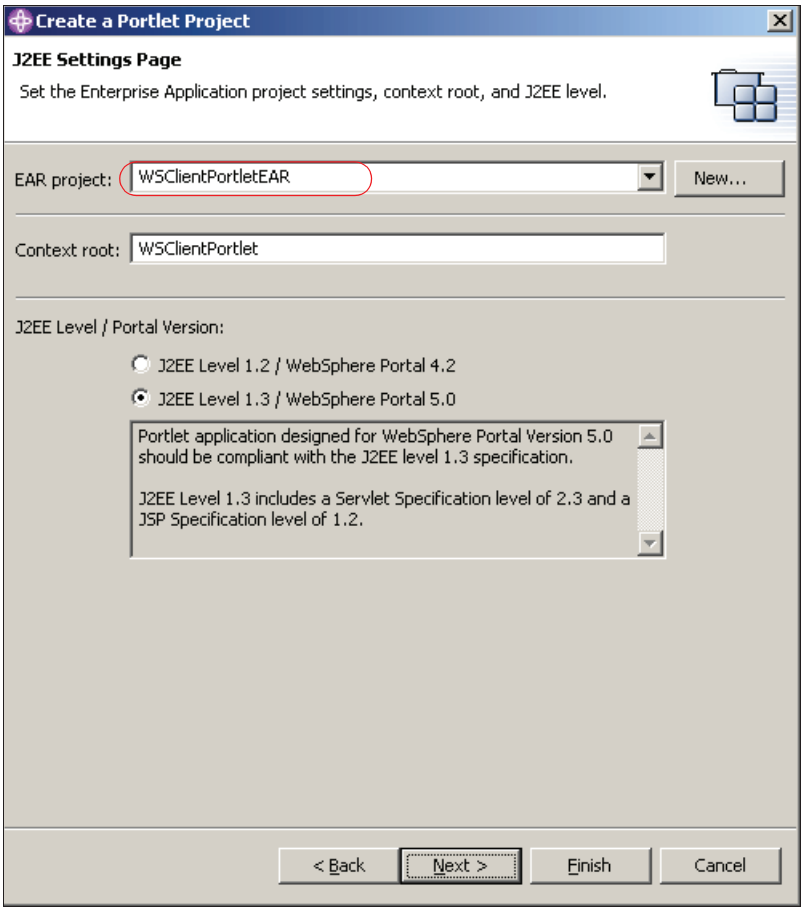


Figure 7-24 New Portlet project wizard

6. In the Portlet Settings window, accept the defaults and click **Next**.
7. In the Event Handling window, accept the defaults and click **Next**.
8. In the Single Sign-On window, accept the defaults and click **Next**.
9. In the Miscellaneous window, select **Add pda markup support** and click **Finish**.

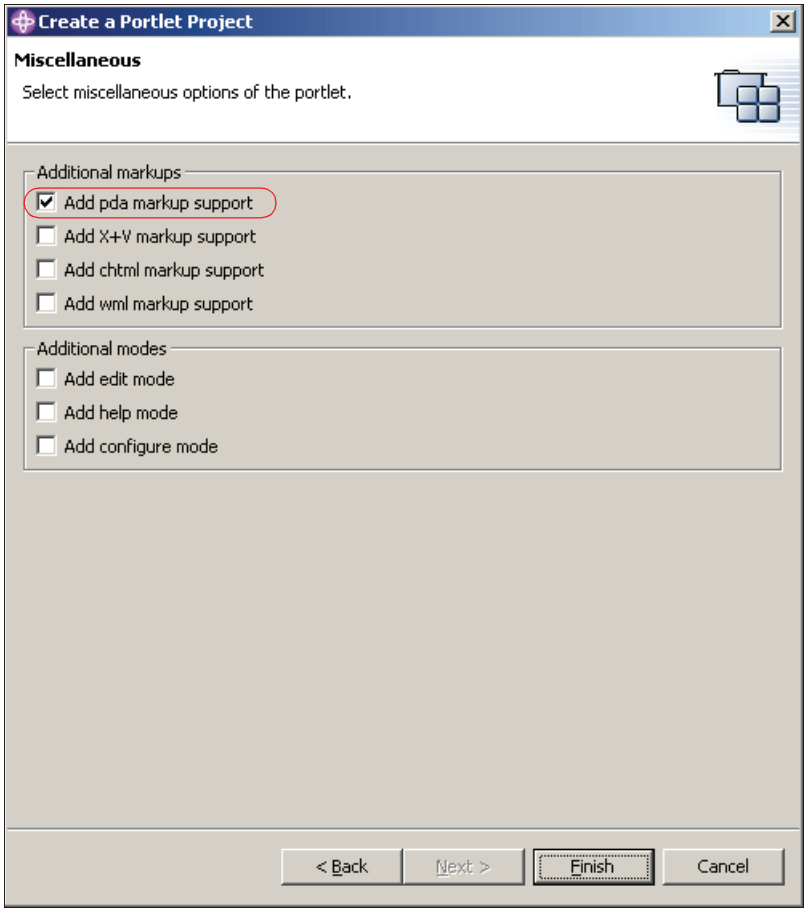


Figure 7-25 New Portlet project wizard

10. Include in the Java classpath of the WSClientPortlet project the JAR files commons-discovery.jar, commons-logging-api.jar, qname.jar, webservices.jar and wsdl4j.jar.
11. A WAR file is provided for this sample scenario (WSClientPortlet.war). Import this WAR file into the WSClientPortlet project.

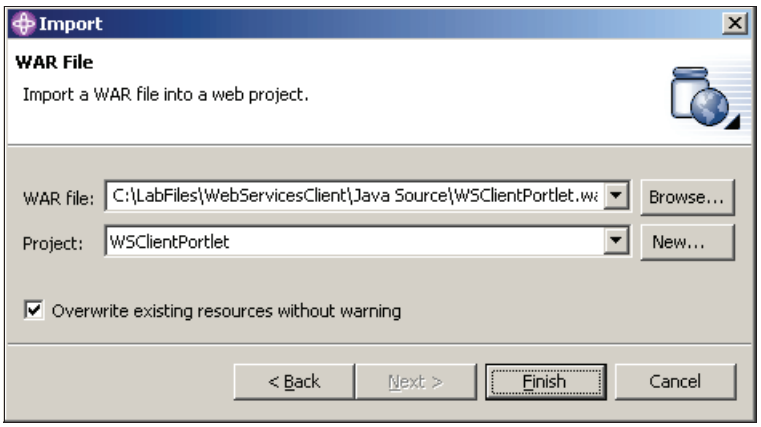


Figure 7-26 Importing the provided WAR file

7.4 Run the WSClientPortlet application

In this section, you will run the Web Services client (WSClientPortlet) to access the prime number generator Web Service. Execute the following steps:

1. Run the WSClientPortlet portlet application by right-clicking **WSClientPortlet** in the Navigator panel and selecting **Run on Server**.

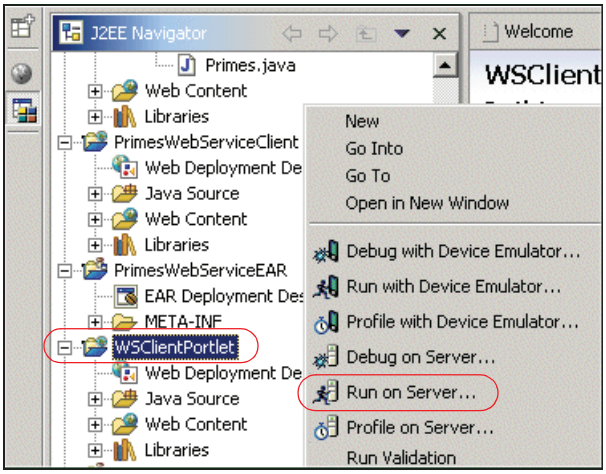


Figure 7-27 Running WSClientPortlet

2. Select the option to **Create a new server**.

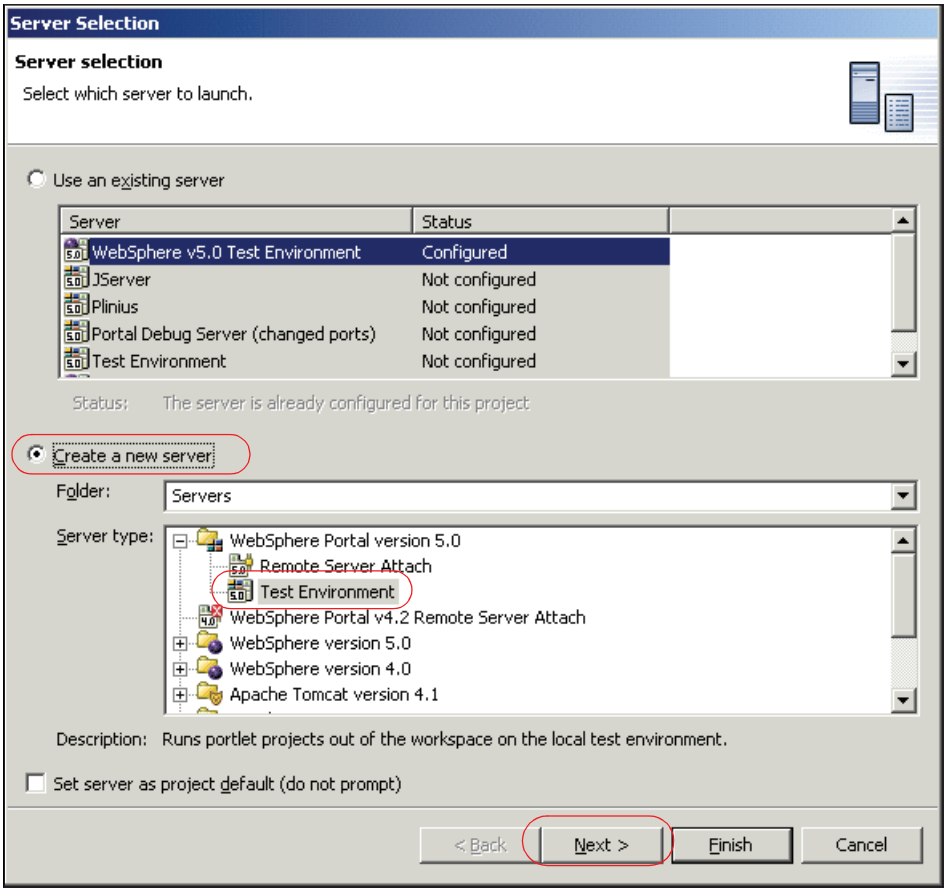


Figure 7-28 Server selection

3. Click **Finish** (Figure 7-28) and wait a few minutes for the server. The portlet will run and will be seen in the built-in browser. Click the method **getPrime(int)** to access the Web Service and generate a prime number.

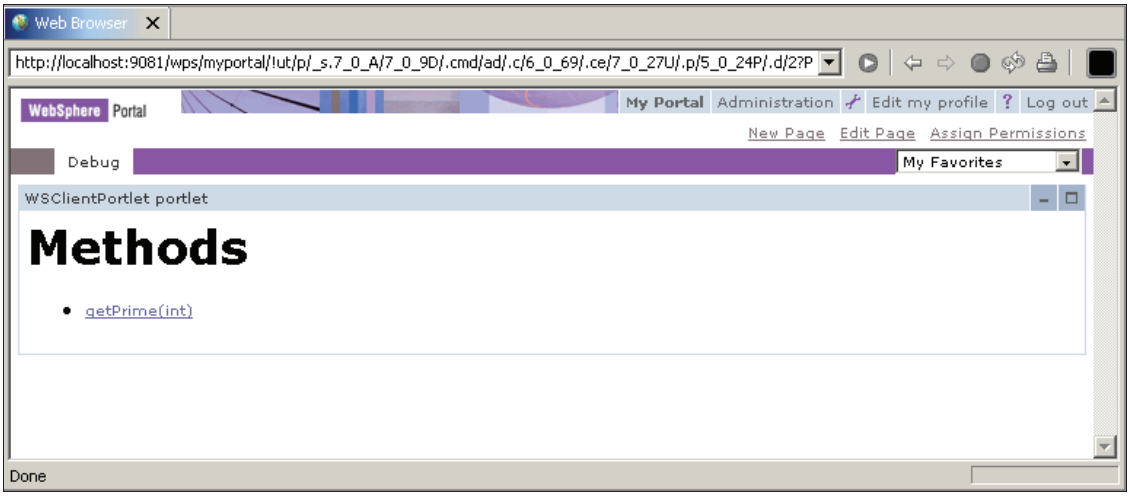


Figure 7-29 Available methods

4. Enter a number of digits, for example 2, and click **Invoke**.

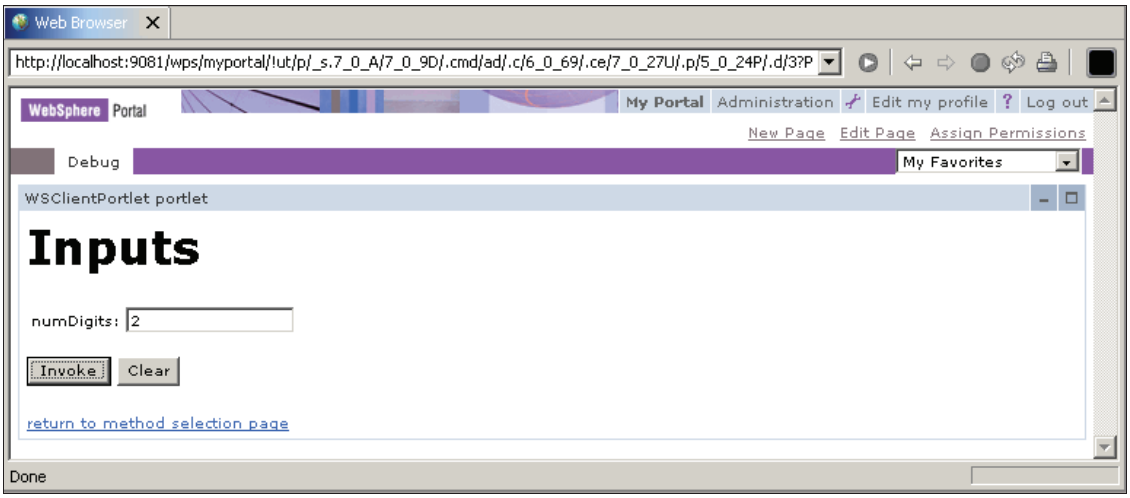


Figure 7-30 Invoking Web Services

5. The result from the Web Service will be shown.

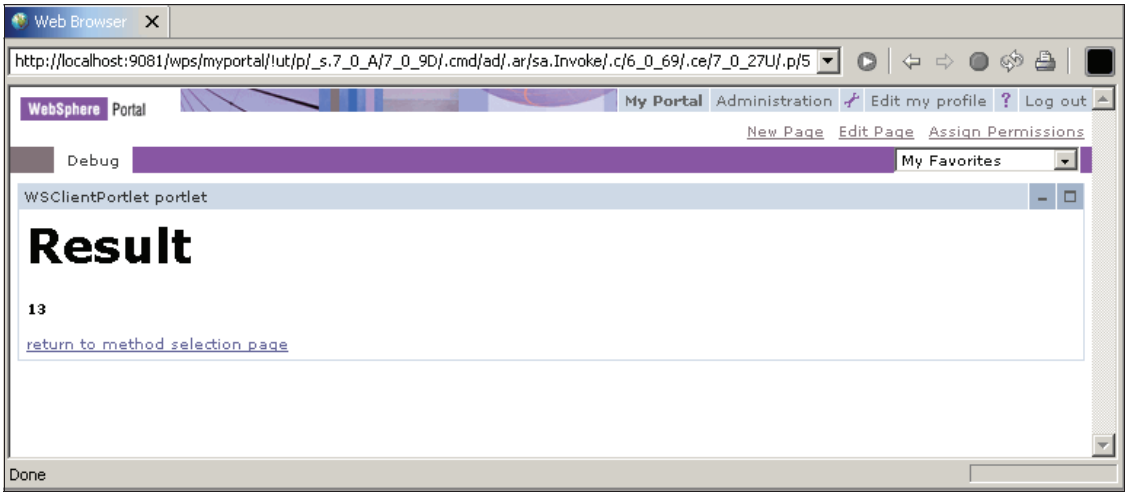


Figure 7-31 Results from the Web Service

Part 2

Transcoding

In this part of the redbook we introduce transcoding online applications in WebSphere Everyplace Access V5. Transcoding Technology, included in WebSphere Everyplace Access, provides the ability to tailor portlet applications to the needs of mobile users and their mobile devices.

© Copyright IBM Corp. 2005. All rights reserved.

8

Transcoding Technology

This chapter provides an introduction to Transcoding Technology, which is part of WebSphere Everyplace Access, and presents an overview of the relevant tools and transcoding options available.

© Copyright IBM Corp. 2005. All rights reserved.

8.1 Overview

Transcoding Technology, included in WebSphere Everyplace Access, provides the ability to tailor Web-based information to the needs of mobile users and their mobile devices. For example, Web pages targeted for the desktop browser can be adapted for viewing on mobile devices, such as PDAs.

With Transcoding Technology, users receive information tailored to the capabilities of the devices they are using. For example, users with small-screen devices access a re-scaled version of the information, while users with devices that require a specialized markup language can access the same information in the format suitable for their device. By providing a single dissemination point for multiple renderings of information, Transcoding Technology eliminates the expense of re-authoring or modifying applications to handle multiple networks and devices. In essence, Transcoding Technology extends the reach of existing information to a new class of users with mobile devices.

Transcoding Technology transforms content based on the information associated with the request, such as device constraints and administrative policies. Also, Web content can be transformed differently to meet the characteristics of the specific devices. Transcoding Technology can support all common types of Web data, including HTML pages and Extensible Markup Language (XML) documents. Transcoding Technology also tailors images by adjusting to screen size, file size, and the color pallet available in the target device.

Transcoding Technology offers three ways to transform content:

- XML stylesheets
- Annotators
- Transcoding plug-ins

These transformation details are also referred to as *resources* in Transcoding Technology. There is another type of resource called *preference profiles*, which are used to represent the characteristics of devices and users. Profiles can be used to determine which stylesheet, annotator, or plug-in to use in a given situation.

Note: In the following sections, the root directory where Transcoding Technology is installed is referred to as TT_Root. You should replace this with your actual directory name, for example:
C:\WebSphere\PortalServer\IBMTrans\

Transcoding Technology helps you ensure the Web data delivered to the devices is appropriate for the device and meets your administrative policy, which can consist of:

- Size: Images are scaled for presentation on the device and text is compressed, which could in effect reduce the size of the file that is transmitted to the device.
- Detail: Content tailored to the specific amount and kind of information your users need and the device can accommodate.
- Format: Content presented in the format supported by your users' device type(s).
- Language: Data is translated into a user's language.

8.1.1 Architecture

Transcoding Technology in WebSphere Everyplace Access is invoked at two different levels:

- Portlet-level transcoding. This is used for content transformations. For example, markup language conversions, stylesheets, and annotators are applied at this level.
- Portal page-level transcoding. This is used for whole-page modifications, for example content fragmentation.

Figure 8-1 on page 242 illustrates Transcoding Technology as implemented in IBM WebSphere Everyplace Access.

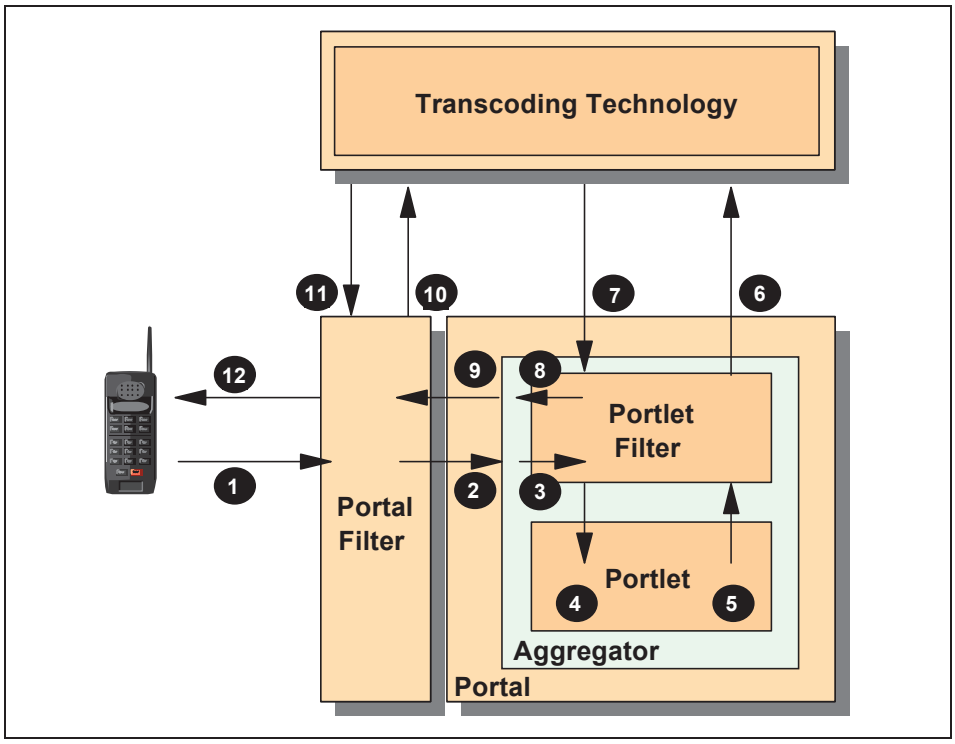


Figure 8-1 Transcoding Technology in WebSphere Everyplace Access

The request flow is explained as follows:

1. The user agent (WAP browser, Pocket PC browser, or desktop browser) makes a request to the portal. In this environment, the request is received by the Portal filter.
2. The Portal filter calls the portal to receive its contents, which calls the aggregator to aggregate the page.
3. The aggregator selects portlets based on a combination of values, such as user credentials and whether the portlet supports the current aggregator. If a portlet is configured for transcoding and it provides a markup that Transcoding Technology can use, it will be selected. For example, if the portlet provides HTML and the client needs WML content, the aggregator will select the portlet because the portlet content must be transformed from HTML to WML.

   Therefore, for a portlet that is configured to use Transcoding Technology, the aggregator calls the portlet filter.
4. The portlet filter calls the portlet to receive its contents.
5. The portlet returns its contents.

6. The portlet filter sends the portlet's contents to Transcoding Technology for processing, associating the portlet's contents with a default URI, unless otherwise specified. This step is known as *portlet-level transcoding*.

   Note that the portlet thinks it is communicating directly to the aggregator, so no special programming style is needed to write a portlet that is filtered.
7. Transcoding Technology returns the processed contents to the portlet filter.
8. The portlet filter returns the processed contents as though it were a proper portlet returning contents normally.
9. The aggregator, having finished aggregating the page, returns the page.
10. The Portal filter sends the Portal's aggregated contents to Transcoding Technology for processing. This step is known as *portal-level transcoding*.
11. Transcoding Technology returns the processed contents.
12. The Portal filter returns the processed contents to the target device.

8.1.2 Preference profiles

Transcoding Technology uses preference profiles to represent the characteristics of a particular device, a group of devices, a user, or user group. Preference profiles are used to decide how to treat documents that will be delivered to different devices and different users. For example, on a device with a small screen, it is desirable to convert tables to lists to reduce horizontal scrolling. A particular user may request that images are eliminated from the content viewed on their device.

When Transcoding Technology processes a document, a device profile and a user profile are selected to apply to that document.

Device preference profiles are represented by .prop files located in <TT_Root>\etc\preferences\device.

User preference profiles (a default profile is included) are represented as .prop files located in <TT_Root>\etc\preferences\user.

If the X-IBM-PVC-Device-Type field is present in the HTTP header, Transcoding Technology uses the device profile whose file name matches the value specified for that field. The value of the X-IBM-PVC-Device-Type field is set by the Portal's preference aggregation.

For example, if the value of X-IBM-PVC-Device-Type is Microsoft!Internet+Explorer!6.0, the <TT_Root>\etc\preferences\device\Microsoft!Internet+Explorer!6.0.prop device preference profile will be used. Figure 8-2 illustrates the properties file for Microsoft Internet Explorer Version 6.

```
#version = 1.0
#Sat Jun 07 08:49:18 EDT 2003
framesSupported=true
deviceRule=(User_Agent%e*MSIE 6.0*)
javaAppletsSupported=true
portalOrdinal=360
portalMarkupVersion=ie
createCHTML=false
portalClient=true
parent=NT.InternetExplorer
desiredContentTypes=[text/html]
javaScriptSupported=true
```

Figure 8-2 Microsoft!Internet+Explorer!6.0.prop

If the X-IBM-PVC-Device-Type field is not present in the HTTP header, Transcoding Technology uses the device profile whose user-agent value matches the value of the user-agent field in the HTTP header. If no matching profile is found, Transcoding Technology uses the default device profile.

User resources are not maintained through Transcoding Technology. However, other programs can pass information to Transcoding Technology specifying the use of a user profile. A user profile is selected for a request in the following way. If a value is specified for userAndSessionExtractor in <TT_Root>/etc/localConfig.prop, Transcoding Technology tries to execute the referenced implementation of the UserAndSessionExtractor interface to obtain user and session names and select the user profile that matches the user name.

You can specify a field in the HTTP header to be used to select a user profile by setting the httpUserIdField value in <TT_Root>/etc/localConfig.prop. For example, if you specify httpUserIdField=X-UserField, then Transcoding

Technology would find the value of the X-UserField keyword in the HTTP header and select the user profile that matches that value.

If the X-IBM-PVC-User field is present in the HTTP header, Transcoding Technology uses the user profile that matches the value specified for that field.

If none of these checks identify a user profile, then Transcoding Technology does not use a specific user profile. If one of these methods is used to specify a user profile and the specified file is not found, Transcoding Technology does not try the other methods; it does not use a specific user profile. For example, if you specify httpUserIdField=myUserField but myUserField is not specified in the HTTP header, Transcoding Technology would not look for the X-IBM-PVC-User field.

If more than one device preference profile matches the incoming request, it is impossible to predict which of the matching profiles would be selected. For example, if you create two device profiles with expressions that could match the same user-agent value, either one could be selected for a request that specifies that user-agent value. Be sure to specify unique values for any new profiles you create.

Each profile contains values for the preferences that are important to the device that the profile represents. If a preference is not important to the device, it can be omitted so that a value can be chosen from a different profile. For most preferences, Transcoding Technology will check profiles for a value in this order:

1. Specific user
2. Specific device
3. Default user
4. Default device

If a value is not specified for a preference in one profile, Transcoding Technology will work down the list until a value is found. The transcoding plug-ins that will be applied to the document are selected based on the combined profiles.

Transcoding Technology provide profiles for several common pervasive devices. There are default profiles to be used if none of the existing profiles matches the device being used.

In addition to determining which transcoding plug-ins will be used, a profile can be used to select a stylesheet or annotator. You can also specify parameters to be used by stylesheets that accept parameters.

8.1.3 XML stylesheets

When Transcoding Technology process documents represented in Extensible Markup Language (XML), it uses XSL stylesheets to convert these documents to another markup language, such as HTML, WML, or other forms of XML.

Stylesheets are associated with the portlet emitting the XML by either adding a config-param parameter to the portlets portlet.xml file or by adding parameters to the portlet by using Portal Administrations Manage Portlet (modify parameters).

Multiple stylesheets can be associated with an XML document, by including the wtp-config processing instruction within the target XML document. One of the stylesheets is selected to process against the XML document based on the values of one or more fields in the HTTP header or the DTD associated with the XML document. A stylesheet can also be selected based on criteria that match a preference profile.

Some stylesheets accept parameters that affect how they operate on documents. A stylesheet that accepts parameters can retrieve values for the parameters from:

- The HTTP header of the request
- Values specified for the Parameters field in the WTPResources file
- Values specified in a preference profile

Stylesheets are used in Chapter 9, “A sample scenario using Transcoding Technology” on page 265.

8.1.4 Annotators

For pervasive devices with limited screen size, it is often desirable to show only the key information from a Web page, and get rid of the less critical information such as graphics and extra content. Annotators can be used to specify which portions of a Web page to include or discard when the Web page is transcoded.

There are two types of annotators:

- Internal annotators

  These are annotation instructions that are created using the Page Designer tool available in WebSphere Studio Application Developer and WebSphere Studio Site Developer. Internal annotation instructions consist of special tags embedded in the Web page. When the page passes through Transcoding Technology, these tags are examined and actions are taken according to the particular instruction(s). For details about and samples of internal annotators, see Chapter 9, “A sample scenario using Transcoding Technology” on page 265.

- External annotators

  These annotation instructions are contained in a separate and independent annotation file that the Transcoding Technology uses to operate on the target Web content. External annotation files can be created with simple text editors, or by using the HTML Annotation Editor supplied with the Everyplace Toolkit (a plug-in for WebSphere Studio).

  External annotation files can be packaged with a portlet .war file. The portlet.xml file can contain the configuration parameters to support annotation. For more details about external annotation, see Chapter 9, “A sample scenario using Transcoding Technology” on page 265.

8.1.5 Transcoding plug-ins

A transcoding plug-in is a program written with programming languages that modifies the content of a document. Transcoding plug-ins are selected to process a document based on conditions specified by the program when the transcoding plug-in is created.

Several transcoding plug-ins are provided with Transcoding Technology in WebSphere Everyplace Access, and you can obtain or develop others:

- The image transcoding plug-in modifies images to better support the display capability of a device.
- The text transcoding plug-in converts textual data, such as HTML or XML, from one format to another and can perform a number of transformations to simplify the output.
- The fragmentation transcoding plug-in fragments XML documents into pieces small enough to be managed by the target device.
- The HTML DOM generator creates a Document Object Model (DOM) version of incoming HTML documents.
- The annotation transcoding plug-in, also called annotation engine, interprets the contents of files written with Transcoding Technology annotation language to perform document clipping.
- The HTML to WML transcoding plug-in converts HTML documents to WML for devices with WAP browsers.
- The HTML to compact HTML transcoding plug-in converts HTML documents to Compact HTML documents for devices with CHTML browsers.

These transcoding plug-ins are installed with the product and are enabled by default (with the exception of the HTML DOM generator). To verify whether the transcoding plug-ins are enabled or to change their status, use the XML configuration utility to export the configuration for reviewing.

8.2 XML configuration utility

The Transcoding Administrative Console (part of the WebSphere Transcoding product) is not available with Everyplace Access. Transcoding Technology provides the XML configuration utility, which provides a set of commands (entered on the command line) that perform the following tasks:

- Export resources: Used to export the resource definitions to an XML document from transcoding resources
- Import resources: Used to import the resource definitions from an XML document into the transcoding resources
- Disable resources: Disables a resource that is currently active in the transcoding resources
- Enable resources: Enables a resource that is currently part of the transcoding resources
- ImportCocoon: Imports a Cocoon properties file to support stylesheet specification within XML documents

Note: Resources such as device profiles, stylesheet registrations, and external annotator registrations can be configured using this utility.

The configuration file can be modified outside of Transcoding Technology, which allows adjustments to the configuration as needed. Figure 8-3 on page 249 shows the XML configuration management consisting of the following steps:

1. Use the ExportResources command to read the existing Transcoding Technology configuration.
2. A Transcoding Technology XML configuration file (WTPResources.xml) is created in the specified directory.
3. The WTPResources.xml file can be modified as needed.
4. Use the ImportResources command to write the modified WTPResources.xml into Transcoding Technology.

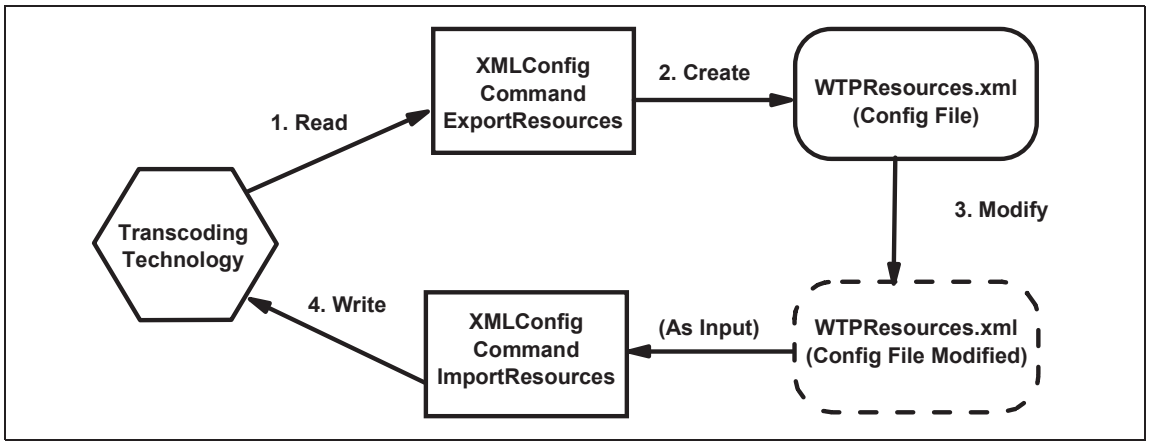


Figure 8-3 XML Configuration management steps

For an example of how to use these commands, see “Preparation steps” on page 252.

8.3 Request Viewer

Request Viewer is a very useful visual tool for monitoring the traffic going through Transcoding Technology. The Request Viewer is included in the Everyplace Toolkit for use in debugging your annotation files and styesheets. You can use Request Viewer to view the configuration and status information of the registered transcoding plug-ins within Transcoding Technology.

Request Viewer is particularly handy as a debugging tool, because it enables you to examine the flow of requests and responses through the server and observe which plug-ins are triggered and when they are triggered. For each transaction, Request Viewer also displays the header and content information as they are manipulated by the transcoding plug-ins.

The primary window of Request Viewer consists of these parts:

1. Menu bar: Provides access to the application options for Files, Actions, Requests and Help. Menu options can be disabled when the action is not available.
2. View options: Two tabs, each of which specifies a particular view selection. One is for the Server Configuration and the other is the Request Processing view. When either of these views is selected, the associated content is displayed in the main window. The Server Configuration view provides the transcoding server information, including the transcoders registered,

preference profiles, and so on. The Request Processing view shows the flow of the request through the server.

3. The Server Configuration view (shown in Figure 8-4) subdivides the window into three parts:

   a. Server Configuration: A list view of the contents of this configuration (including sublayers and transcoding plug-ins). This is found in the top-left pane.

   b. Details: Displays the details of an item within the configuration. Content is displayed in this pane when an item is selected in the Server Configuration tree view. This is found in the top-right pane, across from the Server Configuration.

   c. Output Messages: Displays messages generated at execution time. This is found across the bottom of the window, below the Server Config and Details panes.

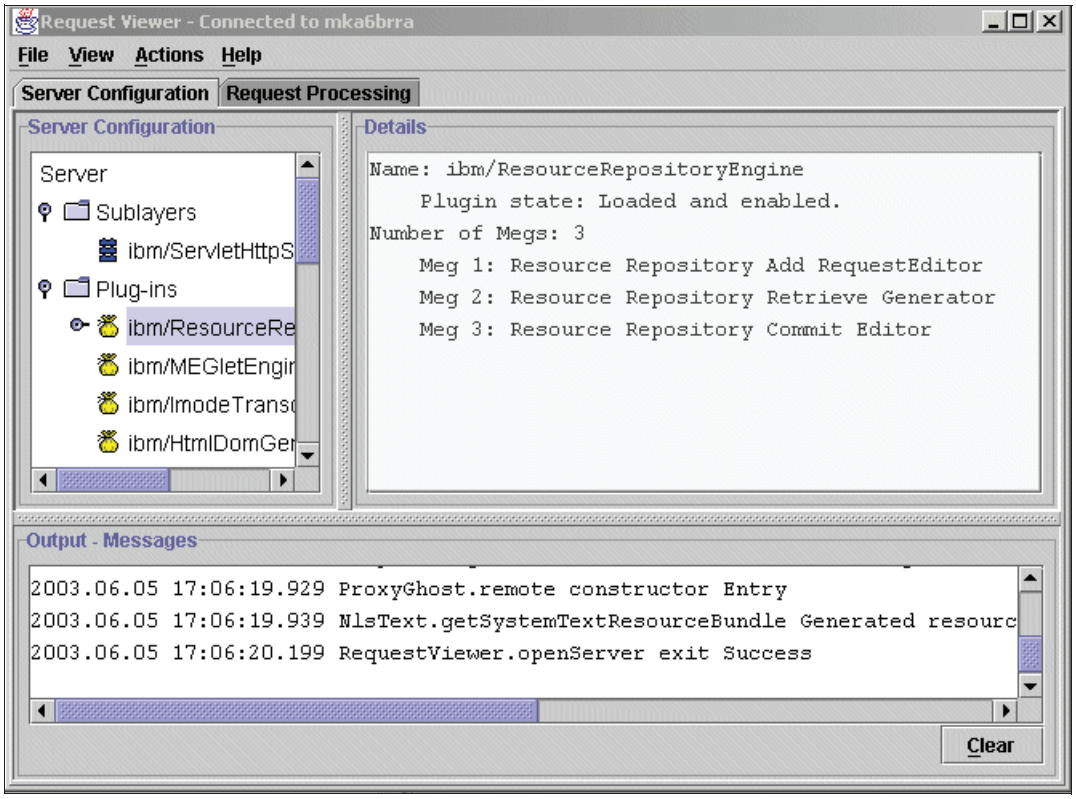


Figure 8-4 Request Viewer server configuration

4. The Request Processing view (shown in Figure 8-5) subdivides the window into three parts:
   a. Request Processing: A tree view of the flow of requests through Transcoding Technology (in the top-left pane).
   b. Transaction Header: Displays the contents of the HTTP header. The content is displayed in this pane when an item is selected in the Request Processing tree view. This is found in the top-right pane, across from Request processing.
   c. Transaction Content: Displays a particular transcoder's input or output (below the Transaction Header).

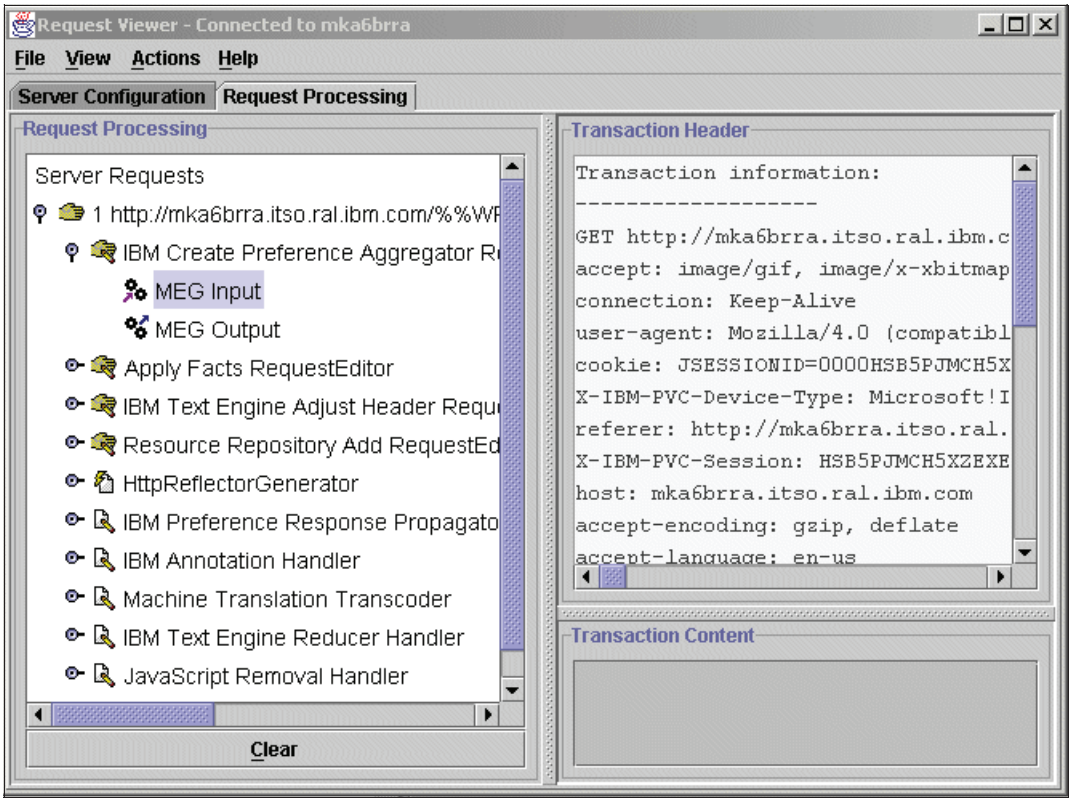


Figure 8-5 Request Viewer request processing

For more details on the Request Viewer, see the *WebSphere Transcoding Publisher Developer Guide V4.0* found at:

`http://www.ibm.com/software/webservers/transcoding/library.html`

8.3.1 How to start Request Viewer

The Request Viewer is now included in the Everyplace Toolkit, which is a plug-in for the WebSphere Studio offering (currently WebSphere Studio Site Developer and WebSphere Studio Application Developer products). We are using WebSphere Studio Site Developer with the Everyplace Toolkit installed.

Preparation steps

Prior to starting Request Viewer in WebSphere Studio, you must perform these steps on the Everyplace Access server:

1. Use the Services to stop the IBM WebSphere Application Server and the IBM Everyplace Sync Server.
2. Start a DOS prompt and change the working directory to the directory where Transcoding Technology is installed. In our case it was installed in the default directory of C:\WebSphere\PortalServer\IBMTrans.
3. You must change the transcoding configuration to enable RMI registry and enable two of the transcoding plug-ins, DatabaseWatcherPlugin and ViewerServerPlugin. To do this, perform the remaining steps.
4. Run the ExportResources command to export the resources that must be modified. The specific command is:

   ```
   ExportResources -rt setup -rt
   plugin,DatabaseWatcherPlugin,ViewerServerPlugin
   ```

5. A WTPResources.xml file is placed in the transcoding directory (IBMTrans). Open this file for editing and make the following changes:
   a. For both the DatabaseWatcherPlugin and the ViewerServerPlugin plug-ins (which are defined as <Plugin> elements) add the statement shown in Example 8-1 before the </Plugin> tag, to enable the plug-ins.

Example 8-1 Enable the plug-ins

```
<Plugin>
.............
<Enable>true</Enable>
</Plugin>
```

   b. Enable the RMI Registry, which is the <RMIRegistry> element, as shown in Example 8-2.

Example 8-2 Enable RMI Registry

```
<RMIRegistry>
.....
<Enable>true</Enable>
```

```
</RMIRegistry>
```

c. Enable the Remote Request Viewer and authorize the system where WebSphere Studio is running by adding the statements shown in Example 8-3 to the <RemoteRequestViewer> element.

Example 8-3 Enable and authorize RemoteRequestViewer

```
<RemoteRequestViewer>
.....
<Enable>true</Enable>
<AuthorizedHostName>MachinewithWebSphereStudio.myco.com</AuthorizedHostName>
</RemoteRequestViewer>
```

d. Save your changes and close the file.

6. Copy the ViewerServerPlugin.jar from the plug-ins folder (in the IBMTrans directory) as ibm_ViewerServerPlugin.jar to the addedPluginJars folder in the IBMTrans directory.
7. Import the changed WTPResources.xml file back into Transcoding Technology by using the ImportResources command.
8. Open the wtpscripts.jar and extract the StartRegistry.bat script into the IBMTrans directory.
9. Always start the RMI registry before starting the WebSphere Application Server or the Everyplace Sync Server. This is done by running the StartRegistry command.
10. Now you can restart the IBM WebSphere Application Server and the IBM Everyplace Sync Server.

Starting Request Viewer

With WebSphere Studio started and an existing portlet project active, you can launch the Request Viewer by selecting **File -> Transcoding Tools -> Launch Request Viewer** as shown in Figure 8-6 on page 254.

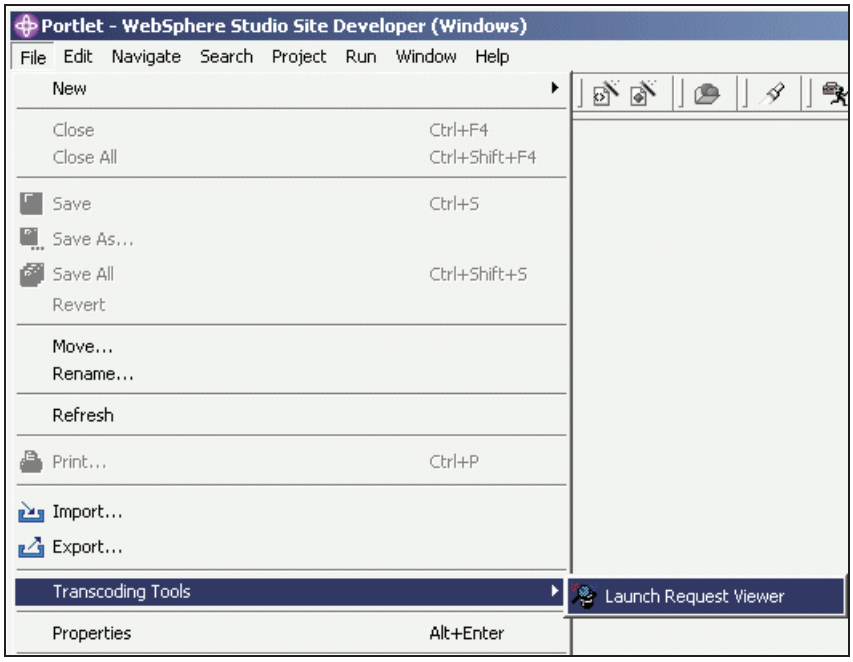


Figure 8-6 Launch Request Viewer

Once the Request Viewer starts, a window pops up asking for information about the machine where Transcoding Technology is running. We are running remote to the Everyplace Access server. Select the **Remote** radio button and in the Host name field enter the remote machine name. Next, select the **Defined in a remote RMI registry** radio button. Both the host name of RMI server and the port number of the RMI server are supplied for you. This window and our entries are shown in Figure 8-7 on page 255. Click **OK** to finish the process.

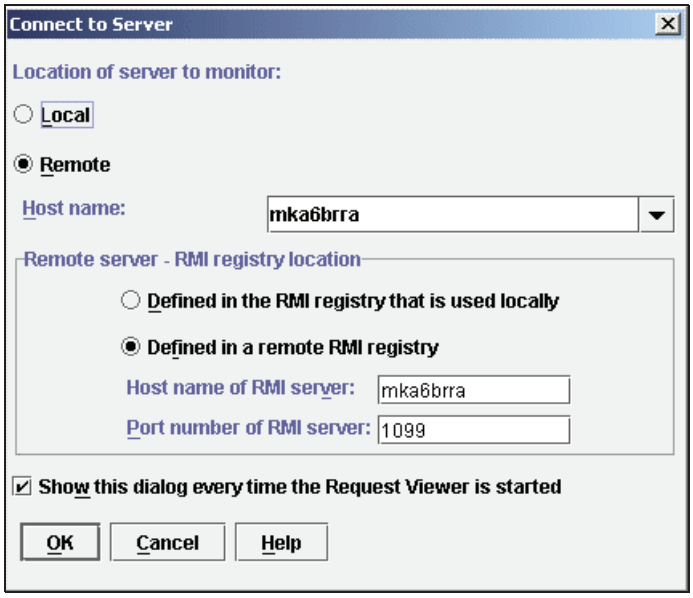


Figure 8-7 Connect to Server

It may take a few seconds for Request Viewer to start. Once it starts you are ready to track requests and view the device-to-server interaction.

Note: Request Viewer is a monitoring tool; therefore, you cannot change the configuration or status of the transcoding plug-ins with it.

8.4 Logging and tracing

To turn on tracing for Transcoding Technology:

1. Log in to the Administration as `wpsadmin` (both user ID and password), then go to the Portal Administration tab. Within the Portal Settings category, select the **Enable Tracing** option. The Enable Tracing tab is shown in Figure 8-8.

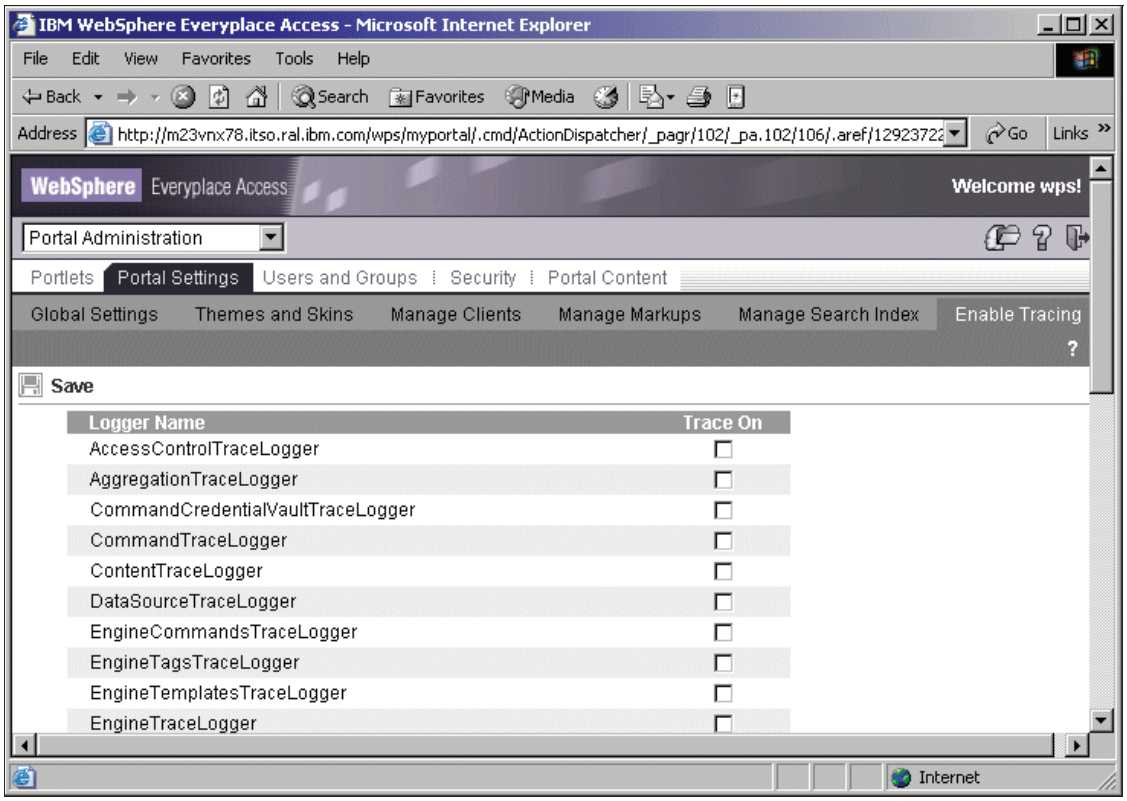


Figure 8-8 Enable tracing portlet

2. Scroll down and check the **TranscodingTraceLogger** box, as shown in Figure 8-9 on page 257.

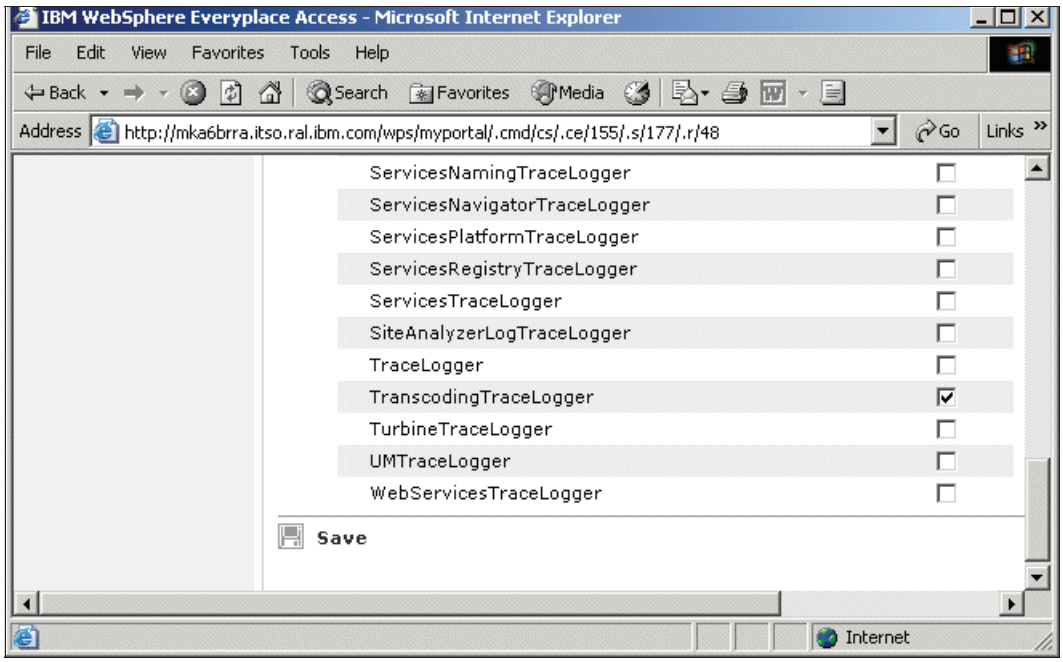


Figure 8-9 Enable tracing for Transcoding Technology

3. Click **Save** after you have selected the desired trace options.

8.4.1 Message files

Message files are created in the Transcoding Technology IBMTrans\log directory. The first message file created is named TranscoderMessages1.log. When the maximum size is reached, this file will be renamed to TranscoderMessages2.log and a new TranscoderMessages1.log file created for new messages. TranscoderMessages1.log is always the newest file. The default message file size is 512 kilobytes.

When the maximum number of message files have been filled, the oldest file will be deleted, the suffix number of each remaining file will be increased by one, and a new TranscoderMessages1.log will be created for new messages. A new file is begun when Transcoding Technology are restarted. The default number of message files is three.

Message file size and maximum number of message files are defined in the <TT_Root>\etc\ras\TranscoderRASMessageFileHandler.properties file.

8.4.2 Trace files

Trace records the specific behavior of Transcoding Technology, which is useful in the diagnosis of problems.

Trace files are created in the Transcoding Technology IBMTrans\log directory. The first trace file created is named TranscoderTrace1.log. When the maximum size is reached, this file will be renamed to TranscoderTrace2.log and a new TranscoderTrace1.log file created for new messages. The default trace file size is 512 kilobytes. TranscoderTrace1.log is always the newest file.

When the maximum number of trace files has been reached, the oldest file will be deleted, the suffix number of each remaining file will be increased by one, and a new TranscoderTrace1.log will be created for new messages. A new file is begun when Transcoding Technology are restarted. The default number of trace files is three.

Tracing file size and maximum number of tracing files are defined in the <TT_Root>\etc\ras\TranscoderRASTraceFileHandler.properties file.

8.4.3 Gather troubleshooting data

When running into problems using Transcoding Technology, there is a certain set of information that will be useful in troubleshooting. There is a batch file named RASCollect.bat in the Transcoding Technology root directory. Executing RASCollect.bat on Windows® will package the necessary files and settings to make it easy to gather diagnostic information. When it finishes execution, a zip file is created in the Transcoding Technology IBMTrans\log directory that you can send to the relevant people for diagnosis. It will be named RASCollect.zip on the Windows server.

8.5 A simple portlet using Transcoding Technology

In this section, we include a sample scenario to show how to enable transcoding in WebSphere Everyplace Access.

8.5.1 Enable transcoding

To give an example of how to enable Transcoding Technology for a portlet, a very simple HelloWorld portlet is used, which does nothing but display a `Hello World` message. The portlet descriptor (portlet.xml) for this portlet is shown in Example 8-4 on page 259. Notice that the portlet.xml has two config-param parameters within the concrete-portlet that set this portlet for transcoding. This ensures that transcoding is set.

Example 8-4 portlet.xml for HelloWorld portlet

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE portlet-app-def PUBLIC "-//IBM//DTD Portlet Application 1.1//EN"
"portlet_1.1.dtd">
<portlet-app-def>
    <portlet-app uid="HelloWorld.50aee4738a9a00171812d35a82af0a5f"
major-version="1" minor-version="0">
        <portlet-app-name>HelloWord application</portlet-app-name>
        <portlet id="Portlet_1" href="WEB-INF/web.xml#Servlet_1"
major-version="1" minor-version="0">
            <portlet-name>HelloWord portlet</portlet-name>
            <cache>
                <expires>0</expires>
                <shared>NO</shared>
            </cache>
            <allows>
                <maximized/>
                <minimized/>
            </allows>
            <supports>
                <markup name="html">
                    <view/>
                </markup>
            </supports>
        </portlet>
    </portlet-app>
    <concrete-portlet-app uid="HelloWorld.50aee4738a9a00171812d35a82af0a5f.1">
        <portlet-app-name>HelloWord application</portlet-app-name>
        <concrete-portlet href="#Portlet_1">
            <portlet-name>HelloWord portlet</portlet-name>
            <default-locale>en</default-locale>
            <language locale="en">
                <title>HelloWord portlet</title>
                <title-short></title-short>
                <description></description>
                <keywords></keywords>
            </language>
<config-param>
<param-name>DoTranscoding</param-name>
<param-value>true</param-value>
</config-param>
<config-param>
<param-name>FilterChain</param-name>
<param-value>Transcoding</param-value>
</config-param>
        </concrete-portlet>
    </concrete-portlet-app>
```

```
</portlet-app-def>
```

Install this portlet into WebSphere Portal. Make sure that transcoding is enabled by doing the following:

1. Log in to WebSphere Everyplace Access Administration as an administrator, for example `wpsadmin`.
2. Go to the Portal Administration tab and select **Global Settings** and make sure the **Enable transcoding of portlet content** box is checked.

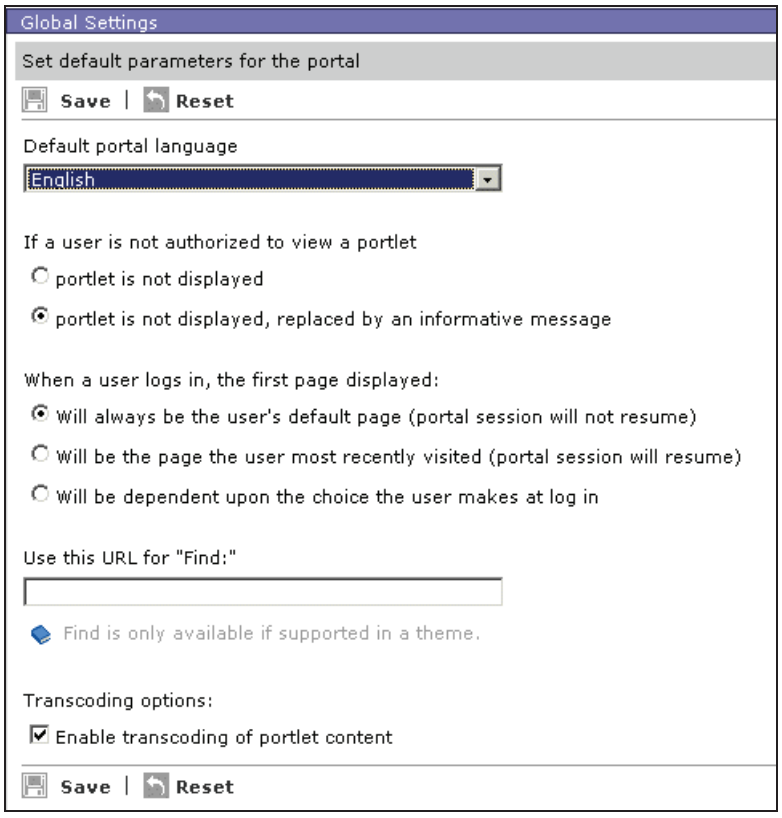


Figure 8-10 Enable global setting for Transcoding Technology

3. To add this portlet to the Welcome page of WebSphere Everyplace Access home page group, perform these steps:
   a. Install the portlet
   b. Set Access Control to allow All anonymous users, retrieve this portlet and select the view for this portlet and save the change.

c. Edit the layout of the Welcome page to add the HelloWorld portlet.

Viewing HelloWorld portlet in a Web browser

Open an Internet Explorer browser and enter the URL. The HelloWorld portlet is shown in Figure 8-11.

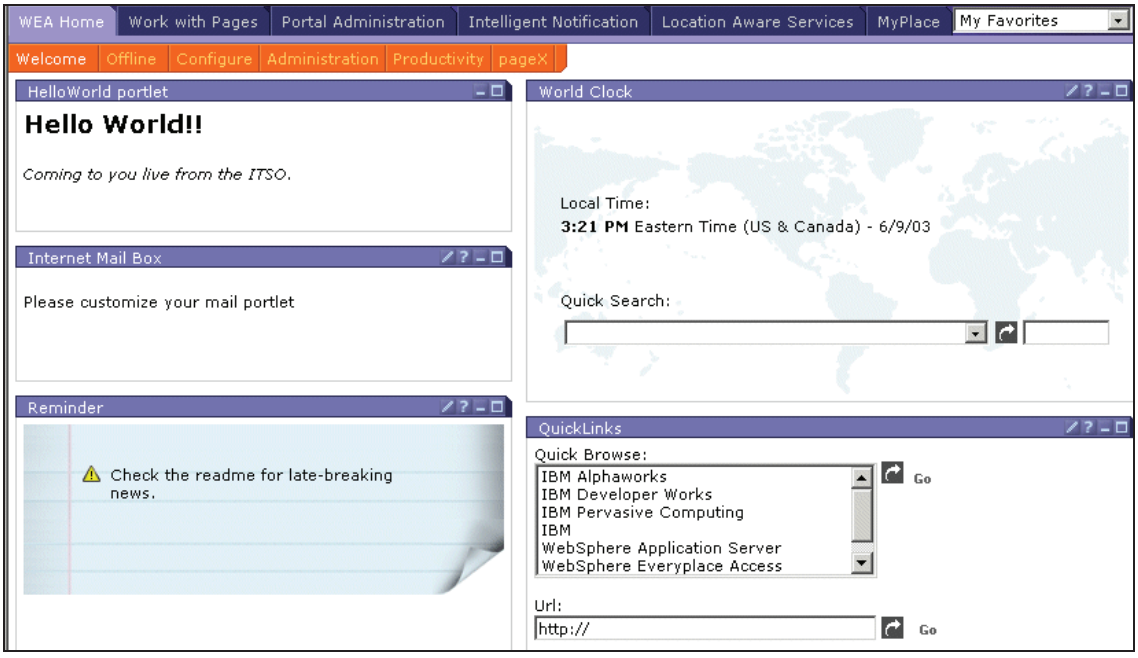


Figure 8-11 HelloWorld portlet displayed in a desktop browser

Viewing HelloWorld portlet in Web browser

Prior to viewing the portlet on the Nokia emulator, we must ensure the device is compatible. To verify this, do the following steps:

1. Use Notepad to open the C:\WebSphere\PortalServer\IBMTrans\etc\plugins\ibm\TextEngine\WMLPrefixer.prop file.
2. Add the statement `!($portletLevel~true)` to the Condition statement. It should look like Example 8-5.

Example 8-5 Condition statement

```
Condition=!($portletLevel~true)&!(x-ibm-wtp~no)&((content-type~text/vnd.wap.wml
)|(content-type~text/x-wap.wml))
```

3. Save the modified file.
4. In the WebSphere Application Server Administration Console, stop and start the WebSphere Portal application server to have the change take effect.

To view the portlet on the Nokia emulator, do the following steps:

1. Use Nokia Mobile Internet Toolkit to view the portlet with a WAP browser. Start Nokia Mobile Internet Toolkit by clicking **Start -> Programs -> Nokia Mobile Internet Toolkit 3.1 -> Mobile Internet Toolkit 3.1**.
2. In the Go field, type the portal URL and press Enter.

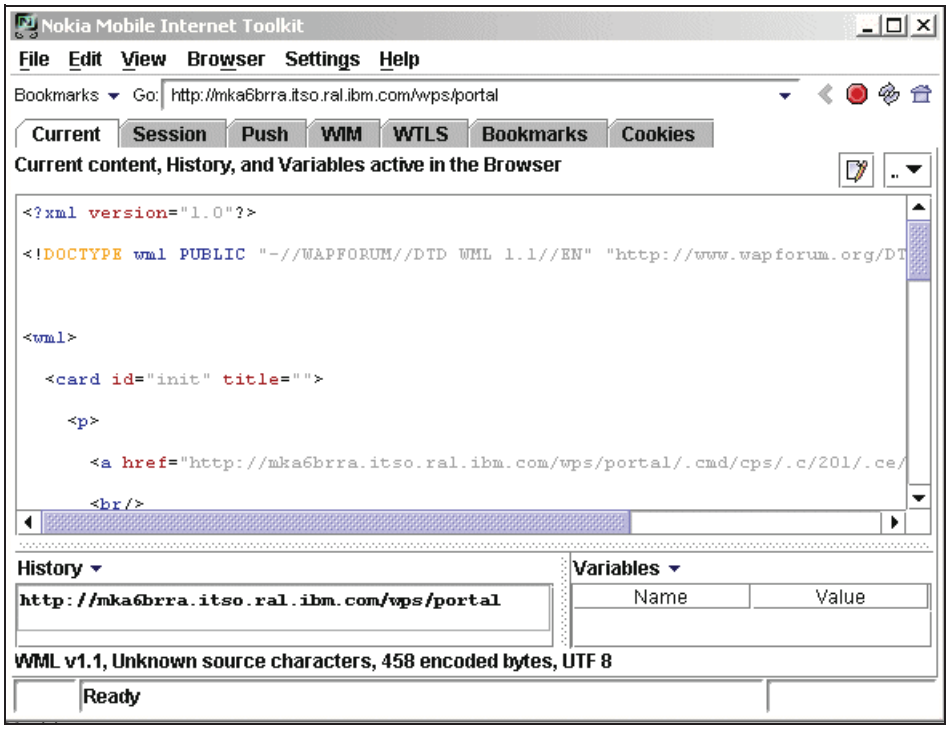


Figure 8-12 Enter URL

3. The Welcome page is shown on the Nokia emulator. Scroll down and select **HelloWorld portlet**. The transcoded portlet is shown in Figure 8-13 on page 263.

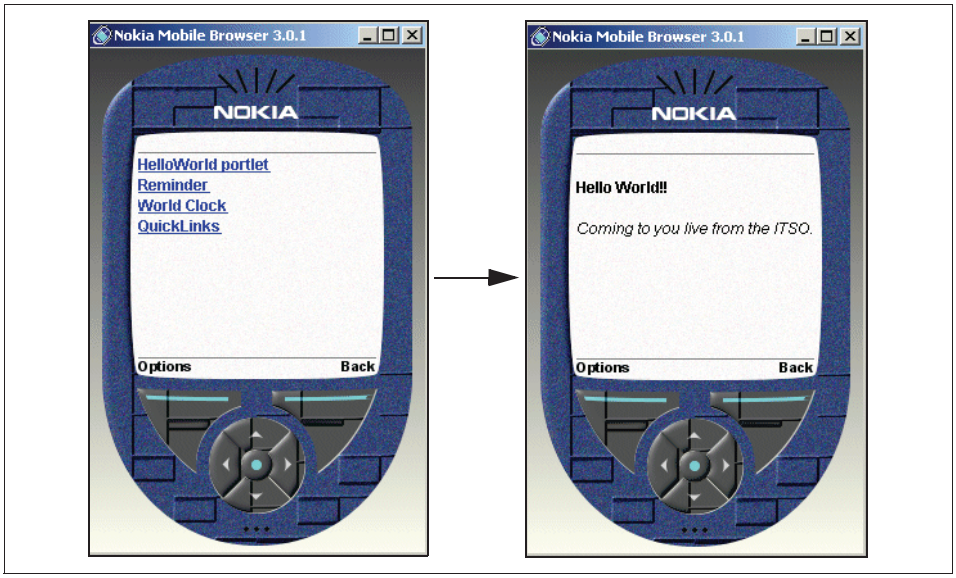


Figure 8-13 HelloWorld portlet displayed in Nokia emulator

8.5.2 Use Request Viewer to monitor the process

Request Viewer can be used to observe and monitor when the portlet and Portal page are transcoded.

Start the Request Viewer (within WebSphere Studio) as detailed in “Starting Request Viewer” on page 253. Use the Nokia emulator to access the HelloWorld portlet. The Request Viewer captures the requests and displays them in its Request Processing window.

As illustrated in Figure 8-14 on page 264, you can expand a request and then expand a transcoder that the request goes through to examine the content changes in MEG Input and MEG Output.

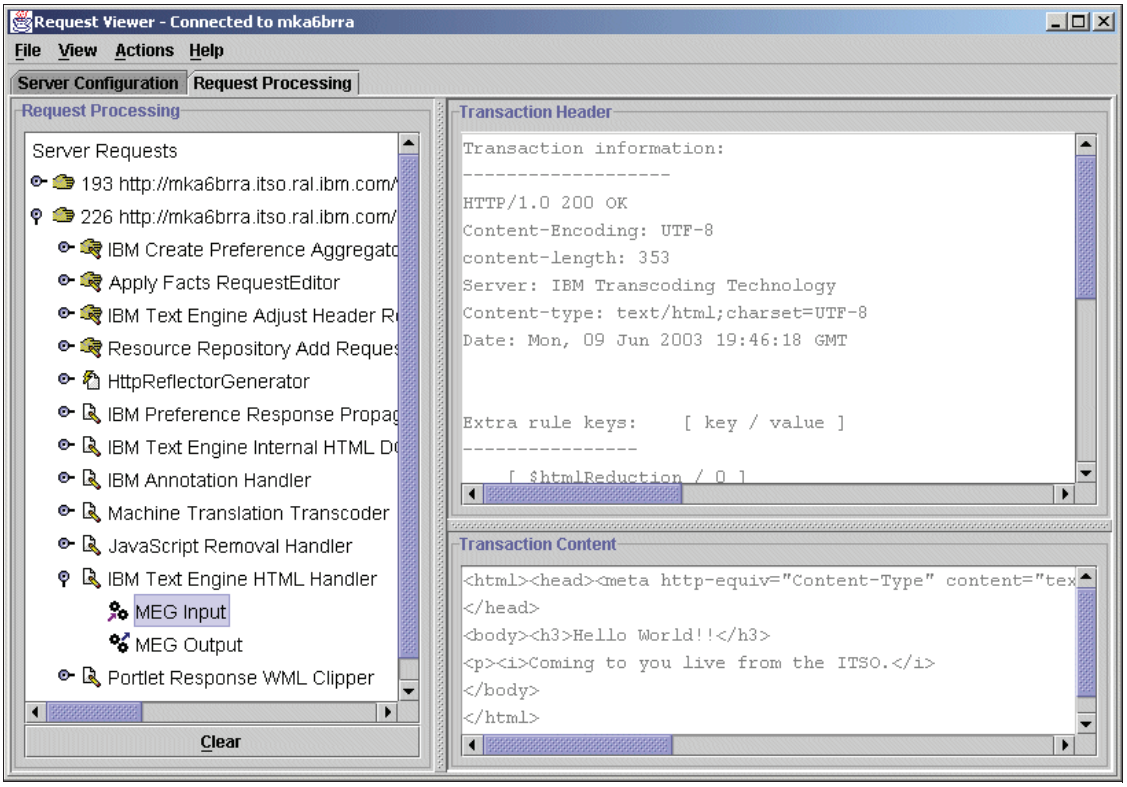


Figure 8-14 Request Viewer

9

A sample scenario using Transcoding Technology

This chapter is designed to give you hands on experience with configuring and deploying Transcoding Technology using WebSphere Everyplace Access. Transcoding technology dynamically translates Web content and applications into mobile device markup languages and optimizes it for delivery to mobile devices, such as mobile phones and handheld computers. This transformation of information from one form to another is a central function of transcoding. Transcoding is particularly suited to bridging the gaps between the representations of Web data and that of various handheld devices.

© Copyright IBM Corp. 2005. All rights reserved.

9.1 Overview

In this sample scenario we show you how to transform Web content to pervasive devices using Transcoding Technology for WebSphere Everyplace Access based on what the requesting device can handle. You explored the three major facets of WEA's Transcoding Technology: transcoding, fragmentation, and annotation. Now a user can use a Web-enabled cell phone to view portlets and transcoding makes all the adjustments necessary.

After completing this lab you should be able to perform content adaptation with Transcoding Technology using WebSphere Everyplace Access.

The following tasks are illustrated in this chapter:

1. Configuring the portlet for Transcoding
2. Text Transcoding
3. Image Transcoding
4. Fragmentation Transcoding
5. Annotation Transcoding
6. XSL Transcoding

Note: You will need to be familiar with Java and understand basic Web concepts such as HTML, XML, HTTP and WML.

Transcoding Technology converts the original HTML to the appropriate device markup language. As part of the transcoding process content may be broken into smaller units (fragments) as needed by the target (receiving) device. The developer can also specify annotations that tell the transcoder to add and/or remove content.

Transcoding Technology supports common Web data representations, including HTML pages, and Extensible Markup Language (XML). Transcoding Technology also tailors images for device limitations including screen size, image file size, and a subset of colors.

You can adapt a portlet's output to the requesting device by passing the output to Transcoding Technology, which uses specialized programs called **transcoding plugins** to perform different conversions. For example, you can transcode a portlet's HTML output into WML (Wireless Markup Language) using the HTML to WML transcoding plugin that is part of Transcoding Technology. You can even write your own transcoding plugin, and use it in Transcoding Technology. Before a portlet can make use of Transcoding Technology, you must configure it by identifying the Transcoding Technology filter you want to use.

By default, portlets are configured to use Transcoding globally. You can turn Transcoding on and off by selecting Enable transcoding of content under the heading Transcoding options in Portal Administration > Portal Settings > Global Settings.

You can also configure individual portlets to use Transcoding, by adding a <config-param> element to the <concrete-portlet> element in the portlet's **portlet.xml** file. To configure a portlet for transcoding, add the parameter **FilterChain** with a value of **Transcoding**. For example, to turn transcoding on for a portlet you will need the following configuration option:

```
<config-param>
   <param-name>FilterChain</param-name>
   <param-value>Transcoding</param-value>
</config-param>
```

Transcoding Technology adapts, reformats, and filters content based on user **preference profiles** to represent the characteristics of devices, and a default user profile to represent organizational policies. Each profile tells Transcoding Technology how to treat documents that will be delivered to that device or user. A preference profile can represent a particular type of device, such as a PDA.

Using the preference profiles, Transcoding Technology can determine the specific set of preference parameters that govern the processing of each request. For example, on a device with limited processing capability, you might want to convert tables to lists to reduce the processing required to present the page.

You can add additional preference profiles to Transcoding Technology for clients not included in the portal, or update or delete Transcoding Technology preference profiles.

If more than one device preference profile matches the incoming request, it is impossible to predict which of the matching profiles would be selected. For example, if you create two device profiles with expressions that could match the same user-agent value, either one could be selected for a request that specifies that user-agent value. Be sure to specify unique values for any new profiles you create.

Each profile contains preference values that are important to the device that the profile represents. If a particular preference is not important to the device, it can be omitted allowing the value to be chosen from a different profile. For most preferences, Transcoding Technology will check profiles for a value in this order:

1. Specific user
2. Specific device

3. Default user
4. Default device

Note: If a value is not specified for a preference in one profile, Transcoding Technology will work down the list until it finds a value.

9.2 WebSphere Everyplace Access configuration

First, you will need to enable transcoding in the global settings. Log in to WEA server as an administrator. For example:

1. Click **Administration** tab.
2. Click **Portal Settings** -> **Global Settings**.
3. Make sure the **Enable transcoding of portlet content** option has been set.



Figure 9-1 Transcoding option

9.3 Text transcoding

You will need to enable transcoding for specific portlets. For example:

1. Login to the WEA server as administrator and then access the welcome page.
2. You will enable the provided *About WebSphere Portal* portlet content to be transcoded to support WML devices.

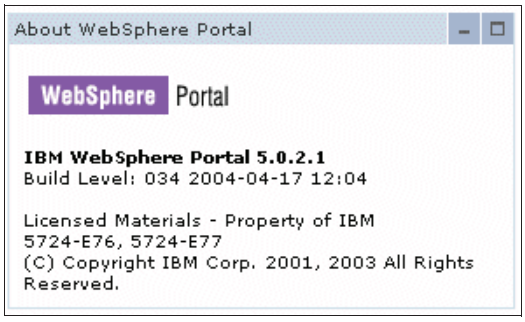


Figure 9-2 About WebSphere Portal portlet

3. Click **Administration**.

4. Click **Portlets** and then **Manage Portlets**.
5. Select **About WebSphere Portal** from the list and click **Modify parameters**.

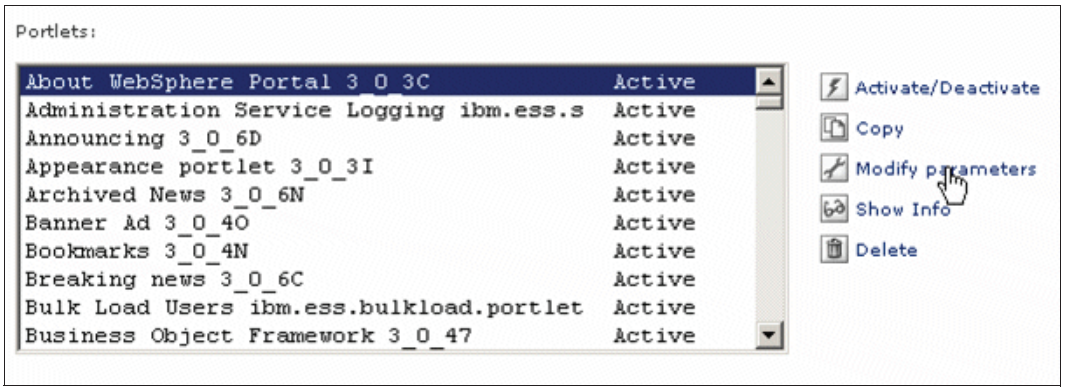


Figure 9-3 Modify parameters option

6. Enter the **FilterChain** parameter with value **Transcoding** and click **Add**.

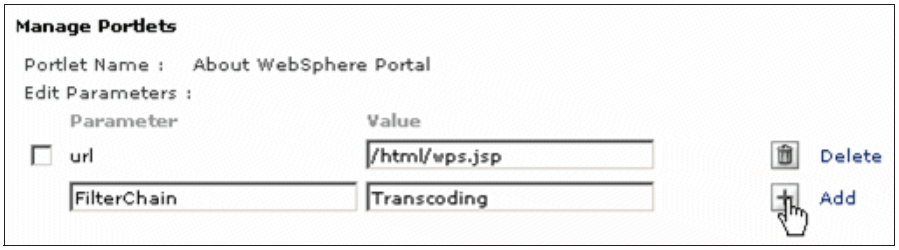


Figure 9-4 Enable transcoding

7. Click **Save** to save the transcoding configuration parameter.
8. Now you will test with a WAP emulator. Open the emulator.
9. For example, perform the following steps to execute the portlet:
   a. Click **http://**
   b. Enter your WEA server URL and click **Go there** link.
   c. Click **About WebSphere Portal**.
   d. You will see the content of the portlet.



Figure 9-5 Accessing the portlet

9.4 Image transcoding

The transcoding plugin converts, reduces or eliminates images to match display capabilities and optimize delivery to constrained device. For example follow these steps:

1. In this section you will also enable transcoding for the **Welcome to WebSphere Portal version 5.0** portlet.

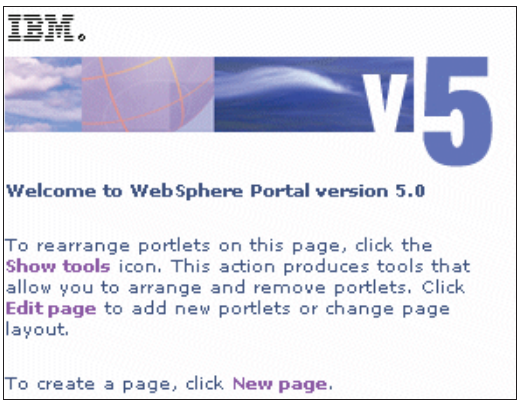


Figure 9-6 Welcome portlet

2. Enable transcoding for this portlet (see 9.3, “Text transcoding” on page 268) by executing the following:
   a. This time select the **Welcome to WebSphere Portal** portlet.
   b. Click **Modify parameters**.
   c. Add the *FilterChain* parameter with value `Transcoding`.
   d. Save the configuration values.

3. Perform the following steps:
   a. From your phone, click left phone button. 
   b. Click **Minimize.**
   c. Click **Welcome to WebSphere Portal**.
   d. Click **Image** and the image is displayed.

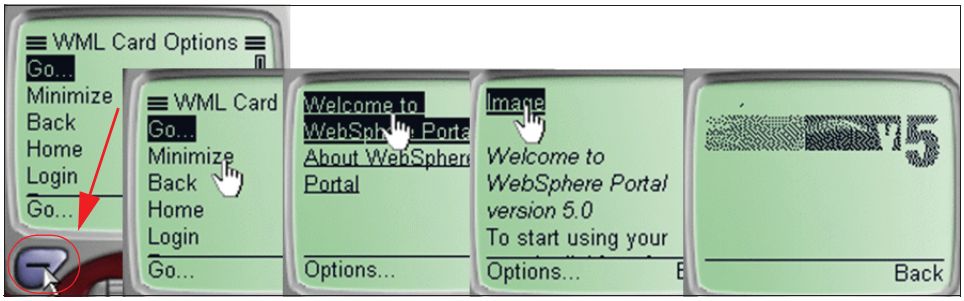


Figure 9-7 Displaying the image

9.5 Fragmentation portlet

The fragmentation transcoding portlet receives requests for WML or Compact HTML. Based on the requesting devices maximum fragment size the fragmentation portlet subdivides the data for that device. As the device needs additional data, Transcoding Technology manages links between the units of data. So as the device wants the next unit of data, the fragment can be sent until all units have been processed. In this way, Transcoding Technology prevents the device from being overwhelmed and potentially losing data.

The fragmentation transcoder maintains a repository of fragmented components specific to a particular request. Transcoding Technology uses specialized URLs for each fragment to ensure that the fragments are sent to the requesting device in proper order.

One clear benefit of the fragmentation transcoder function is that developers do not have to worry as much about the capacity of the device. They can focus on the content they want to generate and let Transcoding Technology handle the process of chunking and sending the data.

For example execute the following steps to create a basic portlet to be used for in this sample scenario:

1. Open toolkit and create a new project by selecting **File** -> **New** -> **Project**…

2. Click **Portlet Development** and select **Portlet Application Project** from the right panel.
3. Enter **FragmentationTest** as Project name. Select **Create basic portlet**. Click **Finish** to create the portlet.
4. If prompted, click the **Yes** button to switch to the portlet perspective.
5. Double-click **FragmentationTestPortletView.jsp.**

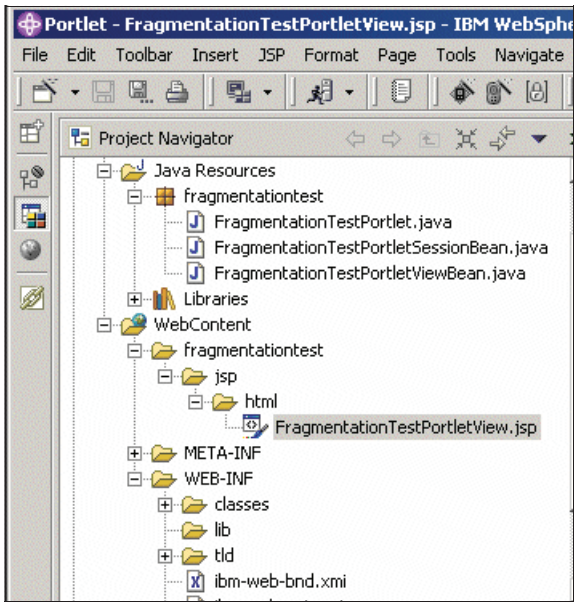


Figure 9-8 Created JSP for portlet view mode

6. Update the JSP content so that it can hold a large amount of data to be displayed as shown in Figure 9-9.

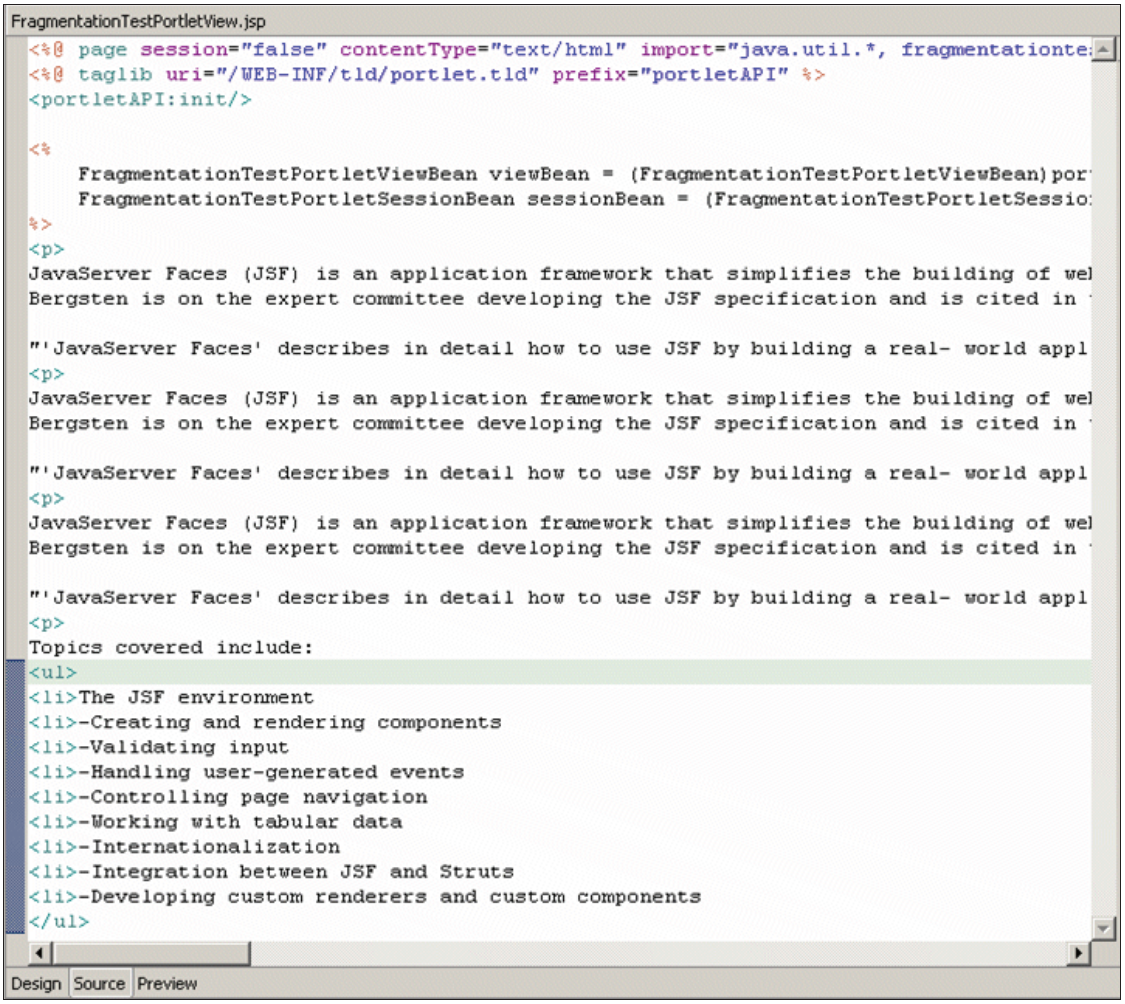


Figure 9-9 Updated JSP

7. Save it using the **Ctrl-S** keys.
8. Select the project and click **Export...** to export the updated portlet as a WAR file.

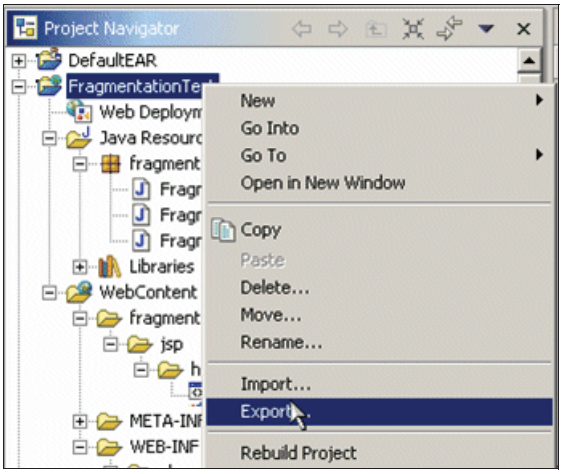


Figure 9-10 Exporting the portlet as a WAR file

9. Select **WAR File** and click **Next**.
10. Click **Browse...** button and select the local directory and WAR file name for Destination entry. Click **Finish** button.

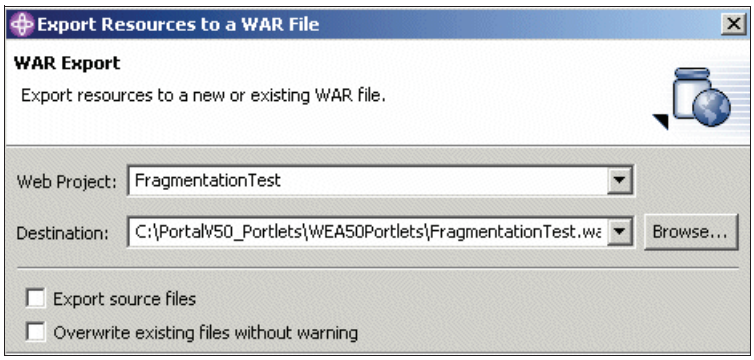


Figure 9-11 Exporting the WAR file

11. Login to WEA Server as an administrator. For example wpsadmin to install the portlet.
12. Click **Administration**.

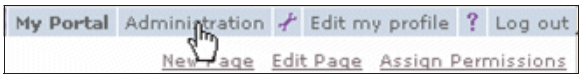


Figure 9-12 Administration

13. Click **Portlets** and then **Install**.
14. Click **Browse…** button.
15. Navigate local file system for the Directory entry. Click **Next**.

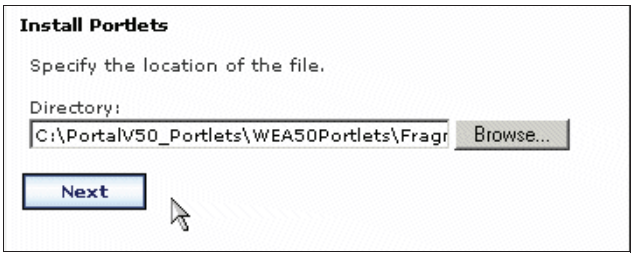


Figure 9-13 Installing the portlet

16. Click **Install** button.
17. After the successful installation, click **Manage Portlets** to enable the portlet for transcoding.

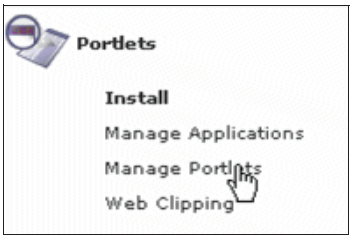


Figure 9-14 Manage portlets

18. Select the **FragmentationTest portlet** from the list and click **Modify parameters**.
19. Enter FilterChain/Transcoding parameter and value. Click **Add**.

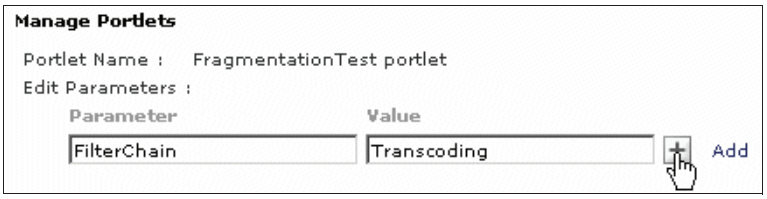


Figure 9-15 Setting the portlet transcoding option

20. Click **Save**.

21. Use the administrative options to create a new page and deploy the portlet. For example, select **Portal User Interface** and then click **Manage Pages**.



Figure 9-16 Manage pages

22. For example, create a new page under My Portal. Click **My Portal**.
23. Click **New Page**.



Figure 9-17 New page

24. Enter **Transcoding** for Title and click **Advanced options**.
25. Check **WML** and **cHTML**. Click **OK**.

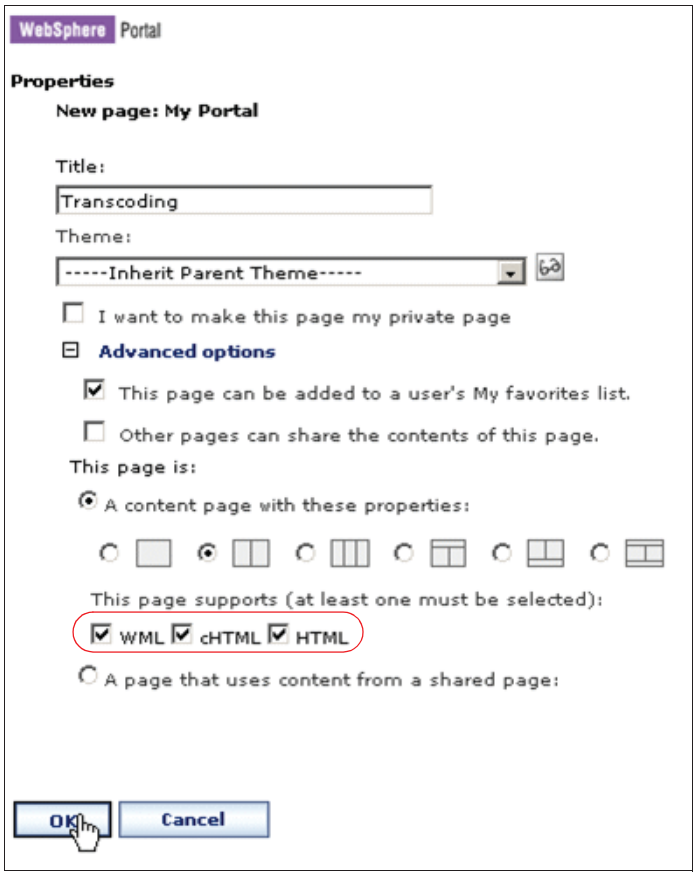


Figure 9-18 Page title and advanced options

26. After a successful page creation, select **OK**.
27. Click **My Portal** tab.

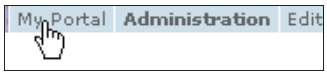


Figure 9-19 My Portal tab

Note: If you do not see new page tab, click the expand mark.

28. Click new page **Transcoding** you just created.



Figure 9-20 Created portal page

29. Click **Edit Page** tab.



Figure 9-21 Edit page

30. From Edit Layout panel, click **Add portlets**.
31. Search for the portlet. For example, enter **frag** and click **Search** button.
32. Check **FragmentationTest portlet** and then click **OK**.
33. Click **Done** to add the portlet to the portal page.
34. Verify that the portlet displays a large amount of html content.

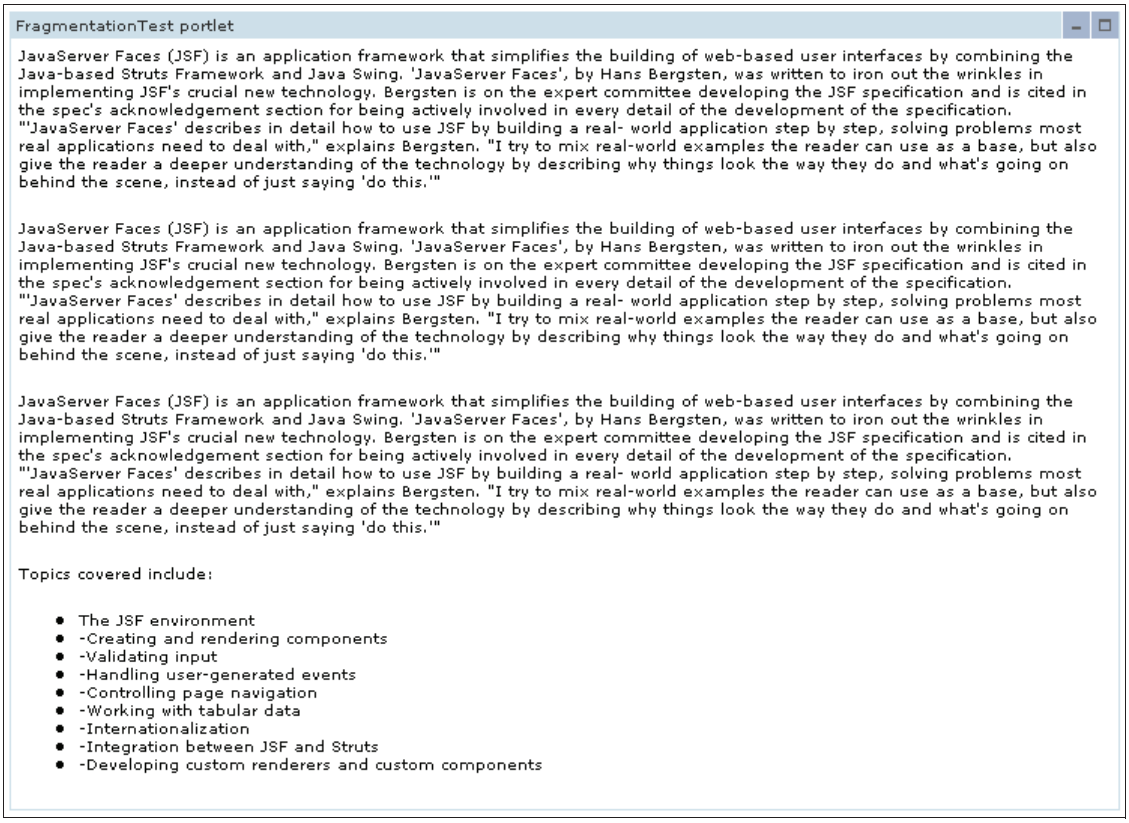


Figure 9-22 Displaying a large amount of data in a desktop browser (html)

35. Open the WAP phone emulator.
36. Right-click on the phone and select **Open URL**.
37. Enter your WEA server URL and then click **Options…** link. Click **Login** and enter user ID and password.

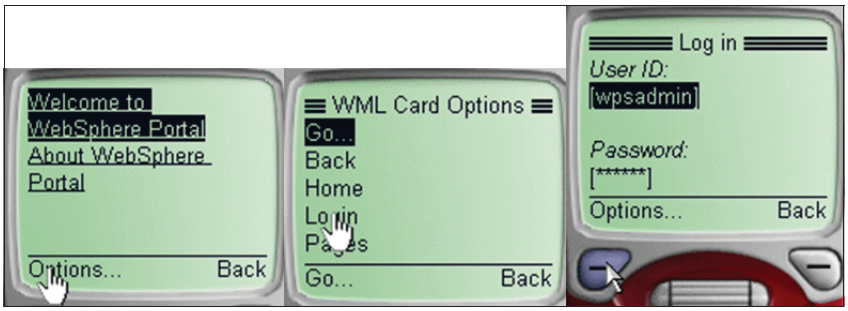


Figure 9-23 Log in to Everyplace Access server

38. After logging in, click the Options button and find the created page with name Transcoding.
39. Click **Pages**.
40. Scroll down until indicator reaches to the bottom.
41. Click Option button.

    Note: You may need to use the Continue option.

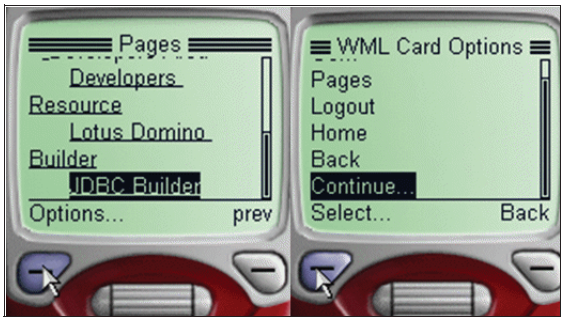


Figure 9-24 Accessing the created portal page

42. Keep clicking **Continue…** until you can see Transcoding page. Click **Transcoding**. Click **Go…**

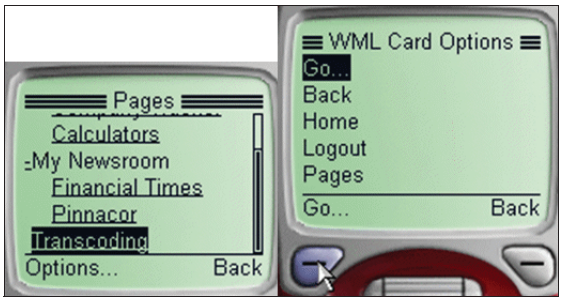


Figure 9-25 Selecting the portal page with name Transcoding

43. Execute the sample portlet to invoke portlet content fragmentation:
    a. Click **FragmentationTest portlet**.
    b. Click **Go…**
    c. Scroll down until the indicator reaches to the bottom.



Figure 9-26 Displaying the first card content

    d. Click Option button .
    e. Click **Continue…**
    f. You will see the next text content loaded.

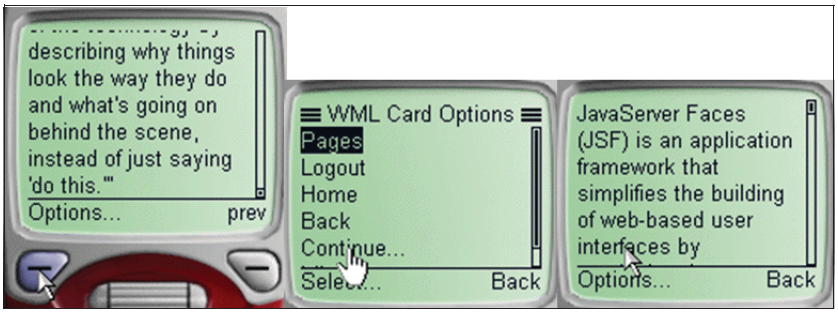


Figure 9-27 Using the Continue option to navigate the content

9.6 Annotations

Transcoding Technology uses a special transcoder called the “annotation engine” to handle annotation processing.

The annotation process follows these general steps:

- Transcoding Technology receives HTML file to be clipped with annotator.
- A Document Object Model (DOM) is generated from the HTML file.
- Elements in HTML DOM that are to be manipulated are identified in annotator with XML Path Language (XPath) statements.
- Annotation engine modifies HTML DOM according to annotations.
- Depending on device profile, the text transcoder can convert modified HTML DOM to a different data format, such as WML.
- Transcoded data is then sent to the requesting device.

The DOM defines the logical structure of an HTML or XML document and essentially provides a standard API for adding, changing, and deleting parts of a document.

When viewed in the DOM format, the document is presented as a hierarchy of nodes or elements. The DOM view is a logical representation of the actual document, which is shown in the **Source** view.

Annotators specify where an annotation takes place by using an **XML Path_ Language (XPath)** expression in a **<description>** element. Other tags are then used to describe the type of annotation being performed.

Bring up Provide Document Clipping Annotation is performed once before transcoding to various markup languages. Transcoding is therefore only acting

on a subset of original HTML content, improving efficiency No Java programming is required.

1. **File** -> **New** -> **Portlet Application Project**.
2. Enter **AnnotationTest** for Project name. Check **Create basic portlet** and then click **Finish**.
3. Enter the following configuration parameter in the portlet descriptor (portlet.xml) to enable transcoding for this project:

```
<config-param>
       <param-name>FilterChain</param-name>
       <param-value>Transcoding</param-value>
</config-param>
```

4. The HTML source shown in Example 9-1 will be used for this sample scenario using annotation.

Example 9-1 HTML code

```
<HTML>
<HEAD>
<TITLE>IBM Stock 297.12
</TITLE>
<META NAME="OWNER" CONTENT="webmaster@www.ibm.com">
<LINK REL="MADE" CONTENT="mailto:webmaster@www.ibm.com">
<META NAME="ALIAS" CONTENT="/IBM/Stock/Ticker/">
<META CONTENT="900" HTTP-EQUIV="refresh">
</HEAD>
<BODY BGCOLOR="#000000">
<A HREF="http://www.ibm.com/" TARGET=_blank>
<IMG SRC="/i/v7/m/ibm-ticker.gif" WIDTH=158 HEIGHT=24 BORDER=0 ALT="Stock"></A>
<TABLE WIDTH=164 UNITS=PIXELS>
<TR HALIGN=CENTER>
<TD COLSPAN=2>
<FONT COLOR="#8888FF" SIZE=6>
<STRONG>
<CENTER>297.12
</CENTER>
</STRONG>
</FONT></TD></TR>
<TR><TD ALIGN=LEFT>
<FONT COLOR="#CC22CC" SIZE=2>High: 297.12
</FONT></TD>
<TD ALIGN=RIGHT><FONT COLOR="#CC22CC" SIZE=2>Low: 295.73
</FONT></TD>
</TR>
<TR><TD COLSPAN=2><P><FONT SIZE=-2 COLOR="#ffffff"><STRONG>Last
update</STRONG>: August 16, 14:44
 EST</TD></TR></TABLE>
```

```
</BODY></HTML>
```

5. Double-click AnnotationTestPortletView.jsp.

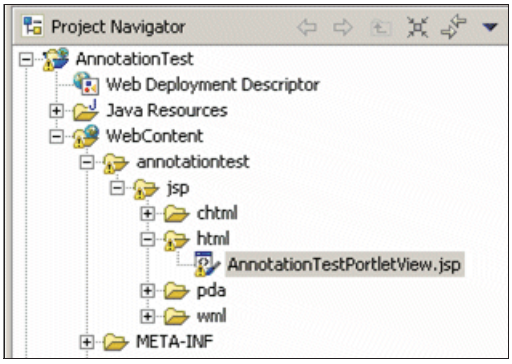


Figure 9-28 Editing the JSP to add new code

6. Overwrite the contents of Example 9-1 on page 283 after the lines shown in Figure 9-29.

```
<%@ page session="false" contentType="text/html" import="java.util.*, annotationtest.*"%>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>
```

Figure 9-29 JavaServer Page initial code

7. Export the WAR file to your local file system.
8. Login to WEA Server as an administrator and install the portlet.
9. Select the previously created Transcoding page and use the edit page option to add the portlet using the **Add Portlet** button.
10. Enter **anno** and click **Search**. You will get the portlet list starting with *anno*. Check **AnnotationTest portlet** and then click **OK**.
11. Click **Done**.
12. You will see the portlet with the broken image link.

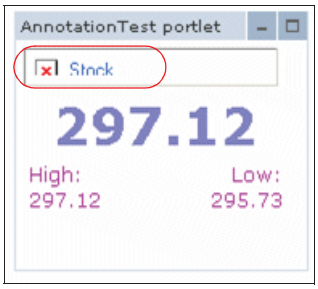


Figure 9-30 Portlet with broken link

13. As you can see in Figure 9-31, the image link is written using a relative link and the link is broken when the portlet runs on the WEA server.

```
<HTML>
<HEAD>
<TITLE>IBM Stock 297.12
</TITLE>
<META NAME="OWNER" CONTENT="webmaster@www.ibm.com">
<LINK REL="MADE" CONTENT="mailto:webmaster@www.ibm.com">
<META NAME="ALIAS" CONTENT="/IBM/Stock/Ticker/">
<META CONTENT="900" HTTP-EQUIV="refresh">
</HEAD>
<BODY BGCOLOR="#000000">
<A HREF="http://www.ibm.com/" TARGET=_blank>
<IMG SRC="/i/v7/m/ibm-ticker.gif" WIDTH=158 HEIGHT=24 BORDER=0
ALT="Stock"></A>
<TABLE WIDTH=164 UNITS=PIXELS>
<TR HALIGN=CENTER>
<TD COLSPAN=2>
<FONT COLOR="#8888FF" SIZE=6>
<STRONG>
<CENTER>297.12
</CENTER>
</STRONG>
</FONT></TD></TR>
<TR><TD ALIGN=LEFT>
<FONT COLOR="#CC22CC" SIZE=2>High: 297.12
</FONT></TD>
<TD ALIGN=RIGHT><FONT COLOR="#CC22CC" SIZE=2>Low: 295.73
</FONT></TD>
</TR>
<TR><TD COLSPAN=2><P><FONT SIZE=-2
COLOR="#ffffff"><STRONG>Last update</STRONG>: August 16, 14:44
 EST</TD></TR></TABLE>
</BODY></HTML>
```

Figure 9-31 JTML code

14. As an example, you will apply the external annotator to replace the relative link with an absolute image link. The annotator shown in Figure 9-32 will find the first image link and replace the source HTML. Let's name this annotator ibmstock.ann.

```
<?xml version='1.0' ?>
<annot version="2.0">
<description take-effect="before" target="/HTML/HEAD">
   <remove/>
</description>
<description take-effect="before" target="/HTML/BODY/A[1]">
   <keep/>
</description>
<description target="/HTML/BODY/A[1]/IMG[1]">
      <replacewithhtml>
         <![CDATA[
           <IMG SRC="http://www.ibm.com/i/v7/m/ibm-ticker.gif"
                  WIDTH=158 HEIGHT=24 BORDER=0 ALT="Stock">]]>
      </replacewithhtml>
</description>
</annot>
```

Figure 9-32 Annotator (ibmstock.ann)

15. You need to include the external annotator in the portlet. You will import ibmstock.ann file under **WEB-INF** directory. In the project navigator, select **WEB-INF** and click **Import….**
16. Select **File System** and click **Next**.
17. Click **Browse…** button
18. Navigate to ibmstock.ann file and click **Finish**. You will now see the ibmstock.ann file under WEB-INF folder.
19. Export the portlet.
20. Login to WEA Server as an administrator and install the portlet.
21. Click **Portlets** -> **Manage Portlets**. Select **AnnotationTest portlet** and click **Modify parameters**.

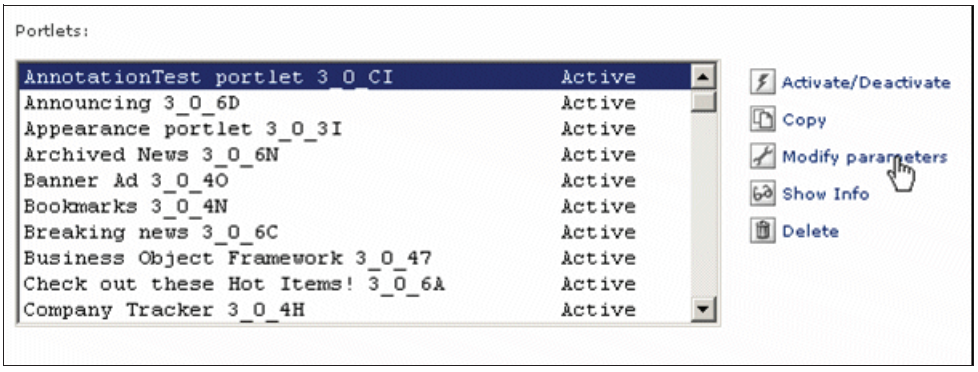


Figure 9-33 Modify parameters

22. Configure the annotator by entering **AnnotationFile** as parameter and **WEB-INF/ibmstock.ann** for its value. Click **Add** as illustrated in Figure 9-34.

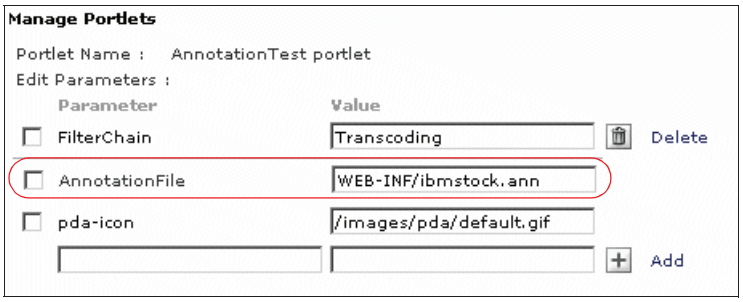


Figure 9-34 Adding the AnnotationFile parameter

23. Go to Transcoding page again and you will see the portlet with the image since the annotator was applied in this case.

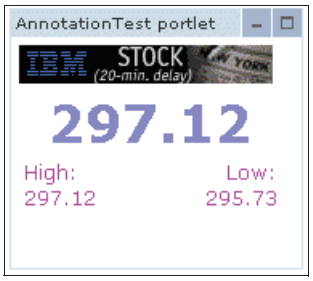


Figure 9-35 Portlet view after applying the annotator

24. However, you will find that the portlet still does not display the last two statements.



Figure 9-36 Statements not displayed

25. You will notice the font color is white (“#ffffff”). You can replace it but for this sample scenario, you will remove the font statement as its default text color is black.

```
<HTML>
<HEAD>
<TITLE>IBM Stock 297.12
</TITLE>
<META NAME="OWNER" CONTENT="webmaster@www.ibm.com">
<LINK REL="MADE" CONTENT="mailto:webmaster@www.ibm.com">
<META NAME="ALIAS" CONTENT="/IBM/Stock/Ticker/">
<META CONTENT="900" HTTP-EQUIV="refresh">
</HEAD>
<BODY BGCOLOR="#000000">
<A HREF="http://www.ibm.com/" TARGET=_blank>
<IMG SRC="/i/v7/m/ibm-ticker.gif" WIDTH=158 HEIGHT=24 BORDER=0
ALT="Stock"></A>
<TABLE WIDTH=164 UNITS=PIXELS>
<TR HALIGN=CENTER>
<TD COLSPAN=2>
<FONT COLOR="#8888FF" SIZE=6>
<STRONG>
<CENTER>297.12
</CENTER>
</STRONG>
</FONT></TD></TR>
<TR><TD ALIGN=LEFT>
<FONT COLOR="#CC22CC" SIZE=2>High: 297.12
</FONT></TD>
<TD ALIGN=RIGHT><FONT COLOR="#CC22CC" SIZE=2>Low: 295.73
</FONT></TD>
</TR>
<TR><TD COLSPAN=2><P><FONT SIZE=-2
COLOR="#ffffff"><STRONG>Last update</STRONG>: August 16, 14:44
 EST</TD></TR></TABLE>
</BODY></HTML>
```

Figure 9-37 Last two statements in HTML code

26. The following ibmstock1.ann will remove the last font statement.

```
<?xml version='1.0' ?>
<annot version="2.0">
<description take-effect="before" target="/HTML/HEAD">
   <remove/>
</description>
<description take-effect="before" target="/HTML/BODY/A[1]">
   <keep/>
</description>
<description target="/HTML/BODY/A[1]/IMG[1]">
      <replacewithhtml>
         <![CDATA[<IMG SRC="http://www.ibm.com/i/v7/m/ibm-ticker.gif" WIDTH=158 HEIGHT=24 BORDER=0 ALT="Stock">]]>
      </replacewithhtml>
</description>
<description take-effect="before" target="/HTML/BODY/TABLE[1]/TBODY[1]/TR[3]/TD[1]/P[1]/FONT[1]">
  <remove/>
</description>
<description take-effect="before" target="/HTML[1]/BODY[1]/TABLE[1]/TBODY[1]/TR[3]/TD[1]/P[1]/FONT[1]/STRONG[1]">
  <keep/>
</description>
</annot>
```

Figure 9-38 Sample annotator to remove the font statement

27. Import the new annotation file and deploy the portlet as you did in Step 15 – 20.

28. You will create a portlet instance to apply different annotators. Click **Manage Portlets**. Select **AnnotationTest portlet** and click **Copy**.

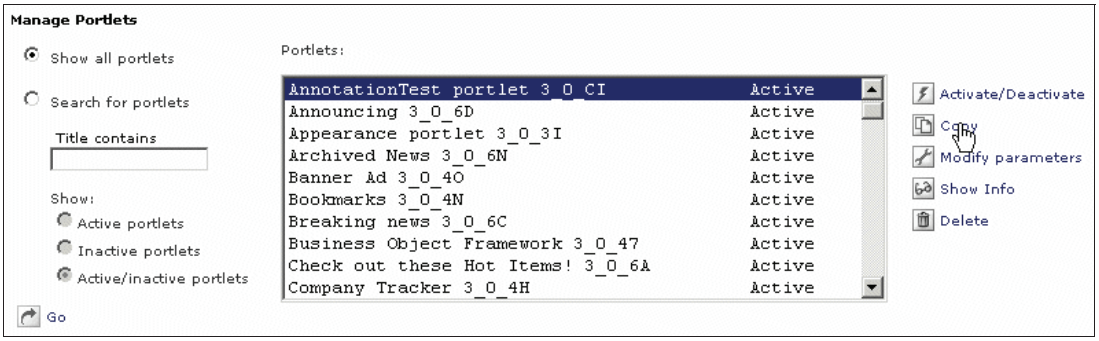


Figure 9-39 Create a portlet instance

29. You will now find two AnnotationTest portlet entries.

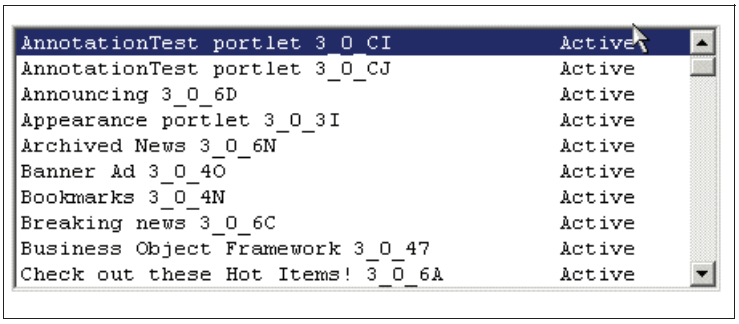


Figure 9-40 Created portlets

30. Select the one you just copied and then click **Modify parameters**.
31. Click **Set title selected locale**.



Figure 9-41 Setting the portlet title

32. Enter **AnnotationTest portlet 1** for the title and then click **OK**.

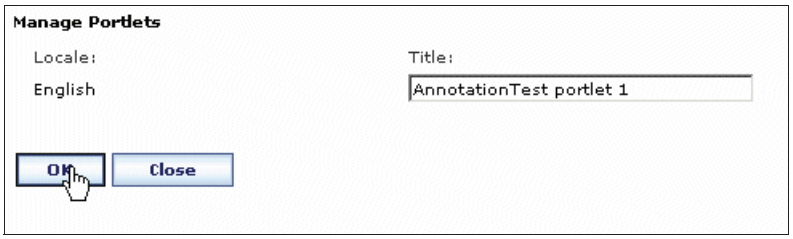


Figure 9-42 Portlet title

33. Check AnnotationFile and enter the new annotator WEB-INF/ibmstock1.ann.

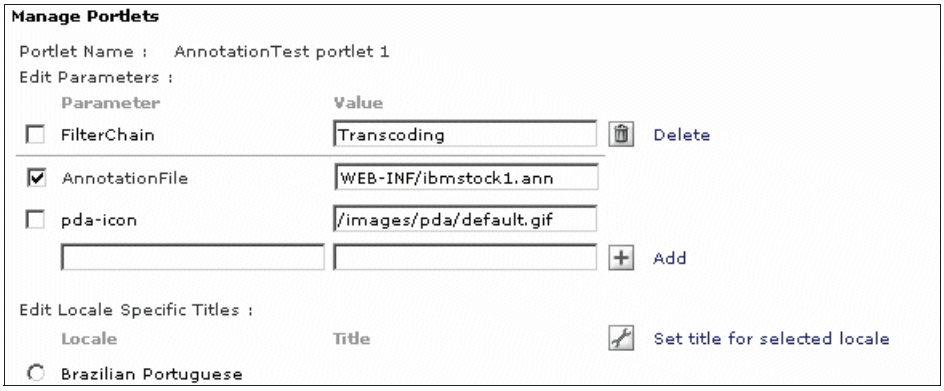


Figure 9-43 Entering the AnnotationFile parameter

34. Click **Save**.
35. Add the AnnotationTest portlet 1 to the previously created Transcoding page.
36. Click **Done**.
37. Click Transcoding page and you will see the two portlets.

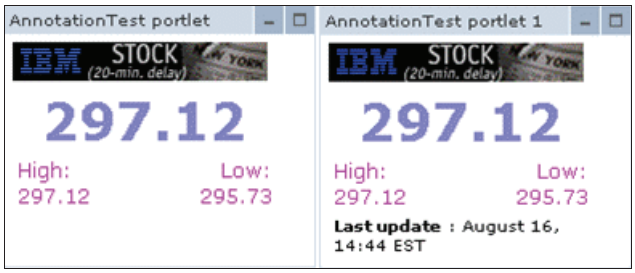


Figure 9-44 Sample portlets after annotators were applied

38. The following ibmstock2.ann will insert more text and an image.

Example 9-2 Sample annotation (ibmstock2.ann)

```
<?xml version='1.0' ?>
<annot version="2.0">
<description take-effect="before" target="/HTML/HEAD">
  <remove/>
</description>
<description take-effect="before" target="/HTML/BODY/A[1]">
   <keep/>
</description>
<description target="/HTML/BODY/A[1]/IMG[1]">
      <replacewithhtml>
        <![CDATA[<IMG SRC="http://www.ibm.com/i/v7/m/ibm-ticker.gif" WIDTH=158
HEIGHT=24 BORDER=0 ALT="Stock">]]>
      </replacewithhtml>
</description>
<description take-effect="before" target="/HTML/BODY/TABLE[1]">
  <keep/>
</description>
<description take-effect="before"
target="/HTML/BODY/TABLE[1]/TBODY[1]/TR[3]/TD[1]/P[1]/FONT[1]">
  <remove/>
<inserthtml>
  <![CDATA[ WEA Testing<br/><img
src="http://www-106.ibm.com/developerworks/websphere/zones/portal/images/headpo
rtalr.jpg"><br/>   ]]>
</inserthtml>
</description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/TABLE[1]/TBODY[1]/TR[3]/TD[1]/P[1]/FONT[1]/STRONG[1]">
  <keep/>
</description>
</annot>
```

39. Import the annotator and create a portlet copy to apply ibmstock2.ann using the same procedure as before.

40. Deploy the AnnotationTest portlet to the Transcoding page.

41. Click Transcoding page and you will see the three portlets.

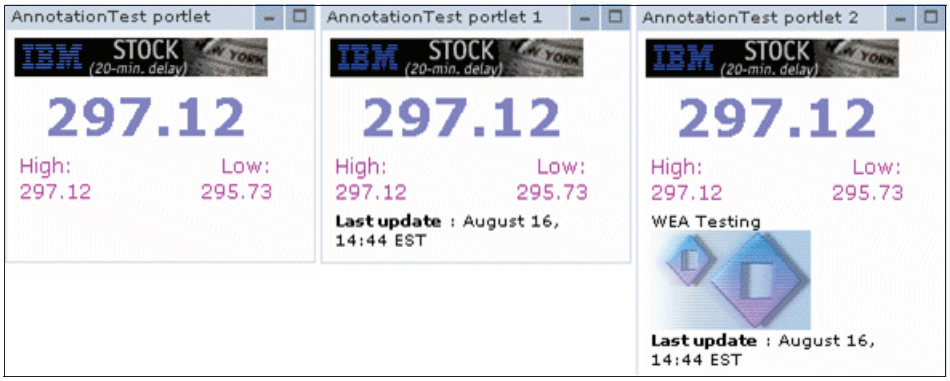


Figure 9-45 Portlet after configured annotators were applied

42. The following ibmstock3.ann will remove all images and anchors.

Example 9-3 Sample annotator (ibmstock3)

```
<?xml version='1.0' ?>
<annot version="2.0">
<description take-effect="before" target="/HTML/HEAD">
   <remove/>
</description>
<description take-effect="before" target="/HTML/BODY/A[1]">
   <keep/>
<remove tag="IMG"/>
<remove tag="A"/>
</description>
<description take-effect="before"
target="/HTML/BODY/TABLE[1]/TBODY[1]/TR[3]/TD[1]/P[1]/FONT[1]">
  <remove/>
</description>
<description take-effect="before"
target="/HTML[1]/BODY[1]/TABLE[1]/TBODY[1]/TR[3]/TD[1]/P[1]/FONT[1]/STRONG[1]">
  <keep/>
</annot>
```

43. Figure 9-46 illustrates the result. You can also view the portlet in the WAP device.

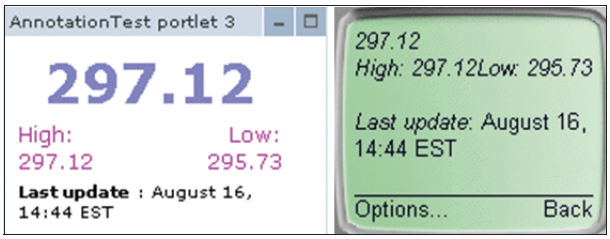


Figure 9-46 Results and WAP device view

9.7 Using external annotators

The following guidelines applied to external annotators:

1. To take out the headings the next line should be:

   ```
   <remove/>
   ```

   This tag will remove everything in the clipping region. In your case, the clipping region starts at the first <h2> tag and extends until the next region is declared.

2. To replace what has been removed, use the next line:

   ```
   <replacewithhtml>
   ```

   This starts the process of replacing the removed text with html.

3. The next line should be:

   ```
   <![CDATA[ <h3> Contact Names and Ext.</h3>]]>
   ```

   Notice your html is inside a comment/CDATA tag to prevent the parser from picking it up at the wrong time.

4. You need to close your open tags. Enter the following:

   ```
   </replacewithhtml>
   </description>
   ```

5. Your next clipping region will be for the table that is generated by the JSP, your description tag will be as follows:

   ```
   <description take-effect="before" target="/descendant::TABLE[1]" >
   ```

   This creates a region starting at the first <table> tag past the start of the last clipping region.

6. To set the clipping state, use the next tag:

   ```
   <keep tag="all" />
   ```

This sets the clipping state to “keep” and is equivalent to <keep/>, just as <remove tag=”all”/> is equivalent to <remove/>

7. Your next tag specifies that you will be clipping within a table:

   ```
   <table majoraxis="row">
   ```

   The parameter “majoraxis=’row’” specifies that you want the rows separated out into a record-type format on the cell phone.

8. The following tag is entered next:

   ```
   <column index="*" clipping="keep" />
   ```

   This tells the annotation engine to keep all of the columns unless otherwise specified.

9. The next tag is similar in form:

   ```
   <column index="3" clipping="remove" />
   ```

   This tag simply removes column 3 from the table.

10. You now just need to close the open tags. Since this is the last part of the file, you need to close the annotation tag as well:

    ```
    </table>
    </description>
    </annot>
    ```

9.8 Using stylesheets

1. Create a project called **FlightInfo**.
2. Import XSLT stylesheet. Select **WEB-INF** directory and click **Import…** .
3. Select **File System** and click **Next.**
4. Click **Browser…** button and navigate **FlightInfoForNet_IE.xsl**. If you installed portal server, you can find it from **<Portal>\IBMTrans\toolkit\stylesheets\samples** directory where <Portal> is the directory you installed Portal server. Check **FlightInfoForNet_IE.xsl** and click **Finish**.

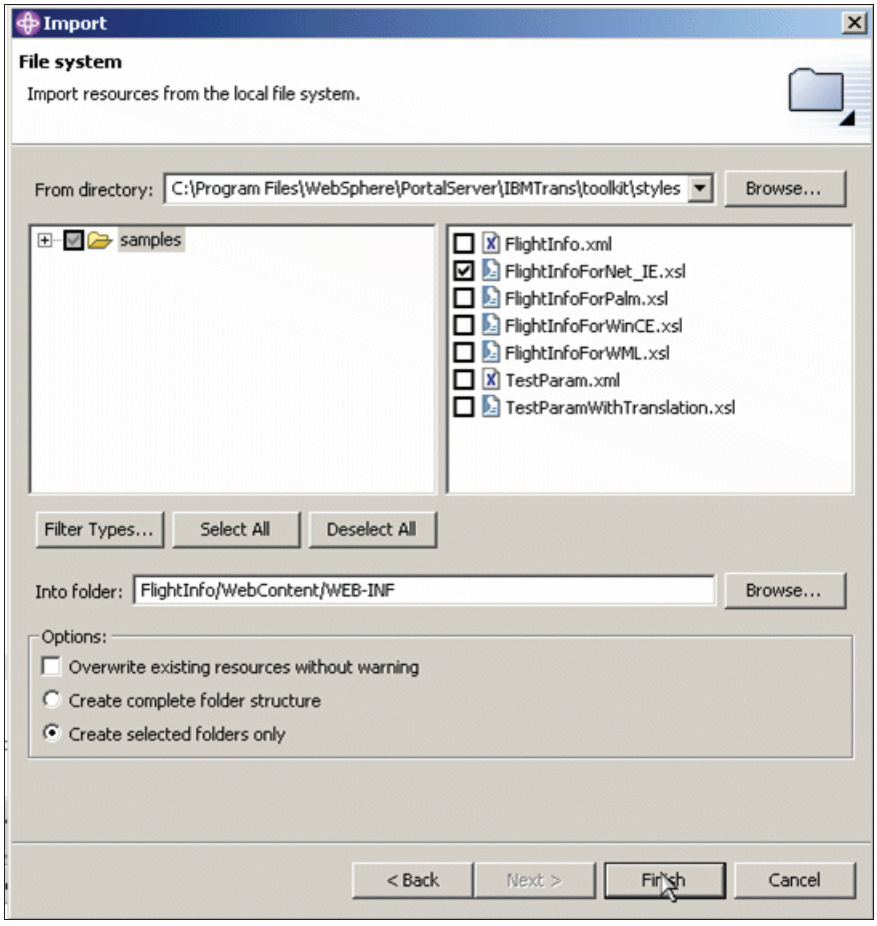


Figure 9-47 Importing resources

5. The **FlightInfoForNet_IE.xsl** is now imported.

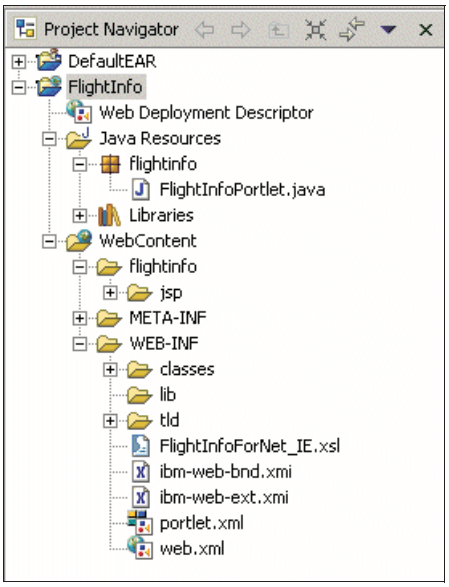


Figure 9-48 Portlet with imported stylesheet

6. Edit portlet.xml. Delete support for markup html and add support for xml.

```
<supports>
    <markup name="xml">
        <view />
    </markup>
</supports>
```

Figure 9-49 Adding xml support

7. Enter the following parameter/value to the portlet.xml.

Example 9-4

```
<config-param>
    <param-name>StylesheetFile</param-name>
    <param-value>WEB-INF/FlightInfoForNet_IE.xsl</param-value>
</config-param>
<config-param>
    <param-name>XML_URL</param-name>
    <param-value> http://home.nc.rr.com/sungik/FlightInfo.xml</param-value>
</config-param>
<config-param>
    <param-name>FilterChain</param-name>
    <param-value>Transcoding</param-value>
```

```
</config-param>
```

8. Double-click FlightInfoPortlet.java.

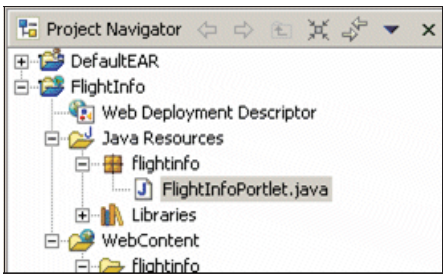


Figure 9-50 FlightInfoPortlet.java

1. Replace the content with the sample code shown in Example 9-5.

Example 9-5 Sample code

```
package flightinfo;

import java.io.IOException;
import java.util.*;
import java.io.*;
import java.io.PrintWriter;
import java.net.*;
import org.apache.jetspeed.portlet.*;


/**
 *
 * A sample portlet based on PortletAdapter
 *
 */
public class FlightInfoPortlet extends PortletAdapter {
   private String stylesheetURL = null;
   private String contentURL = null;

   /**
    * @see org.apache.jetspeed.portlet.Portlet#init(PortletConfig)
    */
   public void init(PortletConfig portletConfig) throws UnavailableException {
      super.init(portletConfig);

   }

   /**
```

```
    * @see org.apache.jetspeed.portlet.PortletAdapter#doView(PortletRequest,
PortletResponse)
    */
   public void doView(PortletRequest request, PortletResponse response) throws
PortletException, IOException {
       PrintWriter writer;
       URLConnection connection;
       URLConnection XSLconnection;
       String line;

       PortletSettings settings = request.getPortletSettings();
           contentURL = settings.getAttribute("XML_URL");

        try {
              writer = response.getWriter();
              connection = new
                                  URL(contentURL).openConnection();
              BufferedReader xmlreader = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
              while ((line=xmlreader.readLine()) != null) {
                     writer.println(line);
                 }
          }
       catch (Exception ioe)
         {
              System.out.println("Error - access the site");
         }

   }

}
```

2. Export the portlet.
3. To use XSL transformation with Transcoding plugins, you must open the PortletFilterService.properties file with a text editor. The file is located within the application server directory at <portalserver>/shared/app/config/services/. After the existing entries within the filtername1 = Transcoding (there are currently three), add the entries shown in Figure 9-51.

```
#-- Transcoding --#
filtername1 = Transcoding
Transcoding.classname = com.ibm.transform.wps.portletfilter.TranscodingPortletFilter
Transcoding.transcodeMarkup.1 = html->wml
Transcoding.transcodeMarkup.2 = html->chtml
Transcoding.transcodeMarkup.3 = html->vxml
Transcoding.transcodeMarkup.4 = xml->html
Transcoding.transcodeMarkup.5 = xml->wml
Transcoding.transcodeMarkup.6 = xml->chtml
```

Figure 9-51 PortletFilterService.properties file

4. You must include XML in the markup languages supported by the portal. Logon as a Portal Administration. Click **Administration**.



Figure 9-52 Administration

5. Click **Portal Settings**.

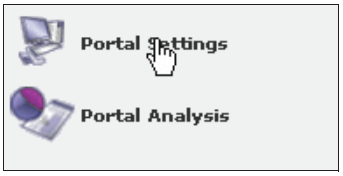


Figure 9-53 Portal settings

6. Click **Supported Markups**.



Figure 9-54 Supported markups

7. Click **Add new markup**.

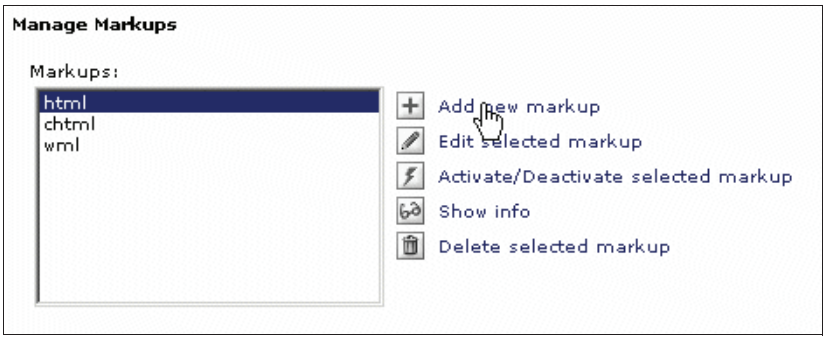


Figure 9-55 Add new markup

8. Enter **xml** for markup name, **text/xml** for MIME Type, and **UTF-8** for Default character set. Click **OK**.

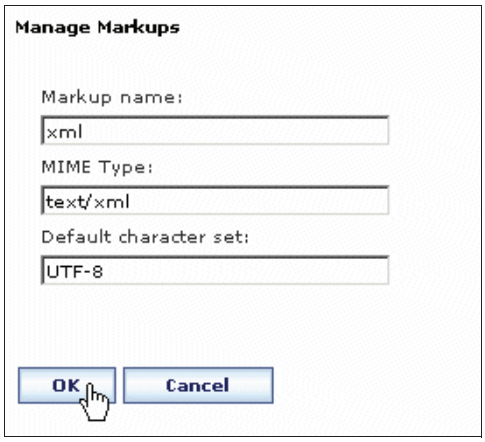


Figure 9-56 Manage markups

9. After a successful markup creation, click **Active/Deactive selected markup**.

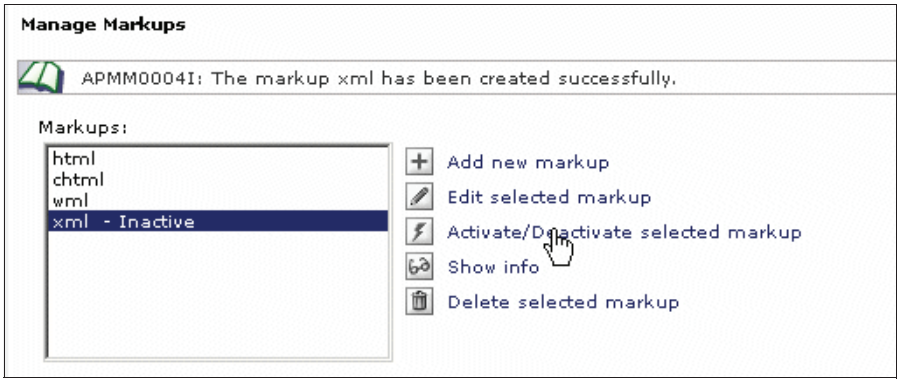


Figure 9-57 Activate selected markup

10. Install the FlightInfo portlet.

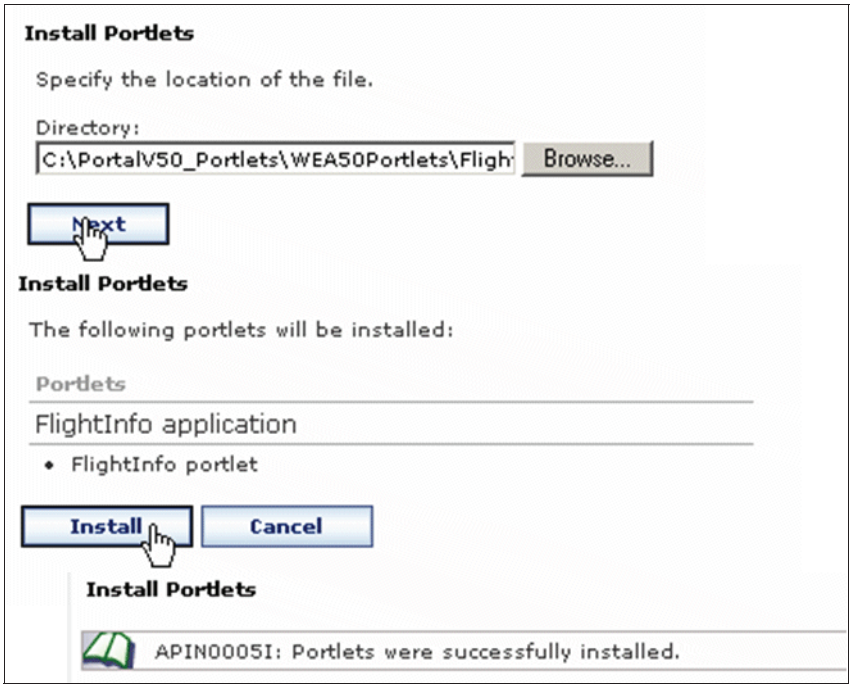


Figure 9-58 Installing the sample portlet

11. Click **My Portal**.
12. Click XML page or any other page where you want to deploy the portlet.
13. Click **Edit Page**.

14. Add FlightInfo portlet. Click **Done**.
15. You will now see the HTML output from the XML after applying the XSL stylesheet.

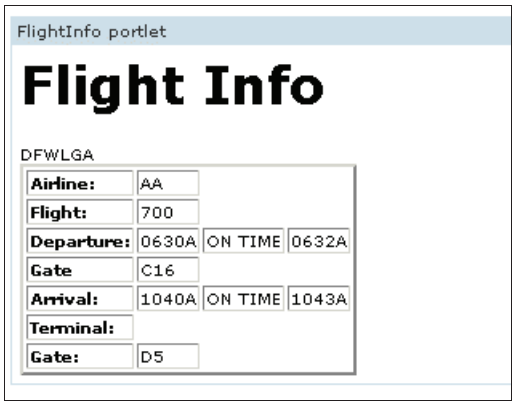


Figure 9-59 Flight Info portlet

16. In this portlet, the source XML file is stored at:

    `http://home.nc.rr.com/sungik/FlightInfo.xml`

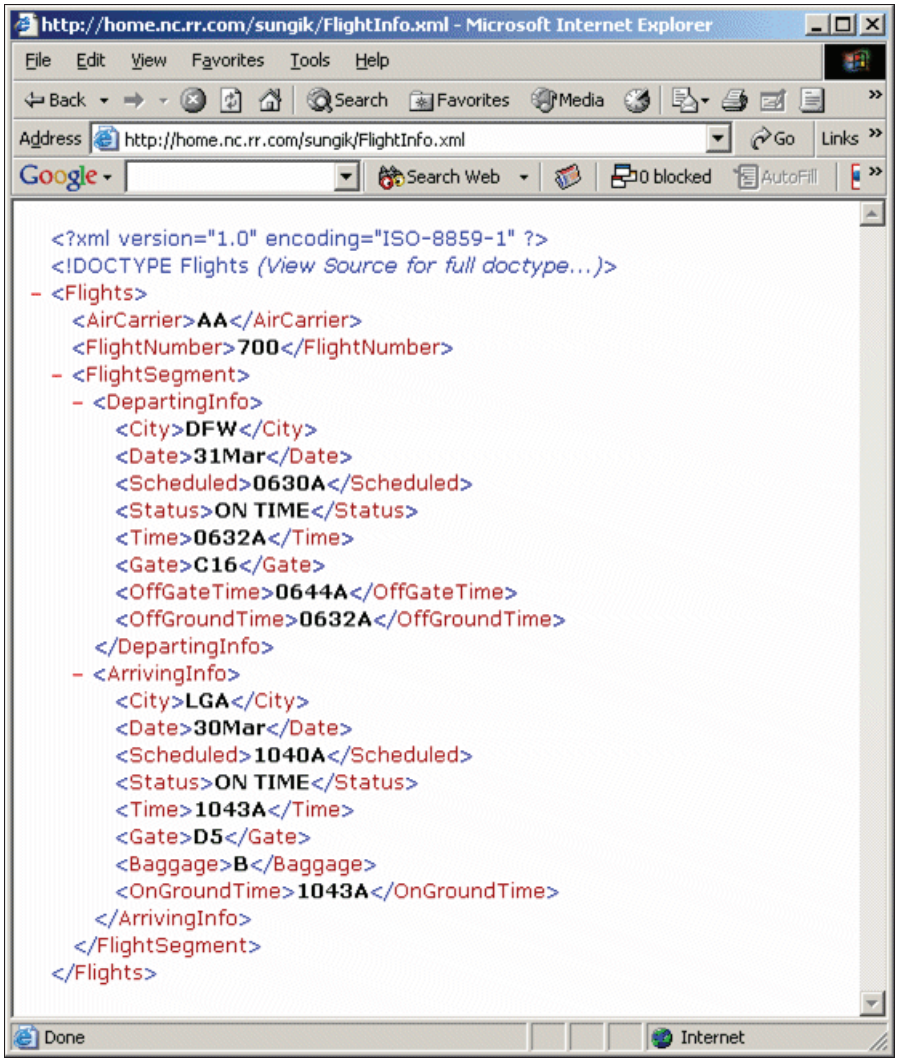


Figure 9-60 XML source code

17. To Change the source XML URL, go to **Manage Portlets** -> **Modify Parameters**.

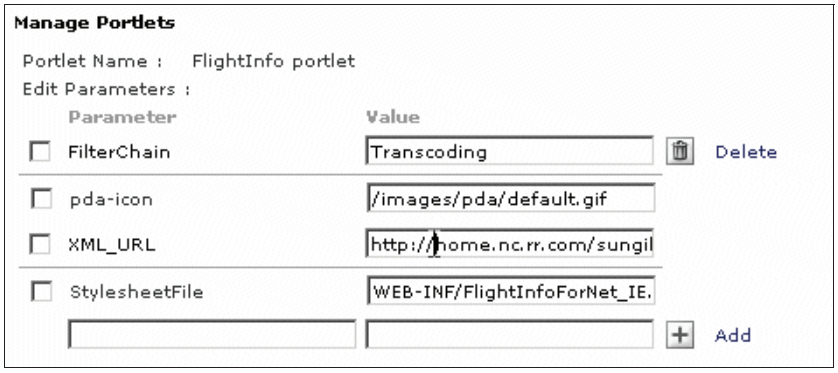


Figure 9-61 Portlet parameters

18. Check XML_URL parameter and enter your Web server location where you stored FlightInfo.xml.

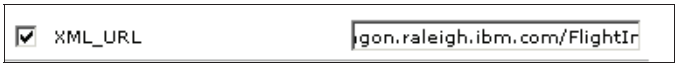


Figure 9-62 Updating the sample XML_URL location

19. Click **Save**.
20. You should obtain the same results.

Part 3

Offline applications

In this part of the redbook we introduce offline application development in WebSphere Everyplace Access V5. Offline applications in WebSphere Everyplace Access run in mobile devices such as PDAs and use synchronization procedures to interact with application servers.

© Copyright IBM Corp. 2005. All rights reserved.

10

Building a portlet for offline browsing

This chapter provides an overview about how to build a portlet that can be browsed offline from PDAs such as Pocket PCs, Palm devices, and others.

The following tasks are described in this chapter:

1. Build a portlet to fetch content from a Web site
2. Use a transcoding annotator plugin for Web clipping
3. Export the portlet to your local file system
4. Deploy and configure the portlet for offline access
5. Running the portlet online
6. Configure WebSphere Everyplace Client for offline access
7. Running offline browsing on a PDA

© Copyright IBM Corp. 2005. All rights reserved.

10.1 Build a portlet to fetch content from a Web site

This section shows how the offline portlet example and wizards can be used to build a form based portlet. The portlet can be used offline using the offline forms capability of WebSphere Everyplace Access.

The sample scenario included in this chapter illustrates a portlet that will access the content of the current IBM stock price from the Web site at `http://www.ibm.com/ibm/stock`. Web clipping will be applied by using an external annotator to fit the necessary information in a PDA device. After building and running the portlet in a test environment, the portlet will be deployed and configured to the WebSphere Everyplace Access server to be offline capable. Finally the portlet will be executed using a PDA with WebSphere Everyplace Client client installed.

The following software is required to successfully run the sample scenario described in this chapter:

- WebSphere Studio Site Developer or WebSphere Studio Application Developer Version 5.1.1
- WebSphere Everyplace Toolkit Version 5.0.1
- WebSphere Everyplace Access Version 5.0
- WebSphere Everyplace Client Version 5.0

In this section you will be shown how to use the toolkit to build and run a portlet that you can later enable for offline browsing. The portlet will access and fetch content from a Web site. It is assumed that WebSphere Studio has been started already.

1. The first step is to launch the portlet development wizard. Inside WebSphere Studio select the menu option **File** –> **New** -> **Portlet Application Project**.
2. For example, enter **PDA_Annotate1** for Project name entry. Select **Create basic portlet**. Check **Configure advanced options**. Click **Next >**.
3. Change EAR Project. Click **New…** button for EAR project and enter **OfflineBrowsing** for Project name entry. Click **Finish**.
4. Check **J2EE Level 1.3/WebSphere Portal 5.0**. Click **Next** button.
5. Click **Next**.
6. Uncheck **Add action listener**. Click **Next**
7. Click **Next** and take J2EE default settings.
8. Check **Add pda markup support** as we will use PDA for browsing.
9. Click **Finish** to generate the portlet application project.

10. Double-click **PDA_Annotate1Portlet.java** for edit.



Figure 10-1 Edit generated portlet code

11. Replace the existing content with the portlet shown in Example 10-1. The portlet connects to www.ibm.com® to fetch the IBM stock value.
12. Save the new code.

Example 10-1 Accessing IBM stock from a portlet

```
package pda_annotate1;
import java.io.IOException;
import java.util.*;
import java.io.*;
import java.io.PrintWriter;
import java.net.*;
import org.apache.jetspeed.portlet.*;
import org.apache.jetspeed.portlet.event.*;
/**
 *
 * A sample portlet based on PortletAdapter
 *
 */
public class PDA_Annotate1Portlet extends PortletAdapter {

   public static final String contentURL = "http://www.ibm.com/ibm/stock/";

   /**
    * @see org.apache.jetspeed.portlet.Portlet#init(PortletConfig)
    */
   public void init(PortletConfig portletConfig) throws UnavailableException {
      super.init(portletConfig);
   }

   /**
    * @see org.apache.jetspeed.portlet.PortletAdapter#doView(PortletRequest,
PortletResponse)
    */
   public void doView(PortletRequest request, PortletResponse response) throws
PortletException, IOException {
```

```
      URLConnection connection;
      PrintWriter writer;
      String line;
      try {
         writer = response.getWriter();
         connection = new URL(contentURL).openConnection();
         BufferedReader xmlreader =
            new BufferedReader(new
InputStreamReader(connection.getInputStream()));
         while ((line=xmlreader.readLine()) != null)
            writer.println(line);

         }
      catch (Exception ioe)
        {
        }
   }
}
```

13. Once the portlet is created you can run and test the portlet. Select the **PDA_Annotate1** project and right-click. From the drop list select **Run on Server...**.
14. If needed, select a proper Portal V5.0 Test Environment server to run the portlet. In this scenario the name of the Portal Test Environment is *Offline Browsing* but any name can be used.
15. You will see the portlet execution and the internal browser will display the IBM Stock site content as illustrated in Figure 10-2.

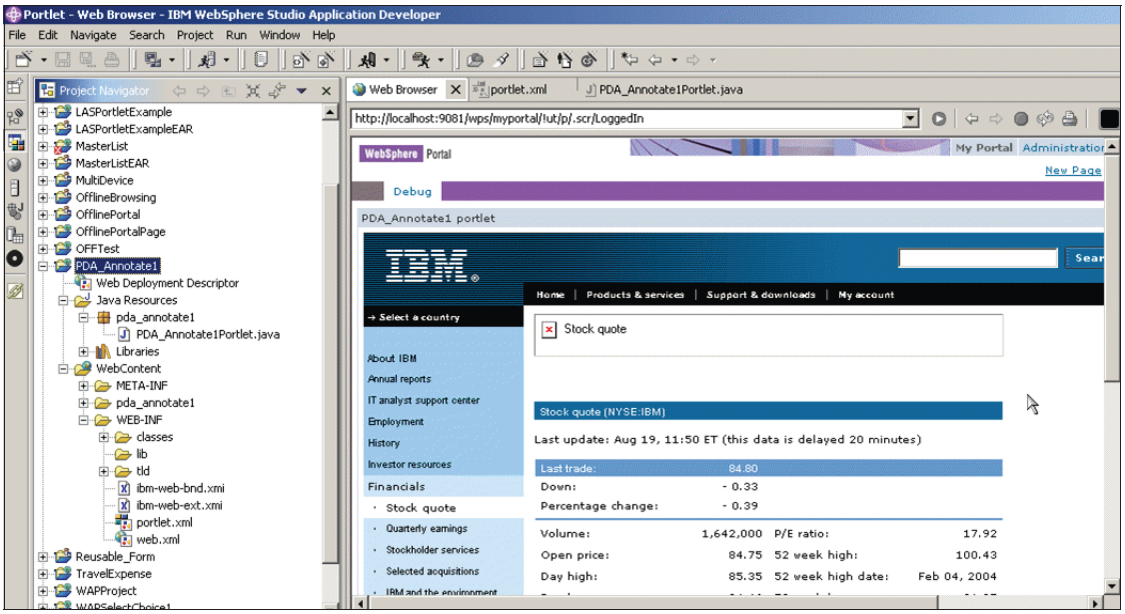


Figure 10-2 Executing the portlet in the Portal Test Environment

10.2 Using a transcoding annotator plugin for Web clipping

In this section you will be given an external annotator to clip the rectangular portion of the page content from the IBM stock site. The content to be clipped is shown in Figure 10-3.

Note: You cannot test Web clipping using the Portal Test Environment since this function is not supported. Therefore, you will need to test the clipping function using a WebSphere Everyplace Access remote server.

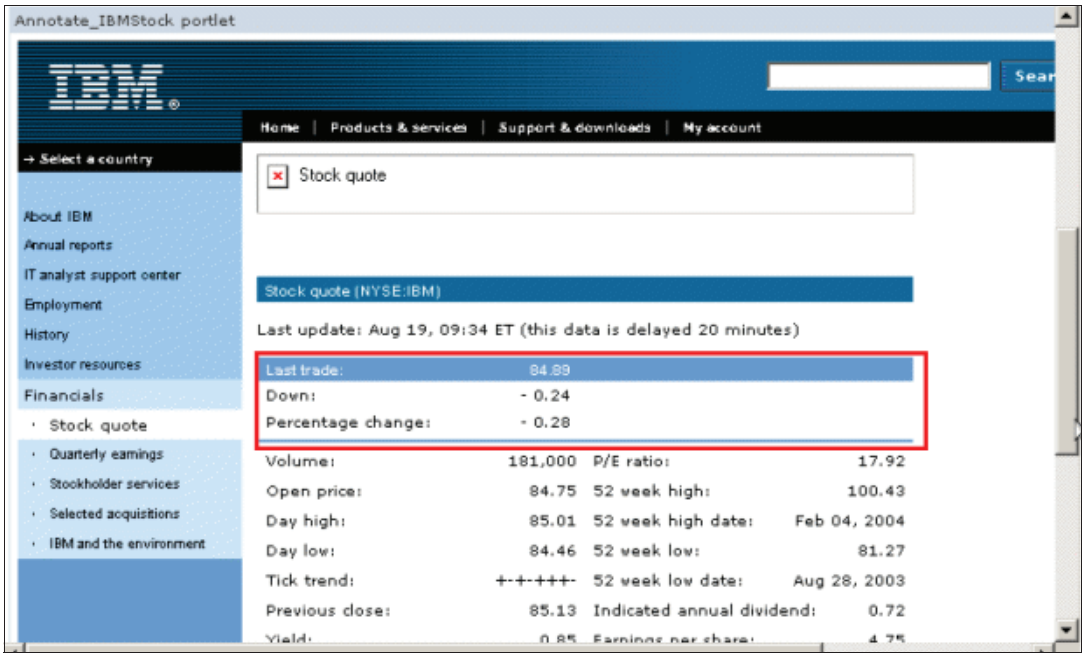


Figure 10-3 Clipping a portion of the page

Execute the following steps to import and apply the external annotator to your portlet application.

1. Import the external annotator. Once you create the external annotator, it will need to be imported into your project. In this scenario the external annotator for the page shown in Figure 10-3 is called ibmstock.ann and it is illustrated in Example 10-2.

Example 10-2 External annotator (ibmstock.ann)

```
<?xml version='2.0' ?>
<annot version="2.0">
    <description take-effect="before" target="/HTML[1]/BODY[1]/*[1]">
        <remove />
    </description>
    <description take-effect="before" target="/descendant::IMG[1]">
        <keep />
        <replace>
      <text>IBM Stock Quote</text>
   </replace>
    </description>
    <description take-effect="after" target="/descendant::IMG[1]">
        <remove />
```

```
    </description>
    <description take-effect="before"
target="/descendant::text()[contains(.,'Last trade:')]/ancestor::TABLE[1]">
        <keep />
        <remove tag="IMG"/>
    </description>
    <description take-effect="after"
target="/descendant::text()[contains(.,'Last trade:')]/ancestor::TABLE[1]">
        <remove />
    </description>
</annot>
```

---

2. Execute the following steps to import the external annotator:
   a. Select the **WEB-INF** directory under **PDA_Annotate1** project and click **Import...**.
   b. Select **File system** and click **Next>**.
   c. Click **Browser...** button to navigate ibmstock.ann file from your local file system for from directory entry.
   d. Click **Finish**.
3. As illustrated in Figure 10-4, after importing you should see the external annotator (ibmstock.ann in this sample scenario), under WEB-INF directory.

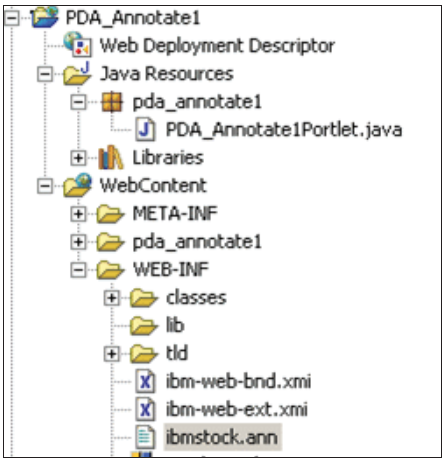


Figure 10-4 Imported external annotator

4. Edit your application portlet descriptor (portlet.xml) and insert the two required configuration parameters shown in Example 10-3. They are:
   – FilterChain, to enable transcoding.

– AnnotationFile, to specify the name of the external annotation file to be applied.

Example 10-3 External annotation required parameters

```
<config-param>
           <param-name>FilterChain</param-name>
           <param-value>Transcoding</param-value>
</config-param>
<config-param>
           <param-name>AnnotationFile</param-name>
           <param-value>WEB-INF/ibmstock.ann</param-value>
</config-param>
```

5. Save all your files.

10.3 Export the portlet to your local file system

Before you deploy the portlet to a WebSphere Everyplace Access server, in this section you will be shown how to export the portlet you have developed.

For example, in WebSphere Studio, execute the following steps:

1. Export the portlet to your local file system. Select **PDA_Annotate1** project and click **Export...**.
2. Select **WAR file** and click the **Next>** button.
3. Click the **Browse...** button to navigate your local directory for destination entry.
4. Click **Finish**.

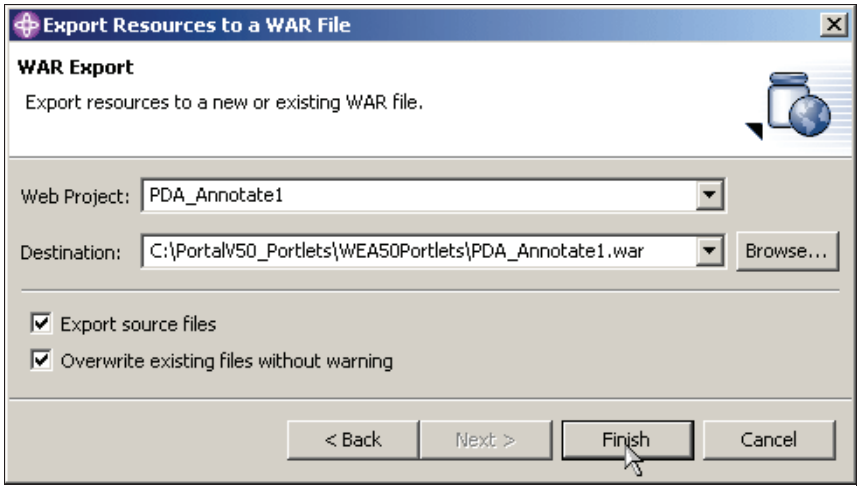


Figure 10-5 Exporting the portlet as a WAR file

10.4 Deploy and configure the portlet for offline access

In this section you will log in to WebSphere Everyplace Access server as administrator, install and deploy the portlet to a WebSphere Everyplace Access server. You will then configure the portlet to support offline capabilities.

Note: If needed you will also need to provide resource permission to the portlet and create a portal page for the portlet be deployed.

Execute the following steps:

1. Logon to WebSphere Everyplace Access server with an administrative user. Click **Administration** tab after a successful login.
2. Click **Portlets** and then click **Install**.
3. Enter the directory where the portlet resides (WAR file). Click **Next**.

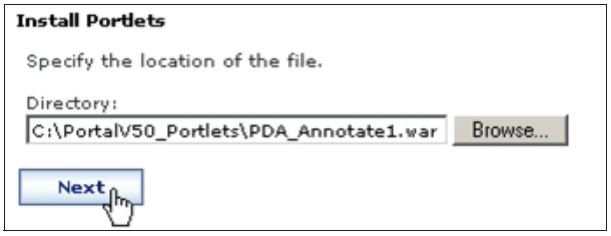


Figure 10-6 Install portlets

4. Click **Install** and make sure the portlet is successfully installed.

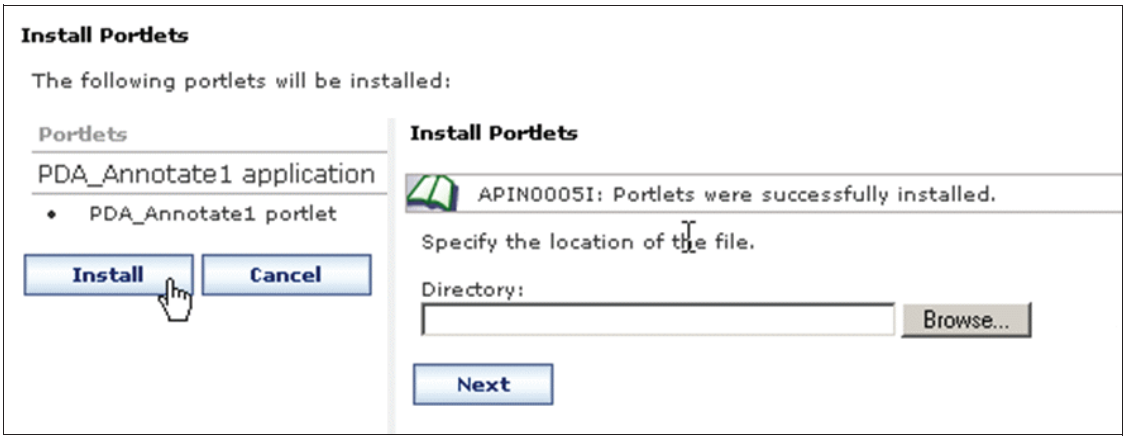


Figure 10-7 Portlet successfully installed

5. Select **Manage Portlets** under **Portlets**.
6. Select **PDA_Annotate1_portlet** and click **Modify parameters**.

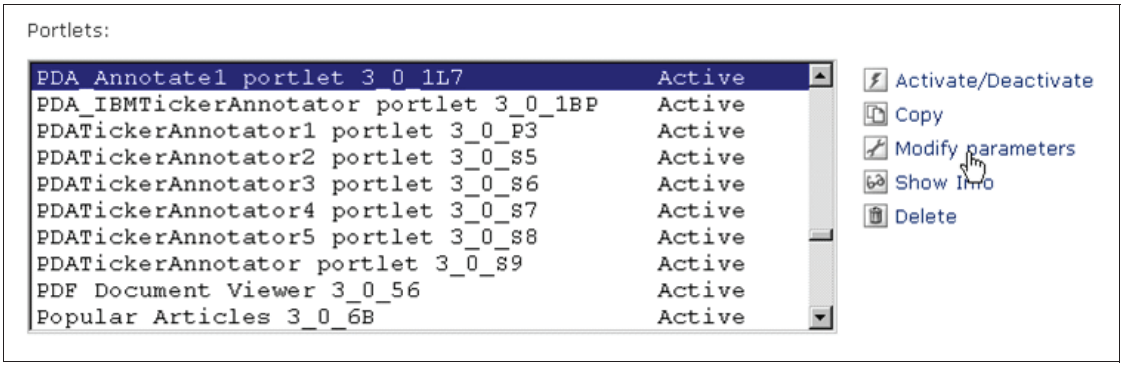


Figure 10-8 Modify portlet parameters

7. Enter the **offline-capable** parameter entry and **true** for its value. Click **Add**.

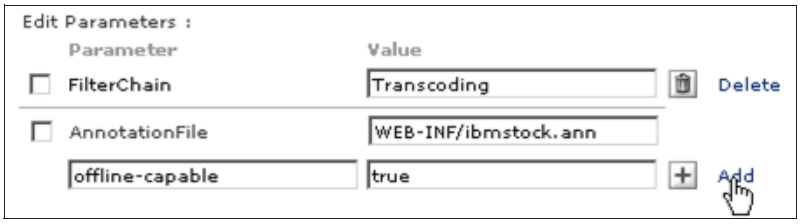


Figure 10-9 Adding the offline-capable parameter

8. Click **Save** button.
9. Click **WebSphere Everyplace** -> **Offline Browsing**.

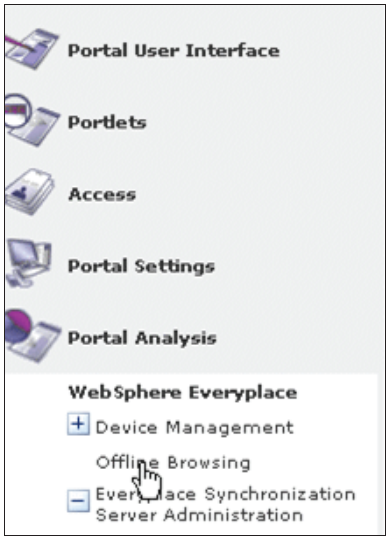


Figure 10-10 Selecting Offline Browsing option

10. Check the portlet **PDA_Annotate1** portlet.



Figure 10-11 Selecting the portlet

11. Click **OK**.
12. You will give resource permission to the portlet so that any authenticated user can view it. Click **Access** -> **Resource Permissions**. Click **Portlets**.

13. Uncheck all for **Allow Inheritance** except Administrator and Security Administrator. Click the pencil mark for privileged User.
14. Click **Add** on Resource Permissions panel.
15. Check all **authenticated portal users** and click **OK**.
16. Click **Done**.
17. Click **OK.**
18. Create a page to deploy the portlet (optional)
19. Click **Manage Pages** under **Portal User Interface**.
20. Click **My Portal**.
21. Click **WEA Home**.
22. lick **New Page**.
23. Enter **Offline Browsing** for title entry. Check **PDA** and click **OK**. Optionally, you can check other markups.
24. Click **OK**.



Figure 10-12 Portlet has been successfully created

25. Now you must see a new page tab **Offline Browsing**. Click **Offline Browsing** page.
26. Click **Edit Page** to add the portlet to the page.
27. Click **Add portlets** button.
28. Check PDA_Annotate1_portlet from the Portlet Titles list.
29. Click **OK**.
30. As shown in Figure 10-13, you will see that the portlet is displayed. Notice that the content is actually clipped using transcoding plugins for annotators.

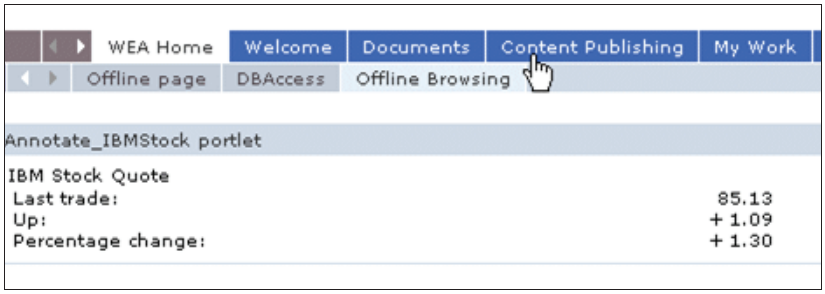


Figure 10-13 Portlet display after text clipping

10.5 Running the portlet online

In this section you will view your portlet online from a Web browser and a PocketPC browser. You will then log in to the WebSphere Everyplace Access server as a user and make the portlet available for offline access.

Execute the following steps:

1. Use your PocketPC to launch `http://<localhost_ip>:9081/wps/portal` where <localhost_ip> can be obtained by executing ipconfig command from Command Prompt window.
2. Login as any valid user to the WebSphere Everyplace Access server. In order to be a valid user to the WebSphere Everyplace Access server, you must sign up to the WebSphere Everyplace Access server.
3. After a successful login, click **Offline Browsing** page tab.
4. You will see the portlet displaying clipped IBM Stock information.

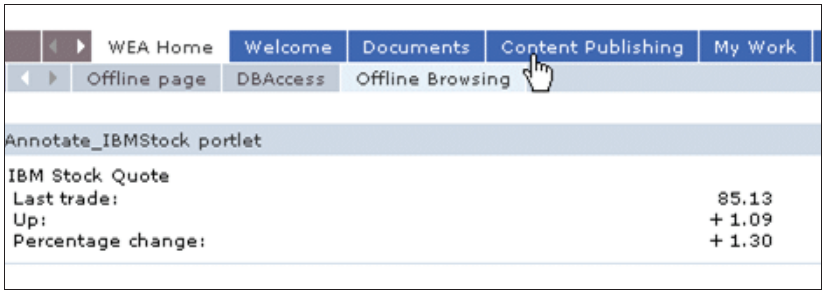


Figure 10-14 Portlet display after clipping

5. Click  next to **My Portal** tab.



Figure 10-15 Configure portlet for offline usage

6. You will find that a new icon appears for the portlet. Click this icon.
7. You will get the message shown in Figure 10-16. Click **OK.**

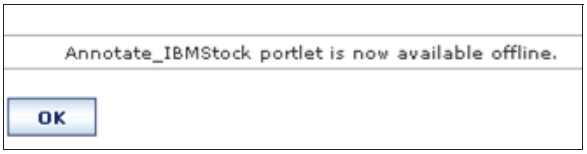


Figure 10-16 Portlet available for offline usage

8. Enter WEA server host name <WEAhost>/wps/portal in your PocketPC to view the portlet using your PocketPC.
9. Login as an user . Enter a proper user ID and password and click **Login**.

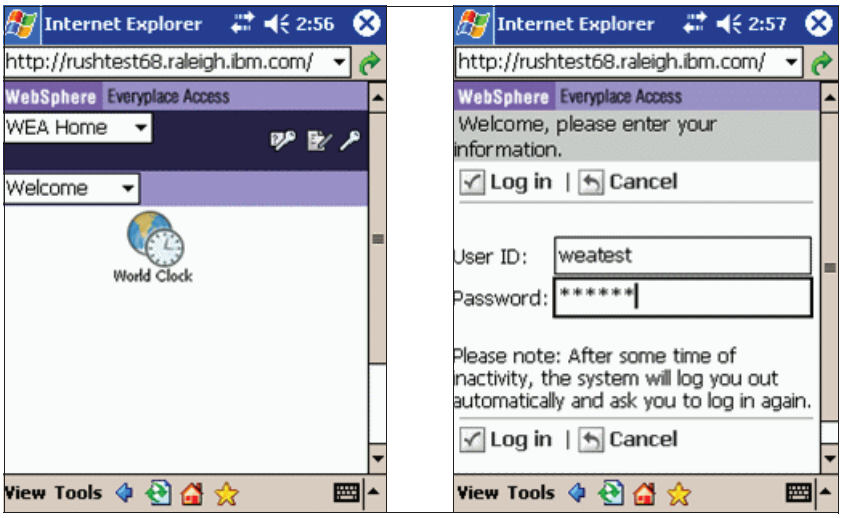


Figure 10-17 Login to WebSphere Everyplace Access

10. Select **Offline Browsing** from the lower combo box.
11. Double-click **PDA_Annotate1 portlet**. You will see the clipped IBM Stock price.

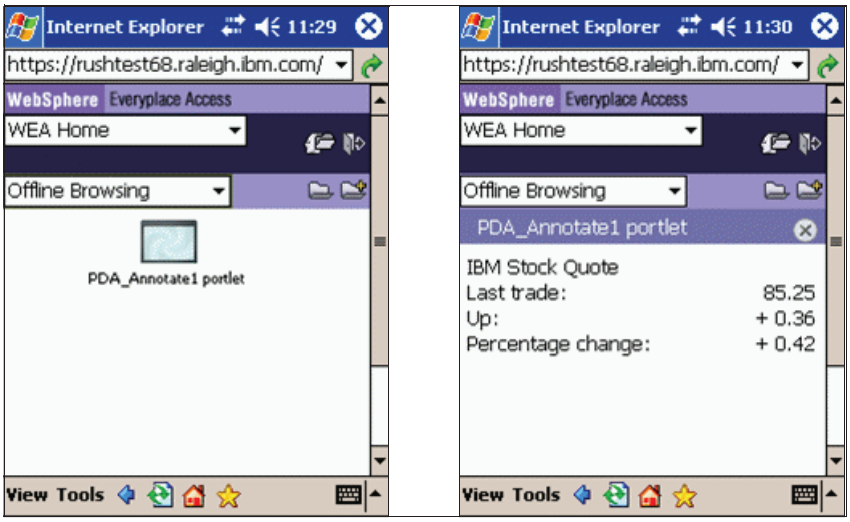


Figure 10-18 Running the portlet online

10.6 Configure Everyplace Client for offline access

In this part you will configure offline administration in the WEA Client. Execute the following steps:

1. Click WEA Client icon 

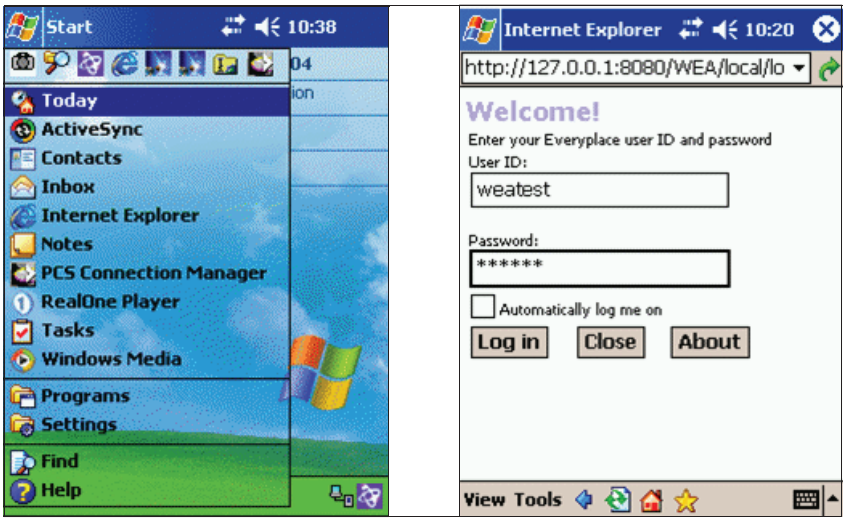


Figure 10-19 Login to WebSphere Everyplace Access

2. Login WEA client as a user you used in Step 5. Click 
3. Click **Network profiles** to configure offline server.

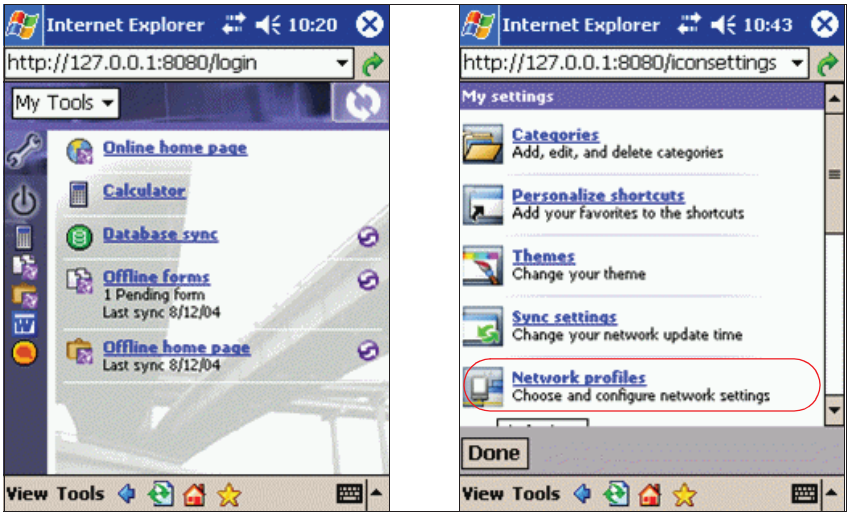


Figure 10-20 Network profiles

4. Enter the server name for Offline Content server entry. Click **OK**. Click **Done**.

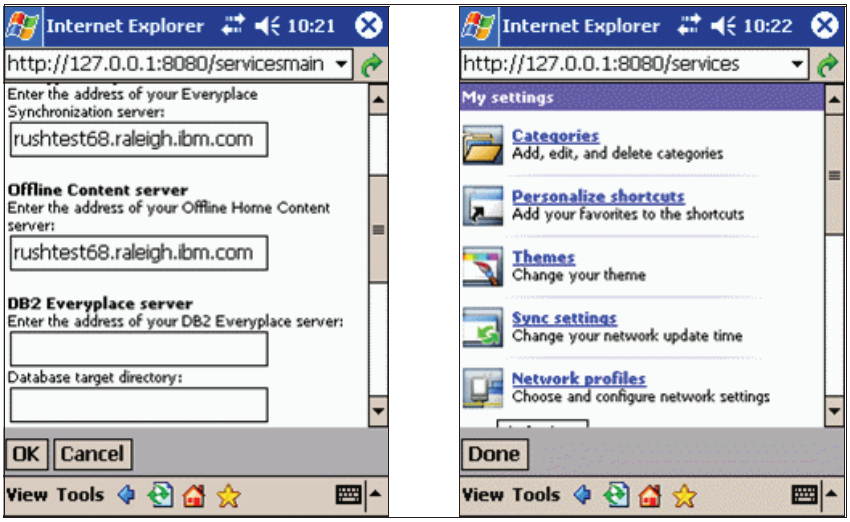


Figure 10-21 Server name for offline content

5. Click  next to **Offline home page.**



Figure 10-22 Synchronization in progress

6. Once the synchronization process has taken place, you will notice that the last synchronization time has been updated.

10.7 Running offline browsing on a PDA

In this section you will run the portlet offline and disconnected from the network. Execute the following steps:

1. Disconnect your Pocket PC from the network.
2. You will notice the icon has been changed to disconnected status 
3. Click **Offline home page**.
4. Click **PDA_Annotate1 portlet**.

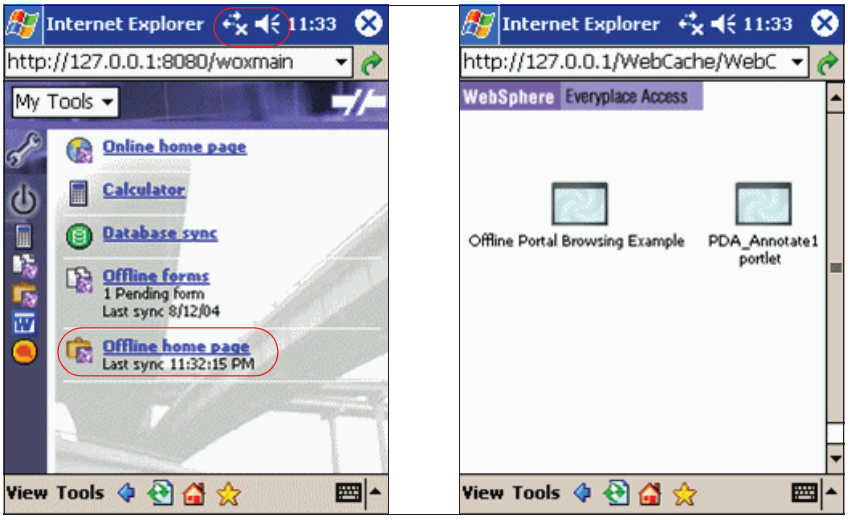


Figure 10-23 Offline home page

5. After some time repeat the process by connecting and disconnecting from the network You will find that the stock value has been updated.

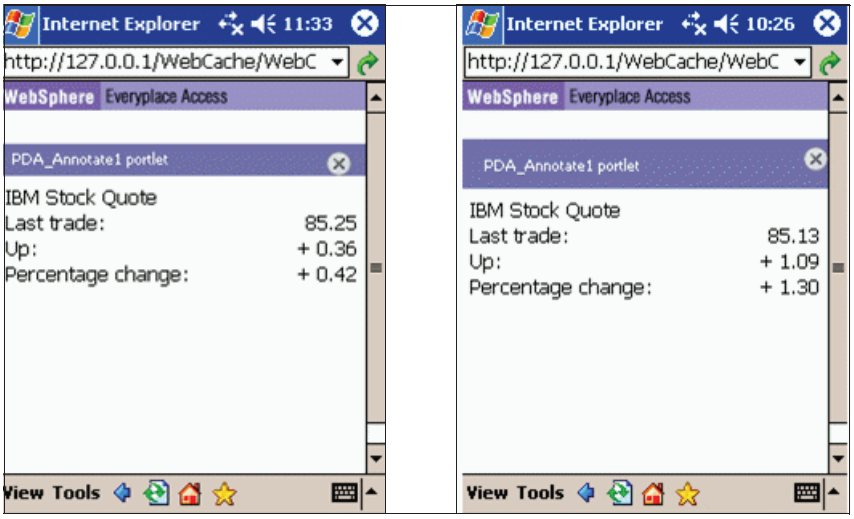


Figure 10-24 Offline browsing

11

Building a portlet for offline form access

This chapter provides an introduction to how the Everyplace Toolkit portlet wizards can be used to build a form portlet, which can then be used for offline submission. The portlet can then be used offline using the offline forms capability of WebSphere Everyplace Server and WebSphere Everyplace Client.

The following tasks are illustrated in this chapter:

1. Building a form based portlet
2. Adding field validations
3. Adding a script to highlight the field in error
4. Deploying the sample portlet as an offline portlet
5. Running the portlet from a Pocket PC
6. Portlet offline configuration
7. Configure the portlet client with offline option

© Copyright IBM Corp. 2005. All rights reserved.

11.1 Building a form based portlet

In this sample scenario you are shown how to use the Toolkit wizards to generate a form suitable for use in offline mode. Additional help and guidance can be found in the online toolkit help available from the Studio Workbench.

The following software is used for this sample scenario included in this chapter:

- WebSphere Studio Site Developer or WebSphere Studio Application Developer V5.1.1
- WebSphere Everyplace Toolkit V5.0.1

In this section you will be shown how to use the toolkit wizards to build a form portlet. WebSphere Studio should have been started. Execute the following steps:

1. The first step is to launch the portlet development wizard. Inside WebSphere Studio select the menu option **File** –> **New** -> **Other**.
2. This will launch a window from which you can select the wizard that you want to use. The window consists of two panels. On the left panel open the branch under **Examples** -> **Portlet Applications**.
3. Click **Mobile**.

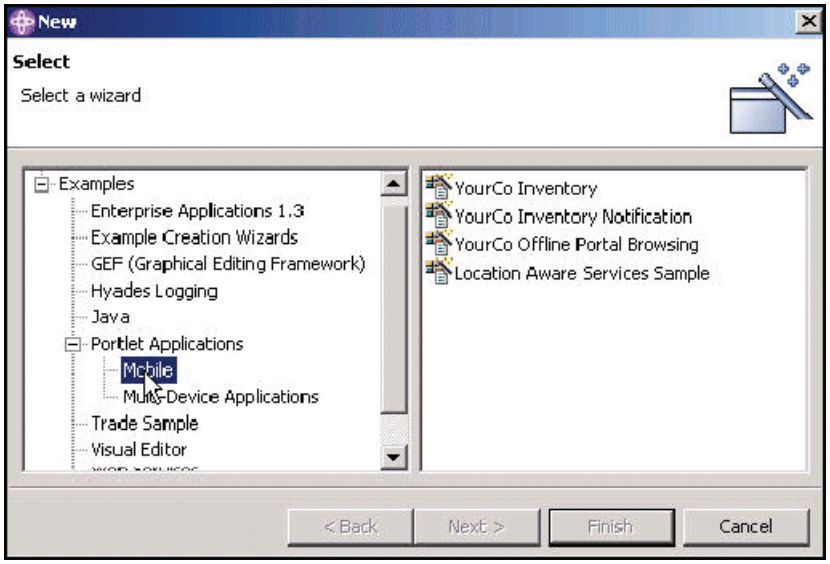


Figure 11-1 Selecting the wizard

4. From the right hand panel, select **YourCo Offline Portal Browsing**. Click **Next.**

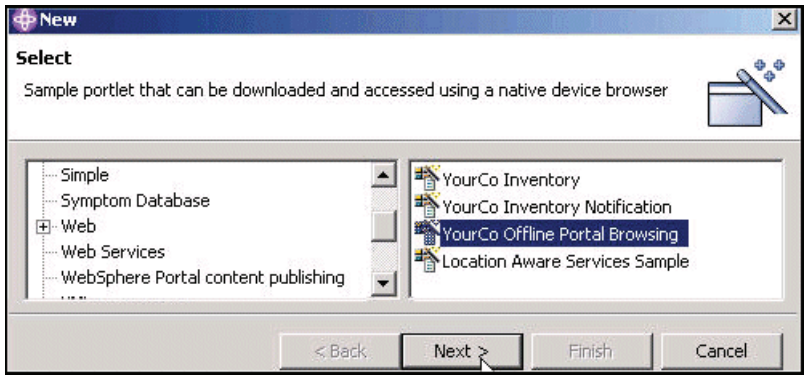


Figure 11-2 Offline browsing application

5. For Figure 11-3, enter a meaningful project name. Click **Finish**.

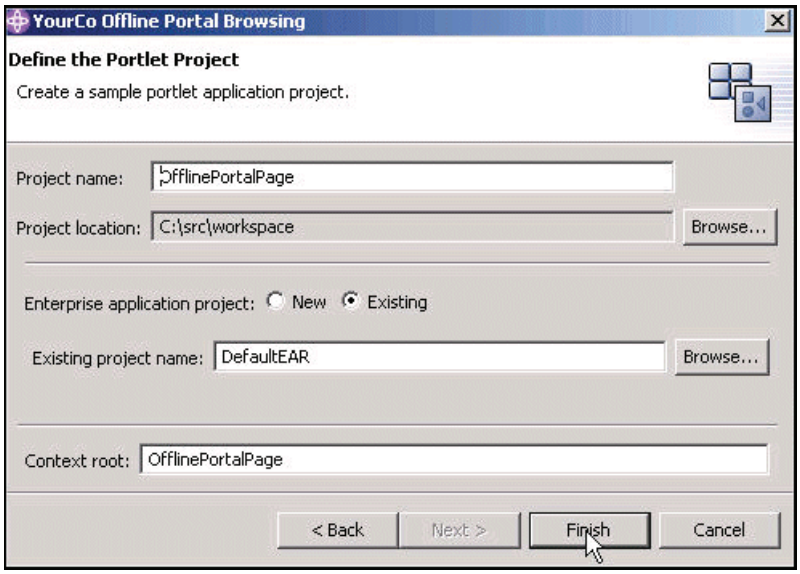


Figure 11-3 Portlet project

6. If at anytime you see a server configuration change window, click **OK** to accept the change.

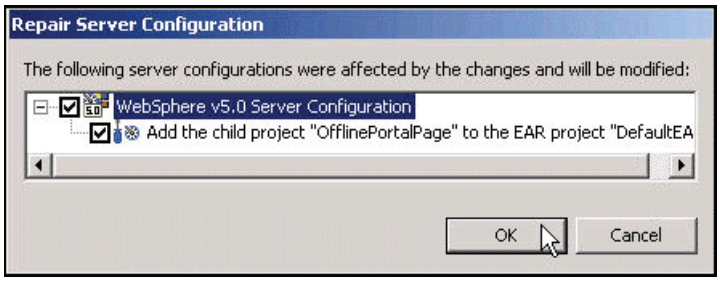


Figure 11-4 Repair server configuration

7. Example code will be generated. This provides a working form portlet that you can use as a base for your form.

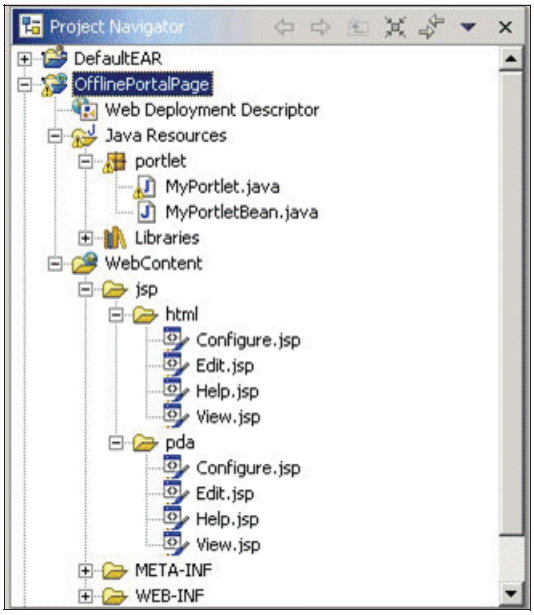


Figure 11-5 Generated sample portlet

8. If you already have configured a test server, you may see some DB2ADMIN and application deployment error messages in the Tasks panel. These are due to a conflict with the existing test server.

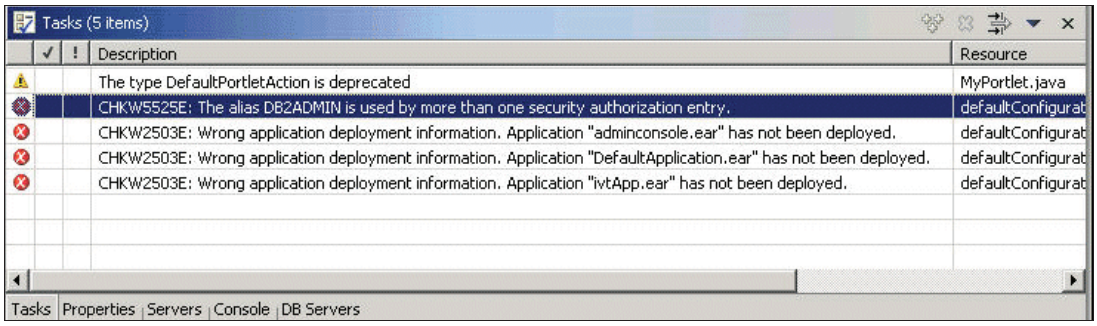


Figure 11-6 Error messages in Tasks panel

9. To remove the error messages, switch to the servers panel.



Figure 11-7 Selecting Server panel

10. Right Click over the server entry and select **Delete** from the context menu.

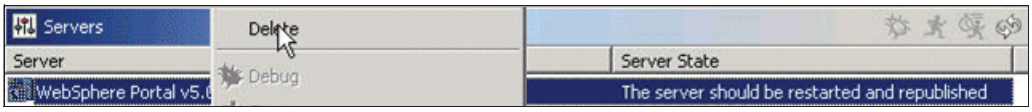


Figure 11-8 Delete the server

11. Confirm the delete by clicking **Yes** on the delete confirmation dialog.
12. Confirm deletion of the Server configuration also, by clicking **Yes** on the next dialog.
13. Open the Tasks panel, and the error message will have been cleared.

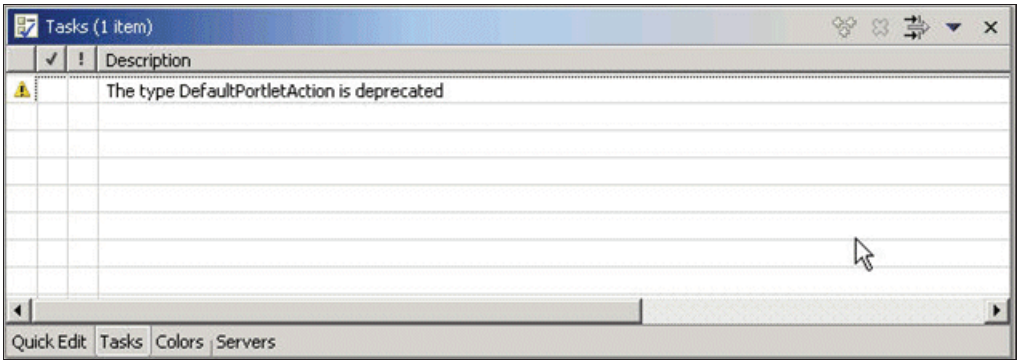


Figure 11-9 Tasks panel

14. Switch back to the Web Perspective
15. Right-click your new form portlet project.
16. From the context menu select **Run on Server.**
17. If there are no portal test environment servers configured, select the **Create a new Server** option.
18. Select **WebSphere Portal v5.0 Test Environment** and click **Next**.

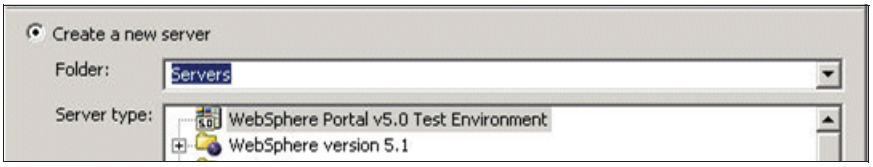


Figure 11-10 WebSphere Portal Test Environment

19. On the next dialog you specify the port. Leave the values as default, but make a note of them. Click **Finish**.
20. If it is not already running, the server will now start up. You will be able to interact with your forms portlet from a browser window.

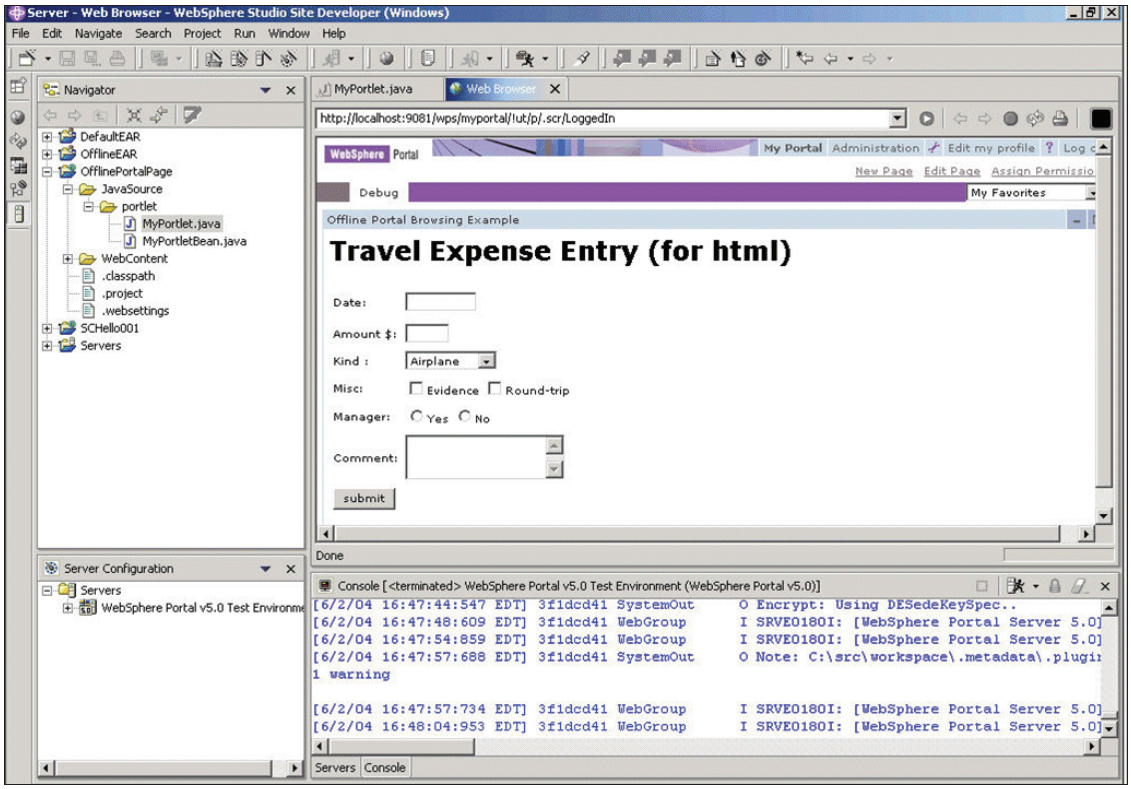


Figure 11-11 Running the portlet

21. Use your PocketPC to launch `http://<localhost_ip>:9081/wps/portal` where <localhost_ip> can be obtained by executing the ipconfig command from the command prompt window.

22. Click 

23. Enter user ID and password wpsadmin/wpsadmin as illustrated in Figure 11-12 and click the offline form portlet.

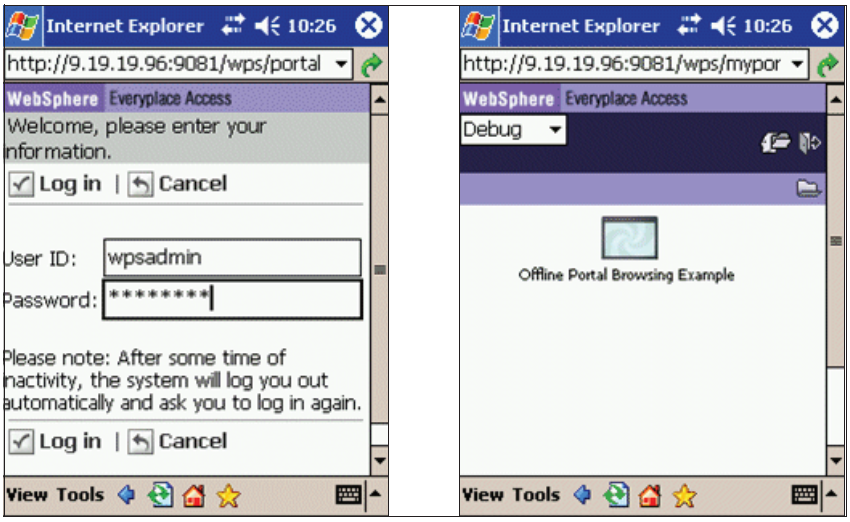


Figure 11-12 Portal login

24. Enter values for the fields and click **submit**.

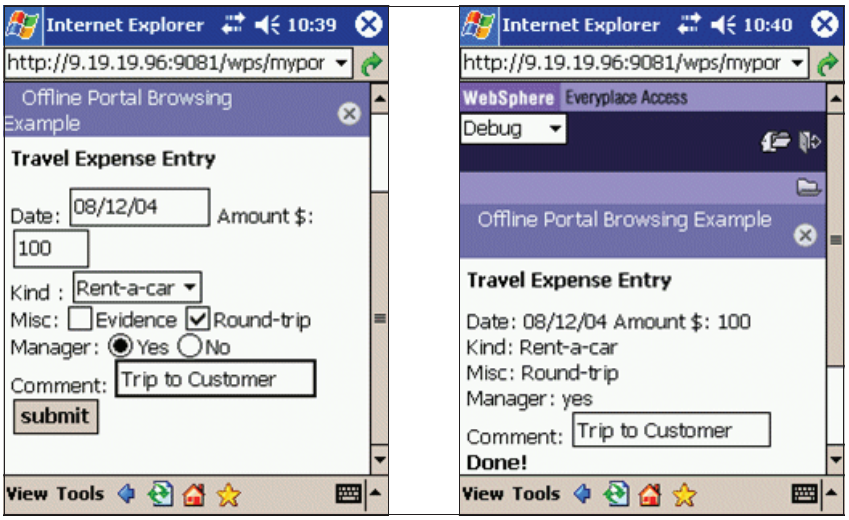


Figure 11-13 Executing the portlet

25. The entry will be accepted.

Note: The submit will be accepted even if the fields are badly entered. For example, enter an alphabetic value for the Amount, the form with the incorrect amount will still be accepted.

11.2 Adding field validations

In this section you will add field validations to the portlet. For example, execute the following steps:

1. Javascript form fields edit needs to be added. Go to the project Navigator panel and open the view.jsp file for the offline portlet.

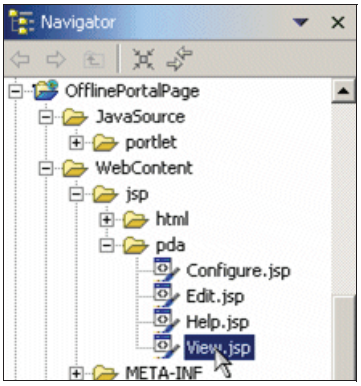


Figure 11-14 Selecting the view mode JavaServer Page

2. In the JSP edit panel click the **Design** view tab.

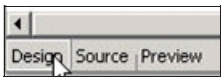


Figure 11-15 Design view

3. In the design view right-click over the amount entry field.
4. From the context menu select **Insert Field Validations** -> **Create New Pattern**.

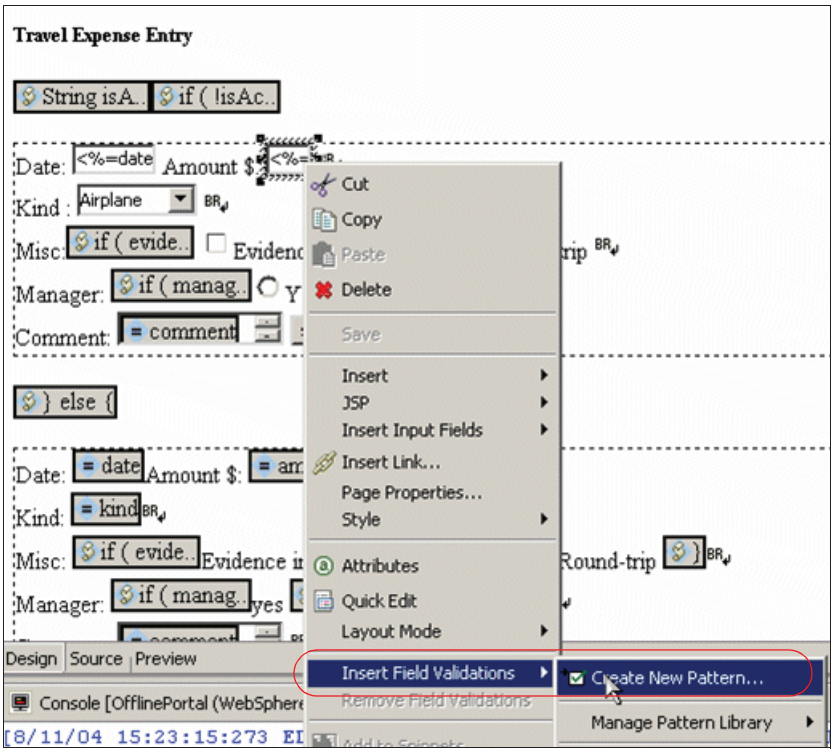


Figure 11-16 Insert field validations

5. A dialog will then show, prompting you for the validation pattern string.
6. A description of pattern formats can be found in the Toolkit help, under **Offline and Reusable Forms**.
7. Enter a pattern **\d+,** for one or more digital characters. Select to Save the pattern and enter a meaningful name for the pattern.
8. Click **OK**.

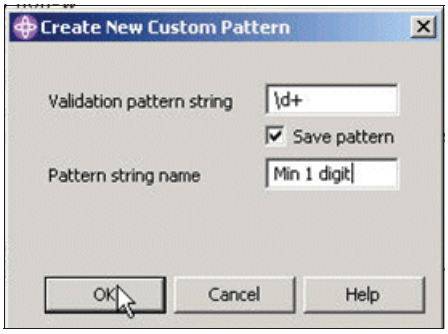


Figure 11-17 Create a new custom pattern

9. If this is the first pattern that you have created, you will be informed that a new library file will be created. Click **OK**.

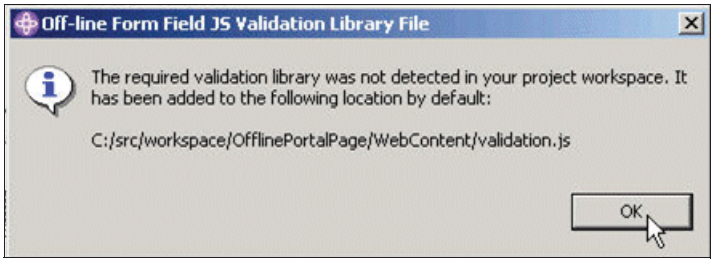


Figure 11-18 Validation library file message

10. In the edit view.jsp panel, click the **Source** tab.

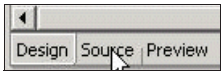


Figure 11-19 Source tab

11. You should see that a new Javascript function has been created. This matches the field with the pattern.

```
<SCRIPT type="text/javascript" src='<%=response.encodeURL("validation.js")%>'></SCRIPT>
<SCRIPT type="text/javascript">
    function validate() {
        var form = document.forms["<portletAPI:encodeNamespace value="OFFLINE_FORMS_TEST"/>"];
        if (!pattern('\\d+',form.elements["amount"].value))
         {
            alert(form.elements["amount"].value + " is an invalid amount.");
            return false;
         }
        return true;
    }
</SCRIPT>
```

Figure 11-20 Generated javascript

12. The form POST method now does a check on the new validate function before posting the request.

```
<form method="POST" name="<portletAPI:encodeNamespace value="OFFLINE_FORMS_TEST"/>"
   action="<%= request.getAttribute ("saveURI") %>" onsubmit="return validate()">
```

Figure 11-21 Form post method

13. Save your changes
14. In the Project Navigator view, right-click over the offline project, In the context menu select **Run on Server**.
15. If you are prompted to save you changes click **OK**
16. Select to run on your pre-generated portal test environment.
17. Fill in the fields, but enter an alphabetic amount.
18. Click **Submit**.

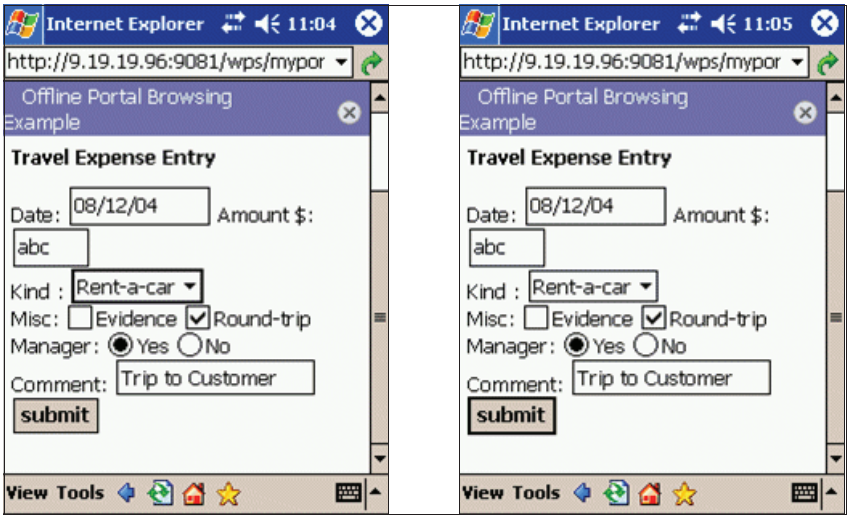


Figure 11-22 Running the portlet with wrong data

19. You should see no action. This is because the form submit has been suppressed, but no error message is set in the JSP.
20. Enter a valid amount and click **Submit.**

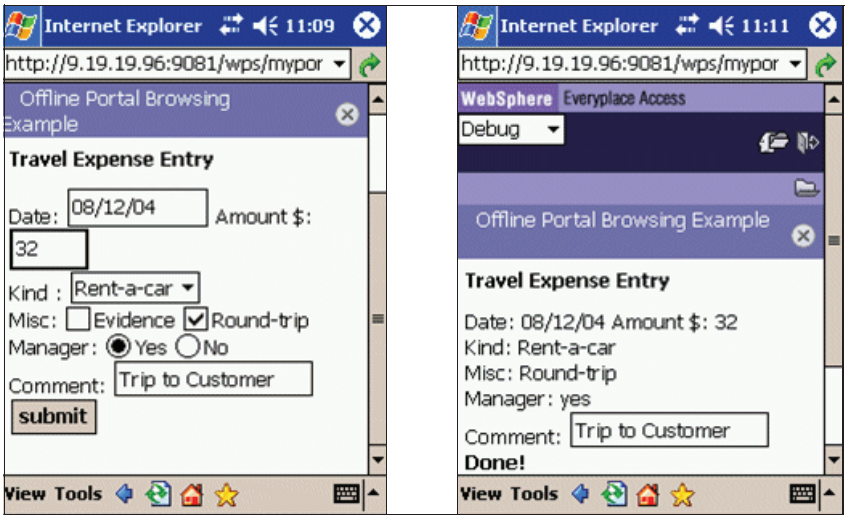


Figure 11-23 Running the portlet with correct data

21. This time the form was successfully submitted. The next stage should be to add a script that will highlight the field in error.

11.3 Adding a script to highlight the field in error

In this part you will modify the Javascript to alert when the amount field is in error. Execute the following steps:

1. Edit the validate Javascript function to pop up an alert when there is an invalid amount.

```
<SCRIPT type="text/javascript">
    function validate() {
        var form = document.forms["<portletAPI:encodeNamespace value="OFFLINE_FORMS_TEST"/>"];
        if (!pattern('\\d+',form.elements["amount"].value))
         {
            alert(form.elements["amount"].value + " is an invalid amount.");
            return false;
         }
        return true;
    }
</SCRIPT>
```

Figure 11-24 Pop up an alert message when field in error

2. Save your changes.
3. Run the new portlet package on the test environment.
4. When the portlet is running, enter an invalid amount and click **submit**. An alert pop-up will display as illustrated in Figure 11-25.

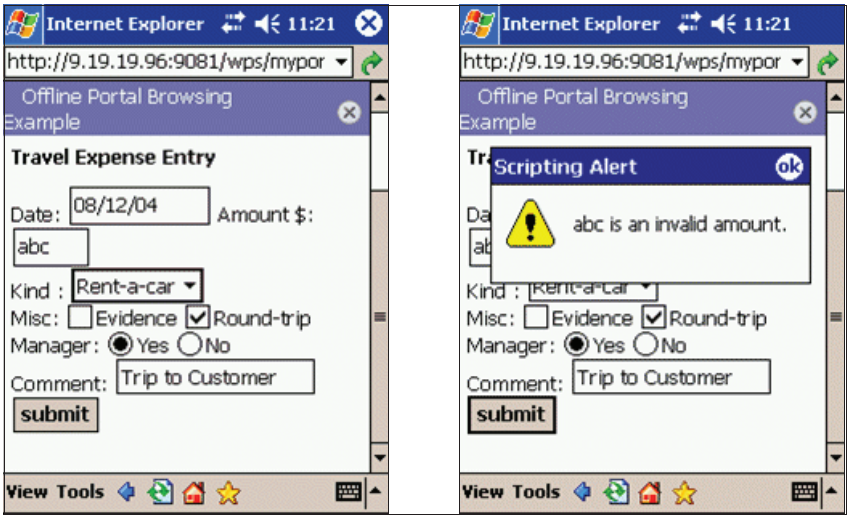


Figure 11-25 Scripting alert

11.4 Deploying the sample portlet as an offline portlet

In this part you will install and run the portlet using your PDA device. Execute the following steps:

1. Export your project as a war file to your local file system. Select your project and then right-click. Select **Export...**
2. Select WAR file and click **Next**.
3. Enter the destination and click **Finish**.

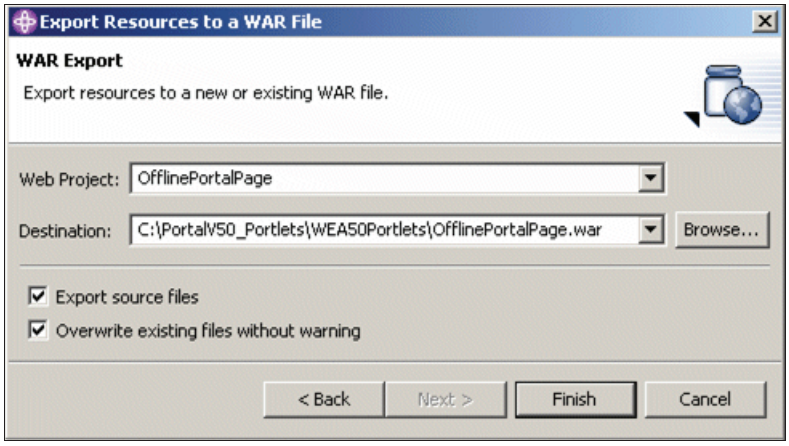


Figure 11-26 Exporting portlet as a WAR file

4. Logon to WebSphere Everyplace Access server with an administrative user. Click **Administration** tab.
5. Click **Portlets** and then click **Install**.

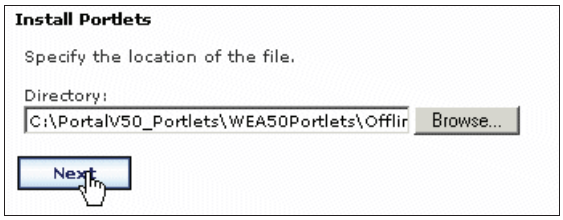


Figure 11-27 Install portlets

6. Enter the portlet location you specified and click **Next**.
7. Click **Install**.

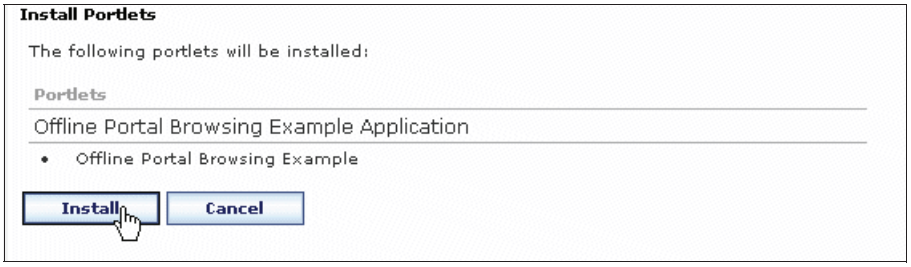


Figure 11-28 Install portlets

8. Make sure the portlet has been successfully installed.

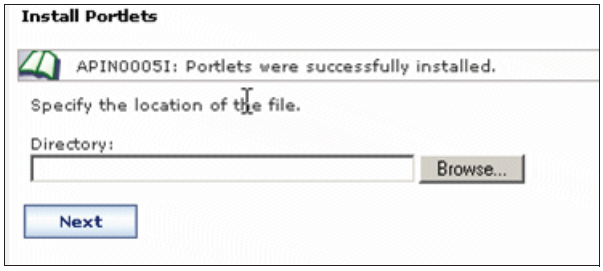


Figure 11-29 Portlet has been installed

9. Select **My Portal** tab
10. Select any target page you want to deploy your portlet and then click **Edit Page**
11. Click **Add Portlets** from the Edit Layout panel.
12. Select **Offline Portal Browsing Example** and click **OK**.

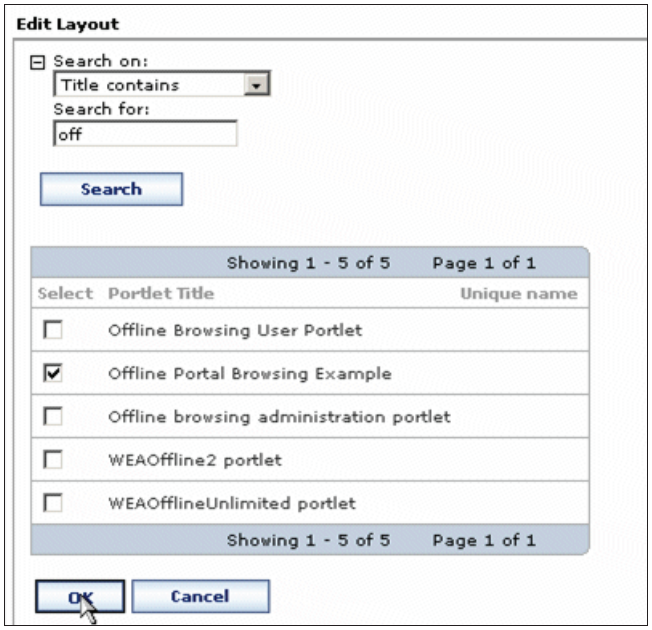


Figure 11-30 Selecting the portlet

13. You will find the portlet is displayed.

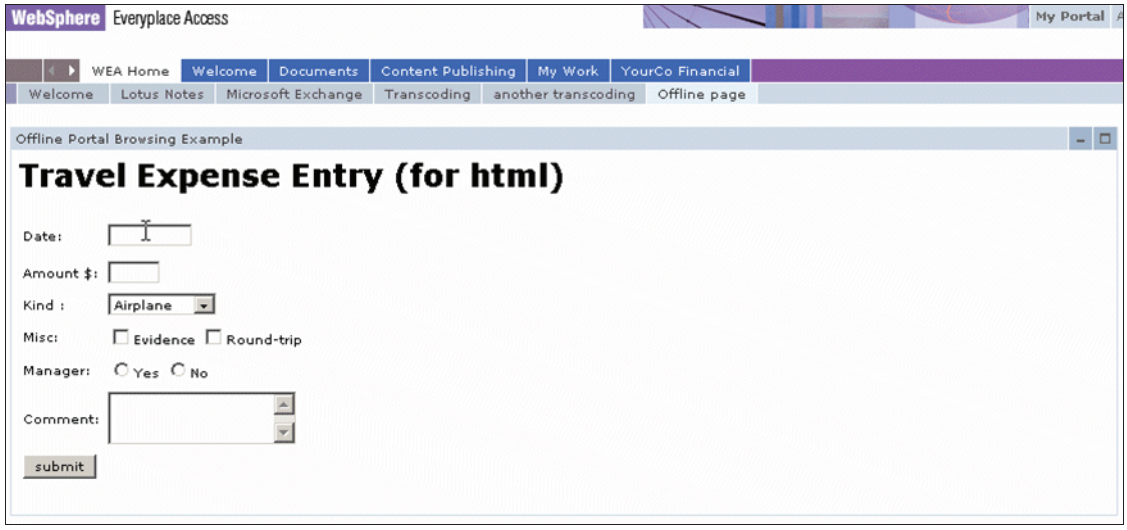


Figure 11-31 Running the portlet

11.5 Running the portlet from a Pocket PC

In this section you will run the form portlet using a PocketPC. Execute the following steps:

1. Enter the WebSphere Everyplace Access server host name. For example <WEAhost>/wps/portal from your PocketPC and login as an administrator.

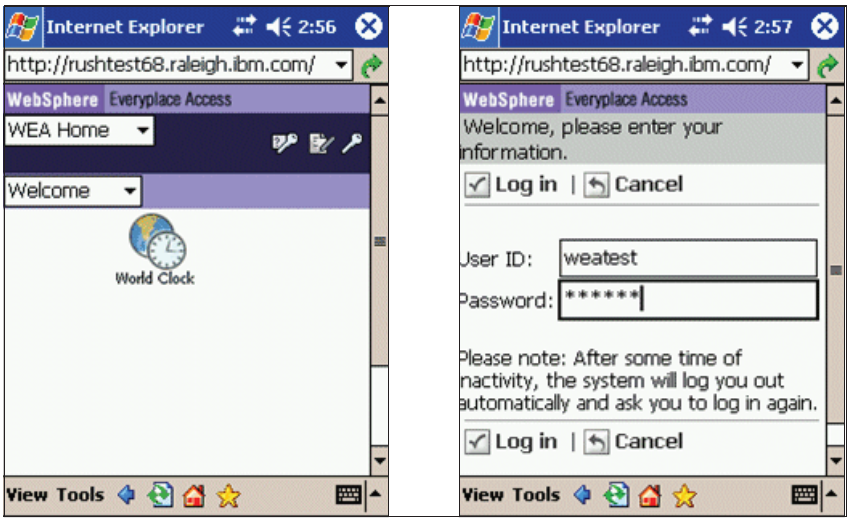


Figure 11-32 Login to WebSphere Everyplace Access

2. Select the page that contains the form portlet and click **Offline Portal Browsing Example** portlet.

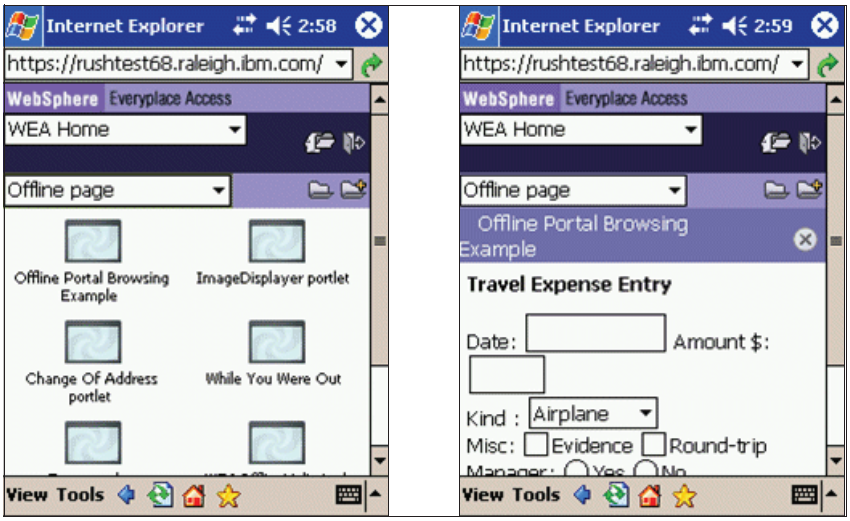


Figure 11-33 Executing the portlet from a Pocket PC

3. Enter an invalid amount and click submit. You will get the error alert.

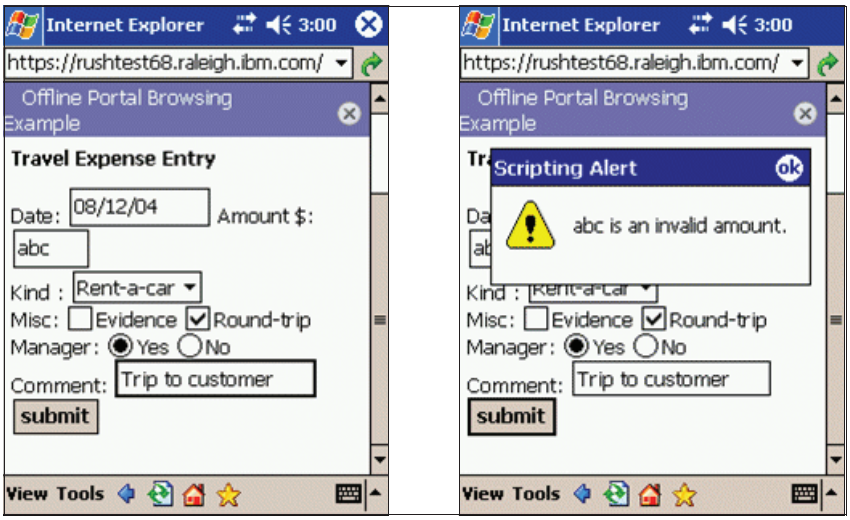


Figure 11-34 Executing the portlet with wrong data

4. Enter the valid digit and the form is successfully submitted.

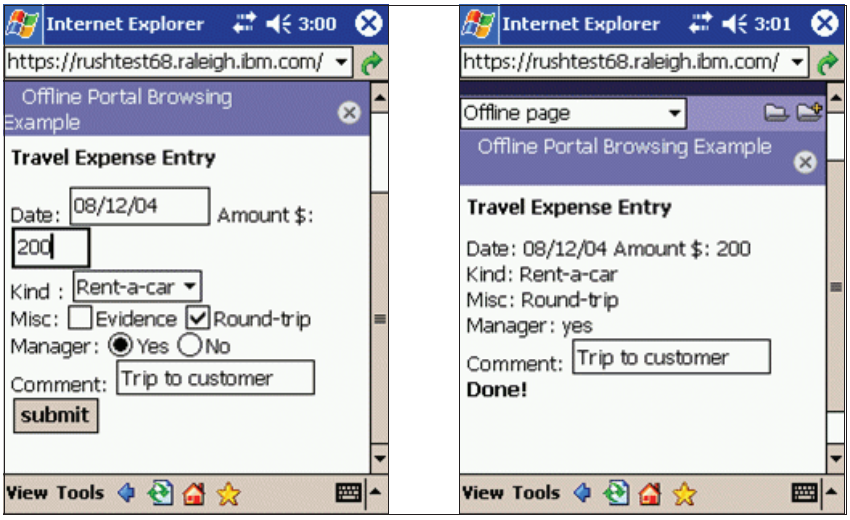


Figure 11-35 Executing the portlet with correct data

11.6 Portlet offline configuration

In this section you will configure the portlet offline capabilities. Execute the following steps:

1. Login to WebSphere Everyplace Access server as an administrator. Click Administration tab and then click **WebSphere Everyplace**.
2. Click **Offline Browsing**.

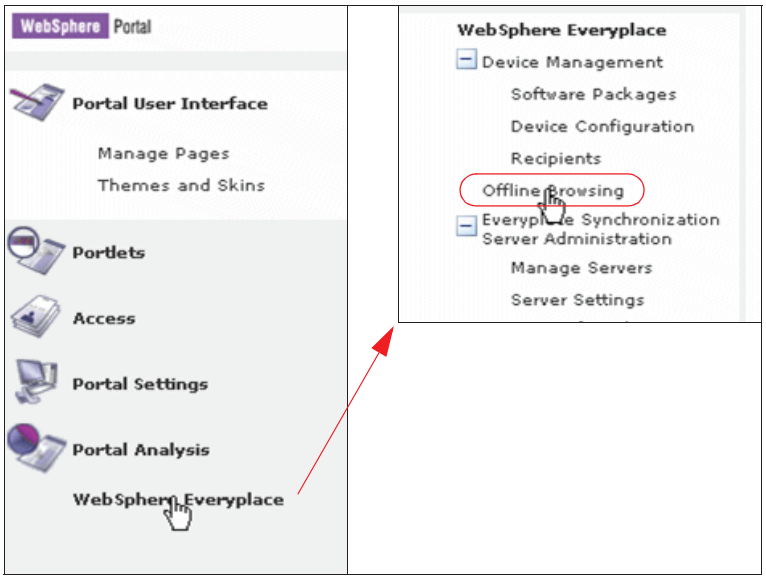


Figure 11-36 Selecting offline browsing

3. Check **Offline Portal Browsing Example** and click **OK**.

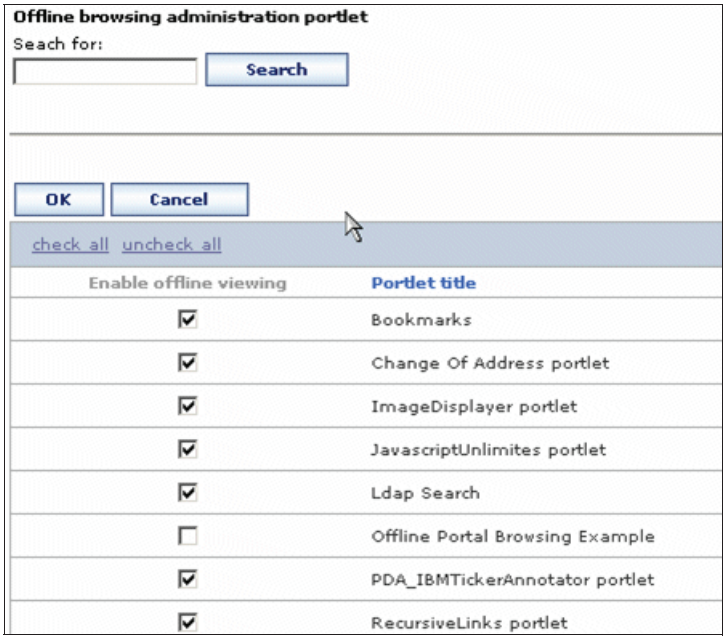


Figure 11-37 Offline browsing administration portlet

4. Configure ACL. Click **Access**. Click **Resource Permissions**.
5. Select **portlets**.

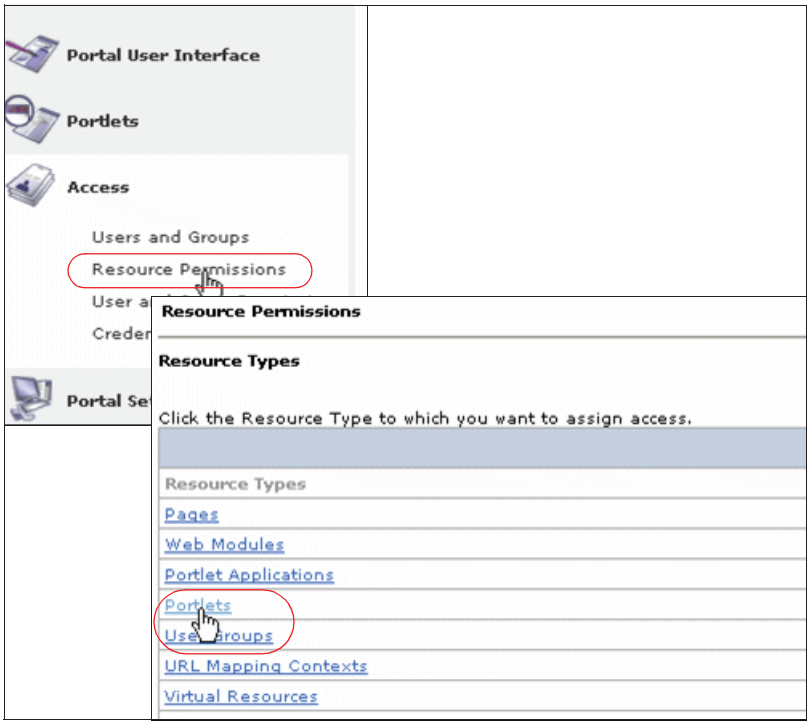


Figure 11-38 Portlets resource permissions

6. Find Offline Portal Browsing Example portlet and click  .

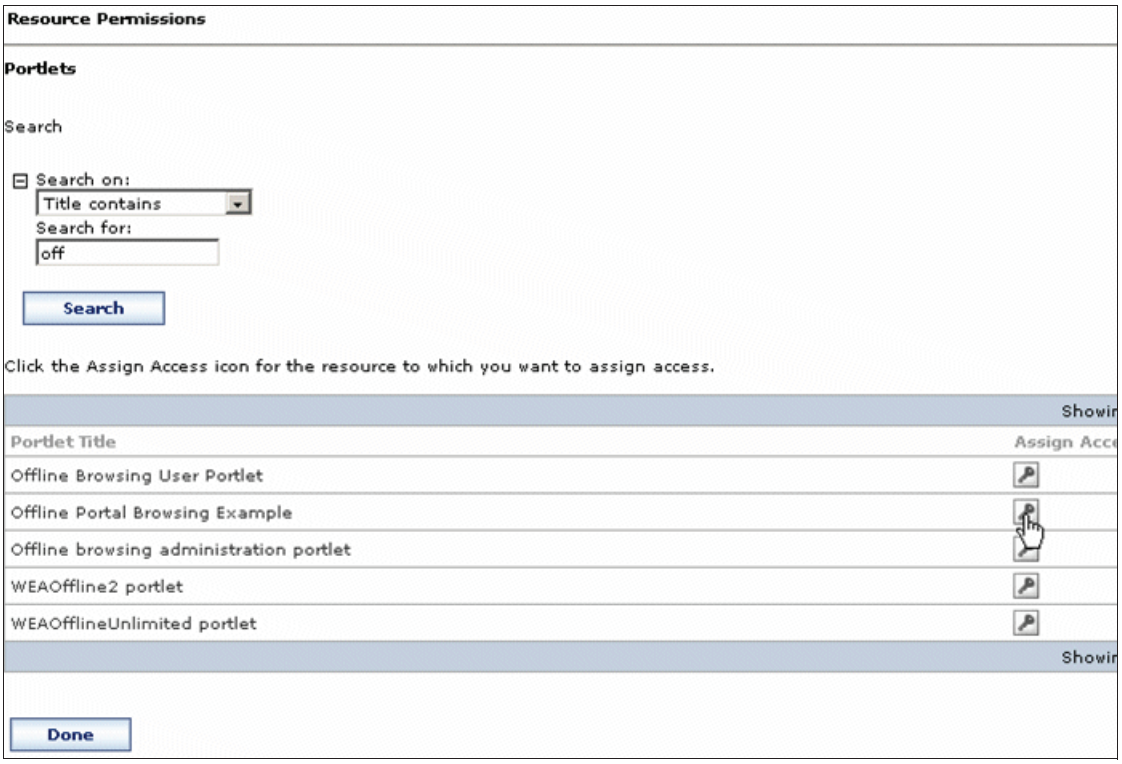


Figure 11-39 Selecting the sample Offline Portal Browsing

7. Delete all check marks for **Allow Inheritance** except **Administrator** and **Security Administrator**. Click pencil mark (Edit mode) on **Privileged User**.

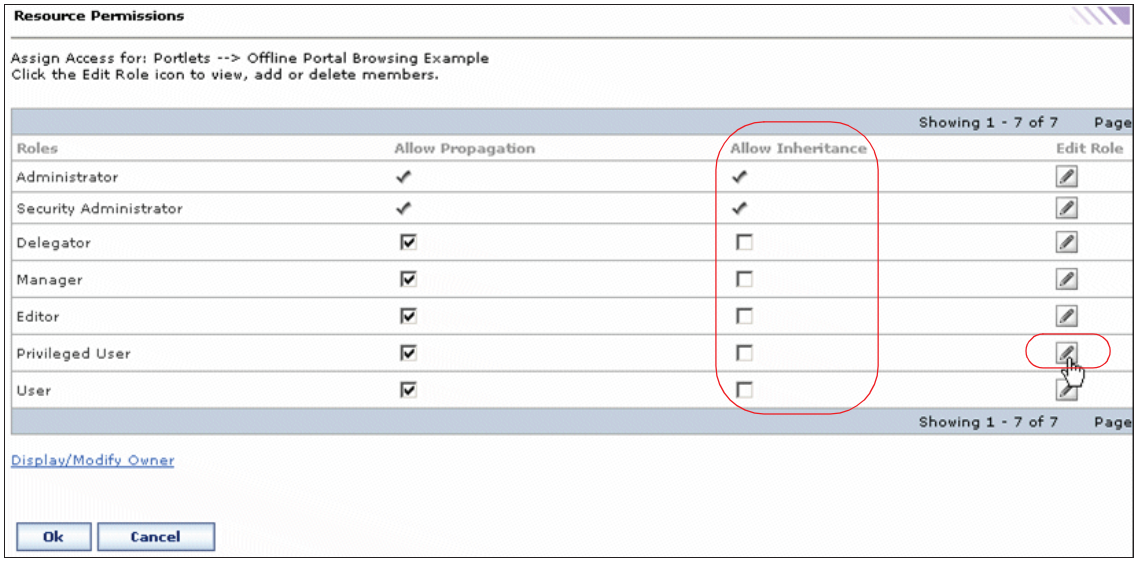


Figure 11-40 Resource permissions

8. Click **Add** button.

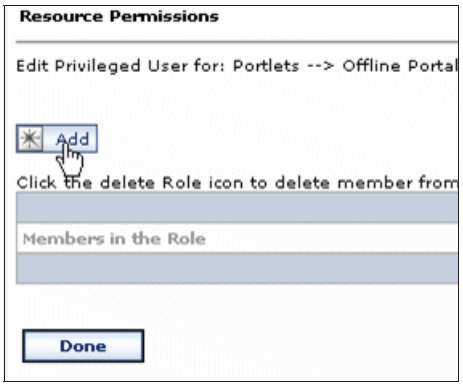


Figure 11-41 Add permissions

9. Check **all authenticated portlet users** and click **OK**. Click **Done**.

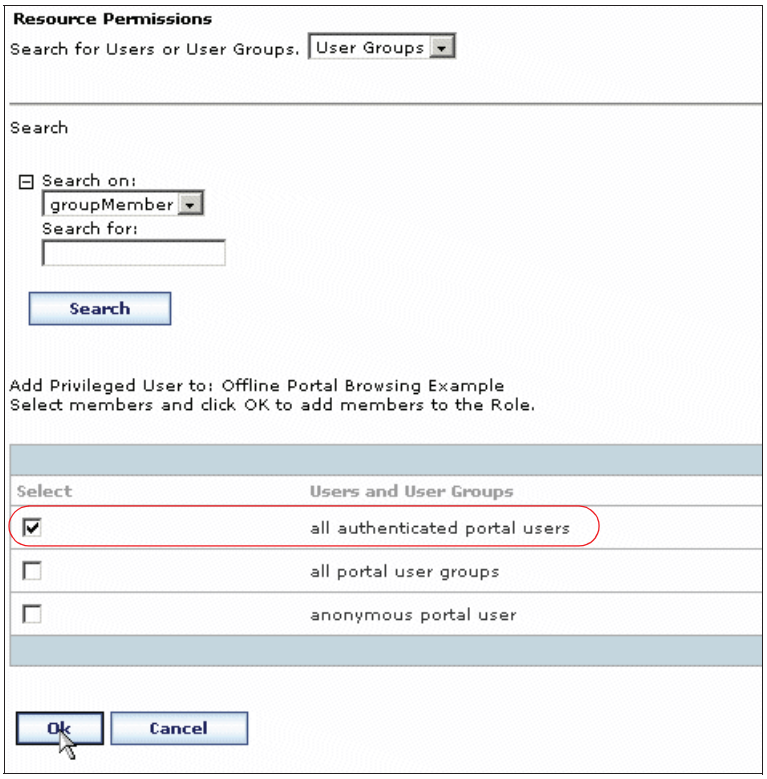


Figure 11-42 Enable for all authenticated portal users

10. Login to WebSphere Everyplace Access. For example as weauser and password weauser.
11. You will see the portlet.

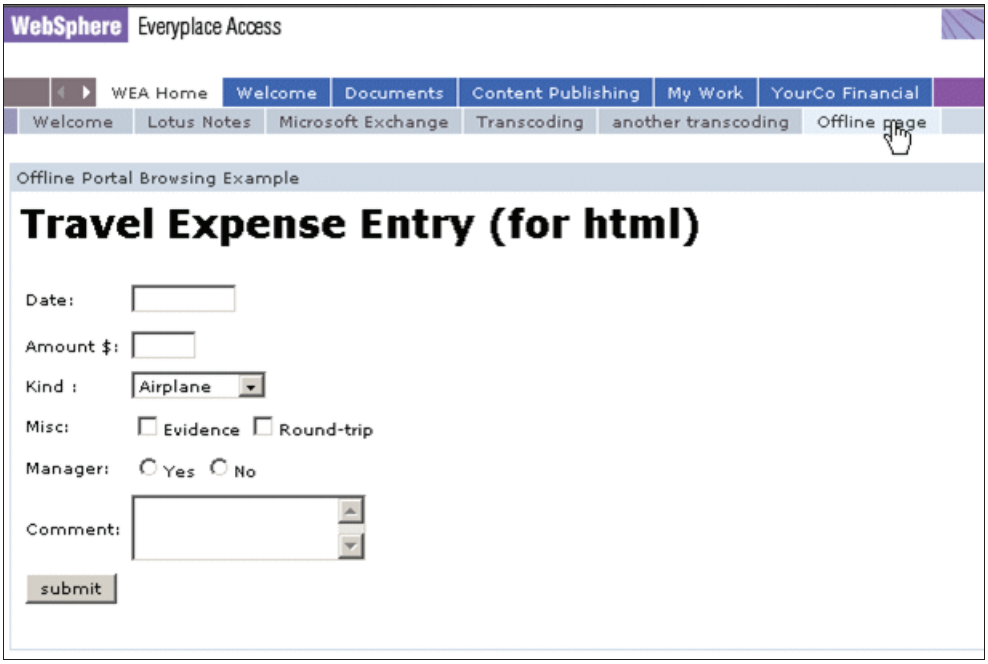


Figure 11-43 Portlet view

12. Click the icon 



Figure 11-44 Configuration icon

13. Click the icon illustrated in Figure 11-45.

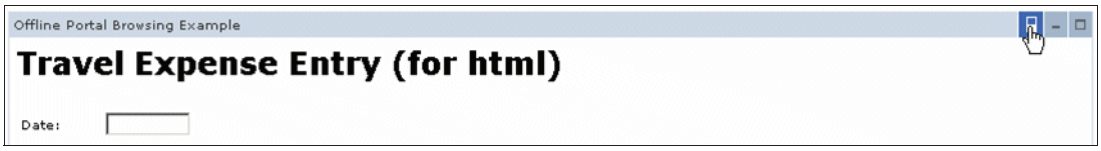


Figure 11-45 Making the portlet available for offline usage

14. You will get the message shown in Figure 11-46. Click **OK**.



Figure 11-46 Portlet available for offline usage

11.7 Configure the portlet client with offline option

In this section you will configure the portlet offline option in the WebSphere Everyplace Access Client and then run the offline form submission. Execute the following steps:

1. Login to the WebSphere Everyplace Access client in the Pocket PC using the following URL:

   `http://127.0.0.1:8080/main.html`

2. In this sample scenario the user ID is weauser and the password is weauser.

3. Click the tool icon  .



Figure 11-47 WebSphere Everyplace Access client

4. Click **Network profiles.** Enter the **Offline Content server** name.
5. Click the icon  next to **Offline home page**.

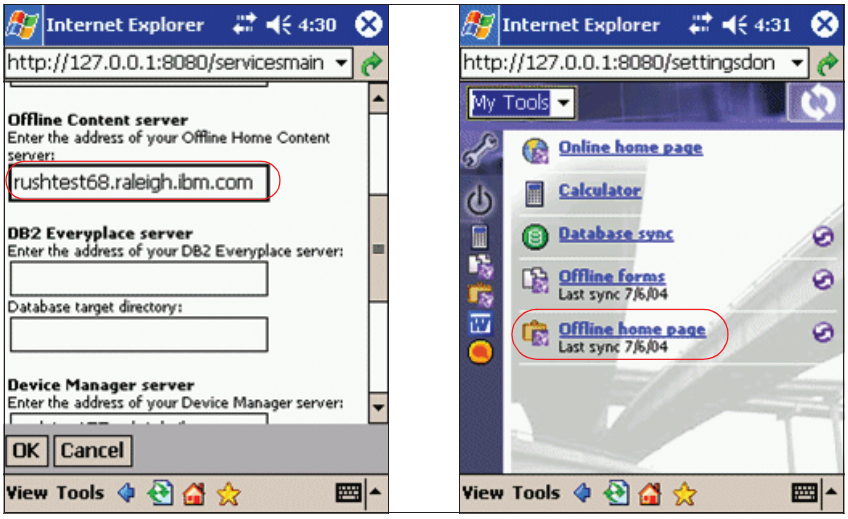


Figure 11-48 Configure Offline Content server and start synchronization

6. After the synchronization completes, click **Offline home page**.
7. Disconnect the PocketPC from the network and then click **Offline Portal Browsing Example** portlet.



Figure 11-49 Offline Portal Browsing Example

8. For testing, enter an invalid data for amount (such as abc for example) and click the **submit** button. An error alert will be prompted.

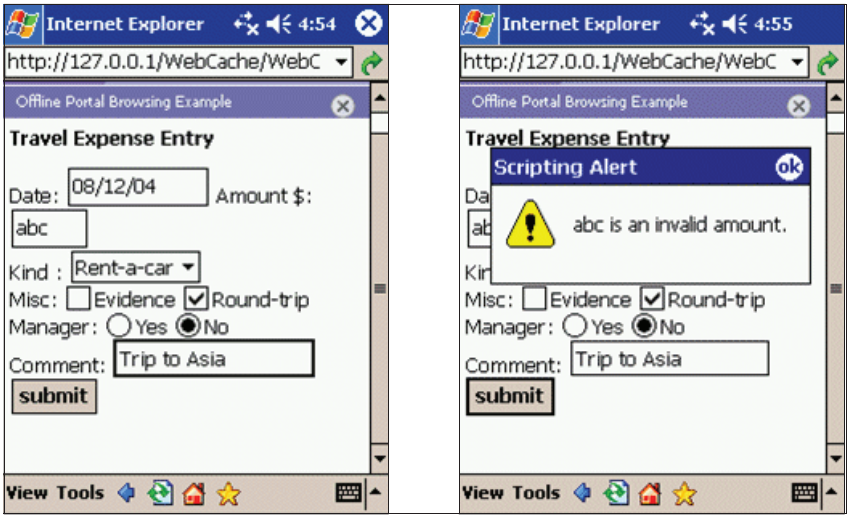


Figure 11-50 Entering invalid data (offline)

9. This time enter a valid amount and submit again. A message indicating that the request has been deferred will appear.

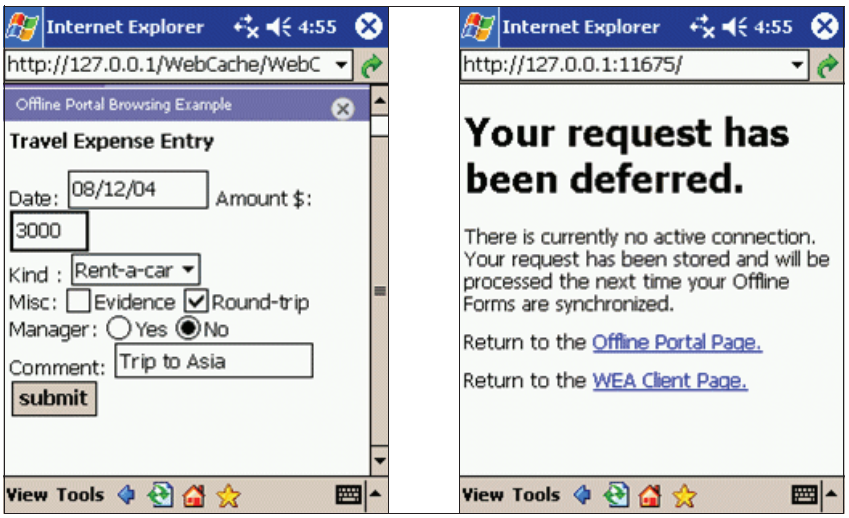


Figure 11-51 Running the portlet offline

10. Click **WEA Client Page.** Connect your PocketPC to the network. To start the synchronization process, click the icon next to **Offline forms.**

11. After the synchronization complete, click **Offline forms**.

    Note: You can also execute these steps after disconnecting the Pocket PC from the network.

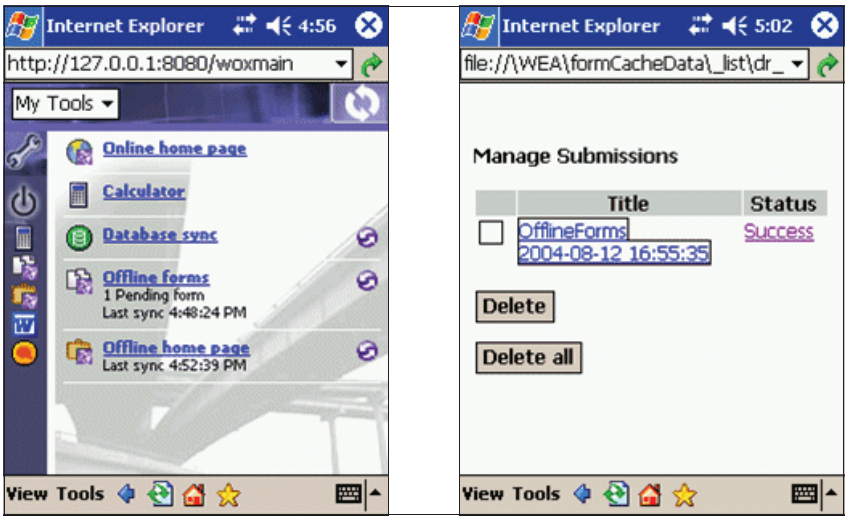


Figure 11-52 Manage offline submissions after synchronization

12. Click **OfflineForms** having status Success. Click **Show response**.

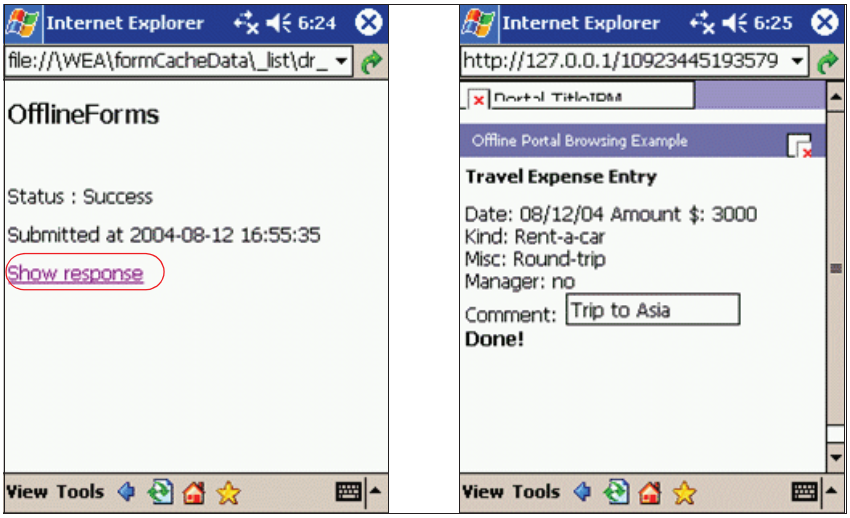


Figure 11-53 Offline portlet response

12

DB2 Everyplace applications with WebSphere Studio Device Developer

This chapter describes the use of WebSphere Studio Device Developer to create Java 2 Micro Edition (J2ME) based applications using DB2 Everyplace database and data synchronization capabilities provided by WebSphere Everyplace Access.

© Copyright IBM Corp. 2005. All rights reserved.

12.1 Overview

WebSphere Studio Device Developer is part of the WebSphere Studio family of products that have been developed based on the Eclipse workbench technology. In this section we provide an overview of this product.

12.1.1 Introduction

The Eclipse workbench is an open source platform, designed by IBM and released to the open source community. It is an open, portable, universal tooling platform that provides frameworks, services and tools for building development tools.

In essence the workbench provides the tool infrastructure. With this infrastructure in place, the tool builders are able to focus on the actual building of their tools. The workbench has been designed for maximum flexibility to support the development of tools for new technologies that may emerge in the future.

The WebSphere Studio family of products is an Eclipse-based integrated platform (IDE) for development, testing, debugging and deploying. It provides support for each phase of the application development life cycle.

WebSphere Studio Device Developer extends the capabilities of the workbench by allowing you to:

- Create WebSphere Studio Device Developer applications and run them locally.
- Create MIDlet Suites and run them locally (in a MIDlet emulator).
- Create builds and launches for embedded and handheld target operating systems (for example, Palm OS and Pocket PC).
- Launch and debug applications on various target devices.

For more information about the WebSphere Studio Device Developer, see:

`http://www.ibm.com/software/wireless/wsdd/`

12.1.2 Java 2 Micro Edition

Java 2 Micro Edition (J2ME) is the Java platform for a broad range of devices, including PDAs, cellular phones and embedded devices. The J2ME platform includes APIs for the development of Java applications in such devices.

The J2ME architecture

J2ME define configurations and profiles for building complete Java runtime environments. Each configuration and profile is targeted to a related category of devices.

Configurations

J2ME is composed of the virtual machine and core class libraries. Each configuration provides the base functionality for developing pervasive applications. Two configurations currently exist:

- Connected Limited Device Configuration (CLDC)

  For devices with sporadic network connections, slow processors, and low memory. Typically includes cell phones and low-end PDAs with 16-bit or 32-bit processors and 512 KB or less of memory for the Java runtime and applications.

- Connected Device Configuration (CDC)

  This configuration is for devices with powerful processors, lots of memory and maybe a large bandwidth, such as high-end PDAs (ARM based). This configuration implements an almost complete subset of the J2ME.

WebSphere Studio Device Developer comes with the J9 Java Virtual Machine and with implementations for the two configurations mentioned above.

Profiles

Profiles are high-level APIs that define the application life cycle model, the user interface, and device-specific properties. The more important profiles are:

- Mobile Information Device Profile (MIDP)

  The MIDP profile is used on cell phones and low-end PDAs. It offers a basic user interface, basic network connectivity, local storage, and application management. The MIDP profile, together with CLDC configuration, provides a complete Java runtime environment for applications that runs on resources-constrained devices.

  MIDP 2.0 is a revised version of the MIDP 1.0 specification, and includes new features such as an enhanced user interface, multimedia and game functionality, greater connectivity, over-the-air (OTA) provisioning, and end-to-end security. MIDP 2.0 is backward compatible with MIDP 1.0, and continues to target mobile information devices such as mobile phones and PDAs. WSDD supports both versions.

- Foundation Profile (FP)

  This is the lowest level profile of the CDC configuration. It provides a broad subset of the J2SE platform including network, I/O, security and text support.

It doesn't include any user interface toolkit. However, it can be combined with others profiles that provide GUI support.

- Personal Basis Profile (PBPro)

  Provides a lightweight implementation of AWT components to build GUI based applications. It is used in conjunction with Foundation Profile to build business-oriented applications.

- Personal Profile (PPro)

  This profile provides a complete implementation of the Abstract Window Toolkit (AWT) for the development of GUI-centric applications. Together, the Foundation and Personal Profiles provide the most complete Java runtime for the development of business-oriented applications.

To complete the scenario, WebSphere Studio Device Developer provides an implementation of the JDBC APIs for database access on the device. However, it is not part of any J2ME configuration or profile, instead it is part of the WebSphere Everyplace Services.

Note: Currently exists a specification, JSR-169 JDBC Optional Package for CDC/Foundation Profile, to provide JDBC access to J2ME CDC environments. WSDD 5.7.1 comes with an implementation of this JSR, but at the moment of this write, it is still in beta stage. You can download it from the WSDD Update site.

To learn more about J2ME, visit `http://java.sun.com/j2me`.

12.1.3 WebSphere Micro Environment

WebSphere Studio Device Developer comes by default with WebSphere Micro Environment (WME) that supports the development of J2ME-based applications. It provides an implementation for the CDC and CLDC configurations, and an implementation for the Mobile Information Device, Foundation and Personal profiles.

The WebSphere Custom Environment (WCE) provides custom characteristics for device and embedded development.

12.2 Installing WebSphere Studio Device Developer

WebSphere Studio Device Developer can be installed stand-alone or as a component of WebSphere Studio Site Developer Version 5.1. We will describe here how to install it.

To install WebSphere Studio Device Developer in stand-alone mode:

1. Run the setup.exe file from WSDD CD.
2. Select a language for the install. Click **OK**.
3. The Welcome window appears. Click **Next**.
4. Read the license agreement and click **Yes**.
5. Select the destination folder for the product. Click **Next**.
6. Select the program folder destination. Click **Next**.
7. Review the installation settings. Click **Next**.
8. Wait while the installation program copies the required files to your PC.
9. If you have WebSphere Studio Site or Application Developer already installed, the installation program will ask if you want to integrate WSDD with it. Choose an option and click **Next.**
10. Click **Finish** to exit the wizard.

12.3 Working with WebSphere Studio Device Developer

In this section, we show you how to use the Device Developer workbench.

12.3.1 Using the workbench

Start WebSphere Studio Device Developer using the shortcut on your Start menu. Select **Programs -> IBM WebSphere Studio -> Device Developer**. When you start the program you see the main window, shown in Figure 12-1 on page 360.

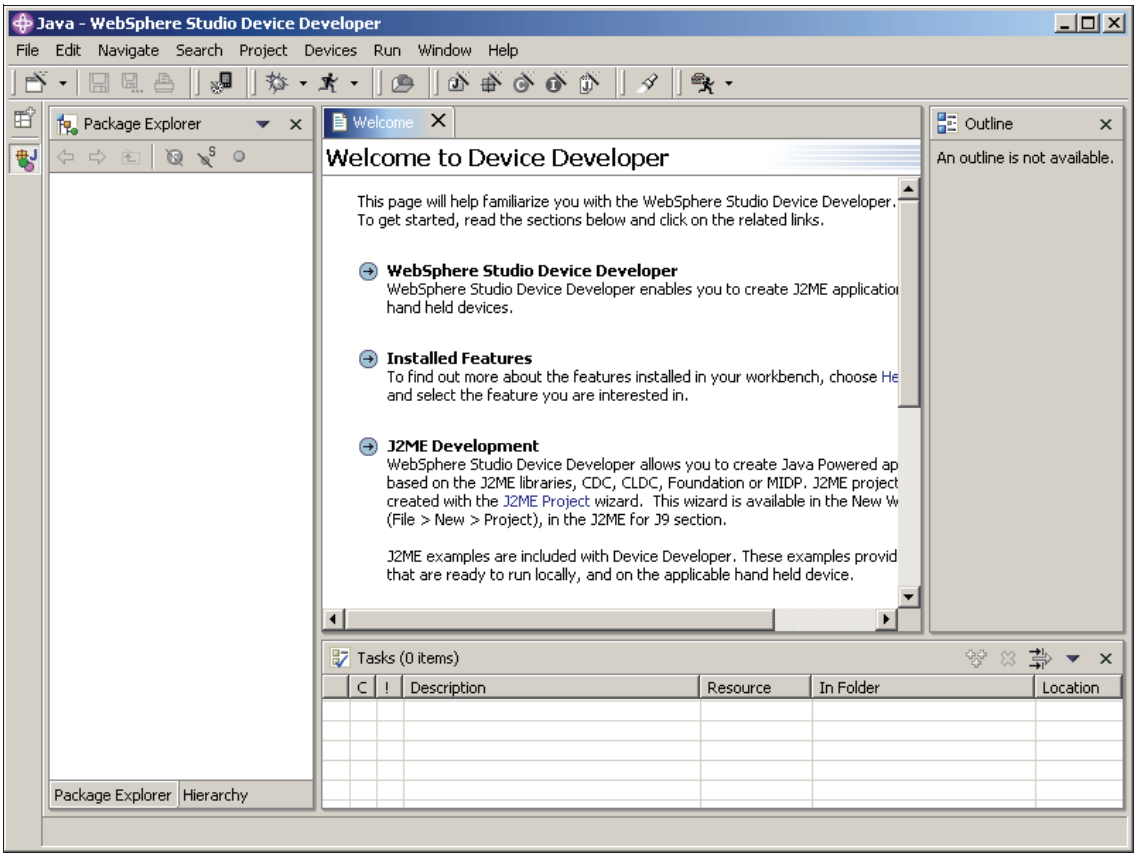


Figure 12-1 The WebSphere Studio Device Developer workbench

For device Java applications, you will work almost always in the Java perspective.

12.3.2 Using the Update Manager

The Update Manager tool allows you to install features that don't come with the packaged version of the product. Among these are:

- WebSphere Micro Environment
  - Extension Services
  - SMF Bundle Development Kit
- Updates for the product

To install these features, you should use the Update Manager perspective. For more information about the workbench, views and perspectives, review 12.3.1, “Using the workbench” on page 359.

For development of DB2 Everyplace (DB2E) applications on Windows CE targets, you will need to install the Database Enabler and the DB2 Everyplace required libraries. These are part of the Extension Services in WSDD 5.7. In order to install the Extension Services, the SMF Bundle Development kit has to be installed.

Note: The Service Managed Framework (SMF) provides a framework above Java that enables you to manage multiple applications and services as components on the platform. It is part of the IBM Workplace™ Client Technology, Micro Edition (WCTME) that is covered in the redbook *IBM Workplace Client Technology Micro Edition Version 5.7.1: Application Development and Case Study*, SG24-6496.

Follow these steps to install the required libraries in WebSphere Studio Device Developer environment:

1. Go to the Update Manager perspective by selecting **Help -> Software Update -> Update Manager**, as shown in Figure 12-2.

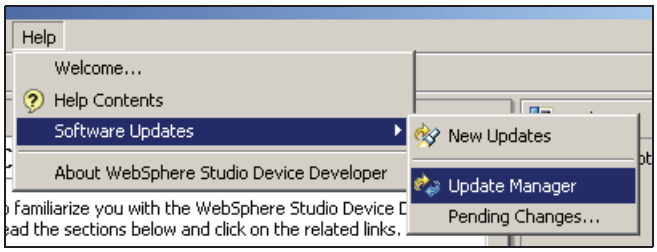


Figure 12-2 Going to the Update Manager perspective

2. In the Features Updates view, select **Sites to Visit -> IBM Micro Environment Toolkit for WebSphere Studio -> Extension Services for WebSphere Everyplace -> SMF Bundle Development Kit 5.7.1**.

   Note: if you can't see the Features Updates view, reset the perspective by selecting **Window -> Reset Perspective**.

3. In the Preview view you see the selected feature details, as shown in Figure 12-3 on page 362. Click **Install Now**.

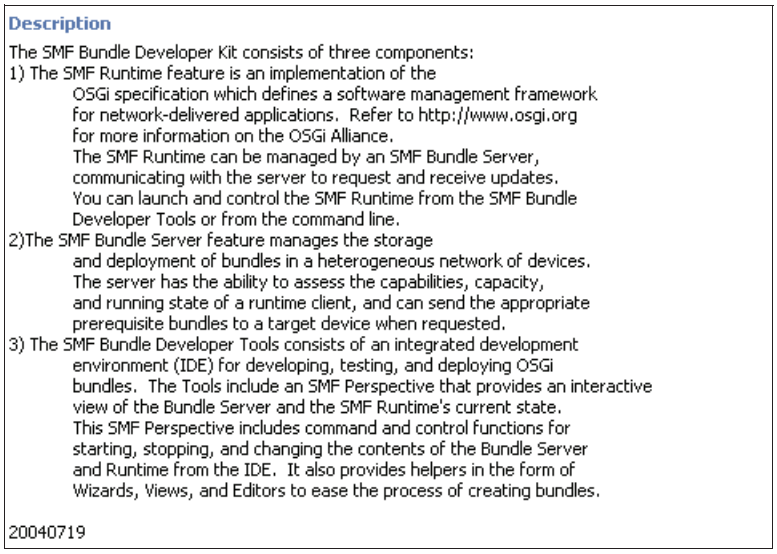


Figure 12-3 Feature details

4. You see the Install Features wizard. Click **Next** to continue
5. .Accept the license agreement by selecting **I accept the terms in the license agreement**, and click **Next**.
6. The wizard shows you where the feature will be installed. Click **Finish**.
7. The wizard warns you about an unsigned feature, as shown in Figure 12-4. This is safe to ignore. Click **Install**.

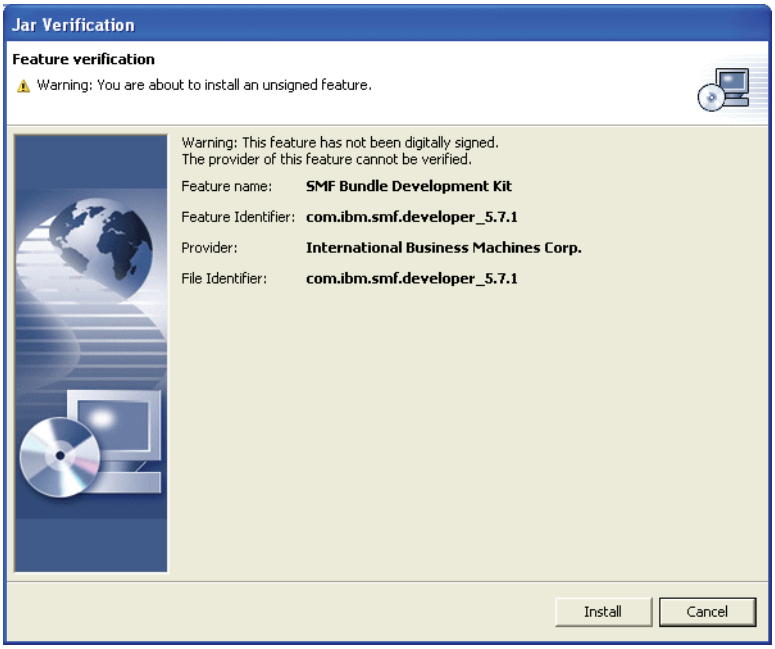


Figure 12-4 Feature verification window

8. Wait while the feature is installed in your workbench.
9. Click **No** to avoid restart the program. We need to install Everyplace Services now.

Tip: If you are installing several features at the same time, respond **No** when the wizard ask you to restart the workbench, except for the last, when you should respond **Yes**.

10. In the Features Updates view, select **Sites to Visit -> IBM Micro Environment Toolkit for WebSphere Studio -> Extension Services for WebSphere Everyplace -> Extension Services 5.7.1**.
11. In the Preview view you see the selected feature details. Click **Install Now**.
12. Repeat the above steps from Step 4 to Step 8 inclusive. Click **Yes** to restart the workbench and make the changes effective.

12.4 Sample scenario

The sample scenario described here illustrates a sales force application using the Personal Profile configuration. We will develop a sales force application for Pocket PC targets using the J2ME Personal Profile (PPro) configuration. This comes with an implementation of the Abstract Window Toolkit (AWT) and is supported by the J9 Foundation Profile.

Note: The PPro configuration is not actually supported on Palm targets. If you want to develop GUI-based applications for Palm devices, you should use the J2ME MIDP profile, or for more advanced applications, the WCE Xtreme configuration.

To develop an application using the PPro profile, follow these steps:

1. Define the database tables for the application.
2. Configure the Pocket PC device for development.
3. Create the project on WebSphere Studio Device Developer.
4. Set up the build and test environment.
5. Write the Java classes that compose the application.
6. Test the application on the device.

12.4.1 Define the database tables for the application

Usually when you are planning to create applications for mobile devices, the idea is to bring data from a central database server to this device and load data entered on the mobile device back into the central database.

This means that the layout of the tables is more or less given and you only have to select whether you want all columns of a table to be included in the database table of the mobile device.

To create the database tables necessary for test the application you have two ways:

- Use the DB2eCLP application that comes with Everyplace Client. You have to run the DDL file to create the database tables.
- Use the DB2 Everyplace ISync Client, which is part of the Everyplace Client, to synchronize the database tables. To use it, you have to configure the DB2 Everyplace SyncServer for the Order System sample database tables. For more information, refer to *IBM WebSphere Everyplace Access V5 Handbook for Developers and Administrators Volume III: E-Mail and Database Synchronization*, SG24-6676. We use this approach in this chapter.

When you have configured the subscription, you should use the DB2 Everyplace ISync Client to bring the database tables and data to the mobile device or to the testing environment. Refer to *IBM WebSphere Everyplace Access V5 Handbook for Developers and Administrators Volume III: E-Mail and Database Synchronization*, SG24-6676 for more information.

A total of four database tables are needed for the application:

- CUSTOMERS
- ORDERS
- PRODUCTS
- ORDER_ITEMS

Important: Be careful when you choose the database tables names. You cannot use a SQL reserved word as a table name. If you do that, your application will fail.

The ORDERS table holds the new orders that are entered during the day, whereas the ORDER_ITEMS table holds the items for every order in the ORDERS table. The CUSTOMERS table contains information about all customers of a sales representative. PRODUCTS shows all available products. Refer to Figure 12-5 on page 366 for the definition of the tables.

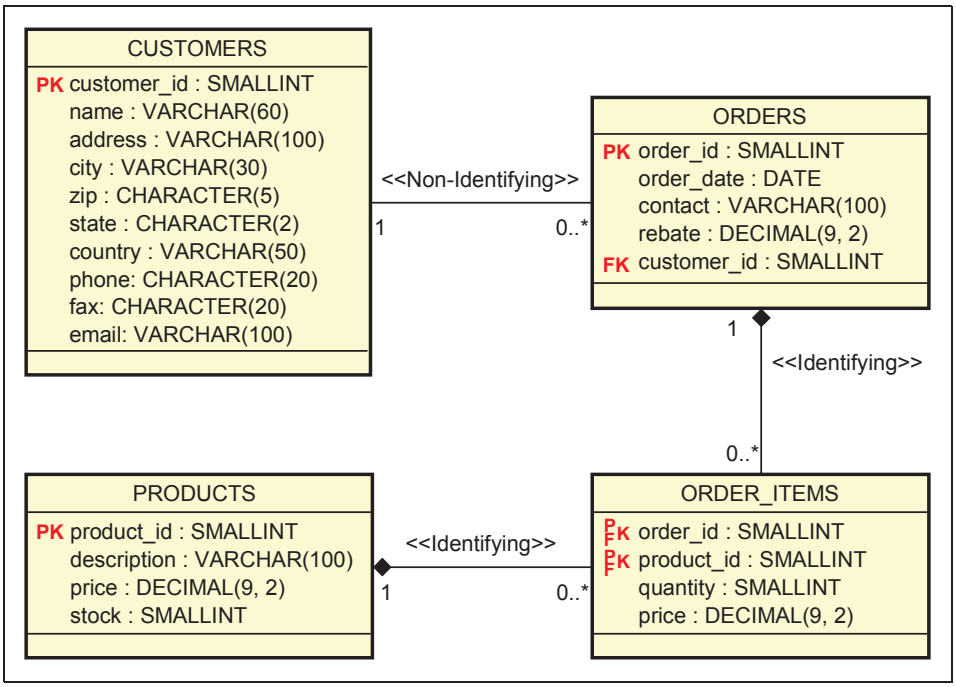


Figure 12-5 Data model diagram

You should create a file containing the necessary SQL table definition statements for the sample application. Make sure that the file extension is .ddl so that the definitions can be imported into the PAB. An example of a DDL file is shown in Example 12-1.

Example 12-1 DDL file for the sample application

```
CREATE TABLE PRODUCTS (
       product_id           SMALLINT NOT NULL,
       description          VARCHAR(100) NOT NULL,
       price                DECIMAL(9,2),
       stock                SMALLINT,
       PRIMARY KEY (product_id)
);

CREATE TABLE CUSTOMERS (
       customer_id          SMALLINT NOT NULL,
       name                 VARCHAR(60) NOT NULL,
       address              VARCHAR(100) NOT NULL,
       city                 VARCHAR(30) NOT NULL,
       zip                  CHAR(5),
       state                CHAR(2),
```

```
       country              VARCHAR(50) NOT NULL,
       phone                CHAR(20) NOT NULL,
       fax                  CHAR(20),
       email                VARCHAR(100),
       PRIMARY KEY (customer_id)
);

CREATE TABLE ORDERS (
       order_id             SMALLINT NOT NULL,
       order_date           DATE NOT NULL,
       contact              VARCHAR(100),
       rebate               DECIMAL(9,2),
       customer_id          SMALLINT NOT NULL,
       PRIMARY KEY (order_id),
       FOREIGN KEY (customer_id)
                             REFERENCES CUSTOMERS
);

CREATE TABLE ORDER_ITEMS (
       order_id             SMALLINT NOT NULL,
       product_id           SMALLINT NOT NULL,
       quantity             SMALLINT NOT NULL,
       price                DECIMAL(9,2) NOT NULL,
       PRIMARY KEY (order_id, product_id),
       FOREIGN KEY (product_id)
                             REFERENCES PRODUCTS,
       FOREIGN KEY (order_id)
                             REFERENCES ORDERS
);
```

Having created a file containing the table specifications, you are now ready to start working on the application.

12.4.2 Configure the Pocket PC device for development

You have to install and configure several components prior to testing and debugging applications on the Pocket PC device using the WebSphere Studio Device Developer workbench. The following components need to be configured:

- Microsoft Active Sync
- J9 runtime for Personal Profile configuration
- Database support
- DB2e runtime
- Database support, JDBC driver and synchronization driver

Microsoft Active Sync

The Microsoft Active Sync software is necessary to provide connectivity between the development machine and the Pocket PC device. You can connect the Pocket PC device using a USB port, a serial port, and an Ethernet card. Table 12-1 offers a resume of the connection methods.

Table 12-1 Connection methods for Pocket PC

Connection Method	Typical Connection Speed
Ethernet/wireless (TCP/IP LAN)	Fastest (10 Mbps for wireless, or more for wired)
USB cradle connection	Faster (up to 12 Mbps)
Serial cradle connection	Slower (56 kbps to 115 kbps)

The serial connection is not recommended because of its slow transfer ratio. With the USB connection, you can connect the Pocket PC to the network using the ActiveSync pass-through feature; however it is not a true TCP/IP connection. For example the USB connection cannot be used to debug the application.

Note: You need a TCP/IP connection to use the debug capability of WebSphere Studio Device Developer. You can obtain one using an Ethernet card for your Pocket PC device. The WebSphere Studio Device Developer debugger will not work with an ActiveSync USB connection.

J9 runtime for Personal Profile configuration

Conveniently WSDD 5.7.1 provides CAB files to install the J9 runtime in the Pocket PC devices. For Personal Profile, the CAB files could be located at <WSDD>\wsdd5.0\ive-2.2\runtimes\wm2003\arm\ppro10\cab directory. For English language you must install the following files:

- j9-ppro10-wm2003-arm_22.CAB
- j9-ppro10_en-wm2003-arm_22.CAB

You can use ActiveSync to install these files in your device. If you want to deploy to several devices consider to use the WEA Device Services.

By default the J9 runtime will be installed at \Program Files\J9\PPRO10 directory on the device.

Note: WSDD 5.7.1 only supports Windows Mobile 2003 (aka Pocket PC 2003) target devices. If you want to work with Pocket PC 2002 devices you should use the previous version, WSDD 5.6.

Database support

Now we need to deploy the Database Enabler library on the J9 runtime already installed. You can locate the library at <WSDD>\wsdd5.0\technologies\eswe\ bundlefile directory. The library is named database_enabler.jar. Copy this library to the <J9_runtime>\PPRO10\lib\jclPPro10\ext directory on your Pocket PC device.

DB2e runtime, JDBC driver and synchronization driver

The DB2E runtime is installed as part of the Everyplace Client installation on the device. For more information about the Everyplace Client installation, refer to *IBM WebSphere Everyplace Access V5 Handbook for Developers and Administrators Volume I: Installation and Administration*, SG24-6462.

Everyplace Client installs the required JDBC and synchronization JAR files on the device, these are installed in the \Windows directory. This means that WSDD will have to put these files in the classpath when creating the shortcut on the device to run the application. The two required libraries are:

- db2ejdbc.jar
- isync4j.jar

12.4.3 Create the project on WebSphere Studio Device Developer

Now that you has configured the Pocket PC device for development, you need to create a J2ME project for the sales force application. Follow these steps:

1. Start WebSphere Studio Device Developer by selecting **Programs -> IBM WebSphere Studio -> Device Developer**.
2. On the workbench, select **File -> New -> Project**.
3. You will see the New Project window. Select **J2ME for J9** on the left pane and select **Create J2ME project** in the right pane, as shown in Figure 12-6 on page 370. Click **Next**.

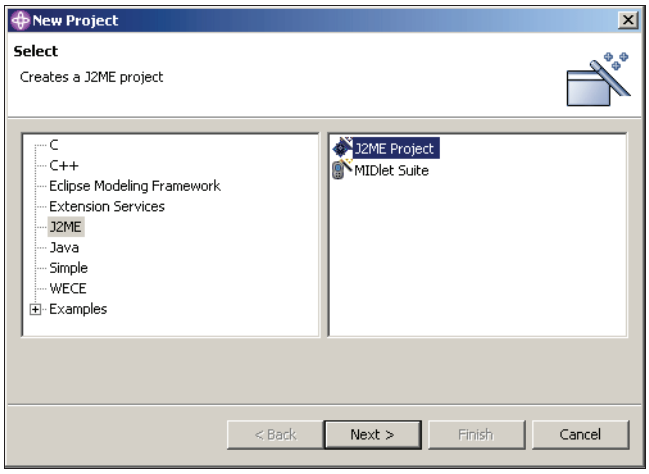


Figure 12-6 Creating a J2ME project

4. Enter `MobileOrderSystem` as the project name, as shown in Figure 12-7. In this window you also can choose the directory for the project. In this case, leave as is. Click **Next**.

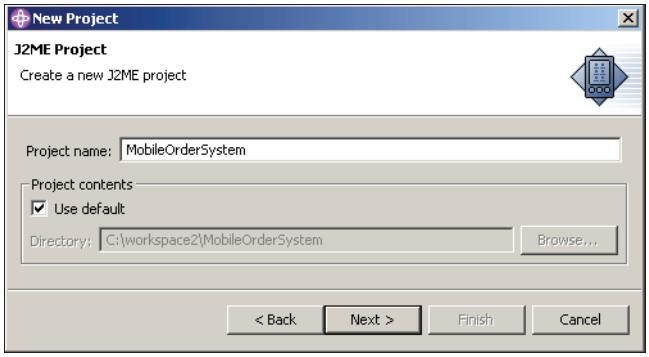


Figure 12-7 Setting the project name

5. In this window you can select the J2ME profile. Select the **2.2.0 -> JCL Personal Profile 1.0**, as shown in Figure 12-8 on page 371. Click **Next**.

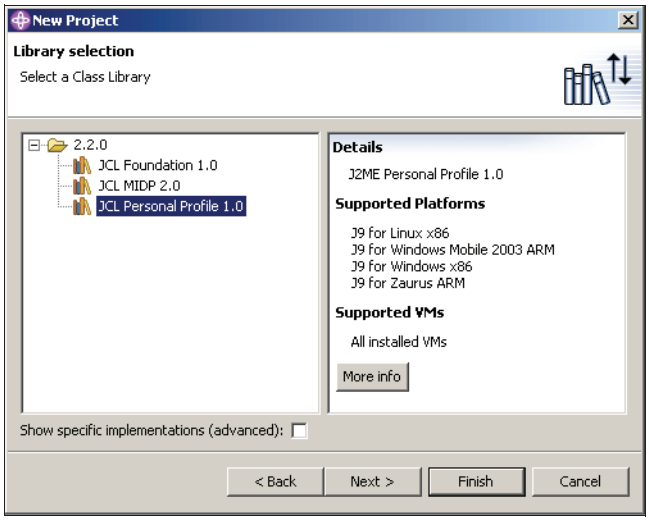


Figure 12-8 Selecting the profile

6. Select the Libraries tab. Click **Add Variable**. In the window shown in Figure 12-9, select ESWE_BUNDLES and click **Extend**.

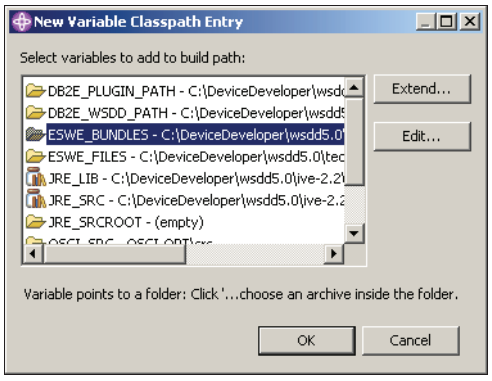


Figure 12-9 Selecting a variable classpath library

7. Select database_enabler.jar, DB2e.jar y ISync.jar, as shown in Figure 12-10 on page 372. Click **OK.**

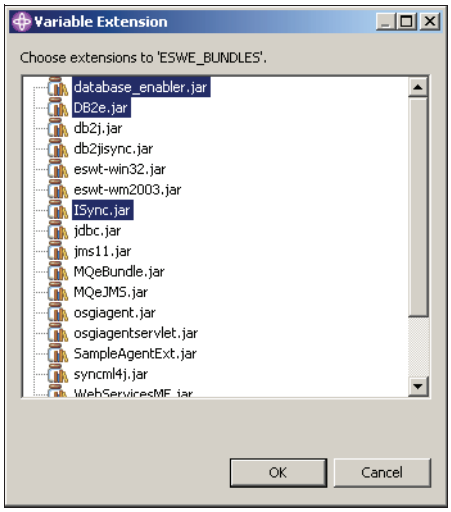


Figure 12-10 Selecting required libraries

Note: The jdbc.jar that you can see in Figure 12-10 is the implementation of the JSP-169 mentioned earlier. We won't use this library in this scenario.

8. Click **Finish.**

You can see the new project in the Package Explorer view. If you click the plus icon near to the project name to expand the tree, you will see the libraries that contain the configuration (PPro in this case), the root source directory (src) among other files, as shown in Figure 12-11.



Figure 12-11 The J2ME project created

12.4.4 Write the Java classes of the application

Now you have to write the classes of the application. You need to create each class using the New Class wizard, and then write the corresponding code.

MobileOrderSystem

This class has the main method used for start the application. The main method opens the database connection, using the path to the database, and shows the first AWT form, NewOrdersFrame in this case. Also, it defines the database path as a static constant.

The code for the class is shown in Example 12-2.

Example 12-2 MobileOrderSystem.java

```
package com.ibm.ral.itso.mos;

import java.sql.SQLException;

/**
 * Main class for Mobile Order System
 *
 */
public class MobileOrderSystem  {

   public final static String DB2E_DATABASE_PATH = "\\ordsysdb\\";

   public static void main(String[] args) throws SQLException {

      // Opens the connection to the database
      Database.getInstance().open(MobileOrderSystem.DB2E_DATABASE_PATH);

      // Shows the first frame of the application
      new NewOrdersFrame().show();
   }
}
```

Database

This class encapsulates the creation of the connection to the database. It uses the singleton pattern to generate an instance of the Database object for use where it is needed. For the sake of simplicity, we only use a connection for all database accesses.

The code for the class is shown in Example 12-3.

Example 12-3 Database.java

```
package com.ibm.ral.itso.mos;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
```

```
/**
 * Encapsulate the access to the DB2e database
 *
 */
public class Database {

   // The database url prefix
   private static final String DB2E_DATABASE_URL_BASE = "jdbc:db2e:";
   // The driver class name
   private static final String DB2E_DRIVER_CLASS
      = "com.ibm.db2e.jdbc.DB2eDriver";

   private static Database instance;
   private Connection connection;

   /**
    * Constructor for Database
    */
   private Database() {
      try {
         // Loads the DB2e driver class
         Class.forName(DB2E_DRIVER_CLASS);
      }
      catch(Exception e) {
         e.printStackTrace();
      }
   }

   /**
    * Returns an instance of the Database object
    * .
    * @return Database
    */
   public static Database getInstance() {
      if (instance == null) {
         instance = new Database();
      }

      return instance;
   }

   /**
    * Returns the database connection
    *
    * @return Connection
    */
   public Connection getConnection() {
      return connection;
```

```
    }

    /**
     * Opens the database connection
     *
     * @param databasePath The DB2e database path
     * @throws SQLException
     */
    public void open(String databasePath) throws SQLException {
       this.connection
           = DriverManager.getConnection(DB2E_DATABASE_URL_BASE+databasePath);
    }

    /**
     * Close the database connection
     *
     * @throws SQLException
     */
    public void close() throws SQLException {
       if (connection != null) connection.close();
    }
}
```

NewOrdersFrame

This class shows a list with the orders for a sales representative. It has three possible actions: Edit an order, Delete an order, and Synchronize the database.

The list uses a PreparedStatement.executeQuery() to perform the SQL SELECT required to obtain the data. The Delete button use a PreparedStatement.executeUpdate() to perform the SQL UPDATE sentence for the order selected in the list.

The code for the class is shown in Example 12-4.

Example 12-4 NewOrdersFrame.java

```
package com.ibm.ral.itso.mos;

import java.awt.Button;
import java.awt.Frame;
import java.awt.List;
import java.awt.MenuBar;
import java.awt.Panel;
import java.awt.ScrollPane;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Vector;

/**
 * AWT Frame for New Orders
 *
 * Shows a list with the order for a salesman. Also it permits edit and delete
 * an order, and synchronize the database
 */
public class NewOrdersFrame extends Frame {

   private static int NO_ORDER_ID = -1;

   // Selects all the orders
   // Selects all the orders
   private static final String SQL_ALLORDERS
      = " SELECT ORDER_ID, ORDER_DATE, C.NAME "
      + " FROM ORDERS O, CUSTOMERS C "
      + " WHERE O.CUSTOMER_ID = C.CUSTOMER_ID";

   // Delete an specific order
   private static final String SQL_DELETE_ORDER_BY_ID
      = "DELETE FROM ORDERS WHERE ORDER_ID = ?";


   // Visual control for the frame
   private Panel panel = null;
    private ScrollPane scrollPane = null;
    private Button editButton = null;
    private List list = null;
    private Button syncButton = null;
    private Button deleteButton = null;
    private MenuBar mainMenuBar = null;

    // Vector of orders keys
    private Vector orders = new Vector();

   /**
    * Constructor for NewOrdersFrame
    *
    */
   public NewOrdersFrame() {
      super();
      initialize();
   }

   /**
    * Initialize the AWT form
```

```
 *
 */
private void initialize() {
     this.add(getPanel(), java.awt.BorderLayout.SOUTH);
     this.add(getScrollPane(), java.awt.BorderLayout.CENTER);
     this.setMenuBar(getMainMenuBar());
     this.setTitle("New orders");
     initNewOrdersList();
}


/**
 * Initialize the menubar
 *
 * @return MenuBar
 */
private MenuBar getMainMenuBar() {
   if (mainMenuBar == null) {
      mainMenuBar = new MenuBar();
   }
   return mainMenuBar;
}

/**
 * Initializes the panel
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getPanel() {
   if(panel == null) {
      panel = new java.awt.Panel();
      panel.add(getSyncButton());
      panel.add(getEditButton());
      panel.add(getDeleteButton());
   }
   return panel;
}
/**
 * Initializes the scrollPane
 *
 * @return java.awt.ScrollPane
 */
private java.awt.ScrollPane getScrollPane() {
   if(scrollPane == null) {
      scrollPane = new java.awt.ScrollPane();
      scrollPane.add(getList(), null);
   }
   return scrollPane;
}
```

```
/**
 * Initializes the Edit button
 *
 * @return java.awt.Button
 */
private java.awt.Button getEditButton() {
   if(editButton == null) {
      editButton = new java.awt.Button();
      editButton.setLabel("Edit");
      editButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent event) {
            editCommand();
         }
      });
   }
   return editButton;
}

/**
 * Initializes the orders list
 *
 * @return java.awt.List
 */
private java.awt.List getList() {
   if(list == null) {
      list = new java.awt.List();
   }
   return list;
}

/**
 * Initializes the Synchronize button
 *
 * @return java.awt.Button
 */
private java.awt.Button getSyncButton() {
   if(syncButton == null) {
      syncButton = new java.awt.Button();
      syncButton.setLabel("Synchronize");
      syncButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent event) {
            syncCommand();
         }
      });
   }
   return syncButton;
}
```

```
/**
 * Initializes the Delete button
 *
 * @return java.awt.Button
 */
private java.awt.Button getDeleteButton() {
   if(deleteButton == null) {
      deleteButton = new java.awt.Button();
      deleteButton.setLabel("Delete");
      deleteButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent event) {
            deleteCommand();
         }
      });
   }
   return deleteButton;
}

/**
 * Obtains the data for the list
 */
private void initNewOrdersList() {

   Connection conn = null;
   try {
      conn = Database.getInstance().getConnection();

      PreparedStatement pstm = conn.prepareStatement(SQL_ALLORDERS);
      ResultSet rs = pstm.executeQuery();

      while (rs.next()) {
         orders.add(new Integer(rs.getInt(1)));
         getList().add(rs.getDate(2)+" "+rs.getString(3));
      }

      rs.close();
      pstm.close();
   }
   catch(SQLException e) {
      e.printStackTrace();
   }
}

/**
 * Closes this frame and shows the Sync frame
 */
private void syncCommand() {
   // Closes this form
   this.setVisible(false);
```

```
   this.dispose();

   // Shows the Sync form
   new SyncFrame().show();
}

/**
 * Close this frame and shows the Edit frame with the order currently
 * selected on the list. If there's no order selected, it does nothing.
 *
 */
private void editCommand() {
   int orderId = getSelectedOrderId();
   if (orderId == NO_ORDER_ID) return;

   this.setVisible(false);
   this.dispose();
   new EditOrderFrame(orderId).show();
}

/**
 * Deletes the order currently selected on the list. If there's no order
 * selected, it does nothing.
 */
private void deleteCommand() {

   int order_id = getSelectedOrderId();

   if (order_id == NO_ORDER_ID) return;

   try {
      PreparedStatement pstm
         = Database.getInstance().getConnection()
                        .prepareStatement(SQL_DELETE_ORDER_BY_ID);
      pstm.setInt(1, order_id);
      int rowsDeleted = pstm.executeUpdate();
      if (rowsDeleted == 1) {
         orders.remove(getList().getSelectedIndex());
         getList().remove(getList().getSelectedIndex());
      }
   }
   catch(SQLException e) {
      e.printStackTrace();
   }
}

/**
 * Looks for the order_id of the currently selected order on the list
 *
```

```
     * @return int The order_id
     */
    private int getSelectedOrderId() {
        int code = NO_ORDER_ID;
        int selectedIndex = getList().getSelectedIndex();

        if (selectedIndex >= 0) {
            code = ((Integer)orders.get(selectedIndex)).intValue();

        }

        return code;
    }
}
```

EditOrderFrame

This class shows a frame for edit an order, previously selected in the New Orders frame. You can edit the Rebate and Contact fields and use the Update order button to reflect the changes in the database. The frame also shows a list with the items for the current order.

The code for the class is shown in Example 12-5.

Example 12-5 EditOrderFrame.java

```
package com.ibm.ral.itso.mos;

import java.awt.Frame;
import java.awt.MenuBar;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.math.BigDecimal;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.DecimalFormat;
import java.text.NumberFormat;

/**
 * AWT Frame for Edit and order
 *
 */
public class EditOrderFrame extends Frame {

    private static final String SQL_ORDER_BYID
        = " SELECT ORDER_DATE, CUSTOMERS.NAME, REBATE, CONTACT"
        + " FROM ORDERS, CUSTOMERS "
        + " WHERE ORDERS.CUSTOMER_ID = CUSTOMERS.CUSTOMER_ID "
```

```
   + " AND ORDER_ID = ?";

private static final String SQL_ORDERITEMS_BYID
   = " SELECT PRODUCTS.DESCRIPTION, QUANTITY, ORDER_ITEMS.PRICE, "
   + " QUANTITY * ORDER_ITEMS.PRICE "
   + " FROM ORDER_ITEMS, PRODUCTS "
   + " WHERE ORDER_ITEMS.PRODUCT_ID = PRODUCTS.PRODUCT_ID "
   + " AND ORDER_ITEMS.ORDER_ID = ?";

private static final String SQL_UPDATE_ORDER
   = " UPDATE ORDERS SET REBATE = ?, CONTACT = ? WHERE ORDER_ID =?";

// Order ID for the editing order
private int orderId;

// Visual controls
private java.awt.Panel buttonPanel = null;
private java.awt.Panel mainPanel = null;
private java.awt.Panel fieldsPanel = null;
private java.awt.ScrollPane orderItemScrollpane = null;
private java.awt.List itemList = null;
private java.awt.TextField customerNameField = null;
private java.awt.TextField orderDateField = null;
private java.awt.TextField orderIdField = null;
private java.awt.Panel rebatePanel = null;
private java.awt.Panel contactPanel = null;
private java.awt.Label label = null;
private java.awt.Label label1 = null;
private java.awt.TextField rebateField = null;
private java.awt.TextField contactField = null;
private java.awt.Panel itemsPanel = null;
private java.awt.Label label2 = null;
private java.awt.Button updateButton = null;
private java.awt.Button backButton = null;
private MenuBar mainMenuBar = null;

/**
 * Constructor for EditOrderFrame
 *
 */
public EditOrderFrame(int orderId) {
   super();
   this.orderId = orderId;
   initialize();
}

/**
 * Initializes the frame
 *
```

```
 */
private void initialize() {
     this.add(getButtonPanel(), java.awt.BorderLayout.SOUTH);
     this.add(getMainPanel(), java.awt.BorderLayout.CENTER);
     this.setMenuBar(getMainMenuBar());
     this.setTitle("Editing order");

}

/**
 * Initializes the main menubar
 *
 * @return MenuBar
 */
private MenuBar getMainMenuBar() {
   if (mainMenuBar == null) {
      mainMenuBar = new MenuBar();
   }
   return mainMenuBar;
}

/**
 * Initializes the button panel
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getButtonPanel() {
   if(buttonPanel == null) {
      buttonPanel = new java.awt.Panel();
      java.awt.FlowLayout layFlowLayout7 = new java.awt.FlowLayout();
      layFlowLayout7.setAlignment(java.awt.FlowLayout.LEFT);
      buttonPanel.setLayout(layFlowLayout7);
      buttonPanel.add(getUpdateButton(), null);
      buttonPanel.add(getBackButton(), null);
      buttonPanel.setName("buttonPanel");
   }
   return buttonPanel;
}

/**
 * Initializes the main panel
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getMainPanel() {
   if(mainPanel == null) {
      mainPanel = new java.awt.Panel();
      java.awt.GridLayout layGridLayout2 = new java.awt.GridLayout();
      layGridLayout2.setRows(2);
```

```
        layGridLayout2.setColumns(1);
        mainPanel.setLayout(layGridLayout2);
        mainPanel.add(getFieldsPanel(), null);
        mainPanel.add(getItemsPanel(), null);
        mainPanel.setName("mainPanel");
    }
    return mainPanel;
}

/**
 * Initializes the order fields panel
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getFieldsPanel() {
    if(fieldsPanel == null) {
        fieldsPanel = new java.awt.Panel();
        java.awt.GridLayout layGridLayout3 = new java.awt.GridLayout();
        layGridLayout3.setRows(5);
        layGridLayout3.setColumns(1);
        fieldsPanel.setLayout(layGridLayout3);
        fieldsPanel.add(getOrderIdField(), null);
        fieldsPanel.add(getOrderDateField(), null);
        fieldsPanel.add(getCustomerNameField(), null);
        fieldsPanel.add(getRebatePanel(), null);
        fieldsPanel.add(getContactPanel(), null);
        initFields();
    }
    return fieldsPanel;
}

/**
 * Initializes the list scrollPane
 *
 * @return java.awt.ScrollPane
 */
private java.awt.ScrollPane getOrderItemScrollpane() {
    if(orderItemScrollpane == null) {
        orderItemScrollpane = new java.awt.ScrollPane();
        orderItemScrollpane.add(getItemList(), null);
    }
    return orderItemScrollpane;
}

/**
 * Initializes the list
 *
 * @return java.awt.List
 */
```

```
private java.awt.List getItemList() {
   if(itemList == null) {
      itemList = new java.awt.List();
      initItemList();
   }
   return itemList;
}

/**
 * Initializes the customer name textfield
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getCustomerNameField() {
   if(customerNameField == null) {
      customerNameField = new java.awt.TextField();
      customerNameField.setFont(new java.awt.Font("sansserif", 1, 12));
      customerNameField.setEditable(false);
   }
   return customerNameField;
}

/**
 * Initializes the order date textfield
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getOrderDateField() {
   if(orderDateField == null) {
      orderDateField = new java.awt.TextField();
      orderDateField.setFont(new java.awt.Font("sansserif", 1, 12));
      orderDateField.setEditable(false);
   }
   return orderDateField;
}

/**
 * Initializes the order id textfield
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getOrderIdField() {
   if(orderIdField == null) {
      orderIdField = new java.awt.TextField();
      orderIdField.setFont(new java.awt.Font("sansserif", 1, 12));
      orderIdField.setEditable(false);
   }
   return orderIdField;
}
```

```
/**
 * Initializes the rebate panel
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getRebatePanel() {
   if(rebatePanel == null) {
      rebatePanel = new java.awt.Panel();
      java.awt.FlowLayout layFlowLayout19 = new java.awt.FlowLayout();
      layFlowLayout19.setHgap(0);
      layFlowLayout19.setVgap(0);
      layFlowLayout19.setAlignment(java.awt.FlowLayout.LEFT);
      rebatePanel.setLayout(layFlowLayout19);
      rebatePanel.add(getLabel(), null);
      rebatePanel.add(getRebateField(), null);
   }
   return rebatePanel;
}

/**
 * Initializes the contact panel
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getContactPanel() {
   if(contactPanel == null) {
      contactPanel = new java.awt.Panel();
      java.awt.FlowLayout layFlowLayout18 = new java.awt.FlowLayout();
      layFlowLayout18.setHgap(0);
      layFlowLayout18.setVgap(0);
      layFlowLayout18.setAlignment(java.awt.FlowLayout.LEFT);
      contactPanel.setLayout(layFlowLayout18);
      contactPanel.add(getLabel1(), null);
      contactPanel.add(getContactField(), null);
   }
   return contactPanel;
}

/**
 * Initializes a label
 *
 * @return java.awt.Label
 */
private java.awt.Label getLabel() {
   if(label == null) {
      label = new java.awt.Label();
      label.setText("Rebate: ");
   }
```

```
   return label;
}

/**
 * Initializes a label
 *
 * @return java.awt.Label
 */
private java.awt.Label getLabel1() {
   if(label1 == null) {
      label1 = new java.awt.Label();
      label1.setText("Contact: ");
   }
   return label1;
}

/**
 * Initializes the rebate textfield
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getRebateField() {
   if(rebateField == null) {
      rebateField = new java.awt.TextField();
      rebateField.setColumns(10);
   }
   return rebateField;
}

/**
 * Initializes the contact textfield
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getContactField() {
   if(contactField == null) {
      contactField = new java.awt.TextField();
      contactField.setColumns(18);
   }
   return contactField;
}

/**
 * Initializes the items panel
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getItemsPanel() {
   if(itemsPanel == null) {
```

```
      itemsPanel = new java.awt.Panel();
      java.awt.BorderLayout layBorderLayout6
         = new java.awt.BorderLayout();
      itemsPanel.setLayout(layBorderLayout6);
      itemsPanel.add(getLabel2(), java.awt.BorderLayout.NORTH);
      itemsPanel.add(getOrderItemScrollpane(),
         java.awt.BorderLayout.CENTER);
   }
   return itemsPanel;
}

/**
 * Initializes a label2
 *
 * @return java.awt.Label
 */
private java.awt.Label getLabel2() {
   if(label2 == null) {
      label2 = new java.awt.Label();
      label2.setText("Items");
      label2.setName("Items");
      label2.setFont(new java.awt.Font("sansserif", 1, 12));
   }
   return label2;
}

/**
 * Initializes the update button
 *
 * @return java.awt.Button
 */
private java.awt.Button getUpdateButton() {
   if(updateButton == null) {
      updateButton = new java.awt.Button();
      updateButton.setLabel("Update order");
      updateButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent event) {
            updateCommand();
         }
      });
   }
   return updateButton;
}

/**
 * Initializes the Back button
 *
 * @return java.awt.Button
 */
```

```
private java.awt.Button getBackButton() {
   if(backButton == null) {
      backButton = new java.awt.Button();
      backButton.setLabel("Back");
      backButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent event) {
            backCommand();
         }
      });
   }
   return backButton;
}


/**
 * Initializes the order fields
 *
 * This method execute a SQL SELECT to bring the data for the order id
 * given in the constructor.
 */
private void initFields() {

   getOrderIdField().setText(String.valueOf(this.orderId));

   try {
      PreparedStatement pstm
         = Database.getInstance().getConnection()
                           .prepareStatement(SQL_ORDER_BYID);
      pstm.setInt(1, this.orderId);
      ResultSet rs = pstm.executeQuery();
      if (rs.next()) {
         getOrderDateField().setText(rs.getDate(1).toString());
         getCustomerNameField().setText(rs.getString(2));


         BigDecimal rebate = rs.getBigDecimal(3);
         if (!rs.wasNull()) {
            getRebateField().setText(
               String.valueOf(rebate.doubleValue()));
         }

         String contact = rs.getString(4);
         if (!rs.wasNull()) {
            getContactField().setText(contact);
         }
      }
      rs.close();
      pstm.close();
   }
```

```
   catch (SQLException e) {
      e.printStackTrace();
      System.err.println(e.getSQLState());
   }
}

/**
 * Initializes the item list
 *
 * This method execute a SQL SELECT to bring the data for the order id
 * given in the constructor. Please note that we have to use the BigDecimal
 * class instead the double primitive, because the ResultSet.getDouble()
 * method is not currently supported on the DB2e JDBC implementation.
 *
 */
private void initItemList() {

   NumberFormat numFormat = DecimalFormat.getInstance();
   numFormat.setMaximumFractionDigits(2);
   numFormat.setMinimumFractionDigits(2);

   try {
      PreparedStatement pstm
         = Database.getInstance().getConnection()
                           .prepareStatement(SQL_ORDERITEMS_BYID);
      pstm.setInt(1, this.orderId);
      ResultSet rs = pstm.executeQuery();
      while(rs.next()) {
         StringBuffer sb = new StringBuffer();
         sb.append(rs.getString(1)).append(" ")
            .append(rs.getInt(2)).append(" ")
            .append(numFormat.format(rs.getBigDecimal(3).doubleValue()))
            .append(" ")
            .append(numFormat.format(rs.getBigDecimal(4).doubleValue()));

         getItemList().add(sb.toString());
      }
      rs.close();
      pstm.close();
   }
   catch (SQLException e) {
      e.printStackTrace();
      System.err.println(e.getSQLState());
   }
}

/**
 * Handles the action for the Back button
 *
```

```
     * Closes this frame and shows the NewOrders Frame
     */
    private void backCommand() {
       this.setVisible(false);
       this.dispose();
       new NewOrdersFrame().show();
    }

    /**
     * Handles the action for the Update order button.
     *
     * This method takes the values from the rebate and contact fields and
     * execute an SQL UPDATE to reflect the changes in the database
     */
    private void updateCommand() {

       try {
          PreparedStatement pstm
             = Database.getInstance().getConnection()
                                .prepareStatement(SQL_UPDATE_ORDER);
          pstm.setInt(3, this.orderId);
          pstm.setBigDecimal(1, new BigDecimal(getRebateField().getText()));
          pstm.setString(2, getContactField().getText());

          pstm.executeUpdate();

          pstm.close();
       }
       catch(SQLException e) {
          e.printStackTrace();
          System.err.println(e.getSQLState());
       }
    }
}
```

SyncFrame

The SyncFrame class shows you how to use the Sync API for Java to develop a customized synchronization client.

The frame has five text fields to enter the synchronization information. When you click the **Sync** button, the sync() method uses this information to connect to the SyncServer and perform the synchronization. The code for the class is shown in Example 12-6.

Note that you have to close the database connection prior to the database synchronization. Refer to the *DB2e Application Development Guide Version 8 Release 1*, SC18-7185, for more information about the Java Sync API.

Example 12-6 SyncFrame.java

```
package com.ibm.ral.itso.mos;

import java.awt.Button;
import java.awt.Frame;
import java.awt.Label;
import java.awt.MenuBar;
import java.awt.Panel;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.SQLException;
import java.util.Properties;

import com.ibm.mobileservices.isync.ISync;
import com.ibm.mobileservices.isync.ISyncConfigStore;
import com.ibm.mobileservices.isync.ISyncDriver;
import com.ibm.mobileservices.isync.ISyncException;
import com.ibm.mobileservices.isync.ISyncManager;
import com.ibm.mobileservices.isync.ISyncProvider;
import com.ibm.mobileservices.isync.ISyncService;
import com.ibm.mobileservices.isync.ISyncSubscriptionSet;

import com.ibm.mobileservices.isync.event.ISyncEvent;
import com.ibm.mobileservices.isync.event.ISyncListener;

/**
 * AWT Frame for synchronization
 *
 * Implements an interface for synchronize the database. It uses the Java API
 * for Synchronization.
 *
 */
public class SyncFrame extends Frame implements ISyncListener {

     // Frame visual controls
     private Panel buttonPanel = null;
     private Button syncButton = null;
     private Panel mainPanel = null;
     private Label label = null;
     private TextField hostNameField = null;
     private TextField hostPortField = null;
     private Label label1 = null;
     private TextField userNameField = null;
     private Label label2 = null;
     private TextField userPasswordField = null;
     private TextField messagesField = null;
     private TextField progressField = null;
```

```
  private Button backButton = null;
  private MenuBar mainMenuBar = null;

/**
 * Constructor for SyncFrame
 *
 */
public SyncFrame() {
   super();
   initialize();
}
/**
 * Initializes the frame
 *
 */
private void initialize() {
     this.add(getButtonPanel(), java.awt.BorderLayout.SOUTH);
     this.add(getMainPanel(), java.awt.BorderLayout.CENTER);
     this.setMenuBar(getMainMenuBar());
     this.setTitle("Sync Orders");

}

private MenuBar getMainMenuBar() {
   if (mainMenuBar == null) {
      mainMenuBar = new MenuBar();
   }
   return mainMenuBar;
}

/**
 * Initializes the button panel
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getButtonPanel() {
   if(buttonPanel == null) {
      buttonPanel = new java.awt.Panel();
      buttonPanel.add(getSyncButton(), null);
      buttonPanel.add(getBackButton(), null);
   }
   return buttonPanel;
}

/**
 * Initializes the Sync button
 *
 * @return java.awt.Button
 */
```

```
private java.awt.Button getSyncButton() {
   if(syncButton == null) {
      syncButton = new java.awt.Button();
      syncButton.setLabel("Synchronize");
      syncButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent event) {
            syncCommand();
         }
      });
   }
   return syncButton;
}

/**
 * Initializes the main panel
 *
 * @return java.awt.Panel
 */
private java.awt.Panel getMainPanel() {
   if(mainPanel == null) {
      mainPanel = new java.awt.Panel();
      java.awt.FlowLayout layFlowLayout1 = new java.awt.FlowLayout();
      layFlowLayout1.setAlignment(java.awt.FlowLayout.LEFT);
      mainPanel.setLayout(layFlowLayout1);
      mainPanel.add(getLabel(), null);
      mainPanel.add(getHostNameField(), null);
      mainPanel.add(getHostPortField(), null);
      mainPanel.add(getLabel1(), null);
      mainPanel.add(getUserNameField(), null);
      mainPanel.add(getLabel2(), null);
      mainPanel.add(getUserPasswordField(), null);
      mainPanel.add(getMessagesField(), null);
      mainPanel.add(getProgressField(), null);
   }
   return mainPanel;
}

/**
 * Initializes a label
 *
 * @return java.awt.Label
 */
private java.awt.Label getLabel() {
   if(label == null) {
      label = new java.awt.Label();
      label.setText("Server: ");
      label.setFont(new java.awt.Font("sansserif", 1, 12));
   }
   return label;
```

```
}

/**
 * Initializes the hostname textfield
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getHostNameField() {
   if(hostNameField == null) {
      hostNameField = new java.awt.TextField();
      hostNameField.setColumns(14);
   }
   return hostNameField;
}

/**
 * Initializes the host port textfield
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getHostPortField() {
   if(hostPortField == null) {
      hostPortField = new java.awt.TextField();
      hostPortField.setColumns(3);
   }
   return hostPortField;
}

/**
 * Initializes a label
 *
 * @return java.awt.Label
 */
private java.awt.Label getLabel1() {
   if(label1 == null) {
      label1 = new java.awt.Label();
      label1.setText("User: ");
      label1.setFont(new java.awt.Font("sansserif", 1, 12));
   }
   return label1;
}

/**
 * Initializes the username textfield
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getUserNameField() {
   if(userNameField == null) {
```

```
      userNameField = new java.awt.TextField();
      userNameField.setColumns(15);
   }
   return userNameField;
}

/**
 * Initializes a label
 *
 * @return java.awt.Label
 */
private java.awt.Label getLabel2() {
   if(label2 == null) {
      label2 = new java.awt.Label();
      label2.setText("Password: ");
      label2.setName("");
      label2.setFont(new java.awt.Font("sansserif", 1, 12));
   }
   return label2;
}

/**
 * Initializes the user's password textfield
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getUserPasswordField() {
   if(userPasswordField == null) {
      userPasswordField = new java.awt.TextField();
      userPasswordField.setEchoChar('*');
      userPasswordField.setColumns(15);
   }
   return userPasswordField;
}

/**
 * Initializes the messages textfield
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getMessagesField() {
   if(messagesField == null) {
      messagesField = new java.awt.TextField();
      messagesField.setColumns(28);
      messagesField.setEnabled(false);
   }
   return messagesField;
}
```

```
/**
 * Initializes the progress textfield
 *
 * @return java.awt.TextField
 */
private java.awt.TextField getProgressField() {
   if(progressField == null) {
      progressField = new java.awt.TextField();
      progressField.setEnabled(false);
      progressField.setColumns(15);
   }
   return progressField;
}

/**
 * Initializes the Back button
 *
 * @return java.awt.Button
 */
private java.awt.Button getBackButton() {
   if(backButton == null) {
      backButton = new java.awt.Button();
      backButton.setLabel("Back");
      backButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent event) {
            backCommand();
         }
      });
   }
   return backButton;
}

/**
 * Starts the synchronization.
 *
 * This method has to close the database connection first. Also it disable
 * all the controls while the synchronization is active. It enables the
 * controls and opens the database when the synchronization end.
 */
private void syncCommand() {
   getHostNameField().setEnabled(false);
   getHostPortField().setEnabled(false);
   getUserNameField().setEnabled(false);
   getUserPasswordField().setEnabled(false);
   getSyncButton().setEnabled(false);
   getBackButton().setEnabled(false);

   try {
      Database.getInstance().close();
```

```
            sync(getHostNameField().getText(), getHostPortField().getText(),
               getUserNameField().getText(), getUserPasswordField().getText());

            Database.getInstance().open(MobileOrderSystem.DB2E_DATABASE_PATH);
         }
         catch(SQLException e) {
            e.printStackTrace();
            System.err.println(e.getSQLState());
         }
         getHostNameField().setEnabled(true);
         getHostPortField().setEnabled(true);
         getUserNameField().setEnabled(true);
         getUserPasswordField().setEnabled(true);
         getSyncButton().setEnabled(true);
         getBackButton().setEnabled(true);
      }

      /**
       * Closes the frame and opens the NewOrders frame.
       */
      private void backCommand() {
         this.setVisible(false);
         this.dispose();
         new NewOrdersFrame().show();
      }

      /**
       * Event handler for synchronization
       *
       * This is a callback method that the sync engine use to communicate what
       * is happening with the synchronization process. You can use the message
       * codes to produce more readables messages.
       *
       * @see com.ibm.mobileservices.isync.event.ISyncListener#eventIssued(
       * ISyncEvent)
       */
      public int eventIssued(ISyncEvent evt ) {
         int evtType = evt.getEventType();
         switch(evtType) {
            // display event status
            case ISync.EVTTYPE_INFO:
            case ISync.EVTTYPE_ERROR:

               getMessagesField().setText(
                  "EventType:"+evtType+". EventCode:"+evt.getEventCode());
               System.out.println ("Event Type: " + evtType );
               System.out.println ("Event Code: " + evt.getEventCode() );
               getProgressField().setText(
```

```
                "Progress: " + evt.getSyncProgress());
             System.out.println ("Progress: " + evt.getSyncProgress());

             return ISync.RTNCB_DONE;
          case ISync.EVTTYPE_RETRY:
             return ISync.RTNCB_REPLY_YES;
          case ISync.EVTTYPE_CONFLICT:
             return ISync.RTNCB_DONE;
          // ignore other event types
          default:
          break;
       }
       // let sync engine take default action
       return ISync.RTNCB_DEFAULT ;
    }

    /**
     * The worker synchronization method
     * .
     * @param host The DB2e SyncServer hostname or IP
     * @param port The port where the sync engine is listening
     * @param user
     * @param password
     */
    private void sync(String host, String port, String user, String password) {

       ISyncProvider provider = null;
       ISyncService service = null;
       ISyncConfigStore config = null;
       ISyncDriver syncer = null;
       String path = MobileOrderSystem.DB2E_DATABASE_PATH;
       ISyncSubscriptionSet ssArr[] = null;
       int rc = 0;
       try {
          // Load the sync provider class

Class.forName("com.ibm.mobileservices.isync.db2e.jni.DB2eISyncProvider");

          // Get an instance DB2eISyncProvider using the DB2e protocol
          provider = ISyncManager.getISyncProvider("isync:db2e:");

          // Create the user settings needed for synchronization
          Properties userProp = new Properties();
          userProp.put("isync.user", user);
          userProp.put("isync.password", password);

          // Get an instance of synchronization service from the provider
          service = provider.createSyncService(host+":"+port, userProp);
```

```
    // Get an instance of the configuration store
    config = service.getConfigStore(path);

    // Get an instance of the sync driver to perform synchronization
    syncer = config.getSyncDriver();


    // Set the listener object for event notification from the
    // syncer object during synchronization
    syncer.setSyncListener(this);

    // Perform synchronization on all enabled subscription sets
    rc = syncer.sync();
    switch (rc) {
       case ISync.RTN_SUCCEEDED:
          getMessagesField().setText("Synchronization succeeded");
          break;
       case ISync.RTN_CANCELED:
          getMessagesField().setText("Synchronization canceled");
          break;
       default:
          getMessagesField().setText("Synchronization failed");
       break;
    }

    ssArr = config.getSubscriptionSets();
    for (int i=0 ; i < ssArr.length; i++ ) {
       System.out.print ("Subscription Set: " +ssArr[i].getName()
          + " Status: ");
       switch(ssArr[i].getStatus()) {
          case ISync.STATUS_READY:
             System.out.println("READY");
             break;
          case ISync.STATUS_COMPLETED:
             System.out.println ("COMPLETED");
             break;
          case ISync.STATUS_CANCELED:
             System.out.println ("CANCELED");
             break;
          default:
             System.out.println ("FAILED");
             break;
       }
    }
}
catch (ISyncException ie) {
    getMessagesField().setText(
          "An exception has ocurred. Code:"+ie.getCode());
```

```
            ie.printStackTrace();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        finally {
            // Close and free all allocated resources
            //
            try {
                if (syncer != null) {
                    syncer.close();
                    syncer = null;
                }
                if (config != null) {
                    config.close();
                    config = null;
                }
                if (service != null) {
                    service.close();
                    service = null;
                }
            }
            catch(ISyncException ie2) {
                getMessagesField().setText(
                    "An exception has ocurred. Code:"+ie2.getCode());
                ie2.printStackTrace();
            }
        }
    } // end

}
```

12.4.5 Creating the device database

You must perform an initial synchronization between the device and the server using the DB2e ISync client to initialize the Db2e database. The directory for the database's files according to the code must be \ordsysdb. Refer to *IBM WebSphere Everyplace Access V5 Handbook for Developers and Administrators Volume III: E-Mail and Database Synchronization*, SG24-6676 for more information about the Db2e Sync process. In a real application you should put some initialization code in the program or a call to the synchronization module if the database has not been initialized.

12.4.6 Set up the build and test environment

To set up the build and test environment, you have to perform the following tasks:

1. Create the device configuration
2. Create the build target
3. Create the launch configuration

Create a device configuration

To configure the device target for the project, follow these steps:

1. Select **Devices -> Configure** from the menu bar to open the Devices Configurations window.
2. On the Devices tree pane, select the Pocket PC Handheld device and click **New** to create a new device profile.
3. For the new device profile you need to specify the J9 runtime location, the Application install location, and the Shortcut install location. For the sample application, use the parameters from Table 12-2, as shown in Figure 12-12 on page 403. Click **OK**.

Table 12-2 Device configuration parameters

Parameter	Value
Device Name	MyPocketPC
J9 Runtime location	\Program Files\J9\PPRO10
Application install location	\MobileOrderSystem
Shortcut install location	\Windows\Start Menu

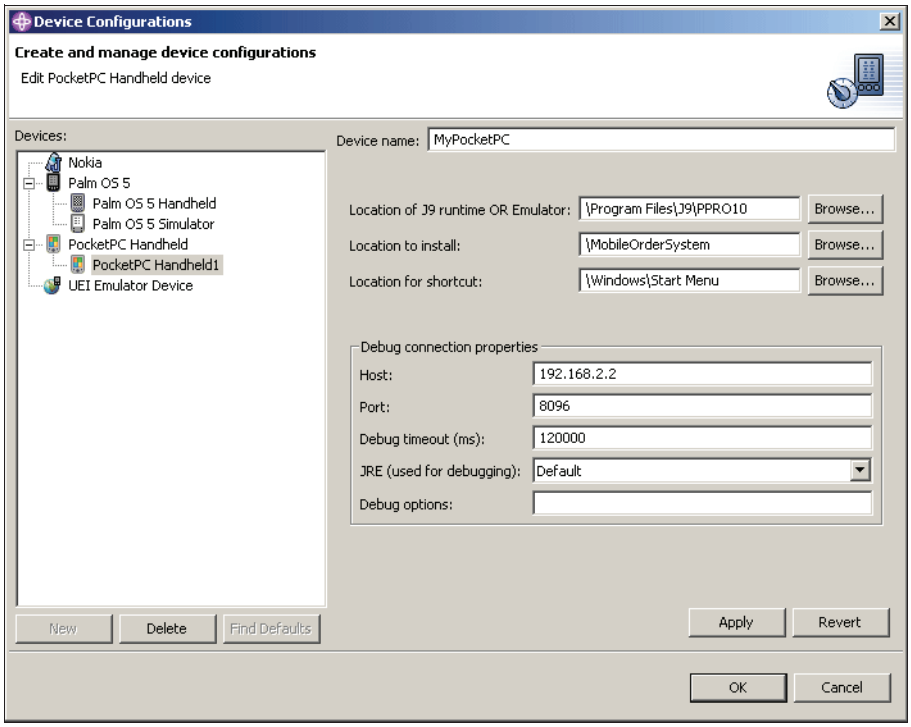


Figure 12-12 Creating a device configuration

Create a build target

You have to define a build target for our application. Follow these steps:

1. Right-click on the project name and select **Device Developer Builds**. Click **Add.**
2. Select **Optimized JXE.** Click **Next**.
3. Verify that **com.ibm.ral.itso.mos.MobileOrderSystem** class as the main class for the application and keep Remove unused classes as shown in Figure 12-13. Click **Next**.

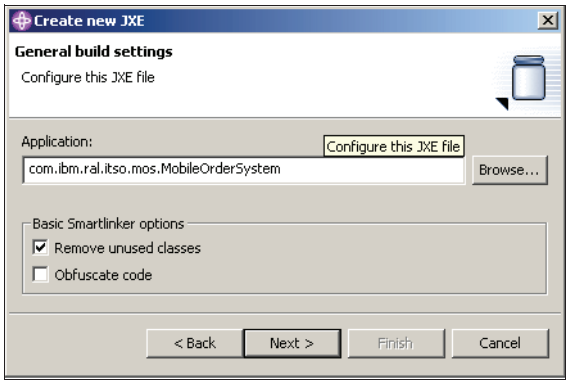


Figure 12-13 Selecting the main class for the build configuration

4. Select **J9 for Windows Mobile 2003** as the platform and click **Next**.
5. Select the libraries to be included in the file as shown in Figure 12-14. Click **Finish**.

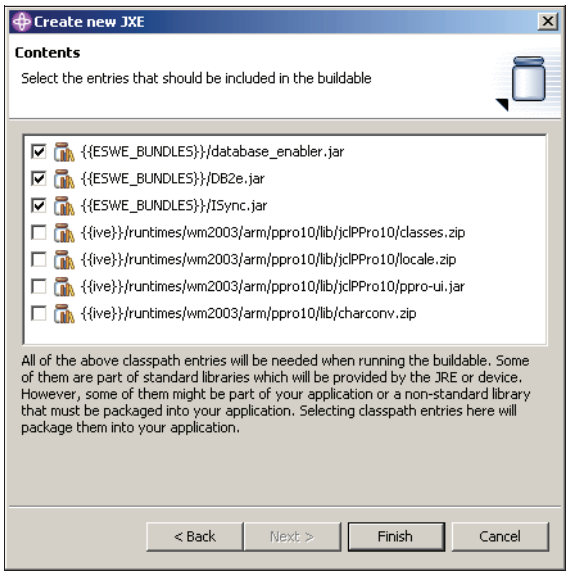


Figure 12-14 Selecting libraries to be included in the application

6. You see the new build configured in the view. Behind the scenes, the wizard has created an Ant task that runs the SmartLinker tool to build the application as a JXE file that the J9 runtime in the device will execute. Click **Close**.

Include the Sync Event handler method in the jxe file

By default the SmartLinker removes all the methods than are not explicitly called in the code. Because the SyncFrame.eventIssued method is a callback one, it will be removed from the build. You should instruct SmartLinker to keep the above mentioned method. Follow these steps:

1. Expand wm2003arm under MobileOrderSystem project and locate the MobileOrderSystem.jxeLinkOptions file, as shown in Figure 12-15. Open it.

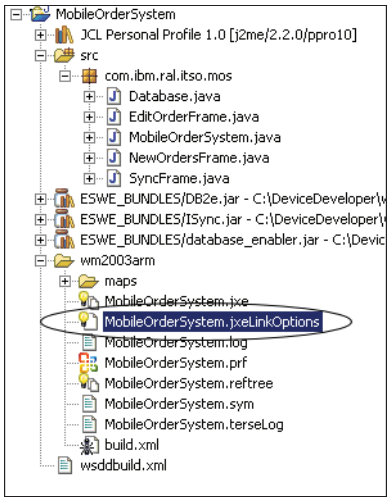


Figure 12-15 JXE Link options

2. Go to the **In/Exclusion** tab and select **Include methods** from the first combo box as shown in Figure 12-16. Click **New**.

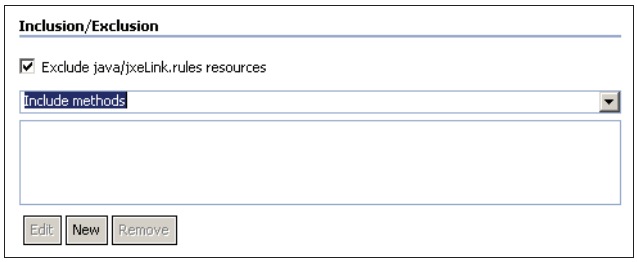


Figure 12-16 Including a method in the build

3. Enter `com.ibm.ral.itso.mos.SyncFrame.eventIssued` in the Rule pattern field and click **OK.** You should see a new entry in the textbox as shown in Figure 12-17 on page 406.

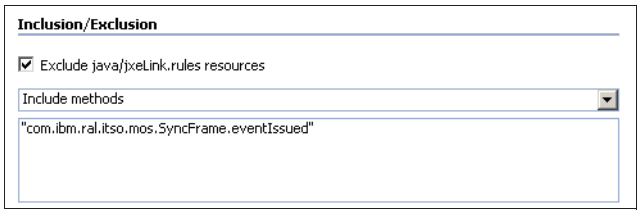


Figure 12-17 SyncFrame.eventIssued method included

4. Save and close the file.

Create the launch configuration

To launch the application on the Pocket PC, you have to create a launch configuration. Follow these steps:

1. Select **Run -> Run...** from the menu bar. Select **Java on Device** to create a new launch configuration for the Pocket PC. Click **New**.
2. A new launch configuration is created. Enter `MyConfiguration` as the name and select **MyPocketPC** as the device. Note that the Project field has `MobileOrderSystem` as the value. Also note the build selected on Java Application field as shown in Figure 12-18 on page 407. Click the **Arguments** tab.

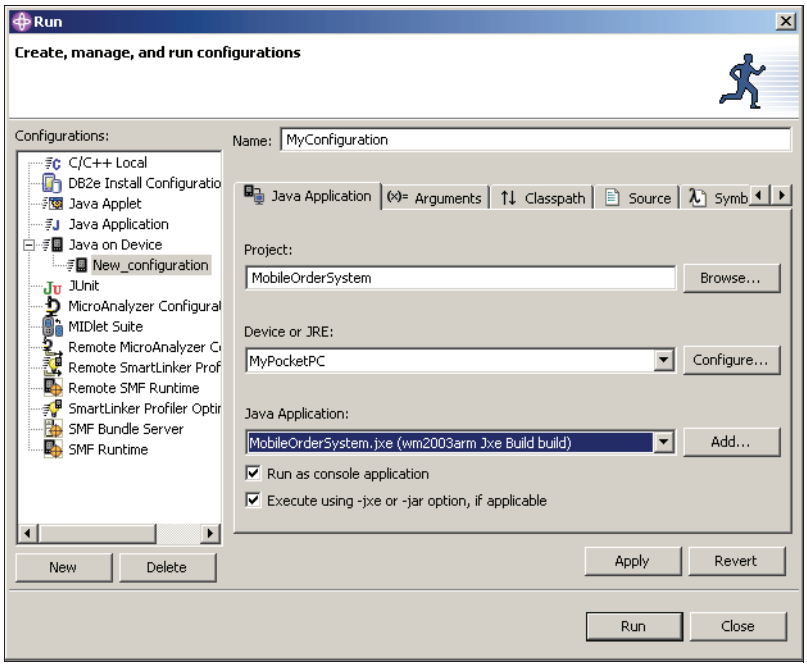


Figure 12-18 Configuring a new Java Application launch

3. Add the following entry to the VM arguments fields, as shown in Figure 12-19 on page 408:

   ```
   -classpath
   \Windows\dbe2jdbc.jar;\Windows\isync4j.jar;\MobileOrderSystem\Mobile
   OrderSyste.jxe.
   ```

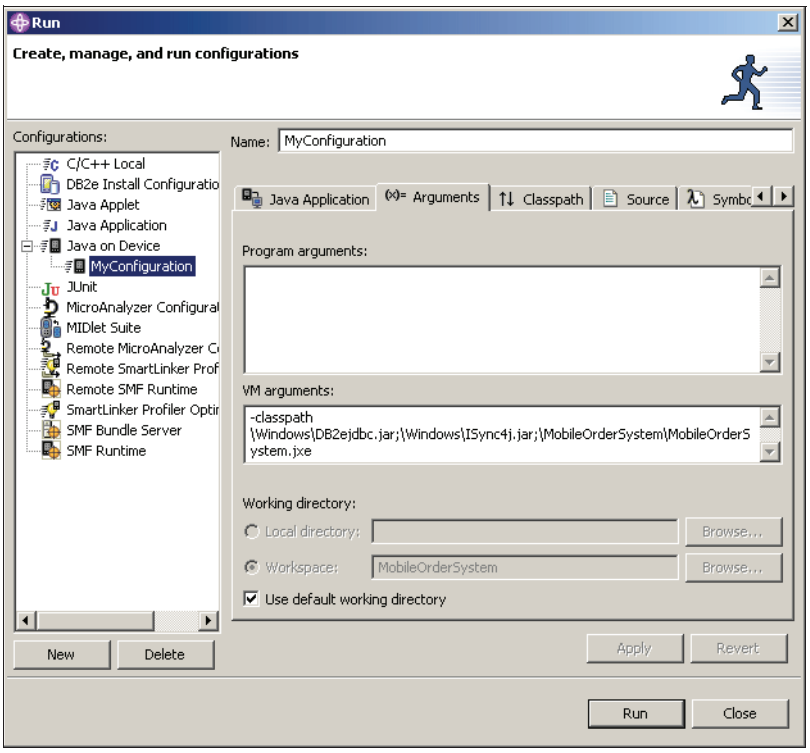


Figure 12-19 Configuring the classpath for the application

4. Click the **Classpath** tab, then click the **User classes** tab. Here, you have to remove the entries in order to avoid to have a duplicate classpath statement in the launch shortcut. To do this, uncheck **Use default classpath** and delete all the entries as shown in Figure 12-20 on page 409.

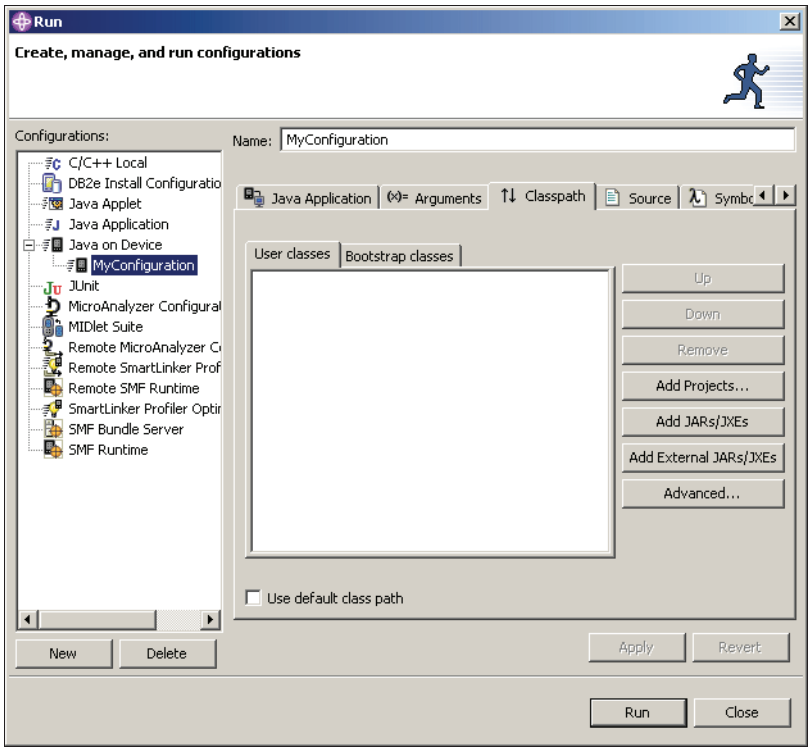


Figure 12-20 Configuring the user classes

5. Click **Close** to finish.

12.4.7 Testing the application

To test the application follow these steps:

1. Make sure that the ActiveSync communication link between your desktop and your Pocket PC is enabled.
2. On WebSphere Studio Device Developer, select **Run -> Run...** from the menu bar. The Launch configuration window opens. Select **Java on Device-> MyConfiguration** from the tree view. Click **Run** to launch the application.
3. Wait while WebSphere Studio Device Developer links and launches your application. A window may warn you that certain methods were not found, as shown in Figure 12-21 on page 410. If you see this window, you can safely ignore it. Just click **OK**.

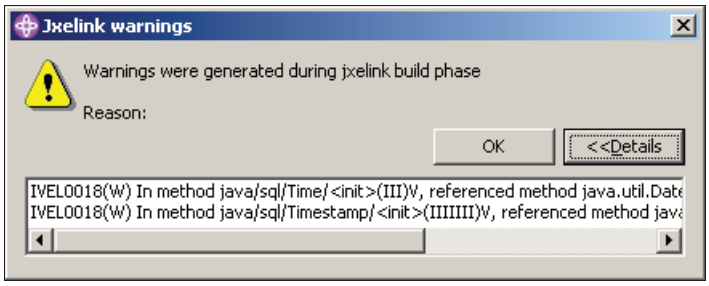


Figure 12-21 SmartLink warning message

Tip: If you experience some problem connecting to the Pocket PC via ActiveSync when the launch process for Pocket PC copies files to the device, try restarting the workbench.

Now you can see on your Pocket PC device the New Orders frame, as shown in Figure 12-22.



Figure 12-22 New orders

If you select an order and click **Delete,** the order is deleted from the database and, of course, from the list. If you select an order and click **Edit**, you can see the Edit Order frame as shown in Figure 12-23 on page 411.

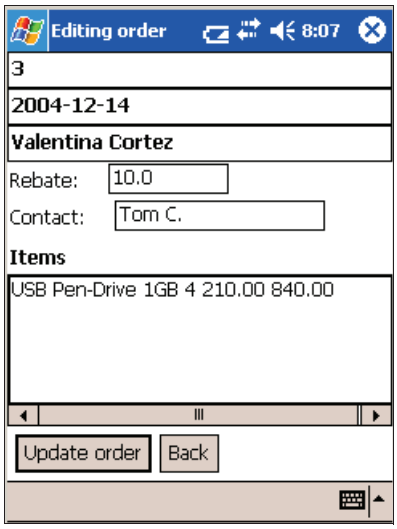


Figure 12-23 Edit order

Enter some values for the Contact and Rebate fields. Consider that the sample application doesn't have any validation code, so enter meaningful values. Click **Update order**, and the order is updated on the database. Click **Back** to return to the New orders frame.

On the New Orders frame, click **Synchronize** to show the Sync Frame. Enter your SyncServer host name, port, user name and password, as shown in Figure 12-24 on page 412.

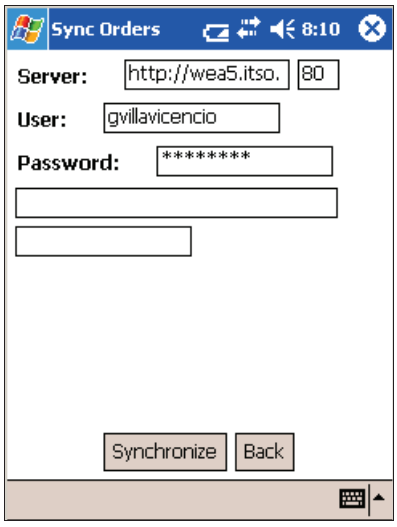


Figure 12-24 Sync orders

Click **Synchronize** to start the synchronization. You see the synchronization messages and the progress in their respective text fields. When the synchronization ends, you see a message that indicate the result of the synchronization, as shown in Figure 12-25.

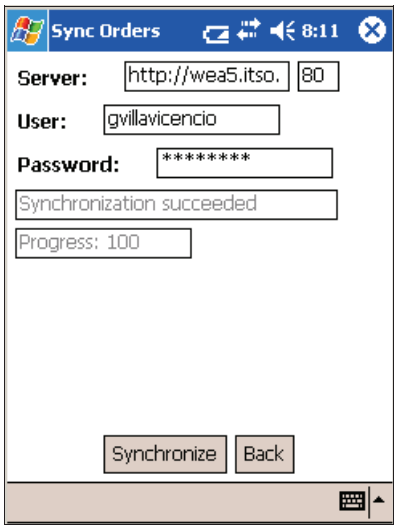


Figure 12-25 Sync ended

Part 4

Appendixes

© Copyright IBM Corp. 2005. All rights reserved.

A

Portlet development platform installation

This appendix describes a sample installation and the steps necessary to prepare a single machine running Windows 2000 Professional or Server to be used for WebSphere application development and testing, including:

- Installation of WebSphere Studio Site Developer V5.1.1
- Installation of WebSphere Studio Device Developer V5.7.0
- Installation of the Everyplace Toolkit V5.0.1 for WebSphere Studio Site Developer which includes the following:
  - Installation of WebSphere Portal Toolkit 5.0.2.1
  - Installation of WebSphere Portal V5.0 for Test Environment
  - Installation of Multi Device Authoring Tools
  - Installation of XHTML + Voice Toolkit Technical Preview
  - Installation of WebSphere Micro Environment Tools

© Copyright IBM Corp. 2005. All rights reserved.

A.1 Prerequisites

In this section, we list the prerequisites for installing the Everyplace Toolkit V5.0.1.

Recommended minimum machine requirements

- 1 GHz Pentium® III or higher
- 1 GB RAM
- 10 GB hard drive
- LAN attached or Loopback support, TCP/IP stack must be installed

Software requirements

- Windows 2000 Professional or Server and Service Pack 3 or higher
- WebSphere Studio Site Developer V5.1.1
- WebSphere Everyplace Toolkit V5.0.1 for WebSphere Studio Site Developer
- WebSphere Portal V5.0

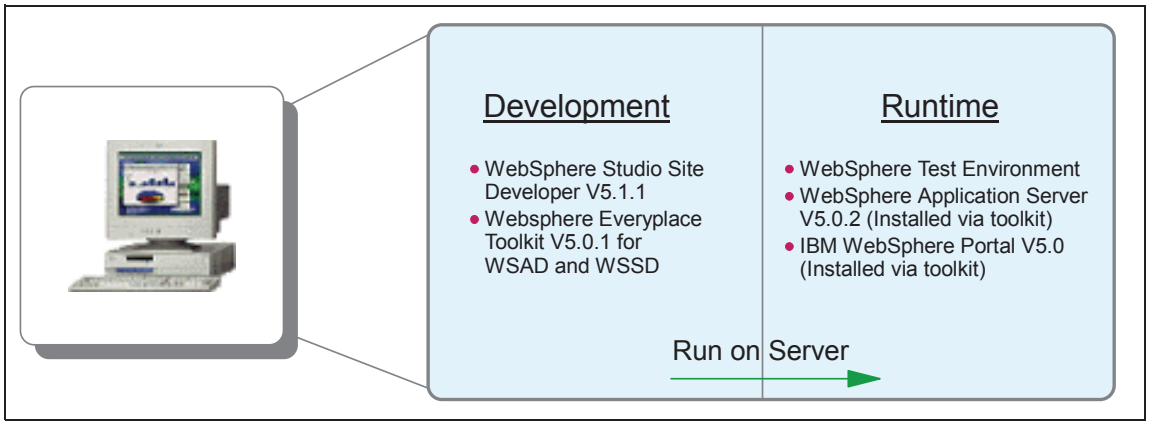


Figure A-1 Portlet development workstation

A.2 WebSphere Studio Site Developer V5.1.1

To install the WebSphere Everyplace Toolkit, you must install WebSphere Studio Site Developer V5.1.1 and associated fixes.

A.2.1 Install WebSphere Studio Site Developer

To install the WebSphere Studio Site Developer, perform the following steps:

1. Insert WebSphere Everyplace Access CD 9-2. Double-click **launchpad.exe**. You will see the Welcome window as illustrated in Figure A-2. Click **Install IBM WebSphere Studio Site Developer** to continue.

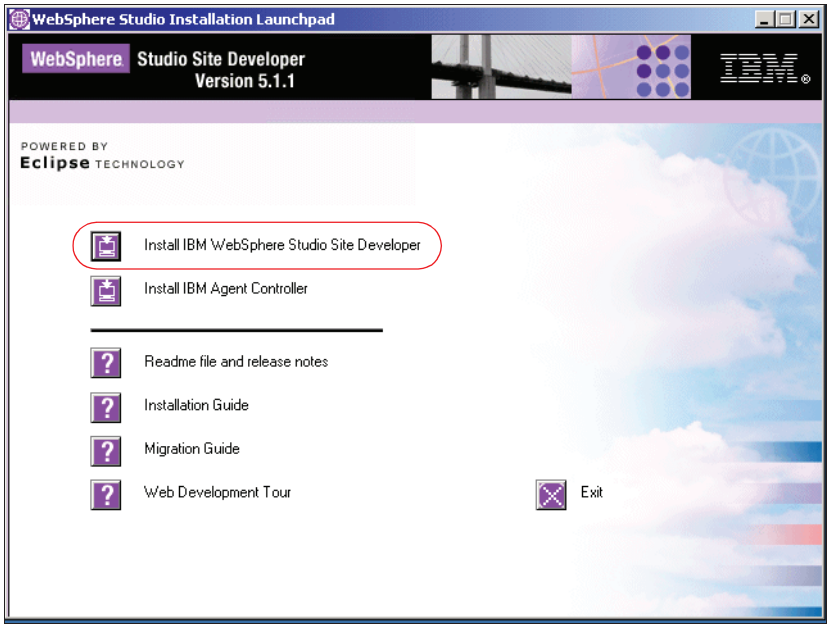


Figure A-2 WebSphere Studio welcome window

2. Click **Next** to continue past the welcome window.
3. Select I accept the terms in the license agreement and click **Next**.
4. Choose a short path name to install WebSphere Studio. Do not use the default install directory. The path name must contain eight or fewer alphanumeric characters and no spaces. Click **Next** to continue.

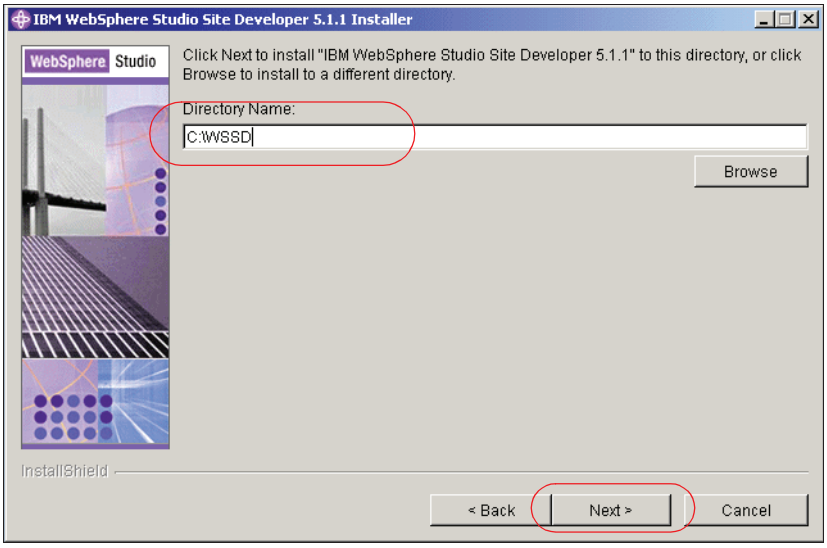


Figure A-3 Installation path window

5. On the features window, select WebSphere Application Server 5.0.2. Then click **Next** to continue.

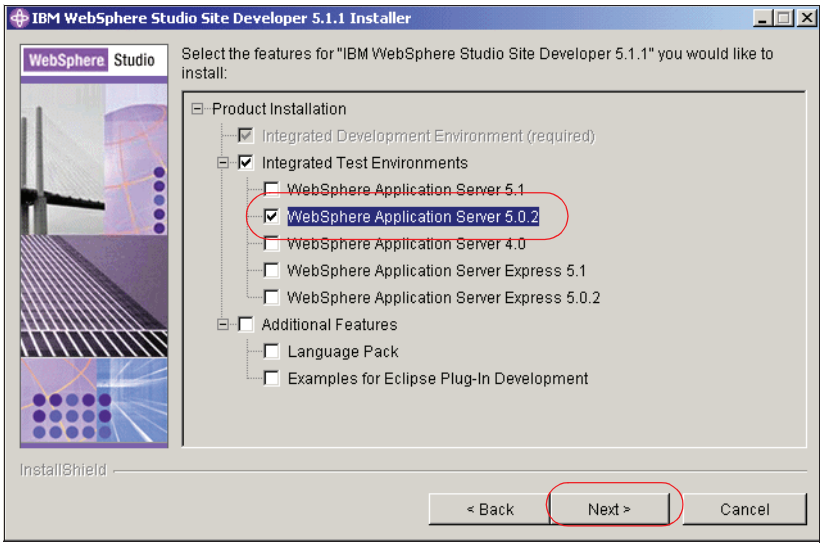


Figure A-4 Test Environment selection

6. Review the summary information and click **Next** to begin the installation.

7. When prompted, insert WSSD install CD 2. Click **OK** to complete the installation.

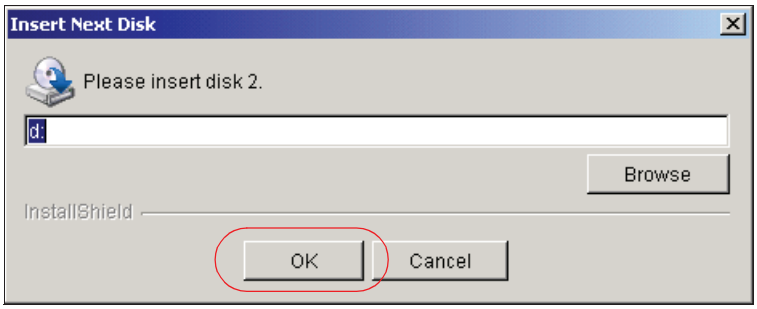


Figure A-5 Insert next CD window

When installation is complete, you will see a window that indicates that the installation was successful. Click **Finish** to complete the installation.

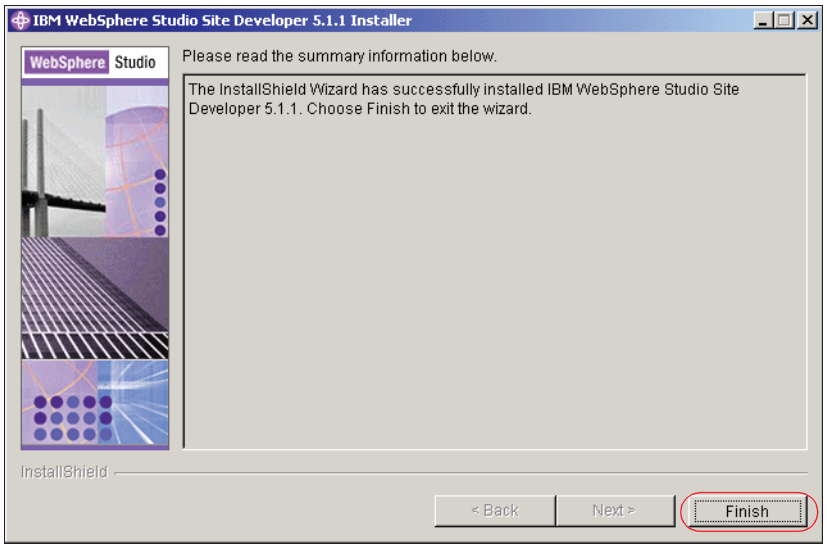


Figure A-6 Installation complete

A.2.2 Apply fixes to WebSphere Studio Site Developer

There are two fixes for WebSphere Studio V5.1.1. The first is Interim Fix 002 for WebSphere Studio. The second applies interim fixes to WebSphere Application Server which is used when you test and debug Web and portlet applications with the WebSphere Studio integrated test environment.

A.2.2.1 Apply Interim Fix 002 for WebSphere Studio

Perform the following steps to apply Interim Fix 002. This fix must be installed prior to installing the Everyplace Toolkit.

1. Locate the WebSphereStudioInterimFix002.zip file on the Everyplace Toolkit CD.
2. Extract this file to a directory on your hard drive.
3. From the WebSphere Studio menu bar, select **Help->Software Updates->Update Manager**.

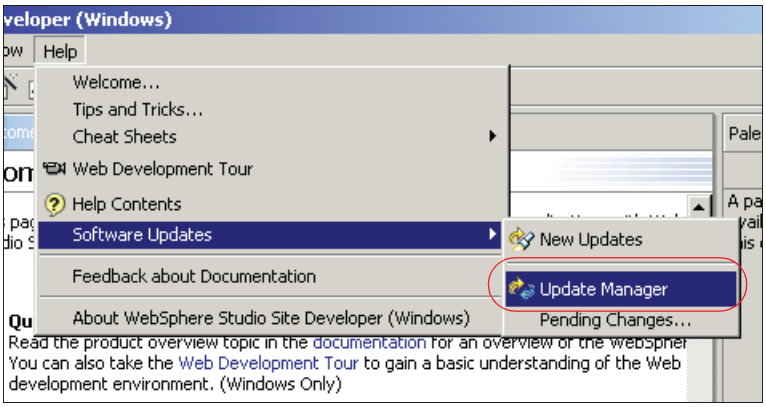


Figure A-7 Opening the update manager

4. In the Features Updates view, expand the My Computer section to locate the directory to where you extracted the update files.

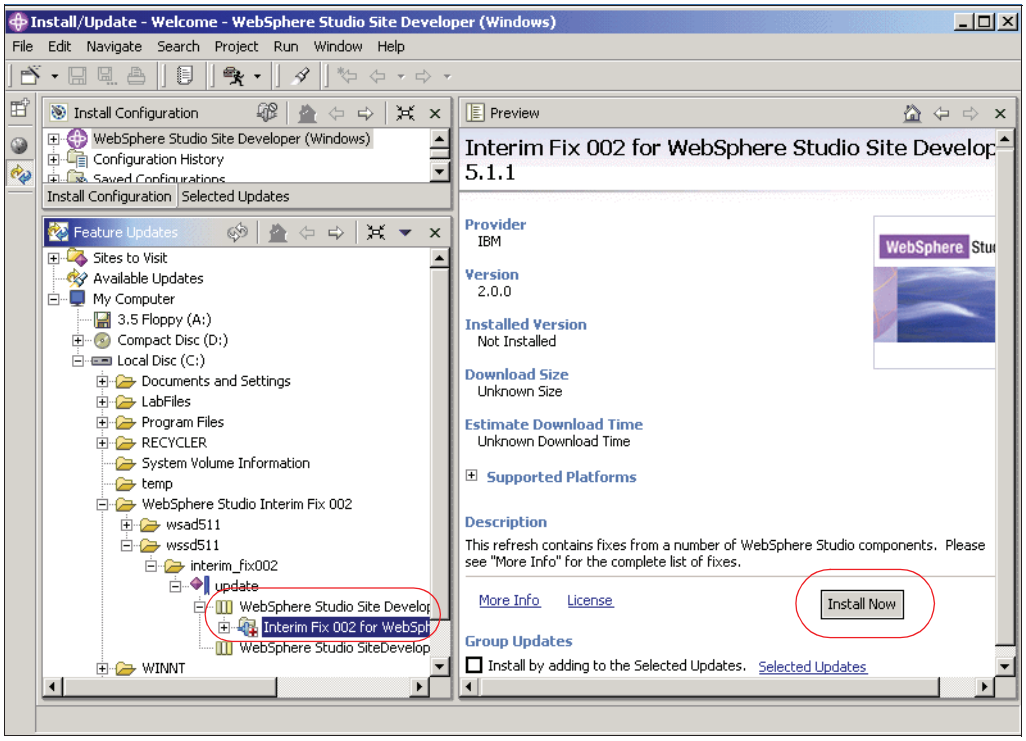


Figure A-8 Installing Interim Fix 002

5. Select **Interim Fix 002 for WebSphere Studio Site Developer** in the /wssd511/interim_fix002/update/WebSphere Studio Site Developer 5.1.1 Interim Fixes folder.
6. In the preview area, click the **Install Now** button as indicated in Figure A-8.
7. Click **Next** on the feature install window to start installation.

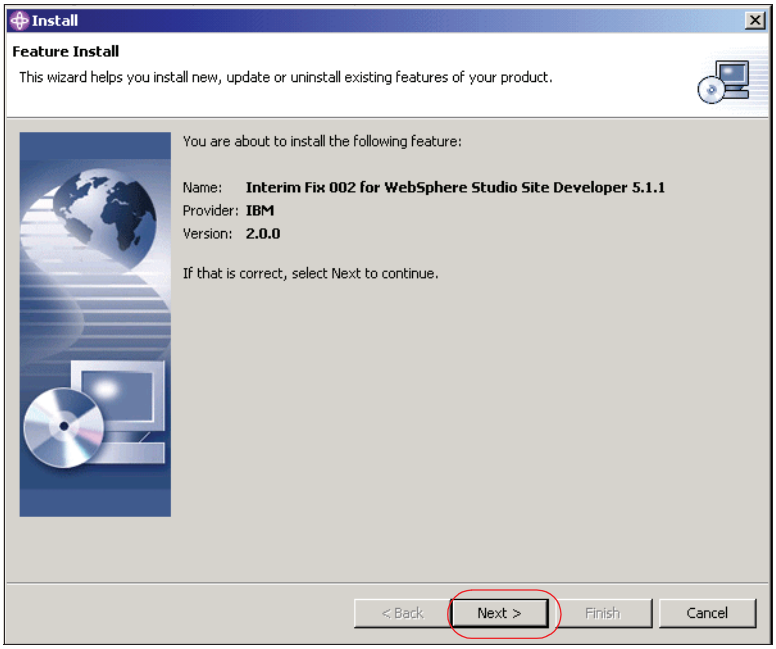


Figure A-9 Feature Install window

8. Accept the license agreement and click **Next** to continue.
9. On the optional features window, accept the default selections. Do not modify these selections as this may result in installation errors.

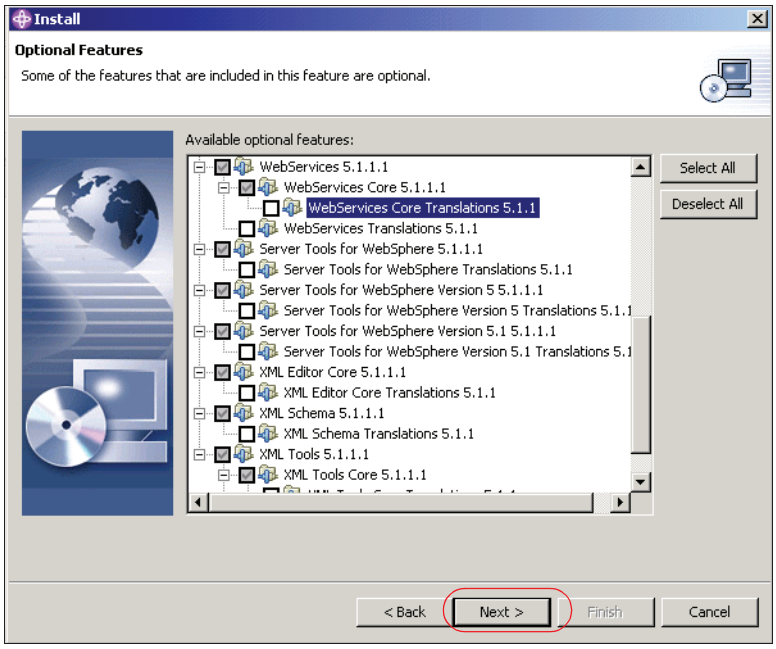


Figure A-10 Optional features window

10. Click **Finish** to begin installation.
11. When you receive the feature verification window, click **Install** to install the update.

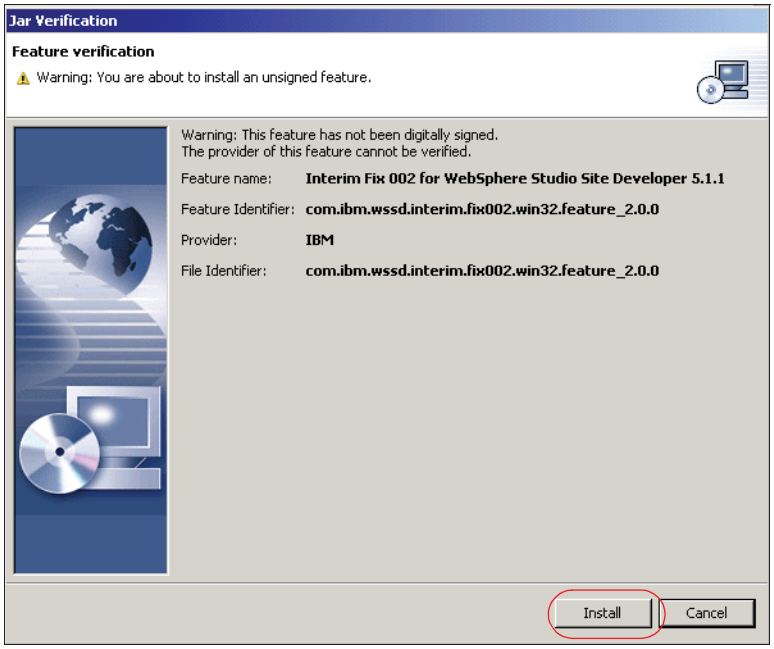


Figure A-11 Feature verification window

12. When installation is complete, you will be prompted to restart the workbench. Click **Yes** to do this and finish this installation.

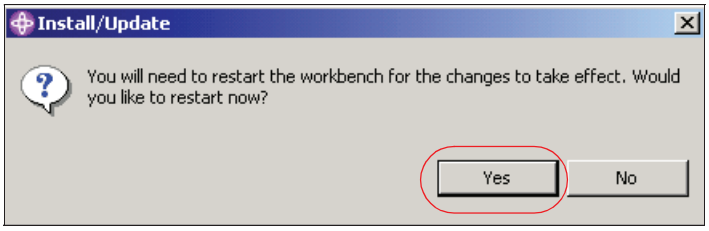


Figure A-12 Restart the workbench

A.2.2.2 Apply interim fixes for WebSphere Application Server

Perform the following steps to apply the interim fixes for WebSphere Application server. This fix must be installed prior to installing the Everyplace Toolkit.

1. Locate the WASInterimFixes.zip file on the Everyplace Toolkit CD.
2. Extract this file to the directory where you installed WebSphere Studio. These files will be extracted to a directory named update. For example, C:\WSSD\update.

3. Modify the InstallWASfixes.bat file in this directory so that WSXD_HOME is set to the directory where you installed WebSphere Studio. For example, SET WSXD_HOME=C:\WSSD.
4. Run InstallWASfixes.bat from a command prompt to apply the fixes.

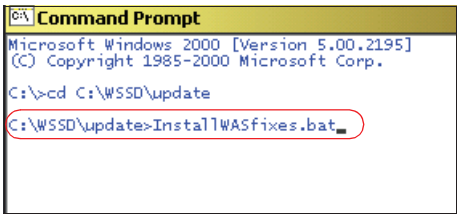


Figure A-13 WAS fixes installation

5. Check the message displayed in the command prompt window to verify a successful installation.

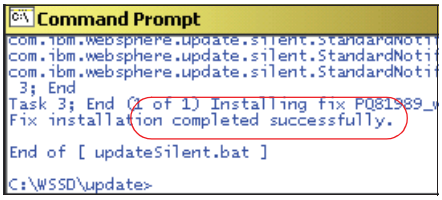


Figure A-14 Installation complete

A.3 WebSphere Studio Device Developer

To install the WebSphere Micro Environment Tools which are included with the WebSphere Everyplace Toolkit, you must install WebSphere Studio Device Developer. If you plan to use the WebSphere Workplace Client Technology™, Micro Edition Tools, you must install Device Developer on top of WebSphere Studio.

If you choose not to install the WebSphere Micro Environment tools, you can choose not to install WebSphere Studio Device Developer. Follow these instructions to install this product:

1. Insert the WSSD installation CD and double-click **launchpad.exe**.
2. Click **Install WebSphere Studio Device Developer** from the installation welcome window.

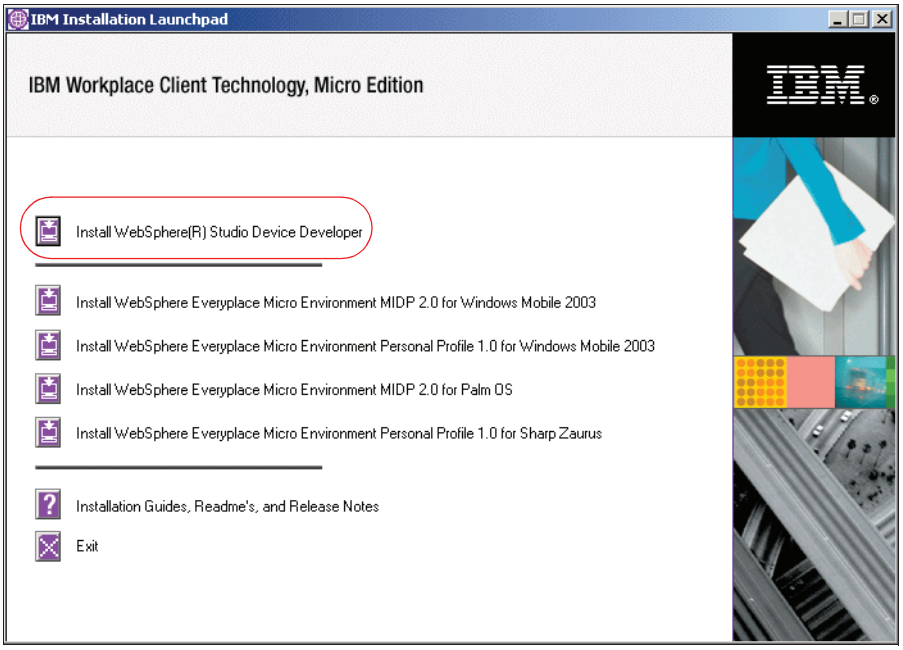


Figure A-15 IBM Workplace Client Technology, Micro Edition

3. Click **Next** to continue past the welcome window.
4. Accept the terms of the license agreement. Click **Next** to continue.
5. Accept the default installation directory and click **Next** to continue.
6. Click **Yes, enable it** to enable WebSphere Studio Device Developer in WebSphere Studio Site Developer.

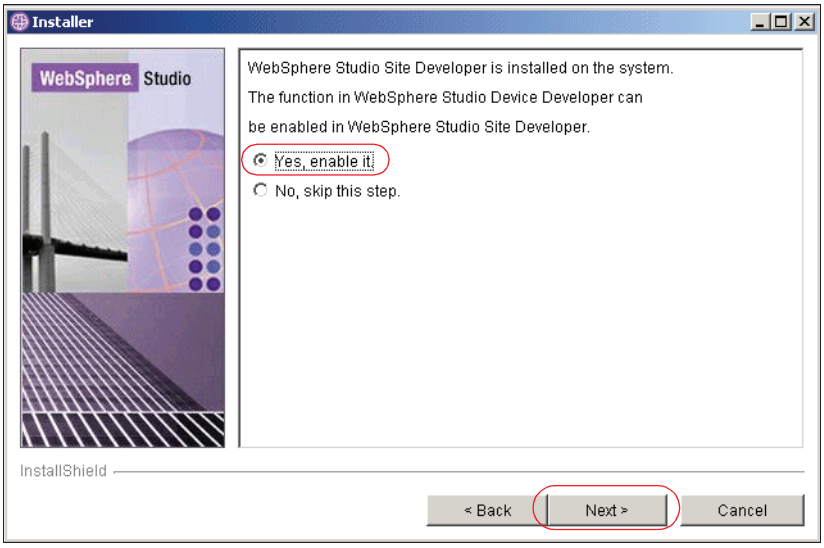


Figure A-16 Enabling Device Developer within Site Developer

7. Review the summary information and click **Next** to begin the installation.
8. Click **Finish** when installation has completed to finish the installation.

A.4 WebSphere Everyplace Toolkit V5.0.1

The WebSphere Everyplace Toolkit enables you to develop Web applications for mobile devices.

A.4.1 Begin Everyplace Toolkit installation

To begin installation of the toolkit, follow these instructions:

1. Run the Everyplace Toolkit installation program located on the WebSphere Everyplace Toolkit CD.
2. Select the language in which you want to install the toolkit. Click **OK** to continue.
3. Click **Next** to continue past the welcome window.
4. Accept the terms of the license agreement and click **Next**.
5. The installation program will detect that the Portal Toolkit is not installed on your machine and will prompt you to install it. Portal Toolkit is a prerequisite

for the Everyplace Toolkit. Continue with the instructions in the following section.

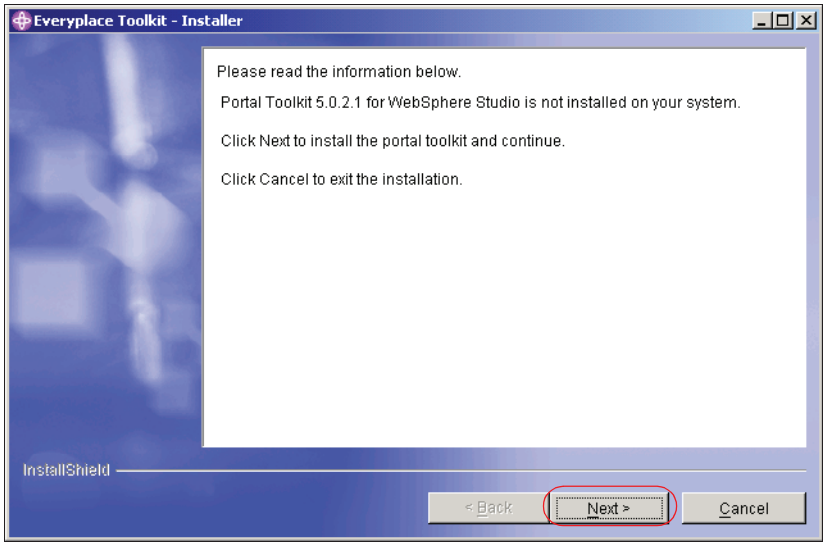


Figure A-17 Install Portal Toolkit window

A.4.2 Install Portal Toolkit V5.0.2.1

The Portal Toolkit must be installed to use the Everyplace Toolkit.

A.4.2.1 Install the Portal Toolkit

To install the Portal Toolkit, follow these instructions:

1. The Portal Toolkit installation will prompt you to select a temporary location to extract the installation files. Select the default and click **Next** to continue.

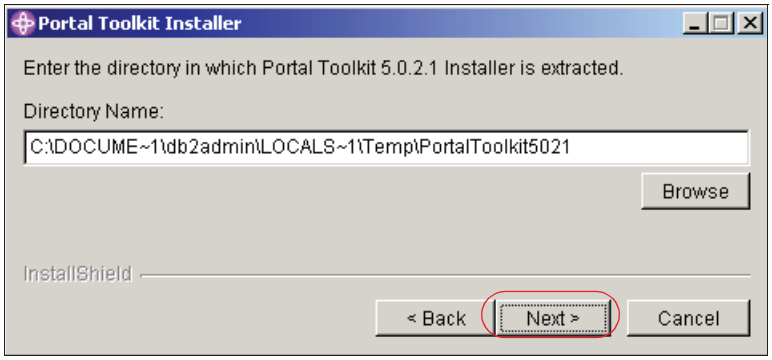


Figure A-18 Temporary location window

2. Select the language in which you want to install the toolkit. Click **OK** to continue.
3. Click **Next** to continue past the welcome window.
4. Accept the terms of the license agreement and click **Next** to continue,
5. Make sure that Portal Toolkit 5.0.2.1 and WebSphere Portal V5.0 for Test Environment are both selected. Click **Next** to continue,

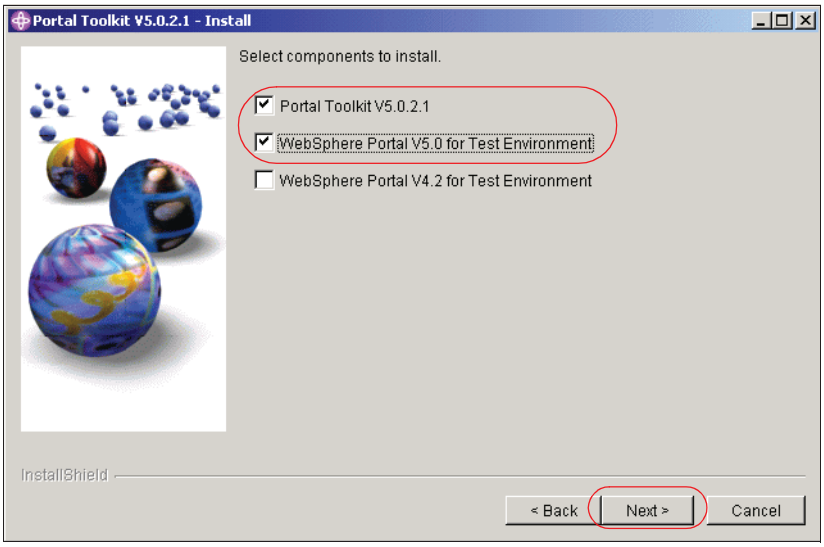


Figure A-19 Components window

6. Click **Next** to accept the default installation directory for the toolkit.

7. Insert the WebSphere Portal V5.0 CD into your drive and select the wpsinstall.jar file from the /wps directory on this CD. Click **Next** to continue the installation.

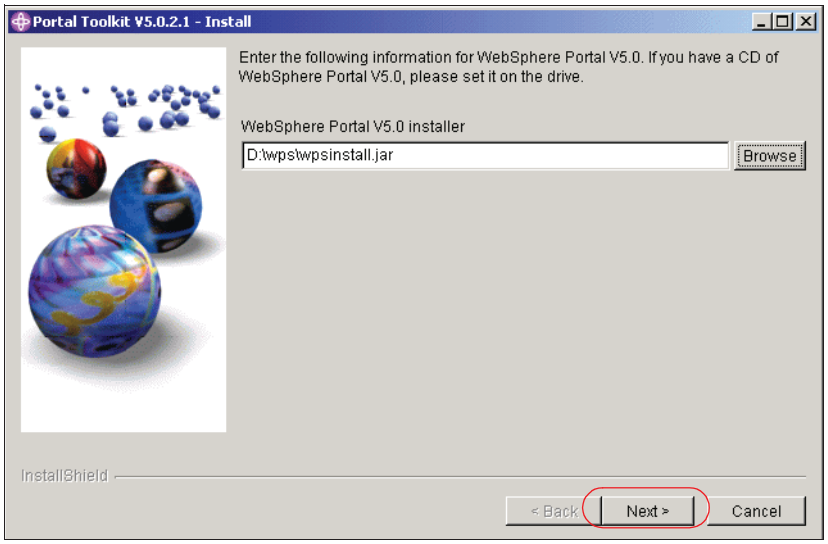


Figure A-20 Installing the Portal runtime

8. Review the summary information and click **Next** to start the installation.
9. Click **Finish** once installation has completed.

A.4.2.2 Install WebSphere Portal 5.0 Fixpack 2

Once the Portal Toolkit installation has completed but before you continue the Everyplace Toolkit, you must apply WebSphere Portal fixpack 2 and WebSphere Portal 5.0.2 Cumulative Fix 1. Leave the Everyplace Toolkit installation running in the background while you complete the installation of these fixes.

To apply Fixpack 2, follow these instructions:

1. Ensure that WebSphere Studio is not running.
2. Insert the CD containing WebSphere Portal V5.0 Fixpack 2.
3. Run the following command from a command prompt to apply the Fixpack: **`C:\<WSXD>\PortalToolkit\updatePortalTestEnv502.bat <d:>`** where <WSXD> is the installation directory for WebSphere Studio and <d:> is the drive where the Fixpack is located.

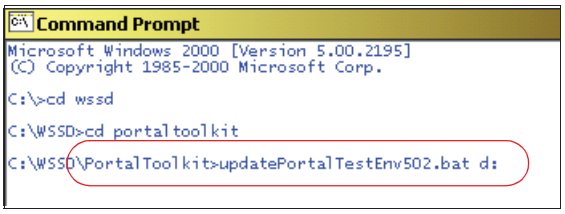


Figure A-21 Updating portal

4. You will receive a message that installation was successful when this process is done.

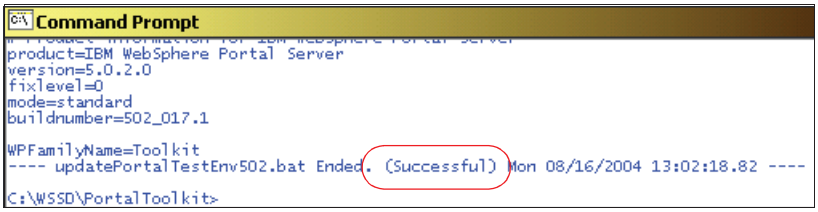


Figure A-22 Update successful

A.4.2.3 Install WebSphere Portal 5.0.2 Cumulative Fix 1

To apply Cumulative Fix 1, follow these instructions:

1. Ensure that WebSphere Studio is not running.
2. Insert the CD containing the WebSphere Portal V5.0.2 Cumulative Fix 1.
3. Run the following command from a command prompt:
   C:\<WSXD>\PortalToolkit\updatePortalTestEnv5021.bat <d:> where <WSXD> is the installation directory for WebSphere Studio and <d:> is the drive where the Cumulative Fix is located.

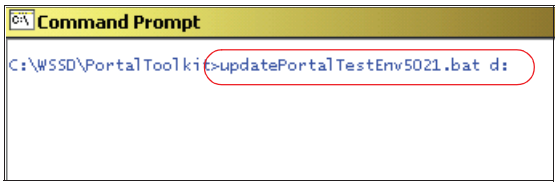


Figure A-23 Updating portal

4. When this installation has completed, you will receive a message saying installation was successful.

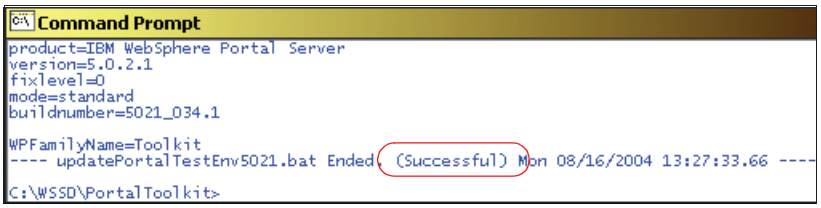


Figure A-24 Update successful

A.4.3 Complete Everyplace Toolkit Installation

Now you will resume the Everyplace Toolkit installation.

1. Accept the default installation directory for the Everyplace Toolkit and click **Next** to start the installation process.

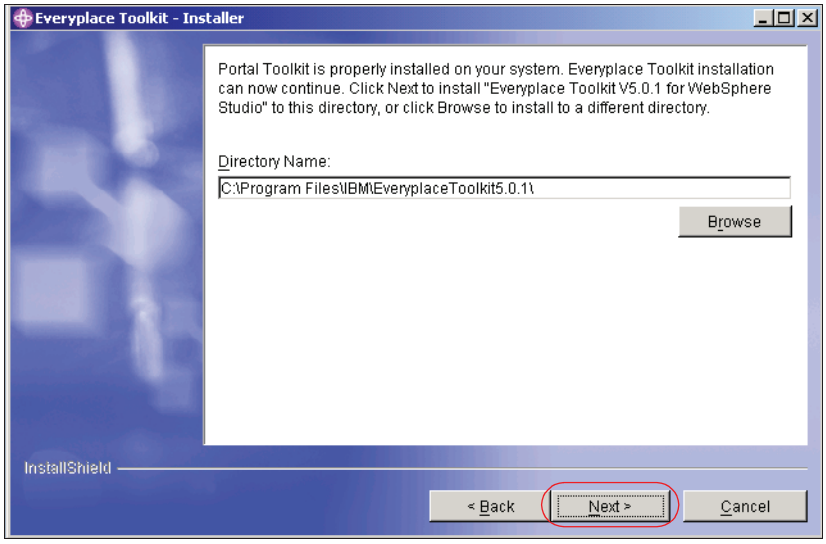


Figure A-25 Resuming Everyplace Toolkit installation

2. Select the **Typical** setup type. This installation type will pre-select and install all of the features including the XHTML+Voice Toolkit (Multimodal Toolkit) which at this time is still a technical preview.

> **Note:** If you chose not to install WebSphere Studio Device Developer, you will need to choose the Custom installation and deselect the Micro Environment Toolkit.

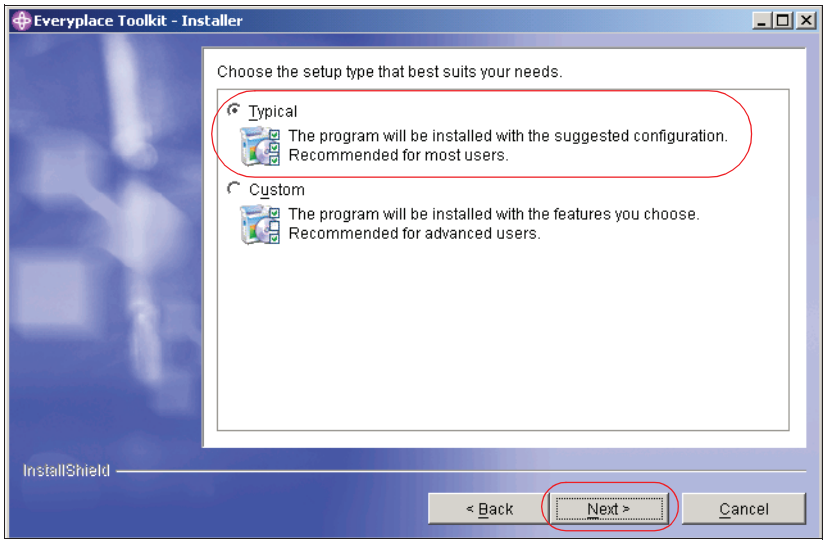


Figure A-26 Typical installation

3. Review the summary information and click **Next** to begin installation.
4. The Multimodal Browser installation will begin as part of this installation process.
   a. Click **Next** to continue past the welcome window.
   b. Accept the license agreement and click **Next** to continue.
   c. Click **Next** after you have read the Readme file.
   d. Select to install all the features of the Multimodal Browser installation. Click **Next** to continue.

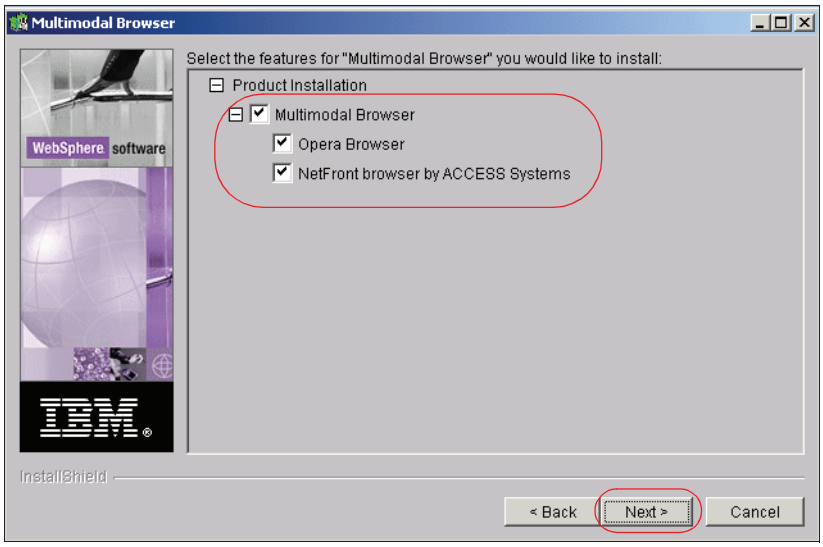


Figure A-27 Installing multimodal browser products

e. Accept the default installation directory for the Multimodal Browser installation. Click **Next** to continue.

f. Click **Finish** to complete installation of the Multimodal Browser.

5. The Multimodal Toolkit installation will begin as part of this installation process.

   a. Click **Next** to continue past the welcome window.

   b. Accept the license agreement and click **Next** to continue.

   c. Click **Next** after you have read the Readme file.

   d. Click **Next** to accept the installation directory for the Multimodal Toolkit.

   e. Click **Finish** to complete installation of the Multimodal Toolkit

6. Click **Next** to install the Micro Environment Tools.

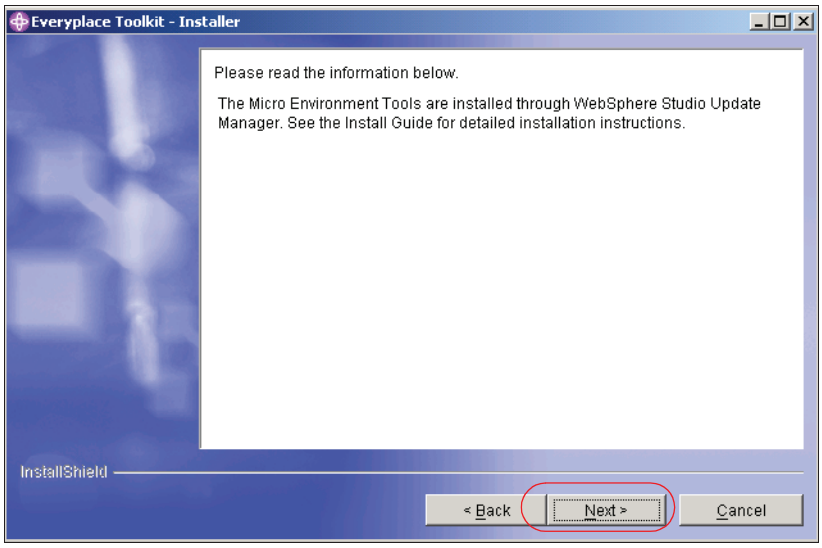


Figure A-28 Installing Micro Environment Tools

7. Click **Finish** to complete installation of the Everyplace Toolkit V5.0.1.

A.4.4 Complete installation of Micro Environment Tools

This series of steps if only necessary if you have installed the WebSphere Micro Environment Tools. This procedure is not done as part of the installation program and is completed using the Update Manager in WebSphere Studio. Follow these steps:

1. Open WebSphere Studio. If you receive the Pending Configuration Changes window, deselect Eclipse C/C++ Standard Make Build and Eclipse C/C++ Managed Make Build. Click **Finish** to apply these changes.

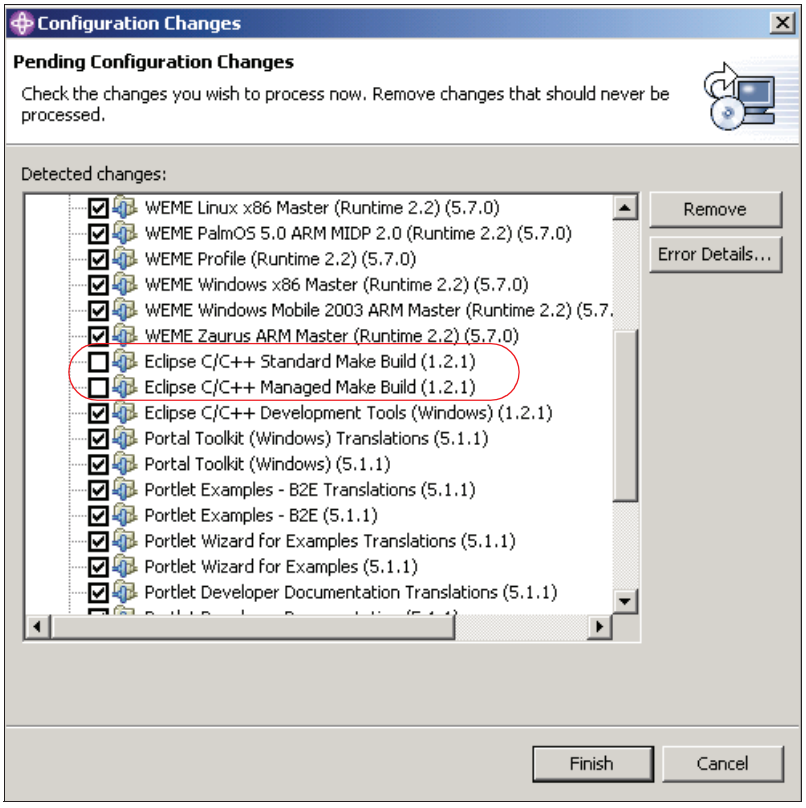


Figure A-29 Configuration changes

2. Click **Yes** to restart the workbench when prompted.
3. Once WebSphere Studio is loaded, from the menu bar click **Window->Open Perspective->Install/Update**.

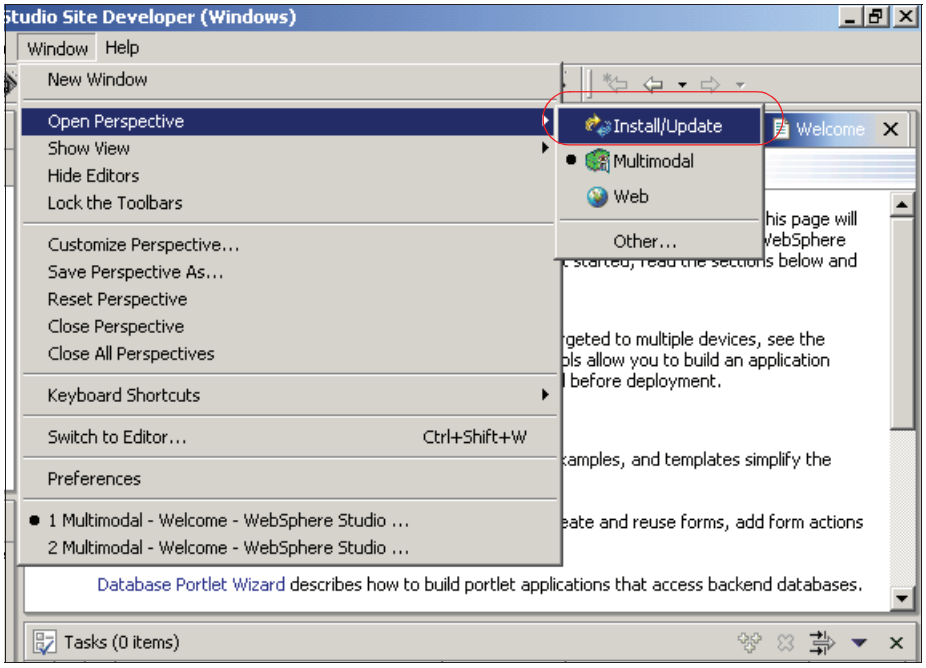


Figure A-30 Install/Update perspective

4. In the Feature Updates view, expand My Computer. Insert the CD containing WebSphere Studio Device Developer and the IBM Micro Environment Toolkit for WebSphere Studio. Expand the CD drive letter. Expand the wsdd/toolkit directory.
5. Install SMF Bundle Development
   a. Expand the Extension Services for WebSphere Everyplace category.
   b. Under this category, select SMF Bundle Development Kit 5.7.0.
   c. In the Preview window, click **Install Now.**

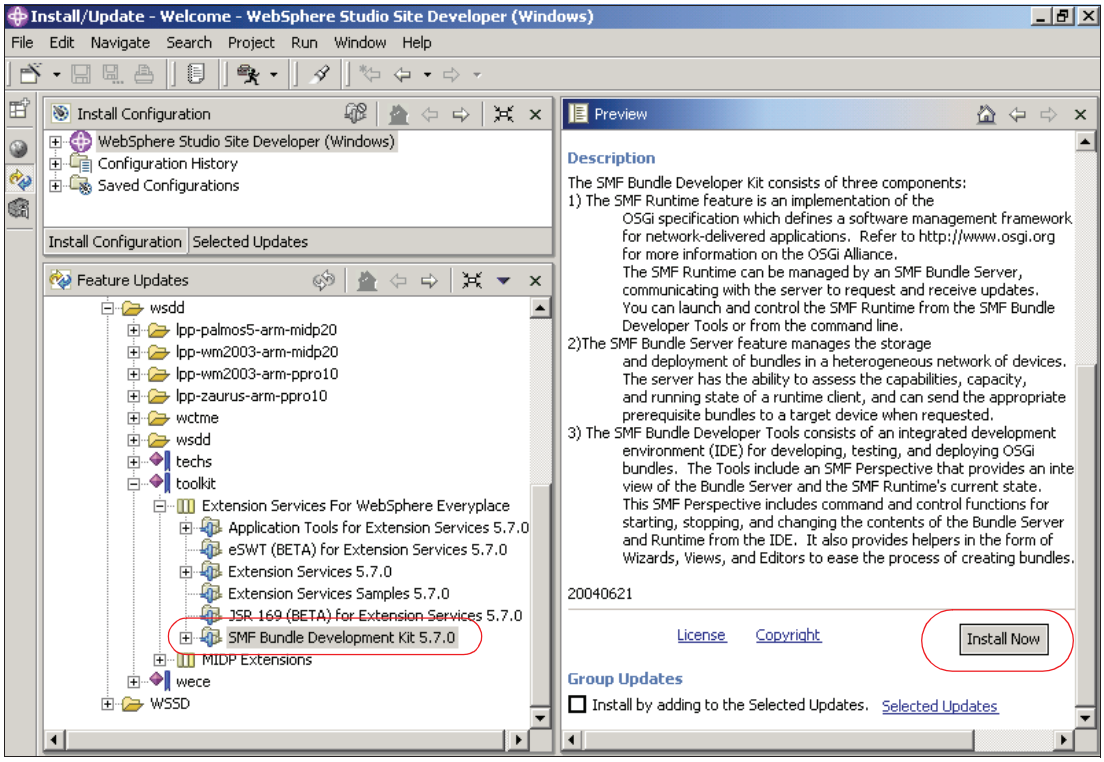


Figure A-31 Install/Update Perspective

d. The Install wizard opens. Click **Next**.

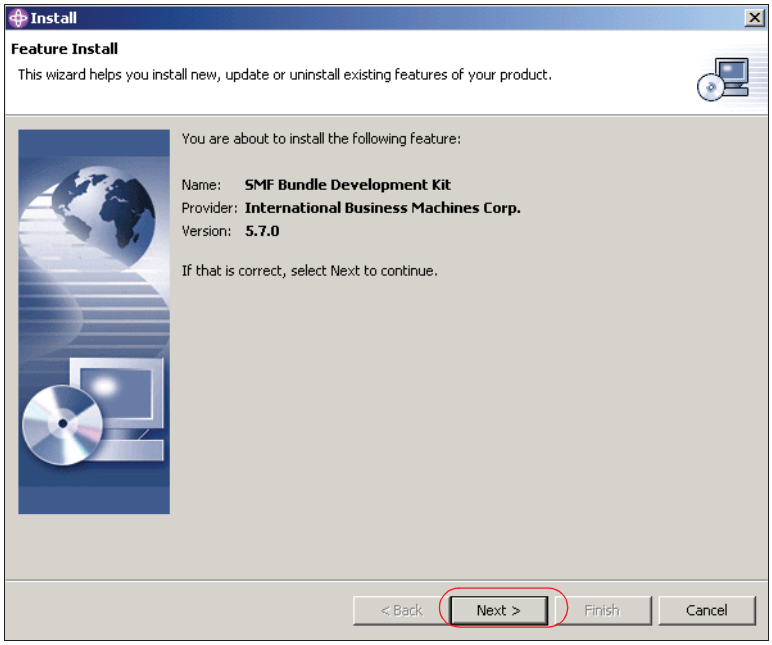


Figure A-32 Beginning installation

e. After accepting the license agreement, click **Next**.

f. Accept the default installation location. Click **Finish** to install.

g. If you are warned that you are about to install an unsigned feature, click **Install** to continue.

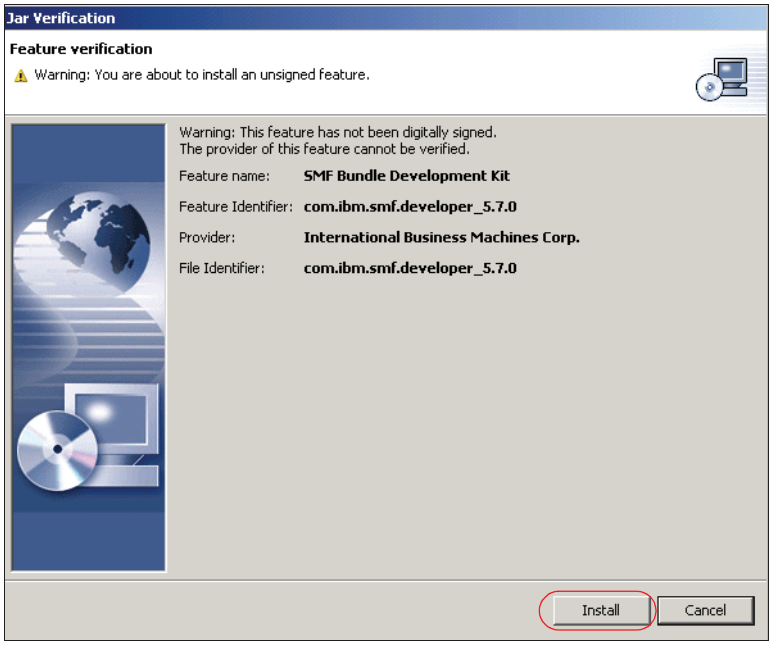


Figure A-33 Feature verification window

h. When prompted to restart the workbench, click **No**. Do not restart the workbench until all Micro Environment installations are complete.

6. Install Application Tools for Extension Services

> **Note:** The following installation procedures are very similar to that of the previous installation step.

a. Under the Extension Services for WebSphere Everyplace category, select Application Tools for Extension Services 5.7.0.

b. In the Preview window, click **Install Now**.

c. The Install wizard opens. Click **Next**.

d. After accepting the license agreement, click **Next**.

e. Accept the default installation location. Click **Finish** to install.

f. If you are warned that you are about to install an unsigned feature, click **Install** to continue.

g. Do not restart the workbench until all Micro Environment installations are complete.

7. Install WebServices for MIDP
    a. Expand the MIDP Extensions category.
    b. Under this category, select Web Services for MIDP 5.7.0.
    c. In the Preview window, click **Install Now**.
    d. The Install wizard opens. Click **Next**.
    e. After accepting the license agreement, click **Next**.
    f. Accept the default installation location. Click **Finish** to install.
    g. If you are warned that you are about to install an unsigned feature, click **Install** to continue.
    h. Do not restart the workbench until all Micro Environment installations are complete.
8. Install Enterprise Tools for MIDP
    a. Under the MIDP Extensions category, select Enterprise Tools for MIDP 5.7.0.
    b. In the Preview window, click **Install Now**.
    c. The Install wizard opens. Click **Next**.
    d. After accepting the license agreement, click b.
    e. Accept the default installation location. Click **Finish** to install.
    f. If you are warned that you are about to install an unsigned feature, click **Install** to continue.
    g. Do not restart the workbench until all Micro Environment installations are complete.
9. Install Extension Services
    a. Under the Extension Services for WebSphere Everyplace category, select Extension Services 5.7.0.
    b. In the Preview window, click **Install Now**.
    c. The Install wizard opens. Click **Next**.
    d. After accepting the license agreement, click **Next**.
    e. Accept the default installation location. Click **Finish** to install.
    f. If you are warned that you are about to install an unsigned feature, click **Install** to continue.
    g. Do not restart the workbench until all Micro Environment installations are complete.
10. Install Extension Services Samples

a. Under the Extension Services for WebSphere Everyplace category, select Extension Services Samples 5.7.0.

b. In the Preview window, click **Install Now**.

c. The Install wizard opens. Click **Next**.

d. After accepting the license agreement, click **Next**.

e. Accept the default installation location. Click **Finish** to install.

f. If you are warned that you are about to install an unsigned feature, click **Install** to continue.

g. Click **Yes** to restart the workbench when prompted.

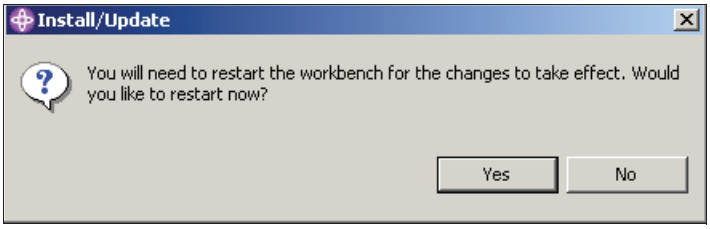


Figure A-34 Restart the workbench

B

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/SG246463

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG246463.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

File name	*Description*
sg246463.zip	Zipped Code Samples

© Copyright IBM Corp. 2005. All rights reserved.

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space: 13 MB minimum
Operating System: Windows
Processor: 1 GB or higher
Memory: 512 MB

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 447. Note that some of the documents referenced here may be available in softcopy only.

- *IBM WebSphere Everyplace Access V5 Handbook for Developers and Administrators Volume I: Installation and Administration,* SG24-6462
- *IBM WebSphere Everyplace Access V5 Handbook for Developers and Administrators Volume II: Application Development,* SG24-6463
- *IBM WebSphere Everyplace Access V5 Handbook for Developers and Administrators Volume III: E-Mail and Database Synchronization,* SG24-6676
- *IBM WebSphere Everyplace Access V5 Handbook for Developers and Administrators Volume IV: Advanced Topics,* SG24-6677
- *IBM WebSphere Everyplace Access Version 4.3 Handbook for Developers,* SG24-7015
- *IBM WebSphere Everyplace Connection Manager V5 Handbook,* SG24-7049
- *RDB Synchronization, Transcoding and LDAP Directory Services in IBM WebSphere Everyplace Access Version 4.1.1,* SG24-6856
- *IBM WebSphere Portal V5 A Guide for Portlet Application Development*, SG24-6076
- *IBM Replication Solutions for Pervasive Computing with DB2 Everyplace and DB2 Satellite Edition*, SG24-6217
- *IBM WebSphere Everyplace Server Service Provider and Enable Offerings: Enterprise Wireless Applications*, SG24-6519
- *Getting the Most From Your Domino Directory*, SG24-5986
- *IBM Workplace Client Technology Micro Edition Version 5.7.1: Application Development and Case Study*, SG24-6496

© Copyright IBM Corp. 2005. All rights reserved.

Other publications

These publications are also relevant as further information sources:

- *IBM DB2 Everyplace Sync Server Administration Guide 7.2*, SC27-0845
- *DB2e Application Development Guide Version 8 Release 1*, SC18-7185
- *IBM DB2 Everyplace Sync Server Administration Guide Version 8 Release 1,* SC18-7186

Online resources

These Web sites and URLs are also relevant as further information sources:

- WebSphere Everyplace Access:

  http://www.ibm.com/software/pervasive/ws_everyplace_access
- WebSphere Portal:

  http://www.ibm.com/software/websphere/portal
- WebSphere Application Server:

  http://www.ibm.com/software/webservers/appserv
- Lightweight Directory Access Protocol standard:

  http://www.ietf.org/rfc/rfc1777.txt
- WebSphere Studio:

  http://www.ibm.com/software/websphere/studio/
- WebSphere Transcoding Publisher:

  http://www.ibm.com/software/webservers/transcoding
- Lotus product information:

  http://www.lotus.com/home.nsf/welcome/products
- WebSphere Everyplace Suite:

  http://www.ibm.com/pvc/products/wes_provider/index.shtml
- Java Community Process, JSR 168:

  http://jcp.org/jsr/detail/168.jsp

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

A

Abstract Window Toolkit 364
Access 346
Accessing resource bundles in JSPs 95
Accessing resource bundles in portlets 94
Accessing Web Services 207
Action Event 4
action event handling 33
action events 33
ActionEvent 35
ActionEvent portlet 34–36
ActionListener 45
actionPerformed 35, 45
actionPerformed method 187
active credential 185, 190
active credential object 187
add field validations 333
Adding a script 338
Adjusting Portal resource bundles 103
alert 338
Annotation Transcoding 266
application development 1, 415
authorization header 187

B

back-end resource 187
back-end secure access 182
basic portlet 33
browser 155

C

cellular telephone simulator 4
central database server 364
class 194
Code samples 33
command prompt 5
components 4
Configure.jsp 226
Configuring the portlet for Transcoding 266
Controller 4
Creating a Web Service 215
credential objects 190
credential slots 183
Credential Vault 181–182, 190
  Active credential objects 185
  Add credential vault handling 193
  administrative slot 184
  Administrator-managed segments 183
  Components of the Credential Vault 183
  Credential slots 184
  Credentials objects 185
  HttpBasicAuth 185
  HttpFormBasedAuth 185
  Import a protected servlet application 187
  J2EE Settings Page 192
  JaasSubjectPassive 185
  JavaMail 185
  LtpaToken 185
  Overview 182
  Passive credential objects 185
  portlet private slot 184
  Portlet Settings 192
  private keys 182
  shared slot 184
  SimplePassive 185
  SiteMinderToken 185
  SSL client certificates 182
  system slot 184
  user credentials 182
  User-managed segments 183
  UserPasswordPassive 185
  Using active credentials 190
  Using passive credentials 204
  Vault segments 183
  WebSealToken 185
Credential Vault organization 183
Credential Vault Portlet Service 183
credentials 182, 194
current version 195

D

data model 4
database 135, 160
database access 135
database access portlet 159

© Copyright IBM Corp. 2005. All rights reserved.

Database portlet
Add pda markup 141
Administrator 146
advanced options 138
basic portlet settings 143
Choose SQL Method 144
columns 146
connection 145
Control Center 136
database 136
DB2 157
DB2 database access portlet 137
default values 139
Deploy 159
EAR Project 139
Event handling 140
generation model 143
list 158
mark-up languages 141
Master information 156
max length 150
modify column names 149
New Project Wizard 137
page title 148
Parse the SQL 147
password 145
perspective 141
populate 136
portlet descriptor 157
portlet development 141
remove columns 149
Result Page 148
sample portlet 157
Server Perspective 152
SQL statement 144
SQL statement types 142
steps 157
Tables tab 146
test database 145
test portal server 152
Toolkit 136
userid 145
WAR file 158
wizard 137, 147
database tables 365
db testdb 160
db wizard 141
DB2 Control Center 161
db2 create 160
DB2 Everyplace 364
DB2e applications
capabilities 355
Configurations 357
Connected Device Configuration 357
Connected Limited Device Configuration 357
data synchronization 355
Development tools 356
Eclipse Workbench 356
Foundation Profile 357
frameworks 356
Introduction 356
J2ME 356
J2ME architecture 357
Java 2 Micro Edition 356
Mobile Information Device Profile 357–358
Overview 356
Personal Profile 358
Profiles 357
standalone mode 359
WebSphere Micro Environment 358
WebSphere Studio Application Developer 358
WebSphere Studio Device Developer 355–356
default resource bundle 92
default settings 37
DefaultPortletMessage 64–65
Define the Portlet Project 191
deployment descriptor 17, 39–40
design pattern 4
devices 13
directory 420
doView 35
Dynamic Web Project wizard 209

E

EAR project 188, 211
Edit mode 187
elements 4
Environment Selection section 219
Event Handling 37, 230
event-processing phase 63
Everyplace Access 305
Everyplace Toolkit 1, 135, 415, 420
Everyplace Toolkit installation program 427
Everyplace Toolkit V5.0.1 326
Example code 328
ExportResources 248
Extensible Markup Language 246, 266

F
fixes 419
Fragmentation 271
Fragmentation Transcoding 266
fragmentation transcoding portlet 271

G
Generate a proxy 217
generated code 194
getCredential 195
getFormText() method 46
getSlotId 195

H
hard drive 420
HelloWorld portlet 4
HTML 3, 266
HTML form 35
HTML pages 87, 266
HTTP 266
HttpBasicAuth 187

I
Image Transcoding 266, 270
ImportCocoon 248
ImportResources 248
individual applications 182
Input.jsp 226
Install Now 421
Install Portal Toolkit 428
installation errors 422
installation files 428
integrated test environment 419
Interim Fixes 421
interim fixes 419

J
J2EE level option 8
J2EE Setting Page 229
J2EE settings 37
J2ME 364
Java Source directory 90
JavaBean 4, 34, 209
JavaBeans 4
Javascript function 338
JSP 195
JSP files 226

K
key-value pairs 88

L
languages 13
license agreement 422, 429
locale 88
log 181

M
main menu 187
markup languages 34
message 154
Message events 34, 63
MessageListener 63–64
MessageReceiver 64
Method.jsp 226
Micro Environment Tools 415
Micro Environment tools 425
Miscellaneous window 38, 230
mobile devices 237, 305, 364
mobile users 237
Model 4
Model-View-Controller 4, 34
multiple languages 87
MVC 4
MVC architecture 46
MVC methodology 4
MVC portlet application 4

N
National Language Support 87
- Accessing resource bundles in JSPs 95
- Accessing resource bundles in portlets 94
- Adjusting Portal resource bundles 103
- Creating Resource Bundles in WebSphere Studio 90
- NLS administration 99
- NLS best practices 104
- Portal NLS administration 102
- Portlet NLS administration 99
- Resource bundles 88
- Sample scenario 105
- Setting NLS titles 102
- Translating Resource Bundles 92
- Translating whole resources 97
- Working with characters 104

Navigator panel 333
NLS 87
NLS administration 87, 99
NLS best practices 87, 104
NLS enablement 87
Nokia Mobile 3

O

offline application development 305
Offline browsing
  connecting 324
  deploy 314
  disconnected status 323
  disconnecting 324
  external annotator 311–312
  fetch content 308
  J2EE default settings 308
  Network profiles 322
  offline access 321
  offline capabilities 315
  Offline Content 322
  offline server 322
  Palm devices 307
  password 320
  Pocket PCs 307
  portal page 315
  Portal Test Environment 311
  portlet descriptor 313
  portlet execution 310
  remote server 311
  resource permission 317
  synchronization process 323
  Test Environment 310
  toolkit 308
  transcoding plugins 318
  user ID 320
  Web browser 319
  Web clipping 308
offline configuration 344
offline form access 325
Offline forms
  Adding field validations 333
  confirmation dialog 329
  Deploying 325
  deployment error messages 328
  design view 333
  field validations 325
  form based portlet 325
  Javascript 335, 338
  offline configuration 325
  offline portlet 339
  PocketPC 331
  portal test environment 336
  portlet package 338
  sample portlet 339
  sample scenario 326
  Tasks panel 329
  toolkit 326
  validation pattern 334
  wizards 326
offline option 325
Offline Portal Browsing 340
offline project 336

P

Palm OS 3, 33
passive credentials 185
PDA 3
PDA device 339
Personal Profile 364
platform installation 415
Pocket PC 135, 325, 342
Pocket PC targets 364
Portal NLS administration 102
Portal test environment 135
Portal Toolkit 415, 428
Portlet API 34
portlet application 4, 44, 135
portlet applications 237, 419
portlet deployment descriptor 12
portlet development 182, 227
Portlet file 226
portlet framework 36
Portlet NLS Administration 99
Portlet NLS administration 99
portlet offline capabilities 344
portlet perspective 90
Portlet Settings 230
portlet View mode 187
Portlet wizard 34
portlet.xml 12
portlet.xml file 11
PortletAction 35
PortletApplicationSettingsAttributeEvent 34
portlet-level transcoding 239
portlets

sequence flow 35
PortletSettingsAttributeEvent 34
Project name 210
Project Navigator 11, 39
properties 88
protected servlet 190
Proxy Page window 222

R

Redbooks Web site 447
Contact us xiii
Request Viewer 249, 252–253, 263
resource bundle 88
Resource bundles 87–88
Resource Permissions 346
Result.jsp 226
retrieving credentials 183
Run on Server 336

S

sample installation
Everyplace Access 417
single machine 415
summary information 418
WebSphere Studio 416
WebSphere Studio Site Developer 416
sample scenario 181, 187, 212, 226
sample Web Service 214
server configuration 327
Server Types 218
Service Deployment Configuration 218
servlet 187
setCredential 195
Setting NLS titles 102
setup 160
Single Sign-On 182, 230
Single Sign-On page 193
slots 184, 195
Software requirements 416
source code 34
SQL DELETE 143
SQL INSERT 143
SQL SELECT 143
SQL UPDATE 143
Studio 90
Studio Application Developer 326
Studio Site Developer 137, 187, 326, 415
Studio Workbench 326
Submit the action 203

T

table 160
Technical Preview 415
test database 135
Test Environment 190, 330, 364, 415
Text Transcoding 266
Tivoli Access Manager lock box 184
Toolkit installation 427
Toolkit V5.0.1 427
Tracing files 258
transcoding online applications 237
transcoding plugin 266
Transcoding Technologies 239–240
a simple portlet 258
aggregator 242
annotation transcoding plug-in 247
annotators 240–241, 246
architecture 241
device 245
Disable resources 248
Document Object Model (DOM) 247
Enable resources 248
enable transcoding 258
Export resources 248
Extensible Markup Language 240
external annotation 247
fragmentation transcoding plug-in 247
HTML pages 240
HTTP header 246
image transcoding plug-in 247
Import resources 248
internal annotation 246
logging 255
markup language conversions 241
message files 257
monitor the process 263
plug-in 247
Portal page level transcoding 241
portlet filter 242
Portlet-level transcoding 241
preference profiles 240, 243
profile 245
Request Viewer 249
stylesheets 241
text transcoding plug-in 247
tracing 255

tracing records 258
transcoding plug-in 240, 247
troubleshooting data 258
user 245
user agent 242
XML config tools 248
XML stylesheets 240, 246
Transcoding Technology 265
translated files 93
Translating Resource Bundles 92
Translating whole resources 87, 97
translations 92

U

user interface 4
Using resource bundles 87

V

vault segment 183
vault segments 183
vaultService data 195
View 4
View mode 187

W

WAP browser 262
WAR file 188, 339
Web applications 427
Web perspective 187
Web Project 218
Web Services Client
Create a Web Service 215
Create a Web Services client portlet 226
web.xml 11, 17
WebSphere Everyplace Access 344
WebSphere Portal 182
WebSphere Portal fixpack 2 430
WebSphere Studio Site Developer 415, 417
WindowEvents 34
WML 266
WML markup 3
workbench 363
working form 328
Working with characters 87, 104
workspace 5

X

XML 266
XSL stylesheets 246
XSL Transcoding 266

IBM
Redbooks
IBM WebSphere Everyplace Access V5
Handbook for Developers and Administrators
Volume II: Application Development
(1.0" spine)
0.875"<->1.498"
460 <-> 788 pages



IBM WebSphere Everyplace Access V5 Handbook for Developers and Administrators

Volume II: Application Development



Develop mobile applications with offline capabilities

Develop online portlet applications for PDAs

Adapt content to multiple markup languages

This IBM Redbook is part of a series of four volumes related to WebSphere® Everyplace® Access Version 5.0. These Redbooks will help you plan, install, administer and develop mobile applications to run in a WebSphere® Everyplace® Access Version 5.0 environment. The information provided in these Redbooks target business-to-employee (B2E) enterprise applications, but most of the scenarios presented will apply to business-to-consumer (B2C) applications as well. In these Redbooks, you will find step-by-step examples and scenarios showing ways to integrate your enterprise applications into a WebSphere Everyplace Access environment using the WebSphere Studio Site Developer and the Everyplace Toolkit, and extend your online and offline application capabilities to use other advanced functions.

In this redbook you will also find step-by-step sample scenarios illustrating procedures to develop online applications using WebSphere Studio Site Developer and WebSphere Application Site Developer along with the WebSphere Everyplace Toolkit. Sample scenarios also include offline applications using Offline Portal Content and Forms, Transcoding, and DB2e applications using the WebSphere Studio Device Developer.

A basic knowledge of Java technologies such as servlets, JavaBeans, EJBs, JavaServer Pages (JSPs), as well as XML applications and the terminology used in Web publishing, is assumed.

The following *IBM WebSphere Everyplace Access V5 Handbook for Developers and Administrators* Redbooks are part of this offering:

Volume I: Installation and Administration, SG24-6462
Volume II: Application Development, SG24-6463
Volume III: E-Mail and Database Synchronization, SG24-6676
Volume IV: Advanced Topics, SG24-6677

SG24-6463-00 ISBN 0738492485

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks