iSeries

Travel Portal scenario

iSeries

Travel Portal scenario

Contents

Travel Portal scenario overview

This report describes the iSeries(TM) Platform Evaluation Test (iPET) team's experience with WebSphere(R) Portal Server on the IBM(R) eServer(TM) i5 platform. The team developed a new test scenario that utilizes WebSphere Portal Server to provide users a portal interface. This interface allows users to view and use business and personal information from another, pre-existing test scenario. The pre-existing scenario was named Travel Agency, so the new portal scenario was given the name Travel Portal. This section of the report describes the Travel Portal development team, and provides background information about the Travel Agency and Travel Portal scenarios.

Development team

The iPET team designs, implements, deploys, and evaluates customer-like solutions in a fashion similar to that used by information technology architects throughout industry. Although the team is restricted to a test laboratory environment, great effort is made to reflect a true customer environment. The team cannot cover every possible customer situation, so it focuses on those scenarios and methodologies that iSeries(TM) customers are most likely to implement in their production environments.

When portal technology started to gain popularity, a small subset of the iPET team began to develop a new test scenario specifically targeting WebSphere(R) Portal Server on the eServer(TM) i5 platform. This report describes this team's work, including the discoveries that were made during the development of the scenario.

Scenario Background

Before the development of the Travel Portal scenario, the iPET team developed the Travel, Flights, and Cruise (TFC) scenarios to test the self service (user-to-online-buying) and the extended enterprise (business-to-business) business patterns. TFC is composed of several fictitious companies, including a cruise company, a travel agency, a flights company, and a bank. Figure 1 illustrates some of the companies within the TFC environment.

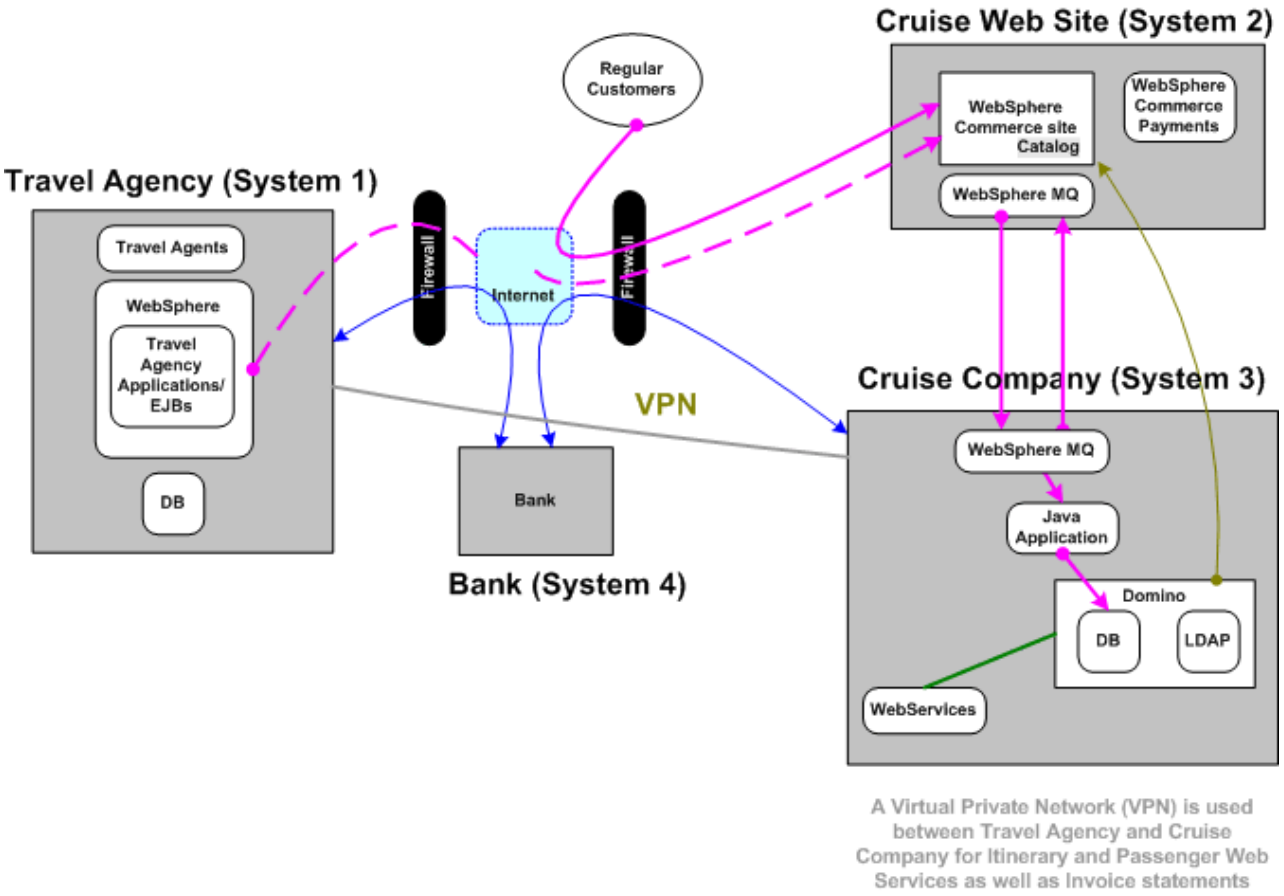


Figure 1. TFC scenario environment

Additional information on the existing TFC scenario can be found in the IBM(R) redpaper

Business-to-Business Integration Guide: Using WebSphere Application Server and Domino for iSeries , REDP0139.

When the TFC scenarios were initially developed, the Travel Agency scenario provided a Web-based application where travel agents could purchase cruises from the cruise company, create and update customer information, book cruises for customers, and so on. However, there was no interface provided for customers to directly access this application, or to view the travel agency's cruise information. To address this need, the iPET team turned to WebSphere Portal Server to provide such an interface for both customers and travel agents. The new portal application became known as the Travel Portal scenario.

By integrating WebSphere Portal Server into its environment, the Travel Agency now has an easy means of obtaining the following:

- Increased programmer productivity by using the portal architecture with its site themes, skins, and Cascading Style Sheet (CSS) support. This allows the developer to focus on the business problem that they must solve, and create the portlet that addresses the problem without having to worry about graphic design.
- Increased employee productivity by allowing the employee easy access to the critical business applications and information from within one portal place.
- Improved customer satisfaction through an easy-to-use public Web site where customers can easily view the availability of future cruises, view the cruises they have already booked, communicate online with a travel agent representative, and so on.

Using WebSphere Portal Server, the Travel Portal scenario provides travel agents and customers access to the following portlets:

- Travel Agents
  - Welcome — Displays the travel agency welcome page
  - Agent Web — Allows access to the Web-based travel agency application
  - Best Deals — Displays the best deals available on different cruises
  - Create Customer Account — Creates accounts for new customers
  - Customer History — Displays the cruise history for a specific customer
  - eInvestments — Allows access to an investment's company application (another test scenario)
  - eMail — Provides e-mail capability
  - Sametime — Provides instant messaging capability
  - SQL Query — Provides the ability to run SQL queries
  - Unbooked Cruises — Displays a list of unbooked cruises
  - World Clock — Displays a clock
- Customers
  - Welcome — Displays the travel agency welcome page
  - Best Deals — Displays the best deals available on different cruises
  - Future Trips — Displays a list of cruises that the customer has booked
  - Sametime — Provides instant messaging capability
  - Trip Reminder — Displays a list of cruises that the customer has booked within a certain time period based on the departure date
  - World Clock — Displays a clock

WebSphere Portal Server

The Travel Portal scenario was developed by the iPET team to directly test the WebSphere(R) Portal Server product on the eServer(TM) i5 platform. WebSphere Portal Server provided the base functionality for the scenario, but the team also incorporated Domino(R) Directory for user authentication, Lotus(R) Sametime(R) for instant messaging capabilities, and Domino mail to enable travel agents to send e-mail to one another. This section of the report provides an overview of WebSphere Portal Server, and descriptions of the server configurations, including WebSphere Portal Server, Domino Directory, and Lotus Sametime. This section also provides brief descriptions of the discoveries the team made during the setup and configuration of these servers.

WebSphere Portal Server overview

WebSphere(R) Portal Server consists of middleware, applications (portlets), and development tools for building and managing secure business-to-business (B2B), business-to-customer (B2C), and business-to-employee (B2E) portals. A portal is a Web site that provides end users with a single point of access to Web-based resources by aggregating those resources in one place. End users of a portal are required to log in only to the portal itself, instead of to each portlet the end users use. WebSphere Portal Server can deliver Web content to wireless application protocol-enabled (WAP-enabled) devices and i-Mode phones, as well as to various Web browsers.

As an administrator, you can customize WebSphere Portal Server to meet the needs of your organization, users, and user groups. You can adapt the look and feel of the portal to fit the standards of your organization, and customize page content for users and groups in accordance with business rules and user profiles. End users, such as business partners, customers, or employees, can further customize their own views of the portal. End users can add portlets to pages and arrange them as they want, and control portlet color schemes. By aggregating portlets in one place and giving end users the power to customize their own desktops, WebSphere Portal Server gives end users a means for doing business more efficiently and with higher satisfaction.

WebSphere Portal Server setup

WebSphere(R) Portal - Express 5.0.2.2 on iSeries(TM) is used to deploy the portlets and to make them available to the travel agents and customers. It also provides the travel agents and customers with a variety of collaboration features, such as instant messaging, e-mail, and so on. Figure 2 illustrates the addition of Travel Portal to the existing TFC environment. The portal server, Lotus(R) Domino(R), and Lotus Sametime(R) are installed on System 5.

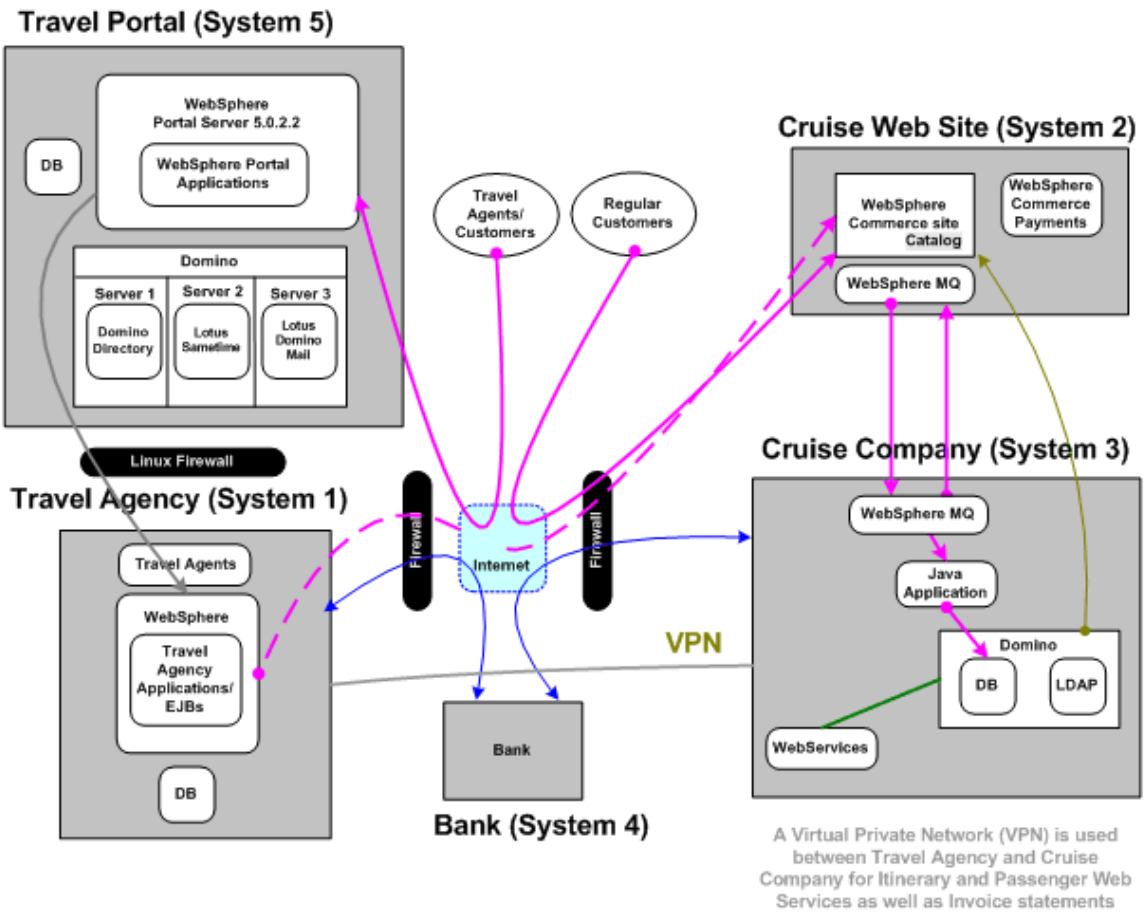


Figure 2. Addition of travel portal to TFC scenario environment

The changes made to the existing TFC environment include the following:

- Install Lotus Domino and Lotus Sametime on the travel portal system
- Install the prerequisites for WebSphere Portal 5.0.2.2 on the travel portal system
- Install WebSphere Portal 5.0.2.2 on the travel portal system
- Create and configure the portal server instance
- Install the portlets

The next three sections of this report provide an overview of the environment used during the portlet integration. The sections include the following:

- An overview of the server environment
- An overview of the use of Domino Directory for the LDAP server
- A list of discoveries the were made during the set up of the environment

Server environment

WebSphere(R) Application Server supports a wide variety of ways to deploy WebSphere Portal Server in a computing environment. Commonly used topologies fall into the following categories:

- **Single-machine topology**. The components are installed on the same machine.
- **Multi-machine topologies**. The components (the Web server, application server, databases, and so forth) are physically separated onto different machines.
- **Vertical scaling topologies**. Additional WebSphere Portal processes are created on a single physical machine through vertical cloning.

- **HTTP server separation topologies**. The Web (HTTP) server is located on a different physical machine from WebSphere Application Server and WebSphere Portal.
- **Demilitarized zone (DMZ) topologies**. Firewalls can be used to create demilitarized zones, which isolate machines from both the public internet and other machines in the configuration. This improves portal security, especially for sensitive back-end resources such as databases.

The Travel Portal scenario uses the single-machine topology. The Web server, portal server, database, Domino(R) Directory server, and Lotus(R) Sametime(R) server all exist on one system.

Two systems are used to implement the Travel Portal scenario:

Existing Travel Agency system

The Travel Agency scenario exists on an eServer(TM) i5 running i5/OS(TM) V5R3. The Travel Agency application is deployed in a WebSphere Application Server Version 5.1 server instance. The application accesses the Travel Agency's database located on the same system through enterprise beans.

During the design of the portlets, a decision was made to utilize the existing enterprise beans within the Travel Agency application. This provides a common interface to access the back-end databases and also limits the amount of coding required for the new portlets. Additional information about the use of enterprise beans within the Travel Agency can be found in the IBM(R) redpaper Enterprise JavaBeans with VisualAge for Java: A Case Study for the IBM eServer iSeries Servers , REDP0136.

New Travel Portal system

Another eServer i5 running i5/OS V5R3 is used for the portal server. The following outlines the steps used to configure the server:

1. Install and set up the Domino Directory and Lotus Sametime servers. More information about the installation and setup of the servers can be found in "Collaboration server setup" on page 9.
2. Install the prerequisites for WebSphere Portal 5.0.2.2. This includes the WebSphere Application Server V5.0, the required Program Temporary Fixes (PTFs), and the WebSphere Application Server V5.0 Enterprise Enablement product. See Supported hardware and software for the hardware requirements and software product levels that are supported for WebSphere Portal.
3. Install the WebSphere Portal Version 5.0.2.2 product (using the **Install only** option).
4. Configure the new portal instance using the WebSphere Portal wizard (available within the IBM Web Administration for iSeries tool). The wizard simplifies the configuration of WebSphere Portal and its core components. It is designed to handle a range of common WebSphere Portal configurations. If additional configurations are required, the configuration can be modified by running WebSphere Portal configuration tasks at a later time.

   The wizard is used to perform the following tasks:

   a. Create the HTTP server and WebSphere Application Server server instance.
   b. Configure the WebSphere Application Server server instance for Portal.
   c. Configure the database for Portal. (A local database is specified for this scenario.)
   d. Deploy the portlets installed with the WebSphere Portal product. (Administration Portlets, Themes and Skins, Business, Document Manager, and Lotus Collaboration are selected for this scenario.)
   e. Configure the Lotus Collaborative Components for Portal (Lotus Sametime and Lotus Domino Directory).
   f. Secure the application server and WebSphere Portal with LDAP. (Lotus Domino Directory located on the system is selected for this scenario.)

5. Customize the portal instance for use by the travel agency. This includes creating the travel agent and customer user IDs/groups, creating the labels and pages used by the travel agency, and granting the user IDs the appropriate authority to the labels, pages, and portlets.
6. Deploy, install, and activate the Travel Portal portlets.
7. Activate the new collaboration portlets (Sametime and eMail).

Additional information regarding the installation of WebSphere Portal Version 5.0.2.2 can be found in the WebSphere Portal Express and Express Plus V5 for the IBM eServer iSeries Server  redbook.

Domino Directory

For the Travel Portal scenario, Domino(R) Directory is used for the LDAP server and security. The LDAP server serves as the central repository for all of the Domino Directory data. The scenario utilizes the replication features of Domino to push LDAP data from the DomHub server to the other Domino servers. The other two servers are used solely for mail and instant messaging. This implementation provides the flexibility to stop and restart each Domino server without impacting the other servers, as well as improved scalability.

To manage and configure the Domino Directory (LDAP) and the server documents for the Domino servers, the Domino Administrator 6 client is used. All users are added and registered through the Administrator client on the LDAP hub server DomHub, and then replicated to the other Domino servers. It is important to register users to give them access to mail on Domino. Adding a user creates a Person record for that user in the Domino Directory, but does not create a mail file for the user. Registering a user through the Domino Administrator client creates a Person document in the Domino Directory and also creates a mail file for the person. Users must have mail files in order to access Domino mail. Additional information about registering and adding users to Domino Directory can be found in the Domino Administrator 6 Help (Doc Pack G210-1425-00) documentation. These documents can be found on the Domino product documentation Web site at http://www.lotus.com/ldd/doc/AS400/6.5/i400help.nsf  .

Discoveries

This section lists the discoveries that were made during the setup of the WebSphere(R) Portal Server Version 5.0.2.2:

- If you need to delete and recreate your WebSphere Application Server (WebSphere Application Server) portal instance and HTTP servers, the easiest way is to use the IBM(R) Web Administration for iSeries(TM) Web page that you used to create the instances. The following steps outline what needs to be done:
  1. Type the following URL into your the address field of your browser:
     `http://MySystem:2001/HTTPAdmin`
  2. Enter your user name and password.
  3. Click the **Manage** tab.
  4. Click the **All Servers** tab.
  5. In the **Manage All Servers** frame, click the **All Application Servers** tab.
  6. Select the server that you want to delete.
  7. If the server is `Running`, click the **Stop** button.
  8. After the server is `Stopped`, click the **Delete** button.
  9. If you want to recreate a new HTTP server, then you would want to do the following after your WebSphere Application Server portal server has been deleted:
     a. Click **All HTTP Servers** tab.
     b. Select your HTTP server.
     c. If the server is `Running`, click the **Stop** button.

d. After the server is Stopped, click the **Delete** button.

10. Start a 5250 session to the system that contains the database collection used by your WebSphere Application Server portal server.
11. Use **DLTLIB** to delete the database collection used by this WebSphere Application Server portal server.

- The Travel Portal development team created a portal server instance using the iSeries WebSphere Portal wizard, but forgot to select the option to deploy the Business Portlets. After the portal server instance was created, the team ran the installportlets.sh script to deploy the Business Portlets. However, the script encountered the following errors, indicating that it could not read files in the config/work directory:

```
XMLA0006I: Connecting to URL http://localhost:19009/wps/config
XMLA0002I: Reading input file /QIBM/UserData/WebAS5/Base/<instanceName>/PortalServer5/config/work/SetupPortal.xml
XMLA0003E: An error occured while reading the file.
XMLA0009E: Could not connect to portal.
```

  It was determined that the wizard runs the cleanup-work-dir task as the final step in the portal configuration to clean up files used in the configuration of the portal. Thus, before running the installportlets.sh script, the WPSConfig.sh script needed to be run in order to the recreate the directory.

- If you see several messages in a WebSphere Portal Server server's SystemOut.log file that contain the text `Application process not in a connected state`, it may be an indication that the QSQSRVR prestart jobs on your system are not functioning correctly. If this is the case, it is necessary to restart these prestart jobs. This is accomplished by executing the following steps:

  1. End any jobs that could be using SQL prestart jobs (for example, LDAP Server):

     ```
     ENDTCPSVR SERVER(*DIRSRV)
     ```

  2. End and restart the SQL prestart jobs:

     ```
     ENDPJ SBS(QSYSWRK) PGM(QSQSRVR)
     STRPJ SBS(QSYSWRK) PGM(QSQSRVR)
     ```

  3. Restart any jobs that were ended in step 1:

     ```
     STRTCPSVR SERVER(*DIRSRV)
     ```

  If this does not fix the problem, it may be necessary to perform a reclaim storage on your system.

Collaboration server setup

WebSphere(R) Portal Server Version 5.0.2.2 provides the ability to integrate many of the Lotus(R) Domino(R) collaboration features and companion products within a single portal environment. Collaboration features help people work together and share information more effectively. Collaborative portlets provide access to a variety of applications that use Lotus Notes(R) databases hosted on Domino servers, including mail, calendar, To Do, Notes View, TeamRoom, and discussion.

Within the TFC environment, the following Domino 6.5.1 servers were created:

- DomHub (LDAP)
- DomMail (e-mail)
- DomST (Lotus Sametime(R))

For step-by-step details on the installation and configuration of Lotus Domino servers, see the Lotus Collaboration and WebSphere Portal Integration on the IBM eServer iSeries Server  redbook.

Sametime

Lotus Sametime(R) is an IBM(R) product and platform for real-time collaboration. The three main facets of Lotus Sametime are:

- **Presence awareness** - Instantly see whether a person is online

- **Instant messaging** - Communicate in real time through text, audio, or video-based information
- **Web conferencing** - Participate in virtual meetings, share information (an application or an entire desktop), or engage in team white boarding

Within the Travel Portal environment, a Domino(R) server (DomST) was created for Sametime. Section 3.5 of the Lotus Collaboration and WebSphere Portal Integration on the IBM eServer iSeries Server redbook contains detailed instructions on configuring Sametime within a Domino server. This redbook provides clear and concise, step-by-step details for configuring Sametime within any environment.

Figure 3 shows the welcome screen when accessing the Sametime server through a Web browser.

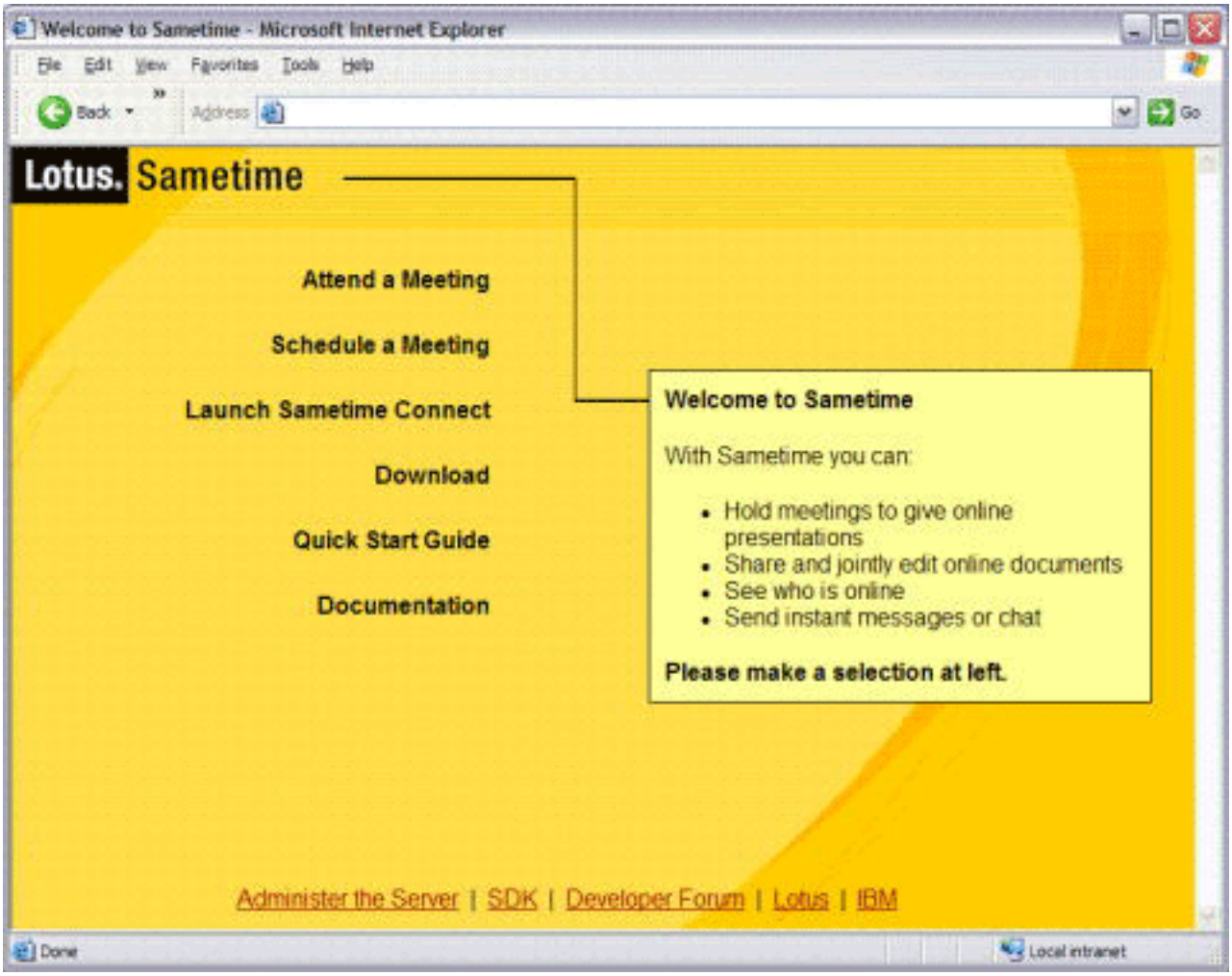


Figure 3. Welcome to Sametime

After users log into the Sametime server, they have access to instant messaging, Web conferencing, and all the other features of Sametime. Figure 4 shows a user signed into a virtual meeting room with team white boarding and instant messaging.

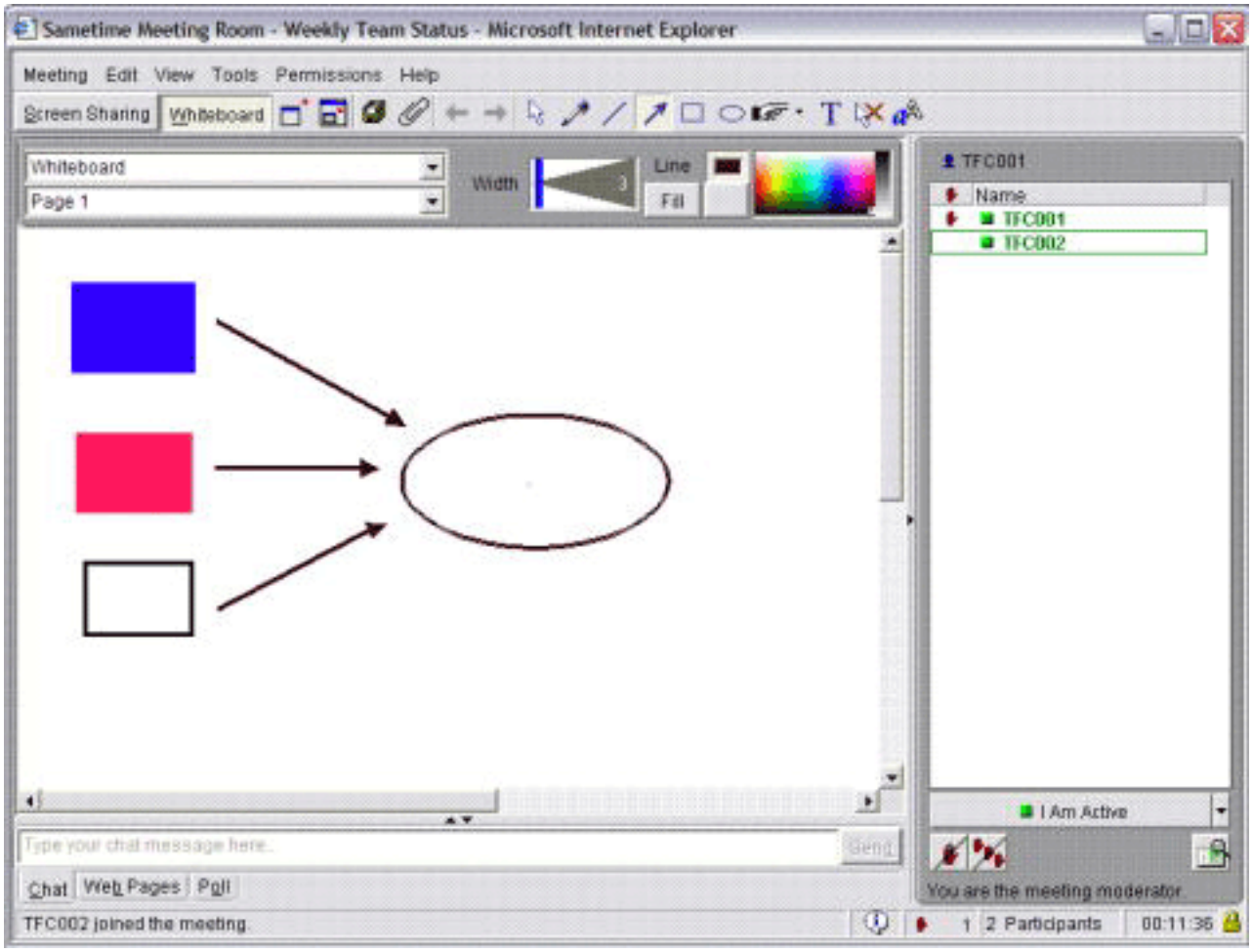


Figure 4. Virtual Meeting through Sametime

Discoveries

Several discoveries were made during the setup and configuration of the Domino(R) collaboration servers:

- It is good practice to run each particular service (LDAP, Sametime(R), and Domino Web Access) on a separate Domino server. When one server is stopped, other Domino servers can still maintain their individual services. For example, with this environment you can start and stop the Domino Web Access and Sametime servers independently of each other.
- Any of the other Domino servers could provide the LDAP directory services, but it is good practice to run LDAP on a separate administration Domino server. This allows the other Domino servers to be stopped while still maintaining the LDAP directory service for authentication, and therefore single sign on, within the Domino domain.
- When running Sametime applications, you need to turn off pop-up blockers in Web browsers or firewalls.
- The LDAP task should only be enabled on one server. This helps to avoid potential problems configuring LDAP within your domain.
- If using Domino 6.5.1 for collaboration with Sametime and WebSphere(R) Portal, be sure to apply Interim Fix Pack 6.5.1F1 to your system.
- If you run into problems logging into the Sametime server, make sure you are using the fully-qualified host name of the Sametime server. You may not be able to log into Sametime if you use the IP address or the short host name.

Travel Portal application

To take advantage of the functionality of WebSphere(R) Portal Server, developers must develop their own portal applications or use existing portal applications, and deploy those applications to their production portal server. The Travel Portal development team developed their own portal application to access business data, and also utilized existing collaborative portlets to give users the ability to quickly and easily communicate with one another.

This section looks at many aspects of portlet development and administration. Before writing their first line of portlet code, the developers need a portlet development environment and a basic understanding of how portlets work. This section addresses both needs by providing an overview of the development environment that was used by the team, as well as descriptions of the application model and flow of the portlets that were developed. With this knowledge, a developer can begin to develop portlets.

This section takes a close look at a subset of the Travel Portal portlets to see exactly how those portlets are implemented. After portlets are developed, they need to be installed and administered, so this section covers the basics of portlet installation and administration. This is followed by a list of discoveries that were made by the team during the development, deployment and administration of the Travel Portal application.

Development environment

IBM(R) WebSphere(R) Studio Application Developer provides an Eclipse-based development environment that allows developers to create, compile, and package Java(TM) Platform 2, Enterprise Edition (J2EE) applications. The Travel Portal team used WebSphere Studio Application Developer version 5.0.2.x to develop the portlets, JSP files, Java beans and enterprise beans that together form the Travel Portal scenario.

For portlet development, WebSphere Studio Application Developer requires the installation of the Portal Toolkit. This toolkit includes plug-ins that provide a portlet development environment, wizards to facilitate the portlet development process, and several portlet examples. Specific versions of the Portal Toolkit are available for each version of WebSphere Studio Application Developer. Thus, it is critical that the proper version of the Portal Toolkit is installed, based on the version of WebSphere Studio Application Developer that is being used. Additional information about this toolkit and a link to download the toolkit itself can be found at the WebSphere Software platform Web site  .

Application model

The Travel Portal application uses the model depicted in Figure 5. In this model, a Web browser makes a request to the initial portal login screen, and upon authentication, accesses the agent or customer welcome page. The Web browser makes a specific portlet request at this time and interacts with JSP files indirectly through a portlet. Portlets and JSP files reside within the portlet container. A portlet interacts with the Travel Agency application to access the travel application business logic through enterprise beans and Java(TM) beans. After receiving a response from the Travel Agency application, the portlet performs necessary validations and computations, and the Java beans are then stored in the request object. The JSP file extracts the information it requires from the Java beans and merges the information with the HTML page. The response is sent back to the browser, which interprets and renders the HTML.

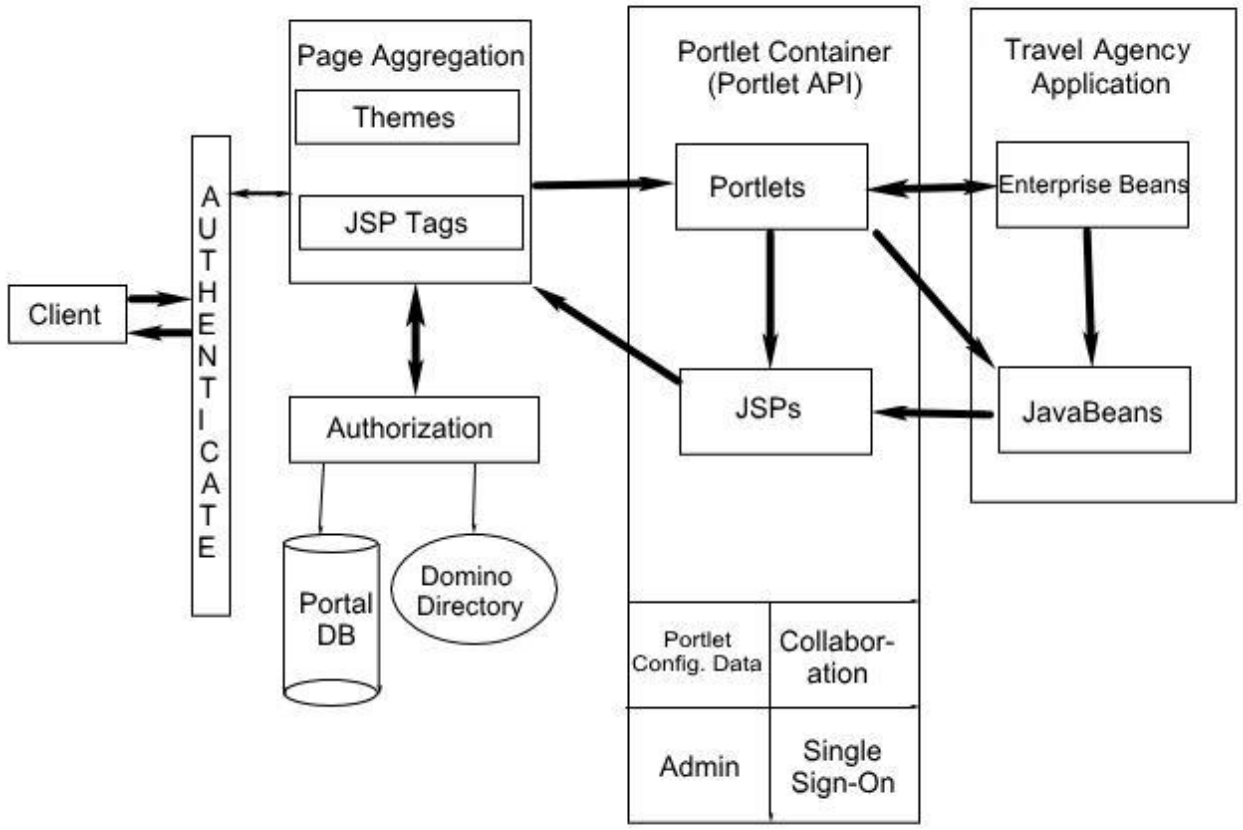


Figure 5. Travel Portal application model

Application flow

In the MVC architectural design pattern, the model represents enterprise data and the rules by which the data are accessed and updated. In the Travel Portal application, the existing Travel Agency application enterprise beans form the model. These enterprise beans contain the data that are requested by a travel agent or a customer. In the MVC architectural design pattern, the view is responsible for rendering the contents of the model. In the Travel Portal application, the JSP pages that are displayed in a user's Web browser form the view. These JSP pages are displayed when a portlet is in its View, Edit, or Help mode. The controller translates interactions from the view into actions to be performed on the model. In the Travel Portal application, this is the responsibility of the portlets. The doView(), doEdit(), and doHelp() methods in each of these portlets perform the various controller tasks.

The portal server instantiates only a single instance of each portlet, which is shared amongst all users. When each portlet is initialized, a PortletConfig object is passed to the portlet's init() method. In each portlet's init() method, a call is made to a method to retrieve the travel agent session bean context.

When a portlet is initially constructed on a Travel Portal page, it is displayed in its View mode. This mode is the normal mode of operation for each of the portlets. Each portlet has a doView() method, which accepts a PortletRequest object and a PortletResponse object as parameters. Based on the contents of the request made by travel agents or customers, the doView() method creates the appropriate travel

enterprise beans and retrieves results from the methods in those enterprise beans. The doView() method then calls the appropriate JSP file to render the data that were retrieved.

The doEdit() and doHelp() methods are responsible for controlling the portlet when the portlet is in Edit and Help modes. These methods call appropriate JSP files to render the data in Edit or Help mode.

Figure 6 illustrates the interactions between the components of the Travel Portal application, as well as the interactions between the Travel Portal and Travel Agency applications.

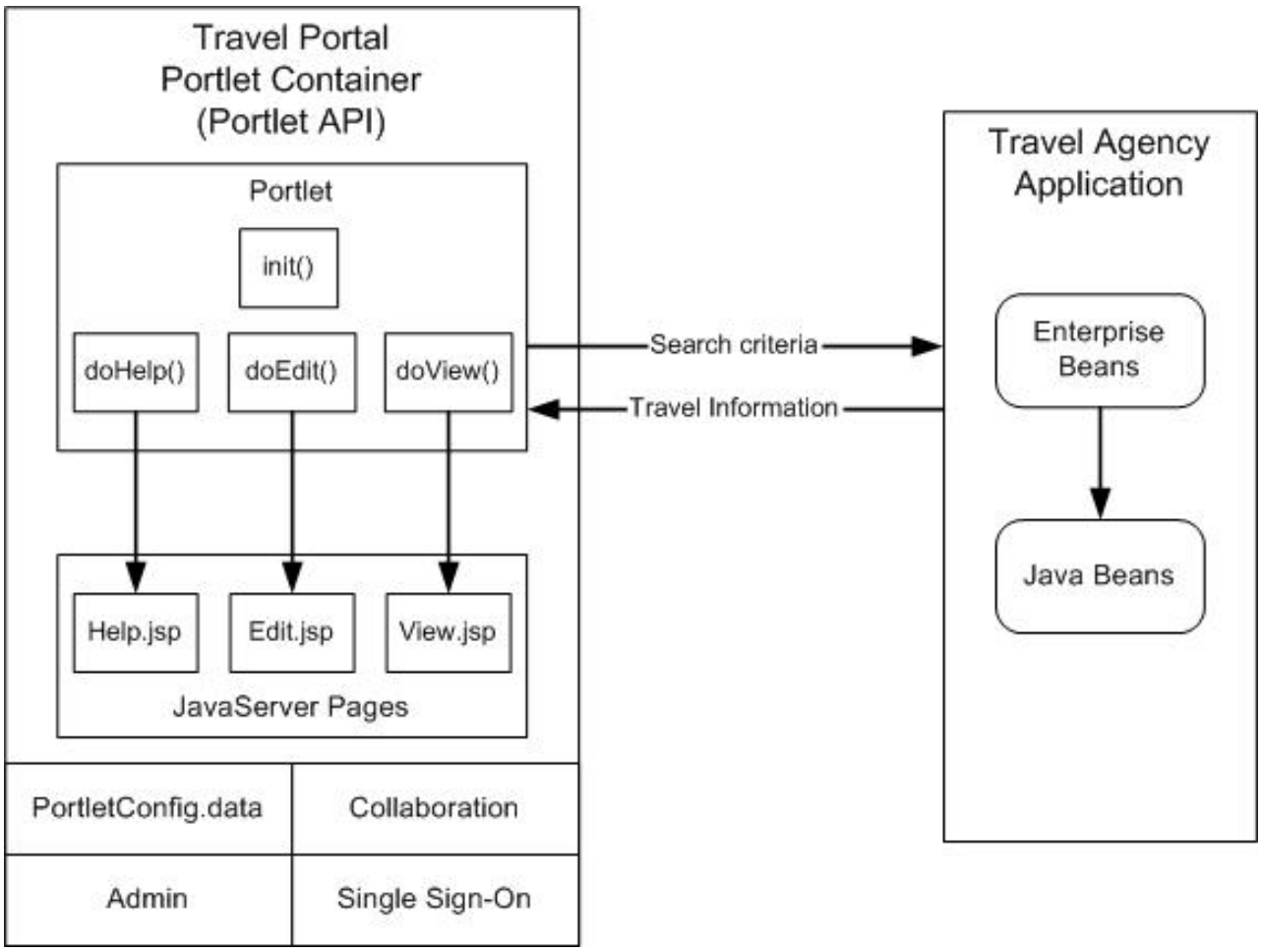


Figure 6. Travel Portal application flow

Application details

Because all of the portlets in the Travel Portal scenario follow the same application model and access much of the same data, each portlet shares some similar application design points with the other portlets in the scenario. This section describes those application design points that are common among the portlets in the scenario.

Because each portlet was created to accomplish a specific task, each portlet also has its own application design points that make it unique. This section takes a subset of portlets from the Travel Portal scenario, and describes the application design points that make each of these portlets unique.

Application design points

The Travel Portal scenario is designed to support the travel agents and customers groups. Users within each group are authorized to access portlets related to their group. Travel Portal is designed to support different places and themes within each group. Different portlets within the Travel Portal scenario share similar skeletons to achieve the required functionality.

The Travel Portal application utilizes WebSphere(R) Portal Server's unique architecture. All the portlets run within the portlet container. Each portlet uses the doView() method as the default execution point, and calls other methods if needed. Portlets first display a JSP page from within the doView() method for user input. User request parameters from the JSP pages are passed to methods within the Travel Agency enterprise beans. Travel Agency enterprise beans access the database and return data to the calling portlet. The portlet then displays this information by calling a JSP page with the results.

The following list explains the responsibility of each portlet method that the Travel Portal application design utilizes:

- The init() method constructs the initial portlet instance and performs portlet initialization. The portal always instantiates only a single instance of the portlet, which is shared among all users. This is similar to the way a servlet instance is shared among all users of an application server.
- The doView() method handles all portlet requests when the portlet is in View mode. When a portlet is initially constructed on the portal page for a user, it is displayed in its View mode. This is the portlet's normal mode of operation. This method is designed to handle most of the Travel Portal's operations and also calls other methods provided by the portlet container to achieve required functionality.
- The doHelp() method handles all portlet requests when the portlet is in Help mode. This method displays Help.jsp, which provides help for a specific portlet.
- The doEdit() method handles all portlet requests when the portlet is in Edit mode. This method allows the travel agent or customer to change the settings for the portlet. Changing the settings for the portlet will affect the way this portlet runs and returns responses to the travel agent or customer.

The BestDeals portlet went through design and coding before the other portlets, and set up a framework for data request - EJB method access - data response - display in the design and coding phases. Other portlets adapted the style used by the BestDeals portlet to capture the same framework, but to provide different functionality. Iteratively, errors in design and coding were removed during the review of the BestDeals portlet; more importantly, errors were detected early before other portlets were developed.

BestDeals portlet

The BestDeals portlet is available in the Travel Agents and Travel Customers labels. It allows any user to view the best deals available on different cruises. The portlet renders a JSP page where the user can specify the search criteria to be used. The portlet allows users to specify a location, specify a departure date, or select cruises departing within the next 30 days. When this information is submitted, the portlet invokes the TravelAgent enterprise bean from within the Travel Agency application. The enterprise bean obtains the cruises based on the search criteria and this information is returned to the user through a second JSP page, which displays a list of cruises that meet the specified criteria. The list of cruises is ordered according to the customer's single occupancy price.

The BestDeals portlet also makes use of portlet messaging, which allows portlets located on the same page to communicate with each other. Portlets can use messages to share information or notify each other of a user's actions. The BestDeals portlet gives a user the ability to select a cruise from the list, and then view the cruise details using the DisplayCruise portlet. For more information about the DisplayCruise portlet, see "DisplayCruise portlet" on page 21.

Figure 7 shows the BestDeals portlet after a user specified the search criteria and obtained the list of available cruises.

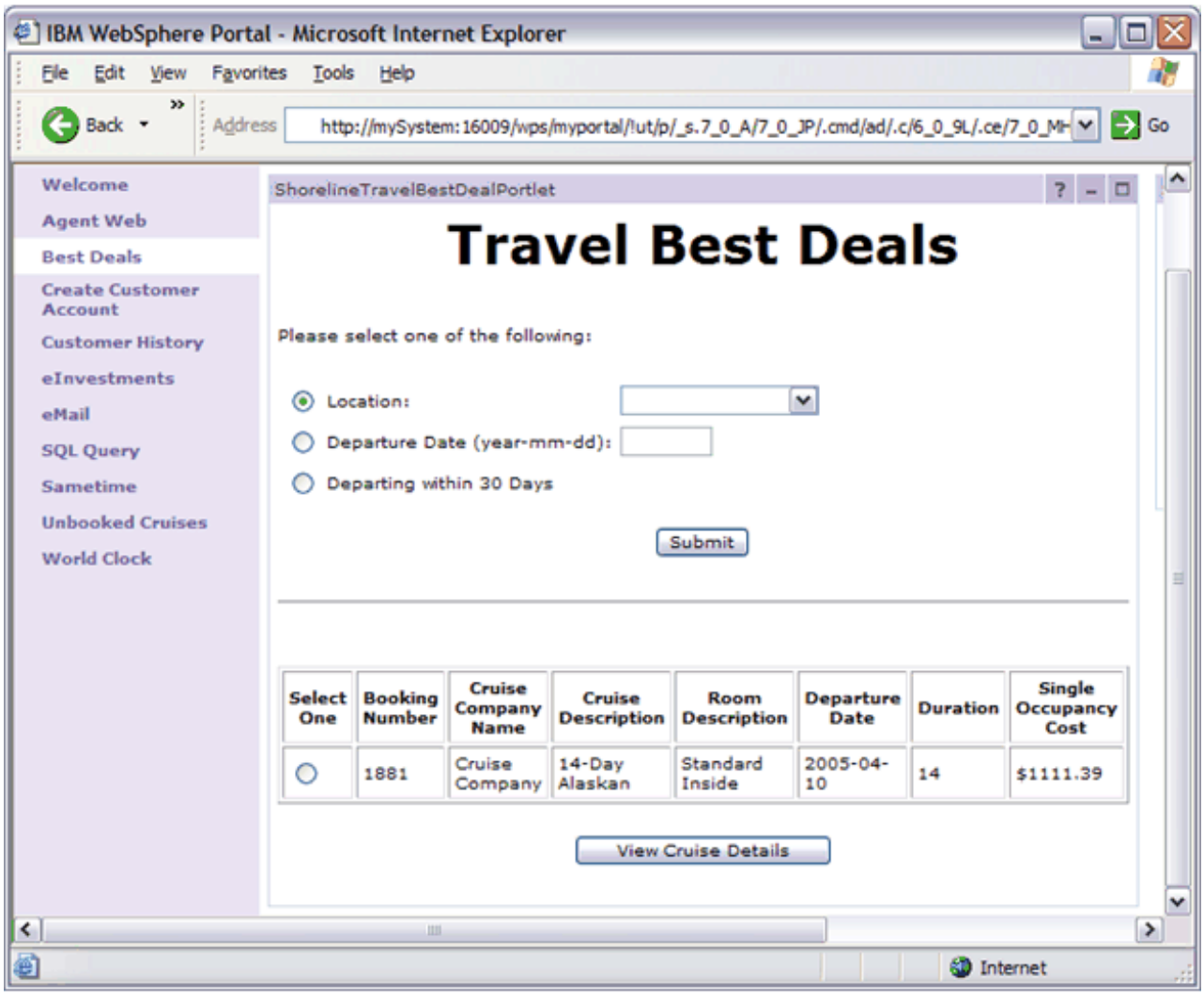


Figure 7. BestDeals portlet

Example Code

The following code snippet is the BestDeals doView() method. It is used to display BestDealsSearch.jsp. Once the data is entered for the search criteria and the submit button is clicked, the doView() method uses the TravelAgent session bean to obtain a list of cruises that match the search criteria. It then displays BestDealsList.jsp.

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

  // Invoke the JSP to render the BestDealsSearch page
   getPortletConfig().getContext().include("/jsp/BestDealsSearch."
      +getJspExtension(request), request, response);

   // Determine if a radio button is selected or if viewDetails has been selected.
   String radioSelected = request.getParameter("selectedRadioButton");
   String viewDetailsSelected = request.getParameter("viewDetailsSelected");

   if ( (radioSelected == null) && ( (viewDetailsSelected == null) ||
      (viewDetailsSelected.equals("no") ) ) )
   {
      radioSelected = "";
```

```
}
else
{
   // Depending on which radio button was selected, check to see if user has
   // entered any values
   String aType = "";
   String aSearchCriteria = "";

   // First check to see if values should exist in session to be used for
   // search criteria
   if (!(viewDetailsSelected == null))
   {
      aType = (String) request.getSession().getAttribute(A_TYPE);
      aSearchCriteria = (String) request.getSession().getAttribute(A_CRITERIA);
   }
   // Location specified
   else if (radioSelected.equals("location"))
   {
      aType = "location";
      aSearchCriteria = request.getParameter("location_name");
   }
   // Departure time specified
   else if (radioSelected.equals("departure"))
   {
      aType = "departureDate";
      aSearchCriteria = request.getParameter("departure_date");
   }
   // 30 Days specified
   else if (radioSelected.equals("30days"))
   {
      aType = "30Days";
      aSearchCriteria = "";
   }

   if (aType == null)
   {
      aType = "";
      aSearchCriteria = "";
   }

   // Get the session values
   request.getSession().setAttribute(A_TYPE, aType);
   request.getSession().setAttribute(A_CRITERIA, aSearchCriteria);

   // If aType is specified, find the cruises and display the BestDealsList page
   if (!(aType.equals("")))
   {
      CruiseInfoBean cruiseInfoBean = null;

      try
      {
         TravelAgent travelAgent = taHome.create();

         cruiseInfoBean = (CruiseInfoBean)
            Beans.instantiate(getClass().getClassLoader(),
            "com.shoreline.CruiseInfoBean");
         cruiseInfoBean = travelAgent.getBestDeals(aType, aSearchCriteria);

         // Add action event for handling the viewing of the cruise details
         // using the DisplayCruises portlet
         PortletURI viewCruiseDetailsURI = response.createURI();
         viewCruiseDetailsURI.addAction("viewCruiseDetails");
         request.setAttribute("viewCruiseDetailsURI",
            viewCruiseDetailsURI.toString());

         // Invoide the JSP to render the BestDealsList JSP
         request.setAttribute("cruiseInfoBean", cruiseInfoBean);
```

```
            getPortletConfig().getContext().include("/jsp/BestDealsList."
               +getJspExtension(request), request, response);
         } // end of try
         catch (Exception ex)
         {
            System.out.println("Error in BestDeals.doView().");
            ex.printStackTrace();
         } // end of catch
      } // end of if
   } // end of if
}
```

The following code snippet is the BestDeals actionPerformed() method. This method is used to extract the bookingNumber, create a new message object, and send the message object to the DisplayCruise portlet; it can display the details for the selected cruise.

```
public void actionPerformed(ActionEvent event) throws PortletException {
   // Getting action and sending appropriate message
   String actionString = event.getActionString();
   if (actionString.equals("viewCruiseDetails"))
   {
      DefaultPortletMessage message = new
         DefaultPortletMessage(event.getRequest().getParameter("bookingNumber"));
      getPortletConfig().getContext().send("DisplayCruisePortlet", message);
   }
}
```

JSP files

The following JSP files are used within the BestDeals portlet:

- BestDealsSearch.jsp: This JSP file is used to display the list of search criteria for the portlet. The JSP file ensures only one item is selected.
- BestDealsList.jsp: This JSP file is used to display the list of cruises that meet the search criteria specified by the user.
- BestDealsHelp.jsp: This JSP file is used to display the help information for the BestDeals portlet.

CreateCustomerAccount portlet

The CreateCustomerAccount portlet is available in the Travel Agents label and is used by travel agents to create customer accounts. This portlet renders a JSP page where the customer name, address, phone number, and date of birth can be entered. There is also a password field where a password for the customer can be entered. When this information is submitted, the portlet invokes the TravelAgent enterprise bean in the Travel Agency application and an account is created with a unique customer number. The customer number, along with the customer information originally entered, is returned to the user through a second JSP page.

This portlet does not directly add users to the portal server. Instead, the travel agent needs to add users using the **Users and Groups** portlet within **Administration**.

Figure 8 shows the CreateCustomerAccount portlet.

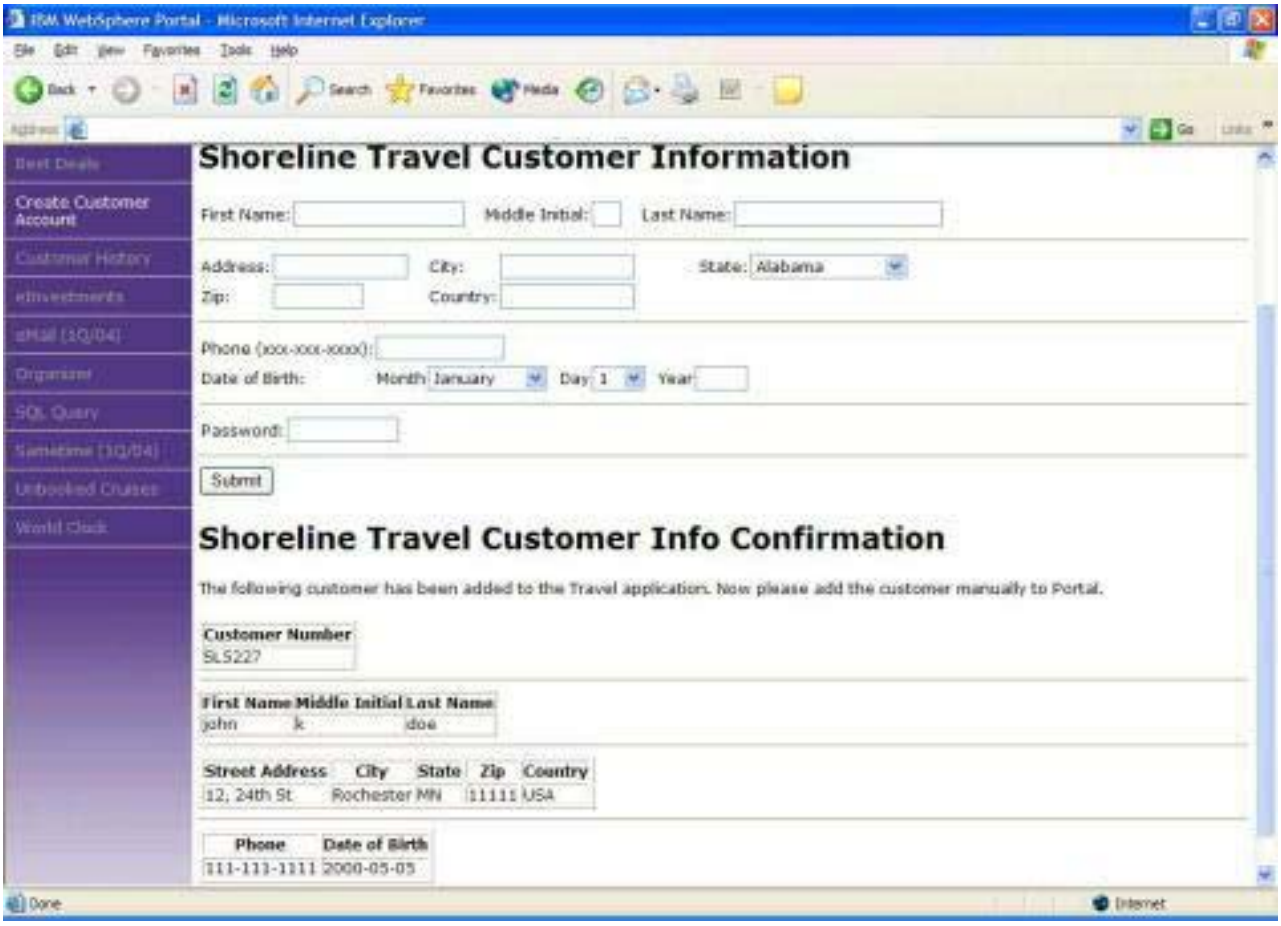


Figure 8. CreateCustomerAccount portlet

Example Code

The following code snippet is the doView() method from the CreateCustomerAccount portlet. The doView method renders the CreateCustomer.jsp to collect the customer information and password. It then invokes the addCustomer() method of the TravelAgent enterprise bean to create the customer account, and finally renders the CustomerInfoConfirmation.jsp to return the customer number generated by the Travel Agency application, and the customer information originally entered.

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

   // Render the CreateCustomer.jsp

   getPortletConfig().getContext().include("/jsp/CreateCustomer."
         +getJspExtension(request), request, response);
   .
   .
   .
   //Invoke Travel Agent addCustomer method
   TravelAgent travelAgent = taHome.create();
   CustomerInfoBean customerInfoBean =
         (CustomerInfoBean) Beans.instantiate(getClass().
          getClassLoader(), "com.shoreline.CustomerInfoBean");

   customerInfoBean = travelAgent.addCustomer(request.getParameter("fname"),
   request.getParameter("mi"),
   request.getParameter("lname"), request.getParameter("addr"),
   request.getParameter("city"),
```

```
    request.getParameter("state"), request.getParameter("zip"),

    request.getParameter("country"),
    request.getParameter("phone"), dateofBirth,
    request.getParameter("password"));


    request.setAttribute("customerInfoBean", customerInfoBean);

    // Render the CustomerInfoConfirmation.jsp details
    getPortletConfig().getContext().include("/jsp/CustomerInfoConfirmation." +

    getJspExtension(request), request, response);
}
```

JSP files

The following JSP files are used in the CreateCustomerAccount portlet:

- CreateCustomer.jsp: This JSP file displays the following fields: first name, middle initial, last name, address, city, state, country, zip code, phone number, date of birth (month, day, year), and the password. The JSP file ensures that all fields have been filled in and returns the data entered to the portlet.
- CustomerInfoConfirmation.jsp: This JSP file extracts the CustomerInfoBean from the request parameter, and displays the customer number and customer information that are contained in the bean.

DisplayCruise portlet

The DisplayCruise portlet uses portlet messaging, which allows other portlets located on the same page to communicate with it. The portlet is available on some of the pages contained in the Travel Agents and Travel Customers labels. It allows a travel agent or a customer to view the detailed information for a specific cruise. It is designed to receive messages from other portlets that specify the booking number of the cruise. The DisplayCruise portlet obtains the detailed information for that cruise using the TravelAgent EJB within the Travel Agency application. The detailed cruise information is returned to the user through a second JSP page. There are two different views of the data, depending on whether the user is an agent or a customer.

Figure 9 shows the DisplayCruise portlet.

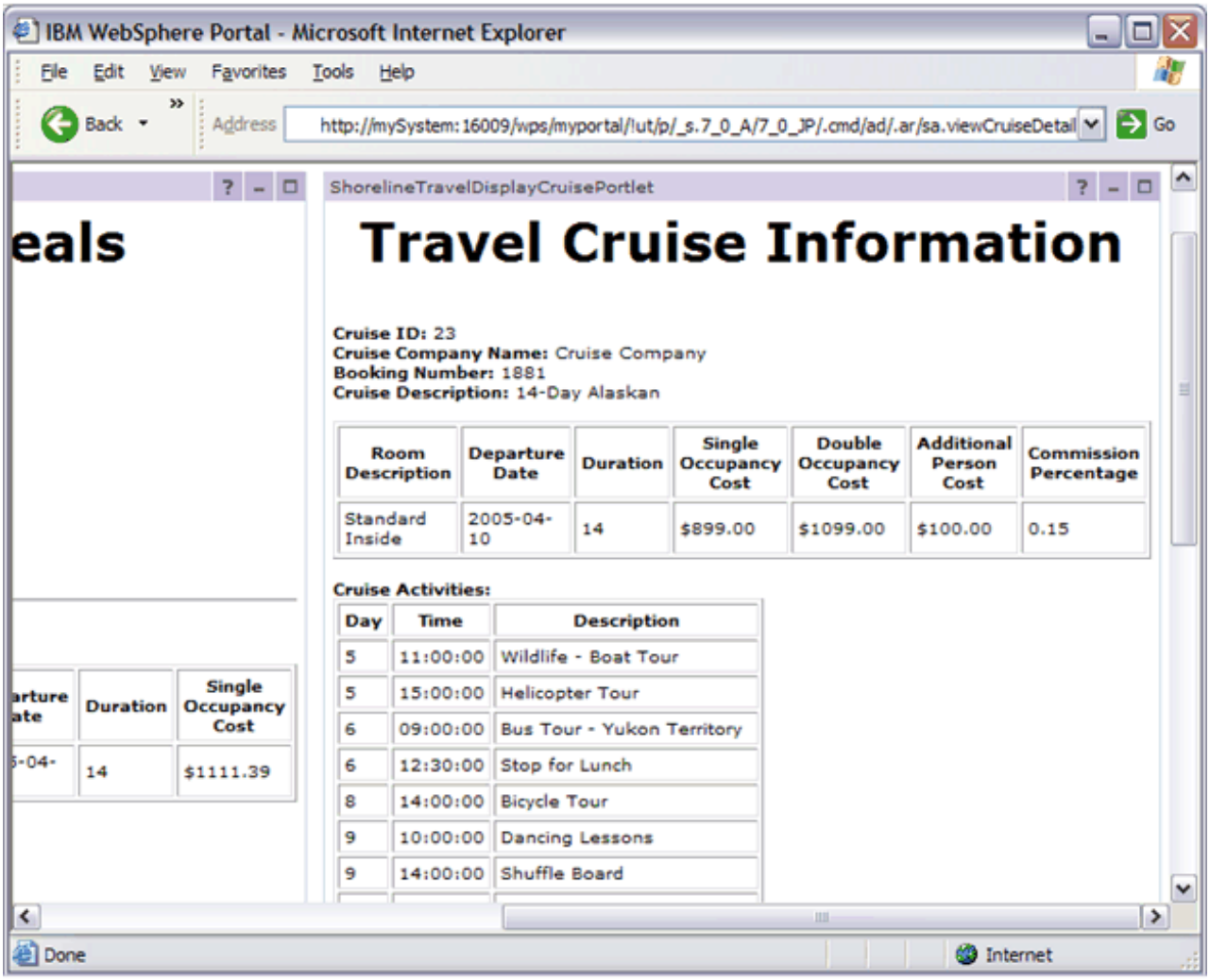


Figure 9. DisplayCruise portlet

Example Code

The following code snippet is the DisplayCruise messageReceived() method. The code snippet accepts the message event; if the event is of type URLMessage, the code snippet casts event to the correct type. It then retrieves the booking number from the message and stores it in the session.

```
public void messageReceived(MessageEvent event) throws PortletException {

   // Obtain booking number from the message and set in session
   if (event.getMessage() instanceof DefaultPortletMessage)
   {
      DefaultPortletMessage msg = (DefaultPortletMessage) event.getMessage();
      PortletSession sess = event.getRequest().getPortletSession();
      event.getRequest().setAttribute("bookingNbr", msg.getMessage());
   }
   else
   {
      System.out.println("Message is not of type URLMessage");
   }
}
```

The following code snippet is the DisplayCruise doView() method. It is used to obtain the detailed information for the specified cruise using the TravelAgent session bean. The code snippet then determines

if the user is an agent or a customer, and displays the appropriate JSP page (AgentCruiseDetail.jsp or CustomerCruiseDetail.jsp) to view the detailed information for the cruise.

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

   // Obtain booking number from session
   String bookingNbr = (String) request.getAttribute("bookingNbr");

   // Use TravelAgent session bean to obtain information for cruise
   CruiseInfoBean cruiseInfoBean = null;
   CruiseActivitiesInfoBean cruiseActivitiesInfoBean = null;

   if (!(bookingNbr == null))
   {
      try
      {
         TravelAgent travelAgent = taHome.create();

         cruiseInfoBean = (CruiseInfoBean)
            Beans.instantiate(getClass().getClassLoader(),
            "com.shoreline.CruiseInfoBean");
         cruiseActivitiesInfoBean = (CruiseActivitiesInfoBean)
            Beans.instantiate(getClass().getClassLoader(),
            "com.shoreline.CruiseActivitiesInfoBean");
         cruiseInfoBean = travelAgent.getCruise(bookingNbr);

         cruiseActivitiesInfoBean =
            travelAgent.getCruiseActivities(cruiseInfoBean.getCruiseCompany(),
            cruiseInfoBean.getCruiseID());
         request.setAttribute("cruiseInfoBean", cruiseInfoBean);
         request.setAttribute("cruiseActivitiesInfoBean", cruiseActivitiesInfoBean);

         // Determine which page to bring up according to User ID
         String user = "";
         if (!(request.getUser() == null))
            user = request.getUser().getUserID().toString().substring(0,3);
         if ( (user.toUpperCase().equals("TFC")) ||
            (user.toUpperCase().equals("WebSphere Portal Server")) )
            getPortletConfig().getContext().include("/jsp/AgentCruiseDetail."
            +getJspExtension(request), request, response);
         else
            getPortletConfig().getContext().include("/jsp/CustomerCruiseDetail."
            +getJspExtension(request), request, response);
      }
      catch (Exception ex)
      {
         log("Error in ShorelineTravelDisplayCruisePortlet.doView().");
         System.out.println(ex);
         ex.printStackTrace(System.out);
      }
   }
   else
   {
      getPortletConfig().getContext().include("/jsp/CruiseEmptyList."
         +getJspExtension(request),
         request, response);
   }
}
```

JSP files

The following JSP files are used within the DisplayCruise portlet:

- CruiseEmptyList.jsp: This JSP file is used when the portlet is started on a page and no booking number has been sent to it yet.

- AgentCruiseDetail.jsp: This JSP file is used to display the detailed information for a cruise when the user is a travel agent. It displays the following information: cruise ID, cruise company name, booking number, cruise description, room description, departure date, duration, single occupancy cost, double occupancy cost, additional person cost, commission percentage, cruise activities, and ports of call.
- CustomerCruiseDetail.jsp: This JSP file is used to display the detailed information for a cruise when the user is a customer. It displays the same information as AgentCruiseDetail.jsp, except that the commission percentage will not be displayed. The commission is already calculated into the single, double, and additional person costs.
- DisplayCruiseHelp.jsp: This JSP file is used to display the help information for the DisplayCruise portlet.

AgentWeb portlet

The AgentWeb portlet is available in the Travel Agents label. It allows travel agents to access the existing Web-based Travel Agency application. The AgentWeb portlet calls the getUser() method provided by the portlet container. It extracts the agent ID of the agent who has logged in to the portal and passes it to the viewShoreline.jsp. The viewShoreline.jsp invokes the existing Travel Agency application by passing the agent ID as a query string to the Travel Agency servlet. The viewShoreline.jsp displays the Travel Agency application within it, after the user is authenticated by the Travel Agency application.

Figure 10 shows the AgentWeb portlet.

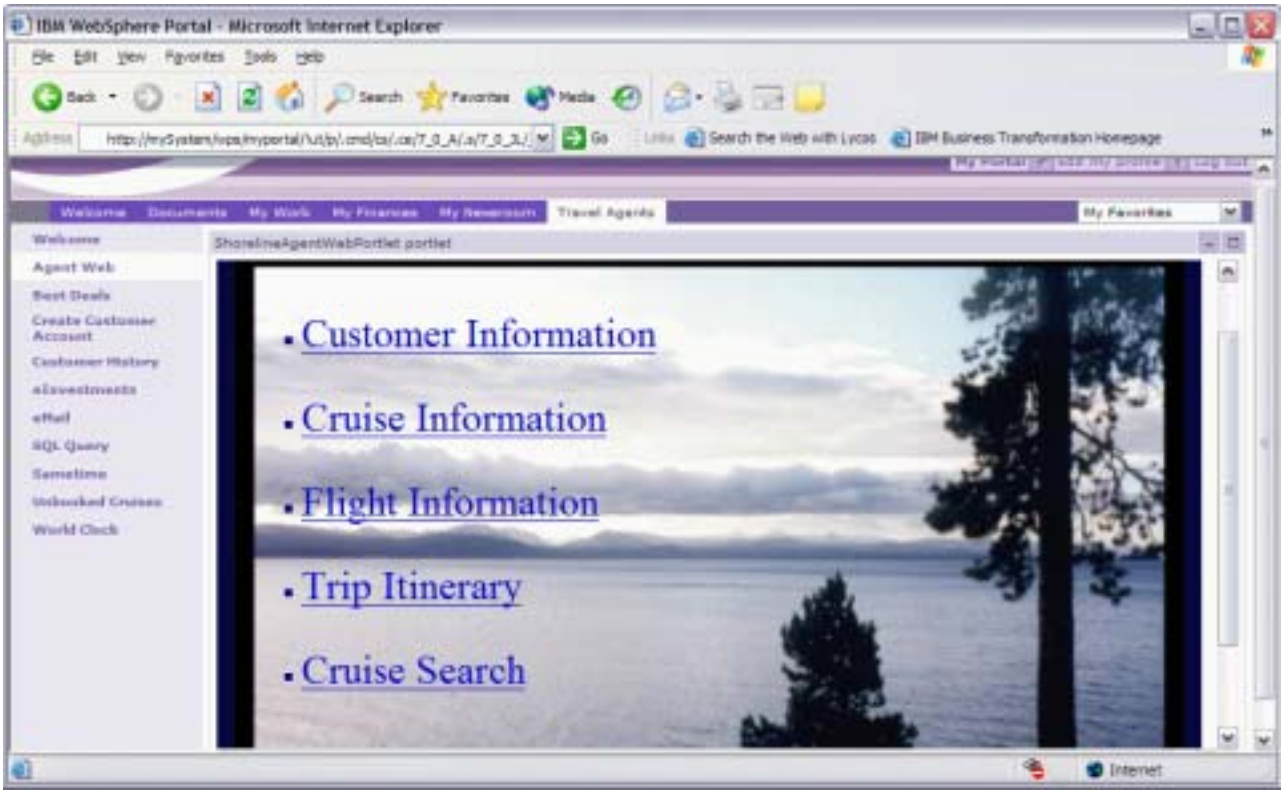


Figure 10. The AgentWeb portlet

Example Code

The following code snippet is the AgentWeb doView() method. This method makes a call to the getUser() method provided by the portlet container, storing the user ID in the bean.

```
public void doView(PortletRequest request, PortletResponse response)
  throws PortletException, IOException {
  // Make a bean
   AgentWebBean bean = new AgentWebBean();

   // Save name in bean
   bean.setPortletName("ShorelineAgentWeb Portlet");

   // Save bean in request
   request.setAttribute("AgentWebBean", bean);

   // Obtain userId
   String userId = request.getUser().getUserID().toString();

   // Pass the userId value to viewShoreline.jsp
   bean.setUsername(userId);

   // Render Jsp to call shoreline application
   getPortletConfig().getContext().include("/jsp/viewShoreline." + getJspExtension(request),request,response);
}
```

JSP files

The following JSP files are used within the AgentWeb portlet :

- viewShoreline.jsp: This JSP file is used to display the Travel Agency application. The following code snippet illustrates the file. The viewShoreline.jsp retrieves the agent ID from the Java(R) bean, passes the agent ID to the Travel Agency application as a query string, and then displays the Travel Agency application.

```
<%@pagecontentType="text/html" %>
<%@ taglib uri='/WEB-INF/tld/portlet.tld' prefix='portletAPI'%>
<portletAPI:init/>
<jsp:useBean id="AgentWebBean" class="com.shoreline.portlets.AgentWebBean" scope="request" />
<IFRAME src='http://MYSYSTEM.SHORELINE.COM:3080/webapp/Shoreline/ShorelineServlet?userIdValue=<%=AgentWebBean.getUsername()%>'
   width = "100%" height="400" scrolling="AUTO" frameborder="0">
Your browser does not support frames
</IFRAME>
```

- AgentWebHelp.jsp: This JSP file is used to display the help information for the AgentWeb portlet.

Collaboration portlets

To take advantage of the collaboration features supported by WebSphere(R) Portal Server 5.0.2.2, the Travel Portal team deployed and configured portlets to give customers and travel agents the ability to send instant messages and send e-mail to one another. The Lotus(R) Sametime(R) portlet was deployed to provide instant messaging capability to users. When this portlet is viewed by a user, a Sametime login window appears, which prompts the user for their user ID and password. Upon authentication, the portlet creates another Sametime window, which displays the list of users who have logged into the Sametime server. The Quick eMail portlet was deployed to provide users the ability to send e-mail to one another. Both of these portlets are part of the Lotus Collaborative Porlets set of portlets, and are deployed by the iSeries(TM) portal wizard during the creation of a WebSphere Portal Server 5.0.2.2 server instance, if specified by the user.

Portal administration

Within WebSphere(R) Portal Server V5.0.2.2, a portal is composed of a hierarchical structure of nodes that can be represented in a parent-child relationship, starting from the content root of the portal. A node is an addressable element in the portal navigation tree belonging to one of the following types:

- **Page**: Pages display content in the form of portlets. Pages can contain child nodes, including other pages that provide content. A page can contain column containers, row containers, and portlets. Containers are columns or rows that you can use to arrange the layout of portlets or other containers on the page.

- **Labels**: Labels do not display any content, but can contain other nodes. They are used primarily to group nodes in the navigation.
- **URL**: URLs can launch any URL-addressable resource, including external Web sites or pages within the portal site.

Nodes are located in a level of the navigation hierarchy relative to the parent node in which they are created. The topmost node in the tree is the content root. After installation, the following nodes are created under the content root:

- **My Portal**: A label containing prepackaged portlets for general business or productive use. By default, this is the first page displayed after login.
- **Page Customizer**: A label containing child pages with portlets for managing page content and layout. Direct access to this mode is hidden from navigation. Instead, the portlets in the page customizer are accessed through context-sensitive links in the portal toolbar.
- **Administration**: A label containing pages with portlets used by portal administrators. A link is provided in the banner to allow users with administrative privileges to access this content.
- **Page Properties**: A page containing the **Properties** portlet, which is used for editing the properties of a page, such as locale-specific titles and description. This page is always hidden from navigation. It is accessed through context-sensitive links in the toolbar or from the **Manage Pages** portlet in **Administration**.
- **Organize Favorites**: A page containing the **Organize Favorites** portlet, which allows users to create, edit, activate, order, and delete labels and URLs. Access is provided through the **My Favorites** drop-down list in the banner.

The Travel Portal development team designed the Travel Portal application to include three labels. The labels include Travel Home, Travel Agents, and Travel Customers. Each of these labels was modified to use the Corporate theme. The theme is specified within the properties for the label and can be managed using the **Manage Pages** portlet within **Administration**. After the labels were created, individual pages for each of the supported portlets were created, the corresponding portlets were added to the page, and the page was then added to the appropriate label.

The TravelAgents and TravelCustomers groups were also created. The travel agent user IDs were added to the TravelAgents group and the customer user IDs were added to the TravelCustomers group. This was accomplished using the **Users and Groups** portlet within **Administration**. The groups were then given the appropriate authority to the labels, pages, and portlets using the **Resource Permissions** portlet within **Administration**.

Each label used within the Travel Portal scenario is described in more detail as follows:

Travel Home label

The Travel Home label is used when a user first accesses the Travel Portal server. It allows users to access the following pages and portlets:

- Welcome
- Best Deals
- Create Customer Account
- World Clock

All anonymous users have User access to the Travel Home label and the pages within it. User access allows a user to view the portal content. The anonymous users also have User access to the individual portlets contained within the pages.

Figure 11 shows the Travel Home label, and the list of pages and portlets available within it.

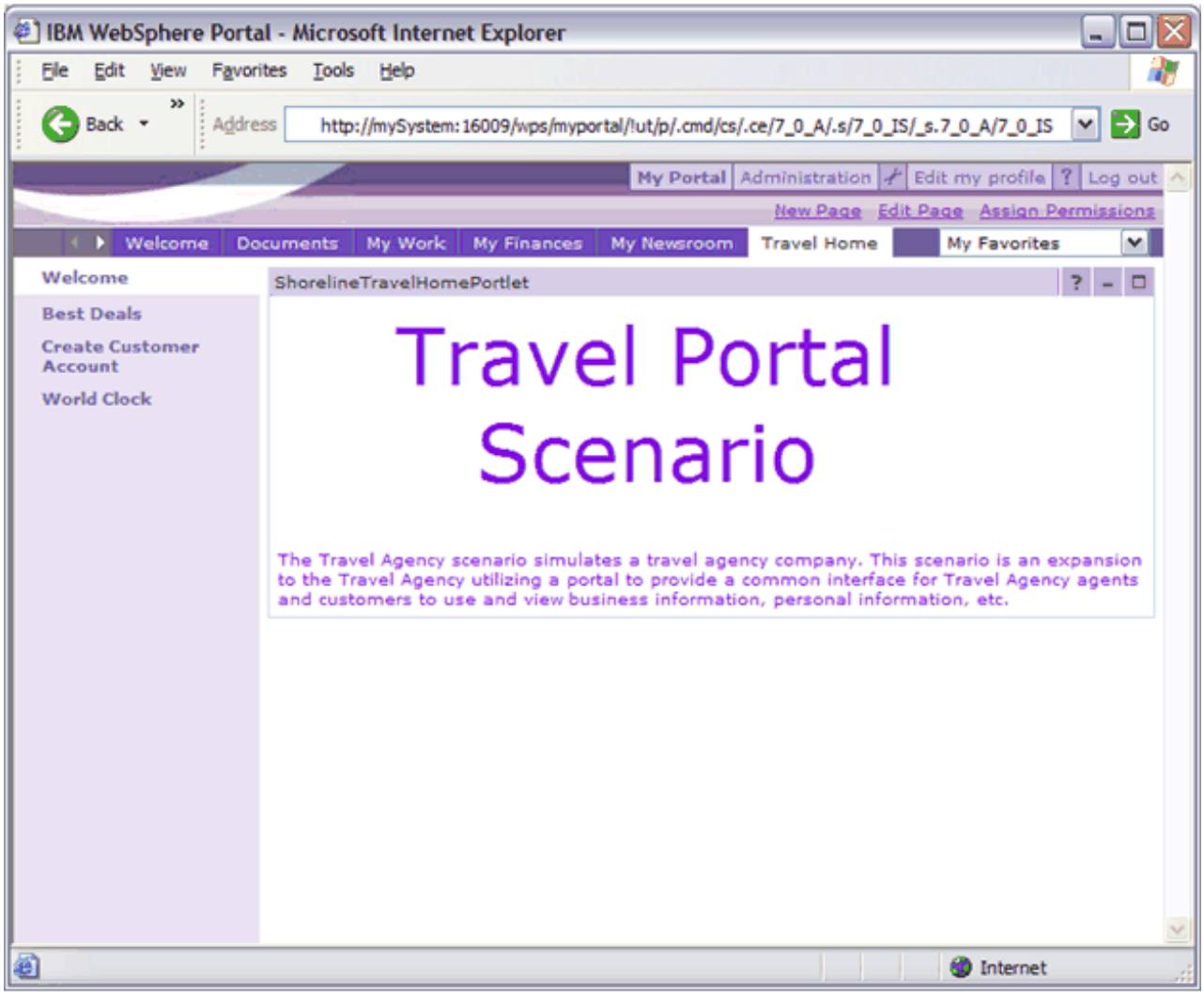


Figure 11. Travel Home label

Travel Agents label

The Travel Agents label is used by the travel agents after they have logged into the WebSphere Portal server. It allows travel agents to view the following pages and portlets:

- Welcome
- Agent Web
- Best Deals
- Create Customer Account
- Customer History
- eInvestments
- eMail
- SQL Query
- Sametime
- Unbooked Cruises
- World Clock

The TravelAgents group has User access to the Shoreline Travel Agents label and the pages within the label. The TravelAgents group also has User access to the individual portlets contained within the pages.

Figure 12 shows the Travel Agents label and the pages and portlets contained within it.

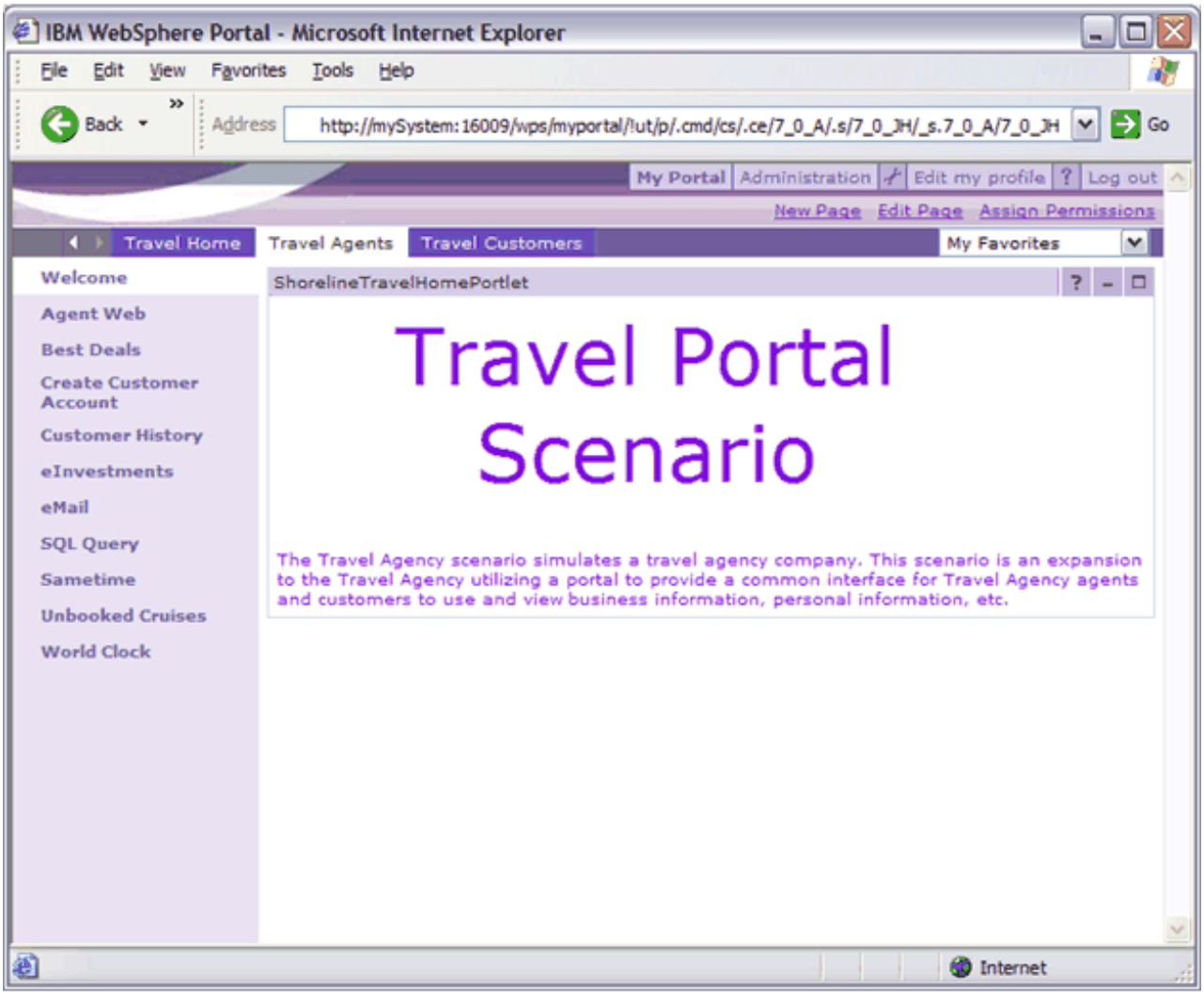


Figure 12. Travel Agents label

Travel Customers label

The Travel Customers label is used by the travel customers after they have logged into the Portal server. It allows travel customers to view the following pages and portlets:

- Welcome
- Best Deals
- Future Trips
- Sametime
- Trip Reminder
- World Clock

The TravelCustomers group has User access to the Travel Customers label and the pages within it. The TravelCustomers group also has User access to the majority of the portlets contained within the pages.

The only exception to this is the Trip Reminder portlet. The TravelCustomers group has Priveldged User access to it, so that customers can customize the portlet's settings.

Figure 13 shows the Travel Customers label and the pages and portlets contained within it.

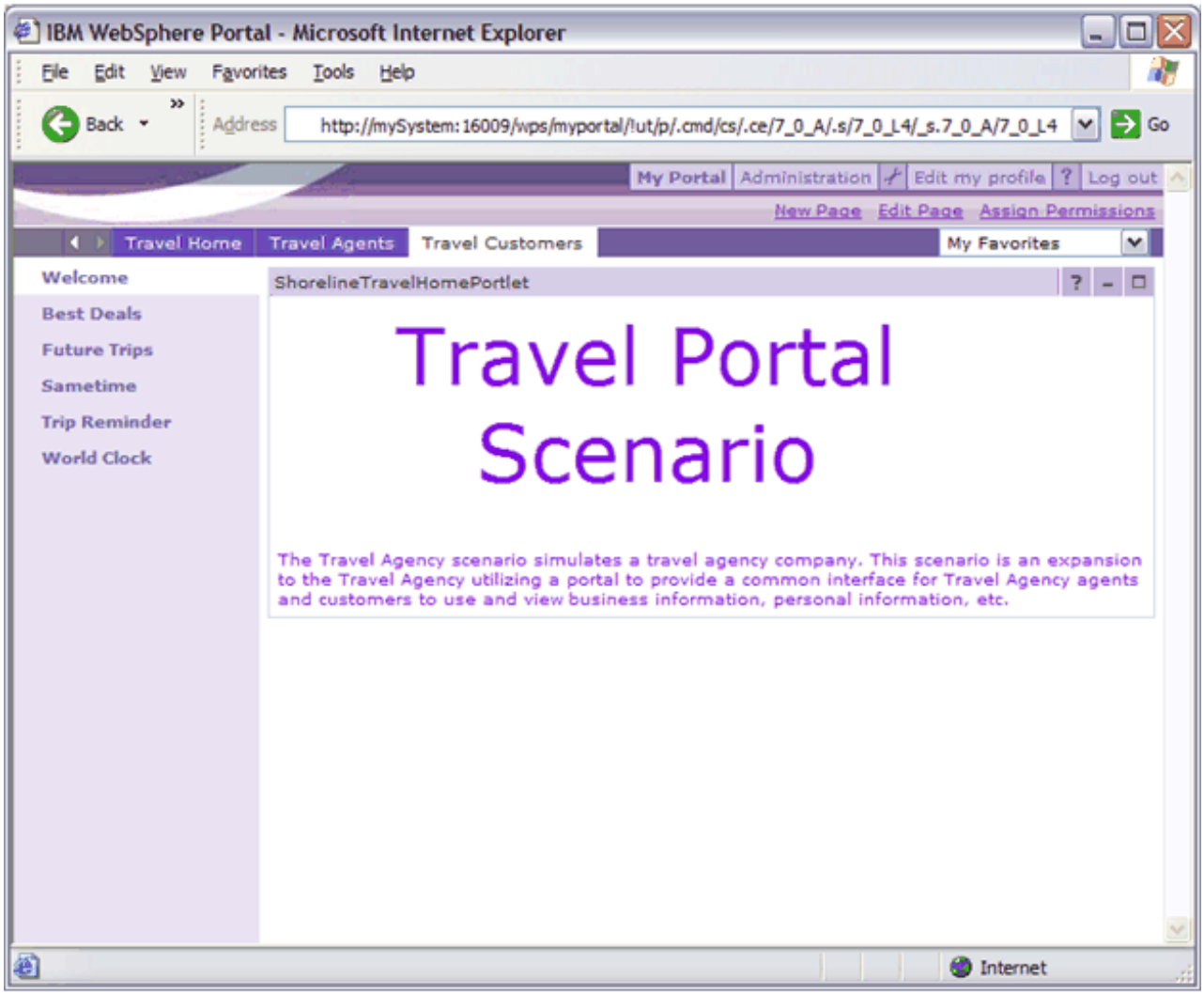


Figure 13. Travel Customers label

Installation of portlets

The installation of a portlet within WebSphere(R) Portal Server consists of the following steps:

- Export the portlet as a Web archive (WAR) file
- Deploy the WAR file to the target system
- Install the portlet within WebSphere Portal Server
- Activate the portlet

Following are links to the instructions for installing portlets within WebSphere Portal Server V5.0. Note that the instructions include references to a specific portlet called the Hello World portlet. As the instructions are followed, details specific to the portlet being installed need to be used in place of those for the Hello World portlet.

The steps to export the portlet to a WAR file from WebSphere Studio Application Developer can be found in chapter 16.4 at IBM WebSphere Portal V5, A Guide for Portlet Application Development  .

The detailed steps to deploy, install and activate a portlet within WebSphere Portal Server V5.0 can be found at Hello World, the simplest portlet for WebSphere Portal V5  . Select the **Deploy the WAR file** link to find the steps to deploy the portlet. Select the **Adding a portlet to a page** link to install and activate the portlet.

Portal application discoveries

This section lists the discoveries that were made during the design, implementation, and installation of the Travel Portal portlets.

- Within a portal application, you can define configuration parameters in the portlet.xml file. The Travel Portal portlets utilize this support by specifying the PROVIDER_URL value, which is used to access the Travel Agency EJBs, within the **Configuration Parameters** table of the portlet.xml file. It is then read and used within the portlet's init() method using the following line of code:

  ```
  providerURL = portletConfig.getInitParameter("PROVIDER_URL");
  ```

- To make the edit and help buttons appear within a particular portlet, you need to enable the edit and help views in the portlet.xml file for each portlet. Within portlet.xml, select **Portlet Applications**, then select the **Markups** section, and change the html markup to specify **Fragment** for the **Edit** and **Help** modes.
- While using IBM(R) WebSphere(R) Studio Application Developer to develop portlets, use the default JSP files provided within the HTML folder rather than using the JSP files in the JSP folder. When the portlet events occur, they call the JSP files located in the HTML folder.
- There is an easy way to make changes to a portlet that has already been installed and added to pages within the portal server. Within the **Portal Administration** place, select the **Portlets** tab, select the **Manage Portlet Applications** tab, select the appropriate Web module, and select **Update**. Then specify the file location for the new WAR file. This installs the updated WAR file and you do not need to add the portlet back into the pages or set up authority for the portlet again.
- The Travel Portal scenario needed a portlet that could be used to display the contents of a specified Web address. The development team found that the IBM Workplace Solution Catalog  contains IBM Web Page Portlet V5.0, which satisfied this need. This portlet was downloaded from the catalog site, installed in the portal server, and made available within the Travel Agents label. After installed and configured, the portlet enables travel agents to access an existing investment company (another test scenario).

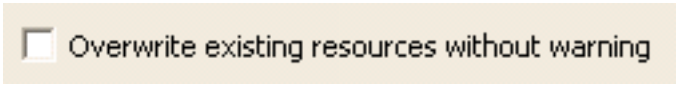
  In addition to the IBM Web Page portlet, the IBM Workplace Solution Catalog also contains the following useful items:

  – Portlets and portlet builders
  – Tools and other applications that integrate with WebSphere Portal
  – Solutions, such as services and value-added offerings

- If you have developed portlets in WebSphere Studio Application Developer 5.0.x using an earlier version of the Portal Toolkit, you must export those portlets from the earlier version and import them into WebSphere Studio Application Developer 5.1.x with the appropriate version of the Portal Toolkit. After this is complete, you can export the portlets to a new WAR file and deploy them to a WebSphere Portal Server 5.x server instance. Execute the following steps to accomplish this :

  1. Export the existing project to a WAR file in earlier version. To do this, in the earlier version of Portal Toolkit, export each project to a WAR file with source files:
     a. Right-click the project and select **Export**.
     b. Select **WAR file** and **Export source files** and click **Finish**.

2. Import the portlet WAR file
   a. In the new version of Portal Toolkit, create a new empty portlet project in the J2EE level in which the existing project was developed.
      1) Select **File** > **New** > **Project** > **Portlet Development** > **Portlet Project** or **Portlet Project (JSR 168)**.
      2) Select **Create empty portlet**, select the **Configure advanced options** checkbox, and click **Next**.
      3) Select the **J2EE Level** in which the existing project was developed and click **Finish**.
   b. Import the WAR file to this new empty portlet project.
      1) Right-click the project and select **Import**.
      2) Select **WAR file** and specify the WAR file you exported in **step 1**.
      3) Do NOT select the **Overwrite existing resources without warning** checkbox.



      4) Click **Yes** to overwrite the **portlet.xml** file and the **web.xml** file during importing.
3. Delete the TLD file. If you do not delete the TLD file from the project it exists, you will get a warning message when you rebuild the project. Leaving the TLD file as is might cause problems when the portlet project is deployed to WebSphere Portal and the TLD file of the portlet is different from the file in the server.

- The following are links to the information that the Travel Portal development team found helpful during the development of the Travel Portal application:
  - WebSphere Portal Zone
  - Information Center for IBM WebSphere Portal - Express for Multiplatforms Version 5.0.2 (iSeries)
  - IBM WebSphere Portal - Express for Multiplatforms V5.0.2 Release Notes for iSeries
  - WebSphere Portal Catalog
  - WebSphere Portal on iSeries

Disclaimer

Information is provided ″AS IS″ without warranty of any kind. Mention or reference to non-IBM products is for informational purposes only and does not constitute an endorsement of such products by IBM.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Printed in USA

Trademarks

IBM, the IBM logo, DB2, Domino, eServer, iSeries, i5/OS, Lotus, Sametime, and WebSphere are registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

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

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