BEA WebLogic Workshop™ Help

Version 8.1 SP4
December 2004

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Samples

WebLogic Workshop includes numerous fully documented samples to help you become familiar with WebLogic Workshop web service design patterns, page flows, features and programming techniques. Samples are provided that illustrate web services and web service clients.

Topics Included in This Section

The Getting Started Samples

Describes a collection of samples that provides a basic demonstration of the core J2EE components you can build with WebLogic Workshop.

TutorialsApp Sample

Demonstrates a complete WebLogic Workshop application that integrates custom Java control, web service and page flow technologies.

EJB Tutorial Sample

Demonstrates a complete WebLogic Workshop application that integrates EJBs, EJB controls, and page flow technologies.

Page Flow and JSP Samples

Demonstrates how to use page flow and JSP technologies to build dynamic web applications and highlight how to use WebLogic Workshop to as a web application development tool.

Web Service Samples

Describes a large collection of web service samples. Each sample focuses on a small set of WebLogic Workshop features and techniques.

Web Service Client Samples

Provides the Java Client and .NET samples, which describes how to use the Java proxy that WebLogic Workshop provides for each web service and how to use a WebLogic Workshop web service from a .NET client, respectively.

WebLogic Workshop Portal Extensions Samples

Provides a sample portal Web application, a sample portal file, and sample portlets that you can view and use in your own applications.

Enterprise JavaBean Samples

Describes a collection of Enterprise JavaBean samples. Each sample describes a small set of EJB development features and techniques.

Related Topics

Tutorials

The Getting Started Samples

The samples in this section provide a basic demonstration of the core J2EE components you can build with WebLogic Workshop. Alternatively, you can create these samples by running The Getting Started Tutorials.

Topics Included in This Section

Getting Started Samples: Web Services

Illustrates a basic web service application.

Getting Started Samples: Web Applications

Illustrates a basic page flow application

Getting Started Samples: Controls

Illustrates a basic application incorporating a custom and a built−in Java control.

Getting Started Samples: Enterprise JavaBeans

Illustrates a basic application incorporating a session and an entity bean.

Getting Started Samples: Portal

Illustrates a basic Portal application.

Related Topics

The Getting Started Tutorials

Getting Started Samples: Web Services

This sample demonstrates a basic application that shows a basic web services that receives a message and returns a response, both in a synchronous and asynchronous manner. Instead of running this finished sample, you can create this application by running the tutorial Getting Started: Web Services.

The sample application allows a name to be passed to a web service which returns a greeting. Both a synchronous and asynchronous approach is implemented. The application contains the following component:

- A *Hello.jws* web service.

Concepts Demonstrated by this Sample

- Using synchronous methods on web services
- Using asynchrony with web services.

Location of Sample Files

This sample is located at:

BEA_HOME\weblogic81\samples\platform\GettingStarted_WebServices_Sample\GettingStarted_WebService

To Run the Sample

1. Start WebLogic Server in the workshop domain.
    - ♦ On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Examples Server***.
    - ♦ On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/workshop/startWebLogic.sh
2. Open GettingStarted_WebServices_Sample\hello\Hello.jws and click the Start button.
3. Enter your name in the *Name* field and click the ***hello_synch*** button.
4. Notice that the method returns the response Hello, <your name>!
5. When the ***Workshop Test Browser*** launches, in the ***Name*** field, enter [your_name] and click ***hello_asynch***.
6. Notice that the method returns immediately. Scroll down to the ***Service Response*** section and notice that the method did not return anything. Also notice that the conversation has started.
7. Click the ***Continue this conversation*** link. Notice in the ***Message Log*** that the the hello_callback method has been invoked and that the conversation is finished. Also notice that no greeting is returned; instead the callback method sends the greeting as a parameter to the invoking client.
8. Return to ***WebLogic Workshop*** and press the ***Stop*** button to close the Test Browser.

Related Topics

Tutorial: Web Services

Getting Started Samples: Web Applications

This sample demonstrates a basic application that shows a basic web application that takes user input and returns a greeting. Instead of running this finished sample, you can create this application by running the tutorial Getting Started: Web Applications.

The sample application allows a user to enter his/her name and returns a greeting to that user. The application contains the following components:

- A *submit.jsp* page, which allows a user to enter a name.
- A *response.jsp* page, which displays a greeting.
- A *HelloController.jpf* file, which implements the navigation logic of the web application.

Concepts Demonstrated by this Sample

- Using page flows to separate web application presentation elements from logic.
- Using data binding.

Location of Sample Files

This sample is located at:

BEA_HOME\weblogic81\samples\platform\GettingStarted_WebApp_Sample\GettingStarted_WebApp_Samp

To Run the Sample

1. Start WebLogic Server in the workshop domain.
    - ♦ On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
    - ♦ On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Open GettingStarted_WebApp_Sample\hello\HelloController.jpf and click the Start button.
3. On the first web page, enter your name in the *Name* field and click the ***hello*** button.
4. Wait for the web site to return the response is Hello, <your name>!
5. Return to ***WebLogic Workshop*** and press the ***Stop*** button to close the Test Browser.

Related Topics

Tutorial: Page Flow

Getting Started Samples: Java Controls

This sample demonstrates a basic application that shows how a custom Java control and a database control are used to model a simplified sign on procedure. Instead of running this finished sample, you can create this application by running the tutorial Getting Started: Java Controls.

The sample application allows a user to enter his/her name, and returns a greeting based on the number of times the user has entered this name before. The application contains the following components:

- A *VisitDB* database control, which processes information about visitors and their number of visits.
- A *Hello* custom control, which receives a visitor's name from a client application, calls the VisitDB control to determine whether this is a new or returning visitor, and computes an appropriate response.
- A test web service that is used as the client application to enter a name. This web service invokes the Hello control's methods.

Concepts Demonstrated by this Sample

- Using custom controls to encapsulate business logic
- Using database controls to execute database queries

Location of Sample Files

This sample is located at:

BEA_HOME\weblogic81\samples\platform\GettingStarted_Control_Sample\GettingStarted_Control_Sample.

To Run the Sample

1. Start WebLogic Server in the workshop domain.

   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Open MyControlTestProject_Sample\hello\HelloTest.jws and click the Start button.
3. If this is the first time you have ever run this application, click the ***createTable*** button to make the table the database control is going to interact with.
4. Scroll down to the ***Service Response*** section, and notice that no exceptions are thrown.
5. Click the Test Operations link.
6. Enter your name in the String field and click the ***hello*** button.
7. Scroll down to the ***Service Response*** section, and notice that the response is Hello, <your name>!

   In the preceding two steps the web service invoked the hello method on the custom control. The control called the getVisits method of the VisitDB database control to find out the number of previous visits. Because you were running this test for the first time, this method returned 0, your name was added to the database using the database contro'ls insertVisitor method, and the Hello control sent the appropriate response.

8. Enter your name again, click the hello button, and notice that the response is Hello again, <your name>!

   In this step the Hello control again invoked the getVisits method of the VisitDB database control to find out the number of previous visits. This method returned 1, the database record was updated to reflect that this was your second visit, and the Hello control sent the appropriate response.
9. Click the Test Operations link, enter your name again, click the hello button, and notice that the response is Hello <your name>! This is visit number 3.
10. Repeat the test with a different name and observe the outcome.
11. Return to ***WebLogic Workshop*** and press the ***Stop*** button to close the Test Browser.

Related Topics

Tutorial: Java Control

Getting Started Samples: Enterprise JavaBeans

This sample demonstrates a basic application that shows how a session and entity bean are used to model a simplified sign on procedure. Instead of running this finished sample, you can create this application by running the tutorial Getting Started: Enterprise JavaBeans.

The sample application allows a user to enter his/her name, and returns a greeting based on the number of times the user has entered this name before. The application contains the following components:

- A *Visit* entity bean, which processes information about visitors and their number of visits.
- A *Hello* session bean, which receives a visitor's name from a client application, calls the Visit bean to determine whether this is a new or returning visitor, and computes an appropriate response.
- A test web service that is used as the client application to enter a name. This web service references the Hello bean via an EJB control.

Concepts Demonstrated by this Sample

- Using session beans to encapsulate business logic
- Using entity beans to model business objects

Location of Sample Files

This sample is located at:

BEA_HOME\weblogic81\samples\platform\GettingStarted_EJB_Sample\GettingStarted_EJB_Sample.work

To Run the Sample

1. Start WebLogic Server in the workshop domain.

   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1−−>Examples−−>WebLogic Workshop Examples−−>Start Workshop with Sample Applications***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Open MyTestProject_Sample\helloControl\HelloBeanCtrlTest.jws and click the Start button.
3. Enter your name in the String field and click the hello button.
4. Scroll down to the ***Service Response*** section, and notice that the response is Hello, <your name>!

   In the preceding two steps the web service invoked the hello method on the Hello bean via the EJB control. The Hello bean invoked the findByPrimary method of the Visit bean to find out if your name is stored in the database. Because you are running this test for the first time, your name could not be found. A record with your name was created in the database and the Hello bean sent the appropriate response.
5. Click the Test Operations link.
6. Enter your name again, click the hello button, and notice that the response is Hello again, <your name>!

In this step the Hello bean again invoked the findByPrimary method of the Visit bean to find out if your name is stored in the database. Your name was found, the database record was updated to reflect that this was your second visit, and the Hello bean sent the appropriate response.

7. Click the Test Operations link, enter your name again, click the hello button, and notice that the response is Hello <your name>! This is visit number 3.
8. Repeat the test with a different name and observe the outcome.
9. Return to ***WebLogic Workshop*** and press the ***Stop*** button to close the Test Browser.

Related Topics

Tutorial: Enterprise JavaBeans

Getting Started Samples: Portal

This sample demonstrates a basic Portal that incorporates various web sites and allows the user to personalize web content. Instead of running this finished sample, you can create this application by running the tutorial Getting Started: Portals.

The sample application allows a user to sign on to the Portal and personalize it by choosing a favorite color. The application contains the following components:

- The Portal framework.
- A login page flow to sign on to the Portal.
- A favorite color application to pick a favorite color.

Concepts Demonstrated by this Sample

- Using a Portals to organize various web applications and web−based content into a single web site.
- Using Portal personalization features to filter content depending on the characteristics of the user.
- Using Portal personalization features to make applications respond to individual patterns of use
- Using Portal personalization features to allow users to personalize the look and feel of the Portal.

Location of Sample Files

This sample is located at:

BEA_HOME\weblogic81\samples\platform\GettingStarted_Portal_Sample\GettingStarted_Portal_Sample.wor

To Run the Sample

1. Start WebLogic Server in the workshop domain.

   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Open PortalProj_Sample\my.portal and click the Start button.
3. When the ***Workshop Test Browser*** launches, click the ***Page 2*** tab.

   Notice that the user must be logged in to access the Favorite Color application.
4. Click the ***Page 1*** tab and click the ***login*** link.
5. Login with the username/password combination ***weblogic/weblogic***.
6. Click the ***Page 2*** tab.

   Notice that the Favorite Color Application is visible once the user is logged in.
7. Select a favorite color and click ***Click***.
8. Click the ***Page 1*** tab and click the ***logout*** link.
9. Login again with the username/password combination ***weblogic/weblogic***.

10. Click the ***Page 2*** tab.

    Notice that the Portal remembered the user's favorite color while the user was logged out.

Related Topics

Portal Tutorials

TutorialsApp Sample

The TutorialsApp Sample demonstrates a complete WebLogic Workshop application that integrates custom Java controls, web services and page flows.

The TutorialsApp application assembles credit−worthiness reports on individual loan applicants. The application consists of three tiers: (1) the application core, a custom Java control, assembles the report from a variety of resources, (2) the web services tier provides web access to the core functionality of the Java control, (3) and a page flow provides a user interface for the application.

The tutorials series Tutorial: Java Control, Tutorial: Web Service, and Tutorial: Page Flow gives step by step instructions for building the TutorialsApp from scratch.

Concepts Demonstrated by this Sample

- ***General concepts:***
    - ♦ Integration of Java control, web service, and page flow technologies
    - ♦ Use of database control
    - ♦ Use of web service control
    - ♦ Use of JMS control
    - ♦ Use of EJB control
    - ♦ XMLBeans
    - ♦ HTTPS and basic authentication
    - ♦ Role−based security
- ***Java control concepts:***
    - ♦ Asynchronous communication
    - ♦ Custom Java control properties
    - ♦ Compiling Java controls into distributable JAR files
- ***Web service concepts:***
    - ♦ Saving state with conversations / advanced asynchronous communication
    - ♦ XML maps and XQuery
    - ♦ Web Service Security (WSSE Security)
- ***Page Flow concepts:***
    - ♦ Separation of data presentation and data proccessing logic
    - ♦ Controlling user navigation with Action methods
    - ♦ Controlling data processing with Action methods and controls
    - ♦ Databinding for data presentation
    - ♦ Databinding for submission of user data input

Location of Sample Files

This sample is located at:

BEA_HOME\weblogic81\samples\platform\TutorialsApp\TutorialsApp.work

To Run the Sample

1. Start WebLogic Server in the workshop domain.

   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

2. You can run either the web service tier or the page flow tier of the application.

   " ***To run the web service tier***, either open TutorialsApp/FirstWebService/investigateJWS/Investigate.jws in WebLogic Workshop and click the Start button or enter http://localhost:7001/FirstWebService/investigateJWS/Investigate.jws in the address bar of a web browser. If WebLogic Server is running in the workshop domain on this machine, you may click here to run the sample.

   Navigate to the ***Test Form*** tab, if necessary.

   Enter a value into the taxID field and click the ***requestCreditReport*** button.

   Click ***Refresh*** until the method ***onCreditReportDone*** appear in the ***Message Log***.

   Click ***onCreditReportDone*** and view the credit−worthiness report.

   " ***To run the page flow tier***, either open TutorialsApp/FirstPageFlow/investigateJPF/InvestigateJPFController.jpf in WebLogic Workshop and click the Start button or enter http://localhost:7001/FirstPageFlow/investigateJPF/InvestigateJPFController.jpf into a web browser. If WebLogic Server is running in the workshop domain on this machine, you may click here to run the sample.

   Login as "Investigator" / "password".

   Enter on the of the following 9−digit taxID values "111111111", "222222222", "333333333", "444444444", "555555555", "123456789" and click ***Submit***.

   The credit−worthiness report will be displayed.

Related Topics

Tutorial: Java Control

Tutorial: Web Service

Tutorial: Page Flow

EJB Tutorial Sample

The EJB Tutorial Sample demonstrates a complete WebLogic Workshop application that integrates EJBs, EJB controls, and page flow technologies.

The EJB Tutorial Sample application allows a user to enter information about bands and a band's recordings, and obtain user rating information on recordings. The application contains the following components:

- A *Band* entity bean, which processes information about bands.
- A *Recording* entity bean, which processes information about recordings.
- A *Music* session bean, which references the Band entity bean, and provides the interface to the page flow application.
- A *Statistics* message−driven bean, which processes messages sent by the *Music* bean, and generates user ratings for recordings.
- A page flow application that is used as the client application to enter new bands and recordings, and to review information about bands, recordings, and recording ratings. This page flow locates and references the Music bean via an EJB control.

The Tutorial: Enterprise JavaBeans gives step by step instructions for building this application from scratch.

Note. The Enterprise JavaBean tutorial uses additional client applications to test intermediate stages of EJB construction. In addition, different database tables are used to store information and other JNDI names are used to avoid naming conflicts. However, in all other aspects the EJB Tutorial sample and EJB Project tutorial applications are identical.

Concepts Demonstrated by this Sample

- Use of stateless session beans
- Use of CMP entity beans
- Entity relations
- Use of message−driven beans
- Use of EJB controls
- Use of Page flows

Location of Sample Files

This sample is located at:

BEA_HOME\weblogic81\samples\platform\EJBTutorial_Sample\EJBTutorial_Sample.work

To Run the Sample

1. Start WebLogic Server in the workshop domain.

    - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1−−>Examples−−>WebLogic Workshop Examples−−>Start Workshop with Sample Applications***.
    - On Linux or Solaris systems, run:

BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

2. Open EJBTutorial_Sample/EJBTutorial_WebApp/Controller.jpf and click the Start button.

Related Topics

Tutorial: Enterprise JavaBeans

Java Control Samples

The samples described in this section illustrate Java controls.

Topics Included in This Section

BufferJCImpl.jcs Sample

A Java control illustrating how you can add message buffering to control operations to handle heavy loads.

HelloImpl.jcs Sample

This control offers a simple "Hello, World!" response. It illustrates a Java control at its most basic.

POVerifyImpl.jcs Sample

The POVerify control is designed to accept the details of an item associated with a purchase order, and to insert the details into two database tables. It exposes a "checkInventory" attribute through which a developer using the control can request this feature.

VerifyFundsImpl.jcs Sample (Local Control)

A Java control demonstrating how you can use multiple Java controls to coordinate potentially complex business logic.

VerifyFundsImpl.jcs Sample (Java Control Project)

A Java control illustrating how you can package Java controls into a control project.

ControlTest.jws Sample

A web service that is a test container for trying out the Java control sample in the JavaControlProject.

LocalControlTest.jws Sample

A web service that is a test container for trying out the Java control samples.

Related Topics

Getting Started with Java Controls

BufferJCImpl.jcs Sample

A Java control illustrating how you can add message buffering to control operations to handle heavy loads. With a buffered method, requests to the method are queued and processed as the control has resources available.

Use the LocalControlTest.jws web service to run this control.

Concepts Demonstrated by this Sample

- Message buffering for Java controls

Location of Sample Files

This sample is located in the localControls/asynch folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\localControls\asynch\BufferJCImpl.j

To Run the Sample

Note that you run this sample by running its test container.

- Start WebLogic Server in the appropriate domain.
- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the test service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/localControls/LocalControlTest.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the test container.
- Navigate to the ***Test Form*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

Getting Started with Java Controls

HelloImpl.jcs Sample

This control offers a simple "Hello, World!" response. It illustrates a Java control at its most basic.

Use the LocalControlTest.jws web service to run this control.

Concepts Demonstrated by this Sample

- Java control fundamentals

Location of Sample Files

This sample is located in the localControls/basics folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\localControls\basics\HelloImpl.jcs

To Run the Sample

Note that you run this sample by running its test container.

- Start WebLogic Server in the appropriate domain.
- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the test service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/localControls/LocalControlTest.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the test web service.
- Navigate to the ***Test Form*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

Getting Started with Java Controls

POVerifyImpl.jcs Sample

The POVerify control is designed to accept the details of an item associated with a purchase order, and to insert the details into two database tables. The control has the capability to make the database submission conditional based on whether there is enough inventory for the item. It exposes a "checkInventory" attribute through which a developer using the control can request this feature.

Use the LocalControlTest.jws web service to run this control.

Concepts Demonstrated by this Sample

- Using multiple nested controls in a Java control
- Java control properties

Location of Sample Files

This sample is located in the localControls/nestedControls folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\localControls\tags\POVerifyImpl.jcs

To Run the Sample

Note that you run this sample by running its test container.

- Start WebLogic Server in the appropriate domain.

- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the test service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/localControls/LocalControlTest.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the test web service.
- Navigate to the ***Test Form*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

Getting Started with Java Controls

VerifyFundsImpl.jcs Sample (Local Control)

A Java control demonstrating how you can use multiple Java controls to coordinate potentially complex business logic. This control coordinates a purchase order transaction that requires a database update and an EJB query. In addition to the transaction semantics of this control's logic, keep in mind that the work of this control is itself part of a transaction that will be rolled back if something goes wrong while the control is executing.

Use the LocalControlTest.jws web service to run this control.

Concepts Demonstrated by this Sample

- Using multiple nested controls in a Java control
- Implicit transactions in a Java control

Location of Sample Files

This sample is located in the localControls/nestedControls folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\localControls\nestedControls\VerifyF

To Run the Sample

Note that you run this sample by running its test container.

- Start WebLogic Server in the appropriate domain.
- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the test service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/localControls/LocalControlTest.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the test web service.
- Navigate to the ***Test Form*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

Getting Started with Java Controls

VerifyFundsImpl.jcs Sample (Java Control Project)

A Java control illustrating how you can package Java controls into a control project.

While this control is identical to a control in the localControls folder of the WebServices sample project, its placement in a control project illustrates how you can package a control in a control project for greater portability. You can build this control's project to generate a JavaControlProject.jar file. That file can be used in multiple other applications. The JAR file will be placed in the Libraries folder.

This control coordinates a purchase order transaction that requires a database update and an EJB query. In addition to the transaction semantics of this control's logic, keep in mind that the work of this control is itself part of a transaction that will be rolled back if something goes wrong while the control is executing.

Use the ControlTest.jws web service in the WebServices/controlProjectTest folder to run this control.

Concepts Demonstrated by this Sample

- Packaging a Java control in a control project
- Using multiple nested controls in a Java control
- Implicit transactions in a Java control

Location of Sample Files

This sample is located in the verifyFunds folder of the JavaControlProject in the SamplesApp WebLogic Workshop application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\JavaControlProject\verifyFunds\VerifyFundsImpl.j

To Run the Sample

Note that you run this sample by running its test container.

- Start WebLogic Server in the appropriate domain.
- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the test service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/controlProjectTest/ControlTest.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the test web service.
- Navigate to the ***Test Form*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

ControlTest.jws Sample

A web service that is a test container for trying out the Java control sample in the JavaControlProject. This web service merely provides a way to run the following local Java control samples:

- VerifyFundsImpl.jcs (Java Control Project)

Concepts Demonstrated by this Sample

- Running a Java control

Location of Sample Files

This sample is located in the controlProjectTest folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\controlProjectTest\ControlTest.jws

To Run the Sample

- Start WebLogic Server in the appropriate domain.
- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/controlProjectTest/ControlTest.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
- Navigate to the ***Test Form*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

Getting Started with Java Controls

LocalControlTest.jws Sample

A web service that is a test container for trying out the Java control samples. This web service merely provides a way to run the following local Java control samples:

- BufferJCImpl.jcs
- HelloImpl.jcs
- VerifyFundsImpl.jcs (Local Control)
- POVerifyImpl.jcs

Concepts Demonstrated by this Sample

- Running a Java control

Location of Sample Files

This sample is located in the localControls folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\localControls\LocalControlTest.jws

To Run the Sample

- Start WebLogic Server in the appropriate domain.
- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/localControls/LocalControlTest.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
- Navigate to the ***Test Form*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

Getting Started with Java Controls

Web Application Samples

The following samples demonstrate how to use Page Flow and JSP technologies to build dynamic web applications. These samples also highlight how to use WebLogic Workshop to as a web application development tool.

Topics Included in This Section

Navigation

These samples demonstrate how to set up navigation for a Page Flow web application.

Navigating Between Page A and Page B Sample

Define Navigation in the Controller file, not the JSPs Sample

Dropdown Navigation Sample

Nesting Page Flows Sample

Handling Exceptions Sample

Handling Data

These samples demonstrate how to collect, process and display data in a Page Flow web application.

Simple Data Submission Sample

Using a Wizard to Collect User Infomation Sample

Data Binding Contexts Sample

Dynamically Populating the Options in a Dropdown Sample

Handling Binary Data Sample

Handling Complex Java Objects Sample

Multiple Forms on a Page Sample

Handling XMLBean Data Sample

Traditional JSP Web Applications vs. Page Flow Web Applications Sample

Using Controls

These samples demonstrate the use of controls to access backend resources.

Calling a Database Sample

Calling a Synchronous Web Service Sample

Calling an Asynchronous Web Service Sample

Security

These samples demonstrate login and role−based security in a Page Flow web application.

Form−based Login Sample

"Shopping Cart" Login Sample

Role−based Security Sample

Struts

These samples show how to integrate your Page Flow with existing Struts applications and how to use Struts features in your Page Flow.

Struts Interoperation Sample

Struts Merging Sample

Validating User Data with Struts Sample

Tag Samples

These samples demonstrate the syntax of the different tags in the <netui...> JSP tag library.

<netui:...> Tags

<netui:anchor> Tag Sample

<netui:bindingUpdateErrors> Tag Sample

<netui:button> Tag Sample

<netui:checkBox> Tag Sample

<netui:checkBoxGroup> Tag Sample

<netui:checkBoxOption> Tag Sample

<netui:content> Tag Sample

<netui:exceptions> Tag Sample

<netui:form> Tag Sample

<netui:formatDate>, <netui:formatNumber>, <netui:formatString> Tags Sample

<netui−data:...> Tags

<netui−template:...> Tags

<netui−template:..> Tags Sample

Custom Tags

Custom Tags Sample

Navigating Between Page A and Page B Sample

A page flow that demonstrates basic Page Flow navigation using action methods in a Controller file.

Concepts Demonstrated by this Sample

- Using action methods to control navigation

Location of Sample Files

In the ***SamplesApp***, the sample is located at:

SamplesApp/WebApp/navigation/simpleNavigation/SimpleNavigationController.jpf

On the local file system, the sample is located at:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebApp\navigation\simpleNavigation\SimpleNavi

To Run the Sample

1. Start WebLogic Server in the Workshop domain.
    - ♦ On Microsoft Windows systems, run

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.cmd
    - ♦ On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and clicking the Start button or by entering http://localhost:7001/WebApp/navigation/simpleNavigation/SimpleNavigationController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.
3. Click ***Link to page_B.jsp***.

Related Topics

How Do I: Define an Action That Forwards Users to Another Page?

Define Navigation in the Controller file, not the JSPs Sample

A Page Flow that demonstrates how to control navigation from the action methods in a Controller file, instead of hard−coding navigation in the presentation−layer JSP files.

Concepts Demonstrated by this Sample

- Controlling navigation with action methods

Location of Sample Files

In the ***SamplesApp***, the sample is located at:

SamplesApp/WebApp/navigation/controllerFileNavigation/controllerFileNavigationController.jpf

On the local file system, the sample is located at:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebApp\navigation\controllerFileNavigation\contr

To Run the Sample

1. Start WebLogic Server in the Workshop domain.
    - ♦ On Microsoft Windows systems, run

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.cmd
    - ♦ On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and clicking the Start button or by entering http://localhost:7001/WebApp/navigation/controllerFileNavigation/controllerFileNavigationController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.
3. Click ***Let's Go!***.

Related Topics

How Do I: Define an Action That Forwards Users to Another Page?

Dropdown Navigation

A Page Flow that controls navigation through submitting different parameters to an action method. Databound JSP tags are used to provide navigation choices to users.

Concepts Demonstrated by this Sample

- Databound JSP tags
- Action methods
- Action Forms

Location of Sample Files

In the ***SamplesApp***, the sample is located at:

SamplesApp/WebApp/navigation/selectTagNavigation/SelectTagNavigationController.jpf

On the local file system, the sample is located at:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebApp\navigation\selectTagNavigation\SelectTa

To Run the Sample

1. Start WebLogic Server in the Workshop domain.
    - ♦ On Microsoft Windows systems, run

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.cmd
    - ♦ On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and clicking the Start button or by entering http://localhost:7001/WebApp/navigation/selectTagNavigation/SelectTagNavigationController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.
3. Select either ***Convertibles***, ***Specialty***, or ***Vintage***, and click ***Go!***.

Nesting Page Flows Sample

This Page Flow sample shows the nesting relationship between two Page Flows: the calling Page Flow, and the nested Page Flow.

Concepts Demonstrated by this Sample

- Using the @jpf:controller annotation's nested="true" attribute in the nested Page Flow.
- Passing flow control between nesting and nested Page Flows.
- Passing data between nesting and nested Page Flows
- Using a form bean to pass data from the nested Page Flow back to the calling Page Flow.

Location of Sample Files

In SamplesApp, the sample is in the following location:

> SamplesApp/WebApp/navigation/nesting/nestingPageFlow/NestingPageFlowController.jpf
> (The nesting Page Flow)

In the example, when the user creates a user profile, control is passed to a nested Page Flow that is located at:

> SamplesApp/WebApp/navigation/nesting/nestedPageFlow/NestedPageFlowController.jpf
> (The nested Page Flow)

On the local file system, the samples are located at:

> <BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\navigation\nesting\nestingPageFlow\N
>
> <BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\navigation\nesting\nestedPageFlow\Ne

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp\navigation\nesting\nestingPageFlow\NestingPageFlowController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

@jpf:controller Annotation

Handling Exceptions Sample

This Page Flow shows how to control navigation after an exception has occured. The @jpf:catch annotation is used to define a central location (showError.jsp) where error information is displayed.

Concepts Demonstrated by this Sample

- Using @jpf:catch to forward users to an error page

Location of Sample Files

In the ***SamplesApp***, the samples are located at:

SamplesApp/WebApp/navigation/handleException/HandleExceptionController.jpf

On the local file system, the samples are located at:
[BEA_HOME]\weblogic81\samples\workshop\SamplesApp\WebApp\navigation\handleException\HandleExc

To Run the Sample

1. Start WebLogic Server in the Workshop domain.
   - ◊ On Microsoft Windows systems, run

     [BEA_HOME]/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ◊ On Linux or Solaris systems, run:

     [BEA_HOME]/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the page flow either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/navigation/handleException/HandleExceptionController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

@jpf:catch Annotation

Using a Wizard to Collect User Infomation Sample

A Page Flow that uses member variables in the Controller file to preserve user state through different steps of a wizard.

Concepts Demonstrated by this Sample

- Using member variables to preserve user state

Location of Sample Files

In the ***SamplesApp***, the sample is located at:

SamplesApp/WebApp/handlingData/dataFlow/dataFlowController.jpf

On the local file system, the sample is located at:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebApp\handlingData\dataFlow\dataFlowControll

To Run the Sample

1. Start WebLogic Server in the Workshop domain.
    - On Microsoft Windows systems, run

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.cmd
    - On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and clicking the Start button or by entering http://localhost:7001/WebApp/handlingData/dataFlow/dataFlowController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

Databinding Contexts Sample

This page flow shows how to use the different data binding contexts.

Concepts Demonstrated by this Sample

- Data binding syntax: {pageFlow...}, {actionForm}, etc.

Location of Sample Files

In the ***SamplesApp***, the sample is located at:

SamplesApp/WebApp/handlingData/databinding/databindingController.jpf

On the local file system, the sample is located at:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebApp\handlingData\databinding\databindingCo

To Run the Sample

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and clicking the Start button or by entering http://localhost:7001/WebApp/handlingData/databinding/databindingController.jpf in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.

Related Topics

Dynamically Populating the Options in a Dropdown Sample

Demonstrates how to make a dynamic dropdown list by databinding a database field to a <netui:select> tag.

Concepts Demonstrated by this Sample

- Databinding
- Use of the <netui−data:select> tag.

Location of Sample Files

In the ***SamplesApp***, the sample is located at:

SamplesApp/WebApp/handlingData/dropdown

On the local file system, the sample is located at:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebApp\handlingData\dropdown

How to Run the Sample

1. Start WebLogic Server in the Workshop domain.
    - ♦ On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop***−−>***Launch Examples Server***.
    - ♦ On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/workshop/startWebLogic.sh
2. Launch the page flow either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/handlingData/dropdown/dropdownController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:select> Tag

<netui:selectOption> Tag

Handling Binary Data Sample

This page flow shows how to handle binary image data in a Page Flow and in a database control.

Concepts Demonstrated by this Sample

- displaying BLOB database types on JSP pages
- uploading image files from JSP pages and storing them as BLOB types in databases
- using Servlets from JSP pages

Location of Sample Files

In the ***SamplesApp***, the sample is located at:

SamplesApp/WebApp/handlingData/binaryFlow/BinaryFlowController.jpf

On the local file system, the sample is located at:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebApp\handlingData\binaryFlow\BinaryFlowCo

To Run the Sample

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and clicking the Start button or by entering http://localhost:7001/WebApp/handlingData/binaryFlow/BinaryFlowController.jpf in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Upload image files (GIF, JPG, BMP) using the Insert button.

Related Topics

Handling Images and Binary Data in Page Flows

Handling Complex Java Objects Sample

This sample shows you how to pre−populate and submit a user input form containing a complex (non−primitive) Java object.

Concepts Demonstrated by this Sample

- pre−populating user input forms
- submitting data with Form Beans
- using the reset() method when submitting complex Java objects

Location of Sample Files

In the ***SamplesApp***, the sample is located at:

SamplesApp/WebApp/handlingData/complexFormBean/complexFormBeanController.jpf

On the local file system, the sample is located at:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebApp\handlingData\complexFormBean\comple

How to Run the Sample

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and clicking the Start button or by entering http://localhost:7001/WebApp/handlingData/complexFormBean/complexFormBeanController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

Multiple Forms on a Page Sample

This page flow sample shows how to handle the submission of multiple forms from a single JSP page.

Concepts Demonstrated by this Sample

- Pre−populating user input forms with data binding expressions
- Handling user submitted data with Action methods

Location of Sample Files

In the ***SamplesApp***, the samples are located at:

SamplesApp/WebApp/handlingData/multipleForms/multipleFormsController.jpf

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\handlingData\multipleForms\multipleF

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
    - ♦ On Microsoft Windows systems, run

      <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
    - ♦ On Linux or Solaris systems, run:

      <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/handlingData/multipleForms/multipleFormsController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

Handling XmlBean Data Sample

This page flow sample shows how to use XMLBean types within a page flow. The sample contains a database control file that returns an XMLBean type (ItemsRowSetDocument). This XMLBean data type is used in the page flow's Form Bean and it is displayed on a JSP page.

Concepts Demonstrated by this Sample

- Using an XMLBean data type as a field in a Form Bean.
- Data binding to an XMLBean type in a JSP page.

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/handlingData/xmlBeans/XmlBeansController.jpf

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\handlingData\xmlBeans\XmlBeansCo

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
    - ♦ On Microsoft Windows systems, run

      <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
    - ♦ On Linux or Solaris systems, run:

      <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the page flow either by opening SamplesApp/WebApp/xmlBeans/XmlBeansController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/handlingData/xmlBeans/XmlBeansController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

Traditional JSP Web Applications vs. Page Flow Web Applications Sample

This page flow shows how Page Flows differ from traditional JSP web applications.

Concepts Demonstrated by this Sample

- Placing data processing logic in the Controller file, not in the JSP pages
- Data binding to data in the Controller file

Location of Sample Files

In the ***SamplesApp***, the sample is located at:

SamplesApp/WebApp/handlingData/traditional_vs_pageFlow_webApp/traditional_vs_pageFlow_webAppCon

On the local file system, the sample is located at:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebApp\handlingData\traditional_vs_pageFlow_w

To Run the Sample

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and clicking the Start button or by entering http://localhost:7001/WebApp/handlingData/traditional_vs_pageFlow_webApp/traditional_vs_pageFlow_web in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.

Related Topics

Calling a Database Sample

A Page Flow that calls a database control.

Concepts Demonstrated by this Sample

- Calling database controls from the Controller file
- Displaying database data on a JSP page

Location of Sample Files

In the ***SamplesApp***, the sample is located at:

SamplesApp/WebApp/controls/database/databaseController.jpf

On the local file system, the sample is located at:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebApp\controls\database\databaseController.jpf

To Run the Sample

1. Start WebLogic Server in the Workshop domain.
    - On Microsoft Windows systems, run

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.cmd
    - On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and clicking the Start button or by entering http://localhost:7001/WebApp/controls/database/databaseController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

Calling a Synchronous Web Service Sample

A Page Flow that makes a invokes a synchonous method of a web service.

Concepts Demonstrated by this Sample

- Use of a web service control
- Synchronous web services

Location of Sample Files

In the ***SamplesApp***, the sample is located at:

SamplesApp/WebApp/controls/webservice/helloworld/HelloWorldController.jpf

On the local file system, the sample is located at:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebApp\controls\webservice\helloworld\HelloWo

To Run the Sample

1. Start WebLogic Server in the Workshop domain.
    - ♦ On Microsoft Windows systems, run

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.cmd
    - ♦ On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and clicking the Start button or by entering http://localhost:7001/WebApp/controls/webservice/helloworld/HelloWorldController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.
3. Click ***Get message from the HelloWorld web service***.

Related Topics

Calling an Asynchronous Web Service Sample

A Page Flow that communicates with a web service through a polling interface.

Concepts Demonstrated by this Sample

- Use of a Web Service control
- Polling an asynchronous web service

Location of Sample Files

In the ***SamplesApp***, the sample is located at:

SamplesApp/WebApp/controls/webservice/polling/PollingController.jpf

On the local file system, the sample is located at:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebApp\controls\webservice\polling\PollingContr

To Run the Sample

1. Start WebLogic Server in the Workshop domain.
    - ♦ On Microsoft Windows systems, run

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.cmd
    - ♦ On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/controls/webservice/polling/PollingController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.
3. Click ***Request Message***.

Related Topics

Using Polling as an Alternative to Callbacks

Form−based Login Sample

This sample uses Form−based authentication to login users to the page flow. This sort of authentication should only be used before a user has entered the page flow. If a user has already entered the page flow, you should use the login stategy described in "Shopping Cart" Login.

Concepts Demonstrated by this Sample

- Linking to a nested Page Flow
- Receiving parameters from a nested Page Flow
- User authentication (username/password authentication)
- Using the @jpf:action login−required="true" annotation
- Using the @jpf:forward return−to... annotation

Location of Sample Files

In the ***SamplesApp***, the samples are located at:

SamplesApp/WebApp/security/formBasedLogin/formBasedLoginFlow/formBasedLoginFlowController.jpf

On the local file system, the samples are located at:
[BEA_HOME]\weblogic81\samples\workshop\SamplesApp\WebApp\security\formBasedLogin\formBasedLo

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     [BEA_HOME]/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     [BEA_HOME]/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the page flows either by opening SamplesApp/WebApp/security/formBasedLogin/formBasedLoginFlow/formBasedLoginFlowController.jpf in WebLogic Workshop and clicking the Start button or by entering http://localhost:7001/WebApp/security/formBasedLogin/formBasedLoginFlow/formBasedLoginFlowControll in the address bar of your browser.

   If WebLogic Server is running in the Workshop domain on your machine, you may click here to run the Form Based sample.

Related Topics

"Shopping Cart" Login Sample

Role−based Security Sample

"Shopping Cart" Login Sample

The sample uses a nested Page Flow to log users into the nesting Page Flow. When unauthorized users try to invoke a particular method in the nesting Page Flow, they are sent to the nested Page Flow for a username password challenge.

This strategy is called a "shopping cart" login, because unauthorized users are allowed to navigate part of the nesting page flow (e.g., the product catalog), but when they try to purchase their selections, they are required to login.

Concepts Demonstrated by this Sample

- Linking to a nested Page Flow
- Receiving parameters from a nested Page Flow
- User authentication (username/password authentication)
- Using the @jpf:action login−required="true" annotation
- Using the @jpf:forward return−to... annotation

Location of Sample Files

In the ***SamplesApp***, the samples are located at:

SamplesApp/WebApp/security/shoppingCartLogin/shopping/shoppingController.jpf

On the local file system, the samples are located at:
[BEA_HOME]\weblogic81\samples\workshop\SamplesApp\WebApp\security\shoppingCartLogin\shopping\sh

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     [BEA_HOME]/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     [BEA_HOME]/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the page flows either by opening SamplesApp/WebApp/security/shoppingCartLogin/shopping/shoppingController.jpf in WebLogic Workshop and clicking the Start button or by entering http://localhost:7001/WebApp/security/shoppingCartLogin/shopping/shoppingController.jpf in the address bar of your browser.

   If WebLogic Server is running in the Workshop domain on your machine, you may click here to run the Shopping Cart sample.
3. Enter the username/password pair "weblogic"/"weblogic" to login.

Related Topics

Form−based Login Sample

Role−based Security Sample

Simple Data Submission Sample

A page flow that demonstrates the data submission process using HTML input elements data bound to a Form Bean instance.

Concepts Demonstrated by this Sample

- Data binding with the <netui:...> tag library
- Submitting data to a Controller file via Form Bean instance (request−scoped)

Location of Sample Files

In the ***SamplesApp***, the sample is located at:

SamplesApp/WebApp/handlingData/simpleSubmit/SimpleSubmitController.jpf

On the local file system, the sample is located at:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebApp\handlingData\simpleSubmit\SimpleSubm

To Run the Sample

1. Start WebLogic Server in the Workshop domain.
    - ♦ On Microsoft Windows systems, run

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.cmd
    - ♦ On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and clicking the Start button or by entering http://localhost:7001/WebApp/handlingData/simpleSubmit/SimpleSubmitController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.
3. In the Workshop Test Browser, enter text in the name field and click SubmitName.

Related Topics

Introduction to Building Web Applications with Page Flows

Role−based Security Sample

This sample shows how to filter access to a Page Flow based on the user's role membership.

Concepts Demonstrated by this Sample

- Linking to a nested Page Flow
- Receiving parameters from a nested Page Flow
- User authentication (username/password authentication)
- Using the @jpf:action login−required="true" annotation
- Using the @jpf:forward return−to... annotation

Location of Sample Files

In the ***SamplesApp***, the samples are located at:

SamplesApp/WebApp/security/roleBasedSecurity/roleBasedSecurityController.jpf

On the local file system, the samples are located at:
[BEA_HOME]\weblogic81\samples\workshop\SamplesApp\WebApp\security\roleBasedSecurity\roleBasedSe

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     [BEA_HOME]/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     [BEA_HOME]/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the page flows either by opening security/roleBasedSecurity/roleBasedSecurityController.jpf in WebLogic Workshop and clicking the Start button or by entering http://localhost:7001/WebApp/security/roleBasedSecurity/roleBasedSecurityController.jpf in the address bar of your browser.

   If WebLogic Server is running in the Workshop domain on your machine, you may click here to run the Form Based sample.

Related Topics

Form−based Login Sample

"Shopping Cart" Login Sample

Struts Interoperation Sample

Demonstrates the interoperability between page flows and Struts. The strutsInteropController page flow instantiates a form bean, JpfFormBean, then passes this shared form to a Struts module named strutsModule. The Struts module alters the contents of the form and passes it back to the page flow. Because page flow forms derive from Struts forms, this sharing of the form bean is possible.

Concepts Demonstrated by this Sample

- The Struts Merge feature
- Interchangeable support for Struts modules and page flow controller classes working together in the same web project.

Location of Sample Files

In the ***SamplesApp***, this sample's files are located in several directories:

Location	***Contents Include...***
SamplesApp/WebApp/struts/strutsInterop	The page flow directory, which includes JSP pages that are part of the page flow, and the shared form bean
SamplesApp/WebApp/struts/strutsModule	A JSP that is part of the Struts module
SamplesApp/WebApp/WEB−INF/src/strutsModule	Two Struts actions classes
SamplesApp/WebApp/WEB−INF	The strutsModule XML, and the project's web.xml where we registered the Struts module

For details about these files, see Interoperating With Struts and Page Flows.

On the local file system, the sample directories listed above are under the WebApp project directory that starts at:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebApp\...

How to Run the Sample

1. Start WebLogic Server in the Workshop domain.

   - ♦ On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1−−>Examples−−>WebLogic Workshop−−>Launch Examples Server***.
   - ♦ On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/workshop/startWebLogic.sh
2. Launch the page flow either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/struts/strutsInterop/strutsInteropController.jpf in the

address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

Interoperating With Struts and Page Flows

Merging Struts Artifacts Into Page Flows

@jpf:controller Annotation

Getting Started with Page Flows

Tutorial: Page Flow

Struts Merge Sample

Demonstrates how to merge Struts artifacts, such as actions, into the configuration XML that is generated for a page flow.

Note: For information about a related sample, see the strutsInteropController Sample.

Concepts Demonstrated by this Sample

- Use of the @jpf:controller struts−merge annotation.
- Struts configuration XML

Location of Sample Files

In the ***SamplesApp***, the sample is located at:

SamplesApp/WebApp/struts/strutsMerge

On the local file system, the sample is located at:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebApp\struts\strutsMerge

How to Run the Sample

1. Start WebLogic Server in the Workshop domain.
    - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop***−−>***Launch Examples Server***.
    - On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/workshop/startWebLogic.sh
2. Launch the page flow either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/struts/strutsMerge/strutsMergeController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

Merging Struts Artifacts Into Page Flows

strutsInteropController Sample

Interoperating With Struts and Page Flows

@jpf:controller Annotation

Getting Started with Page Flows

Tutorial: Page Flow

Form Validation Sample

This page flow sample shows several ways to perform validation of data entered on input forms.

Concepts Demonstrated by this Sample

- Client−side validation using JavaScript and <netui*:*> tag event attributes like onBlur and onClick.
- Server−side validation using Java to implement the validate method in the form bean.
- Server−side validation using the Struts ValidatorPlugIn to do XML−based validation.
- The use of the @jpf:validation−error−forward annotation.
- The use of the @jpf:message−resources annotation and a message properties file.

Location of Sample Files

In the ***SamplesApp***, the validation samples start in the following location:

SamplesApp/WebApp/struts/validation/Controller.jpf

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\struts\validation\Controller.jpf

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
    - ♦ On Microsoft Windows systems, run

      <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
    - ♦ On Linux or Solaris systems, run:

      <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/struts/validation/Controller.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

Validating User Input

@jpf:validation−error−forward Annotation

@jpf:message−resources Annotation

Merging Struts Artifacts Into Page Flows

<netui:anchor> Tag Sample

This page flow sample shows how to use the <netui:anchor> tag.

Concepts Demonstrated by this Sample

- Linking to a JSP page
- Invoking an action method on the Controller file
- Submitting form data to the Controller file
- Submitting form data to the Controller file using a custom JavaScript function

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/anchor/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\anchor\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/anchor/AnchorController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/anchor/AnchorController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:anchor> Tag

<netui:bindingUpdateErrors> Tag Sample

This page flow sample shows how to use the <netui:bindingUpdateErrors> tag.

Concepts Demonstrated by this Sample

- Displaying errors inline on a JSP page

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/bindingUpdateErrors/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\bindingUpdateErrors

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/bindingUpdateErrors/BindingUpdateErrorsController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/bindingUpdateErrors/BindingUpdateErrorsController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:bindingUpdateErrors> Tag

<netui:button> Tag Sample

This page flow sample shows how to use the <netui:button> tag.

Concepts Demonstrated by this Sample

- Invoking an action method on the Controller file
- Submitting form data to the Controller file

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/button/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\button\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/button/ButtonController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/button/ButtonController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:button> Tag

<netui:checkBox> Tag Sample

This page flow sample shows how to use the <netui:checkBox> tag.

Concepts Demonstrated by this Sample

- Rendering an HTML check box with a <netui:checkBox> tag.
- Invoking an action method on the Controller file
- Submitting form data to the Controller file
- Setting default values in a form

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/checkBox/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\checkBox\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/checkBox/CheckBoxController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/checkBox/CheckBoxController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:checkBox> Tag

<netui:checkBoxGroup> Tag

<netui:checkBoxOption> Tag

<netui:checkBoxGroup> Tag Sample

This page flow sample shows how to use the <netui:checkBoxGroup> tag.

Concepts Demonstrated by this Sample

- Rendering a set of HTML checkboxes with a <netui:checkBoxGroup> tag
- Invoking an action method on the Controller file
- Submitting form data to the Controller file
- Setting default values in a form
- Dynamically populating checkbox options
- Data binding to objects in the Controller file

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/checkBoxGroup/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\checkBoxGroup\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/checkBoxGroup/CheckBoxGroupController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/checkBoxGroup/CheckBoxGroupController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:checkBox> Tag

<netui:checkBoxGroup> Tag

<netui:checkBoxOption> Tag

<netui:checkBoxOption> Tag Sample

This page flow sample shows how to use the <netui:checkBoxOption> tag.

Concepts Demonstrated by this Sample

- Rendering an HTML check box with a <netui:checkBoxOption> tag.
- Invoking an action method on the Controller file
- Submitting form data to the Controller file
- Setting default values in a form
- Data binding to objects in the Controller file

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/checkBoxOption/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\checkBoxOption\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/checkBoxOption/CheckBoxOptionController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/checkBoxOption/CheckBoxOptionController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:checkBox> Tag

<netui:checkBoxOption> Tag

<netui:checkBoxGroup> Tag

<netui:content> Tag Sample

This page flow sample shows how to use the <netui:content> tag.

Concepts Demonstrated by this Sample

- Databinding to member variables in the Controller file.

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/content/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\content\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
    - ♦ On Microsoft Windows systems, run

      <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
    - ♦ On Linux or Solaris systems, run:

      <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/content/ContentController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/content/ContentController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:content> Tag

<netui:exceptions> Tag Sample

This page flow sample shows how to use the <netui:exceptions> tag.

Concepts Demonstrated by this Sample

- Displaying errors inline with the <netui:exceptions> tag.
- Using the @jpf:catch annotation to handle exceptions.

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/exceptions/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\exceptions\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
    - ♦ On Microsoft Windows systems, run

      <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
    - ♦ On Linux or Solaris systems, run:

      <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/exceptions/ExceptionsController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/exceptions/ExceptionsController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:exceptions> Tag

@jpf:catch Annotation

<netui:form> Tag Sample

This page flow sample shows how to use the <netui:form> tag.

Concepts Demonstrated by this Sample

- Databinding to fields in a Form Bean
- Submitting file data to an action method in the Controller file
- Pre−populating form fields

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/form/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\form\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/form/FormController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/form/FormController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:form> Tag

<netui:format...> Tag Samples

This page flow sample shows how to use the <netui:formatDate>, <netui:formatNumber>, and <netui:formatString> tag.

Concepts Demonstrated by this Sample

- Databinding to getter methods in the Controller file
- Formatting Java data types into strings for display in a browser

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/formatTags/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\formatTags\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/formatTags/formatTagsController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/formatTags/formatTagsController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:formatDate> Tag

<netui:formatNumber> Tag

<netui:formatString> Tag

<netui:imageAnchor> Tag Sample

This page flow sample shows how to use the <netui:imageAnchor> tag.

Concepts Demonstrated by this Sample

- Linking to a page
- Invoking an action method on the Controller file
- Submitting data to an action method on the Controller file
- Submittind data to an action method via a custom JavaScript function

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/imageAnchor/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\imageAnchor\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/imageAnchor/ImageAnchorController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/imageAnchor/ImageAnchorController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:imageAnchor> Tag

<netui:label> Tag Sample

This page flow sample shows how to use the <netui:label> tag.

Concepts Demonstrated by this Sample

- Databinding to the javax.servlet.jsp.Request object
- Escaping text for output to a browser

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/label/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\label\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/label/LabelController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/label/LabelController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:label> Tag

<netui:parameter> Tag Sample

This sample shows how to use the <netui:parameter> tag.

Concepts Demonstrated by this Sample

- Adding parameters to the URL

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/parameter/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\parameter\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/parameter/index.jsp in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/parameter/index.jsp in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:parameter> Tag

<netui:parameterMap> Tag Sample

This page flow sample shows how to use the <netui:parameterMap> tag.

Concepts Demonstrated by this Sample

- Databinding to member variables in the Controller file
- Adding parameters to the URL

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/parameterMap/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\parameterMap\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
    - ♦ On Microsoft Windows systems, run

      <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
    - ♦ On Linux or Solaris systems, run:

      <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/parameterMap/ParameterMapController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/parameterMap/ParameterMapController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:parameterMap> Tag

<netui:radioButton...> Tag Samples

This page flow sample shows how to use the <netui:radioButtonGroup> and <netui:radioButtonOption> tags.

Concepts Demonstrated by this Sample

- Databinding to fields in a Form Bean
- Dynamically determining a group of radio button options
- Submitting data to an action method in the Controller file

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/radioButtons/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\radioButtons\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/radioButtons/RadioButtonsController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/radioButtons/RadioButtonsController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:radioButtonGroup> Tag

<netui:radioButtonOption> Tag

<netui:rewrite...> Tag Samples

This page flow sample shows how to use the <netui:rewriteName> and <netui:rewriteURL> tags.

Concepts Demonstrated by this Sample

- Rewriting names, ids, and other page elements to ensure uniqueness

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/rewrite/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\rewrite\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/rewrite/RewriteController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/rewrite/RewriteController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:rewriteName> Tag

<netui:rewriteURL> Tag

<netui:scriptContainer> Tag Sample

This page flow sample shows how to use the <netui:scriptContainer> tag.

Concepts Demonstrated by this Sample

- Organizing JavaScript on a page
- Using the JavaScript method getNetuiTagName() to retrieve id and name attributes

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/scriptContainer/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\scriptContainer\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/scriptContainer/ScriptContainerController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/scriptContainer/ScriptContainerController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:scriptContainer> Tag

<netui:select> Tag Sample

This page flow sample shows how to use the <netui:select> and <netui:selectOption> tags.

Concepts Demonstrated by this Sample

- Dynamically populating a selection list
- Enabling multiple selection

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/select/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\select\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/select/selectController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/select/selectController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:select> Tag

<netui:selectOption> Tag

<netui:textArea> Tag Sample

This page flow sample shows how to use the <netui:textArea> tag.

Concepts Demonstrated by this Sample

- Submitting data to an action method in the Controller file
- Data binding to a field in a Form Bean

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/textArea/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\textArea\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/testArea/TextAreaController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/testArea/TextAreaController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:textArea> Tag

<netui:textBox> Tag Sample

This page flow sample shows how to use the <netui:textBox> tag.

Concepts Demonstrated by this Sample

- Submitting data to an action method in the Controller file
- Data binding to a field in a Form Bean

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/textBox/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\textBox\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/textBox/TextBoxController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/textBox/TextBoxController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:textBox> Tag

<netui:tree> Tag Sample

This page flow sample shows how to use the <netui:tree> and <netui:node> tags.

Concepts Demonstrated by this Sample

- Using <netui:tree> and <netui:node> tags to create a navigation tree
- Using frame sets

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/tree/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\tree\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/tree/treeController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/tree/treeController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:tree> Tag

<netui:node> Tag

TreeNode Class

<netui:tree> Tag: Dynamic Tree Sample

This page flow sample shows how build a navigation tree dynamically by calling directly into the the <netui:tree> and <netui:node> APIs.

Concepts Demonstrated by this Sample

- Using the TreeNode object to create a dynamic navigation tree
- Using the <netui:tree> tag on a JSP page
- Data binding to a TreeNode object in a Controller file
- Using frame sets

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui/tree_dynamic/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui\tree_dynamic\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
    - ♦ On Microsoft Windows systems, run

      <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
    - ♦ On Linux or Solaris systems, run:

      <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui/tree_dynamic/treeController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui/tree_dynamic/treeController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui:tree> Tag

<netui:node> Tag

TreeNode Class

<netui−data:callControl> Tag Sample

This page flow sample shows how to use the <netui−data:callControl> tag.

Concepts Demonstrated by this Sample

- Calling a synchronous web service control
- Calling resources directly from a JSP page (unmediated by a Controller file)

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui_databinding/callControl/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui_databinding\callCon

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
    - ♦ On Microsoft Windows systems, run

      <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
    - ♦ On Linux or Solaris systems, run:

      <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui_databinding/callControl/callControl.jsp in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui_databinding/callControl/callControl.jsp in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui−data:callControl> Tag

<netui−data:callMethod> Tag Sample

This page flow sample shows how to use the <netui−data:callMethod> tag.

Concepts Demonstrated by this Sample

- Calling a method on the Controller file

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui_databinding/callMethod/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui_databinding\callMet

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui_databinding/callMethod/CallMethodController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui_databinding/callMethod/CallMethodController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui−data:callMethod> Tag

<netui−data:callPageFlow> Tag Sample

This page flow sample shows how to use the <netui−data:callPageFlow> tag.

Concepts Demonstrated by this Sample

- Calling a method on the Controller file

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui_databinding/callPageFlow/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui_databinding\callPage

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui_databinding/callPageFlow/CallPageFlowController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui_databinding/callPageFlow/CallPageFlowController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui−data:callPageFlow> Tag

<netui−data:cellRepeater> Tag Sample

This page flow sample shows how to use the <netui−data:cellRepeater> tag.

Concepts Demonstrated by this Sample

- Rendering a data set as an HTML table

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui_databinding/cellRepeater/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui_databinding\cellRep

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui_databinding/cellRepeater/CellRepeaterController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui_databinding/cellRepeater/CellRepeaterController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui−data:cellRepeater> Tag

<netui−data:choice> Tag Sample

This page flow sample shows how to use the <netui−data:choice> and <netui:choicMethod> tags.

Concepts Demonstrated by this Sample

- Conditionally rendering data sets using decision methods
- Rendering data sets as HTML tables.

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui_databinding/choice/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui_databinding\choice\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
    - ♦ On Microsoft Windows systems, run

      <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
    - ♦ On Linux or Solaris systems, run:

      <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui_databinding/choice/choiceController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui_databinding/choice/choiceController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui−data:choice> Tag

<netui−data:choiceMethod> Tag

<netui−data:choice> Tag Sample

This page flow sample shows how to use the <netui−data:choice> and <netui:choicMethod> tags.

Concepts Demonstrated by this Sample

- Conditionally rendering data sets using decision methods
- Rendering data sets as HTML tables.

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui_databinding/choice/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui_databinding\choice\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui_databinding/choice/choiceController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui_databinding/choice/choiceController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui−data:choice> Tag

<netui−data:choiceMethod> Tag

<netui−data:declarePageInput> Tag Sample

This page flow sample shows how to use the <netui−data:declarePageInput> tag.

Concepts Demonstrated by this Sample

- Using the {pageInput...} data binding context
- Clarifying dependencies of JSP on the Controller file
- Using the Data Palette tab
- Calling a database control

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui_databinding/declarePageInput/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui_databinding\declareI

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui_databinding/declarePageInput/PageInputController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui_databinding/declarePageInput/PageInputController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui−data:declarePageInput> Tag

<netui−data:getData> Tag Sample

This page flow sample shows how to use the <netui−data:getData> tag.

Concepts Demonstrated by this Sample

- Making Controller data available to JSP scriptlets

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui_databinding/getData/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui_databinding\getData

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui_databinding/getData/GetDataController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui_databinding/getData/GetDataController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui−data:getData> Tag

<netui−data:grid> Tag Sample

This page flow sample shows how to use the <netui−data:grid> tag.

Concepts Demonstrated by this Sample

- Using the <netui−data:grid> tag to render javax.sql.RowSet objects as HTML
- Calling a database control

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui_databinding/grid/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui_databinding\grid\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui_databinding/grid/GridController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui_databinding/grid/GridController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui−data:grid> Tag

<netui−data:anchorColumn> Tag

<netui−data:basicColumn> Tag

<netui−data:columns> Tag

<netui−data:grid> Tag

<netui−data:expressionColumn> Tag

<netui−data:imageColumn> Tag

<netui−data:message> Tag Sample

This page flow sample shows how to use the <netui−data:message> and <netui−data:messageArg> tags.

Concepts Demonstrated by this Sample

- Rending customizable message strings

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui_databinding/message/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui_databinding\message

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui_databinding/message/MessageController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui_databinding/message/MessageController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui−data:message> Tag

<netui−data:pad> Tag Sample

This page flow sample shows how to use the <netui−data:pad> tag.

Concepts Demonstrated by this Sample

- Rending data sets a HTML
- Setting maximum and minimum iterations over a data set

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui_databinding/pad/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui_databinding\pad\

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui_databinding/pad/PadController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui_databinding/pad/PadController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui−data:pad> Tag

<netui−data:repeater> Tag Sample

This page flow sample shows how to use the <netui−data:repeater> tag.

Concepts Demonstrated by this Sample

- Rending data sets as HTML
- Rendering XMLBeans as HTML
- Calling a database control

Location of Sample Files

In the ***SamplesApp***, the sample is in the following location:

SamplesApp/WebApp/tagSamples/netui_databinding/repeater/

On the local file system, the samples are located at:

<BEA_HOME>\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\netui_databinding\repeater

To Run the Samples

1. Start WebLogic Server in the Workshop domain.
   - ♦ On Microsoft Windows systems, run

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - ♦ On Linux or Solaris systems, run:

     <BEA_HOME>/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the Page Flow either by opening SamplesApp/WebApp/tagSamples/netui_databinding/repeater/repeaterController.jpf in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/netui_databinding/repeater/repeaterController.jpf in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

<netui−data:repeater> Tag

<netui−data:repeaterItem> Tag

<netui−data:repeaterHeader> Tag

<netui−data:repeaterFooter> Tag

Custom Tags Sample

The JSP page CustomTags.jsp demonstrates the use of three custom tags.

Concepts Demonstrated by this Sample

Use of a custom tag with no attributes or body text.

Use of a custom tag with an attribute but body text.

Use of a custom tag with an attribute and body text.

Location of Sample Files

In the ***SamplesApp***, the sample is located at:

SamplesApp/WebApp/tagSamples/custom_tags/CustomTags.jsp

On the local file system, the sample is located at:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebApp\tagSamples\custom_tags\CustomTags.jsp

To Run the Sample

1. Start WebLogic Server in the Workshop domain.
   - On Microsoft Windows systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.cmd
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the JSP page either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebApp/tagSamples/custom_tags/CustomTags.jsp in the address bar of your browser. If WebLogic Server is running in the Workshop domain on this machine, you may click here to run the sample.

Related Topics

Getting Started with Page Flows

Tutorial: Page Flow

Web Service Samples

The following sample web services are provided with WebLogic Workshop. They are located in the samples project, which is the default project when you start WebLogic Workshop for the first time.

Follow the links below for more detailed information on each sample, including instructions for running the sample.

Topics Included in This Section

AccountEJBClient.jws

A web service that demonstrates use of an EJB control (AccountEJBControl.jcx), which represents the AccountEJB Entity Bean and exposes its business interface to web services. This sample is *not* related to AccountPublish.jws and AccountSubscribe.jws.

AccountPublish.jws

A web service that demonstrates use of a JMS control (AccountPublishJMSControl.jcx) to publish a message to a JMS topic. This sample is paired with AccountSubscribe.jws, which subscribes to the JMS topic to which AccountPublish.jws publishes. This sample is *not* related to AccountEJBClient.jws.

AccountSubscribe.jws

A web service that demonstrates use of a JMScontrol (AccountPublishJMSControl.jcx) to subscribe to a JMS topic. This sample is paired with AccountPublish.jws, which publishes to the JMS topic to which AccountSubscribe.jws subscribes. This sample is *not* related to AccountEJBClient.jws.

AdvancedTimer.jws

A web service that demonstrates an asynchronous interface to a legacy system (LegacySystem.jws) that does not support asynchrony. It does so by "polling" the legacy system: calling it repeatedly to see if it's done. AdvancedTimer does the polling for the client, invoking a client callback if and when the legacy system responds.

AdvancedTimer uses the @jc:timer timeout= repeats−every= tag.

BasicAuthentication.jws

A web service that demonstrates how to secure a web service using basic authentication (username and password authentication).

Buffer.jws

A web service that demonstrates the @common:message−buffer tag to queue high−traffic requests. Uses a Timer control to delay sending a response back to the client, simulating waiting for a slow back−end service to respond to requests from this web service. The start method is buffered, allowing clients to continue processing immediately after submitting a request.

ClientCert

Demonstrates asynchronous communication between two web services, where the target web service requires a client certificate.

Conversation.jws

A web service that demonstrates use of the @jws:conversation tag to control the lifetime of a conversational instance of the web service and provide data persistence.

Conversation.jws implements a synchronous polling interface for the asynchronous HelloWorldAsync web service. A polling interface is necessary if you wish to serve clients that cannot accept asynchronous callbacks due to security arrangements such as firewalls.

Conversations also provide correlation, whereby requests from multiple simultaneous clients are tracked and responses are directed to the right client.

createUser.jws

Demonstrates the referencing of externally defined roles and principals.

CreditReport.jws

Demonstrates use of conversations and Web Service controls.

CustomerDBClient.jws

A sample that demonstrates use of a database control by managing customer records. CustomerDBClient is a client of CustomerDBControl.jcx. Together they demonstrate construction of a Database control and use of a Database control by a web service. CustomerDBControl.jcx demonstrates use of SQL's CREATE, DROP, INSERT, UPDATE, and SELECT statements.

CustomJMSClient.jws

A web service that demonstrates use of a JMS control defined in a JCX file. CustomJMSClient is a client of the CustomJMSControl.jcx JMS control.

DynamicSQL.jws

A web service that demonstrates how to pass dynamically generated SQL statements to a database control.

HelloWorld.jws

A very simple web service with a single synchronous method. Illustrates a WebLogic Workshop web service in its simplest form.

HelloWorldAsync.jws

A simple asynchronous web service that illustrates the use of a callback. Uses a Timer control to provide a delay that simulates waiting for a slow back−end service to do some work, then notifies the client of the result via a callback. HelloWorldAsync is used by the Conversation.jws sample, but is also a standalone web service.

HelloWorldSecureClient.jws

A web service that demonstrates some of the features of the declarative and the programmatic security models supported by WebLogic Workshop.

Investigate.jws

A web service that demonstrates exposing a Java control with an asynchronous web service. Also demonstrates the use XQuery maps, and WS−Security.

InvestigateSync.jws

A web service that demonstrates exposing a Java control with a synchronous web service.

JMS_XMLProtocol.jws

A web service that receives SOAP requests via a JMS queue.

LuckyNumberDBClient.jws

A sample that uses a database to store players and their lucky numbers. LuckyNumberDBClient uses the LuckyNumber.jws web service to generate random numbers in the range [1,20] inclusive, then checks for winners in a database and returns the number drawn and the list of winners, if any. The database is managed using the LuckyNumberDBControl.jcx Database control. LuckyNumberDBControl.jcx creates and manages a PLAYERS table in the database and implements queries against it. Demonstrates using SQL's CREATE, INSERT, SELECT and DROP statements in a database control.

QuoteClient.jws

Demonstrates the ability to modify a Web Service control's JCX file to change the Java signature of one or more methods while still adhering to the public contract of the called web service. QuoteServiceControl.jcx was originally generated from the QuoteService.jws web service. But one method in QuoteServiceControl.jcx has had a parameter removed and replaced with a hard−coded value in the accompanying XML map. Thus, the outgoing XML message is the same, but the Java signature visible to clients of the Web Service control is simplified.

ServiceControlFactory.jws

A web service that demonstrates creating a control factory with a Web Service control.

SimpleJMS.jws

A web service that demonstrates use of a JMS control declared locally in a web service's JWS file.

SimpleTimer.jws

A simple web service that demonstrates use of the Timer control. Uses the @jc:timer timeout= tag.

TraderEJBClient.jws

A web service that demonstrates use of the EJB control TraderEJBControl.jcx, which represents the TraderEJB Stateless Session Bean and exposes its business interface to web services.

VeriCheck.jws

A web service that demonstrates the declarative and the programmatic security models supported by WebLogic Workshop, including basic authentication to secure a web service, calling a secure service through a service control, and calling back a secure service through a callback interface.

WS−Security Callback Sample

The WS−Security Callback sample demonstrates asynchronous communication between two web services both of which are secured with WS−Security security.

WS−Security ReqResp Sample

The WS−Security ReqResp sample demonstrates synchronous communication between two web services both of which are secured with WS−Security security.

WS−Security UserToken Sample

The WS−Security UserToken sample demonstrates synchronous communication between two web services both of which are secured with WS−Security security, especially the use of username and password in WS−Security.

Related Topics

Tutorial: Web Services

AccountEJBClient.jws Sample

A web service that demonstrates use of an EJB control AccountEJBControl.jcx, which represents the AccountEJB Entity Bean and exposes its business interface to web services.

Concepts Demonstrated by this Sample

- Use of an EJB control

Location of Sample Files

This sample is located in the ejbControl folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\ejbControl\AccountEJBClient.jws

To Run the Sample

To run this web service:

1. Start WebLogic Server in the appropriate domain.
   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/ejbControl/AccountEJBClient.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Create one or more accounts by entering values for key, openingBalance and type and invoking the createNewAccount method.
5. Click the ***Message Log*** title to return to the methods page. Experiment with the other methods in the interface.
6. Examine the AccountEJBClient.jws and AccountEJBControl.jcx files to explore the relationship between the control and its client. The AccountEJBControl.jcx file was created using the Add EJB Control dialog.
7. Select log entries in the Message Log to see the message traffic involved in each interaction.

Related Topics

Using WebLogic Built−In Controls

EJB Control

TraderEJBClient.jws Sample

Test View

AccountPublish.jws Sample

A web service that demonstrates use of a JMS control to publish messages to a JMS topic. AccountSubscribe.jws is a companion to this sample.

Concepts Demonstrated by this Sample

- Use of a JMS control with a topic
- Use of custom JMS control methods
- Use of JMS message properties

Location of Sample Files

This sample is located in the jms folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\jms\AccountPublish.jws

To Run the Sample

The AccountPublish.jws and AccountSubscribe.jws samples work together. The instructions below describe how to use both services:

1. Start WebLogic Server in the Workshop domain. You can do this in one of the following ways:
   - Open the SamplesApp application from the Workshop IDE. Select **File−−>Open−−>Application**, then navigate to BEA_HOME\weblogic81\samples\workshop\SamplesApp\SamplesApp.work. Select **Tools−−>WebLogic Server−−>Start WebLogic Server** to start the server.
   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
   - On Linux or Solaris systems, run BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the AccountPublish.jws service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/jms/AccountSubscribe.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run AccountSubscribe.jws.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Invoke the startListening method. The AccountSubscribe.jws web service is now listening for messages that are published to a JMS topic named jms.AccountUpdate.
5. Launch the AccountPublish.jws by entering http://localhost:7001/WebServices/jms/AccountPublish.jws in the address bar of your browser (not from the WebLogic Workshop visual development environment). If WebLogic Server is running in the appropriate domain on this machine, you may click here to run AccountPublish.jws.
6. Navigate to the ***Test Form*** tab of Test View, if necessary.
7. Enter a string value for accountID and numeric value for amount and invoke the deposit method. At this point a message is published to the jms.AccountUpdate JMS topic.
8. In the browser that is testing AccountSubscribe.jws, click ***Refresh***. You can see that the accountUpdateReceived callback has been sent to the client.

9. Select the ***accountUpdateReceived*** log entry to see the payload of the callback. It should contain the same information you entered for the deposit method in Step 8.
10. The message containing the account transaction is published to the topic by the AccountPublishJMSControl.jcx JMS control used by AccountPublish.jws. The JMS server then sends the message to all active subscribers.
    Since AccountSubscribe.jws is subscribed to the topic via the AccountSubscribeJMSControl.jcx JMS control, it receives the message. If you examine the two JCX files, you will see that the information you entered was encoded in both the message properties and the message body using the @jc:jms−property and @jc:jms−header properties of the JMS controls. When using JMS messaging, the senders and receivers of messages must agree on the message format at design time.

Related Topics

Using WebLogic Built−In Controls

JMS Control

Designing Asynchronous Interfaces

AccountSubscribe.jws Sample

CustomJMSClient.jws Sample

SimpleJMS.jws Sample

Test View

AccountSubscribe.jws Sample

A web service that demonstrates use of a JMS control to subscribe to a JMS topic. AccountPublish.jws is a companion to this sample.

Concepts Demonstrated by this Sample

- Use of a JMS control with a topic
- Use of a custom JMS callback
- Use of JMS message properties

Location of Sample Files

This sample is located in the jms folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\jms\AccountSubscribe.jws

To Run the Sample

The AccountPublish.jws and AccountSubscribe.jws samples work together. The following instructions describe how to use both services:

1. Start WebLogic Server in the appropriate domain.
   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the AccountSubscribe.jws service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/jms/AccountSubscribe.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run AccountSubscribe.jws.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Invoke the startListening method. The AccountSubscribe.jws web service is now listening for messages that are published to a JMS topic named jms.AccountUpdate.
5. Launch the AccountPublish.jws service by by entering http://localhost:7001/WebServices/jms/AccountPublish.jws in the address bar of your browser (*not* from the WebLogic Workshop visual development environment). If WebLogic Server is running in the appropriate domain on this machine, you may click here to run AccountPublish.jws.
6. Navigate to the ***Test Form*** tab of Test View, if necessary.
7. Enter a string value for accountID and numeric value for amount and invoke the deposit method. A message is published to the jms.AccountUpdate JMS topic.
8. In the browser that is testing AccountSubscribe.jws, click ***Refresh***.
9. You should see that the accountUpdateReceived callback has been sent to the client.
10. Select the accountUpdateReceived log entry to see the payload of the callback. It should contain the same information you entered for the deposit method in Step 7.

The message containing the account transaction was published to the topic by the AccountPublishJMSControl.jcx JMS control used by AccountPublish.jws. The JMS server then sent the message to all active subscribers. Since AccountSubscribe.jws is subscribed to the topic via the AccountSubscribeJMSControl.jcx JMS control, it receives the message. If you examine the two JCX files, you will see that the information you entered was encoded in both the message properties and the message body using the @jc:jms−property and @jc:jms−header properties of the JMS controls. When using JMS messaging, the senders and receivers of messages must agree on the message format at design time.

Related Topics

Using WebLogic Built−In Controls

JMS Control

AccountPublish.jws Sample

CustomJMSClient.jws Sample

SimpleJMS.jws Sample

Test View

AdvancedTimer.jws Sample

A web service that demonstrates an asynchronous interface to a simulated legacy system (LegacySystem.jws) that does not support asynchrony. It does so by *polling* the legacy system, which means that the web service calls the legacy system at intervals to see if it is finished. AdvancedTimer does the polling for the client, invoking a client callback if and when the legacy system responds.

Concepts Demonstrated by this Sample

- Use of a Timer control
- Use of a Web Service control
- Declaration and use of a client callback
- Use of the JwsContext interface
- Use of Conversations
- Polling

Location of Sample Files

This sample is located in the timer folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\timer\AdvancedTimer.jws

To Run the Sample

1. Start WebLogic Server in the appropriate domain.

    - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1−−>Examples−−>WebLogic Workshop Examples−−>Start Workshop with Sample Applications***.
    - On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/timer/AdvancedTimer.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Invoke the start method to create a new conversational instance and invoke the operation on the simulated legacy system.
5. After the legacy system completes its operation, AdvancedTimer invokes the onDone callback on the client. If the legacy system does not complete in 15 seconds, AdvancedTimer invokes the onDone callback anyway, but with a failure indication.
6. Refresh the browser periodically until the callback.onDone callback entry appears in the Message Log.
7. Select log entries in the Message Log to see the message traffic involved in each interaction.

Related Topics

BasicAuthentication.jws Sample

The BasicAuthentication web service demonstrates how to secure a web service with basic authentication.

Concepts Demonstrated by this Sample

- Use of basic authentication to secure a web service

Location of Sample Files

This sample is located in the security/transport folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\security\transport\basicAuthentication

To Run the Sample

1. Start WebLogic Server in the appropriate domain.
    - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1*****−−>*****Examples*****−−>*****WebLogic Workshop Examples*****−−>*****Start Workshop with Sample Applications***.
    - On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/security/transport/basicAuthentication/BasicAuthentication.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. When prompted for a username and password, enter:
    - ***username:*** weblogic
    - ***password:*** weblogic
4. Navigate to the ***Test Form*** tab of Test View, if necessary.
5. Invoke the hello method.
6. The message returned by the BasicAuthentication web service appears in the message traffic information for the hello method.

Related Topics

VeriCheck.jws Sample

WebLogic Workshop Security Overview

Buffer.jws Sample

A web service that demonstrates the @common:message−buffer tag to queue high−traffic requests. The sample uses a Timer control to delay sending a response back to the client, simulating waiting for a slow back−end service to respond to requests from this web service. The start method is buffered, allowing clients to continue processing immediately after submitting a request.

Note: In previous releases, the @common:message−buffer tag was known as the @jws:message−buffer tag. This tag is still supported for backward compatibility.

Concepts Demonstrated by this Sample

- Use of a Timer control
- Declaration and use of a client callback
- Use of the JwsContext interface
- Use of Conversations

Location of Sample Files

This sample is located in the async folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\async\Buffer.jws

To Run the Sample

1. Start WebLogic Server in the appropriate domain.

    - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1−−>Examples−−>WebLogic Workshop Examples−−>Start Workshop with Sample Applications***.
    - On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/async/Buffer.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Invoke the startBufferAsync method to create a new conversational instance and invoke the operation on the simulated legacy system.
5. After the artificial delay, Buffer will invoke the BufferResult callback.
6. Refresh the browser periodically until the callback.BufferResult entry appears in the Message Log.
7. Select log entries in the Message Log to see the message traffic involved in each interaction.

Related Topics

Using WebLogic Built−In Controls

Timer Control

Designing Asyncronous Interfaces

JwsContext Interface

Conversation.jws Sample

Test View

clientCert Sample

The clientCert sample demonstrates asynchronous communication between two web services where the target web service is secured with Two−Way SSL.

Concepts Demonstrated by this Sample

- Use of SSL (HTTPS protocol) to secure a web service
- Requiring client certificate authentication
- Calling a secure service through a Web Service control
- Callingback a secure service through a callback interface

Location of Sample Files

This sample is located in the security/transport folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\security\transport\clientCert

To Run the Web Service

1. Start WebLogic Server in the appropriate domain.
    - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
    - On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service by entering http://localhost:7001/WebServices/security/transport/clientCert/WebServiceA.jws?.EXPLORE=.TEST in the address bar of a web browser. If WebLogic Server is running in the samples domain on your machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Click the invokeWebServiceB method.
5. Click ***Refresh***, until the Message Log displays the callback ctrl.result.
6. Select log entries in the Message Log to see the message traffic involved in each interaction.

Related Topics

HelloWorldSecureClient.jws Sample

WebLogic Workshop Security Overview

Conversation.jws Sample

A web service that demonstrates use of the @jws:conversation tag to control the lifecycle of a conversational instance of the web service and provide data persistence.

Conversation.jws implements a synchronous polling interface for the asynchronous HelloWorldAsync web service. A polling interface is necessary if you wish to serve clients that cannot accept asynchronous callbacks due to security arrangements such as firewalls.

Conversations also provide correlation, whereby requests from multiple simultaneous clients are tracked and responses are directed to the right client.

Concepts Demonstrated by this Sample

- Use of a Web Service control
- Use of the JwsContext interface
- Use of Conversations
- Polling

Location of Sample Files

This sample is located in the async folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\async\Conversation.jws

To Run the Sample

1. Start WebLogic Server in the appropriate domain.

    - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
    - On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/async/Conversation.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Invoke the startRequest method to create a new conversational instance and invoke the operation on the simulated legacy system.
5. Click on the conversation ID (the large number at the top of each section in the Message Log) to access that conversation's continue and finish methods.
6. Notice the call from Conversation.jws to helloAsync.HelloAsync. helloAsync is an instance of a Web Service control and HelloAsync is one of its methods.
7. Invoke the getRequestStatus method to see if the request is complete. If you do this before the helloAsync_onHelloResult callback arrives from the Service control, you will get a response telling

you the back end system hasn't yet replied.
8. If you invoke getRequestStatus after the helloAsync_onHelloResult callback has arrives, getRequestStatus will return the result.
9. Click on the conversation ID again to access the conversation's continue and finish methods.
10. Invoke terminateRequest to finish the conversation. This releases the resources associated with this conversational instance of the web service.
11. Select log entries in the Message Log to see the message traffic involved in each interaction.

Related Topics

Using WebLogic Built−In Controls

Web Service Control

Designing Conversational Web Services

Using Polling as an Alternative to Callbacks

JwsContext Interface

Buffer.jws Sample

Test View

CreateUser.jws Sample

A web service that demonstrates the reference of externally defined principals and roles. Using the annotations @common:security run−as−principal and run−as, the web service references an externally defined principal and role. The sample also shows how to access WebLogic Workshop's Security API.

Concepts Demonstrated by this Sample

- Use of @common:security run−as−principal
- Use of @common:security run−as
- Referencing externally defined principals and roles
- Use of WebLogic Workshop Security API

Location of Sample Files

This sample is located in the security/createUser folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\security\createUser\createUser.jws

To Run the Sample

1. Start WebLogic Server in the appropriate domain.
    - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop***−−>***Launch Examples Server***.
    - On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/security/createUser/createUser.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Invoke methods to add, delete and list users in WebLogic Workshop's default authenticator.

Related Topics

@common:security Annotation

CreditReport.jws Sample

Similar to the Investigate web service developed during Tutorial: Your First Web Service, but more fully developed. Demonstrates use of conversations and Web Service controls.

Concepts Demonstrated by this Sample

- Use of a Timer control
- Use of a Web Service control
- Declaration and use of a client callback
- Use of the JwsContext interface
- Use of Conversations
- Use of XML maps
- Polling

Location of Sample Files

This sample is located in the creditreport folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\creditReport\CreditReport.jws

To Run the Sample

1. Start WebLogic Server in the appropriate domain.
    - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
    - On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/creditreport/CreditReport.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Enter any numeric value for ssn and invoke the requestReport method.
5. If you click Refresh, log entries for calls to the two external services will appear, to bank.startCustomerAnalysis and irs.requestTaxReport.
6. After 10 seconds, bank will report back by sending the onDeliverAnalysis callback. The partial results will be forwarded to the client with an invocation of CreditReport's onProgressNotify callback. You must continually click Refresh to see method invocations on controls and callbacks because there is no way to push these events to a browser.
7. At any time, you may navigate to the continue methods for CreditReport by clicking on the conversation ID in the Message Log.
8. After 20 seconds, the irs Web Service control will respond. Another onProgressNotify callback is sent to the client, then finally a onReportDone callback ends the conversation.
9. At any time before the conversation ends you may invoke getCurrentStatus.

10. At any time before the conversation ends you may invoke cancelReport, which will in turn invoke cancel methods on the external services if they are still pending.
11. Select log entries in the log to see the message traffic involved in each interaction.

Related Topics

Using WebLogic Built−In Controls

Timer Control

Web Service Control

Designing Conversational Web Services

Using Polling as an Alternative to Callbacks

JwsContext Interface

CustomerDBClient.jws Sample

A sample that demonstrates use of a database control by managing customer records. CustomerDBClient is a client of CustomerDBControl.jcx. Together they demonstrate construction of a Database control and use of a Database control by a web service. CustomerDBControl.jcx demonstrates use of SQL's CREATE, DROP, INSERT, UPDATE, and SELECT statements.

Concepts Demonstrated by this Sample

- Use of a Database control

Location of Sample Files

This sample is located in the database folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\database\CustomerDBClient.jws

To Run the Sample

1. Start WebLogic Server in the appropriate domain.
    - ♦ On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
    - ♦ On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/database/CustomerDBClient.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the Test Form tab of Test View, if necessary.
4. Invoke the createCustomerTable method to create the database table and populate it with test data.
5. Use other methods in the interface to query the database in various ways. Until you insert additional records, valid customer IDs are 1, 2 or 3.
6. Select log entries in the Message Log to see the essage traffic involved in each interaction.
7. Examine the source code for CustomerDBClient.jws and CustomerDBControl.jcx to see how the JCX file defines database operations and method shape and the web service uses the methods and data structures provided by the Database control.

Related Topics

LuckyNumberDBClient.jws Sample

Using WebLogic Built−In Controls

Database Control Design Issues

Test View

CustomJMSClient.jws Sample

A web service that demonstrates use of a JMS control defined in a JCX file. CustomJMSClient is a client of the CustomJMSControl.jcx JMS control.

Concepts Demonstrated by this Sample

- Use of a JMS control with a queue
- Declaration and use of a client callback
- Use of Conversations

Location of Sample Files

This sample is located in the jms folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\jms\CustomJMSClient.jws

To Run the Sample

1. Start WebLogic Server in the appropriate domain.

   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/jms/CustomJMSClient.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Enter values for firstname and lastname and invoke the sendPerson method. The values are packaged in a message and sent to a JMS queue via the JMS control myCustomQ.
5. Since the JMS control in this simple example is configured to listen to the same queue it sends to, the message immediately arrives and is forwarded to the web service via the onResponse callback. Refresh the browser to see the callback entry appear in the Message Log.
6. Select log entries in the Message Log to see the message traffic involved in each interaction.

Related Topics

SimpleJMS.jws Sample

Using WebLogic Built−In Controls

JMS Control

Designing Conversational Web Services

Test View

DynamicSQL Sample

This sample demonstrates passing dynamically generated SQL statements to a database control.

Concepts Demonstrated by this Sample

- Passing dynamically generated SLQ statements and phrases to a database control.
- Substitution syntax for SQL statements and phrases
- Substitution syntax for individual SQL terms.

Location of Sample Files

This sample is located in the database/dynamicSQL folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\database\dynamicSQL

To Run the Sample

- Start WebLogic Server in the appropriate domain.

  On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop***−−>***Launch Examples Server***.

  On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/database/dynamicSQL/DynamicSQL.jws?.EXPLORE=.TEST in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
- Navigate to the Test Form tab of Test View, if necessary.
- Invoke the methods listed to query the database.

Related Topics

Parameter Substitution in @jc:sql Statements

HelloWorld.jws Sample

A very simple web service with a single synchronous method. Illustrates a WebLogic Workshop web service in its simplest form.

Concepts Demonstrated by this Sample

- Basic WebLogic Workshop web service structure

Location of Sample Files

This sample is located in the root folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\HelloWorld.jws

To Run the Sample

1. Start WebLogic Server in the appropriate domain.
   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop*** −−>***Launch Examples Server***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/HelloWorld.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Invoke the Hello method. The return value is the string "Hello, World!", of course.
5. Select log entries in the Message Log to see the message traffic involved in each interaction.

Related Topics

Guide to Development with WebLogic Workshop

Structure of a JWS File

@common:operation Tag

Test View

HelloWorldAsync.jws Sample

A simple asynchronous web service that illustrates the use of a callback. Uses a Timer control to provide a delay that simulates waiting for a slow back−end service to do some work, then notifies the client of the result via a callback. HelloWorldAsync.jws is used by the Conversation.jws sample, but is also a standalone web service.

Concepts Demonstrated by this Sample

- Use of a Timer control
- Declaration and use of a client callback
- Use of Conversations

Location of Sample Files

This sample is located in the async folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\async\HelloWorldAsync.jws

To Run the Sample

1. Start WebLogic Server in the appropriate domain.
    - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
    - On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/async/HelloWorldAsync.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the Test Form tab of Test View, if necessary.
4. Invoke the HelloAsync method to create a new conversational instance and invoke the simulated legacy system operation.
5. After 5 seconds, HelloWorldAsync invokes the onHelloResult callback on the client.
6. Refresh the browser periodically until the callback.onHelloResult callback entry appears in the Message Log.
7. Select log entries in the Message Log to see the message traffic involved in each interaction.

Related Topics

Using WebLogic Built−In Controls

Timer Control

Designing Conversational Web Services

Test View

HelloWorldSecureClient.jws Sample

The HelloWorldSecureClient web service demonstrates synchronous communication between two web service where the target web service is secured using SSL.

Concepts Demonstrated by this Sample

- Use of SSL (HTTPS protocol) to secure a web service
- Calling a secure service through a Web Service control

Location of Sample Files

This sample is located in the security/sync folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\security\transport\helloWorldSecure\

To Run the Sample

1. Start WebLogic Server in the appropriate domain.
    - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
    - On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/security/transport/helloWorldSecure/HelloWorldSecureClient.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Invoke the HelloWorldSecureClient method.
5. Refresh the browser periodically until the hws.HelloWorldSecure method entry appears in the Message Log.
6. Select log entries in the Message Log to see the message traffic involved in each interaction. The message returned by the HelloWorldSecure web service appears in the message traffic information for the invokeHWSecure method.

Related Topics

VeriCheck.jws Sample

WebLogic Workshop Security Overview

Investigate.jws Sample

The Investigate.jws sample demonstrates how to provide web access for a Java control with an asynchronous web service.

For step−by−step instructions for building Investigate.jws, see Tutorial: Web Services.

Concepts Demonstrated by this Sample

- Use of an asynchronous web service to expose the functionality of a Java control
- Use of XQuery maps to modify the SOAP output of a web service
- Use of ECMA script to modify an XQuery map
- WS−Security

Location of Sample Files

This sample is located in the FirstWebService/investigateJWS folder of the TutorialsApp WebLogic Workshop application. In the file system the location is:

BEA_HOME\weblogic81\samples\platform\TutorialsApp\FirstWebService\investigateJWS

To Run the Sample

1. Start WebLogic Workshop.
    - ♦ On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Quick Start***
2. On the Quick Start menu click ***Experience WebLogic Workshop 8.1***
3. In ***WebLogic Workshop***, select ***File−−>Open−−>Application***.
4. In the ***Open Workshop Application*** dialog, navigate to BEA_HOME/weblogic81/samples/platform/TutorialsApp/TutorialsApp.work and click ***Open***.
5. On the ***Application*** tab, right−click the ***TutorialsApp*** folder and select ***Build Application***.
6. Start WebLogic Server by selecting ***Tools−−>WebLogic Server−−>Start WebLogic Server***.
7. Launch the sample by entering http://localhost:7001/FirstWebService/investigateJWS/Investigate.jws?.EXPLORE=.TEST in the address bar of a web browser. Or you may click here to run the sample.
8. Enter a taxID and invoke the ***requestCreditReport*** method.
9. Click ***Refresh***, until the Message Log displays the method callback.onCreditReportDone.
10. Select log entries in the Message Log to see the message traffic involved in each interaction.

Related Topics

Tutorial: Web Services

InvestigateSync.jws Sample

InvestigateSync.jws Sample

The InvestigateSync.jws sample demonstrates how to provide web access for a Java control using a synchronous web service.

For step−by−step instructions for building InvestigateSync.jws, see Tutorial: Web Services.

Concepts Demonstrated by this Sample

- Use of an synchronous web service to expose the functionality of a Java control
- Polling as an alternative to callbacks

Location of Sample Files

This sample is located in the FirstWebService/investigateJWS folder of the TutorialsApp WebLogic Workshop application. In the file system the location is:

BEA_HOME\weblogic81\samples\platform\TutorialsApp\FirstWebService\investigateJWS

To Run the Sample

1. Start WebLogic Workshop.
    - ♦ On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1−−>Quick Start***
2. On the Quick Start menu click ***Experience WebLogic Workshop 8.1***
3. In ***WebLogic Workshop***, select ***File−−>Open−−>Application***.
4. In the ***Open Workshop Application*** dialog, navigate to BEA_HOME/weblogic81/samples/platform/TutorialsApp/TutorialsApp.work and click ***Open***.
5. On the ***Application*** tab, right−click the ***TutorialsApp*** folder and select ***Build Application***.
6. Start WebLogic Server by selecting ***Tools−−>WebLogic Server−−>Start WebLogic Server***.
7. Launch the sample by entering http://localhost:7001/FirstWebService/investigateJWS/InvestigateSync.jws?.EXPLORE=.TEST in the address bar of a web browser. Or you may click here to run the sample.
8. Enter a taxID and invoke the ***requestCreditReport*** method.
9. Click ***Continue this Conversation***.
10. Invoke the ***isReportReady*** method until it returns the value ***true***.
11. Click ***Continue this Convesation***.
12. Invoke the ***getCreditReportDone*** method to view the service response.
13. Select log entries in the Message Log to see the message traffic involved in each interaction.

Related Topics

Tutorial: Web Services

Investigate.jws Sample

JMS−XML Protocol Sample

This sample shows how to invoke a web service through a JMS message queue.

Concepts Demonstrated by this Sample

- Using the JMS−XML protocol
- Invoking a web service through a JMS message queue

Sample Files

In your local file system, the sample is located at:

[BEA_HOME]\weblogic81\samples\workshop\SamplesApp\WebServices\jms_xmlProtocol\

The sample contains the following three files:

1. ***ClientJWS.jws***: This web service calls the java client file ClientXML.java.
2. ***ClientXML.java***: This JAVA file connects to the JMS queue, constructs SOAP requests, and sends those messages (via JMS) to invoke the web service JMS_XMLProtocol.jms.
3. ***JMS_XMLProtocol.jws***: This web service receives SOAP requests through the JMS queue. Responses are writen to System.out.

To Run the Sample

- Start WebLogic Server in the workshop domain.

- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1−−>Examples−−>WebLogic Workshop Examples−−>Launch Examples Server***.
- On Linux or Solaris systems, run:

  [BEA_HOME]/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/jms_xmlProtocol/ClientJWS.jws in the address bar of your browser. If WebLogic Server is running in the workshop domain on your machine, you may click here to run the sample.
- In the browser, click the ***Test XML*** tab, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

Building a JMS Client

LuckyNumberDBClient.jws Sample

A sample that uses a database to store players and their lucky numbers. LuckyNumberDBClient uses the LuckyNumber.jws web service to generate random numbers in the range [1,20] inclusive, then checks for winners in a database and returns the number drawn and the list of winners, if any. The database is managed using the LuckyNumberDBControl.jcx Database control. LuckyNumberDBControl.jcx creates and manages a PLAYERS table in the database and implements queries against it. Demonstrates using SQL's CREATE, INSERT, SELECT and DROP statements in a database control.

Concepts Demonstrated by this Sample

- Use of a Database control

Location of Sample Files

This sample is located in the database folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\database\LuckyNumberDBClient.jws

To Run the Sample

1. Start WebLogic Server in the appropriate domain.
   - ♦ On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
   - ♦ On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/database/LuckyNumberDBClient.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Invoke the start method to create the database table and populate it with test data.
5. Click on the conversation ID (the large number in the Message Log) to access the continue and finish methods for that conversation.
6. Invoke the drawNumber method. The web service invokes the getLuckyNumber method of the LuckyNumber web service to obtain a random number. LuckyNumberDBClient then queries the database, via the LuckyNumberDBControl Database control, to see if there are any players holding that number.
7. Any winners found are returned in a string, as the return value of drawNumber.
8. Test View always show the details of a Message Log entry when it arrives. Since the invocation of getLuckyNumber occurred after the invocation of drawNumber, Test View is now displaying the details of the getLuckyNumber invocation. Select the drawNumber entry in the Message Log to see its formatted return string.
9. Select log entries in the Message Log to see the message traffic involved in each interaction.

10. Examine the source code for LuckyNumberDBClient.jws and LuckyNumberDBControl.jcx to see how the JCX file defines database operations and method shape and the web service uses the methods and data structures provided by the Database control.

Related Topics

CustomerDBClient.jws Sample

Using WebLogic Built−In Controls

Database Control

@jc:sql Tag

Test View

QuoteClient.jws Sample

Demonstrates the ability to modify a Web Service control's JCX file to change the Java signature of one or more methods while still adhering to the public contract of the called web service. QuoteServiceControl.jcx was originally generated form the QuoteService.jws web service. But one method in QuoteServiceControl.jcx has had a parameter removed and replaced with a hard−coded value in the accompanying XML map. Thus, the outgoing XML message is the same, but the Java signature visible to clients of the Web Service control is simplified.

Concepts Demonstrated by this Sample

- Use of a Web Service control
- Customization of a Web Service control
- Declaration and use of a client callback
- Use of Conversations

Location of Sample Files

This sample is located in the service folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\service\QuoteClient.jws

To Run the Sample

1. Start WebLogic Server in the appropriate domain.
    - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1−−>Examples−−>WebLogic Workshop Examples−−>Start Workshop with Sample Applications***.
    - On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/service/QuoteClient.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Enter a value for tickerSymbol (BEAS, of course) and invoke the getQuote method.
5. QuoteClient then calls the getQuote method of QuoteService, via the QuoteServiceControl Web Service control, which will return a price quote for the specified ticker symbol.

Running this sample isn't the interesting aspect of it. If you examine QuoteService.jws, you will see that its getQuote method expects two parameters. But QuoteClient is calling the method with a single parameter. This is accomplished via an XML map on the QuoteServiceControl Web Service control.

QuoteServiceControl.jcx was originally generated from QuoteService.jws. It was then modified: the first parameter of getQuote was removed from the Java signature, meaning callers are no longer expected to pass it. An XML map was then added to getQuote, with the value of the <customerID> element hard−coded. This

leaves the message shape as QuoteService expects it, with two parameters. The calling service passes one parameter, which the Web Service control combines with the hard−coded parameter to produce the two−parameter message the target service expects.

Related Topics

Using WebLogic Built−In Controls

Web Service Control

Defining Java to XML Translation with XML Maps

Designing Conversational Web Services

Using Callbacks to Notify Clients of Events

JwsContext Interface

Test View

SimpleJMS.jws Sample

A web service that demonstrates use of a JMS control that sends to and receives from a queue.

Concepts Demonstrated by this Sample

- Use of a JMS control with a queue
- Declaration and use of a client callback
- Use of Conversations

Location of Sample Files

This sample is located in the jms folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\jms\SimpleJMS.jws

To Run the Sample

1. Start WebLogic Server in the appropriate domain.
    - ♦ On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1−−>Examples−−>WebLogic Workshop Examples−−>Start Workshop with Sample Applications***.
    - ♦ On Linux or Solaris systems, run:

      BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/jms/SimpleJMS.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Enter a value for name and invoke the start method to start a conversation.
5. Navigate to the continue and finish methods for the conversation by clicking on the conversation ID (the large number) at the top of the section in the Message Log.
6. Enter values for msg and invoke the sendString method. The values are packaged in a message and sent to a JMS queue via the JMS control myCustomQ, which is an instance of CustomJMSControl.
7. Since the JMS control in this simple example is configured to listen to the same queue it sends to, the message immediately arrives and is forwarded to the web service via the onMessageReceived callback. Refresh the browser to see the callback entry appear in the Message Log.
8. Select log entries in the Message Log to see the message traffic involved in each interaction.

Related Topics

CustomJMSClient.jws Sample

Using WebLogic Built−In Controls

JMS Control

Designing Conversational Web Services

Test View

SimpleTimer.jws Sample

A simple web service that demonstrates use of the Timer control. Uses the @jc:timer timeout= tag.

Concepts Demonstrated by this Sample

- Use of a Timer control
- Declaration and use of a client callback
- Use of Conversations

Location of Sample Files

This sample is located in the timer folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\timer\SimpleTimer.jws

To Run the Sample

1. Start WebLogic Server in the appropriate domain.

   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1−−>Examples−−>WebLogic Workshop Examples−−>Start Workshop with Sample Applications***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/timer/SimpleTimer.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Invoke the createTimer method to create a new conversational instance.
5. Click on the conversation ID (the large number) in the Message Log to access this conversation's continue and finish methods.
6. Invoke the setAlarm method. This starts the Timer control, which expires in 5 seconds. When the timer expires, it calls its onTimeout callback, which arrives at SimpleTimer via the timer_onTimeout callback handler. The timer_onTimeout callback handler in turn invokes SimpleTimer's onAlarm callback to the client.
7. Refresh the browser to see callbacks arriving from the service. The use of a browser for Test View does not allow WebLogic Server to push notification to the browser when callbacks execute.

   Note: The setAlarm method uses the Timer control's restart method, so invoking setAlarm before the timer expires resets it so that it expires 5 seconds from the invocation of setAlarm.
8. Refresh the browser periodically until the callback.onAlarm callback entry appears in the Message Log.
9. Select log entries in the Message Log to see the message traffic involved in each interaction.

Related Topics

AdvancedTimer.jws Sample

Using WebLogic Built−In Controls

Timer Control

Designing Conversational Web Services

Using Callbacks to Notify Clients of Events

Test View

TraderEJBClient.jws Sample

A web service that demonstrates use of the EJB control TraderEJBControl.jcx, which represents the TraderEJB Stateless Session Bean and exposes its business interface to web services.

Concepts Demonstrated by this Sample

- Use of an EJB control
- Use of Conversations

Location of Sample Files

This sample is located in the ejbControl folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\ejbControl\TraderEJBClient.jws

To Run the Sample

1. Start WebLogic Server in the appropriate domain.
   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1−−>Examples−−>WebLogic Workshop Examples−−>Start Workshop with Sample Applications***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/ejbControl/TraderEJBClient.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Enter values for tickerSymbol and numberOfShares in for either the buy or sell method and invoke the method.
5. Click the ***Message Log*** title to return to the methods.

   Note: TraderEJB, the EJB that is represented by TraderEJBControl, is a Stateless Session Bean and therefore does not store any information. Its sole purpose is to qualify transacations: it limits all transactions to 500 shares.
6. Examine the TraderEJBClient.jws and TraderEJBControl.jcx files to explore the relationship between the control and its client. The TraderEJBControl.jcx file was created using the Add EJB Control dialog.
7. Select log entries in the Message Log to see the message traffic involved in each interaction.

Related Topics

AccountEJBClient.jws Sample

Using WebLogic Built−In Controls

EJB Control

Test View

VeriCheck.jws Sample

The VeriCheck web service demonstrates asynchronous communication between two web servers both of which are secured with SSL and role−based security.

Concepts Demonstrated by this Sample

- Use of SSL (HTTPS protocol) to secure a web service
- Use of basic authentication to secure a web service
- Calling a secure service through a Web Service control
- Callingback a secure service through a callback interface

Location of Sample Files

This sample is located in the security/roleBased folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\security\roleBased\VeriCheck.jws

To Run the Web Service

1. Start WebLogic Server in the appropriate domain.
   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service by entering https://localhost:7002/WebServices/security/roleBased/VeriCheck.jws?.EXPLORE=.TEST in the address bar of a web browser. If WebLogic Server is running in the samples domain on your machine, you may click here to run the sample.
3. When prompted to accept the digital certificate, click ***Yes***.
4. When prompted for a username and password, enter:
   - ***username:*** weblogic
   - ***password:*** weblogic
5. Navigate to the ***Test Form*** tab of Test View, if necessary.
6. Enter values for checkingAccountID and amount in the checkForSufficientBalance method. The checkingAccountID parameter can be any string value, the checkForSufficientBalance parameter must be an integer value. Invoke the method by clicking the button labeled checkForSufficientBalance.

   The VeriCheck web service sends the checking account and amount parameters to the Bank web service. The Bank web service checks to see if the checking account has sufficient funds to cover the amount specified. Once the Bank web service has completed its task, it calls back the VeriCheck web service informing VeriCheck whether the account has sufficient funds. The VeriCheck web service then calls back the original client informing the client whether the account has sufficient funds.

Both the VeriCheck and Bank web services require basic authentication. When the VeriCheck service invokes the Bank service's doesAccountHaveSufficientBalance method, it sends authentication information via the method calls setUsername() and setPassword().

Similarly, when the Bank callsback the VeriCheck service, it sends authentication information via the method calls setCallbackUsername() and setCallbackPassword().

7. Click ***Refresh***, until the Message Log displays the callback callback.onCheckDone.
8. Select log entries in the Message Log to see the message traffic involved in each interaction.

Related Topics

HelloWorldSecureClient.jws Sample

WebLogic Workshop Security Overview

WS−Security Callback Sample

The WS−Security Callback sample demonstrates asynchronous communication between two web services both of which are secured with WS−Security security.

Concepts Demonstrated by this Sample

- Use of WS−Security to secure a web service
- Invoking a secure service through a Web Service control
- Callingback a secure service through a callback interface

Location of Sample Files

This sample is located in the security/wsse folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\security\wsse\callback

To Run the Sample

1. Start WebLogic Server in the appropriate domain.
   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

2. Launch the sample by entering http://localhost:7001/WebServices/security/wsse/callback/client/Client.jws?.EXPLORE=.TEST in the address bar of a web browser. If WebLogic Server is running in the samples domain on your machine, you may click here to run the sample.
3. Click the invokeHello method.
4. Click ***Refresh***, until the Message Log displays the callback callback.callback.
5. Select log entries in the Message Log to see the message traffic involved in each interaction.

Related Topics

Web Service Security

WS−Security ReqResp Sample

The WS−Security ReqResp sample demonstrates synchronous communication between two web services both of which are secured with WS−Security security.

Concepts Demonstrated by this Sample

- Use of WS−Security to secure a web service
- Invoking a secure service through a web service control

Location of Sample Files

This sample is located in the security/wsse folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\security\wsse\reqResp

To Run the Sample

1. Start WebLogic Server in the appropriate domain.
   - ♦ On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
   - ♦ On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the sample by entering http://localhost:7001/WebServices/security/wsse/reqResp/client/Client.jws?.EXPLORE=.TEST in the address bar of a web browser. If WebLogic Server is running in the samples domain on your machine, you may click here to run the sample.
3. Click the invokeHello method.
4. Click ***Refresh***, until the Message Log displays the method myCompanyControl.hello.
5. Select log entries in the Message Log to see the message traffic involved in each interaction.

Related Topics

Web Service Security

WS−Security ReqResp Sample

The WS−Security ReqResp sample demonstrates synchronous communication between two web services both of which are secured with WS−Security security.

Concepts Demonstrated by this Sample

- Use of WS−Security to secure a web service
- Invoking a secure service through a web service control

Location of Sample Files

This sample is located in the security/wsse folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\security\wsse\reqResp

To Run the Sample

1. Start WebLogic Server in the appropriate domain.
   - ♦ On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
   - ♦ On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the sample by entering http://localhost:7001/WebServices/security/wsse/reqResp/client/Client.jws?.EXPLORE=.TEST in the address bar of a web browser. If WebLogic Server is running in the samples domain on your machine, you may click here to run the sample.
3. Click the invokeHello method.
4. Click ***Refresh***, until the Message Log displays the method myCompanyControl.hello.
5. Select log entries in the Message Log to see the message traffic involved in each interaction.

Related Topics

Web Service Security

Web Service Client Samples

The samples in this section help you build web services with Java clients and .NET clients.

Topics Included in This Section

Java Client Samples

Describes three examples of using the Java proxy that WebLogic Workshop provides for each web service. The same web service is used from a JSP page; from a Java console application and from a Java Swing application.

.NET Client Sample

Illustrates how to use a WebLogic Workshop web service from a .NET client, including full participation in conversations and receipt of a callback. A fully developed and documented ASP.NET web service that is a client of a WebLogic Workshop web service is provided.

Related Topics

WebLogic Workshop Samples

Java Client Samples

Any application can communicate with a web service if it can generate and consume XML messages and use one of the protocols supported by the target web service. WebLogic Workshop provides Java proxy classes that allow any Java program to use a particular WebLogic Workshop web service. The proxy classes allow the Java application to treat the web service as though it is a normal Java class.

The proxy classes, along with supporting classes provided by WebLogic Server, perform the following work:

- Converting Java method invocations on the proxy to appropriate XML messages
- Managing sending the XML messages to the web service over an appropriate protocol
- Receiving response messages
- Converting received XML response messages into Java types

Concepts Demonstrated by these Samples

- Use of a web service's Java proxy from a JSP
- Use of a web service's Java proxy from a Java console application
- Use of a web service's Java proxy from a Java Swing application.
- Building a client to call a conversational web service

Location of the Maze Generator Sample Files

The SamplesApp application provides two Java client applications, a standalone Java console application and a Swing application, that demonstrate basic use of the proxy classes. These are located in the ProxyClient project of the SamplesApp application. Step−by−step instructions for building and running these sample clients are available in the mazegen folder of the ProxyClient project.

The SamplesApp application also provides a JSP client that uses the proxy classes to call a web service. It is located in the WebApp project, in the \jspProxyClient\mazegen folder, along with a readme file explaining how to run the JSP client. If WebLogic Server is running in the appropriate domain on this machine, you may click here to try the JSP client sample.

The web service which provides the proxy classes, MazeGenerator.jws, is located in the WebServices project, in the \proxy\mazegen folder. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the web service.

Location of the Register Sample Files

The SamplesApp application includes a standalone Java application that demonstrates how to call a conversational web service from a Java client. This sample is located in the register folder of the ProxyClient project. Step−by−step instructions for building and running this sample is available in this folder.

Related Topics

How Do I: Communicate with a Web Service from a JSP or Servlet?

How Do I: Communicate with a Web Service from Another Java Application?

How Do I: Tell Developers of Non−WebLogic Workshop Clients How to Participate in Conversations?

How Do I: Use the Java Proxy for a Web Service?

.NET Client Sample

Any application can communicate with a web service if it can generate and consume XML messages and use one of the protocols supported by the target web service. This includes applications and web services built in other tools and other languages. The central reason for the development of the web service concept was to enable inter−operation of software components regardless of the operating system or language with which they are implemented. The key to this inter−operation is the WSDL file, which describes a web service in an implementation−independent way.

Visual Studio .NET can use the WSDL file of a web service to produce a web service proxy. A .NET component can then use the proxy class to communicate with the web service as though it were a local object.

Concepts Demonstrated by this Sample

- Use of a WebLogic Workshop web service from an ASP.NET web service.
- Non−WebLogic Workshop web service client participating in a conversation.
- Non−WebLogic Workshop web service accepting a callback.

Location of Sample Files

This sample is located in the interop\dotNET folder of the WebServices project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\interop\dotNET

To Run the Sample

Step by step instructions for building and running the .NET web service client may be found in the readme.html file in the interop\dotNET sample directory (above). If you are viewing this documentation on a machine running WebLogic Server in the WebLogic Workshop samples domain, you may click here to view the readme.html file.

Related Topics

How Do I: Communicate with a Web Service from a JSP or Servlet?

How Do I: Communicate with a Web Service from Another Java Application?

How Do I: Tell Developers of Non−WebLogic Workshop Clients How to Participate in Conversations?

How Do I: Use the Java Proxy for a Web Service?

XMLBeans Samples

The samples in this section illustrate how to work with XML using XMLBeans.

Topics Included in This Section

CustomerDB_XMLBean Sample

A database control and associated web service client that show how to return XMLBean types from a database control.

ItemsDB_XMLBean Sample

A database control and associated web service client that show how to return XMLBean types from a database control.

MixedContent.jws Sample

A web service illustrating how you can use an XML cursor to manipulate the content of an XML element.

SchemaChoice.jws Sample

A web service illustrating how you can construct an XML document with XMLBeans and types compiled from schema, and how you can validate the document in cases where your code may create a document that is invalid.

SchemaEnum.jws Sample

A web service illustrating how you can access XML values that are defined in schema as enumerations.

SelectPath.jws Sample

A web service demonstrating how to use the XmlCursor.selectPath method to execute an XPath.

SimpleAccess.jws Sample

A web service illustrating basic use of XMLBeans compiled from XML schema.

SimpleExpressions.jws Sample

A web service illustrating XQuery by showing a few simple expressions in use.

ThresholdService.jws Sample

A web service to use as a test client for the ThresholdEnums Java control, which contains code that illustrates how you can use the schema type system to retrieve information about a compiled schema.

TokenTypes.jws Sample

A web service illustrating how tokens correspond to portions of an XML document.

TypeHierarchyPrinterService.jws Sample

A web service that prints a hierarchical rendering of the schema types represented by the schema type system available to its application.

XsdConfig.jws Sample

A web service illustrating how you can use an XSDCONFIG file to guide naming translation when the schema compiler generates an API corresponding to your schema.

Related Topics

Getting Started with XMLBeans

CustomerDB_XMLBean Sample

A database control and web service client that demonstrate how to return XMLBean types from the database, and pass these types to a web service.

Concepts Demonstrated by this Sample

- Querying a database with database control methods
- Returning XMLBean types from a database control
- Using XMLBean types to handle communication between a database control and its web service client

Location of Sample Files

This sample is located in the database/xmlbean folder of the WebServices project in the SamplesApp sample application. In the file system the locations are:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\database\xmlbean\CustomerDB_XML
BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\database\xmlbean\CustomerDB_XML

To Run the Sample

To run this web service:

1. Start WebLogic Server in the Workshop domain.
   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/database/xmlBean/CustomerDB_XMLBeanClient.jws?.EXPLORE=.TEST in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Invoking the web service methods calls corresponding methods on the database control.
5. Select log entries in the Message Log to see the message traffic involved in each interaction.

Related Topics

Returning XMLBeans from a Database Control

ItemsDB_XMLBean Sample

ItemsDB_XMLBean Sample

A database control and web service client that demonstrate how to return XMLBean types from the database, and pass these types to a web service.

Concepts Demonstrated by this Sample

- Querying a database with database control methods
- Returning XMLBean types from a database control
- Using XMLBean types to handle communication between a database control and its web service client

Location of Sample Files

This sample is located in the database/xmlbean folder of the WebServices project in the SamplesApp sample application. In the file system the locations are:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\database\xmlbean\Items_XMLBean.j
BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\database\xmlbean\Items_XMLBeanC

To Run the Sample

To run this web service:

1. Start WebLogic Server in the Workshop domain.
   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Workshop with Sample Applications***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh
2. Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/database/xmlBean/Items_XMLBeanClient.jws?.EXPLORE=.TEST in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Invoking the web service methods calls corresponding methods on the database control.
5. Select log entries in the Message Log to see the message traffic involved in each interaction.

Related Topics

Returning XMLBeans from a Database Control

CustomerDB_XMLBean Sample

MixedContent.jws Sample

A web service illustrating how you can use an XML cursor to manipulate the content of an XML element. While working with strongly−typed XML (in which you are accessing the XML through an API generated from schema) provides easy access for getting and setting the entire value of an element or attribute, it does not easily provide finer grained access to an element's content. This web service shows how you can use an XML cursor to "dive into" an element's content, manipulating it on a character−by−character level.

The code in this web service is designed to look at the description of each item in an inventory list, creating a link wherever the description contains a reference to another item in the inventory list. This alters the description element so that it contains a mix of text and link elements. Such an element is said to have "mixed content."

Concepts Demonstrated by this Sample

- XML cursor navigation
- Using a cursor to alter XML

Location of Sample Files

This sample is located in the xmlBeans/cursor folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\xmlBeans\cursor\MixedContent.jws

To Run the Sample

- Start WebLogic Server in the appropriate domain.
- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/xmlBeans/cursor/MixedContent.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
- Navigate to the ***Test XML*** tab of Test View, if necessary.
- To test the linkItems method, paste the contents of the InventoryItem.xml file over the <inventory> element (and its children) provided by Test View. Click ***linkItems*** to view the XML produced by this code.
- Under ***Service Response***, notice that <link> elements have been inserted into the resulting XML.

Related Topics

Navigating XML with Cursors

XmlCursor Interface

SchemaChoice.jws Sample

A web service illustrating how you can construct an XML document with XMLBeans and types compiled from schema. It also shows how you can validate the document in cases where your code may create a document that is invalid.

The schema for the document created by the sample, EmployeeMarital.xsd, defines a snippet of data about an employee that specifies their marital status (for example, "Married", "Single", or "Domestic partner"). The schema defines a "choice" structure in which only one of a set of elements is allowed. The set in this case includes partner−name and spouse−name elements. Only one of these would be applicable for a given employee, so only one should be allowed. However, to illustrate validation, the code here tries to add both.

Validation is done with the XmlObject.validate method, which in this case takes an XmlOptions argument. The option specifies an ArrayList as a "listener" to use for collecting errors that occur while validating −− but any object implementing the java.util.Collection interface will work for this purpose. During validation, the XMLBeans runtime components call the listener's add method as errors are discovered, adding XmlError instances that this code can use to record and report what happened with the validation.

Concepts Demonstrated by this Sample

- Enumerations generated from schema
- Validation against schema using the XMLBeans API
- xs:choice schema structure translated into Java types

Location of Sample Files

This sample is located in the xmlBeans/schema folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\xmlBeans\schema\SchemaChoice.jws

To Run the Sample

- Start WebLogic Server in the appropriate domain.
- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/xmlBeans/schema/SchemaChoice.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
- Navigate to the ***Test Form*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

XmlObject interface

XmlCursor Interface

SchemaEnum.jws Sample

A web service illustrating how you can access XML values that are defined in schema as enumerations. When a schema containing enumerations is compiled, the generated Java types represent the enumerations with enumerations of their own.

Concepts Demonstrated by this Sample

- Enumerations generated from schema

Location of Sample Files

This sample is located in the xmlBeans/schema folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\xmlBeans\schema\SchemaEnum.jws

To Run the Sample

- Start WebLogic Server in the appropriate domain.
- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/xmlBeans/schema/SchemaEnum.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
- Navigate to the ***Test XML*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

Getting Started with XMLBeans

XmlObject interface

XmlCursor Interface

SelectPath.jws Sample

A web service demonstrating how to use the XmlCursor.selectPath method to execute an XPath expression. Unlike the execQuery method, which returns an XmlCursor containing the results of the query, the selectPath method updates the content of the cursor it is called from.

In other words, for a method call such as myCursor.selectPath("$this/elementName"), the myCursor variable would afterward contain the query results. If there were multiple elementName elements returned, you could use "selection" methods to navigate among the elementName elements and access them individually. "selection" methods include toNextSelection, toPreviousSelection, getSelectionCount, addToSelection, and so on.

For more information on using XPath and XQuery with the selectPath method, see Selecting XML with XQuery and XPath.

Concepts Demonstrated by this Sample

- Using XPath and XQuery with an XML cursor

Location of Sample Files

This sample is located in the xmlBeans/xquery folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\xmlBeans\xquery\SelectPath.jws

To Run the Sample

- Start WebLogic Server in the appropriate domain.
- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/xmlBeans/xquery/SelectPath.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
- Navigate to the ***Test XML*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

Getting Started with XMLBeans

Selecting XML with XQuery and XPath

SimpleAccess.jws Sample

A web service illustrating basic use of XMLBeans compiled from XML schema. With XMLBeans, you can compile XML schemas to generate types through which you can access XML instances that conform to the schema. With the generated types, you can use get and set methods in the way that you would with other JavaBeans.

In WebLogic Workshop, you can compile schema files simply by copying them to a Schema project in the application. WebLogic Workshop compiles the schemas and places the resulting JAR file among the application's libraries. From there, types generated from the schema are available to other code in the application.

Concepts Demonstrated by this Sample

- Java types generated by compiling XML schema
- Accessing generated types by using JavaBeans−style get and set methods

Location of Sample Files

This sample is located in the xmlBeans/schema folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\xmlBeans\schema\SimpleAccess.jws

To Run the Sample

- Start WebLogic Server in the appropriate domain.
- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/xmlBeans/schema/SimpleAccess.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
- Navigate to the ***Test XML*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

Getting Started with XMLBeans

SimpleExpressions.jws Sample

A web service illustrating XQuery by showing a few simple expressions in use. XQuery offers a variety of ways to operate on XML. It supports simple path expressions that are similar to XPath, as well as more complex expressions for looping through XML.

All of the XQuery expressions in the source code use the special word "$this". This is not an XQuery word, but for XQuery expressions in WebLogic Workshop it signifies "the current context in the XML instance document". Contrast this with XQuery maps, which use $input to indicate "the start of the XML instance document".

Also, notice that these expression are executed with an XmlCursor instance. Contrast this with using an instance of XmlObject (or a type that inherits from it), as described in Selecting XML with XQuery and XPath.

For more information about XQuery, see http://www.w3.org/TR/xquery/.

Concepts Demonstrated by this Sample

- Using XPath and XQuery with an XML cursor

Location of Sample Files

This sample is located in the xmlBeans/xquery folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\xmlBeans\xquery\SimpleExpressions

To Run the Sample

- Start WebLogic Server in the appropriate domain.
- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/xmlBeans/xquery/SimpleExpressions.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
- Navigate to the ***Test XML*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

Getting Started with XMLBeans

Selecting XML with XQuery and XPath

ThresholdService.jws Sample

A web service to use as a test client for the ThresholdEnums Java control. That control contains code that illustrates how you can use the schema type system to retrieve information about a compiled schema. This service simply retrieves the values of an enumeration defined in schema.

The schema type system API is a powerful way to get information about compiled schema. It is, in a sense, an API about your schema API. The SchemaTypeSystem object represents compiled schemas within reach of your code — for example, schemas in JAR files in your classpath.

Concepts Demonstrated by this Sample

- Getting information about compiled schemas through the schema type system.

Location of Sample Files

This sample is located in the xmlBeans/schema folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\xmlBeans\schema\ThresholdService.j

To Run the Sample

- Start WebLogic Server in the appropriate domain.
- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/xmlBeans/schema/ThresholdService.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
- Navigate to the ***Test XML*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

Getting Started with XMLBeans

SchemaTypeSystem Interface

TokenTypes.jws Sample

A web service illustrating how tokens correspond to portions of an XML document. A token represents a logical piece of XML, such as the start or end of an element, text, a comment, and so on. A token type, on the other hand, represents a kind of token. In other words, a token is a STARTDOC (the very beginning of an XML instance, before all markup), a START (the start of an element), a COMMENT, and so on.

When navigating XML with a cursor, you move the cursor past tokens using methods such as toNextToken, toNextElement, and so on. It's important to remember that a cursor is almost always immediately before some token. When at a START token, for example, the cursor is just before the start of an element. This also means that it is before the start of an element and after some other token. In other words, calling currentTokenType on an XmlCursor instance will usually return the token that it is immediately before. The exception to this rule is for an ENDDOC token, which is after all other markup. Because it is at the very end, it can't be before anything else.

Concepts Demonstrated by this Sample

- XML cursor navigation
- Using a cursor to alter XML
- Overview of XML tokens

Location of Sample Files

This sample is located in the xmlBeans/cursor folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\xmlBeans\cursor\TokenTypes.jws

To Run the Sample

- Start WebLogic Server in the appropriate domain.
- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/xmlBeans/cursor/TokenTypes.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
- Navigate to the ***Test XML*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

Getting Started with XMLBeans

Navigating XML with Cursors

XmlCursor Interface

TypeHierarchyPrinterService.jws Sample

A web service that prints a hierarchical rendering of the schema types represented by the schema type system available to its application. The logic for this sample is actually in TypeHierarchyPrinter.java, which this web service calls.

A schema type system is a set of Java types corresponding to XML schema types. When you compile schema into Java types, those Java types (along with the built−in types they use or inherit from) become part of a schema type system.

The type system used by this sample is available to this code because the types are included in a JAR file on this application's classpath. These types are used by the various schema−related samples in the SamplesApp application.

The unique "signatures" output by this sample are built from several characteristics, including:

- The type's position in the schema (for example, is it global or local?).
- Whether the type is a datatype (complex or simple) or an element or attribute.
- Whether the type is derived by restriction, union, and so on.

There isn't yet a public standard for such signatures, but the style used here is useful for seeing the hierarchy. Keep in mind that if a signature standard does become adopted, that standard will likely be used by the schema type system API. For more information, see Introduction to Schema Type Signatures.

Concepts Demonstrated by this Sample

- Getting information about compiled schemas through the schema type system
- Schema type system "signatures"

Location of Sample Files

This sample is located in the xmlBeans/schema folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\xmlBeans\schema\TypeHierarchyPri

To Run the Sample

- Start WebLogic Server in the appropriate domain.
- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/xmlBeans/schema/TypeHierarchyPrinterService.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this

machine, you may click here to run the sample.
- Navigate to the ***Test XML*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

Getting Started with XMLBeans

SchemaTypeSystem Interface

XsdConfig.jws Sample

A web service illustrating how you can use an XSDCONFIG file to guide naming translation when the schema compiler generates an API corresponding to your schema. The schema for the XML created by this code is EasyPOLocal.xsd, in the Schemas project of the SamplesApp application. Without any configuration, compiling that schema would result in the package and type names defined by that schema.

The EasyPOConfig.xsdconfig file included with the XSD file in the Schemas project tells the schema compiler how to name and package the types it generates. The XSDCONFIG file presents a one−to−one mapping between schema element and proposed API name. It also gives the compiler a package name to use instead of the namespace URI. A name such as "PurchaseOrder2" is the schema compiler's effort to avoid a name conflict. Here, there would be a conflict between the purchase order "document" type that allows you to add a new PURCH_ORDER element to the document, and the purchase order element that gives you access to its children.

Note that guiding the compiler−generated naming does not affect names and namespaces for the underlying XML. The createPO method of this web service returns the XML as it should be shaped according to the original schema.

Concepts Demonstrated by this Sample

- Guiding type naming with an XSDCONFIG file

Location of Sample Files

This sample is located in the xmlBeans/schema folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\xmlBeans\schema\XsdConfig.jws

To Run the Sample

- Start WebLogic Server in the appropriate domain.
- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/xmlBeans/schema/XsdConfig.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
- Navigate to the ***Test XML*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

Getting Started with XMLBeans

How Do I: Guide XMLBeans Type Naming During Schema Compilation?

XQuery Map Samples

The samples described in this section illustrate XQuery maps.

Topics Included in This Section

InputMapMultiple.jws Sample

A web service illustrating how you can use an XQuery map on a method whose parameter values are arrays.

OutputMap.jws Sample

A web service illustrating a simple return−XML XQuery map.

OutputScriptMap.jws Sample

Demonstrates use of XQuery maps in combination with ECMAScript to do special formatting and processing when tranforming Java objects to XML.

SimpleMap.jws Sample

A web service illustrating a simple parameter−XML map that uses XQuery.

Related Topics

Introduction to XQuery Maps

InputMapMultiple.jws Sample

A web service illustrating how you can use an XQuery map on a method whose parameter values are arrays.

The method in this web service accepts three arrays as parameters and returns a simple float. In order to map to the set of arrays, the XQuery map loops through the incoming XML message, extracting values and inserting them into the template represented by the map.

In general, you may find that while the XQuery expression in the source code looks complex, it resembles programming conventions with which you're already familiar, such as variables, loops, and return values.

Here are a few things to note about the XQuery in this sample's maps:

- An XQuery let clause binds variables (represented by $i, $n, and so on) to the XML resulting from a path to an item element.
- for clauses evaluate their expressions and iterate over the items in the resulting sequence, binding a variable to each item in turn. The bound variables are then used to insert values into the template of the outgoing message.
- return clauses return the result of the expressions that follow them.

The return−xml XQuery map includes a simple path expression that finds the result of the method and inserts it into the template for the outgoing message.

Concepts Demonstrated by this Sample

- parameter−xml and return−xml XQuery maps
- Creating variables in XQuery with the let keyword
- Constructing XML with XQuery and returning it with the return keyword
- Simple path expression in an XQuery map
- XQuery data function

Location of Sample Files

This sample is located in the xqueryMap folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\xqueryMap\InputMapMultiple.jws

To Run the Sample

- Start WebLogic Server in the appropriate domain.

- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/xqueryMap/InputMapMultiple.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
- Navigate to the ***Test XML*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

Introduction to XQuery Maps

How Do I: Add or Edit an XQuery Map with the XQuery Mapper Dialog?

OutputMap.jws Sample

A web service illustrating a simple return−XML XQuery map.

The method in this web service uses a simple XQuery map to ensure that the outgoing message carrying its return value conforms to a particular schema. The elements that make up the map's template (elements such as ns1:EMPLOYEE) match those defined by the schema associated with this map. XQuery expressions (such a $input/ns0:HelloResult/ns0:fname) select values from the default XML generated by WebLogic Workhop from the method's Java signature. At runtime, WebLogic Server runtime components execute the XQuery expressions against the default XML, then insert the resulting values into the template formed by the map.

Concepts Demonstrated by this Sample

- return−xml XQuery map
- Simple path expression in an XQuery map
- XQuery data function

Location of Sample Files

This sample is located in the xqueryMap folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\xqueryMap\OutputMap.jws

To Run the Sample

- Start WebLogic Server in the appropriate domain.
- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/xqueryMap/OutputMap.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
- Navigate to the ***Test XML*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

Introduction to XQuery Maps

How Do I: Add or Edit an XQuery Map with the XQuery Mapper Dialog?

OutputScriptMap.jws Sample

Demonstrates use of XQuery maps in combination with ECMAScript to do special formatting and processing when tranforming Java objects to XML.

You can call script in a JSX file from an XQuery map to incorporate logic that may not be possible in the XQuery itself. The method in this web service returns the XML response of this method as translated by a JSX file called from an XQuery map. The JSX file contains ECMAScript that formats into another shape the XML that would have otherwise been generated from the Person object this method returns.

Concepts Demonstrated by this Sample

- ECMAScript in a JSX file
- return−xml XQuery map
- Calling script from an XQuery map

Location of Sample Files

This sample is located in the xqueryMap folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\xqueryMap\OutputScriptMap.jws

To Run the Sample

- Start WebLogic Server in the appropriate domain.
- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/xqueryMap/OutputScriptMap.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
- Navigate to the ***Test XML*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

Handling XML with ECMAScript Extensions

Introduction to XQuery Maps

How Do I: Add or Edit an XQuery Map with the XQuery Mapper Dialog?

SimpleMap.jws Sample

A web service illustrating a simple parameter−XML map that uses XQuery.

The method in this web service uses a simple XQuery map to ensure that the parameter values in the incoming XML message are correctly mapped to the parameters themselves. The map itself (with its acceptPerson root element) reflects the method's default XML schema — a kind of default template for what an incoming message should look like. However, the actual incoming message will be different; for example, its root element is person, rather than acceptPerson.

Concepts Demonstrated by this Sample

- parameter−xml XQuery map
- XQuery data function

Location of Sample Files

This sample is located in the xqueryMap folder of the SamplesApp WebLogic Workshop project. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\WebServices\xqueryMap\SimpleMap.jws

To Run the Sample

- Start WebLogic Server in the appropriate domain.
- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Launch Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/domains/workshop/startWebLogic.sh

- Launch the service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/WebServices/xqueryMap/SimpleMap.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
- Navigate to the ***Test XML*** tab of Test View, if necessary, then follow the instructions for testing the sample in the comments provided.

Related Topics

Introduction to XQuery Maps

How Do I: Add or Edit an XQuery Map with the XQuery Mapper Dialog?

Portal Samples

The WebLogic Workshop Portal Extensions include a sample portal application that contains two sample portals: Sample Portal and Tutorial Portal. You can view these samples in the WebLogic Workshop development environment and in a browser. You can also reuse many of the portlets in your own portals.

This topic includes the following sections:

- Sample Portal
- Tutorial Portal
- Viewing the Samples
- Making Samples Available in the WebLogic Administration Portal

Sample Portal

The Sample Portal, Avitek Inweb, is an example business−to−enterprise portal that illustrates collaboration and other services. For details, see Sample Portal.

Tutorial Portal

The Tutorial Portal contains development samples of portal functionality, including Authentication, Controls, Look & Feel, Java Page Flows, Internationalization, Backing Files, Window Modes, Portlet Preferences, and Portlet Events. Samples contain documentation on how they were implemented.

WebLogic Portal Tutorial Samples

Viewing the Samples

This section shows you how to view the sample portlets and the tutorial samples.

Viewing the Sample Portal

To view the samples in a browser:

1. Start the platform samples server:
   <BEA_HOME>/<WEBLOGIC_HOME>/samples/domains/portal/startWebLogic.cmd(.sh).
2. Use the following URLs:

   http://<server>:<port>/sampleportal/sample.portal

   If the server is running on your machine, use ***http://localhost:7001/sampleportal/sample.portal***.

To open these samples in WebLogic Workshop Platform Edition:

1. Open the following application:

   <BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/portalApp.work.
2. In the Application window, double−click the sampleportal/sample.portal file to open it. In the

Document Structure window, or in the Portal Designer window, select components and look at their property values in the Property Editor window.

Viewing the Tutorial Portal

To view the samples in a browser:

1. Start the platform samples server: <BEA_HOME>/<WEBLOGIC_HOME>/samples/domains/portal/startWebLogic.cmd(.sh).
2. Use the following URLs:

   http://<server>:<port>/tutorial/tutorialPortal.portal

   If the server is running on your machine, use ***http://localhost:7001/tutorial/tutorialPortal.portal***.

To open these samples in WebLogic Workshop Platform Edition:

1. Open the following application:

   <BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/portalApp.work.
2. In the Application window, double−click the tutorial/tutorialPortal.portal file to open it. In the Document Structure window, or in the Portal Designer window, select components and look at their property values in the Property Editor window.

Making Samples Available in the WebLogic Administration Portal

For instructions on importing the sample portal into the WebLogic Administration Portal so administrators can modify portal resources, construct new portals, and set up visitor entitlements and delegated administration, see Deploying Portal Applications.

Related Topics

WebLogic Administration Portal Online Help

Sample Portal

The Sample Portal, Avitek Inweb, is an example business−to−enterprise portal that illustrates collaboration and other services. Sample Portal is a fully functional portal Web project that contains many portal resources you can reuse in your own portals. In particular, Sample Portal contains navigation examples and many portlets you can explore and reuse.

Sample Portlets

Each of the sample portlet topics includes instructions for using the sample in your own portals.

JSP Portlets

Login to Portal Portlet

Login Director Portlet

Targeted Menu Portlet

RSS News Feed Portlet

Portal Search Portlet

Java Page Flow Portlets

My Mail Portlet

My Task List Portlet

My Calendar Portlet

My Contacts Portlet

Discussion Forums Portlet

Discussion Forum Administration Portlet

My Content Portlet

HTML Portlets

Dev2Dev Portlet

Navigation Samples

Left Navigation Shell

Targeted Menu Portlet

Login to Portal Portlet

The Login to Portal portlet illustrates login functionality for a portal desktop.

Concepts Demonstrated by this Sample

The JSP portlet uses a backing file to authenticate users.

Location of Sample Files

This sample is located in the `<BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/portalApp.work` application.

How to Run the Sample

See Viewing the Samples in Portal Samples.

How to Use the Sample in Your Portals

1. Create a portal application. Make sure you create a Portal Application and add a Portal Web Project to it.
2. Import or copy the following directories and files into your portal application and portal Web project. You may need to create the appropriate directories in your application:

Import or copy this	***to this directory*** (creat
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/includes/Login.portlet`	<PORTAL_APP>/<pr
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/login.jsp`	<PORTAL_APP>/<pr
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/WEB-INF/src/examples/login/LoginBacking.java`	<PORTAL_APP>/<pr WEB-INF/src/exampl
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/WEB-INF/lib/yahoo_servlet.jar`	<PORTAL_APP>/<pr WEB-INF/lib/

3. Open your portal file and navigate the page where you want the portlet to appear.
4. In the ***Data Palette*** window, drag the ***Login to Portal*** portlet onto a placeholder on the page.
5. In the ***Property Editor*** window, set any relevant properties.
6. Save the portal file.
7. View your portal with the WebLogic Test Browser or with your default browser.
   - ***WebLogic Test Browser*** − In the WebLogic Workshop toolbar, click the ***Start*** button (or press ***Ctrl+F5***).
   - ***Default Browser*** − In the WebLogic Workshop menu, choose ***Portal−−>Open Current Portal***.

Related Topics

Creating a Portal File

Portal Samples

Login Director Portlet

This login portlet directs the user to a "default" desktop the first desktop to which the user is entitled and returns the user to that desktop when the user logs out.

Concepts Demonstrated by this Sample

The JSP portlet uses a backing file to authenticate users and direct them to the default desktop.

Location of Sample Files

This sample is located in the `<BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/portalApp.work` application.

How to Run the Sample

While you can run the portlet in the development environment, the full functionality of this portlet is visible only in streaming mode where multiple desktops and visitor entitlement rules exist. For more information, see Single File vs. Streamed Rendering.

To run the portlet in the development environment, see Viewing the Samples in Portal Samples.

To run the portlet in streaming mode, add the portlet to a page in sample.portal, then use the WebLogic Administration Portal to create a new desktop that uses sample.portal as a template. On the Desktop Properties page for the new desktop, click ***View Desktop*** to view the portlet.

How to Use the Sample in Your Portals

1. Create a portal application. Make sure you create a Portal Application and add a Portal Web Project to it.
2. Import or copy the following directories and files into your portal application and portal Web project. You may need to create the appropriate directories in your application:

Import or copy this	***to this directory*** (c
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/login/director.portlet`	<PORTAL_APP>
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/login/director.jsp`	<PORTAL_APP>
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/WEB-INF/src/examples/login/DirectorBacking.java`	<PORTAL_APP> WEB-INF/src/exa
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/WEB-INF/src/examples/login/DirectorUtil.java`	<PORTAL_APP> WEB-INF/src/exa
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/WEB-INF/lib/yahoo_servlet.jar`	<PORTAL_APP> WEB-INF/lib/

3. Open your portal file and navigate the page where you want the portlet to appear.
4. In the ***Data Palette*** window, drag the ***Login Director*** portlet onto a placeholder on the page.
5. In the ***Property Editor*** window, set any relevant properties.

6. Save the portal file.
7. To view the portlet in a desktop, create a desktop in the WebLogic Administration Portal using the current .portal file as a template, and in the Desktop Properties page for the desktop, click ***View Desktop***.

Related Topics

Portal Samples

Login Portlet

Targeted Menu Portlet

This portlet demonstrates using navigation from a portlet to control a specific book.

Concepts Demonstrated by this Sample

The JSP portlet provides portlet−based navigation by acquiring a book's context, sub−books, and pages and constructing a tree−like list of links to the sub−books and pages.

This portlet uses functionality similar to the Left Navigation Shell. The difference between the two samples is scope. This portlet provides navigation for a single book, and the left navigation shell provides navigation for the entire desktop.

Location of Sample Files

This sample is located in the `<BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/portalApp.work` application.

How to Run the Sample

To run the sample:

1. Add the portlet to a page in sample.portal.
2. In the Portal Designer, select the book for which you want to provide navigation. In the Property Editor window, note the value for the book's Definition Label property.
3. Open the portlet in the Portlet Designer, and select the ***TargetBook*** portlet preference.
4. In the Property Editor window, set the ***Preference Value*** property value to the book's Definition Label.
5. Save the portlet and portal files.
6. In the WebLogic Workshop menu, choose ***Portal −−> Open Current Portal***.

How to Use the Sample in Your Portals

1. Create a portal application. Make sure you create a Portal Application and add a Portal Web Project to it.
2. Import or copy the following directories and files into your portal application and portal Web project. You may need to create the appropriate directories in your application:

Import or copy this	***to this direc***
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/navigation/targeted/targetedMenu.portlet`	<PORTAL_
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/navigation/targeted/targetedMenu.jsp`	<PORTAL_
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/WEB−INF/src/examples/navigation/NavigationNode.java`	<PORTAL_ WEB−INF/s

<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/WEB-INF/src/examples/navigation/NavigationUtil.java	<PORTAL_ WEB-INF/s

3. Open your portal file and navigate the page where you want the portlet to appear.
4. In the ***Data Palette*** window, drag the ***Targeted Menu*** portlet onto a placeholder on the page.
5. In the Portal Designer, select the book for which you want to provide navigation. In the Property Editor window, note the value for the book's Definition Label property.
6. Open the portlet in the Portlet Designer, and select the ***TargetBook*** portlet preference.
7. In the Property Editor window, set the ***Preference Value*** property value to the book's Definition Label.
8. Save the portlet and portal files.
9. In the WebLogic Workshop menu, choose ***Portal --> Open Current Portal***.

Related Topics

Left Navigation Shell

Portal Samples

dev2dev Portlet

The dev2dev portlet illustrates a portlet that uses static HTML links for accessing valuable BEA resources and information.

Concepts Demonstrated by this Sample

This HTML portlet illustrates that portlets can be created with only HTML tags. No knowledge of JSP development is necessary. Simply give an HTML file a .jsp extension and create a portlet with it.

The file does not need <HTML>, <HEAD>, <TITLE>, or <BODY> tags. Simply enclose your HTML content in opening and closing tags.

Location of Sample Files

This sample is located in the `<BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/portalApp.work` application.

How to Run the Sample

See Viewing the Samples in Portal Samples.

How to Use the Sample in Your Portals

1. Create a portal application.
2. Import or copy the following directories and files into your portal application and portal Web project. You may need to create the appropriate directories in your application:

Import or copy this	***to this directory*** (create if necessary)
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/includes/dev2dev.portlet`	<PORTAL_APP>/<project>/portlets/in
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/dev2dev/`	<PORTAL_APP>/<project>/portlets/

3. Open your portal file and navigate the page where you want the portlet to appear.
4. In the ***Data Palette*** window, drag the ***Dev2Dev*** portlet onto a placeholder on the page.
5. In the ***Property Editor*** window, set any relevant properties.
6. Save the portal file.
7. View your portal with the WebLogic Test Browser or with your default browser.

    - ***WebLogic Test Browser*** − In the WebLogic Workshop toolbar, click the ***Start*** button (or press ***Ctrl+F5***).
    - ***Default Browser*** − In the WebLogic Workshop menu, choose ***Portal−−>Open Current Portal***.

Related Topics

Creating a Portal File

Portal Samples

RSS News Feed Portlet

The RSS News Feed Portlet retrieves news content and links based on visitor news feed preferences. This portlet provides an edit mode that lets you change the news feed and set portlet preferences.

Concepts Demonstrated by this Sample

This JSP portlet, which uses a backing file, illustrates a news aggregator portlet that connects to external news feeds and weblogs that provide Really Simple Syndication (RSS) content. This portlet provides an edit mode.

Location of Sample Files

This sample is located in the `<BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/portalApp.work` application.

How to Run the Sample

See Viewing the Samples in Portal Samples.

How to Use the Sample in Your Portals

1. Create a portal application.
2. Import or copy the following directories and files into your portal application and portal Web project. You may need to create the appropriate directories in your application:

Import or copy this	***to this directory*** (create if ne
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/rss/`	<PORTAL_APP>/<project>
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/WEB-INF/src/examples/rss/`	<PORTAL_APP>/<project> WEB-INF/src/examples/

`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/WEB-INF/`lib/xmlx-tags.jar
 	
Add the following entry in the tag library section of <project>/WEB-INF/web.xml to register the tag library:
 	
`<taglib>`
`  <taglib-uri>xmlx.tld</taglib-uri>`
`  <taglib-location>/WEB-INF/lib/xmlx-tags.jar</taglib-location>`
`</taglib>`
	<PORTAL_APP>/<project> WEB-INF/lib/

3. Open your portal file and navigate the page where you want the portlet to appear.
4. In the ***Data Palette*** window, drag the ***RSS News Feed*** portlet onto the portal page.
5. In the ***Property Editor*** window, set any relevant properties.
6. Save the portal file.
7. View your portal with the WebLogic Test Browser or with your default browser.
   - ***WebLogic Test Browser*** − In the WebLogic Workshop toolbar, click the ***Start*** button (or press ***Ctrl+F5***).

- ***Default Browser*** − In the WebLogic Workshop menu, choose ***Portal−−>Open Current Portal***.

To Change the News Feed

1. In the ***Portal Designer***, double−click the portlet to open it.
2. With the portlet file open, click the arrow icon on the Portlet Preferences bar on the portlet footer to expand the preferences.
3. Select the ***contentURL*** preference.
4. In the ***Property Editor*** window, enter an absolute HTTP path to a valid news feed (.rdf file) in the ***Preference Value*** field. For example: http://www.theserverside.com/rss/theserverside−1.0.rdf.

You can also let users change news feeds for a portlet with the portlet's edit mode.

The portlet's ***Edit*** mode uses the rss.properties file to let you select from a list of valid RSS feeds. Modify rss.properties to remove or add feeds. The ***rss*** portlet shows one feed at a time. To show multiple feeds, add more ***rss*** portlets and set a different ***contentURL*** for each. Set the ***Edit URI*** field to /portlets/rss/rssedit.jsp to allow users to change the portlet feed using the rss.properties list. Save the portal file.
Related Topics

Creating a Portal File

Portal Samples

<pref:getPreference> Tag

<pref:ifModifiable> Tag

<pref:else> Tag

Portal Search Portlet

The Portal Search portlet lets you perform searches in your enterprise databases. This portlet also provides an edit mode to let you set search preferences, including selecting available databases to search.

Concepts Demonstrated by this Sample

This is a JSP portlet that provides edit mode.

Location of Sample Files

This sample is located in the `<BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/portalApp.work` application.

How to Run the Sample

See Viewing the Samples in Portal Samples.

How to Use the Sample in Your Portals

1. Create a portal application.
2. Import or copy the following directories and files into your portal application and portal Web project. You may need to create the appropriate directories in your application:

Import or copy this	***to this directory*** (create if necessary)
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/includes/search.portlet`	<PORTAL_APP>/<project>/portlets/inc
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/search/`	<PORTAL_APP>/<project>/portlets/
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/WEB-INF/lib/autonomyClient1.5.0.jar and autonomySupport.jar`	<PORTAL_APP>/<project>/WEB-INF

3. Open your portal file and navigate the page where you want the portlet to appear.
4. In the ***Data Palette*** window, drag the ***Portal Search*** portlet onto the portal page.
5. In the ***Property Editor*** window, set any relevant properties.
6. Save the portal file.
7. View your portal with the WebLogic Test Browser or with your default browser.
   - ***WebLogic Test Browser*** − In the WebLogic Workshop toolbar, click the ***Start*** button (or press ***Ctrl+F5***).
   - ***Default Browser*** − In the WebLogic Workshop menu, choose ***Portal−−>Open Current Portal***.

Related Topics

Creating a Portal File

Portal Samples

My Mail Portlet

The My Mail portlet lets you configure an e−mail account that uses a POP3 or IMAP standard e−mail protocol. Once your e−mail account is configured, you can send, receive, store, and delete e−mail.

Concepts Demonstrated by this Sample

This Java Page Flow portlet illustrates group collaboration functionality within a portlet. The portlet provides edit mode.

Location of Sample Files

This sample is located in the `<BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/portalApp.work` application.

How to Run the Sample

See Viewing the Samples in Portal Samples.

How to Use the Sample in Your Portals

When this portlet is used in a domain (for example, in the portalApp in the Sample Portal Domain), the EJBs it uses are registered with JNDI names that can be used only once in the domain. That means you can use the following collaboration portlets in only one portal application in a domain: My Mail, My Task List, My Calendar, My Contacts, Discussion Forums, and Discussion Forum Administration. Within that portal application you can create multiple portal Web projects that can each contain multiple portals that reuse these portlets.

1. Create a portal application in a domain that has not used the collaboration portlets.
2. Make sure your portal application is open and the server is running (***Tools−−>WebLogic Server−−>Start WebLogic Server***).
3. Import/add the following directories and files into your portal application and portal Web project using WebLogic Workshop. (Right−click−−>***Import*** or ***Add Module*** or ***Add Library*** on the target directory). You may need to create the appropriate directories in your application.

   Be sure to add the harmony_portlets.jar library first, as shown in the following table.

Import this	***into this WebLogic Workshop directory*** (create if necessary)
`<WEBLOGIC_HOME>/samples/portal/portalApp/APP−INF/lib/harmony_portlets.jar`	<PORTAL_APP>/Libraries/
`<WEBLOGIC_HOME>/samples/portal/portalApp/security_ejb.jar uniqueid_ejb.jar`	<PORTAL_APP>/Modules/
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/includes/collaboration/native_mail.portlet`	<PORTAL_APP>/<project>/portlets/includes/collaboration

<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/collaboration/nativedb/mail/	<PORTAL_APP>/<project>/collaboration/native

Note: If you add the non−JAR resources to directories other than those shown, you must open the portlet file in WebLogic Workshop and edit the Content URI for both the portlet's main content and Edit page content; you must modify the package path and <view−properties> paths in the Java Page Flow files; and you must modify the import statement to the ContentController and modify any other relevant paths in the JSPs.

4. Add the collaboration tables to your database. If you have already performed this for another collaboration portlet, skip this step.

   Note: If you ran the create_* database script to set up your database, the collaboration tables already exist. Do not run create_* if, for example, you are using the default PointBase database and have already added records to the database. Follow these instructions instead.

   To add the collaboration tables to an existing database, run the following database script:

   <BEA_HOME>/<WEBLOGIC_HOME>/portal/db/<DB_TYPE>/<DB_VERSION>/collaboration_create_tabl

   for example

   bea/weblogic81/portal/db/pointbase/44/collaboration_create_tables.sql
5. To run this script for PointBase:
   a. Start the PointBase Console. In a command window, run

      <DOMAIN>/startPointBaseConsole.cmd(.sh)
   b. Log into the console. The default login is weblogic/weblogic.
   c. Choose ***File−−>Open***.
   d. Open the collaboration_create_tables.sql script. The script opens in the Enter SQL Commands window.
   e. Click the ***Execute All*** button. The collaboration tables are created.
   f. Close the PointBase Console.
6. Add entries to <PORTAL_APP>/<PROJECT>/WEB−INF/web.xml.

   a. Open <PORTAL_APP>/<PROJECT>/WEB−INF/web.xml.
   b. Open <BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/WEB−INF/web.xml
   c. From the sampleportal web.xml file, copy the following sections into your project web.xml file in the appropriate locations:

```
<!−− Compoze Collaboration Mail Attachment Servlet −−>
<servlet>
                <servlet−name>CompozeNativeMailFileAttachmentServlet</servl
                <servlet−class>com.compoze.mail.FileAttachmentServlet</serv
</servlet>
.
.
.
<!−− Compoze Collaboration Mail Attachment Servlet Mapping −−>
<servlet−mapping>
                <servlet−name>CompozeNativeMailFileAttachmentServlet</servl
                <url−pattern>*.compozenativemailfileattachmentservlet</url−
</servlet−mapping>
```

```
.
.
.
<ejb-ref>
    <description>Unique ID Generator</description>
    <ejb-ref-name>com.compoze.ejb.uniqueid.IUniqueIDGeneratorHome</ejb-ref-
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.compoze.ejb.uniqueid.IUniqueIDGeneratorHome</home>
    <remote>com.compoze.ejb.uniqueid.IUniqueIDGenerator</remote>
    <ejb-link>UniqueIDGenerator</ejb-link>
</ejb-ref>
<ejb-ref>
    <description>Access Control Manager</description>
    <ejb-ref-name>com.compoze.security.acl.IAccessControllerHome</ejb-ref-na
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.compoze.security.acl.IAccessControllerHome</home>
    <remote>com.compoze.security.acl.IAccessController</remote>
    <ejb-link>AccessController</ejb-link>
</ejb-ref>
```

d. Save and close your project web.xml file.

9. Perform the following steps only if you are using a database other than PointBase.
   a. Stop the server. Choose ***Tools−−>WebLogic Server−−>Stop WebLogic Server***.
   b. Modify your domain's setDomainEnv.cmd(.sh) to use the correct database.

      In the entry set HARMONY_PORTLETS_PROPERTIES=−Dejbruntime.database=pointbase44, use the commented area above this entry to replace the pointbase44 entry with the name of your database driver. Possible values are listed above that entry.
   c. Save setDomainEnv.cmd(.sh).
   d. Restart the server.

10. Add the Login to Portal Portlet to your portal Web project. Users must log in to use the collaboration portlets.
11. Open your portal file and navigate the page where you want the portlet to appear.
12. In the ***Data Palette*** window, drag the ***My Mail*** portlet onto the portal page.
13. In the ***Property Editor*** window, set any relevant properties.
14. Save the portal file.
15. View your portal with the WebLogic Test Browser or with your default browser.
    - ***WebLogic Test Browser*** − In the WebLogic Workshop toolbar, click the ***Start*** button (or press ***Ctrl+F5***).
    - ***Default Browser*** − In the WebLogic Workshop menu, choose ***Portal−−>Open Current Portal***.

For instructions on using the portlet's features, see *Compoze Portlets for BEA WebLogic Portal User's Guide* at http://e−docs.bea.com/wlp/docs81/pdf/compoze_portlets_users_guide.pdf.

Related Topics

Creating a Portal File

Portal Samples

Getting Started with Page Flows

My Task List Portlet

The My Task List portlet lets you create a To Do list, set dates, status, priorities, completion percentages, and other details on tasks. This portlet also provides an edit mode to let you customize your task views.

Concepts Demonstrated by this Sample

This Java Page Flow portlet illustrates group collaboration functionality within a portlet. The portlet provides edit mode.

Location of Sample Files

This sample is located in the `<BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/portalApp.work` application.

How to Run the Sample

See Viewing the Samples in Portal Samples.

How to Use the Sample in Your Portals

When this portlet is used in a domain (for example, in the portalApp in the Sample Portal Domain), the EJBs it uses are registered with JNDI names that can be used only once in the domain. That means you can use the following collaboration portlets in only one portal application in a domain: My Mail, My Task List, My Calendar, My Contacts, Discussion Forums, and Discussion Forum Administration. Within that portal application you can create multiple portal Web projects that can each contain multiple portals that reuse these portlets.

1. Create a portal application in a domain that has not used the collaboration portlets.
2. Make sure your portal application is open and the server is running (***Tools−−>WebLogic Server−−>Start WebLogic Server***).
3. Import/add the following directories and files into your portal application and portal Web project using WebLogic Workshop. (Right−click−−>***Import*** or ***Add Module*** or ***Add Library*** on the target directory). You may need to create the appropriate directories in your application.

   Be sure to add the harmony_portlets.jar library first, as shown in the following table.

Import this	***into this WebLogic Workshop directory*** (create if necessary)
`<WEBLOGIC_HOME>/samples/portal/portalApp/APP−INF/lib/harmony_portlets.jar`	<PORTAL_APP>/Libraries/

`<WEBLOGIC_HOME>/samples/portal/portalApp/`
`todo_ejb.jar`
`security_ejb.jar`
`uniqueid_ejb.jar`
	<PORTAL_APP>/Modules/

`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/includes/collaboration/`	<PORTAL_APP>/<project>/portlets/includes/collaboration

native_task.portlet	
<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/collaboration/nativedb/todo/	<PORTAL_APP>/<project>/collaboration/native

Note: If you add the non−JAR resources to directories other than those shown, you must open the portlet file in WebLogic Workshop and edit the Content URI for both the portlet's main content and Edit page content; you must modify the package path and <view−properties> paths in the Java Page Flow files; and you must modify the import statement to the ContentController and modify any other relevant paths in the JSPs.

4. Add the collaboration tables to your database. If you have already performed this for another collaboration portlet, skip this step.

   Note: If you ran the create_* database script to set up your database, the collaboration tables already exist. Do not run create_* if, for example, you are using the default PointBase database and have already added records to the database. Follow these instructions instead.

   To add the collaboration tables to an existing database, run the following database script:

   <BEA_HOME>/<WEBLOGIC_HOME>/portal/db/<DB_TYPE>/<DB_VERSION>/collaboration_create_tabl

   for example

   bea/weblogic81/portal/db/pointbase/44/collaboration_create_tables.sql
5. To run this script for PointBase:
   a. Start the PointBase Console. In a command window, run

      <DOMAIN>/startPointBaseConsole.cmd(.sh)
   b. Log into the console. The default login is weblogic/weblogic.
   c. Choose ***File−−>Open***.
   d. Open the collaboration_create_tables.sql script. The script opens in the Enter SQL Commands window.
   e. Click the ***Execute All*** button. The collaboration tables are created.
   f. Close the PointBase Console.
6. Add entries to <PORTAL_APP>/<PROJECT>/WEB−INF/web.xml.
   a. Open <PORTAL_APP>/<PROJECT>/WEB−INF/web.xml.
   b. Open <BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/WEB−INF/web.xml
   c. From the sampleportal web.xml file, copy the following sections into your project web.xml file in the appropriate locations:

```
<ejb−ref>
    <description>Unique ID Generator</description>
    <ejb−ref−name>com.compoze.ejb.uniqueid.IUniqueIDGeneratorHome</ejb−ref−
    <ejb−ref−type>Session</ejb−ref−type>
    <home>com.compoze.ejb.uniqueid.IUniqueIDGeneratorHome</home>
    <remote>com.compoze.ejb.uniqueid.IUniqueIDGenerator</remote>
    <ejb−link>UniqueIDGenerator</ejb−link>
</ejb−ref>
<ejb−ref>
    <description>Access Control Manager</description>
    <ejb−ref−name>com.compoze.security.acl.IAccessControllerHome</ejb−ref−na
    <ejb−ref−type>Session</ejb−ref−type>
    <home>com.compoze.security.acl.IAccessControllerHome</home>
```

```
        <remote>com.compoze.security.acl.IAccessController</remote>
        <ejb-link>AccessController</ejb-link>
    </ejb-ref>
    <ejb-ref>
        <description>To Do Manager</description>
        <ejb-ref-name>com.compoze.todo.ejb.IToDoManagerHome</ejb-ref-name>
        <ejb-ref-type>Session</ejb-ref-type>
        <home>com.compoze.todo.ejb.IToDoManagerHome</home>
        <remote>com.compoze.todo.ejb.IToDoManager</remote>
        <ejb-link>ToDoManager</ejb-link>
    </ejb-ref>
```

d. Save and close your project web.xml file.

7. Perform the following steps only if you are using a database other than PointBase.
   a. Stop the server. Choose ***Tools−−>WebLogic Server−−>Stop WebLogic Server***.
   b. Modify your domain's setDomainEnv.cmd(.sh) to use the correct database.

      In the entry set HARMONY_PORTLETS_PROPERTIES=−Dejbruntime.database=pointbase44, use the commented area above this entry to replace the pointbase44 entry with the name of your database driver. Possible values are listed above that entry.
   c. Save setDomainEnv.cmd(.sh).
   d. Restart the server.
8. Add the Login to Portal Portlet to your portal Web project. Users must log in to use the collaboration portlets.
9. Open your portal file and navigate the page where you want the portlet to appear.
10. In the ***Data Palette*** window, drag the ***My Task List*** portlet onto the portal page.
11. In the ***Property Editor*** window, set any relevant properties.
12. Save the portal file.
13. View your portal with the WebLogic Test Browser or with your default browser.
    - ***WebLogic Test Browser*** − In the WebLogic Workshop toolbar, click the ***Start*** button (or press ***Ctrl+F5***).
    - ***Default Browser*** − In the WebLogic Workshop menu, choose ***Portal−−>Open Current Portal***.

For instructions on using the portlet's features, see *Compoze Portlets for BEA WebLogic Portal User's Guide* at http://e−docs.bea.com/wlp/docs81/pdf/compoze_portlets_users_guide.pdf.

Related Topics

Creating a Portal File

Portal Samples

Getting Started with Page Flows

My Calendar Portlet

The My Calendar portlet provides a full−featured calendar system to let you manage and configure appointments and reminders. The portlet also provides an edit mode to let you set calendar preferences and options.

Concepts Demonstrated by this Sample

This Java Page Flow portlet illustrates group collaboration functionality within a portlet. The portlet provides edit mode.

Location of Sample Files

This sample is located in the `<BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/portalApp.work` application.

How to Run the Sample

See Viewing the Samples in Portal Samples.

How to Use the Sample in Your Portals

When this portlet is used in a domain (for example, in the portalApp in the Sample Portal Domain), the EJBs it uses are registered with JNDI names that can be used only once in the domain. That means you can use the following collaboration portlets in only one portal application in a domain: My Mail, My Task List, My Calendar, My Contacts, Discussion Forums, and Discussion Forum Administration. Within that portal application you can create multiple portal Web projects that can each contain multiple portals that reuse these portlets.

1. Create a portal application in a domain that has not used the collaboration portlets.
2. Make sure your portal application is open and the server is running (***Tools−−>WebLogic Server−−>Start WebLogic Server***).
3. Import/add the following directories and files into your portal application and portal Web project using WebLogic Workshop. (Right−click−−>***Import*** or ***Add Module*** or ***Add Library*** on the target directory). You may need to create the appropriate directories in your application.

   Be sure to add the harmony_portlets.jar library first, as shown in the following table.

Import this	***into this WebLogic Workshop directory*** (create if necessary)
`<WEBLOGIC_HOME>/samples/portal/portalApp/APP−INF/lib/harmony_portlets.jar`	<PORTAL_APP>/Libraries/

`<WEBLOGIC_HOME>/samples/portal/portalApp/`
`calendar_ejb.jar`
`security_ejb.jar`
`uniqueid_ejb.jar`
	<PORTAL_APP>/Modules/

`<WEBLOGIC_HOME>/samples/portal/portalApp/`	<PORTAL_APP>/<project>/

sampleportal/portlets/includes/collaboration/native_calendar.portlet	portlets/includes/collaboration
<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/collaboration/nativedb/calendar/	<PORTAL_APP>/<project>/p collaboration/native

Note: If you add the non−JAR resources to directories other than those shown, you must open the portlet file in WebLogic Workshop and edit the Content URI for both the portlet's main content and Edit page content; you must modify the package path and <view−properties> paths in the Java Page Flow files; and you must modify the import statement to the ContentController and modify any other relevant paths in the JSPs.

6. Add the collaboration tables to your database. If you have already performed this for another collaboration portlet, skip this step.

   Note: If you ran the create_* database script to set up your database, the collaboration tables already exist. Do not run create_* if, for example, you are using the default PointBase database and have already added records to the database. Follow these instructions instead.

   To add the collaboration tables to an existing database, run the following database script:

   <BEA_HOME>/<WEBLOGIC_HOME>/portal/db/<DB_TYPE>/<DB_VERSION>/collaboration_create_tabl

   for example

   bea/weblogic81/portal/db/pointbase/44/collaboration_create_tables.sql

   To run this script for PointBase:

   a. Start the PointBase Console. In a command window, run

      <DOMAIN>/startPointBaseConsole.cmd(.sh)
   b. Log into the console. The default login is weblogic/weblogic.
   c. Choose ***File−−>Open***.
   d. Open the collaboration_create_tables.sql script. The script opens in the Enter SQL Commands window.
   e. Click the ***Execute All*** button. The collaboration tables are created.
   f. Close the PointBase Console.

7. Add entries to <PORTAL_APP>/<PROJECT>/WEB−INF/web.xml.
   a. Open <PORTAL_APP>/<PROJECT>/WEB−INF/web.xml.
   b. Open <BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/WEB−INF/web.xml
   c. From the sampleportal web.xml file, copy the following sections into your project web.xml file in the appropriate locations:

```
<!−−  Compoze Collaboration V Calendar Servlet −−>
<servlet>
    <servlet−name>CompozeNativeCalendarVCalendarServlet</servlet−name>
    <servlet−class>com.compoze.calendar.AppointmentVCalendarServlet</servle
</servlet>
```

```
.
.
.
<!--  Compoze Collaboration V Calendar Servlet Mapping -->
<servlet-mapping>
    <servlet-name>CompozeNativeCalendarVCalendarServlet</servlet-name>
    <url-pattern>*.compozevcalendarservlet</url-pattern>
</servlet-mapping>
.
.
.
<ejb-ref>
    <description>Unique ID Generator</description>
    <ejb-ref-name>com.compoze.ejb.uniqueid.IUniqueIDGeneratorHome</ejb-ref-
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.compoze.ejb.uniqueid.IUniqueIDGeneratorHome</home>
    <remote>com.compoze.ejb.uniqueid.IUniqueIDGenerator</remote>
    <ejb-link>UniqueIDGenerator</ejb-link>
</ejb-ref>
<ejb-ref>
    <description>Access Control Manager</description>
    <ejb-ref-name>com.compoze.security.acl.IAccessControllerHome</ejb-ref-na
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.compoze.security.acl.IAccessControllerHome</home>
    <remote>com.compoze.security.acl.IAccessController</remote>
    <ejb-link>AccessController</ejb-link>
</ejb-ref>
<ejb-ref>
    <description>Calendar Manager</description>
    <ejb-ref-name>com.compoze.calendar.ejb.ICalendarManagerHome</ejb-ref-nam
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.compoze.calendar.ejb.ICalendarManagerHome</home>
    <remote>com.compoze.calendar.ejb.ICalendarManager</remote>
    <ejb-link>CalendarManager</ejb-link>
</ejb-ref>
```

d. Save and close your project web.xml file.

8. Perform the following steps only if you are using a database other than PointBase.
   a. Stop the server. Choose ***Tools-->WebLogic Server-->Stop WebLogic Server***.
   b. Modify your domain's setDomainEnv.cmd(.sh) to use the correct database.

      In the entry set HARMONY_PORTLETS_PROPERTIES=-Dejbruntime.database=pointbase44, use the commented area above this entry to replace the pointbase44 entry with the name of your database driver. Possible values are listed above that entry.
   c. Save setDomainEnv.cmd(.sh).
   d. Restart the server.

9. Add the Login to Portal Portlet to your portal Web project. Users must log in to use the collaboration portlets.
10. Open your portal file and navigate the page where you want the portlet to appear.
11. In the ***Data Palette*** window, drag the ***My Calendar*** portlet onto the portal page.
12. In the ***Property Editor*** window, set any relevant properties.
13. Save the portal file.
14. View your portal with the WebLogic Test Browser or with your default browser.
    - ***WebLogic Test Browser*** - In the WebLogic Workshop toolbar, click the ***Start*** button (or press ***Ctrl+F5***).

- ***Default Browser*** − In the WebLogic Workshop menu, choose ***Portal−−>Open Current Portal***.

For instructions on using the portlet's features, see *Compoze Portlets for BEA WebLogic Portal User's Guide* at http://e−docs.bea.com/wlp/docs81/pdf/compoze_portlets_users_guide.pdf.

Related Topics

Creating a Portal File

Portal Samples

Getting Started with Page Flows

My Contacts Portlet

The My Contacts portlet provides full−featured address book management. The portlet also provides an edit mode to let you set contact management preferences.

Concepts Demonstrated by this Sample

This Java Page Flow portlet illustrates group collaboration functionality within a portlet. The portlet provides edit mode.

Location of Sample Files

This sample is located in the `<BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/portalApp.work` application.

How to Run the Sample

See Viewing the Samples in Portal Samples.

How to Use the Sample in Your Portals

When this portlet is used in a domain (for example, in the portalApp in the Sample Portal Domain), the EJBs it uses are registered with JNDI names that can be used only once in the domain. That means you can use the following collaboration portlets in only one portal application in a domain: My Mail, My Task List, My Calendar, My Contacts, Discussion Forums, and Discussion Forum Administration. Within that portal application you can create multiple portal Web projects that can each contain multiple portals that reuse these portlets.

1. Create a portal application in a domain that has not used the collaboration portlets.
2. Make sure your portal application is open and the server is running (***Tools−−>WebLogic Server−−>Start WebLogic Server***).
3. Import/add the following directories and files into your portal application and portal Web project using WebLogic Workshop. (Right−click−−>***Import*** or ***Add Module*** or ***Add Library*** on the target directory). You may need to create the appropriate directories in your application.

   Be sure to add the harmony_portlets.jar library first, as shown in the following table.

Import this	***into this WebLogic Workshop directory*** (create if necessary)
`<WEBLOGIC_HOME>/samples/portal/portalApp/APP−INF/lib/harmony_portlets.jar`	<PORTAL_APP>/Libraries/

`<WEBLOGIC_HOME>/samples/portal/portalApp/`
`contact_ejb.jar`
`security_ejb.jar`
`uniqueid_ejb.jar`
	<PORTAL_APP>/Modules/

`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/includes/collaboration/`	<PORTAL_APP>/<project>/portlets/includes/collaboration

native_contact.portlet	
<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/collaboration/nativedb/contact/	<PORTAL_APP>/<project>/collaboration/native

Note: If you add the non−JAR resources to directories other than those shown, you must open the portlet file in WebLogic Workshop and edit the Content URI for both the portlet's main content and Edit page content; you must modify the package path and <view−properties> paths in the Java Page Flow files; and you must modify the import statement to the ContentController and modify any other relevant paths in the JSPs.

4. Add the collaboration tables to your database. If you have already performed this for another collaboration portlet, skip this step.

   Note: If you ran the create_* database script to set up your database, the collaboration tables already exist. Do not run create_* if, for example, you are using the default PointBase database and have already added records to the database. Follow these instructions instead.

   To add the collaboration tables to an existing database, run the following database script:

   <BEA_HOME>/<WEBLOGIC_HOME>/portal/db/<DB_TYPE>/<DB_VERSION>/collaboration_create_tabl

   for example

   bea/weblogic81/portal/db/pointbase/44/collaboration_create_tables.sql
5. To run this script for PointBase:
   a. Start the PointBase Console. In a command window, run

      <DOMAIN>/startPointBaseConsole.cmd(.sh)
   b. Log into the console. The default login is weblogic/weblogic.
   c. Choose ***File−−>Open***.
   d. Open the collaboration_create_tables.sql script. The script opens in the Enter SQL Commands window.
   e. Click the ***Execute All*** button. The collaboration tables are created.
   f. Close the PointBase Console.
6. Add entries to <PORTAL_APP>/<PROJECT>/WEB−INF/web.xml.
   a. Open <PORTAL_APP>/<PROJECT>/WEB−INF/web.xml.
   b. Open <BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/WEB−INF/web.xml
   c. From the sampleportal web.xml file, copy the following sections into your project web.xml file in the appropriate locations:

```
<ejb−ref>
    <description>Unique ID Generator</description>
    <ejb−ref−name>com.compoze.ejb.uniqueid.IUniqueIDGeneratorHome</ejb−ref−
    <ejb−ref−type>Session</ejb−ref−type>
    <home>com.compoze.ejb.uniqueid.IUniqueIDGeneratorHome</home>
    <remote>com.compoze.ejb.uniqueid.IUniqueIDGenerator</remote>
    <ejb−link>UniqueIDGenerator</ejb−link>
</ejb−ref>
<ejb−ref>
    <description>Access Control Manager</description>
    <ejb−ref−name>com.compoze.security.acl.IAccessControllerHome</ejb−ref−na
    <ejb−ref−type>Session</ejb−ref−type>
    <home>com.compoze.security.acl.IAccessControllerHome</home>
```

```
        <remote>com.compoze.security.acl.IAccessController</remote>
        <ejb-link>AccessController</ejb-link>
    </ejb-ref>
    <ejb-ref>
        <description>Contact Manager</description>
        <ejb-ref-name>com.compoze.contact.ejb.IContactManagerHome</ejb-ref-name
        <ejb-ref-type>Session</ejb-ref-type>
        <home>com.compoze.contact.ejb.IContactManagerHome</home>
        <remote>com.compoze.contact.ejb.IContactManager</remote>
        <ejb-link>ContactManager</ejb-link>
    </ejb-ref>
```

d. Save and close your project web.xml file.

9. Perform the following steps only if you are using a database other than PointBase.
   a. Stop the server. Choose ***Tools−−>WebLogic Server−−>Stop WebLogic Server***.
   b. Modify your domain's setDomainEnv.cmd(.sh) to use the correct database.

      In the entry set HARMONY_PORTLETS_PROPERTIES=−Dejbruntime.database=pointbase44, use the commented area above this entry to replace the pointbase44 entry with the name of your database driver. Possible values are listed above that entry.
   c. Save setDomainEnv.cmd(.sh).
   d. Restart the server.

10. Add the Login to Portal Portlet to your portal Web project. Users must log in to use the collaboration portlets.
11. Open your portal file and navigate the page where you want the portlet to appear.
12. In the ***Data Palette*** window, drag the ***My Contacts*** portlet onto the portal page.
13. In the ***Property Editor*** window, set any relevant properties.
14. Save the portal file.
15. View your portal with the WebLogic Test Browser or with your default browser.
    - ***WebLogic Test Browser*** − In the WebLogic Workshop toolbar, click the ***Start*** button (or press ***Ctrl+F5***).
    - ***Default Browser*** − In the WebLogic Workshop menu, choose ***Portal−−>Open Current Portal***.

For instructions on using the portlet's features, see *Compoze Portlets for BEA WebLogic Portal User's Guide* at http://e−docs.bea.com/wlp/docs81/pdf/compoze_portlets_users_guide.pdf.

Related Topics

Creating a Portal File

Portal Samples

Getting Started with Page Flows

Discussion Forums Portlet

The Discussion Forums portlet lets you participate in threaded forum conversations. To administer discussion forums in this portlet, use the Discussion Forum Administration portlet.

Concepts Demonstrated by this Sample

This Java Page Flow portlet illustrates group collaboration functionality within a portlet. The portlet provides edit mode.

Location of Sample Files

This sample is located in the `<BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/portalApp.work` application.

How to Run the Sample

See Viewing the Samples in Portal Samples.

How to Use the Sample in Your Portals

When this portlet is used in a domain (for example, in the portalApp in the Sample Portal Domain), the EJBs it uses are registered with JNDI names that can be used only once in the domain. That means you can use the following collaboration portlets in only one portal application in a domain: My Mail, My Task List, My Calendar, My Contacts, Discussion Forums, and Discussion Forum Administration. Within that portal application you can create multiple portal Web projects that can each contain multiple portals that reuse these portlets.

1. Create a portal application in a domain that has not used the collaboration portlets.
2. Make sure your portal application is open and the server is running (***Tools−−>WebLogic Server−−>Start WebLogic Server***).
3. Import/add the following directories and files into your portal application and portal Web project using WebLogic Workshop. (Right−click−−>***Import*** or ***Add Module*** or ***Add Library*** on the target directory). You may need to create the appropriate directories in your application.

   Be sure to add the harmony_portlets.jar library first, as shown in the following table.

Import this	***into this WebLogic Workshop directory*** (create if necessary)
`<WEBLOGIC_HOME>/samples/portal/portalApp/APP−INF/lib/harmony_portlets.jar`	<PORTAL_APP>/Libraries/

`<WEBLOGIC_HOME>/samples/portal/portalApp/discussion_ejb.jar`
`security_ejb.jar`
`uniqueid_ejb.jar`
	<PORTAL_APP>/Modules/

`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/includes/collaboration/`	<PORTAL_APP>/<project>/portlets/includes/collaboration

native_discussion.portlet	
<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/collaboration/nativedb/discussion/	<PORTAL_APP>/<project>/collaboration/native

Note: If you add the non−JAR resources to directories other than those shown, you must open the portlet file in WebLogic Workshop and edit the Content URI for both the portlet's main content and Edit page content; you must modify the package path and <view−properties> paths in the Java Page Flow files; and you must modify the import statement to the ContentController and modify any other relevant paths in the JSPs.

6. Add the collaboration tables to your database. If you have already performed this for another collaboration portlet, skip this step.

   Note: If you ran the create_* database script to set up your database, the collaboration tables already exist. Do not run create_* if, for example, you are using the default PointBase database and have already added records to the database. Follow these instructions instead.

   To add the collaboration tables to an existing database, run the following database script:

   <BEA_HOME>/<WEBLOGIC_HOME>/portal/db/<DB_TYPE>/<DB_VERSION>/collaboration_create_tabl

   for example

   bea/weblogic81/portal/db/pointbase/44/collaboration_create_tables.sql

   To run this script for PointBase:

   a. Start the PointBase Console. In a command window, run

      <DOMAIN>/startPointBaseConsole.cmd(.sh)
   b. Log into the console. The default login is weblogic/weblogic.
   c. Choose ***File−−>Open***.
   d. Open the collaboration_create_tables.sql script. The script opens in the Enter SQL Commands window.
   e. Click the ***Execute All*** button. The collaboration tables are created.
   f. Close the PointBase Console.

7. Add entries to <PORTAL_APP>/<PROJECT>/WEB−INF/web.xml.
   a. Open <PORTAL_APP>/<PROJECT>/WEB−INF/web.xml.
   b. Open <BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/WEB−INF/web.xml
   c. From the sampleportal web.xml file, copy the following sections into your project web.xml file in the appropriate locations:

```
<!−− Compoze Collaboration Discussion Thread Attachment Servlet −−>
<servlet>
    <servlet−name>CompozeDiscussionMessageFileAttachmentServlet</servlet−na
    <servlet−class>com.compoze.discussion.MessageFileAttachmentServlet</ser
</servlet>
```

```
<!-- Compoze Collaboration Discussion Thread Topic Attachment Servlet -->
<servlet>
    <servlet-name>CompozeDiscussionTopicFileAttachmentServlet</servlet-name
    <servlet-class>com.compoze.discussion.TopicFileAttachmentServlet</servl
</servlet>
.
.
.
<!-- Compoze Collaboration Discussion Thread Attachment Servlet Mapping -->
<servlet-mapping>
    <servlet-name>CompozeDiscussionMessageFileAttachmentServlet</servlet-na
    <url-pattern>*.compozediscussionmessagefileattachmentservlet</url-patte
</servlet-mapping>

<!-- Compoze Collaboration Discussion Thread Topic Attachment Servlet Mappi
<servlet-mapping>
    <servlet-name>CompozeDiscussionTopicFileAttachmentServlet</servlet-name
    <url-pattern>*.compozediscussiontopicfileattachmentservlet</url-pattern
</servlet-mapping>
.
.
.
        <resource-ref>
                <res-ref-name>ebusinessDataSource</res-ref-name>
                <res-type>javax.sql.DataSource</res-type>
                <res-auth>Container</res-auth>
        </resource-ref>
.
.
.
<ejb-ref>
    <description>Unique ID Generator</description>
           <ejb-ref-name>com.compoze.ejb.uniqueid.IUniqueIDGeneratorHome</e
           <ejb-ref-type>Session</ejb-ref-type>
    <home>com.compoze.ejb.uniqueid.IUniqueIDGeneratorHome</home>
    <remote>com.compoze.ejb.uniqueid.IUniqueIDGenerator</remote>
    <ejb-link>UniqueIDGenerator</ejb-link>
</ejb-ref>
<ejb-ref>
    <description>Discussion Forum Manager</description>
    <ejb-ref-name>com.compoze.discussion.ejb.IDiscussionManagerHome</ejb-re
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.compoze.discussion.ejb.IDiscussionManagerHome</home>
    <remote>com.compoze.discussion.ejb.IDiscussionManager</remote>
    <ejb-link>DiscussionManager</ejb-link>
</ejb-ref>
<ejb-ref>
    <description>Access Control Manager</description>
    <ejb-ref-name>com.compoze.security.acl.IAccessControllerHome</ejb-ref-na
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.compoze.security.acl.IAccessControllerHome</home>
    <remote>com.compoze.security.acl.IAccessController</remote>
    <ejb-link>AccessController</ejb-link>
</ejb-ref>
```

d. Save and close your project web.xml file.

8. Add entries to <PORTAL_APP>/<PROJECT>/WEB-INF/weblogic.xml.
   a. Open <PORTAL_APP>/<PROJECT>/WEB-INF/weblogic.xml.
   b. Open <BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/WEB-INF/weblogic

c. From the sampleportal weblogic.xml file, copy the following sections into your project weblogic.xml file inside the <reference−descriptor> element:

```
<resource−description>
    <res−ref−name>ebusinessDataSource</res−ref−name>
    <jndi−name>weblogic.jdbc.jts.ebusinessPool</jndi−name>
</resource−description>
```

d. Save and close your project weblogic.xml file.

9. Perform the following steps only if you are using a database other than PointBase.
   a. Stop the server. Choose ***Tools−−>WebLogic Server−−>Stop WebLogic Server***.
   b. Modify your domain's setDomainEnv.cmd(.sh) to use the correct database.

      In the entry set HARMONY_PORTLETS_PROPERTIES=−Dejbruntime.database=pointbase44, use the commented area above this entry to replace the pointbase44 entry with the name of your database driver. Possible values are listed above that entry.
   c. Save setDomainEnv.cmd(.sh).
   d. Restart the server.

9. Add the Login to Portal Portlet to your portal Web project. Users must log in to use the collaboration portlets.
10. Open your portal file and navigate the page where you want the portlet to appear.
11. In the ***Data Palette*** window, drag the ***Discussion Forums*** portlet onto the portal page.
12. In the ***Property Editor*** window, set any relevant properties.
13. Save the portal file.
14. View your portal with the WebLogic Test Browser or with your default browser.
    - ***WebLogic Test Browser*** − In the WebLogic Workshop toolbar, click the ***Start*** button (or press ***Ctrl+F5***).
    - ***Default Browser*** − In the WebLogic Workshop menu, choose ***Portal−−>Open Current Portal***.

For instructions on using the portlet's features, see *Compoze Portlets for BEA WebLogic Portal User's Guide* at http://e−docs.bea.com/wlp/docs81/pdf/compoze_portlets_users_guide.pdf.

Related Topics

Creating a Portal File

Portal Samples

Getting Started with Page Flows

Discussion Forum Administration Portlet

The Discussion Forum Administration portlet lets you administer threaded discussion forums in the Discussion Forums portlet.

Concepts Demonstrated by this Sample

This portlet surfaces a Java Page Flow and provides edit mode.

Location of Sample Files

This sample is located in the `<BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/portalApp.work` application.

How to Run the Sample

See Viewing the Samples in Portal Samples.

How to Use the Sample in Your Portals

When this portlet is used in a domain (for example, in the portalApp in the Sample Portal Domain), the EJBs it uses are registered with JNDI names that can be used only once in the domain. That means you can use the following collaboration portlets in only one portal application in a domain: My Mail, My Task List, My Calendar, My Contacts, Discussion Forums, and Discussion Forum Administration. Within that portal application you can create multiple portal Web projects that can each contain multiple portals that reuse these portlets.

1. Create a portal application in a domain that has not used the collaboration portlets.
2. Make sure your portal application is open and the server is running (***Tools−−>WebLogic Server−−>Start WebLogic Server***).
3. Import/add the following directories and files into your portal application and portal Web project using WebLogic Workshop. (Right−click−−>***Import*** or ***Add Module*** or ***Add Library*** on the target directory). You may need to create the appropriate directories in your application.

   Be sure to add the harmony_portlets.jar library first, as shown in the following table.

Import this	***into this WebLogic Workshop directory*** (create if necessary)
`<WEBLOGIC_HOME>/samples/portal/portalApp/APP−INF/lib/harmony_portlets.jar`	<PORTAL_APP>/Libraries/

`<WEBLOGIC_HOME>/samples/portal/portalApp/discussion_ejb.jar`
`security_ejb.jar`
`uniqueid_ejb.jar`
	<PORTAL_APP>/Modules/

`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/includes/collaboration/native_discussion_admin.portlet`	<PORTAL_APP>/<project>/portlets/includes/collaboration

<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/portlets/collaboration/nativedb/discussion/	<PORTAL_APP>/<project>/collaboration/native

Note: If you add the non−JAR resources to directories other than those shown, you must open the portlet file in WebLogic Workshop and edit the Content URI for both the portlet's main content and Edit page content; you must modify the package path and <view−properties> paths in the Java Page Flow files; and you must modify the import statement to the ContentController and modify any other relevant paths in the JSPs.

4. Add the collaboration tables to your database. If you have already performed this for another collaboration portlet, skip this step.

   Note: If you ran the create_* database script to set up your database, the collaboration tables already exist. Do not run create_* if, for example, you are using the default PointBase database and have already added records to the database. Follow these instructions instead.

   To add the collaboration tables to an existing database, run the following database script:

   <BEA_HOME>/<WEBLOGIC_HOME>/portal/db/<DB_TYPE>/<DB_VERSION>/collaboration_create_tabl

   for example

   bea/weblogic81/portal/db/pointbase/44/collaboration_create_tables.sql

   To run this script for PointBase:

   a. Start the PointBase Console. In a command window, run

      <DOMAIN>/startPointBaseConsole.cmd(.sh)
   b. Log into the console. The default login is weblogic/weblogic.
   c. Choose ***File−−>Open***.
   d. Open the collaboration_create_tables.sql script. The script opens in the Enter SQL Commands window.
   e. Click the ***Execute All*** button. The collaboration tables are created.
   f. Close the PointBase Console.

7. Add entries to <PORTAL_APP>/<PROJECT>/WEB−INF/web.xml.
   a. Open <PORTAL_APP>/<PROJECT>/WEB−INF/web.xml.
   b. Open <BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/WEB−INF/web.xml
   c. From the sampleportal web.xml file, copy the following sections into your project web.xml file in the appropriate locations:

```
<!−− Compoze Collaboration Discussion Thread Attachment Servlet −−>
<servlet>
    <servlet−name>CompozeDiscussionMessageFileAttachmentServlet</servlet−nam
    <servlet−class>com.compoze.discussion.MessageFileAttachmentServlet</serv
</servlet>

<!−− Compoze Collaboration Discussion Thread Topic Attachment Servlet −−>
<servlet>
    <servlet−name>CompozeDiscussionTopicFileAttachmentServlet</servlet−name>
```

```
    <servlet-class>com.compoze.discussion.TopicFileAttachmentServlet</servl
</servlet>
.
.
.
<!-- Compoze Collaboration Discussion Thread Attachment Servlet Mapping -->
<servlet-mapping>
    <servlet-name>CompozeDiscussionMessageFileAttachmentServlet</servlet-na
    <url-pattern>*.compozediscussionmessagefileattachmentservlet</url-patte
</servlet-mapping>

<!-- Compoze Collaboration Discussion Thread Topic Attachment Servlet Mappi
<servlet-mapping>
    <servlet-name>CompozeDiscussionTopicFileAttachmentServlet</servlet-name
    <url-pattern>*.compozediscussiontopicfileattachmentservlet</url-pattern
</servlet-mapping>
.
.
.
        <resource-ref>
                <res-ref-name>ebusinessDataSource</res-ref-name>
                <res-type>javax.sql.DataSource</res-type>
                <res-auth>Container</res-auth>
        </resource-ref>
.
.
.
<ejb-ref>
    <description>Unique ID Generator</description>
    <ejb-ref-name>com.compoze.ejb.uniqueid.IUniqueIDGeneratorHome</ejb-ref-n
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.compoze.ejb.uniqueid.IUniqueIDGeneratorHome</home>
    <remote>com.compoze.ejb.uniqueid.IUniqueIDGenerator</remote>
    <ejb-link>UniqueIDGenerator</ejb-link>
</ejb-ref>
<ejb-ref>
    <description>Discussion Forum Manager</description>
    <ejb-ref-name>com.compoze.discussion.ejb.IDiscussionManagerHome</ejb-re
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.compoze.discussion.ejb.IDiscussionManagerHome</home>
    <remote>com.compoze.discussion.ejb.IDiscussionManager</remote>
    <ejb-link>DiscussionManager</ejb-link>
</ejb-ref>
<ejb-ref>
    <description>Access Control Manager</description>
    <ejb-ref-name>com.compoze.security.acl.IAccessControllerHome</ejb-ref-na
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.compoze.security.acl.IAccessControllerHome</home>
    <remote>com.compoze.security.acl.IAccessController</remote>
    <ejb-link>AccessController</ejb-link>
</ejb-ref>
```

d. Save and close your project web.xml file.

8. Add entries to <PORTAL_APP>/<PROJECT>/WEB−INF/weblogic.xml.
   a. Open <PORTAL_APP>/<PROJECT>/WEB−INF/weblogic.xml.
   b. Open <BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/WEB−INF/weblogic
   c. From the sampleportal weblogic.xml file, copy the following sections into your project weblogic.xml file inside the <reference−descriptor> element:

```
<resource-description>
    <res-ref-name>ebusinessDataSource</res-ref-name>
    <jndi-name>weblogic.jdbc.jts.ebusinessPool</jndi-name>
</resource-description>
```

d. Save and close your project weblogic.xml file.

9. Perform the following steps only if you are using a database other than PointBase.
   a. Stop the server. Choose ***Tools−−>WebLogic Server−−>Stop WebLogic Server***.
   b. Modify your domain's setDomainEnv.cmd(.sh) to use the correct database.

      In the entry set HARMONY_PORTLETS_PROPERTIES=−Dejbruntime.database=pointbase44, use the commented area above this entry to replace the pointbase44 entry with the name of your database driver. Possible values are listed above that entry.
   c. Save setDomainEnv.cmd(.sh).
   d. Restart the server.
10. Add the Login to Portal Portlet to your portal Web project. Users must log in to use the collaboration portlets.
11. Open your portal file and navigate the page where you want the portlet to appear.
12. In the ***Data Palette*** window, drag the ***Discussion Forum Administration*** portlet onto the portal page.
13. In the ***Property Editor*** window, set any relevant properties.
14. Save the portal file.
15. View your portal with the WebLogic Test Browser or with your default browser.
    - ***WebLogic Test Browser*** − In the WebLogic Workshop toolbar, click the ***Start*** button (or press ***Ctrl+F5***).
    - ***Default Browser*** − In the WebLogic Workshop menu, choose ***Portal−−>Open Current Portal***.

For instructions on using the portlet's features, see *Compoze Portlets for BEA WebLogic Portal User's Guide* at http://e−docs.bea.com/wlp/docs81/pdf/compoze_portlets_users_guide.pdf.

Related Topics

Creating a Portal File

Portal Samples

Getting Started with Page Flows

My Content Portlet

The My Content portlet lets you completely manage your content in the BEA Virtual Content Repository without having to use the WebLogic Administration Portal. With My Content portlet users can create, update, and delete content directories and nodes, and they can browse content hierarchies and and search for content.

My Content portlet supports delegated administration, letting users view and manage only the content nodes delegated to them.

My Content Portlet does not support BEA Library Services−Enabled repositories. To access a library services−enabled repository via a portlet, use the Content Management Portlet.

Concepts Demonstrated by this Sample

This Java Page Flow portlet supports full create, read, update, and delete (CRUD) capabilities and provides security through delegated administration.

Location of Sample Files

This sample is located in the `<BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/portalApp.work` application.

How to Run the Sample

See Viewing the Samples in Portal Samples.

How to Use the Sample in Your Portals

For detailed instructions for Setting Up the My Content Portlet, see Setting Up My Content Portlet.

Using My Content Portlet

The following procedures show you how to use My Content portlet:

Creating and Modifying Content with My Content Portlet

Searching with My Content Portlet

Related Topics

Portal Samples

For information on setting up content management in the BEA Virtual Content Repository and setting up delegated administration, see the WebLogic Administration Portal documentation.

Content Management Portlet

The Content Management portlet lets you manage your content in the BEA Virtual Content Repository without having to use the WebLogic Administration Portal. With Content Management portlet users can create, update, and delete content directories and nodes.

The Content Management portlet does not allow you to view and modify content types, modify repositories, or set Delegated Administration policies. To set up types, configure repositories or set Delegated Administration policies, you must use the WebLogic Administration portal.

Concepts Demonstrated by this Sample

This portlet supports full create, read, update, and delete (CRUD) capabilities for content, BEA Library Services and provides security through previously set delegated administration.

Location of Sample Files

This sample is located in the `<BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/portalApp.work` application.

How to Run the Sample

See Viewing the Samples in Portal Samples.

How to Use the Sample in Your Portals

For detailed instructions for Setting Up the My Content Portlet, see Setting Up the Content Management Portlet.

Related Topics

Portal Samples

For information on setting up content management in the BEA Virtual Content Repository, library services and setting up delegated administration, see the WebLogic Administration Portal documentation.

Left Navigation Shell

The left navigation shell demonstrates using a navigation tree in the left frame of a portal desktop.

Concepts Demonstrated by this Sample

This sample uses a shell to create a left table cell that contains a navigation tree of links for the selected book. The main page book is rendered in the right table cell.

This portlet uses functionality similar to the Targeted Menu Portlet. The difference between the two samples is scope. The left navigation shell provides navigation an entire desktop, and the Targeted Menu portlet provides navigation for a single book.

Following is the leftNav.shell file, which shows how the header, footer, and left navigation JSPs are included to provide a left navigation area in the shell:

```
<?xml version="1.0" encoding="UTF-8"?>

<netuix:markupDefinition xmlns:netuix="http://www.bea.com/servers/netuix/xsd/controls/netuix/1.
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bea.com/servers/netuix/xsd/controls/netuix/1.0.0 markup-netuix-1
    <netuix:locale language="en"/>
    <netuix:markup>
        <netuix:shell
            title="Left Nav Shell" description="A shell with a left nav section." markupType="S
            <netuix:head/>
            <netuix:body>
                <netuix:header>
                    <netuix:jspContent contentUri="/portlets/header/header.jsp"/>
                </netuix:header>
                <netuix:jspContent contentUri="/navigation/shell/beginBody.jsp"/>
                <netuix:break/>
                <netuix:jspContent contentUri="/navigation/shell/endBody.jsp"/>
                <netuix:footer/>
            </netuix:body>
        </netuix:shell>
    </netuix:markup>
</netuix:markupDefinition>
```

Following are descriptions of the JSPs used in the shell:

- header.jsp is included to provide a header only. It is not required for left navigation functionality.
- beginBody.jsp file provides two table cells: one for the left navigation area and the other to contain the main page book of the desktop, represented by the <netuix:break/> tag. The left cell includes a <jsp:include page="/navigation/shell/leftNav.jsp"/> tag to insert the contents of leftNav.jsp in the left navigation cell. leftNav.jsp includes all the navigation functionality.
- endBody.jsp supplies the closing HTML tags for the right table cell, the table row, and the table, wrapping the main page book area within the shell (which is why the JSP is referenced after the <netuix:break/> tag).

Location of Sample Files

This sample is located in the `<BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/portalApp.work` application.

How to Run the Sample

See Viewing the Samples in Portal Samples.

How to Use the Sample in Your Portals

1. Create a portal application. Make sure you create a Portal Application and add a Portal Web Project to it.
2. Import or copy the following directories and files into your portal application and portal Web project. You may need to create the appropriate directories in your application:

Import or copy this	***to this directory*** (crea
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/framework/markup/shell/leftNav.shell`	<PORTAL_APP>/<p
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/navigation/shell/beginBody.jsp endBody.jsp leftNav.jsp`	<PORTAL_APP>/<p
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/framework/markup/skins/<skin>/css/treenav.css`	<PORTAL_APP>/<p
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/framework/markup/skins/<skin>/images/treenav*.gif`	<PORTAL_APP>/<p `<skin>/images/`
`<WEBLOGIC_HOME>/samples/portal/portalApp/sampleportal/framework/markup/skins/<skin>/skin.properties`	If you already have a that you do not want entry for treenav.css into your skin.proper entire sample skin.pr <PORTAL_APP>/<p

3. Open your portal file and navigate the page where you want the portlet to appear.
4. In the ***Data Palette*** window, drag the ***Login to Portal*** portlet onto a placeholder on the page.
5. In the ***Property Editor*** window, set any relevant properties.
6. Save the portal file.
7. View your portal with the WebLogic Test Browser or with your default browser.
   - ***WebLogic Test Browser*** − In the WebLogic Workshop toolbar, click the ***Start*** button (or press ***Ctrl+F5***).
   - ***Default Browser*** − In the WebLogic Workshop menu, choose ***Portal−−>Open Current Portal***.

Related Topics

How the Shell Determines Header and Footer Content

Targeted Menu Portlet

Creating Shells

Creating Skins and Skin Themes

Portal Samples

Tutorial Portal

Tutorial Portal is a portal Web project that provides self−documented examples of portal development implementations. The portal resources in Tutorial Portal are designed more for viewing and reference than for component reuse in your portals.

Concepts Demonstrated by this Sample

The following functionality is illustrated by the development examples:

- Authentication
- Controls
- Look & Feel
- Java Page Flows
- Internationalization
- Backing Files
- Window Modes
- Portlet Preferences
- Portlet Events
- Multichannel (mobile device support)

Location of Sample Files

This sample is located in the `<BEA_HOME>/<WEBLOGIC_HOME>/samples/portal/portalApp/portalApp.work` application. With that application open, the sample is located in the tutorial project. Open the tutorialPortal.portal file in WebLogic Workshop Platform Edition.

How to Run the Sample

See Viewing the Samples in Portal Samples.

Related Topics

Portal Samples

Enterprise JavaBean Samples

The samples described in this section illustrate Enterprise JavaBeans development features and techniques.

Topics Included in This Section

Automatic Primary Key Generation Sample

A CMP entity bean sample showing how a named sequence table that generates unique key values is used to create primary keys for a simple entity bean.

BMP Entity Bean Sample

A Bean−Managed Persistence (BMP) entity bean sample showing the main implementation differences with Container−Managed Persistence (CMP) entity beans.

EJB Security Sample

An Enterprise JavaBean sample that shows how role−based security is used to limit access to methods and to ensure that the bean is run assuming a certain security role.

Finder Methods Sample

A CMP entity bean sample that shows common finder methods, using EJB QL and WebLogic QL queries.

Home Methods Sample

A CMP entity bean sample that uses home methods to obtain information about the data it represents.

Message−Driven Bean Sample

A message−driven bean sample that shows how to process JMS messages, implementing a message and a session façade pattern.

Select Methods Sample

A CMP entity bean sample that shows common select methods, using EJB QL and WebLogic QL queries.

Value Object Sample

A sample that uses a value object to limit network traffic between a page flow and various EJBs, following a session façade pattern.

Related Topics

EJB Tutorial Sample

Developing Enterprise JavaBeans

Tutorial: Enterprise JavaBeans

Automatic Primary Key Generation Sample

This sample shows a CMP entity bean that uses a named sequence table to auto−generate primary keys. For more information on the various methods available to auto−generate primary keys, see Automatic Primary Key Generation.

Concepts Demonstrated by this Sample

- Use of a named sequence table
- Use of a CMP entity bean
- Use of an EJB Control
- Use of a conversational Test Web Service

Location of Sample Files

The code of the CMP entity bean is located in the automaticPK folder of the EJBs project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\EJBs\automaticPK\Customer_APK.ejb

The web service that you run as a client application to test the EJB, and the EJB control that you use to locate and reference the EJB, are located in the automaticPK folder of the EJBs_ClientApps project in the SamplesApp sample application. In the file system the location of the web service is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\EJBs_ClientApps\automaticPK\CustomerControlT

To Run the Sample

1. Start WebLogic Server in the appropriate domain.

   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Examples Server***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/workshop/startWebLogic.sh

2. Launch the test web service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/EJBs_ClientApps/automaticPK/CustomerControlTest.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Invoke the startTestDrive method to create a new conversational instance.
5. Click continue this conversation.
6. Locate the ***create*** section, enter a first and last name, and click ***create***. Notice that the EJB's create method is invoked via the EJB control.
7. Click continue this conversation.
8. Locate and click ***getCustomer_ID***. Notice that the primary key value for the newly created entity bean is returned.

Related Topics

Automatic Primary Key Generation

How Do I: Test an Enterprise JavaBean?

Test View

BMP Entity Bean Sample

The Bean−Managed Persistence (BMP) entity bean sample shows how to build a basic entity bean using bean−managed persistence instead of container managed persistence. This example is intended to provide a comparison between how a BMP entity bean and a CMP entity bean implement similar behavior, and to that effect implements the *BMPItemBean* entity bean, which is similar to the CMP entity bean *ItemsBean_F* in the Finder Methods Sample. The entity bean implements several persistence fields and their accessor methods, an ejbCreate method, a finder method ejbFindByPrice, and the methods that you need to explicitly define in BMP to handle database queries, that is, findByPrimaryKey, ejbLoad, ejbStore, ejbRemove, as well as setEntityContext and unsetEntityContext. The example uses JDBC to talk to a relational database.

Concepts Demonstrated by this Sample

- Implementing bean−managed persistence.
- Examining an EJB at runtime using a conversational web service via an EJB control.

Location of Sample Files

The code of the BMP entity bean is located in the beanManagedPersistence folder of the EJBs project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\EJBs\beanManagedPersistence

The web service that you run as a client application to test the EJB, and the EJB control that you use to locate and reference the EJB, are located in the beanManagedPersistence folder of the EJBs_ClientApps project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\EJBs_ClientApps\beanManagedPersistence

To Run the Sample

1. Start WebLogic Server in the appropriate domain.

   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Examples Server***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/workshop/startWebLogic.sh

2. Launch the BMPItemBeanCtrlTest web service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/EJBs_ClientApps/beanManagedPersistence/BMPItemBeanCtrlTest.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Invoke the startTestDrive method to create a new conversational instance.
5. Click continue this conversation.
6. Locate the create method, provide a unique number, name, and price, and click the ***create*** button.
7. Click continue this conversation.

8. Repeat the last two steps a number of times to enter additional items to the database.
9. Locate the findByPrimaryKey method, enter the number of a previously entered product, and click the ***findByPrimaryKey*** button.
10. Click continue this conversation.
11. Use any of the getter methods to verify that the correct product has been returned.
12. Click continue this conversation.
13. Locate the findByPrice method, enter a price, and click the ***findByPrimaryKey*** button. Click continue this conversation, and examine the returned set of items.
14. Return to ***WebLogic Workshop*** and press the ***Stop*** button to close the Test Browser.

Related Topics

Bean−Managed Persistence

EJB Security Sample

In this EJB sample two session beans are used to demonstrate various uses of role−based security. Specifically, this sample shows how a certain security role can be assumed when executing code and how to use a security role to grant or block access to the execution of certain methods.

In the example, a *RoleCheckerBean* EJB, running under the security role *engineer*, is called by a web service *RoleCheckerControlTest.jws*, running under the security role *manager*. A third EJB, *InvokedBean*, is in turn invoked by the RoleCheckerBean EJB. The RoleCheckerBean EJB has the following methods:

- ceoOnly has a method permission setting that limits access only to callers running under the ceo security role.
- discoverCallerRole uses the SessionContext's isCallerInRole method to determine the security of the caller.
- discoverRoleCheckersRole invokes the InvokedBean, which in turn uses uses the SessionContext's isCallerInRole method to determine the security of its caller, that is, RoleCheckerBean.
- managerOnly has a method permission setting that limits access only to callers running under the manager security role.

Concepts Demonstrated by this Sample

- Running EJB code with certain security permissions
- Blocking method execution to security roles

Location of Sample Files

The code of the EJBs is located in the security folder of the EJBs project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\EJBs\security

The web service that you run as a client application to test the EJBs and the EJB control that you use to locate and reference the RoleCheckerBean EJB are located in the security folder of the EJBs_ClientApps project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\EJBs_ClientApps\security

To Run the Sample

1. Start WebLogic Server in the appropriate domain.

   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Examples Server***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/workshop/startWebLogic.sh

2. Launch the RoleCheckerControlTest web service either by opening it in WebLogic Workshop and selecting the Start operation or by entering

http://localhost:7001/EJBs_ClientApps/security/RoleCheckerControlTest.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.

3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Invoke the ceoOnly method on the web service, which will invoke the corresponding method on the EJB. The EJB's ceoOnly method can only be invoked by the ceo security role. Because the web service runs under the manager role, a javax.ejb.AccessLocalException security violation exception is thrown.
5. Click the Test operations link.
6. Invoke the managerOnly method on the web service, which will invoke the corresponding method on the EJB. The EJB's managerOnly method can only be invoked by the manager security role. Because the web service runs under the manager role, the method executes normally.
7. Click the Test operations link.
8. Invoke the discoverCallerRole method on the web service, which will invoke the corresponding method on the EJB. The EJB's discoverCallerRole method checks the security role of the invoking client and determines that the web service runs under the manager role.
9. Click the Test operations link.
10. Invoke the discoverRoleCheckersRole method on the web service, which will invoke the corresponding method on the EJB. The EJB's discoverRoleCheckersRole method checks the security role of the EJB itself and determines that it runs under the engineer role.
11. Return to ***WebLogic Workshop*** and press the ***Stop*** button to close the Test Browser.

Related Topics

Role−Based Security

Finder Methods Sample

A finder method is invoked through a CMP entity bean's (local or remote) home interface, and returns (local or remote) references to the entity beans that match an EJB QL query. This sample shows common finder methods, using EJB QL and WebLogic QL queries.

In the example, an *ItemsBean_F* EJB models the persistent data in a prefilled database table, representing an inventory of products. Each items has a product name, a quantity that is in stock, and a price. In addition, this EJB has an entity relationship with the *ManufacturerBean_F* EJB, which holds information about the various manufacturers known for these products. Each product has one manufacturer, and a manufacturer has multiple products. Various finder methods are used to obtain information about the records in the database tables.

Concepts Demonstrated by this Sample

- Using a finder method to return references to entity beans.
- Using the IN operator to return an entity bean from a relation.
- Using a literal value in a WHERE clause
- Using the DISTINCT keyword for a finder method to return unique instances.
- Using a finder method with multiple input parameters.
- Using the ORDERBY keyword.
- Using of the aggregate function AVG.
- Using subqueries.
- Examining an EJB at runtime using a conversational web service via an EJB control.

Location of Sample Files

The code of the CMP entity beans is located in the finderMethods folder of the EJBs project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\EJBs\finderMethods

The web services that you run as a client application to test the EJBs, and the EJB controls that you use to locate and reference the EJBs, are located in the finderMethods folder of the EJBs_ClientApps project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\EJBs_ClientApps\finderMethods

To Run the Sample

1. Start WebLogic Server in the appropriate domain.

   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Examples Server***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/workshop/startWebLogic.sh

2. Launch the ItemsBeanControlTest web service either by opening it in WebLogic Workshop and selecting the Start operation or by entering

http://localhost:7001/EJBs_ClientApps/finderMethods/ItemsBeanControlTest.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.

3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Invoke the startTestDrive method to create a new conversational instance.
5. Click continue this conversation.
6. Browse through the finder methods and select one you would like to run. Follow the instructions given for the method and observe the outcome.
7. Repeat the last two steps for the other finder methods you would like to run.
8. Return to ***WebLogic Workshop*** and press the ***Stop*** button to close the Test Browser.

9. Launch the ManufacturerBeanControlTest web service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/EJBs_ClientApps/finderMethods/ManufacturerBeanControlTest.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
10. Repeat steps 3 through 8 to test the finder methods available through this web service.

Related Topics

Home Methods Sample

@ejbgen:select Annotation

Home Methods Sample

A home method is a business method that is not specific to one bean instance, and that is invoked on an entity bean's (local or remote) home interface. This sample shows a CMP entity bean that uses various home methods to obtain information about the persistent data the bean represents. Specifically, an *ItemsBean_H* EJB models the persistent data in a prefilled database table *Items*, representing an inventory of products. Each items has a product name, a quantity that is in stock, and a price. Various home methods are used to obtain information about the items in stock on the basis of their price.

Concepts Demonstrated by this Sample

- A home method is defined in the home interface (local home interface in the example). In other words, you do not first have to obtain an instance of an entity bean before you can invoke a home method.
- A home method is typically used to obtain some information about (a subset of) the entity beans. Typically a select method is used to retrieve a collection of entity beans, and some business logic is used for this collection to obtain the desired information. Specifically, the home method ejbHomeGetNumberOfExpensiveItems retrieves all items above a certain price and returns the number of different products that are expensive. The method ejbHomeGetAverageExpensiveItems also retrieves all items above a certain price but instead it retrieves the average price of those expensive items. The method ejbHomeGetExpensiveItemsInStock determines whether any expensive items are in stock. All three methods use the method ejbSelectExpensiveItems to obtain a collection of items that are above the given price.
- The *ItemsBean_H* EJB is run using a conversational test web service via an EJB control.

Location of Sample Files

The code of the CMP entity bean is located in the homeMethods folder of the EJBs project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\EJBs\homeMethods\ItemsBean_H.ejb

The web service that you run as a client application to test the EJB, and the EJB control that you use to locate and reference the EJB, are located in the homeMethods folder of the EJBs_ClientApps project in the SamplesApp sample application. In the file system the location of the web service is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\EJBs_ClientApps\homeMethods\ItemsBeanContro

To Run the Sample

1. Start WebLogic Server in the appropriate domain.

   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Examples Server***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/workshop/startWebLogic.sh

2. Launch the test web service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/EJBs_ClientApps/homeMethods/ItemsBeanControlTest.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Invoke the startTestDrive method to create a new conversational instance.
5. Click continue this conversation.
6. Locate the ***ejbHomeGetNumberOfExpensiveItems*** section, enter a valid price, and click ***ejbHomeGetNumberOfExpensiveItems***. The number of items above that price are returned.
7. Repeat the last two steps for the other home methods.

Related Topics

How Do I: Add a Home Method to an Entity Bean?

@ejbgen:local−home−method Annotation

@ejbgen:remote−home−method Annotation

Select Methods Sample

Message−Driven Bean Sample

This sample shows how message−driven beans can be used by a client application to invoke the methods of multiple session or entity beans in an asynchronous manner. Specifically, a page flow sends multiple JMS messages to various message−driven beans, which in turn create new entity bean instances or remove entity beans, a pattern known as message façade. In addition, the page flow interacts with a session bean to invoke read operations on the same entity bean, a pattern known as session façade. The combination of message façade for write operations and session façade for read operations is a common design approach in full−fledged EJB applications with asynchrony requirements.

The following EJBs are used in this sample:

- Two message−driven beans AddViaQMDBean and AddViaQMDBean_Dup, whose definitions are functionally identical. These beans listen to the message queue jms/SamplesAppMDBQ and accept messages with the message selector Command = 'Add'. Upon receiving a message, a message bean instance creates a new SimpleToken object, with the token name as specified in the message. Because these beans listen to a queue (a point−to−point system), exactly one message−driven bean instance will receive a given message.

  For demonstration purposes a pause of 1 second is built in during the processing of a message, and the maximum number of beans instances allowed at one time for each bean is set to two.
- A message−driven bean DeleteViaQMDBean. This bean listens to the message queue jms/SamplesAppMDBQ and accepts messages with the message selector Command = 'Delete'. Upon receiving a message, this bean uses the entity bean SimpleToken to delete all SimpleToken records from the database. Notice that this bean listens to the same queue as AddViaQMDBBean and AddViaQMDBBean_Dup, and that message selectors are used in both cases to be selective about the messages that are accepted.
- Two message−driven beans AddViaTopicMDBean and AddViaTopicMDBean_Dup, whose definitions are functionally identical. These beans listen to the message topic jms/SamplesAppMDBTopic for messages. Upon receiving a message, a message bean instance creates a new SimpleToken object, with the token name as specified in the message. Because these two bean classes listen to a topic (a publish−and−subscribe system), a message is received by both (causing the creation of two SimpleTokens for every message sent).

  For demonstration purposes a pause of 1 second is built in during the processing of a message, and the maximum number of beans instances allowed at one time for each bean is set to two.
- An entity bean SimpleTokenBean to manipulate SimpleTokens. Its definition contains the methods create and remove, as well as the finder methods findAll and findAllOrdered.
- A session bean FacadeBean to act as the intermediary between the SimpleTokenBean's read operations and the page flow. It contains the business methods findAllTokens and findAllTokensOrdered, which invoke the equivalent methods on the entity bean and return a Collection of SimpleToken value objects (for more information, see the Value Object Sample).

In addition, the following components are used in this sample:

- A page flow application that displays a table of SimpleTokens, and enables a user to add simple tokens via a queue or a topic, delete all tokens, or find, unordered or ordered by name, all simple tokens currently known.
- An EJB control FacadeBeanControl used by the page flow to communicate with the FacadeBean.

- A custom control MessageSendImpl used by the page flow to send JMS messages. This custom control uses the JMS controls sendToQueueControl and sendToTopicControl to send messages to the queue jms/SamplesAppMDBQ and topic jms/SamplesAppMDBTopic.

Note. This sample demonstrates the session and message façade pattern, and the use of JMS messages. For more detailed information on these topics, see your favorite documentation on EJB design patterns and JMS messaging.

Concepts Demonstrated by this Sample

- JMS messaging, and the difference between queues and topics
- Use of message−driven beans to receive messages
- Use of message selectors to be selective about the messages a message−driven bean will process
- Combined use of the message façade pattern to model asynchronous write operations and the session façade pattern to model read operations.
- Use of EJB controls, custom controls, and JMS controls to encapsulate business logic invoked by a page flow
- Use of a page flow

Location of Sample Files

The EJBs are located in the messageDriven folder of the EJBs project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\EJBs\messageDriven

The page flow files that you run as a client application, the various controls used by the page flow, are located in the messageDriven folder of the EJBs_ClientApps project in the SamplesApp sample application. In the file system of the JPF controller file is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\EJBs_ClientApps\messageDriven\Controller.jpf

To Run the Sample

1. Start WebLogic Server in the appropriate domain.

   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Examples Server***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/workshop/startWebLogic.sh

2. Launch the page flow controller either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/EJBs_ClientApps/messageDriven\Controller.jpf in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.

Related Topics

Value Object Sample

Getting Started with Message−Driven Beans

@ejbgen:message−driven Annotation

JMS Control

EJB Control

Building Custom Java Controls

Select Methods Sample

A select method is a private method that can only be used internally by a CMP entity bean class, that is, it is not defined in the bean's interfaces. The method can return (local or remote) references to the entity beans or the values of an individual CMP field that match an EJB QL or WebLogic QL query. This sample shows mostly common select methods that return the values of an individual CMP fields. Because select methods can only be used internally, various business methods are defined that use these select methods.

Note. For more examples of (WebLogic) EJB QL queries that returns references entity beans, see the Finder Methods Sample. For examples of a select method that is used by various home methods, see the Home Methods Sample.

In the example, an *ItemsBean_S* EJB models the persistent data in a prefilled database table, representing an inventory of products. Each items has a product name, a quantity that is in stock, and a price. In addition, this EJB has an entity relationship with the *ManufacturerBean_S* EJB, which holds information about the various manufacturers known for these products. Each product has one manufacturer, and a manufacturer has multiple products. Various select methods are used to obtain information about the items in the database. For each select method a home method is created with a matching name that simply returns the results of the select method as a String. For more information on common uses of home methods, see the Home Methods Sample mentioned above.

Concepts Demonstrated by this Sample

- Using a select method to return references to entity beans.
- Using a select method to return the values of individual CMP fields.
- Using the IN operator to return an entity bean from a relation.
- Using the ORDERBY keyword.
- Using WebLogic QL to return the values of multiple CMP fields.
- Having a business method invoke a select method.
- Examining an EJB at runtime using a conversational web service via an EJB control.

Location of Sample Files

The code of the CMP entity bean is located in the selectMethods folder of the EJBs project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\EJBs\selectMethods\ItemsBean_S.ejb

The web service that you run as a client application to test the EJB, and the EJB control that you use to locate and reference the EJB, are located in the selectMethods folder of the EJBs_ClientApps project in the SamplesApp sample application. In the file system the location of the web service is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\EJBs_ClientApps\selectMethods\ItemsBeanContro

To Run the Sample

1. Start WebLogic Server in the appropriate domain.

- On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Examples Server***.
- On Linux or Solaris systems, run:

  BEA_HOME/weblogic81/samples/workshop/startWebLogic.sh

2. Launch the test web service either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/EJBs_ClientApps/selectMethods/ItemsBeanControlTest.jws in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.
3. Navigate to the ***Test Form*** tab of Test View, if necessary.
4. Invoke the startTestDrive method to create a new conversational instance.
5. Click continue this conversation.
6. Browse through the select methods and decide which one you would like to run. Follow the instructions given for the method and observe the outcome.
7. Repeat the last two steps for the next select method you would like to run.

Related Topics

Home Methods Sample

@ejbgen:select Annotation

Value Object Sample

This sample demonstrates how value objects whose class definitions are auto−generated by WebLogic, can be used to limit network traffic between a client application and a set of EJBs arranged in a session façade pattern.

The following components are used in this sample:

- An ItemsBean entity bean defining an item. For this EJB a value class *ItemValue* is auto−generated as specified by its ejbgen:file−generation tag. The ItemsBean EJB has a one−to−many bidirectional relationship with a ManufacturerBean.
- A ManufacturerBean entity bean defining a manufacturer. For this EJB a value class *ItemValue* is auto−generated as specified by its ejbgen:file−generation tag.
- An AdminBean session bean with business methods to manipulate manufacturers and (albeit not yet implemented) items. The session bean is the intermediary between the client application and the entity beans, following a session façade pattern. It has the following methods:
    - The method void addManufacturer(ManufacturerVO mVO) takes a manufacturer value object and creates a new manufacturer.
    - The method ManufacturerVO getManufacturer(Integer PK) returns a manufacturer value object for a manufacturer. This value object contains all information regarding this information, including complete details of the items produced by this manufacturer.
    - The method Collection getManufacturers() returns a collection of manufacturer value objects, representing all manufacturers. This value object does not contain information of the items produced by these manufacturers.
- An EJB Control AdminBeanControl used by the page flow to communicate with the AdminBean.
- A page flow application that displays tables of manufacturers and items and enables a user to enter a new manufacturer.

Note. This sample shows only one possible way in which value objects can be used. For more information on value objects, data transfer object pattern, and the session façade pattern, see your favorite documentation on EJB design patterns.

Concepts Demonstrated by this Sample

- Enabling auto−generation of a value class definition using the @ejbgen:file−generation Annotation on an entity bean.
- Use of an auto−generated value class.
- Limiting the information a value object holds to curb the amount of information that needs to be transmitted over the network.
- Session façade pattern.
- Use of an EJB Control.
- Use of a page flow.

Location of Sample Files

The code of the session bean is located in the valueObject folder of the EJBs project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\EJBs\valueObject\AdminBean.ejb

The code of the CMP entity beans is located in the finderMethods folder of the EJBs project in the SamplesApp sample application. In the file system the location is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\EJBs\finderMethods

The page flow files, and the EJB control that you use to locate and reference the EJB, are located in the valueObject folder of the EJBs_ClientApps project in the SamplesApp sample application. In the file system the location of the JPF controller file is:

BEA_HOME\weblogic81\samples\workshop\SamplesApp\EJBs_ClientApps\valueObject\Controller.jpf

To Run the Sample

1. Start WebLogic Server in the appropriate domain.
   - On Microsoft Windows systems, from the ***Start*** menu, choose ***BEA WebLogic Platform 8.1***−−>***Examples***−−>***WebLogic Workshop Examples***−−>***Start Examples Server***.
   - On Linux or Solaris systems, run:

     BEA_HOME/weblogic81/samples/workshop/startWebLogic.sh

2. Launch the page flow controller either by opening it in WebLogic Workshop and selecting the Start operation or by entering http://localhost:7001/EJBs_ClientApps/valueObject/Controller.jpf in the address bar of your browser. If WebLogic Server is running in the appropriate domain on this machine, you may click here to run the sample.

Related Topics

@ejbgen:file−generation Annotation

@ejbgen:value−object Annotation