Creating a J2C application for an IMS phone book transaction

Time required

To complete this tutorial, you will need approximately **30 minutes**. If you decide to explore other facets of the J2C wizard while working on the tutorial, it could take longer to finish.

Prerequisites

In order to complete this tutorial end to end, you should be familiar with:

- J2EE and Java programming
- Basic IMS Transaction Manager (IMS TM) concepts

Learning objectives

This tutorial is divided into several exercises that must be completed in sequence for the tutorial to work properly. This tutorial teaches you how to use the J2C Java bean wizard to submit a transaction to IMS. While completeing the exercises, you will:

- Use the J2C Java bean wizard to submit a transaction to IMS.
- Create a Java method, `runPhonebook.java`, which accepts a customer number and returns the customer's information.
- Create a JSP to deploy the application on WebSphere Application Server.

When you are ready, begin Exercise 1.1: Selecting the resource adapter

Exercise 1.1: Selecting the resource adapter

This tutorial illustrates how to use the J2C wizard to build a simple Web application that processes an IMS transaction that returns a phone book record.

Before you can begin this tutorial, you must first obtain the required resources:

- **Connection to a IMS server**: In this tutorial, your application interacts with an IMS program on a server. Specifically, you need to perform some setup work on the IMS server machine, where you want the Phonebook IMS program to run. These steps are not covered: contact your IMS server administrator for setup information and to obtain connection properties for your IMS server.
- **A copy of the COBOL copy book `EX01.cbl`** You may locate this file in your product installation directory: `\rad\eclipse\plugins\com.ibm.j2c.cheatsheet.content_6.0.0\Samples\IMS\phonebook`. If you wish to store it locally, you can copy the code from here:

EX01.cbl

```
IDENTIFICATION DIVISION.
        ENVIRONMENT DIVISION.
        CONFIGURATION SECTION.
        DATA DIVISION.
       *
       *    IMS Connector for Java, COBOL Transaction Message Source
       *
       ************************************************************************
       *                                                                      *
       * (c) Copyright IBM Corp. 2003                                       *,
       * All Rights Reserved                                                  *
       * Licensed Materials - Property of IBM                                 *
       *                                                                      *
       * DISCLAIMER OF WARRANTIES.                                            *
       *                                                                      *
       * The following (enclosed) code is provided to you solely for the      *
       * purpose of assisting you in the development of your applications.    *
       * The code is provided "AS IS." IBM MAKES NO WARRANTIES, EXPRESS OR    *
       * IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF      *
       * MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING      *
       * THE FUNCTION OR PERFORMANCE OF THIS CODE.                            *
       * IBM shall not be liable for any damages arising out of your use      *
       * of the generated code, even if they have been advised of the         *
       * possibility of such damages.                                         *
       *                                                                      *
       * DISTRIBUTION.                                                        *
       *                                                                      *
       * This generated code can be freely distributed, copied, altered,      *
       * and incorporated into other software, provided that:                 *
       *   - It bears the above Copyright notice and DISCLAIMER intact        *
       *   - The software is not for resale                                   *
       *                                                                      *
       ************************************************************************
       *
        LINKAGE SECTION.

        01  INPUT-MSG.
            02  IN-LL          PICTURE S9(3) COMP.
            02  IN-ZZ          PICTURE S9(3) COMP.
            02  IN-TRCD        PICTURE X(10).
            02  IN-CMD         PICTURE X(8).
            02  IN-NAME1       PICTURE X(10).
            02  IN-NAME2       PICTURE X(10).
            02  IN-EXTN        PICTURE X(10).
```

```
    02  IN-ZIP          PICTURE X(7).

01  OUTPUT-MSG.
    02  OUT-LL         PICTURE S9(3) COMP VALUE +0.
    02  OUT-ZZ         PICTURE S9(3) COMP VALUE +0.
    02  OUT-MSG        PICTURE X(40) VALUE SPACES.
    02  OUT-CMD        PICTURE X(8) VALUE SPACES.
    02  OUT-NAME1      PICTURE X(10) VALUE SPACES.
    02  OUT-NAME2      PICTURE X(10) VALUE SPACES.
    02  OUT-EXTN       PICTURE X(10) VALUE SPACES.
    02  OUT-ZIP        PICTURE X(7) VALUE SPACES.
    02  OUT-SEGNO      PICTURE X(4) VALUE SPACES.

PROCEDURE DIVISION.
```

- **A clean workspace.**

Switching to the J2EE Perspective

If the J2EE icon,  , does not appear in the top right tab of the workspace, you need to switch to the J2EE perspective.

1. From the menu bar, select **Window** > **Open Perspective** > **Other**. The Select Perspective window opens.
2. Select **J2EE**.
3. Click **OK**. The J2EE perspective opens.

Selecting the IMS Resource Adapter and connecting to the IMS server

1. In the J2EE perspective, select **File** > **New** > **Other**.
2. In the New page, select **J2C** > **J2C Java Bean**. Click **Next**
   Note: If you do not see the J2C option in the wizard list, you need to Enable J2C Capabilities.
   1. From the menu bar, click **Window** > **Preferences**.
   2. On the left side of the Preferences window, expand Workbench.
   3. Click **Capabilities**. The Capabilities pane is displayed. If you would like to receive a prompt when a feature is first used that requires an enabled capability, select **Prompt when enabling capabilities**.
   4. Expand Enterprise Java.
   5. Select **Enterprise Java**. The necessary J2C capability is now enabled. Alternatively, you can select the Enterprise Java capability folder to enable all of the capabilities that folder contains. To set the list of enabled capabilities back to its state at product install time, click **Restore Defaults**.
   6. To save your changes, click **Apply**, and then click **OK**. Enabling Enterprise Java capabilities will automatically enable any other capabilities that are required to develop and debug J2C applications.
3. In the **Resource Adapters Selection** page, , select either the J2C 1.0 or J2C 1.5 IMS resource adapter. For this tutorial select **IMS Connector for Java (IBM: 9.1.0.1.1)**. Click **Next**.
4. In the **Connection Properties** page, de-select the Managed Connection check box and select **Nonmanaged connection**. (For this tutorial, you will use a non-managed connection to directly access IMS.) Accept the default Connection class name of `com.ibm.connector2.ims.ico.IMSManagedConnectionFactory`. In the blank fields, enter all the required connection information. Required fields, indicated by an asterisk (*), include the following:
   For TCP/IP connection:
   - **Host name**: (Required) The IP address or host name of IMS Connect.
   - **Port Number**: (Required) The number of the port used by the target IMS connect.

   For local option connection:
   - **IMS Connect name**: (Required) The name of the target IMS connect.

   For both:

- **Data Store Name**: (Required) The name of the target IMS datastore.

You may obtain the connection information from your IMS system administrator. When you have provided the required connection information, click **Next**.

Now you are ready to begin Exercise 1.2: Setting up the Web project and Java Interface and Implementation.

Exercise 1.2: Setting up the Web Project and Java Interface and Implementations

Before you begin, you must complete Exercise 1.1: Selecting the Resource Adapter.

Exercise 1.2 steps you through the creation of a J2C application. In this exercise you will

- Create a J2C Java bean
- Create a dynamic Web project

Creating a J2C Java bean

All work done in the workbench must be associated with a project. Projects provide an organized view of the work files and directories, optimized with functions based on the type of project. In the workbench, all files must reside in a project, so before you create the J2C Java bean, you need to create a project to put it in.

1. Type the value `IMSPhoneBook` in the **Java Project Name** field.
2. Click the **New** button beside the **Java Project Name** field to create the new project.
3. In the New Source Project Creation page, select **Web project**, and click **Next**.
4. In the New Dynamic Web Project page, click **Show Advanced**.
5. Ensure that the following values are selected:
    - **Name**: `IMSPhoneBook`
    - **Project location**: accept default
    - **Servlet version**: `2.4`
    - **Target server**: `WebSphere Application Server v6.0`
    - **EAR Project**: `IMSPhoneBookEAR`
    - **Context Root**: `IMSPhoneBook`
6. Click **Finish**.
7. A dialog box may appear asking if you would like to switch to the Web perspective. Click **Yes**. You will now be back on the New J2C Java Bean page
8. On the New J2C Java Bean page, ensure that the following values appear:
    - In the **Package Name** field, type `sample.ims`
    - In the **Interface Name** field, type `PhoneBook`.
    - In the **Implementation Name** field, type `PhoneBookImpl`.
9. Click **Finish**.

Now you are ready to begin Exercise 1.3: Creating the Java method.

Exercise 1.3: Creating the Java Method

Before you begin, you must complete Exercise 1.2: Setting up the Web project and Java interface and implementations .

Exercise 1.3 leads you through the creation of a Java method. In this exercise you will

- Create a Java method
- Create the input and output data mapping between COBOL and Java

Creating a Java method

1. In the Snippets view, select **J2C**. Right click **Add Java method to J2C Java bean**.
2. In the Java method name field, type `runPhoneBook` for the name of the method. Click **Next**.

Creating the input data mapping between COBOL and Java

In this step, you will import the Ex01.cbl (COBOL) file that is needed to create your applicaion. The Ex01.cbl file is located in <RSDP_installdir>\rad\eclipse\plugins\com.ibm.j2c.cheatsheet.content_6.0.0 \Samples\IMS\phonebook, where <RSDP_installdir> is the directory where this product is installed. The COBOL file contains the application program that runs on the IMS server. It has the definition of the structure to be passed to the IMS server via the communications area. This structure represents the customer records being returned from the IMS application program. Before you can work with a file, you must import it from the file system into the workbench.

1. In the **Specify the input/output type** of the Java Method page, click **New**.
2. In the Data Import page, ensure that the **Choose mapping** field is **COBOL_TO_JAVA**. Click **Browse** beside the **Cobol file name** field.
3. Locate the `Ex01.cbl` file in the file system, and click **Open**.
4. Click **Next**.
5. In the COBOL Importer page, Click **Show Advanced**.
    - Select the following options:

Parameter	Value
Platform Name	Z/OS
Code Page	037
Floating point format name	IBM 390 Hexadecimal
External decimal sign	EBCDIC
Endian name	Big
Remote integer endian name	Big
Quote name	DOUBLE
Trunc name	STD
Nsymbol name	DBCS

    - Beside the **Data structures** list, click the **Query** button.
    - The data structures from the `Ex01.cbl` file are displayed. Select **INPUT-MSG**.Click **Next**.
6. In the Saving properties page, select the following values for input type:
    - The GenerationStyle is Default.
    - Click **Browse** beside the **Project Name** field to choose the Java project **IMSPhoneBook**
    - In the **Package Name** field, enter sample.ims.data
    - In the **Class Name** field, accept the default name **INPUTMSG**. Click **Finish**.

Creating the output data mapping between COBOL and Java

1. In the **Java method** page, click **New** next to the **Output type** field.
2. In the **Data Import** page, ensure that the **Choose mapping** field is **COBOL_TO_JAVA**.
3. Click the Browse button besides the **Cobol file name** field:

4. Locate the Ex01.cbl file. Click **Open**.
5. Once the file is added, Click **Next**.
6. In the **COBOL Importer** page, Click on the **Show Advanced**.
7. In the **COBOL Importer** page, select the following communication data structures:

Parameter	Value
Platform Name	Z/OS
Floating point format name	IBM 390 Hexadecimal
codepage	037
External decimal sign	EBCDIC
Endian name	Big
Remote integer endian name	Big
Quote name	DOUBLE
Trunc name	STD
Nsymbol name	DBCS

8. Beside the **Data structures** text area, click the **Query** button.
9. The data structures from the **Ex01.cbl** file are displayed. Select **OUTPUT-MSG**. Click **Next**.
10. In the Saving properties page, select the following values for output type:
    - The GenerationStyle is Default.
    - Click **Browse** beside the **Project Name** field to choose the Java project **IMSPhoneBook**
    - In the **Java package Name** field, enter sample.ims.data
    - In the **Java Class Name** field, accept the default name **OUTPUTMSG**. Click **Finish**.
11. On the **Java Method** page , click **Finish** to complete the operation.
12. In the **Binding Details** page, ensure that **interactionVerb** is **SYNC_SEND_RECEIVE(1)** to indicate that the interaction with IMS involves a send followed by a receive interaction. Click **Next**.

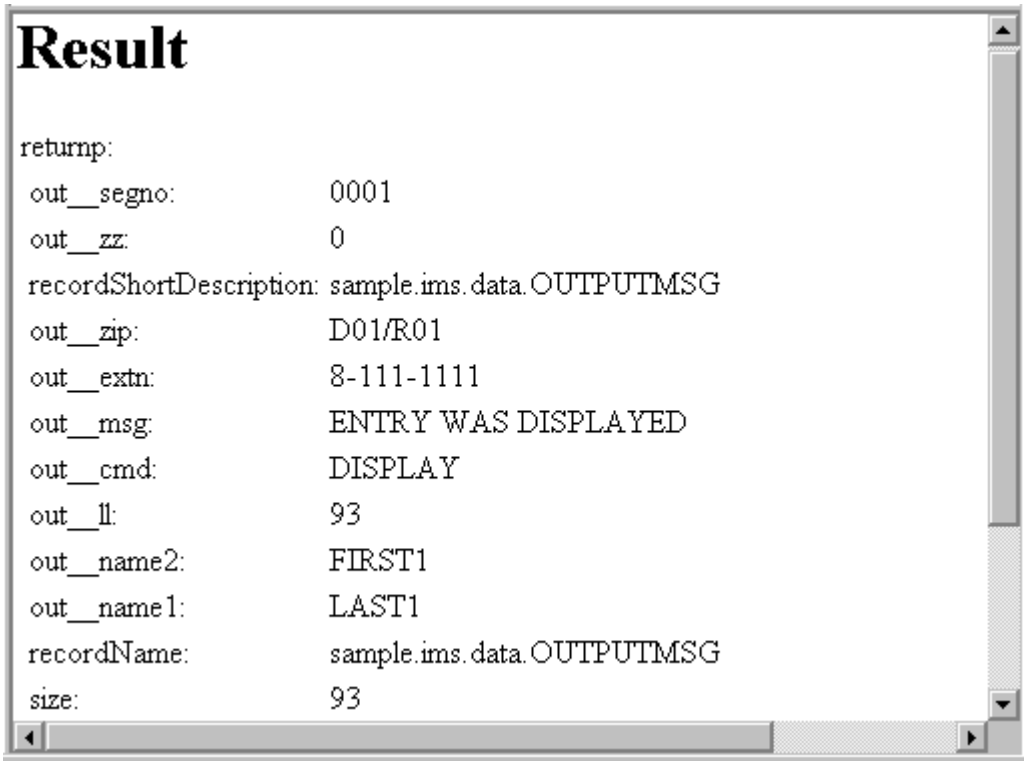
Now you are ready to begin Exercise 1.4: Deploying the application.

Exercise 1.4: Deploying the application

Before you begin, you must complete Exercise 1.3: Creating the Java method.

Creating a JSP

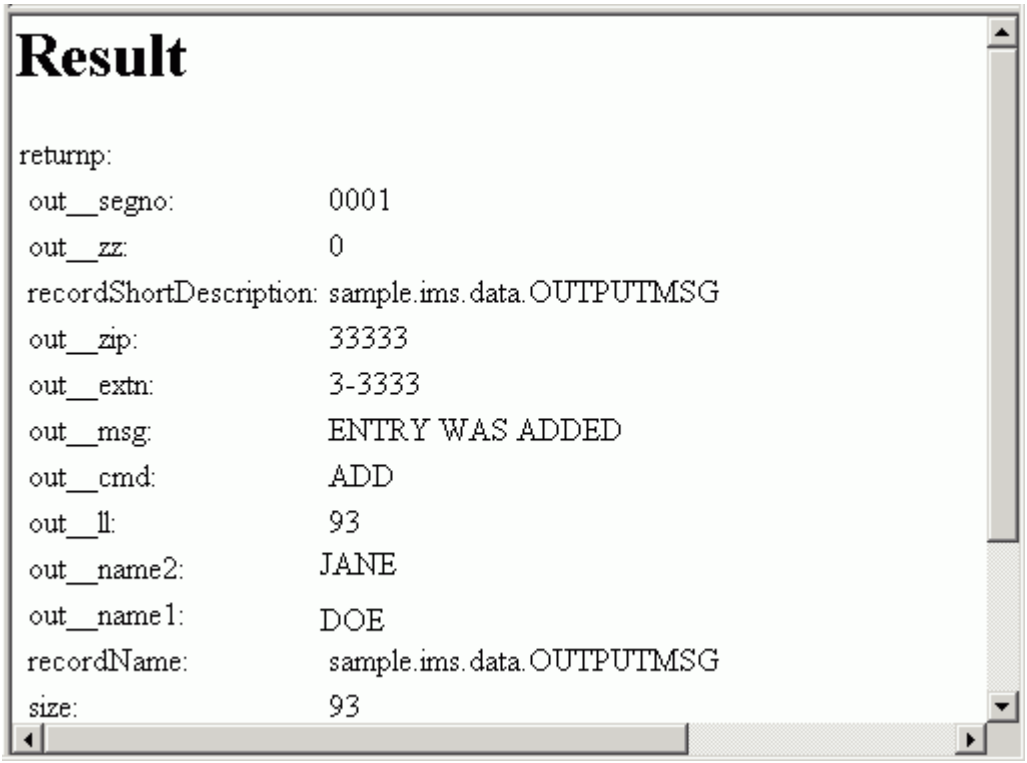
1. In the **Deployment Information** page, select **Create J2EE resource**.
2. In the **J2EE Resource Type**, select **JSP**. Click **Next**.
3. In the **JSP Creation** page, select **Generate simple JSPs with default input modes**.
4. In the **JSP folder** field, enter a JSP Folder name, such as SampleJSP. Click **Finish**.
5. Once the bean has been generated, you can run it on WebSphere Application Server. In the J2EE perspective, open Servers view, and right click **New** > **Server**.
6. Select **WebSphere V6. Server**. Click **Next**
   Note: If you do not see the WebSphere V6. Server option, change the **View by** option to **Name** to view the options.
7. Accept the default port number; if it is already in use, modify port settings if needed. Click **Next**.
8. Select `IMSPhoneBookEAR` from **Available projects**. Click **Add**.
9. Click **Finish**.
10. Start the server.
11. When the server is successfully started, right-click TestClient.jsp and select **Run on Server**.
12. A browser window with the **Test Client** will launch. Click on **runPhoneBook method**.
13. Use the following values as **Inputs**:
    - Enter IVTNO in In__trcd
    - Enter 0 in the In___zz field
    - Enter LAST1 in In__name1 field
    - Enter DISPLAY in In__cmd field
    - Enter 59 in In__ll field
    - Enter 93 in size field
14. Click **Invoke**, and this output will appear in the **Result** field.



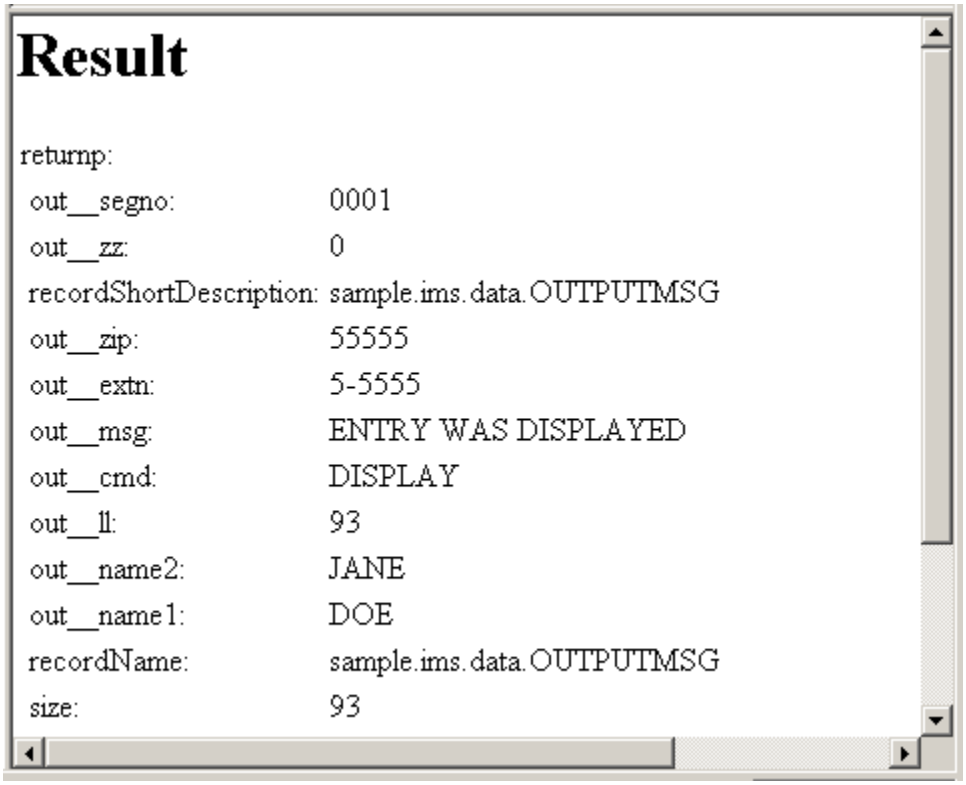
15. Now submit another cmd to add a phone book entry. Click on **runPhoneBook method**.
    - Enter 59 in In__ll field
    - Enter 0 in the In___zz field

- Enter IVTNO in In__trcd
- Enter Add in In__cmd field
- Enter Jane in In__name2 field
- Enter Doe in In__name1 field
- Enter 55555 in In__zip
- Enter 5-5555 in In__extn
- Enter 93 in the size field

16. Click **Invoke**, and this output will appear in the **Result** field.



17. Now submit another cmd to display the phone book entry you just added. Click on **runPhoneBook method**.
    - Enter 59 in In__ll field
    - Enter IVTNO in In__trd
    - Enter 0 in the In___zz field
    - Enter DISPLAY in In__cmd field
    - Enter Doe in In__name1 field
    - Enter 93 in the size field
18. Click **Invoke**, and this output will appear in the **Result** field.



Creating a Faces JSP to deploy the J2C Java bean

This section outlines the steps for deploying your J2C Java bean through a faces JSP.

1. Expand the IMSPhoneBook project, and find the WebContent folder.
2. Right click on WebContent folder in your IMSPhoneBook project and select **New** > **Other** > **Web** > **Faces JSP file**.
3. Give your new new faces JSP the name Test.
4. Accept defaults for all other fields.
5. Click **Finish**.

Adding the java bean to faces JSP

1. Once you have created the Faces JSP file, the page should open Test.jsp in the **Design** page. If it is not in the **Design** page of the editor, expand the WEB-INF folder under the WebContent folder. Right click on Test.jsp, click **Open With**, and click on Page Designer. Test.jsp will open in the Design page.
2. The Palette view should appear on the right panel. If it does not appear, in the top menu, click on **Window** > **Show view** > **Palette**.
3. In the Data folder of the Palette view, click on the JavaBean option of the **Palette**
4. Drag and drop the JavaBean to the Test.jsp editor; the Add JavaBean wizard will open.
5. Select **Add new JavaBean**.
6. In the **Name** field, type phonebookLookup
7. Click the open book icon, , beside the **Class** field. The Class Selection window appears.
8. In the Class Selection page, type **PhoneBookImpl** in the **Search** field
9. Clear the **Add input/output controls to display the JavaBean on the Web page** check box.
10. Click **Finish**.
11. You will see **PhoneBookImpl** in the Page Data view.

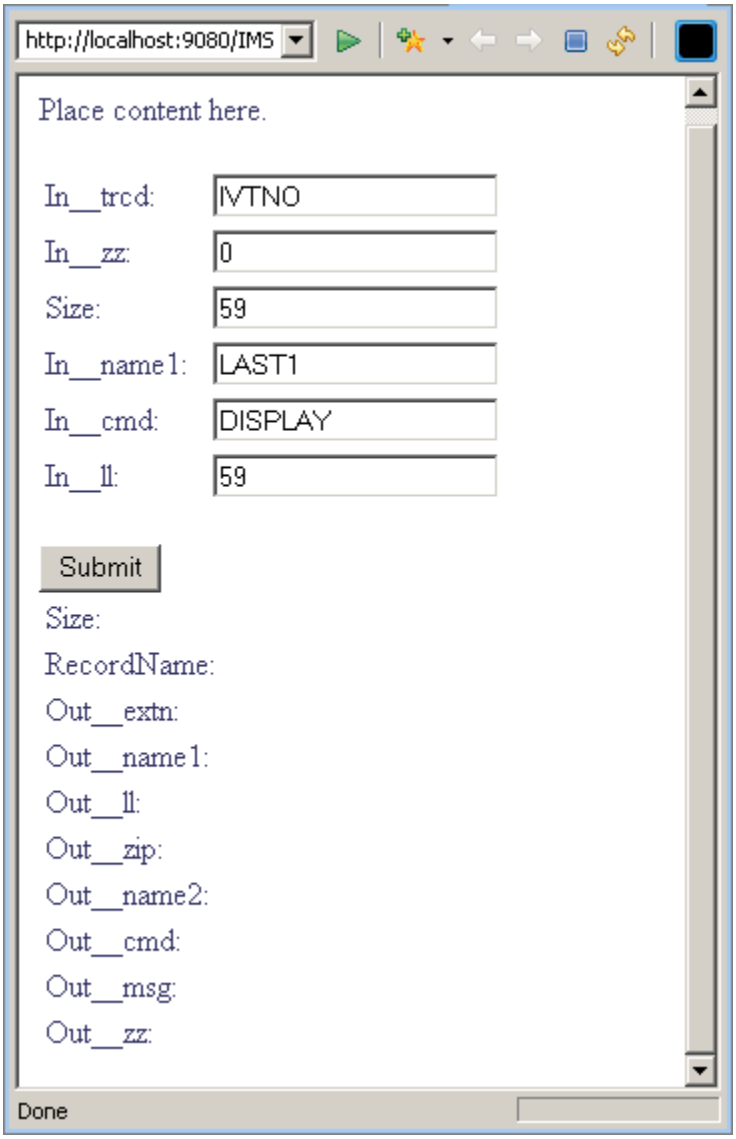
Adding input and output controls to the faces JSP

1. Right-click **phonebookLookup** Java Bean in the Page Data view, and click **Add New JavaBean Method**.
2. From the list of available methods, click on **runPhoneBook** .
3. Click **OK**.
4. Expand *phonebookLookup* Java Bean in the Page Data view, and select the runPhoneBook() method.
5. Drag and drop the runPhoneBook() method onto the editor. The Insert JavaBean wizard appears.
6. In the **Create controls for:** field, select **Inputting data**.
7. In the **Fields to display** field, select **None**, to clear the form.
8. In the **Fields to display** field, select these input fields
    - arg.in__trcd
    - arg.in__zz
    - arg.size
    - arg.in__name1
    - arg.in__cmd
    - arg.in__ll
9. Click **Finish**.
10. Accept defaults for the other fields.
11. Click **Next**.
12. In the Configure Date Controls page, select **Create controls for displaying the results**.
13. In the **Fields to display** field, select **None**, to clear the form.
14. In the **Fields to display** field, select these output fields
    - out__zz
    - out__zip
    - out__extn
    - out__msg
    - out__cmd
    - out__ll
    - out__name2
    - out__name1
    - size
15. Click **Finish**.
16. Save your Faces JSP page, by pressing **Ctrl-S** or by clicking **File** > **Save** in the toolbar.

Testing the Faces JSP

1. Select the **Servers** tab. Start the test server, if it is not already running. To start the server, right click on the WebSphere Application Server v6.0, and click **Start**.
2. Right click on Test.jsp (the faces JPS that you just created) in the Project Explorer view.
3. Select **Run** > **Run on Server**.
4. Select WebSphere Application Server v6.0 and click **Finish**
5. The browser will open to Test.jsp. Type these values in the text boxes:
    - Enter IVTNO in In__trcd
    - Enter 59 in In__ll field
    - Enter 0 in the In___zz field
    - Enter DISPLAY in In__cmd field
    - Enter LAST1 in In__name1 field
    - Enter 93 in size field

    Note: Ensure that there are no extra



6. Click **Submit**
7. You will see the output displayed in your browser.

Congratulations! You have completed the PhoneBook Tutorial.

Finish your tutorial by reviewing the materials in the Summary.

Summary

This tutorial has taught you how to use the J2C Java bean wizard to connect to an IMS server.

Completed learning objectives

If you have completed all of the exercises, you should now be able to

- Use the J2C Java bean wizard to create a J2C application that interfaces with an IMS transaction.
- Create a Java method, `runPhoneBook.java` , which accepts an individual's name and returns the individual's contact information.
- Create a JSP to deploy the application on WebSphere Application Server