Build a J2C Application for a CICS COBOL copybook: Same input and output

Time required

To complete this tutorial, you will need approximately **30 minutes**. If you decide to explore other facets of the J2C Java bean 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
- COBOL programming language
- CICS ECI server technology

Learning objectives

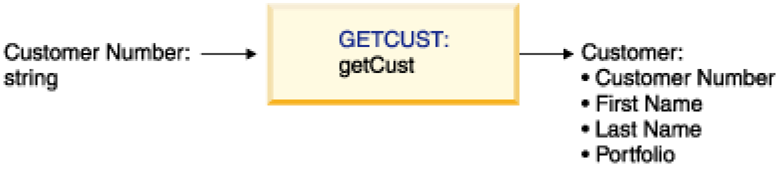
This tutorial is divided into several exercises that must be completed in sequence for the tutorial to work properly. This tutorial teaches you how to use the J2C Java bean wizard to connect to a CICS ECI server. While completing the exercises, you will:

- Use the J2C Java bean wizard to create a J2C application that interfaces with a CICS transaction using an External Call Interface (ECI).
- Create a Java method, getCustomer, which accepts a customer number and returns the customer's information
- Create a JSP to deploy the application on a WebSphere application server

When you are ready, begin Exercise 1.1: Using the J2C Java bean wizard to select the Resource Adapter

Exercise 1.1: Selecting the resource adapter

This tutorial will lead you through the detailed steps to generate a J2C application that interfaces with a CICS transaction using an External Call Interface (ECI). The service is built from a CICS COBOL function, `getCustomer`, which accepts a customer number and returns the customer's information, as shown in this diagram.



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

- **Connection to a CICS ECI server**: In this tutorial, your application interacts with a CICS program on a server. Specifically, you need to set up a CICS transaction gateway on a machine to access the server. You also need to perform some setup work on the CICS server machine, where you want the CICS to execute. These steps are not covered.
- **A copy of the COBOL file taderc99.cbl.** You may locate this file in your product installation directory: `\rad\eclipse\plugins\com.ibm.j2c.cheatsheet.content_6.0.0 \Samples\CICS\taderc99`. If you wish to store it locally, you can copy the code from here:

`taderc99.cbl`

```
identification division.
program-id. TADERC99.
environment division.
data division.
working-storage section.
LINKAGE SECTION.
01  DFHCOMMAREA.
    02  CustomerNumber     PIC X(5).
    02  FirstName  PIC A(15).
    02  LastName   PIC A(25).
    02  Street     PIC X(20).
    02  City       PIC A(20).
    02  Country    PIC A(10).
    02  Phone      PIC X(15).
    02  PostalCode PIC X(7).
procedure division.
start-para.
    IF CustomerNumber EQUAL '12345'
      move 'Alan' to FirstName
      move 'Turing' to LastName
      move '1150 Eglinton Ave.' to Street
      move 'New York' to City
      move 'USA' to Country
      move '(416) 444-4444' to Phone
      move '  94041' to PostalCode
    ELSE IF CustomerNumber EQUAL '44444'
      move 'Enrico' to FirstName
      move 'Fermi' to LastName
      move '11 Maple Ave.' to Street
      move 'Austin' to City
      move 'USA' to Country
      move '(416) 444-4444' to Phone
      move '  10121' to PostalCode
    ELSE
      move 'Mary' to FirstName
```

```
  move 'Poppins' to LastName
  move '51 Sweets Dr.' to Street
  move 'Chicago' to City
  move 'USA' to Country
  move '(416) 444-4444' to Phone
  move '  30326' to PostalCode
END-IF.
EXEC CICS RETURN
END-EXEC.
```

- **A clean workspace.**

Using the J2C Java bean wizard to select the resource adapter

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.

Connecting to the CICS ECI 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 page, under **View by**, select **JCA version**. Expand 1.5, and select **ECIResourceAdapter (IBM:6.0.0)** . Click **Next**.
4. In the Connection Properties page, select **Nonmanaged connection** check box. (For this tutorial, you will use the non-managed connection to directly access the CICS server, so you do not need to provide the JNDI name.) Accept the default Connection class name of `com.ibm.connector2.cics.ECIManagedConnectionFactory`. In the blank fields, provide connection information. Required fields are indicated by an asterisk (*):
    - **Server name**: (Not required) The name of the CICS Transaction Gateway server.
    - **Connection URL***: (Required) The server address of the CICS ECI server
    - **Port number**: (Not required) The number of the port that is used to communicate with the CICS Transaction Gateway. The default port is 2006.
    - **User name**: (Not required) The user name for the connection.
    - **Password**: (Not required) The password for the connection.

   You may obtain the connection information from your CICS Server system administrator.
5. 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. In the New J2C Java Bean page, type the value `CustomerProj` in the **Project Name** field.
2. Click the **New** button beside the **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**: CustomerProj
    - **Project location**: accept default
    - **Servlet version**: 2.4
    - **Target server**: WebSphere Application Server v6.0
    - **EAR Project**: CustomerProjEAR
    - **Context Root**: CustomerProj
6. Click **Finish**.
7. A dialog box may appear asking if you would like to switch to the Dynamic Web perspective. Click **Yes**.
8. In the New J2C Java Bean page, ensure that the following values appear:
    - In the **Package name** field, type `sample.cics`
    - In the **Interface name** field, type `Customer`.
    - In the **Implementation name** field, type `CustomerImpl`.
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 .

Creating a Java method

You will now create a Java method that will use the COBOL importer to map the data types between the COBOL source and the data in your 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 `getCustomer` for the name of the operation. Click **Next**.

Creating the input and output data mapping between COBOL and Java

In this step, you will import the taderc99.cbl (COBOL) file that is needed to create your application. The taderc99.cbl file is located in <RSDP_installdir>\rad\eclipse\plugins\com.ibm.j2c.cheatsheet.content_6.0.0 \Samples\CICS\taderc99, where <RSDP_installdir> is the directory where this product is installed. The COBOL file contains the program that runs on the CICS server. It has the definition of the structure to be passed to the CICS server via the communications area (COMMAREA). This structure represents the customer records being returned from the CICS application. 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**.
3. Click **Browse** beside the **Cobol file** field.
4. Locate the `taderc99.cbl` file in the file system, and click **Open**.
5. Click **Next**.
6. In the COBOL Importer page, select a communication data structure.
   - Select Win32 for **Platform Name**.
   - Select ISO-8859-1 for **Code page**
   - Click **Query**.
   - Select **DFHCOMMAREA** for **Data structures**.
7. In the Saving properties page,
   - Select **Default** for **Generation Style**.
   - Click **Browse**.
   - Select the Web project **CustomerProj**. Click**Open**.
   - In the **Package Name** field, enter sample.cics.data
   - In the **Class Name** field, enter CustomerInfo. Click **Finish**.
8. In the Java method page, select **Use Input for output**. Click **Finish**.
9. On the Java Method page, click **Finish** to complete the operation.
10. In the **functionName** field, type the COBOL program id (`TADERC99`). Click **Next**.

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

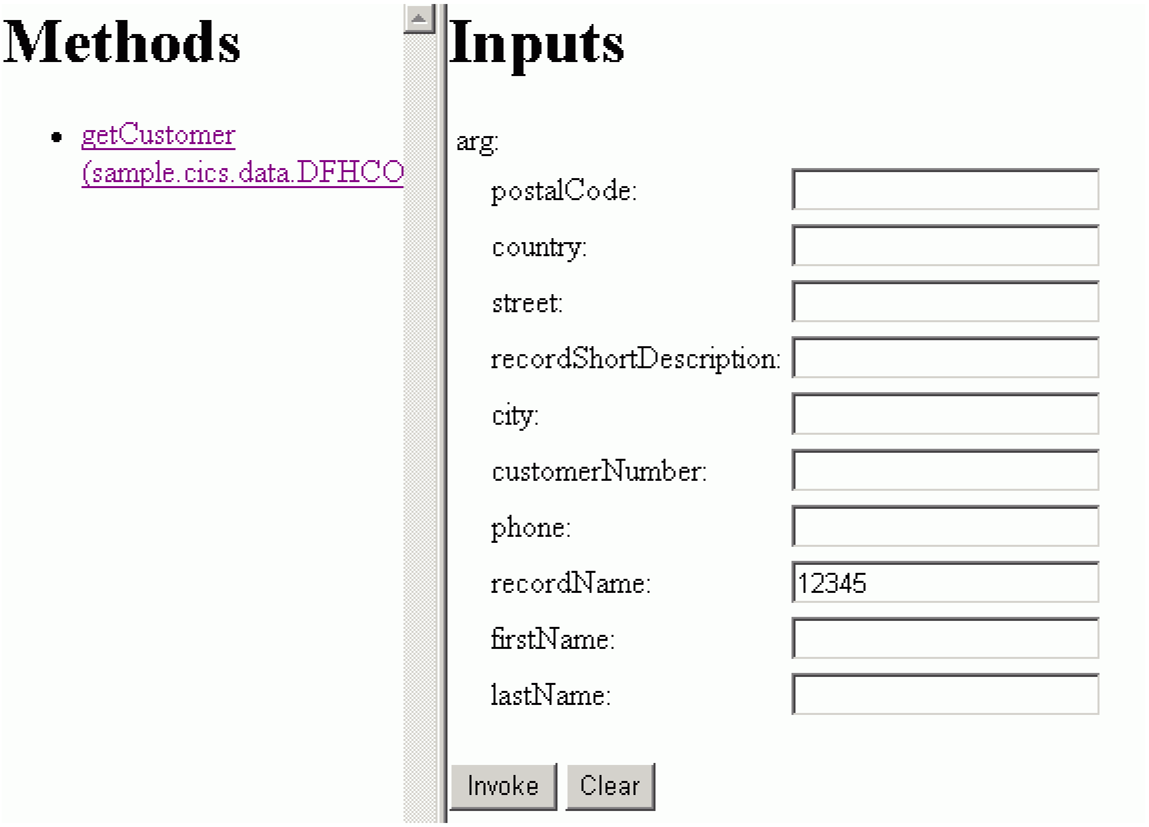
Exercise 1.4: Deploying the application

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

Creating a JSP

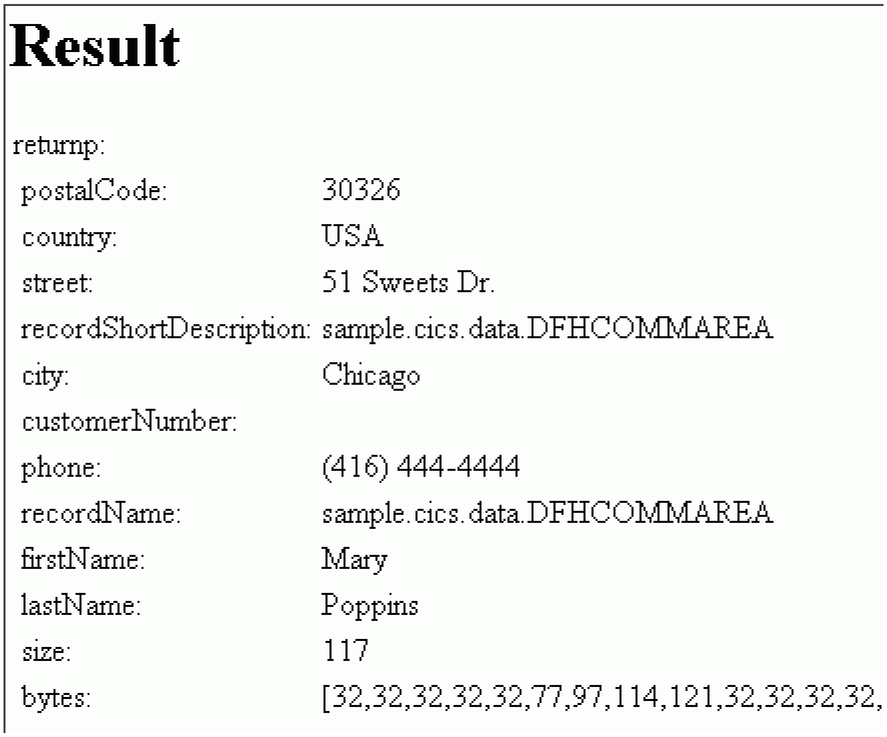
1. In the **Deployment Information** page, select **Create J2EE resource**.
2. In the **J2EE Resource Type**, select **JSP**. Click **Next**.
3. In the **JSP Creation** page, select **Generate simple JSPs with default input modes**.
4. In the **JSP folder** field, enter a JSP Folder name. Click **Finish**.
5. In the **J2EE** perspective, open **Servers** view, Right click **New** > **Server**.
6. Select **WebSphere Application Server V6 Server**. Click **Next**
   NOTE: If you do not see the **WebSphere Application Server V6 Server**, go to the **View by** field, and select **name**.
7. Accept the default port number; if it is already in use, modify the port settings.
8. Click **Next**.
9. Select `CustomerProjEAR` from **Available projects**. Click **Add**.
10. Click **Finish**.
11. Start the server.
12. When server is successfully started, right click on **TestClient.jsp** and select **Run on Server**.
13. A browser window with the Test Client will be launched. Click on the **getCustomer** method.

- Enter 12345 in customer_id



recordName: 12345

- Click **Invoke**. The customer information will appear in the output console.



```
Result

returnp:
 postalCode:             30326
 country:                USA
 street:                 51 Sweets Dr.
 recordShortDescription: sample.cics.data.DFHCOMMAREA
 city:                   Chicago
 customerNumber:
 phone:                  (416) 444-4444
 recordName:             sample.cics.data.DFHCOMMAREA
 firstName:              Mary
 lastName:               Poppins
 size:                   117
 bytes:                  [32,32,32,32,32,77,97,114,121,32,32,32,32,
```

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

Adding the Java bean to a Faces JSP

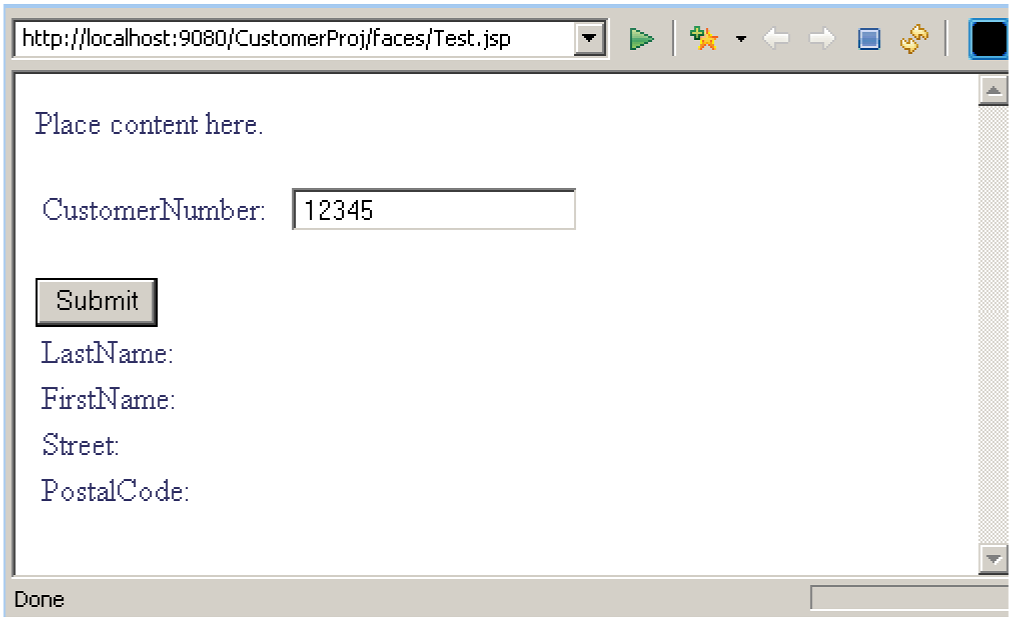
1. Once you have created the Faces JSP file, the page should open Test.jsp in the **Design** page of the editor. If it is not in the **Design** page, expand the WEB-INF folder under the WebContent folder. Right click on Test.jsp, click **Open With**, and click on **Page Designer**. Test.jsp will open in the Design page of the editor.
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 JavaBean option of the **Palette.**
4. Drag and drop the JavaBean to the Test.jsp Design window; the Add JavaBean wizard will open.
5. Select **Add new JavaBean.**
6. In the **Name** field, type `customerLookup`.
7. Click the open book icon, 📖, beside the **Class** field. The Class Selection dialog will appear. Type **CustomerImpl** in the search field and click on the found class. Click **OK**.
8. In the Class Selection page, type **CustomerImpl** in the **Search** field
9. Uncheck **Add input/output controls to display the JavaBean on the web page**.
10. Click **Finish**.
11. You will see **CustomerImpl** in the Page Data view.

Adding input and output controls to the Faces JSP

1. Right click on **customerLookup** Java Bean in the Page Data view, and click **Add New JavaBean Method**.
2. From the list of available methods, click on **getCustomer** .
3. Click **OK**.
4. Expand **customerLookup** Java Bean in the Page Data view, and select the getCustomer() method.
5. Drag and drop the getCustomer() method onto the JSP's canvas.
6. The **Insert JavaBean** wizard will appear. The Configure Date Controls page allows you to select data fields.
7. In the **Create controls for:** field, select **Inputting data**.
8. In the **Fields to display** field, select **None**, to clear the form.
9. In the **Fields to display** field, select the field that has the label **arg.customerNumber.**.
10. Accept defaults for other fields.
11. Click **Next**.
12. In the Configure Data 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 the output fields LastName, FirstName, Street, and PostalCode.
15. Click **Finish**.
16. Save your Faces JSP 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 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. In the **CustomerNumber** field, type 12345.


http://localhost:9080/CustomerProj/faces/Test.jsp

Place content here.

CustomerNumber: 12345

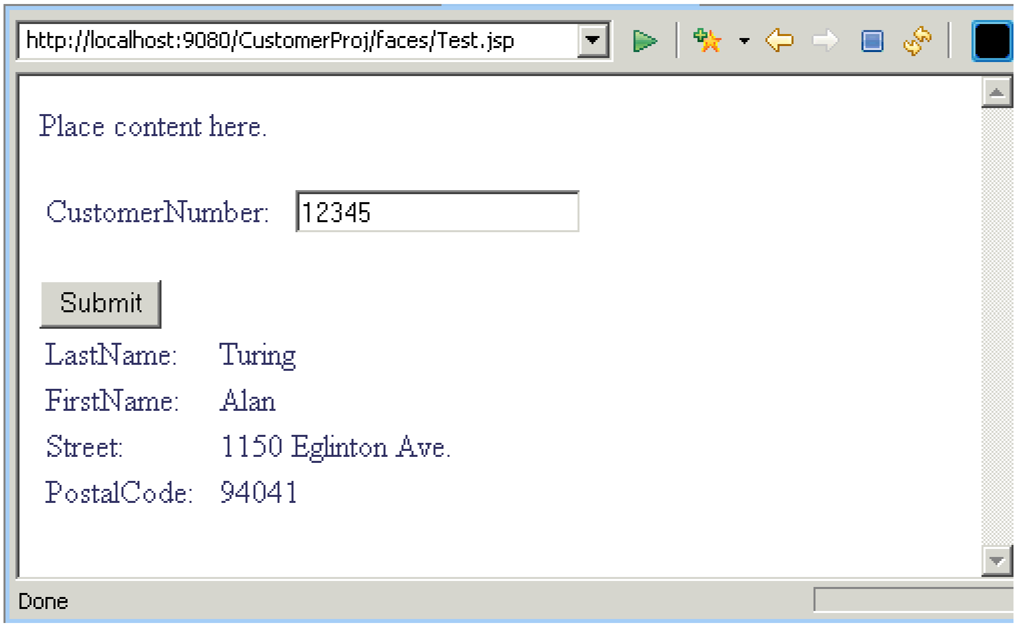
LastName:

FirstName:

Street:

PostalCode:

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



http://localhost:9080/CustomerProj/faces/Test.jsp

Place content here.

CustomerNumber: 12345

LastName: Turing
FirstName: Alan
Street: 1150 Eglinton Ave.
PostalCode: 94041

Testing the application using the TestCustomer program

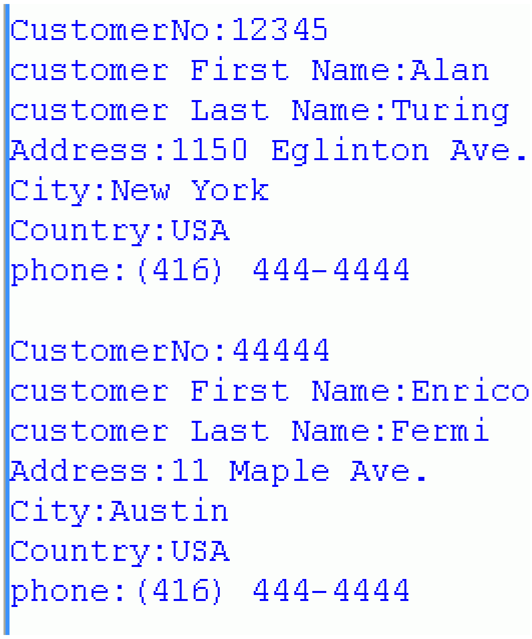
1. Expand the **CustomerProj** > **Java Resources** > **JavaSource**.
2. Right click on the **sample.cics** package.
3. Select **New** > **class**.
4. Type TestCustomer in the **Name** field.
5. In the Java editor window, replace all the code in your workspace with the following code:

TestCustomer.java

```
/**************************************************************************
 * Licensed Materials - Property of IBM
 *
 * com.ibm.j2c.cheatsheet.content
 *
 * Copyright IBM Corporation 2004. All Rights Reserved.
 *
 * Note to U.S. Government Users Restricted Rights:  Use, duplication or di
 **************************************************************************
/*
 * Created on Aug 30, 2004
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
package sample.cics;
import sample.cics.data.*;

/**
 * @author ivyho
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class TestCustomer {

        public static void main(String[] args) {
                try {

                        CustomerInfo input = new CustomerInfo();
                        input.setCustomerNumber("12345");

                        CustomerImpl proxy = new CustomerImpl();
                        CustomerInfo output = proxy.getCustomer(input);
                        System.out.println("\nCustomerNo:"+output.getCusto


                         input.setCustomerNumber("44444");

                         proxy = new CustomerImpl();
                         output = proxy.getCustomer(input);

                         System.out.println("\nCustomerNo:"+output.getCust


                }catch (Exception e)
                {
                        e.printStackTrace();
                }



        }
}
```

6. In the Project Explorer view, right click on the TestCustomer.java file. Select **Run** > **Run As Application**.

   You will see the following in the console:



```
CustomerNo:12345
customer First Name:Alan
customer Last Name:Turing
Address:1150 Eglinton Ave.
City:New York
Country:USA
phone:(416) 444-4444

CustomerNo:44444
customer First Name:Enrico
customer Last Name:Fermi
Address:11 Maple Ave.
City:Austin
Country:USA
phone:(416) 444-4444
```

Congratulations! You have completed the CICS Taderc99 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 a CICS ECI 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 a CICS transaction using an External Call Interface (ECI).
- Create a Java method, getCustomer, which accepts a customer number and returns the customer's information
- Create a JSP to deploy the application on a WebSphere application server