Create a J2C application for an IMS MFS-based transaction using IMS MFS SOA Support

Skill Level: Intermediate

Maria G. Querales (querales@us.ibm.com)
Software Developer, IBM®

Kevin D. Kelley (kdkelley@us.ibm.com)
Information Developer, IBM

Shahin Mohammadi-Rashedi (shahin@us.ibm.com)
IMS™ SOA Demonstration Team Lead

IBM Corporation
15 March 2009

Abstract

In this tutorial you will use IBM Rational® Developer for System z® Software Version 7.5 to build a J2C Java™ Bean based on the input and output formats defined in the MFS source of an IMS transaction. Once the Java Bean is created, you will reuse the invoke and use the IMS business logic from a Web application such as JavaServer Pages component (JSP).

About this tutorial

This tutorial will take you through the steps of using the Java EE Connector (J2C) component of IBM Rational Developer for System z and IBM IMS MFS SOA Support and will help you to familiarize yourself with the Java Platform Enterprise Edition (Java EE, previously known as J2EE) development environment and the J2C. You will be working with the sample Phonebook IMS transaction (IVTNO), which is one of the IMS installation verification programs shipped with IMS.

Rational Developer for System z
Is a superset of Rational Application Developer. The Java EE perspective, J2C wizard and Web tools that you will use in this tutorial are also available in Rational Application Developer.

Rational Application Developer
Simplifies application development for enterprise information systems (EIS) such as IMS by providing wizard-based tools and a list of adapters ready to use. Rational Application Developer provides J2C wizards that enable you to create J2C applications, either as standalone programs or as added function to existing applications.

Java EE Connector Architecture

Java EE Connector Architecture (JCA) is a standard for connecting a Java-based technology solution for connecting application servers and EIS.

J2C JavaBeans™
A JavaBean that communicates with an EIS through JCA.

J2C Application
A typical J2C application consists of a J2C JavaBean with one or more methods that call EIS functions. For IMS, the input and outputs to these functions are data binding classes. After you have created a J2C JavaBean, you can create Web pages, an EJB, or a Web service for the J2C JavaBean.

MFS
Message Format Service (MFS) enables application programmers to specify screen formats, application input and output fields, and various device characteristics that define the end-user interface to an IMS transaction. While MFS manages device-specific information, the IMS transaction defines the application logic.

IMS MFS SOA Support
Next generation of MFS Web services support introduced in IBM Rational Application Developer for WebSphere® Software Version 7.5 (Rational Application Developer). Designed to use the latest programming model, J2EE Connector (J2C) that is supported in Rational Application Developer.

Objectives

To gain hands-on experience extending IMS MFS-based applications to the Web as a part of a Web page. Software tools that are part of Rational Developer for System z and IMS MFS SOA Support make the transformation processes easy, as the tutorial will demonstrate.

Upon completion of this tutorial, you will be able to:

- Use Rational Developer for System z and its built-in J2C tools.
- Enable an IMS MFS-based application as J2C JavaBean
- Generate a JSP for a J2C JavaBean

System requirements for the tutorial:

For purposes of this demo, you will be using IBM Rational Developer for System z Software Version 7.5 with IBM WebSphere Application Server Version 7. If you decide to test MFS SOA Support in your local environment, use the following software and platform:

- Software installed on Microsoft® Windows®
    - Rational Developer for System z Software Version 7.5.0 iFix001
    - WebSphere Application Server Version 6.1
- System software installed on IBM z/OS®
    - IMS Version 9 or Version 10
    - IMS Connect Version 9 or Version 10
    - OTMA
    - TCP/IP

Checklist for the first time implementation

You may find it helpful to have the following checklist available before proceeding with your own implementation for the first time.

	Your environment	**This tutorial:**
MFS Source file(s)	This can be obtained from IMS application programmers.	C:\mfs_source_file\dfsivf1.mfs
IMS Connect host name (or IP address) and port number.	This can be obtained from IMS system programmers.	Host name: ZSERVEROS.DFW.IBM.COM Port number: 9999
IMS Data store.	This can be obtained from IMS system programmers.	IMSC
Workspace directory and project name will be used by Rational Developer for System z when generating artifacts.	A naming standard is recommended.	C:\Workspaces7.5\SANDBOX

Overview of development tasks

To complete this tutorial you will perform the following tasks:

Task 1: Create the J2C JavaBean for the IMS MFS-based transaction

For the first task of the lab you will create a J2C JavaBean that is based on the input and output definitions of an MFS-based IMS application. The definitions reside in the MFS source file of the IMS application (you will import this MFS source file as part of task 1). You will use these definitions to populate the service definition, including the XML schema, for both input and output types.

Task 2: Create a JSP for the J2C JavaBean

For task 2 you will use a JSP file to embed the J2C JavaBean that you created in Task 1. You need to embed the J2C JavaBean into a JSP because a J2C JavaBean is like a method call... it is not an application. The JSP is the actual application that displays things as a Web page.

> **What is a JSP?** A Java Server Page (JSP) is a Java technology that allows software developers to dynamically generate HTML, XML or other types of documents in response to a Web client request.

Task 3: Test the J2C application using the JSP on IBM WebSphere Application Server

In Task 3 you test what you have built. You will use the JSP you created in Task 2 (TestClient.jsp) to test the J2C application on IBM WebSphere Application Server.

Figure 1 shows how to create and test a J2C application for an IMS MFS-based transaction using IMS MFS Support.

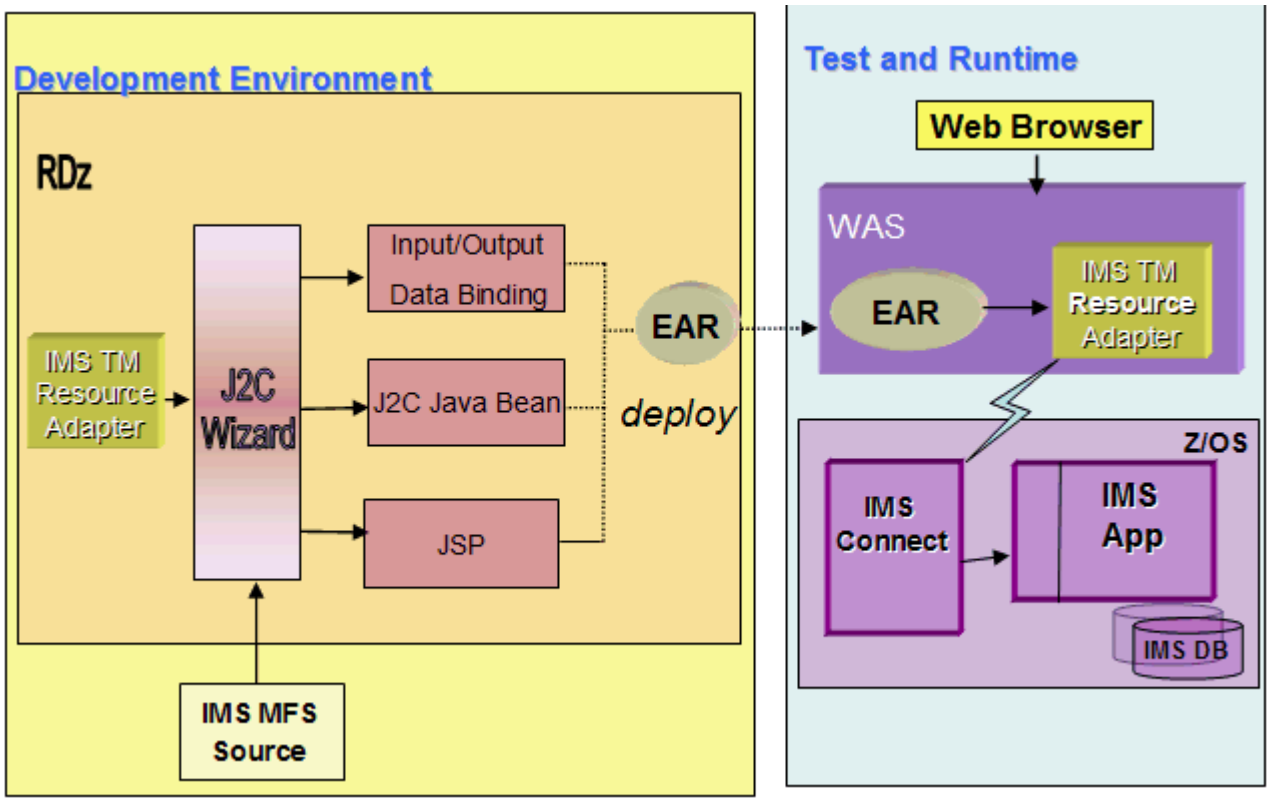

Figure 1. Using RDz and system z to accomplish lab objectives

Task 1: Create the J2C JavaBean for the IMS MFS-based transaction

KEY POINTS

What are you about to do in Task 1?

For the first task of the lab you will create a J2C JavaBean that is based on the input and output definitions of an MFS-based IMS application. These definitions reside in the MFS source file of the IMS application and they will be used to populate the service definition, including the XML schema for both input and output types.

Item	Description
MFS Source file	The MFS Source file (dfsivf1.mfs) defines the structure of the input and output messages of the IMS IVP Phonebook application.
MID	MFS input formatting occurs when a message input descriptor (MID) name is provided with an input message.
MOD	The MFS message output descriptor (MOD) can supply a MID name to be used for formatting the next input message.

1.1 ⏭ The IBM Rational Developer for System z V 7.5 is started and you are using the Workspaces7.5 \SANDBOX.

Important!

This Lab uses the directory ***C:\Workspaces7.5\SANDBOX*** as your work workspace.

What is a workspace?

A workspace is a directory that stores files for your projects. You can select your own directory or take the default directory. Artifacts created by Rational Developer for System z will be stored in this directory.

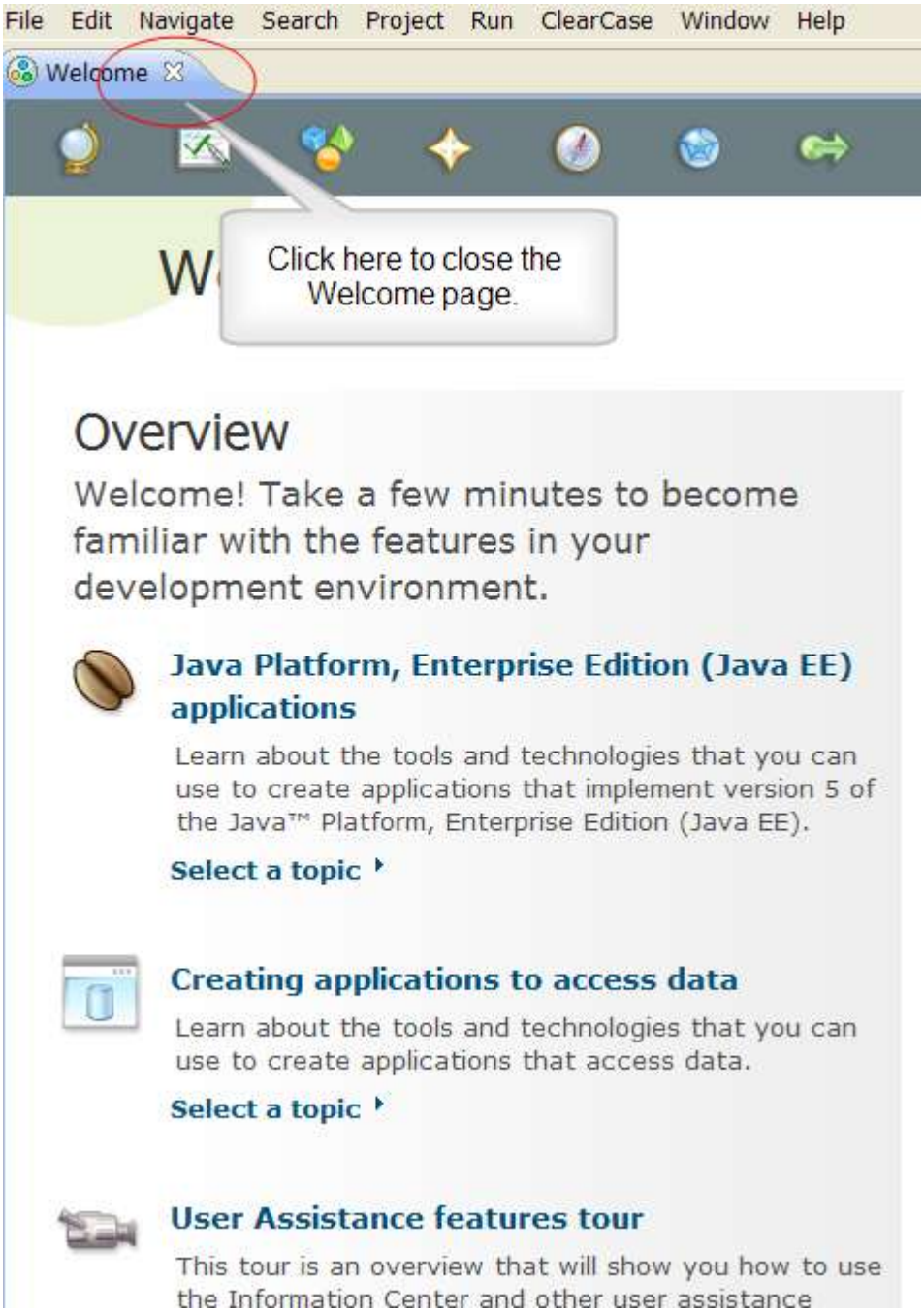

Figure 2. The Welcome page

1.2 ⏭ If the Welcome page displays you can close it from the Welcome tab. Open the Java EE perspective on the Rational Developer for System z workspace.

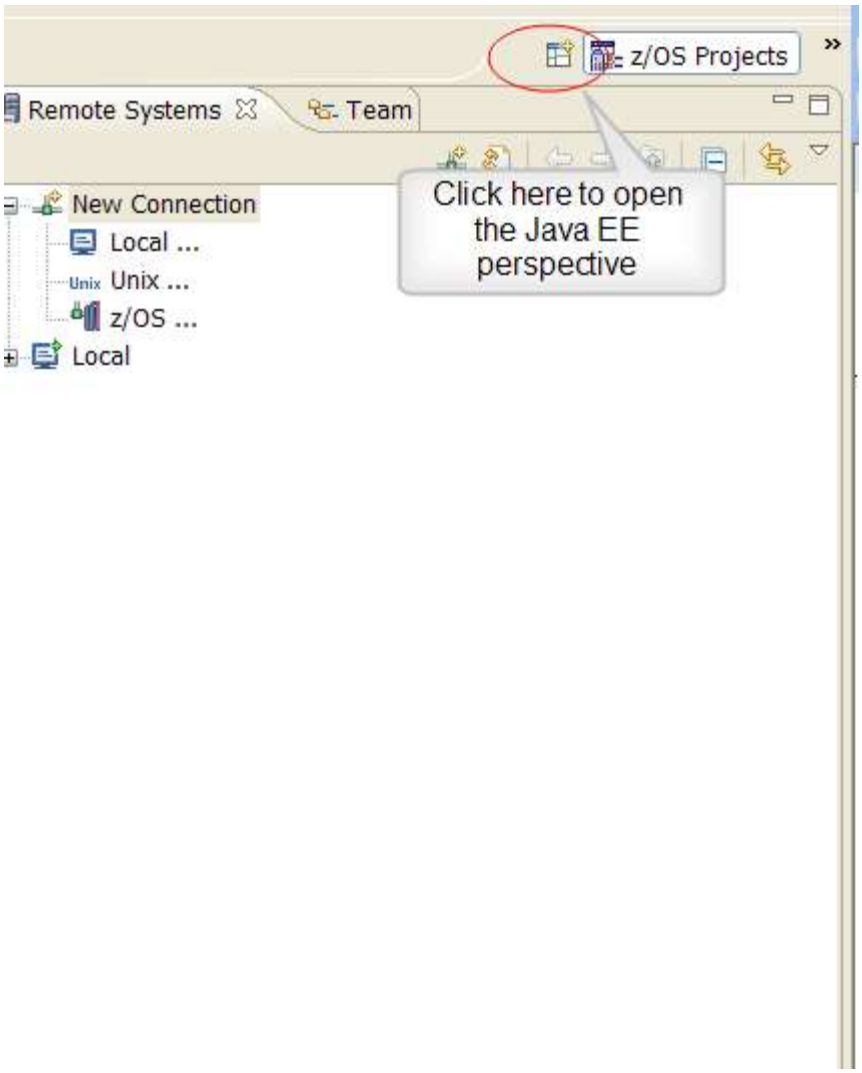

Figure 3. Setting the Java EE perspective

1.3 ⏭ You can also close the Welcome to z/OS projects panel, if it is displayed.
The z/OS Projects – IBM Rational Developer for System z workspace displays.

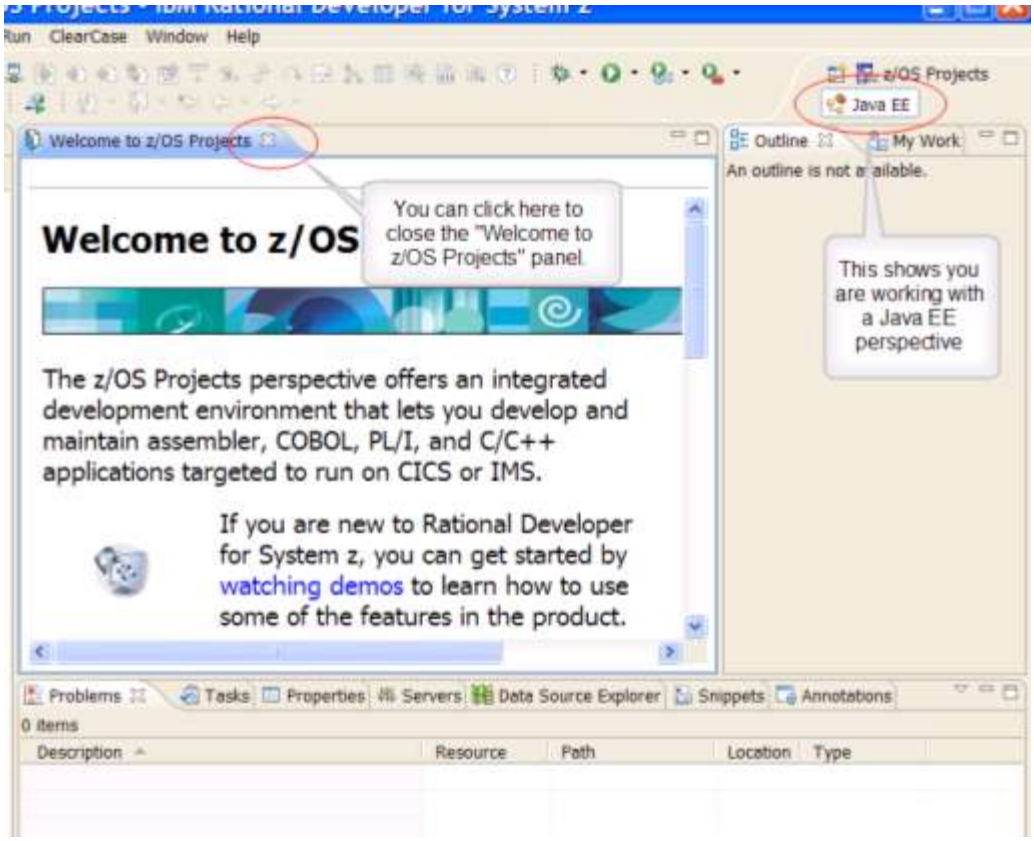

Figure 4. The IBM Rational Developer for System z workspace

What is Java EE ?

The Java Platform, Enterprise Edition (previously known as Java 2 Platform, Enterprise Edition, or J2EE) provides a standard for developing component-based, multi-tier, enterprise applications.

A Java EE application system typically includes the following tiers:

- **Client tier**: In the client tier, Web components, such as servlets and JSPs, or standalone Java applications provide a dynamic interface to the middle tier.
- **Middle tier**: In the server tier, or middle tier, enterprise beans and Web Services encapsulate reusable, distributable business logic for the application. These server-tier components are contained on a Java EE Application Server, which provides the platform for these components to perform actions and store data.
- **Enterprise data tier**: In the data tier, the enterprise's data is stored and persisted, typically in a relational database.

J2EE applications are comprised of components, containers, and services. Web components, such as servlets and JSPs, provide dynamic responses to requests from a Web page. EJB components contain server-side business logic for enterprise applications. Web and EJB component containers host services that support Web and EJB modules.

1.4 From the Rational Developer for System z workspace, click File > New > Other to start the wizard.

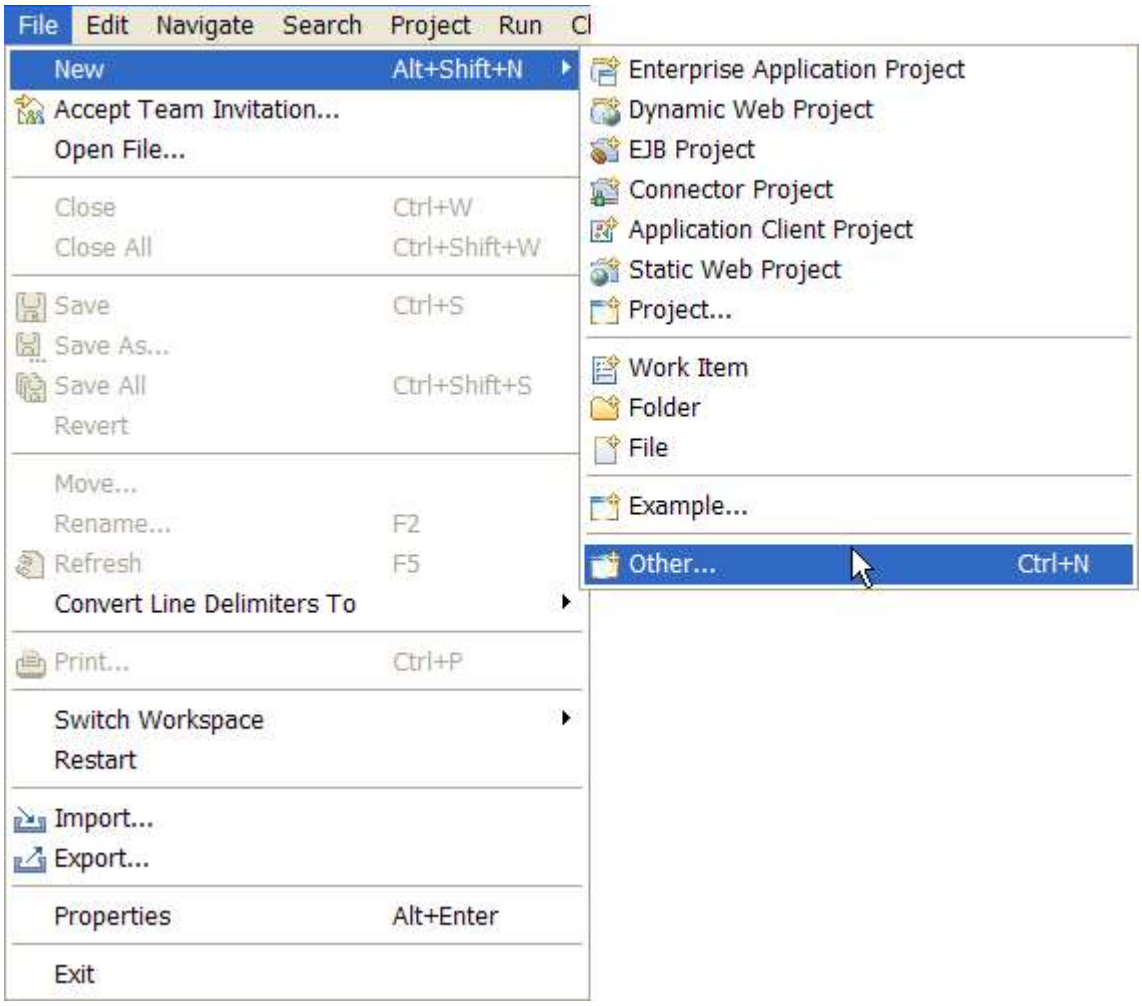

Figure 5. The menu path for selecting a wizard

In this section you will create a Java bean that communicates with IMS using J2C. Here we will define whether the Java bean is a managed or non-managed resource, along with defining the TCP/IP address, port and IMS datastore name.

The Select a wizard page opens.

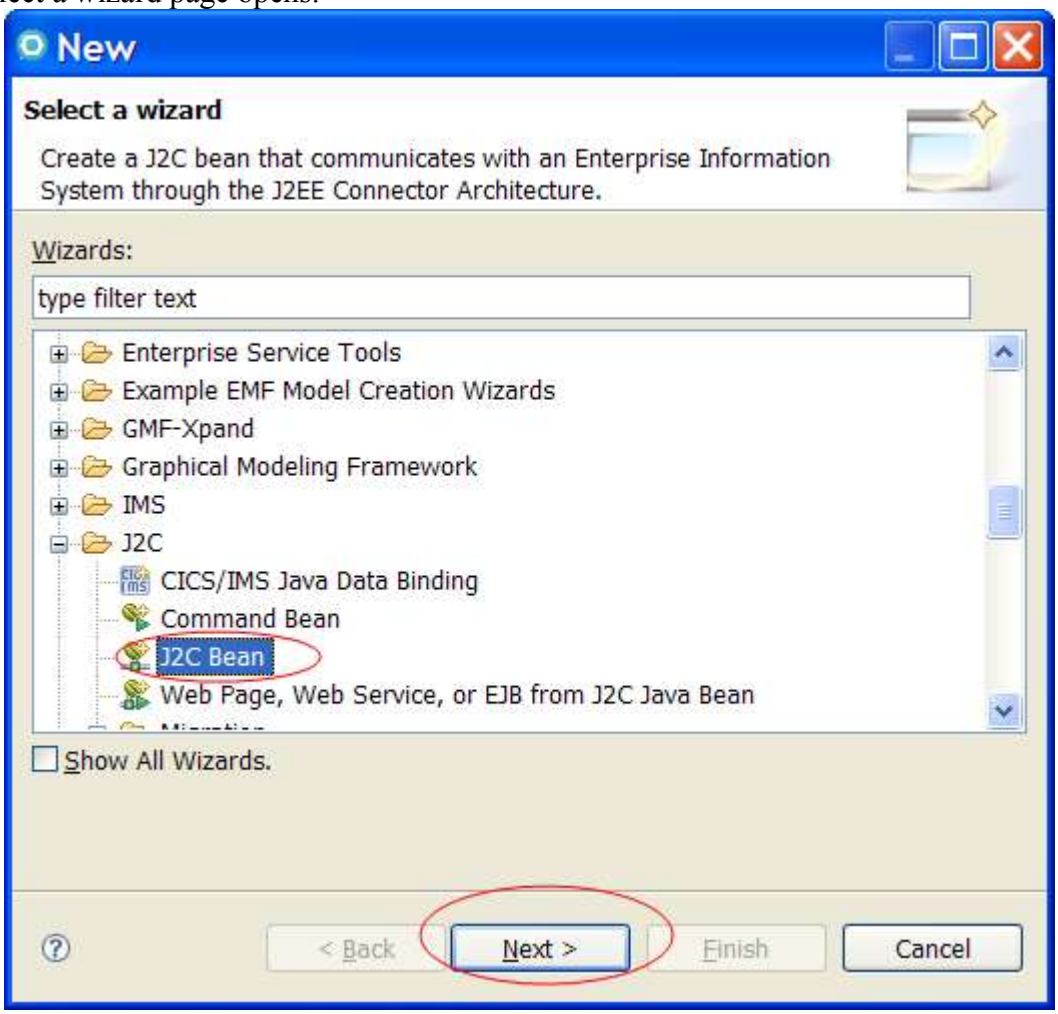

Figure 6. The Select a wizard page

1.5 ⏭ Expand the J2C folder and select the J2C Bean wizard. Click Next. The Resource Adapter Selection page opens.

Why J2C Bean?

You will use this wizard to create a JavaBean that communicates with IMS through JCA. J2C wizards enable you to create J2C applications, either as standalone programs or as added function to existing applications.
The wizard:

- dynamically imports your selected resource adapter
- enables you to set the connection properties to connect to the EIS servers
- guides you through the file importing and data mapping steps
- facilitates the creation of Java classes and methods to access the transformed source data

1.6 ⏭ Expand the IMS TM folder and choose the IMS Connector for Java (IBM: 9.1.0.2.5a) for your project. Click Next. The Connector Import page opens.

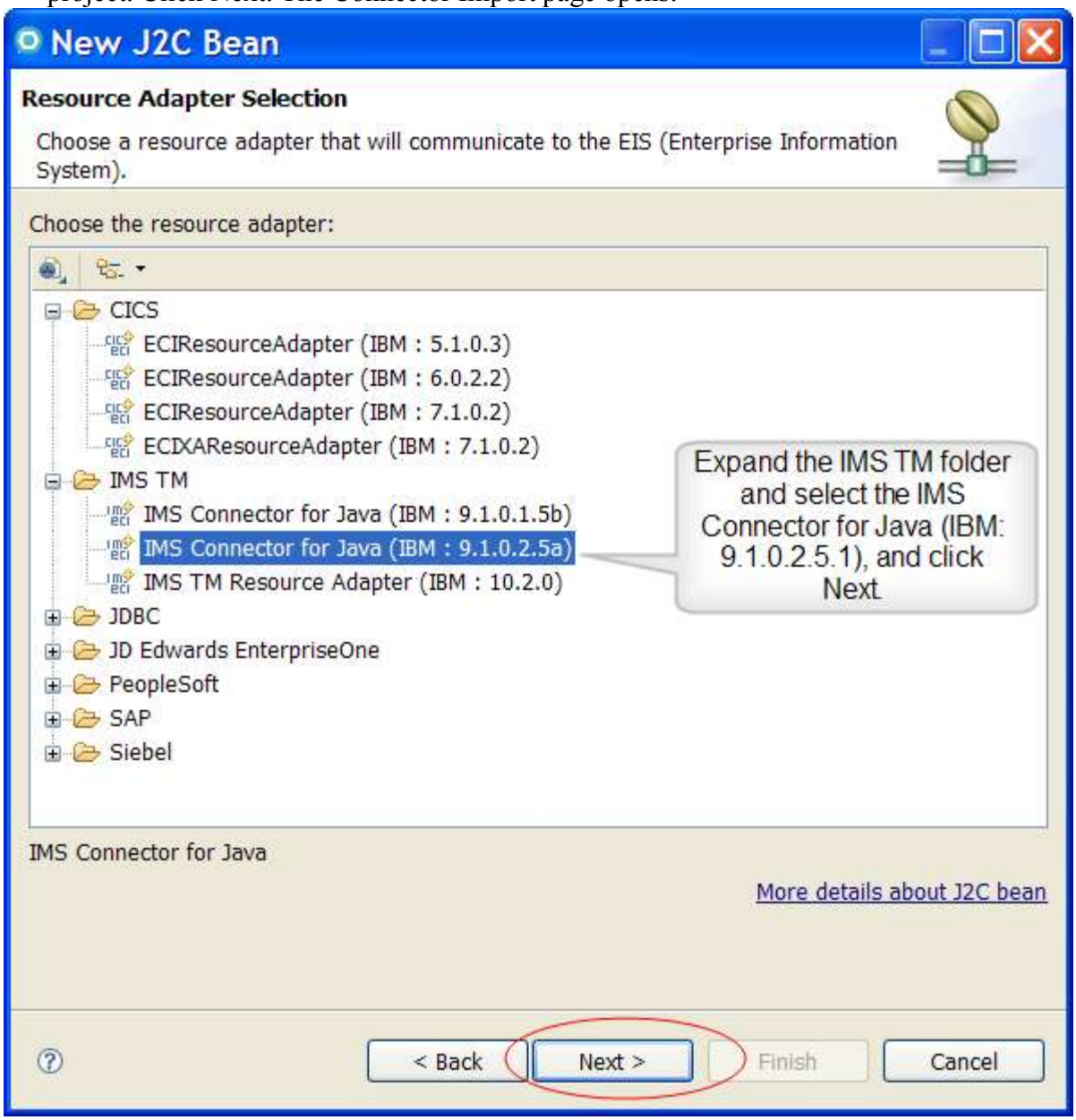

Figure 7. The Resource Adapter Selection page

What is a resource adapter?

Resource adapters allow your application to communicate with the enterprise information system (EIS). A resource adapter is a system-level software driver that is used by a Java application to connect to an EIS. The resource adapters reside on the application server and provide connectivity between the EIS, the application server, and the enterprise application. Applications deployed on the application server communicate with the resource adapter using the Common Client Interface (CCI). The RAR file (a type of archive file format) contains all the information necessary for installing, configuring and running a Resource Adapter. Resource Adapters comply

with the J2EE Java EE Connector Architecture (**J2C**). In this lab we are using IMS TM Resource Adapter to connect to IMS.

IMS TM Resource Adapter Version 10 is based on the newer JCA 1.5 standards, and therefore it is located under the 1.5 section.

Rational Developer for System z supports Java EE Connector Architecture (JCA) version 1.0 and version 1.5.

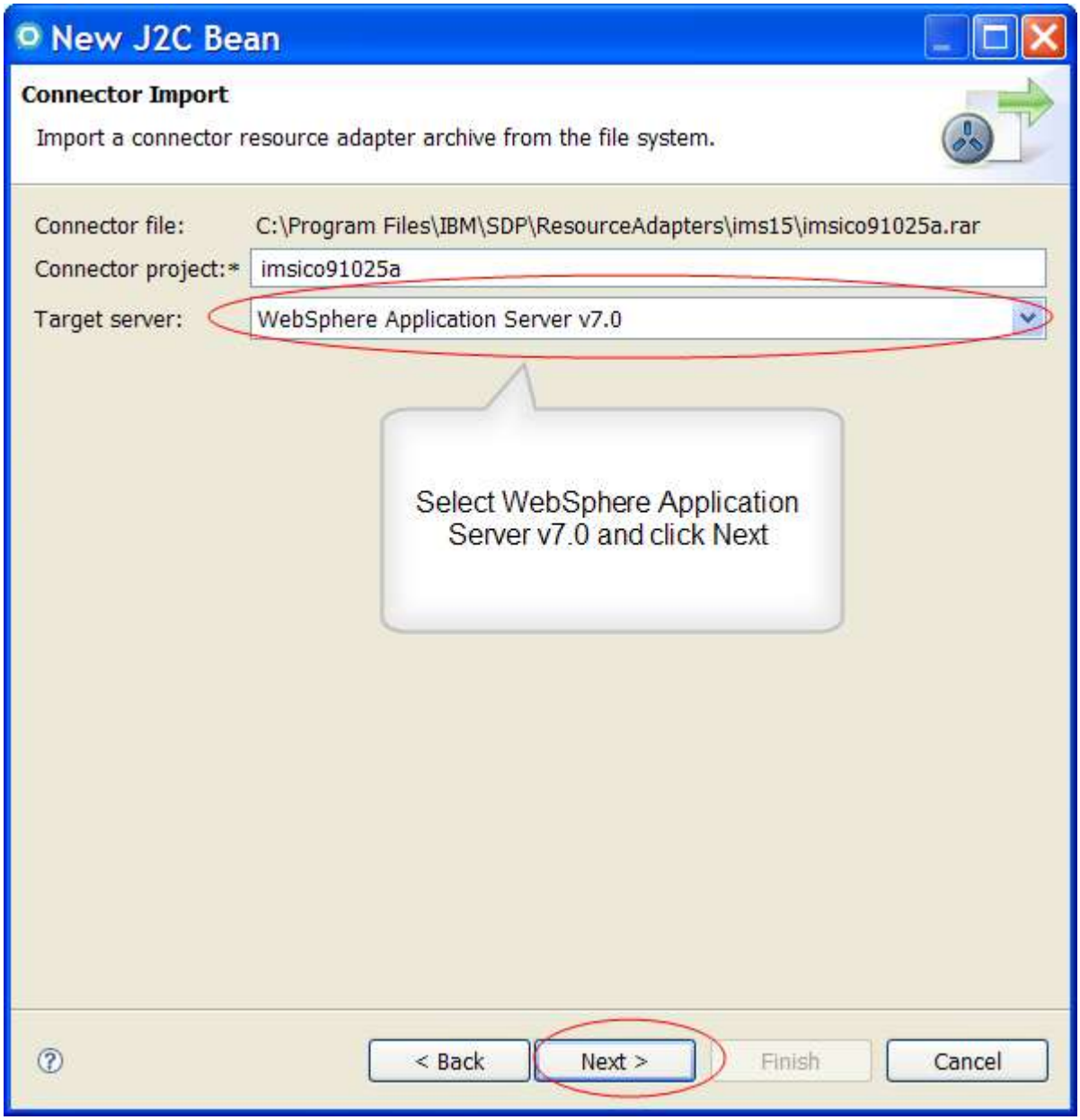

Figure 8. The Connector Import page

1.7 ⏭ Select WebSphere Application Server v7.0 runtime environment from Target server, and click Next. The Scenario Selection page opens.

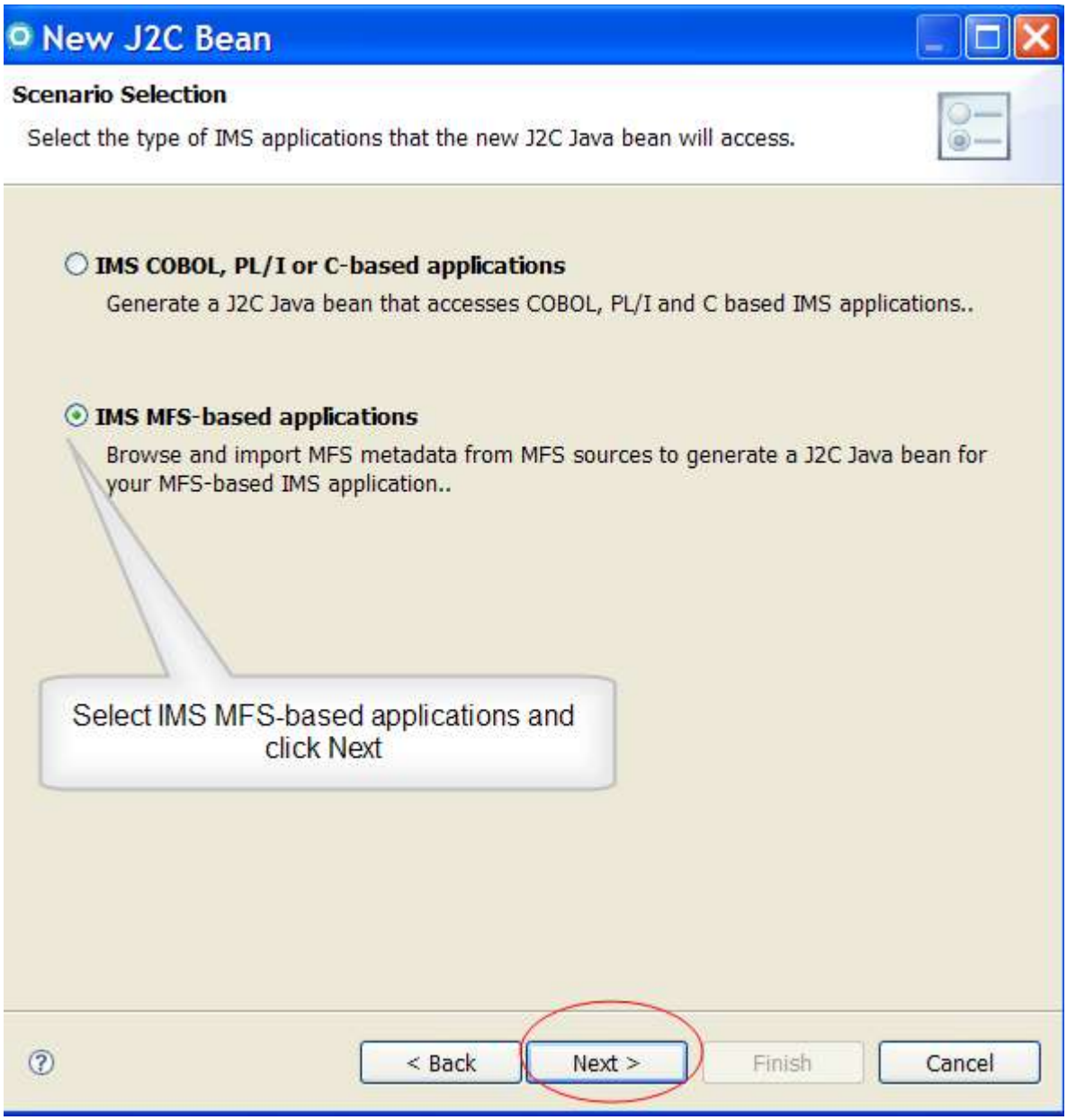

Figure 9. The Scenario Selection page

1.8 ⏭ Select IMS MFS-based applications and click Next. The Discovery Configuration page opens.

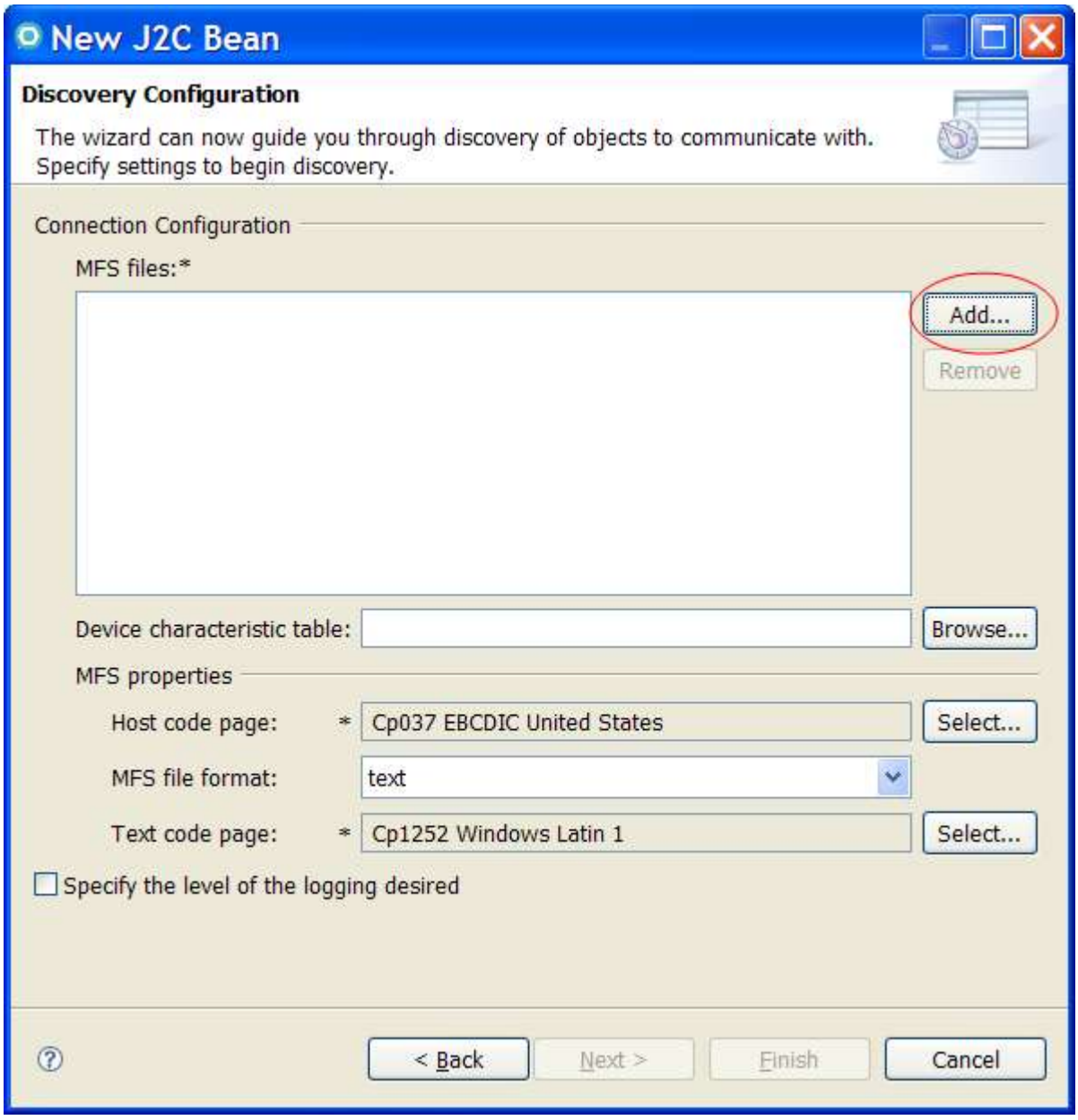

Figure 10. The Discovery Configuration page

1.9 ⏭ Click Add and navigate to the folder where the dfsivf1.mfs source file is (C:\mfs_source_file).

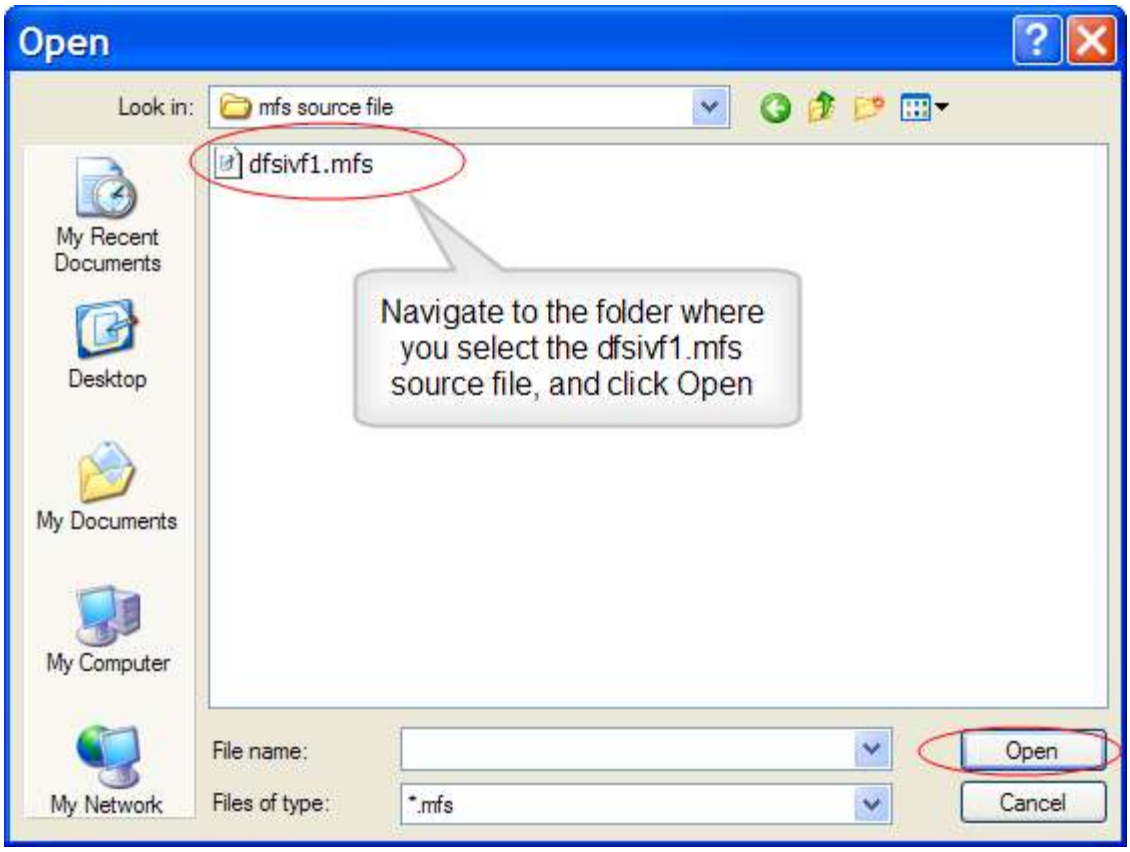

Figure 11. Finding the MFS Source file

The MFS Source file dfsivf1.mfs defines the structure of the input and output messages of the IMS IVP Phonebook application.

1.10 Select the dfsivf1.mfs source file and click Open to return to the Discovery Configuration page.

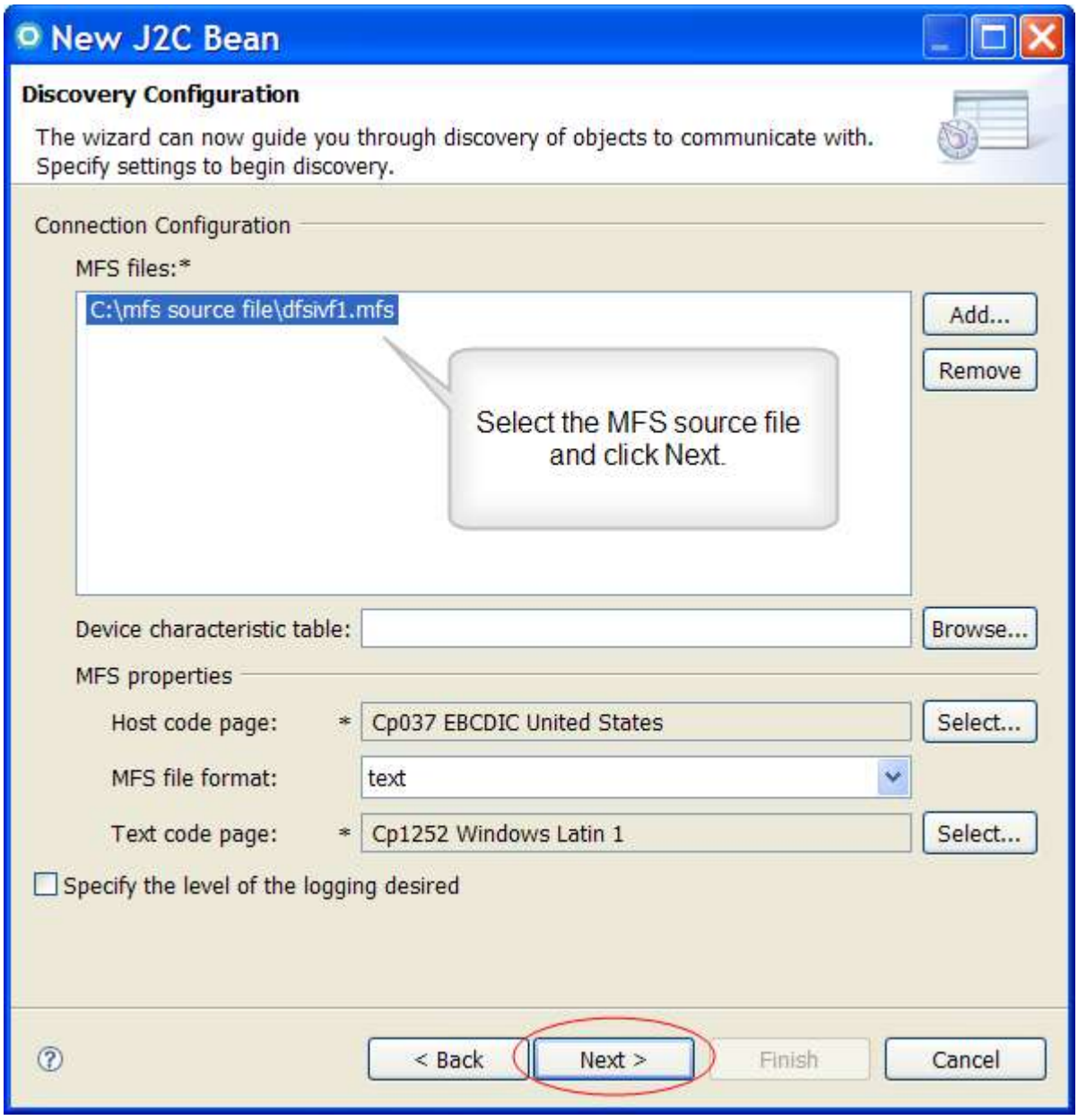

Figure 12. Click Next on the Discovery Configuration page

The dfsivf1.mfs source file is parsed through the MFS Importer and returns an operation that contains both MID and MOD names. The operation displays on the Object Discovery and Selection page.

1.11 ⏭ Accept all the defaults and click Next. The Object Discovery and Selection page opens.

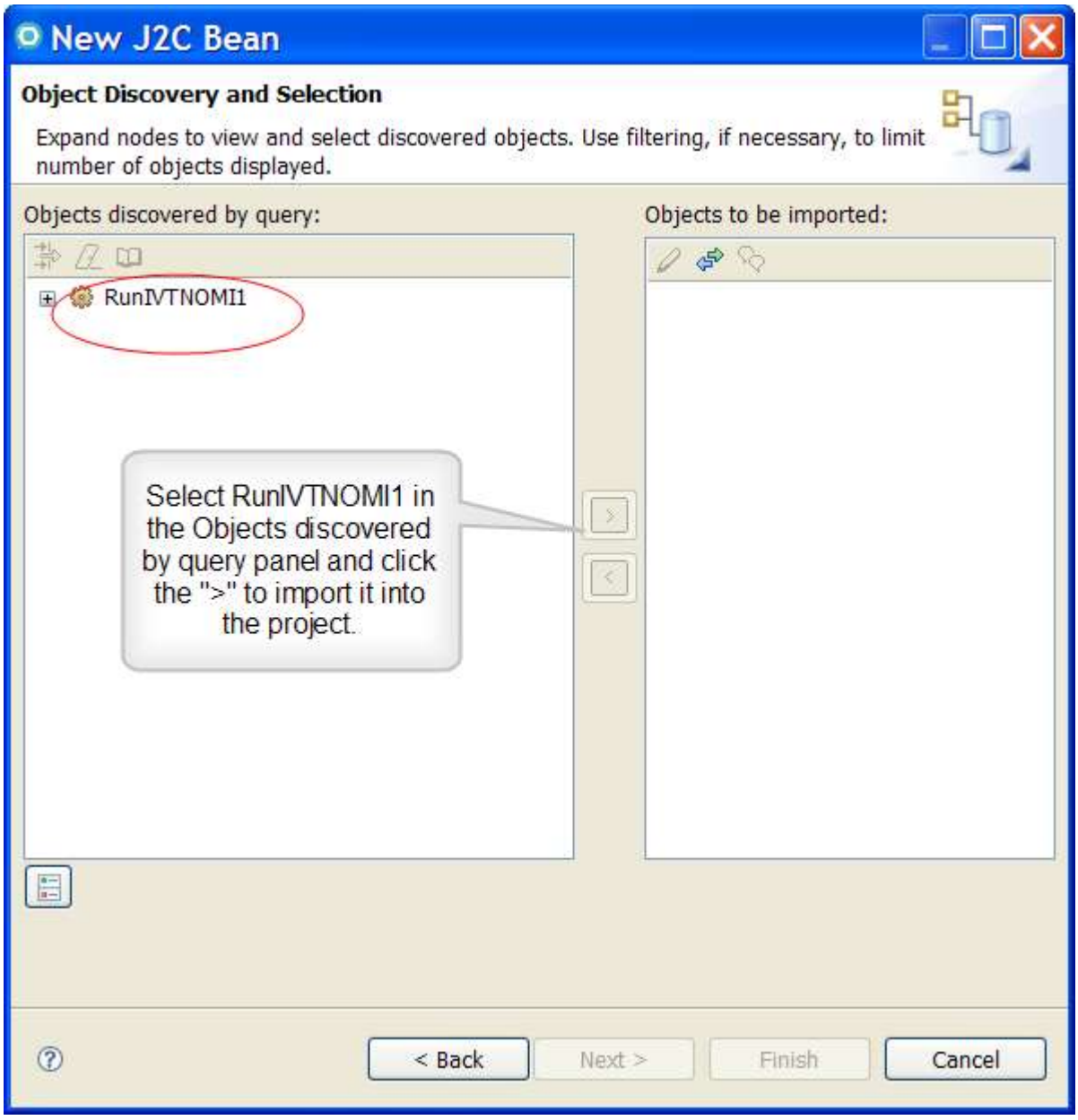

Figure 13. The Object Discovery and Selection page

1.12 ⏭ Select RunIVTNOMI1 in the Objects discovered by query panel and click the “>” to import it into the project. The Configuration Parameters for RunIVTNOMI1 page opens.

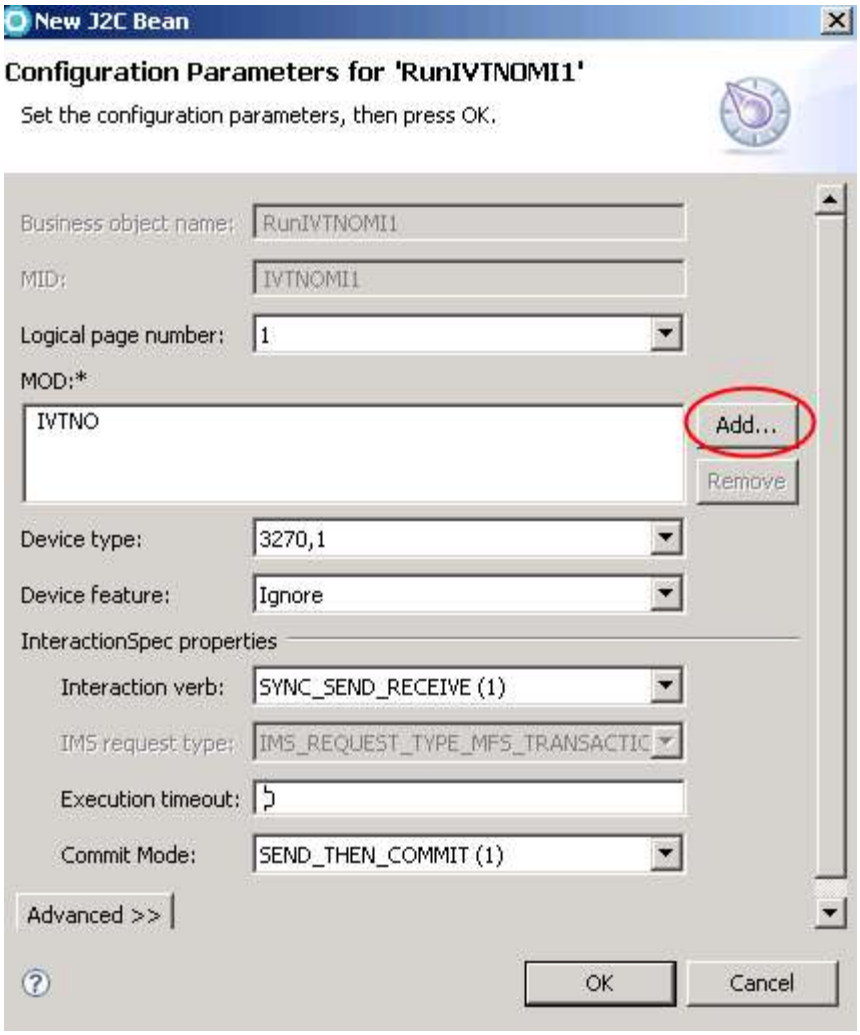

Figure 14. The Configuration Parameters for RunIVTNOMI1

1.13 ⏭ Click Add. The Add Value page opens.

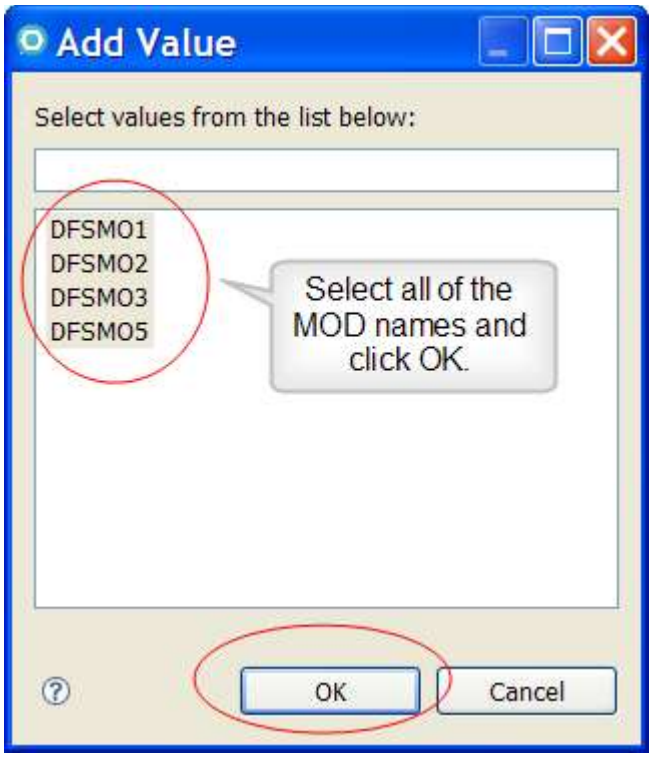

Figure 15. The Add Value page

Select the following MOD names and click OK:

- DFSMO1
- DFSMO2
- DFSMO3
- DFSMO5

Tip: You can select all of the MOD names at one time.

What are MID and MOD?
MFS input formatting occurs when a message input descriptor (MID) name is provided with an input message. The MFS message output descriptor (MOD) can supply a MID name to be used for formatting the next input message.

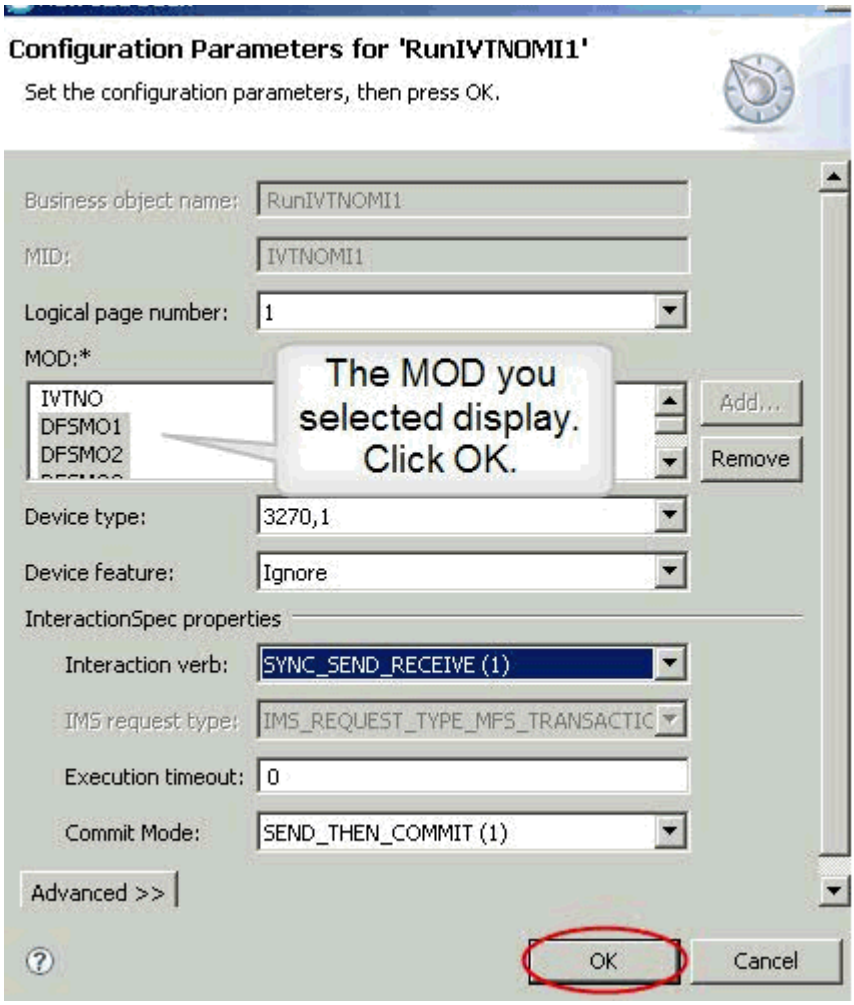

Figure 16. The MODs you selected are included in the Configuration Parameters page

1.14 ⏭ Accept the defaults on the Configuration Parameters for 'RunIVTNOMI1' page and then click OK to return to the Object Discovery and Selection page.

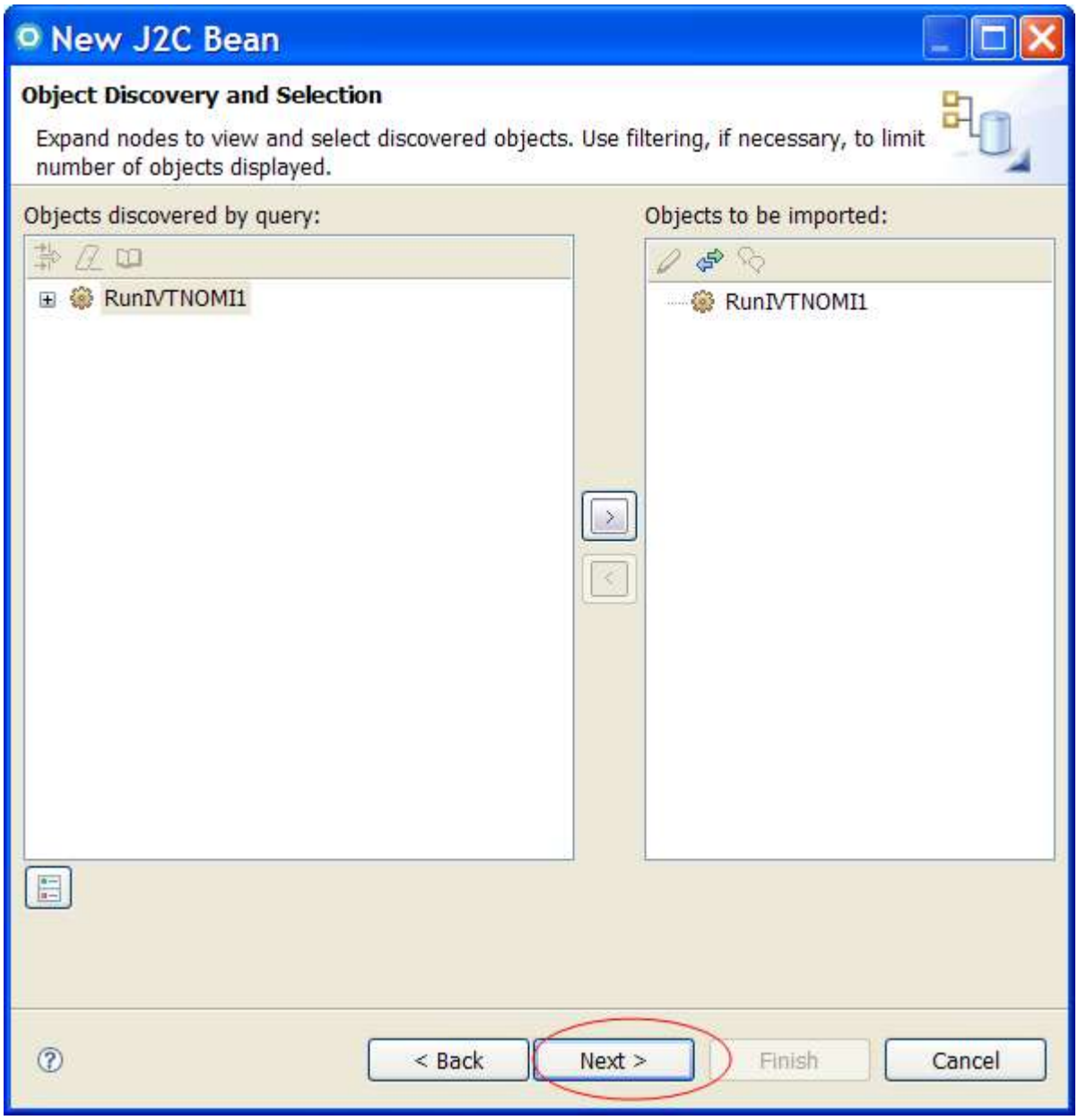

Figure 17. The Object Discovery and Selection page shows RunIVTNOMI1 added into the Objects to be imported panel.

1.15 ⏭ Click Next. The J2C Bean Creation and Deployment Configuration page opens.

Figure 18. The J2C Bean Creation and Deployment Configuration

1.16 ⏭ From the J2C Bean Creation and Deployment Configuration page, click New next to the Project name field to create a new project into which the J2C JavaBean will be generated. The New

Source Project page displays.

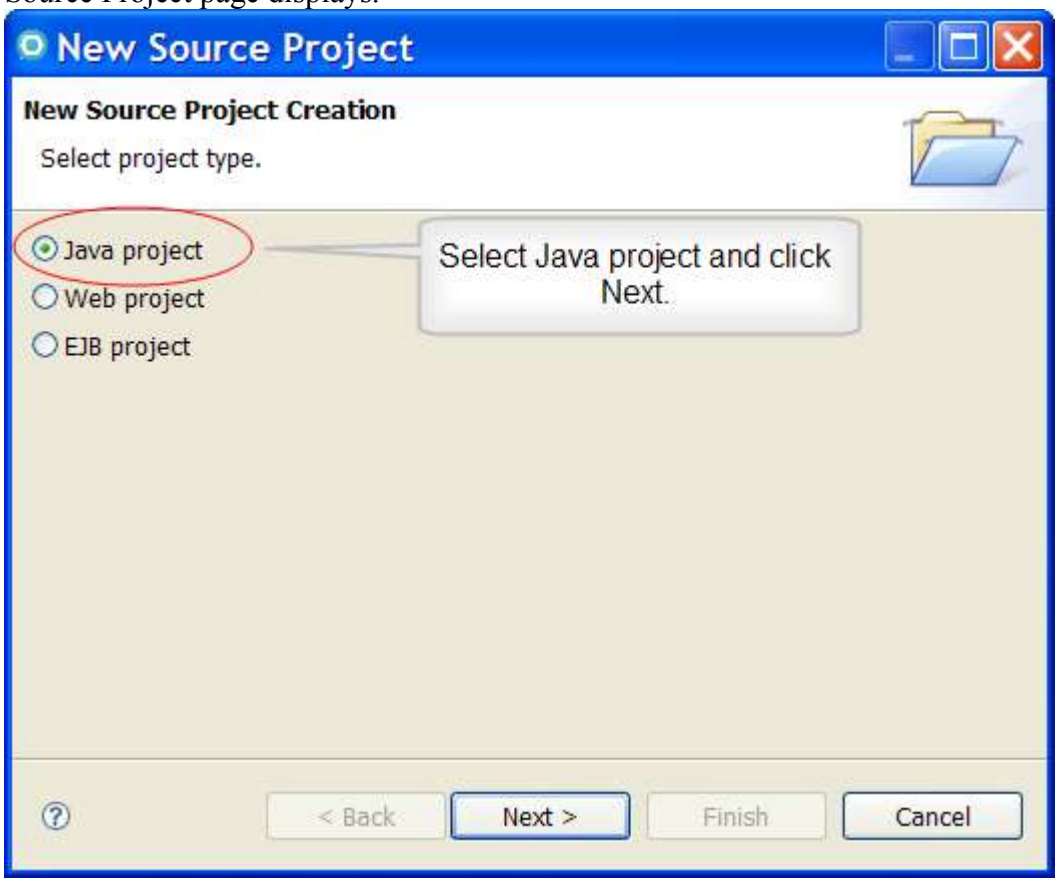

Figure 19. The New Source Project Creation page

Why Java project?
The J2C Wizard gives a choice between creating Java, Web or EJB project. We choose Java project because we will use it as the container for the J2C JavaBean source.
Java projects are not defined in the Java SE specification. They are used as the lowest unit to organize the workspace and contain all resources needed for a Java application as images, source, class and properties files.

1.17 ⏭ Select Java project and click Next. The Create a Java Project page opens.

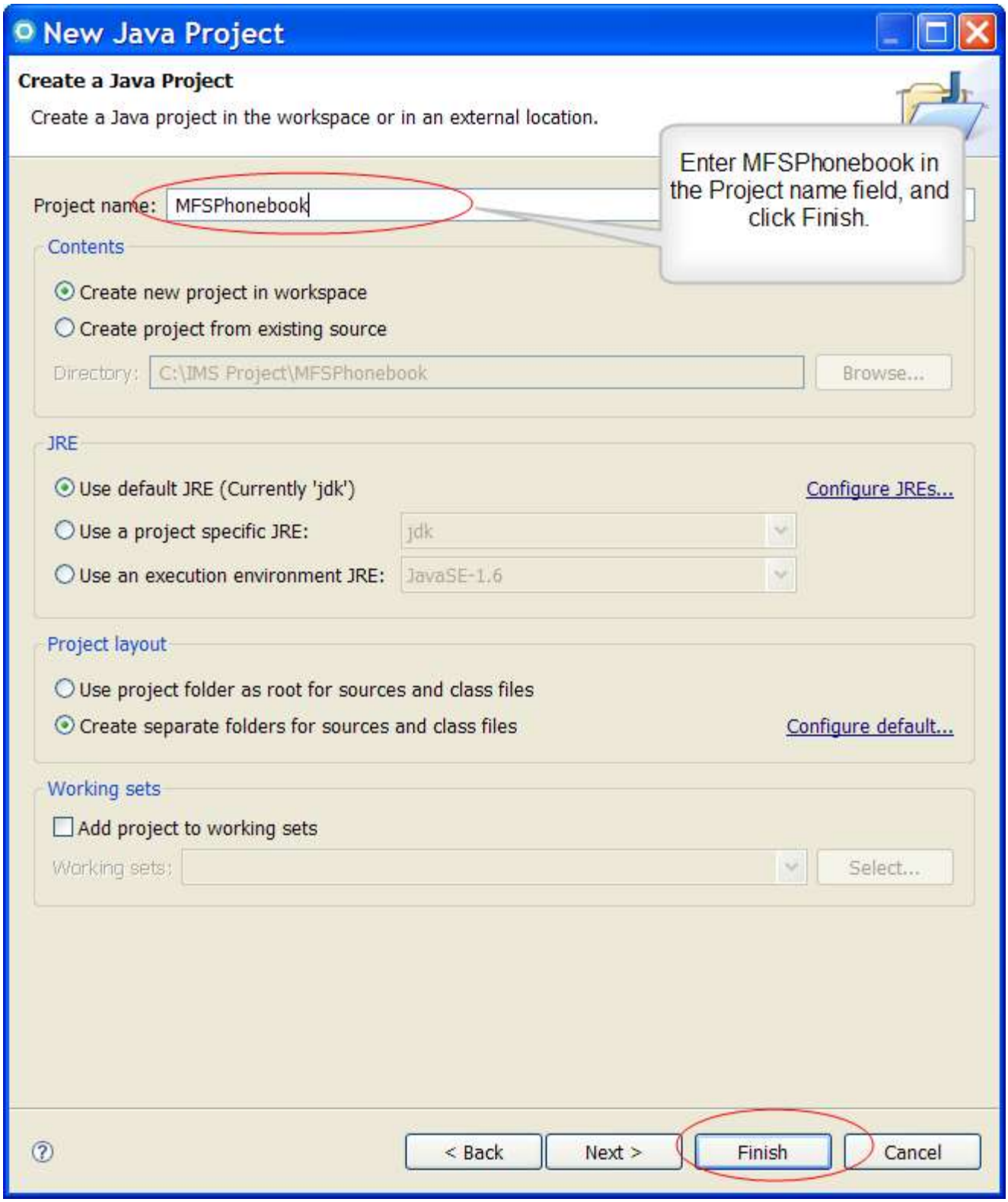

Figure 20. The Create a Java Project page

1.18 ⏭ Enter `MFSphonebook` in the Project Name field, and click Finish to return to the J2C Bean Creation and Deployment Configuration page.mt

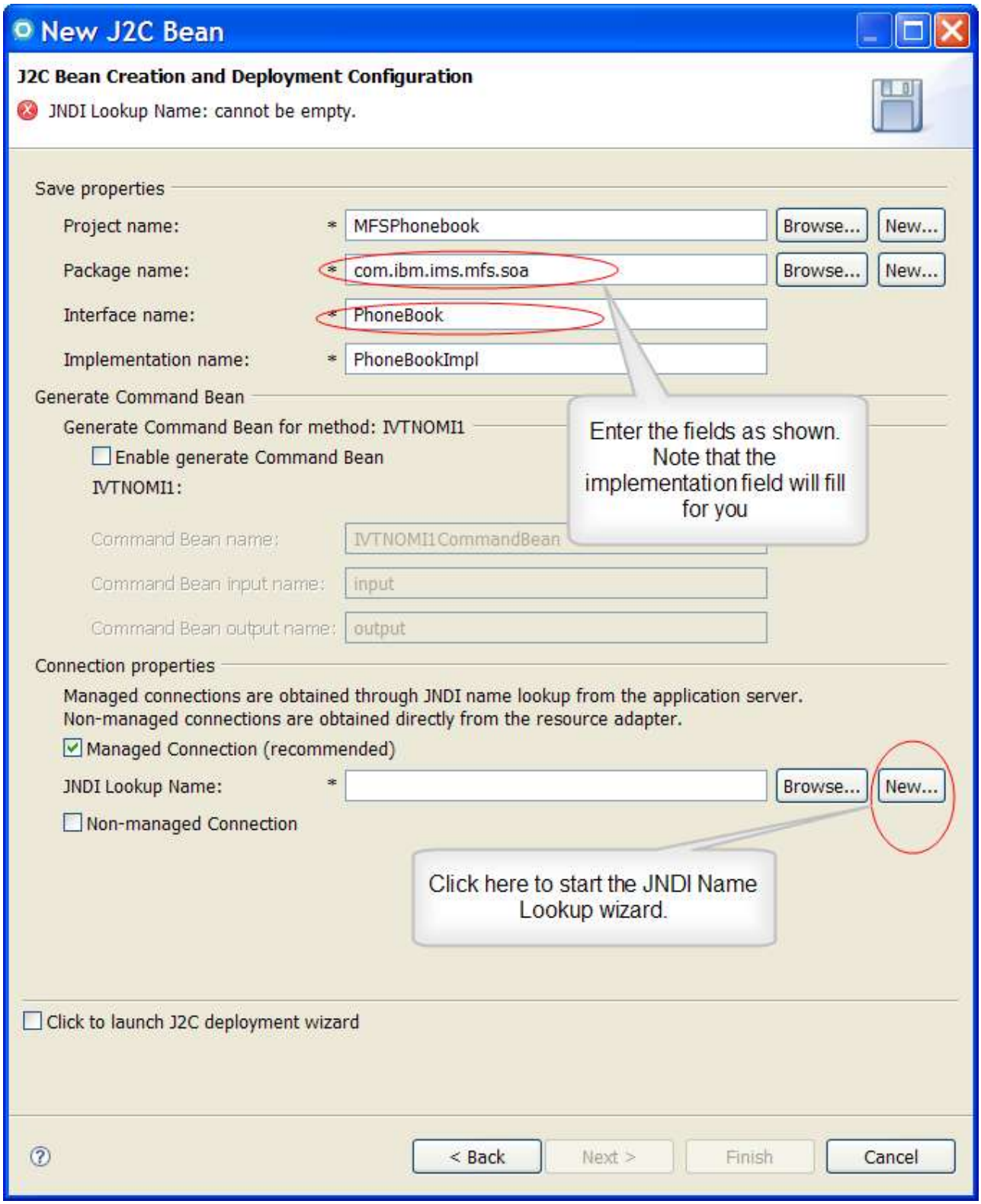

Figure 21. The J2C Bean Creation and Deployment page

1.19 ⏭ Enter com.ibm.ims.mfs.soa in the Package name field and PhoneBook in the Interface name field. The Implementation name will auto-complete for you. Click New under Connection properties. The JNDI Lookup Wizard displays

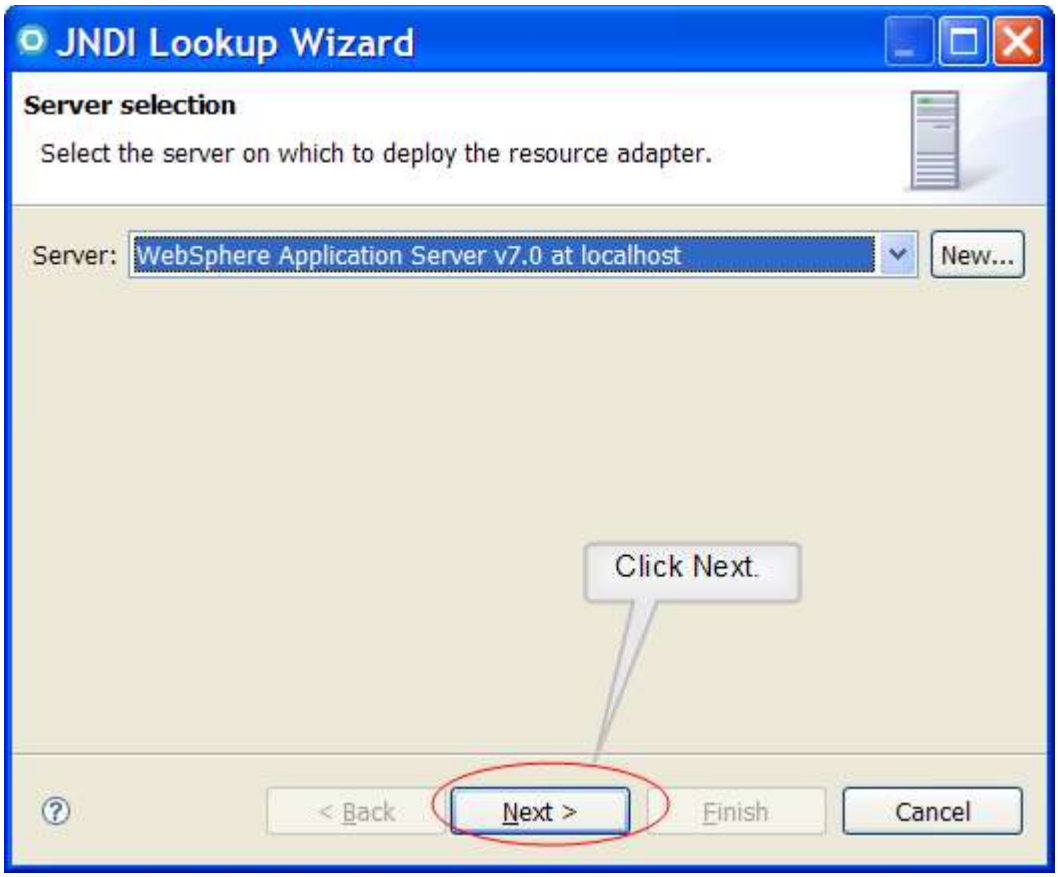

Figure 22. The Server selection page

What is an Application Server?
An application server is a component-based product that resides in the middle-tier of a server centric architecture. It provides middleware services for security and state maintenance, along with data access and persistence.

What is JNDI?
The **Java Naming and Directory Interface (JNDI)** is an API for directory service that allows clients to discover and look up data and objects via a name. In this lab we assign a unique JNDI name to our managed connection. Our J2C JavaBean will then use this JNDI name to look up the connection on the WebSphere Application Server.

1.20 ⏭ Click Next and the New J2C Connection Factory page opens.

Figure 23. The JNDI Lookup Wizard

Our application will interact with the IMS TM resource adapter through an object called a **connection factory**. IMS connection factories are used to create pre-configured connections to the **IMS transaction manager** (IMS TM).

When an application uses the IMS TM Resource Adapter, it interacts with IMS using connections between the IMS TM resource adapter and IMS Connect that are created by the IMS TM Resource Adapter. These connections can be **managed** or **non-managed**.

1.21 ⏭ Enter MyJNDIname in the JNDI Name field. Ensure that **TCP/IP** is checked (default) and enter the required connection information as indicated by the asterisk (*):

Host name: **ZSERVEROS.DFW.IBM.COM**
Port: **9999**
Data store name: **IMSC**

1.22 ⏭ Click Finish to return to the J2C Bean Creation and Deployment Configuration page.

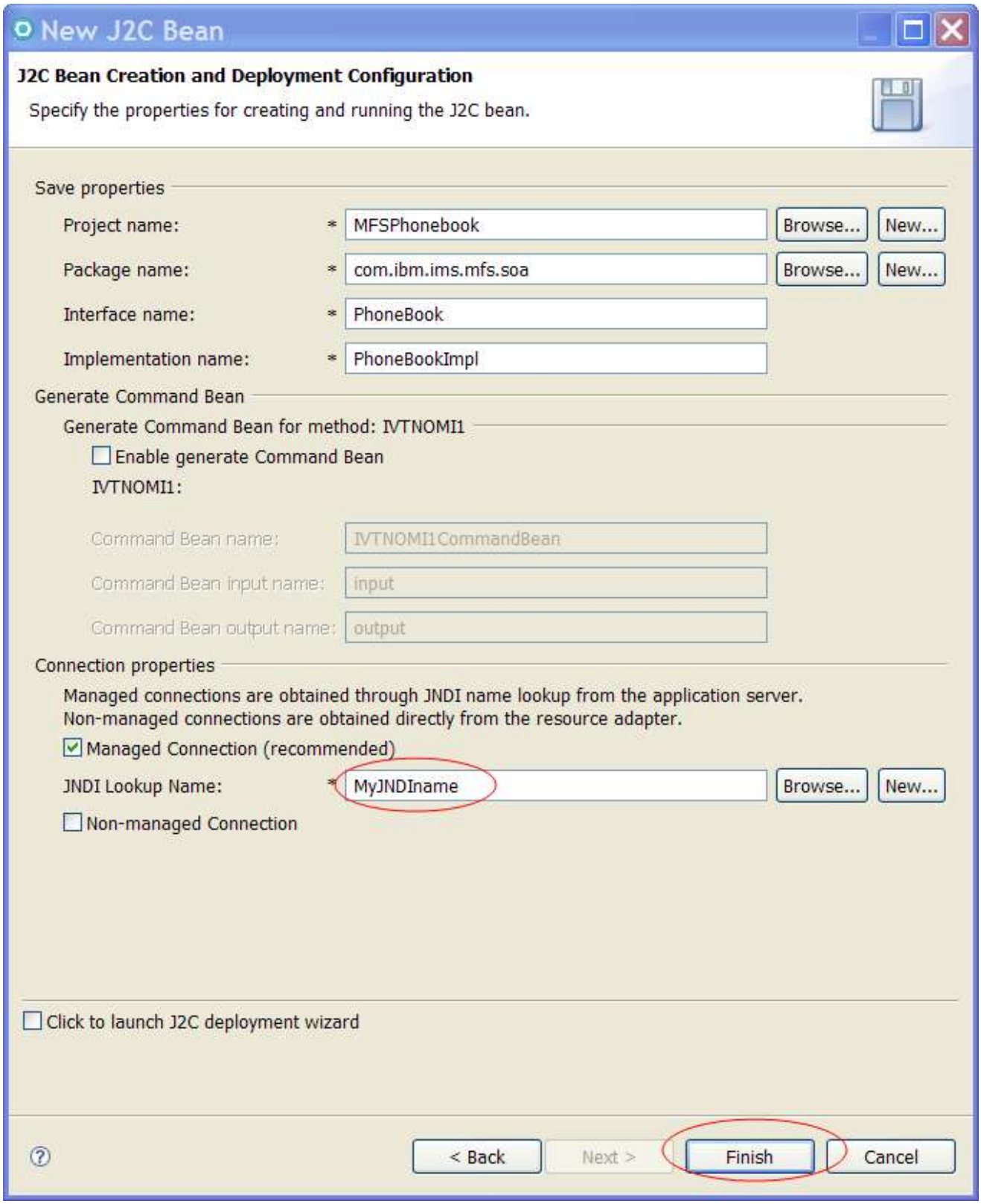

Figure 24. The J2C Bean Creation and Deployment Configuration page

1.23 ⏭ Click Finish.

Managed and Non-Managed connections

A **managed connection** runs inside a Web application server. With a managed connection, the application server provides transaction management and connection pool management, and the application server can send security information. In addition, managed connections allow connection information to be maintained by the system administrator. As connection information changes (the type of communications, the port, etc.), the system administrator can adjust the connection characteristics and no Java objects need to be regenerated.

A **non-managed connection** is designed to run where connections management supplied by an application server is not available. The characteristics of the connection must be specified and are hard-coded into the generated object. You can change the connection characteristics from your program, but you will need to generate your J2C code appropriately. Because non-managed connections are not always convenient to change, and they do not take advantage of the connection pooling, transaction management, and security management that are provided by an application server, it is easy to see why managed connections are recommended.

For this lab example, we will use a **managed connection** to IMS,

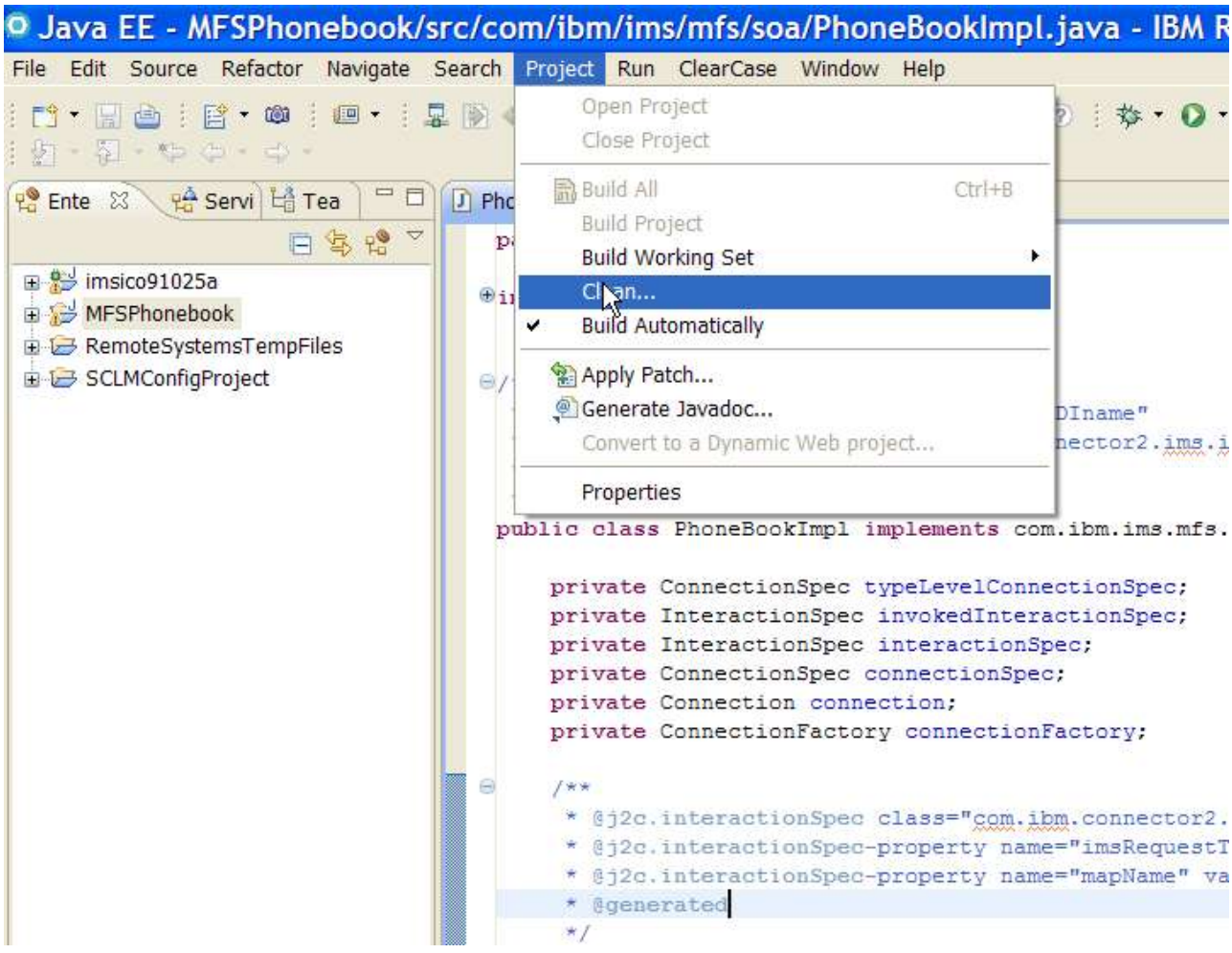

Figure 25. Select Project > Clean…

1.24 ⏭ Click Project > Clean …. The Clean pane comes up

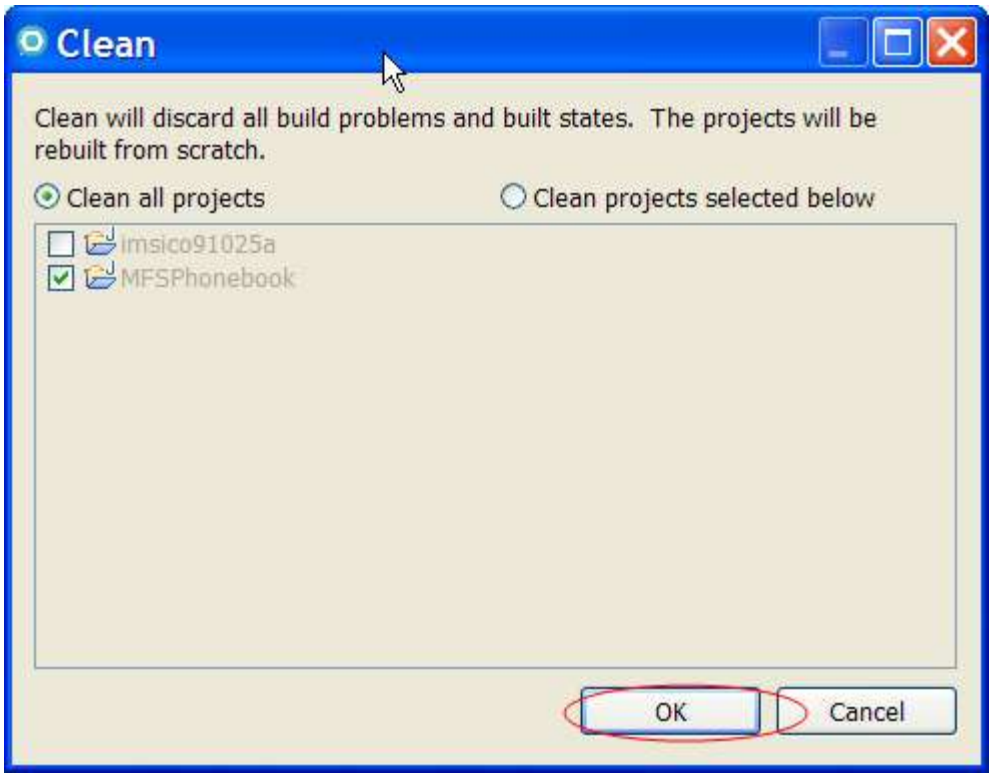

Figure 26. The Clean Pane

1.25 ⏭ Click OK.

You just completed Task 1: "Create the J2C JavaBean for the IMS MFS-based transaction"

What is next? Now that you have created the J2C JavaBean, you can proceed to generate other J2EE resources, such as JSP based on this J2C Javabean.

Number of tasks completed: 1
Number of tasks left: 2

Task 2: Create a JSP for the J2C JavaBean

KEY POINT
What are you about to do in Task 2?
For task 2 you will generate a JSP file in which you will embed the J2C JavaBean you created in Task 1. You need to embed the J2C JavaBean into a JSP because a J2C JavaBean is like a method call... it is not an application. The JSP is the actual application that displays things as a Web page.

In this task you will create a Simple JSP file (Web pages) to embed the previously generated J2C JavaBean.

2.1 ⏭ From the IBM Rational Developer for System z workspace, select File > New > Other. The Select a wizard page opens.

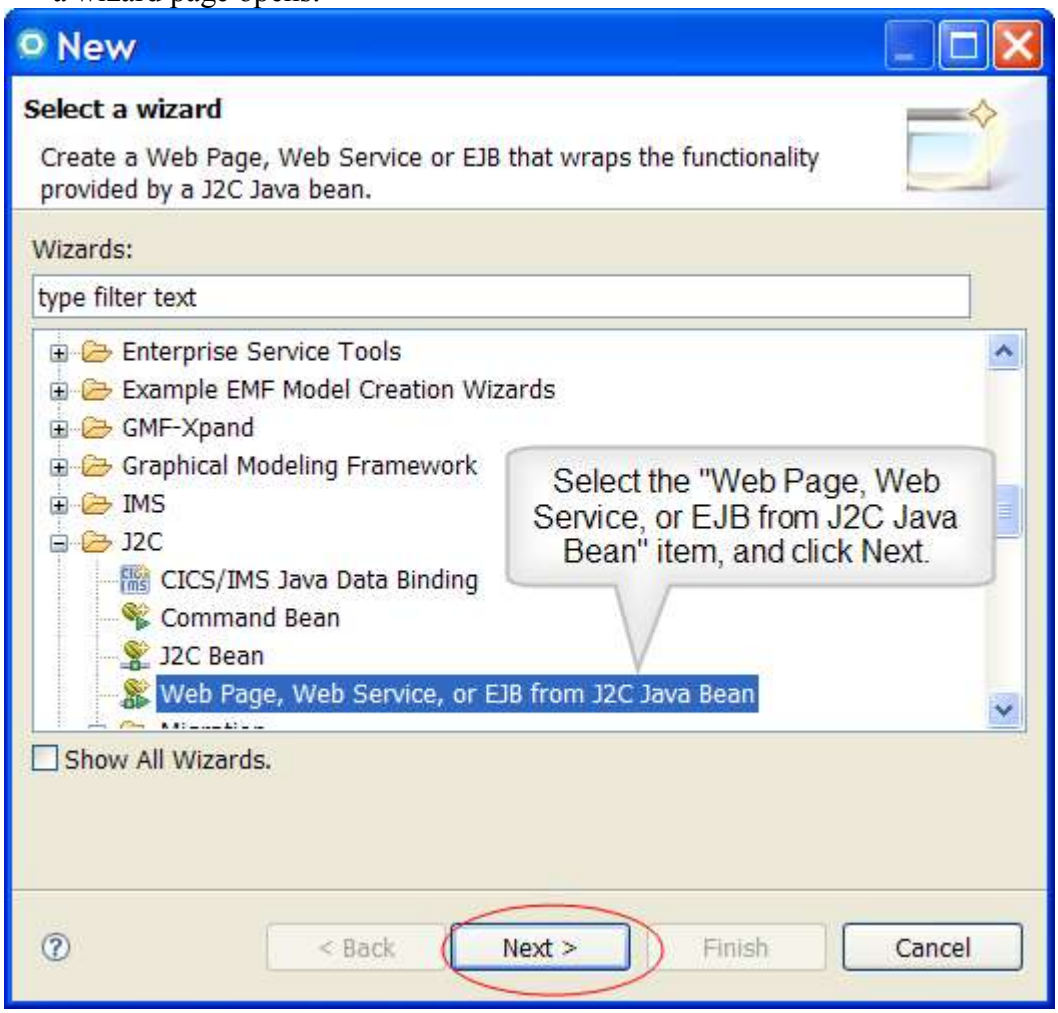

Figure 27. The Select a wizard page with the Web page, Web service, or EJB from J2C Java Bean item selected

2.2 ⏭ Expand the J2C folder and select the Web page, Web service, or EJB from J2C Java Bean item, and click Next. The Java bean selection pane opens.

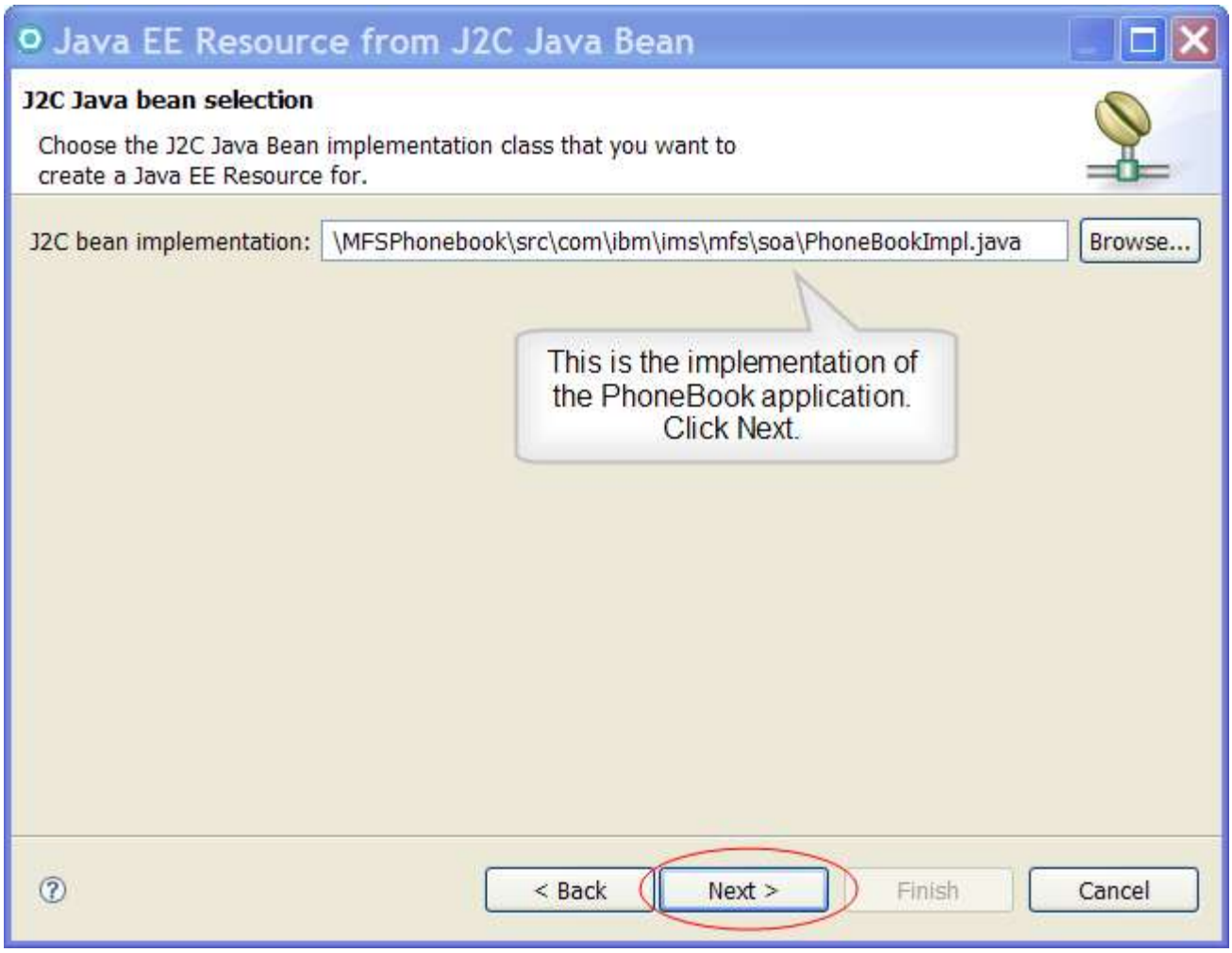

Figure 28. The implantation of the PhoneBook application is selected

2.3⏭ Click Next. The Deployment Information page opens.

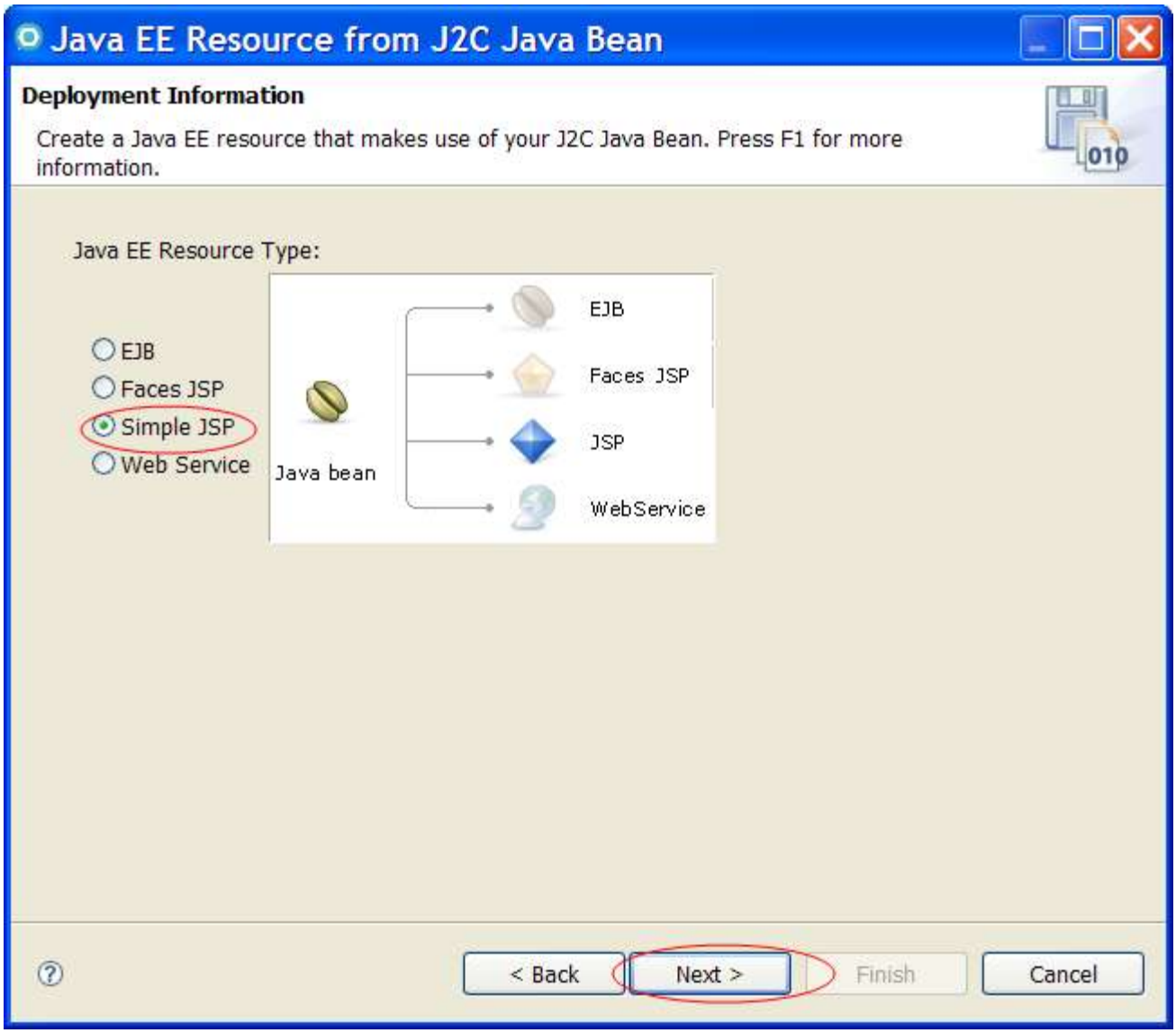

Figure 29. The Deployment Information page

JSP technology enables you to generate dynamic web content, such as HTML, DHTML, XHTML, and XML files, to include in a Web application. JSP files are one way that the product implements server-side dynamic page content. JSP files allow a Web server, such as WebSphere Application Server, to add content dynamically to your HTML pages before they are sent to a requesting browser.
EJB (Enterprise JavaBeans) and **Web Services** are other powerful architectures that can interface with our J2C JavaBean. Web Services are covered in the optional Task 4 of this lab.

2.4 ⏭ Select Simple JSP and click Next. The Simple JSP Creation page opens.

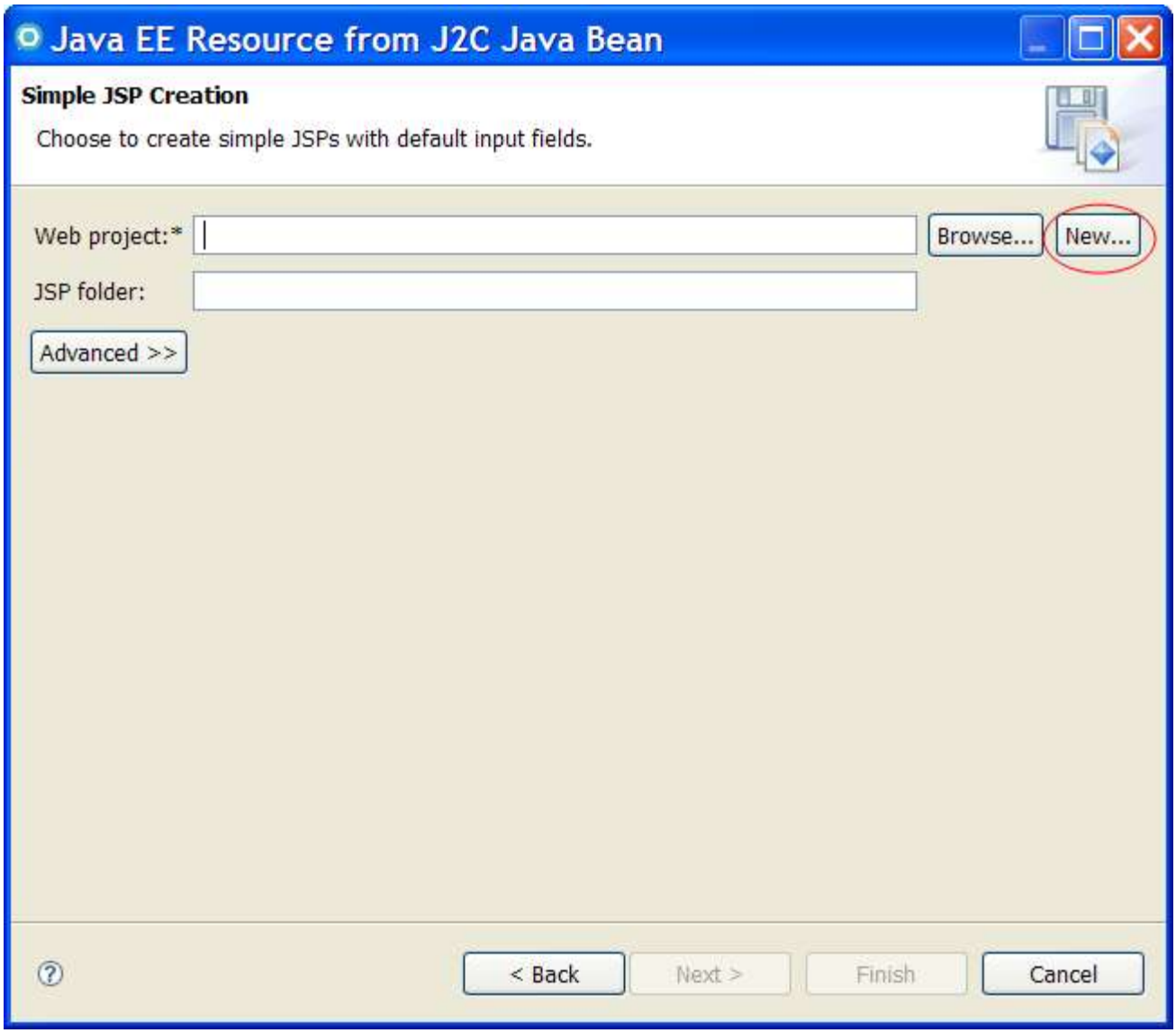

Figure 30. The Simple JSP Creation page

2.5 ⏭ Click New. The Dynamic Web Project page opens.

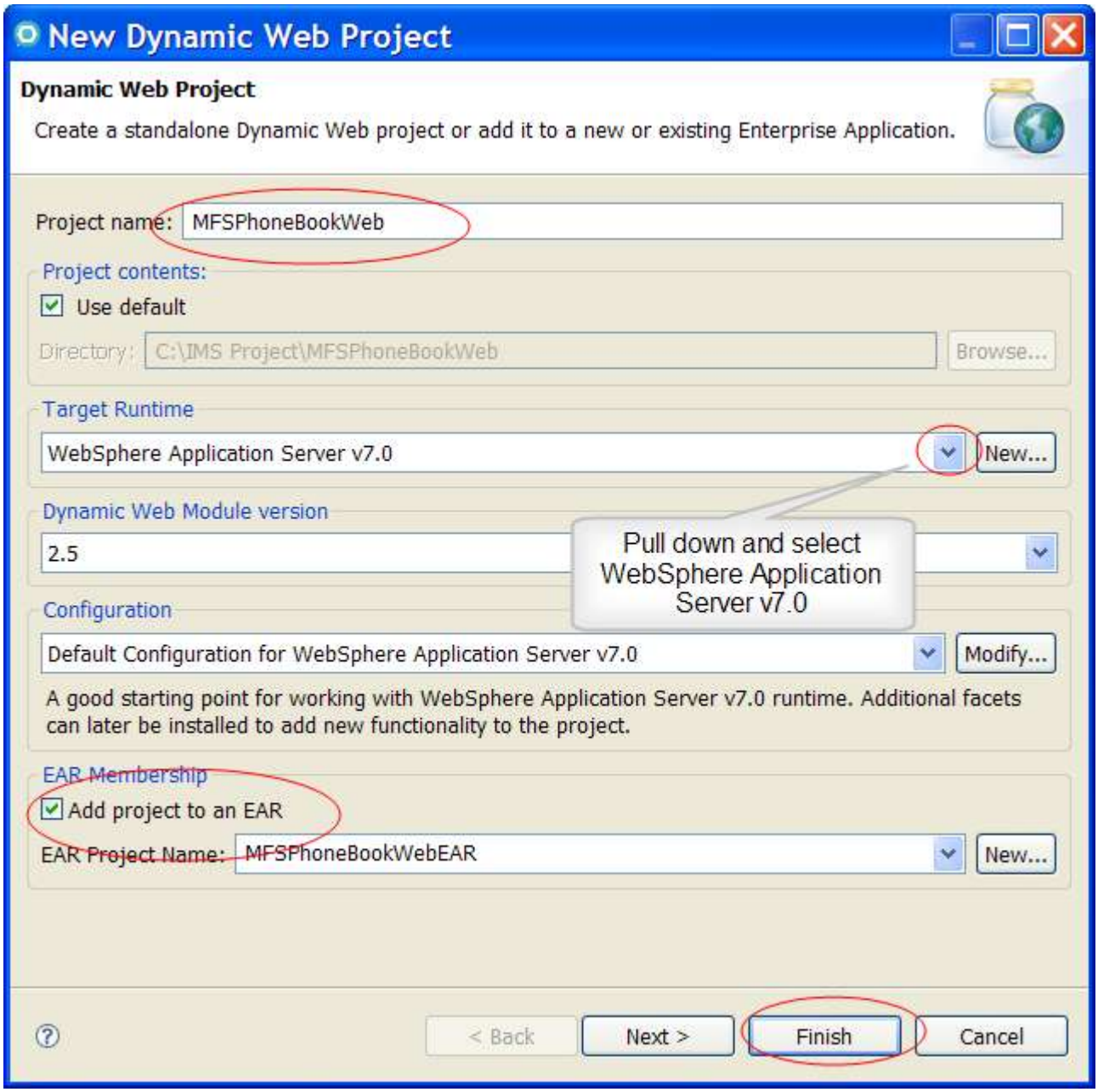

Figure 31. The Dynamic Web Project page

2.6 ⏭ Enter MFSPhoneBookWeb in the Project Name field, and from the pull-down menu, select WebSphere Application Server v7.0 for the Target Runtime. Verify that Add project to EAR box is checked and click Finish. The Simple JSP Creation page opens.

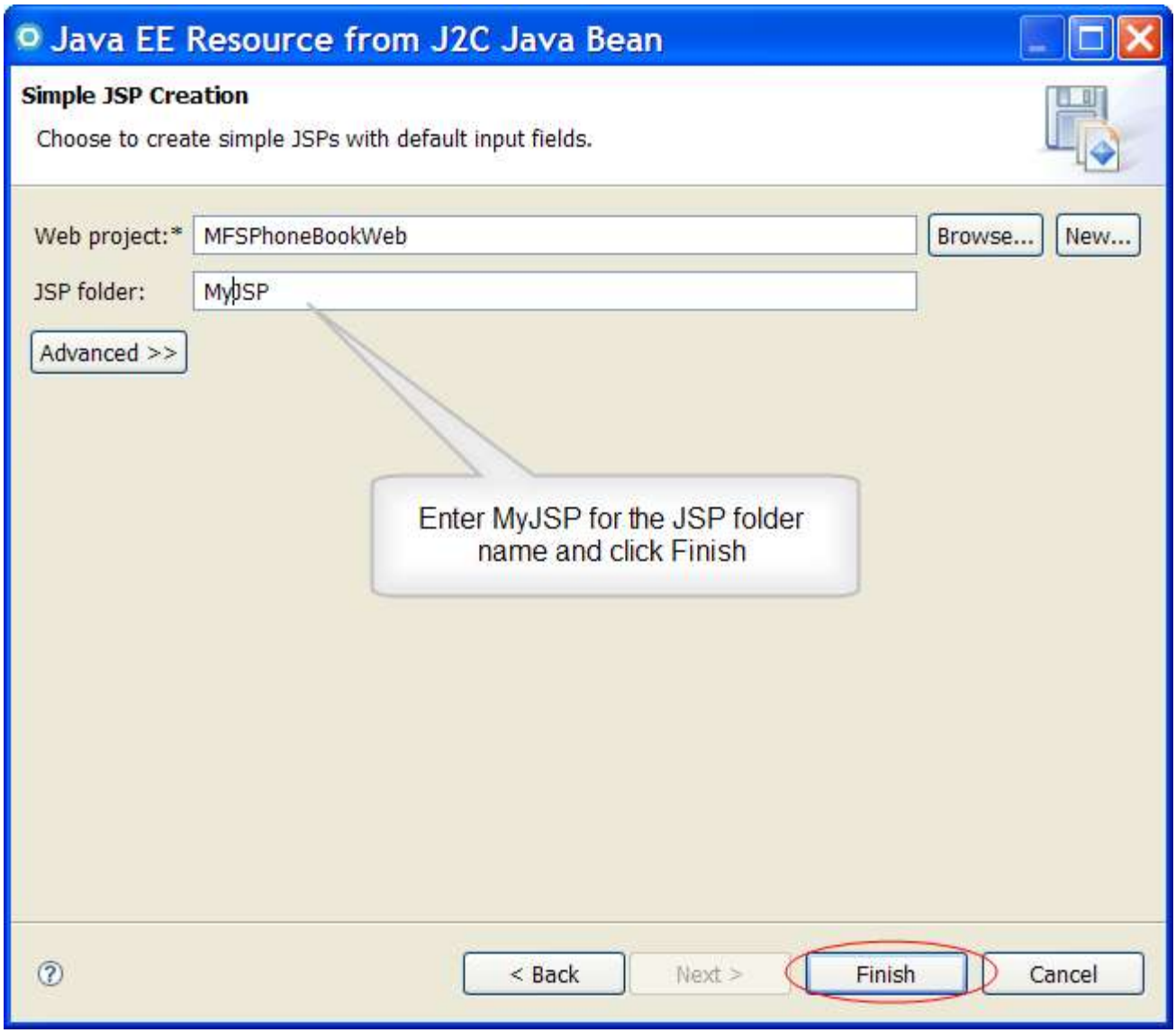

Figure 32. The Simple JSP Creation page

2.7 ⏭ Name the JSP folder MyJSP and click Finish. Notice that the MFSPhoneBookWeb web project and the PBSimpleJSPEAR items are created in the Project Explorer panel.

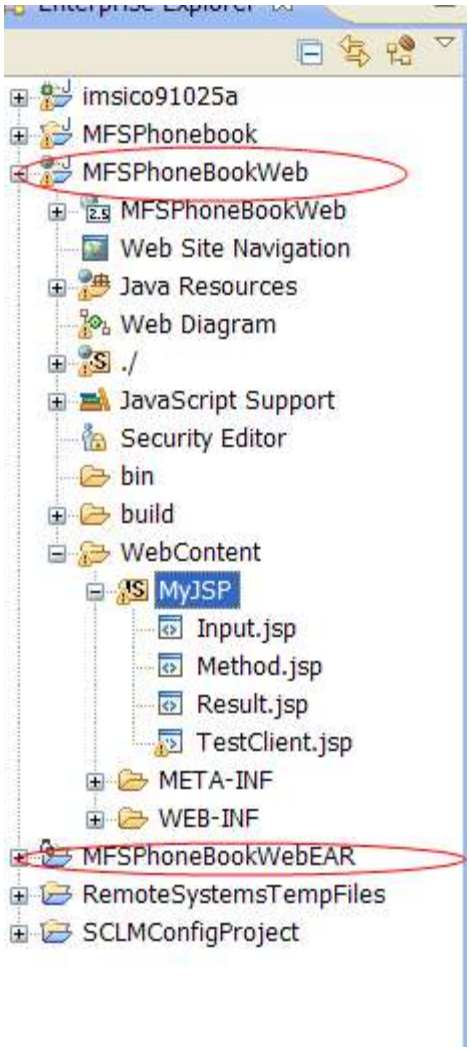

Figure 33. MFSPhoneBookWeb web project and the PBSimpleJSPEAR items are created in the Project Explorer panel

You have completed Task 2: "Create a JSP for the J2C JavaBean "

What is next? Test the J2C application using the JSP

Number of tasks completed: 2

Number of tasks Left: 1

Task 3: Test the J2C Application using the JSP on IBM WebSphere Application Server

What are you about to do in Task 3?
All that you have to do now is test. In Task 3 you will start WebSphere Application Server, add your project to the application server runtime environment, and test your application using the simple JSP client that you created.

3.1 ⏭ In the Project Explorer panel on the left, expand MFSPhoneBookWeb > WebContent > MyJSP.

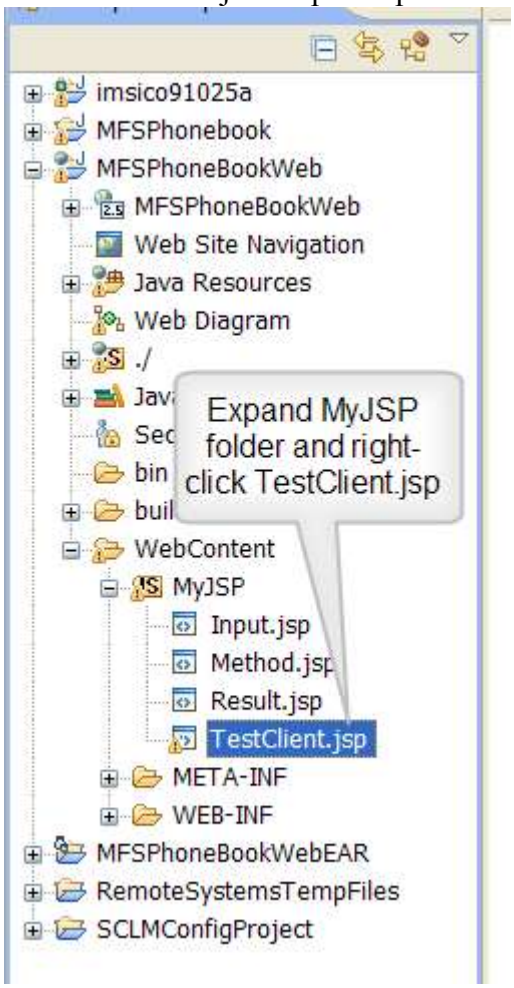

Figure 34. Expand MyJSP and right click TestClient.jsp

3.2 ⏭ Right click TestClient.jsp. Select Run As > Run on Server.

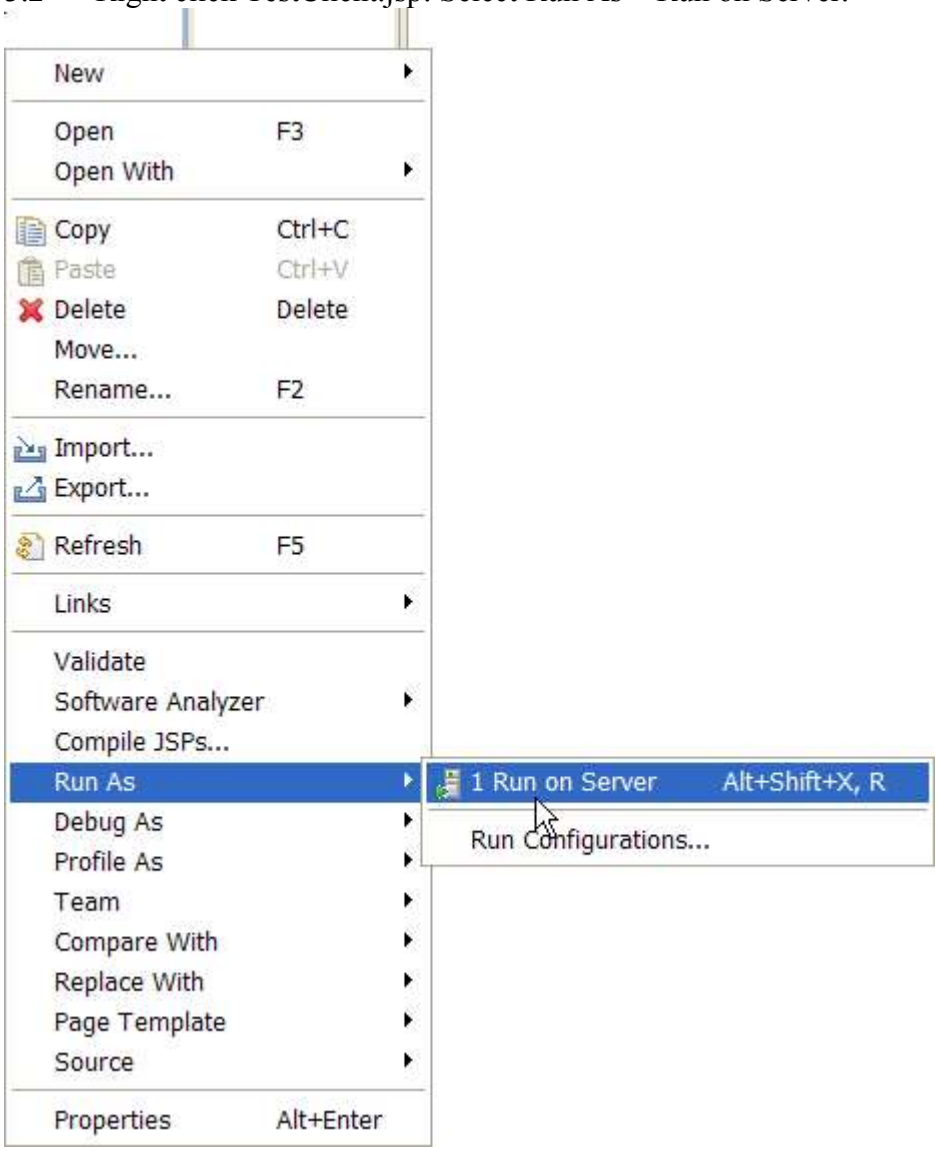

Figure 35. The menu path to the Run on Server page

The Run On Server page opens.

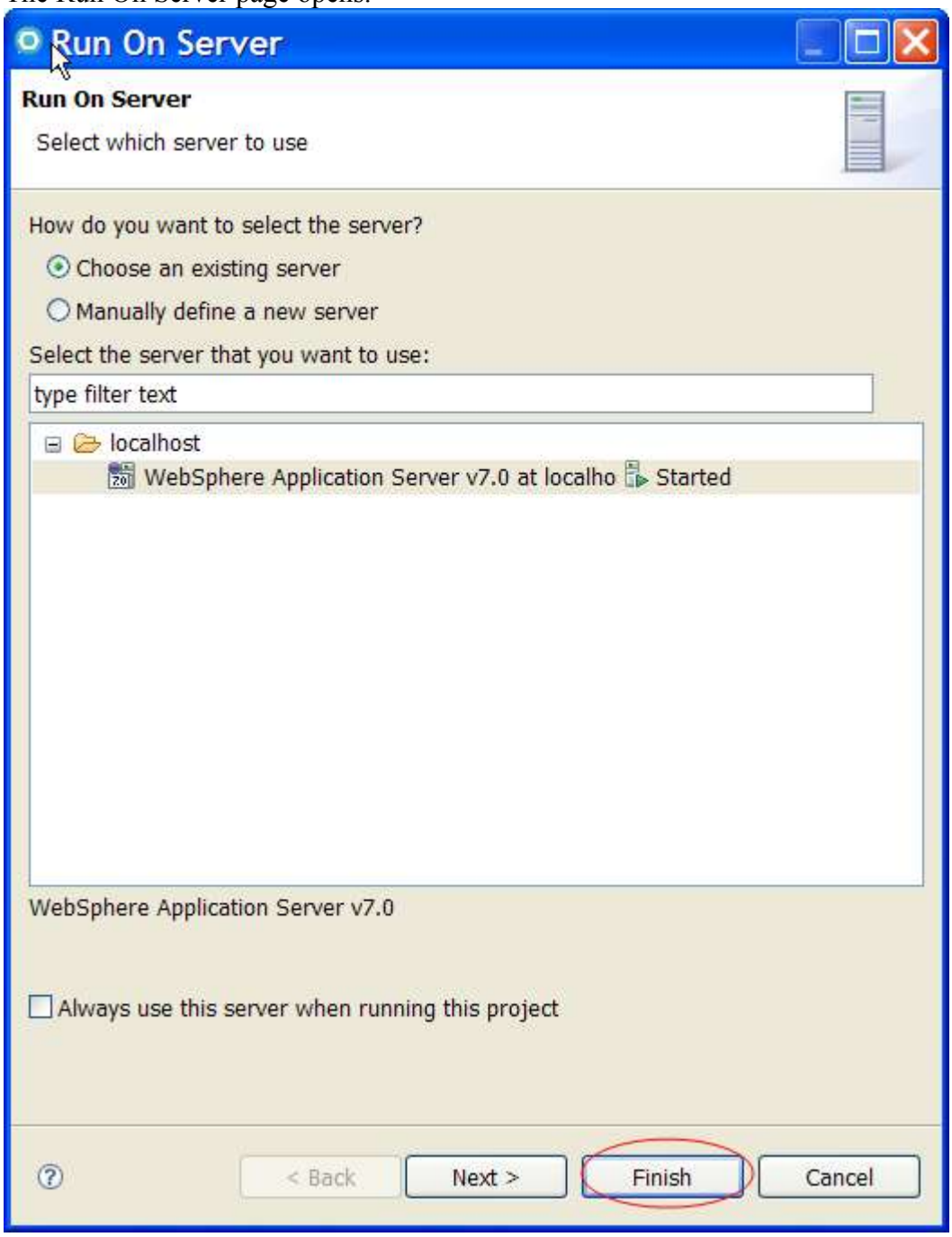

Figure 36. Run on Server Page

3.3 ⏭ Accept the defaults and click Finish. It will take a moment for the server to start. The workspace will display separate panels for Methods, Inputs, and Result. This may take a few minuets. ?

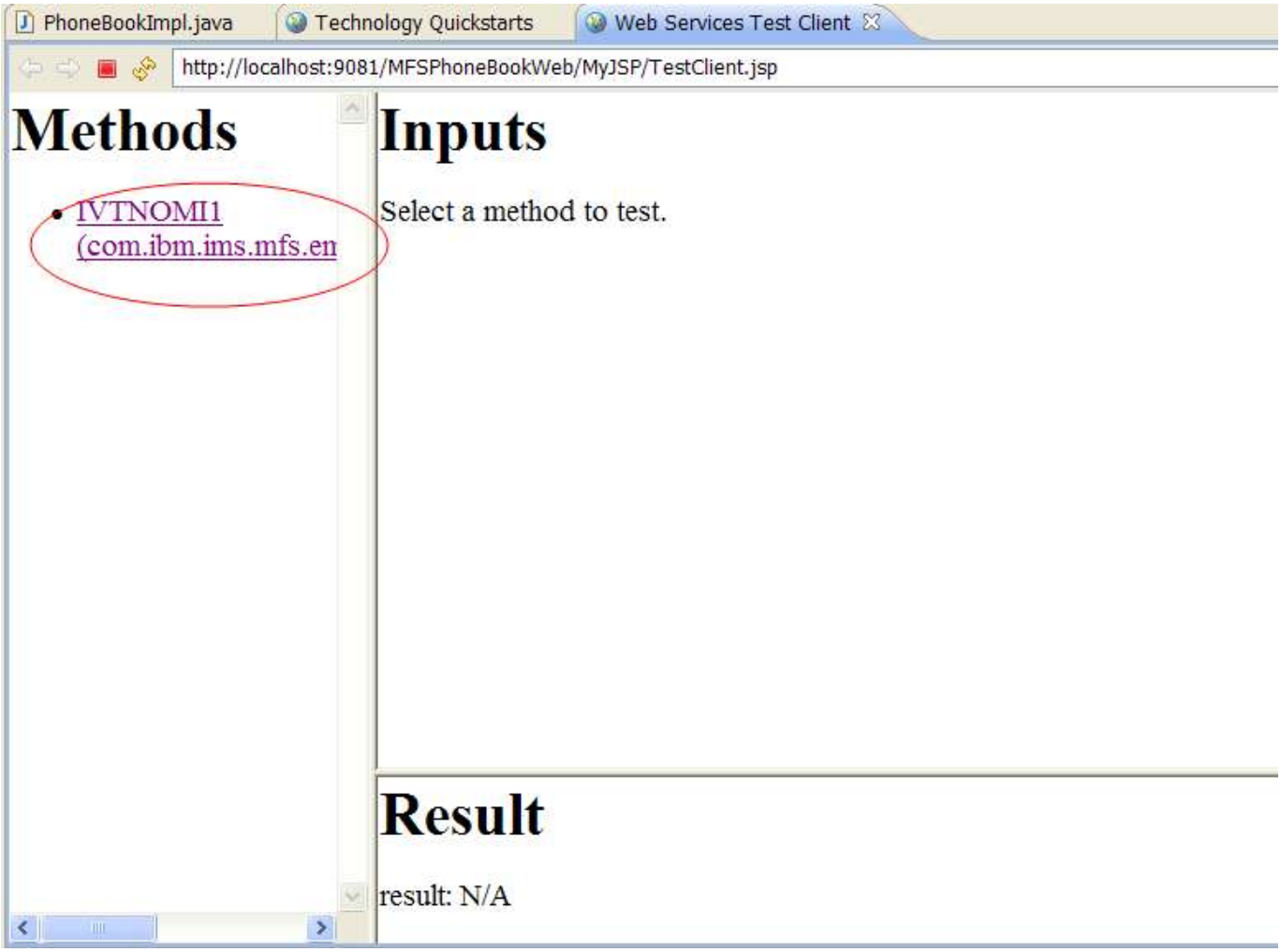

Figure 37. The workspace displays Methods, Inputs, and Result in separate panels in the workspace

3.4 ⏭ Under Methods, click IVTNOMI1.

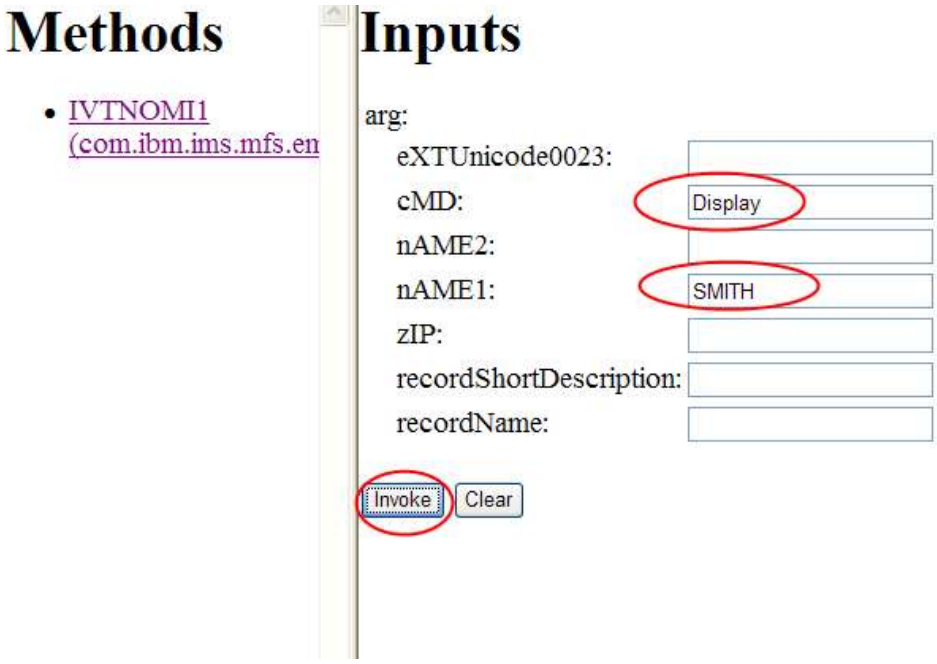

Figure 38. The PhoneBook interface opens in the Inputs panel.

3.5 ⏭ Enter `Display` in the Cmd field (command field).
3.6 ⏭ Enter `LAST1` in the Name1 field (last name field).

3.7 ⏭ Click Invoke. The results display at the bottom of the Inputs panel. Scroll down in the Results panel to see that the entry is displayed.

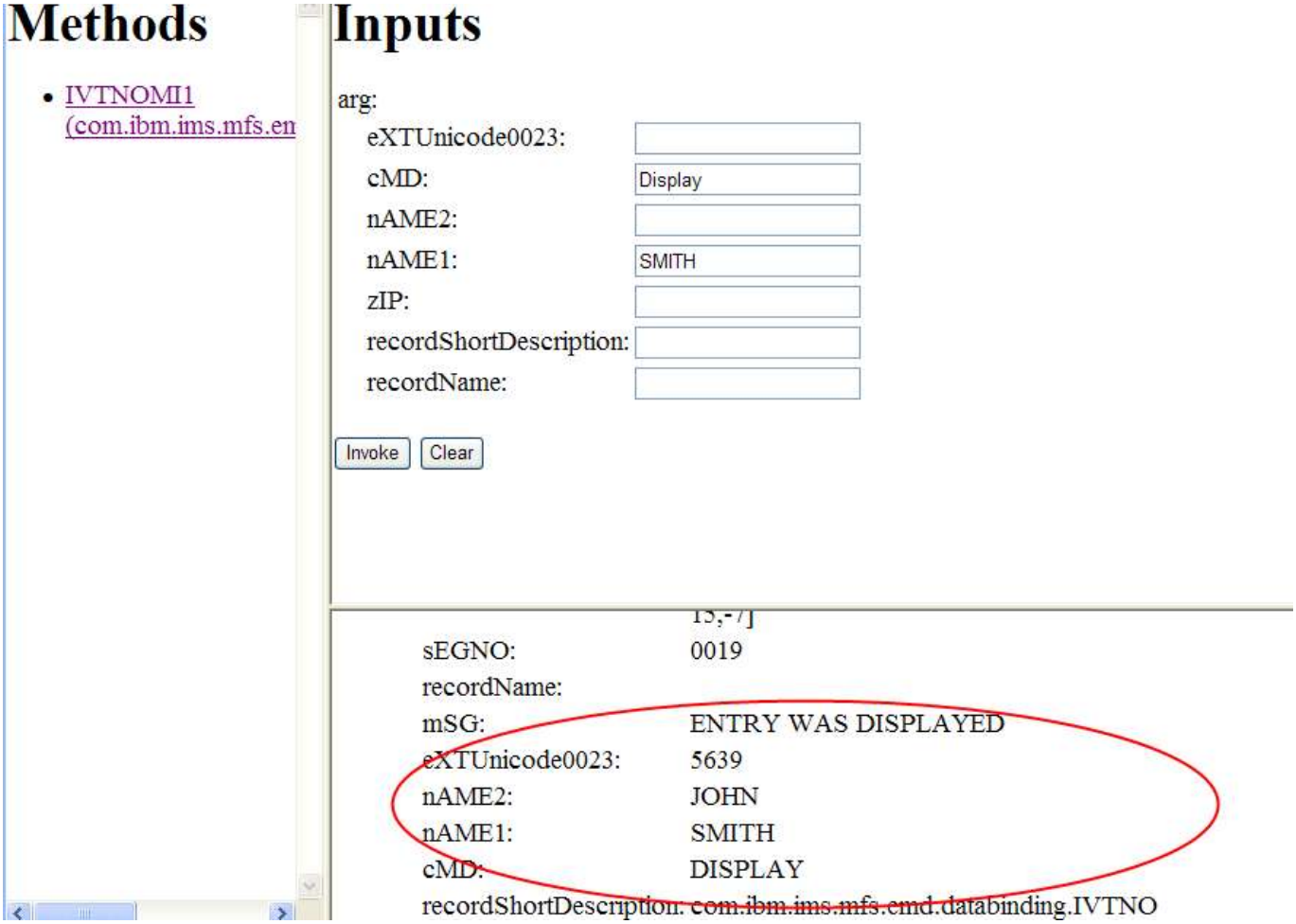

Figure 39. The results display in the Result panel

3.8 ⏭ You can click Clear to empty the fields so you can enter another name.

Congratulations! You have completed the Lab!

Resources

Learn

- If you want to learn more about MFS SOA Support there is an article in the IMS Newsletter: http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.imsnews.doc/newsletters/v803/v803.htm#article4
- More information about MFS SOA can be found at the IMS Integration Suite page: http://www-01.ibm.com/software/data/ims/toolkit/
- More information about MFS SOA can be found in the IMS Information Center: http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.etools.mfs.doc/mfs.htm

Get products and technologies

- Download trial versions of IBM Rational software.

- Download a trial version of IBM Rational Developer for system z.

Discuss

- Participate in the discussion forum.

- Check out developerWorks® blogs and get involved in the developerWorks community.

About the authors

Maria G Querales is a Software Developer at IBM
Kevin D. Kelley is an Information Developer at IBM
Shahin Mohammadi-Rashedi is an IMS SOA Demonstration Team Lead at IBM

Trademark notice