Create a J2C application for IMS Phonebook transaction using IMS TM Resource Adapter

Level: Intermediate

Evgueni Liakhovitch (evgueni@us.ibm.com) Software Developer, IBM®

Yee-Rong Lai (ylai@us.ibm.com) Information Developer, IBM

<u>Shahin Mohammadi-Rashedi</u> (<u>shahin@us.ibm.com</u>) IMS™ SOA Demonstration Team Lead, IBM

15 March 2009

Abstract

This tutorial takes you through the steps of using the J2EE Connector (J2C) component of the IBM Rational[®] Developer for System z[®] Version 7.5, and IBM IMS TM Resource Adapter Version 10 to access IMS transactions through a J2C Java[™] bean. After the J2C bean is created, it can be easily consumed by a Web application (JavaServer Pages component, or JSP), Web service, or Enterprise Java Beans (EJB) component.

About this tutorial

This tutorial will take you through the steps of using the J2EE Connector (J2C) components of the IBM Rational Developer for System *z*, version 7.5 and IBM IMS TM Resource Adapter Version 10.

Many customers have invested heavily in IMS to perform critical transactions for their businesses. The J2C tools available in Rational Developer for System z can expose the interaction specification properties of the IMS TM Resource Adapter and generate a Java class that accesses IMS transactions. Once the J2C bean is created, it can easily be consumed by a Web application (JavaServer Pages component, or JSP), Web service, or Enterprise Java Beans (EJB) component.

This tutorial will help to familiarize you with the Java Platform, Enterprise Edition (Java EE, previously known as J2EE) development environment and the J2C. In this tutorial, you will be working with the sample Phonebook IMS transaction (IVTNO), which is one of the IMS installation verification programs shipped with IMS. As an optional exercise you will also experiment with creating a Web service from the generated J2C bean class, and testing that Web service with the Web Services Explorer provided with Rational Developer for System z.

Rational Developer for System z Version 7.5 has two packages. The Rational Developer for System z with Java is the package that is a superset of IBM Rational Application Developer, Therefore, the J2EE perspective, J2C wizard, and Web tools that are offered in Rational Application Developer are also available in Rational Developer for System z and other IDEs that also include Rational Application Developer. The tutorial instructions also apply to these IDEs, with perhaps minor naming and interface differences.

Objectives

To understand and gain hands on experience extending IMS applications to Web as a part of Web pages or Web services. Software tools that are part of Rational Developer for System z and IMS TM Resource Adapter make the transformation processes easy, as the tutorial will demonstrate. The total cost of ownership for customers, therefore, is reduced.

Upon completion of this study, you will be able to perform these tasks:

- Use Rational Developer for System z and its built-in J2C tools.
- Configure IMS TM Resource Adapter connection factory
- Enable IMS applications as JSP components and Web services

System requirements for the tutorial:

- Software installed on Windows®
 - Rational Developer for System z Version 7.5
- System software installed on IBM z/OS[®]
 - IMS Version 9 or Version 10
 - IMS Connect Version 9 or Version 10 with XML Adapter configured
 - o OTMA
 - o 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.

A tutorial checklist is provided for this exercise.

Table 1. Implementation checklist

	Your environment	This tutorial:
COBOL copybook	This can be obtained from IMS application programmers.	C:\IMS PhoneBook\IMSPHBK.cpy
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.	Workspace directory: C: \Workspaces7.5\SANDBOX Project name: J2CPhoneBook

Overview of development tasks

To complete this tutorial you will perform the following tasks:

- 1. Install and configure the IMS TM Resource Adapter. Import the IMS TM Resource Adapter from the file system.
- 2. Use the J2C Java Bean wizard to create a bean that executes a transaction in IMS. Use the J2C Java Bean wizard to set up the J2EE project, Java interface, and implementations. Create sample JSP client for testing.
- 3. Deploy and test the sample application. Deploy and test the sample application within the IBM WebSphere Application Server.
- 4. (Optional) Generate Web Service implementation. Use Rational Developer for System z to generate a Web service implementation from the sample application and test it by using the Web services explorer.





Figure 1. Using Rational Developer for System z to accomplish lab objectives

Task 1. Install the IMS TM Resource Adapter

In this section, you will validate that the WebSphere Application Server runtime environment is available, and then import the IMS TM Resource Adapter.

Using Rational Developer for System z and the J2EE Projects perspective

Start Rational Developer for System z if it is not already started.

1. Select Start > Programs > IBM Software Development Platform > IBM Rational Developer for System z > IBM Rational Developer for System z

Switch to the J2EE perspective

Switch from the default z/OS Projects perspective to the J2EE perspective. Within Eclipse, there are several ways to change perspectives.

1. From the **Window** pull down, select **Open Perspective > Other**, as shown in the following Figure.

Figure 2. Opening a perspective in Rational Developer for System z

z/OS Projects - Welcome to z/OS Projects -	IBM Rational Developer fo	r System z
File Edit Navigate Search Project Data Run	Window Help	
	New Window New Editor	<u>1</u> 2 - 18 - +
Z/US Projects 25	Open Perspective Show View	∜≄ Debug ∰: z/OS Projects
	Customize Perspective Save Perspective As	Other

What is a perspective?

A perspective defines the initial set and layout of views in the Workbench window. Within the window, each perspective shares the same set of editors. Each perspective provides a set of functionality aimed at accomplishing a specific type of task, or that works with specific types of resources. For example, the Java perspective combines views that you would commonly use while editing Java source files, while the Debug perspective contains the views that you would use while debugging Java programs.

2. Scroll down and select **J2EE** from the Open Perspective dialog box, as shown in the following Figure:



Figure 3. Choosing the J2EE perspective

What is Java EE?

The Java Platform, Enterprise Edition (previously known as Java Platform, Enterprise Edition, or JEE) 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, JSP components, 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.

Java EE 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 serverside business logic for enterprise applications. Web and EJB component containers host services that support Web and EJB modules.

3. Press **OK** to switch to the J2EE Perspective.

Import the IMS TM Resource Adapter

- 1. Click **File** > **Import** to open the Import dialog box.
- 2. Enter RAR, select **RAR file**, and then click **Next** to continue, as shown in the following Figure.

Import		
Select Import an exte	rnal Connector RAR file into a Connector projec	* 🖄
Select an impo RAR	rt source;	
🖃 🗁 Java	EE KAR file 	

a A Importing a Connector DAD fil _..

3. From the Import dialog, click **Browse** to locate and import the IMS TM Resource Adapter file.

onnector Impo	ort					
mport a Connector	r module from the f	ile system			(~
onnector file:	C:\Program Files	IBM\SDP70\Reso	urceAdapters\i	ms15\imsico1(021.r. 💌 🛛	Browse
onnector module:	imsico1021				•	
arget runtime:	WebSphere Appli	cation Server v7.()		•	New
EAR Membership - Add project to	an EAR					
EAR Project Name	EAR				~	New
	14				2 < 10	
			ř – –	3	6	

. . . - -___

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 contains all the information necessary for installing, configuring, and running a resource adapter. Resource adapters comply with the J2EE Connector Architecture specification. In this lab, you are using IMS TM Resource Adapter to connect to IMS.

4. Navigate to the file C:\Program Files\IBM\SDP70\ResourceAdapters\ims15\imsico1021.rar and click **Open**, as shown in the following Figure. .

pen					
Look in:	🗀 ims15		•	3 🕫 📂 🖽-	
My Recent Documents Desktop My Computer My Network Places	imsico1021.r	a.rar			
	File name:	imsico1021.rar			Open

Figure 6. Importing the IMS TM Resource Adapter RAR file

- 5. For Target runtime, choose WebSphere Application Server 7.0, as shown in the following Figure.
- 6. Click Finish.

The IMS TM Resource Adapter module should now be visible in your J2EE perspective, as shown in the following Figure.

Figure 7. IMS TM Resource Adapter Version 10.2.1 module in J2EE project explorer

🧿 Java EE - IBM Rational Developer for Syste	em z
File Edit Navigate Search Project Run Clev	arCase Window Help
➡ +	
Project Explorer 🛛 🖓 🖓	
 HostConnectProjectFiles 	

Task 2. Create a bean that communicates with IMS

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

Using the J2C Java Bean wizard

1. Start the J2C Java Bean wizard by clicking **File > New > Other** to open the Select a Wizard dialog, as shown in the following Figure.

J2EE - Welcome to	z/OS Projects - II	BM Rational Deve	loper for Syste
File Edit Navigate Se	arch Project Data	a Run Window I	Help
New	Alt+Shift+N 🕨	Project	
Open File		Project	-
Close	Ctrl+W	Polder	- IOC Designation
Close All	Ctrl+Shift+W	File	2/US Projects
Save	Ctrl+S	= 9 Evample	2
Save As		Control Control	
Save All	Ctrl+Shift+S	📑 Other 💦	pme to

2. Enter j2c, select **J2C Java Bean**, as shown in the following Figure, and click **Next:**

Figure 9. Selecting the J2C Java Bean wizard

🔄 New		
Select a wizard Create a J2C bean that communicates with an Er Connector Architecture.	nterprise Information System through the J2	EE 📩
Wizards:		
ji2c		E _R
 J2C J2C Bean Web Page, Web Service, or EJB from Migration J2C Service Migration 	m J2⊂ Java Bean	
Show All Wizards.		
0	< Back Next >	iish Cancel

3. Select the appropriate resource adapter for the J2C Java Bean. Expand **1.5**, expand **IMS TM Resource Adapter (IBM: 10.2.1**), and select **imsico1021**, as shown in the following Figure.

Figure 10. Selecting IMS TM Resource Adapter Version 10.2.1

🔮 New J2L Bean	
Resource Adapter Selection Choose a resource adapter that will communicate to th	e EIS (Enterprise Information System).
Choose the resource adapter:	
CICS IMS TM IMS Connector for Java (IBM : 9.1.0.1.5b) IMS Connector for Java (IBM : 9.1.0.2.5a) IMS TM Resource Adapter (IBM : 10.2.1) Imsico1021 DDBC DDBC DDBC PeopleSoft SAP DEdwards EnterpriseOne Siebel	
	More details about J2C bean
0	<back next=""> Finish Cancel</back>

4. Click **Next** to continue. The Scenario Selection page opens.

Rational Developer for System z supports J2EE Connector Architecture Version 1.0 and Version 1.5. IMS TM Resource Adapter Version 10 is based on the newer JCA 1.5 standards, and is therefore located under the 1.5 section.

5. In the Scenario Selection page, select IMS COBOL, PL/I or C-based applications and click Next.

gure 11. S New J2C B	electing the type of IMS ean	applications th	at the new J2C Ja	va bean will access
cenario S	election			
Select the ty	pe of IMS applications that the n	ew J2C Java bean w	vill access.	
C mt				
(• IM5	COBOL, PL/I or C-based app	olications		17
Ger	erate a J2C Java bean that acc	esses COBOL, PL/I a	nd C based IMS applica	tions
	MFS-based applications			
Bro IMS	wse and import MFS metadata fr application	om MFS sources to g	generate a J2C Java be	an for your MFS-based
?		< Back	Next >	Finish Cancel
		12	2	

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 it 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, and so on), 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 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.

The example in this tutorial uses a managed connection to IMS.

After you have selected the scenario, you must provide the connection properties information. These properties will be stored in a connection factory.

Create a J2C application for IMS Phonebook transaction using IMS TM Resource Adapter © Copyright IBM Corporation 1994, 2009. All rights reserved. 13 of 54 This application will interact with the IMS TM Resource Adapter through an object called the **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**.

6. Select **Managed Connection** from the Connection Properties page and click **New**, as shown in the following Figure.

Figure	12.	New	12C Java	a Bean	dialog
Iguie	12.	14644	JZCJAV	a Dean	ulaiog

💿 New J2C Java Bean		×
Connection Properties		1
Managed cannections are obtained through obtained arectly from the resource adapter	n JNDI from the application server. Non-managed connectio (two-tiered application).	ons are
Managed Connection (recommen	ded)	
JNDI Name:" MyDefaultJNDIName		Browse New
Non-managed Connection		
Note: If both connection types are sel non-managed connection.	ected, a managed connection is attempted first, followed by	à
0	<back next=""> Fa</back>	nish Cancel

7. Make sure that **WebSphere Application Server v7.0** is selected and click **Next**, as shown in the following Figure.

Figure 13. Selecting the server on which to deploy the IMS TM Resource Adapter

💽 JNDI Lookup	Wizard	_ 🗆 🗙
Server select Select the serve	ion r on which to deploy the resource ada	apter.
Server: WebSp	here Application Server v7.0 at localh	ost 💌 New
0	< Back Next >	Finish Cancel

8. The New J2C Connection Factory dialog displays. Enter a new JNDI Name (for example, PhoneBookJNDIName).

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 tutorial you will assign a unique JNDI name to your managed connection. Our J2C bean will then use this JNDI name to look up the connection on the WebSphere Application Server.

9. Make sure that **TCP/IP** is selected (the default) and enter the required connection information (indicated by the asterisk [*]), as shown in the following Figure:

- a. Host name: zserveros.dfw.ibm.com
- b. **Port:** 9999
- C. Data store name: IMSC

Figure 14. Specifying connection information

💿 JNDI Lookup Wiz	rard	×
New J2C Connection Create a new J2C Connection first page of the J2C wize	on Factory ection Factory for the resource adapter selected at the ard.	NIN
Resource Adapter:	IMS TM Resource Adapter (IBM : 18.1.1)	
JNDI Name:	PhoneBookJNDIName	
Connection class name:	com.ibm.connector2.ims.ico.IMSManagedConnectionFactory	7
Host name:	* ZSERVEROS.DFW.IBM.COM	1000
Port Number:	* 9999	
Local option		
IMS Connect nam	e.	1
Data store name:	* IMSC	
Advanced >>		
0	Kexto Finish Cancel	

10. Click Finish.

11. On the Connection Properties screen, click **Next** to continue.

12. On the J2C Java Bean Output Properties page, enter J2CPhoneBook as the **Project Name** and click **New** to define the project properties, as shown in the following Figure.

Figure	15.	Specify	ing 12C	Java	bean	output	properties
inguic	T O .	Speciny	1119 520	Juvu	DCull	output	properties

💿 New J2C Java Bean	E
J2C Java Bean Output Properties	
OProject Name: The project does not exist in the workspace.	
Project Name: * J2CPhoneBook	Browse New
Package Name	Browse New
Interface Name	
Implementation Name:	

13 On the New Source Project Creation page, select **Web project** and click **Next** to continue, as shown in the following Figure.

Figure 16. Selecting Web project as the project type

💿 New Sou	rce Project			×
New Sourc Select project	e Project Creatio lype.	n		
C Java proj	ect			
Web proj	ect «			
C EJB proje	ect			
	N			
۲	< Bapk	Next>	Finish	Cancel

Why Web project?

The J2C wizard gives you a choice between creating a Java, Web, or EJB project. Choose Web project because you will be creating a Web interface for your J2C bean in the form of a simple JSP component and as a Web Service (see optional Task 4).

Web projects hold all of the Web resources that you create and use to develop your Web applications.

- 14. On the New Dynamic Web Project page shown in the following Figure.
 - a. In Project contents, leave Use default checked.
 - b. For the **Target Runtime**, make sure **WebSphere Application Server v7.0 is selected** as the server.
 - c. Leave the Dynamic Web Module version and Configurations settings as is.
 - d. Select the **Add Project to an EAR** check box. Allow the wizard to add "EAR" to your EAR project name. Web projects and EAR projects must have different names.
 - e. Click **Finish** to create the Dynamic Web Project.

Figure 17. Specifying the target runtime and EAR membership for the Web project

Project name: 320	PhoneBook			
Contents				
🔽 Use default				
Directory: C:\Wor	kspaces7.5\RDz	\J2CPhoneBook	÷	Browse
Target Runtime				
WebSphere Applica	ation Server v7,I	Dis		▼ New
Dynamic Web Modu	ule version			
2.5				<u>•</u>
Configuration				
Default Configurat	ion for WebSphe	ere Application S	erver v7.0	▼ Modify
A good starting po	int for working w	ith WebSphere	Application Serve	r v7.0 runtime.
EAR Membership				
Add project to a	an EAR			
EAD Droject Name	J2CPhoneBook	EAR.		▼ New

Rational Developer for System z will now complete the creation of the J2EE components that support the J2C bean. Notice that both a dynamic Web project and an EJB project have been added to your work space. Also, now that the supporting projects are created, the J2C Java Bean wizard returns to define the J2C bean Output Properties.

15. On the J2C Java Bean Output Properties page, leave the project name as **J2CPhoneBook** . Enter the following required fields (respect the upper/lower case) as shown in the following Figure.

- a. Package Name: sample.ims
- b. Interface Name: PhoneBook Notice that PhoneBookImpl in the Implementation Name field will be created for you once the Interface Name is supplied.

Figure 18. Specifying the package name and interface name

Create a J2C application for IMS Phonebook transaction using IMS TM Resource Adapter © Copyright IBM Corporation 1994, 2009. All rights reserved.

Project Name: * J2CPhoneBook	Browse New
Package Name: sample.ims	Browse New
nterface Name: PhoneBook	
mplementation Name:* PhoneBookImpl	
Save session as Ant script	

16. Click **Next** to continue.

17. From the Java Methods page, click the **Add** button to add a Java method to the sample.ims package defined in the previous page.

18. From the Add Java Methods page, enter **runPhoneBook** as the Java Method **Name**:

Figure 19. Adding a Java method

O New J2C Bean	
Java Methods	60
Add a Java method for each function or service you want to access from the Additional methods can be added later in the Java source editor through the	EIS. Snippets View.
Java methods:	
	Add
	Edit/V
🖸 Add Java Method	
Java Method	
Specify the name of the new method.	
Name: * runPhoneBook	
Input type:	Browse
Use the input type for output	
Output type:	Browse
	New
User defined InteractionSpec or ConnectionSpec parameters	
Method summary:	
🞯 runPhoneBook () : void	

What's Next

You have just started creating a Java method that will provide translation between COBOL data that IMS can understand and Java data types suitable for Java EE applications. Next, you will import a COBOL copybook for the IVTNO transaction. The wizard will parse the copybook and identify input and output fields. It will then generate translation code that will provide the mapping between COBOL and Java. As elsewhere in this tutorial, all the work is done by the Rational Developer for System z tooling, and no manual coding is required. Now you will create the input and output data mappings between COBOL and Java. In this step you will import the data definitions from the Ex01.cbl (COBOL) copybook.

20. Click **New** next to the **Input type** to create the input mappings for the *runPhoneBook* Java method as shown in the following Figure.

Create a J2C application for IMS Phonebook transaction using IMS TM Resource Adapter © Copyright IBM Corporation 1994, 2009. All rights reserved. 20 of 54

Figure 20. Creating a new input type

ava Method Specify the name of the	e new method.	M
Name: * runPho	neBook	
Input type:		Browse
🗖 Use the input type	for output	
Output type:		Browse
		New
User defined In	teractionSpec or ConnectionSpec parameters	
1ethod summary:		
Method summary: 	.() : void	

21. From the Data Import page, accept the default **COBOL to Java** mapping.

22 Click the **Browse** button and find the file **C:\IMS PhoneBook\IMSPHBK.cpy** as shown in the following Figure. Click **Open** to accept it:

Create a J2C application for IMS Phonebook transaction using IMS TM Resource Adapter © Copyright IBM Corporation 1994, 2009. All rights reserved. 21 of 54

Figure 21. Selecting the COBOL to Java mapping and the COBOL file

Import				×
Data Import Specify data import configuration propertie	85			BO,
Choose mapping: COBOL to Java	РНВК.сру			Browse
Ø	< Back	Next>	Finish	Cancel

23. Click Next to continue.

24. The Importer page opens, allowing you to define the appropriate COBOL Import Parameter settings. Default settings are for the Win32 platform.

- a. From the **Platform** list, select **z/OS**. The code page and other parameter settings will correctly change for the z/OS platform.
- b. You can press the **Advanced>>** button to observe the other parameter settings for COBOL. However, do not change any of these advanced settings for this tutorial.
- c. Click **Query** to select the appropriate input data from the IMSPHBK.cpy COBOL file as shown in the following Figure.

Figure 22. Specifying the COBOL import parameter settings

💿 Import	×
Importer	P
🛞 Data structures cannot be empty.	
Platform: 2/05	
Code page: IBM-037	Select
Advanced >>	×
Data structures:	<u> </u>
 	Query

The COBOL program IMSPHBK.cpy contains the following **Data structures** in the Linkage Section: **INPUT-MSG** and **OUTPUT-MSG**. Notice that Rational Developer for System z inspected the IMSPHBK.cpy COBOL program and provided the data areas to map to the input record.

d. Select the **INPUT-MSG** data structure to map this data area to the input record, as shown in the following Figure, and then click **Next** to continue:

💿 Import	×
Importer Select a communication data structure.	P.O.
Platform: z/OS Code page: IBM-037 Advanced >>	Select
Data structures:	Query
(2)	< Back Next > Finish Cancel

Figure 23. Selecting the input data structure

25. The Import page displays. Notice that you can change the generation style and data bindings. Accept the default generation style and data bindings.

26. Click **Finish** to generate the INPUTMSG Java class for the input mapping:

Figure 24. Specifying the data binding information for input messages

o mpore			
Saving Properties Specily the saving properties.			
Data Binding			
Project Name: * J2CPhoneBook			Browse New
Package Name:* sample.ims.data			Browse New
Class Name: * INPUTMSG			
☑ Overwrite existing class			
Generation Style: Default			•
		×	
	i	1	
0	< Back	Next>	Finish Cancel

The Add Java Method page redisplays. Notice that the **Input type** has been defined with the INPUTMSG Java class. *Create a J2C application for IMS Phonebook transaction using IMS TM Resource Adapter* © Copyright IBM Corporation 1994, 2009. All rights reserved. 27 Click **New** to define the output type, as shown in the following Figure.

Add Java Method		
iva Method ipecify the name of the r	new method.	M
Name: * runPhone	eBook	
Input type:		Browse
		New
📕 Use the input type fo	or output	
Output type:		Browse
Output type:		Browse
Output type:	aractionSpec or ConnectionSpec parameters	Browse
Output type:	eractionSpec or ConnectionSpec parameters	Browse
Output type:	eractionSpec or ConnectionSpec parameters	Browse
Output type: User defined Inte lethod summary: Concerning runPhoneBook ()	eractionSpec or ConnectionSpec parameters	Browse
Output type: User defined Inte	eractionSpec or ConnectionSpec parameters	Browse
Output type:	eractionSpec or ConnectionSpec parameters	→ New
Output type:	eractionSpec or ConnectionSpec parameters	Browse

Figure 25. Creating an output type

You must now create the **Output type**, repeating similar actions that you already performed for the Input type.

28. From the Data Import page, accept the default COBOL to Java mapping.

29. Click **Browse** and select the C:\IMS PhoneBook\IMSPHBK.cpy file again, as shown in the following Figure.

30. Click **Open** to accept it.

Figure 26. Selecting the COBOL to Java mapping and the COBOL file

Import		×
Data Import Specify data import configuratio	n properties	B.O.
Choose mapping: COBOL to COBOL Re: COBOL Re:	Java	Browse
0	CERC Nexts	Finite Cancel 1

31. Click **Next** to continue.

32. The Importer page opens, in which you define the appropriate COBOL Import Parameter settings, as shown in the following Figure. . Default settings are for the Win32 platform.

- a. Select **z/OS** as the platform. The code page and other parameter settings will correctly change for the z/OS platform.
- b. Click Query to select the appropriate input data from the IMSPHBK.cpy COBOL file:

Figure 27. Specifying the COBOL import parameter settings

🕘 Import	×
Importer	B
🚱 Data structures cannot be empty.	
Platform: 2/05	
Code page: IBM-037	Select
Advanced >>	
Data structures:	
 	Query

33. Select the **OUTPUT-MSG** as the data structure for the output type, as shown in the following Figure, and then click **Next** to continue:

Figure	28.	Selecting	the	output	data	structure
--------	-----	-----------	-----	--------	------	-----------

💿 Import		×
Importer Select a communication data structure.		80
Platform: z/OS	<u> </u>	-
Code page: IBM-037 Advanced >>		Select
Data structures:		
		Query
2	A Bank Newt > Finish	Cancel
U	K BACK NEXC / FINISH	Cancer

34. Accept the default generation style and data bindings. Click **Finish** to generate the OUTPUTMSG Java class, as shown in the following Figure.

Figure 29. Specifying the data binding information for output messages

Caulus Desperties			
Saving Properties			0
Specify the saving properties.			
Data Binding			
Project Name: * J2CPhone	Book	Browse	New
Package Name:* sample,ims.data		Browse	New
Class Name: * OUTPUTM	MSG		
Overwrite existing class			
Generation Style: Default		*	
	×		
		<u>></u>	
(?)	< Back Next >	Finish	Cancel

Create a J2C application for IMS Phonebook transaction using IMS TM Resource Adapter © Copyright IBM Corporation 1994, 2009. All rights reserved. 26 of 54 35. The Add Java Method page should now be defined with an Input type: INPUTMSG and an Output type: OUTPUTMSG, as shown in the following Figure.

idd Java M	lethod					
ecify the nar	d me of the new meth	nod.				M
ame: *	runPhoneBook					
nput type:	INPUTMSG					Browse
						New
Use the in	put type for output					
utput type:	OUTPUTMSG					Browse
						New
沟 🛛 User de	efined InteractionSp	pec or Conne	ectionSpec para	meters		
thod summar	ry: 	C) 01/17	DUTHCO			
a 🌚 runph	опероок (пльп нар	Garg): OUI	PUTMDG			
						_
2			2 Back	MAYE ~	Einich	Cancel

36. Click **Finish** to continue to the Java methods summary page.

37. The Java method summary page should look like that shown in the following Figure.

New J2C Bean		
ava Methods		(TT)
Add a Java method fo Additional m <mark>a</mark> thods ca	or each function or service you want to access from the EIS. an be added later in the Java source editor through the Snippets View.	
ava methods:		Sec. 1
🏶 runPhoneBo	bk (INPUTMSG arg) : OUTPUTMSG	Add
		Edit
		Remove
nteractionSpec class:	com.ibm.connector2.ims.ico.IMSInteractionSpec	
InteractionSpec pro	perties for 'runPhoneBook'	
Interaction verb:	SYNC_SEND_RECEIVE (1)	•
IMS request type:	IMS_REQUEST_TYPE_IMS_TRANSACTION (1)	•
Execution timeout:	0	
Commit mode:	SEND_THEN_COMMIT (1)	•
Advanced >>		
0	< Back Next > Einish	Cancel

Figure 21. Summary for the Java method

38. Click **Next** to complete the Java method creation.

39. The Deployment Information page displays, as shown in the following Figure. From here you could complete the J2C Java bean generation. However, Rational Developer for System z provides additional generation capabilities (Web page, Web service, or EJB) to consume this J2C Java bean.

- a. Select the **Create a Web Page, Web Service, or EJB from the J2C bean** check box. More options will become available, as shown in the following Figure.
- b. Select **Simple JSP**.
- c. Click **Next** to continue:

Figure 32. Specifying the J2EE resource type for deployment

New J2C Bean - 🗆 × Deployment Information Create a Java EE resource that makes use of your J2C Java Bean. Press F1 for more information. Create a Web page, Web Service, or EJB from the J2C bean. Java EE Resource Type: EJB C EJB Faces JSP C Faces JSP Simple JSP JSP C Web Service Java bean WebService ? < Back Next > Cancel

The **JavaServer Pages (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

Create a J2C application for IMS Phonebook transaction using IMS TM Resource Adapter © Copyright IBM Corporation 1994, 2009. All rights reserved. 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 your J2C bean. Web services are covered in the optional Task 4 of this tutorial.

40. From the Simple JSP Creation dialog, enter m_{yJSPs} for the **JSP folder** name and click **Finish** to complete the simple JSP, as shown in the following Figure.

igure 33. Cr New J2C Be	eating a JSP folder ean			_ _ _ X
Simple JSP (Choose to cre	C reation ate simple JSPs with default	input fields.		
Web project: JSP folder: Advanced >>	J2CPhoneBook			Browse
0		< <u>B</u> ack	Next >	Einish Cancel

Notice that after clicking **Finish**, the Rational Developer for System z workspace opens up the **PhoneBookImpl.java** file. This is the implementation of the mapping between COBOL and Java. It contains the <code>runPhoneBook()</code> method, as well as other generated methods. It also contains the reference to the JNDI name <code>PhoneBookJNDIName</code>, which is used to look up the Managed Connection factory that we created. The connection information (along with the input data) is passed to the IMS TM Resource Adapter, which in turn calls IMS.

41. Expand **Java Resources: src > sample.ims** under the J2CPhoneBook project, as shown in the following Figure. Take a moment and look at the generated components in the Project Explorer. The **myJSPs** folder contains the newly generated JSPs that will be used to test our J2C bean implementation.

Figure 34. J2CPhoneBook project in the Project Explorer and the PhoneBookImp.java file

File Edit Source Refactor Navigate Search P] ➡️ • 🖃 📥] 🏇 • 💽 • 🚱 • 💁 •] ½ • 🗐 • *- 🔑 • •	rroject Run Window Help] 🥖] 📸 + Sỹ +] 🧐 🔗 +] 🖓 ∋ J 😵] 🖧 🛛 🖺 😭
Project Explorer 🛛 📄 🔩 🏱 🗖 🗋	Sechnology Quickstarts
 imsico1021 J2CPhoneBook Deployment Descriptor: J2CPhoneBook Web Site Navigation Java Resources Java Resources Java Resources Java Resources PhoneBook, java PhoneBook, java Image Sampleims, data Image Libraries 	<pre>package sampleims; import javax.resource.ResourceException; import javax.resource.cci.Connection; import javax.resource.cci.Interaction; import javax.resource.cci.ConnectionFactory; import javax.resource.cci.ConnectionSpec; import javax.resource.cci.InteractionSpec; import javax.resource.cci.Record; import javax.resource.cci.ResourceAdapterMet;</pre>
Web Diagram JavaScript Support Security Editor WebContent WebContent META-INF Method.jsp Method.jsp Method.jsp WeB-INF WeB-INF WeB-INF	<pre>/** * @j2c.connectionFactory indi-name="MyDefau * @j2c.connectionSpec class="com.ibm.connec * @generated */ */ * Problems 2 Tasks Properties 2 Data Source Exp Server * State * * * * * * * * * * * * * * * * * * *</pre>

Task 3: Deploy and test your application

In this section, you will start the WebSphere Application Server, add your project to the application server runtime environment, and test your application using the simple JSP client that was created as part of the J2C Java Bean wizard.

Deploying your application

- 1. Select the **Servers** view within the J2EE perspective.
- Using the Servers view, select the WebSphere Application Server v7.0 and click the green arrow icon to start the server, as shown in the following Figure. It will take a few moments for the application server to start:

Figure 35. Starting the WebSphere Application Server

🕺 Problems	🤕 Tasks		🕫 Servers 🗵	🙀 Data Source Explor	🚡 Snippets	Annotations	
						\$ O \$	- 80
Server 🔺			State		Status	43	
webS	iphere Appl	ication Server vi	7.0 at loca 晶 Stopp	ed			

- 3. After the Console displays the "Server server1 open for e-business" message, click the **Servers** view and check the Status indicator.
- 4. When the WebSphere Application Server is started, the **Servers** view will display **Started** in the **Status** field, and **Synchronized** in the **State** field, as shown in the following Figure.

Figure 36. Status and State of the WebSphere Application Server

🛃 Problems	🧟 Tasks	🔲 Propertie	해 Servers 🛛	Data Sou	🛅 Snippets	Ca Annotati	📮 Console	
						Ť,	00	•
Server 🔺			State			Status		
zo Webs	iphere Appl	ication Server v	/7.0 at loca 指 Sta	rted		Synchronize	d	

 You now must add your project to the started application server. Using the Servers view, right-click WebSphere Application Server v7.0 and select Add and Remove Projects, as shown in the following Figure.

Figure 37. Adding your project to the started application server

import javax.resource.cc: import javax.resource.cc: import javax.resource.cc:	 Restart in Profile Stop Publish Clean 	Ctrl+Alt+S Ctrl+Alt+P	onnectionSpec : Co onnection : Connec onnectionFactory : unPhoneBook(INPU etConnectionFactor
<pre> # @j2c.connectionFactory # @j2c.connectionSpec c: # @generated #</pre>	Add and Remove Projects Monitoring	÷,	etConnectionFactor etConnection() etConnection(Conn
*/ */ Problems 🕢 Tasks 🗔 Propertie 👫 Si	 Create tables and data sources Reconnect debug process Server configuration Universal test client Administration 	• • •	etConnectionSpec(
Server - WebSphere Application Server v7.0 at	Properties loca 📻 started	Alt+Enter Synchroniz	ed

6. From the Add and Remove Projects page, select **J2CPhoneBookEAR** and click **Add** > to add your project to the Configured Projects, as shown in the following Figure.

Figure 38. Adding your project to configured projects

Add and Remove Projects	×
Add and Remove Projects Modify the projects that are configured on the serve	er 📕
Move projects to the right to configure them on the Available projects:	e server Configured projects:
() KBack	Next > Finish Cancel

Create a J2C application for IMS Phonebook transaction using IMS TM Resource Adapter © Copyright IBM Corporation 1994, 2009. All rights reserved. 33 of 54 7. Click **Finish** to add your project to application server. The WebSphere Application Server will publish the application, as shown in the following Figure.

Figure 39. Publishing the application

🖹 Problems 🧟 Tasks 🗖 Propertie 👫 Serve	rs 🛛 🙀 Data Sou 🚡 Snippets	🗔 Annotati 🛛 📮 Console 🕅 🗖
		🅸 🔾 🖉 📕 🚇
Server 🔺	State	Status
😐 📷 WebSphere Application Server v7.0 at loca	🖡 Started	Publishing

8. WebSphere Application Server then returns to a **Synchronized** state, as shown in the following Figure.

Figure 40. Returning to synchronized state after publishing the application

🧾 Problems 🛛 🖉 Tasks 🔲 Propertie	🖧 Servers 🔀 📲 Data	a Sou 📔 Snippets	🔤 Annotati	📮 Console	- 8
			\$	\$ 🔾 🖉	e (P
Server 🔺	State		Status		
😐 📷 WebSphere Application Server 🕯	/7.0 at loca 👬 Started		Synchronize	ed	

9. If not switched automatically, switch to the **Console** view and verify that the application has started successfully, as shown in the following Figure.

Figure 41. Console view

	Pro	blems 😔	Tasks	Propertie	해 Servers	Data Sou	🛅 Snippets	Ca Annotati	Console	2 - 0
We	bSpl	here Appli	cation Se	erver v7.0 at lo	calhost (Web	Sphere Applicat			F 🚱 🛃	🖸 • 👩 •
r	I	Sessio	onCont	extRegist	ry getSe	essionCont	ext SESNO	0176I: Wi	ll create	: a new 🛌
:	I	com.ik	om.ws.	wswebcont	ainer.Vi	irtualHost	addWebAp	plication	n SRVEO25	OI: Web
Ig	A	WSVF	202211	: Applics	tion sta	arted: J2C	PhoneBook	CEAR		and the second second second
Jn	A	WSVE	201911	[: Composi	tion uni	it WebSphe	re:cuname	=J2CPhone	BookEAR	in BLA [
4			1							•

Testing your application

In this section you will test the simple JSP client that was created as part of the J2C Java Bean wizard.

- Select the generated **TestClient.JSP** file that is located in the *myJSPs* folder. Recall that this is the folder you created to store the simple JSP test client. If the **TestClient.JSP** file is not visible, navigate to the **J2CPhoneBook > WebContent > myJSPs** folder in the Project Explorer.
- 2. With the **TestClient.jsp** selected, Right-click and select **Run As > Run On Server**, as shown in the following Figure.

Figure 42. Running the TestClient.jsp

💽 Java EE - J2CPhoneBool	Open F3	mpl.java - IBM Rational Developer for Syste		
File Edit Navigate Search ● ● ● • ● ● ● ● •	Open With Ctrl+C	·] 🤞 🖏 • 🞯 • 🥮 🔗 • 🎱		
Project Explorer	X Delete Delete Move Rename F2	ology Quickstarts PhoneBookImpl.java Σ pkage sampleims;		
E 22CPhoneBook E 25 Deployment Desc	≧n Import ≧ Export	port javax.resource.ResourceExc port javax.resource.cci.Connect		
⊡ 🤔 Java Resources È 🥵 src È - 🕀 sampleim:	Refresh F5	<pre>port javax.resource.cci.Interac port javax.resource.cci.Connect port javax.resource.cci.Connect</pre>		
⊕… D Phon ⊕… D Phon ⊕… D Phon ⊕…∰ sampleim:	Transform Validate	<pre>port javax.resource.cci.Interac port javax.resource.cci.Record; port javax.resource.cci.Resourc</pre>		
🗄 🛋 Libraries	Compile JSPs Run As	🔓 1 Run on Server 🛛 Alt+Shift+X, R		
⊡	Debug As Profile As	Run Configurations		
Build Build WebContent META-INF Book myJSPs Input.jsp Method is	Team Compare With Replace With Page Template Source	©j2c.connectionSpec class="com @generated ms		
Result.js; Result.js; TestClient	Properties Alt+Enter	e Application Server v7.0 at localhost (WebSphere SessionContextRegistry getSessio)		
		com.ipm.ws.wswepcontainer.Virtua.		

3. Choose the started WebSphere Application Server v7.0 and click **Finish** to run the TestClient.jsp, as shown in the following Figure.

Figure 43. Choosing the server to run the JSP

🖸 Run On Server				
Run On Server				
Select which server to	use			
How do you want to sel	ect the server?			
Choose an existin	g server			
C Manually define a	new server			
Select the server that y	ou want to use:			
type filter text				
WebSphere Application	Server v7.0 ver when running this pro	oject		
0	< Back	Next >	Finish	Cancel

4. The Web Services Test Client will launch the TestClient.jsp. Notice that there are three separate panes (Java Methods, Input parameters, and Results) within the TestClient.jsp, as shown in the following Figure.

Figure 44. The Web Services Test Client view



5. Double-click the **Web Services Test Client** title to maximize the window within the workbench, and then click the runPhoneBook method to test.

Create a J2C application for IMS Phonebook transaction using IMS TM Resource Adapter © Copyright IBM Corporation 1994, 2009. All rights reserved. 37 of 54

- 6. Enter the following values to test the J2C bean, as shown in the following Figure.
 - **a.** in__ll: 59
 - $b. \quad \text{in_trcd: in} \\$
 - C. in_zz: 0
 - $d. \quad \text{in_cmd:} \text{ display}$
 - e. in__name1: LAST1
- 7. Make sure not to enter any spaces before or after any of the input fields.
- 8. Click **Invoke** to run the application.

Figure 45. Testing TestClient.jsp in the Web Services Test Client

Methods Inputs <	WebDiagram.gph	PhoneBookImpLjava	Web Services Te	st Client X
Methods Inputs • runPhoneBook (sample.ims.data.INPUTIv/ (sample.ims.data.INPUTIv/ (sample.ims.data.INPUTiv/ (sample.ims.data.INPUTiv/ (sample.ims.data.INPUTiv/ (sample.ims.data.INPUTiv/ in_extn: in_ll: in_trcd: recordShortDescription: in_zip: recordName:	🗢 🔿 🔳 🤗 [http:	//localhost:9082/J2CPhoneBo	ok/myJSPs/TestClient.jsj	p
• <u>runPhoneBook</u> <u>(sample ims. data INPUTIv</u> in_extn: in_11: in_11: in_trcd: IVTNO recordShortDescription: in_zip: recordName:	Methods	[≞] Inp	uts	
in_extn: in_ll: in_trcd: in_zip: recordShortDescription: in_zip: recordName:	runPhoneBoo (sample ims d	k arg		
in_trcd: IVTNO recordShortDescription: in_zip: recordName:	(semple.ans.o	in	_extn: II:	59
recordShortDescription:			trod:	IVTNO
recordName:		reco	ordShortDescription	
		reco	ordName:	
		in	<u>,77.</u>	0
in name2:		in	_cmd: name2:	DISPLAY
in_name1:		in	- _name1:	LAST1
Invoke, Clear		Invoke	Clear	

9. Resize the result area to check your results, as shown in the following Figure.

Figure 46. Resizing the result area



You should see something similar to the results shown in the following Figure.

Fia	ure	47.	Results	from	the	testina
••9	a. c		Resalts			cesting

Result	*
retump:	
out_segno:	0009
out_zz:	768
recordShortDescri	ption: sample.ims.data.OUTPUTMSG
out_zip:	D01/R01
out_extn:	8-111-1111
out_msg	ENTRY WAS DISPLAYED
out_cmd:	DISPLAY
out_1	93
outname2:	FIRST1
out_name1:	LAST1
recordName:	sample.ims.data.OUTPUTMSG
bytes:	[0,93,3,0,-59,-43,-29,-39,-24,64,-26,-63,-30,64,-60,-55,-30,- 63,-24,-59,- 60,64,64,64,64,64,64,64,64,64,64,64,64,64,
٠	

- 10. Select **Windows** > **Reset** perspective and **OK**.
- 11. Using the **Servers** tab, click the red square icon to stop the WebSphere Application Server, as shown in the following Figure.

Figure 48. Stopping the WebSphere Application Server

扎 Problems 🤕 Tasks 🔲 Propertie 👭 Serve	rs 🖾 🛛 🙀 Data Sou	🛅 Snippets 🗖 🗖 An	notati 📃	Console 🗖 🗖
			参 0) 🖉 🔍 🕑
Server 🔺	State	Stat	üs	NG
😐 📷 WebSphere Application Server v7.0 at loca	🖥 Started	Sync	hronized	

12. Close all open editor windows (pressing Ctrl+Shift+F4 should accomplish this).

Create a J2C application for IMS Phonebook transaction using IMS TM Resource Adapter © Copyright IBM Corporation 1994, 2009. All rights reserved. 39 of 54



If you have extra time, you can do a bit more. How about taking the generated J2C bean and wrapping that as a Web service? This feature extends the usage of your J2C bean past simple Web clients and offers the transaction as a Web service.

Task 4 (Optional): Create a Web service to invoke the J2C bean

In the previous sections, you installed a J2C Resource Adapter, created a J2C bean that executed against the IMS sample transaction: IVTNO. You also created a simple JSP Test Client that tested your J2C bean. In this optional section, you will wrap the J2C bean as a Web service, and then test it by using the generated WSDL file. You will then generate a Web service client and test it as well.

Creating a Web service

1. Click **File > New > Other** to open the Select a Wizard dialog, as shown in the following Figure.

J2EE - Welcome to	o z/OS Projects - II	3M Rational Deve	eloper for Syste
File Edit Navigate S	earch Project Data	a Run Window	Help
New	Alt+Shift+N 🕨	Project	
Open File		Designat	-
Close	Ctrl+W		2005/2008 - 55
Close All	Ctrl+Shift+W	File	z/OS Projects
Save	Cul+S	E Sevemple	2
Save As		La campio	
Save All	Ctrl+Shift+S	📑 Other 🔥	pme to

What is a Web service?

Web services are self-contained, self-describing modular applications that can be published, located, and invoked across the Web. Businesses can dynamically mix and match Web services to perform complex transactions with minimal programming. Web services allow buyers and sellers all over the world to discover each other, connect dynamically, and execute transactions in real time with minimal human interaction.

The following standards play key roles in Web services: Web Services Description Language (WSDL) and Simple **Object Access protocol (SOAP).**

4. In the New wizard screen, select **Web Page**, **Web Service**, or **EJB from J2C Java Bean** and click **Next**, as shown in the following Figure.

New	the state of the s			
Select a wizard Create a Web Page, Web S J2C Java bean.	ervice or EJB that wra	ps the functional	lity provided by	a 📄
Wizards:				
type filter text				
Command Bear 22C Bean 22C Bean Web Page, We 32C Servic 32C Se	h •b Service, or EJB from e Migration hs	12C Java Bean		
Show All Wizards.	< Back	Next >	Finish	Cancel

Figure 50. Selecting the wizard to wrap the J2C Java bean into a Web service

5. Press Browse to look up the J2C bean Implementation class, as shown in the following Figure.

Figure 51. Clicking Browse to select the J2C bean



6. Type P, select the **PhoneBookImpl** class in package **sample.ims** and click **OK**, as shown in the following Figure.

Figure 52. Selecting the PhoneBookImpl class

elect entries:	~
P fatching types: PhoneBookImpl - sam	nole ims <
- I nonobestanipi san	ipio:into

- 7. This returns the fully-qualified implementation class in the dialog, as shown in the following Figure.
- 8. Press Next to continue.

Figure 53. The PhoneBookImpl class selected

J2EE Resource	e from J2C Java Bean 🛛 🔀
J2C Java bean s Choose the J2C Java Resource for.	selection a Bean implementation class that you want to create a J2EE
J2C bean implemen	tation: \J2CPhoneBook\src\sample\ims\PhoneBookImpl.java Browse
0	<back next=""> Finish Cancel</back>

- 9. Select Web Service as the J2EE Resource Type, as shown in the following Figure.
- 10. Click **Next**, and then **Finish**. The J2EE Resource from a J2C Java Bean wizard will now create the necessary J2EE resources for the Web service implementation. This might take a few moments.

ation				L. L
at makes use of yo	our J2C Java Bea	n. Press F1 for more i	nformation.	010
e;				
	`)) 	EJB Faces JSP		
Java bean	🔶 🧿	JSP WebService		
	x S Java bean	t makes use of your J2C Java Bear	s EJB Faces JSP Java bean WebService	x x y y y y y y y y y y y y y

Figure 54. Selecting Web Service as the resource type

11. Examine the generated J2EE resources for Web services. Expand J2CPhoneBook > WebContent > WEB-INF > wsdl. Note the generated PhoneBookImpl.wsdl file and associated xml mapping files, as shown in the following Figure.

Figure 55. Generated J2EE resources for Web services based on the PhoneBookImpl J2C bean



13. Double-click the **PhoneBookImpl.wsdl** file to open up the WSDL editor. Observe the visual representation of the wsdl file in the **Design** view, as shown in the following Figure. There is also a source view, if you prefer.

Note that the Web service operation is **runPhoneBook** and that the operation has one input and one output, and that these map to the INPUTMSG and OUTPUTMSG defined previously.

Figure 56. PhoneBookImpl.wsdl in the Design view

A PhoneBookImplService		🕕 Phon	eBookImpl
PhoneBookImpl	🕸 runPhor	eBook	
http://localhost:9080/LAB	(input)	☐ parameters	runPhoneBook
	Output	A P parameters	e runPhoneBookResponse



Note that the http://localhost:portnumber/... is generated. This tutorial uses port 9081.

Create a J2C application for IMS Phonebook transaction using IMS TM Resource Adapter © Copyright IBM Corporation 1994, 2009. All rights reserved. 44 of 54 If the port number is not 9081, in the **Design** view, click the http that is displayed in http://localhost:*portnumber*/J2C....

Click the **Properties** tab and change the port number to 9081, as shown in the following Figure.

Save the changes (Ctrl+S).

Figure 57. Port number to use for this tutorial 🌌 *PhoneBookImpl.wsdl 🖾 🎱 Technology Quickstar 🎱 Web Services Test Cl Web Services Explore 1 View: default 💽 PhoneBookImpl 🔏 PhoneBookImplService 🎡 runPhoneBook PhoneBookImpl http://localhost:9080/J2C. parameters 🕑 runPhoneBook input] 🗘 output e runPhoneBookResponse Design Source 🖧 Servers 🙀 Data Source Explorer 🚼 Problems 📈 Tasks 🔲 Properties 🔀 🔚 Snippets 🗔 Annotations Console port Ensure that the port Name: PhoneBookImpl number is 9081 General PhoneBookImplSoapBinding Binding: Documentation Extensions http://localhost:9080/J2CPhoneBook/services/PhoneBookImpl Address: SOAP Destacale

Testing the Web service using Web Service Explorer

In this part of the tutorial, you will use the Web Services Explorer to test your Web service.

1. Make sure that WebSphere Application Server is running. You will test your Web service using the WebSphere Application Server test environment. Using the Servers view, note the **Status**. If the WebSphere Application Server is not running, press the **green arrow** on the **Servers** tab to start it, as shown in the following Figure.. This will take a few moments.

Figure 58. Starting the WebSphere A	pplication Server	•			
🖹 Problems 🖉 Tasks 🔲 Propertie 👯 Serve	rs 🖾 🛛 🎁 Data Sou	🛅 Snippets	Ca Annotati	Console	
			Ŕ	¢ Q Ø	- 66
Server 🔺	State		Status	NS	
😐 📷 WebSphere Application Server v7.0 at loca	🖥 Stopped		Republish		

 From the Project Explorer view, right-click the PhoneBookImpl.wsdl file and select Web Services > Test with Web Services Explorer, as shown in the following Figure.

Figure 59. Testing with Web Services Explorer



3. The **Web Services Explorer** window will launch. It contains two views: a **Navigator** view and an **Actions** view. To make more space, maximize the window by double-clicking the title, as shown in the following Figure.

Figure 60. Double-clicking the Web Services Explorer tab to maximize the window



4. The Actions view specifies both the operations and endpoints for the Web service. The Endpoint is defined as http://localhost:9081/J2CPhoneBook/services/PhoneBookImpl. Note that this is the location of the Web service and that it points to the WebSphere Application Server running on the workstation and listening on port **9081**, as shown in the following Figure.

Figure 61. The location of the Web service

Web Services Explorer

🍄 Navigator 🥏 🏑	Actions	
WSDL Main @ file:/C:/Workspaces7.5/SANDBOX/J2CPhone	WSDL Binding De	tails
E-22 PhoneBookImplService E-® PhoneBookImplSoapBinding	Shown below are the de additional endpoints.	tails for this SOAP <binding> element. Click on an operation</binding>
	Name	Documentation
	runPhoneBook	
	▼ Endpoints Add	Remove
		Endpoints
	http://localhost:90	081/J2CPhoneBook/services/PhoneBookImpl
	Go Reset	3.5

5. Click the runPhoneBook operation, as shown in the following Figure.

Figure 62. The runPhoneBook operation for the Web service in Web Services Explorer

😵 🖉	Actions
器 WSDL Main 白 - 盈 file:/C:/Workspaces7.5/RDz/J2CPhoneBook/W 白 - 盈 PhoneBookImplService 由 - ⑧ PhoneBookImplSoapBinding	WSDL Binding Details Shown below are the details for this SOAP <binding> element. Click on a additional endpoints. Operations</binding>
	Name Doc

6. This will invoke the WSDL operation, as shown in the following Figure.

Figure 63. The WSDL operation

रेड- Navigator 🔗 📿	Actions
K WSDL Main	Invoke a WSDL Operation
	Enter the parameters of this WSDL operation and click Go to invoke.
	http://localhost:9082/J2CPhoneBook/services/PhoneBookImpl
	✓ Body
	▼ runPhoneBook
	💌 arg 🗖 nil?
	recordName string Ini?
	recordShortDescription string Init?
	bytes base64Binary

- Enter the necessary information to run the IVTNO transaction, as shown in the following Figure.
 67. These will be the same parameters used for TestClient.jsp:
 - **a**. **in_ll:** 59
 - b. in_zz: 0
 - c. in_trcd: IVTNO
 - d. in_cmd: display
 - e. in__name1: LAST1

8. Click **Go** to run the Web service:

Figure 64. Specifying the values for the parameters and clicking Go to run the service
Web Services Explorer

🕾 Navigator	🖉 🖉 🛛 Actions
WSDL Main 	CPhoneBook/W in trcd string nil? IVTNO in cmd string nil? DISPLAY
	in name1 string Inil? LAST1 in name2 string Inil?
	in <u>extn</u> string Inil?
	Go Reset

9. The Web service executes, and the results are displayed in the **runPhoneBookResponse** object, as shown in the following Figure. Scroll through the Status window or resize the window, as necessary, to see the results:

/eb Services Explorer								
🛠 Navigator 🖉 📿	2 Actions							
WSDL Main WSDL Main Solution PhoneBookImplService PhoneBookImplSoapBinding Solution PhoneBookImplSoapBinding Solution PhoneBookImplSoapBinding Solution PhoneBookImplSoapBinding Solution PhoneBookImplSoapBinding PhoneBookImplSoapBinding PhoneBookImplService PhoneBookImplService P	Solution							
	I							
	1 Status							
	▼ runPhoneBookResponse							
	✓ sunPhoneBookReturn							
	recordName (string): sample.ims.data.DUTPUTMSG recordShortDescription (string): sample.ims.data.DUTPUTMSG bytes (base64Binary): AF0DAMXV49noQDbB4kDEyeLX08HoxcR/ out_l (short): 93 out_zz (short): 98							
out_mog (string): ENTRY WAS DISPLAYED								
						out_cmd (string); DISPLAY		
	outname1 (string): LAST1							
	out_name2(string): FIRST1							
	out_extn (string): 8-111-1111							
	out_zip (string): D01/R01							
	out_segno (string): 0013							

Figure 65. Resizing to maximize the Status window

10. Click the **Source** link to display the Web service results in the Form view, as shown in the following Figure.

Figure 66. Clicking the Source link in the Status window

1 Status	4
	Source
▼ runPhoneBookResponse	
🝷 runPhoneBookReturn	*
recordName (string): sample.ims.data.OUTPUTMSG	
recordShortDescription (string) sample.ins.data.OUTPUTMSG	
bytes (base64Binaty): AF0DAM04/49noQ0bB4kDEyeL/08HoxcRAQEB	AQEBAQEBAQBAQEBAQEDEyeLX08HoQNPB4uPxQE
out_II(short): 93	

11. This will display the actual SOAP response and SOAP request envelope. Scroll down to see the data being transmitted to and from IMS, as shown in the following Figure.

Figure 67. SOAP request and response envelopes



What is SOAP?

Simple Object Access Protocol (**SOAP**) is an XML-based standard for messaging over HTTP and other Internet protocols. It is a lightweight protocol for the exchange of information in a decentralized, distributed environment. It is based on XML, and consists of three parts:

- An envelope that defines a framework for describing what is in a message and how to process it.
- A set of encoding rules for expressing instances of application-defined data types.

A convention for representing remote procedure calls and responses.

11. When you are done testing, in the **Server** view, click the red square icon to stop the WebSphere Application Server, as shown in the following Figure.

Figure 68. Stopping the WebSphere Application Server

扎 Problems 🧟 Tasks 🔲 Propertie 🍀 Se	rvers 🛛 🚺	🗎 Data Sou	5 Snippets	Ca Annotati	Console	- 8					
🏧 🖓 🖉 🛒 🔛											
Server 🔺	State	State			Status						
😐 📷 WebSphere Application Server v7.0 at l	loca 🚡 Started	R.		Synchronize	ed 🛛						

13. Close all open editor windows (pressing Ctrl+Shift+F4 to accomplish this).

Create a J2C application for IMS Phonebook transaction using IMS TM Resource Adapter © Copyright IBM Corporation 1994, 2009. All rights reserved. 52 of 54

You have completed the optional Web service generation part of the J2C bean tutorial!

Resources

Learn

- Visit the <u>Rational Developer for System z zone on developerWorks</u> for technical resources and best practices for using this product.
- Visit the IMS TM Resource Adapter Web site.
- Visit the <u>IMS zone on developerWorks</u> for technical resources on Information Management System.
- Visit the Information Management System (IMS) page for product and purchasing information.
- Subscribe to the <u>IBM developerWorks newsletter</u>, a weekly update on the best of developerWorks tutorials, articles, downloads, community activities, Webcasts and events.
- Subscribe to the <u>Rational Edge newsletter</u> for articles on the concepts behind effective software development.
- Browse the <u>technology bookstore</u> for books on these and other technical topics.

Get products and technologies

- Download a trial version of Rational Developer for System z.
- Download IMS TM Resource Adapter.
- Download <u>IBM product evaluation versions</u> and get your hands on application development tools and middleware products from all IBM brands including DB2®, Lotus®, Tivoli®, and WebSphere.

About the authors

- Evgueni Liakhovitch is a Software Developer at IBM.
- Yee-Rong in an Information Developer at IBM.
- Shahin Mohammadi-Rashedi is an IMS SOA Demonstration Team Lead at IBM.

Trademark notice

IBM, the IBM logo and ibm.com are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark

information" at www.ibm.com/legal/copytrade.shtml.