H01

IMS ConnectWorkshop

IMS Technical Conference

St. Louis, Mo September 30 - Oct 3, 2002

RETURN TO INDEX

Contents

This workshop features hands-on labs using IMS Connect and the new WebSphere Application Developer Integration Edition (WSAD IE) toolkit. During the exercises, you will have the opportunity to:

- configure and work with your own IMS Connect system
- get first-hand experience on how it interacts with IMS
- and create a Web Service to access an IMS transaction using the WSAD IE tool.

The workshop provides step-by-step instructions and is designed to be self-paced. Instructors are available to assist in any problems encountered.

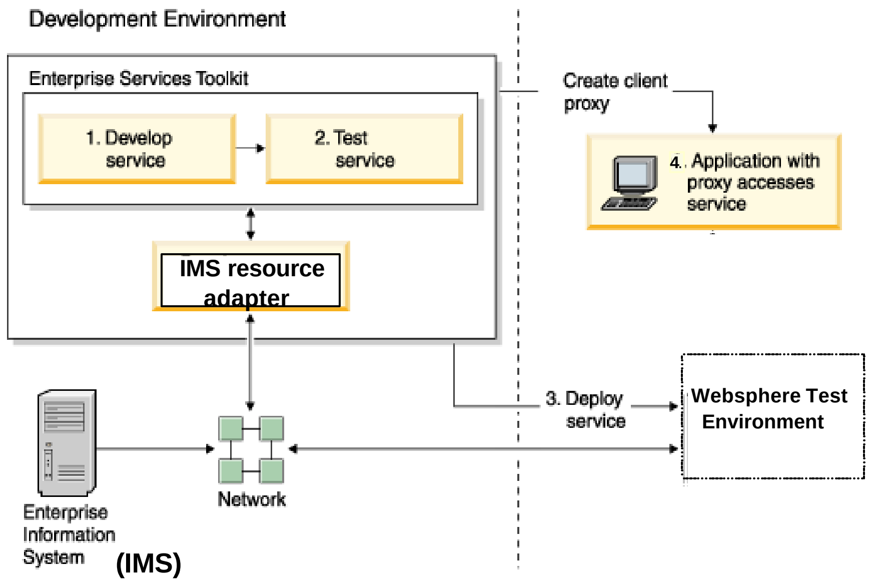
Notice

The goal of this document is to provide an overview of the configuration of IMS Connect and to allow IMS customers with the opportunity to work with some of the available products that can assist in web-enabling their IMS systems.

Objectives of this Workshop

This workshop is designed to provide hands-on experience in the configuration of IMS Connect and the creation of a service to drive a transaction in IMS.

A “service”, more specifically, an “enterprise service” is the term used to refer to code that runs on an application server such as Websphere that drives a request to invoke access to an Enterprise System such as IMS. The creation of the “service” follows standard rules. The following diagram shows the process that we will use. It begins by developing the service with the set of tools in WSAD IE. After testing, the service can be moved to the production environment on an application server. In this workshop we will simulate the deployment by using the WebSphere$^{(R)}$ Test Environment in WSAD IE. This part of the tool emulates the actual Websphere EE application server within the development environment. Finally, we will use the tool to create a client proxy and add that proxy to the client application accessing the service.


Development Environment
Enterprise Services Toolkit
1. Develop service
2. Test service
IMS resource adapter
Create client proxy
4. Application with proxy accesses service
3. Deploy service
Websphere Test Environment
Enterprise Information System
(IMS)
Network

Exercise 1. Review the Environment.

What This Exercise is About

We will begin by reviewing the workstation environment and then proceed to a familiarization with the host.

Tasks:

1. If your workstation has not already been started, proceed through the following:
   a. Start your system.
   b. When the logon panel appears, use the following **USERID ==> admin, PASSWORD ==> admin**

2. Identify your workstation hostname:
   a. Start a DOS command window.
   b. Type **hostname** on the command prompt. This is your workstation hostname and should be the same name as **teamxx** where **'xx'** is the number assigned to your team.
   c. An alternative way to identify your workstation hostname is:

      - Select (Highlight Only) Network ICON from Control Panel or Desktop.
      - Right Mouse Click, Select Properties. Under the Identification Tab, check Computer Name.

3. Determine your IP address and those of other workstations in the class.

   On a DOS command window prompt, enter **'PING teamxx'** where **'xx'** is your team number. Repeat this for other teams.

4. To logon to the MVS host, the following has been set up for you.

   Userid ==> TEAMxx, Password ==> TEAMxx where **'xx'** is the number assigned to you.

MVS Information

Left mouse click the CM650 icon. This will establish the connection to the MVS system.

Logon information: Userid=TEAMxx   Password=TEAMxx

```
                CUSTOMPAC MASTER APPLICATION MENU
OPTION ===>                                           SCROLL ===> PAGE

  IS  ISMF     - Interactive Storage Management Facility
  P   PDF      - ISPF/Program Development Facility
  IP  IPCS     - Interactive Problem Control Facility
  HC  HCD      - Hardware Configuration Definition
  SM  SMP/E    - SMP/E Dialogs
  R   RACF     - Resource Access Control Facility
  SD  SDSF     - System Display and Search Facility
  OS  Support  - OS/390 ISPF System Support Options
  OU  User     - OS/390 ISPF User Options
  DI  DITTO    - Data Interfile Transfer, Testing and Operations
  S   SORT     - DF/SORT Dialogs
  BMR BMR READ - BOOKMANAGER READ (Read Online Documentation)
  BMI BMR INDX - BOOKMANAGER READ (Create Bookshelf Index)
  BMB BMR BLD  - BOOKMANAGER BUILD (Create Online Documentation)
  IC  ICSF     - Integrated Cryptographic Service Facility
```

Use OPTION ‘P’ to access the ISPF/PDF environment

Use OPTION ‘SD’ to access the Console

Exercise 2. Configure IMS Connect

What This Exercise is About

In this exercise you will logon to the MVS system and work with IMS Connect. The activities will involve configuration, set-up and starting an IMS Connect address space and ensuring that it can communicate with IMS.

Tasks:

1. Logon to MVS. Your USERID ==> TEAMxx, PASSWORD ==> TEAMxx.

2. Access “TEAMxx.WORKSHOP.JCL” and edit member BPECFGXX.   Make sure the profile is “NUMS OFF”. The TSO command is “unnum”. Check that columns 72-80 are blank.

3. Create a new member BPECFGxx where xx is your team number. This will be used as your own BPE configuration member for tracing. For purposes of this workshop, delete all TRCLEV statements. We will not be requesting any internal traces.

4. Copy your BPECFGxx member to IMS610.PROCLIB.

5. Browse IMS610.PROCLIB(DFSPBIM6).

   a. Identify GRNAME, APPLID1 and IMSID values.

   b. Note all three values because you will need them for the IMS Connect Configuration. Recommended DATASTORE ID is the value in IMSID, though not required (easier to monitor which IMS the Client is communicating with).

6. Edit TEAMXX.WORKSHOP.JCL(ICONNXX)

   a. Change XX to your team value.

   b. Leave Security OFF.

```
******************************************************************
* IMS CONNECT CONFIGURATION FILE
******************************************************************
HWS (ID=ICONNXX)
TCPIP (HOSTNAME=TCPIP,PORTID=(34XX),EXIT=(HWSIMSO0,HWSSMPL0))
```

c. Add a DATASTORE statement.

DATASTORE statement	**FUNCTION**	**DFSPBxx member**
GROUP	XCF group name	GRNAME
ID	Name that client specifies for an IMS	Any value, e.g., IMSID
TMEMBER	XCF name for the target IMS	APPLID1, or USERVAR, or OTMANM
MEMBER	XCF name for this IMS Connect	------

```
DATASTORE (ID=IM7P, MEMBER=ICONNxx,GROUP=IM7POTMA,TMEMBER=IM7P)
```

d. Copy File to IMS610.PROCLIB as ICONNxx where xx is your team number. Again make sure that columns 72-80 are blank.

7. Edit TEAMXX.WORKSHOP.JCL(ICONNJXX)
   a. Change XX to your team number.
   b. Submit the job.

```
//ICONNXX  JOB CLASS=A,MSGCLASS=O,REGION=4M
// USER=TEAMXX,NOTIFY=TEAMXX
//****************************************************
//*  BRING UP AN IMS CONNECT SYSTEM
//****************************************************
//STEP1    EXEC PGM=HWSHWS00,TIME=1440,
//              PARM='BPECFG=BPECFGXX,HWSCFG=ICONNXX'
//STEPLIB  DD  DSN=IMSCON.SHWSRESL,DISP=SHR
//PROCLIB  DD  DSN=IMS710.PROCLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=O
//SYSUDUMP DD  SYSOUT=O
//HWSSVSTC DD  DSN=TEAMxx.ICONN.TRACE,DISP=SHR
//HWSRCORD DD  DSN=TEAMxx.ICONN.HWSRCORD,DISP=SHR
```

8. View the status of your IMS Connect (ICONNxx) job.

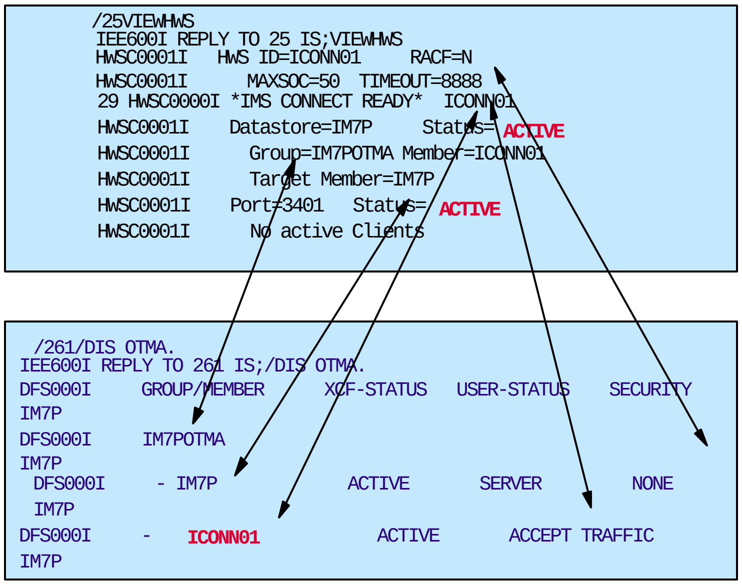
   From the SDSF - LOG panel, enter ‘**/rr**VIEWHWS’ where ‘**rr’** is the outstanding prompt for your ICONNxx job. Note that the commands on this console must always start with a ‘/’. This is a requirement for this system and does not imply an IMS command.

```
/25VIEWHWS
IEE600I REPLY TO 25 IS;VIEWHWS
HWSC0001I   HWS ID=ICONN01     RACF=N
HWSC0001I      MAXSOC=50  TIMEOUT=8888
29 HWSC0000I *IMS CONNECT READY*  ICONN01
HWSC0001I    Datastore=IM7P     Status=ACTIVE
HWSC0001I      Group=IM7POTMA Member=ICONN01
HWSC0001I      Target Member=SCSIM7P
HWSC0001I    Port=3401   Status=ACTIVE
HWSC0001I      No active Clients
```

9. View the status of your connection to IMS from the IMS side. Enter ‘/**rr**/DIS OTMA’ where ‘**rr’** is the outstanding prompt for IMS. Note also that there is a ‘/’ after **rr** since this is an IMS command.

```
/261/DIS OTMA.
IEE600I REPLY TO 261 IS;/DIS OTMA.
DFS000I     GROUP/MEMBER      XCF-STATUS   USER-STATUS    SECURITY
IM7P
DFS000I     IM7POTMA
IM7P
DFS000I     -IM7P              ACTIVE       SERVER         FULL
IM7P
DFS000I     -ICONN01          ACTIVE       ACCEPT TRAFFIC
IM7P
```

10. Review IMS Connect and OTMA Relationships.



```
/25VIEWHWS
IEE600I REPLY TO 25 IS;VIEWHWS
HWSC0001I   HWS ID=ICONN01      RACF=N
HWSC0001I      MAXSOC=50  TIMEOUT=8888
29 HWSC0000I *IMS CONNECT READY*  ICONN01
HWSC0001I    Datastore=IM7P      Status= ACTIVE
HWSC0001I      Group=IM7POTMA Member=ICONN01
HWSC0001I      Target Member=IM7P
HWSC0001I    Port=3401    Status= ACTIVE
HWSC0001I      No active Clients
```

```
/261/DIS OTMA.
IEE600I REPLY TO 261 IS;/DIS OTMA.
DFS000I     GROUP/MEMBER      XCF-STATUS   USER-STATUS     SECURITY
IM7P
DFS000I     IM7POTMA
IM7P
 DFS000I     - IM7P              ACTIVE       SERVER          NONE
 IM7P
DFS000I     -    ICONN01            ACTIVE       ACCEPT TRAFFIC
IM7P
```

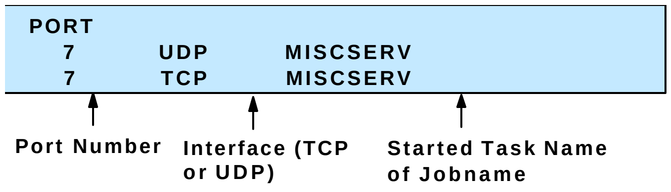
IMS Connect Display	OTMA Display
Datastore=IM7P	IM7P

RACF=N
RACF-Y
	SECURITY=NONE
	SECURITY=Profile, CHECK or FULL

Status=ACTVE	USER-STATUS Accept Traffic
MEMBER=ICONN01	GROUP/MEMBER ICONN01
Target Member = SCSIM7P	GROUP/MEMBER SCSIM7P
IM7POTMA	GROUP/MEMBER IM7POTMA

11. Things to review and try.

a. In your own environment, you might want to consider reserving the TCP/IP port that you specified in the TCPIP statement of the configuration file (HWSCFG=). This would be done by the TCP/IP administrator. You can check this by looking in the 'TCPIP.PROFILE.TCPIP' dataset. This is the default TCP/IP data set name and may be different in your particular environment. For this class, the dataset is 'TCPIP.TCPIP.TCPPARMS(TCPPROF)'. Browse this data set (do not edit it). Issue a 'F PORT 1' command. You will notice that this will bring you to the list of reserved ports.



```
PORT
   7        UDP        MISCSERV
   7        TCP        MISCSERV
```

To add your IMS Connect ports to this environment, the TCP/IP administrator would add:

(note - do not do this for this environment)

```
3401 TCP ICONN1         ; IMS Connect Port 1
3402 TCP ICONN2         ; IMS Connect Port 2
...                       ...
```

b. Review and test the following commands:

The format is: **'/rr**COMMAND**'** where **rr** is the outstanding prompt of your IMS Connect address space, e.g., /**45**VIEWHWS.

IMS Connect command	**Description**
CLOSEHWS	Terminates IMS Connect
OPENDS *datastore id*	Starts communication with an IMS
OPENPORT *port id*	Reestablishes communications with TCP/IP
SETRACF *ON* | *OFF*	Defines the security environment
STOPCLNT *port id client id*	Terminates communications with a specific client on a port
STOPDS *datastore id*	Terminates communications with an IMS
STOPPORT *port id*	Terminates listening on a specific TCP/IP port
VIEWDS *datastore id*	Displays current activity with an IMS
VIEWHWS	Displays current activity of IMS Connect from both the port and datastore perspectives
VIEWPORT *port id*	Displays the current activity of a port

Exercise 3. Bring up WSAD IE.

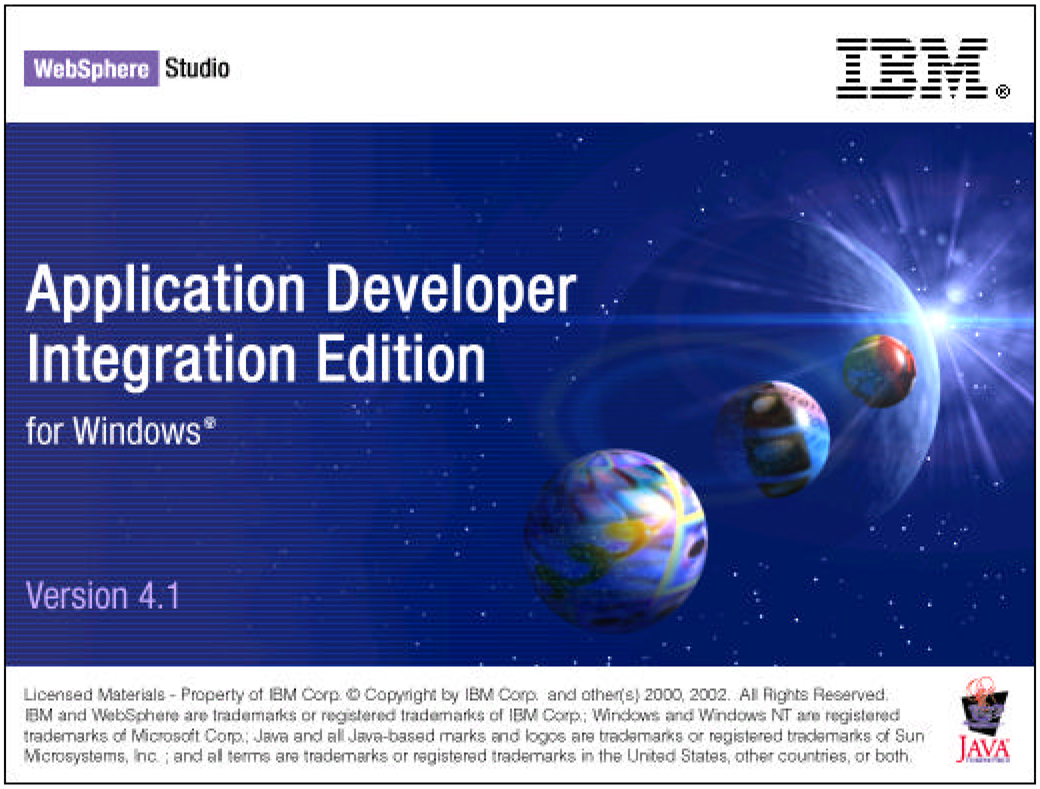
What This Exercise is About

The purpose of this exercise is to provide a familiarization with the WSAD IE (Websphere Studio Application Developer - Integration Edition) toolkit.

Application Developer Integration Edition consists of the following tools and components:
- **A set of tools to develop a service, collectively called the Enterprise Services Toolkit. These tools include a subset of tools to let you work with multiple services, creating another service out of the many. It is called a service flow. A service flow connects the output of one service to the input of another.**
- **A set of resource adapters. Resource adapters let services access Enterprise Information Systems (EIS). Examples of EIS systems are IMS and CICS.**

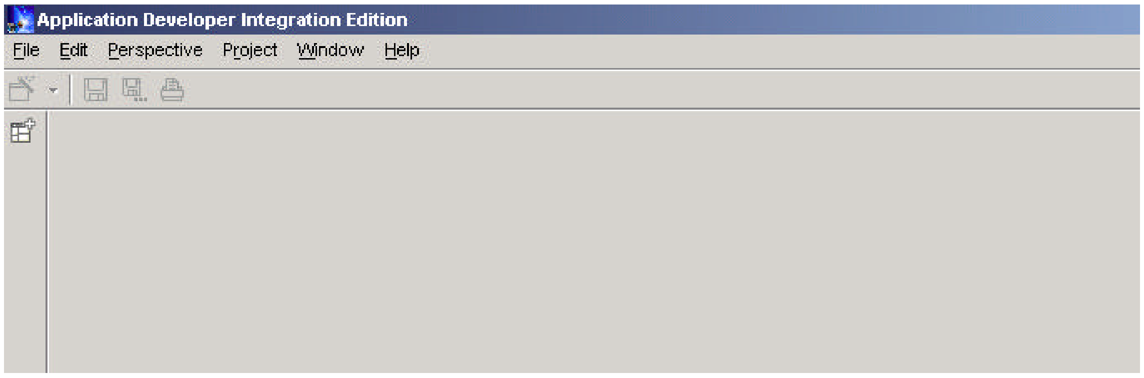
WSAD IE is a general-purpose application development environment and also includes the capabilities provided by its predecessor toolkits. For example, the wizards to create EJBs, JSPs, servlets, XML-based documents, Web pages, and so forth. Additionally, WSAD IE has access to the operating system resources such as the file system, databases, and application servers. The toolkit, therefore, is one of the most complete development environments for those designing services for enterprise applications.

1. At the bottom left of your workstation screen, select **Start ==> Programs ==> IBM Websphere Studio Application Developer Integration Edition ==> IBM Websphere Studio Application Developer Integration Edition.** This brings up the toolkit



Licensed Materials - Property of IBM Corp. © Copyright by IBM Corp. and other(s) 2000, 2002. All Rights Reserved. IBM and WebSphere are trademarks or registered trademarks of IBM Corp.; Windows and Windows NT are registered trademarks of Microsoft Corp.; Java and all Java-based marks and logos are trademarks or registered trademarks of Sun Microsystems, Inc.; and all terms are trademarks or registered trademarks in the United States, other countries, or both.

2. Once the tool has been initialized, your workspace is created and presented for your use.


Application Developer Integration Edition
File Edit Perspective Project Window Help

You are now ready to begin creating an IMS Service.

.....................................(This page intentionally left blank)..

Exercise 4. Create an IMS Service.

There are three basic steps that need to be done to successfully create an IMS Service. These include:

4a. Create a service project
4b. Import a Cobol file
4c. Create an IMS service definition

4a. Create a service project

****
Some terminology - "project" and "perspective".

A project is a mechanism for organizing resources. It is composed of folders (directories) and files. There are different kinds of projects in the WSAD IE tool. For example: Java projects are for stand-alone applications; Web projects for Web applications; EJB (Enterprise Java Bean) projects for EJB development; EAR projects tie together Web and EJB (and Client) projects into a J2EE hierarchy; Server projects are used to define application servers for testing.

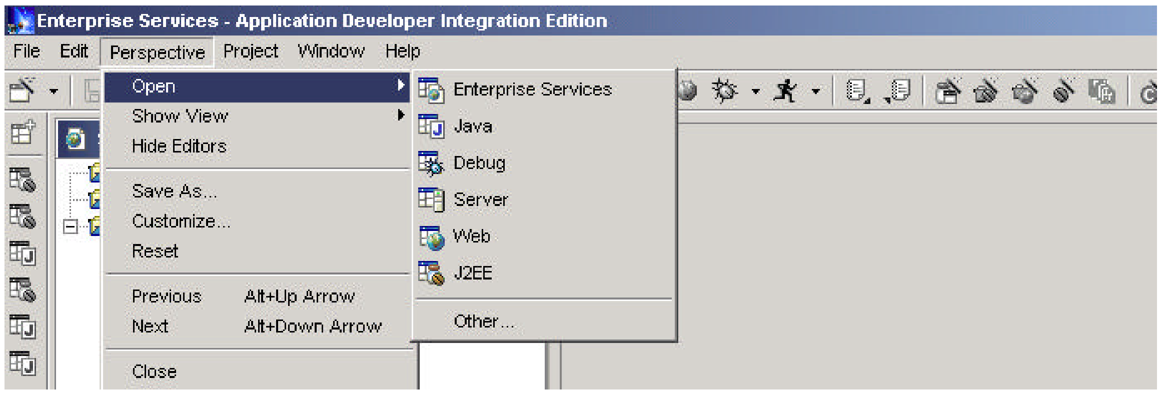
A perspective is the way a developer sees the projects. A perspective is a set of views, arranged into the Workbench window, and a set of editors and tools that are used to manipulate the resources.Perspectives are tailored for certain tasks, based on the role of the developer. For example, in the Java perspective you can compile Java code; in the Web perspective, you can edit and customize Web applications; in the J2EE perspective, you develop J2EEhierarchies and EJBs.

As we go through the remainder of the workshop, you will be told which type of project to create and which perspective to open.
****

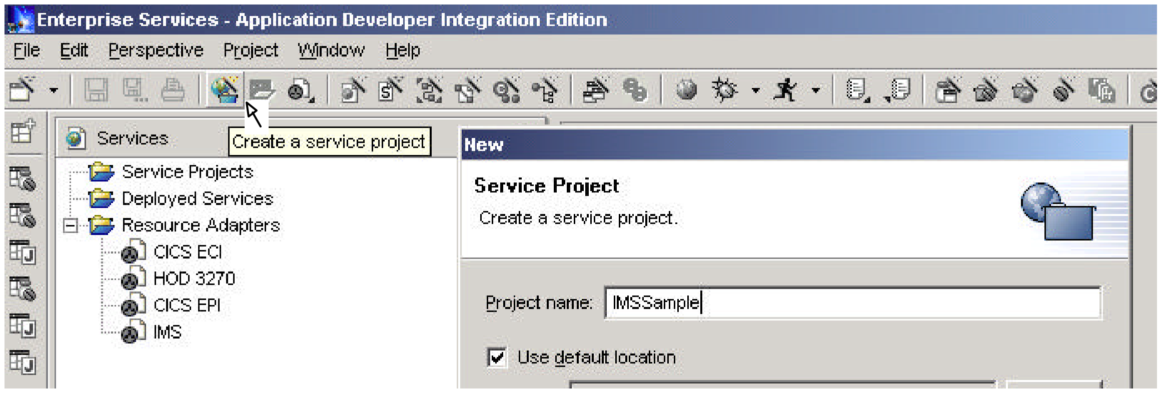
In this exercise, we will create an IMS service that will be contained in a Java project. All the resources required by this service, e.g., the COBOL copybook and service settings will be kept in this project.

To create a project for the service, follow these steps
:

1. Open the Enterprise Services perspective. Go to the tool bar at the top of the screen and click on "Perspective". In the pop-up window, choose **Open > Enterprise services.**



2. Launch the Service project wizard to create a project for your service. From the toolbar click the **Create a service project** icon. This opens the Service project wizard.



Service Project

Create a service project.

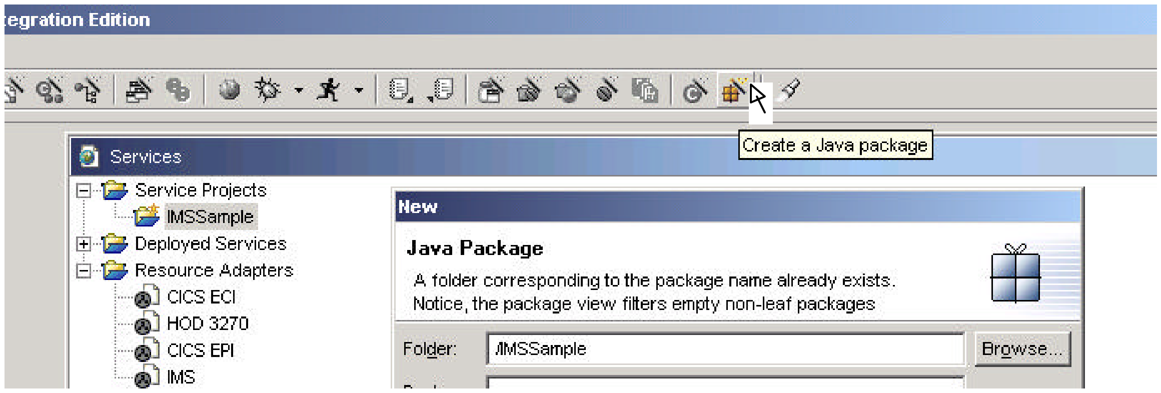
Project name: IMSSample

☑ Use default location

3. Type **IMSSample** for the name of the project. Use the default location to store the new project. Click **Finish** when done.

4b. Import a COBOL File

1. Create a **Java package** by expanding the **Service Projects** folder. Select **IMSSample** and click on the **Create a Java package** icon from the toolbar. On the Java Package page, make sure that **/IMSSample** is displayed as the name of the folder. **(A package is a component in a project that contains a group of files.)**



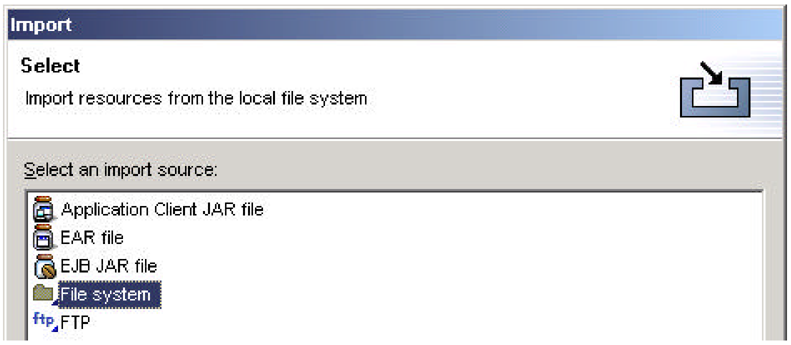
Java Package

A folder corresponding to the package name already exists.
Notice, the package view filters empty non-leaf packages

Folder: /IMSSample

2. Type **sample.ims** for the package name. Click Finish.

3. Go back to the Services perspective window. Expand the **IMSSample** project and select the **sample.ims** package that you just created. Go to the top of the screen and click **File** in the toolbar. In the pop-up window, select **Import** to open up the Import wizard.


Import

Select

Import resources from the local file system

Select an import source:

Application Client JAR file
EAR file
EJB JAR file
File system
FTP

At this point we are now ready to work with the COBOL definitions from the IMS transaction. A sample COBOL copybook file for the IMS IVP transaction - IVTNO - is shipped with the WSAD IE toolkit. What we need out of this file is the structure of the input and output messages that the IMS transaction needs. We will begin by importing the file into our WSAD IE workspace. The next set of steps show you how to access the file.

4. Select **File system** to import the resources from the local file system and then click **Next.**

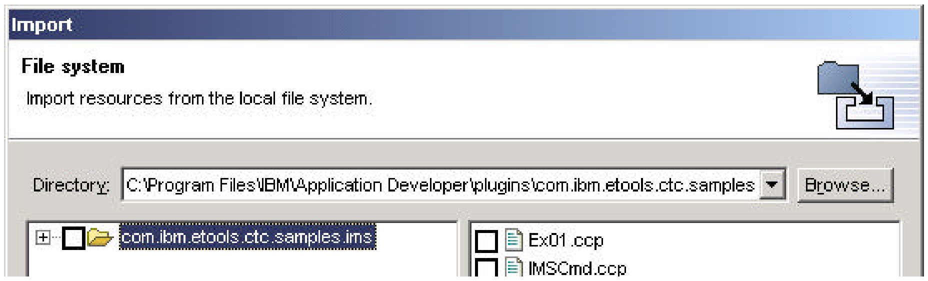
5. To access the directory, click **Browse** and select the following folder:

   C:\Program Files\IBM\Application Developer\plugins\com.ibm.etools.ctc.samples.ims

   Click **OK.**

6. On the **File System** page of the Import wizard, select the com.ibm.etools.ctc.samples.ims folder **without** selecting the check box

.


Import

File system

Import resources from the local file system.

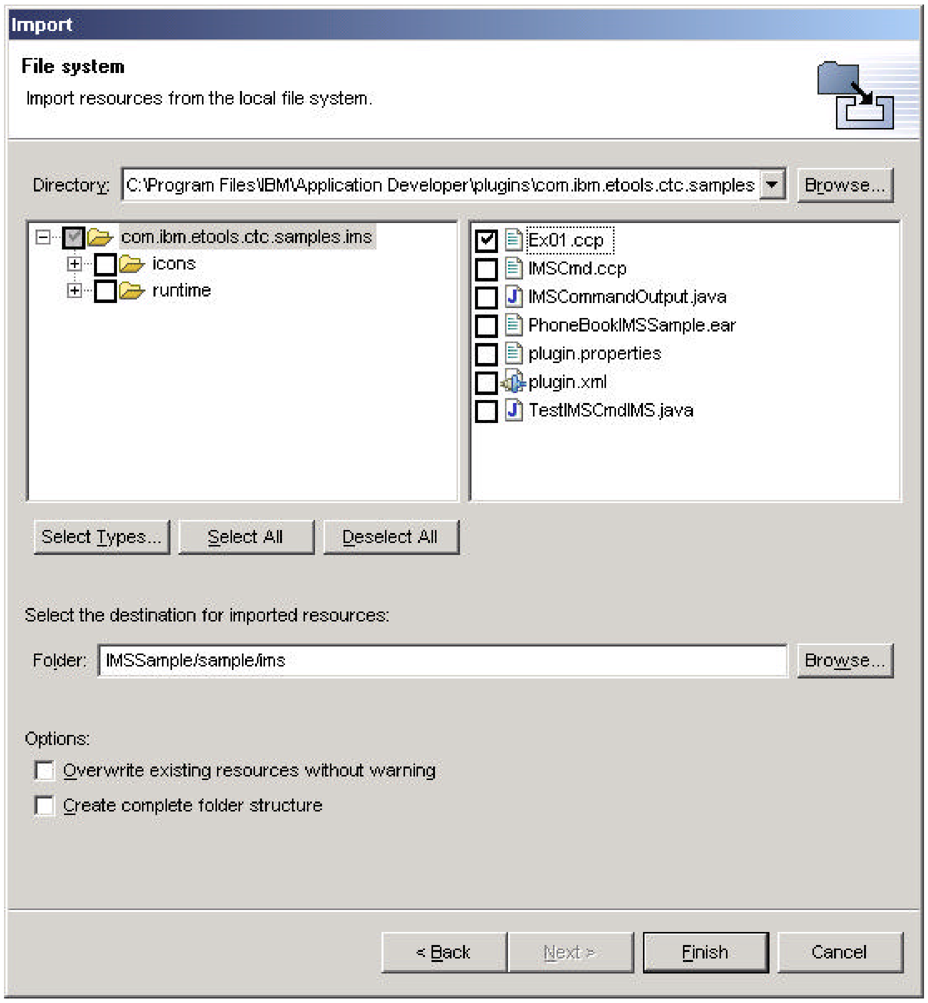
Directory: C:\Program Files\IBM\Application Developer\plugins\com.ibm.etools.ctc.samples

Browse...

com.ibm.etools.ctc.samples.ims

Ex01.ccp
IMSCmd.ccp

Select the **Ex01.ccp** check box to import the COBOL copybook file.


Import

File system

Import resources from the local file system.

Directory: C:\Program Files\IBM\Application Developer\plugins\com.ibm.etools.ctc.samples

- [✓] com.ibm.etools.ctc.samples.ims
  - [ ] icons
  - [ ] runtime

- [✓] Ex01.ccp
- [ ] IMSCmd.ccp
- [ ] IMSCommandOutput.java
- [ ] PhoneBookIMSSample.ear
- [ ] plugin.properties
- [ ] plugin.xml
- [ ] TestIMSCmdIMS.java

Select the destination for imported resources:

Folder: IMSSample/sample/ims

Options:

- [ ] Overwrite existing resources without warning
- [ ] Create complete folder structure

7. Make sure that **IMSSample/sample/ims** is the name of the destination folder for the imported resource. Click **Finish** to import the file and close the wizard. If you are successful in importing the file, the **Tasks** view (bottom panel of your screen) will not contain any (red) errors and the sample.ims package in the Packages view will contain the imported file.

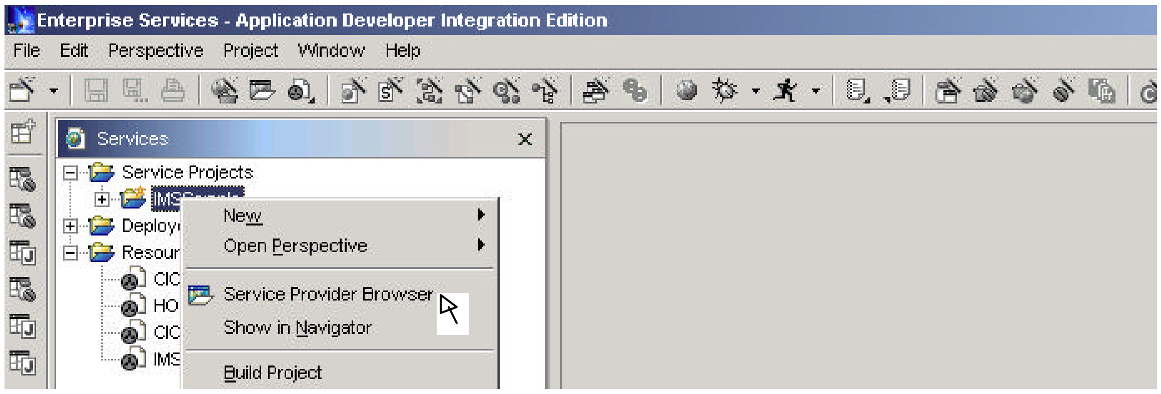
You are now ready to create an IMS service definition from that file.

4c. Create an IMS service definition

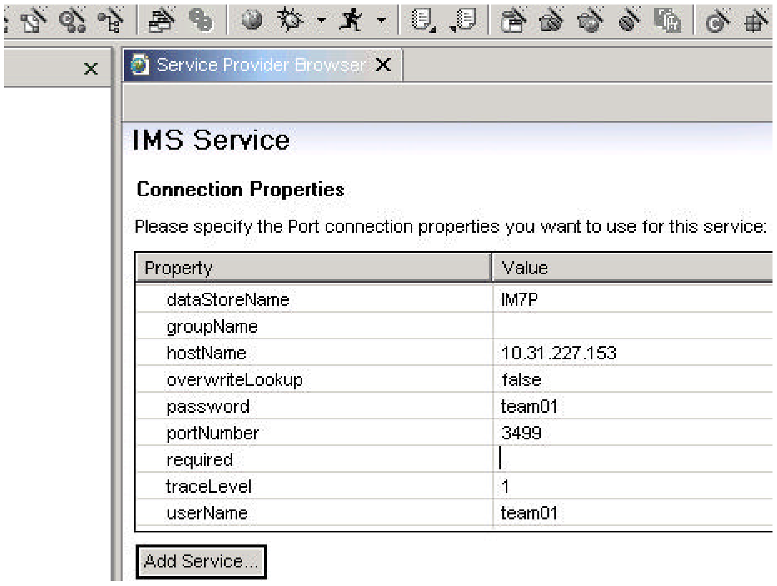
A service definition describes the basic format of system requests - where the service is deployed and what operations this service provides. It is described in the Web Services Description Language (WSDL) which is a standard for describing networked, SML-based services. WSDL provides a simple way to describe the basic format of system requests regardless of the underlying run-time implementation.

In this workshop, you will create an IMS service definition. It is also worth noting that WSAD IE uses the term EIS Service. EIS is the acronym for Enterprise Information System.

1. Select the **IMSSample** service project (single left mouse click to highlight) then right mouse click to bring up the selection window. Select the **Service provider browser** icon and click on **IMS Services.**



2. In the IMS Service page type the appropriate values as shown below. Make sure that **after** you enter the appropriate values, that you also click on the Value field for **required.** This is simply to move the cursor to a field other than the last one you entered to ensure that the wizard actually accepted the values you entered. When completed, click on **Add Service.**



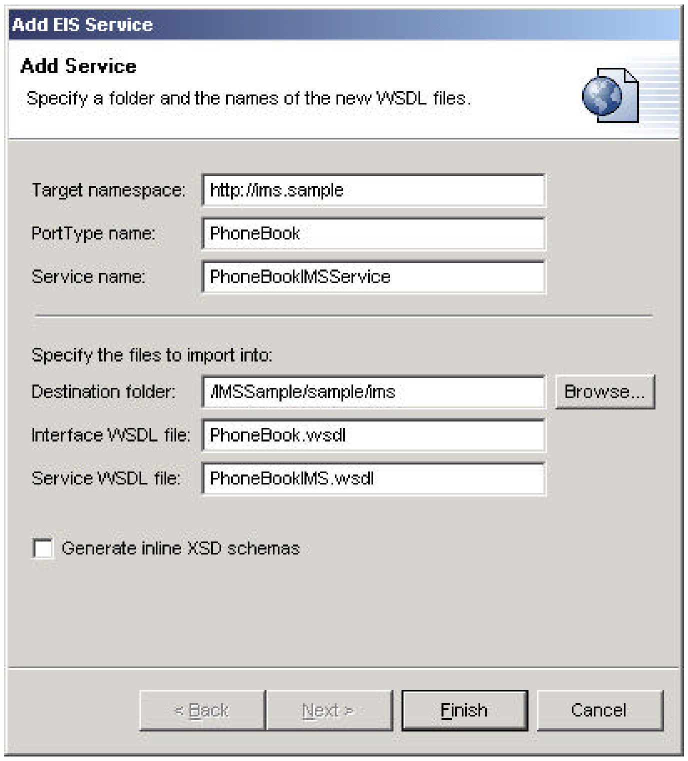
Please specify the Port connection properties you want to use for this service:

Property	Value
dataStoreName	IM7P
groupName	
hostName	10.31.227.153
overwriteLookup	false
password	team01
portNumber	3499
required	
traceLevel	1
userName	team01

3. In the **Add EIS Service** page which follows:

   a. Type **http://ims.sample** for the target namespace. This is a unique logical namespace for your service.

   b. Type **PhoneBook** for the name of the port type. This field represents the interface for retrieving the output from the IMS transaction using the service. Click **Finish** to accept all other default names. An editor opens with the Binding page. This is the Design view. To see the source view, click on the **Source** tab at the bottom of the Binding page. Now you can add the operations to the service.



Target namespace: http://ims.sample
PortType name: PhoneBook
Service name: PhoneBookIMSService

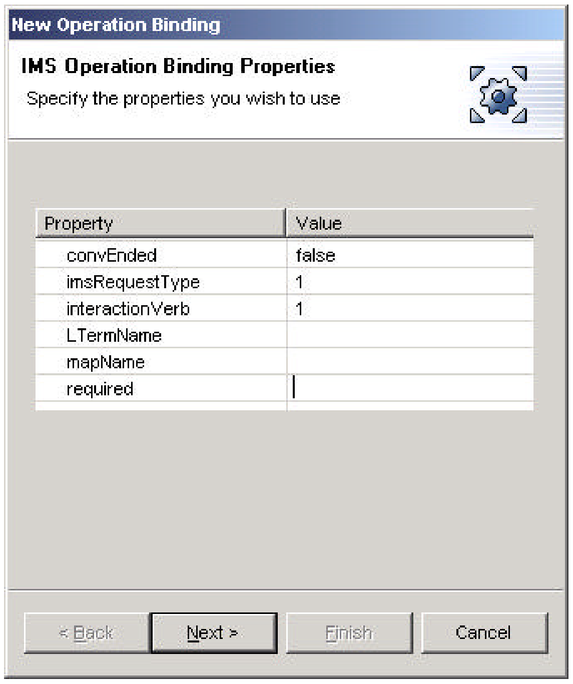
Specify the files to import into:
Destination folder: /IMSSample/sample/ims
Interface WSDL file: PhoneBook.wsdl
Service WSDL file: PhoneBookIMS.wsdl

☐ Generate inline XSD schemas

4. In the Design view, under **Operations**, click **New.**

5. In the IMS Operation Binding Properties page, ensure that the following are set:
   - **imsRequestType** field is set to 1. This indicates that the interaction with IMS consists of running a transaction.
   - **interactionVerb** field is set to 1. This indicates that the interaction with IMS involves a send (of the input message) followed by a receive (of the output reply).

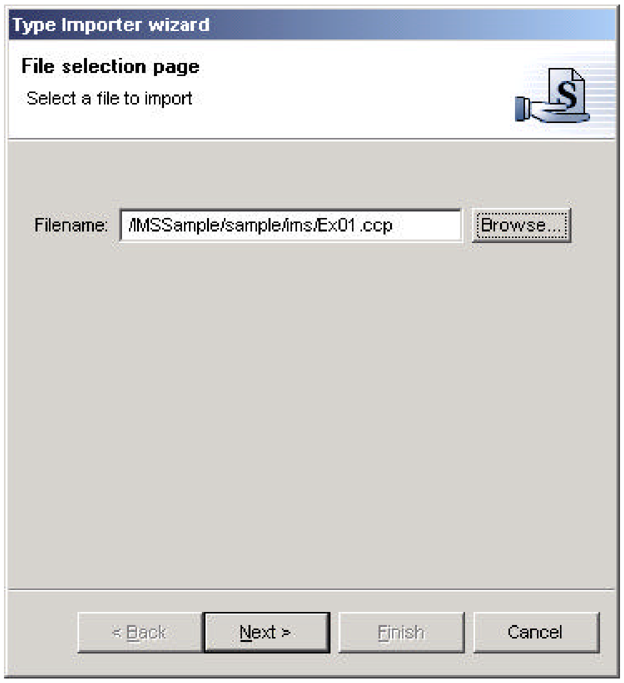
   Click on the Value field for **required.** This is simply to move the cursor to a field other than the last one you entered. When completed, click **Next.**


New Operation Binding

IMS Operation Binding Properties

Specify the properties you wish to use

Property	Value
convEnded	false
imsRequestType	1
interactionVerb	1
LTermName	
mapName	
required	

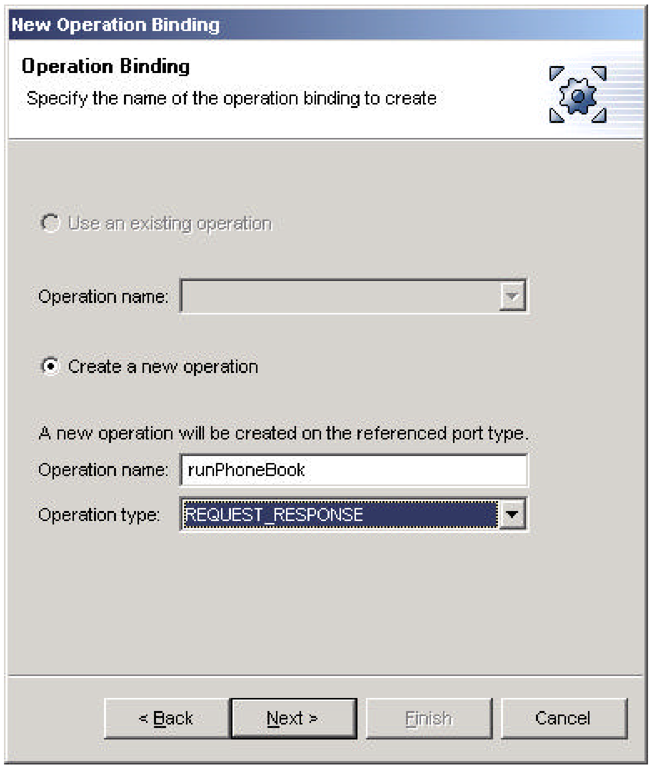

Type Importer wizard

File selection page

Select a file to import

Filename: /IMSSample/sample/ims/Ex01.ccp

6. The next page that comes up is the Operation Binding page. This is where you provide the information needed to create new input and output messages based on the existing input and output operations. Leave **Create a new operation** selected. In the **Operation name** field, type **runPhoneBook.** Leave the type of operation as REQUEST_RESPONSE because there will be two messages, one for the request to run the IMS transaction and one for the response from the IMS transaction. Click **Next.**


New Operation Binding

Operation Binding

Specify the name of the operation binding to create

Use an existing operation

Operation name:

[✓ Create a new operation]

A new operation will be created on the referenced port type.

Operation name: runPhoneBook

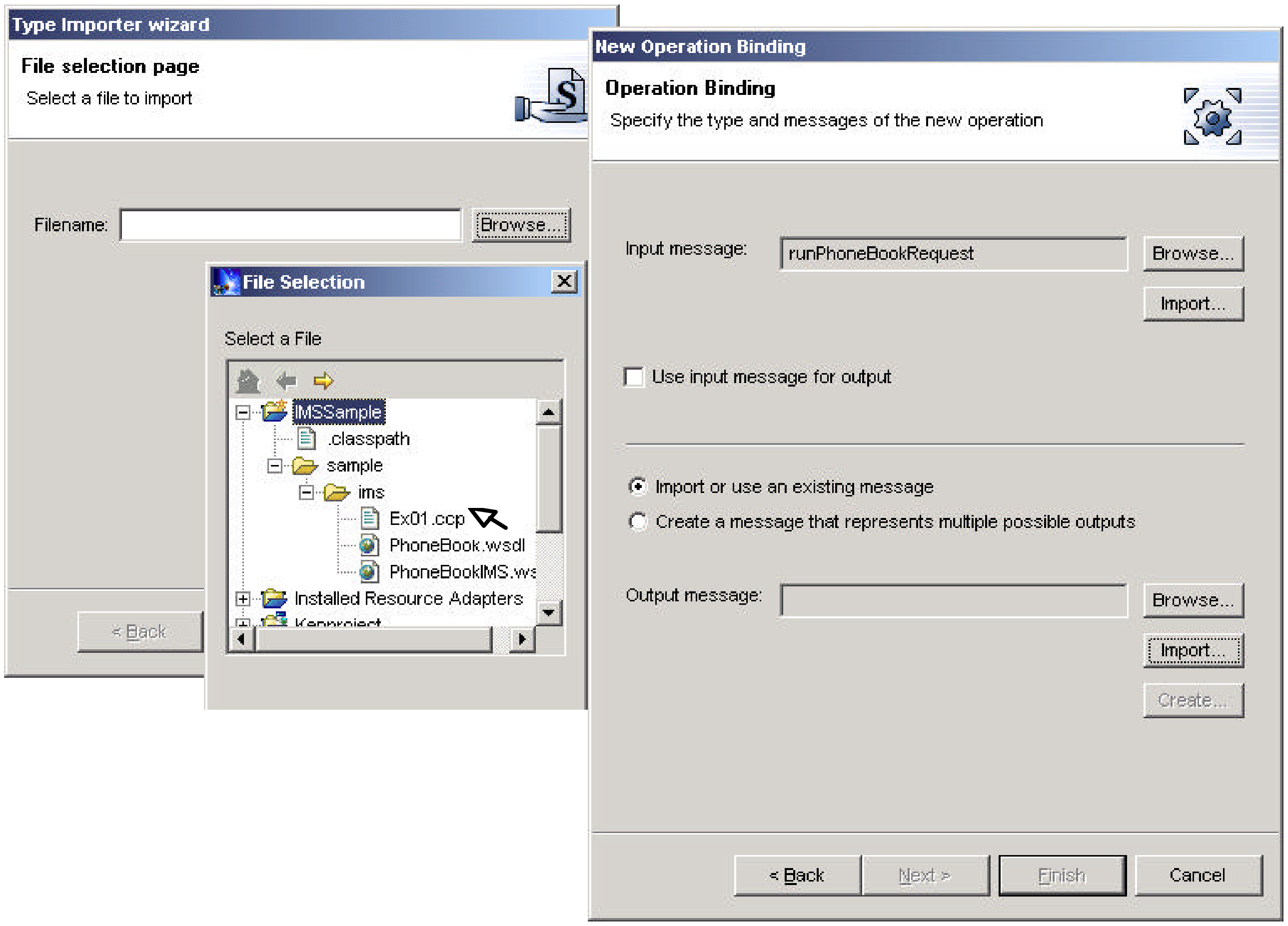
Operation type: REQUEST_RESPONSE

7. In the page that comes up, click the **Import** button next to **Input Message.** The Type Importer wizard opens.

   a. Click on **Browse** to find the filename.

   b. Import the Ex01.ccp file to specify the XML schema definition for the Input message. Click **OK.**

      Click **Next.**



Type Importer wizard

File selection page

Select a file to import

Filename:

File Selection

Select a File

IMSSample
.classpath
sample
ims
Ex01.ccp
PhoneBook.wsdl
PhoneBookIMS.ws
Installed Resource Adapters

New Operation Binding

Operation Binding

Specify the type and messages of the new operation

Input message: runPhoneBookRequest

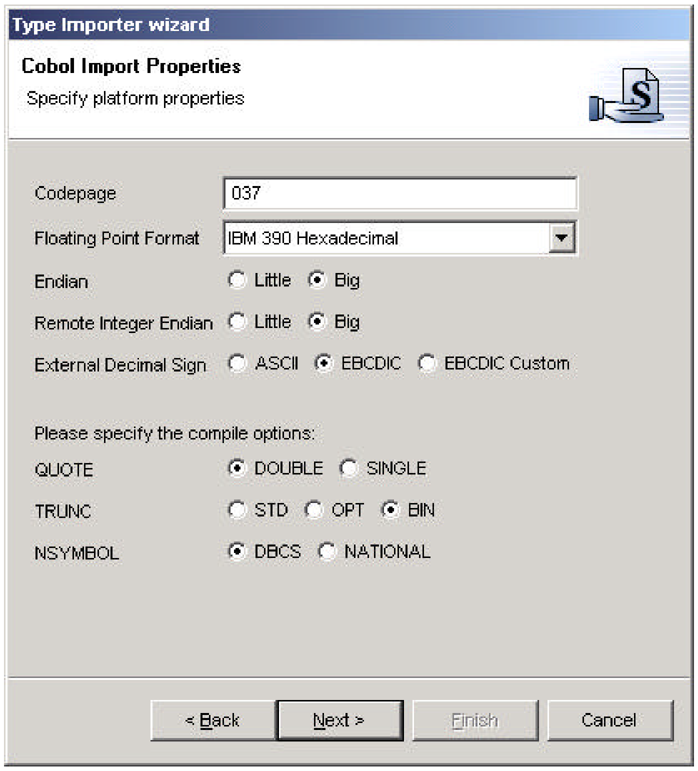
☐ Use input message for output

● Import or use an existing message

○ Create a message that represents multiple possible outputs

Output message:

c. In the Cobol Import Properties page, enter the following values and click **Next.**



Cobol Import Properties

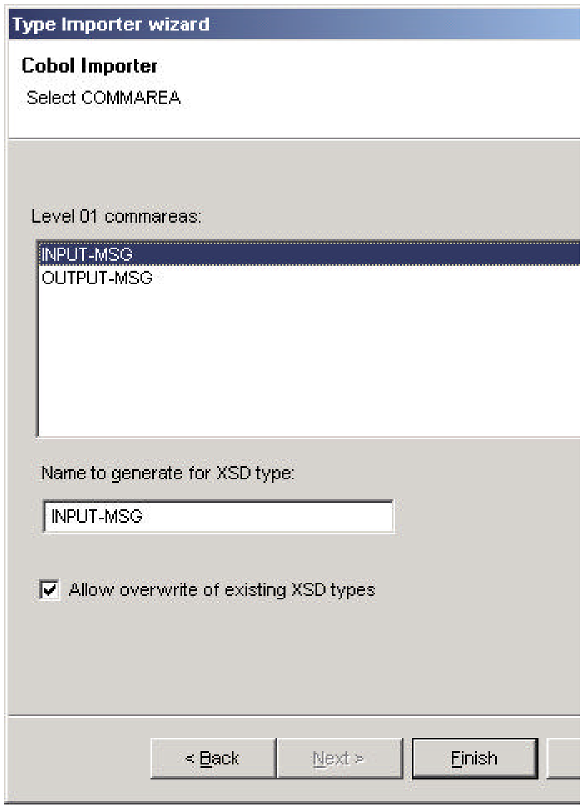
Specify platform properties

Codepage	037
Floating Point Format	IBM 390 Hexadecimal
Endian	Little [✓ Big]
Remote Integer Endian	Little [✓ Big]
External Decimal Sign	ASCII [✓ EBCDIC] EBCDIC Custom

Please specify the compile options:

QUOTE	[✓ DOUBLE] SINGLE
TRUNC	STD OPT [✓ BIN]
NSYMBOL	[✓ DBCS] NATIONAL

d. In the Cobol Importer page, the level 01 commareas from the file are displayed. Select INPUT-MSG which will populate the SXD type name in INPUT-MSG. You can accept the default to overwrite the XSD=types. Click **Finish.**



Cobol Importer

Select COMMAREA

Level 01 commareas:

[✓ INPUT-MSG]
OUTPUT-MSG

Name to generate for XSD type: INPUT-MSG

[✓] Allow overwrite of existing XSD types

This process provides the definitions necessary to support the runPhoneBook method. Invocation of this method will result in transaction input information being passed from the application to the input message and then to the EIS. For the corresponding transaction output to be returned from the EIS to the output message and then to the application we now have to process the information for the output message.

8. In the Operation Binding page, click on **Import** next to the Output message. The process you will follow is similar to that of processing the input message.

   a. Click **Browse** in the Type Importer wizard.

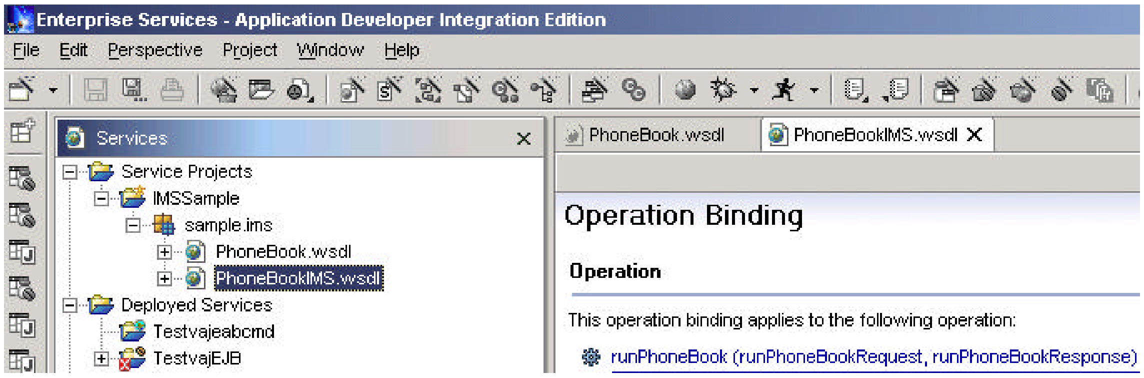
   b. In the File selection page, click **Browse** and then click **OK** to select the same Ex01.ccp from the sample.ims package. In this sample, both the input and output message definitions are contained in the same COBOL source file, Ex01.ccp. Click **Next.**

   d. In the Cobol Import Properties page, specify the same values that you entered above for input. Click **Next.**

   e. In the Cobol Import window, select OUTPUT-MSG in the **level 01 commareas** list which will populate the XSD type name with the OUTPUT-MSG. You can accept the default to overwrite the XSD types. Click **Finish** to return to the Operation binding page.

9. Click **Finish.**

10.) Close the editor by clicking the **X** next to PhoneBook.wsdl then click **Yes** to save the changes. Note that you must save the changes in order to successfully proceed with the sample. Also close the Service Provider page in the editor. The service interface file PhoneBook.wsdl and the service binding file PhoneBookIMS.wsdl are updated in the sample.ims package of the IMSSample project.



This operation binding applies to the following operation:

runPhoneBook (runPhoneBookRequest, runPhoneBookResponse)

The binding file defines the <service> part of a WSDL description and then imports the service interface file. The interface file contains the <message>, <portType>, and <binding> parts of a WSDL description that describes the interface to the Web service. Messages describe input and output requests and responses to the service while port types are the collection of operations a service supports. Bindings are the implementation of the service such as an IMS binding to manage communication requests/responses between the service and IMS.

Now you are ready to create a proxy to directly access the service you just created.

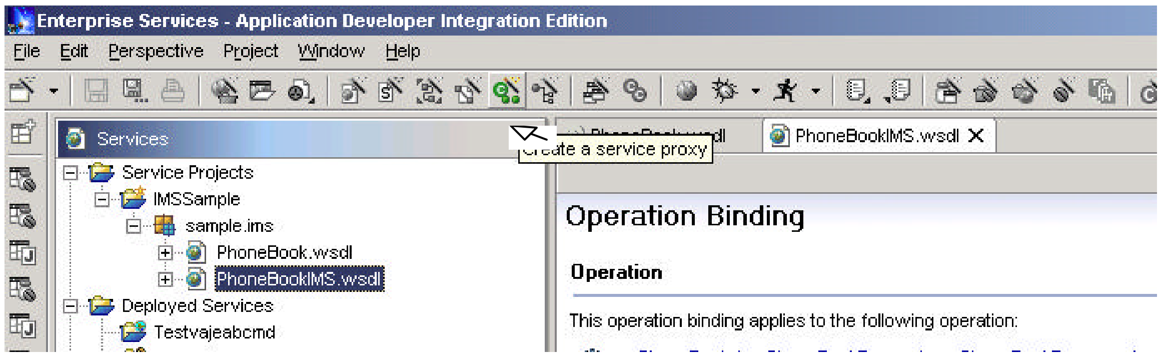
Exercise 5. Create an IMS Proxy and use this to test the IMS Service.

The proxy that you will now create provides a remote procedure call interface to the service. Using this proxy, the application will call a remote method on the service. Once the remote call is made, the proxy handles all of the communication details between the application and the IMS service.

5a. Create the IMS Proxy

The following steps will lead you through the creation of a proxy:

1. Expand the IMSSample project to select the service binding file PhoneBookIMS.wsdl.
2. From the toolbar, click the **Create a service proxy** icon**.** The Proxy wizard opens.



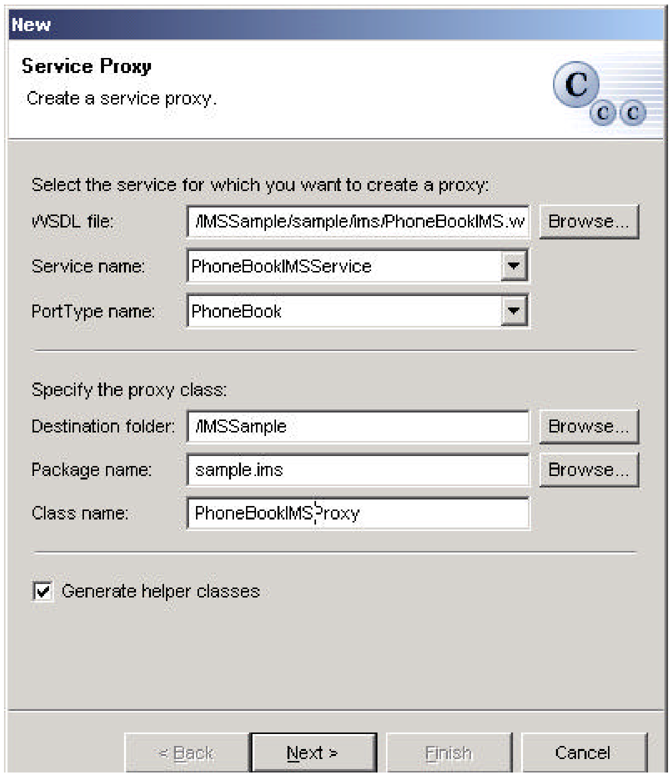
This operation binding applies to the following operation:

3. In the Service Proxy page, check to see if the service for which you want to create a proxy is shown and the proxy class properties are correct. Because you selected a file in the first step, most fields should be populated with default values. These defaults are generated based on the contents of the selected service binding file.

   a. Change the class name to **PhoneBookIMSProxy**

   b. Ensure the package name is **sample.ims**

   c. **Generate helper classes** is selected by default. Your service will need these Java helper classes since you are creating a service that includes complex types. Click **Next**.



Create a service proxy.

Select the service for which you want to create a proxy:

WSDL file: /IMSSample/sample/ims/PhoneBookIMS.w

Service name: PhoneBookIMSService

PortType name: PhoneBook

Specify the proxy class:

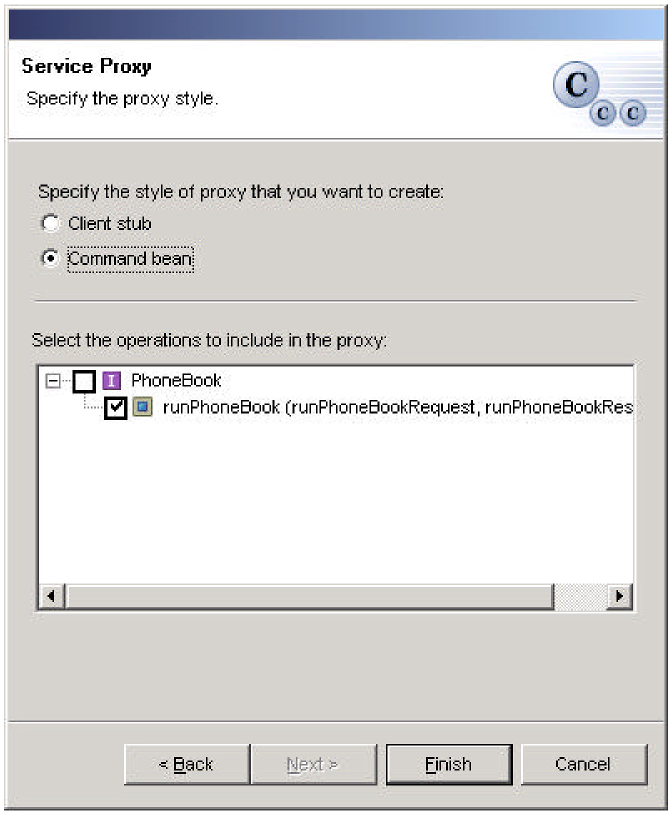
Destination folder: /IMSSample

Package name: sample.ims

Class name: PhoneBookIMSProxy

[✓] Generate helper classes

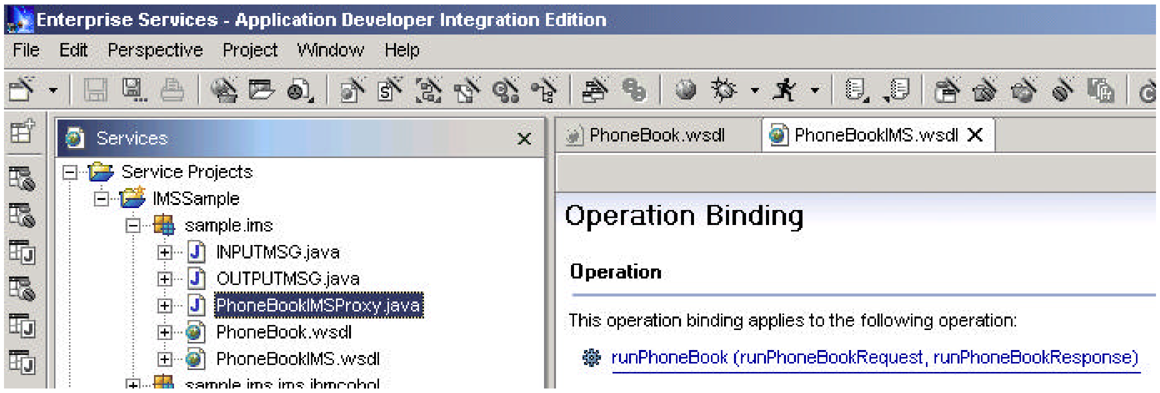
4. In the Service Proxy Style page, specify the style of the proxy and the operations to expose in the proxy:
   a. You can select the **Command bean** proxy type because the service binding file contains one method.
   b. Select the **runPhoneBook** method.


Service Proxy
Specify the proxy style.

Specify the style of proxy that you want to create:
Client stub
[✓ Command bean]

Select the operations to include in the proxy:
PhoneBook
[✓] runPhoneBook (runPhoneBookRequest, runPhoneBookRes

5. Click **Finish**. This will generate the IMS proxy (PhoneBookIMSProxy) in the IMSSample project.



Service Projects
IMSSample
sample.ims
INPUTMSG.java
OUTPUTMSG.java
PhoneBookIMSProxy.java
PhoneBook.wsdl
PhoneBookIMS.wsdl
sample.ims.ims.ibmcobol

Operation Binding

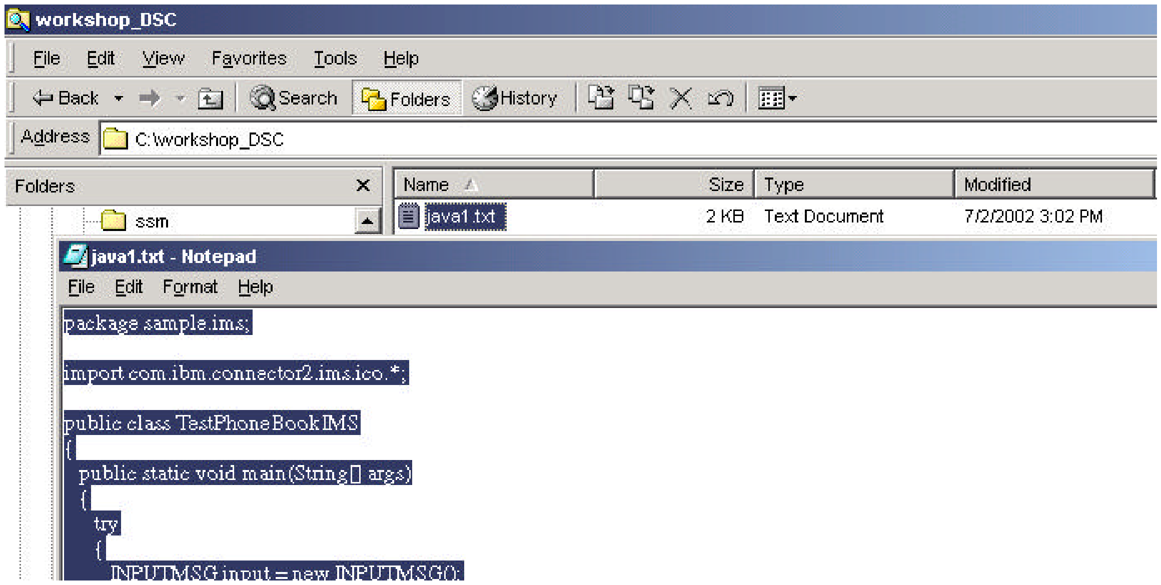
Operation

This operation binding applies to the following operation:
runPhoneBook (runPhoneBookRequest, runPhoneBookResponse)

5b. Write a Java class to test the IMS Proxy

You will now create a java class to execute the proxy. This section will step you through the process. You will prepare a client application that will bind the client to the implementation of the service and then invoke the service. The code will be stored in the service project you created named IMSSample.

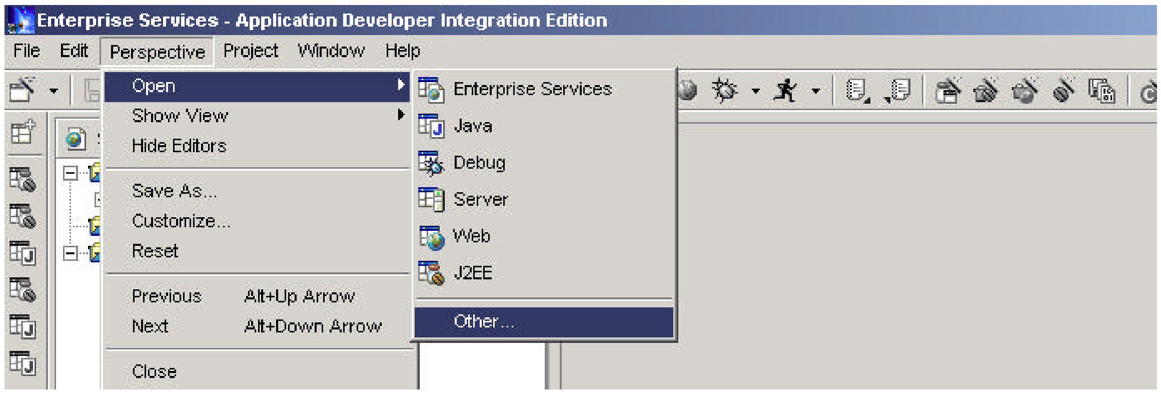
1. You will be adding pre-written code into a new class you will create. Find this code in one of two ways:
   - The code is readily available on your PC in C:\workshop_DSC\java1.txt.

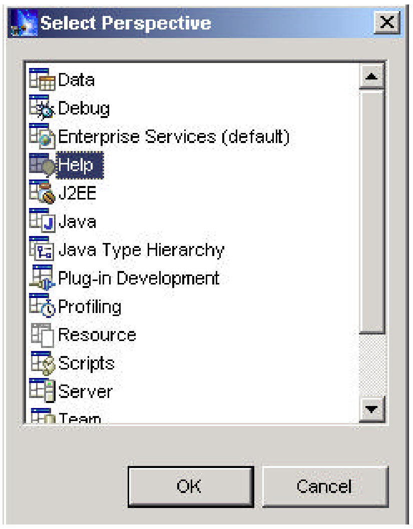


```
package sample.ims;

import com.ibm.connector2.ims.ico.*;

public class TestPhoneBookIMS
{
  public static void main(String[] args)
  {
    try
    {
      INPUTMSG input = new INPUTMSG();
```

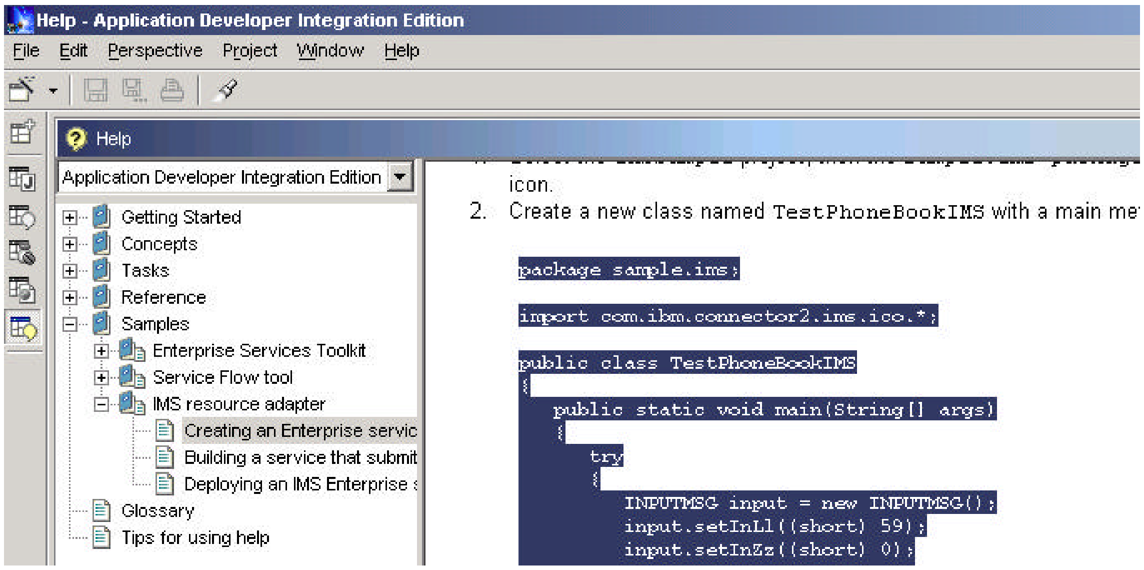
OR

   - It is also available in the Help section of the WSAD IE tool. This is where you would get it in your own environment (outside this workshop). To do so go to the top of the screen and in your toolbar find Perspective => Open => Other, then click on Help





Click **OK.** This opens up the Help perspective. Open up the Samples folder => IMS Resource adapter folder => Creating an Enterprise Service for an IMS Transaction. On the right pane, scroll down until you get to the sample code as shown below.

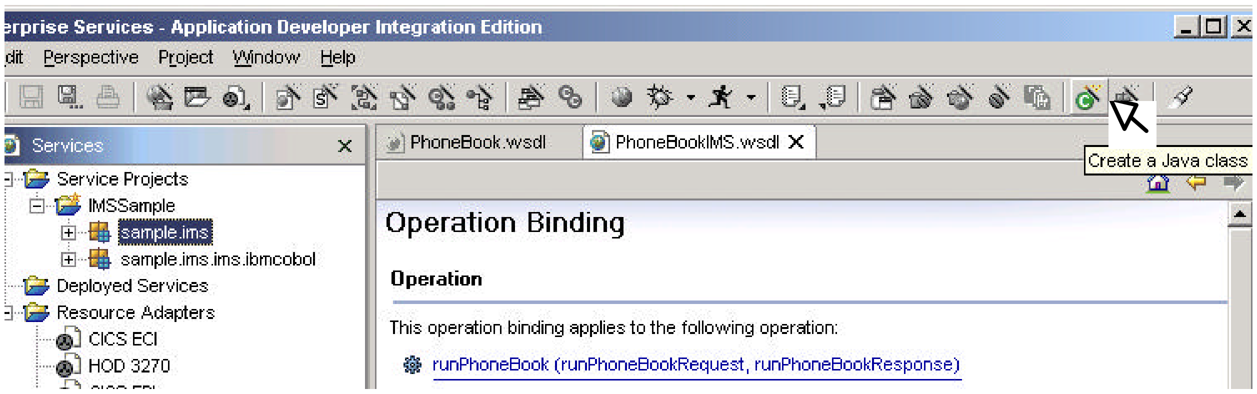

Help - Application Developer Integration Edition
File
Edit
Perspective
Project
Window
Help
Help
Application Developer Integration Edition
Getting Started
Concepts
Tasks
Reference
Samples
Enterprise Services Toolkit
Service Flow tool
IMS resource adapter
Creating an Enterprise servic
Building a service that submit
Deploying an IMS Enterprise s
Glossary
Tips for using help

icon.

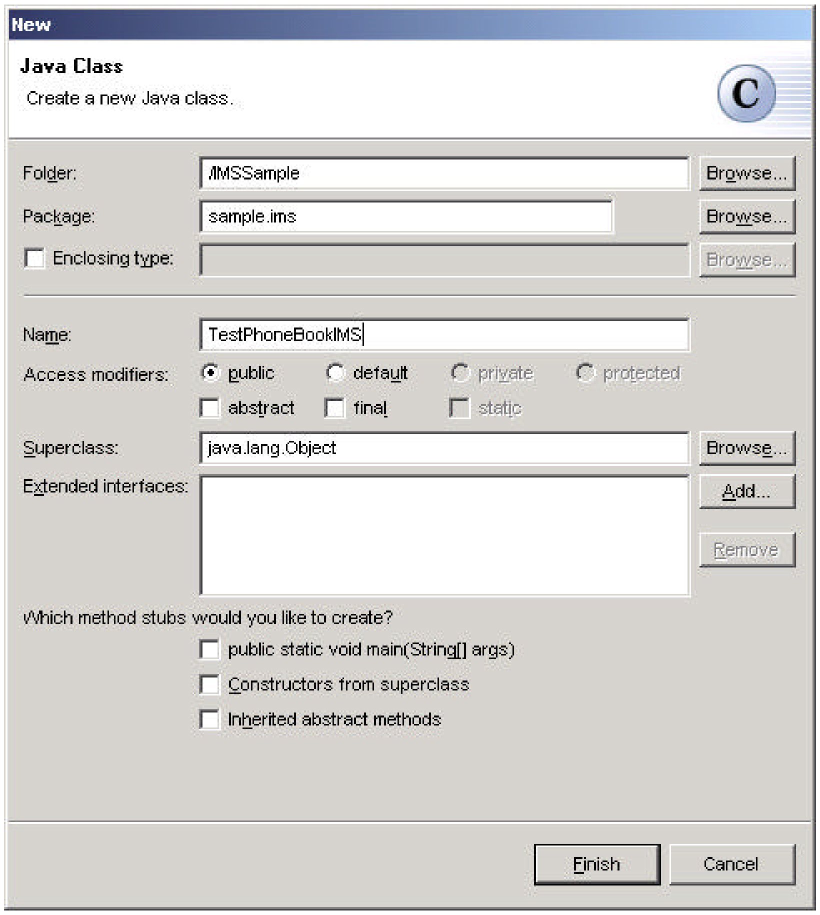
2. Create a new class named `TestPhoneBookIMS` with a main me

```
package sample.ims;

import com.ibm.connector2.ims.ico.*;

public class TestPhoneBookIMS
{
    public static void main(String[] args)
    {
        try
        {
            INPUTMSG input = new INPUTMSG();
            input.setInLl((short) 59);
            input.setInZz((short) 0);
```

2. You will now create the class and copy the above code into it. Select the IMSSample project, then the sample.ims package. From the toolbar, click the **Create a Java class** icon.


erprise Services - Application Developer Integration Edition
dit Perspective Project Window Help
Services
Service Projects
IMSSample
sample.ims
sample.ims.ims.ibmcobol
Deployed Services
Resource Adapters
CICS ECI
HOD 3270
PhoneBook.wsdl
PhoneBookIMS.wsdl
Create a Java class
Operation Binding
Operation
This operation binding applies to the following operation:
runPhoneBook (runPhoneBookRequest, runPhoneBookResponse)

Create a new class named **TestPhoneBookIMS.**


New

Java Class

Create a new Java class.

Folder: /IMSSample

Package: sample.ims

Enclosing type: [ ]

Name: TestPhoneBookIMS

Access modifiers: [✓ public] default private protected; [ ] abstract [ ] final [ ] static

Superclass: java.lang.Object

Extended interfaces:

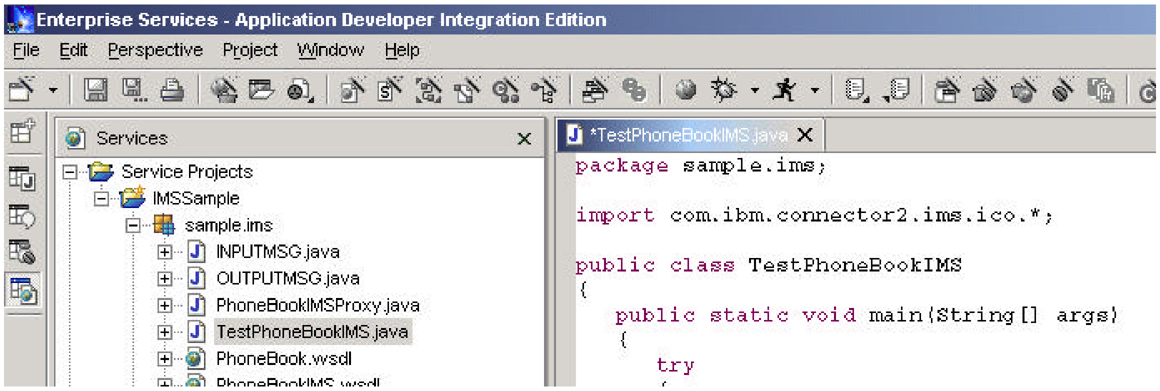
Which method stubs would you like to create?

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

[ ] Constructors from superclass

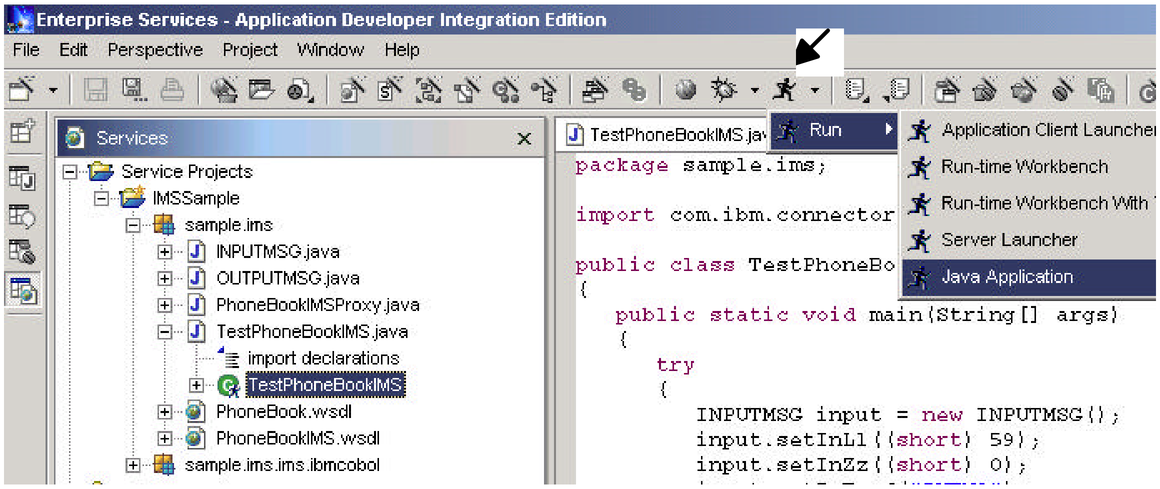
[ ] Inherited abstract methods

Replace the skeleton code with the code that you retrieved in the previous step.



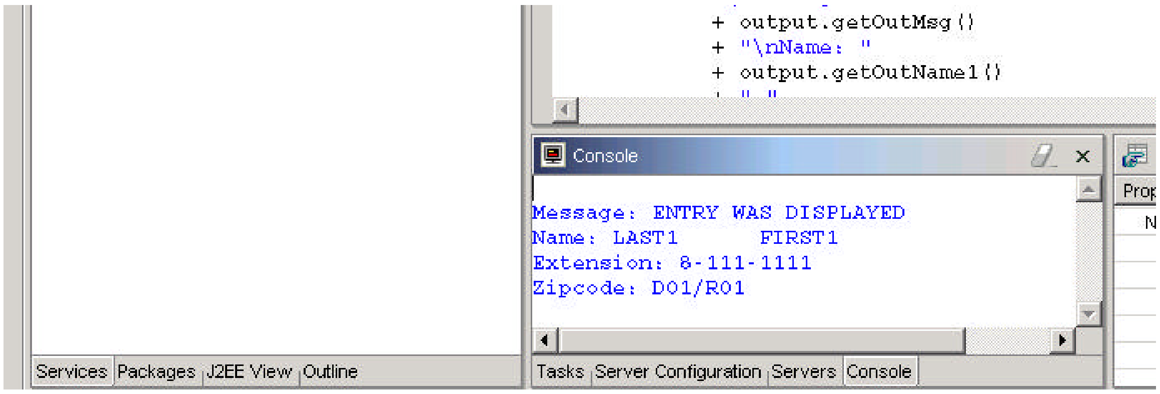
```
package sample.ims;

import com.ibm.connector2.ims.ico.*;

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

3. Close the editor by clicking on the **X** and click **Yes** to save the changes.
4. Select the TestPhoneBookIMS class and expand the **Run** icon on the toolbar by selecting the arrow beside it. From the pop-up menu, select **Run => Java Application.**



```
package sample.ims;

import com.ibm.connector

public class TestPhoneBo
{
   public static void main(String[] args)
   {
      try
      {
         INPUTMSG input = new INPUTMSG();
         input.setInLl((short) 59);
         input.setInZz((short) 0);
```

5. You should get a clean run without exceptions and see the following on the console window at the bottom of your screen.


Console
Services Packages J2EE View Outline
Tasks Server Configuration Servers Console
Prop

```
+ output.getOutMsg()
+ "\nName: "
+ output.getOutName1()
```

```
Message: ENTRY WAS DISPLAYED
Name: LAST1       FIRST1
Extension: 8-111-1111
Zipcode: D01/R01
```

If you do not get this message, then retrace your steps to see where you might have made a mistake.

Exercise 6. Deploy the IMS Service

You can deploy the service you have just created to the Web sphere Application Serve as an EJB service and as a SOAP service. We will deploy as a SOAP service in this workshop because it is evolving as a more popular mechanism.

****
Before we start, a little bit of information about SOAP.

> **SOAP is a protocol that leverages HTTP as its transport layer and XML as its data layer, to execute remote methods, known as SOAP services.**
>
> **One of the methods that SOAP provides for invoking services is called a Remote Procedure Call (RPC). The RPC method is a synchronous technique which uses a client-server model to execute remote SOAP services. The RPC model can be defined using the following steps:**

****

We will start out by creating a session bean and deploying it to a test server that is running as part of the WSAD IE toolkit. The IMS service that you created earlier will rely on this session bean to access the phone book data in the target IMS system. You will also create and use a client proxy which is just a simplified client interface to the session bean.

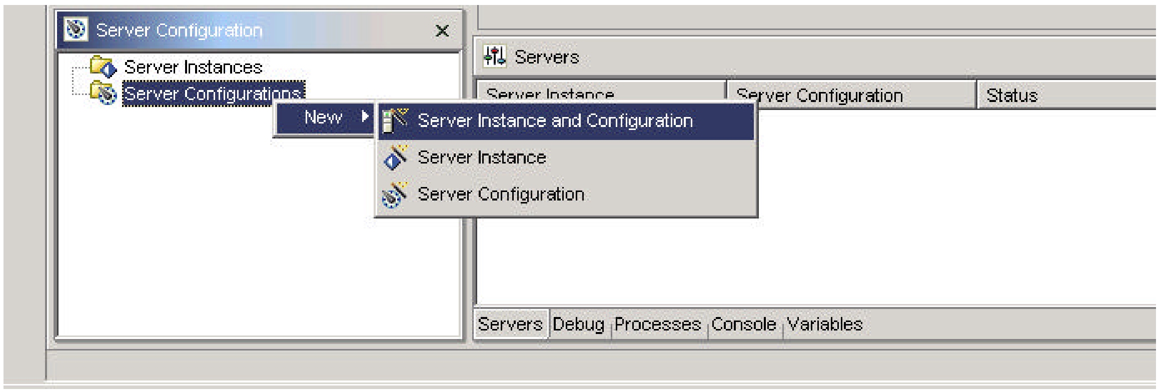
To deploy the service, we have to go through the following steps:

6a. Configure the server
6b. Create an EJB project and an EAR project
6c. Create a Web project
6d. Add a Connection Factory to the server configuration
6e. Create and deploy a session bean

6a. Configure the Server

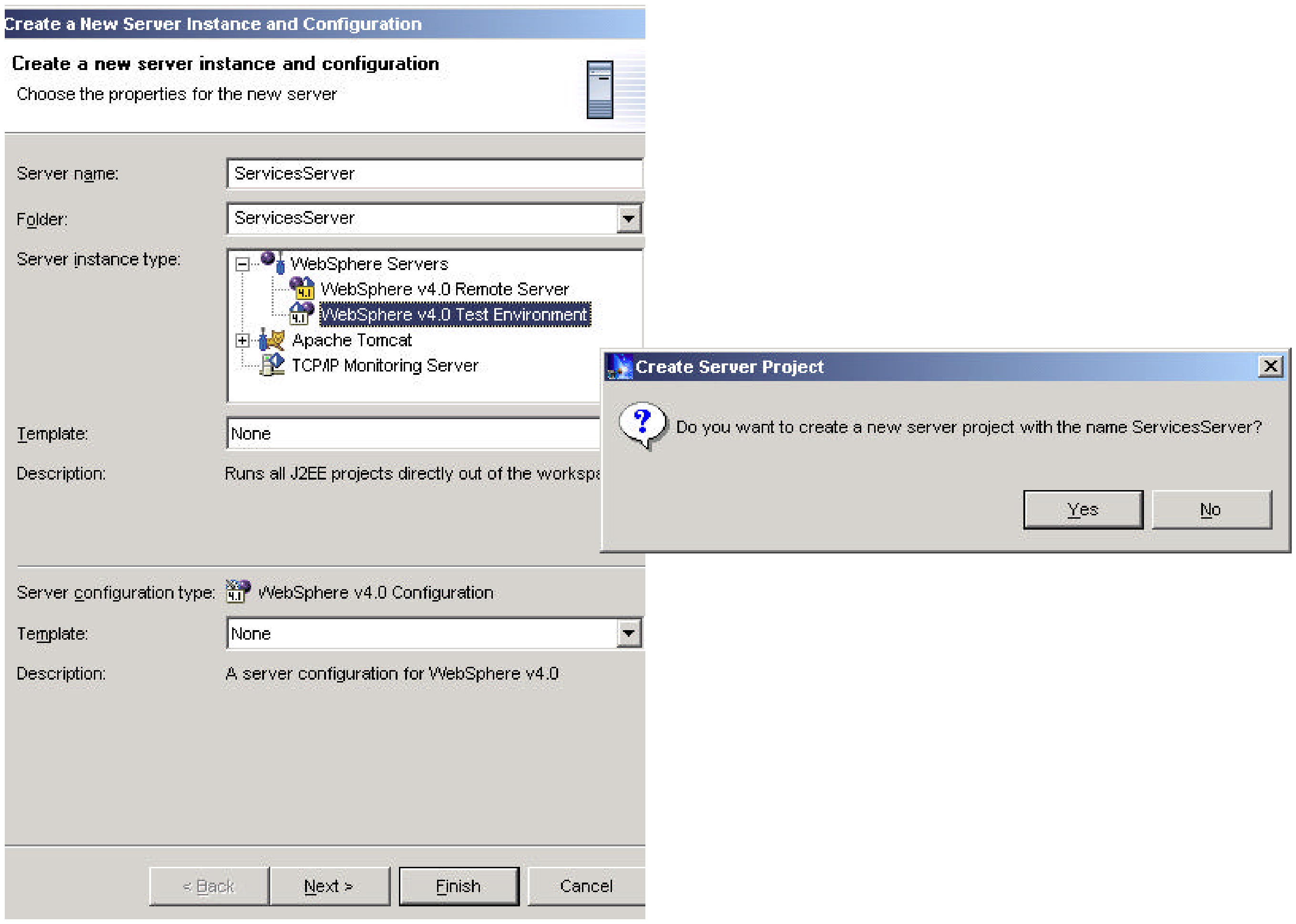
We will create an instance of a test server that runs in WSAD IE and configure it using these steps:

1. Open up a server perspective. This is done by going to the top of your panel, clicking on **Perspective => Open => Other** and then selecting **Server** in the Select Perspective panel.
2. In the **Server Configuration** window, highlight **Server Configurations**, right mouse click and then select **New > Server Instance and Configuration**. The Create a New Server Instance and Configuration wizard opens.



3. Type **ServicesServer** for both the server name and the folder name.
4. Expand **Websphere Servers** from the **Server instance type** list-box.
5. Select **Websphere v4.0 Test Environment**. Leave the template set to **None**. Click **Next**.

6. Click **Yes** to create a ServicesServer server project.



Create a New Server Instance and Configuration

Create a new server instance and configuration

Choose the properties for the new server

Server name: ServicesServer

Folder: ServicesServer

Server instance type:
- WebSphere Servers
  - WebSphere v4.0 Remote Server
  - WebSphere v4.0 Test Environment
- Apache Tomcat
- TCP/IP Monitoring Server

Template: None

Description: Runs all J2EE projects directly out of the workspa

Server configuration type: WebSphere v4.0 Configuration

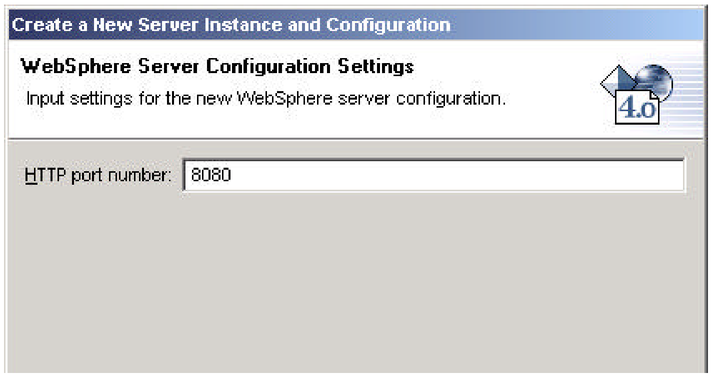
Template: None

Description: A server configuration for WebSphere v4.0

Create Server Project

Do you want to create a new server project with the name ServicesServer?

The server port number defaults to **8080**. The port identifies the location of the service. Click **Finish**. The new server instance appears in the Server Configuration view and in the Servers view. The server configuration appears in the Server Configuration view.



Create a New Server Instance and Configuration

WebSphere Server Configuration Settings

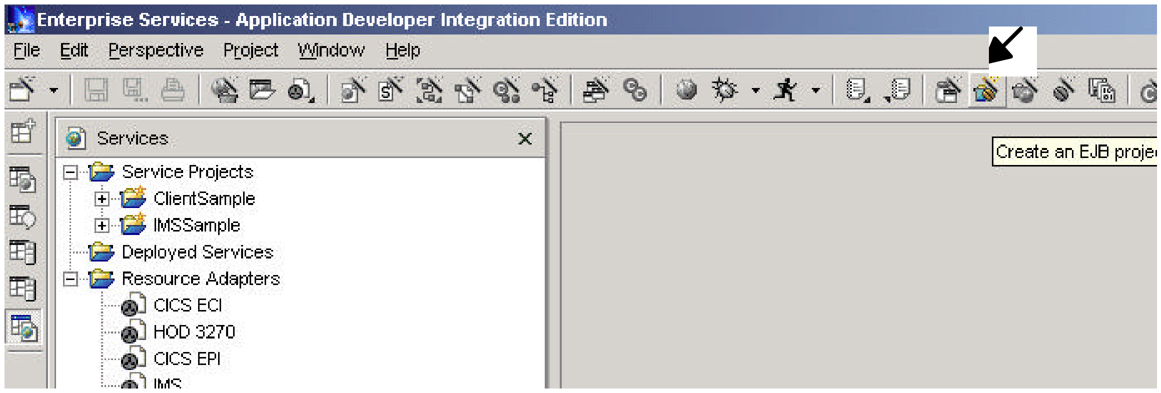
Input settings for the new WebSphere server configuration.

HTTP port number: 8080

You have just created an instance of the WebSphere Application Server that is emulated by the Websphere Test Environment running on your local host on port 8080. This now allows you to use this environment to locally deploy and test the service. If it works in this environment, it can later be deployed to a production server.

6b. Create an EJB Project and an EAR project

1. Select the Create an EJB project icon from the toolbar. The EJB Project wizard opens.

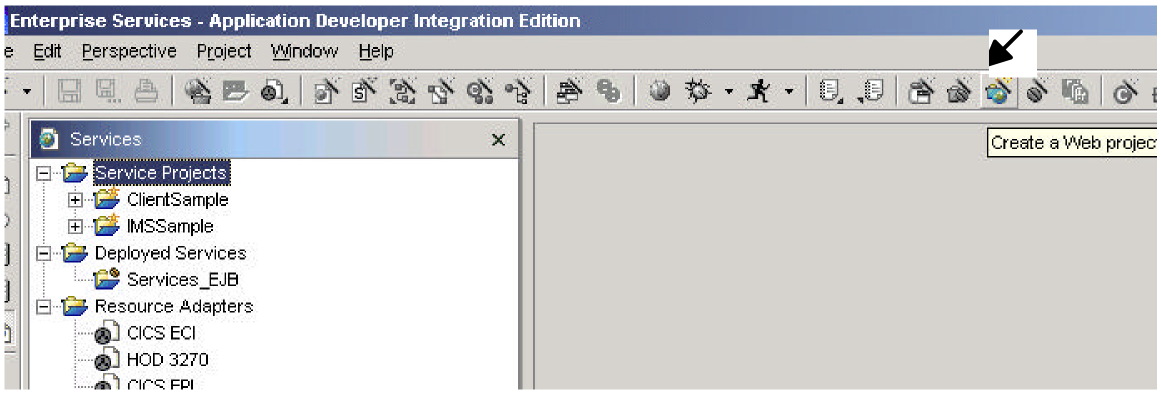


2. Type Services_EJB for the name of the EJB project.
3. Type ServicesEAR for the name of the Enterprise application project.
4. Click Finish. An Enterprise application project named ServicesEAR and an EJB module project named Services_EJB are created. (If an error symbol appears beside the EJB project, it will probably automatically disappear as you progress in your EJB development activities. So ignore it for now.)

6c. Create a Web project

The next step is to create a Web project.

1. Click the Create a Web Project icon from the toolbar. The Create a Web Project wizard opens.



2. Type Services_SOAP for the name of the Web project.
3. Ensure that the Enterprise application project name is ServicesEAR. It is not necessary to change the root context, to create a CSS file, or to set any module dependencies or Java$^{(TM)}$ build settings. Click **Finish**.

The Web module that was just created contains all the resources required to create a Web application. (Again, if an error symbol appears beside Services_EJB, it will probably automatically disappear as you progress through the remainder of the sample.)

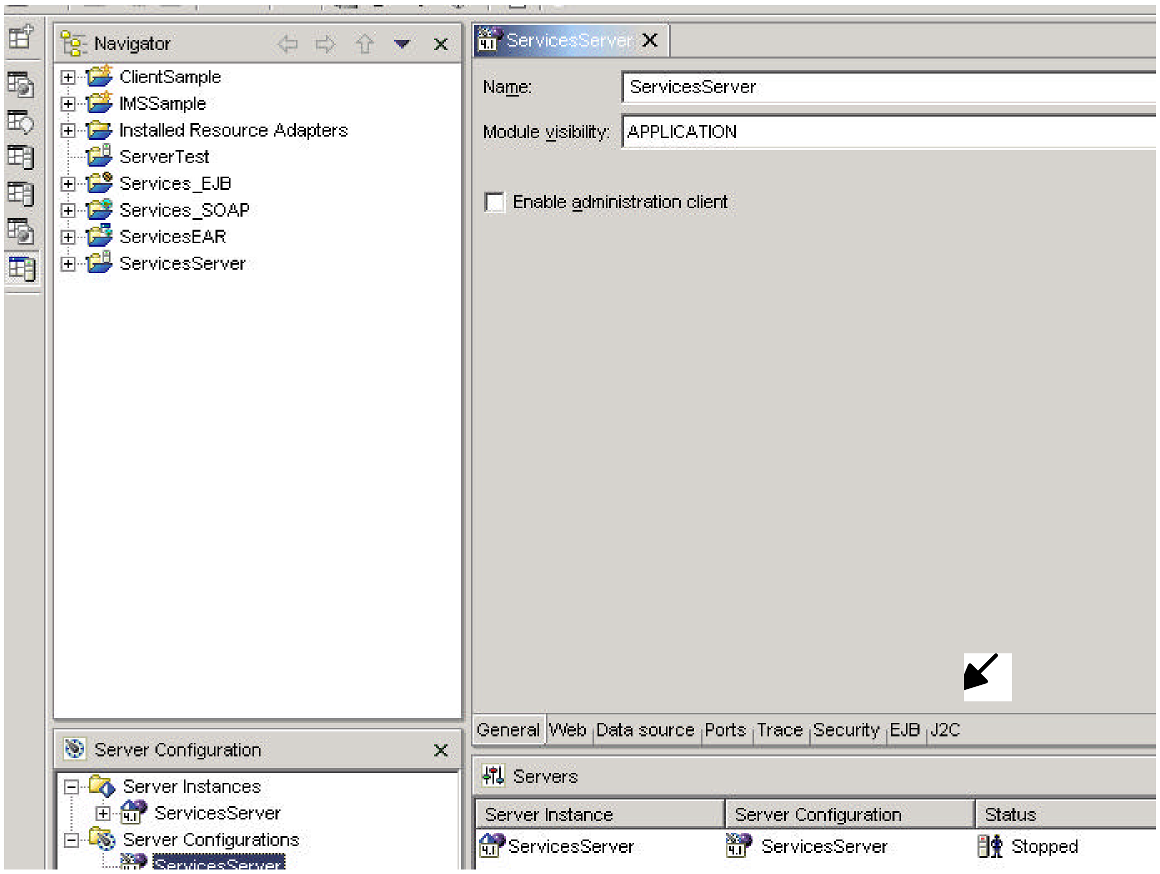
6d. Add a Connection Factory to the Server Configuration

You will now add an instance of the JCA connection factory and configure its properties.

The connection factory, as its name implies, provides connections to the EIS on demand. You specify all the information needed by the resource adapter to connect to the particular instance of the EIS. For the IMS resource adapter, you must specify at least the HostName, PortNumber and DataStoreName properties. These values will be used to determine the target IMS that will be accessed through all the connections created by this instance of the connection factory. You will also specify the JNDI lookup name under which the new connection factory instance will be made available to components. With this lookup name, the components can quickly make a connection to the EIS.

To add a connection factory, follow these steps:

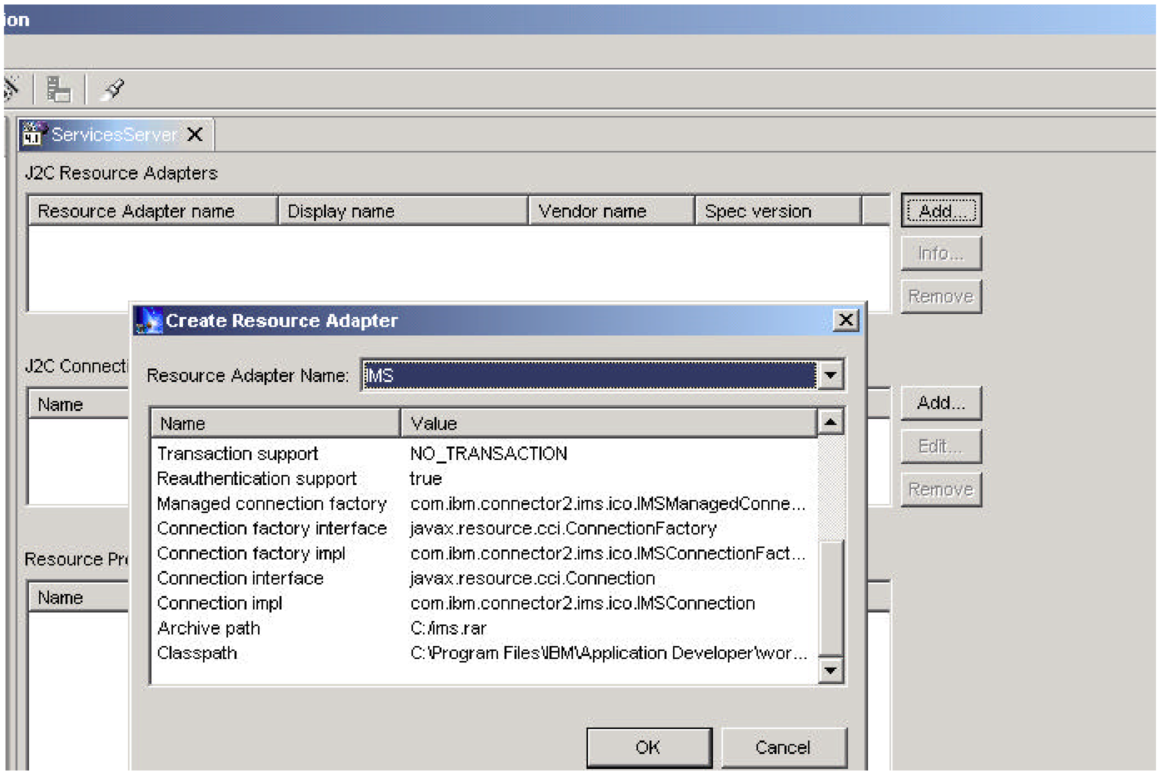
1. Open up the Server perspective. Expand **Server Configurations** in the Server Configuration view.
2. Double-click the server configuration **ServicesServer**. An editor opens.



Name: ServicesServer
Module visibility: APPLICATION

Server Instance	Server Configuration	Status
ServicesServer	ServicesServer	Stopped

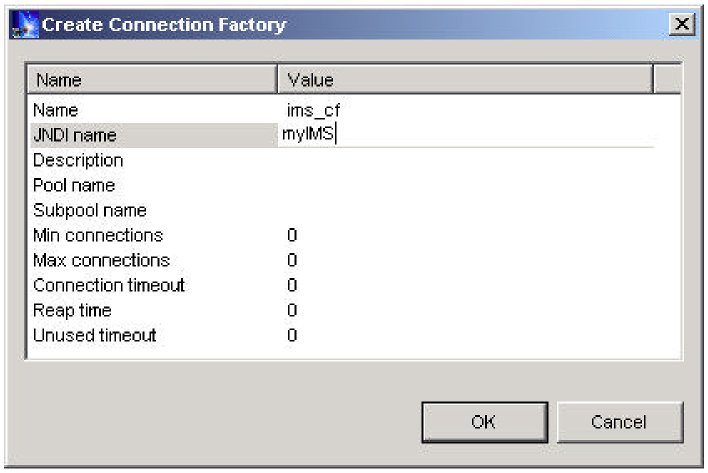
3. Click the **J2C** tab. In the Create Resource Adapter window click **Add** beside the J2C Resource Adaptors pane.
4. From the Resource Adapter Name drop-down list, select the resource adaptor named IMS. Click **OK**.



Resource Adapter Name: IMS

Name	Value
Transaction support	NO_TRANSACTION
Reauthentication support	true
Managed connection factory	com.ibm.connector2.ims.ico.IMSManagedConne...
Connection factory interface	javax.resource.cci.ConnectionFactory
Connection factory impl	com.ibm.connector2.ims.ico.IMSConnectionFact...
Connection interface	javax.resource.cci.Connection
Connection impl	com.ibm.connector2.ims.ico.IMSConnection
Archive path	C:/ims.rar
Classpath	C:\Program Files\IBM\Application Developer\wor...

5. Click **Add** beside the J2C Connection Factories pane. The application client will look up this connection factory instance using the JNDI interface. The application client will use this connection factory instance to get a connection to the underlying IMS.
6. On the Create Connection Factory window type the name ims_cf. You may need to click several times on this field to enter a value. Type the JNDI name myIMS. Click **OK**. You may need to click on another field before you can click **OK**.



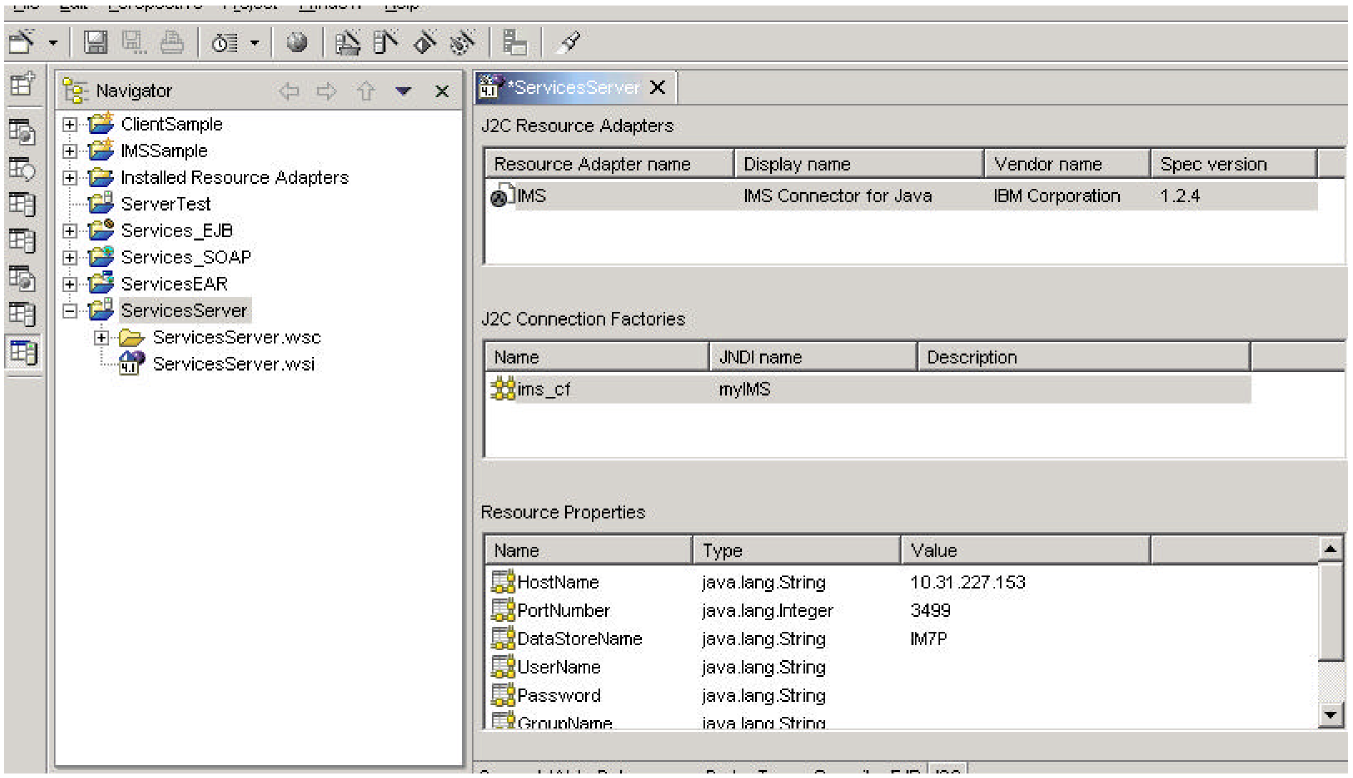
Name	Value
Name	ims_cf
JNDI name	myIMS
Description	
Pool name	
Subpool name	
Min connections	0
Max connections	0
Connection timeout	0
Reap time	0
Unused timeout	0

7. In the Resource Properties pane, type the property values appropriate for your environment.

   In the **HostName** field, type 10.31.227.153

   In the **DataStoreName** field, type IM7P

   In the **PortNumber** field, type 3499



J2C Resource Adapters

Resource Adapter name	Display name	Vendor name	Spec version
IMS	IMS Connector for Java	IBM Corporation	1.2.4

J2C Connection Factories

Name	JNDI name	Description
ims_cf	myIMS	

Resource Properties

Name	Type	Value
HostName	java.lang.String	10.31.227.153
PortNumber	java.lang.Integer	3499
DataStoreName	java.lang.String	IM7P
UserName	java.lang.String	
Password	java.lang.String	
GroupName	java.lang.String	

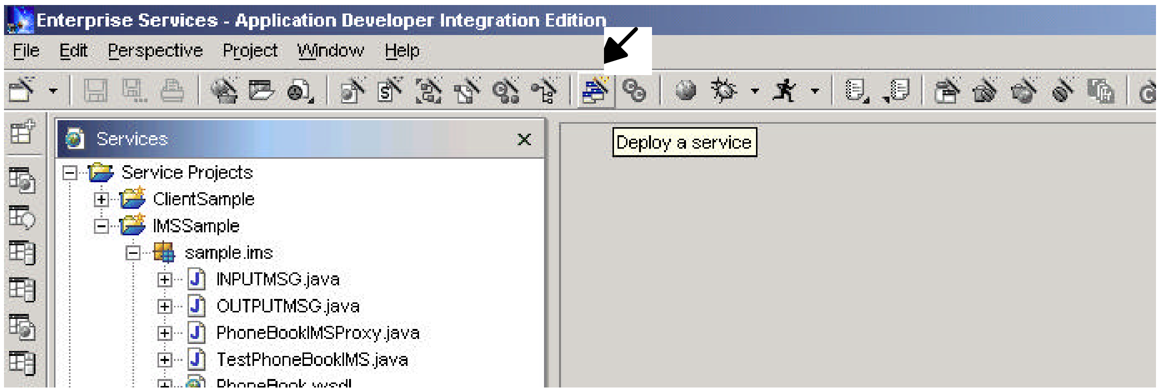
8. Close the editor and click **Yes** to save the changes.

6e. Create and deploy a Session Bean

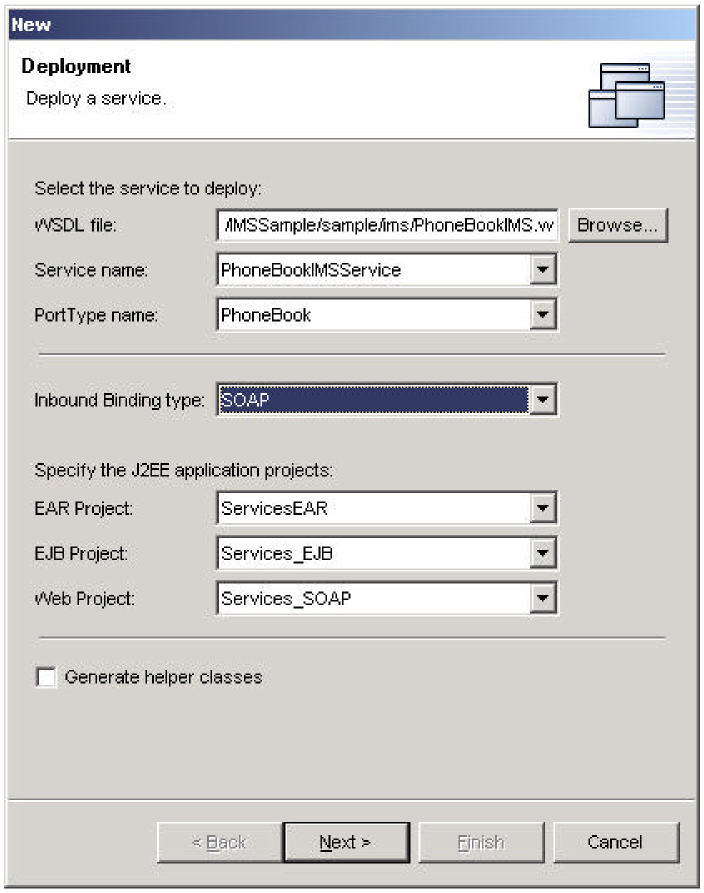
The Service Deployment wizard generates the session bean. The session bean will handle the client request to pass a transaction request, in our case the PhoneBook transaction request, to the service. The PhoneBook request is an input message request to run the IMS IVP transaction IVTNO to either: add, delete, update, or display information in the IMS PhoneBook database. It also generates deployed classes that allow the session bean to operate on an EJB server such as the Websphere Application Server.

The following steps create the session bean:

1. Go to IMSSample project and package sample.ims then select the service binding file named PhoneBookIMS.wsdl.
2. From the toolbar click Deploy a service icon. The Deployment wizard opens.



3. In the Deployment page, the WSDL file, service name and port name default to values based on the service binding file you selected earlier. Since you are a creating service that will use SOAP you need to select SOAP from the Inbound Binding type drop-down list. Make sure that the EAR project, EJB project and Web project are the same as the projects created earlier. You already generated helper classes earlier so deselect the Generate helper classes check box. Click Next.



Select the service to deploy:

WSDL file: /IMSSample/sample/ims/PhoneBookIMS.w

Service name: PhoneBookIMSService

PortType name: PhoneBook

Inbound Binding type: SOAP

Specify the J2EE application projects:

EAR Project: ServicesEAR

EJB Project: Services_EJB

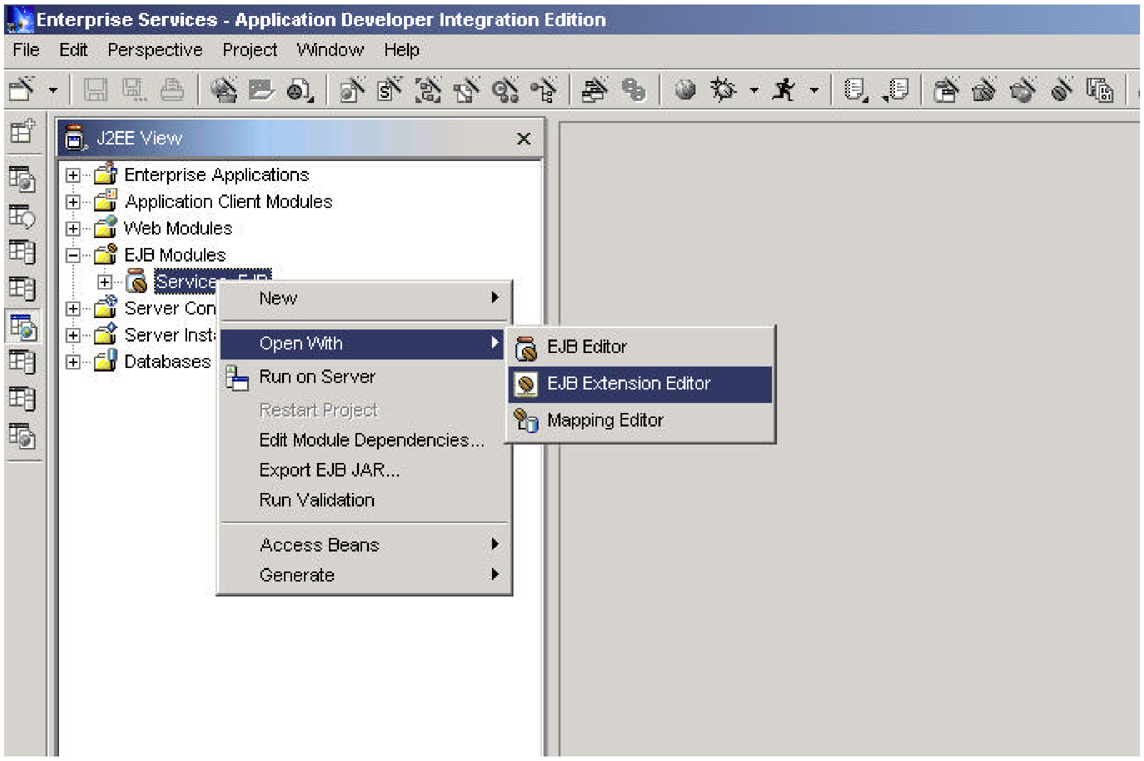
Web Project: Services_SOAP

☐ Generate helper classes

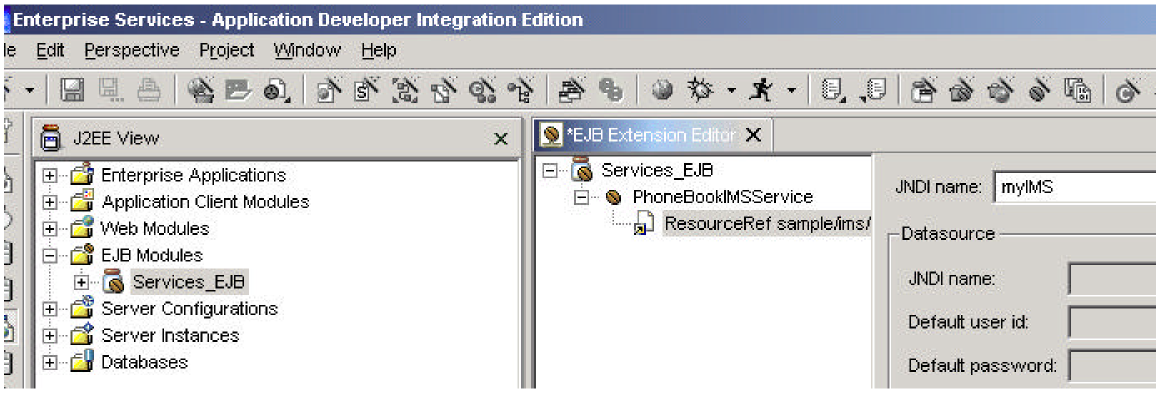
4. In the EJB Port page leave the default for the JNDI name. Although you are deploying the service as a SOAP service, you are creating two inbound bindings (SOAP and EJB) to access the service.
5. In the SOAP page you specify the SOAP binding properties. You can accept default transport, style, action and encoding properties. These properties are all specific to the SOAP specification. Click Next.
6. In the SOAP Port page you specify the SOAP Port address. You can accept the default address and Web project. Click Finish. The service definition file PhoneBookSOAP.wsdl is created in the Services_SOAP project. The SOAP deployment descriptor tells the SOAP server about the service and is created in the Web module project named Services_SOAP. It contains servlet initialization and mapping information as well as other settings to run the Web module within an application server. The Services_EJB project contains all the resources for EJB applications, including the session bean (PhoneBookIMSService), the deployment descriptor, the remote interface, and the EJB home. It also contains a service definition file (PhoneBookEJB.wsdl) that describes the generated session bean. The service project (IMSSample) contains a service definition and Bean class zipped as a JAR file and placed in the EAR project.

Now you will need to bind the resource reference which is the Java Naming and Directory Interface (JNDI) related information. With this information, a factory object at run time can generate a connection when needed by the application, locate the target IMS host system and invoke the transaction in IMS. The directions are as follows:

1. From the **J2EE View** in the Enterprise Services perspective by clicking on the J2EE tab at the bottom of the panel. Expand **EJB Modules**. Right-click Services_EJB. Select **Open with > EJB Extension Editor**.



2. Click the **Bindings** tab at the bottom of the editor panel.
3. Expand Services_EJB and expand PhoneBookIMSService. Select **ResourceRef.**
4. Type myIMS for the JNDI name.



JNDI name: myIMS

5. Close the editor and click **Yes** to save the changes.

The next step is to add the EAR project (ServicesEAR) to the server configuration that you created earlier in order to correctly start the server to test your service. To add the project, follow these steps:

1. Open up the Server Configuration view by going to the toolbar at the top of your screen and clicking on: Perspective => Open => Other => Server. In the Server Configuration view under Server Configurations, right-click **ServicesServer**.

2. Select **Add Project** > ServicesEAR which is the name of the Enterprise Application Project that you created above.

You have now completed the service provider piece of your service, that is, you have created an IMS service from an IMS transaction and deployed that service to the Application Server. Next you take on the role as a service consumer and you will create the client piece that can be used to invoke the service as a SOAP service

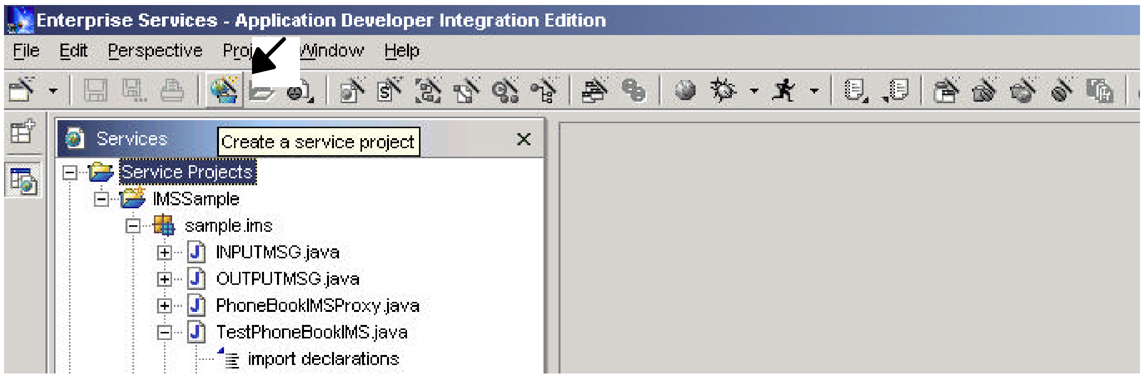
Exercise 7. Create a SOAP Proxy to test the IMS Service

There are several steps that must be done to create a SOAP Proxy:
7a. Create a java project
7b. Create the SOAP proxy
7c. Test the SOAP proxy

7a. Create a java project

We will first need to create a Java project for the client side of the service:

1. From the toolbar, click the **Create a service project** icon. The Service Project wizard opens.



2. Type **ClientSample** for the name of the project. Use the default location to store the new project. You do not need to specify Java Build Path settings or dependent JARS because these are automatically set for you. Click Finish. The Service Provider Browser opens. Close the Service Provider Browser.

7b. Create the SOAP proxy

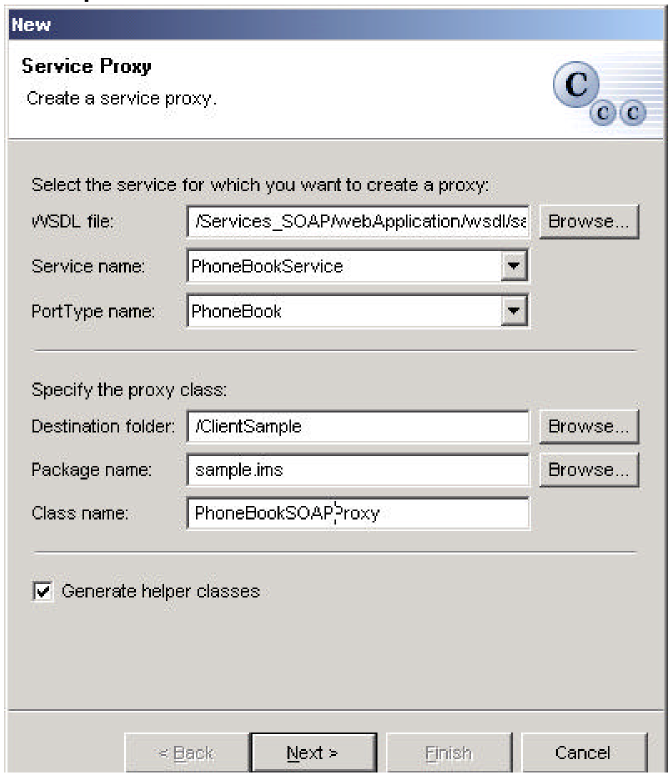
The proxy is needed to provide a remote procedure call interface to the service. Once the application makes the remote call, the proxy handles all of the communication details between the application and the service using SOAP.

To create the SOAP proxy, follow these steps:

1. In the Services view, expand the **Services_SOAP** project and the **sample.ims** package. Select the SOAP binding file PhoneBookSOAP.wsdl. The service bindings are used to generate a Soap proxy.
2. From the toolbar, click the Create a service proxy icon. The Proxy wizard opens.
3. In the Service Proxy page, check to see if the service you want to create a proxy for is shown and the proxy class properties are correct. Because you selected a file in the first step, most fields are populated with default values. These default values are generated based on the selected SOAP deployment descriptor file.
   a. Change the destination folder to /ClientSample since you want to place the proxy in the ClientSample project since it is the client application that will use the proxy.
   b. Ensure the package name is sample.ims

c. Change the class name to PhoneBookSOAPProxy.

d. Ensure that the **Generate helper classes box** is selected. Click **Next**.



New

Service Proxy

Create a service proxy.

Select the service for which you want to create a proxy:

WSDL file: /Services_SOAP/webApplication/wsdl/sa

Service name: PhoneBookService

PortType name: PhoneBook

Specify the proxy class:

Destination folder: /ClientSample

Package name: sample.ims

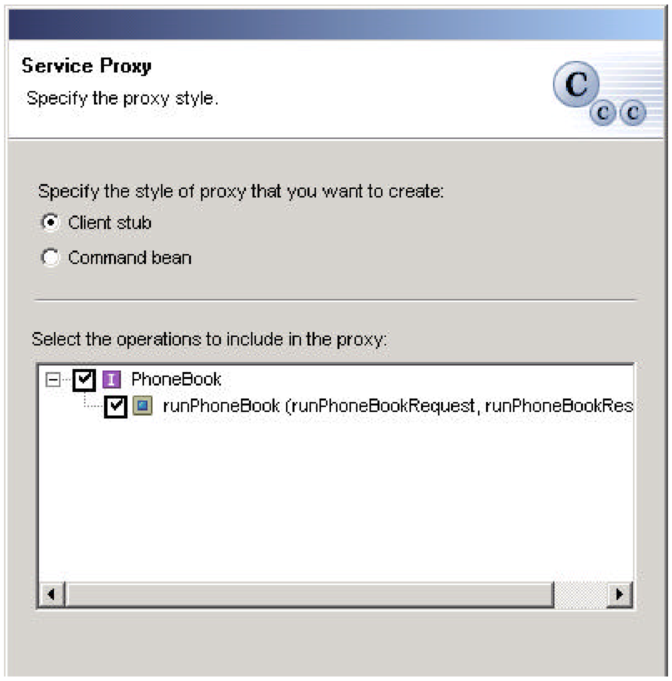
Class name: PhoneBookSOAPProxy

☑ Generate helper classes

4. In the Service Proxy Style page specify the style of the proxy and the operations to expose in the proxy:

   a. Select the client stub bean proxy style.

   b. Select **runPhoneBook** method.



Service Proxy

Specify the proxy style.

Specify the style of proxy that you want to create:

● Client stub

○ Command bean

Select the operations to include in the proxy:

☑ PhoneBook

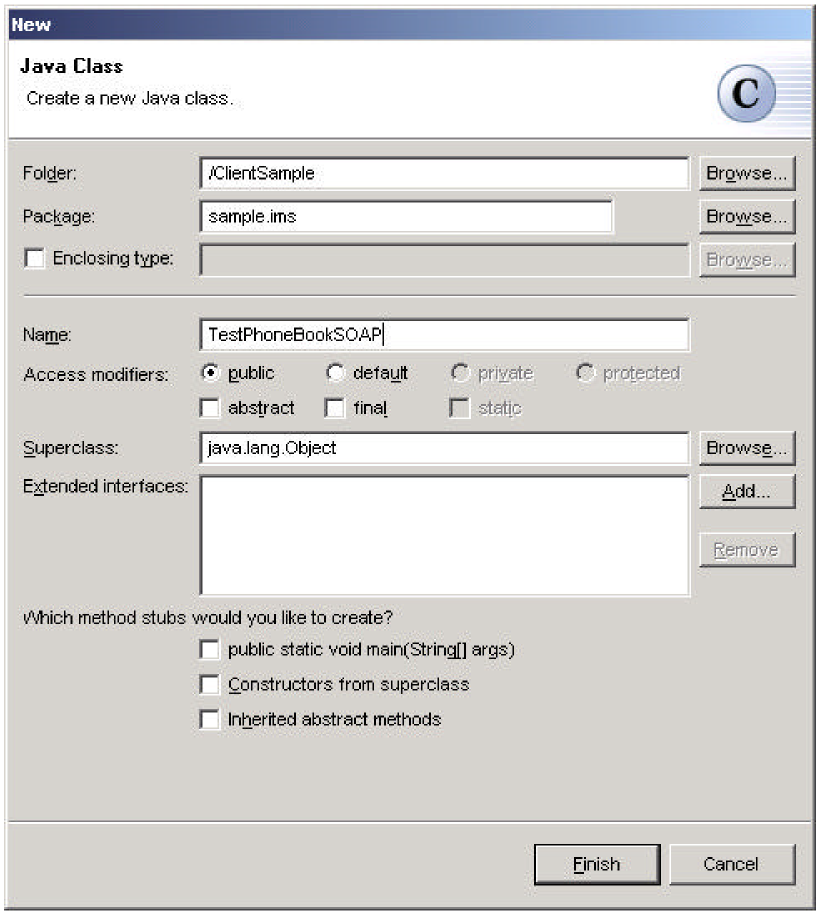
☑ runPhoneBook (runPhoneBookRequest, runPhoneBookRes

5. Click **Finish**. The SOAP proxy (PhoneBookServiceSOAPProxy) is generated in the ClientSample project.

7c. Test the SOAP proxy

You are now ready to test your SOAP proxy but will first need to create some code (a Java class) to do so.
To create the class, follow these steps:

1. Select the ClientSample project. Select sample.ims package.
2. From the toolbar click Create a Java class. In the sample.ims package, create another new class with a main method TestPhoneBookSOAP.


New

Java Class

Create a new Java class.

Folder: /ClientSample

Package: sample.ims

☐ Enclosing type:

Name: TestPhoneBookSOAP

Access modifiers: (● public) ○ default ○ private ○ protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

Extended interfaces:

Which method stubs would you like to create?

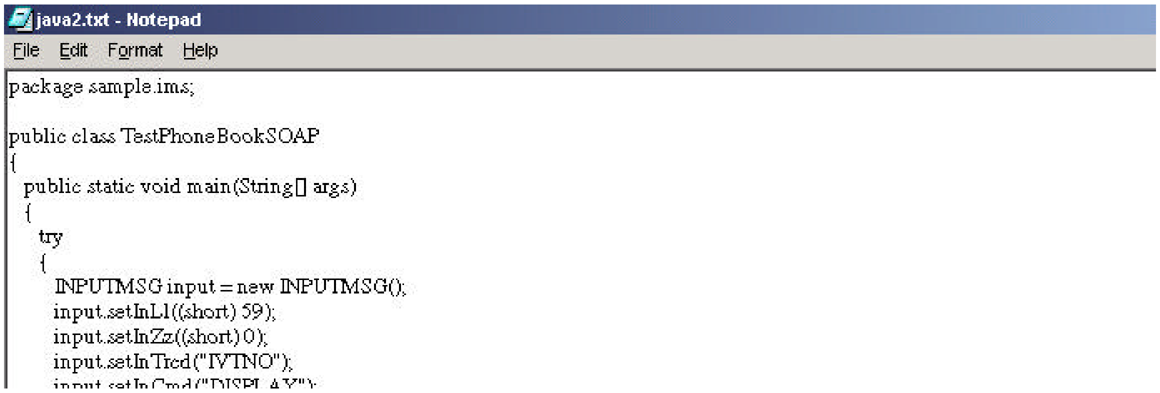
☐ public static void main(String[] args)

☐ Constructors from superclass

☐ Inherited abstract methods

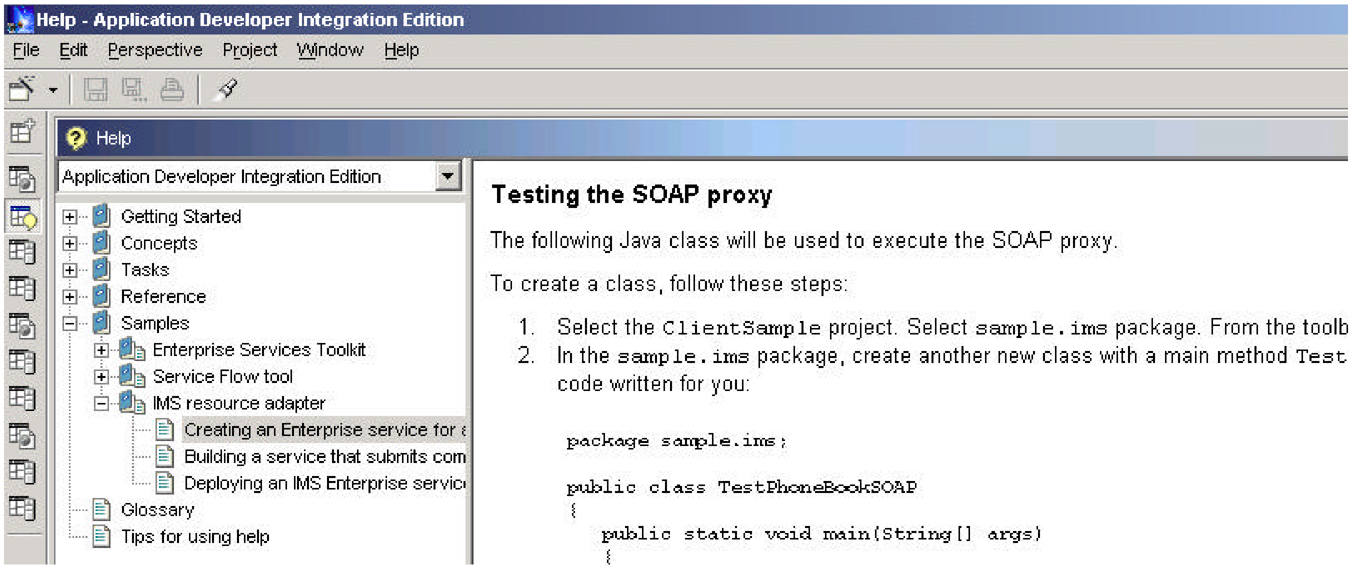
3. In the TestPhoneBookSoap.java window that opens, replace the skeleton code with the sample code that has been provided. You can find the sample code you need either in:

   - C:\workshop_DSC\java2.txt



```
package sample.ims;

public class TestPhoneBookSOAP
{
  public static void main(String[] args)
  {
    try
    {
      INPUTMSG input = new INPUTMSG();
      input.setInLl((short) 59);
      input.setInZz((short) 0);
      input.setInTrcd("IVTNO");
```

- **Or,** in the WSAD IE Help section:


Help - Application Developer Integration Edition
File Edit Perspective Project Window Help
Help
Application Developer Integration Edition
Getting Started
Concepts
Tasks
Reference
Samples
Enterprise Services Toolkit
Service Flow tool
IMS resource adapter
Creating an Enterprise service for
Building a service that submits com
Deploying an IMS Enterprise servic
Glossary
Tips for using help

Testing the SOAP proxy

The following Java class will be used to execute the SOAP proxy.

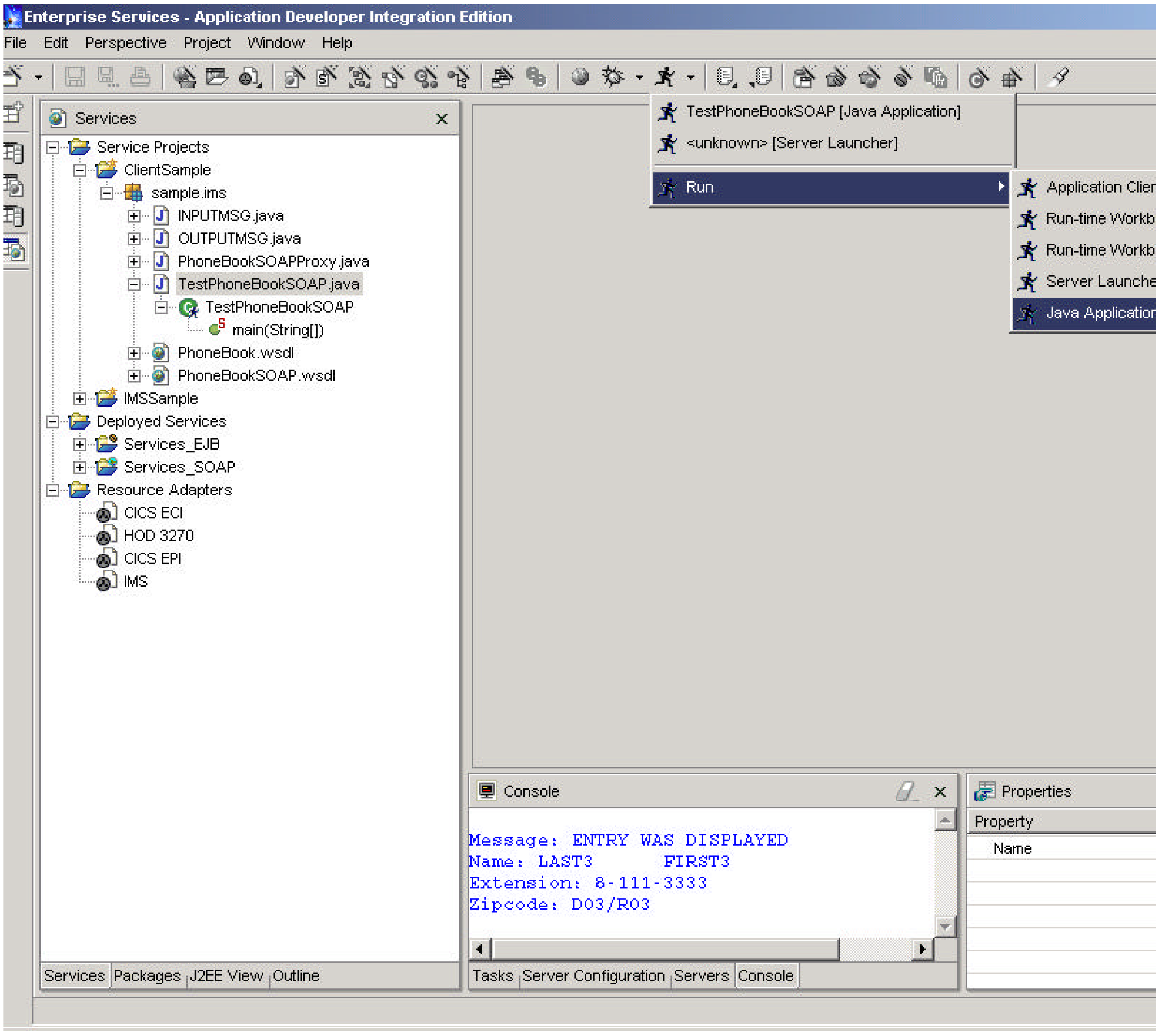
To create a class, follow these steps:

1. Select the `ClientSample` project. Select `sample.ims` package. From the toolb
2. In the `sample.ims` package, create another new class with a main method `Test` code written for you:

```
package sample.ims;

public class TestPhoneBookSOAP
{
   public static void main(String[] args)
   {
```

4. Exit out of the window and save the changes.,
5. Now we can run the **TestPhoneBookSOAP** class.
   a. Start the server. You do this by opening the Server perspective. Under Server Instance in the Servers view, right-click ServicesServer and then from the pop-up menu, select Start or Restart. The server status should change to "Started". (If it doesn't change to "Started", click the Console tab and look for exceptions or load module errors.) If the Server perspective opens when the server starts, click the Services tab to return to the Services view.
   b. In the Services perspective, open up the ClientSample project and the sample.ims package. Select TestPhoneBookSOAP. Expand the **Run** icon on the toolbar by selecting the arrow beside it. From the pop-up menu, select **Run > Java Application**.
   c. You should get a clean run without exceptions and see the following on the Console:

```
Message: ENTRY WAS DISPLAYED
Name: LAST3      FIRST3
Extension: 8-111-3333
Zipcode: D01/R03
```



```
Message: ENTRY WAS DISPLAYED
Name: LAST3      FIRST3
Extension: 8-111-3333
Zipcode: D03/R03
```

RETURN TO INDEX