



Hands-on experience with IBM WebSphere Development Studio Client for iSeries – Scenario application

Version 4.0 for Windows



Hands-on experience with IBM WebSphere Development Studio Client for iSeries – Scenario application

Version 4.0 for Windows

Contents

Chapter 1. Introduction to the iSeries

scenario application 1

Project-based scenarios 2

Chapter 2. Technology concepts 5

Chapter 3. Running the scenario 7

Before you begin 7

Installing the sample files 7

Restoring the .sav files 7

Restoring a Development Studio Client .sav file to the iSeries host 8

Importing and running the application in

Development Studio Client 9

Creating the Web projects. 10

Importing the Web project .zip files 10

Importing SV000501.zip 11

Server configuration 12

Configuring the IBM WebFacing tool 12

Defining host information 13

Running the application in the workbench 13

Running the application as a customer in the workbench 13

Running the application as an administrator in the workbench 14

Deploying the scenario application to the

WebSphere Application Server 15

Configuring WebSphere Application Server. 15

Securing the administrator's page 16

Creating EAR files for iSeries WebSphere

Application Server deployment. 17

Deploying the EAR files to iSeries WebSphere

Application Server 17

Running the application in WebSphere

Application Server 18

Chapter 4. Building the application . . . 19

Introduction 19

Before you begin 19

Step-by-step module 1: SV000514 – Create a Web service to return product prices. 19

Creating a new Web project 20

Defining the iSeries information 20

Creating the RPG service program. 20

Testing the sample 23

Step-by-step module 2: SV000501 – Creating the administrator interface to view inventory and order. 23

Creating a WebFacing project 24

Converting the DDS source 25

Creating a style sheet (optional) 27

Creating the Web interaction using iSeries Web development tools 28

Link the project to a Web interaction 30

Restarting the server 31

Testing the interface 32

Advanced module 1: SV001585 – Create HTML, servlets, and JSP files that place customer orders on an iSeries host 33

Summary of high level steps 34

Creating the Web page, servlets, and JSP files . . . 35

Advanced module 2: SV000618 – Create the Web page that uses the SV000514 and SV001586 Web services. 41

Creating the Web pages, servlets, JSPs, and RPG code. 43

Before you deploy to WebSphere Application Server 44

Notices 47

Programming Interface Information 48

Trademarks and Service Marks 48

Chapter 1. Introduction to the iSeries scenario application

This scenario is a sample application developed using IBM WebSphere Development Studio Client for iSeries version 4.0, the Eclipse-based technology for the iSeries host. The scenario is designed for developers wanting to use an iSeries host for Java development, Web development, and the management and transformation of RPG code into Web applications.

The application consists of five .zip files and two .sav files that you restore onto your iSeries host. They form a series of Web pages with URLs for various points in the process.

Important note: Service pack 1 installation is required before you can work on the exercises. Check our support page for all downloads and fixes:
ibm.com/software/ad/wdt400/support/.

This scenario takes you through various parts of the product with an emphasis on iSeries-specific components such as the IBM WebFacing Tool, Web services, Web development tools for iSeries (including the Web Interaction wizard and Program Call wizard), Java development tools for iSeries, and the IBM Toolbox for Java.

The scenario illustrates the situation of two companies, a wholesale supplier and a clothing retail store, who do business together and both use their iSeries host for business logic and data. In the past, the businesses have communicated with each other through e-mail, telephone, and fax to check inventory, submit orders, and track orders to fulfilment. They now hope to use the Web to carry out regular business transactions.

The retail store hopes to have a Web site that their customers can use to purchase products and their employees can use to order inventory from the wholesale supplier. The wholesale supplier hopes to receive retailer orders online for tracking, and wants to service multiple potential customers. In this scenario, you will take the role of a programming consultant for both of these companies, helping move their businesses to the Web.

The application has two different entry points based on the type of user. As a customer, you begin by viewing the products the store has to offer, in this case, casual clothing. If you want to make a purchase, you can click to access an order screen. After you order, a summary page is generated, and you can continue to shop, cancel the order, or submit the order.

As an administrator, you have a secure user ID for the application, requiring the security policy for the application to be defined during deployment. Your entry point is a login screen where you can view your orders, view the inventory, and purchase from the wholesale supplier. You could select items, check the latest wholesale price, and order the size and quantity you want. The application verifies if the wholesaler has the size and quantity you want, and either confirms the order or tells you that it cannot fulfill the order at this time.

Underneath the surface of the application, many actions are taking place in the various parts of the product. The following table illustrates the process and component of the product responsible for each part of the application. Continue on to the detailed information about how to perform each task.

Table 1.

Customer application tasks	Administrator application tasks	Underlying process
Display product prices		Use an iSeries RPG program to create a Web service, and use Web development tools to view and display the prices.
Place an order from the store		Use servlets and JSP files along with iSeries Java development tools, iSeries Web development tools, and the IBM Toolbox for Java to access and view inventory on an iSeries host as well as place orders and show a purchase summary.
	View inventory	Use the IBM WebFacing Tool to convert an existing RPG program into a Web application and use Web development tools to customize the Web page.
	Order from the wholesale supplier with the merchandise ID and quantity.	Create a Web service that is invoked when you click the Purchase button.
View initial web page for the store.	View initial web page for ordering inventory.	Use Web development tools to create both home pages.

Project-based scenarios

This scenario comes in the form of five projects:

SV000501 project: Create a Web page to view outstanding orders, inventory, and product details – This project is created with iSeries Web development tools and the IBM WebFacing Tool, and is designed for RPG programmers with limited knowledge of Web application development who want to use the IBM WebFacing Tool to put their RPG applications on the Web.

SV001585 project: Create HTML code, servlets, and JSP files that place client orders on an iSeries host – This project uses the IBM Toolbox for Java’s SQL and JDBC classes, the RecordIOManager bean of iSeries Java development tools, and the iSeries Program call wizard. These elements show various ways to access and manipulate data and programs that exist on the iSeries host. This project is designed for Java programmers and Web application developers who want to develop Web pages to access iSeries data and code. In addition, you should have working knowledge of iSeries host management and RPG programming.

SV000514 project: Create an iSeries Web service to provide product prices – The Web Services wizard uses a Java bean generated by the iSeries Program Call wizard to call one or more program procedures on the iSeries host, and convey the information back to a browser. This project is designed for RPG programmers who want to use Web services to create self-contained, modular applications that can be described, published, located, and invoked over the World Wide Web.

SV001586 project: Create a Web service to place orders to the wholesale supplier through an iSeries host – The Web service accepts a merchandise ID, plus the required quantity, and then places an order with the wholesale supplier. This project is a component of SV000514, and is designed for RPG programmers who want to create Web services.

SV000618 project: Create a Web page to interface the order form, inventory form, and purchase order generated by the IBM WebFacing Tool – This project requires iSeries Web development tools, and involves creating HTML and JSP files to use and connect the Web services developed in SV000514 and SV001586. The project is designed for developers who want to work with Web services, and who have knowledge in RPG and Java programming.

Chapter 2. Technology concepts

In order to work through the scenario application, you need to be familiar with a number of technology concepts, especially if you are new to Web application development. The following is a brief list of some of the things you will encounter when you work through the application.

Design Time Control (DTC)

DTCs are used to define iSeries objects such as data entry fields and push buttons, which can exchange information between iSeries host programs and the Web page. Developers can use DTCs to capture user events such as syntax checking of entry fields and button clicking.

Enterprise Archive file (EAR)

An EAR file is a standard Java Archive (JAR) file with an .ear extension. In the GUI version of the J2EE SDK application deployment tool, you create an EAR file first and add JAR and Web Archive (WAR) files to the EAR file. If you use the command line packager tools, however, you create the JAR and WAR files first and then create the EAR file.

IBM WebFacing Tool

The IBM WebFacing Tool converts existing 5250 interfaces to browser-based graphical user interfaces. With little or no modification to your original iSeries applications, you can extend the use of your programs to the Internet or an intranet.

Java Archive file (JAR)

A JAR file is a compressed package of Java files, similar to a .zip file. It contains the class, image, and sound files for a Java applet gathered into a single file and compressed for faster downloading to your browser.

Java Server Pages (JSP)

JSP provide the ability to display dynamic content in static HTML pages. Written in Java, JSP are server and platform-independent. By effectively separating the Web presentation from the Web content, JSP can help developers who need to quickly change the design and display of their Web pages.

Program Call bean

These are the Java beans generated by the Program Call wizard. One type is a regular Java bean used by Java applications. The other type can be used by the Web service wizard to create a Web service.

Program Call wizard

The Program Call wizard helps you create the Java beans and associated PCML file needed to invoke an iSeries program or procedure. The wizard prompts you for

information regarding program or service program objects, along with the parameters for the objects, and then creates the desired Java beans (and PCML file).

Report Program Generator (RPG)

A procedural programming language used by iSeries programmers. You can use RPG to create business applications such as invoicing programs and order entry programs. The latest version, ILE RPG IV, expands the capabilities of the RPG language, while supporting programmers' experience with previous versions.

Servlet

Server-side programs, written in Java, that run in Java-enabled servers or application servers such as IBM WebSphere Application Servers. Servlets perform tasks specified by the server, such as responding to requests by generating an HTML response. For example, you can use servlets in an online banking application to respond to the user while sending data to the server.

Web Interaction wizard

This wizard is part of iSeries Web development tools. It creates and manages the interactions between ILE programs and Web pages. The wizard controls where input, output, and error messages are displayed, and directs the data from the input and output fields to the ILE programs. The Web Interaction wizard can be used to map error messages to the area where the error occurred so that the user can identify the source of the error easily.

Web services

A self-contained application that is designed and implemented to be used over the Internet. Web services are created using open standards such as SOAP, WSDL, and XML. There are numerous business situations with which you can use a Web service, including an inventory management system where clients can check their inventory levels through the Internet. Another example is the creation of a Web service to track a product order directly from a supplier.

Web services definition language (WSDL)

WSDL is an XML-based language that defines the interface of a Web service. WSDL understands a Web service, and manages the flow of information between the Web service and the host program. For example, a developer would use WSDL to create an interface for a Web site that shows updated stock quotes.

WebSphere Studio Workbench

IBM WebSphere Development Studio Client for iSeries is built on WebSphere Studio Workbench, IBM's implementation of the Eclipse platform. The extensible, universal workbench integrates all of the tools necessary to build and maintain applications. Developers can use Development Studio Client to incorporate new objects into the development environment through the use of plug-ins, and seamlessly add Java files, graphics, video, and so on.

Chapter 3. Running the scenario

You can run the *Wholesale* and *Retailstore* applications inside the Development Studio Client workbench or on WebSphere Application Server for any platform including the iSeries platform. See Chapter 1, “Introduction to the iSeries scenario application” on page 1 for an overview of the applications.

Before you begin

To test the applications from the workbench, you need to ensure that:

- You have created or have access to an HTTP server configuration and instance on the iSeries host, and the instance is running.
- You have NET USE access to the iSeries host, and have mapped the /QIBM directory of the iSeries host to a drive letter on your Windows® system.

Installing the sample files

To use the iSeries scenario application you need to work with the seven files downloaded from the online package available in the education section of our Web site: ibm.com/software/ad/wdt400/education/course_downloads.html:

- Wholesale.sav
- Retailstor.sav
- SV000501.zip
- SV000514.zip
- SV000618.zip
- SV001585.zip
- SV001586.zip

The .sav files contain iSeries data and RPG programs, and the .zip files contain the Web applications that interact with the iSeries programs to manipulate iSeries data. First, you need to restore the .sav files, and then you can import the .zip files into the workbench and run the application in the IDE.

Restoring the .sav files

To work with the samples in this guide, you need to restore the WHOLESALE and RETAILSTOR libraries on your iSeries host. You should do this even if you have already restored the libraries for a previous release of the product, because their contents are different. The instructions describe how to restore the Wholesale library. To restore the RETAILSTOR library, repeat the instructions exactly except use the word “RETAILSTOR” wherever you see the word “WHOLESALE”.

Note: The .sav files used to install the sample library are for use with a V5R1 or later iSeries host. For the purposes of this scenario, both libraries are restored to the same iSeries host, but if you were developing this application for a real business, you would restore the two libraries to two different iSeries hosts. You would restore the WHOLESALE library to the iSeries host that is providing the Web services, and you would restore the RETAILSTOR library to the iSeries host that belongs to the retail store.

To restore the Wholesale.sav file:

1. Log on to your iSeries host through a green-screen emulator.

- a. Create a library to contain the save files. To create a new library in the emulator, enter CRTLIB.
- b. Give your library a name, such as SCENARIO.
- c. Tab to the next line, specify *TEST as the library type and press enter.
- d. Create a save file using the CRTSAVF command:
CRTSAVF FILE(MYLIB/WHOLESALE)

where MYLIB is the library you created in step two.

2. On your workstation, open a Command Prompt window.
 - a. Change to the directory where you downloaded the application files, for example, c:\temp.
 - b. On the command line, enter: ftp *hostname*, where *hostname* is the name of your iSeries host, for example, PROD400.
 - c. Enter your iSeries user ID and password.
 - d. On the command line, enter cd MYLIB where MYLIB is the same library you created to contain the save files.
 - e. Enter the following:
bin
put WHOLESale.sav WHOLESale
quit
3. Back in the iSeries console:
 - a. Enter RSTLIB and press F4.
 - b. In the **Saved Library** field, enter WHOLESale and press the tab key.
 - c. In the **Device** field, enter *savf and press the tab key
 - d. Press enter in the next field to display additional values and tab to the **Save file** field.
 - e. Enter "WHOLESale" and press the tab key.
 - f. In the **Library** field, enter MYLIB, where MYLIB is the library you created to contain the save files. This restores the WHOLESale library on your iSeries host.
 - g. Press enter to save your action.
4. Repeat this procedure for the Resailstor.sav file to restore the RETAILSTOR library (except for 1.a, b, and c since the library is already created).

Restoring a Development Studio Client .sav file to the iSeries host

For the SV000618 project to run, you need to restore an object to the iSeries host:

1. Logon to your iSeries host through a green-screen emulator. If you do not have a 5250 session set up from your workstation:
 - a. Open the Remote Systems Explorer by clicking **Perspective > Open > Remote Systems Explorer**.
 - b. Expand **New Connection** to invoke the **New Connection** window.
 - c. In the Connection name field, enter the name of your iSeries host, for example, PROD400.
 - d. In the **System Type** field, select **iSeries** from the drop-down list if it is not already selected.
 - e. In the host name field, enter the name of your iSeries host, for example, PROD400.
 - f. Click **Finish**.

- g. In the Remote Systems view, expand your new connection, which would be the name of your iSeries host.
 - h. Right-click **iSeries Commands** and select **Open 5250 emulator**.
 - i. When the emulator opens, sign on with your user ID and password.
2. Create a save file using the CRTSAVF command:
CRTSAVF FILE(MYLIB/QDTSSFL)

where *MYLIB* is the library you created in the previous section, e.g., "SCENARIO".

3. On your workstation, open a Command Prompt window.
4. Change to the following directory: cd
x:\WDSC\WSSD\plugins\com.ibm.etools.iseries.webtools\lib\ where x is the directory where you installed Development Studio Client.

Note: If you installed Development Studio Client on top of WebSphere Studio Application Developer, the beginning of your plugins directory would be x:\WSAD\WDSC\plugins...

5. Enter: ftp *hostname* where *hostname* is the name of your iSeries host, for example, PROD400.
6. Enter your iSeries user ID and password.
7. On the command line, enter cd *MYLIB* where *MYLIB* is the same library you created to contain the save file.
8. Enter the following:
bin
put QDTSSFL.sav QDTSSFL
quit
9. Back in the iSeries console, enter RSTOBJ and press F4.
10. In the **Objects** field, enter *all and press the tab key.
11. In the Saved Library field, enter QGBL and press the tab key.
12. In the **Device** field, enter *savf and press the tab key.
13. Press enter.
14. In the **Save file** field, enter qdtssf1 and press the tab key.
15. In the **Library** field, enter *MYLIB* where *MYLIB* is the library you previously created.
16. Press enter to save your changes.

Importing and running the application in Development Studio Client

When you downloaded the application, you extracted five zip files to a directory on your local system:

- SV000501.zip
- SV000514.zip
- SV000618.zip
- SV001585.zip
- SV001586.zip

The first file corresponds to a WebFacing project, and the others correspond to Web projects. Therefore, the steps for importing differs slightly between the two types of files.

Creating the Web projects

The Web project creation steps are very similar for SV000514.zip, SV000618.zip, SV001585.zip, and SV001586.zip, with a few differences, concerning which projects are associated with which EAR files, outlined below.

To create the Web projects:

1. Open IBM WebSphere Development Studio Client for iSeries from the start menu.
2. Open the Web perspective from the menu bar by clicking **Perspective > Open > Other > Web** and click **OK**.
3. Click **File > New > Web Project**.
 - a. In the **Project name** field, enter SV000514.
 - b. In the **Enterprise Application project name** field, enter SVWholeSaleEAR.
 - c. Click **Finish** and note that the **SV000514** project and the **SVWholeSaleEAR** file are added to your workspace.
4. Click **File > New > Web Project**.
 - a. In the **Project name** field, enter SV001586.
 - b. In the **Enterprise Application project name** field, enter SVWholeSaleEAR and click **Finish**.
5. Click **File > New > Web Project**.
 - a. In the **Project name** field, enter SV000618.
 - b. In the **Enterprise Application project name** field, enter SVRetailStorEAR and click **Finish**.
6. Click **File > New > Web Project**.
 - a. In the **Project name** field, enter SV001585.
 - b. In the **Enterprise Application project name** field, enter SVRetailStorEAR and click **Finish**.

Importing the Web project .zip files

To import the .zip files into your workspace, the following process outlines how to import SV000514.zip. The process is the same for the other three zip files, except for the names of the files.

To import the first .zip file:

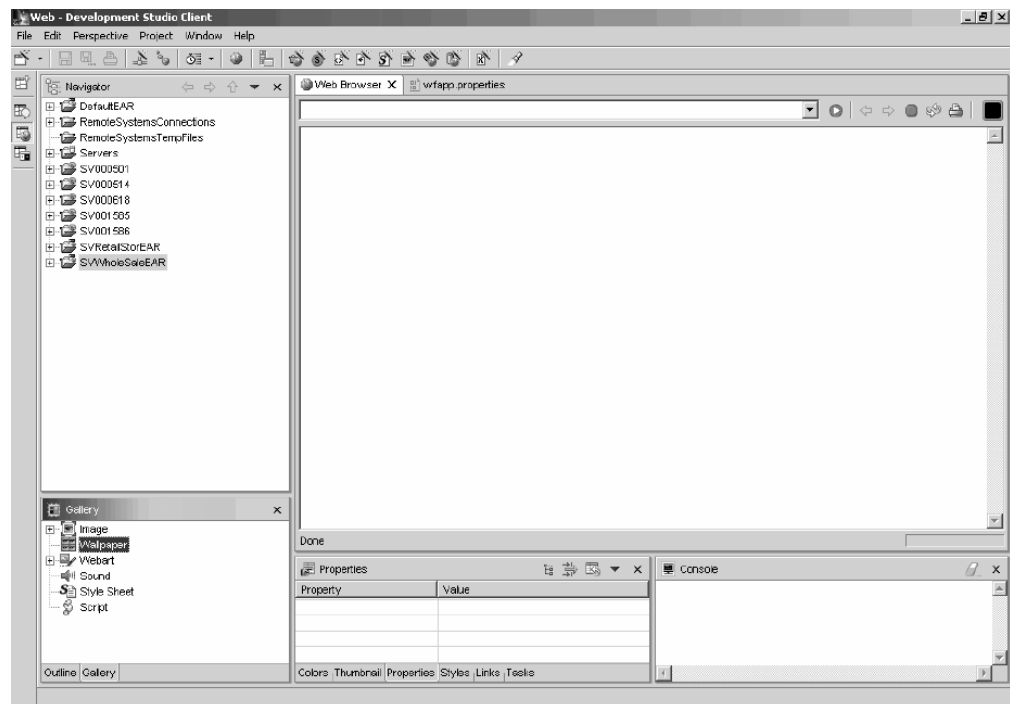
1. In the Navigator view, click **SV000514** to select it.
2. From the menu bar, click **File > Import**.
3. From the Import wizard, click **Zip file** and then click **Next**.
4. Browse to the directory where you downloaded the .zip files, for example, c:\temp.
5. Click **SV000514.zip** and click **Open**.
6. Click **Select All** to make sure that all components of SV000514.zip are selected.
7. In the Folder field, enter **SV000514** if it is not already filled in by default.
8. Select the **Overwrite existing resources without warning** check box.
9. Click **Finish**.
10. Repeat steps 1 to 9 for SV000618, SV001585, and SV001586, making sure to use the different names.

Importing SV000501.zip

To import the SV000501.zip file into the Development Studio Client workbench:

1. Make a temporary directory on your local workbench named SV000501. For example, c:\temp\SV000501.
2. Go to where you extracted all of the .zip files when you downloaded the application.
 - a. Double-click **SV000501** to open the contents in WinZip. (If you are using a different unzip program, complete the tasks as you would with that program.)
 - b. Click the **Extract** button.
 - c. Navigate to where you created the temporary directory, for example, c:\temp\SV000501 and click **Extract**.
3. From the workbench menu bar, click **File > Import**.
4. Click **WebFacing Projects** and then click **Next**.
5. Click **Browse** and navigate to the directory that contains the extracted files from the SV000501.zip file, for example, c:\temp\SV000501.
6. Click the **SV000501** folder and click **OK**.
7. Back in the **Import** wizard, click **Next**.
8. Click **Select All** and click **Finish** to import all elements of the WebFacing project into your workspace.

Now that you have imported all of the files, the your workspace should look something like this:



Altering the EAR files

Now that you have imported the SV000501 project, you need to alter the location of the .war files so that they appear under the correct EAR file:

1. In the Navigator view, expand **DefaultEAR > META-INF**.

2. Double-click **application.xml** to open it in the application editor. Click **Yes** if you receive any messages.
3. Click the **Modules** tab.
4. Click **SV000501.war** and click **Remove**.
5. From the menu bar, click the save icon or click **File > Save application.xml**.
6. In the Navigator view again, expand **SVRetailStorEAR > META-INF**.
7. Double-click **application.xml** to open it in the application editor.
8. Click the **Modules** tab.
9. Click **Add**.
10. From the **Folder Selection** window, click **SV000501** and click **OK**.
11. Click the save icon or click **File > Save application.xml** and close the file.

Server configuration

Now that you have imported all of the source files, you need to configure the WebSphere administrative server so that it recognizes the WholeSale and RetailStor EAR files.

To configure the WebSphere administrative server:

1. In the Web perspective, you need to start a Server instance by expanding **SV000501 > Web Application**, right-clicking **index.html** and selecting **Run on Server**.
2. After the server starts and loads the index page into the workspace, click **Perspective > Open > Other > Server** and click **OK** to open the Server perspective.
3. In the Server Configuration view, expand **Server Configuration**.
4. Right-click **WebSphere Administrative Domain** again and select **Add Project > SVWholeSaleEAR**.

To check that the server has picked up the projects:

1. Click the **Servers** tab in the Servers view.
2. Right-click **WebSphere v4.0 Test Environment**.
3. Select **Restart Project** and confirm that the five SV files and two EAR files, plus the DefaultEAR file, display in the pop-up menu (You do not need to restart anything.)

Note: If **Restart Project** is not enabled, you need to first start the server by selecting **Start** from the right-click menu.

Configuring the IBM WebFacing tool

Before you run the application or work on the SV000501 module, you need to start the WebFacing server for the application to run, and you need to configure your wfapp.properties file in the workbench so that the application picks up the correct port from the iSeries host.

To start the WebFacing server:

1. Open a green-screen 5250 emulator and sign on with your user ID and password.
2. At the command line, enter `strtcpsvr *webfacing`.

You need to change the WebFacing properties file . To change the wfapp.properties file:

1. Switch to the Web perspective and expand **SV000501 > webApplication > WEB-INF > classes > conf**.
2. Double-click **wfapp.properties** to open it in the default editor.
3. Change the value beside {WFAppHostName} to the name of your iSeries host, for example, PROD400.
4. Confirm that the value beside {WFAppPortName} is 4004. (If it is not, change it to this value.)
5. Leave the values beside {WFAppHostUID} (user ID) and {WFAppHostPWD} (password) blank unless you want to be logged on automatically, in which case you can add these values.
6. Click the save icon or click **File > Save wfapp.properties**.

Defining host information

After you import all of the EAR files into Development Studio Client, you need to define the iSeries host information for all five projects, ensuring that they are configured to run with your iSeries host under your user ID and password. To define host information:

1. Make sure you are in the Web perspective.
2. In the Navigator view, right-click **SV000501** and select **iSeries host information setup**.
3. In the **iSeries host name** field, enter the name of the iSeries host where you restored your RetailStor save file, for example PROD400.
4. Enter your user ID and password.
5. Click **Finish**.
6. Repeat steps 2 to 5 for the other four projects: SV000514, SV000618, SV001585, and SV001586.

Running the application in the workbench

At this point, you can run the application in the workbench, and later on, after you recreate the EAR files for iSeries WebSphere Application Server deployment, you can run the application on the iSeries host, which is covered towards the end of this chapter. There are two points of entry for the Web application, as a customer, and as the administrator.

Running the application as a customer in the workbench

As a customer, you would begin on the shopping Web page for the retail store, and then browse through items and ordering quantity and size.

Note: You may experience difficulties if you attempt to run the application behind a firewall, because your web.xml file looks for the following file:

`http://java.sun.com/j2ee/dtds/web-app_2_2.dtd`. To solve this problem, before you run the application, change the DOCTYPE statement in all of your web.xml files to:

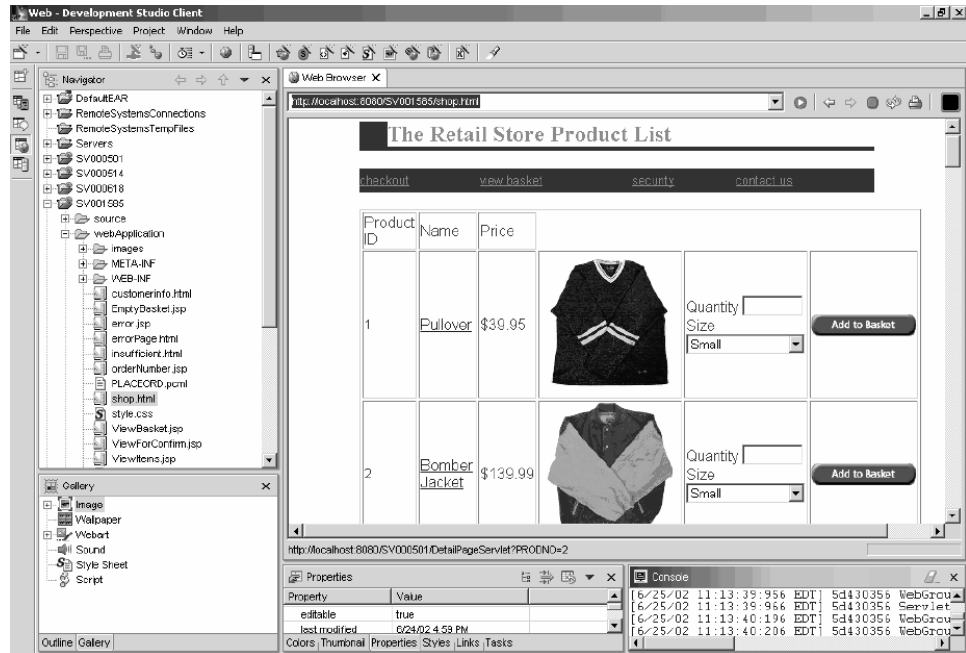
```
!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
"x:/WDSC/WSSD/plugins/com.ibm.etools.j2ee/dtds/web-app_2_2.dtd"
```

where x:/wdsc is the directory where you installed the product.

To run the application as a customer in the workbench:

1. In the Web perspective, expand **SV001585 > webApplication**.

2. Right-click **shop.html** and select **Run on Server**. This launches the application in the workbench browser.
3. Enter the application by clicking the image of the T-shirts.
4. Try entering values from the following page, pretending that you are a customer ordering a Pullover or a Bomber Jacket, selecting the size and quantity, and adding items to your basket:



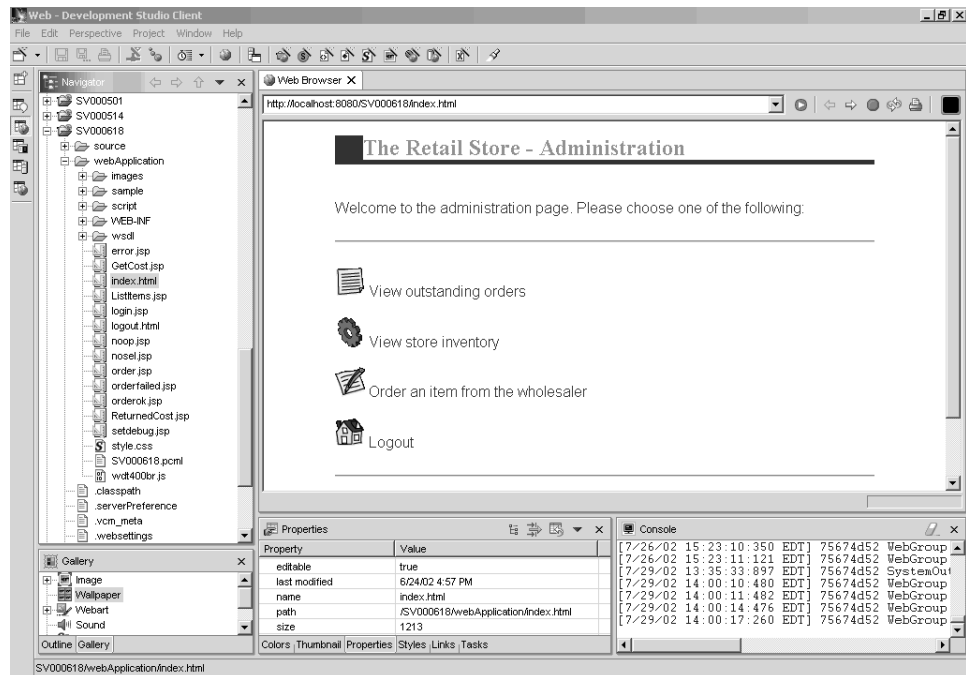
The page that displays the text "your order has been processed" is the last page of the sample. When you are satisfied with the sample, you can close the browser.

Running the application as an administrator in the workbench

As an administrator, you would begin on the administration Web page for the retail store, check on orders or store inventory. To run the application as an administrator in the workbench:

1. In the Web perspective, expand **SV000618 > Web Application**.
2. Right-click **index.html** and select **Run on Server**. This launches the application in the workbench browser.

- Click the icon to the left of **View store inventory** to display the following page, as if you were an administrator deciding what to buy for your store:



Try clicking the icons to display the administrative pages.

Deploying the scenario application to the WebSphere Application Server

Now that you have run the application in the the workbench test environment, you can deploy the iSeries application to WebSphere Application Server, as if the application were being run in the real world. Before you deploy the application, however, you need to make a few adjustments, such as securing the administrator's page, and changing a Web service's URL so that the application points to the right place, as explained in the next sections.

Note: Deploying to WebSphere Application Server is optional; you can still continue to the next chapter and complete the modules without testing the application in WebSphere Application Server.

Configuring WebSphere Application Server

Earlier in the chapter, we discussed "Running the application in the workbench" on page 13. Now that you have deployed the files to the iSeries host, you can run the application on your iSeries using WebSphere Application Server.

To deploy the application to WebSphere Application Server (optional) you need to ensure that:

- You have created or have access to a WebSphere Application Server version 4.0 instance on the iSeries host, and the instance is running (only if you want to test deployment to WebSphere Application Server).
- You know the port numbers for the HTTP and WebSphere Application Server instances on the iSeries host.
- You have the WebSphere Administrative Console version 4.0 installed on your workstation.

Your Web-enabled iSeries applications use WebSphere Application Server to run the Java™ servlets and JavaServer Pages™ (JSPs) that communicate between the Web user's browser and the iSeries programs or data. To serve your HTML pages and JSP files from the same iSeries system, you also need an HTTP server on that system. We recommend that you use the IBM HTTP Server powered by Apache. You can find documentation about how to use this server in the: IBM HTTP Server for iSeries Documentation Center at http://publib.boulder.ibm.com/pubs/html/iseries_http/v5r1/index.htm.

WebSphere Application Server executes the JavaServer pages, Java beans™, and Java servlets that are generated for various processes. The primary documentation resources for IBM WebSphere Application Server for iSeries and IBM WebSphere Administrative Console for iSeries are available at the following Web sites:

- IBM WebSphere Application Server Version 4.0 Advanced Edition for iSeries at <http://publib.boulder.ibm.com/was400/40/AE/english/docs/>
- IBM WebSphere Application Server Version 4.0 Advanced Single Server Edition for iSeries at <http://publib.boulder.ibm.com/was400/40/AEs/english/docs/>

Becoming familiar with the IBM WebSphere® Application Server documentation, in particular, the sections on *J2EE modules*, *Installing WebSphere Application Server*, and *Setting up multiple instances of the WebSphere administrative server*, is highly recommended. Minimally, you need to carry out the steps under the *Installation* link.

Use the site map to find information about how to install, configure, and obtain the required PTFs for WebSphere Administrative Console.

Securing the administrator's page

Because the administrator's page, `index.html`, should only be accessible by authorized people it should be properly secured. This can be done programmatically, in the Web application logic, or by using WebSphere's security feature. In this scenario, we used WebSphere security to secure the page. Note that we used WebSphere Application Server V4.0 Advanced Edition. You can find information about WebSphere Application Server at their Web sites:

- IBM WebSphere Application Server Version 4.0 Advanced Edition for iSeries at <http://publib.boulder.ibm.com/was400/40/AE/english/docs/>
- IBM WebSphere Application Server Version 4.0 Advanced Single Server Edition for iSeries at <http://publib.boulder.ibm.com/was400/40/AEs/english/docs/>

If you are using a different version of WebSphere Application Server, refer to that version's documentation on securing Web resources.

You can configure security for Web resources, such as Web pages and servlets, within Development Studio Client, or in the Application Assembly Tool. For this scenario we use Development Studio Client.

To review the security configuration and properties for this Web application:

1. Open Development Studio Client from the Start menu and switch to the Web perspective by clicking **Perspective > Open > Other > Web** and then clicking **OK**.
2. In the Navigator view, expand **SV000618 > webApplication > WEB-INF**.
3. Double-click **web.xml** to open the `web.xml` view.
4. Click the **Security** tab in the middle section of the view.

5. To secure index.html, the administration page, a Security constraint has also been defined. Click the first **SecurityConstraint** item in the **Security constraints** window.
6. Click **AdminPage** in the **Web resource collection** window.
7. Click **Edit** to invoke the **Web resource collections** dialog. Note that the GET and POST methods for index.html are pre-selected.
8. Click **OK**.

In the **Security roles** section, notice the defined security role named "Administrator." During deployment, individuals are assigned to this role and therefore given access to the index.html page. In the **Authorized roles** section, note that we have given the role Administrator access to this security constraint. With this security in place, only users assigned to the Administrator role are granted access to the index.html page, after they have provided the proper credentials such as user ID and password. When a resource is secured, WebSphere Application Server first attempts to authenticate the user. Authentication is done using certificates, or by prompting the user for a user ID and password. The prompting can be done with the basic authentication dialog, or by using a custom form.

In this scenario, we designed our own logon page named login.jsp. To configure its authentication prompt, select the **Pages** tab in the web.xml view. In the **Login** section, note that **Form** is pre-selected as the Authentication method. Also note that the name of the Login page is login.jsp. The Error page is displayed when the logon is unsuccessful. In this case, the application re-displays the login.jsp page.

Creating EAR files for iSeries WebSphere Application Server deployment

You need to create EAR files to deploy your application to the iSeries WebSphere Application Server. An EAR file is a standard Java Archive (JAR) file with an .ear extension. To create the EAR files:

1. Open Development Studio Client from the Start menu.
2. Switch to the Web perspective.
3. In the Navigator view, right-click **SVRetailStorEAR** and select **Export EAR file**.
4. Click **Browse** and navigate to a directory on your iSeries Integrated File System where you can keep the EAR files (you need to map your network drive to an iSeries IFS).
5. Enter SVRetailStorEAR.ear in the **File name** field and click **Open**.
6. Click **Finish**.
7. Repeat steps 3 to 6 for SVWholeSaleEAR.

Deploying the EAR files to iSeries WebSphere Application Server

Now that you have created the EAR files, you can deploy them to WebSphere Application Server.

1. Open the WebSphere Administrative Console.
2. Right-click **Enterprise Applications** and select **Install Enterprise Application**.
3. Select the **Install Application (*.ear)** radio button.
4. Click the upper **Browse** button (the lower is unavailable).
5. Navigate to the IFS directory where you exported the EAR files.
6. Select **SVRetailStorEAR.ear**.

7. Enter "RETAILSTOR" in the **Application name** field.
8. Click **Next** and enter your iSeries host user ID for the administrator's role.
9. Click **Next** repeatedly until you are at the page titled **Selecting Virtual Hosts for Web Modules**.
10. For all three Web modules, click **Select Virtual Host** and select your preferred virtual host from the drop-down list. (If you are not sure which one to select, use **default** or **default_host**.)
11. Click **Next**.
12. For all three Web modules click **Select Server** and select the server that you want to use. (If you are not sure which one to select, use **Default Server**.)
13. Click **Next**.
14. Click **Finish** and click **OK** in the dialog box.
15. Right-click **Enterprise Applications** again and select **Install Enterprise Application**.
16. Click the bottom **Browse** button (the upper one is unavailable).
17. Navigate to the IFS directory where you placed the EAR files.
18. Select **SVWholesaleEAR.ear**.
19. Enter "WHOLESALE" in the **Application name** field.
20. Click **Next** repeatedly, until **Finish** is enabled.
21. Click **Finish** and click **OK** on the dialog box.

Note: It might take a few minutes for a confirmation message to appear.

Running the application in WebSphere Application Server

To take on the role of the customer and run the retail store entry point, enter the following URL into a Web browser:

```
http://your iSeriesHostName:yourHTTPPortNumber/SV001585/shop.html
```

To take on the role of the administrator and run the WholeSale entry point, enter the following URL into a Web browser:

```
http://your iSeriesHostName:yourHTTPPortNumber/SV000618/index.html
```

If you do not know the HTTP port number, ask your WebSphere Application Server administrator.

Chapter 4. Building the application

Introduction

The entire scenario application is made up of five projects, named SV000501, SV000514, SV000618, SV001585, and SV001586. This chapter gives you step-by-step instructions on how to construct some of these projects, as if you were developing the application yourself. The step-by-step modules are intended for developers who are relatively new to application development, as well as to the Development Studio Client IDE. The advanced modules are designed for developers who are experienced in application development and more familiar with the IDE.

Note: Although there are five projects, there are only four modules because project SV001586 is just the Web Services component of project SV000514.

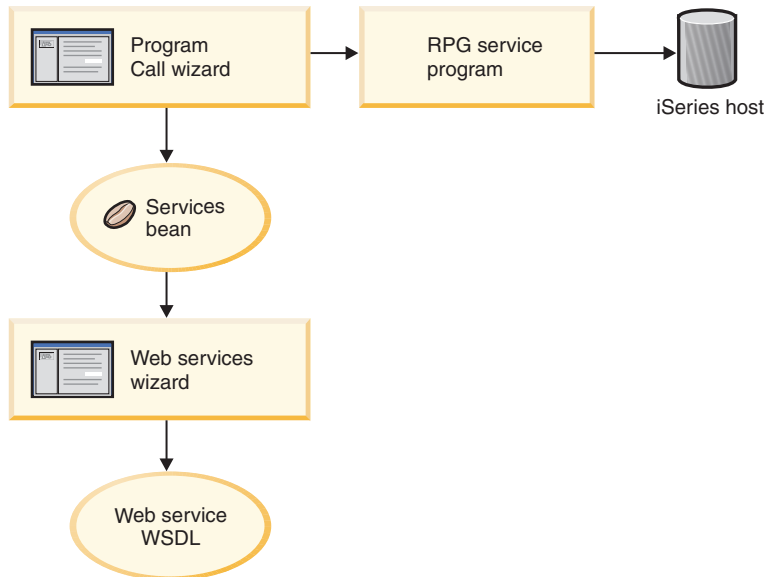
Before you begin

You can complete the exercises only if the following prerequisites are met. Prerequisites are discussed in more detail in Chapter 3, “Running the scenario” on page 7.

- You have TCP/IP access to an iSeries host.
- You have started the iSeries host servers with the command `STRTCPSVR *ALL`
- You have started the WebFacing server with the command `STRTCPSVR *WEBFACING`
- You have restored the `WHOLESALE` and `RETAILSTOR` libraries to your iSeries host.
- You have completed all the tasks in the previous chapter, (except the optional WebSphere Application Server tasks, which are not required to test the application in the workbench).

Step-by-step module 1: SV000514 – Create a Web service to return product prices

In this module, you need to create a Web service from an RPG program in the iSeries to display the product prices. First, you create an RPG service program with a procedure that can retrieve the cost of an item from the iSeries database, given an item number. You use the Program Call wizard from iSeries Java development tools to invoke the RPG program, and create a services bean. You then use the Web Services wizard to create a Web service, and use the generated sample to verify the Web service.



Creating a new Web project


The first step in creating this Web service is to create a new Web project to hold your information.

1. From the workbench IDE, switch to the Web perspective or open the Web perspective by clicking **Perspective > Open > Other > Web > OK**.
2. Click **File > New > Other > Web > Web project** and click **Next**.
3. In the Project name, enter Project514.
4. In the Enterprise Application project name, enter Project514EAR.
5. Leave all other defaults and click **Finish**.

Now, you can see that the Project514 and Project514EAR projects are added to your workspace in the Navigator view.

Defining the iSeries information

After you create the Web project, you need to define which iSeries the project uses to obtain information.

1. Click the **Project514** project in the Navigator view to select it.
2. Click the Specify Host Information icon  on the toolbar.
3. Enter the name of your iSeries host where the restored WHOLESale library resides, for example, PROD400.
4. Enter your user ID and password for your iSeries host.
5. Click **Add** to add a library to your library list.
6. In the Runtime library list field, erase newlib, enter WHOLESale.
7. Press Enter to save your addition.
8. Click **Finish**.

Creating the RPG service program

You want your application to be able to retrieve the price of an item, given the item number. This is handled by an RPG service program that contains a procedure called QryProdCost. The WHOLESale library contains an RPG service

program named CWWSSRV. This program contains a QryProdCost procedure that can take the item numbers as input, open the Inventory file within the WHOLESale library, retrieve the price from the inventory database, and return the price. To accommodate this, the interface has two parameters, one for the item number, and the other for the price. If the item number or price is not found, the RPG program returns a message to the interface.

To create this Web service, you use the Program Call wizard to create a Java bean that invokes the QryProdCost RPG procedure. Then, you use Web services to take the Java bean and enable RPG procedure as a Web service.

To create the Java bean:

1. Switch to the Web perspective.
2. Right-click **Project514** and select **New > Other**.
3. In the **New** window, select **iSeries > Java > Program Call Bean**.
4. Click **Next** to invoke the Program Call wizard.
5. In the **Java bean name** field underneath **Add Program**, enter **Inventory**.
6. In the **Program object** field, enter **CWWSSRV**, the name of the RPG service program.
7. In the **Library** field, enter **WHOLESale**.
8. From the **Program type** drop-down list, select ***SRVPGM**.
9. In the **Entry point** field, enter **QryProdCost**.
10. Click **OK** to add the program definition.

Creating the parameters and generate the Java bean

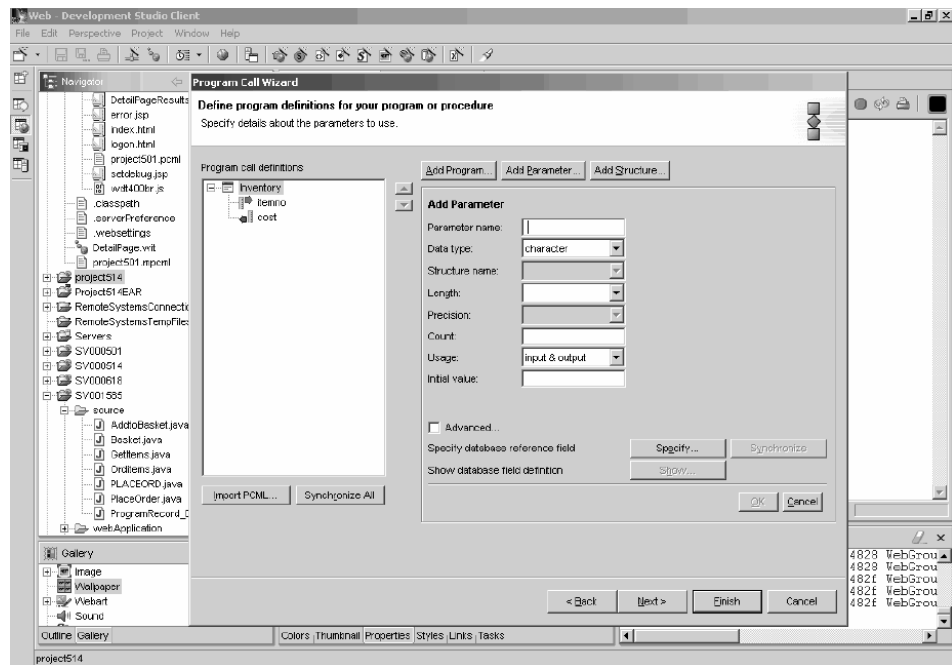
Now that the program is specified, you can add parameters. The CWWSSRV RPG program contains the following two parameters:

- Item number parameter - the program uses this number to find the item in the database
- Item cost parameter - the program uses this number to find the item's cost in the database

To add these parameters:

1. In the left panel of the Program Call wizard, click the **Inventory** program call definition to select it. This action re-populates the fields on the right side of the page.
2. Click **Add Parameter**.
3. In the **Parameter name** field, enter **itemno**.
4. From the **Data type** drop-down list, select **packed decimal**.
5. In the **Length** field, enter **5**.
6. In the **Precision** field, enter **0**.
7. From the **Usage** drop-down list, select **input**.
8. Click **OK** to add this parameter. In the left pane, note that **itemno** appears beneath **Inventory**. Now you are ready to add the second parameter.
9. In the **Parameter name** field, enter **cost**.
10. From the **Data type** drop-down list, select **packed decimal**.
11. In the **Length** field, enter **7**.
12. In the **Precision** field, enter **2**.
13. From the **Usage** drop-down list, select **output**.

- Click **OK** to add this parameter. In the left pane, notice that **cost** appears beneath **Inventory**. At this point, the wizard should look like this. Note that the icon to the left of the a parameter shows if it is of type input, input & output, or output:



- Click **Next**.
- In the **Package** field, enter **scenario** as a package name. Leave the defaults in the other fields.
- Clear the **Java Application** check box.

Note: Review the list of files under "These files will be generated by the wizard" and note that the name of the generated Java bean will be **InventoryServices.java**.

- Click **Finish** to generate the files.

Making a Web service from the Java bean

After creating the Java bean that invokes the RPG program, the next step is to convert the bean into a Web service so that other programs can access the same RPG program over the Internet.

To create the Web Service, use the Web Services wizard to create the WSDL files that are distributed to users who need to use the Web Service. To create the files:

- In the Navigator view of the Web perspective, expand **Project514 > source > scenario**.
- Right-click **InventoryServices.java** and select **New > Other**.
- In the New window, click **Web Services > Web Service** and then click **Next**.
- Select the **Generate a sample** check box from the **Wizard defaults** section of the window, and leave the defaults for the other check boxes.
- Click **Next** three times, until you reach the **Web Service Java Bean Methods** page.
- Select the check box beside **scenario.InventoryResult Inventory (java.math.BigDecimal itemno)** and deselect any other check boxes. This method retrieves an item's cost given the product number.

7. Click **Finish**.

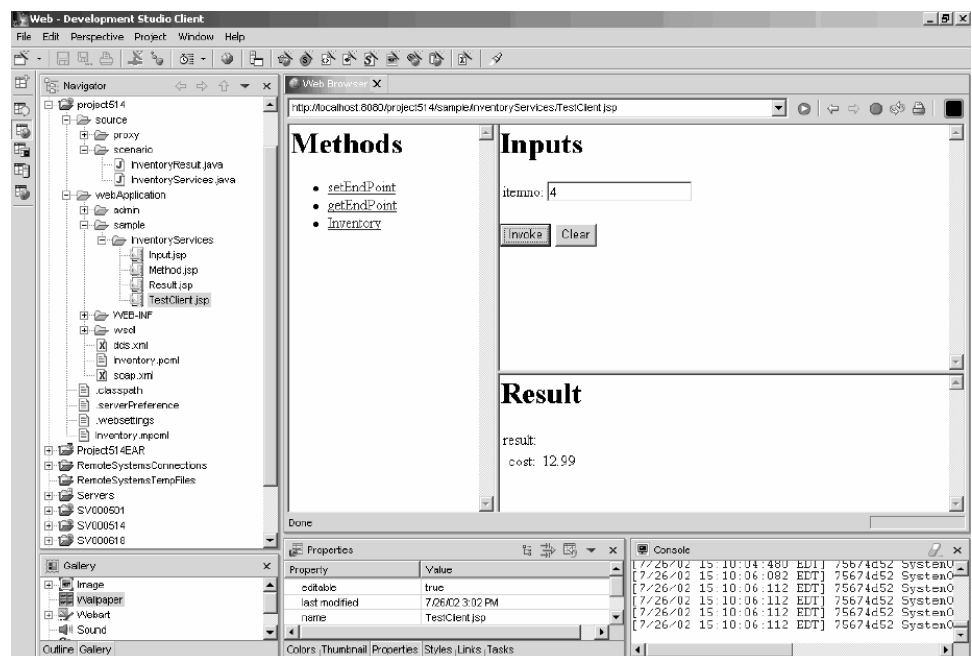
Note: The Web Service generation creates a **wsdl** folder under the webApplication folder of your workspace. This folder contains two WSDL files and an XSD file that you would distribute to the users of your Web service.

Testing the sample

When you created the Web service, one of the instructions was to request that a sample be generated. Because that option was selected, the Web Services wizard created test pages that you can use to test the Web Service. To test the sample:

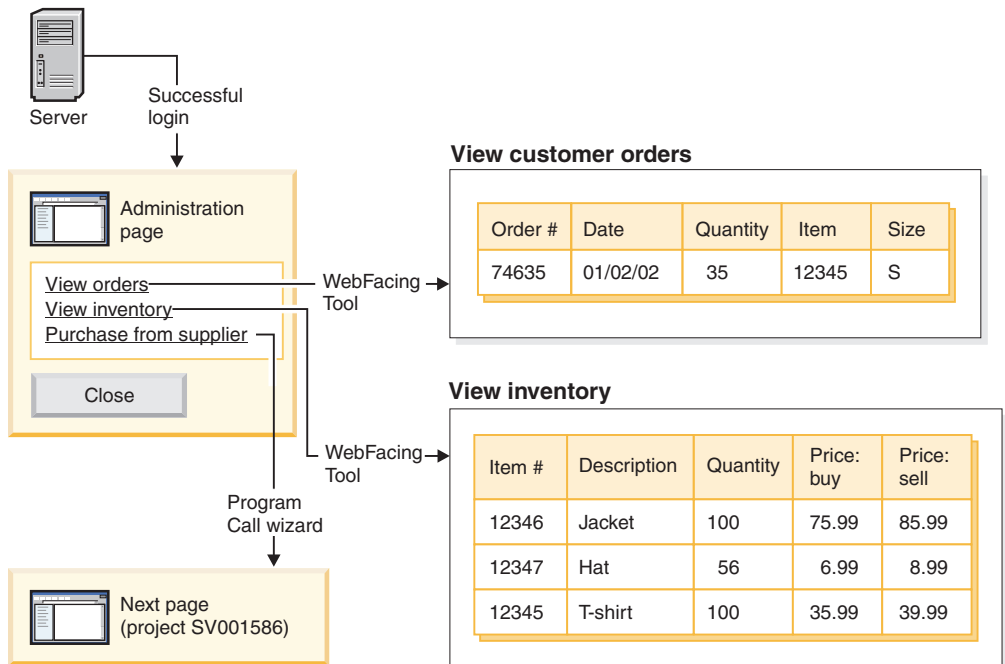
1. In the **Navigator** view, expand **Project514 > webApplication > sample > InventoryServices**.
2. Right-click **TestClient.jsp** and select **Run on Server** to open a browser in the IDE and launch the sample.
3. When the Web page is generated, click the **Inventory** method on the left.
4. In the **itemno** field, enter 4 and click **Invoke**.
5. Verify that the output returned is:

```
result:  
cost: 12.99
```



Step-by-step module 2: SV000501 – Creating the administrator interface to view inventory and order

In this module, you create an interface using the IBM WebFacing Tool that displays an administrator's operation options. After a successful login to the iSeries host, the interface displays the available inventory and you can check existing orders. As an administrator, you can then purchase merchandise from the wholesale supplier.



In this project, you work with two programs and two display files restored to your iSeries host. The program names are ViewInventory and ViewOrder. The programs use the following two display files: ORDERDSP and QUERY. These two files contain Web settings customized for the JSP files generated by the WebFacing Tool, and are used for images and hyperlinks. The image Web setting let you use a field's content to generate the image file's name and display the image in the JSP file. The hyperlink Web setting lets you invoke another Web application when you click the image in the JSP file. You can use either CODE Editor or CODE Designer to check the display file source code, to determine how to write the Web settings.

Creating a WebFacing project

The first thing you need to do is create a WebFacing project and specify the relevant CL commands. To create the WebFacing project:

1. In the workbench, switch to the WebFacing perspective by clicking one of the perspective icons down the left side of the workspace, or by clicking **Perspective > Open > Other > WebFacing** and then clicking **OK**.
2. Create a new WebFacing project by clicking **File > New > WebFacing project**.
3. Name the project project501.
4. In the **Enterprise Application project name** field, enter SVRetailStorEAR and click **Next**.
5. Beside the **Connection** field, click **New**.
6. In the **Connection name** and **Host name** fields, enter the name of your iSeries host to which you restored the WHOLESale and RETAILSTOR libraries, for example, PROD400.
7. In the **Default User ID** field, enter your user ID on the iSeries host you specified and click **Finish**.
8. Click **Refresh list** and enter your password in the pop-up dialog to refresh the list.
9. In the generated list of libraries, expand **RETAILSTOR**.
10. Click **QDDSSRC** and then click the right arrow to move the files over.

11. Click **Next** twice, until you arrive at the **Specify CL commands** page.
12. In the **Command label** field, enter View inventory.
13. In the **CL command** field, enter call call viewinvent.
14. Select the **Sign on with specified values** radio button.
15. Click **Add** and note the addition in the list at the bottom of the window.
16. In the **Command label** field, delete the old value and enter View orders.
17. In the **CL command** field, delete the old value and enter call vieworder.
18. Select the **Prompt for signon** radio button.
19. Click **Add** and note the addition in the list at the bottom of the window and click **Next**.
20. On the **Choose a Web style** window, scroll through the styles to see what is available. Because we need to retrieve information from this style later on, for this module select **avenue** and click **Next**.
21. Select the **No. I only want to create the project now** check box and click **Finish**.

You now need to copy the image files from SV000501 to project501 so that project501 displays correctly:

1. Switch to the Web perspective.
2. In the Navigator view, expand **SV000501 > webApplication > images**.
3. Right-click **generated** and select **Copy**.
4. In the folder selection dialog, expand **project501 > webApplication**.
5. Click **images** to select the folder and click **OK**.

Converting the DDS source

Now that you have created the project, you can convert the DDS display files into JSP files that will display on your Web page. When you convert your DDS display files, JSPs and Java beans are generated for you that substitute for the DDS code and make Web access possible. The tool creates three Java beans and two JSPs per record format; the Java beans hold the data for the record format, or control its appearance, and the JSP file displays the Web version of the screen, prompts for data, and handles input errors. The wizard generates an application home page to launch the Web-enabled version of your program.

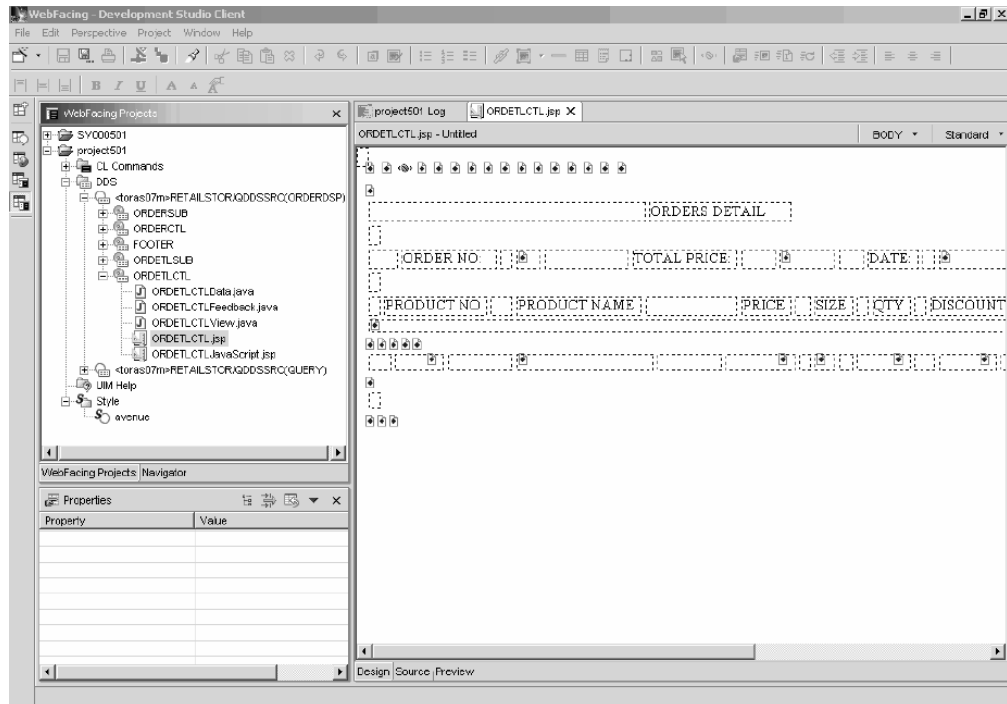
To convert the DDS source:

1. Switch to the WebFacing perspective.
2. In the WebFacing projects view, expand **project501 > DDS**.
3. Select *iSeriesHost* > **RETAILSTOR/QDDSSRC(ORDERDSP)** and *iSeriesHost* > **RETAILSTOR/QDDSSRC(QUERY)** by holding down the Shift key and clicking both of them.
4. Right-click and select **Convert** to begin conversion.

When conversion is complete, a conversion log displays with information about any error that occurred. To see one of your application pages:

1. Expand **project501 > DDS > iSeriesHost > RETAILSTOR/QDDSSRC(ORDERDSP) > ORDERCTL**.
2. Double-click **ORDETLCCTL.jsp** to display the page in Page Designer.
3. Observe the different views of your JSP file from the **Design**, **Source**, and **Preview** tabs.

Here is what your workspace should look like from the Design view:



Configuring UTF-8 support in the workbench

IBM WebFacing Tool applications have multiple language support. Because languages use different character sets, data streams between the browser and the WebSphere Application Server are UTF-8 encoded. For the IBM WebFacing Tool to function correctly, you need to configure UTF-8 support in the wfapp.properties file in the workbench.

To configure UTF-8 support:

1. Switch to the Server perspective by clicking **Perspective > Open > Other > Server** and then clicking **OK**.
2. In the Navigator view, expand **Servers** folder.
3. Double-click **defaultInstance.wsi** to open it in the default editor.
4. Click the **Environment** tab and click the **Add** button.
5. In the **Name** field, enter `client.encoding.override`.
6. In the **Value** field, enter `UTF-8` and click **OK**.
7. Click the save icon or click **File > Save WebSphere v4.0 Test Environment**.

Configuring UTF-8 support for WebSphere Application Server

(Optional) If you want to deploy your iSeries application to WebSphere Application Server, you also need to configure UTF-8 support in WebSphere Application Server as well as the workbench.

To configure UTF-8 support in WebSphere Application Server 4.0 Advanced Edition:

1. Start the WebSphere Administrative Console.
2. Expand the **Nodes** icon and expand **Node name > Application servers > Default server**.
3. Select the **JVM Settings** tab and click the **Advanced JVM settings** button to open the **Advanced JVM settings** dialog.

4. In the **Command line arguments** field, enter:
`-Dclient.encoding.override=UTF-8`
5. Click **OK** and click **Apply** under the **JVM settings** tab.
6. For this change to go into effect for your WebSphere applications, stop the default server and then restart it. To stop the server, right-click **Default server** and select **Stop**. After this process is complete, right-click **Default server** and select **Start**.

To configure UTF-8 support in WebSphere Application Server 4.0 Advanced Single Server Edition:

1. Start the WebSphere Administrative Console.
2. In the browser-based Administrative Console, expand the **Nodes** icon and expand **Node name > Application servers > Default server > Process definition > JVM settings**.
3. Scroll to the Advanced Settings section of the JVM Settings page and click the System Properties link. The System Properties page is displayed.
4. Click **New** to add a new System Property.
5. In the **Name** field, enter `client.encoding.override`.
6. In the **Value** field, enter UTF-8.
7. Click **OK**. If you receive a **Configuration needs to be saved** message with a link at the top of the **JVM settings** page, click the link to go to the **Save configuration** page. Select **Save** and then click **OK**.
8. For this change to go into effect for your WebSphere applications, stop the application server and then restart it. How you stop and start the application server can vary depending on the platform you have installed WebSphere Application Server on. Refer to the WebSphere Application Server documentation for your platform for information on stopping and starting the application server.

Creating a style sheet (optional)

If you want to integrate additional pages with a cascading style sheet (CSS), you need to customize either the style of the WebFacing project or the cascading style sheet to make them look alike. After you finish customizing the style sheet, you can use the Web Interaction wizard to create a detailed Web page using the style sheet to display item details such as price and color. For the purposes of this exercise, you will incorporate the `DetailPageResults.jsp` style sheet from the SV000501 project, detailed in “Creating the Web interaction using iSeries Web development tools” on page 28. For future reference, you can manually customize the CSS file in two ways. (Optional) For the first method:

1. Switch to the WebFacing perspective and expand **SV000501 > Style**.
2. Right-click SV000501 and select **Save as**.
3. Give the style a name that you will recognize, and click **OK**.
4. Right-click your new style and select **Properties**.
5. Alter the properties and click **OK**.

(Optional) For the second method, which gives you more control over fine details:

1. Switch to the WebFacing perspective and expand **SV000501 > webApplication > styles > apparea**.
2. Double-click **apparea.css** to open it in a text editor and change its properties.
3. Click the save icon or click **File > Save apparea.css**.


4. Expand **SV000501 > webApplication > styles > chrome** and double-click **gradient.css** to open it in a text editor and change its properties.
5. Click the save icon or click **File > Save gradient.css**.

Creating the Web interaction using iSeries Web development tools

Now that you have converted your DDS source, you can use the Web Interaction wizard to create the JSP files and servlets for your Web page. This Web service will let the administrator view wholesale inventory and check on existing orders. When working with this Web interaction, you will:

- define the host information
- copy over the correct style sheet
- create the interaction
- add programs and parameters to the interaction
- change the usage of the parameters

First, you need to define the host information:

1. Switch to the Web perspective.
2. In the Navigator view, click **project501** to select it.
3. Click the Specify Host Information icon  on the toolbar.
4. For the **iSeries host name** field, enter the name of your iSeries host, for example, **PROD400**.
5. Enter your user ID and password for the iSeries host and click **Add**.
6. Click the first line under **Runtime library list**, delete **newlib** and enter **RETAILSTOR**.
7. Click Enter to save the addition.
8. Click **Finish**.

If you decided not to create your own style sheet, explained in “Creating a style sheet (optional)” on page 27, then before you create the Web interaction, you need to copy over the correct style sheet so that project501 displays the proper JSP file format:

1. In the Navigator view, expand **SV000501 > webApplication**.
2. Right-click **DetailPageResults.jsp** and select **Copy**.
3. In the Folder Selection dialog, expand **project501**.
4. Click **webApplication** to select it.
5. Click **OK**.

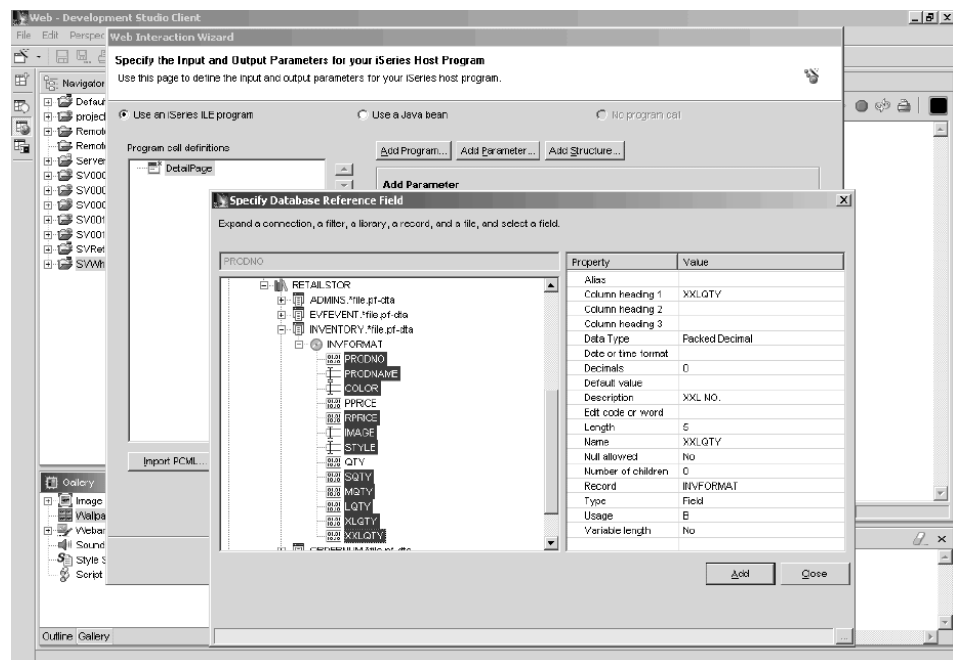
Now you can create the Web interaction:

1. Click **File > New > Other > Web > Web Interaction** and click **Next** to invoke the Web Interaction wizard.
2. Beside **Destination folder**, click **Browse**.
3. Expand **project501**, click **webApplication**, and click **OK**.
4. For the **Web Interaction name** field, enter **DetailPage** and click **Next**.
5. Select the **Generate Input JSP** radio button.
6. Select the **Use output pages** radio button and click **Add**.

7. From the **Output JSP** dialog, expand **webApplication**, select **DetailPageResults.jsp** and click **OK** so that your output page displays correctly.
8. Click **Next**.

Now, you can add programs and parameters to your interaction. You need to add 11 parameters to the same program. Instead of adding each parameter with its individual values manually, you can add them in a slightly faster way:

1. Select **Use an iSeries ILE program** if it is not already selected by default.
2. Click **Add Program**.
3. For the **Program alias** field, enter **DetailPage**.
4. For the **Program object** field, click **Browse**.
 - a. Expand *iSerieshost* > *LIBL > RETAILSTOR your library list and then expand **RETAILSTOR**.
 - b. Click **DETAILPAGE.*pgm.rpgle** and click **OK**.
5. Back in the Web Interaction wizard, click **OK**.
6. In the Program call definitions section in the left pane of the wizard, click **DetailPage** once to select it.
7. Click **Add Parameter**.
8. Beside **Specify database reference field**, click **Specify**.
9. Expand *iSerieshost* > *LIBL > RETAILSTOR > INVENTORY.*file.pf-dta > **INVFORMAT** to display a list of 13 parameters. You need to add the following 11 by clicking each one once and clicking **Add**: **PRODNO**, **PRODNAME**, **COLOR**, **RPRICE**, **IMAGE**, **STYLE**, **SQTY**, **MQTY**, **LQTY**, **XLQTY**, **XXLQTY**. Or, you can hold the CTRL key, click all of them, and then click **Add**.

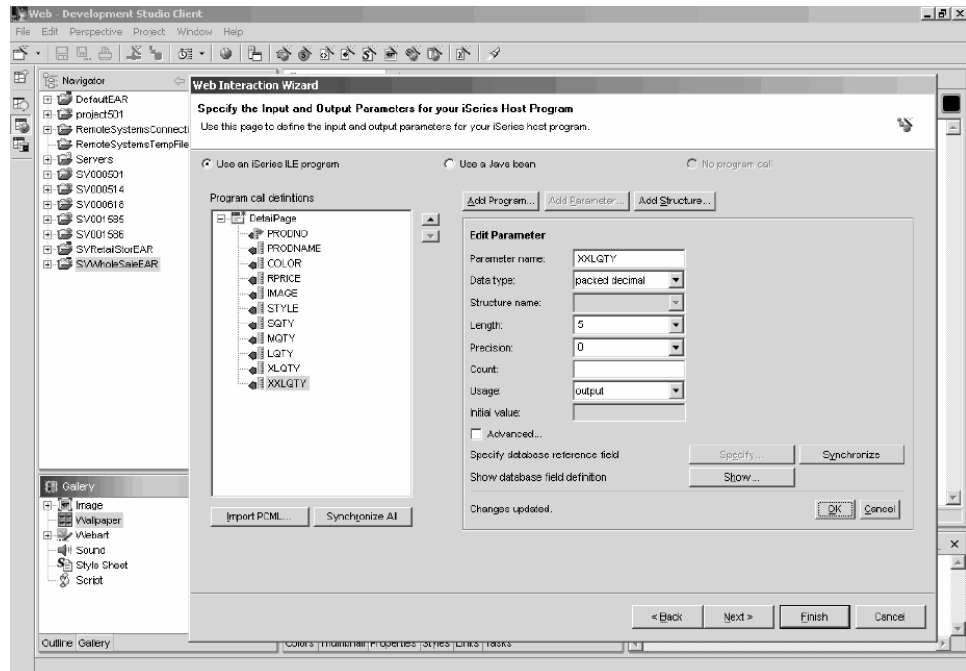


10. Click **Close**.

You now need to change the usage 10 parameters (all parameters except **PRODNO**) to "output".

1. Still in the Web Interaction wizard, click **PRODNAME** to select it.

- In the **Usage** combo box, switch the selected value to **output** and click **OK**. Repeat this step (with the previous step) for all of the parameters except the first one (PRODNO). This is what your workspace should look like once you have adjusted all of the parameters. Note that the icon to the left of the a parameter shows if it is of type input, input & output, or output.



- Click **Next** to check the **Input Form** preview.
- Click **Finish** to create the Web interaction.
- Select **Yes to all** if you receive any messages.

Link the project to a Web interaction

Now that you have created a Web interaction JSP file that uses input and output parameters, you need to customize the JSP file so that it also works with the WebFacing component. You need to enter code to create a link from your WebFacing application to invoke this Web interaction. To do this you need to add a JavaScript function in the `webface.js` file so that you can call the `DetailPageServlet` servlet with the `PRODNO` parameter in a new window.

To create the link:

- In the Navigator view, expand **project501 > webApplication > ClientScript**.
- Double-click **webface.js** to open it.
- Scroll to the bottom of the file and enter the following lines:

```
var mywindow
function next(app)
{
mywindow = window.open(app,"Details","RESIZABLE=YES, HEIGHT=700, WIDTH=800");
}
```

- Click the save icon or click **File > Save webface.js**.

If you want to create an application similar to `SV000501` in the future, you also need to change a Web setting in your DDS source to enable the image you added and the close-window link for the JavaScript function. The RPG code included in this application is altered to show the change, however, you would need to

manually make the change in future applications. After changing the Web settings, you would need to reconvert the DDS source. You can check the code to duplicate the result.

To view the DDS source:

1. Switch to the WebFacing perspective.
2. In the WebFacing projects view, expand **project501 > DDS**.
3. Right-click **<iSerieshost > RETAILSTOR/QDDSSRC(QUERY)** and select **Open With > CODE Designer**.
4. After CODE Designer opens, expand **SCREEN1 > ITEMSUB**.
5. Click **IMAGESRC**.
6. Click the **Source** tab.
7. Click the **Web Settings** tab on the bottom-right part of the window.

Note: Check the Web setting properties such as the width in pixels and the file name. In future, you have to make the same changes to your DDS source and then re-convert the source.

8. Note the following lines in the source:

```
A   PRODNO  R   0 5 6
A   PRODNAME R   0 5 16
A   IMAGESRC 19A 0 5 33
A*%WB 13 FLD 100|100|&{IMAGESRC}
A*%WB 12 FLD 1 javascript:next('/SV000501/DetailPageServlet?PRODNO=&{PRODNO}')
```

9. You need to make one of more alterations in the last line of this code sample. This line is defined in the field in the Web Settings view. To change values in the line, you need to change them in this field:
 - a. Change SV000501 to project501.
 - b. Make sure that a single quotation mark ' comes after javascript:next(
 - c. Make sure that a single quotation mark plus an ending parenthesis ') come after {PRODNO} .
 - d. Save the file by clicking the save icon or by clicking **File > Save**.

For more information on how to work with DDS Source, switch to the Help perspective of the workbench and see the IBM WebFacing Tool documentation.

Restarting the server

Before you test the interface, you need to adjust module visibility and restart the server.

Module visibility specifies the classloader isolation mode to use for the application server. You need to adjust this module visibility to use one classloader per module, rather than one for the entire server or enterprise application. To adjust module visibility:

1. Switch to the Server perspective.
2. In the Navigator view, expand **Servers > defaultConfiguration.wsc**.
3. Double-click **server-cfg.xml** to open it in the right pane of the workspace.
4. Make sure you are in the General tab. Click the drop-down list beside **Module visibility** and select **MODULE**.
5. Click the save icon or click **File > Save WebSphere Administrative Domain**.

To restart the server:

1. Click the **Servers** tab at the bottom of the screen.
2. Right-click WebSphere v4.0 Test Environment and select **Start** or **Restart** (whichever one is available).

Testing the interface

You have now completed the necessary steps to create an interface for viewing orders and viewing inventory. To test your interface:

1. Switch to the Web perspective.
2. Expand **project501 > webApplication**.
3. Right-click **index.html** and select **Run on Server**.
4. Click **View orders – Launch in main browser window** to invoke the administrator application. After logging in with your iSeries user ID and password (for the iSeries host you used while developing this project), you will be directed to the following page:

The screenshot shows the IBM WebSphere Development Studio Client interface. The main browser window displays a web page titled "View orders" with a table of orders. The table has the following data:

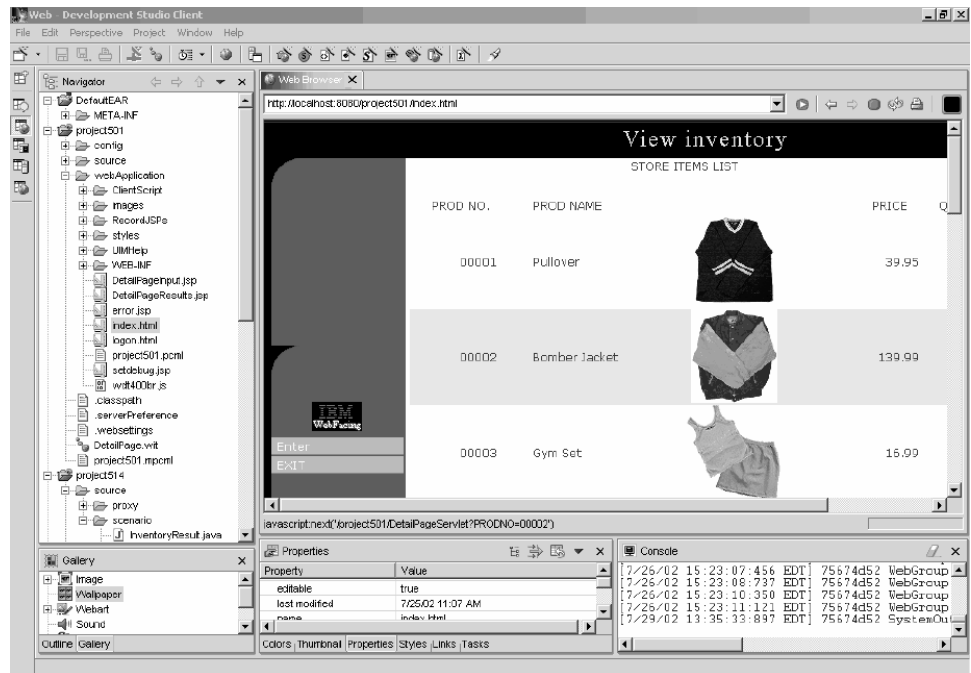
SELECT	ORDER NO	TOTAL PRICE	DATE	STATUS
<input type="checkbox"/>	000043	49.79	2002-07-26	0
<input type="checkbox"/>	000042	164.79	2002-07-26	0
<input type="checkbox"/>	000041	49.74	2002-07-19	0
<input type="checkbox"/>	000040	49.78	2002-07-19	0
<input type="checkbox"/>	000039	45.18	2002-07-19	0
<input type="checkbox"/>	000038	64.29	2002-07-19	0
<input type="checkbox"/>	000037	49.74	2002-07-17	0
<input type="checkbox"/>	000035	34.84	2002-07-15	0
<input type="checkbox"/>	000035	49.74	2002-07-11	0
<input type="checkbox"/>	000034	49.74	2002-07-11	0
<input type="checkbox"/>	000033	164.78	2002-07-10	0
<input type="checkbox"/>	000032	95.69	2002-07-10	0
<input type="checkbox"/>	000031	95.69	2002-07-10	0
<input type="checkbox"/>	000030	80.83	2002-07-10	0

The IDE interface also shows a Navigator window with the project structure, a Properties window, and a Console window displaying system logs.

Try entering X beside any of the product numbers to see details about that product.

5. Click the back arrow to get to the index.html page, and click **View inventory – Launch in main browser window** to invoke the customer application, and you

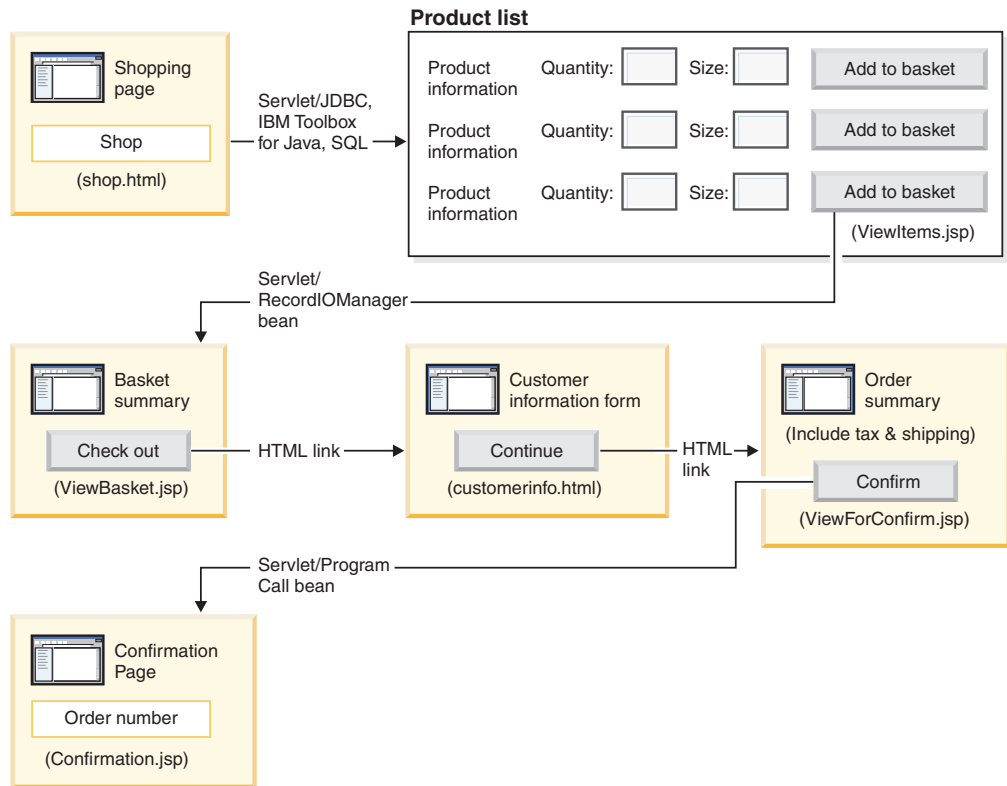
will be directed to the following page:



Try clicking an image to display details about a particular item.

Advanced module 1: SV001585 – Create HTML, servlets, and JSP files that place customer orders on an iSeries host

This project is intended for users with good Java programming knowledge and some knowledge of iSeries data management and RPG. The project shows you how to work with the IBM Toolbox for iSeries data access classes, RecordIOManager bean, and the Program call bean to create HTML code, servlets, and JSPs that place customer orders on an iSeries host. As a user, you go from the shopping page to view the products available and add items to your basket. Once you are satisfied with all of the items you have selected (in your Basket summary) you click a check out button and are directed to a customer information form. Once the form is complete, the project returns an order summary with a confirmation button, which takes you to a confirmation page that displays your order and order number.



Summary of high level steps

Because this is an advanced module, the instructions do not take you through each step of creating the project, but outline the iSeries-specific development steps taken to create such a project. These are the high-level steps:

1. Write an HTML shopping Web page.
2. Write a servlet that populates a Java bean (using JDBC and SQL) with available items for the customer to purchase.
3. Write a JSP file to view items for sale and allow customers to enter the quantity and size of the desired item, and to select the item by clicking an **Add to basket** button
4. Using the RecordIOManager bean, write a servlet invoked from clicking the **Add to basket** button that updates the iSeries INVENTORY database by subtracting the quantity and size required for the item, and adding this selection to a Java bean called "basket". The servlet then redirects the response to ViewBasket.jsp if the operation is successful. If the operation is unsuccessful, the servlet displays an error page.
5. Write an HTML form for the customer to enter personal information.
6. Write a JSP file purchase confirmation page that displays basket contents, taxes, plus shipping and handling charges. The page also must contain a confirmation button for the customer.
7. Write a servlet that is called when the customer clicks the confirmation button, which uses a Java bean created with the iSeries Program call wizard. One of the Java bean's methods calls an RPG program to create a new order in the ORDERS database on the iSeries host corresponding to the content of the customer's basket. The servlet then returns an order number, places the Java bean on the Web application's session, and loads an order confirmation JSP file containing the order number.

Creating the Web page, servlets, and JSP files

To construct the components of project SV001585:

1. Create a web project.
2. Write a shop.html page with Page Designer containing a link that invokes a GetItems servlet
3. Import the jt400.jar file for the iSeries Toolbox for Java classes into the Web project *lib* folder. You can find this jar file in `x:\wdsc\wssd\plugins\com.ibm.etools.iseries.toolbox\runtime` where *x* is the directory in which you installed Development Studio Client.

Note: See `GetItems.java` and `ViewItems.jsp` in the SV001585 project to see the main iSeries Toolbox for Java JDBC and SQL related parts, for the servlet and JSP. In the Navigator view of the Web perspective, you can find `GetItems.java` by expanding **SV001585 > source**, and you can find `ViewItems.jsp` by expanding **SV001585 > webApplication**.

4. Write a `GetItems` servlet that uses the iSeries Toolbox for Java JDBC and SQL to retrieve clothing items from the `iSeriesINVENTORY` database that:
 - a. Places the `ResultSet` bean containing the SQL query result on the session
 - b. Redirects the request to `ViewItems.jsp`

Code sample for `GetItems.jsp`:

```
public void init() {
    .
    .
    .
    // Load the IBM Toolbox for Java JDBC driver.
    DriverManager.registerDriver(new com.ibm.as400.access.AS400JDBCdriver());
    // Note that we have retrieved the as400 name, userid, and password from
    // web.xml file using and xml parser.
    as400conn =
        DriverManager.getConnection(
            "jdbc:as400://" + as400 + ";naming=sql;errors=full",
            userid,
            password);

    dmd = as400conn.getMetaData();
    .
    .
}

public void service(HttpServletRequest request, HttpServletResponse response){
    .
    .
    .
    Statement select =
        as400conn.createStatement(
            ResultSet.TYPE_SCROLL_INSENSITIVE,
            ResultSet.CONCUR_READ_ONLY);
    ResultSet rs =
        select.executeQuery(
            "SELECT PRODNO, PRODNAME, RPRICE, IMAGE FROM "
            + retailLibrary
            + dmd.getCatalogSeparator()
            + inventoryFile);

    HttpSession session = request.getSession(true);
    session.setAttribute("resultset", rs);

    response.sendRedirect("/ViewItems.jsp");
}
```

```

        .
        .
        .
    }
}
5. Write a ViewItems JSP file that retrieves the clothing items from the ResultSet
bean obtained in the previous step, to display the clothing items in a table
format. The JSP file should also include a form for each item that you can use
to select size and quantity, and then add the item to your basket. You can use
Page Designer in iSeries Web development tools to write the JSP. More
specifically, you can lay out the page in the Design view, and add appropriate
code in the Source view. Code sample for ViewItems.jsp:
<!--Getting the ResultSet Object from the session--><%
    int columnCount = 0;
    ResultSet rs = (ResultSet)session.getAttribute("resultset");
    if(rs !=null)
%>
<%
{
    rs.beforeFirst();
        ResultSetMetaData rsmd = rs.getMetaData ();
        columnCount = rsmd.getColumnCount ();

%>
<TABLE border="1">
<TBODY>
<TR>
    <TD>Product ID</TD>
    <TD>Name</TD>
    <TD width="551">Price</TD>
    <TD colspan="2"></TD>
</TR>
<%while (rs.next ()) {
<TR>
<!--Creating a form for this row (or this item)-->
<FORM name="myform" action="/SV001585/AddtoBasket" onsubmit="return errorChecking(this);">

<!--Getting each column data from this row of ResultSet object-->
<!--Process data is a user defined method to modify the data for display if needed-->
        <%
            for (int i = 1; i <= columnCount; ++i){
                String value = rs.getString(i);
                if (rs.wasNull ())
                    value = "<null>";
                else{
                    if(i==1)
                        prodID=value;
                    value = processData(i,value);

                }
            }
        %>
        <TD><%=value%></TD>
        <%
            }
        %>

<!--Creating quantity input field and size drop down menu-->
<!--Note that we are using product id as the name of the field-->
        <TD width="290">Quantity
        <INPUT size="5" type="text" name='<%=prodID+"Q"%>' ><BR>
        Size <SELECT name='<%=prodID+"S"%>' >
            <OPTION value="s" selected>Small</OPTION>
            <OPTION value="m" selected>Medium</OPTION>
            <OPTION value="l" selected>Large</OPTION>
            <OPTION value="XL" selected>Extra Large</OPTION>

```

```

        <OPTION value="XXL" selected>Extra Extra Large</OPTION>
    </SELECT>
</TD>
<TD><INPUT type="image" name="submit" src="images/Add_to_basket.gif"></TD>
</FORM>
</TR>
<%
}
%>
</TBODY>
</TABLE>
<%
}

```

6. Import the iSeries Java development tools iseriesut.jar file into the *lib* folder for your your Web project. You can find this JAR file in `x:\wdsc\wssd\plugins\com.ibm.etools.iseries.toolbox\runtime`, where *x* is the directory in which you installed the product. See `AddtoBasket.java` and `ViewBasket.jsp` in the SV001585 project to see the implementation. In the Navigator view of the Web perspective, you can find `AddtoBasket.java` by expanding **SV001585 > source**, and you can find `ViewBasket.jsp` by expanding **SV001585 > webApplication**.
7. Use the RecordIOManager bean from iSeries Java development tools to write an `AddtoBasket` servlet called by the **Add to basket** button, which updates the iSeries INVENTORY database by subtracting the quantity requested by the customer and adding the items to a Basket Java bean in the session. `AddtoBasket.jsp` code sample:

```

public class AddtoBasket extends HttpServlet {

    //Inner class of AddtoBasket
    public class MyRecordIOManager extends RecordIOManager {
        .
        .
        .
        public MyRecordIOManager(
            String hostInfo1,
            String hostInfo2,
            String hostInfo3,
            String file,
            String lib)throws Exception{
            super(hostInfo1, hostInfo2,hostInfo3,file,lib);
            setFileType(RecordIOManager.FILEACCESS_KEYED);
            setCommitLevel(RecordIOManager.COMMITLOCKLEVEL_ALL);
            //journal has the same name as the database file
            setJournal(file);
            //journal is in the same library as the database file
            setJournalLibrary(lib);
        }
        .
        .
        .
        public synchronized String updatedDBFile(
            String id,
            String size,
            String quantity
        ) {
            .
            .
            .
            //opening the file
            try {
                if (openFile()) {
                    record = readRecord(key);
                    quantityAvailable = ((BigDecimal)
                        record.getValueAt(0,sizeColumn)).intValue();
                }
            }
        }
    }
}

```

```

totalQuantityAvailable = ((BigDecimal)
    record.getValueAt(0, 8)).intValue();
if (quantityRequested <= quantityAvailable) {
    newQuantity =
        new BigDecimal(quantityAvailable - quantityRequested);
    totalNewQuantity =
        new BigDecimal(totalQuantityAvailable - quantityRequested);
    record.setValueAt(newQuantity, 0, sizeColumn);
    record.setValueAt(totalNewQuantity, 0, 8);
    // Note that we update the record but we don't commit
    // in case the customer decides to empty the basket in which
    // case we call the rollBack method
    updateRecord(record);
    status = success;
} else {
    status = notEnough;
}
} else
    status = accessError;
} catch (Exception e) {
    e.printStackTrace();
    status = accessError;
}

//closing the file and adding
try {
    closeFile();
} catch (Exception e) {
    //in case of error rollback
    try {
        rollback();
    } catch (Exception e1) {
        e1.printStackTrace();
    }
    status = accessError;
}

return status;
}
}

//init method of AddtoBasket servlet
public void init() {
    hostInfo = GetItems.getHostInfo();
}

public void doGet(HttpServletRequest req, HttpServletResponse res) {
    .
    .
    .
    Basket basket = (Basket) session.getAttribute("basket");
    MyRecordIOManager recIO = (MyRecordIOManager) session.getAttribute("recIO");
    if (basket == null) {
        basket = new Basket();
        session.setAttribute("basket", basket);
    }

    if(recIO == null){
        if (recIO == null) {
            try {
                recIO =
                    new MyRecordIOManager(
                        hostInfo[0],
                        hostInfo[1],
                        hostInfo[2],
                        GetItems.getInventoryFile()),

```


- d. The last window of the wizard gives you the option to create a bean for a Java application, a Web service, or both. In this project, you only need to create one for a Java application.

Note: The PLACEODR service program takes an array of structures and places each element into one record of an ORDERS database, generating one order number as output for each array.

12. Write a PlaceOrder servlet invoked by the ViewForConfirm.jsp **Confirm** button.
- The servlet uses the bean generated by the iSeries Program Call wizard to access an iSeries host and place orders in the RETAILSTOR library's ORDERS database.
 - Orders are the items in the basket, sent to the ILE program as an array of structures.
 - Each structure of this array is an item in the basket.
 - The PLACEORD RPG service program called by the bean returns the order number as an output parameter and places it on the session.

The following code segment shows how the PlaceOrder servlet uses the PLACEORD bean:

```
.
.
.
public void init() throws ServletException {
    hostInfo = GetItems.getHostInfo();
    super.init();

    try {
        /* creating an instance of the PLACEORD bean created
           by iSeries Program Call Bean wizard */
        orderBean = new PLACEORD();
        orderBean.setConnectionData(hostInfo[0], hostInfo[1], hostInfo[2]);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

.
.
.

public void doPost(HttpServletRequest request, HttpServletResponse response) {

    ...

    PLACEORD.Orditems_Struct inputStruct = null;

    // retrieving the order items from the basket
    Basket basket = (Basket) request.getSession().getAttribute("basket");
    AddtoBasket.MyRecordIOManager recIO = (AddtoBasket.MyRecordIOManager)
        request.getSession().getAttribute("recIO");
    if (basket == null || basket.size() == 0 || recIO == null) {
        try {
            response.sendRedirect("errorPage.html");
        } catch (IOException e) {
            e.printStackTrace();
        }
    } else {
        items = basket.elements();
        // setting array of structure elements
        while (items.hasMoreElements()) {
            item = (String[]) items.nextElement();
        }
    }
}
```

```

        inputStruct = orderBean.getOrdItemAr(j);
        inputStruct.setItemNo(new BigDecimal(item[0]));
        inputStruct.setQuantity(new BigDecimal(item[1]));
        inputStruct.setSizeOrd(item[2]);
        j = j + 1;
    }
    // setting the rest of the array elements to dummy values
    for (int i = j - 1; i < 100; i++) {
        inputStruct = orderBean.getOrdItemAr(i);
        inputStruct.setItemNo(new BigDecimal(0));
        inputStruct.setQuantity(new BigDecimal(0));
        inputStruct.setSizeOrd("s");
    }

    // setting the other two input parameters of the bean
    orderBean.setNumOfItems(new BigDecimal(j));
    orderBean.setBalance((BigDecimal) request.getSession().getAttribute("balance"));
    try {
        // invoking the iSeries program
        orderBean.invoke();

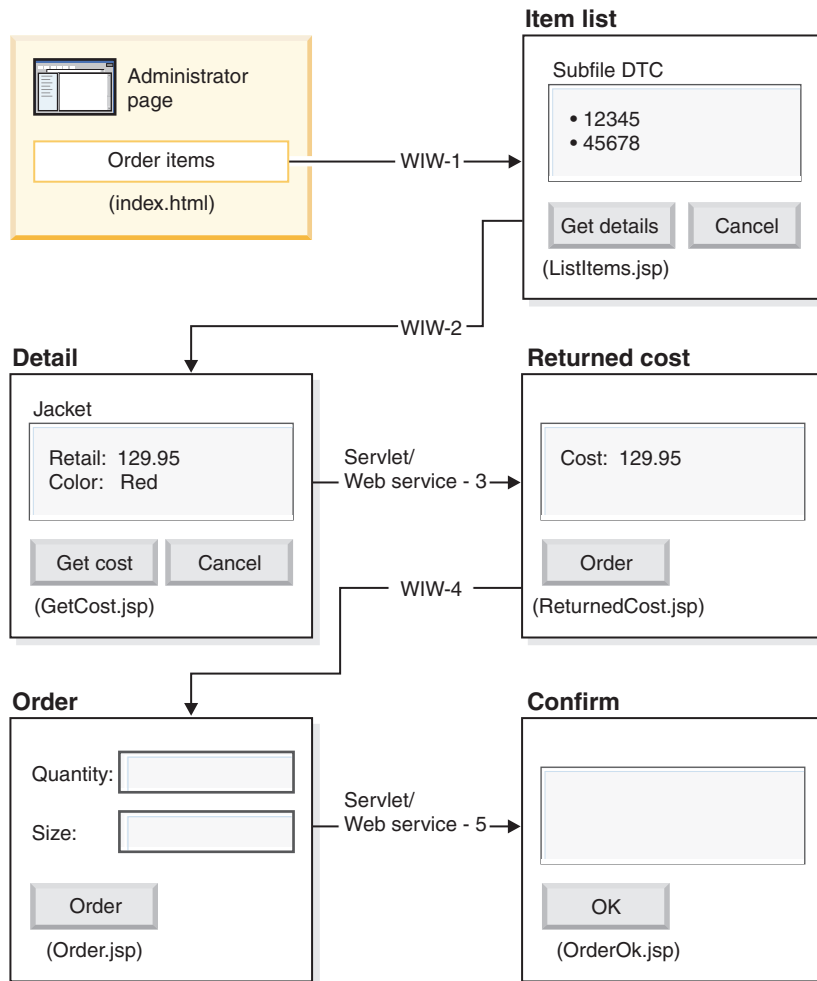
        // retrieving the order number from PLACEORD bean
        orderNumber = (orderBean.getRetCode()).toString();
        request.getSession().setAttribute("orderNumber", orderNumber);
        basket.empty();
        // commit this order now
        recIO.commit();
        response.sendRedirect("orderNumber.jsp");
        return;
    } catch (Exception e) {
        response.sendRedirect("errorPage.html");
        e.printStackTrace();
    }
}
}
}

```

13. Write an OrderNumber servlet that retrieves the number and displays it for the customer along with a confirmation message. If the customer has not added any items to his or her basket, ensure that an error page is returned instead.

Advanced module 2: SV000618 – Create the Web page that uses the SV000514 and SV001586 Web services

This project demonstrates how you can use RPG programming knowledge to create Web clients for iSeries Web Services and RPG programs. In this project, you play the administrator's role, stepping through a series of Web pages to determine inventory quantity, and to order additional inventory for your retail store from the wholesale supplier. You enter the item number, view the details of the item, order the quantities and size, and accept the confirmation.



(WIW = Web Interaction Wizard)

Because this is an advanced module, the instructions do not take you through each step of creating the project, but outline the development steps taken to create such a project. This project uses the following components of Development Studio Client:

- iSeries Web development tools to create the Web pages with Page Designer, incorporating the Web Interaction wizard and various design-time controls (DTC)
- The Remote Systems Explorer to create a TNLSTITM RPG service program that returns item information
- The Web Services wizard to generate servlet proxy code, which finds item prices and orders the items
- iSeries Java development tools to create the necessary servlets
- The WebSphere Test Environment to verify the application before deployment to the iSeries host through WebSphere Application Server

The next section explains how to develop the project.

Creating the Web pages, servlets, JSPs, and RPG code

To construct the components of SV000618:

1. Create a Web project to hold all of the files you will create.
2. Write a ListItems JSP file that lists items in the iSeries inventory database. You can use Page Designer in iSeries Web development tools to write the servlet. More specifically, you can lay out the page in the Design view, and add appropriate code in the Source view. You also need to insert a subfile Design Time Control (DTC) to interact with the TNLSTITM RPG service program, filling the subfile with database records. You can specify the service program in the DTC control settings.

Next, you need to use the Web Interaction wizard to create the input page:

- Specify ListItems.jsp as the output page to list inventory items, which ensures that the Web Interaction wizard creates a ListItems.wit file.
- Make sure not to specify any program calls in the Web Interaction wizard since the subfile DTC automatically invokes the TNLSTITM RPG service program. The wizard also generates a ListItemsWitServlet that serves as a link to invoke the ListItems.jsp page.
- To review the ListItemsWit.wit file generated by the Web Interaction wizard:
 1. Expand **SV001618** and double-click **ListItems.wit** to display the interaction wizard for the file.
 2. Click **Next** through the wizard to review the values specified for the interaction.

Next, you need to write a GetCost JSP output page with Page Designer, which takes input from the ListItems.jsp input page. When a user clicks an item on the ListItems.jsp page, a GetCost.jsp page displays details for the item.

After creating the GetCost.jsp page, use the Web Interaction wizard to create a WitOrder interaction between ListItems.jsp (that you select as the input page) and GetCost.jsp (that you select as the output page):

- On the Program Call page of the wizard, specify an invocation of the GetDetail procedure and parameter from the TNLSTITM RPG service program.
- In the procedure, subfile DTC APIs are incorporated to determine which subfile record has been selected. The procedure uses this information to retrieve the selected record from the INVENTORY database and displays details, including the image, of the selected item on GetCost.jsp.
- To review the WitOrderWit.wit file generated by the Web Interaction wizard:
 1. Expand **SV001618** and double-click **WitOrder.wit** to display the interaction wizard for the file.
 2. Click **Next** through the wizard to review the values specified for the interaction.

Notice that the *flow* parameter is specified as a flow controller on the output page. This makes the parameter's value ensure that the appropriate JSP files display.

Because the GetDetail parameter was specified in the Web Interaction wizard, you need to add Java code to GetCost.jsp to display images of the selected item. You can use the following lines of code to access and display the image in GetCost.jsp:

```
<TD><%  
String[] strimageValueArray = null;  
String strimageValue = "";
```

```

if( session.getAttribute("Affinity_0_info.image") != null )
{
    strimageValueArray = (String[])session.getAttribute("Affinity_0_info.image");
    strimageValue = strimageValueArray[0];
}
%>
</TD>

```

Next, you need to import Web Services Definition Language (WSDL) files from project SV000514 so that the administrator can retrieve the current cost of an item from the wholesale supplier by pressing the **Get cost** button.

- The **Get cost** button invokes `QryProdCostServlet.jsp` and corresponding Web service from project SV000514.
- Use the Web Services wizard and imported WSDL files to generate the Java proxy code required to invoke the Web service.
- The `QryProdCostServlet.jsp` takes input from the `GetCost.jsp` page, uses Java proxy code to invoke the SV000514 Web service to find the cost of the selected item, and displays the cost in a page called `ReturnedCost.jsp`.
- To view the `QryProdCostServicesProxy.java` code and `QryProdCostServlet.java`:
 1. Expand **SV001618 > source > proxy > soap**.
 2. Double-click **QryProdCostServicesProxy.java**.
 3. For `QryProdCostServlet.java`, double-click **QryProdCostServlet.java**, under **SV001618 > source**, and note how it instantiates the Java proxy code.

Next, you need to use the Web Interaction wizard to link `ReturnCost.jsp` as input and `Order.jsp` as output so that the administrator can click an **Order** button to order the selected item from the wholesale supplier.

- With this interaction, you do not need to use a program call, as the linking of the two pages is enough to display the correct information.
- To view `WitPlaceOrder.wit`:
 1. Expand **SV001618**.
 2. Double-click **WitPlaceOrder.wit** to open the interaction.
 3. Click **Next** through the wizard to review the specified values.

Next, use the SV001586 Web service so that the administrator can specify the size and quantity of ordered items.

- Import the SV001586 WSDL files into this project, generate Java proxy code to invoke the Web service, and write an `OrderSupplyServlet` invoked when the user presses the **Order** button from `Order.jsp`.
- The servlet gathers information from `Order.jsp`, invokes the Web service Java proxy code, which invokes the SV001586 Web service and orders the item.
- The servlet displays `OrderOK.jsp` if the order is successful, and an error page if unsuccessful.
- To see how the generated Web service proxy is instantiated and invoked to use the SV001586 Web service:
 1. Expand **SV000618 > source**.
 2. Double-click **OrderSupplyServlet.java** and examine the contents.

Before you deploy to WebSphere Application Server

Before you deploy your application to WebSphere Application Server, you need to change a specific URL within the SV000514 and SV001586 Java class files for the application to function properly.

For SV000514:

- In the Web services proxy class, `QryProdCostServicesProxy`, a variable is defined that contains the URL of the Web service to invoke.
- When the proxy is first created, this URL is set to `http://localhost:8080/SV000514/servlet/rpcrouter`.
- With the variable set to this value, the Web service in project SV000514 in the IDE is invoked, as was demonstrated in “Running the application in the workbench” on page 13.
- Before you deploy this application, you need to change this URL value to point to where you have deployed the EAR file, `SVWholeSale.ear`, on the iSeries IFS directory.

For project SV001586:

- In the Web services proxy class, `OrderSupplyServicesProxy`, a variable is defined that contains the URL of the Web service to invoke.
- When the proxy is first created, this URL is set to `http://localhost:8080/SV001586/servlet/rpcrouter`.
- With the variable set to this value, the Web service in project SV001586 in the IDE is invoked, as was demonstrated in “Running the application in the workbench” on page 13.
- Before you deploy this application, you need to change this URL value to point to where you have deployed the `SVWholeSaleEAR.ear` file on the iSeries IFS directory.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

Director of Licensing
Intellectual Property & Licensing
International Business Machines Corporation
North Castle Drive, MD - NC119
Armonk, New York 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Canada Ltd. Laboratory
Information Development
B3/KB7/8200/MKM
8200 Warden Avenue
Markham, Ontario
Canada L6G 1C7

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Programming Interface Information

This publication is intended to help you to create and manage applications and user interfaces on the workstation, in a client/server environment. It contains examples of data and reports that are used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses that are used by an actual business enterprise is entirely coincidental. This publication documents General-Use Programming Interface and Associated Guidance Information provided by IBM WebSphere Development Studio Client for iSeries.

Trademarks and Service Marks

The following terms are trademarks or registered trademarks of the International Business Machines Corporation in the United States or other countries or both:

- Application System/400
- AS/400
- AS/400e
- DB2/400
- IBM
- iSeries
- Integrated Language Environment
- OS/400
- VisualAge
- WebSphere

Java and all Java-based trademarks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, and Windows NT are trademarks or registered trademarks of Microsoft Corporation in the United States, or other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.



Program Number: 5724-A81

Printed in U.S.A.