



IBM WebSphere Development Studio Client for iSeries: iSeries Retail Store Scenario Application

Version 5.1 for Windows



IBM WebSphere Development Studio Client for iSeries: iSeries Retail Store Scenario Application

Version 5.1 for Windows

Contents

Chapter 1. Introduction to the iSeries

retail store 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 .savf files 8

Modifying your library list 9

Importing and running the application in

Development Studio Client 10

Creating the Web projects. 10

Starting the WebSphere Test Environment 11

Importing the Web project files 11

Extracting the SV000501 .zip files on to your local system 12

Importing the WebFacing project files and creating the WebFacing project 12

Configuring the WebFacing project 13

Defining server information 14

Server configuration 14

Running the application in the workbench 15

Note: Running your application behind a firewall 15

Running the application as a customer in the workbench 16

Running the application as an administrator in the workbench 17

Deploying the scenario application to the

WebSphere Application Server 17

Configuring WebSphere Application Server. 17

Securing the administrator's page 18

Creating EAR files for iSeries WebSphere Application Server deployment. 19

Deploying the EAR files to iSeries WebSphere Application Server 20

Running the application in WebSphere Application Server 20

Chapter 4. Step-by-step module 1:

Create a Web service to return product prices (SV000514) 21

Introduction 21

Before you begin 21

Creating a new Web project 22

Defining the iSeries server information 22

Creating the RPG service program. 22

Creating the parameters and generating the Java bean. 23

Making a Web service from the Java bean 24

Testing the sample 25

Chapter 5. Step-by-step module 2:

Create an interface to view inventory and order items (SV000501). 27

Introduction 27

Before you begin 27

Testing the project on your iSeries server (optional) 28

Create a connection to your iSeries server 28

Creating a WebFacing project 28

Converting the DDS source 30

Configuring UTF-8 support in the workbench — For WAS Version 4.0 users only. 30

Configuring UTF-8 support for WebSphere Application Server 31

Creating a style sheet 32

(Optional) Manually customizing the cascading style sheet. 32

Extending and enhancing your WebFacing project with a Web interaction. 32

Link the project to a Web interaction 35

Publishing your files and restarting the server. 36

Testing the interface 37

Chapter 6. Advanced module 1: Create

HTML, servlets, and JSP files that place customer orders on an iSeries server (SV001585) 39

Introduction 39

Summary of high level steps 39

Before you begin 40

Creating the Web page, servlets, and JSP files 40

Chapter 7. Advanced module 2: Create

the Web project that uses the SV000514 and SV001586 Web services (SV000618) 49

Introduction 49

Before you begin 50

Creating the Web pages, servlets, JSPs, and RPG code. 50

Before you deploy to WebSphere Application Server 52

Notices 53

COPYRIGHT LICENSE: 54

Programming interface information 54

Trademarks and service marks 55

Chapter 1. Introduction to the iSeries retail store scenario application

The scenario package is a sample application developed using IBM WebSphere Development Studio Client for iSeries, the Eclipse-based technology for the iSeries server. The scenario is designed for developers wanting to use an iSeries server for:

- Java development
- Web development
- The management and transformation of RPG code into Web applications

The application consists of two .savf files that you restore onto your iSeries server and five .zip files that you import into the client product. They form a series of Web pages with URLs for various points in the process.

Important note: Check our support page for all service packs, PTFs, and other information: ibm.com/software/awdtools/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
- 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 server 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 fulfillment. They now hope to use the Web to carry out regular business transactions.

The retail store wants to have a Web site that:

- Customers can use to purchase products
- Employees can use to order inventory from the wholesale supplier

The wholesale supplier wants to:

- Receive retailer orders online for tracking
- 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 a link 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	Scenario number
Display product prices		Use an iSeries RPG program to create a Web service, and use Web development tools to view and display the prices.	SV000514
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 server as well as place orders and show a purchase summary.	SV001585
	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.	SV000501
	Order from the wholesale supplier with the merchandise ID and quantity.	Create a Web service that is invoked when you click the Purchase button.	SV001586
View initial web page for the store.	View initial web page for ordering inventory.	Use Web development tools for iSeries to create both home pages.	SV000618

Project-based scenarios

The iSeries scenario application is composed of five projects, named SV000501, SV000514, SV000618, SV001585, and SV001586. This guide contains instructions on how to run the pre-packaged application. The guide also contains step-by-step modules and advanced modules, showing you how to build the various projects by yourself. The step-by-step modules are intended for developers who are relatively new to application development and Development Studio Client. The advanced modules are intended for developers who are experienced in application development and more familiar with Development Studio Client.

Note: Although there are five projects, there are only four modules because project SV001586 and SV000514 are part of the same application.

This scenario comes in the form of five projects, belonging to the two applications:

- **RetailStore Application:**

- **SV000501 project: Create a Web project 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.
- **SV000618 project: Create a Web project 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.
- **SV001585 project: Create HTML code, servlets, and JSP files that place client orders on an iSeries server** – 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 server. 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 server management and RPG programming.

- **Wholesale Application:**

- **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 server, 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 server** – 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.

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.

Enterprise Archive file (EAR)

An EAR file is a standard Java Archive (JAR) file with an .ear extension. They can contain multiple Web project, and you use them to package and deploy your Web applications to the WebSphere Application Server (WAS). **Note:** 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)

JSPs provide the ability to display dynamic content in static HTML pages. Written in Java, JSPs are server and platform-independent. By effectively separating the Web presentation from the Web content, JSPs 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 Components

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

Web Interaction wizard

This wizard is part of iSeries Web development tools. It creates and manages the interactions between iSeries programs and Web pages. The wizard controls where input, output, and error messages display, and directs the data from the input and output fields to the ILE programs. You can also use the Web Interaction wizard 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

Web services are self-contained applications designed and implemented for use over the Internet. They are created with 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, or if you want 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 server 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 *Retail store* 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 retail store scenario application,” on page 1 for an overview of the applications.

By the end of this chapter, you should be able to:

- Restore the sample files and Development Studio Client objects to your iSeries server
- Create Web projects and a WebFacing project to hold the files
- Import the restored files from your iSeries server into Development Studio Client
- Configure the WebSphere Application Server (WAS)
- Configure the WebFacing project
- Run the application as a customer in the WebSphere Test Environment
- Run the application as an administrator in the WebSphere Test Environment
- Configure your WebSphere Application Server
- Secure the administrator’s page
- Create EAR files for external WAS deployment
- Deploy the application to WAS
- Run the application in WAS

Before you begin

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

- You have a V5R1 or later iSeries server
- You have the latest WebFacing PTFs applied. See our support page for PTFs: <http://www.ibm.com/software/awdtools/wdt400/support/>
- You have NET USE access to the iSeries server
- You have the latest fixes applied to Development Studio Client. To apply fixes, open the product and select **Help > Software Updates > New Updates**. If there are fixes available, a window appears to guide you through the updates. If this is the first time you have applied updates, repeat the procedure. There may be additional fixes to apply.

Installing the sample files

To use the iSeries scenario application you need to work with the following files:

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

The .savf 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 .savf files to your iSeries server, and then you can import the .zip files into Development Studio Client and run the application in the workbench.

Restoring the .savf files

To work with the samples in this guide, you need to restore the WHOLESALE and RETAILSTOR libraries to your iSeries server. You should do this even if you have already restored the libraries for a previous release of the product, because their contents are different.

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

To restore the Wholesale.savf file:

1. Log on to your iSeries server.
 - a. Create a library to contain the save files. To create a new library, enter CRTLIB.
 - b. Name your library SCENARIO.
 - c. Tab to the next line; specify *TEST as the library type and press Enter to save your changes.
 - d. Create two save files using the CRTSAVF command, pressing enter between the two lines:

```
CRTSAVF FILE(SCENARIO/WHOLESALE)
CRTSAVF FILE(SCENARIO/RETAILSTOR)
```

These lines specify that you want to create save files in your Scenario library.

2. On your workstation, open a Command Prompt window.
 - a. You need to change to the directory where the .savf files exist. By default, they are created in c:\wdsc\wdscsampl. If you installed the product to a different drive, or you chose not to use "wdsc" as the home directory for Development Studio Client, then change to the wdscsampl directory in the correct place where you installed the product.
 - b. On the command line, enter: ftp *hostname*, where *hostname* is the name of your iSeries server, for example, PROD400.
 - c. Enter your user ID and password for the iSeries server.
 - d. On the command line, enter cd /qsys.lib/scenario.lib to switch to your Scenario library.
 - e. Enter the following lines:

```
bin
put WHOLESALE.savf WHOLESALE.savf
put RETAILSTOR.savf RETAILSTOR.savf
quit
```

These lines (before you quit) specify that you want to take the save files file from your local system and put them onto the iSeries server.

3. Back in the iSeries console, restore your Wholesale library:

- a. Enter RSTLIB and press F4 to define how you want to restore the library.
 - 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 in the **Save file** field and press the Tab key.
 - f. In the **Library** field, delete the existing value and enter scenario.
 - g. Press Enter to restore the WHOLESale library to your iSeries server.
4. Repeat this procedure for the Retailstor.savf file:
- a. Enter RSTLIB and press F4 to define how you want to restore the library.
 - b. In the **Saved Library** field, enter RETAILSTOR 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 RETAILSTOR in the **Save file** field and press the Tab key.
 - f. In the **Library** field, delete the existing value and enter scenario.
 - g. Press Enter to save your action and restore the RETAILSTOR library to your iSeries server.

Modifying your library list

Although you have restored the RETAILSTOR and WHOLESale libraries to your iSeries server, you might also want to ensure that the RETAILSTOR library is in your library list that you use when you sign on. This is so that you can run this application in the green-screen environment, which is good for testing purposes, in Chapter 4, "Step-by-step module 1: Create a Web service to return product prices (SV000514)," on page 21. (The WHOLESale library is added at run time.) To do this, you need to update your job description and user profile so that the changes to your library list are permanent.

Different users have different ways of modifying their library list, but if you do not know how, here is an example:

1. On your iSeries server, type dsplibl and press Enter to display your library list.
2. Check to see if the RETAILSTOR library is in your library list. If it is there, you can exit the green-screen and continue on with the next section, "Importing and running the application in Development Studio Client" on page 10. If the RETAILSTOR library is not there, then:
 - a. Press F12 to exit your current screen.
 - b. Locate the name of your job description. To do so, type dspusrprf *username* and press Enter to display your user profile, where *username* is the ID you use to sign on to the iSeries server. Your job description is on one of the pages of your user profile (use Page Down). When you find your job description, make a note of it and press F12 to exit from the screen.
 - c. At the command line, type chgjobd and press F4.
 - d. In the **Job Description** field, type in your job description name and press F10 to see additional parameters.
 - e. Page down to **Initial library list**.
 - f. In the field below **Initial library list**, you will see "+ for more values". Type a plus sign and press Enter.

- g. On the **Specify more values** page, delete any entry that exists, and type RETAILSTOR. Press Enter.
- h. Press Enter again to save your changes.
- i. At the command line, type: chgusrprf and press Enter.
- j. Type your username and press Enter.
- k. Press F10 to see additional parameters.
- l. Page down to **Job description**.
- m. If the name of your job description does not match the job description you found with the dspusrprf command, type the name of the job description you found with that command and press Enter to save your changes.

Importing and running the application in Development Studio Client

Five zip files exist in your installed program files for Development Studio client, by default in the c:\wdsc\wdscsampl directory. These files are:

- Retail store application .zip files:
 - SV000501.zip: This project was created using the Webfacing Tool. It displays the current inventory and outstanding orders for the retail store.
 - SV000618.zip: This project uses the Web Interaction Wizard to call an RPG program, and display a list of available items on a Web page.
 - SV001585.zip: This project uses Java to allow users to shop on the retail store's Web site and add items to their shopping baskets.
- Wholesale application .zip files:
 - SV000514.zip: This project uses an RPG program on the iSeries server as a Web service to return item price information.
 - SV0001586.zip: This project uses an RPG program as a Web service to check the availability of items from the wholesaler.

You will import each one separately into the Development Studio Client workbench and specify a corresponding Enterprise Archive (EAR) file for each. An EAR file is a standard Java Archive (JAR) file with an .ear extension. They can contain multiple Web projects, and you use them to package and deploy your Web applications to the WebSphere Administrative Server (WAS).

The first four files correspond to the Web projects, and the fifth one corresponds to a WebFacing project. First, you create the Web projects, and then you create the WebFacing project. Afterwards, you import the contents of the zip files into the five projects.

Creating the Web projects

In this section, you will create Web projects for SV000514, SV000618, SV001585, and SV001586. The steps to create each project are outlined separately. Although the steps might seem repetitious, keep in mind that there are a few minor but important differences you need to take note of when creating the four projects, (concerning which EAR files are associated with which projects).

To create the Web projects:

1. Open IBM WebSphere Development Studio Client for iSeries from the Start menu.
2. In Development Studio Client, open the Web perspective with the menu bar by clicking **Window > Open Perspective > Other > Web** and click **OK**.
3. Click **File > New > Dynamic Web Project**.

- a. In the **Project name** field, enter SV000514.
 - b. Check the **Configure advanced options** check box and click **Next**.
 - c. Click the **New** button beside the **EAR project** field.
 - d. In the **Project name** field, enter SVWholeSaleEAR and click **Finish** on both dialog boxes.
4. Click **File > New > Dynamic Web Project**.
 - a. In the **Project name** field, enter SV001586 and click **Next** (The **Configure advanced options** check box should be checked automatically).
 - b. Verify that SVWholeSaleEAR is selected in the **EAR project** drop-down field.
 - c. Click **Finish**.
 5. Click **File > New > Dynamic Web Project**.
 - a. In the **Project name** field, enter SV000618 and click **Next**.
 - b. Click the **New** button beside the **EAR project** field.
 - c. In the **New project name** field, enter SVStoreEAR and click **Finish** on both dialog boxes.
 6. Click **File > New > Dynamic Web Project**.
 - a. In the **Project name** field, enter SV001585 and click **Next**.
 - b. Verify that SVStoreEAR is selected in the **EAR project** drop-down field.
 - c. Click **Finish**.

Starting the WebSphere Test Environment

Before you import the content for all of your Web projects, you need to start the WebSphere Test Environment.

Importing the Web project files

To import the file systems into your workspace, the following procedure outlines how to import the files in SV000514. The procedure is the same for the other three Web project file systems: SV001586, SV001585, and SV000618. The section after this one explains how to create and import the WebFacing project.

To import the files:

1. If you are not in the Navigator view, click the **Navigator** tab.
2. Right-click the **SV000514** folder and select **Import**.
3. From the **Import** wizard, click **Zip file** and then click **Next**.
4. Browse to the directory that contains the .zip files; by default this should be c:\wdsc\wdscsampl.
5. Click **SV000514.zip** and click **Open**.
6. Delete "SV000514" from the **Into folder** field, and enter "/" (a single forward slash) in its place.
7. Select the **Overwrite existing resources without warning** check box.
8. Click **Finish** and click **Yes** if you receive any dialog boxes during the importation.
9. Repeat steps 2 to 8 for SV001586, SV001585, and SV000618.

Extracting the SV000501 .zip files on to your local system

Before you can import the contents of the SV000501 WebFacing project into the Development Studio Client workbench, you need to extract the .zip file into a separate directory on your local system, so that you can import the project as a file system instead of a .zip file.

Note: If you are using a program other than WinZip, complete the tasks as you would with that program.

1. On your local system, navigate to c:\wdsc\wdscsampl. The c:\wdsc directory could be different if you chose to install to a different drive or a different base folder. If so, navigate to that directory, and then find \wdscsampl.
2. Double-click on the following .zip file: SV000501.zip.
3. Click the Extract button and navigate to the same folder where the zip files are, by default, c:\wdsc\wdscsampl.
4. Select the **All Files** radio button, and then click **Extract**.
5. If you look in your wdscsampl directory, you will see that a SV000501 folder is created, with all the necessary files inside.
6. Now that all of the files are extracted into c:\scenario\SV000501, delete the SV000501.zip file so that you do not accidentally select it when you import your file system.

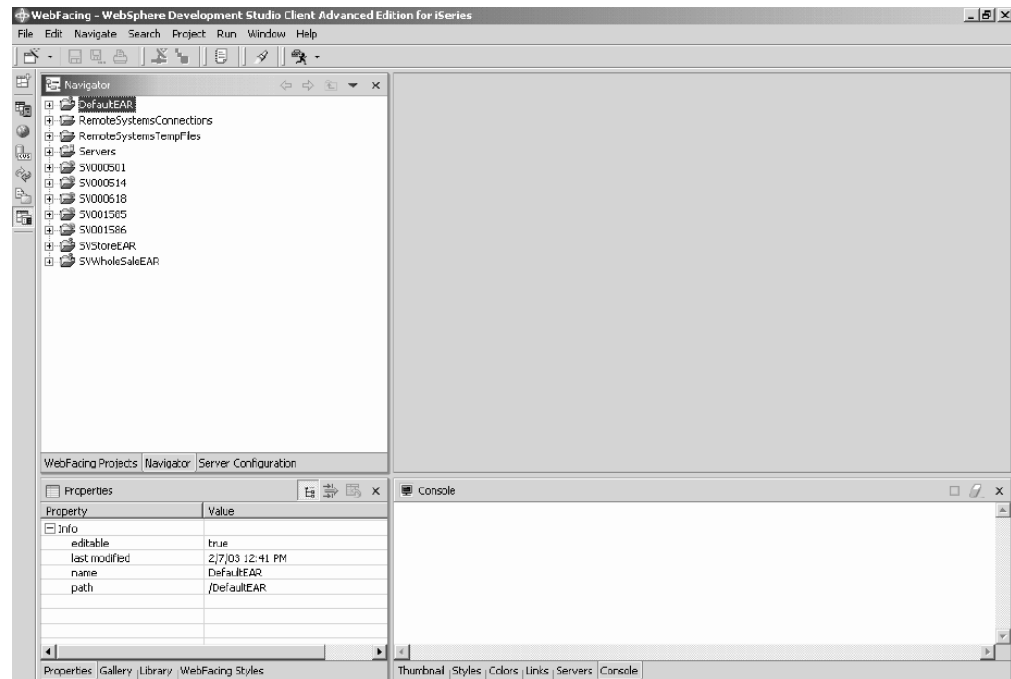
Importing the WebFacing project files and creating the WebFacing project

Project SV000501 is different from the other four Web projects you just created, because SV000501 is a WebFacing project. WebFacing projects are different because they include different files, and have a unique structure specific for handling DDS files from your iSeries server.

To import the files and create a WebFacing project:

1. If you are not in the Project Navigator view, click the **Project Navigator** tab.
2. Right-click inside the view and select **Import**.
3. In the Import window, select **WebFacing Projects** and click **Next**.
4. Browse to the directory where you extracted the scenario files; by default this should be c:\wdsc\wdscsampl.
5. Select **SV000501** and click **OK**. Click **Next**.
6. In the **WebFacing projects found** area of the window, select the **SV000501** checkbox.
7. In the **Enterprise Application Project (EAR)** field, type **SVStoreEAR**.
8. Click **Finish**.

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



If you see errors beside your imported projects, (indicated by red x marks), right-click each project and select **Rebuild Project**. Depending on what is in your workspace, this should clear up most of the errors so that the application runs properly. Any errors you still see could exist because some of the code has not fully migrated to the 5.1 release of Development Studio Client. You can continue with the exercises in this chapter to help introduce yourself to the product, or you can proceed to the next chapter to try building the applications from scratch.

Configuring the WebFacing project

Before you run the application or work on the SV000501 project, you need to start the WebFacing server for the application to run, and you need to configure the project so that it uses the correct iSeries server.

To start the WebFacing server:

1. Sign-on to your iSeries server.
2. At the command line, enter `strtcpsvr *webfacing`.
3. At the bottom of the screen you should see the message, "WEBFACING server starting".

You now need to change WebFacing properties, back inside the Development Studio Client workbench:

1. Switch to the Web perspective (you can switch between perspectives by clicking the icons clustered on the left side of the screen).
2. Click the **Project Navigator** tab so that you can see your project structures.
3. Expand **SV000501** and double-click **Web Deployment Descriptor**.
4. Click the **Parameters** tab.
5. Click **WFDefaultHost**.

6. In the **Value** field, erase the existing value and enter the name of your iSeries server.
7. From the workbench menu bar, click the Save icon or click **File > Save Web Deployment Descriptor**. Click **OK** if you receive any messages. Close the file.

Defining server information

Now you need to define the iSeries server information for all five projects, ensuring that they are configured to run with your iSeries server under your user ID and password. To define server information:

1. Make sure you are in the Web perspective.
2. In the Project Navigator view, right-click **SV000501** and select **Specify iSeries Web Tools Run-time Configuration**.
3. In the **iSeries server name** field, delete the existing value and enter the name of the iSeries server.
4. Delete the other existing values and enter your user ID and password.
5. Click **Finish** (and click **Finish** again if necessary).
6. Repeat steps 2 to 5 for the other four projects: SV000514, SV000618, SV001585, and SV001586.

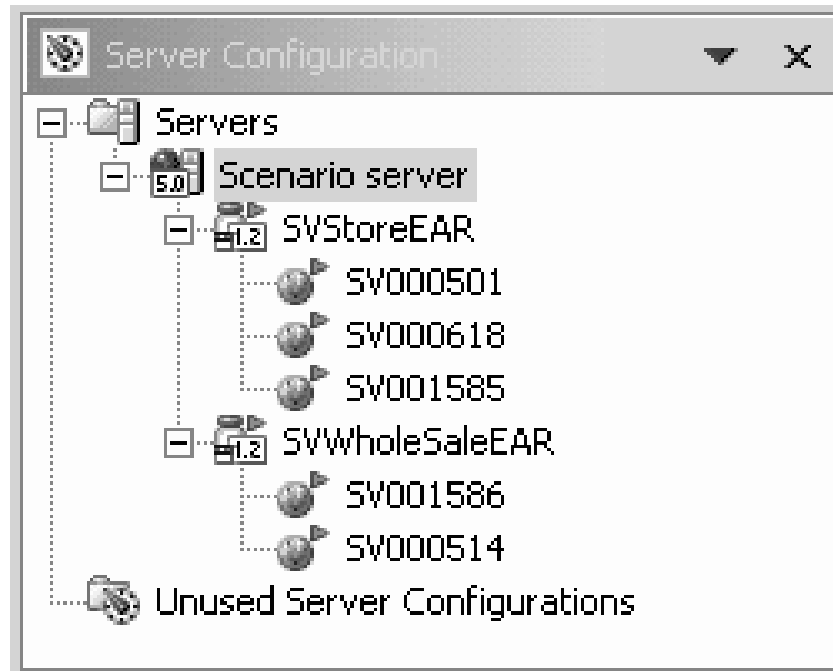
Server configuration

Now that you have imported all of the source files, you need to configure the WebSphere Test Environment server so that it recognizes the SVWholesaleEAR and SVStoreEAR applications.

To configure the WebSphere Test Environment server:

1. Switch to the Server perspective. From the workbench menu bar, click **Window > Open Perspective > Other > Server** and click **OK**.
2. In the Server Configuration view, right-click **Servers** and select **New > Server and Server Configuration**.
3. In the **Server name** field, enter Scenario server.
4. In the **Folder** field, enter Scenario folder.
5. In the **Server type** box, expand **WebSphere version 5.0** and click **Test Environment**, if it is not already selected by default.
6. Click **Finish** and click **Yes** if you receive any messages.
7. To add the EAR files to the server configuration:
 - a. Expand **Servers**.
 - b. Right-click **Scenario server** and select **Add > SVWholeSaleEAR**.
 - c. Expand **Servers**.
 - d. Right-click **Scenario server** again and select **Add and Remove Projects**.
 - e. Select click the **Add All** button to move both SVStoreEAR and SVWholeSaleEAR to the right side of the window.
 - f. Click **Finish**.

8. In the server configuration view, expand **Scenario server** and then expand **SVStoreEAR** and **SVWholeSaleEAR** to see all of your applications listed:



To ensure that the server has picked up the projects:

1. In the Servers view, to the right of the Server Configuration view, (click the **Servers** tab if you cannot see the Servers view), right-click **Scenario server** and select **Publish**. A **Publishing** dialog box appears to show you the progress of publishing.
2. Click OK to publish your applications to the WebSphere Application Server test environment.
3. When publishing is finished, click **OK** on the dialog box.
4. In the same view, right-click **Scenario server** again and click **Start**. The Console view opens to show you the server's activities. The server is started when you see the line "Server *servername* open for e-business" message at the bottom of the view. Scroll down to see all of the messages.

Running the application in the workbench

At this point you can run the application in the WebSphere Test Environment in the workbench. When you are satisfied that it is running properly, you can export the EAR file to the iSeries server and deploy it to the iSeries WebSphere Application Server.

Note: Running your application behind a firewall

You might encounter problems if you try 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`. If you are going to be running the application behind a firewall, then 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/eclipse/plugins/com.ibm.etools.j2ee/dtds/web-app_2_2.dtd"
```

where x:/wdsc is the directory where you installed the product. You can usually find your web.xml files under *Project name* > **Web Content** > **WEB-INF**. After you open the file, click the **Source** tab to edit the file directly. If you change any web.xml files, you need to restart your projects:

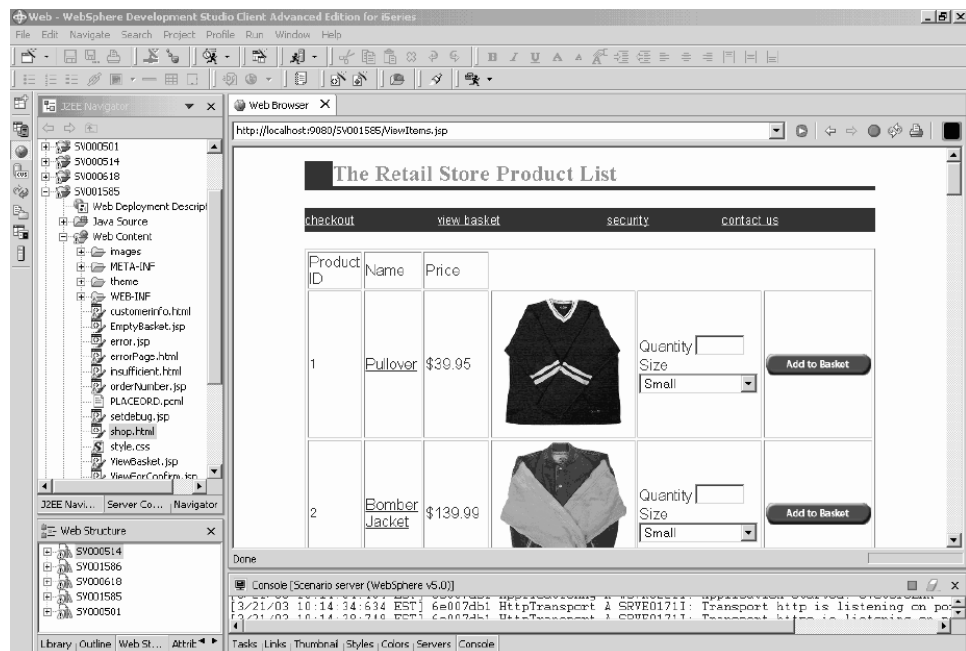
1. Switch to the Servers view.
2. Right-click on your **Scenario** server and select **Restart Project** > **SVWholeSaleEAR**.
3. Wait until you see the "Application started" message in the console. Then restart your SVStoreEAR project as well. Right-click on your **Scenario** server and select **Restart Project** > **SVStoreEAR**.
4. Wait until you see the "Application started" message in the console, and then continue on to the next section.

Running the application as a customer in the workbench

Now you can run the application and view the contents of the Web pages as if you were a customer shopping on the retail store's Web site. You can browse through items and then order a quantity and size.

To run the application as a customer in the workbench:

1. In the Web perspective, expand **SV001585** > **Web Content**.
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 on 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:

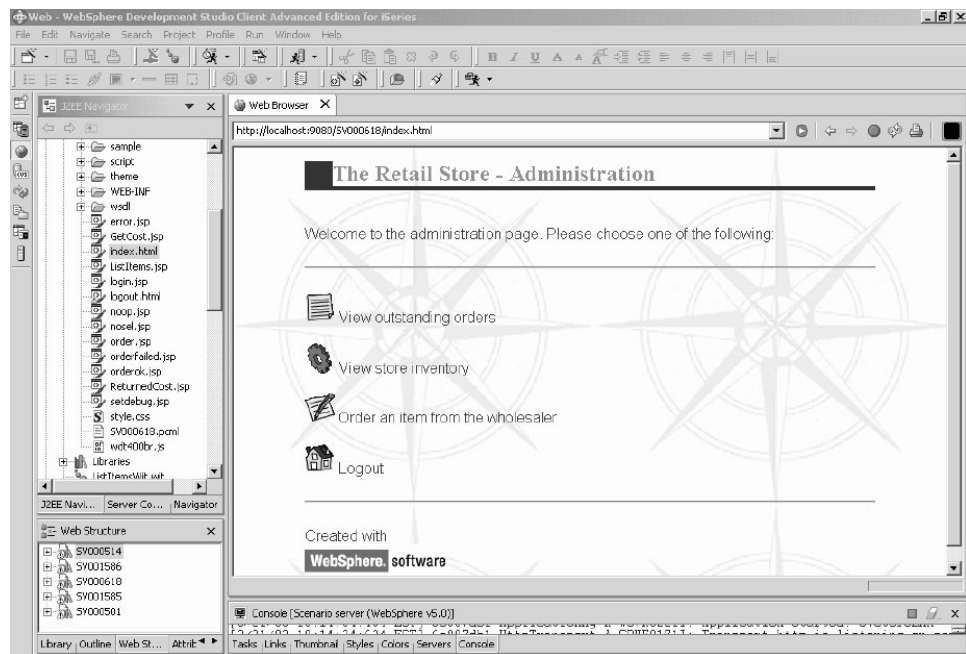


The page that displays the text "your order has been processed" is the last page of the sample. When you are satisfied with the sample, you can close the browser.

Running the application as an administrator in the workbench

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

1. In the Web perspective, expand **SV000618 > Web Content**.
2. Right-click **index.html** and select **Run on Server**. This launches the application in the workbench browser.
3. 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 iSeries scenario application in the the workbench test environment, you can deploy the application to WebSphere Application Server, as if you were running the application 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 15. Now that you have deployed the files to the iSeries server, 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, version 5.0, or WebSphere Application Server Express instance on the iSeries server, 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 server.
- You have the WebSphere Administrative Console version 4.0 or 5.0 installed on your workstation.

Note: For version 5.0 of WAS, the console is browser-based, so you do not need to have a console on your workstation. Consult the on-line documentation to find more information about the various version of WAS.

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/>
- RedBooks page for WebSphere: Search for RedBooks for WebSphere Application Server Version 5.0 and WebSphere Application Server Express Edition Version 5.0 at <http://publib-b.boulder.ibm.com/redbooks.nsf/portals/WebSphere>

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. In Development Studio Client, switch to the Web perspective.
2. In the Navigator view, expand **SV000618 > Web Content > WEB-INF**.
3. Double-click **web.xml** to open the web.xml view.
4. Click the **Security** tab.
5. To secure index.html, the administration page, a Security constraint has also been defined. At the top of the view, click **Security Constraints**.
6. Click the first instance of **SecurityConstraint** in the list.
7. To the right, click **AdminPage**.
8. Click **Edit** to invoke the **Web Resource Collections** dialog box. Note that the GET and POST methods for index.html are pre-selected.
9. 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. They can contain multiple Web project, and you use them to package and deploy your Web applications to the WebSphere Administrative Server (WAS).

To create the EAR files:

1. Switch to the Web perspective.
2. In the Navigator view, right-click **SVStoreEAR** and select **Export**.
3. In the Export window, click **EAR file** and click **Next**.
4. Under **Where do you want to export the resources to?**, click **Browse** and navigate to a directory on your iSeries Integrated File System where you can keep the EAR files. (You need to have mapped 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 2 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 server 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 server 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:

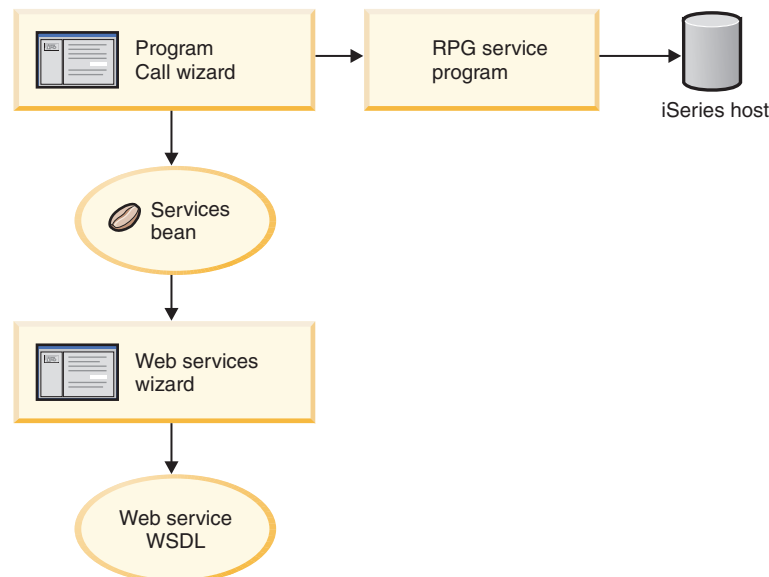
`http://your iSeriesHostName:yourHTTPPortNumber/SV000618/index.html`

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

Chapter 4. Step-by-step module 1: Create a Web service to return product prices (SV000514)

Introduction

In this module, you create a Web service from an RPG program that exists on the iSeries server, to display product prices for your inventory. 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.



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 server.
- You have the latest iSeries server PTFs applied. See our support page for PTF and Service pack information:
<http://www.ibm.com/software/awdtools/wdt400/support/>
- You have started the iSeries servers with the command `STRTCPSVR *ALL`
- You have started the WebFacing server with the command `STRTCPSVR *WEBFACING`
- You have completed all the tasks in Chapter 3, "Running the scenario," on page 7, (except the optional WebSphere Application Server tasks, which are not required to test the application in the workbench).

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 **Window > Open Perspective > Other > Web > OK**.
2. Click **File > New > Dynamic Web project**.
3. In the **Project name** field, enter Project514.
4. Check the **Configure advanced options** check box and click **Next**.
5. Click the **New** button beside the **EAR project** field.
6. In the **Project name** field, enter Project514EAR and click **Finish** on both dialog boxes.

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

Defining the iSeries server information

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

1. Right-click your **Project514** project and select **Specify iSeries Web Tools Run-time configuration**.
2. Enter the name of your iSeries server where the restored WHOLESale library resides, for example, PROD400.
3. Enter your user ID and password for this iSeries server.
4. Type Wholesale in the **Library** field and click **Add**.
5. Click **Finish**.

Creating the RPG service program

You want your application 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 task, 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 first create a Java bean with the Program Call wizard, to invoke the QryProdCost RPG procedure. Then, you use Web services to enable the RPG procedure as a Web service through the Java bean.

To create the Java bean:

1. In the Web perspective, right-click **Project514** and select **New > Other**.
2. Select **iSeries > Java** from the left pane of the window and then select **Program Call Bean** on the right pane.
3. Click **Next** to invoke the Program Call wizard.
4. In the **Java bean name** field, underneath the **Add Program** title, enter Inventory.

5. In the **Program object** field, enter CWSSRV, the name of the RPG service program.
6. In the **Library** field, enter WHOLESAL.
7. From the **Program type** drop-down list, select *SRVPGM.
8. In the **Entry point** field, enter QryProdCost.
9. Click **OK** to add the program definition.

Creating the parameters and generating the Java bean

Now that you have specified the program, you can add parameters. The QryProdCust procedure contains the following two parameters:

- Item number parameter - The program uses this number to find the item in the database.
- Item cost parameter - This parameter is set to the cost of the item as read from the database.

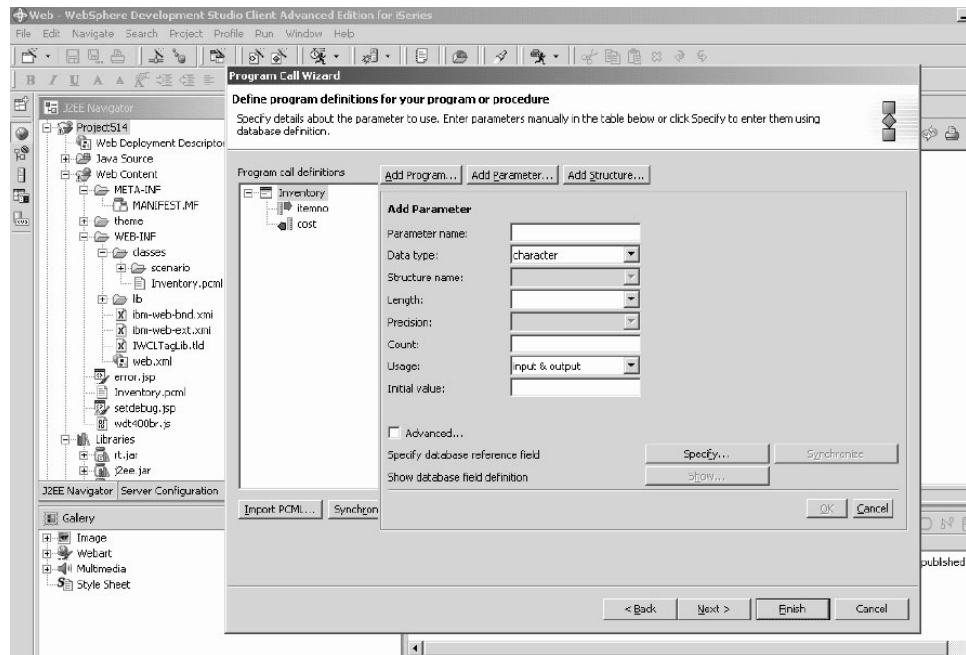
To add the item number parameter:

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.

To enter the item cost parameter:

1. In the left panel of the Program Call wizard, click the **Inventory** program call definition again 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 cost.
4. From the **Data type** drop-down list, select **packed decimal**.
5. In the **Length** field, enter 7.
6. In the **Precision** field, enter 2.
7. From the **Usage** drop-down list, select **output**.
8. 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:



To define the package name for these parameters, in the Program Call Wizard,

1. Click **Next**.
2. In the **Package** field, enter `scenario` as a package name. Leave the defaults in the other fields.
3. 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`.

4. 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.

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

1. In the Navigator view of the Web perspective, expand **Project514 > Java Source > scenario**.
2. Right-click `InventoryServices.java` and select **New > Other**.
3. In the New window, click **Web Services**, click **Web Service** on the right side of the window, and then click **Next**.
4. Select the **Generate a proxy** and the **Test the generated proxy** check boxes.
5. Click **Next**.
6. In the Server-Side Deployment Selection area, click **Edit** and select **IBM SOAP**.

7. In the Client-Side Environment Selection area, click **Edit** and select **IBM SOAP**.
8. Click **Next**.
9. Click **Next** on the Web Service Java Bean Selection page.
10. Click **Next** on the Web Service Java Bean Identity page.
11. On the Web Service Java Bean Methods page, press **Deselect All**.
12. Select the check box for the following method: scenario.inventoryResult inventory(java.math.BigDecimal).
13. Click **Next**.
14. When the Web Services Warning dialog displays, click **Yes to All**.
15. On the Web Service Binding Proxy Generation page, click **Next**.
16. On the Web Service Test page, click **Next**.
17. On the Web Service Publication page, leave all of the default values and click **Finish**.

The Web Services wizard will now generate the files required to run and test the Web Service. This step may take a few minutes. The generation creates a **wSDL > scenario** folder under the **Web Content > WEB-INF** folder of your project. This folder contains four WSDL files and two XSD files that you would distribute to the users of your Web service.

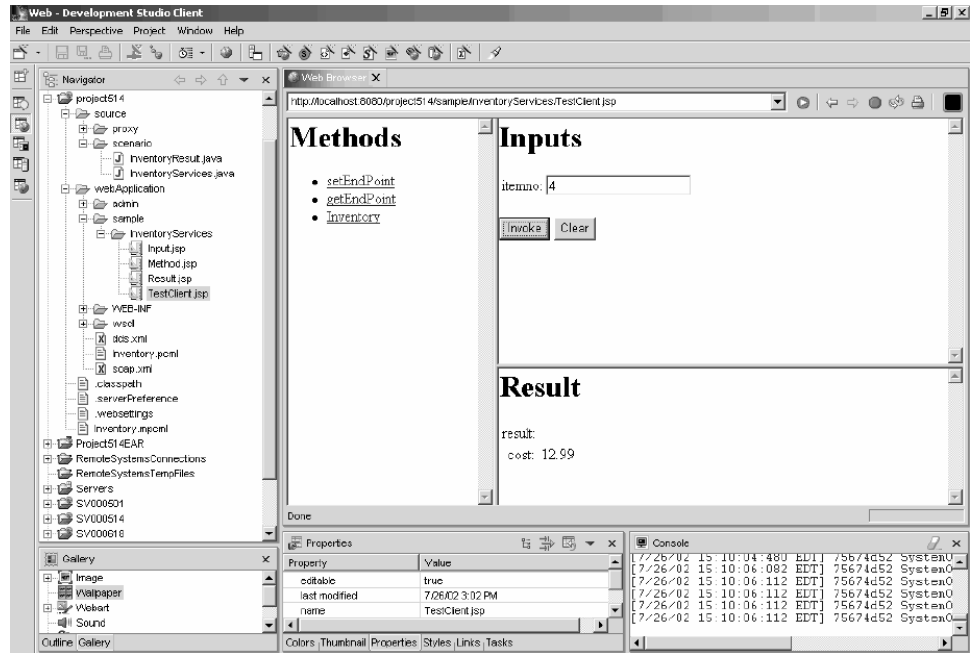
Testing the sample

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 a new project with test pages that you can use to test the Web service. In the **Navigator** view, note that this new project name is **Project514Client**. Also note that the tool automatically opened **TestClient.jsp** on the right side of the workbench. This file is located under **Project514Client > Web Content > sample > InventoryServices**.

To test the sample:

1. In the loaded page, scroll down the methods pane until you are at the very bottom. Click the **Inventory (java.math.BigDecimal)** method.
2. In the **itemno** field, enter 4 and click **Invoke**.
3. Verify that the output returned is:

```
result:  
cost: 12.99
```

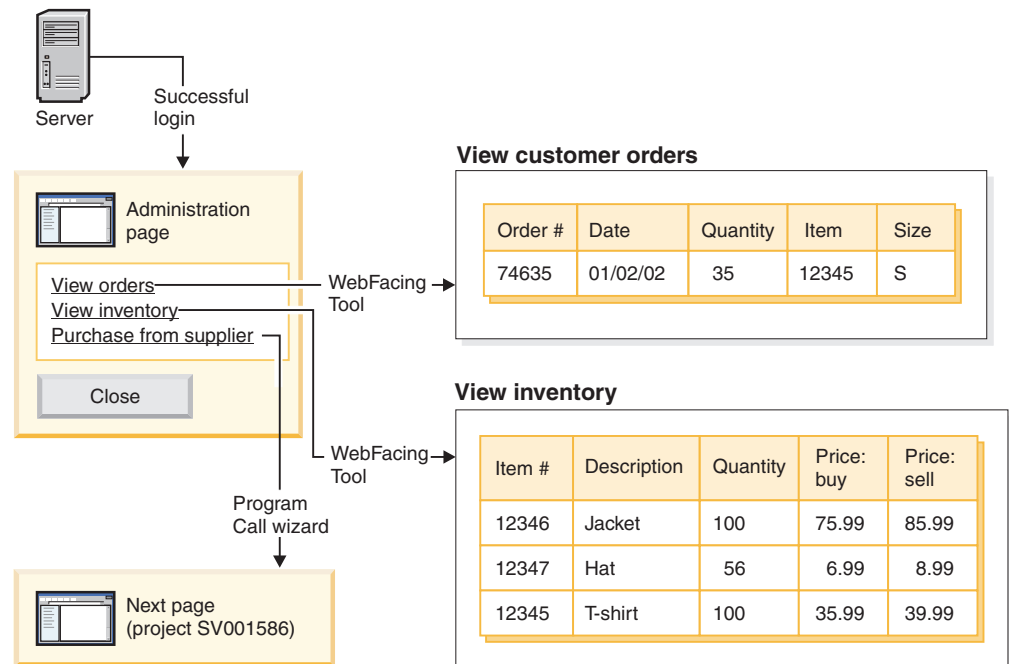


Once you confirm the result, you are finished the module! You have created a Web service that returns product prices.

Chapter 5. Step-by-step module 2: Create an interface to view inventory and order items (SV000501)

Introduction

In this module, you create an interface that displays an administrator's operation options, using the IBM WebFacing Tool. After a successful log in to the iSeries server, 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 server. 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 specify the Web settings.

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 server.
- You have started the iSeries servers with the command `STRTCPSVR *ALL`

- You have started the WebFacing server with the command STRTCPSVR *WEBFACING
- You have completed all the tasks in Chapter 3, “Running the scenario,” on page 7, (except the optional WebSphere Application Server tasks, which are not required to test the application in the workbench).

Testing the project on your iSeries server (optional)

The files for this module are located in the RETAILSTOR library, that you restored to your iSeries server in “Restoring the .savf files” on page 8. Before you create the application in Development Studio Client, you can test the application on your iSeries server, to see how it behaves in a green-screen environment. This process will help you evaluate the look and feel of the application after you create it in Development Studio Client.

To run the application on your iSeries server:

1. Log on to your iSeries server.
2. Ensure that the RETAILSTOR library is on your library list, as described in “Modifying your library list” on page 9.
3. At the command line, enter CALL VIEWINVENT to see the current inventory.
4. Exit from the inventory, and at the command line, enter CALL VIEWORDER to see a list of outstanding orders.

Create a connection to your iSeries server

Before completing the exercises in this chapter, you need to define a profile and connection to your iSeries server.

1. Switch to the Remote System Explorer perspective.
2. In the Remote Systems view, **New Connection** is automatically expanded to show the various remote systems types you can connect to through the Remote System Explorer. Expand **iSeries** to configure a connection to an iSeries server.
3. If you are defining your first connection, you will see the **New - Name personal profile** dialog. Enter a name for your first profile, or accept the default, and click **Next**.
4. In the **Connection name** field, enter MyiSeries.
5. In the **Host name** field, enter the name or TCP/IP address of your iSeries server.
6. Click **Finish**.

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 **Window > Open Perspective > Other > WebFacing** and then clicking **OK**.
2. Create a new WebFacing project by clicking **File > New > WebFacing Project**.
3. Name the project Project501.
4. Click the **Existing** radio button beside **Enterprise Application Project**.
5. Enter SVStoreEAR in the value in the **Existing project name** field; not that the value is case-sensitive. Click **Next**.

6. For the **J2EE level** drop-down list, select **1.3**. Click **Next**.
7. In the **Connection** field, MyiSeries should be filled in automatically, (where the connection name is MyiSeries, as defined in the previous section). Otherwise, click **New** and enter the name of your iSeries server in the dialog, specifying MyiSeries for the connection name, and click **Finish**.
8. Click **Refresh DDS list** and enter your password and user ID 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. Now you need to add a command and command label for the administrator:
12. In the **CL command** field, enter call call viewinvent.
13. In the **Command label** field, delete any existing value and enter View inventory.
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. Now you need to add a second command and command label for the customer:
16. In the **CL command** field, delete the existing value and enter call vieworder.
17. In the **Command label** field, delete the existing value and enter View orders.
18. Select the **Prompt for signon** radio button.
19. Click **Add** and note the addition in the list at the bottom of the window. 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**. Click **Next**.
21. Verify that the **No. I only want to create the project now** radio button is selected and click **Finish**.
22. If you see a Repair Server Configuration message, click **OK**. This message is just to confirm that you want the project added to the SVStoreEAR file.

Because you are constructing Project501 as a mirror of the complete SV000501 project, you now need to copy some of the image files from SV000501 into Project501, so that Project501 displays correctly. (If you want, however, you can add your own images for the interface.) To copy the images:

1. Switch to the Web perspective.
2. In the Navigator view, expand **SV000501 > Web Content > images**.
3. (Optional) Expand **generated** to take note of all the images you are importing into the project.
4. Right-click **generated** and select **Copy**.
5. Scroll up in the Navigator view until you can see **Project501** again.
6. Expand **Project501 > Web Content**.
7. Right-click the **images** folder and select **Paste** from the pop-up menu.
8. Now, the **generated** folder exists for Project501. Expand **generated** to take note of the images you added.

Converting the DDS source

Now that you have created the project, you can convert the DDS display files into JSP files for your Web page. When you convert your DDS display files, the WebFacing Tool generates JSPs and XML files that substitute for the DDS code and make Web access possible. The generated files hold the data for the record format, or control its appearance, display the Web version of the screen, prompt for data, and handle input errors. The wizard also generates an application home page to launch the Web-enabled version of your program.

First, however, you need to alter your source to reference the correct iSeries server. More specifically, you need to perform a modification in one of your DDS display files so that the link for the application will function. You need to change the default name, SV000501, to your project name, Project501:

1. Switch back to the WebFacing perspective and switch to the WebFacing Projects view if it is not already open by default.
2. Expand **Project501 > DDS**.
3. Double-click the second entry,
`<iSeriesserver>RETAILSTOR/QDDSSRC(QUERY)`
4. Scroll down to about line 16, where you see the line:
`A*%WB 12 FLD 1 next ('/SV000501/DetailPage.do?PRODNO={PRODNO}')`
5. Delete SV000501 and enter Project501.
6. Save and close the file.

Note: If at some point you want to run the SV000501 project again, you need to change the Project501 value back to SV000501, since this file exists on the iSeries server and is accessed by both projects.

To convert the DDS source:

1. In the WebFacing perspective, click the **WebFacing Projects** tab to switch to the WebFacing Projects view if it is not already open by default.
2. Expand **Project501**.
3. Right-click the **DDS** folder and select **Convert** to begin conversion. You might need to enter your user ID and password if at some point you were disconnected from your iSeries server.

Configuring UTF-8 support in the workbench — For WAS Version 4.0 users only

IBM WebFacing Tool applications support the display of multiple language on a screen. 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 application's properties file in the workbench.

Note: This section only applies to WAS 4.0 users. WAS version 5.0 performs this task automatically.

To configure UTF-8 support:

1. Switch to the Web perspective (you can switch between perspectives by clicking the icons clustered on the left side of the screen).
2. Click the **Project Navigator** tab so that you can see your project structures.
3. Expand **Project501** and double-click **Web Deployment Descriptor**.

4. Click the **Environment** tab.
5. Click the **Add** button.
- 6.
7. Switch to the Server perspective by clicking its icon on the upper left task bar, or click **Perspective > Open > Other > Server** from the menu bar, and then clicking **OK**.
8. In the Navigator view, expand **Servers** folder.
9. Double-click **defaultInstance.wsi** to open it in the default editor.
10. Click the **Environment** tab and click the **Add** button.
11. An editable value appears, "(New Variable)". Delete this default string and enter: `client.encoding.override`.
12. In the **Value** field, enter UTF-8.
13. Click the save icon or click **File > Save Web Deployment Descriptor**.

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

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 information about store items, such as price and color. For the purposes of the module, you will incorporate the `DetailPageResults.jsp` style sheet from the SV000501 project. For future reference, however, you can manually customize the CSS file (outlined in the next section).

(Optional) Manually customizing the cascading style sheet

As mentioned previously, rather than incorporating the `DetailPageResults.jsp` style sheet from the SV000501 project for Project501, you can customize your own style sheets:

1. Switch to the Project Navigator view and expand **Project501 > Web Content > styles > apparea**.
2. Double-click **apparea.css** to open it in the CSS Designer. This style sheet controls the appearance of your application area.
3. Modify and save the file.
4. Back in the Project Navigator view, navigate to **Project501 > Web Content > styles > chrome**.
5. Double-click **avenue.css** to open it in the CSS Designer. This style sheet controls the appearance of the entire page.
6. Modify and save the file.

Extending and enhancing your WebFacing project with a Web interaction

In this section, you enhance the WebFacing project. When the list of items is displayed, you want to click on an item image to find details for that item. To do this, you use the Web Interaction wizard to invoke an RPG program to retrieve the item's details and display them on another web page. You will perform the following steps:

- define the server 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 server information:

1. Switch to the Web perspective.
2. In the Navigator view, right-click **Project501** and select **Specify iSeries Web Tools run-time configuration**.
3. Enter your iSeries server name, user ID, and password.
4. Type `Retailstor` in the **Library** field and then click **Add**.

5. Click **Finish**. Click **Finish** again if necessary.

Before you create the Web interaction, you need to copy over the correct styles and style sheet so that Project501 displays the proper JSP file format. (You would not need to perform this task if you completed the exercise, “(Optional) Manually customizing the cascading style sheet” on page 32). You first copy over the styles folder, and then you copy over the `DetailPageResults.jsp`:

1. In the Navigator view, expand **SV000501 > Web Content**.
2. Right-click **Styles** and select **Copy**.
3. Scroll up in the Navigator view until you see your **Project501** again.
4. Expand the project, right-click **Web Content**, and select **Paste** from the pop-up menu. Click **Yes** to overwrite the existing styles.
5. Scroll back down to **SV000501 > Web Content**.
6. Right-click **DetailPageResults.jsp** and select **Copy**.
7. Scroll back up until you see your **Project501** again.
8. Expand the project, right-click **Web Content**, and select **Paste** from the pop-up menu.

Note the new elements added underneath Project501’s Web Content folder.

Now you can create the Web interaction:

1. Click **File > New > Other**.
2. In the **New** dialog, click **Web** on the left side, and then click **Web Interaction** on the right side. Click **Next**.
3. In the **Web Interaction name** field, enter `DetailPage` and click **Next**.
4. Select the **Generate Input JSP** radio button.
5. Select the **Use output pages** radio button (if it is not selected already) and click **Add**.
6. From the **Output JSP** dialog, expand **Web Content**, select **DetailPageResults.jsp**, your pre-formatted output page, and click **OK**.
7. Click **Next**.

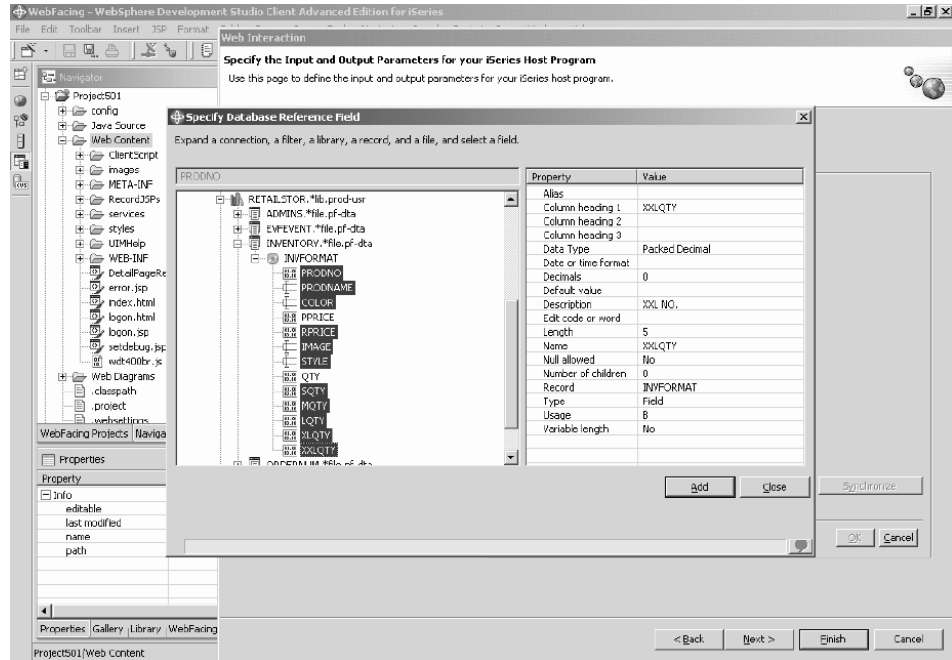
Now, you can add programs and parameters to your interaction. You need to add 11 parameters to the same program. Instead of adding each parameter with its individual values manually, you can add them in a slightly faster way:

1. Select **Use an iSeries ILE program** if it is not selected already.
2. Click **Add Program**.
3. In the **Program alias** field, enter `DetailPage`.
4. For the **Program object** field, click **Browse**.
 - a. Expand `iSeriesserver > *LIBL > RETAILSTOR`.
 - b. Click `DETAILPAGE.*pgm.rpgle` (the first file underneath RETAILSTOR) and click **OK**.
5. Back in the Web Interaction wizard, click **OK** (towards the lower-right).

Note that `DetailPage` is added on the left side of the wizard under Program call definitions.

6. In the Program call definitions section in the left pane of the wizard, click **DetailPage** to select it.
7. Click **Add Parameter**.
8. Beside **Specify database reference field** towards the bottom of the wizard, click **Specify**.

9. Expand *iSeries*server > *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**, or holding down the CTRL key and clicking each one, and then clicking **Add**: PRODNO, PRODNAME, COLOR, RPRICE, IMAGE, STYLE, SQTQ, MQTY, LQTY, XLQTY, XXLQTY. Basically, you are selecting all of the fields except PPRICE and QTY.

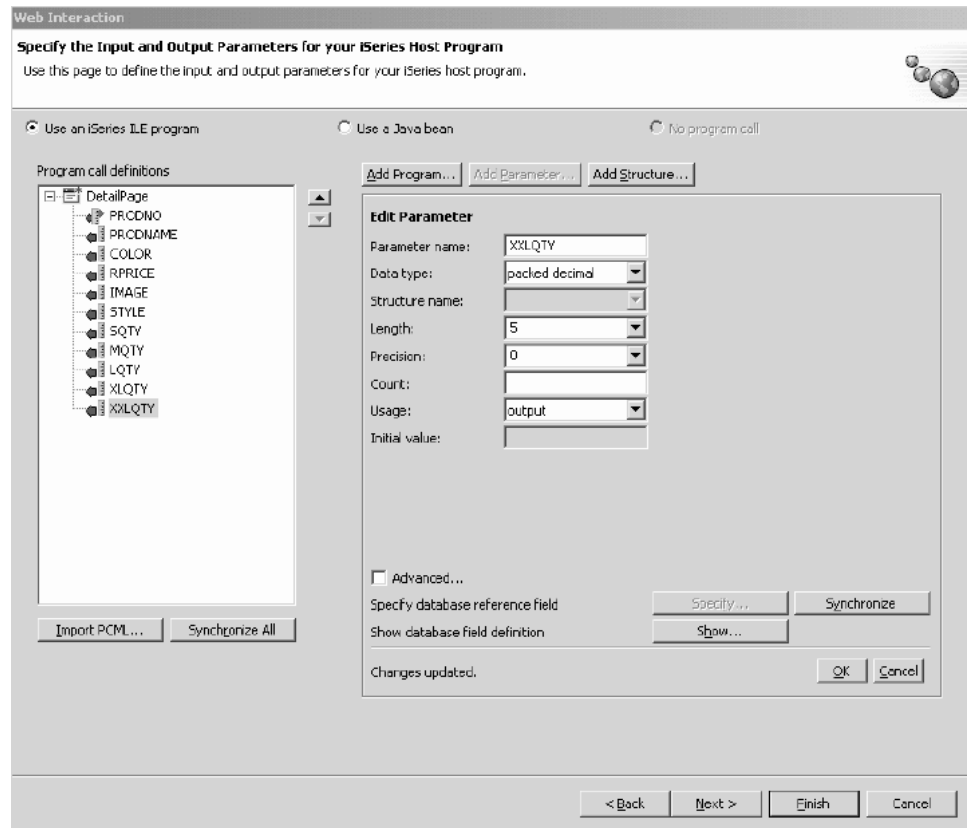


10. Click **Close**.

You now need to change the usage for 10 of the 11 parameters to "output" (all parameters except PRODNO).

1. Still in the Web Interaction wizard, click **PRODNAME** to select it.
2. 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.



3. Click **Next** to look at the **Input Form** preview.
4. Click **Finish** to create the Web interaction.
5. Select **Yes to all** or **OK** and **Finish** again 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:

1. In the Navigator view, expand **Project501 > Web Content > ClientScript**.
2. Double-click **webface.js** to open it in an editor.
3. 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");
}
```

4. 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 also enable 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. Furthermore, 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 **<i>Seriesserver > RETAILSTOR/QDDSSRC(QUERY)** and select **Open With > CODE Designer**.
4. After CODE Designer opens, expand **SCREEN1 > ITEMSUB**.
5. Click **IMAGESRC** to select it.
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
      ('/Project501/DetailPageServlet?PRODNO=&{PRODNO}')
```

Note especially that Project501 is specified in the last line. As mentioned earlier, if you want to run the SV000501 application again, you would need to change the Project501 value back to SV000501.

9. Save the file by clicking the save icon or by clicking **File > Save** from the menu bar.
10. Close the file and close CODE Designer.

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.

Publishing your files and restarting the server

In this section, you publish your files and restart so that your Project501 application can pick up all of your changes.

Before you restart the server, however, you need to verify that the Project501.war is added to the SVStoreEAR file.

To verify your configuration:

1. In the Navigator view, expand **Scenario folder**.
2. Double-click **Scenario server.wsi**.

You might see a dialog asking you if you want the editor to fix the project entries

automatically. Click **Yes**. If you do not see a dialog, you do not need to take any action. Either way, the file is automatically modified when it opens.

3. Save and close the file.

To restart the server:

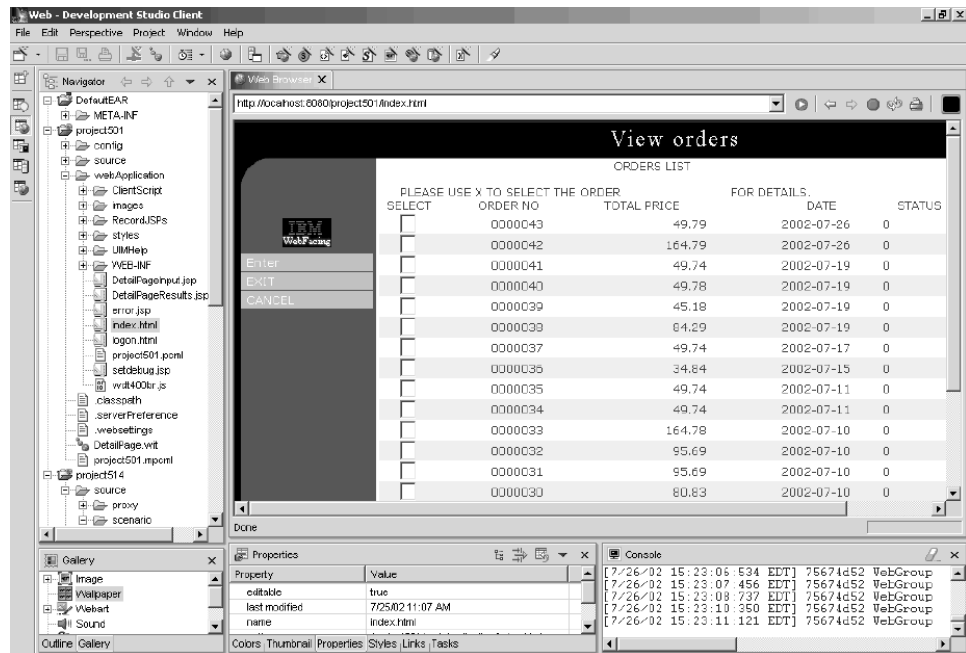
1. Switch to the Server perspective.
2. In the Server Configuration view, towards the bottom left, expand **Servers** and double-click **Scenario server**.
3. Click the **Servers** tab at the bottom right of the screen to see the status of your server in the Servers view.
4. Right-click **Scenario server** and select **Publish**. Click **OK** when publishing is complete.
5. Right-click your server in the Servers view and select **Start** or **Restart** (whichever is enabled). If you are prompted about unsaved files, cancel the dialog, save all open files, and proceed again.
6. Check the Console log information (opened automatically). You will know that the server is started when you see the following line at the bottom of the log:
Server *server_name* open for e-business.

Testing the interface

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

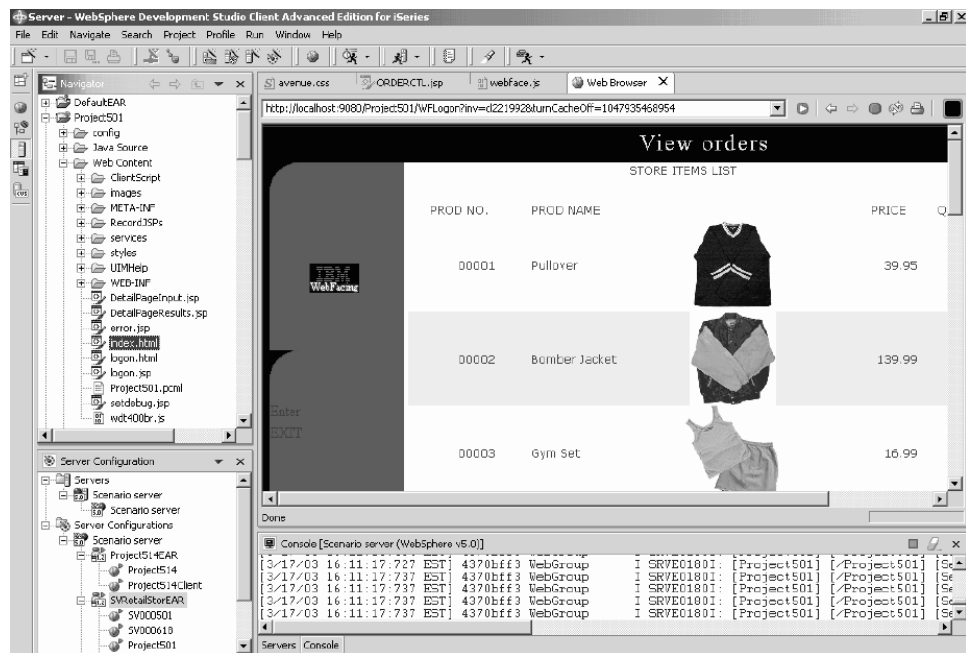
1. Switch to the Web perspective.
2. Expand **Project501 > Web Content**.
3. Right-click **index.html** and select **Run on Server**.
4. A **Server Selection** dialog opens. Verify that **Scenario server** is selected by default and click **Finish**.
5. Click **View orders – Launch in main browser window** to invoke the administrator application. After logging in with your iSeries user ID and password (for the iSeries server you used while developing this project), you

will be directed to the following page:



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

- 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 (after you log on to your iSeries server with your user ID and password):



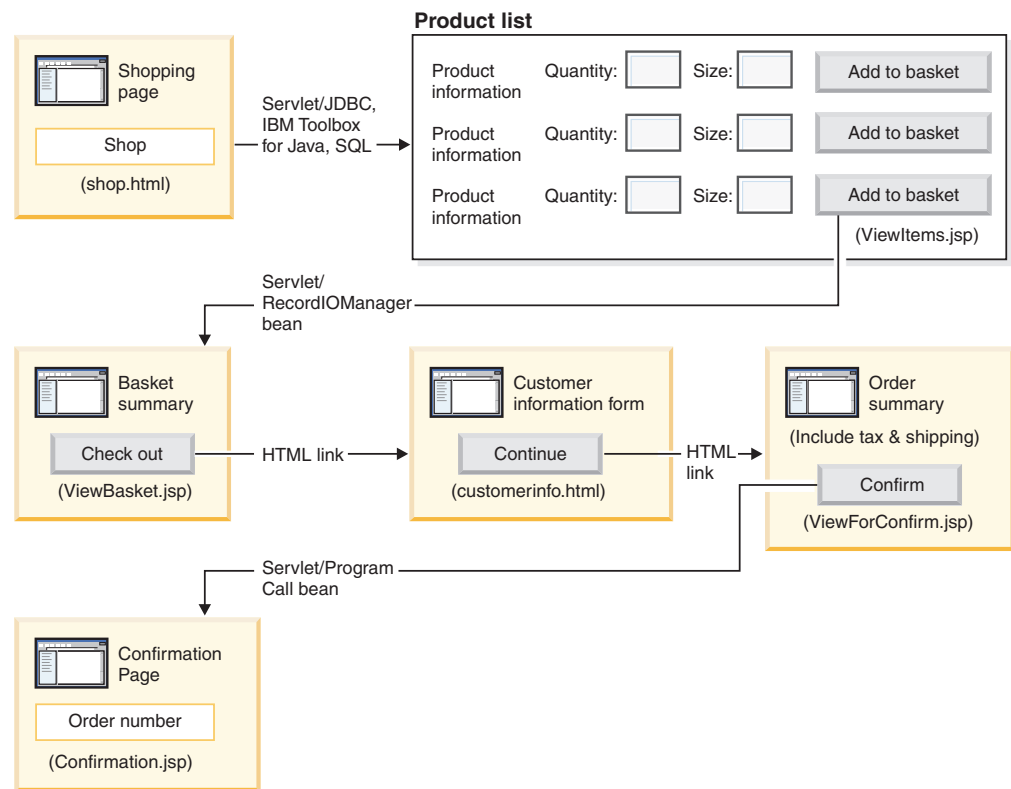
Try clicking an image to display details about a particular item.

You are now finished the module! You have created a user interface to view inventory and order items.

Chapter 6. Advanced module 1: Create HTML, servlets, and JSP files that place customer orders on an iSeries server (SV001585)

Introduction

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 server. 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 server 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.

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 server.
- You have started the iSeries 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 server.
- You have completed all the tasks in Chapter 3, "Running the scenario," on page 7, (except the optional WebSphere Application Server tasks, which are not required to test the application in the workbench).

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.ertools.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);
            setCommitLockLevel(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 updateDBFile(
            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(),
                    GetItems.getRetailLibrary());
            } catch (Exception e) {
                try {
                    res.sendRedirect("errorPage.html");
                    return;
                } catch (Exception e1) {
                    e1.printStackTrace();
                }
            }
        }
    }
}
}

```

```

}
id = req.getParameter("id");
size = req.getParameter(id + "S");
quantity = req.getParameter(id + "Q");

status = recIO.updateDBFile(id, size, quantity);
session.setAttribute("recIO", recIO);

if (status.equals("SUCCESS")) {
    basket.addItem(id, quantity, size);
    try {
        res.sendRedirect("ViewBasket.jsp");
        return;
    } catch (Exception e) {
        e.printStackTrace();
    }
} else {
    if (status.equals("NOT_ENOUGH")) {
        try {
            res.sendRedirect("insufficient.html");
            return;
        } catch (Exception e) {
            e.printStackTrace();
        }
    } else
        if (status.equals("ACCESS_ERROR")) {
            try {
                res.sendRedirect("errorPage.html");
                return;
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}

}

}

```

8. Write a ViewBasket JSP file that displays the contents of the basket.
9. Write a customerinfo.html form for the entering of payment information, including a **Continue** button that invokes ViewForConfirm.jsp.
10. Write a ViewForConfirm.jsp that shows contents of the entire basket plus the total balance. You can develop ViewForConfirm.jsp in the same manner as ViewBasket.jsp, with the addition of a calculated shipping charge and a **Confirm** button that places an order in the iSeries ORDERS database.
11. Use the iSeries Program Call wizard of iSeries Java development tools to create a PLACEORD.java bean, which accesses the PLACEODR service program in the RETAILSTOR library. The wizard creates beans for use by Java applications or the Web Services wizard to access iSeries ILE programs.
 - a. To open the wizard, right-click SV001585 in the Navigator view and select **New > Other**.
 - b. In the **New** window, click **iSeries > Java > Program Call Bean**.
 - c. In the Program Call wizard, you enter information about the iSeries ILE program name, library, type of program, input and output parameters.
 - 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 server 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
        rec10.commit();
        response.sendRedirect("orderNumber.jsp");
        return;
    } catch (Exception e) {
        response.sendRedirect("errorPage.html");
        e.printStackTrace();
    }
}
}
}

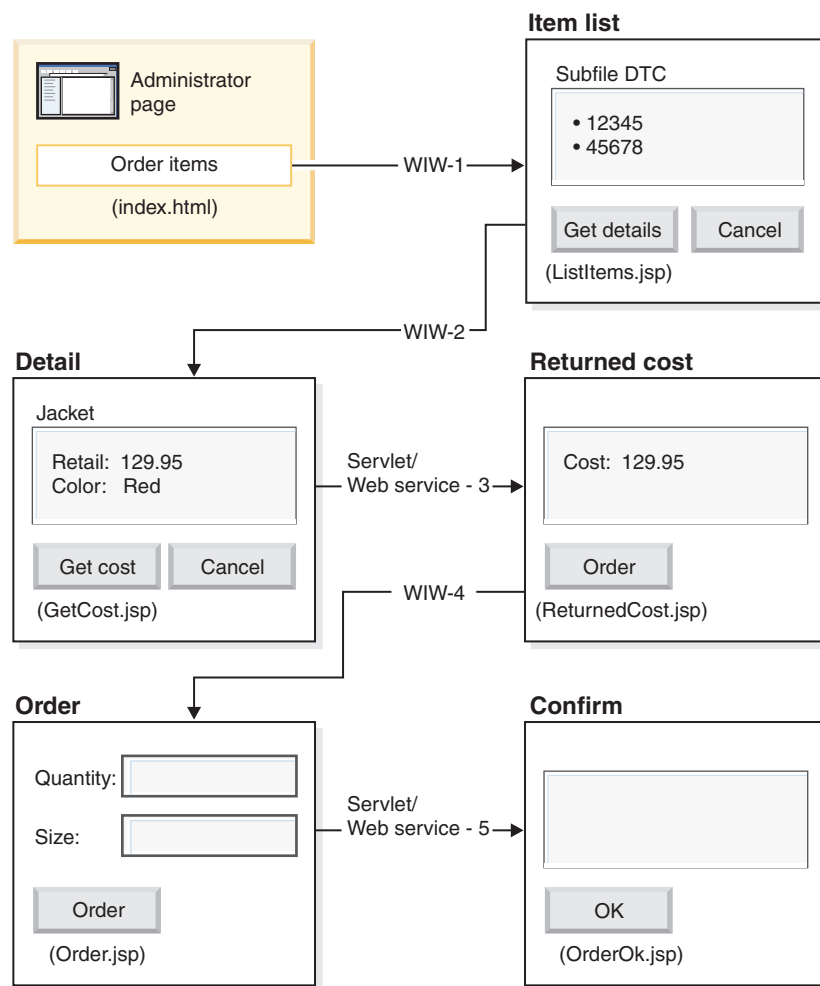
```

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.

Chapter 7. Advanced module 2: Create the Web project that uses the SV000514 and SV001586 Web services (SV000618)

Introduction

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 output from the Web Interaction wizard and various Web Components
- The Remote System 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 server through WebSphere Application Server

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 server.
- You have started the iSeries 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 server.
- You have completed all the tasks in Chapter 3, “Running the scenario,” on page 7, (except the optional WebSphere Application Server tasks, which are not required to test the application in the workbench).

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 **SV000618** 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 **SV000618** 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.

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 **SV000618 > source > proxy > soap**.
 2. Double-click **QryProdCostServicesProxy.java**.
 3. For QryProdCostServlet.java, double-click **QryProdCostServlet.java**, under **SV000618 > 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 **SV000618**.
 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:9080/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 15.
- 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:9080/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 15.
- 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

Note to U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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 Documentation 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 described in this Documentation. The furnishing of this Documentation does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing IBM Corporation North Castle Drive Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:
IBM World Trade Asia Corporation Licensing 2-31 Roppongi 3-chome, Minato-ku Tokyo 106, Japan

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 OR CONDITIONS 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.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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:

Lab Director IBM Canada Ltd. Laboratory 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 Documentation 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.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

(C) (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. (C) Copyright IBM Corp. 1992, 2002. All rights reserved.

Programming interface information


Programming interface information is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

- 400
- AFP
- AIX
- AIX windows
- APPN
- Application System/400
- AS/400
- AS/400e
- BookManager
- C Set ++
- C/400
- CICS
- CICS/400
- CICS/ESA
- COBOL/2
- COBOL/400
- Common User Access
- CUA
- DB2
- DB2 Extenders
- DB2 Universal Database
- 
- GDDM
- IBM
- IBMLink
- Integrated Language Environment
- iSeries
- Language Environment
- MQSeries
- Network Station
- Open Class
- Operating System/2
- Operating System/400
- OS/2
- OS/390
- OS/400
- POWER2
- PowerPC
- PROFS
- RPG/400
- RS/6000
- S/390
- SAA
- SQL/400
- System/36
- System/38
- VisualAge
- VTAM
- WebSphere

InstallShield is a trademark of InstallShield Corporation.

Intel and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

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

Rational is a trademark of International Business Machines Corporation and Rational Software Corporation, in the United States, other countries or both.

Lotus, Lotus Notes and Domino are trademarks of Lotus Development Corporation in the United States, other countries, or both.

ActiveX, Microsoft, SourceSafe, Visual C++, Visual SourceSafe, Windows, Windows NT, Win32, Win32s and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Netscape Navigator is a trademark of Netscape Communications Corporation.

UNIX is a registered trademark of The Open Group.

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



Program Number: 5724-A81

Printed in USA