IBM WebSphere Development Studio Client for iSeries

Build a Web user interface for an RPG application using iSeries Web Tools

Version 5.1.2

IBM WebSphere Development Studio Client for iSeries

Build a Web user interface for an RPG application using iSeries Web Tools

Version 5.1.2

Contents

Introduction v

Chapter 1. Module 1. Introducing the Web customer inquiry application . . . 1
Introducing iSeries Web Application Development . . 1
Introducing the Web customer inquiry application . . 2

Chapter 2. Module 2. Creating an RPG program. 7
Exercise 2.1: Starting the product 8
Exercise 2.2: Opening the Remote System Explorer perspective 9
Exercise 2.3: Creating a connection to the iSeries server 11
Exercise 2.4: Creating the IFS Directory for the Web project 12
Exercise 2.5: Finding an iSeries source member . . 14
Exercise 2.6: Opening a source member for edit . . 15
Exercise 2.7: Reviewing Remote System LPEX Editor features. 16
Exercise 2.8: Editing an RPG source member and creating a *PGM object 19

Chapter 3. Module 3. Creating the RPG service program for your Web application (optional) 25
Exercise 3.1: Starting the product 26
Exercise 3.2: Opening the Remote System Explorer perspective 27
Exercise 3.3: Creating a connection to the iSeries server 29
Exercise 3.4: Creating the IFS Directory for the Web project 30
Exercise 3.5: Finding an iSeries source member . . 32
Exercise 3.6: Open a source member for edit . . . 33
Exercise 3.7: Reviewing Remote System LPEX Editor features. 34
Exercise 3.8: Editing a member and creating a service program module 37

Chapter 4. Module 4. Creating a Web project. 45
Exercise 4.1: Opening the Web perspective 45
Exercise 4.2: Creating a dynamic Web project . . . 47
Exercise 4.3: Setting up the iSeries server information 51

Chapter 5. Module 5. Creating a Web application 57
Exercise 5.1: Invoking the Web Interaction wizard 58
Exercise 5.2: Specifying the input and output page 59
Exercise 5.3: Defining the iSeries GETDATA program invocation and parameters 61
Exercise 5.4: Defining the iSeries GETDATAS service program invocation and parameters 66
Exercise 5.5: Defining the input page content . . . 72
Exercise 5.6: Defining the output page content. . . 74
Exercise 5.7: Specifying error handling 76
Exercise 5.8: Enhancing the output page. 78
Exercise 5.9: Ensuring the customer number field cannot be modified. 82
Exercise 5.10: Visualizing the flow structure of your Web application 84

Chapter 6. Module 6. Running the Web application 89
Exercise 6.1: Opening the Web perspective 89
Exercise 6.2: Finding the Web application and running it 90

Chapter 7. Module 7. Debugging a Web application 95
Exercise 7.1: Introducing the iSeries Integrated Debugger 95
Exercise 7.2: Starting a Debug session using service entry points 96
Exercise 7.3: Adding and deleting breakpoints. . . 97
Exercise 7.4: Displaying a variable. 98
Exercise 7.5: Viewing the call stack 99
Exercise 7.6: Closing the debug session 99

Chapter 8. Module 8. Adding an error page 101
Exercise 8.1: Creating the flow control page . . . 101
Exercise 8.2: Modifying your Web interaction. . . 103
Exercise 8.3: Testing the new error page 110

Chapter 9. Module 9. Enhancing the input page using Web tools 113
Exercise 9.1: Opening Page Designer. 114
Exercise 9.2: Working with page properties . . . 115
Exercise 9.3: Linking a cascading style sheet to the Web page. 116
Exercise 9.4: Designing and adding a logo. . . . 118
Exercise 9.5: Adding a heading 1 tag to the page 127
Exercise 9.6: Adding a picture to the page. . . . 128
Exercise 9.7: Adding moving text to the page. . . 130
Exercise 9.8: Changing the text color. 134

Chapter 10. Summary 137

Appendix. Notices 139
Programming interface information 140
Trademarks 141

© Copyright IBM Corp. 2001, 2005

Introduction

This tutorial teaches you how to create a simple e-business customer inquiry application that uses a Web-based front end to communicate with the business logic written in ILE RPG residing on an iSeries™ server. While creating a browser user interface, you will learn how to use the iSeries Web Interaction wizard to generate input and output JSP files as well as Page Designer to enhance the input and output Web pages. You will also add iSeries Web components to your pages, for example, Web equivalents of iSeries command keys, input fields that accept only particular types of data, or output fields such as subfile names. You will then learn how to run the application in the WebSphere® Test Environment that is part of the product.

Prerequisites

In order to complete this tutorial end to end, you should already have working knowledge of the following:

- Basic Microsoft® Windows® operations such as working with the desktop and basic mouse operations such as opening folders and performing drag-and-drop operations.
- How Web applications work.
- How to use a browser to navigate the Internet.

It is also useful, but not necessary, for you to have basic knowledge of the following:

- DDS
- Servlets and Java™ Server Pages (to understand the generated output of the Web Interaction wizard)

The file(s) required for this tutorial are available for download at http://ibm.com/software/awdtools/wdt400/library.

Time required

To complete the modules of this tutorial, you will need approximately **2 hours and 20 minutes**.

Learning objectives

The tutorial is broken into 8 modules, each with its own learning objectives. You can choose to complete one or all of the modules. Each module contains several exercises that must be completed in order for the tutorial to work properly.

Chapter 1, “Module 1. Introducing the Web customer inquiry application,” on page 1 teaches you about the completed e-business customer inquiry application that you will build in this tutorial. In this module, you will:

- View a concept diagram of the Web development tools process
- Recognize the capabilities of the iSeries Web tools
- View the finished Web customer inquiry application

© Copyright IBM Corp. 2001, 2005

Chapter 2, “Module 2. Creating an RPG program,” on page 7 teaches you how to use the Remote System Explorer perspective to edit and compile an iSeries program that will be invoked by the Web application to get data from the iSeries. In this module, you will:

- Start the product
- Set the default workspace
- Check the Remote System Explorer perspective is open
- Set up a new connection to an iSeries server
- Set up an IFS folder to contain the compiled iSeries source member
- Locate and RPG source file you want to edit
- Edit an RPG source member with the Remote Systems LPEX Editor
- Highlight the tokens of your RPG source
- Show only comment lines in the RPG source
- Show all lines in the RPG source
- Display a format line
- Compile a program object
- Create a Program Call Markup Language (PCML) program interface description

Chapter 3, “Module 3. Creating the RPG service program for your Web application (optional),” on page 25 teaches you how to use the Remote System Explorer perspective to edit and compile an iSeries service program that will be invoked by the Web application to get data from the iSeries. In this module, you will:

- Start the product
- Set the default workspace
- Check the Remote System Explorer perspective is open
- Set up a new connection to an iSeries server
- Set up an IFS folder to contain the compiled iSeries source member
- Locate and RPG source file you want to edit
- Edit an RPG source member with the Remote Systems LPEX Editor
- Highlight the tokens of your RPG source
- Show only comment lines in the RPG source
- Show all lines in the RPG source
- Display a format line
- Compile a service program object
- Create a Program Call Markup Language (PCML) program interface description

Chapter 4, “Module 4. Creating a Web project,” on page 45 teaches you about the Web perspective and the tools in this perspective to help you create a Web project to contain your Web application files and to supply the information which the iSeries server will use for serving business information for this Web application. In this module, you will:

- Access tools and views for iSeries Web application development
- Know the difference between a static Web project and a dynamic Web project
- Set up a dynamic Web project
- Define where the RPG program or service program resides on an iSeries server

Chapter 5, “Module 5. Creating a Web application,” on page 57 teaches you how to create a Web interaction to use an input and output page, and to create the servlet

to invoke an RPG program to get data from the iSeries database. You will also learn how to visualize the flow structure of your Web application using the Web Diagram editor. In this module, you will:

- Start the Web Interaction wizard
- Create an input page that invokes the interaction
- Create an output page that displays the results of the interaction
- Get the program interface definitions
- Specify the input and output parameters
- Specify the input parameters to show on the input page
- Specify what data to show on the output page
- Specify what will happen when an incorrect customer number is entered
- Change the background color of the output page
- Make sure the customer number field is read only
- Know what Struts is all about
- View the flow structure of a Struts-based Web application

Chapter 6, “Module 6. Running the Web application,” on page 89 teaches you how to run the Web application in the WebSphere Test Environment. You also learn about the WebSphere Test Environment. In this module, you will:

- Check the Web perspective is open
- Locate input page of the Web application
- Invoke the Web application to run in the WebSphere Test Environment
- Test the Web application

Chapter 7, “Module 7. Debugging a Web application,” on page 95 teaches you about the iSeries integrated debugger, helps you learn how to start a debug session using service entry breakpoints, and helps you learn how to add and delete breakpoints, display a variable, view the call stack, run, step and step into a program, set watch breakpoints and close the debug session. In this module, you will:

- Recognize the features of the iSeries Integrated Debugger
- Set breakpoints
- Monitor an expression
- Expand a thread to show every program, module, procedure and method on the call stack at the current execution point
- Run the program to termination

Chapter 8, “Module 8. Adding an error page,” on page 101 teaches you how to create an informative error page for customers when an incorrect customer number is entered.

- Create an error page using Page Designer
- Change the Web interaction to use the new error page
- View the result of adding flow control to the Web application

Chapter 9, “Module 9. Enhancing the input page using Web tools,” on page 113 explains how to add color and some pictures to make the input page of the Web application more attractive. You will learn how to use the Page Designer tool and some other related Web tools. In this module, you will:

- Locate the Web application input page and start Page Designer

- View the title of the input page
- Add a style to the input page
- Preview the new style for the input page
- Start WebArt Designer
- Create a logo
- Resize the logo
- Save the object as a WebArt object
- Save the object for a Web page
- Place the object on the Design page
- Add a company name to the input page
- Add an image to the input page
- Add a marquee to the input page
- Apply color to certain areas of text

When you are ready, begin Chapter 1, “Module 1. Introducing the Web customer inquiry application,” on page 1.

Chapter 1. Module 1. Introducing the Web customer inquiry application

This module teaches you about the Web customer inquiry application that you will build in this tutorial. You will also learn about the iSeries Web tools included with the product.

In this module, you will:

- View a concept diagram of the Web development tools process
- Recognize the capabilities of the iSeries Web tools
- View the finished Web customer inquiry application

Exercises

The exercises within this module must be completed in order:

- "Introducing iSeries Web Application Development"
- "Introducing the Web customer inquiry application" on page 2

Time required

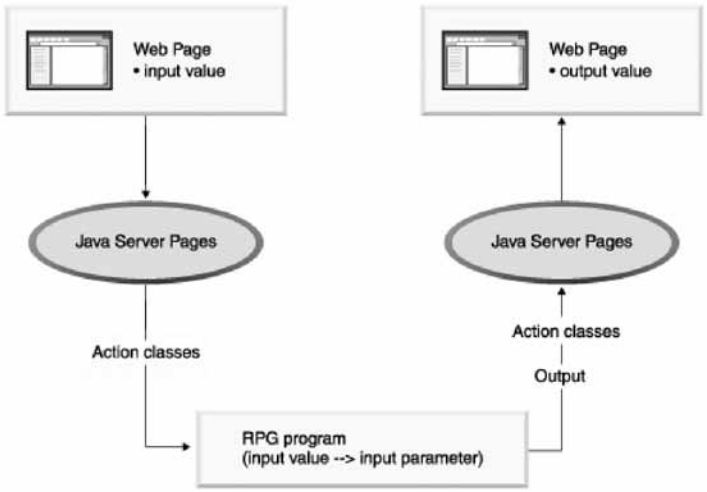
This module will take approximately **5 minutes** to complete.

Introducing iSeries Web Application Development

With the product you can make your iSeries applications and data accessible beyond the green-screen interface. You can create a new Web interface that connects directly to your program's input and output parameters.

You can create interactive Web pages using Web development tools. You use the Web Interaction wizard to define how your pages interact with one or more ILE or non-ILE applications. This wizard generates Java Action classes and JSP files for use with data from HTML forms. When the end user enters data in a form, the input becomes data to your ILE and non-ILE programs, and the output from the programs is formatted for the Web. Your ILE and non-ILE logic can be separated into different programs for each stage of input and output (known as a Web interaction), or can be a single service program with entry points to handle each Web interaction. The following diagram illustrates the Web development tools process:

© Copyright IBM Corp. 2001, 2005



You can easily customize your HTML and JSP files using the editing tools in Web development tools. You can use these tools to create and update input forms, change the appearance or placement of blocks of text, and add backgrounds and images to your pages. The iSeries-specific Web components help you create Web versions of your input and output pages with the same kinds of input validation, output formatting, and subfile controls that native DDS screens provide. You do not need a detailed knowledge of HTML or Java to accomplish these tasks.

You have an easy test mechanism for your Web-enabled applications. You can easily run your program in the WebSphere test environment of the workbench, quickly make changes, and retest, rather than redeploy your application every time you want to verify functions. When you are finished, you can package and deploy your Web applications as J2EE-based Web archive (WAR) and Enterprise Archive (EAR) files, and then install them in a WebSphere Application Server.

You have looked at the concept diagram of the Web development tools process and can recognize the capabilities of the iSeries Web tools and now you are ready to begin “Introducing the Web customer inquiry application.”

Introducing the Web customer inquiry application

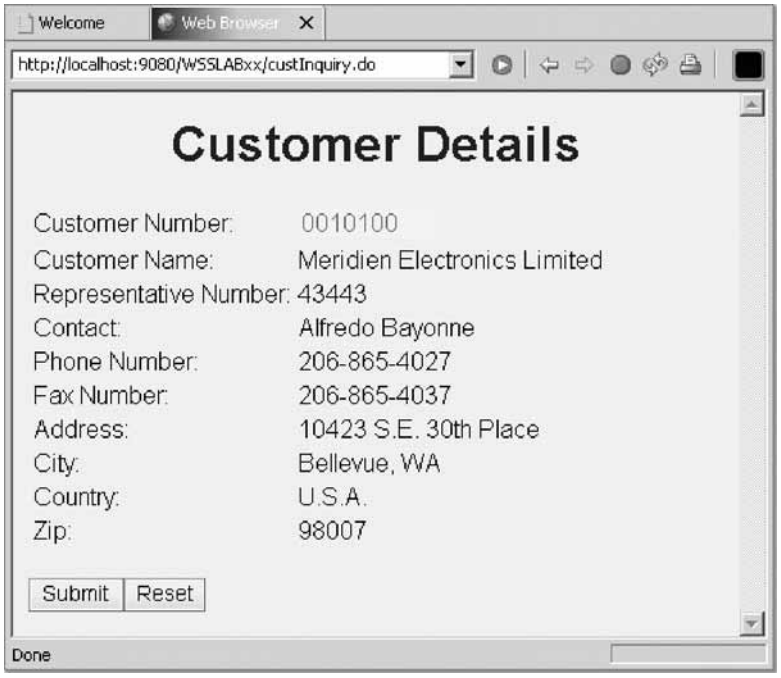
Before you begin, you must complete “Introducing iSeries Web Application Development” on page 1.

By the end of this tutorial you will have created a simple e-business customer inquiry application that uses a Web-based front-end to communicate with RPG business logic residing on an iSeries server. You will first create the RPG program or service program for your Web application. You will then create a Web project and invoke the Web Interaction wizard to create the input and output pages and to create the servlet to invoke the RPG program or service program to get data from the iSeries database. Next, you will run the application in the local test environment on your workstation. You will also create an informative error page for when a customer enters an incorrect customer number.

Here is what the finished e-business customer inquiry application looks like.

After the WebSphere server has started, your Web application Customer Inquiry input page (inquiry.jsp) will display in the workbench browser:

When you enter a customer number and click the **Submit** push button, the Customer Details output page (result.jsp) appears.



Next, you click the Back button in the browser to return to the Customer Inquiry input page. Here you enter an incorrect customer number and click **Submit**.

An error message appears.

You then add a new output error page to improve the message information so that if a wrong customer number is entered,

an error page opens instead of a message.

Module recap

You have completed Chapter 1, "Module 1. Introducing the Web customer inquiry application," on page 1. You have learned how to:

- View a concept diagram of the Web development tools process
- Recognize the capabilities of the iSeries Web tools
- View the finished Web customer inquiry application

Now that you have this introductory knowledge of the product iSeries Web application development, you can continue with Chapter 2, "Module 2. Creating an RPG program," on page 7 if you want to work with an RPG program or Chapter 3, "Module 3. Creating the RPG service program for your Web application (optional)," on page 25 if you want to work with an RPG service program.

Chapter 2. Module 2. Creating an RPG program

This module teaches you how to create the iSeries program that you will be using for your Web application using the Remote System Explorer perspective. Remote System Explorer provides a PDM like environment but with a GUI interface, and full integration with all iSeries programmer tools in the workbench.

In this module, you will:

- Start the product
- Set the default workspace
- Check the Remote System Explorer perspective is open
- Set up a new connection to an iSeries server
- Set up an IFS folder to contain the compiled iSeries source member
- Locate an RPG source file you want to edit
- Open an RPG source member with the Remote Systems LPEX Editor
- Highlight the tokens of your RPG source
- Show only comment lines in the RPG source
- Show all lines in the RPG source
- Display a format line
- Edit an RPG source member with the Remote Systems LPEX Editor
- Compile a program object
- Create a Program Call Markup Language (PCML) program interface description

If you want to work with a procedure in a service program instead, go to Chapter 3, "Module 3. Creating the RPG service program for your Web application (optional)," on page 25.

If you haven't used the LPEX editor before, you may want to spend some time trying some LPEX editor features.

Exercises

The exercises in this module must be completed in order:

- "Exercise 2.1: Starting the product" on page 8
- "Exercise 2.2: Opening the Remote System Explorer perspective" on page 9
- "Exercise 2.3: Creating a connection to the iSeries server" on page 11
- "Exercise 2.4: Creating the IFS Directory for the Web project" on page 12
- "Exercise 2.5: Finding an iSeries source member" on page 14
- "Exercise 2.6: Opening a source member for edit" on page 15
- "Exercise 2.7: Reviewing Remote System LPEX Editor features" on page 16
- "Exercise 2.8: Editing an RPG source member and creating a *PGM object" on page 19

Time required

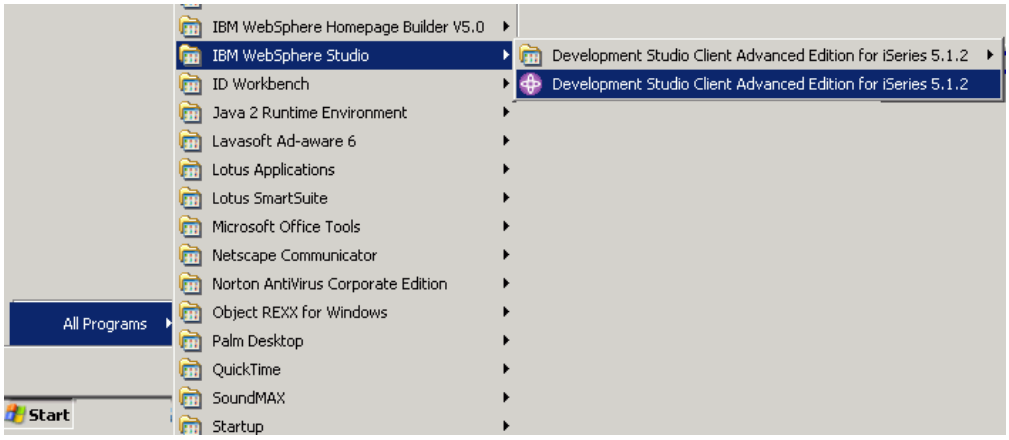
This module will take approximately **30 minutes** to complete.

© Copyright IBM Corp. 2001, 2005

Exercise 2.1: Starting the product

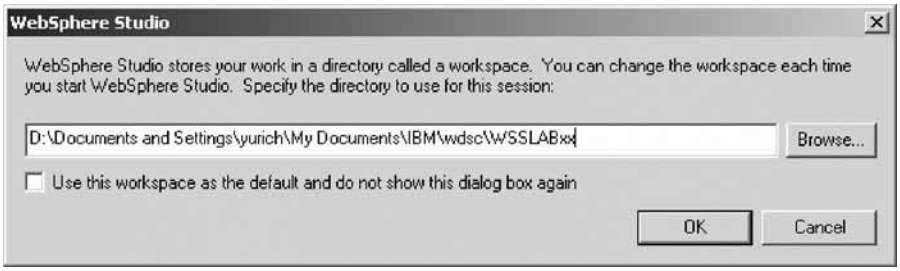
Now go ahead and start the product.

On your desktop taskbar:



1. Click **Start > All Programs > IBM® WebSphere Studio > Development Studio Client Advanced Edition for iSeries 5.1.2**

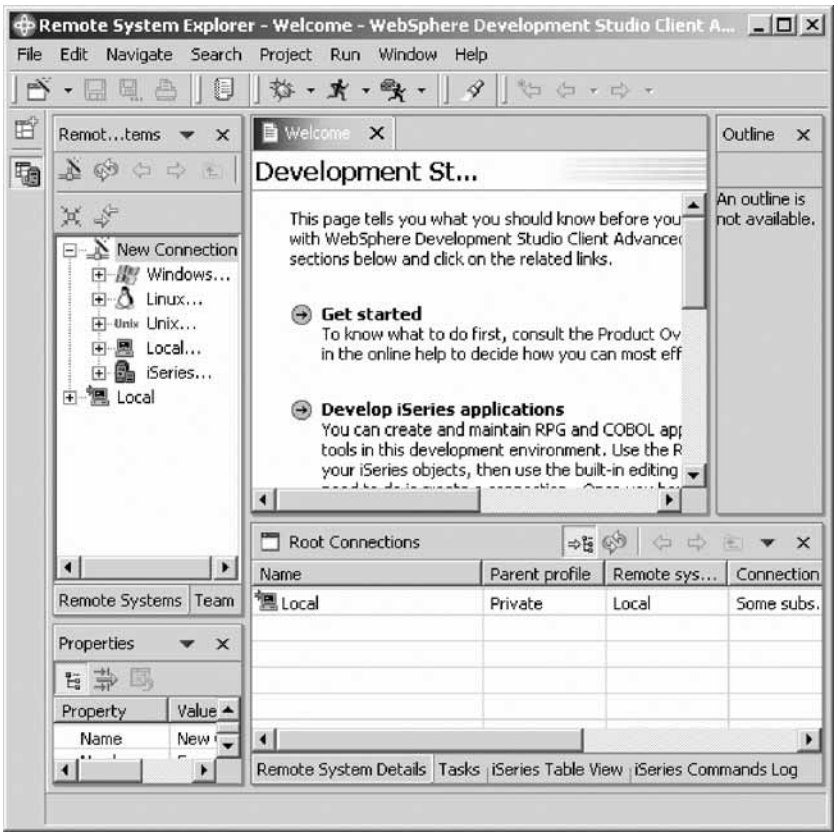
   A dialog for workspace selection will appear asking you for the workspace location, unless you used the product before and selected not to show this dialog again.



   The workspace contains all the information about your Development Studio projects. You can accept the default or store the work related to this tutorial, in a separate workspace.

2. To name the directory of the workspace as shown above, use the directory name `WSSLABxx` where XX is your unique team number.
3. Click **OK**.

   After a few moments of loading, the workbench opens, and the initial window of the product appears on your desktop.



You have just started the product, set the default workspace and now you are ready to begin "Exercise 2.2: Opening the Remote System Explorer perspective."

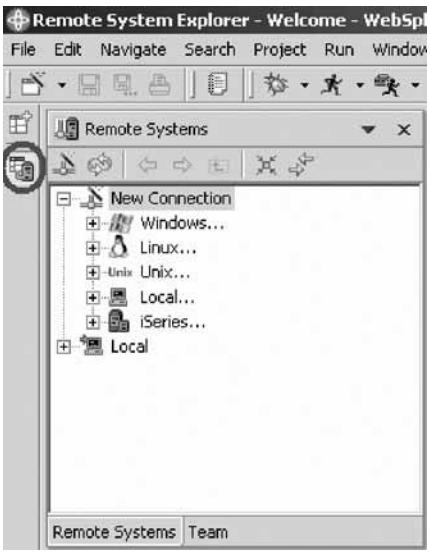
Exercise 2.2: Opening the Remote System Explorer perspective

Before you begin, you must complete "Exercise 2.1: Starting the product" on page 8.

Check if the window title bar says: Remote System Explorer. If it does you are all set. Skip the next steps and go directly to "Exercise 2.3: Creating a connection to the iSeries server" on page 11.

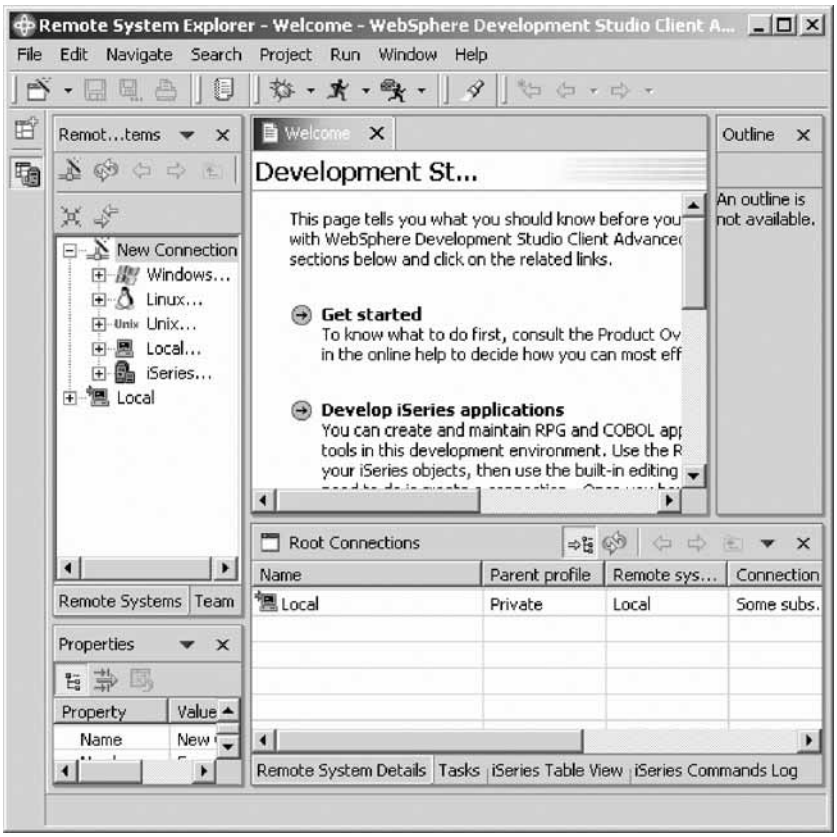
To open the Remote System Explorer perspective:

1. In the workbench, check whether you have the Remote System Explorer perspective already open. Look for the icon in the left taskbar of the workbench.
2. If you see it click the RSE icon.



3. Otherwise, click **Window > Open Perspective > Remote System Explorer** on the workbench menu.

   The Remote System Explorer perspective opens.

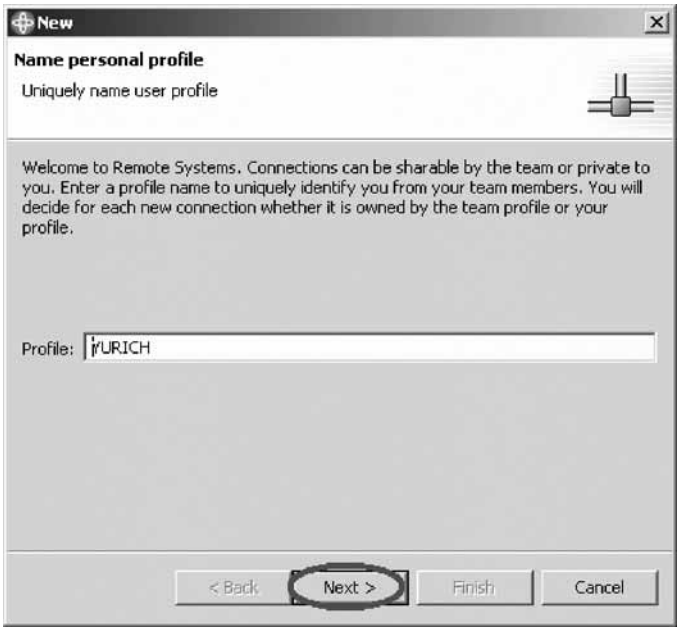


You have checked that the Remote System Explorer perspective is open and now you are ready to begin “Exercise 2.3: Creating a connection to the iSeries server” on page 11.

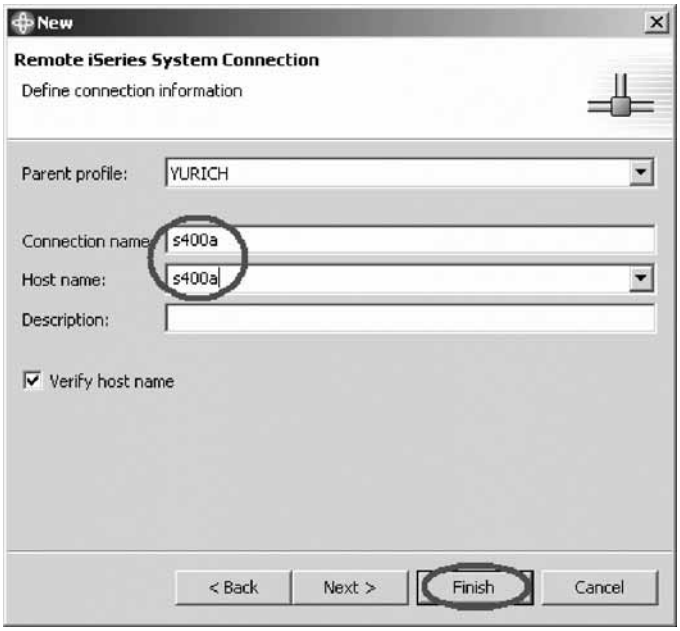
Exercise 2.3: Creating a connection to the iSeries server

Before you begin, you must complete “Exercise 2.2: Opening the Remote System Explorer perspective” on page 9.

1. Click the plus sign + beside **iSeries** under **New Connection** in the Remote Systems view.
2. Accept the default profile name.



3. Click **Next**.



In this exercise s400a is used as the name for the iSeries server. You will need to use the name or IP address of your particular iSeries server.

4. Type a host name, for example, s400a in the **Host name** field.
5. The **Connection name** field will be filled automatically with the same entry.
6. Click **Finish**.

Your new connection appears in the Remote Systems view and you are ready to work with objects on this iSeries system.

You have set up a new connection to an iSeries server and now you are ready to begin "Exercise 2.4: Creating the IFS Directory for the Web project."

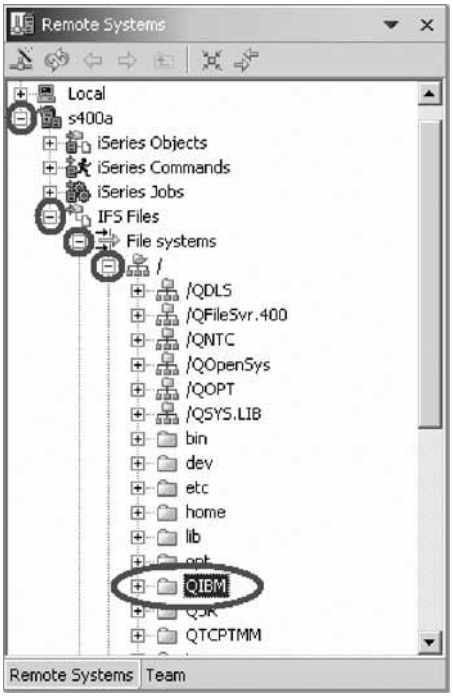
Exercise 2.4: Creating the IFS Directory for the Web project

Before you begin, you must complete "Exercise 2.3: Creating a connection to the iSeries server" on page 11.

IFS Files allow you to explore folders and files in the Integrated File System of the remote iSeries system. You will now create a directory in the IFS that will be used when you compile your iSeries source member.

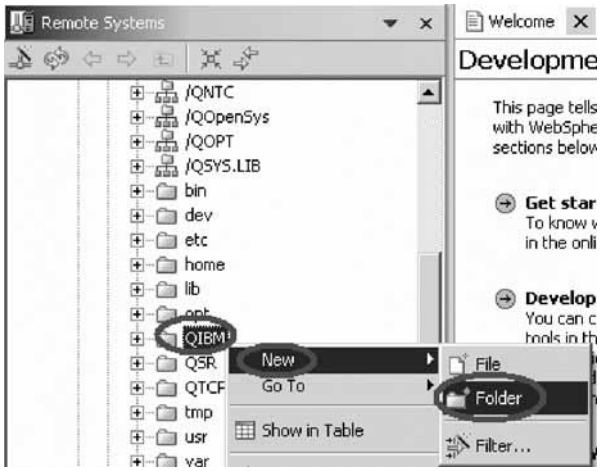
Locate the newly created iSeries connection.

1. Expand your iSeries connections.

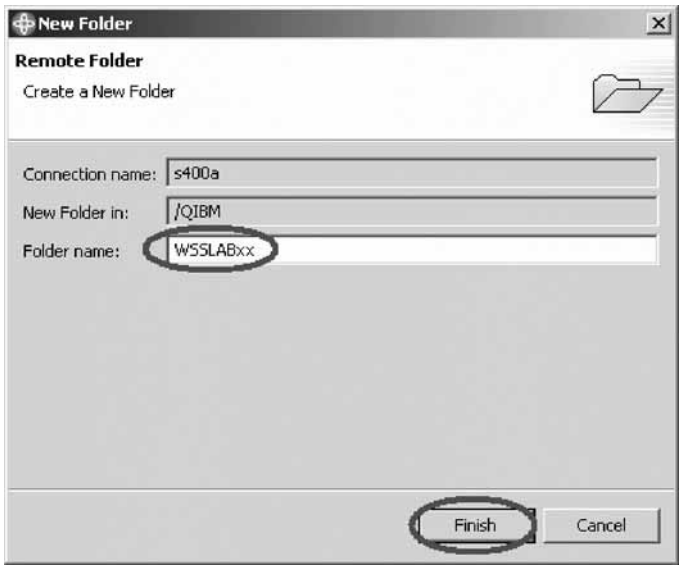


2. Expand the **IFS Files** by clicking the plus sign + beside it.
3. Expand the **File systems**.
4. If prompted, type your **User ID** and **Password**.

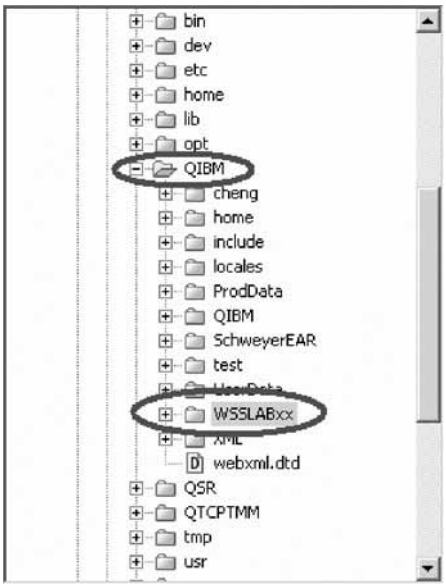
5. Select the **Save user ID** check box.
6. Select the **Save password** check box.
7. Expand the / Root node.
8. Select **QIBM**. Right-click **QIBM** and click **New > Folder** on the pop-up menu.



9. Enter WSSLABxx where XX is your unique team number in the **Folder name** field.



10. Click **Finish**.



You should see the new WSSLABxx directory under QIBM.

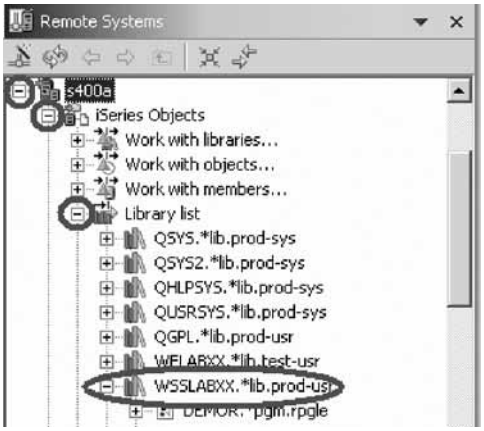
You have set up an IFS folder to contain the compiled iSeries source member and now you are ready to begin "Exercise 2.5: Finding an iSeries source member."

Exercise 2.5: Finding an iSeries source member

Before you begin, you must complete "Exercise 2.4: Creating the IFS Directory for the Web project" on page 12.

You will see a list of libraries under Library list in the Remote Systems view. Now you locate the source you want to edit.

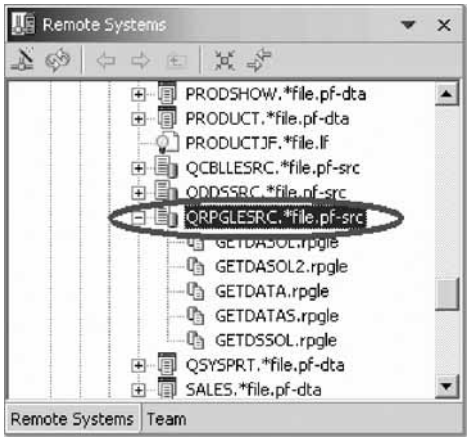
To find an iSeries source member:



Note: If you don't see WSSLABxx in your library list, then you need to add the library. You can right-click the Library list and click Add Library List Entry from the pop-up menu to add WSSLABxx to your library list.

1. Expand your iSeries connections.
2. Expand **iSeries Objects** by clicking the plus sign + beside it.

3. Expand **Library List**.
4. Expand the library WSSLABxx.
   You will see all objects in this library.
5. Expand the source file QRPGLESRC.



6. Scroll down to the source members in the expanded list.

You have located an RPG source member you want to edit and now you are ready to begin "Exercise 2.6: Opening a source member for edit."

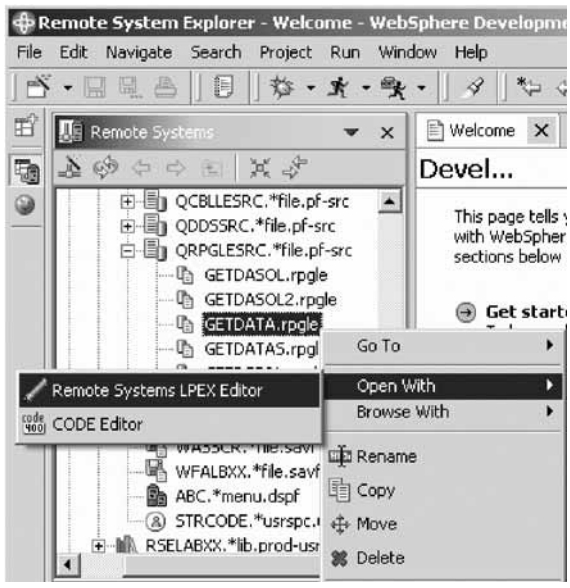
Exercise 2.6: Opening a source member for edit

Before you begin, you must complete "Exercise 2.5: Finding an iSeries source member" on page 14.

A shell source member is provided for you to use.

To edit a source member:

1. Right-click member **GETDATA**.
2. Click **Open with > Remote Systems LPEX Editor** on the pop-up menu.



The Editor window opens and you are ready to write your program.

Note: You can also double-click on a member to open the Remote Systems LPEX editor.

You have opened an RPG source member for edit and now you are ready to begin "Exercise 2.7: Reviewing Remote System LPEX Editor features."

Before you work with the RPG program source, if you have not worked with the Remote Systems LPEX editor, go to "Exercise 2.7: Reviewing Remote System LPEX Editor features."
If you already know the LPEX editor, then you are ready to write the RPG program; go to "Exercise 2.8: Editing an RPG source member and creating a *PGM object" on page 19.

Exercise 2.7: Reviewing Remote System LPEX Editor features

Before you begin, you must complete "Exercise 2.6: Opening a source member for edit" on page 15.

Before you start, here are some useful tips on using the LPEX editor. Let's look at some of the features of the LPEX editor, so you can later easily find your way around and use them:

- **ALT+L** marks a line
- **ALT+U** unmarks selection
- **ALT+D** deletes a selected text
- **ALT+C** copies selected text
- **ALT+M** moves selected text
- **Enter** inserts a new line

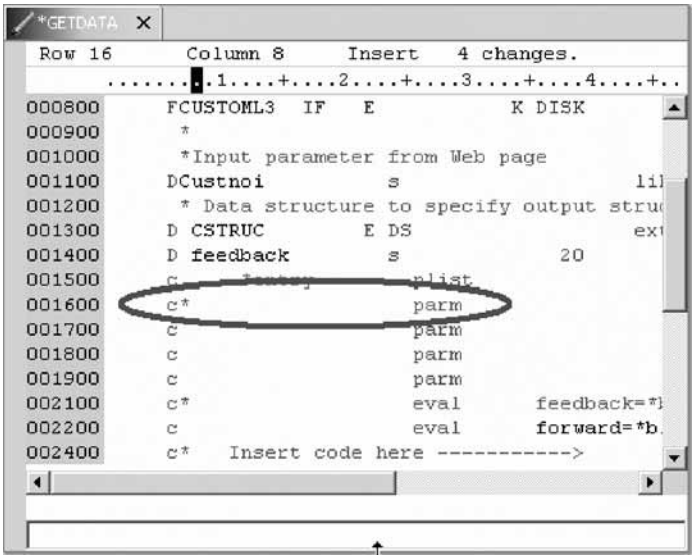
Highlighting specification fields

The LPEX editor highlights the Tokens (specification fields) of your RPG program source, providing a separate color for each, improving readability. When you make changes to a line, the token colors get updated only after you move your cursor off the line.

To see how token highlighting works:

1. Move the cursor to a Calculation statement.
2. Position the cursor to column 7 (right next to the C).
3. Type an asterisk (*).
4. Move the cursor off the line and watch what happens.

   The line where you typed the asterisk (*) becomes a comment line and its color changes accordingly.



5. Move the cursor back to column 7 and remove the asterisk (*).
6. Move the cursor off that line of code and the statement is tokenized.

   The token highlighting changes to reflect that this is a non-commented C specification.

Displaying types of lines

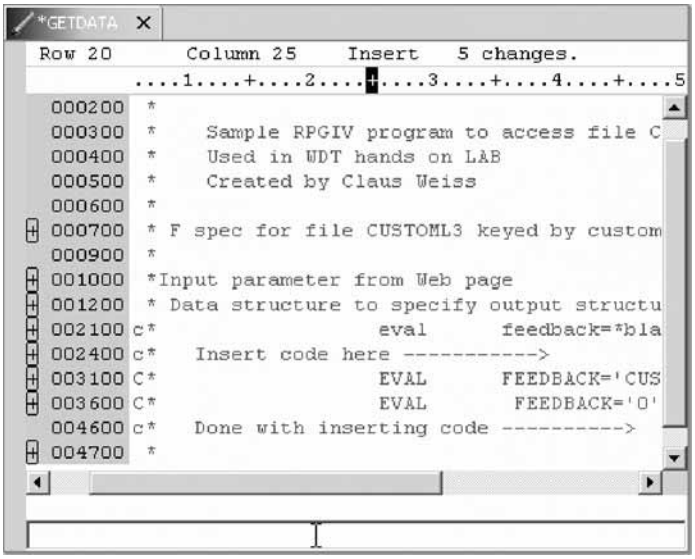
Using the LPEX editor, it is possible to have only particular types of source lines displayed at a time.

To display types of lines:

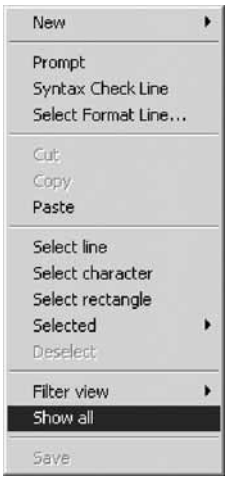
1. Right-click anywhere in the Editor window.
2. Click **Filter view** on the pop-up menu.

   The menu items list the types of line selections that can be made.
3. Click **Comments**.

   The Editor window now contains only those RPG statements that are comments.



4. Right-click anywhere in the Editor window.
5. Click **Show all** on the pop-up menu.



All statements types are displayed. In addition, the choices in the pop-up menu can be used to include only control specifications, user subroutines, and procedures and others. The Filter Selection option under the Selected option in the Edit menu allows the selection of only those lines containing a particular selected character string.

Checking the syntax of a file

Syntax checking is available for RPG code, and by default will be active. The syntax of RPG code is checked automatically when a change is made to a line, and the cursor is moved off the line.

When errors are found, they are displayed following the statement with the error.

Keeping track of columns in a specification line

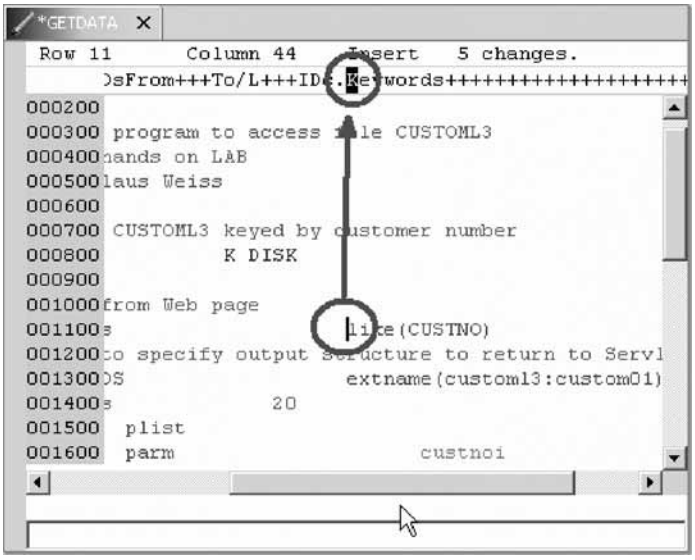
The format line is at the top of the Editor window, just above the first statement. A format line is used to help keep track of the columns in a particular specification line. The content of the format line can vary to reflect the particular type of specification being keyed such as F specs, C specs, D specs and so on.

To display a format line:

1. Click a line or use the arrow keys and click the left mouse button to make a line active.

   The format line gets updated as a line gets focus.

   You can move the cursor right or left with the arrow keys to go from character to character, or with the **Tab** key to go from field to field. An indicator on the format line moves with the cursor to show in which column the cursor is positioned.



You can select a format line for any specification you want.

2. Click **Source** from the Editor menu.
3. Click **Select Format Line** on the pop-up menu.

   The Format Line Selection window opens.

Now you can start and create the RPG program.

4. Click **OK**.

You have highlighted tokens of your RPG source, shown only comment lines, shown all lines, displayed a format line and now you are ready to begin "Exercise 2.8: Editing an RPG source member and creating a *PGM object."

Exercise 2.8: Editing an RPG source member and creating a *PGM object

Before you begin, you must complete "Exercise 2.6: Opening a source member for edit" on page 15 and if you were unfamiliar with the editing features of the Remote Systems LPEX Editor then you must complete "Exercise 2.7: Reviewing Remote System LPEX Editor features" on page 16.

To make it easier for you, most of the RPG source is already prepared for you. You have already opened the correct source member GETDATA in the previous exercise using the Remote Systems view.

Read through the source. It contains an F spec to access file CUSTOML3, which contains the customer data keyed by customer number. The D specs define the data structure CSTRUC that you pass back to your Web page and the CUSTNOI variable that gets passed from the Web page to this program. As well the Feedback variable is defined as a 20 length character field.

The Entry parameter list is defined as:

- CUSTNOI field (this is the input parameter).
- CSTRUC structure (this the data structure for the customer output data).
- Feedback field to indicate no success for file access.

  The code you have to write here, fetches a customer record by chaining to the file with the CUSTNOI that gets passed into the program.

To edit then create a *PGM object with RPG:

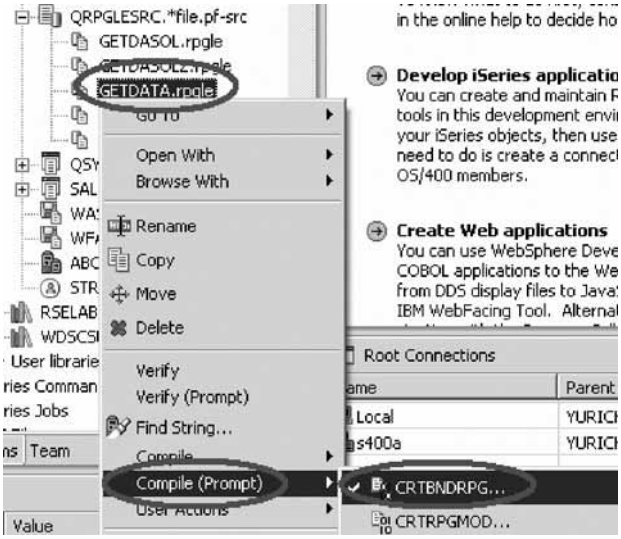
1. Add the code to get the customer record and check whether the chain was successful.

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
C     CUSTNOI       CHAIN(e)  CUSTOML3
C                   IF        not %found(CUSTOML3)
C                   EVAL      FEEDBACK='CUS0001 ' + CUSTNOI
C                   ELSE
C                   EVAL      FEEDBACK='0'
C                   ENDIF
```

   Your coding is complete, so let's create the program.

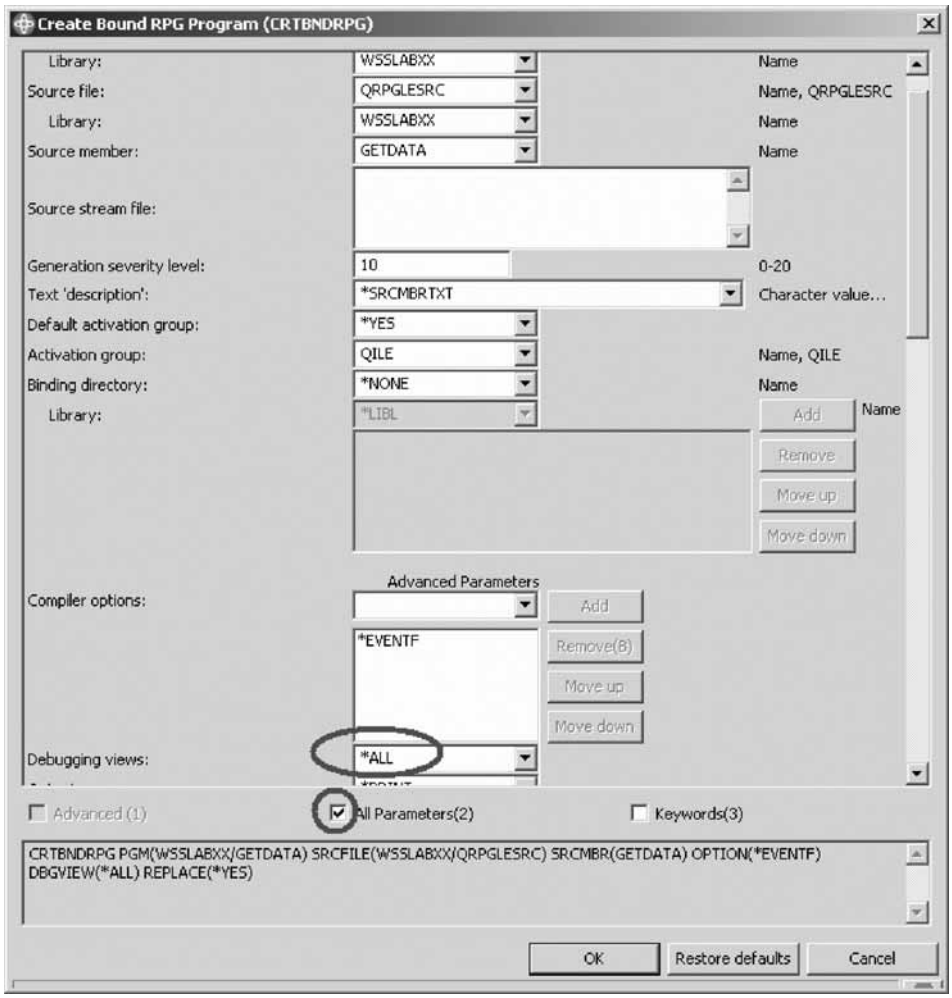
2. Click the Save icon on the workbench toolbar to save the member.
3. Click the **X** in the Editor window title bar to close the member.

   Be careful not to click the X on the workbench window title bar.



4. In the Remote Systems view, right-click the GETDATA member in source file QRPGLESRC and click **Compile (Prompt) > CRTBNDRPG** on the pop-up menu.

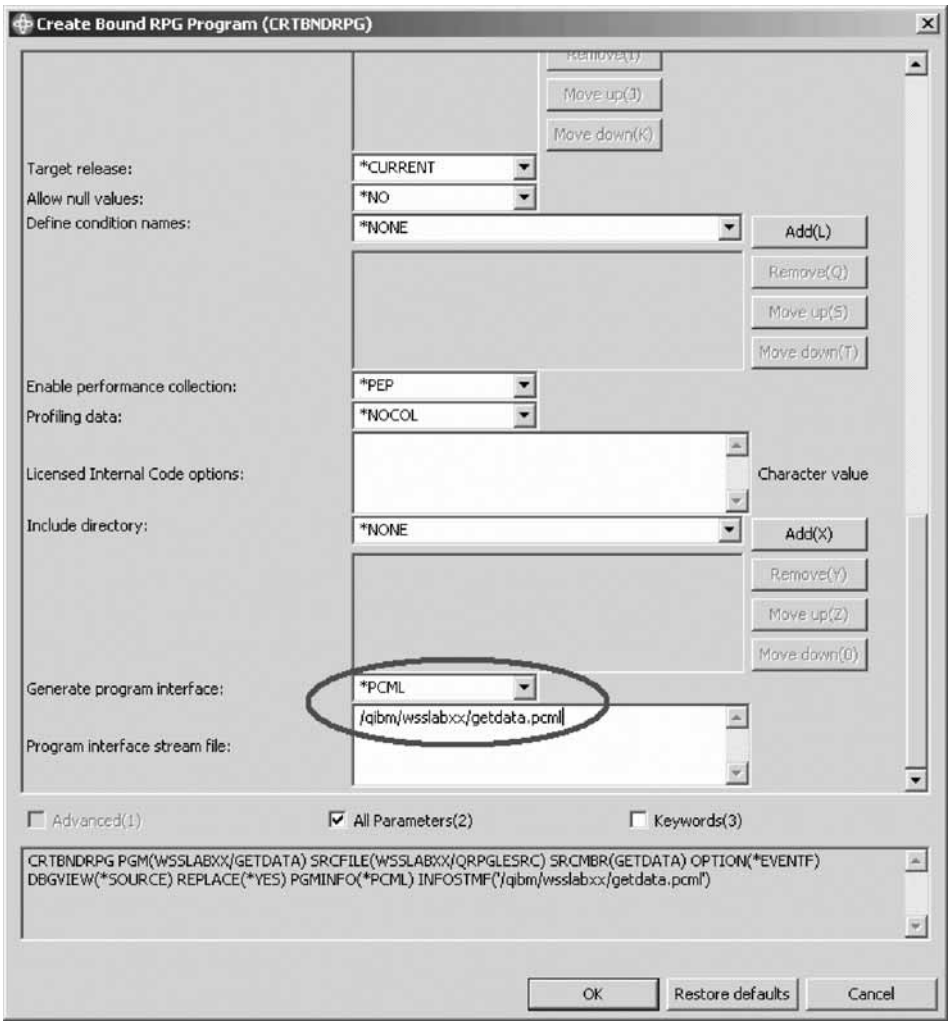
   The Create Bound RPG Program (CRTBNDRPG) dialog opens.



5. In the **Debugging views** list, select ***ALL**

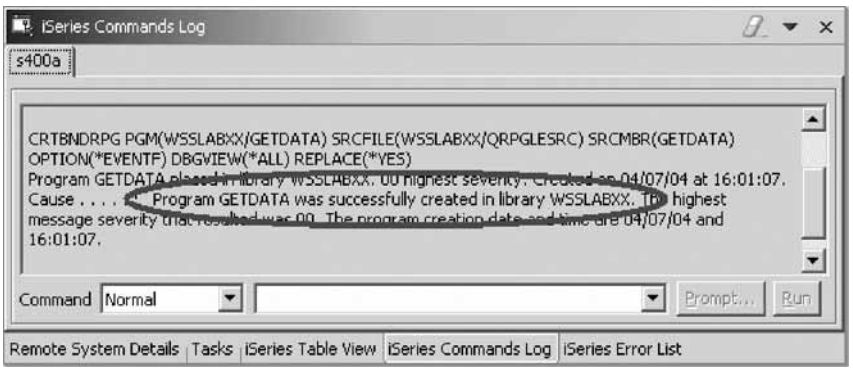
   You need to specify two more parameters, these parameters are used to tell the compiler to generate the program interface description in a language called Program Call Markup Language (PCML). Later on you can use this description in the Web Interaction wizard.

6. Check the **All Parameters** check box at the bottom of the dialog.
7. Scroll down in the dialog until you see the **Generate program interface** field.
8. Select the ***PCML** value from the list.



9. Enter `/qibm/wsslabxx/getdata.pcml` for the **Program interface stream file** field.
10. Click **OK** to start to compile.

    After compile, the iSeries Error List view is shown, listing all compile errors. You may see only information and warning messages which means your compile was completed and the program object was created.



11. Click the **iSeries Commands Log** tab to check that the compile was successful. This log shows the Remote System Explorer job messages.

If you get a message that there are errors in your program, go through the edit compile fix cycle.

Fixing errors

In the error list view, check the errors:

1. Double-click the error.
   This positions the cursor in the Editor window on the statement that is wrong.
2. Fix all errors.
3. Save the source member.
4. Go back to the Remote Systems view.
5. Right-click the member to run the CRTBNDRPG command.
6. Continue this cycle until you get a clean compile.

You have edited an RPG source member and created a program object.

Module recap

You have completed Chapter 2, "Module 2. Creating an RPG program," on page 7. You have learned how to:

- Start the product
- Set the default workspace
- Check the Remote System Explorer perspective is open
- Set up a new connection to an iSeries server
- Set up an IFS folder to contain the compiled iSeries source member
- Locate an RPG source file you want to edit
- Open an RPG source member with the Remote Systems LPEX Editor
- Highlight the tokens of your RPG source
- Show only comment lines in the RPG source
- Show all lines in the RPG source
- Display a format line
- Edit an RPG source member with the Remote Systems LPEX Editor
- Compile a program object
- Create a Program Call Markup Language (PCML) program interface description

Now that you have created the RPG program, you can continue to Chapter 4, "Module 4. Creating a Web project," on page 45.

Skip Chapter 3, "Module 3. Creating the RPG service program for your Web application (optional)," on page 25 as you don't need to do this set of exercises.

Chapter 3. Module 3. Creating the RPG service program for your Web application (optional)

You do not need to do this module if you have done Chapter 2, "Module 2. Creating an RPG program," on page 7.

This module teaches you how to add the iSeries service program to your Web application using the Remote System Explorer perspective. Remote System Explorer provides a PDM like environment but with a GUI interface, and full integration with all iSeries programmer tools in the product workbench.

In this module, you will:

- Start the product
- Set the default workspace
- Check the Remote System Explorer perspective is open
- Set up a new connection to an iSeries server
- Set up an IFS folder to contain the compiled iSeries source member
- Locate an RPG source file you want to edit
- Open an RPG source member with the Remote Systems LPEX Editor
- Highlight the tokens of your RPG source
- Show only comment lines in the RPG source
- Show all lines in the RPG source
- Display a format line
- Edit an RPG source member with the Remote Systems LPEX Editor
- Compile a service program object
- Create a Program Call Markup Language (PCML) program interface description

If you haven't used the LPEX editor before, you may want to spend some time trying some LPEX editor features.

Exercises

The exercises in this module must be completed in order:

- "Exercise 3.1: Starting the product" on page 26
- "Exercise 3.2: Opening the Remote System Explorer perspective" on page 27
- "Exercise 3.3: Creating a connection to the iSeries server" on page 29
- "Exercise 3.4: Creating the IFS Directory for the Web project" on page 30
- "Exercise 3.5: Finding an iSeries source member" on page 32
- "Exercise 3.6: Open a source member for edit" on page 33
- "Exercise 3.8: Editing a member and creating a service program module" on page 37

Time required

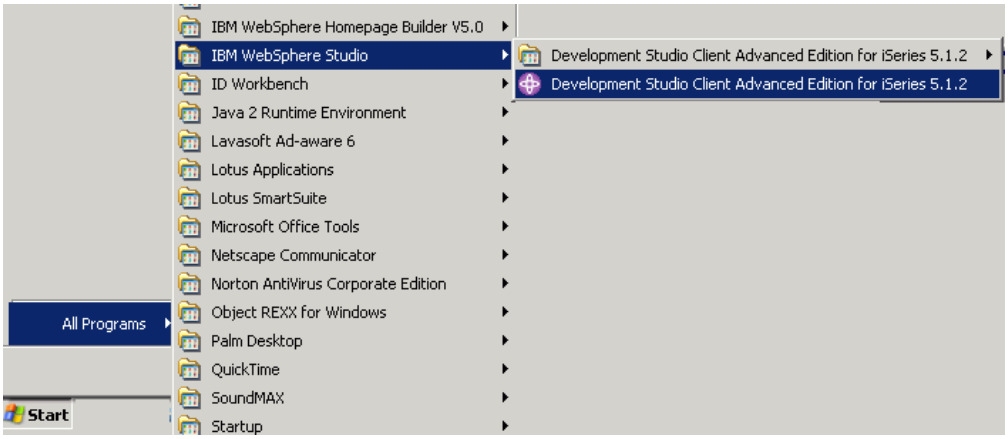
This module will take approximately **30 minutes** to complete.

© Copyright IBM Corp. 2001, 2005

Exercise 3.1: Starting the product

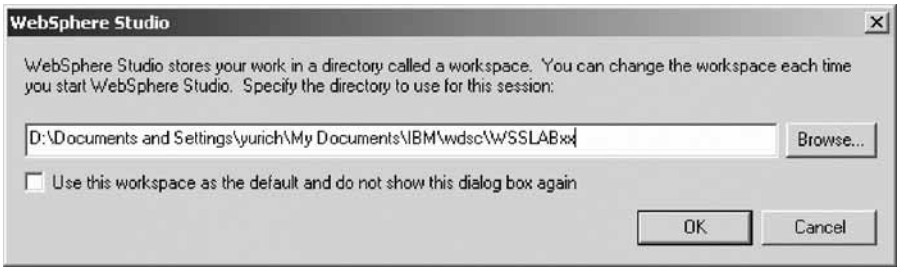
Now go ahead and start the product.

On your desktop taskbar:



1. Click **Start > All Programs > IBM WebSphere Studio > Development Studio Client Advanced Edition for iSeries 5.1.2**

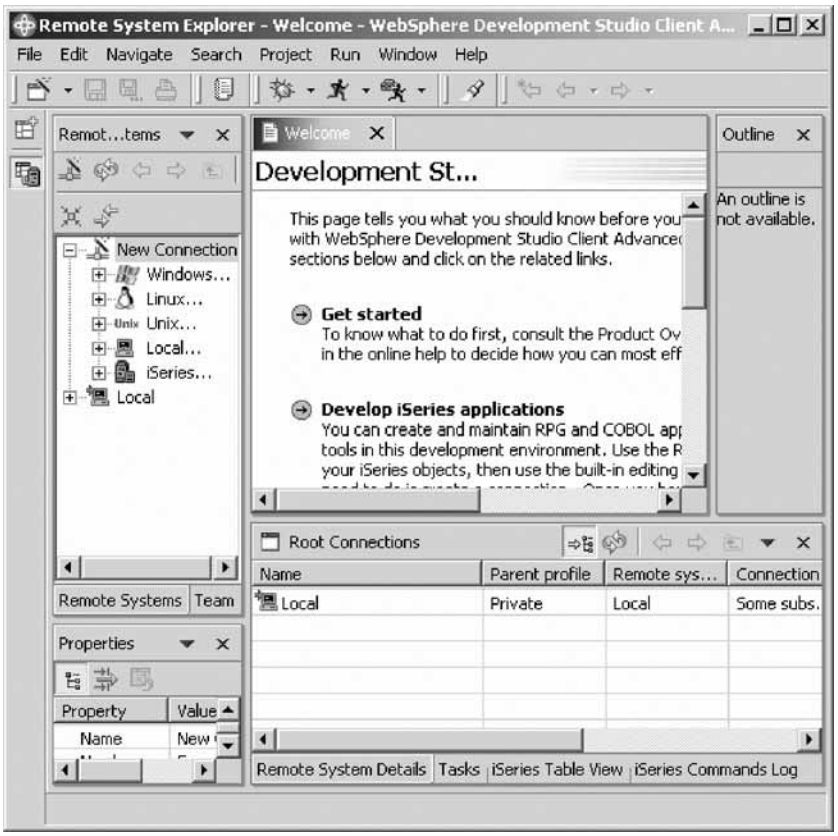
   A dialog for workspace selection will appear asking you for the workspace location, unless you used the product before and selected not to show this dialog again.



   The workspace contains all the information about your Development Studio projects. You can accept the default or store the work related to this tutorial, in a separate workspace.

2. To name the directory of the workspace as shown above, use the directory name `WSSLABxx`.
3. Click **OK**.

   After a few moments of loading, the workbench opens, and the initial window of the product appears on your desktop.



You have just started the product, set the default workspace and now you are ready to begin "Exercise 3.2: Opening the Remote System Explorer perspective."

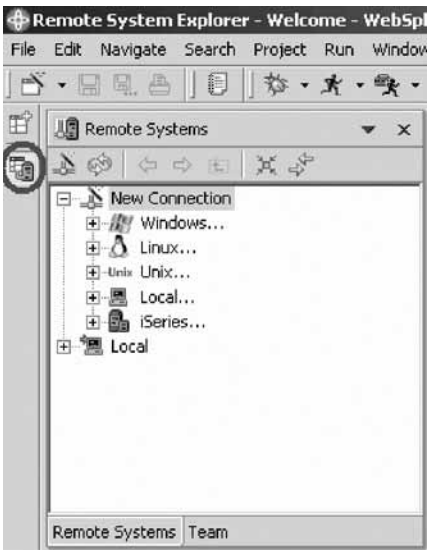
Exercise 3.2: Opening the Remote System Explorer perspective

Before you begin, you must complete "Exercise 3.1: Starting the product" on page 26.

Check if the window title bar says: Remote Systems Explorer. If it does you are all set. Skip the next steps and go directly to "Exercise 3.3: Creating a connection to the iSeries server" on page 29.

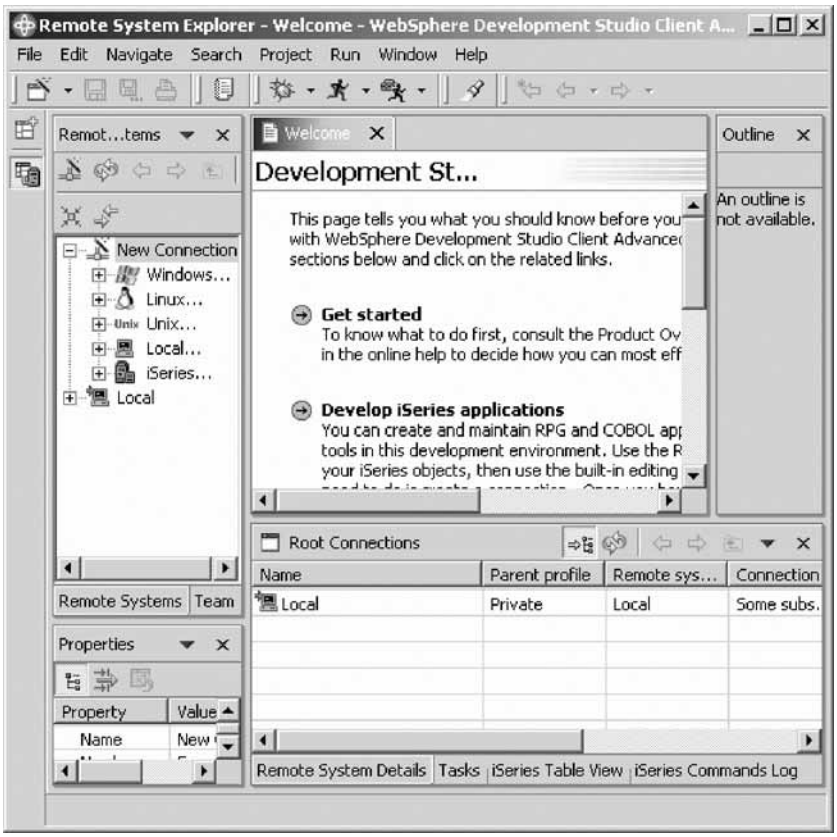
To open the Remote System Explorer perspective:

1. In the workbench, check whether you have the Remote System Explorer perspective already open. Look for the icon in the left taskbar of the workbench.
2. If you see it, click the RSE icon.



3. Otherwise, click **Window > Open Perspective > Remote System Explorer** on the workbench menu.

   The Remote System Explorer perspective opens.



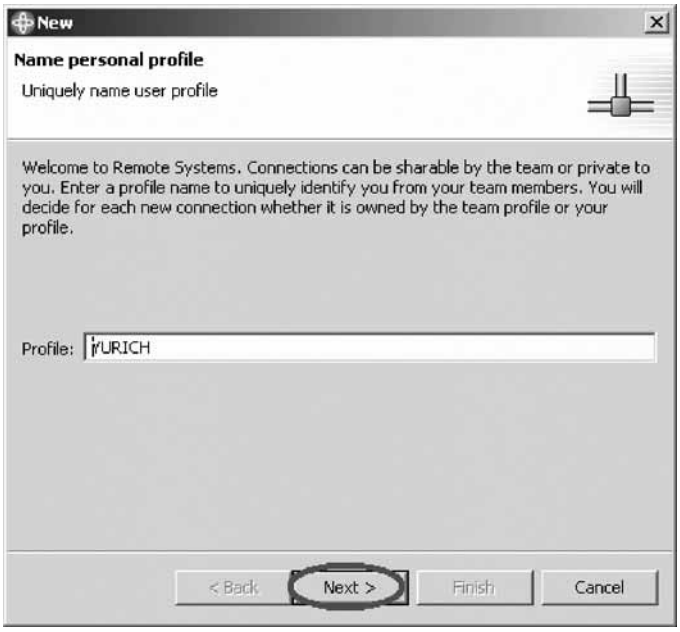
You have checked that the Remote System Explorer perspective is open and now you are ready to begin “Exercise 3.3: Creating a connection to the iSeries server” on page 29.

Exercise 3.3: Creating a connection to the iSeries server

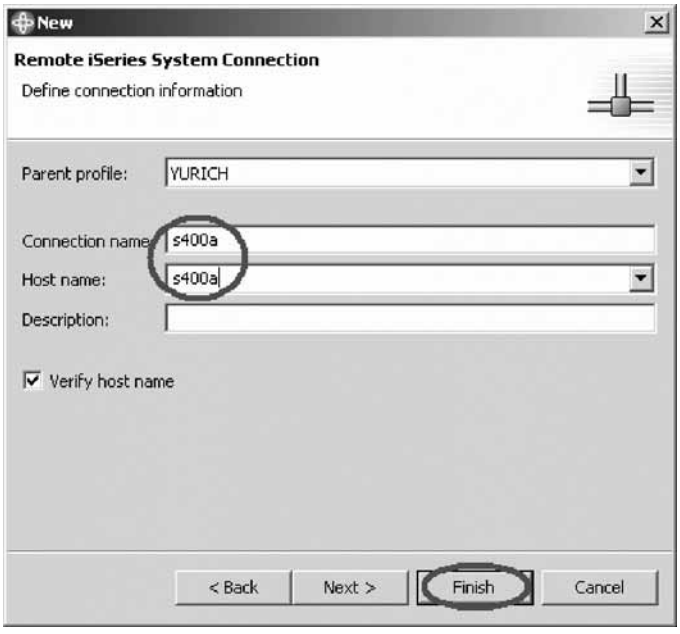
Before you begin, you must complete “Exercise 3.2: Opening the Remote System Explorer perspective” on page 27.

To create a connection:

1. Click the plus sign + beside **iSeries** under **New Connection** in the Remote Systems view.
2. Accept the default profile name.



3. Click **Next**.



In this exercise s400a is used as the name for the iSeries server, you will need to use the name or IP address of your particular iSeries server.

4. Type a host name, for example, `s400a` in the **Host name** field.

5. The **Connection name** field will be filled automatically with the same entry.
6. Click **Finish**.

   Your new connection appears in the Remote Systems view and you are ready to work with objects on this iSeries system.

You have set up a new connection to an iSeries server and now you are ready to begin "Exercise 3.4: Creating the IFS Directory for the Web project."

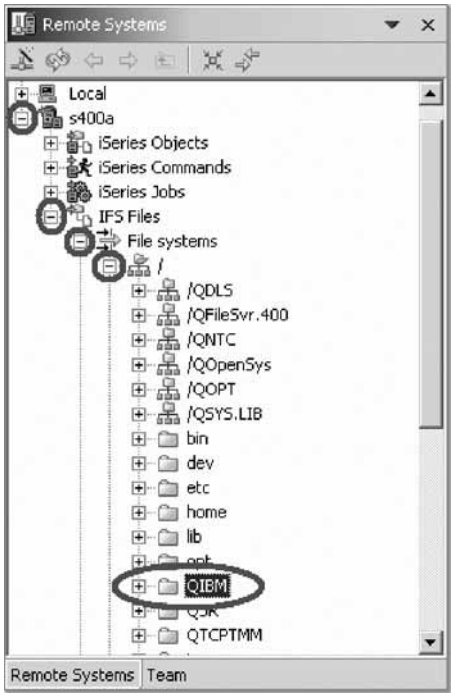
Exercise 3.4: Creating the IFS Directory for the Web project

Before you begin, you must complete "Exercise 3.3: Creating a connection to the iSeries server" on page 29.

IFS Files allow you to explore folders and files in the Integrated File System of the remote iSeries system. You will now create a directory in the IFS that will be used when you compile your iSeries source member.

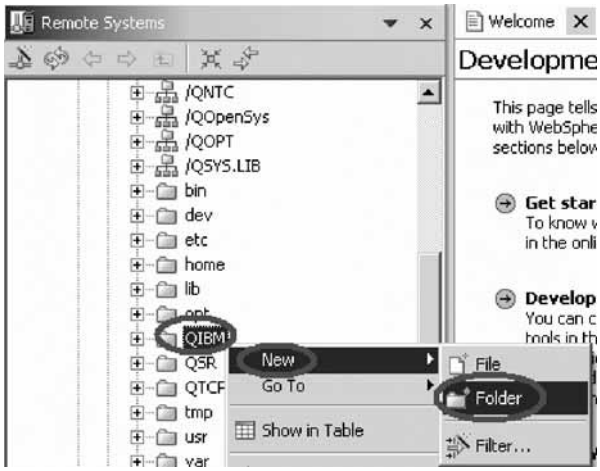
Locate the newly created iSeries connection.

1. Expand your iSeries connections.

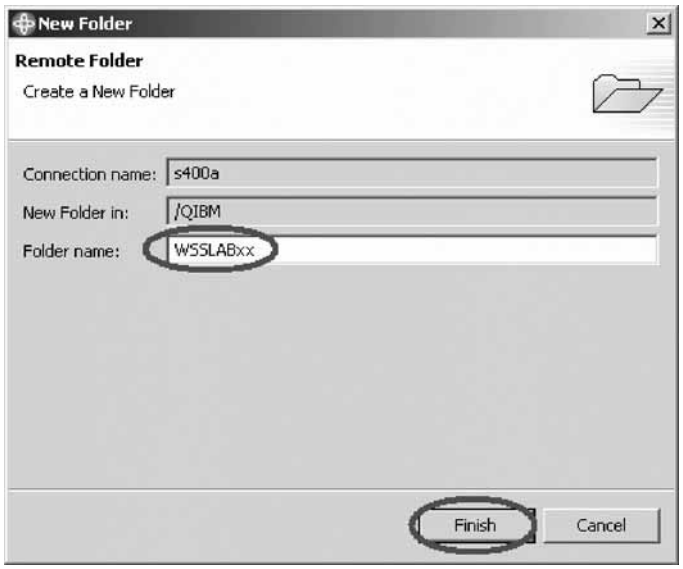


2. Expand the **IFS Files** by clicking the plus sign + beside it.
3. Expand the **File systems**.
4. If prompted, type your **User ID** and **Password**.

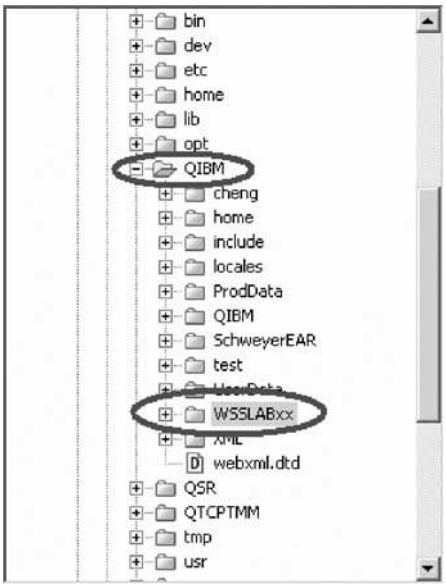
5. Select the **Save user ID** check box.
6. Select the **Save password** check box.
7. Expand the **/** Root node.
8. Select **QIBM**. Right-click **QIBM** and click **New > Folder**.



9. Enter `WSSLABxx` in the **Folder name** field.



10. Click **Finish**.



You should see the new WSSLABxx directory under QIBM.

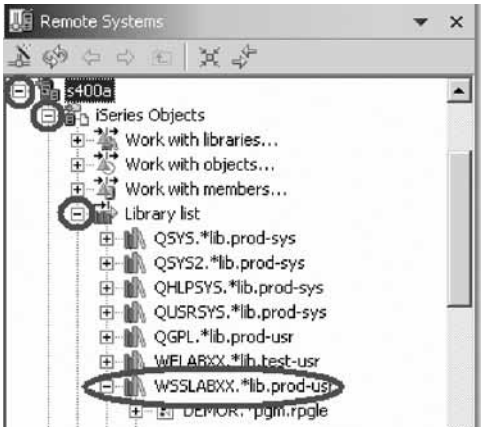
You have set up an IFS folder to contain the compiled iSeries source member and now you are ready to begin "Exercise 3.5: Finding an iSeries source member."

Exercise 3.5: Finding an iSeries source member

Before you begin, you must complete "Exercise 3.4: Creating the IFS Directory for the Web project" on page 30.

You will see a list of libraries under Library list in the Remote Systems view. Now you locate the source you want to edit.

To find an iSeries source member:



Note: If you don't see WSSLABxx in your library list, then you need to add the library. You can right-click the Library list and click Add Library List Entry from the pop-up menu to add WSSLABxx to your library list.

1. Expand your iSeries server. Use the same server that you have used to specify the reference fields in the data structure.

2. Expand **iSeries Objects** by clicking the plus sign + beside it.
3. Expand **Library List**.
4. Expand the library WSSLABxx.

   You will see all objects in this library.
5. Expand the source file QRPGLESRC.
6. Scroll down to the source members in the expanded list.

You have located an RPG source member you want to edit and now you are ready to begin "Exercise 3.6: Open a source member for edit."

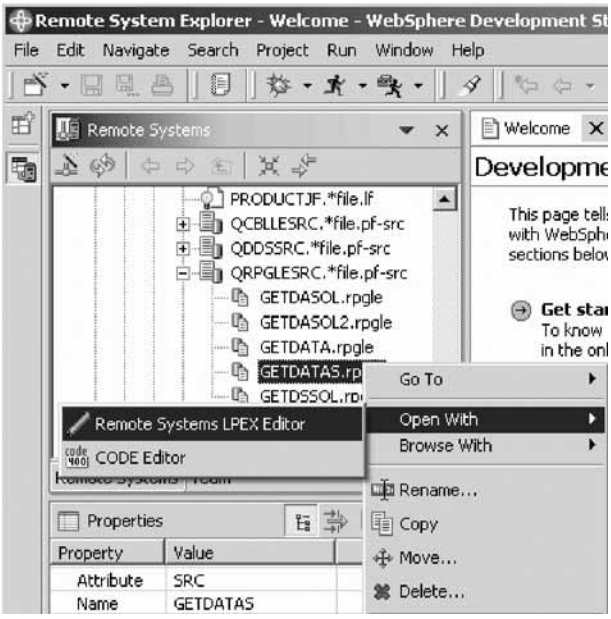
Exercise 3.6: Open a source member for edit

Before you begin, you must complete "Exercise 3.5: Finding an iSeries source member" on page 32.

One shell source member is provided for you to use.

To open a source member for edit:

1. Right-click member GETDATAS.
2. Click **Open with > Remote Systems LPEX Editor** on the pop-up menu.



The Editor window opens and you are ready to write your program.

Tip: You can also double-click on a member to open the Remote Systems LPEX editor.

You have opened an RPG source member for edit and now you are ready to begin "Exercise 3.7: Reviewing Remote System LPEX Editor features" on page 34.

Now before you edit the member, if you have not worked with the Remote Systems LPEX editor, go to "Exercise 3.7: Reviewing Remote System LPEX Editor features" on page 34.
If you already know the LPEX editor, you are ready to write your service program, go "Exercise 3.8: Editing a member and creating a service program module" on page 37.

Exercise 3.7: Reviewing Remote System LPEX Editor features

Before you begin, you must complete “Exercise 3.6: Open a source member for edit” on page 33.

Before you start, here are some useful tips on using the LPEX editor. Let’s look at some of the features of the LPEX editor, so you can later easily find your way around and use them:

- **ALT+L** marks a line
- **ALT+U** unmarks selection
- **ALT+D** deletes a selected text
- **ALT+C** copies selected text
- **ALT+M** moves selected text
- **Enter** inserts a new line

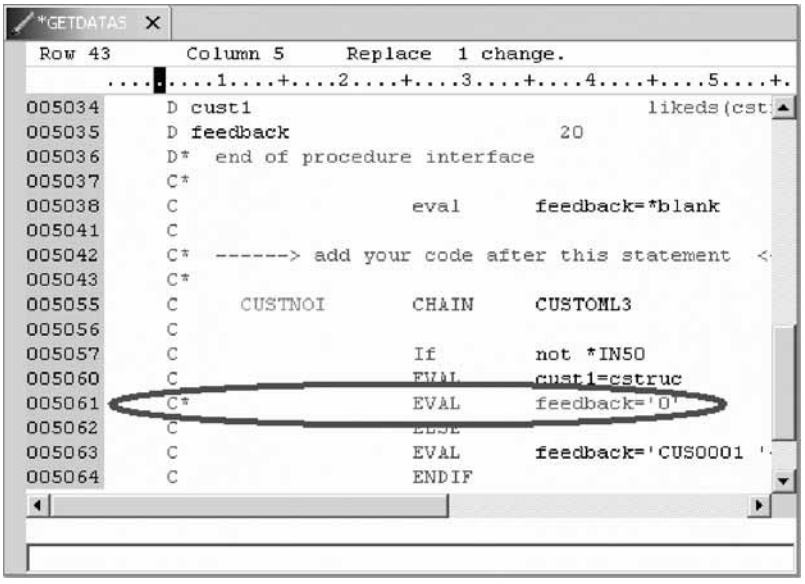
Highlighting specification fields

The LPEX editor highlights the Tokens (specification fields) of your RPG program source, providing a separate color for each, improving readability. When you make changes to a line, the token colors get updated only after you move your cursor off the line.

To see how token highlighting works:

1. Move the cursor to a Calculation statement.
2. Position the cursor to column 7 (right next to the C).
3. Type an asterisk (*).
4. Move the cursor off the line and watch what happens.

   The line where you typed the asterisk (*) becomes a comment line and its color changes accordingly.



5. Move the cursor back to column 7 and remove the asterisk (*).
6. Move the cursor off that line of code and the statement is tokenized.

   The token highlighting changes to reflect that this is a non-commented C specification.

Displaying types of lines

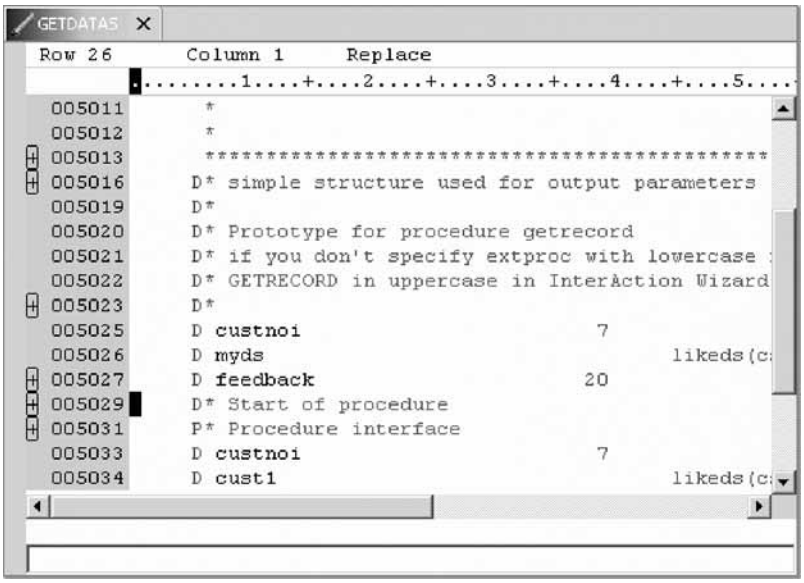
Using the LPEX editor, it is possible to have only particular types of source lines displayed at a time.

To display types of lines:

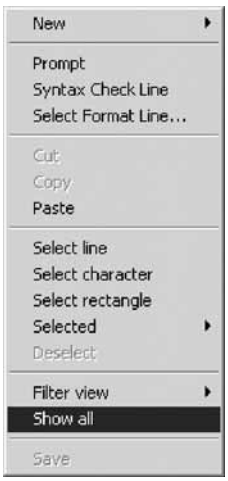
1. Right-click anywhere in the Editor window.
2. Click **Filter view** on the pop-up menu.

   The menu items list the types of line selections that can be made.
3. Click **Comments**.

   The Editor window now contains only those RPG statements that are comments.



4. Right-click anywhere in the editor window.
5. Click **Show all** on the pop-up menu.



All statements types are displayed. In addition, the choices in the pop-up menu can be used to include only control specifications, user subroutines, procedures and others. The Filter Selection option under the Selected option in the Edit menu allows the selection of only those lines containing a particular selected character string.

Checking the syntax of a file

Syntax checking is available for RPG code, and by default will be active. The syntax of RPG code is checked automatically when a change is made to a line, and the cursor is moved off the line.

When errors are found, they are displayed following the statement with the error.

Keeping track of columns in a specification line

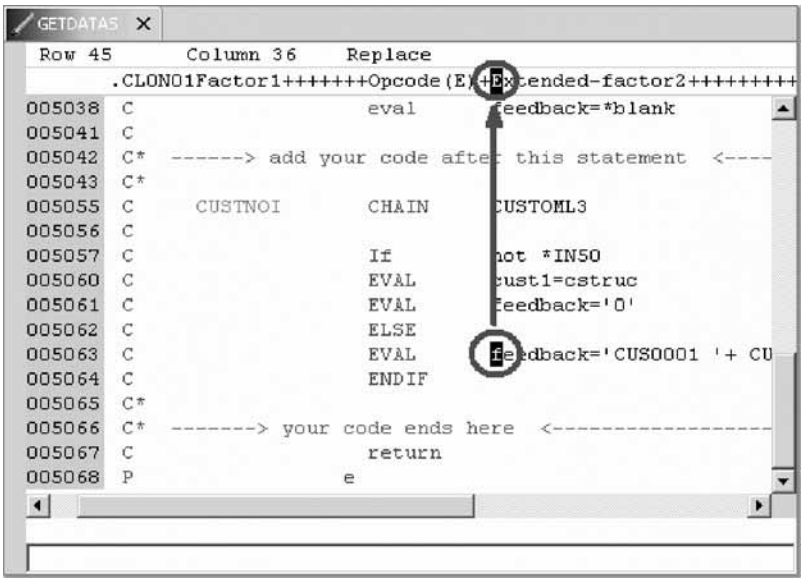
The format line is at the top of the Editor window, just above the first statement. A format line is used to help keep track of the columns in a particular specification line. The content of the format line can vary to reflect the particular type of specification being keyed such as F specs, C specs, D specs and so on.

To display a format line:

1. Click a line or use the arrow keys and click the left mouse button to make a line active.

   The format line gets updated as a line gets focus.

   You can move the cursor right or left with the arrow keys to go from character to character, or with the **Tab** key to go from field to field. An indicator on the format line moves with the cursor to show in which column the cursor is positioned.



   You can select a format line for any specification you want.

2. Click **Source** from the Editor menu.
3. Click **Select Format Line** on the pop-up menu.

   The Format Line Selection window opens.

   Now you can start and create the service program.

4. Click **OK**.

You have highlighted tokens of your RPG source, shown only comment lines, shown all lines, displayed a format line and now you are ready to begin "Exercise 3.8: Editing a member and creating a service program module."

Exercise 3.8: Editing a member and creating a service program module

Before you begin, you must complete "Exercise 3.6: Open a source member for edit" on page 33 and if you were unfamiliar with the editing features of the Remote Systems LPEX Editor then you must complete "Exercise 3.7: Reviewing Remote System LPEX Editor features" on page 34.

To make it easier for you most of the RPG source is already prepared for you. You have already opened the correct source member GETDATAS in the previous exercise using the Remote Systems view. Read through the source. It contains an F spec to access file CUSTOML3 which contains the customer data keyed by customer number. The D specs define the data structure CSTRUC that gets passed back to your Web page and the CUSTNOI variable that gets passed from the Web page to this program. Also variable FEEDBACK is defined as 20 characters, to return message id and customer number in case the customer record was not found.

The prototype for procedure getrecord with the parameters is defined as the:

- CUSTNOI field (this is the input parameter)
- CSTRUC structure (this is the data structure for customer data)
- FEEDBACK field (returns error or success)

Note: The external name is case sensitive. It has to match exactly what you specified in the Web Interaction wizard.

The procedure interface defines the parameters in the same way as the prototype. Inside the procedure after the procedure interface you need to add the RPG code.

Adding RPG code

To add the RPG code:

1.
```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
C     CUSTNOI       CHAIN(E)  CUSTOML3
C                   IF        %found(CUSTOML3)
C                   EVAL      FEEDBACK='0'
C                   EVAL      cust1=cstruc
C                   ELSE
C                   EVAL      FEEDBACK='CUS0001 ' + CUSTNOI
C                   ENDIF
```
2. You do a CHAIN to the database and move the database fields to the local procedure structure. If the database access fails, move the MSGID and Customer number to the FEEDBACK variable.

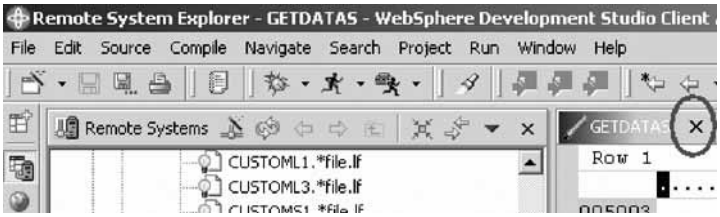
Your coding is complete, so let's create the service program module.

Creating the service program module

To create the service program module:

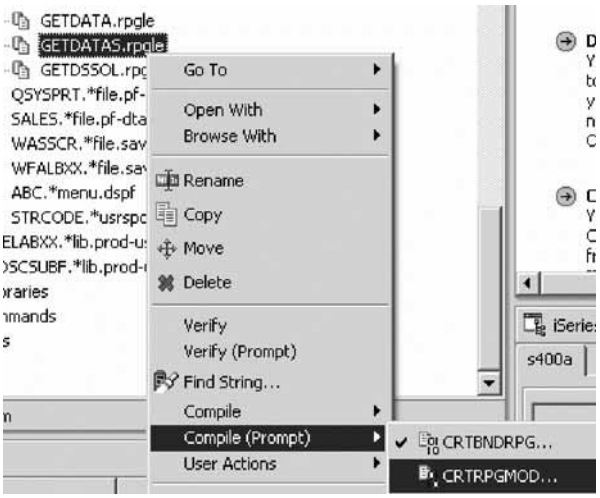
1. Click the Save icon on the workbench toolbar to save the member.

2. Click **X** in the Editor window title bar to close the member.

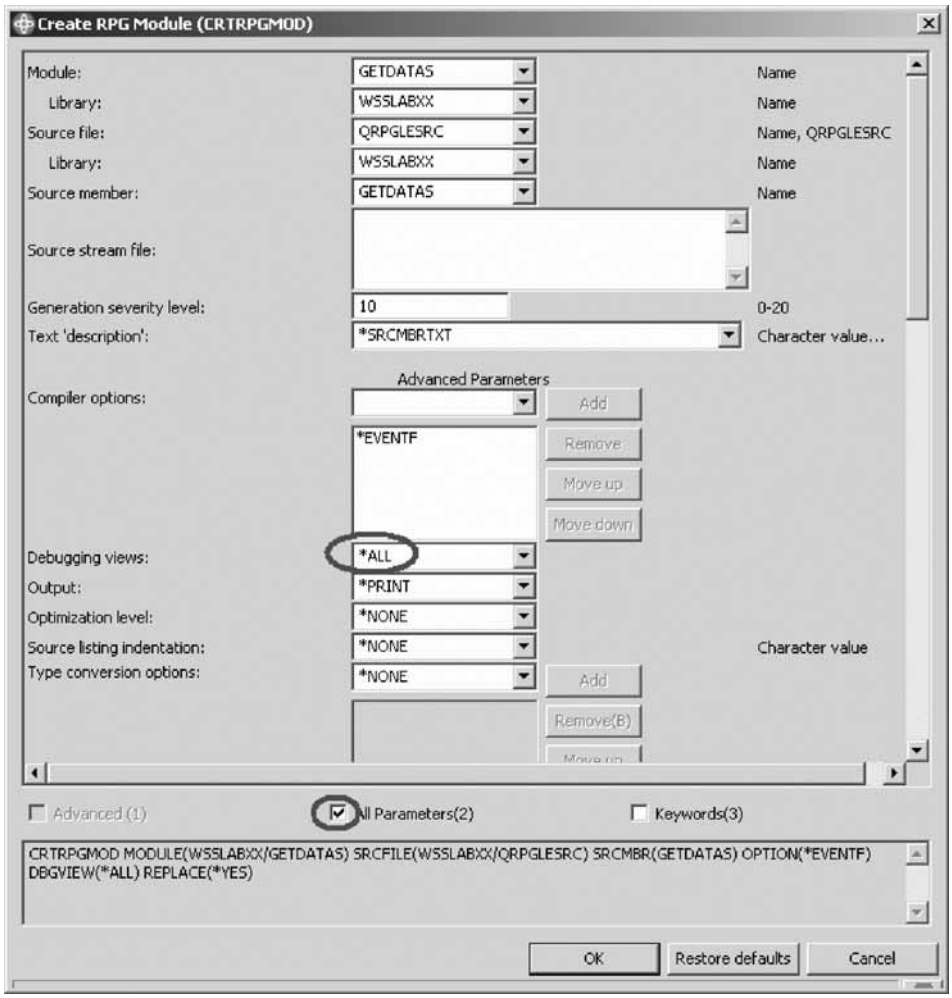


Be careful, don't click the X on the workbench window title bar.

3. In the Remote Systems view, right-click GETDATAS member in QRPGLESRC and select **Compile (Prompt) > CRTRPGMOD** on the pop-up menu.

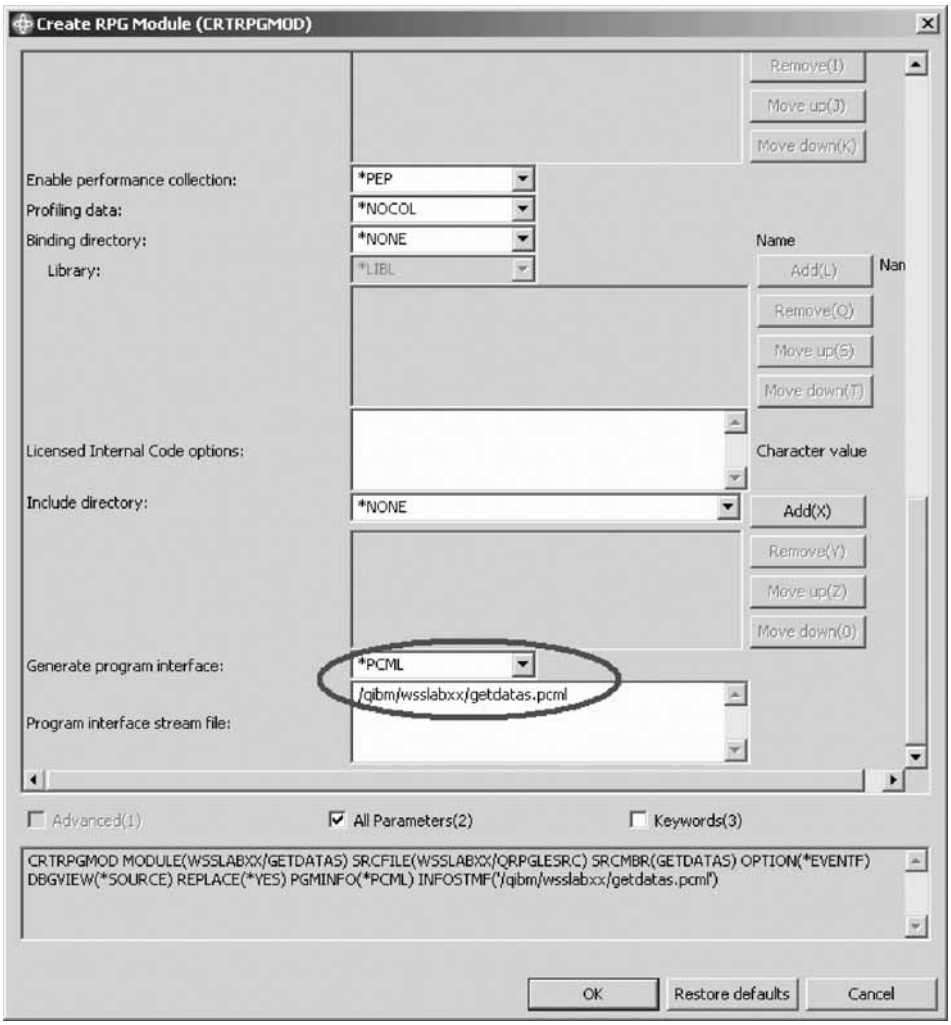


The Create RPG Module (CRTRPGMOD) dialog opens.



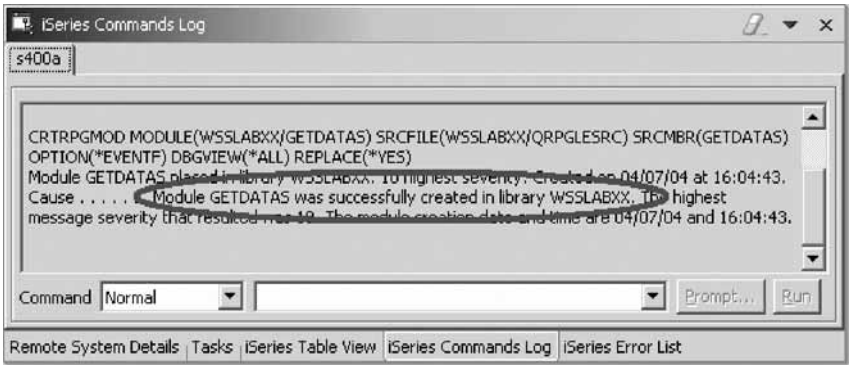
4. In the **Debugging views** list, select ***ALL**

   You need to specify two more parameters, these parameters are used to tell the compiler to generate the program interface description in a language called Program Call Markup Language (PCML). Later on you can use this description in the Web Interaction wizard.
5. Select the **All Parameters** check box at the bottom of the dialog.
6. Scroll down in the dialog until you see the **Generate program interface** field.
7. Select the ***PCML** value from the list.



8. Enter `/qibm/wsslabxx/getdatas.pcml` for the **Program interface stream file** field.
9. Click **OK** to submit the command.

   After the compile the iSeries Error List view is shown, listing all compile errors. You may only see information and warning messages that means your compile was completed and the program module was created.



10. Click the **iSeries Commands Log** tab to check if the compile was successful. This log shows the Remote System Explorer job messages. If you get a message that there have been errors found in your program, go through the edit compile fix cycle.

Fixing errors

In the iSeries Error List view, check the errors:

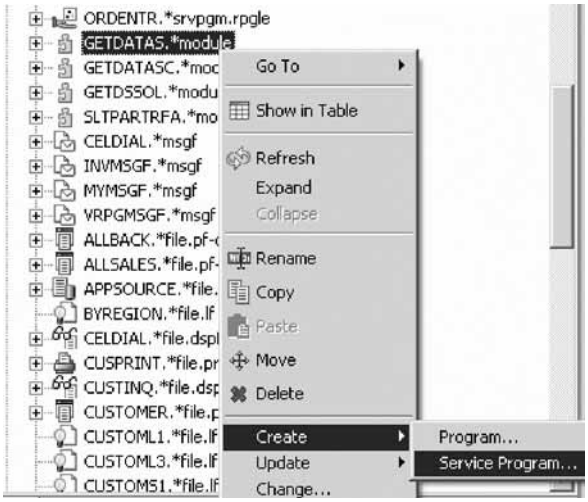
1. Double-click on the error.

   This positions the cursor in the Editor window on the statement that is wrong.
2. Fix all errors.
3. Save the source member.
4. Go back to the Remote Systems view.
5. Right-click the member to run the CRTRPGMOD command.
6. Continue this cycle until you get a clean compile.

Creating a service program

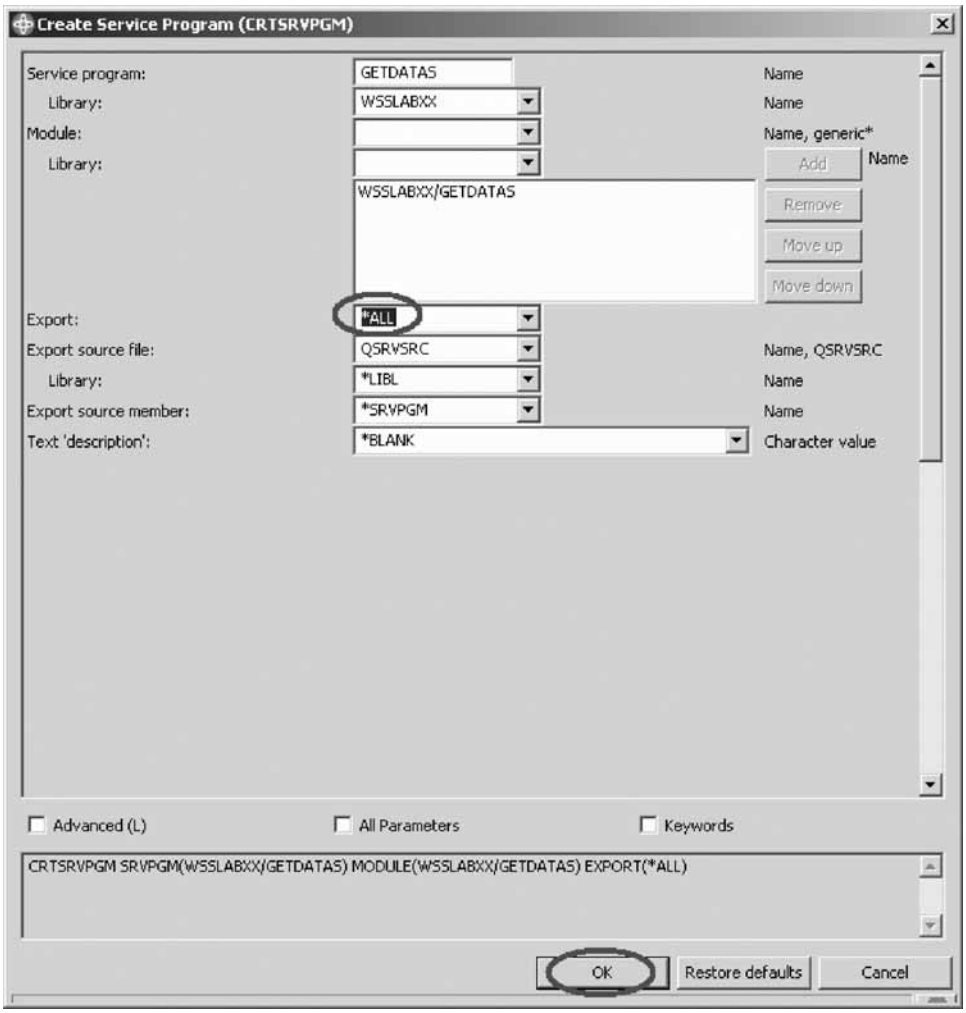
To create a service program:

1. In the Remote Systems view find the GETDATAS module in library WSSLABxx.

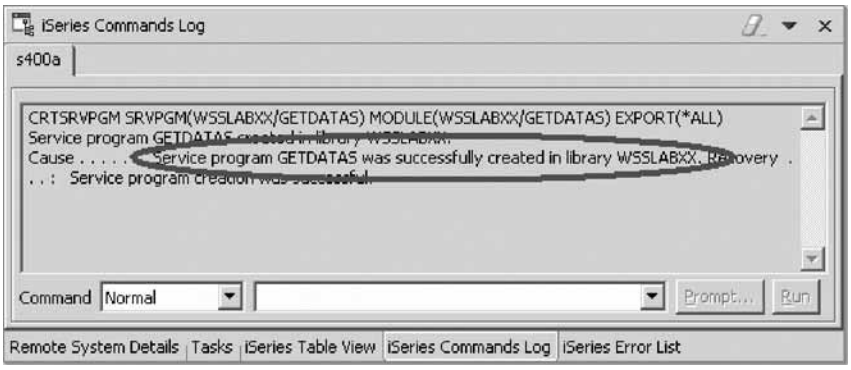


   Note: If you don't see the module, right-click library WSSLABxx and click **Refresh** on the pop-up menu.
2. Right-click GETDATAS and click **Create > Service Program** on the pop-up menu.

   The Create Service Program (CRTSRVPGM) dialog opens.



3. In the Export list, select ***ALL**.
4. Click **OK**.
   Check the command log to see that the Service program was created.
5. Click the **iSeries Commands Log** tab at the bottom of the workbench.
   You should see a message that the Service program was created.



You have edited an RPG source member and created a service program object.

Module recap

You have completed Chapter 3, “Module 3. Creating the RPG service program for your Web application (optional),” on page 25. You have learned how to:

- Start the product
- Set the default workspace
- Check the Remote System Explorer perspective is open
- Set up a new connection to an iSeries server
- Set up an IFS folder to contain the compiled iSeries source member
- Locate an RPG source file you want to edit
- Open an RPG source member with the Remote Systems LPEX Editor
- Highlight the tokens of your RPG source
- Show only comment lines in the RPG source
- Show all lines in the RPG source
- Display a format line
- Edit an RPG source member with the Remote Systems LPEX Editor
- Compile a service program object
- Create a Program Call Markup Language (PCML) program interface description

Now that you have created a RPG service program, you can continue to Chapter 4, “Module 4. Creating a Web project,” on page 45.

Chapter 4. Module 4. Creating a Web project

This module teaches you how to create a Web project and then supply the information which the iSeries server will use for serving the business information for this Web application. You will also learn about perspectives, J2EE settings, Web projects, and the iSeries Web Tools Run-Time Configuration wizard.

In this module, you will:

- Access tools and views for iSeries Web application development
- Know the difference between a static Web project and a dynamic Web project
- Set up a dynamic Web project
- Define where the RPG program or service program resides on an iSeries server

Exercises

The exercises in this module must be completed in order.

- "Exercise 4.1: Opening the Web perspective"
- "Exercise 4.2: Creating a dynamic Web project" on page 47
- "Exercise 4.3: Setting up the iSeries server information" on page 51

Time required

This module will take approximately **15 minutes** to complete.

Exercise 4.1: Opening the Web perspective

In this exercise you will use the Web perspective. This allows you to access tools and views specifically related to Web application development. However, you are still in the Remote System Explorer (RSE) perspective which gives you the tools to do your iSeries programming tasks as you have discovered in the previous module.

Perspectives

Before we go further, let's review perspectives. Perspectives are a collection of views and tools assembled to help different kinds of users to do their job when they use the workbench. These workbench users will have different jobs (roles), for example, one user needs to work with iSeries objects related to RPG/COBOL, another user is dealing with Java programs, and another user with Web page development. Each of these different user types will need different views of the files/objects they are working with and they need different tools. The perspectives present the user with a selection of specific views/tools geared towards the different roles the users have. The workbench has many predefined perspectives, like the:

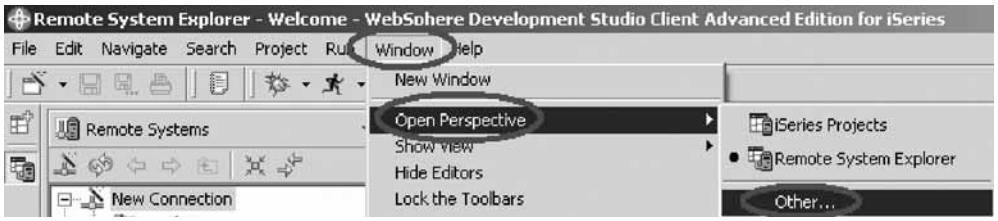
- Web perspective for Web developers.
- Remote System Explorer perspective for iSeries programmers.

You can also create your own perspective by using the **Save Perspective as** option under the **Window** menu in the workbench, after you have modified a perspective.

© Copyright IBM Corp. 2001, 2005

Now that you know what a perspective is we can go ahead with our exercise. You are a Web developer in this tutorial and that is the reason you will select the Web perspective. The Web environment provides its own perspective since it needs to give its users access to unique views and tools targeted towards Web tasks. You create a Web project in the Web perspective.

To open the Web perspective:

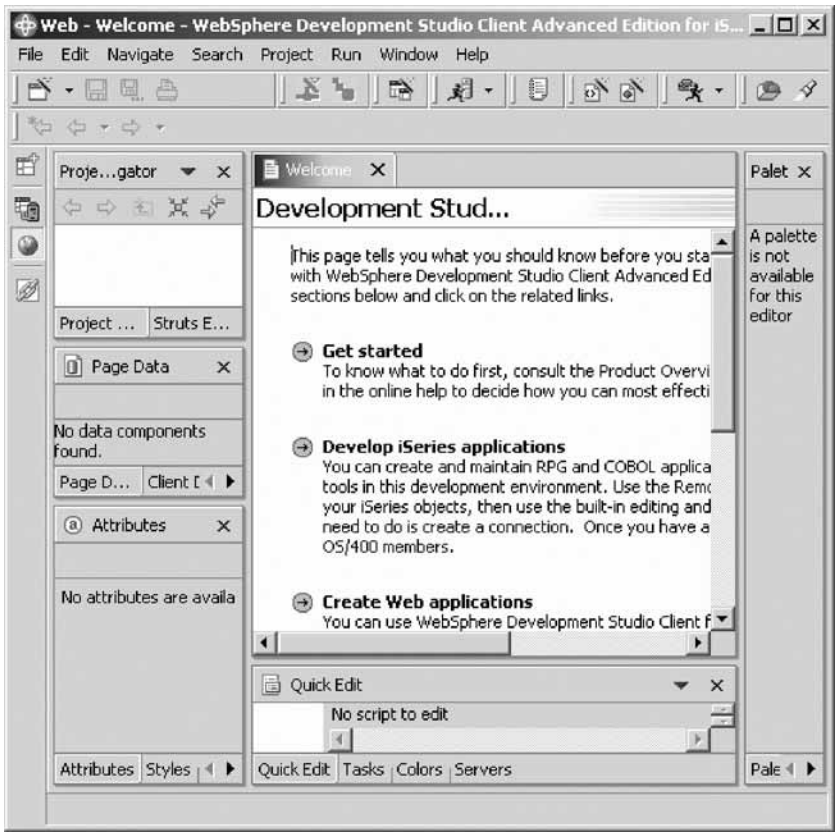


1. Click **Window > Open Perspective** on the workbench menu.
2. Then click **Other** on the pop-up menu.

   The Select Perspective dialog opens.

3. Select **Web** from the list.
4. Click **OK**.

   The workbench will now show the Web perspective with the Project Navigator view open on the left hand side.



You have accessed tools and views for iSeries Web application development and now you are ready to begin “Exercise 4.2: Creating a dynamic Web project.”

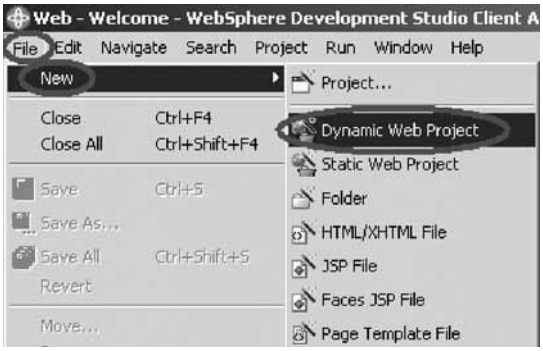
Exercise 4.2: Creating a dynamic Web project

Before you begin, you must complete “Exercise 4.1: Opening the Web perspective” on page 45.

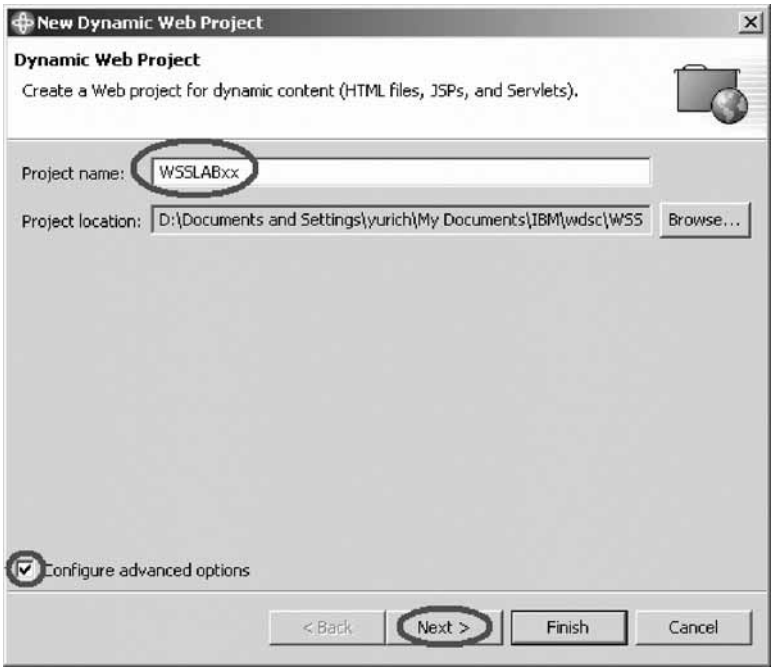
Web projects hold all of the Web resources that are created and used when developing your Web application. The first step to creating or importing a Web application is to create either a static or a dynamic Web project. Static Web projects are meant to contain only simple Web site resources, such as HTML files. Dynamic Web projects are used to structure Web applications that will use more complicated, dynamic Web technologies, such as JavaServer Page files, and possibly data access resources.

To create a dynamic Web project:

1. Click **File > New** on the workbench menu.
2. Then click **Dynamic Web Project** on the pop-up menu.



This starts the Dynamic Web Project wizard. The initial page of the wizard opens.

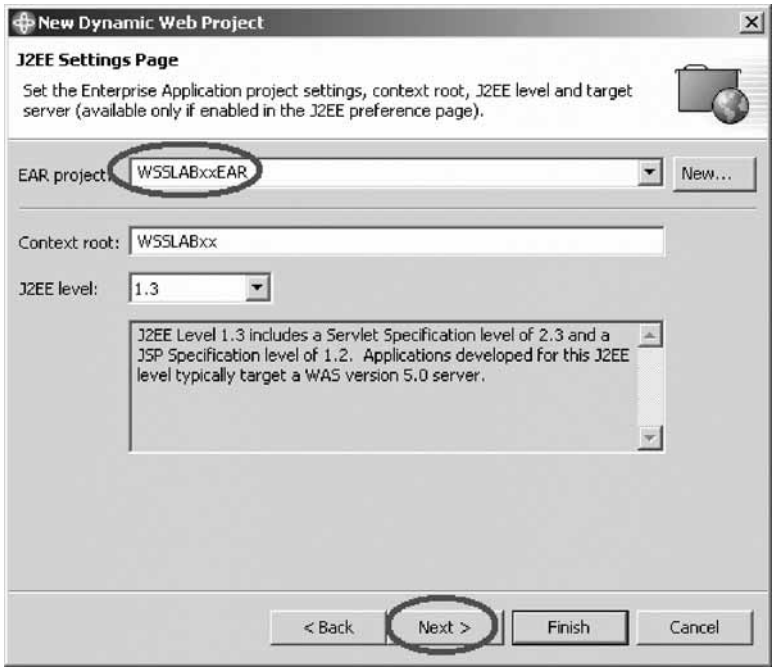


3. Accept the default value in the **Project location** field.

   This is where your project is stored in your file system.
4. In the **Project name** field, type WSSLABxx.
5. Select the **Configure advanced options** check box to specify or customize the project options, such as adding Struts support to your Web project.
6. Click **Next**.

   The J2EE Settings page opens.

   This page allows you to specify the J2EE level for the Web project. If you plan to use the WebSphere Application Server Version 5 only, then you select J2EE level 1.3. If you are using previous versions of WebSphere Application Server in your environment and are planning to deploy the Web application to these versions then you use J2EE level 1.2 since these earlier versions of WebSphere Application Server don't support the J2EE level 1.3. In this exercise you use the Web application in the WebSphere Test Environment, which is Version 5, so you use J2EE level 1.3 which is the default.



7. In the **EAR project** field, type `WSSLABxxEAR`.

   A new or existing Enterprise Application project must be associated with your new Web project for purposes of deployment.

   The Enterprise Application project is new, so you typed the name of the new project. When your Web project is created at the end of the wizard, the new Enterprise Application project (EAR file) is also created. The default is an Enterprise Application project named DefaultEAR located in the same directory as your new Web project.

8. Leave the default value for the **Context root** field.

   The default value for the context root is the name you provided for your Web project. The context root is the Web application root, the top-level directory of your application when it is deployed to a Web server. The context root can also be used by the links builder to ensure that your links remain ready to publish as you move and rename files inside your project.

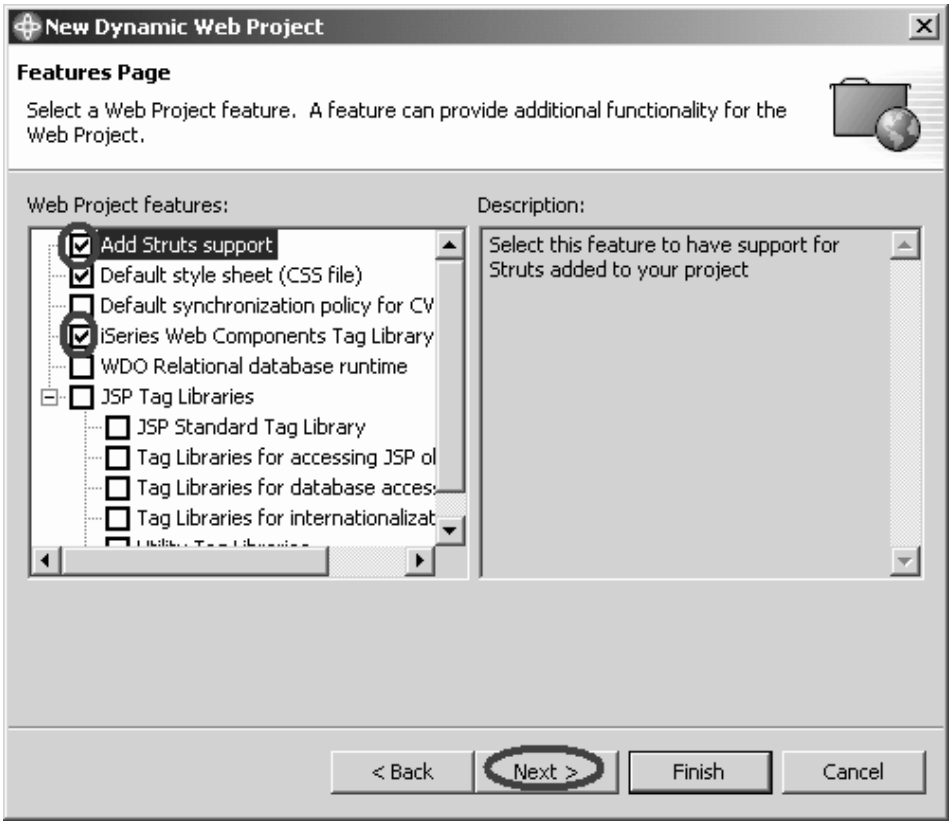
9. Leave the default value in the **J2EE level** list.

   By default, the Web project's J2EE level is set to the Workbench's J2EE level.

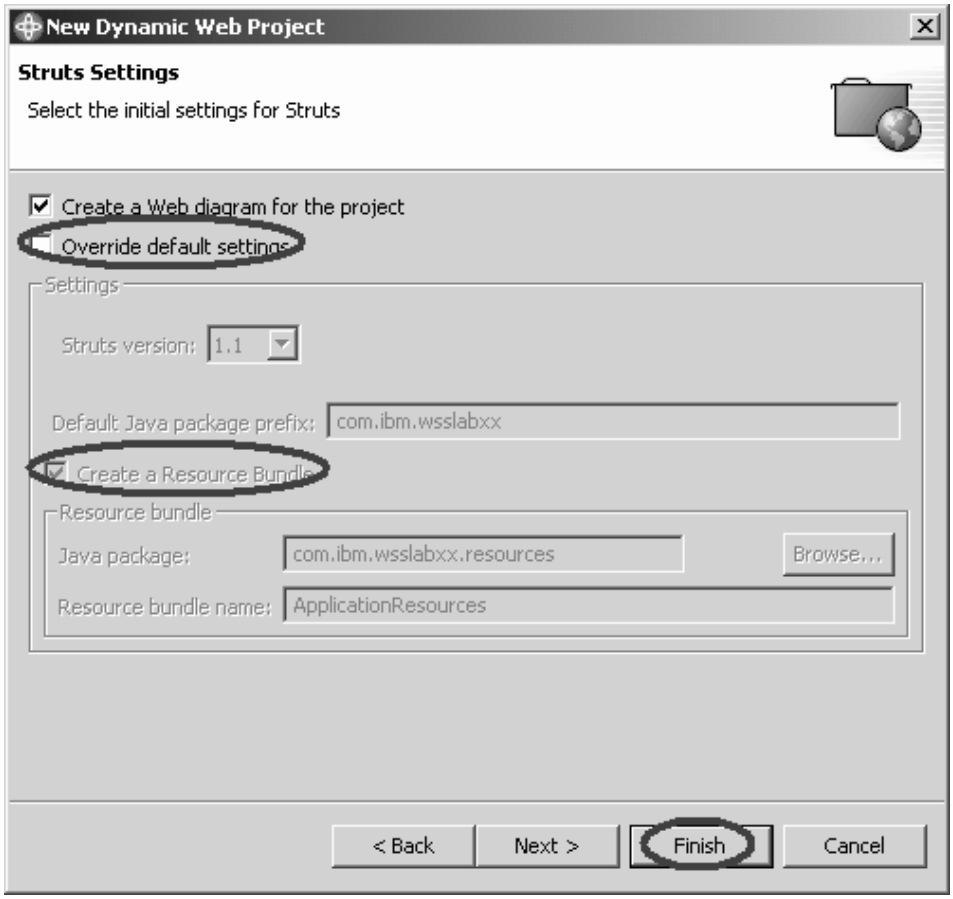
   Any new servlets and JSP files that you create should adhere to the latest specification level available; previous specification levels are offered to accommodate any legacy dynamic elements that you expect to import into the project.

10. Click **Next**.

    The Features Page opens.



11. Leave the **Default style sheet (CSS)** check box selected to create a default CSS file (called Master.css) for any HTML and JSP files included in the project.
12. Select the **Add Struts support** check box as you want to create a project that uses Struts technology.
13. Select the **iSeries Web Components Tag Library** check box as you want to add iSeries fields and controls to the Web pages of your Web application.
14. Click **Next** to go to the Struts Settings page and ensure that **Create a Resource Bundle for the Struts Project** is selected. If not, select the **Override default settings** check box and then select **Create a Resource Bundle for the Struts Project**.



15. Click **Finish** to create the Web project.

    Now you are back in the workbench and your new project and project files appear in the Project Navigator.

You know the difference between a static Web project and a dynamic Web project and have set up a dynamic Web project and now you are ready to begin "Exercise 4.3: Setting up the iSeries server information."

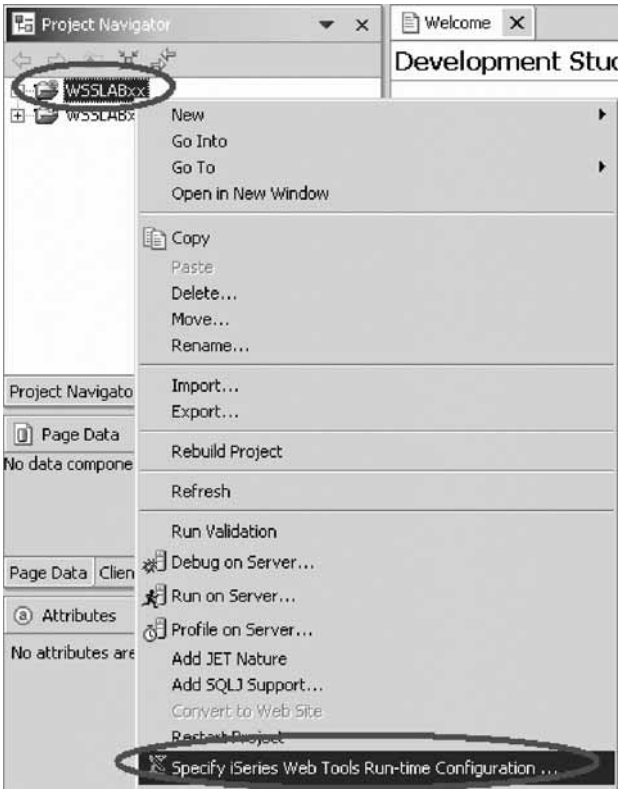
Exercise 4.3: Setting up the iSeries server information

Before you begin, you must complete "Exercise 4.2: Creating a dynamic Web project" on page 47.

Before you can start creating your application, you will need to define where the RPG program you will connect to later on in this tutorial is located. You need to specify the iSeries server name and the user-ID and password to be used when starting the job on the iSeries to run your program.

To set up the iSeries server:

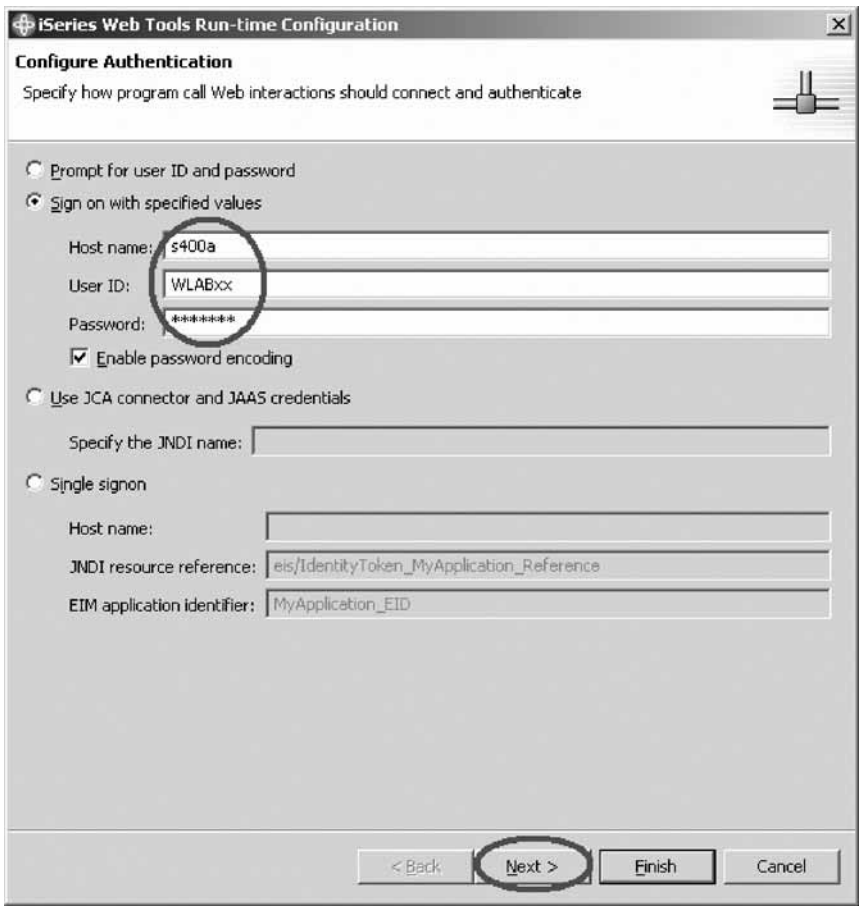
1. In the Web perspective, right-click the WSSLABxx project in the Project Navigator.



2. Click **Specify iSeries Web Tools Run-Time Configuration** on the pop-up menu.

   Tip: You could also use the button from the workbench toolbar to open the Run-Time Configuration wizard.

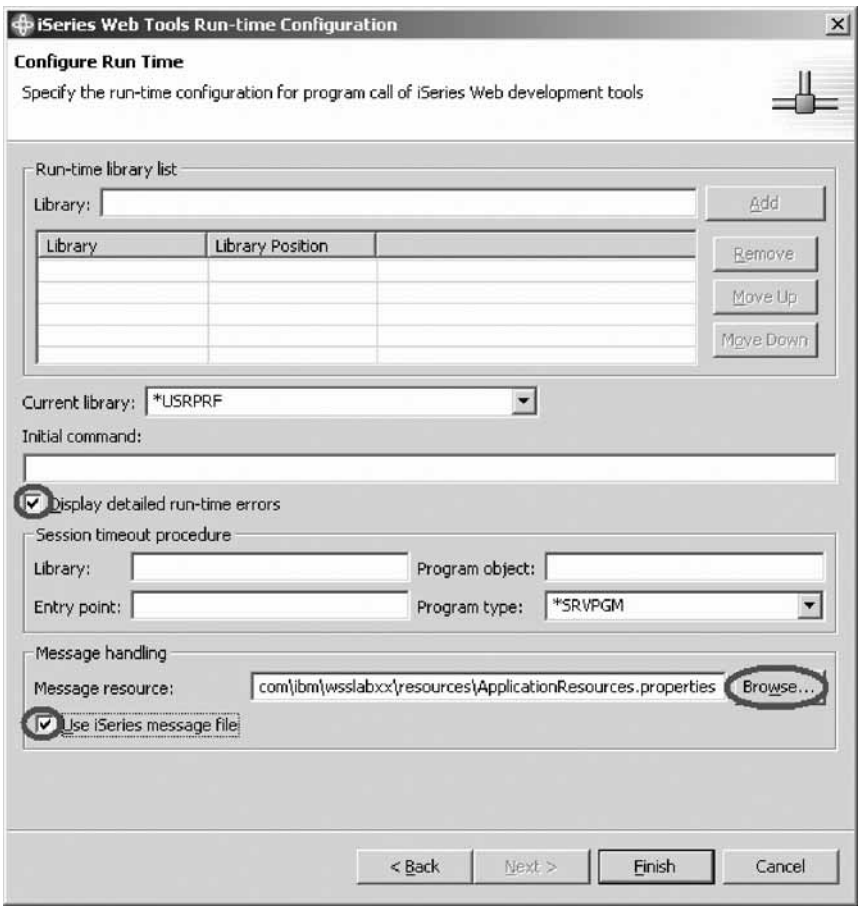
   The iSeries Web Tools Run-Time Configuration wizard opens.



You use this wizard to specify how Web interactions that perform program calls to an iSeries host should be connected and authenticated.

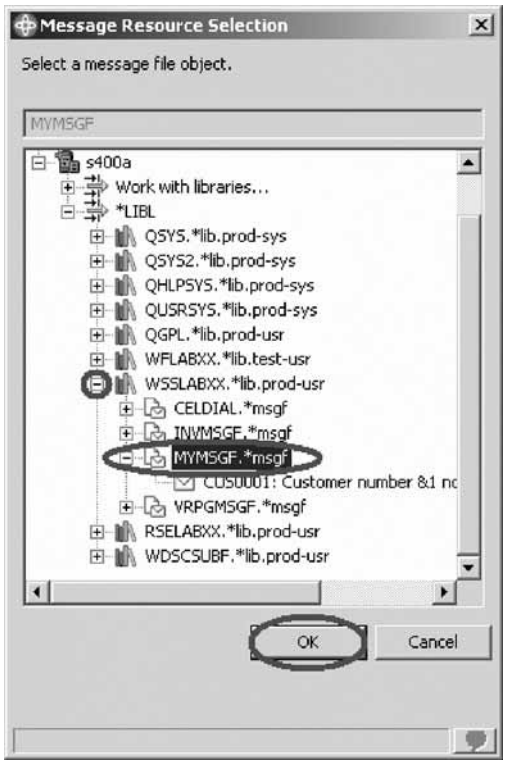
3. In the **Host name** field, type the name of the iSeries host where your program is located, for example `s400a`.
4. Enter your user ID and password for the iSeries host in the appropriate fields.

   You need to ensure that your User ID has been set up in a way that it contains the correct library list. If not, you will have to use the table on the next page to add a library to the library list when the job gets started to run your program.
5. **Enable password-encoding** is selected by default. This option ensures that the password appears in encoded format in the web.xml file. If you clear the check box for this option, the password appears as plain text in the web.xml file.
6. Click **Next**.
7. (Optional) If you need to add WSSLABxx to your library list, then type `WSSLABxx` in the **Library** field and click **Add**.
8. Select the **Display detailed run-time errors** check box to see the details for any errors that occur during the runtime of your Web application. This is useful for tracing and debugging development errors that may occur.



9. Select the **Use iSeries message file** check box to retrieve error messages from the iSeries host.
10. Click **Browse** to specify the iSeries message file.
11. Click the plug sign + beside s400a.
12. Click the plus sign + beside *LIBL.

13. Drill down to WSSLABXX > MYMSGF.*msgf.



14. Click **OK**.
15. Click **Finish** to define authentication and run-time values for your host program or procedure call to be made by your Web interaction in the Web project.

You have defined where the RPG program or service program resides on the iSeries server.

Module recap

You have completed Chapter 4, “Module 4. Creating a Web project,” on page 45. You have learned how to:

- Access tools and views for iSeries Web application development
- Know the difference between a static Web project and a dynamic Web project
- Set up a dynamic Web project
- Define where the RPG program or service program resides on an iSeries server

Now that you have created a Web project, you can continue to Chapter 5, “Module 5. Creating a Web application,” on page 57.

Chapter 5. Module 5. Creating a Web application

This module teaches you how to create the interaction to use an input and output page, and to create the servlet to invoke an RPG program to get data from the iSeries database. You will also learn about interactions, Web input and output pages and Program Call Markup Language (PCML). Finally you will learn how to see the flow structure of your Web application.

In this module, you will:

- Start the Web Interaction wizard
- Create an input page that invokes the interaction
- Create an output page that displays the results of the interaction
- Get the program interface definitions
- Specify the input and output parameters
- Specify the input parameters to show on the input page
- Specify what data to show on the output page
- Specify what will happen when an incorrect customer number is entered
- Change the background color of the output page
- Make sure the customer number field is read only
- Know what Struts is all about
- View the flow structure of a Struts-based Web application

Exercises

The exercises in this module must be completed in order.

- "Exercise 5.1: Invoking the Web Interaction wizard" on page 58
- "Exercise 5.2: Specifying the input and output page" on page 59
- "Exercise 5.3: Defining the iSeries GETDATA program invocation and parameters" on page 61
- "Exercise 5.4: Defining the iSeries GETDATAS service program invocation and parameters" on page 66
- "Exercise 5.5: Defining the input page content" on page 72
- "Exercise 5.6: Defining the output page content" on page 74
- "Exercise 5.7: Specifying error handling" on page 76
- "Exercise 5.8: Enhancing the output page" on page 78
- "Exercise 5.9: Ensuring the customer number field cannot be modified" on page 82
- "Exercise 5.10: Visualizing the flow structure of your Web application" on page 84

Time required

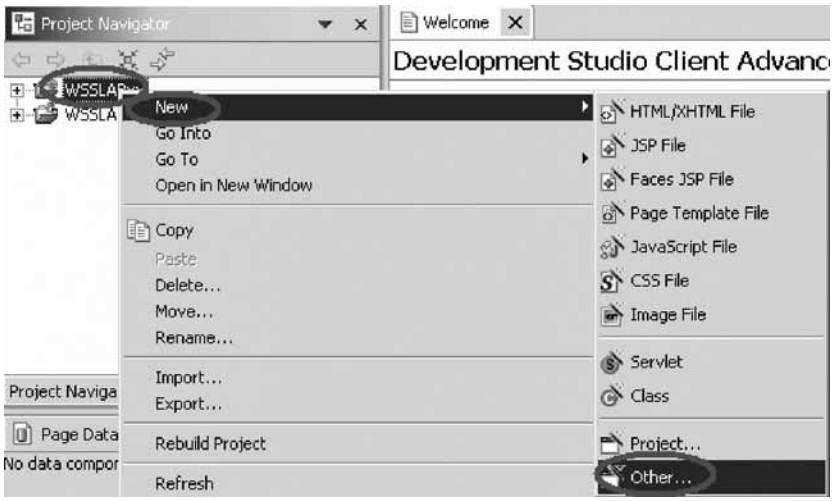
This module will take approximately **30 minutes** to complete.

© Copyright IBM Corp. 2001, 2005

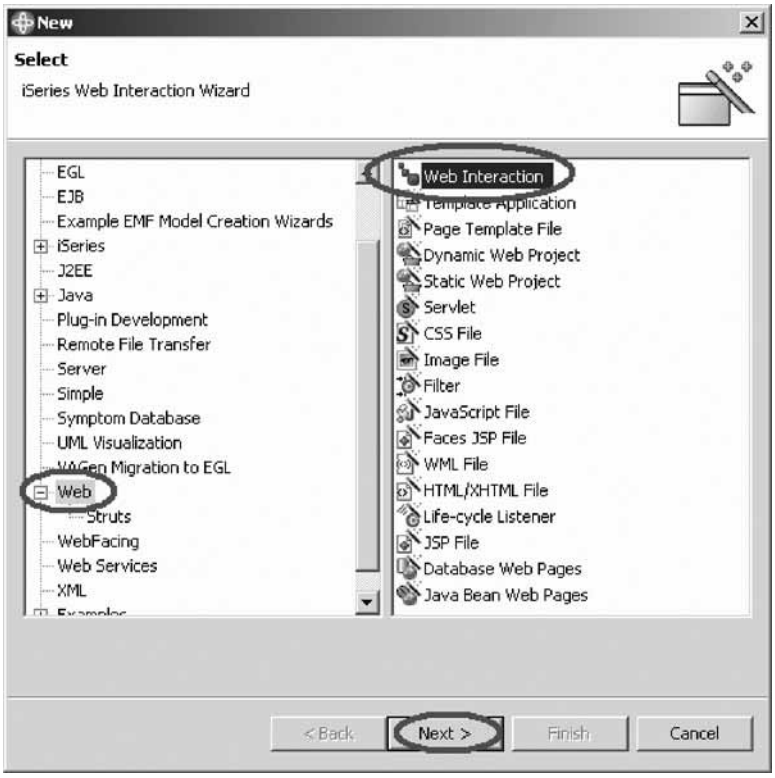
Exercise 5.1: Invoking the Web Interaction wizard

During this exercise you will use the Web Interaction wizard to create the interface description for parameters to be passed to an iSeries program. You will also generate a Web page based on the parameters definition used by the program. Some of the parameters will be defined based on existing iSeries database field definitions.

To invoke the Web Interaction wizard:



1. Right-click WSSLABxx in the Project Navigator.
2. Click **New > Other** on the pop-up menu.
3. Select **Web** from the left pane.



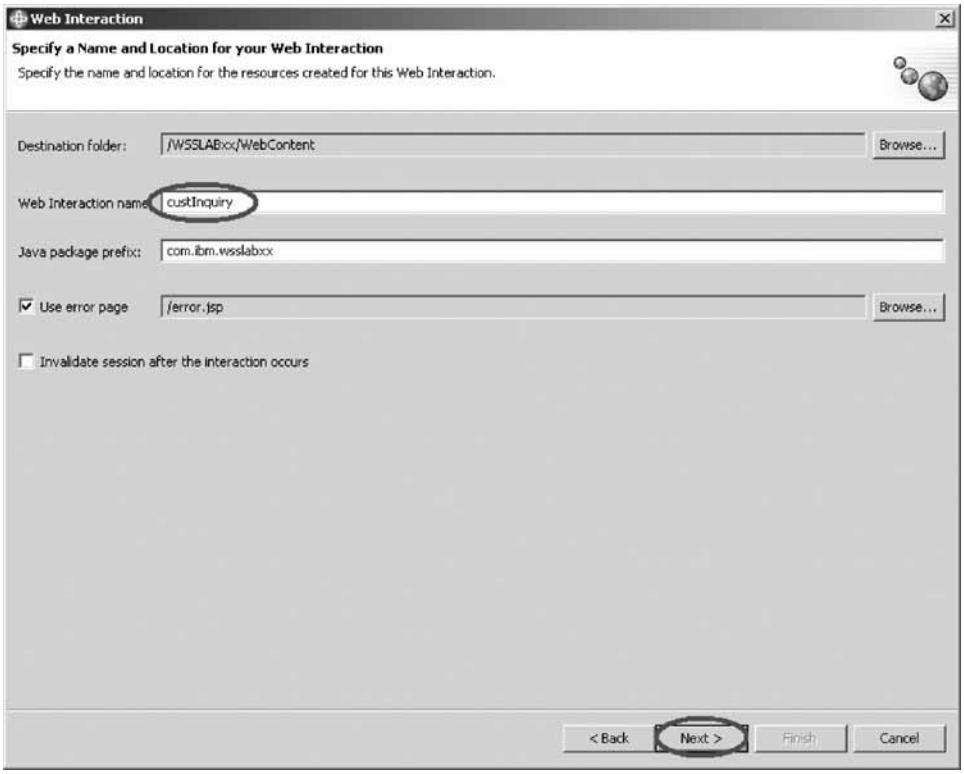
4. Select **Web Interaction** from the right pane.

5. Click **Next**.

   An interaction is the span between the submit of a request from the current Web page to the post of a new Web page. An interaction could be a simple request to the HTTP server to load another page. In this environment an interaction will be made up of a call to an iSeries program, waiting for the return of the program with data and then invoking a JSP to show the next Web page containing the data returned from the program.

   First you have to give the interaction a name so you can reference it later on.

   The Specify a Name and Location for your Web Interaction page opens.



6. Type `custInquiry` in the **Web Interaction name** field.
7. Click **Next** in the Web Interaction wizard.

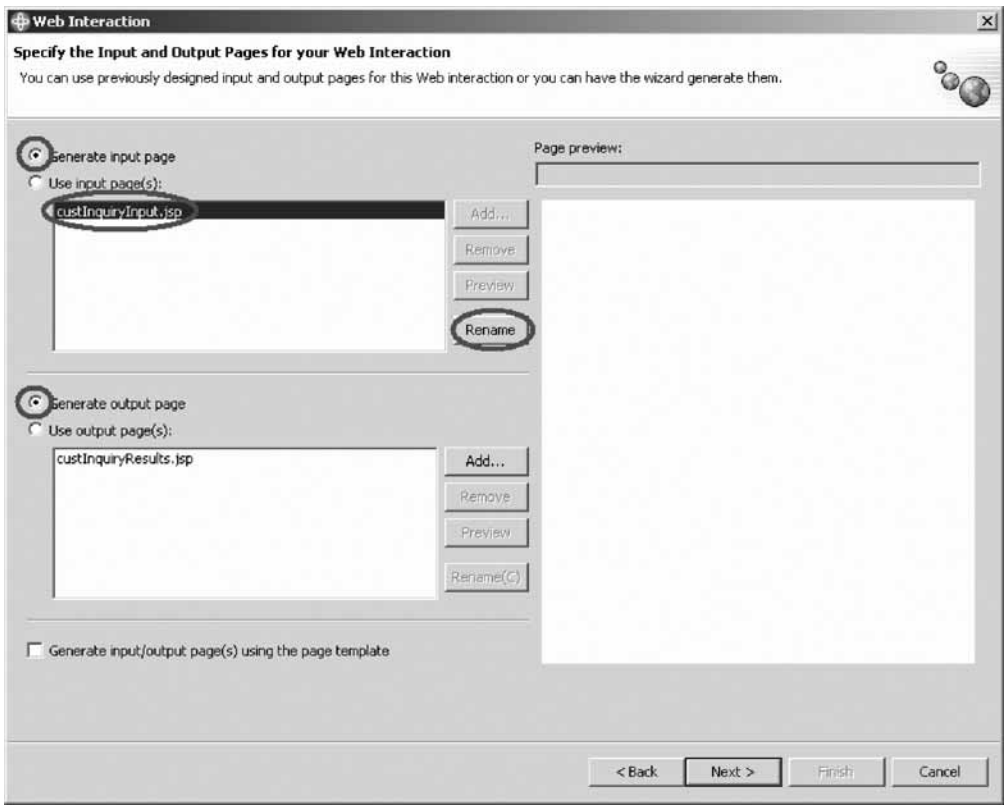
   The Specify the Input and Output Pages for your Web Interaction page opens.

You have started the Web Interaction wizard and now you are ready to begin "Exercise 5.2: Specifying the input and output page."

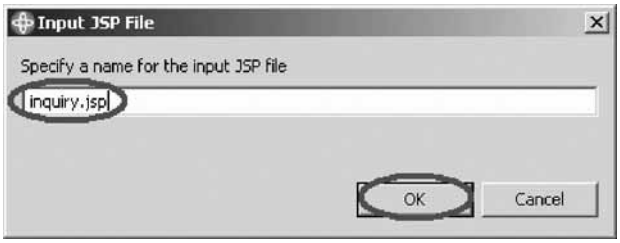
Exercise 5.2: Specifying the input and output page

Before you begin, you must complete "Exercise 5.1: Invoking the Web Interaction wizard" on page 58.

The next page of the Interaction wizard asks for information about the Web pages that are involved in this interaction. You have an input page that invokes the interaction and an output page that displays the results of interaction. The output page shows in the browser at the end of the interaction. The wizard generates both Web pages for you.



1. Click **Generate input page** radio button.
2. Click **Generate output page** radio button.
3. Select custInquiryInput.jsp and click **Rename**.



4. Enter `inquiry.jsp` as the new input JSP file name.
5. Click **OK**.
6. Select custInquiryResults.jsp and click **Rename**.
7. Enter `result.jsp` as the new output JSP file name.
8. Click **OK**.
9. Click **Next**.

   The Specify the Input and Output Parameters for your iSeries Host Program page opens.

   You have created an input page that invokes the interaction and an output page that displays the results of the interaction and now you are ready to begin "Exercise 5.3: Defining the iSeries GETDATA program invocation and parameters" on page 61.

To define the iSeries GETDATA program invocation and parameters go to "Exercise 5.3: Defining the iSeries GETDATA program invocation and parameters" on page 61.

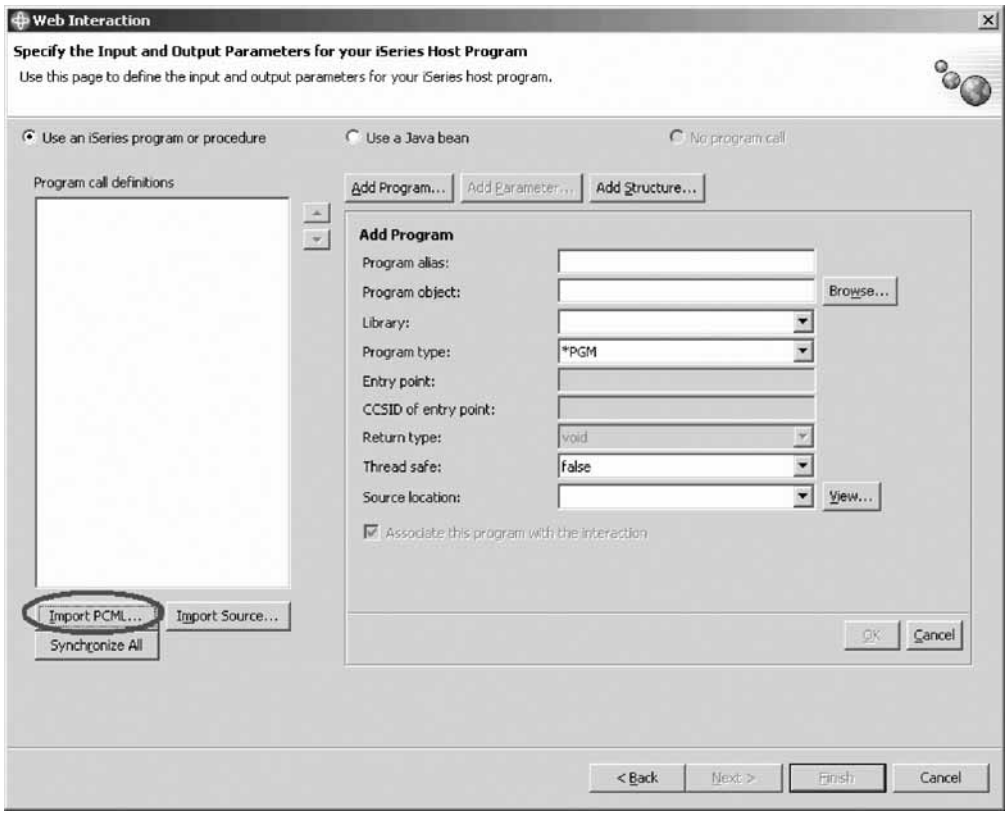
To define the iSeries GETDATAS service program invocation and parameters go to "Exercise 5.4: Defining the iSeries GETDATAS service program invocation and parameters" on page 66.

Exercise 5.3: Defining the iSeries GETDATA program invocation and parameters

Before you begin, you must complete "Exercise 5.2: Specifying the input and output page" on page 59.

The next page asks you to define the parameters you want to pass between the Web pages and the iSeries program that processes the requests.

Here you will re-use the PCML program interface definition that was created when you compiled your source member.

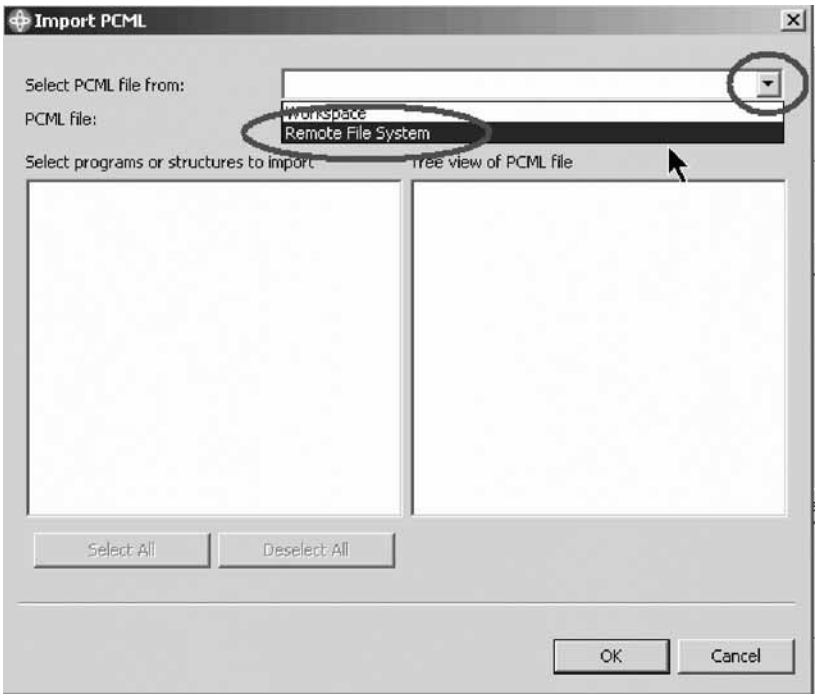


Getting the program interface definitions

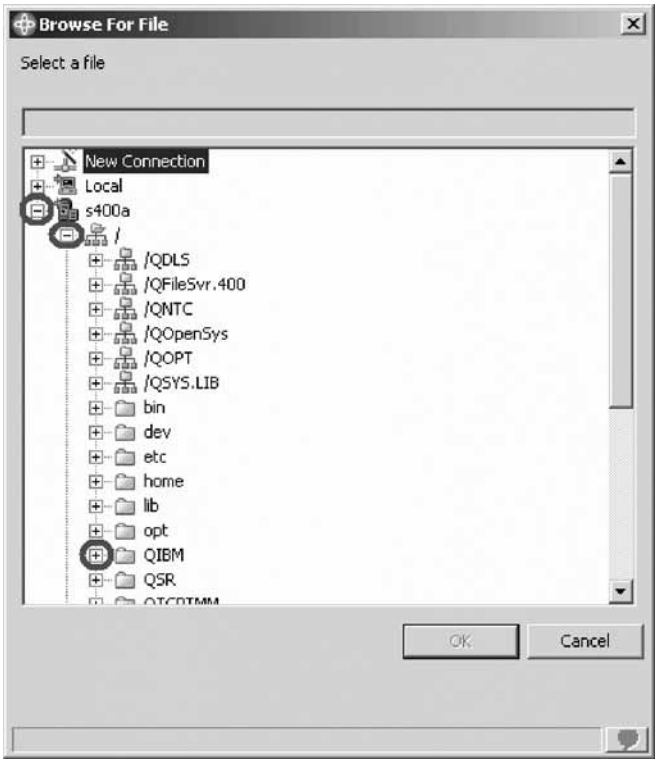
To get the program interface definitions that you want to call on the iSeries:

1. Click Import **PCML**

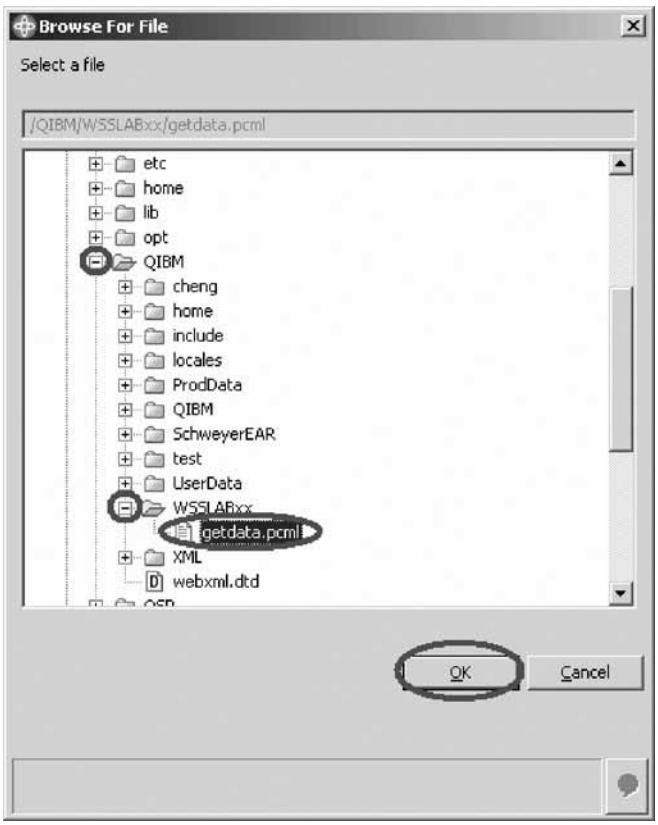
   The Import PCML dialog opens.



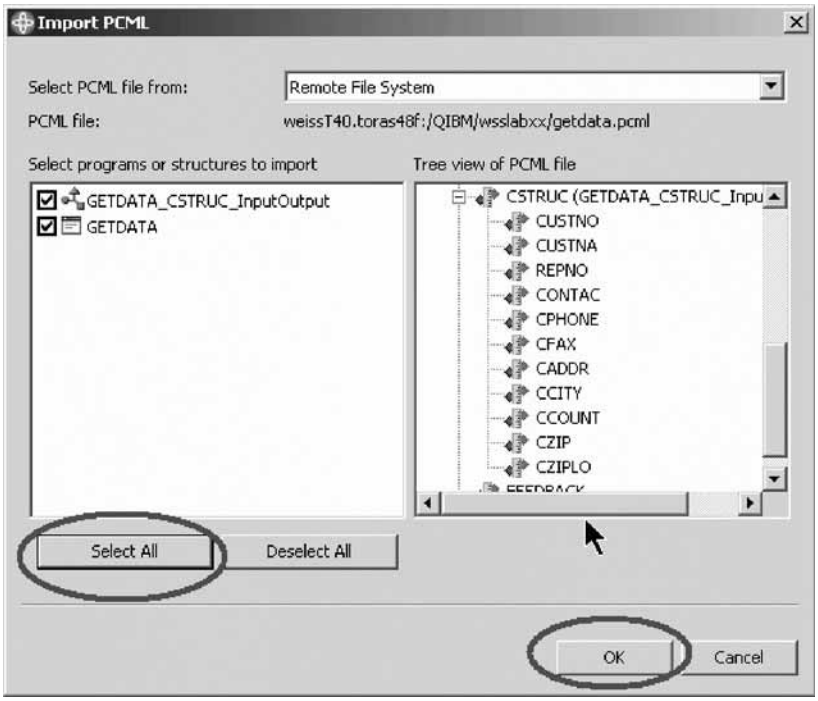
2. Click the arrow on the **Select PCML file from** field
3. Select **Remote File System** from the list.
   The **Browse For File** dialog opens.



4. Expand the tree view of the IFS drive to the directory where you stored the PCML file generated by the RPG compiler:
   Since you followed the instructions and used a program call, it should be in `QIBM/WSSLABxx/getdata.pcml`



5. Select the correct file.
6. Click **OK**.
7. The Import PCML dialog appears with a list of the program interface constructs contained in the PCML file.



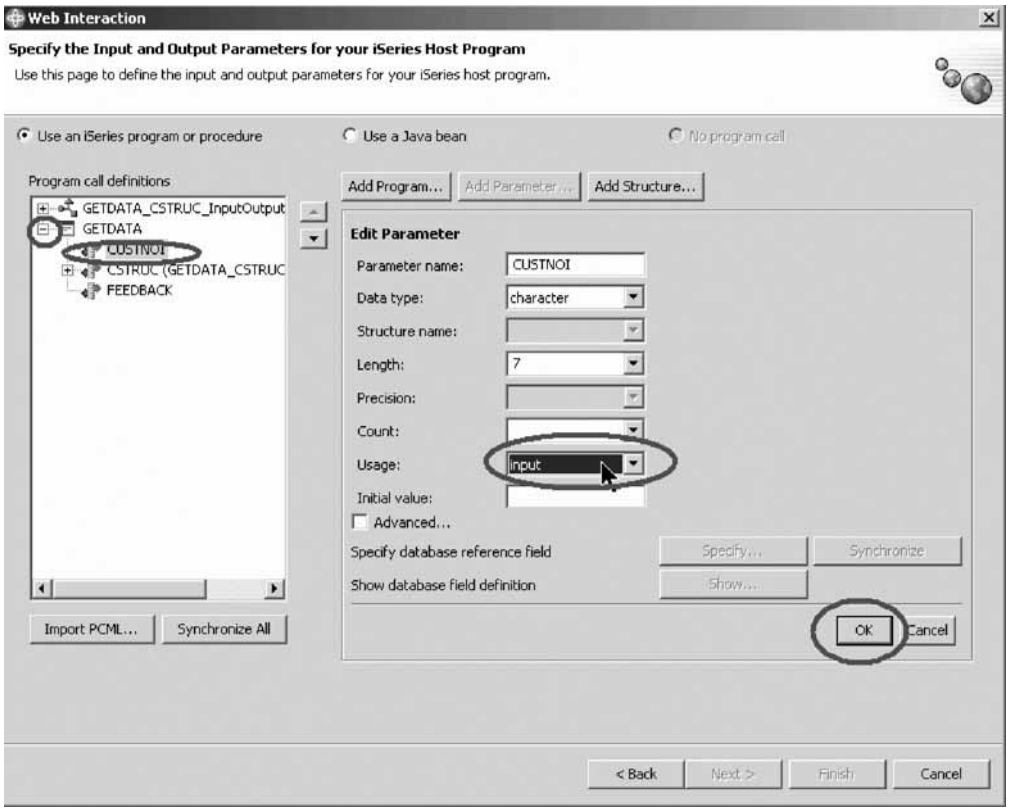
- A data structure

- A call interface

8. Click **Select All**.
9. Click **OK**.

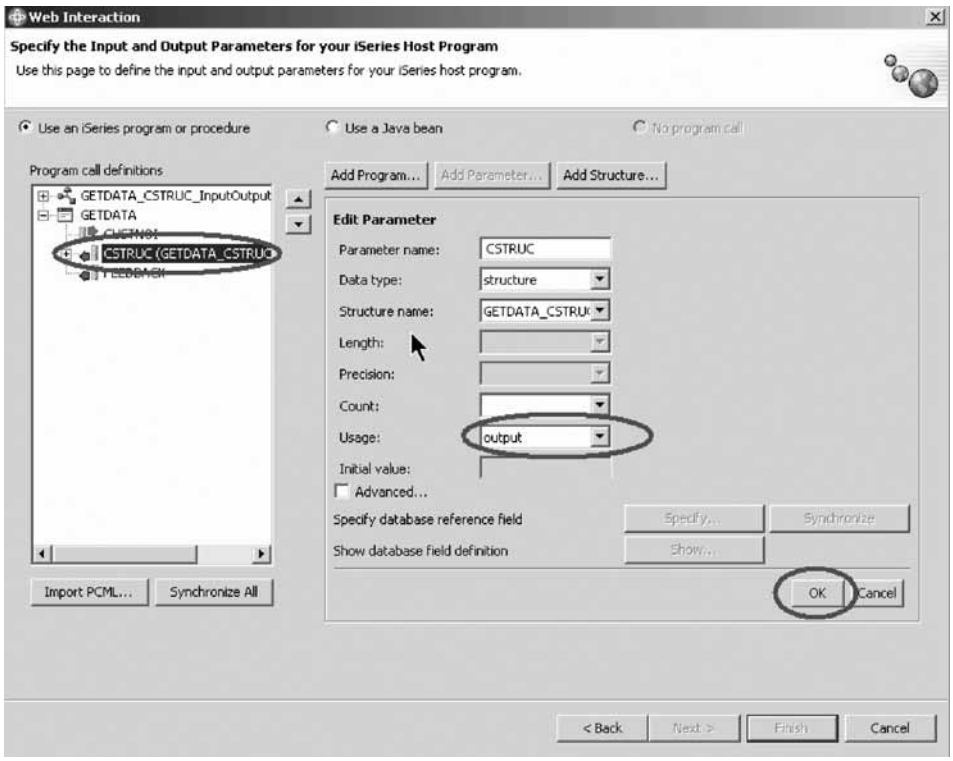
Specifying the input and output parameters

The Specify the Input and Output Parameters for your iSeries Host Program page opens again, now containing the program description you just imported.

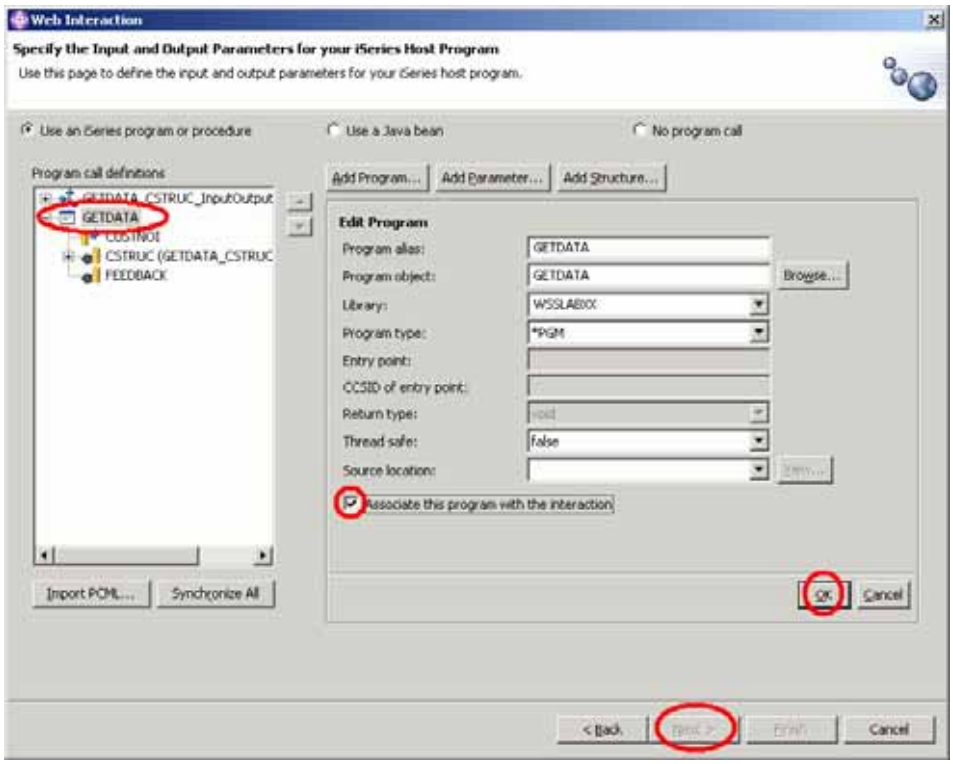


1. Expand GETDATA in the left pane under **Program call definitions**.

   The information from the PCML file gets included in the right pane and the parameters defined for this program appear.
2. Select the first parameter CUSTNOI.
3. In the right pane, change the **Usage** field from Input/Output to **Input** to indicate that this parameter is used to pass a value from the Web page to the program.
4. Click **OK**



5. In the left pane select the CSTRUC parameter.
6. In the right pane, change the **Usage** field from Input/Output to **Output**, indicating that this parameter is used to pass a value from the program to the result Web page.
7. Click **OK**.
8. Do the same for the **FEEDBACK** parameter.



9. Now select GETDATA in the left pane under **Program call definitions**. This will fill the right pane with the program information.
10. In right pane, select the check box **Associate this program with the interaction**.
11. Click **OK**.
12. Click **Next**.

You have defined the program interface definitions and now you are ready to begin "Exercise 5.5: Defining the input page content" on page 72.

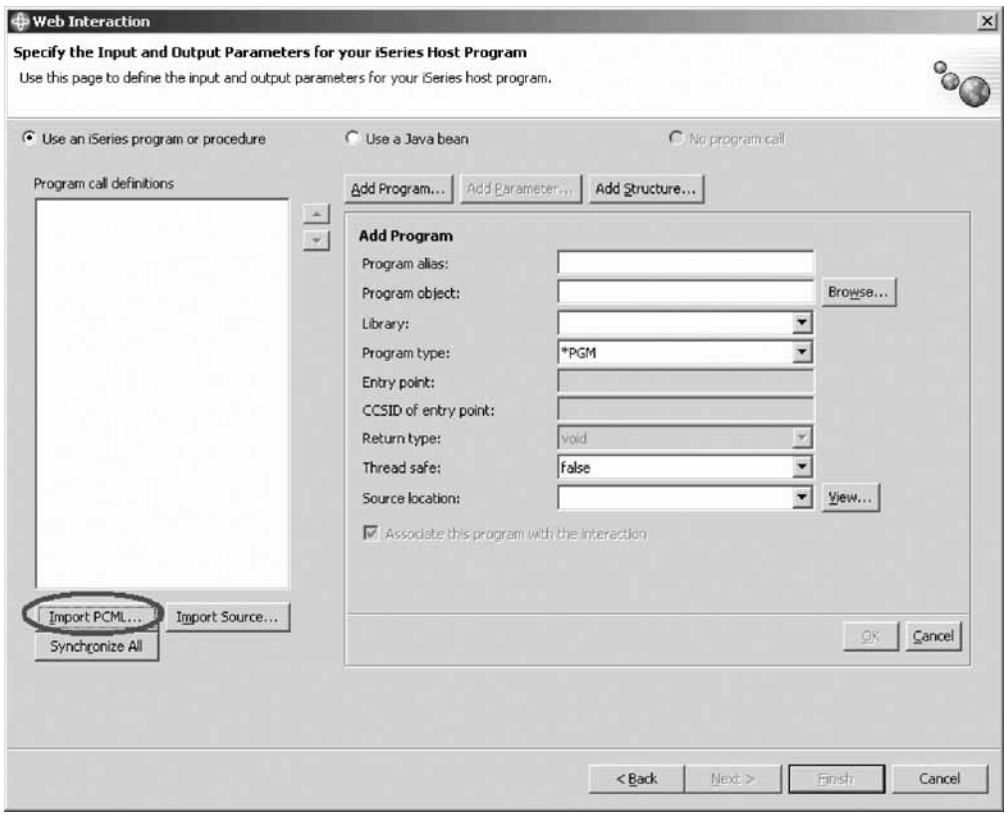
Skip the next exercise and continue with "Exercise 5.5: Defining the input page content" on page 72.

Exercise 5.4: Defining the iSeries GETDATAS service program invocation and parameters

Before you begin, you must complete "Exercise 5.2: Specifying the input and output page" on page 59.

The next page asks you to define the parameters you want to pass between the Web pages and the iSeries service program that processes the requests.

Here you will re-use the PCML program interface definition that was created when you compiled your source member.

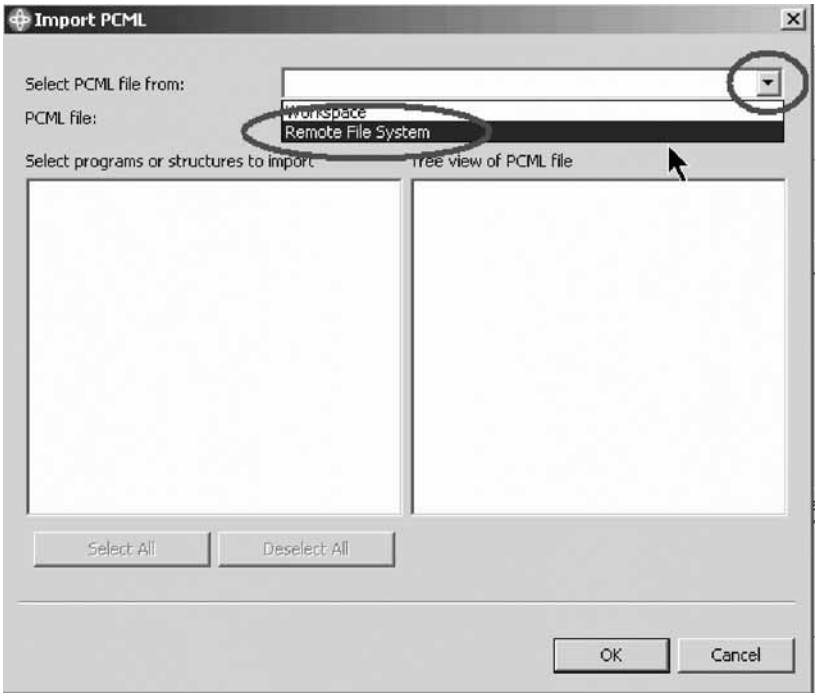


Getting the program interface definitions

To get the program interface definitions that you want to call on the iSeries:

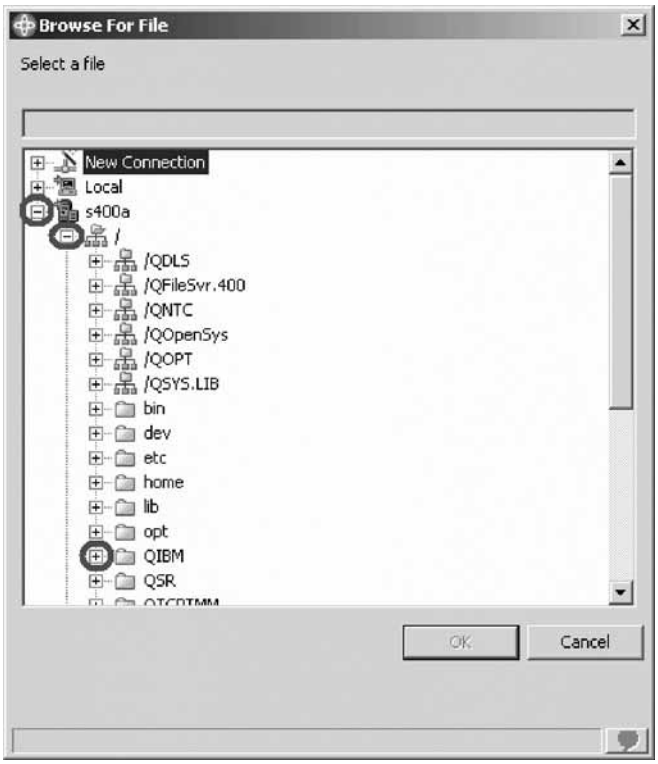
1. Click Import **PCML**

   The Import PCML dialog opens.



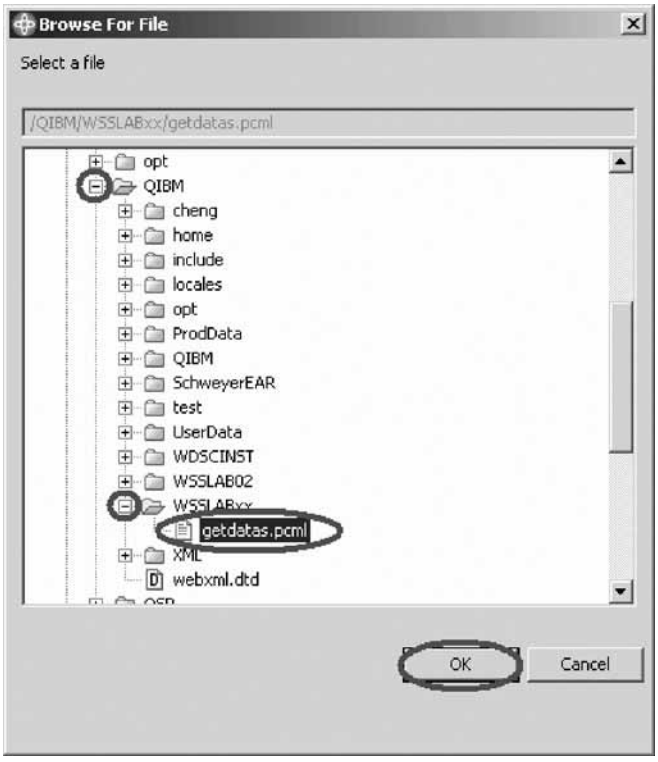
2. Click the arrow on the **Select PCML file from** field
3. Select **Remote File System** from the list.

   The **Browse For File** dialog opens.

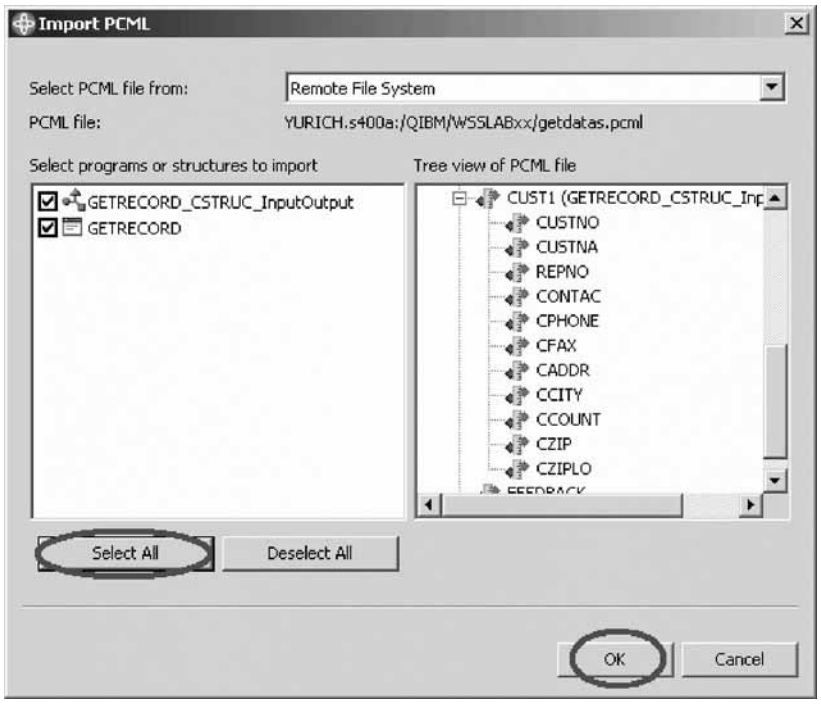


4. Expand the tree view of the IFS drive to the directory where you stored the PCML file generated by the RPG compiler:

   Since you used a service program, it should be in `QIBM/WSSLABxx/getdatas.pcml`.



5. Select the correct file.
6. Click **OK**.
7. The Import PCML dialog appears with a list of the program interface constructs contained in the PCML file.

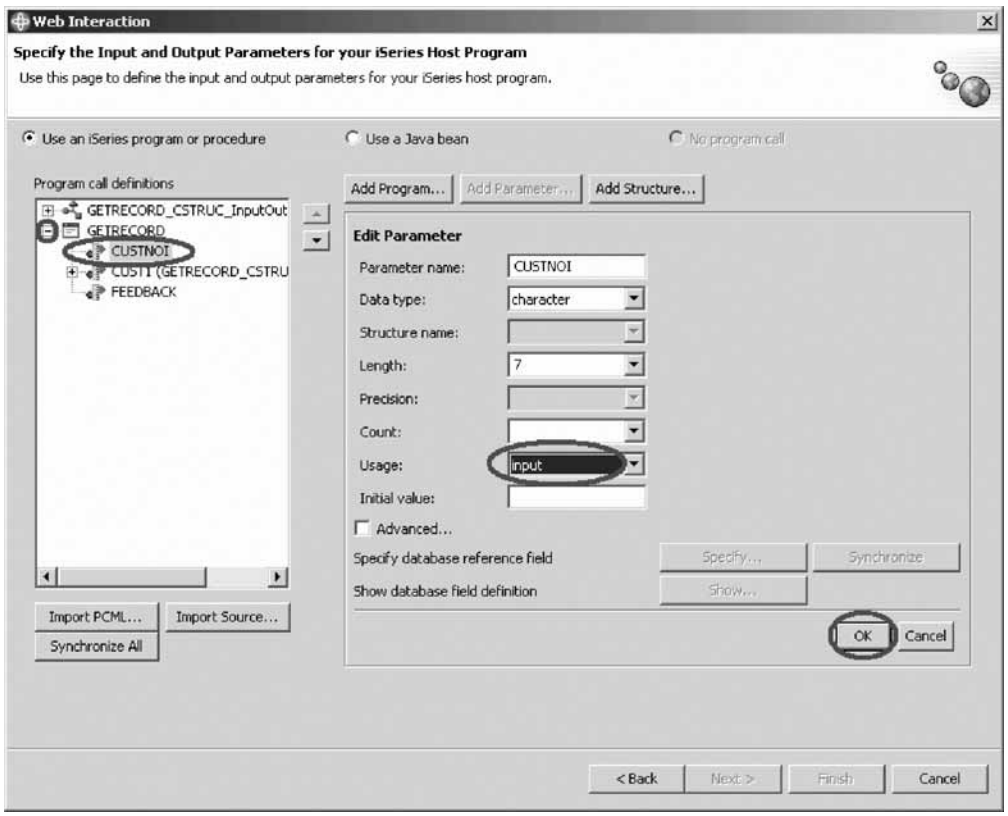


- A data structure
- A call interface

8. Click **Select All**.
9. Click **OK**.

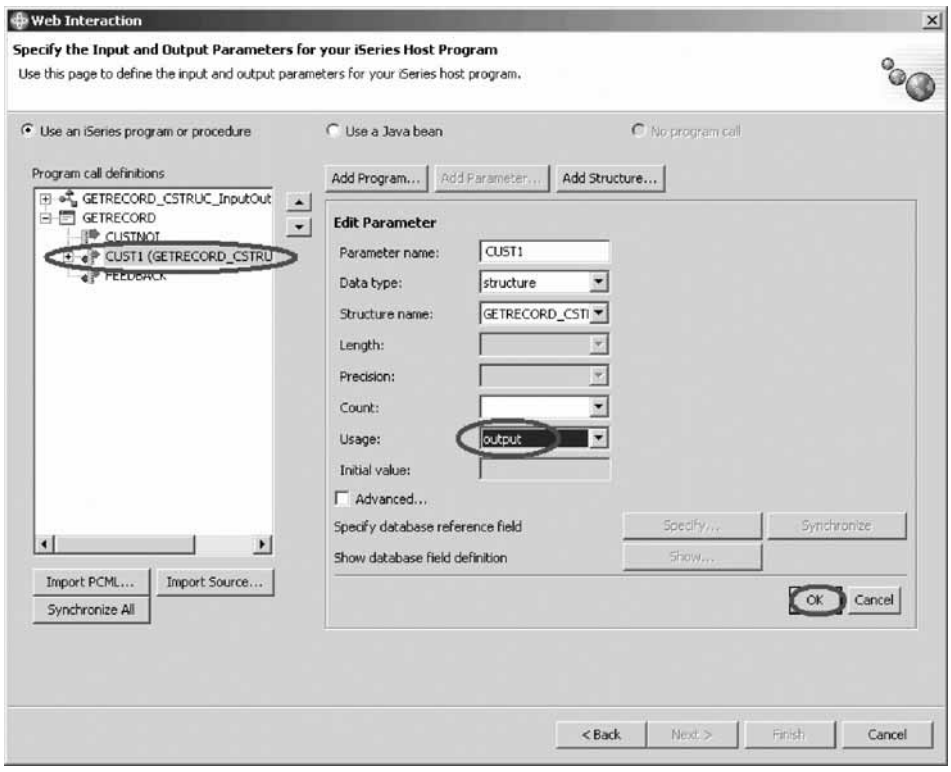
Specifying the input and output parameters

The Specify the Input and Output Parameters for your iSeries Host Program page opens again, now containing the program description you just imported.

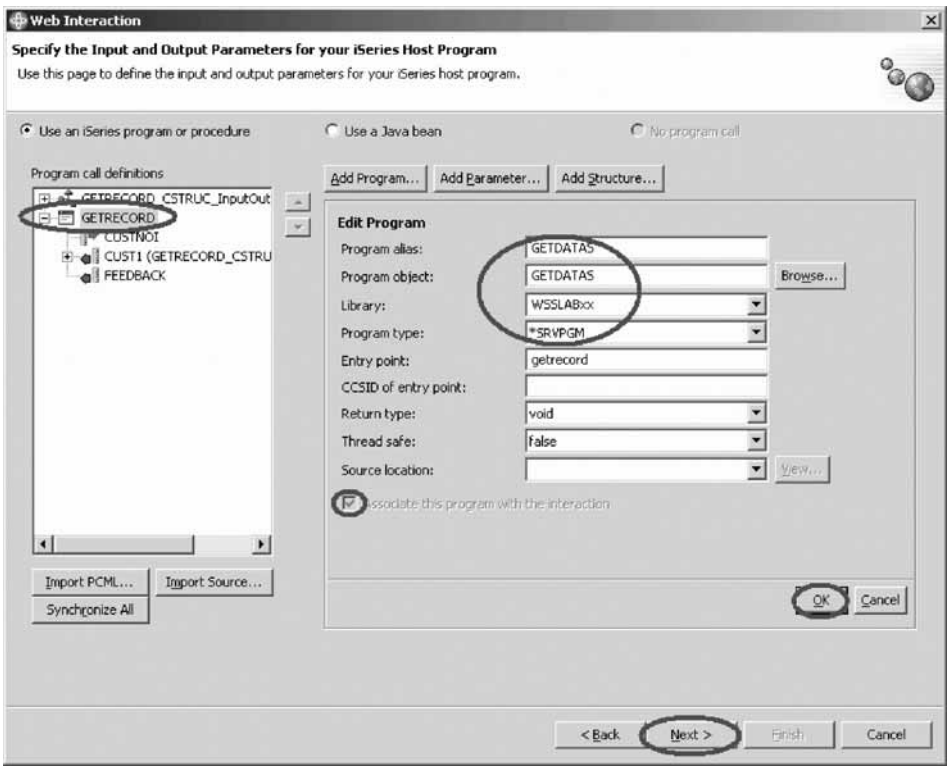


1. Expand GETRECORD in the left pane under **Program call definitions**.

   The information from the PCML file gets included in the right pane and the parameters defined for this program appear.
2. Select the first parameter CUSTNOI.
3. In the right pane, change the **Usage** field from Input/Output to **Input** to indicate that this parameter is used to pass a value from the Web page to the program.
4. Click **OK**



5. In the left pane select the CUST1 parameter.
6. In the right pane, change the **Usage** field from Input/Output to **Output**, indicating that this parameter is used to pass a value from the program to the result Web page.
7. Click **OK**.
8. Do the same for the **FEEDBACK** parameter.



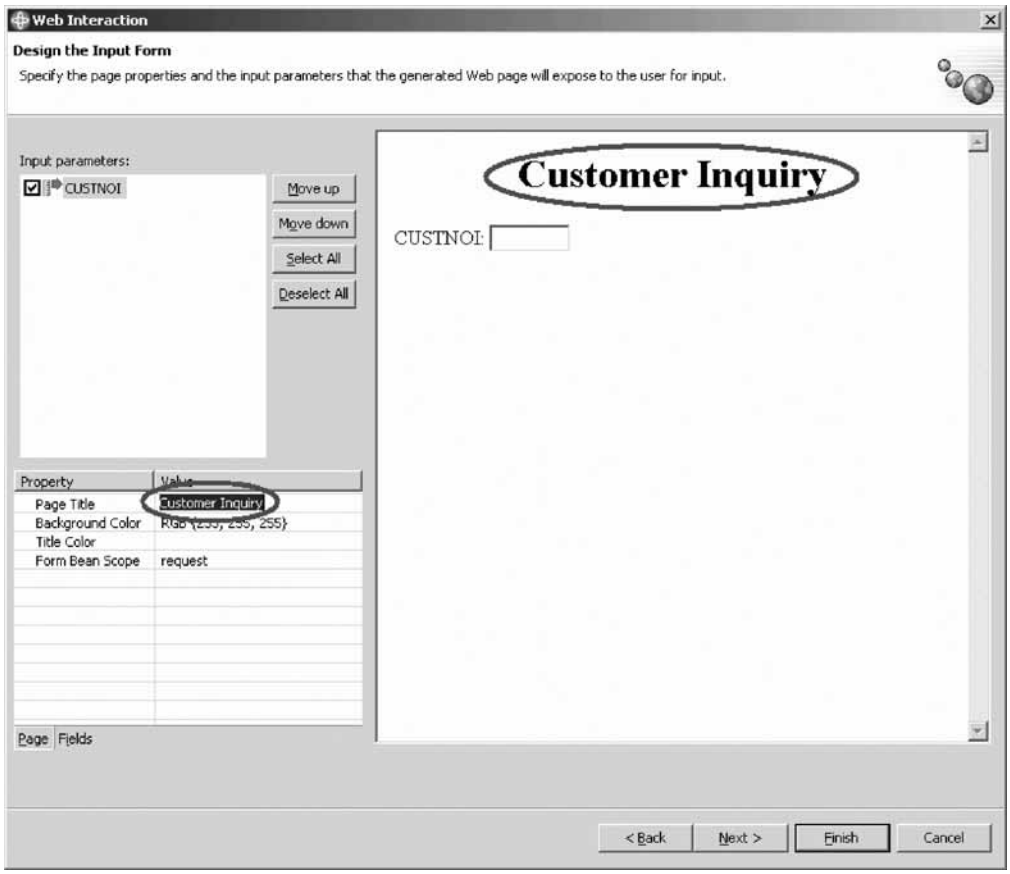
9. Now select GETRECORD in the left pane under **Program call definitions**. This will fill the right pane with the program information.
10. Change the name of the service program to GETDATAS.
11. Select the Program type *SRVPGM.
12. In right pane, select the check box **Associate this program with the interaction**.
13. Click **OK**.
14. Click **Next**.

You have defined the program interface definitions and now you are ready to begin "Exercise 5.5: Defining the input page content."

Exercise 5.5: Defining the input page content

Before you begin, you must complete "Exercise 5.3: Defining the iSeries GETDATA program invocation and parameters" on page 61or "Exercise 5.4: Defining the iSeries GETDATAS service program invocation and parameters" on page 66.

Now you need to specify which of the input parameters you want to see on the input page for this interaction that gets generated by the wizard. This is easy in this application since you only have the customer number input field to handle. You also have the capability to enhance the page layout by changing colors of the page background and changing the heading. The Design the Input Form page contains a properties table that shows all input capable parameters, and a sample of the Web page that will be created.



Changing the heading

To change the heading:

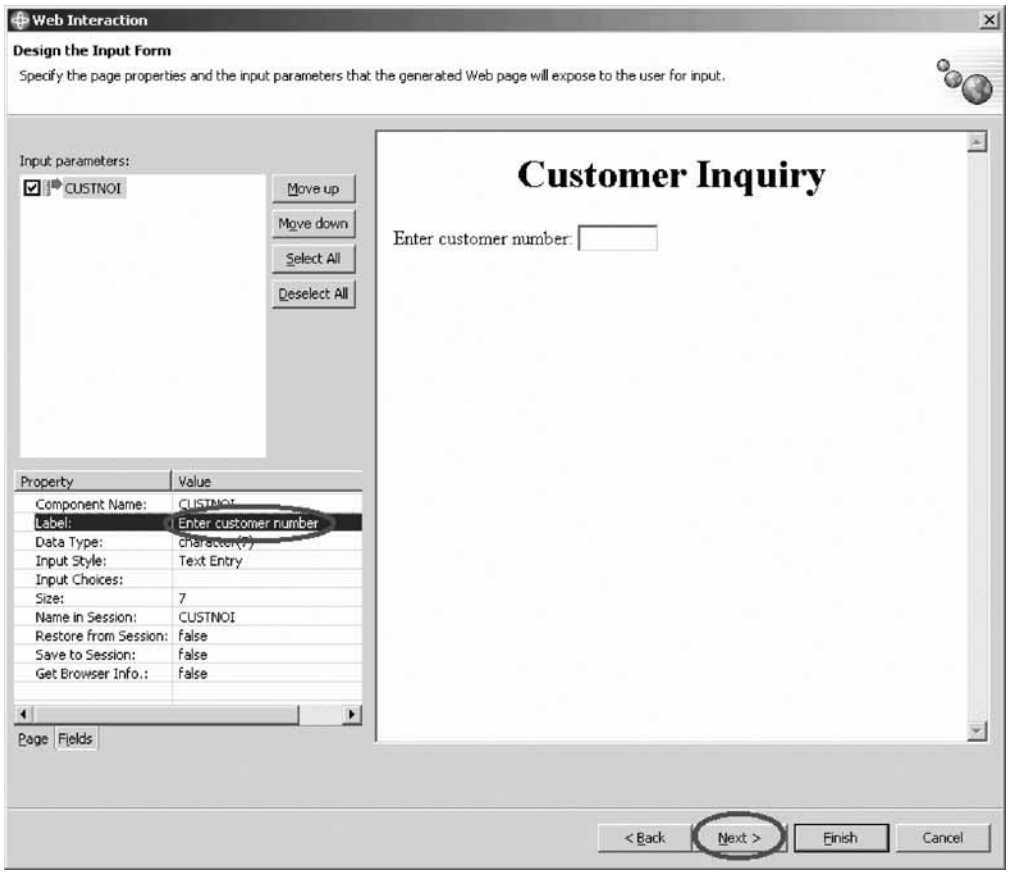
1. Click the field beside the **Page Title** property in the Property table and type `Customer Inquiry`.
2. Press **Enter** to apply this change to the sample page.

   Now you want to change the customer number field properties.
3. Click the **Fields** tab on the Properties table.

   This opens the field properties for the selected field in the list.

Changing a field label

To change a field label:



1. In the left pane of the Design the Input Form page, select CUSTNOI.
2. Click the field beside the **Label** property, and type `Enter customer number`.
3. Press **Enter**.
4. Click **Next** to advance to the next wizard page.

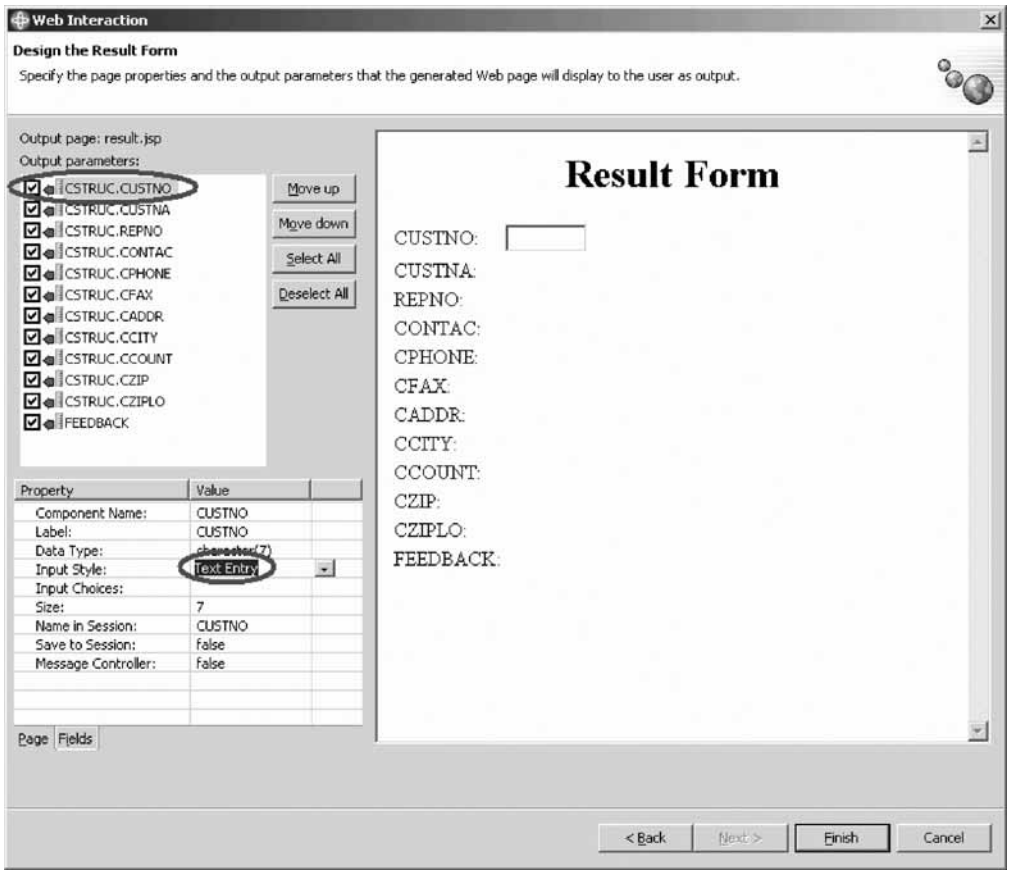
You have specified the input and output parameters and the input parameters to show on the input page and now you are ready to begin "Exercise 5.6: Defining the output page content."

Exercise 5.6: Defining the output page content

Before you begin, you must complete "Exercise 5.5: Defining the input page content" on page 72.

Now you can specify what data should be displayed on the output Web page generated for you. In the right pane of the wizard you see a sample of the page that will be generated. First, change the Input style of the output parameter CSTRUC.CUSTNO. This will enable you to use customer number as an input parameter to future features of your Web project.

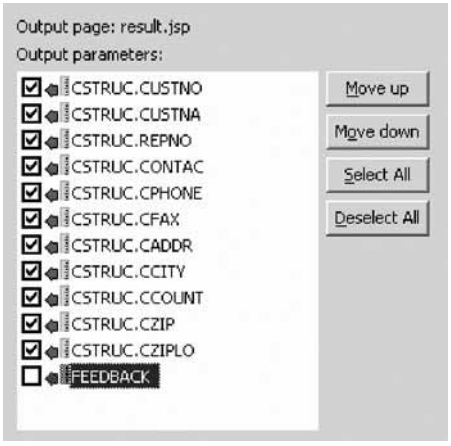
To define the output page:



1. Select CSTRUC.CUSTNO.
2. Click the **Fields** tab.
3. From the **Input Style** list, select **Text Entry**.

   By default all output parameters will be added to the Web page, but you don't want the data in the FEEDBACK parameter to be displayed.

   Let's go through the steps to erase the FEEDBACK parameter from the output page.



4. In the left pane of the Design the Result Form, clear the **FEEDBACK** check box. This will remove it from the Web page.

   You will stay on this page to specify the error handling.

You have specified what data to show on the output page and now you are ready to begin “Exercise 5.7: Specifying error handling.”

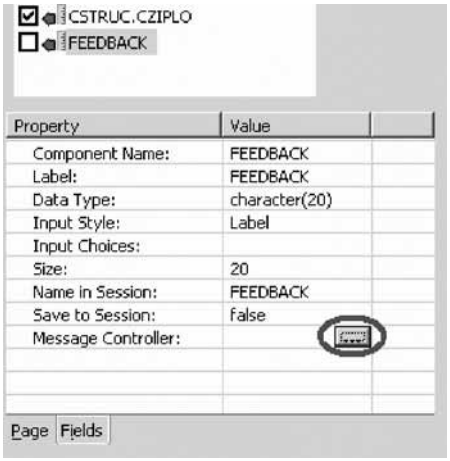
Exercise 5.7: Specifying error handling

Before you begin, you must complete “Exercise 5.6: Defining the output page content” on page 74.

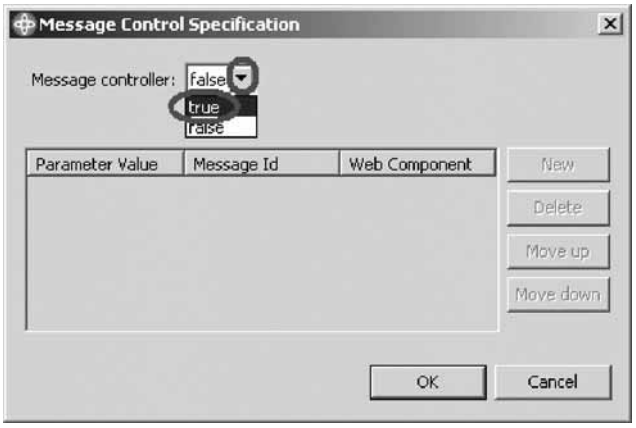
Now you can specify what should happen in case a customer number didn’t match the customer numbers in your database.

1. In the left pane of the Design the Result Form, select the **FEEDBACK** parameter.

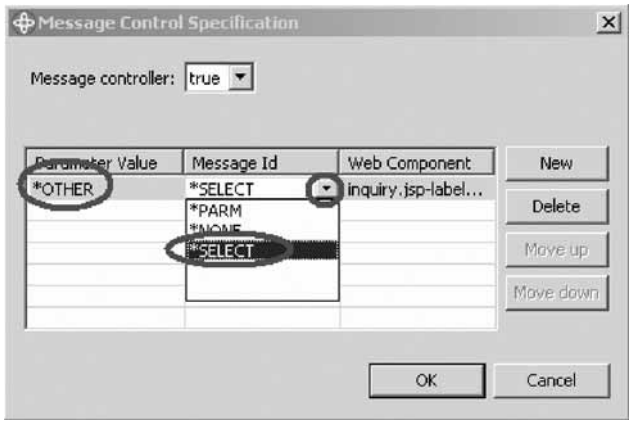
   You will see the properties for this variable in the Properties table.



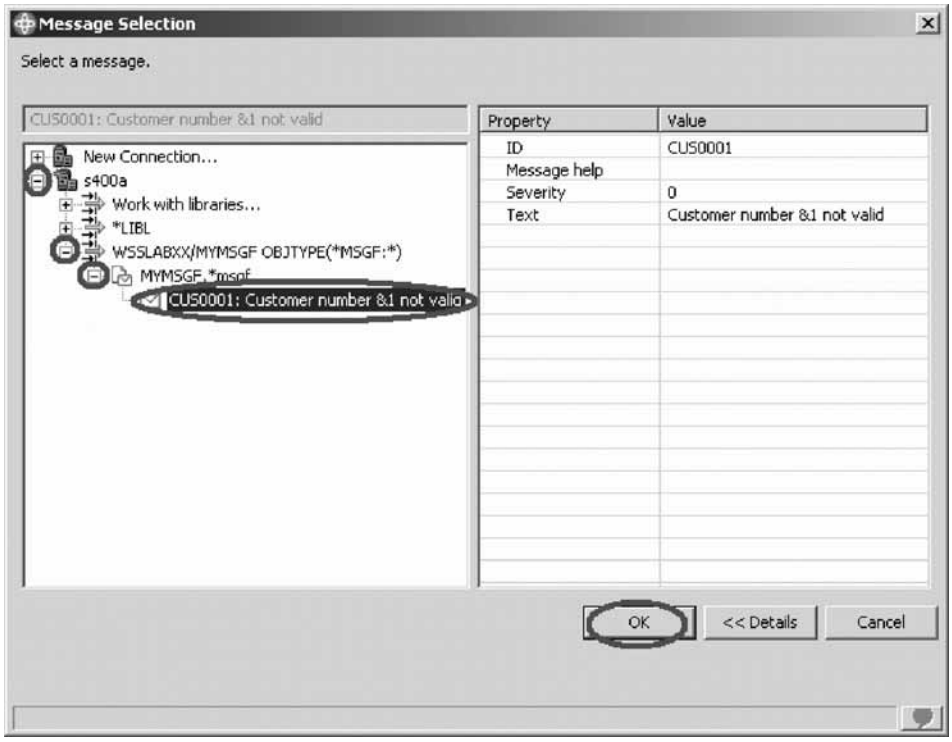
2. Locate the **Message Controller** heading in this Properties table.



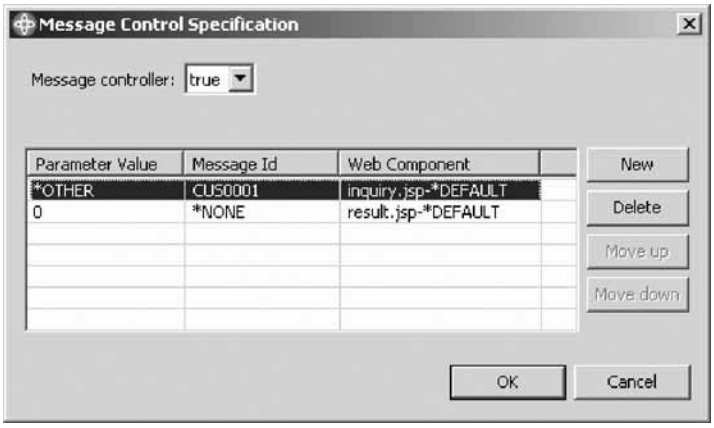
3. Click the field beside the **Message Controller** and set it to **true**.
4. Click **New**.
5. Set the **Parameter Value** to *OTHER from the list.



6. Set the **Message ID** to ***SELECT** from the list.
   This opens a Message Selection Window.
7. Drill-down to `s400a > WSSLABXX/MYMSGF > MYMSGF > CUS0001`



8. Click **OK**.
9. Leave the default value of the **Web Component**.
10. Click **New**.
11. Enter `0` as the **Parameter Value**.



12. Press **Enter**.
13. Set **Message ID** to ***NONE** using the list.
14. Set **Web Component** to **result.jsp-DEFAULT** using the list.
15. Click **OK**.

    Here is what you just did:

    - If the return value of the FEEDBACK parameter is **0**, display the customer details in the **result.jsp** Web page.
    - Any other return value in the FEEDBACK parameter, use the message ID CUS0001 in the specified message file, place the Customer number input in the message substitution variable, and display the Web page **inquiry.jsp** with the error message CUS0001.

    The message handling is defined.

You have specified what will happen when an incorrect customer number is entered in the customer number field and now you are ready to "Exercise 5.8: Enhancing the output page."

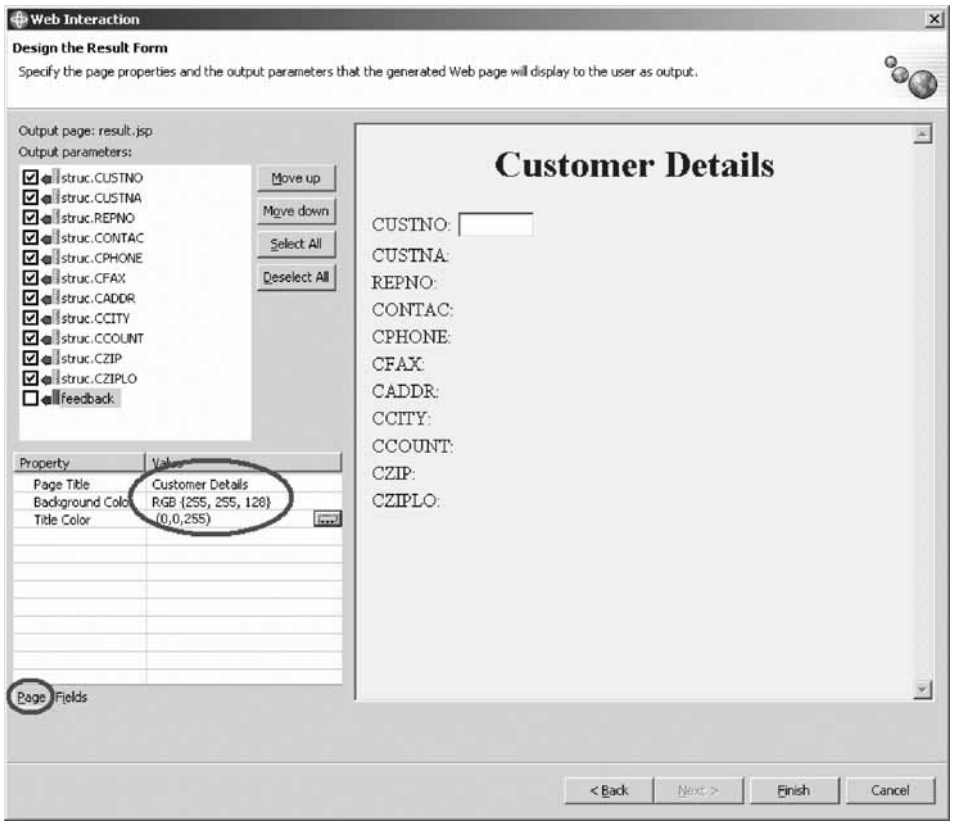
Exercise 5.8: Enhancing the output page

Before you begin, you must complete "Exercise 5.7: Specifying error handling" on page 76.

Next you enhance the look of the Web page.

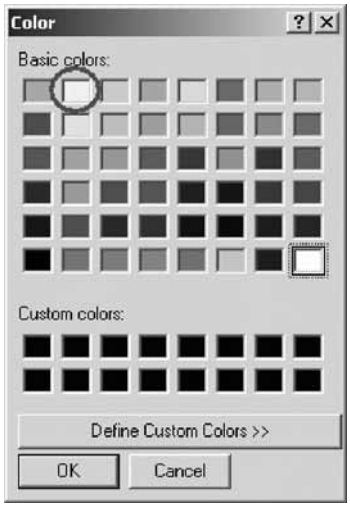
To enhance the output page:

1. Click the **Page** tab in the Properties table.



2. Click the field beside the **Page title** and type `Customer Details`.
3. Click the field beside the **Background Color** and click the selection button on the right side of the field.

   The Color palette opens.
4. Select a **light yellow** from the Color palette.



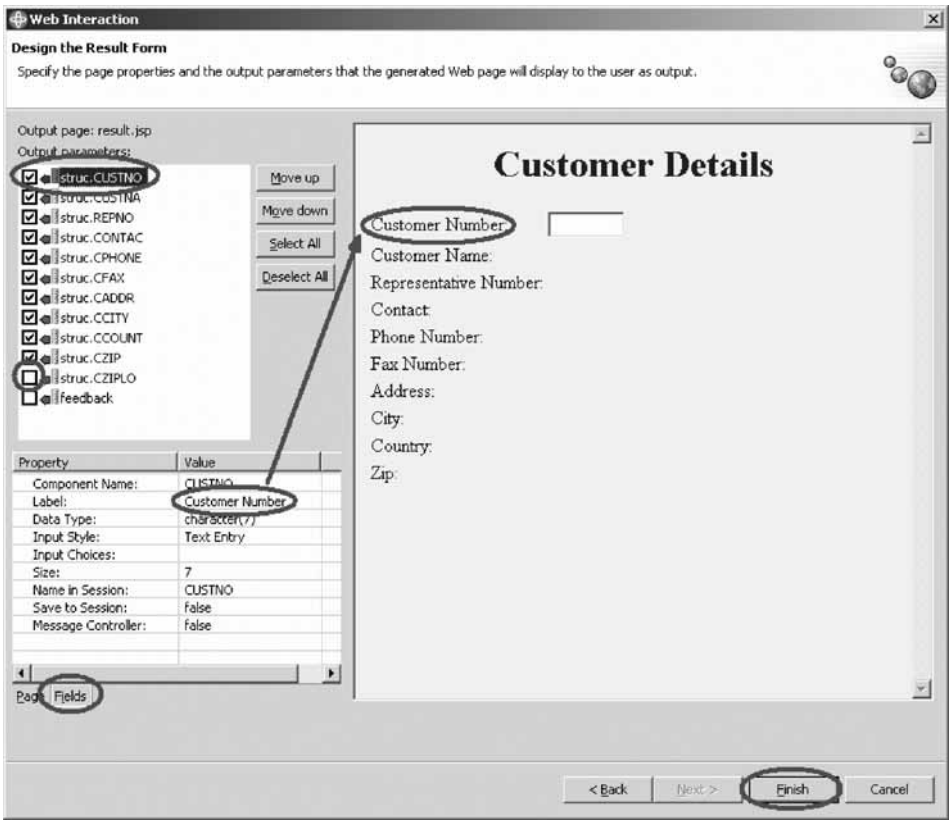
5. Click **OK** on the Color palette.
6. Click the field beside the **Title Color** and click the selection button on the right side of the field.
7. Select **Blue** from the Color palette.
8. Click **OK** on the Color palette.

You notice the sample page gets updated as you choose different properties.

If you don't like the suggested color choices, go ahead and create your own design.

You need to modify the labels of the fields that show up on the output JSP.

9. Click the **Fields** tab.
10. Select each field in the top left, and for each one, change the **Label** value to the values described in the table below.



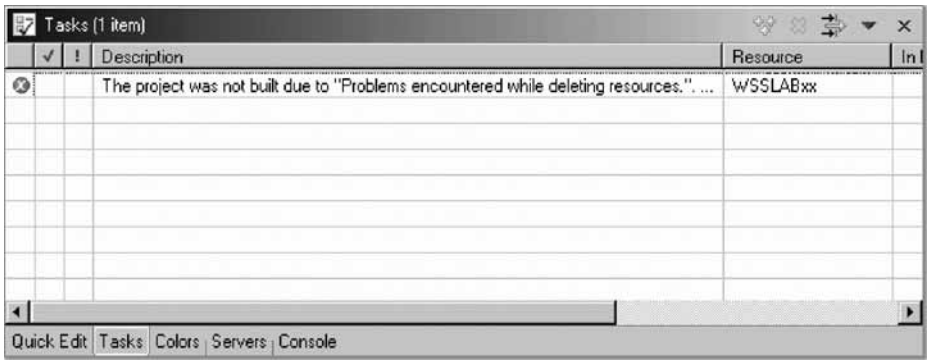
CUSTNO	Customer Number
CUSTNA	Customer Name
REPNO	Representative Number
CONTAC	Contact
CPHONE	Phone Number
CFAX	Fax Number
CADDR	Address
CCITY	City
CCOUNT	Country
CZIP	Zip

11. In the left pane of the Design the Result Form page, clear the **struc.CZIPLO** check box. You don't need to display this variable.
12. Click **Finish** on the Design the Result Form page.

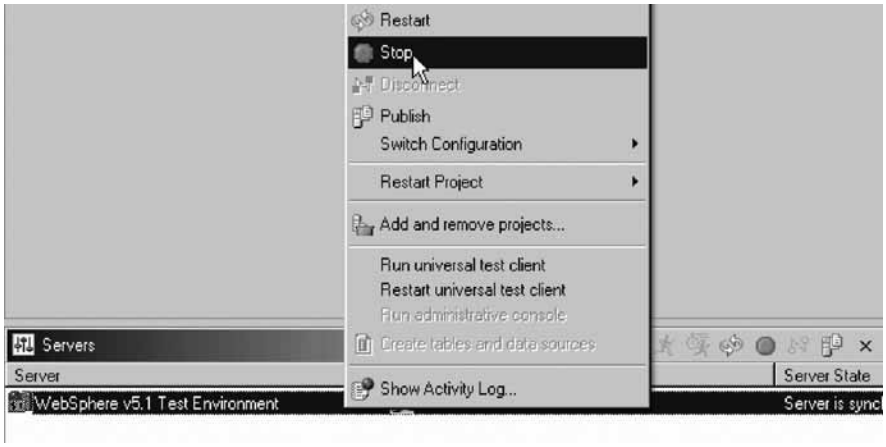
    The Interaction wizard creates the necessary files for your application:

- The file **custInquiry.wit** is created in the project. This file reflects the data you entered on the pages of the Web Interaction wizard for the interaction you identified as cust. To modify these settings, double-click on the file to open the Web Interaction wizard.
- The default package under the Java Source folder contains the .java files generated by the wizard.
- The classes folder under Web Content/WEB-INF contains the .class files generated from the .java files that were created by the wizard.

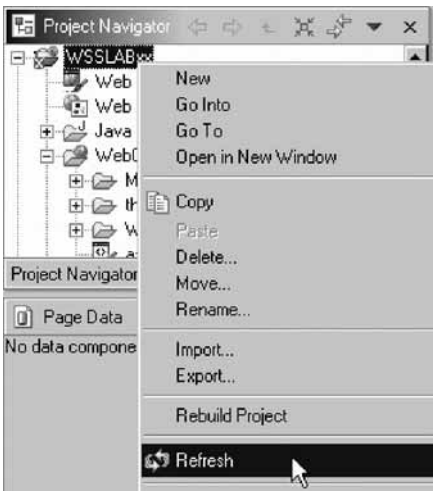
Note: If you encounter the following error message in your **Tasks** list:



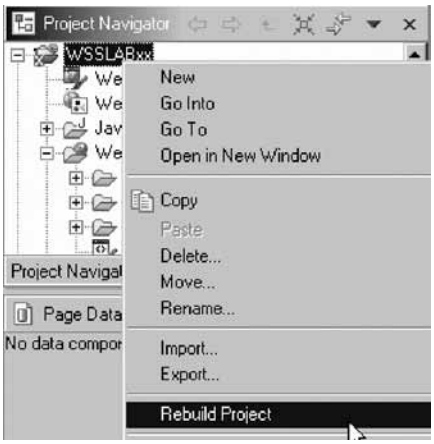
1. Click the **Server** tab to access the Server view.
2. Right-click the test server and select **Stop** on the pop-up menu.



3. Switch to your Project Navigator and right-click your project folder.
4. Click **Refresh** on the pop-up menu.



5. Right-click again your project folder and click **Rebuild Project**.



This should remove the error message in your Task view.

You have changed the background color of the output page and now you are ready to begin "Exercise 5.9: Ensuring the customer number field cannot be modified."

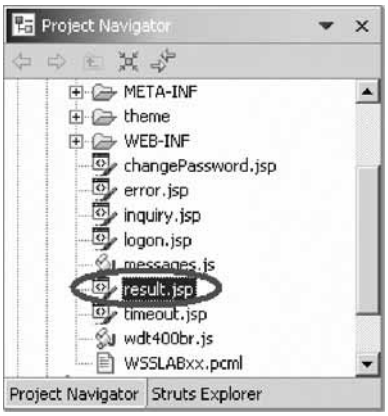
Exercise 5.9: Ensuring the customer number field cannot be modified

Before you begin, you must complete "Exercise 5.8: Enhancing the output page" on page 78.

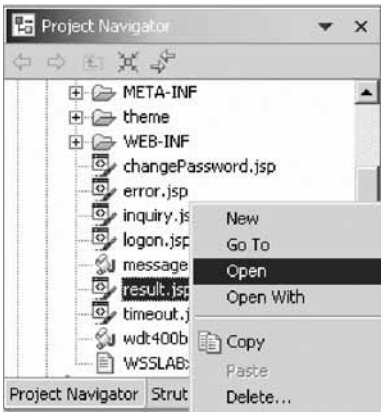
Let's make sure that the Customer Number field on your results.jsp page cannot be modified.

To ensure the customer number field cannot be modified:

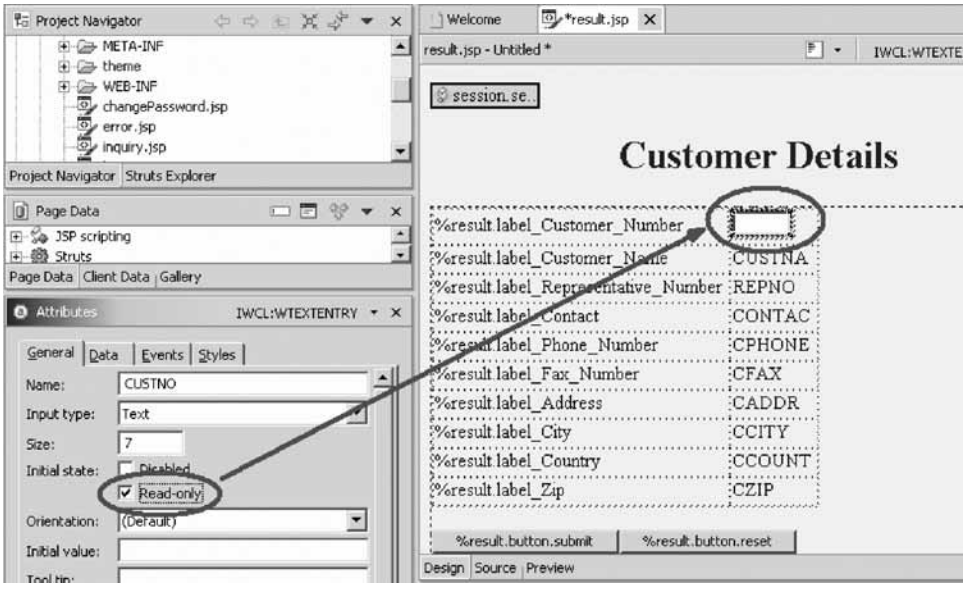
1. In the Project Navigator, expand **WebContent** and locate result.jsp.



2. Right-click **result.jsp**.



3. Click **Open** on the pop-up menu.
   This opens the result.jsp page on the Page Designer.



4. Select the **Text Entry** field of Customer Number in Page Designer.
   Notice the **Attributes** view of that Text field appears.
5. Select the **Read-only** check box.
   This prevents any modifications to the Customer Number.

6. Save your changes and close result.jsp.

You have made sure that the customer number field is read only and now you are ready to begin "Exercise 5.10: Visualizing the flow structure of your Web application."

Exercise 5.10: Visualizing the flow structure of your Web application

Before you begin, you must complete "Exercise 5.9: Ensuring the customer number field cannot be modified" on page 82.

A Web diagram is a view that helps you visualize the flow structure of a Struts-Based Web Application. Because of the indirectness involved with a Struts application, being able to visually see the application's flow can help you to better understand the application.

Struts

Before you create a Web diagram, let's first look at what Struts is all about. First, Struts is a set of Java classes and JSP tag libraries that provide a conceptual framework for developing Web applications. The Struts technology is open source and was developed as part of the Apache Software Foundation's Jakarta project.

Second, Struts provides numerous, custom JSP tags that are simple to use but are powerful in the sense that they hide information. The Page Designer does not need to know much about form beans, for example, beyond the bean names and the names of each field in a given bean.

Third, you can use the diagram editor to show all or part of a Struts application. For example, suppose you have a three-part Struts application. One part handles the login process, one part handles product inquiries, and a third part handles product updates. In this case you could draw three diagrams to represent this system, or you could draw the entire system in a single diagram. Because one diagram can be included inside another, it would probably make more sense to represent this Struts application using a set of three diagrams.

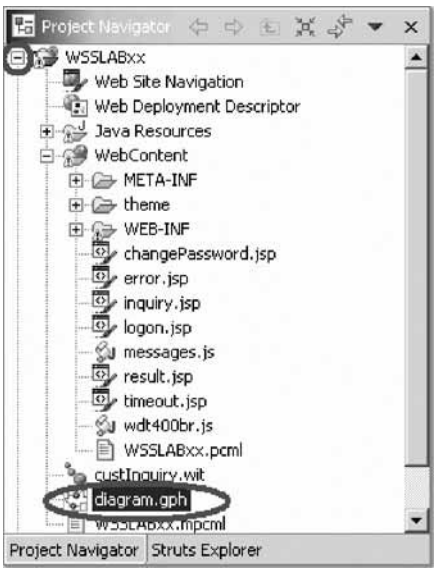
Now, that you know what Struts is, you can go ahead with this exercise.

You will now create a Web diagram that will show you the visual components of the Web application you just created.

An empty Web diagram (diagram.gph) should have been generated for you during project creation.

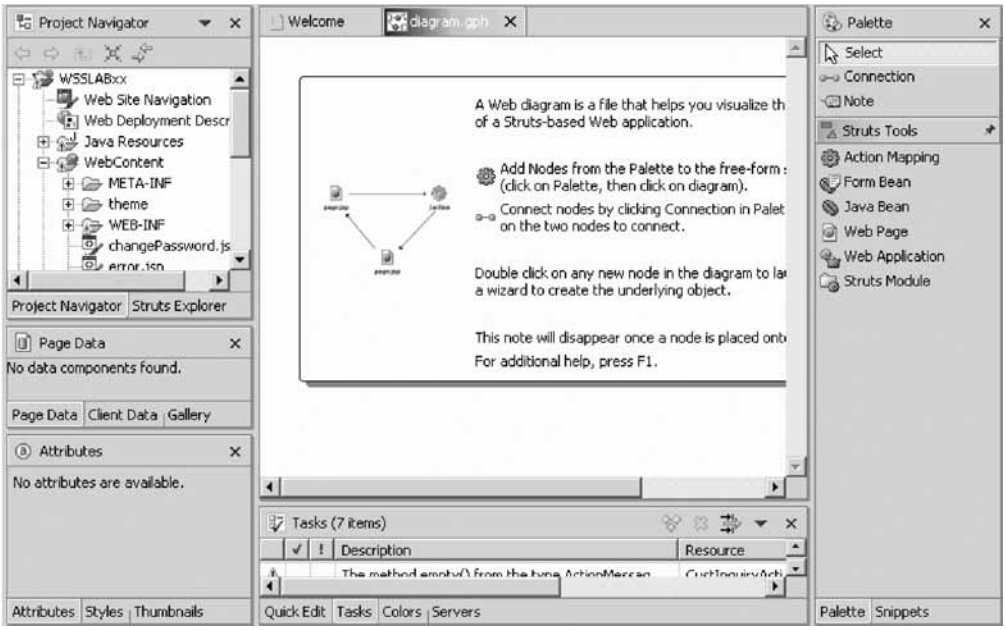
To create a Web diagram:

1. If you don't see it (diagram.gph) in the editor, in the Project Navigator view, expand WSSLABxx.



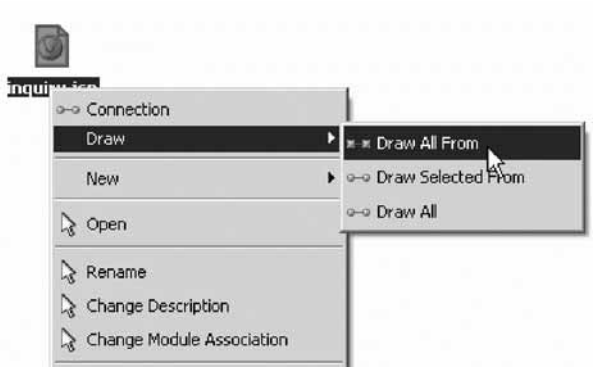
2. Locate and double-click **diagram.gph**.

You will now see diagram.gph in the editor.

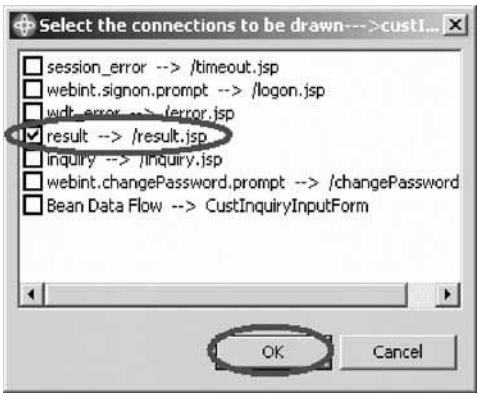


Now, let's populate the Web diagram:

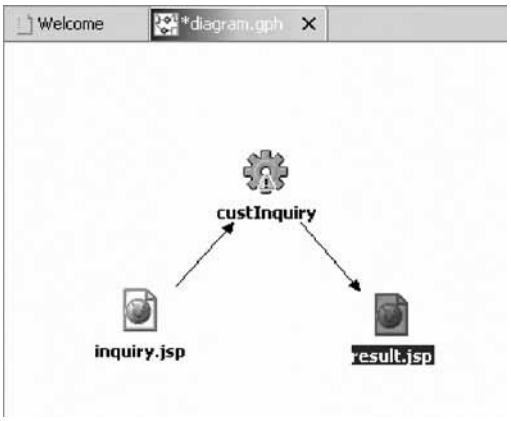
1. Locate inquiry.jsp under WSSLABxx and then WebContent in the Project Navigator.
2. Click inquiry.jsp and drag it to diagram.gph. You will notice the cursor change from to .
3. In the Web diagram, right click inquiry.jsp.



4. Click **Draw > Draw All From** on the pop-up menu.
   You will now see the custInquiry interaction.
5. Right-click custInquiry and click **Draw > Draw Selected From**.
6. Select the result.jsp check box



7. Click **OK**.
   You should now see the graphical representation of the Web interaction that you've created in a previous exercise.



8. Save and close diagram.gph.

You have seen the flow structure of a Struts-based Web application.

Module recap

You have completed Chapter 5, “Module 5. Creating a Web application,” on page 57. You have learned how to:

- Start the Web Interaction wizard
- Create an input page that invokes the interaction
- Create an output page that displays the results of the interaction
- Get the program interface definitions
- Specify the input and output parameters
- Specify the input parameters to show on the input page
- Specify what data to show on the output page
- Specify what will happen when an incorrect customer number is entered
- Change the background color of the output page
- Make sure the customer number field is read only
- Know what Struts is all about
- View the flow structure of a Struts-based Web application

You have created your Web application. Continue to Chapter 6, “Module 6. Running the Web application,” on page 89.

Chapter 6. Module 6. Running the Web application

This module teaches you how to test the Web application locally without deploying the files in the Web project to the host server. Testing your Web application locally includes using the iSeries Web Tools Run-time Configuration wizard to specify the authentication and run-time values for every host program or procedure call made by any Web interaction in the Web project. This is something you have already done in Chapter 2, “Module 2. Creating an RPG program,” on page 7 or Chapter 3, “Module 3. Creating the RPG service program for your Web application (optional),” on page 25.

In this module, you will:

- Check the Web perspective is open
- Locate input page of the Web application
- Invoke the Web application to run in the WebSphere Test Environment
- Test the Web application

Exercises

The exercises in this module must be completed in order:

- “Exercise 6.1: Opening the Web perspective”
- “Exercise 6.2: Finding the Web application and running it” on page 90

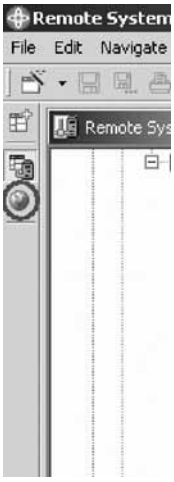
Time required

This module will take approximately **10 minutes** to complete.

Exercise 6.1: Opening the Web perspective

To open the Web perspective:

1. Click the Web perspective icon, on the left taskbar of the workbench.



2. If there is no Web perspective icon, select **Window > Open Perspective > Web** from the workbench menu.

© Copyright IBM Corp. 2001, 2005

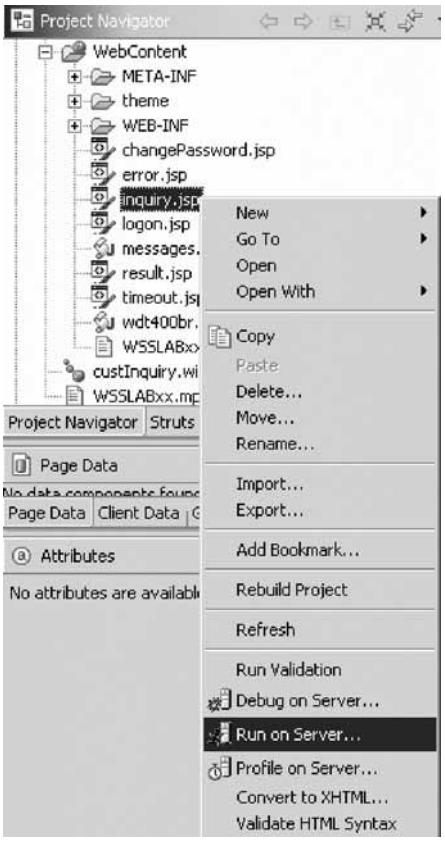
You have checked that the Web perspective is open and now you are ready to begin “Exercise 6.2: Finding the Web application and running it.”

Exercise 6.2: Finding the Web application and running it

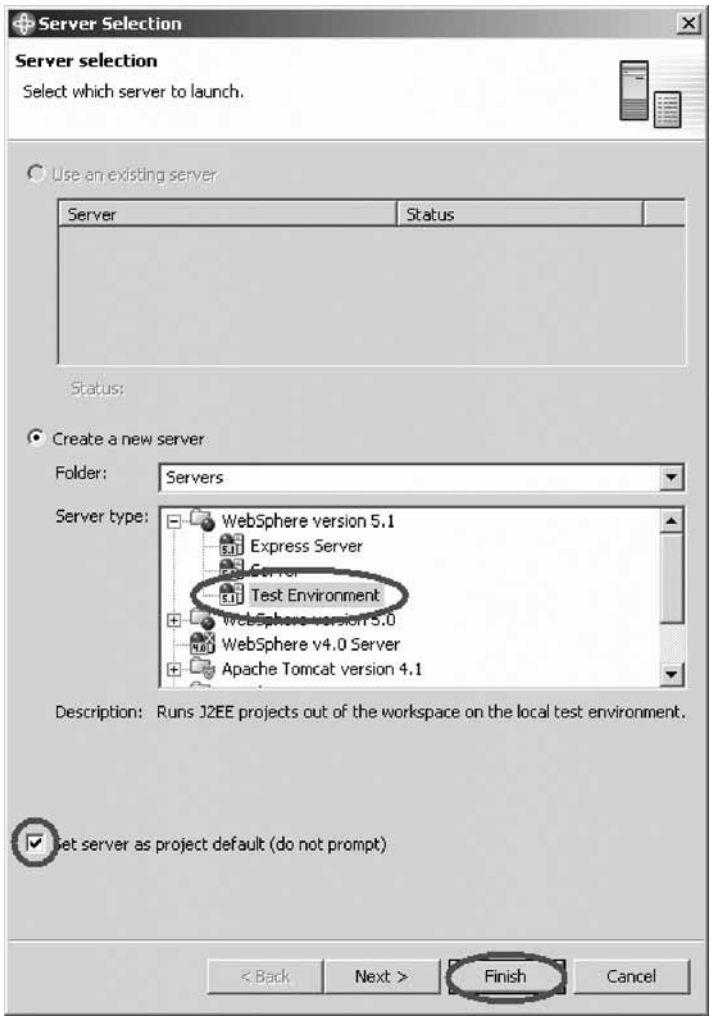
Before you begin, you must complete “Exercise 6.1: Opening the Web perspective” on page 89.

To test your application locally using the WebSphere test environment:

1. In the Project Navigator, expand your project WSSLABxx.
2. Expand the **WebContent** folder.
3. Right-click inquiry.jsp and click **Run on Server** on the pop-up menu.



You will see a message to select the Server to run in the Test Environment.



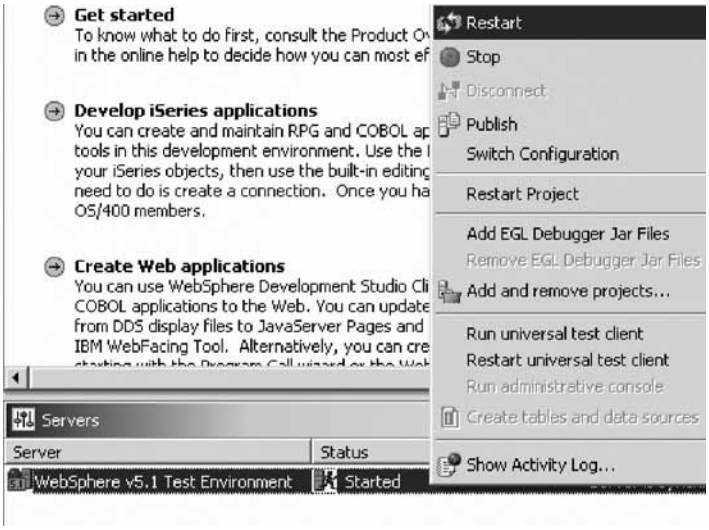
4. Make sure **Test Environment** is selected in the **Server type** list.
5. Select the **Set server as project default (do not prompt)** check box.
6. Click **Finish** to configure and launch the server.

   After the WebSphere server has started, your Web application input page will display in the workbench browser.

7. Type `0010100` as the **Customer Number**.
8. Click **Submit**.

   This launches the generated output or result page.

   a. If you see a **Page not found error**, go to the **Servers** view.



   b. Restart the server instance. Then go back and try the **Run on Server** option again; the Server sometimes needs to be restarted to pick up the changes.

   You will see the Result page with customer data appearing in the workbench browser.

Next, test the message handling.

9. Click the Back button in the browser to get back to the first page.
10. Enter a wrong customer number, for example, 9999.

11. Click **Submit**.

    An error message appears.

This capability is generated by the Web Interaction wizard. Remember that you specified something for the feedback field; you specified the message file to be accessed if the feedback field contains a non 0 value; in this case the feedback parameter will contain the message ID to be used for the error message.

You have found and run the Web application.

Module recap

You have completed Chapter 6, “Module 6. Running the Web application,” on page 89. You have learned how to:

- Check the Web perspective is open
- Locate input page of the Web application
- Invoke the Web application to run in the WebSphere Test Environment
- Test the Web application

Now that you have run your Web application you learn how to debug your Web application. Continue to Chapter 7, “Module 7. Debugging a Web application,” on page 95.

Chapter 7. Module 7. Debugging a Web application

This module teaches you how to debug a Web application. You will learn how to set up the debug environment, use a service entry point, add and remove breakpoints, display a variable, and view the call stack.

You will debug a Web application that invokes the iSeries GETDATA program.

In this module, you will:

- Recognize the features of the iSeries Integrated Debugger
- Start a debug session with service entry points
- Set breakpoints
- Monitor an expression
- Expand a thread to show every program, module, procedure and method on the call stack at the current execution point
- Run the program to termination

Exercises

The exercises in this module must be completed in order.

- "Exercise 7.1: Introducing the iSeries Integrated Debugger"
- "Exercise 7.2: Starting a Debug session using service entry points" on page 96
- "Exercise 7.3: Adding and deleting breakpoints" on page 97
- "Exercise 7.4: Displaying a variable" on page 98
- "Exercise 7.5: Viewing the call stack" on page 99
- "Exercise 7.6: Closing the debug session" on page 99

Time required

This module will take approximately **15 minutes** to complete.

Exercise 7.1: Introducing the iSeries Integrated Debugger

The Integrated Debugger is a source-level debugger that enables you to debug and test an application that is running on an iSeries system. It provides a functionally rich interactive graphical interface that allows you to:

- View source code or compiler listings, while the program is running on the iSeries system.
- Set, change, delete, enable and disable line breakpoints in the application program. You can easily manage all your breakpoints using the Breakpoints view.
- Set watch breakpoints to make the program stop whenever a specified variable changes.
- View the call stack of your program in the Debug view. As you debug, the call stack gets updated dynamically. You can view the source of any debug program by clicking on its call stack entry.
- Step through your code one line at a time.
- Step into or step over program calls and ILE procedure calls.

© Copyright IBM Corp. 2001, 2005

- Display a variable and its value in the Monitors view. The value can easily be changed to see the effect on the program's execution.
- Locate procedure calls in a large program quickly and easily using the Modules/Programs view.
- Debug multithreaded applications, maintaining separate stacks for each thread with the ability to enable and disable any individual thread.
- Load source from the workstation instead of the iSeries – useful if you don't want the source code on a production machine.
- Debug client/server and distributed applications.

The Debugger supports RPG/400® and ILE RPG, COBOL and ILE COBOL, C, C++ and CL.

In the following exercises you will be given the opportunity to learn about some of the basic features of the Debugger.

You have been introduced to the debugger and now you are ready to begin "Exercise 7.2: Starting a Debug session using service entry points."

Exercise 7.2: Starting a Debug session using service entry points

Before you begin, you must complete "Exercise 7.1: Introducing the iSeries Integrated Debugger" on page 95.

In this exercise you will use service entry points to debug a Web application. The service entry point feature is designed to allow easy debugging of Web applications that invoke business logic written in RPG, COBOL, CL, or even C or C++. The service entry point is a special kind of entry breakpoint that can be set directly from the Remote System Explorer. It is triggered when the first line of a specified procedure is executed in a job that is not under debug. Service entry points allow you to gain control of your job at that point. A new debug session gets started and execution is stopped at that location.

To start a debug session:

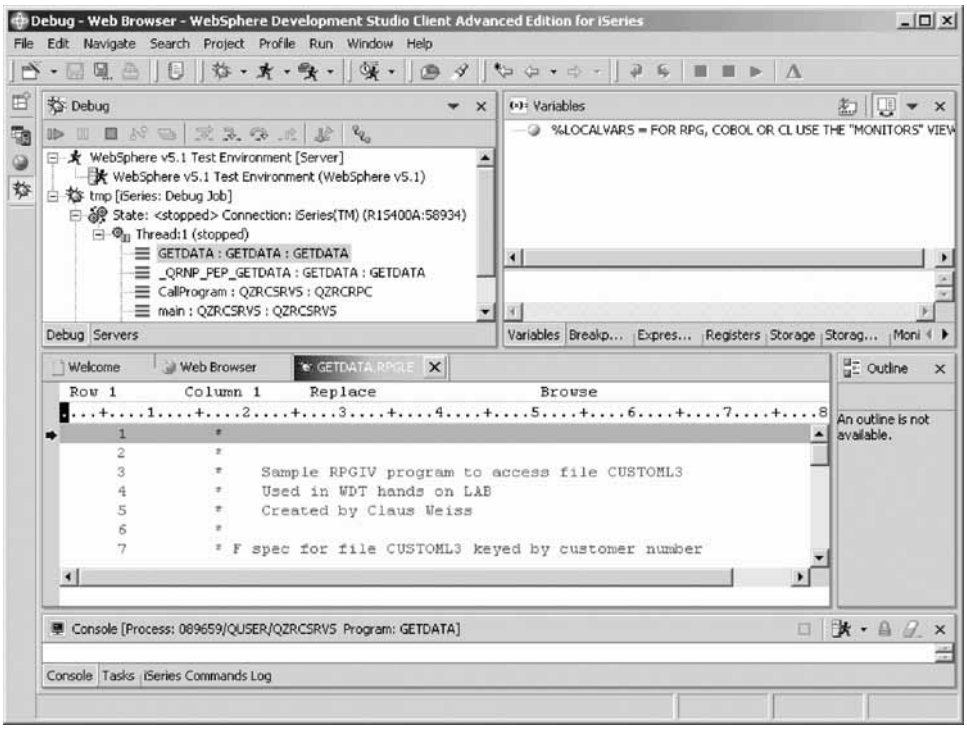
1. In the Remote System Explorer perspective, expand the iSeries server icon (s400a)
2. Expand **iSeries Objects**
3. Expand the **Library List**.

   If prompted, type your User-ID and password.
4. Expand the library WSSLABxx.

   You will see all objects in this library.

   Now you will start the debug session and set a service entry point.
5. Right-click **GETDATA.*pgm.rpgle**
6. Click **Service Entry Point > Set** on the pop-up menu.
7. Click the **Web Browser** tab to switch back to the Web browser.
8. Type `0010100` in the **Enter customer number** field.
9. Click the **Submit** button, which will invoke GETDATA*pgm.rpgle on the iSeries host.

   At this point, the Service Entry point is hit and the debug session is started.



Notice that it also opens an instance of the GETDATA source code in the debug editor. You now have the access to debug this source.

Let's look at some more debug features.

You have started a debug session with service entry points and now you are ready to begin "Exercise 7.3: Adding and deleting breakpoints."

Exercise 7.3: Adding and deleting breakpoints

Before you begin, you must complete "Exercise 7.2: Starting a Debug session using service entry points" on page 96.

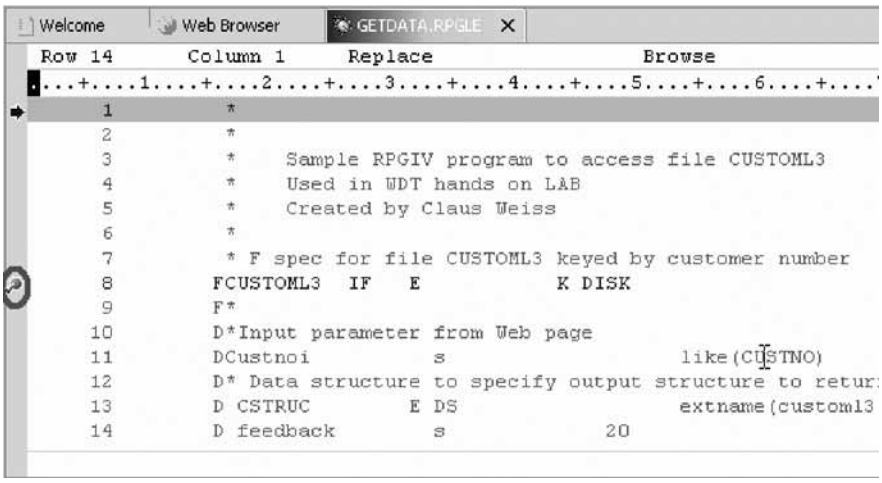
Breakpoints are markers you place in your program to tell the debugger to stop your program whenever execution reaches that point. For example, if you suspect that a particular statement in your program is causing problems, you could set a line breakpoint on the line containing the statement, then run your program. Execution stops at the breakpoint before the statement is executed. You can then check the contents of variables and view the call stack, and execute the statement to see how the problem arises.

You can only set breakpoints at executable lines. All executable lines are displayed in blue. The easiest way to set a breakpoint is to right-click on the line in the Source view.

To set a breakpoint:

1. Click anywhere on line 8.
2. Right-click and click **Add breakpoint** on the popup menu.

   A dot with a check mark in the prefix area indicates that a breakpoint has been set for that line. The prefix area is the small grey margin to the left of the source lines.



3. Click on line 31.
4. Right-click and click **Add breakpoint** on the pop-up menu.
5. Click the **Breakpoints** tab in the upper right pane of the workbench.
6. In the Breakpoints view right-click the breakpoint for line 8.

   Here you can manage your breakpoints. You can disable, or remove breakpoints accordingly.

You have set breakpoints and now you are ready to begin "Exercise 7.4: Displaying a variable."

Exercise 7.4: Displaying a variable

Before you begin, you must complete "Exercise 7.3: Adding and deleting breakpoints" on page 97.

You can monitor variables in the Monitors view. Now you will monitor the variable CUSTNOI.

To monitor a variable:

1. In the Source view double-click the variable CUSTNOI on line 22.
2. Right-click CUSTNOI.
3. Click **Monitor Expression** on the pop-up menu.

The variable appears in the **Monitors** view.

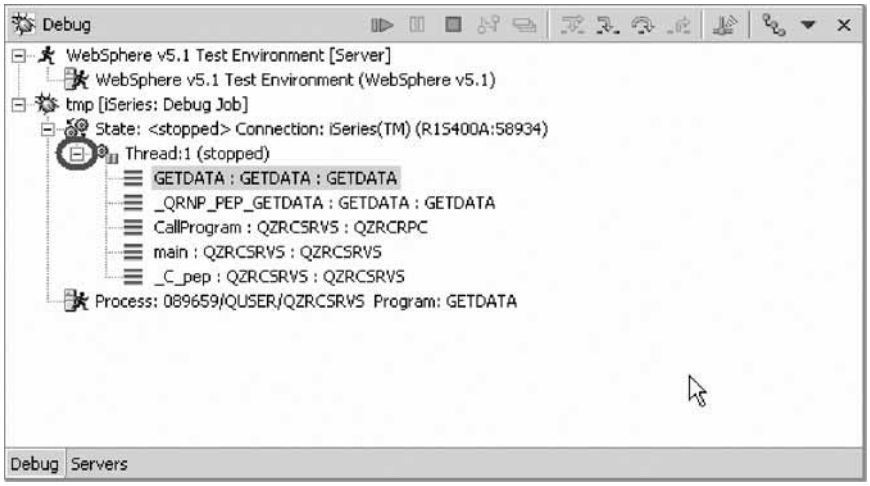
If you quickly want to see the value of a variable without adding it to the Monitor, you can place the cursor on a variable to see its value in a pop-up window. This works well for scalars, but not for arrays or data structures.

You have monitored an expression and now you are ready to begin "Exercise 7.5: Viewing the call stack."

Exercise 7.5: Viewing the call stack

Before you begin, you must complete "Exercise 7.4: Displaying a variable" on page 98.

The Debug view in the upper left pane, lists all call stack entries. It contains a tree view for each thread. The thread can be expanded to show every program, module, procedure and method that is on the stack at the current execution point. If you double-click a stack entry that has debug data, the corresponding source will show up. Otherwise the message No Debug data available appears in the debug editor. In the Debug view, expand the stack entry of **Thread:1** if it is not expanded already, by clicking the plus sign + in front of it.



You have expanded a thread to show every program, module, procedure and method on the call stack at the current execution point and now you are ready to begin "Exercise 7.6: Closing the debug session."

Exercise 7.6: Closing the debug session

Before you begin, you must complete "Exercise 7.5: Viewing the call stack."

To close a debug session:

1. Click **Resume** on the Debug toolbar.

   The session will stop at the first breakpoint on line 8.
2. Click **Resume** again.

   The session will stop at the breakpoint on line 31.
3. Click **Resume** again.
4. Click **OK** on the Program Terminated window.

   Next, close the Debug perspective.

5. Right-click the Debug icon on the left taskbar of the workbench.
6. Click **Close** on the pop-up menu.

7. Switch to the Remote System Explorer perspective to remove the Service Entry Point on GETDATA.*pgm.rpg.
8. Drill down to s400a > iSeries Objects > Library List > WSSLABXX > GETDATA.*pgm.rpg
9. Right-click on GETDATA.*pgm.rpg
10. Click **Service Entry Point > Remove** on the pop-up menu.

You have run the program to termination.

Module recap

You have completed Chapter 7, “Module 7. Debugging a Web application,” on page 95. You have learned how to:

- Recognize the features of the iSeries Integrated Debugger
- Start a debug session with service entry points
- Set breakpoints
- Monitor an expression
- Expand a thread to show every program, module, procedure and method on the call stack at the current execution point
- Run the program to termination

Now that you have debugged your Web application, you learn how to create an informative error page using Page Designer. Continue to Chapter 8, “Module 8. Adding an error page,” on page 101.

Chapter 8. Module 8. Adding an error page

This module teaches you how to create an informative error page using Page Designer for when a customer enters an incorrect customer number. This way, the customer won't just be given an error message, but will have a more informative and helpful response from the Web Application, which is easy to customize. You will also learn how to add the error page to your existing Web application using the Web Interaction wizard and to use the RSE perspective to modify the program or service program to call the error page. In this module, you will:

- Create an error page using Page Designer
- Change the Web interaction to use the new error page
- View the result of adding flow control to the Web application

Exercises

The exercises in this module must be completed in order.

- "Exercise 8.1: Creating the flow control page"
- "Exercise 8.2: Modifying your Web interaction" on page 103
- "Exercise 8.3: Testing the new error page" on page 110

Time required

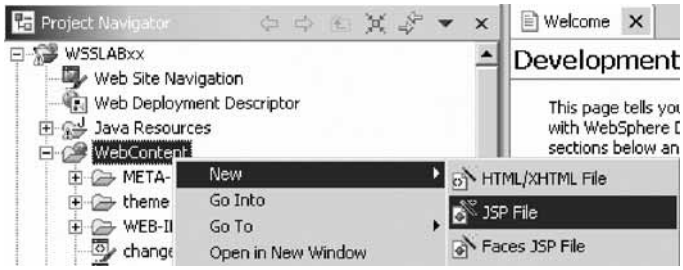
This module will take approximately **20 minutes** to complete.

Exercise 8.1: Creating the flow control page

First we need to create the error page using Page Designer.

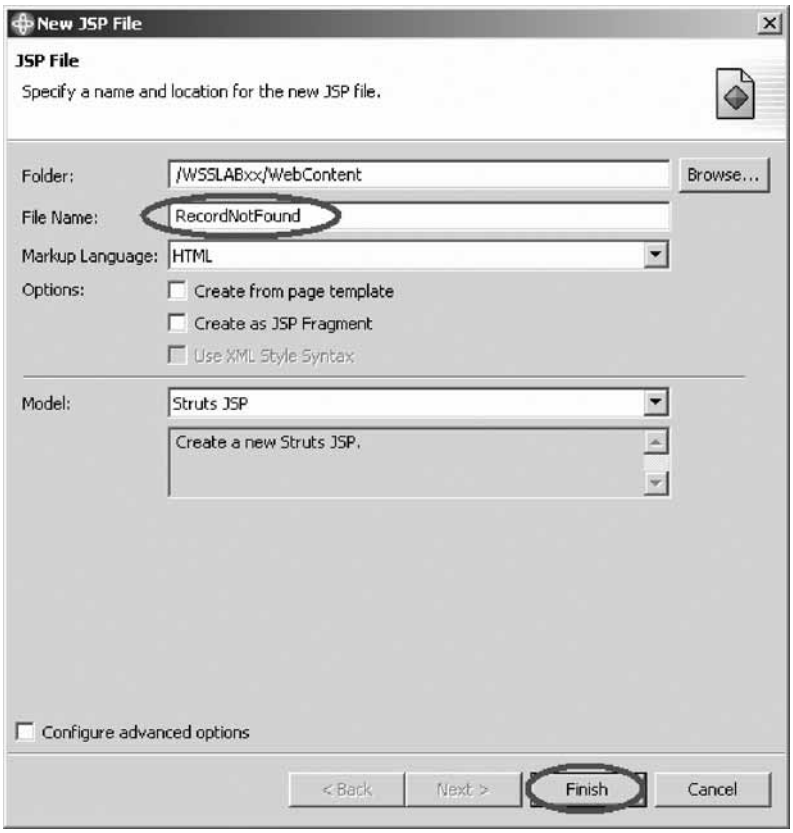
To create the error page:

1. In the Web Perspective, expand your project and drill down to the **Web Content** folder.



2. Right-click the **Web Content** folder and click **New > JSP** on the pop-up menu. The New JSP File page opens.

© Copyright IBM Corp. 2001, 2005



3. In the **File Name** field, type `RecordNotFound`.
4. Click **Finish**.

   The Page Designer automatically opens.

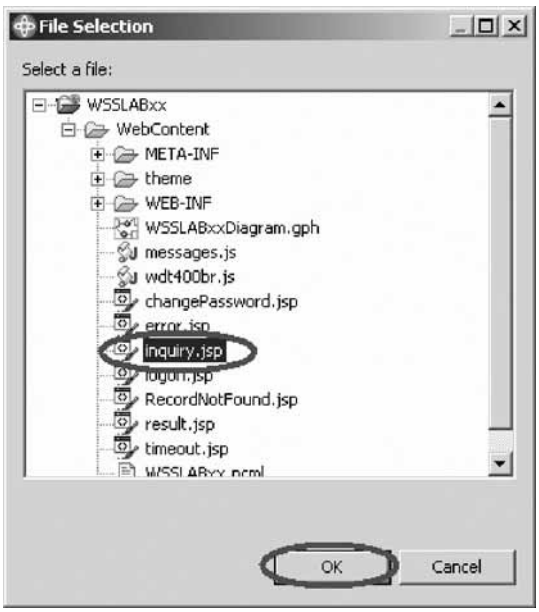
   Your window will look like this.

5. Click on the words **Place content here.** and change it to something like `Sorry, there is no record for that customer number.`
6. Press **Enter**.
7. Type `Please check the number and try again.`
8. Highlight the words `try again` and right-click.
9. Select **Insert Link** on the pop-up menu.

   The Insert Link window opens.

10. Click **Browse** next to the **URL** field.
11. Click **File** on the pop-up menu.

    The File Selection window opens.



12. Select **inquiry.jsp**.
13. Click **OK**.
14. Click **OK** in the Insert Link dialog.

    You have completed creating the flow control page.
15. Save and close the file.

    You could spend more time customizing this page, but for the purpose of this exercise you will leave the page as is.

You have created an error page using Page Designer and now you are ready to begin "Exercise 8.2: Modifying your Web interaction."

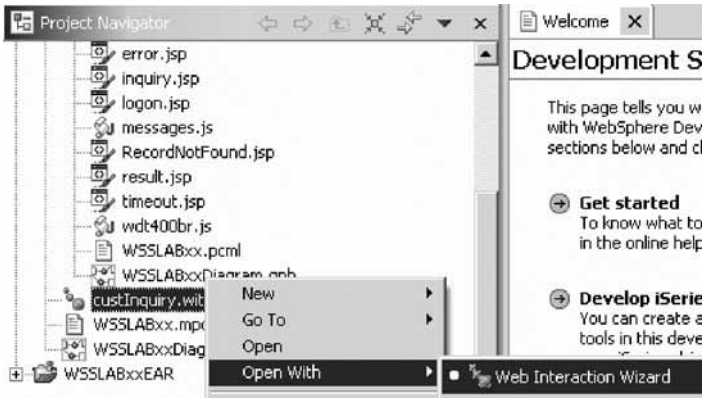
Exercise 8.2: Modifying your Web interaction

Before you begin, you must complete "Exercise 8.1: Creating the flow control page" on page 101.

Now you need to change the Web interaction to use the new error page.

To modify the Web interaction:

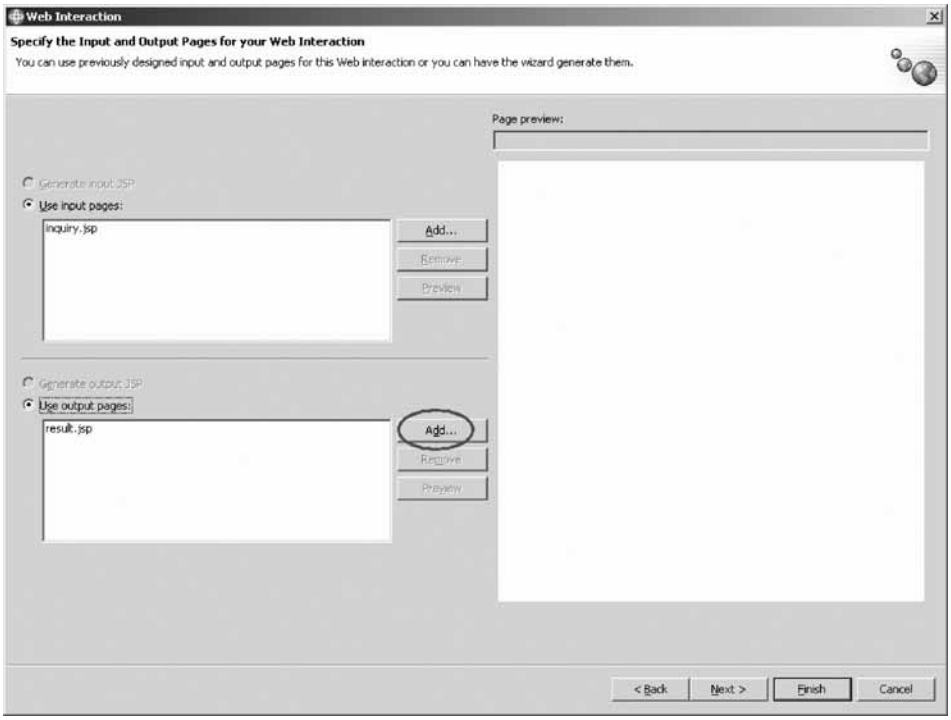
1. Expand your project **WSSLABxx** if not already expanded.
2. Right-click **custInquiry.wit** and click **Open With > Web Interaction Wizard** on the pop-up menu.



The Web Interaction wizard opens.

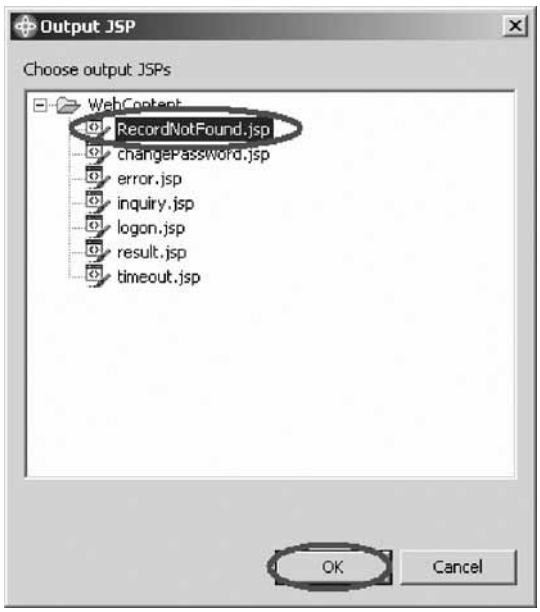
The name and location of your Web interaction are already specified.

3. Click **Next** to add a new output page.

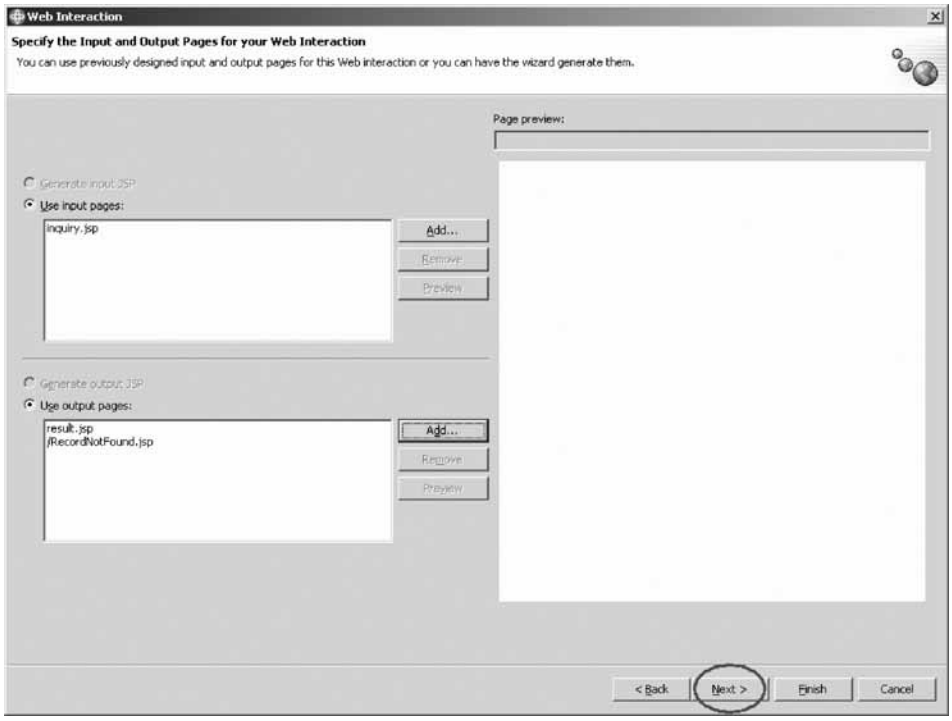


4. Next to the **result.jsp** section, click **Add**.

The Output JSP window opens.



5. Expand **Web Content**.
6. Select **RecordNotFound.jsp**.
7. Click **OK** to add the new output page.

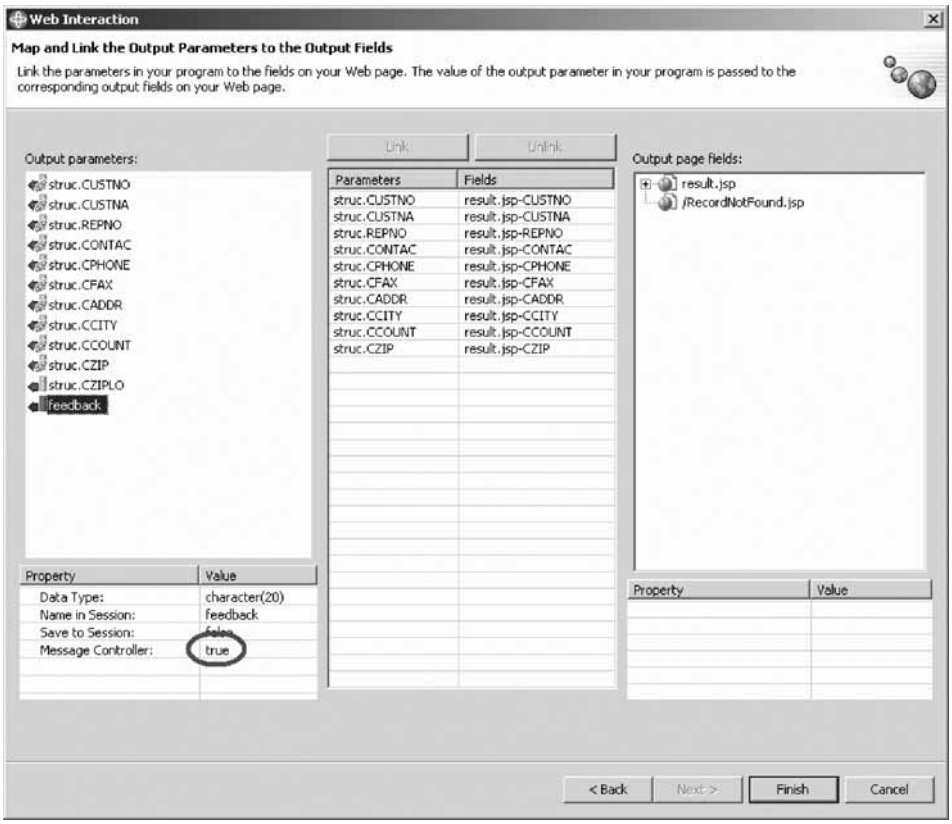


8. Click **Next** in the Web Interaction wizard.

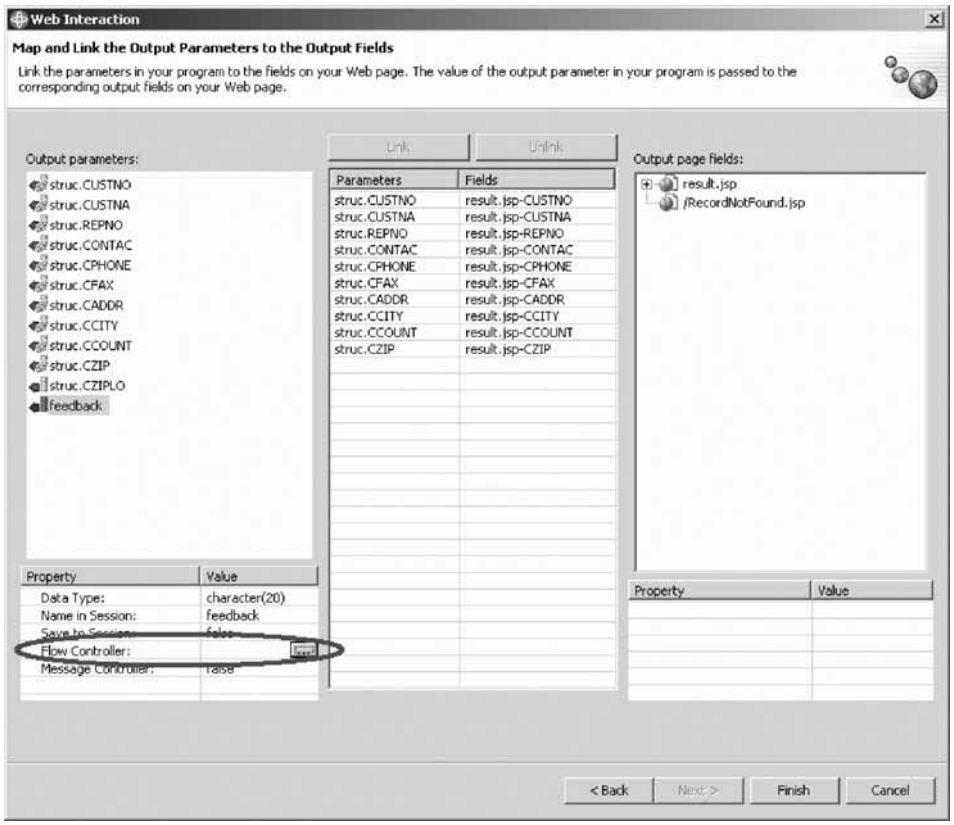
   You will re-use the Feedback parameter to control which page the Web Interaction will send next after it performs its action; the result.jsp or the RecordNotFound.jsp. So you don't have to add another parameter or change the parameters, you just change the values the feedback parameter returns depending on whether a customer record was found or not found.
9. Click **Next** since you don't need to define the program interface.

   This opens the Input Parameters page.

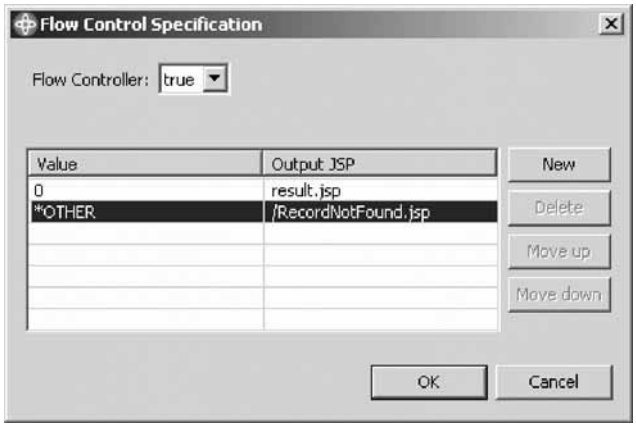
10. Click **Next** again since you don't need to change any input parameters. This opens the Output Parameters page.



11. Select feedback under the **Output Parameters** list.
12. Change the **Message Controller property** to **false**.
13. Select feedback under the **Output Parameters** list again.
    Now a **Flow controller property** is added to the Properties table.



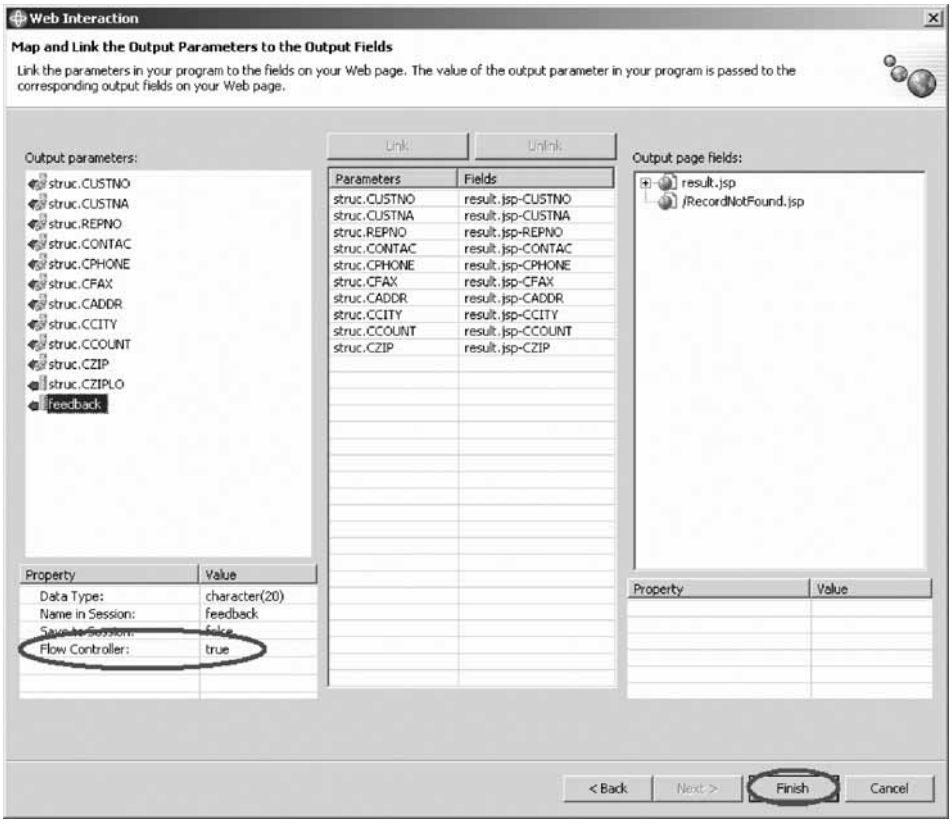
14. Change the **Flow Controller property** to **true**.
    The Flow Control Specification dialog opens.



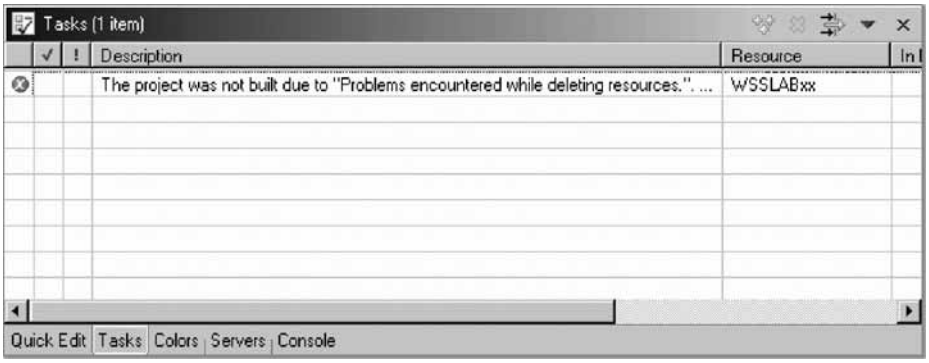
15. The value *OTHER is already in the **Value** field for the /error.jsp.
16. Click the **/error.jsp** field to change the Output JSP page.
17. Select **RecordNotFound.jsp** from the list.
18. Click **New**.
    This creates a new flow condition. The default page is result.jsp.
19. The value 0 is already in the **Value** field with **result.jsp** in the Output JSP field.
20. Click **OK**.
    This will set the flow control such that if the value of Feedback is 0, then the program will output the customer information to the result.jsp page and display it.

However, if the customer number doesn't exist in the database, the Feedback value will be something else other than 0, and instead of displaying a message near the input field, it will show the user the RecordNotFound.jsp page for a more helpful response.

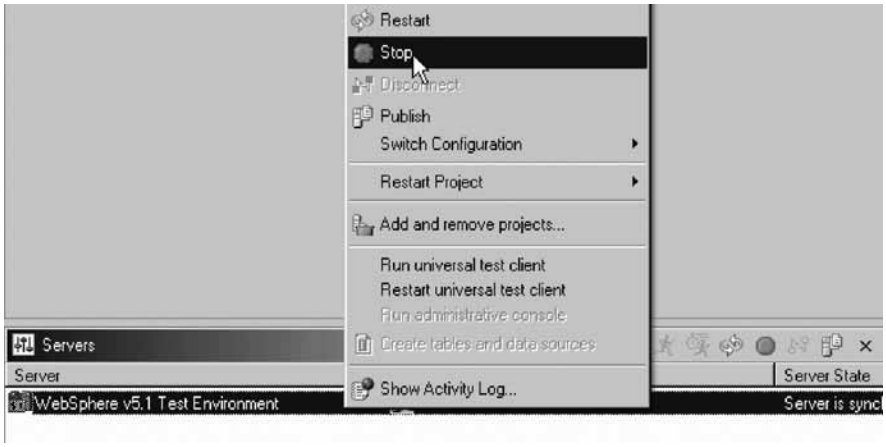
21. Click **Finish** on the Output Parameters page to apply your changes to the Web Interaction.



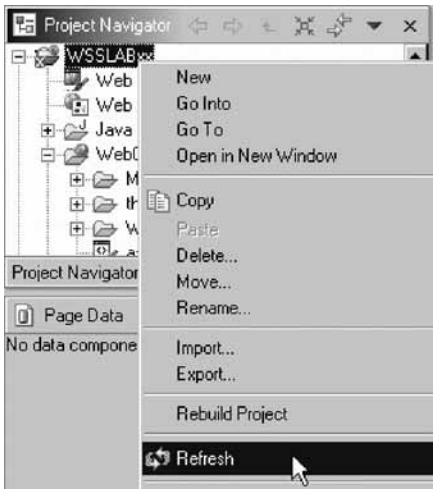
Note: If you encounter the following error message in your **Tasks** list:



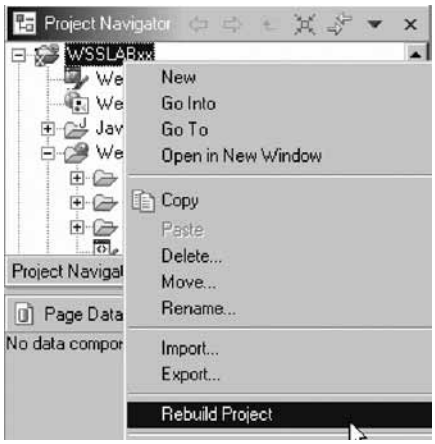
1. Click the **Server** tab to access the Server view.
2. Right-click the test server and select **Stop** on the pop-up menu.



3. Switch to your Project Navigator and right-click your project folder.
4. Click **Refresh** on the pop-up menu.



5. Right-click again your project folder and click **Rebuild Project**.



This should remove the error message in your Task view.

You have changed the Web interaction to use the new error page and now you are ready to begin "Exercise 8.3: Testing the new error page" on page 110.

Exercise 8.3: Testing the new error page

Before you begin, you must complete “Exercise 8.2: Modifying your Web interaction” on page 103.

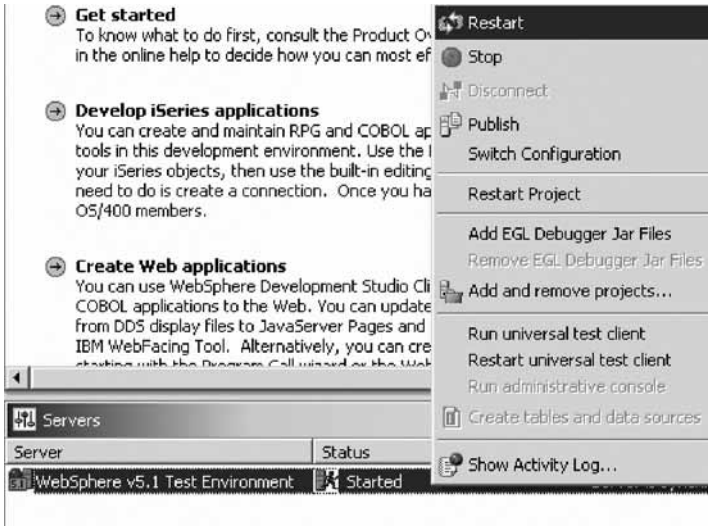
Now that the flow control is in place, if you enter a wrong customer number,

you will get the error page instead of a message.

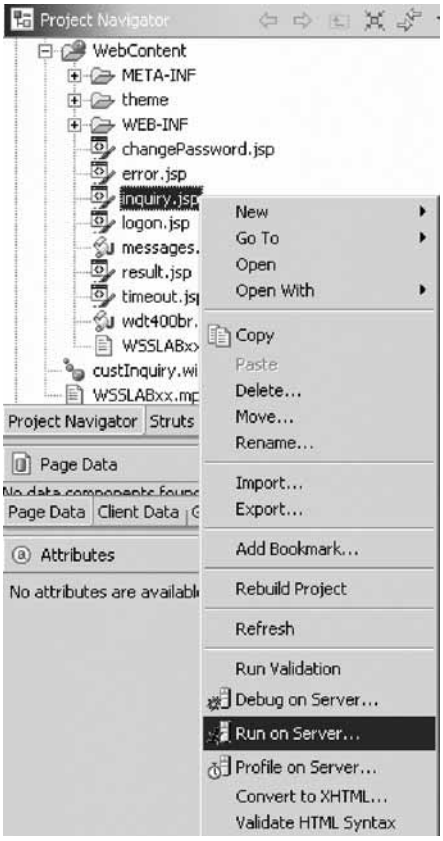
Now follow the instructions below to run the modified project on the WebSphere Test Environment to see the result of adding flow control to your Web application.

The server should still be running from the last time you ran the Web application. Every time you make a change to the Web application, you need to restart the server to pick up the changes.

To restart the server:



1. Click the **Servers** tab to open the Servers view.
2. Select the server instance. Right-click and click **Restart** on the pop-up menu. The server instance should restart.
3. In the Web perspective Project Navigator locate the **Web Content** folder, and right-click **inquiry.jsp**.



4. Click **Run on Server** on the pop-up menu.

You have viewed the results of adding flow control to the Web application.

Module recap

You have completed Chapter 8, “Module 8. Adding an error page,” on page 101. You have learned how to:

- Create an error page using Page Designer
- Change the Web interaction to use the new error page
- View the result of adding flow control to the Web application

Now that you have added an error page to your Web application to handle incorrect customer numbers, you can learn how to customize your input page. Continue to Chapter 9, “Module 9. Enhancing the input page using Web tools,” on page 113.

Chapter 9. Module 9. Enhancing the input page using Web tools

this module teaches you how to use the Web tools in the workbench to update the input page of your customer inquiry Web application. The input page named inquiry.jsp is one of the files created by the Interaction wizard. This page is very plain. You will learn how to use Page Designer and some related Web tools in the workbench to add some color to the input page, as well as add some pictures to make it a more interesting Web page.

In this module, you will:

- Locate the Web application input page and start Page Designer
- View the title of the input page
- Add a style to the input page
- Preview the new style for the input page
- Start WebArt Designer
- Create a logo
- Resize the logo
- Save the object as a WebArt object
- Save the object for a Web page
- Place the object on the Design page
- Add a company name to the input page
- Add an image to the input page
- Add a marquee to the input page
- Apply color to certain areas of text

Exercises

The exercises in this module must be completed in order

- "Exercise 9.1: Opening Page Designer" on page 114
- "Exercise 9.2: Working with page properties" on page 115
- "Exercise 9.3: Linking a cascading style sheet to the Web page" on page 116
- "Exercise 9.4: Designing and adding a logo" on page 118
- "Exercise 9.5: Adding a heading 1 tag to the page" on page 127
- "Exercise 9.6: Adding a picture to the page" on page 128
- "Exercise 9.7: Adding moving text to the page" on page 130
- "Exercise 9.8: Changing the text color" on page 134

Time required

This module will take approximately **30 minutes** to complete.

© Copyright IBM Corp. 2001, 2005

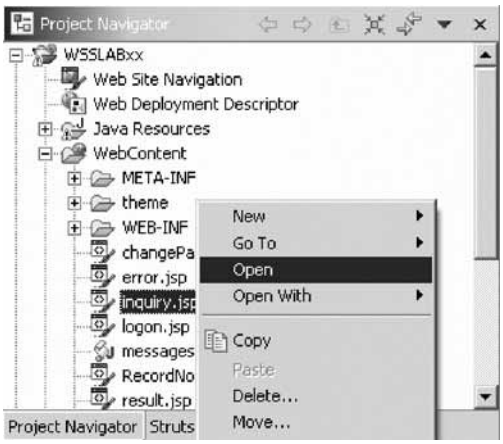
Exercise 9.1: Opening Page Designer

The first step is to locate the inquiry.jsp file and to start Page Designer. In Page Designer you will link the page to a cascading style sheet that comes as a sample with the workbench.

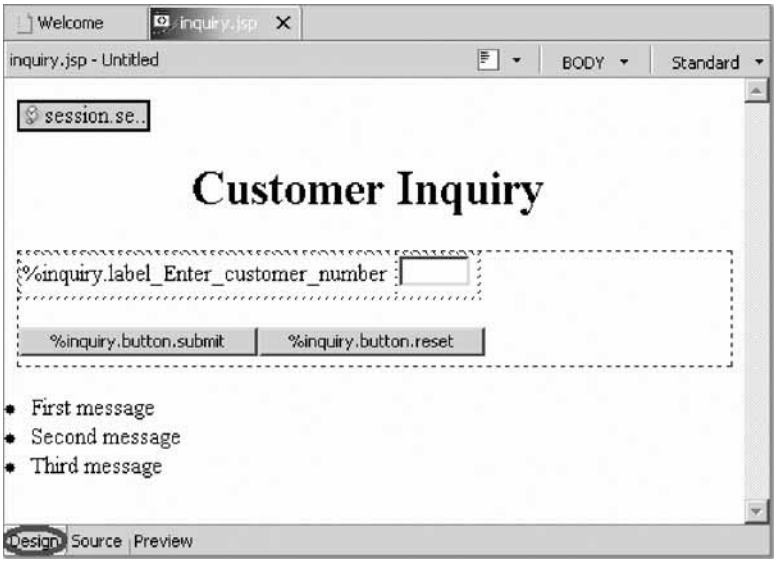
To open the Web perspective:

1. In the workbench, open the Web perspective:

   You should now have the Project Navigator in your workbench.



2. Expand the WSSLABxx web project.
3. Expand the **Web Content** folder.
4. Right-click inquiry.jsp and click **Open** on the pop-up menu.



   The Page Designer appears in the upper right pane of the workbench and shows the inquiry.jsp page as the Web Interaction wizard created it.

   Make sure that you are on the Design page in Page Designer.

5. Click the **Design** tab.

You have located the Web application input page and started Page Designer and now you are ready to begin “Exercise 9.2: Working with page properties” on page 115.

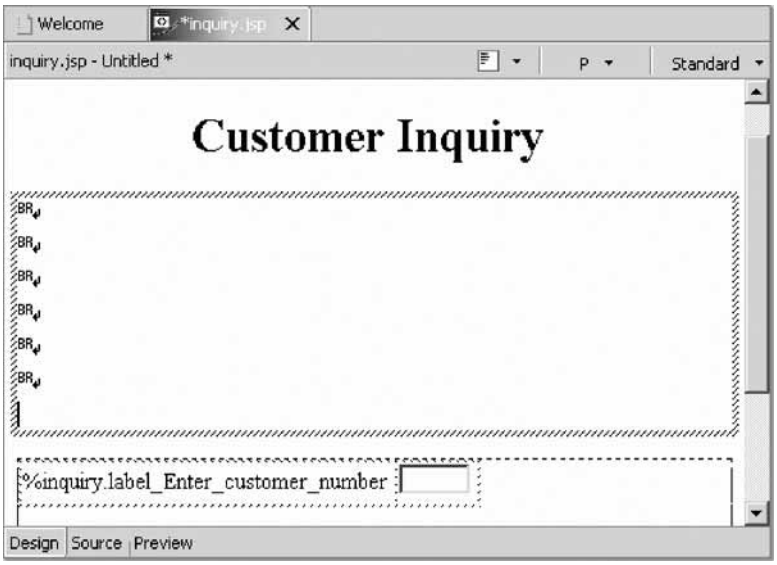
Exercise 9.2: Working with page properties

Before you begin, you must complete “Exercise 9.1: Opening Page Designer” on page 114.

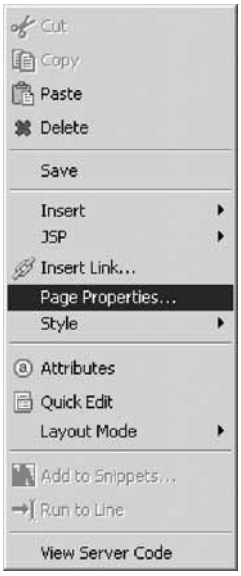
Next you move the form down.

To work with page properties:

1. Click underneath the heading **Customer Inquiry**.

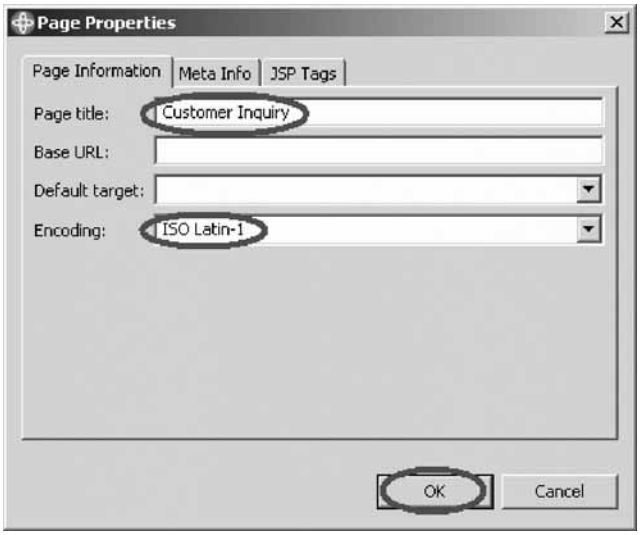


2. Press **Enter** number of times as shown above.
3. To change the page properties, right-click the background of the inquiry.jsp page in Page Designer.



4. Click **Page Properties** on the pop-up menu.

   The Page Properties window opens.



This window allows you to change some of the page properties.

For example you could change the title. When this page is shown in a browser, the Window title bar of the browser will display Customer Inquiry.

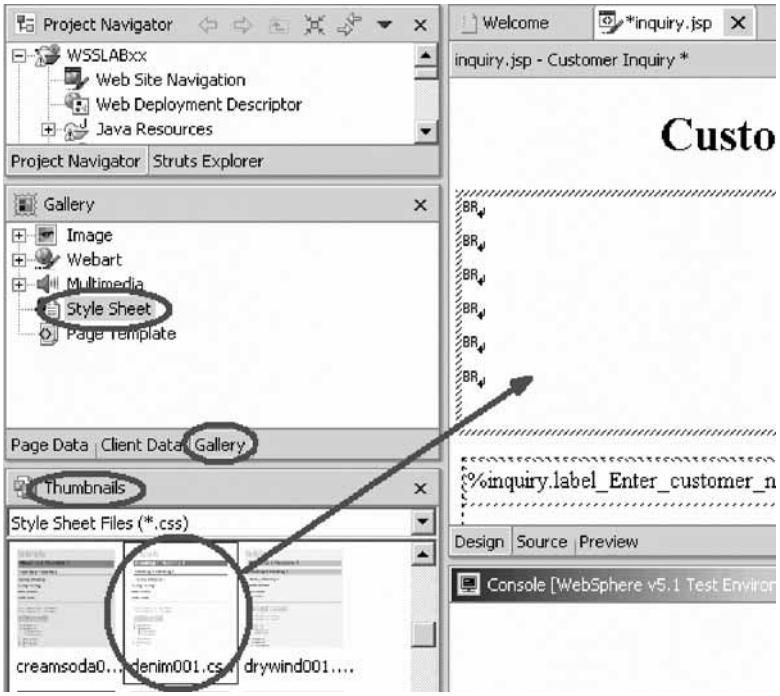
You have viewed the title of the input page and now you are ready to begin "Exercise 9.3: Linking a cascading style sheet to the Web page."

Exercise 9.3: Linking a cascading style sheet to the Web page

Before you begin, you must complete "Exercise 9.2: Working with page properties" on page 115.

Now you will add a style to the input page. You can use a style that is used in your company or you can use one of the sample style sheets that are provided in the product.

To link a style sheet to an input page:



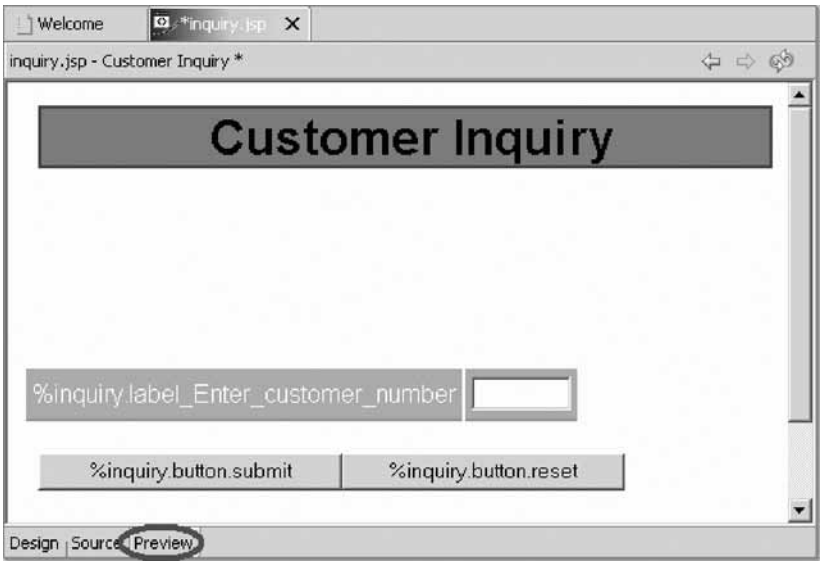
1. Click the **Gallery** tab, if the Gallery view isn't already displayed.
2. Click the **Style Sheet** icon in the Gallery list.
3. Click the **Thumbnail** tab in the left bottom stacked view in the workbench.

   You should see thumbnail icons of all the styles available as shown above.
4. In the thumbnail view, scroll down to the bottom, until you see style sheet **denim001.css** in the list, or select a style sheet that you like best.
5. Click the thumbnail picture of **denim001.css**.
6. Hold the left mouse button down and drag the mouse cursor to the Page Designer window.

   The cursor will change from this shape , to this shape . When the latter cursor shape appears in the Page Designer window then, let go of the mouse left button.

   After a short while the Style sheet properties will be applied.
7. Click the **Preview** tab.

   You will see the colors in the page have changed to the style sheet definitions.



8. Click the **Design** tab to get back to the Design page.

You have added a style to the input page and previewed the new style for the input page and now you are ready to begin "Exercise 9.4: Designing and adding a logo."

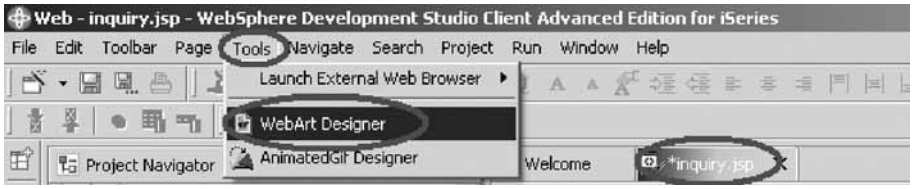
Exercise 9.4: Designing and adding a logo

Before you begin, you must complete "Exercise 9.3: Linking a cascading style sheet to the Web page" on page 116.

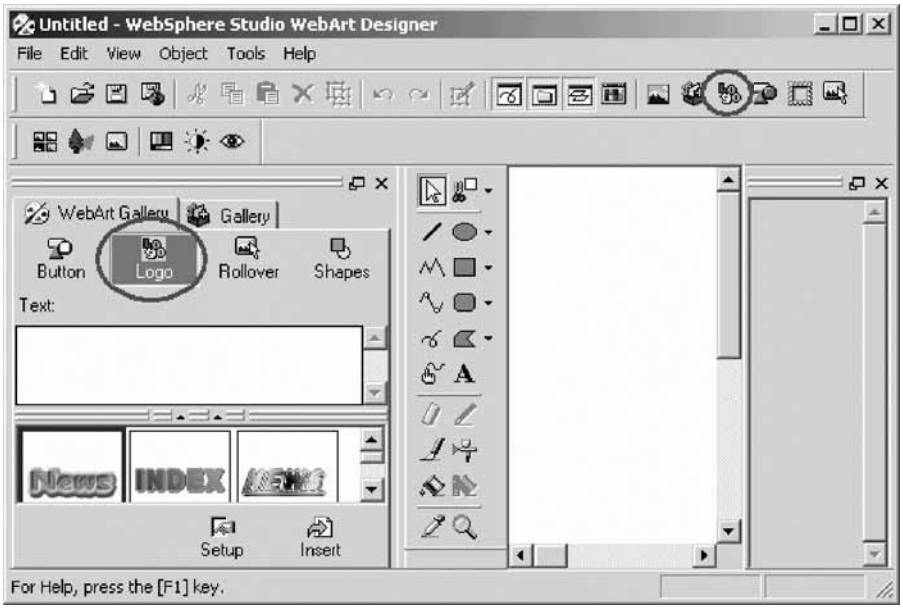
Now that you have the overall Web page look specified you will use the WebArt Designer to create a Logo that you then will add to this page.

Starting the WebArt Designer

To start the WebArt Designer:



1. Make sure the Page Designer is in focus.
2. Click **Tools > WebArt Designer** in the workbench menu.
   The WebArt designer opens.



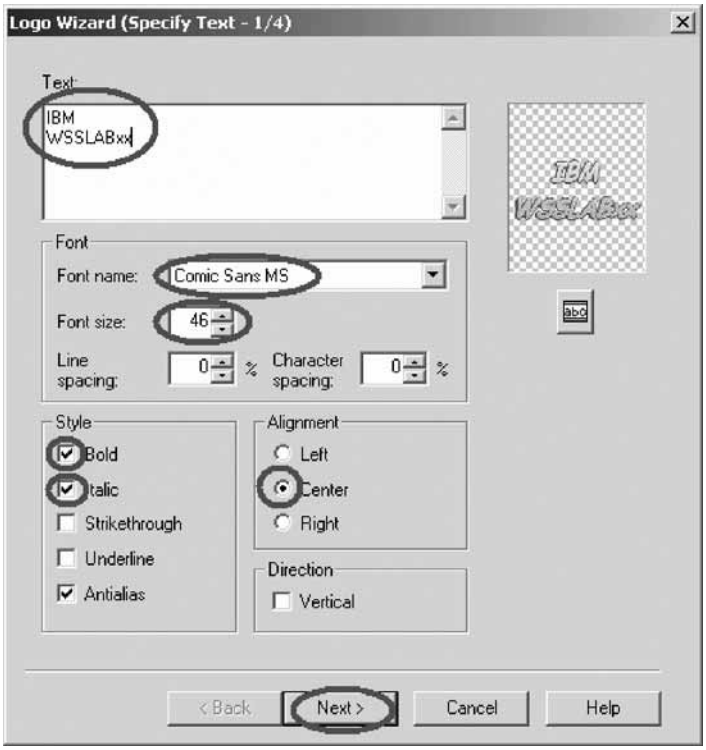
The WebArt Designer shows the Template Gallery on the left, where you find samples of logos, buttons, rollovers, images and more. The large white area in middle of the dialog is the canvas that is used to work with objects that you want to create or change. Now you will create a logo from scratch.

You could also select one from the template gallery as the base for your own logo.

Creating the logo

To create the logo:

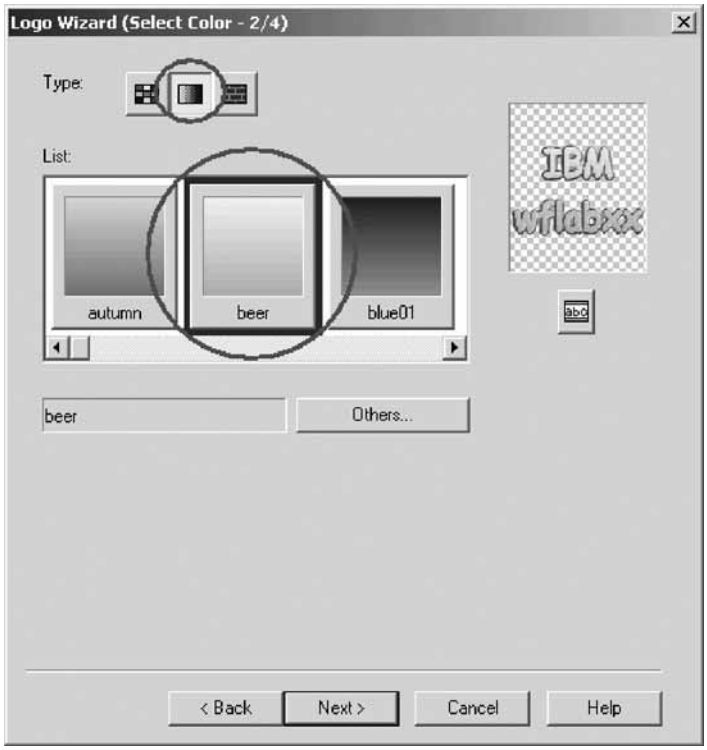
1. Click the **Create Logo** button above the canvas or click **Object > Create Logo**.
   The Logo Wizard opens.



2. In the **Text** field, type your company name and in the next line in the **Text** field, type WSSLABxx.
3. In the **Font name** list, select **Comic Sans MS**.
4. In the **Font size** list, select **46**.
5. Under **Alignment**, select the **Center radio** button.

   Notice in the upper right corner of the dialog, a sample of the logo as specified is displayed.
6. Click **Next** to go to the next page of the wizard.

   The Select Color page opens.



7. Select the **gradation type** button; the middle one of the three type push buttons.

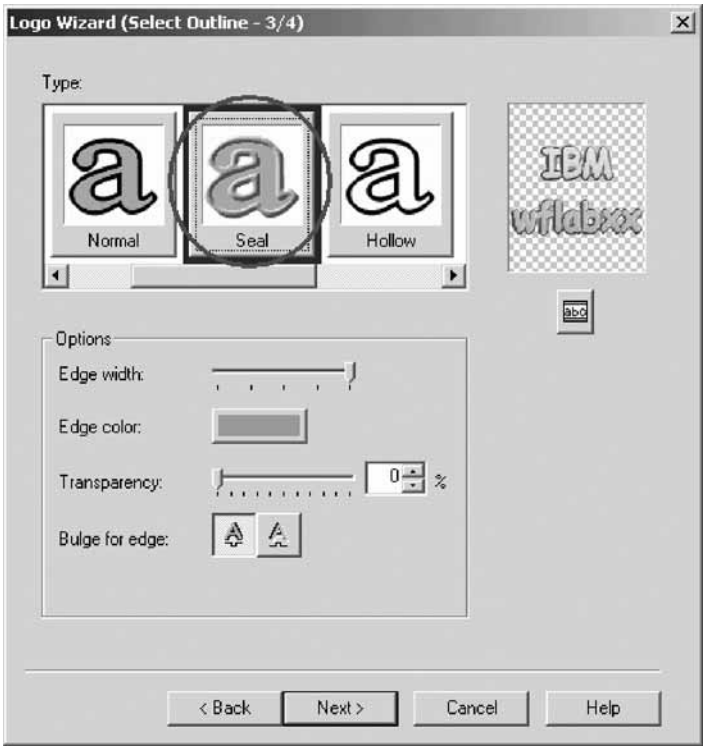
   Tip: The other buttons select color types: solid and textured.

8. Select beer from the colors available, or any other color you like best, just scroll through the list to find a gradation you like.

   Tip: You can change the colors by clicking the Others button on this dialog and create you own gradation.

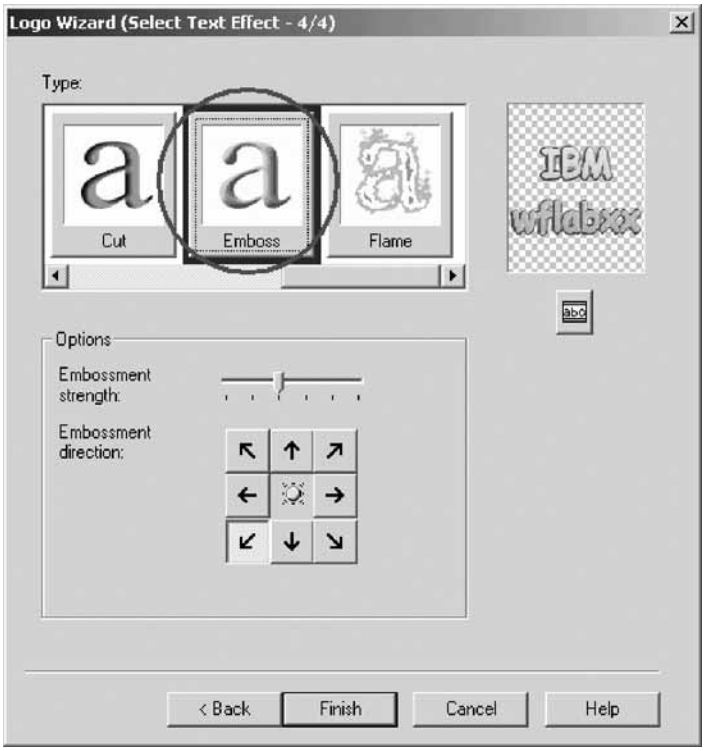
9. Click **Next** to go to the next page of the wizard.

   The Select Outline page opens.



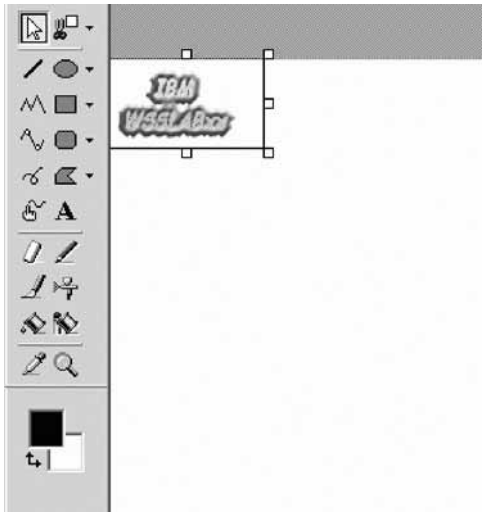
10. Select the **Seal outline** from the list, or any outline you like best.
11. Click **Next**.

    The Select Text Effect page opens.



12. Select the **Emboss text** effect, or one that you like best.
13. Click **Finish**.

    You return to the WebArt designer window.



Resizing the logo

To resize the logo object on the canvas:

1. Click the logo object to select it.
2. Move the cursor to the rectangle at the right bottom corner of the object; watch the cursor changing shape.
3. Drag the rectangle up and to the left so the object becomes smaller.

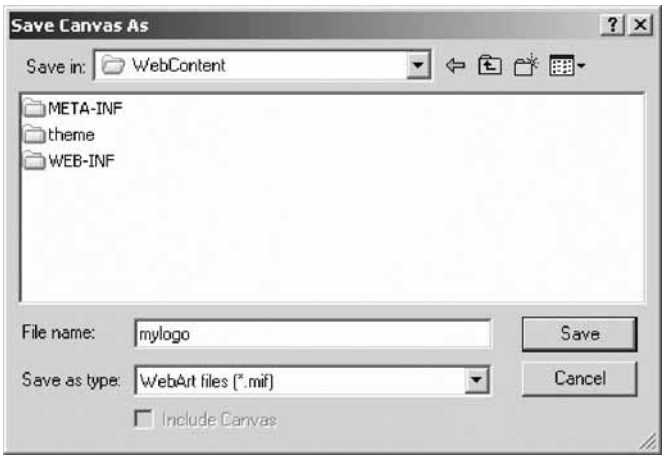
Saving the object as a WebArt object

Now you need to save this object. First save it as a WebArt object. This allows you later on to work with the object again in WebArt Designer, but you can't use that format for your Web page.

To save this object:

1. Click **File > Save Canvas As** on the WebArt Designer menu.

   The Save Canvas As window opens. Make sure you see the WebContent folder in the Save Canvas As window.



2. In the **File name** field, type `mylogo`.
3. Click **Save**.

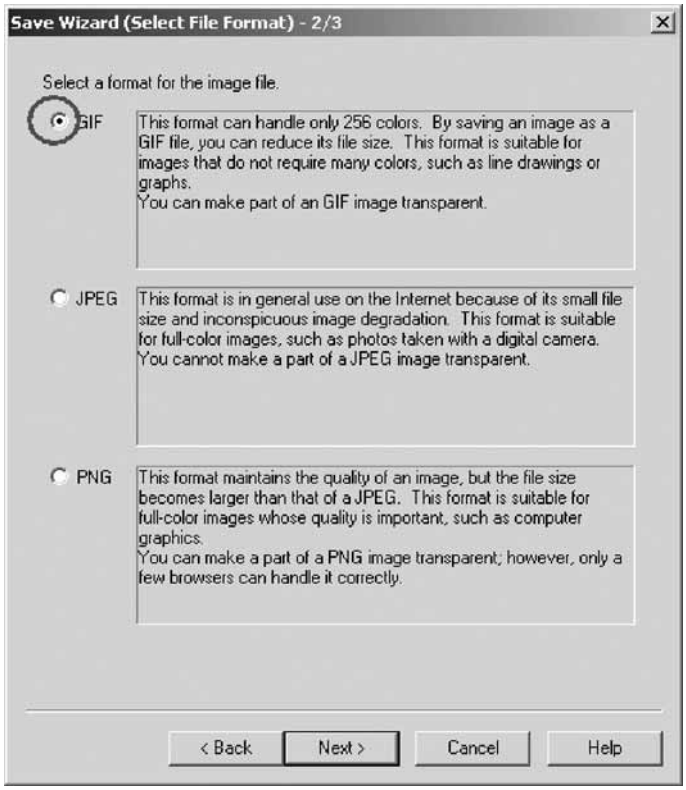
Now you need to save the object in a form that can be displayed on a Web page.

Saving the object for a Web page

To save the logo for a Web page:

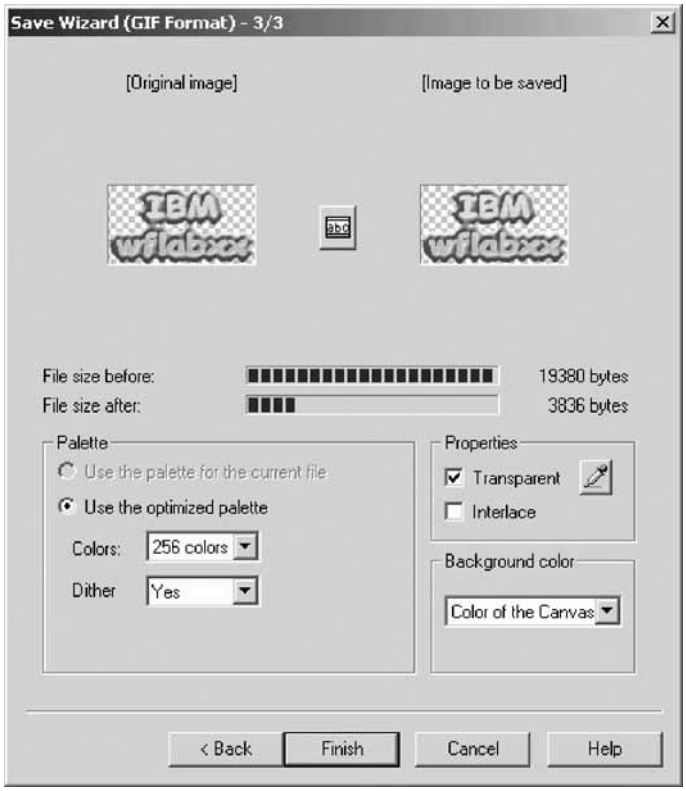
1. Click the logo object on the canvas.
2. Click **File > Save wizard for Web** from the WebArt Designer menu.
   The Save Wizard opens.

3. Click the **Save the selected object** radio button.
4. Click **Next**.
   The Select File Format page opens.



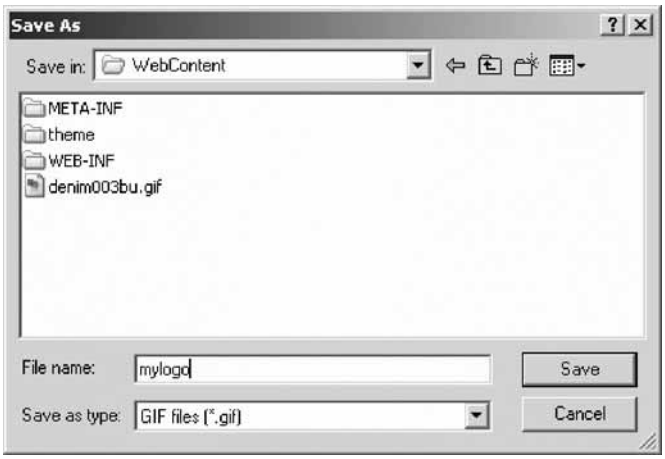
5. Click the **GIF** radio button.
6. Click **Next**.

   The GIF Format page opens.



7. Click **Finish**.

The Save As window opens.



Make sure the directory is pointing to the Web Content directory in the workspace where your Web project is located.

8. In the **File name** field, type `mylogo`.
9. Click **Save**.

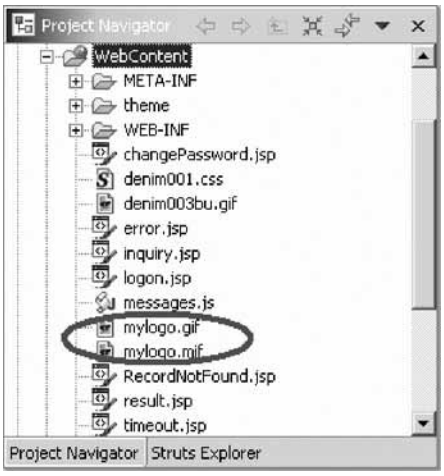
   Close the WebArt designer.
10. Select **File > Exit** from the menu.

    You return to the workbench in the Web perspective and see the Project Navigator.

Placing the logo on the Design page

To place the logo on the Design page:

1. Make sure you switch from the Gallery view to the Project Navigator.
2. Expand the WSSLABxx Web project.
3. Expand the **Web Content** folder.



The mylogo.gif file should appear in the list as shown below. If it doesn't appear, the list might need to be refreshed.

4. Right-click the Web Project WSSLABxx folder icon.
5. Select **Refresh**.

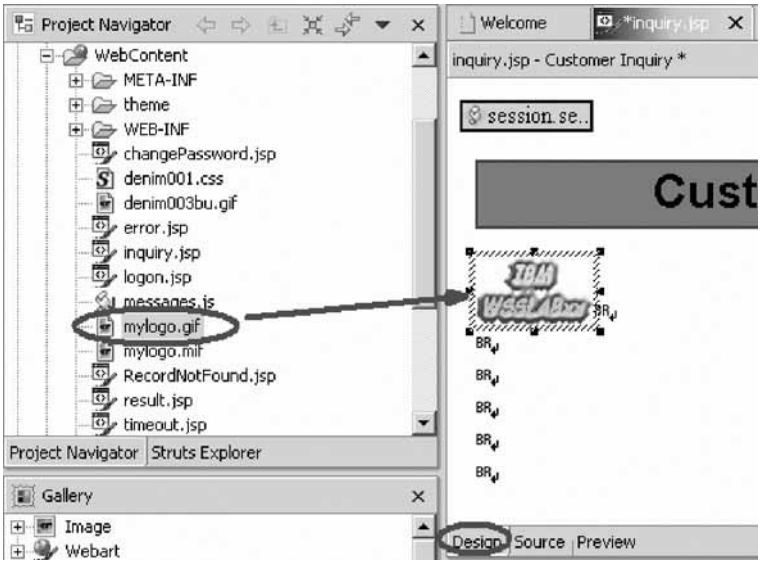
   Hopefully you can see the file now in the Web Content folder, if not, go to Windows Explorer and search for the mylogo.gif on your hard drive. Move it

to the Web Content folder in the product workspace. If you didn't use the default location as shown above, do a search for the workspace and the WSSLABxx directory in it. Move the mylogo.gif file into the Web Content sub directory under the WSSLABxx directory. Now you can take the logo and put it on your Web page that is still open in Page Designer. If Page Designer has been closed just open the inquiry.jsp file.

Note: Make sure your are on the Design page, not the Preview page.

6. In the Project Navigator view select the **mylogo.gif** file.
7. Hold the left mouse button down.
8. Drag the file to the upper left side in the Page Designer window.
9. Release the mouse button.

   The logo is placed on the Design page.



You have started WebArt Designer, created a logo, resized a logo, saved the object as a WebArt object, saved the logo for a Web page, placed the object on the Design page and now you are ready to begin "Exercise 9.5: Adding a heading 1 tag to the page."

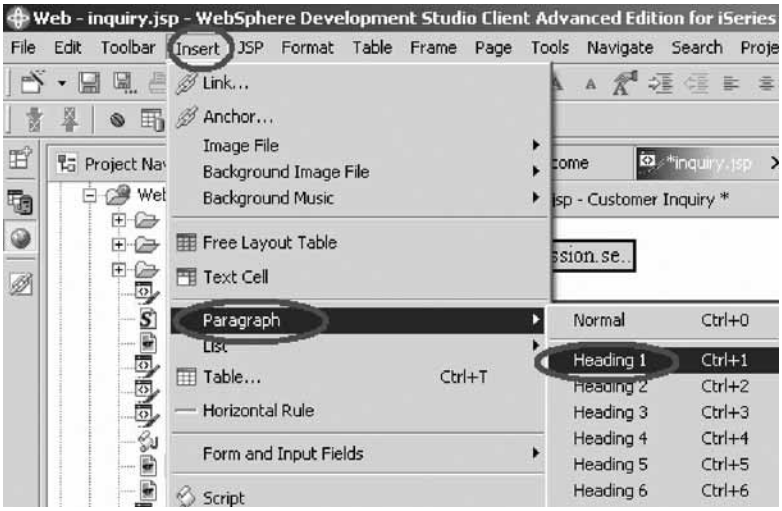
Exercise 9.5: Adding a heading 1 tag to the page

Before you begin, you must complete "Exercise 9.4: Designing and adding a logo" on page 118.

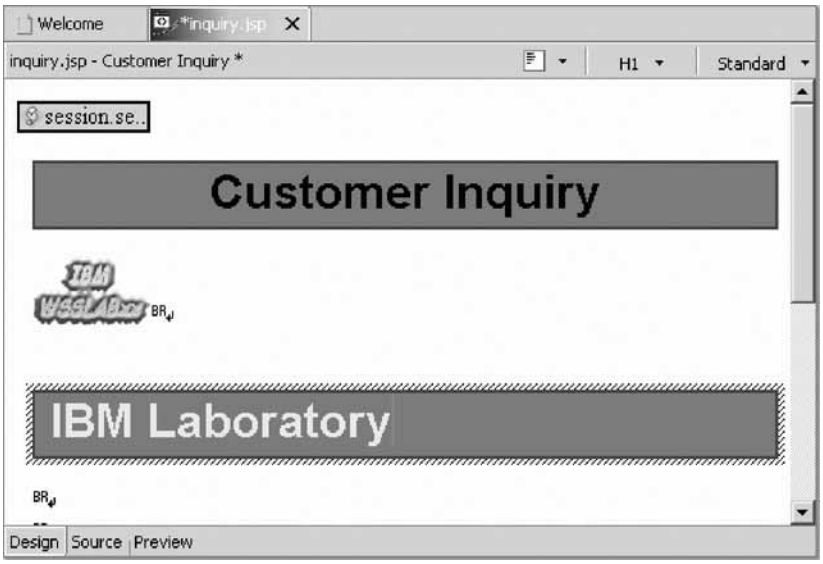
Now you want to insert a heading below the logo.

To add a heading 1 tag:

1. Position the cursor just below the logo at the first BR tag.



2. Click **Insert** from the workbench menu.
3. Click **Paragraph > Heading 1** from the pop-up menu.
   A frame appears that allows you to enter text.
4. Enter your company name.



Now you want to use one of the sample pictures that comes with the product and place this picture on the page.

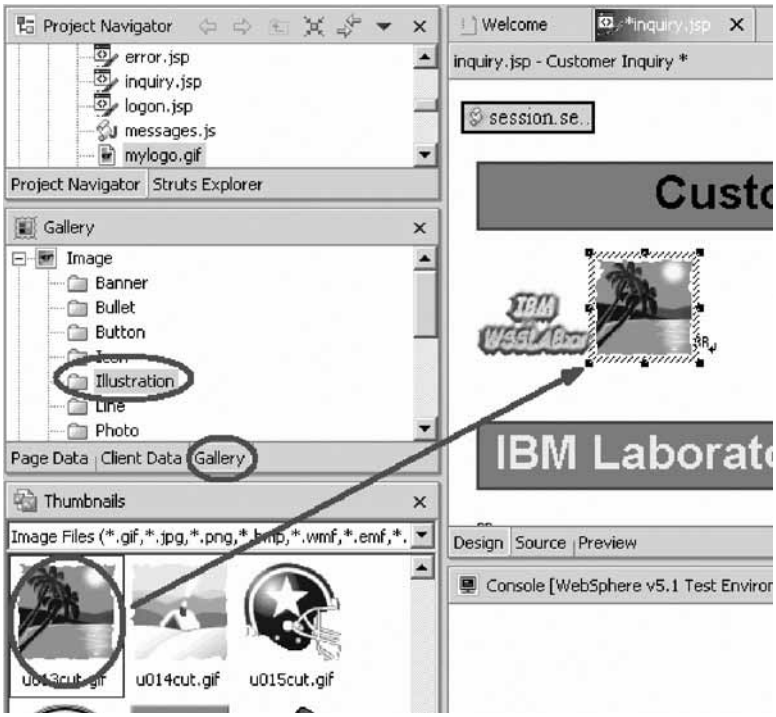
You have added a company name to the input page and now you are ready to begin "Exercise 9.6: Adding a picture to the page."

Exercise 9.6: Adding a picture to the page

Before you begin, you must complete "Exercise 9.5: Adding a heading 1 tag to the page" on page 127.

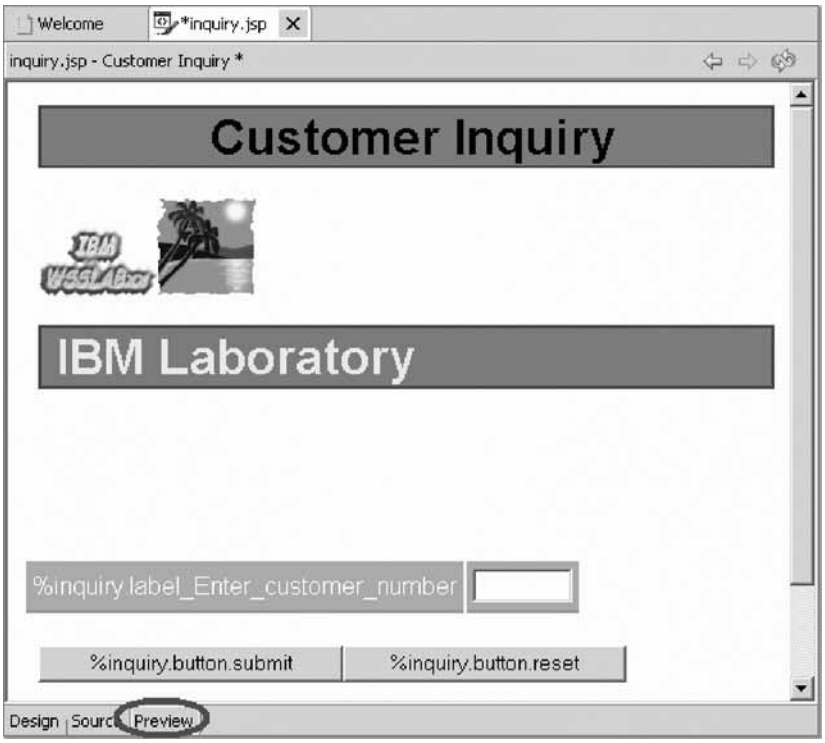
To add a picture to the page:

1. In the Project Navigator, click the **Gallery** tab.



2. Expand the **Image** folder.
3. Select the **Illustration** folder.
4. On the Design page beside the Gallery view select one of the sample illustrations, for example, use file u013cut.gif.
5. Drag the picture onto the Design page beside the logo.

   The Design page now contains a picture.



You are almost done. Next you add moving text to the page.

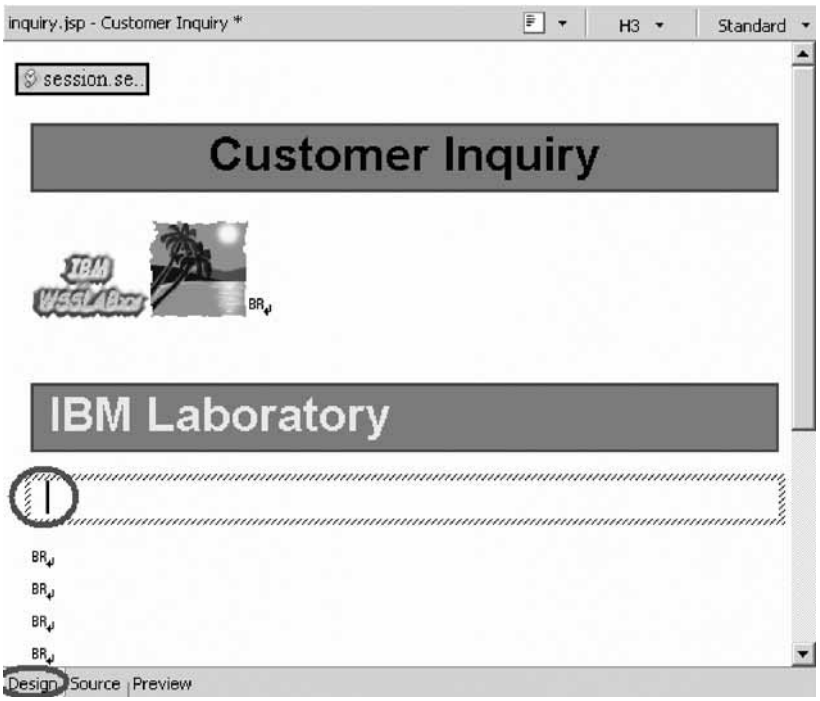
You have added an image to the input page and now you are ready to begin "Exercise 9.7: Adding moving text to the page."

Exercise 9.7: Adding moving text to the page

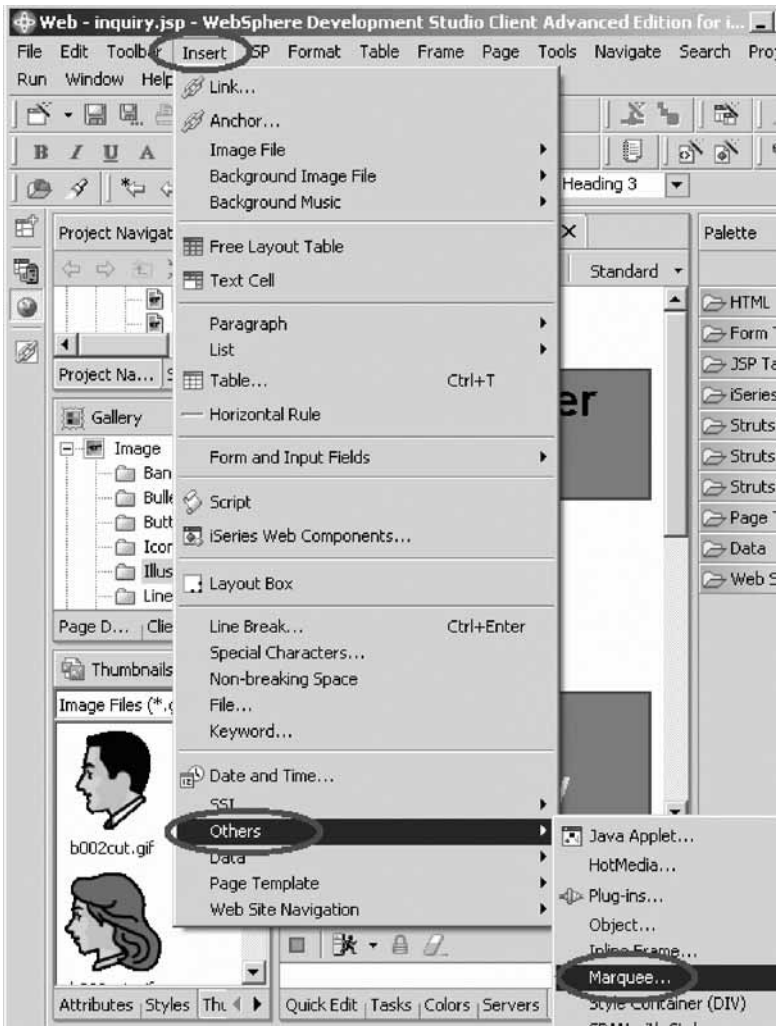
Before you begin, you must complete "Exercise 9.6: Adding a picture to the page" on page 128.

To add moving text:

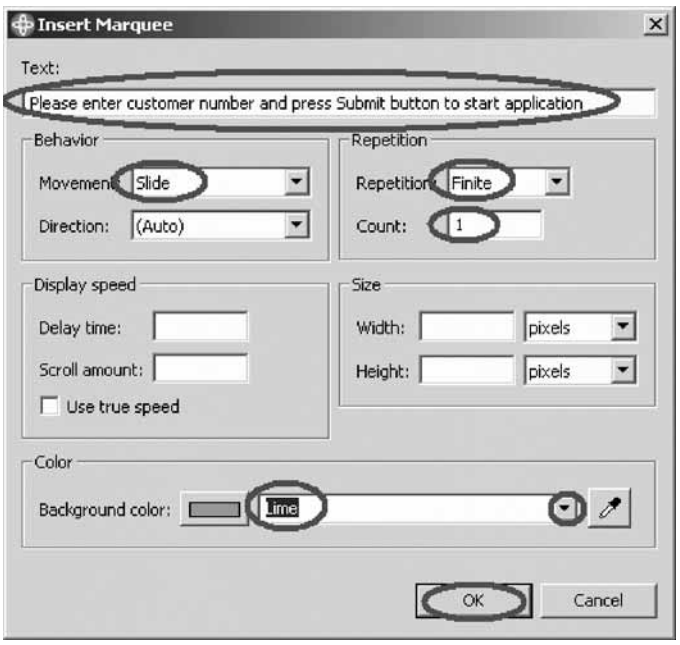
1. Back to the Design page, position the cursor underneath your company name.
2. Click **Insert** from the workbench menu.
3. Click **Paragraph > Heading 3** on the pop-up menu.



4. Leave the cursor positioned inside the heading 3 frame.
5. Click **Insert** from the workbench menu.



6. Click **Others > Marquee** on the pop-up menu
   The Insert Marquee window opens.



7. Enter into the **Text** field `Please enter customer number and press Submit button to start application.`
8. Select **Slide** from the **Movement** list.
9. Select **Finite** from the **Repetition** list.
10. Select **1** from the **Count** list.

    The two last selections just avoid the text sliding in forever and not standing still. If you want more movement on the page, you can change these settings.
11. Select **Lime** in the **Background color** list.
12. Click **OK**.

    The Design page should look like:

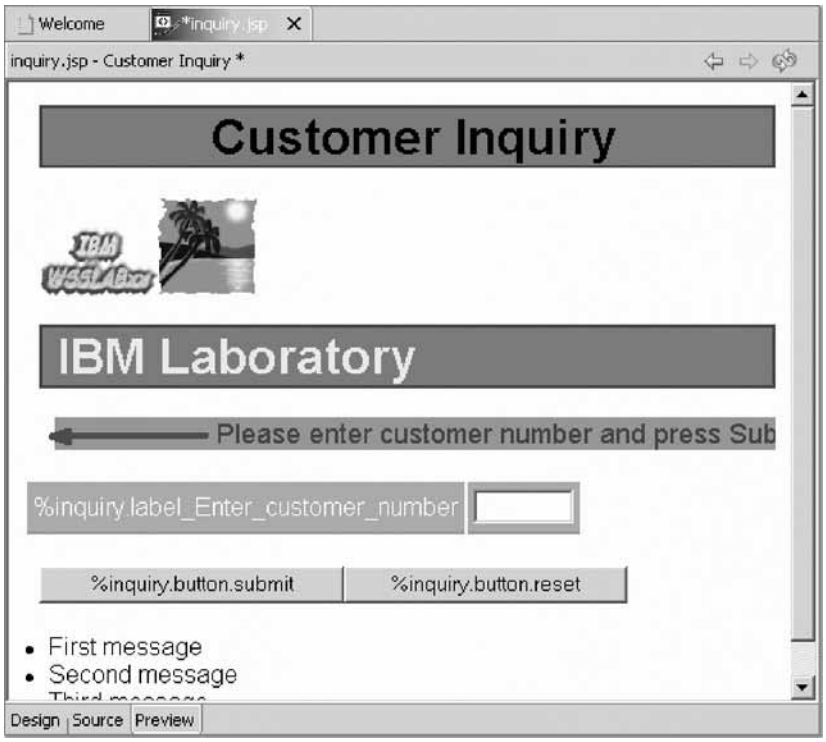
    To save some space, you can remove some of the line break tags.
13. Position the cursor on the **BR** tag.

14. Press the **Delete** key until the entry field and push buttons appear on your page:

    Next you view the page as it would appear in a browser.

15. Click the **Preview** tab at the bottom of the Design page.

    You will notice that your heading 3 text is sliding in.



You have added a marquee to the input page and now you are ready to begin "Exercise 9.8: Changing the text color."

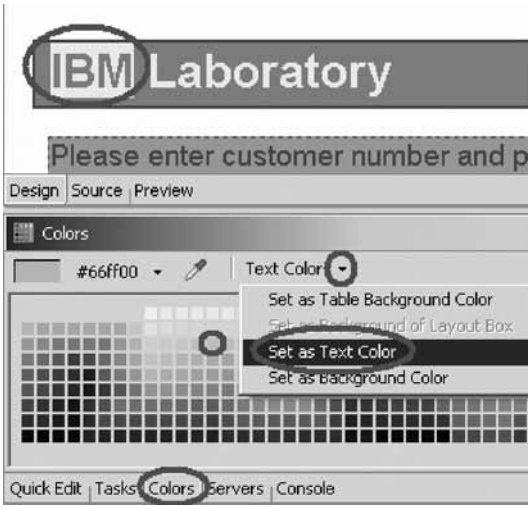
Exercise 9.8: Changing the text color

Before you begin, you must complete "Exercise 9.7: Adding moving text to the page" on page 130.

Sometimes you want to change the text color. There is an easy way to apply another color to certain areas of text. To do that you return to the Design page.

To change the text color:

1. Click the **Design** tab.



2. Select the company text (IBM) you want to change color on, by swiping the text area with the mouse cursor.

   Below the Page Designer window there are several tabs.
3. Click the **Colors** tab.
4. Select a color from the Color palette.
5. Click **Text color** from the Color menu.
6. Click **Set as Text Color** on the pop-up menu.

   You are done; now the selected text will be displayed with the color you selected for it.
7. Click the **Preview** tab to view your completed page.
8. Save the file inquiry.jsp.

9. Exit Page Designer.
10. In the Project Navigator expand the WSSLABxx folder.
11. Expand the **Web content** folder.
12. Right-click inquiry.jsp.
13. Click **Run on server** on the pop-up menu.
    Your new designed input page opens and you can run your Web application.

You have applied color to certain areas of text on the input page.

Module recap

You have completed Chapter 9, “Module 9. Enhancing the input page using Web tools,” on page 113. You have learned how to:

- Locate the Web application input page and start Page Designer
- View the title of the input page

- Add a style to the input page
- Preview the new style for the input page
- Start WebArt Designer
- Create a logo
- Resize the logo
- Save the object as a WebArt object
- Save the object for a Web page
- Place the object on the Design page
- Add a company name to the input page
- Add an image to the input page
- Add a marquee to the input page
- Apply color to certain areas of text

Finish your tutorial by reviewing the materials in Chapter 10, “Summary,” on page 137.

Chapter 10. Summary

In this tutorial you learned how to create a simple e-business customer inquiry application that used a Web-based front end to communicate with the business logic written in ILE RPG residing on an iSeries server. While creating a browser user interface, you used the iSeries Web Interaction wizard to generate input and output JSP files as well as Page Designer to enhance the input and output Web pages. You also added iSeries Web components to your pages, for example, Web equivalents of iSeries command keys, input fields that accept only particular types of data, or output fields such as subfile names. You then ran the application in the WebSphere Test Environment that is part of the product.

Completed learning objectives

If you have completed all of the modules, you should now be able to:

- Use the Remote System Explorer perspective to edit and compile an iSeries program or service program
- Use the Web perspective tools and views to create a Web project for your Web application
- Create a Web interaction to use an input page and an output page and to create a servlet to invoke an RPG program or service program to get data from the iSeries
- Run the Web application in the WebSphere Test Environment
- Use service entry breakpoints to debug your Web application
- Create an informative error page for customers when an incorrect customer number is entered
- Add color, pictures, change text to make the input page more attractive

More information

For more information on the product and the iSeries Web Tools, see http://ibm.com/software/adwtools/iseries.

© Copyright IBM Corp. 2001, 2005

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

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

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 106-0032, 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 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.

© Copyright IBM Corp. 2001, 2005

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

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 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:

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

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 1992, 2004. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

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

IBM
iSeries
OS/400
RPG/400
VisualAge
WebSphere

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.

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.

UNIX® is a registered trademark of The Open Group.

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

Printed in USA