WebSphere Development Studio Client for iSeries
Bringing New Life into 5250 Applications

Learn a new way to build applications for the IBM iSeries Server

Bring your 5250 applications to the Web with the IBM WebFacing Tool

Catch up with the latest trends in application development

Aleksandr Nartovich
Wilfried Blankertz
Camille Paquay
Pentti Rasanen
Claus Weiss

ibm.com/redbooks
Take Note! Before using this information and the product it supports, be sure to read the general information in “Notices” on page vii.
Contents

Notices .......................................................... vii
Trademarks ....................................................... viii

Preface ............................................................. ix
The team that wrote this redbook ................................ ix
Become a published author ........................................ xi
Comments welcome ............................................... xi

Part 1. Navigating the WebSphere maze ........................................ 1

Chapter 1. What is under the WebSphere umbrella ............................... 3
1.1 Deploying the Web for e-business ..................................... 4
   1.1.1 What is a Web server ........................................ 4
1.2 Adding dynamic content to Web sites .................................. 5
   1.2.1 Non-Java-based applications and database serving standards .... 5
   1.2.2 Java-based technologies ..................................... 9
1.3 Developing and deploying Web applications ............................ 18
   1.3.1 What is a Web application ................................... 18
   1.3.2 The purpose and use of the WebSphere Application Server ....... 19
1.4 iSeries: A flexible application server for the Web ......................... 22
1.5 The IBM suite of host/server access products ............................ 24
   1.5.1 Methods to combine new Web applications with existing applications 25
1.6 Web-to-host integration tools comparison ................................ 29
   1.6.1 Host On-Demand ............................................ 29
   1.6.2 WebFacing Tool ............................................ 31
   1.6.3 Host Publisher ............................................. 32
   1.6.4 Web Interaction wizard .................................... 35
   1.6.5 Comparison table .......................................... 36

Chapter 2. Creating applications for iSeries: A new set of tools ............ 39
2.1 IBM WebSphere Development Studio for iSeries .......................... 40
   2.1.1 ILE RPG .................................................. 41
   2.1.2 ILE C/C++ ................................................ 42
   2.1.3 ILE COBOL ............................................... 42
   2.1.4 Application Development ToolSet (ADTS) ....................... 42
2.2 WebSphere Development Studio Client for iSeries ......................... 43
   2.2.1 WebSphere Studio family .................................... 43
   2.2.2 A new concept ............................................. 44
   2.2.3 Follow-on product for WebSphere Development Tools ............. 47
   2.2.4 WebFacing Tool ............................................ 49
   2.2.5 Web Tools - iSeries Extensions ................................ 49
   2.2.6 Java Tools - iSeries Extensions ................................ 51
   2.2.7 CoOperative Development Environment (CODE) .................... 52
   2.2.8 VisualAge RPG ............................................ 52
2.3 More information ............................................... 53

Chapter 3. New GUI framework ...................................................... 55
3.1 Eclipse: The start of a new era of tools .................................. 56
3.2 A new approach to the application development process ................... 56
3.3 Components of the Eclipse GUI framework ........................................... 57
  3.3.1 Perspectives ................................................................. 57
  3.3.2 Views .................................................................. 59
  3.3.3 Editors .................................................................. 59
3.4 Development Studio Client features .................................................. 60
  3.4.1 Common features ......................................................... 60
  3.4.2 iSeries-specific features ................................................ 62
3.5 Online help .................................................................. 62

Part 2. Installation and configuration ...................................................... 65

Chapter 4. Setting up the iSeries environment ........................................ 67
  4.1 Overview of the required software components .................................. 68
  4.2 Configuring IBM WebSphere Application Server for iSeries .............. 69
    4.2.1 Installing and configuring the WebSphere Application Server instance ... 70
    4.2.2 Starting the WebSphere Application Server .................................... 71
    4.2.3 Creating and starting additional administrative servers ................... 73
  4.3 Installing and starting the IBM WebSphere Administrative Console on the PC client ............................................................. 75
    4.3.1 Installing the WebSphere Administrative Console ......................... 76
    4.3.2 Starting the WebSphere Administrative Console ............................ 78
    4.3.3 Working with the WebSphere Administrative Server installation .......... 80
  4.4 Setting up and configuring the IBM HTTP Server for iSeries .............. 82
    4.4.1 Starting an administrative server instance ...................................... 82
    4.4.2 Creating a new HTTP server ....................................................... 86
    4.4.3 Starting the HTTP Server (powered by Apache) instance ................. 93
  4.5 Verifying the installation ................................................................ 95
  4.6 Starting WebFacing jobs .................................................................. 96

Chapter 5. Setting up the client environment ........................................... 99
  5.1 Installing WebSphere Application Server Administrative Console ........ 100
  5.2 Installing WebSphere Development Studio Client for iSeries ............. 100

Part 3. Working with iSeries applications on a workstation ...................... 103

Chapter 6. Introduction to the sample applications .................................. 105
  6.1 Overview of the Order Entry application ........................................... 106
    6.1.1 The ABC Company .......................................................... 106
    6.1.2 The ABC Company database ................................................ 106
    6.1.3 Application files ................................................................ 106
    6.1.4 Database table structure ....................................................... 107
    6.1.5 Customer transaction flow ...................................................... 107
  6.2 Overview of the Customer Master Inquiry ......................................... 111
    6.2.1 Starting the application .......................................................... 111

Chapter 7. iSeries extensions (plug-ins) in Development Studio Client ........ 113
  7.1 Introduction to using Remote Systems Explorer .................................. 114
  7.2 Perspectives .................................................................. 114
  7.3 Profiles .................................................................. 116
  7.4 Connections ................................................................ 117
  7.5 Subsystems and filters .............................................................. 118
    7.5.1 Subsystems ................................................................ 118
    7.5.2 Filters .................................................................. 120

Chapter 8. Working with CODE ............................................................. 127
  8.1 What is code ................................................................ 128
# Part 4. Bringing your existing applications to the Web

<table>
<thead>
<tr>
<th>Chapter 9. Introduction to VisualAge RPG</th>
<th>157</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1 VisualAge RPG features</td>
<td>158</td>
</tr>
<tr>
<td>9.2 VisualAge RPG Language Reference.</td>
<td>158</td>
</tr>
<tr>
<td>9.3 Programming with VARPG</td>
<td>159</td>
</tr>
<tr>
<td>9.3.1 Starting the VARPG Designer.</td>
<td>161</td>
</tr>
<tr>
<td>9.3.2 The VARPG GUI Designer</td>
<td>161</td>
</tr>
<tr>
<td>9.4 Creating a simple GUI subfile application</td>
<td>164</td>
</tr>
<tr>
<td>9.5 Additional information</td>
<td>175</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 10. WebFacing Tool</th>
<th>179</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1 Sample application</td>
<td>180</td>
</tr>
<tr>
<td>10.1.1 Using the WebFacing Tool to convert DDS</td>
<td>180</td>
</tr>
<tr>
<td>10.1.2 Testing your application inside Development Studio Client for iSeries</td>
<td>187</td>
</tr>
<tr>
<td>10.1.3 Creating your own styles.</td>
<td>190</td>
</tr>
<tr>
<td>10.1.4 The user ID and password the application runs under.</td>
<td>195</td>
</tr>
<tr>
<td>10.1.5 Exporting a WebFacing project.</td>
<td>196</td>
</tr>
<tr>
<td>10.1.6 Deploying a Web application with WebSphere Administrative Console</td>
<td>197</td>
</tr>
<tr>
<td>10.1.7 Launching the converted application in a browser</td>
<td>200</td>
</tr>
<tr>
<td>10.1.8 Creating a file share in the iSeries integrated file system (IFS)</td>
<td>203</td>
</tr>
<tr>
<td>10.2 Security considerations</td>
<td>205</td>
</tr>
<tr>
<td>10.3 Hints and tips for working with the WebFacing Tool</td>
<td>205</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 11. iSeries Web development tools in Development Studio Client</th>
<th>207</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1 Introduction to design-time control</td>
<td>208</td>
</tr>
<tr>
<td>11.2 Developing the Web application</td>
<td>209</td>
</tr>
<tr>
<td>11.2.1 Using design-time control</td>
<td>209</td>
</tr>
<tr>
<td>11.2.2 Using the Web Interaction wizard</td>
<td>227</td>
</tr>
<tr>
<td>11.2.3 Testing application in Development Studio Client.</td>
<td>239</td>
</tr>
<tr>
<td>11.2.4 Debugging a Web application with a host program call</td>
<td>245</td>
</tr>
<tr>
<td>11.2.5 Exporting the Web application to production.</td>
<td>247</td>
</tr>
<tr>
<td>11.2.6 Subfile DTC APIs</td>
<td>249</td>
</tr>
<tr>
<td>11.2.7 Example code</td>
<td>251</td>
</tr>
<tr>
<td>11.3 Program Call beans</td>
<td>257</td>
</tr>
<tr>
<td>11.3.1 Creating Program Call beans</td>
<td>257</td>
</tr>
<tr>
<td>11.3.2 Calling an iSeries program or procedure</td>
<td>261</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 12. XML is everywhere</th>
<th>263</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1 Why XML.</td>
<td>264</td>
</tr>
</tbody>
</table>
Notices

This information was developed for products and services offered in the U.S.A.

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

IBM may have patents or pending patent applications covering subject matter 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.

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.

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

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

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

- Redbooks®
- AIX®
- AS/400®
- AS/400e™
- Balance®
- COBOL/400®
- DB2®
- DB2 Universal Database™
- e Strategy™
- IBM®
- IBM.COM™
- IBM eServer®
- Infoprint®
- Integrated Language Environment®
- iSeries™
- Language Environment®
- Net.Data®
- OS/2®
- OS/390®
- OS/400®
- Perform™
- PowerPC®
- pSeries™
- RPG/400®
- S/390®
- SecureWay®
- SP™
- SP1®
- System/38™
- VisualAge®
- WebSphere®
- xSeries™
- zSeries™

The following terms are trademarks of International Business Machines Corporation and Lotus Development Corporation in the United States, other countries, or both:

- Lotus®
- Domino™
- Word Pro®

The following terms are trademarks of other companies:

- ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

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

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

- C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

- UNIX is a registered trademark of The Open Group in the United States and other countries.

- SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

Web applications emerged quickly with the advent of the Internet and growth in the number of Internet users. These applications offer easy access from anywhere in the world since they run on a server and use a Web browser as the client.

However, acceptance of the Web application model has been hindered by the lack of support for Web-enabling existing applications. IBM iSeries and AS/400 customers understand this well. For years, they created their own applications that provided adequate support for their business processes. There was no reason to invest in new applications. That is until they saw the appeal of the universal accessibility and automation of business processes (including B2B capabilities).

Whether you want to write your own Web applications or move existing applications to the Web, IBM lets you take advantage of both worlds with a number of host access products. The newest addition to this product family is the IBM WebFacing Tool, which is packaged as part of the new WebSphere Development Studio Client for iSeries (Development Studio Client). The WebFacing Tool converts existing Data Description Specifications (DDS) into Java artifacts that are deployed in an application server, such as WebSphere Application Server. With the WebFacing Tool, existing iSeries applications require minor or no modifications and are accessible from the Web and iSeries display devices.

This IBM Redbook discusses WebSphere Development Studio Client for iSeries, the WebFacing Tool, and several more powerful tools. This includes:

- **WebSphere Studio Site Developer Advanced**: This integrated development environment (IDE) for developing applications offers the unique ability to add new features in the form of plug-ins. Anyone can develop a new plug-in and install it into WebSphere Studio Site Developer, while maintaining a simple infrastructure. WebSphere Studio Site Developer includes the development environment for creating plug-ins.

- **Cooperative Development Environment (CODE)**: This workstation-based tool supports application development in many different host languages, including RPG and Java.

- **VisualAge RPG (VARPG)**: This visual development environment allows you to create and maintain client/server applications on the workstation. You can use your existing RPG skills to create graphical user interface (GUI) applications. These applications can be deployed as native Windows applications, or as Java applets that can run on any Java-capable Web browser.

- **IBM Distributed Debugger**: This tool lets you detect and diagnose errors in code developed with OS/400 Integrated Language Environment (ILE) and Original Program Model (OPM) languages and Java.

**Note:** In this redbook, the name *WebSphere*, without other designation, refers to WebSphere Application Server 4.0, Advanced Edition for iSeries.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Rochester Center.

© Copyright IBM Corp. 2002. All rights reserved.
Aleksandr V. Nartovich is a Senior I/T Specialist in the IBM ITSO Rochester Center. He joined the ITSO in January 2001 after working as a developer in the IBM WebSphere Business Components organization. Aleksandr began his career as a developer in AS/400 communications. Later he shifted his focus to business components development on WebSphere. Aleksandr holds two degrees: one in computer science from the University of Missouri-Kansas City and one in electrical engineering from Minsk Radio Engineering Institute. You can reach Aleksandr by e-mail at: alekn@us.ibm.com

Wilfried Blankertz is a Technical Support Specialist for iSeries in the IBM EMEA Central region, located in Frankfurt/Germany. From 1995 to 1998, he was assigned to the ITSO, Rochester Center, and wrote extensively and taught IBM classes worldwide on all areas of AS/400 Groupware solutions and Systems Management. Before joining the ITSO, he worked as a systems engineer for IBM Germany supporting customers with the AS/400 system and its predecessor systems (IBM System /3, /32, /34, /36, and /38) for over 25 years. Wilfried is a Certified Lotus Professional for Domino Administration for Domino R5 Application Development and Domino Administration. You can reach him by e-mail at: WillBlank@de.ibm.com

Camille Paquay is a technical consultant for iSeries for CMG in the Netherlands. As a technical consultant, he advises and supports customers in advanced architectural problems. Camille has worked with IBM iSeries and its predecessor systems (IBM System /36, and /38) for over 17 years. His experience includes positions from programming to application manager. You can reach Camille via e-mail at: camille.paquay@cmg.nl

Pentti Rasanen is an IT Specialist for IBM Finland. He joined IBM in 1999, specifically the iSeries System Support Team. His areas of expertise include High Availability software support, application development tools, and design and operating system technical support. Before joining IBM, he spent over 15 years working as a software developer and designer in the AS/400 and System/38 platforms. You can reach Pentti by e-mail at: pentti.rasanen@fi.ibm.com

Claus Weiss is an advisory software consultant with the iSeries Application Development Tools team at the IBM Canada Toronto Laboratory. He has a computer science degree from the University in Hamburg, Germany, as well as an industrial engineering degree. Claus worked as a systems engineer for IBM Germany and joined the IBM Toronto Laboratory in 1984. He was a developer and team leader for System/38 AD tools and has been a member of the Design Control Group for the AS/400 AD tools. He worked as a planner for AS/400 AD tools for the past 10 years specializing in Visual tools. Claus is a frequent speaker at COMMON and other iSeries conferences.

We especially acknowledge and thank David Muir and Claus Weiss at the Toronto lab. They were outstanding in providing support for our team.

We also thank the following people for their contributions to this project:

Abe Batthish
Vadim Berestetsky
Alison Butterill
Chris Eden
Satish Gungabeesoon
Mike Hockings
Paul Kao
Howard Lau
Maha Masri
Edmund Reinhardt  
Perry Smith  
Felix Wang  
IBM Toronto, Canada

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

► Use the online Contact us review redbook form found at:

ibm.com/redbooks

► Send your comments in an Internet note to:

redbook@us.ibm.com

► Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. JLU Building 107-2  
3605 Highway 52N  
Rochester, Minnesota 55901-7829
Part 1

Navigating the WebSphere maze

This part introduces you to the WebSphere brand and its core product – WebSphere Application Server. It discusses the importance of an application server in support of e-business.

An integral part of the suit of the WebSphere software offering is the broad array of the development tools for building Web applications. Among these tools is WebSphere Development Studio Client for iSeries (Development Studio Client), which is targeted for iSeries application developers. This part introduces this tool, explains the reasons behind its creation, and shows you the basic techniques for navigating in the Development Studio Client.

This part sets a firm ground to help you understand the importance of WebSphere Development Studio Client for iSeries and its impact on the iSeries community. It also outlines the advantages of using Development Studio Client for all iSeries application developers.

Note: In this redbook, the name WebSphere, without other designation, refers to WebSphere Application Server 4.0, Advanced Edition for iSeries.
What is under the WebSphere umbrella

The Internet offers a new and ever-evolving economic environment. It has fundamentally altered the way businesses operate. Now, in addition to the core operations (accounting, payroll, inventory, and so on) needed to run an enterprise, businesses have moved to e-commerce thanks to the rise of the Internet. The iSeries server provides a broad set of application development tools and technologies that help extend many aspects of an enterprise to implement superior business-to-business (B2B) and business-to-consumer (B2C) solutions.

Initially only used as the name for the IBM application server, the term *WebSphere* evolved to become the brand name for all IBM products that enable, develop, and deploy e-business applications. These applications support Web sites that go beyond showing static documents.

This chapter introduces you to the WebSphere development environment for the iSeries server. It covers the following topics:

- An overview on the components and infrastructure to create, maintain, and deploy Web sites with dynamic contents.
- An introduction to the wide variety of application development environments that can be used on the iSeries to help meet your goals.
- An overview of the wide range of tools that IBM uses to help leverage your core application suite running on the iSeries. Such tools can do everything from dynamically converting your 5250 into HTML to programming environments that allow you to support both traditional 5250 clients and Web clients, all with one set of source code.
- IBM WebSphere Development Studio for iSeries
1.1 Deploying the Web for e-business

Since its inception as a U.S. Defense department special project in the late 1960s, the Internet has become an economic and social phenomenon, particularly in the 1990s. In the beginning, the primary purpose was to share more or less static HyperText Markup Language (HTML) documents that are seldom or never changed. Over time, especially in the late 1990s, the Web became a communication device that allows two-way communication between the browser user and an application running on a server.

This section introduces you to the concepts and underlying technologies that support Web sites.

1.1.1 What is a Web server

The primary software backbone of a Web presence is the HTTP server, commonly known as a Web server. It is a TCP/IP application similar in concept to a network file server.

The key uses of the HTTP server include:

- File server
  - Serves any kind of file to a browser (for example .html, .gif, or .avi)
  - Caches files for performance
- Application server
  - Numerous application models
- Security server
  - Trusted component of the system
  - SSL support
  - Client authentication and access control
- Proxy server
  - Caching proxy

The purpose of the HTTP server is to respond to browser requests for files, graphics, and other media, such as sound, in the form of sending HTML pages back to the browser. On the iSeries server, Web pages are stored either in directories in the integrated file system (IFS) or as members in database files in libraries.

HyperText Markup Language is the standard language for Web pages. World Wide Web (WWW) documents are written in HTML script. The HTML coding describes, to the browser, the appearance of the displayed document. Therefore, an HTML page is a mixture of text and special HTML tags that describe the page. There are tags that describe the structure and presentation of the page, such as heading tags, ordered list tags, and table tags. There are also tags for including an image on the page and tags for creating a link to another page.

The general sequence of operation for a Web server is:

1. A Web browser sends a request for a document.
2. The HTTP server fetches the requested file and sends it to the Web browser.
3. The Web browser interprets the document and displays it.

The HTTP servers can also serve the Java applets (Java programs that run in a Web browser) and forward browser-generated requests to Java servlets on the application server.
Application servers (such as the IBM WebSphere Application Server) and transaction servers run on top of HTTP servers to provide the dynamic data and transaction capabilities of the customer e-business Web site respectively.

For the iSeries platform, from an IBM product viewpoint, three HTTP servers are available:

- IBM HTTP Server for iSeries (5769-DG1 or 5722-DG1)
- HTTP Server (powered by Apache) (also part of 5722-DG1)
- Domino for iSeries

The IBM HTTP Server for iSeries (5769-DG1 for OS/400 Version 4 or 5722-DG1 for OS/400 Version 5) is the current Web server packaged with OS/400. It provides the Web server component for the WebSphere Application Server for iSeries. However, WebSphere Application Server Version 4 also includes a built-in HTTP server that can be used for testing purposes.

### 1.2 Adding dynamic content to Web sites

This section discusses the major standards and technologies of adding logic to Web pages that the industry has or is adopting. It focuses on the standards related to applications and database serving that are the essential components in linking Web browsers and the server system. By definition, they are platform independent, but we include iSeries-specific information wherever appropriate.

#### 1.2.1 Non-Java-based applications and database serving standards

This section discusses two non-Java-based standards for applications and database serving in the e-business environment:

- Common Gateway Interface (CGI)
- Net.Data

**The Common Gateway Interface (CGI)**

On most non-iSeries Web serving platforms, CGI programming implies working with scripts written in the PERL language. Although an unsupported version of PERL is available for the iSeries server, most iSeries CGI programs are created using ILE RPG, ILE COBOL, ILE C, or Java.

**Understanding CGI processing**

The basic idea of CGI is based on the fact that the incoming URL from a browser may not only specify the name of an HTML file to be served, but can also carry additional information including the name of a program and parameters for it. Instead of serving the static contents of an HTML file, the program is executed, which in turn, has to deliver the HTML stream to be formatted and displayed by the browser.

The following steps describe the process in more detail:

1. A CGI program is requested on an incoming URL. When a browser user enters a URL containing a request in a Web page, the request is sent to the HTTP Server for iSeries with data that was entered on the Web page.

2. The CGI program is invoked. The HTTP Server for iSeries finds the CGI program. The program name is specified in MAP, PASS, or EXEC directives of the HTTP configuration file of the original HTTP server.

3. The CGI program retrieves the data that was entered by the browser user by invoking the QzhhCgiParse API. It runs as a normal OS/400 batch program and can open files; read,
update, and write records in the DB2 UDB for iSeries database; run SQL statements; or
call other programs, commands, or APIs.

At this point, the CGI program is conceptually identical to any one of traditional programs
running under OS/400. It receives input from the browser form, works with DB2 UDB for
iSeries database files, and prepares a response to the browser. However, it is not
considered an interactive job, since it was not started through a display device.1

4. Based on the logic and possibly the database access the program performed, it creates a
stream of HTML data. The resulting HTML is then returned to the HTTP Server. When the
response HTML to be returned to the browser is created, the CGI program uses
application programming interface (API) calls to send the resulting HTML to the HTTP
Server for iSeries program.

5. The resulting HTML is sent back to the browser. HTTP Server for iSeries sends the
completed HTML page to the browser. The process is complete.

**Note:** The HTML code that defines the contents and the format of the Web page to be
displayed are dynamically created, typically by merging literals defined within the
program’s source code with actual DB2 and calculated data. Static Web contents are
typically created by a person (a Web designer) and are stored without modification in a text
file. Under OS/400, such a text file typically resides as a stream file in an directory of the
integrated file system (IFS).

**Persistent CGI**

Persistent CGI is an extension to the CGI interface that allows a CGI program to remain active
across multiple browser requests and maintain a session with that browser client. This allows
files to be left open, the state to be maintained, and long running database transactions to be
committed or rolled-back based on end-user input. The OS/400 CGI program must be written
using named activation groups, which allow the program to remain active after returning to the
server.

See Chapter 4 of HTTP Server for iSeries Web Programming Guide, GC41-5435, for more
information on persistent CGI.

**Benefits of using CGI**

The primary reason to use CGI programing on the iSeries server is that you or your staff may
already be familiar with one of the iSeries server programming languages such as ILE RPG,
ILE COBOL, and ILE C. For example, if you already know RPG, it is relatively simple to learn
how to incorporate CGI processing techniques into an RPG program, compared with learning
Net.Data or Java.

When you write a CGI program, you have access to all of the OS/400 programming tools. For
example, you can use native database operations in your CGI programs. You can also use
string handling operations in the language to create the exact HTML statements that you
need. Finally, you can use the same debugging tools to put your CGI program into production.

**Considerations of using CGI**

A major concern of using CGI compared to other options (predominantly compared to
Java-based technologies) is that it is somewhat labeled as legacy technology.

Generally considerably more code is required for an RPG CGI program as compared to
Net.Data. Once you create a working RPG CGI program, you can easily copy code for further
programming.

---

1 In general terms, you may consider a Web browser as a display device. However, OS/400 does not treat it as such. Here, the term
stands for a local, remote, or virtual display device of one of the 5250 or 3270 types of terminals.
The most difficult part of the RPG CGI program is the code that parses the buffer returned from the QzhbCgiParse API. Again, once you develop several programs using this API, you can extract the routines easily.

**Positioning CGI**

CGI programming is your optimal choice to Web-enable your iSeries server applications if your primary goal is to use existing programming skills. CGI is supported on the iSeries server using IBM HTTP Server for iSeries (5769-DG1) and it is shipped with OS/400 at no additional charge.

**More information on CGI**

You can find more information on CGI with the IBM HTTP Server (original) in the following Redbooks:
- *Cool Title About the AS/400 and Internet*, SG24-4815 (Chapter 9)
- *Unleashing AS/400 Applications on the Internet*, SG24-4935

Also, refer to Chapter 6 of *HTTP Server (powered by Apache): An Integrated Solution for IBM @server iSeries Servers*, SG24-6716, for more information on CGI with the IBM HTTP Server (powered by Apache).

The best implementation guide for CGI programming both with the HTTP Server (original) and HTTP Server (powered by Apache) is *HTTP Server for iSeries Web Programming Guide*, GC41-5435.

For more information, consult the following resources:
- Sample CGI programs:
- Easy400 is an open source site that enables RPG and COBOL programmers to develop full-functional e-business solutions by exploiting their current skills. This site has free development and education tools:
  - [http://www.easy400.ibm.it/](http://www.easy400.ibm.it/)
- Ignite/400 iSeries e-business user group has plenty of sample programs and tips at:
  - [http://www.ignite400.org](http://www.ignite400.org)

**Net.Data: A scripting language**

Part of the IBM HTTP Server (5722-DG1 or 5769-DG1) comes as a server-side scripting language called *Net.Data*. It extends Web servers by enabling the dynamic generation of Web pages using data from a variety of data sources. The data sources can include relational and non-relational database management systems such as DB2, DRDA-enabled databases, and flat file data.

Net.Data applications can be built rapidly using a macro language that is simple, yet powerful. They allow the reuse of existing business logic by supporting calls to applications written in a variety of programming languages, including Java, C/C++, RPG, COBOL, CL, REXX, and others. Net.Data provides several features for high performance, including persistent connections to databases.

Using macros that you develop as input to the CGI program, Net.Data allows users to achieve the same tasks as CGI programs do without writing their own CGI programs. The system-provided Net.Data CGI program, Net.Data, uses the macro to:
- Send HTML to your browser
- Run SQL commands
- Call system services, such as programs, compiled in other languages
Understanding Net.Data processing

The steps for Net.Data processing are somewhat similar to CGI, except for the fact that the program being invoked is not written by a user. Instead, a program supplied with the HTTP server is called and controlled by a script written by the programmer. The process is explained here in depth:

1. A Net.Data macro is requested on an incoming URL. The request is sent through the URL to the IBM HTTP Server for iSeries program.

2. The IBM HTTP Server for iSeries program recognizes that the request is for Net.Data and invokes Net.Data.

3. The Net.Data configuration options are retrieved from the INI file.
   Upon starting, the Net.Data program retrieves the initialization options from the optional INI file. Although it is optional, if you do not have an INI file, the URLs are considerably more complicated.

4. The macro and start-at section within the macro are identified, and Net.Data determines where on the system the macro source file is located.

5. Net.Data now retrieves the macro, which is stored in text format.

6. Net.Data starts the execution of the macro at the start-at section by parsing the macro, and processing any global function calls. The start-at section is typically an HTML block that contains statements describing the initial page to be sent to the browser.

7. If any DB2 UDB for iSeries data is to be accessed with SQL statements or function calls to other iSeries should be performed, Net.Data does this now.

8. The resulting HTML is returned from Net.Data to the IBM HTTP Server for iSeries program.

9. The resulting HTML is sent back to the browser by the IBM HTTP Server for iSeries.

Benefits of using Net.Data

Net.Data is a superb tool for creating simple HTML forms that act as front ends to database queries. In addition to the default behavior of Net.Data (display SQL results in an HTML table), you can customize the resulting table or use Net.Data functions to add other HTML options to the table, such as listbox fields and check box fields.

If you do not have iSeries programming skills (usually RPG, COBOL, or C) or if you are uncomfortable with using CGI APIs required to get, parse, and return data to the browser, you may find it easier to work with Net.Data. You can concentrate on the application, rather than the mechanics of communicating with the browser.

Net.Data also provides built-in support for working with the results of SQL queries. It takes care of getting and parsing requests from the browser and preparing output to return to the browser.

Net.Data is available on a variety of platforms, including AIX, OS/2, OS/390, and Windows NT.

Considerations of using Net.Data

A major concern of using Net.Data, especially when you expect a heavy transaction at any given time, can be a performance issue. Net.Data is interpreted, not compiled. On one hand, this can be a benefit. For example, you can develop or make changes more quickly than the equivalent compiled program. On the other hand, it can cause some performance concerns.
**Positioning of Net.Data**
Net.Data can be your optimal choice to Web-enable your iSeries server applications if your primary goal is to complete the project without serious investment on internal or external programming skills acquisition.

**More information on Net.Data**
You can find more information on Net.Data with IBM HTTP Server (original) in the following Redbooks:
- *Cool Title About the AS/400 and Internet*, SG24-4815 (Chapter 9)
- *Unleashing AS/400 Applications on the Internet*, SG24-4935

Also, refer to Chapter 6 of *HTTP Server (powered by Apache): An Integrated Solution for IBM @server iSeries Servers*, SG24-6716, for more information on Net.Data with the IBM HTTP Server (powered by Apache).

You can find a user forum for Net.Data at:
http://63.209.20.13/cgi-bin/db2www/forum.d2w/main

You access or download the Net.Data manuals from the Net.Data home page at:

Bits & Bytes Programming has a library of Net.Data and other programming samples at:
http://www.web400.com

### 1.2.2 Java-based technologies

During the past three or four years, Java has become the premier language to develop Web applications. Besides the advantages of Java for object-oriented programming, this brought and still brings more and more reusable standards and tools while Java becomes increasingly popular.

This section provides a brief overview of the Java environment. It also discusses Java-based applications and database serving standards including:
- Java applets
- Java servlets
- JavaServer Pages (JSP)
- Enterprise JavaBeans (EJB)

**The Java platform**
Java is both a programming language and a platform. As a high-level programming language, Java is unique in its architecture-neutral, object-oriented configuration.

The Java language is unusual in that each Java program is both compiled and interpreted. With a compiler, you can translate the Java program into an intermediate language known as Java bytecodes. These are platform-independent codes that are interpreted by the Java interpreter. With this interpreter, each Java bytecode instruction is parsed and run on the computer. Compilation happens once. Interpretation occurs each time the program is executed.

Think of Java bytecodes as the machine code instructions for the *Java Virtual Machine* (JVM), the base for the Java platform. Every Java interpreter (for example, a Web browser for running applets or a Java development tool) is an implementation of the JVM.
The JVM allows the Java platform to be independent from the processor. The Java bytecodes help make Java's famous “Write Once, Run Anywhere” capability possible. Once the Java program is compiled into bytecodes on any platform having a Java compiler, these bytecodes can run on any implementation of the Java Virtual Machine. The same program, for example, can run on the iSeries server, Windows NT, or Sun Solaris.

The Java platform refers to the environment in which a program runs. While most platforms consist of both hardware or software (for example, Intel-based Windows NT), the Java platform differs in that it is a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

- The Java Virtual Machine
- The Java application programming interface (Java API)

The Java API is a large collection of ready-made software components that provide many capabilities, such as GUI widgets. The Java API is grouped into libraries, or packages, of related components.

Java manifests itself in the e-business arena by being the platform for several standards, including applets, servlets, JavaServer Pages, and Enterprise JavaBeans.

**Java applets: Running application logic in a Web browser**

Java applets are dynamic and interactive programs that can run inside a Web page displayed by a browser enabled for Java, such as Microsoft Internet Explorer or Netscape Navigator. Applets were the first application of Java to gain widespread notice. They first gained popularity in the mid-1990s as a tool to add animation and other effects to Web pages.

Browsers that provide support for applets include:

- Netscape Navigator Version 4.04 or later
- Microsoft Internet Explorer Version 4.01 or later
- Sun Microsystems HotJava

Java applets are delivered to the browser in HTML files that contain a reference to the applet. Upon arriving in the browser, the applet begins execution. The applet usually displays what appears to the user as a Web page. However, all of the user interface elements are contained within the applet itself and are not rendered by HTML.

Most importantly, the applet can communicate directly back to the server. When the server responds, it is up to the applet to display the response. Because the applet is in control of its user-interface elements, it can update them to display the server response, again without requiring HTML.

**Java applet processing on the iSeries server**

One of the most important features of applets for iSeries developers is the ability of the applet to interact with the iSeries server. When you create an applet, you can include Java classes from the IBM Toolbox for Java (5722-JC1) that let you access objects on the iSeries server.

For example, you can:

- Access records from the iSeries database using SQL statements record-level access techniques
- Call programs or invoke commands on the iSeries server
- Send and receive entries from data queues on the iSeries server
The steps of the applet process follow this sequence:

1. An HTML page that contains an APPLET tag is requested on an incoming URL.

2. A requested HTML page is returned to the browser. As the page is sent to your browser, the Java classes used in the applet are also sent to the browser. After receiving the applet, the browser starts it and passes control to the applet.

3. A request is sent from the applet to the iSeries server. Once the applet is started, you interact with it as you would with other forms displayed in the browser. The applet may require that you enter data or make selections. You usually have one or more buttons in the applet that you can click. The buttons are not associated with a FORM statement in an HTML form, but rather are used to invoke methods in the applet’s Java code. It is up to the Java code in the applet to use methods to send requests to the iSeries server.

4. The iSeries server services the request from the applet. The request from the applet is serviced the same as other types of program requests running on the iSeries server. For example, if the request is to create an SQL result set, the SQL processor is invoked to query the iSeries database.

5. After the iSeries processes the request, the results of servicing the request are sent back to the applet. The results are in the format that pertains to the request. For example, an SQL request generates a result set. A data queue read operation generates a packet of bytes that contains the data queue entry. The applet receives the results using the Java class and methods appropriate to the request.

6. Once the results are available to the applet, it displays those results using any of the user interface components included in the applet when it was designed. The display is not limited to HTML only. After formatting and displaying the results, the applet is available for additional user interaction, which can include additional requests to the applet for more data from the iSeries server.

**Benefits of using applets**

Applets are useful when you need to work with iSeries resources and you do not want to create Net.Data or CGI programs on the iSeries server. You can also work with applets for greater control to design the user interface and interact with the user at run time.

Another reason to use applets is to use the industry standard Java language, rather than the proprietary OS/400 Net.Data or CGI programming languages.

Although the Java classes to access the iSeries server are proprietary, the classes are readily usable by any Java programmer with a basic knowledge of the iSeries server.

**Considerations of using applets**

Applets depend on the capabilities of the browser under which they run. Browsers are available from numerous vendors. Each of these browsers use a different security model. You may have to program an applet differently based on the browser under which you are running. In other words, there is a chance that your application may not run as it was designed depending on your clients’ browser environment.

Another point of consideration is the size of the applets and communications link speed. Applets and their associated classes should be stored at a central location and downloaded as needed. This results in downloading the classes as needed by the browser. Depending on the size and number of classes, and the speed of the communications line, the download time can be an expensive use of resource and time.
Using servlets to run Java code on the Web server

A servlet is a Java program that runs on an application server and works with a Web server to generate dynamic content. Like a CGI program (see “The Common Gateway Interface (CGI)” on page 5 for more information), it receives client requests, handles them, and sends a response. If a servlet is called through HTTP, the response is typically an HTML flow.

Unlike CGI programs that are loaded in memory each time a client makes a request, a servlet is loaded in memory once by the application server and can serve multiple requests in parallel using threads. On the iSeries server, you can use persistent CGI to allow the CGI program to remain active across multiple browser requests and maintain a session with that browser client. See “Persistent CGI” on page 6 for more information.

The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets. HTTP servlets are a specialized servlet type that provide a framework to handle the HTTP protocol, such as the GET and POST methods. All HTTP servlets must inherit from the javax.servlet.http.HttpServlet class. A servlet's life cycle is composed of three phases (see Figure 1-1):

- **init() method**: This method is called by the application server when the servlet is first loaded into memory. You can provide initialization parameters for a servlet in the Web application's configuration files.

- **service() method**: This method is called for each client request. For HTTP requests, the service() method is specialized to dispatch the request to the appropriate doGet, doPost, doPut, or doDelete methods, depending on the HTTP request method (DO, POST, PUT, DELETE). If you write HTTP servlets, you should not override the service() method, but instead override the appropriate doXXX method.

- **destroy() method**: This method is called when the application server unloads the servlet from memory. You should free any resources used by the servlet in this method.

![Diagram](image)

**Figure 1-1  The servlet’s life cycle**

Servlets are managed by the application server (see 1.3.2, “The purpose and use of the WebSphere Application Server” on page 19, for information on the application server). They are loaded in memory upon the first client request or at server startup. Each client request is then served on a different thread.
**Comparing servlets and applets**

Java servlets have many advantages over Java applets for a number of reasons. Two of the main advantages are in:

- **Performance**
- **Ready access to the iSeries database**

Here, the advantages are because Java servlets run entirely on the iSeries server. Therefore, the servlets can access the iSeries database and other system resources with much less interaction over the network link.

**Java servlet processing on the iSeries server**

Support for Java servlets on the iSeries server is provided by the WebSphere Application Server, rather than by the IBM HTTP Server for iSeries (5769-DG1 or 5722-DG1). However, the HTTP Server (powered by Apache) does include an industry-standard Java servlet and JavaServer Pages engine based on technology from the Apache Software Foundation's Jakarta Tomcat open source code base.

Apache Software Foundation's Jakarta Tomcat for iSeries support can be used as a simple starting point for business partners and customers interested in learning about or piloting Java servlet and JSP applications. For more information, refer to Chapter 7 in *HTTP Server (powered by Apache): An Integrated Solution for IBM @server iSeries Servers*, SG24-6716.

If you are already familiar with CGI programming, covered earlier in “The Common Gateway Interface (CGI)” on page 5, servlet processing is similar. The most important differences are:

- **The servlet must be written in Java.**
- **The servlet can be pre-started in the multi-threaded job pool so that there is no startup overhead when it is invoked.**
  
  If a servlet is not currently active when it is invoked, you incur the startup overhead on its first usage. After that point, the servlet is available for subsequent invocations.
- **Several Java classes exist to facilitate communication with the Web browser.**

Each step in the servlet process is explained here:

1. **The servlet is invoked by an incoming request of a Web browser.**
   
   The servlet is identified in a URL that you either type into the browser address field or click as a link. Or, it can be specified on an HTML FORM statement that is used when a Submit button is clicked. The request is sent using the HTTP protocol to the HTTP Server for iSeries program, which passes the request to the **application server**.

2. **The application server identifies the servlet and invokes it.**
   
   See 1.3.2, “The purpose and use of the WebSphere Application Server” on page 19, for information on the roles of the application server.

3. **The servlet reads the form data.**
   
   The servlet uses the **doGet** or **doPost** method to read the form data, and the input data is available to the servlet in the HttpServletteRequest input stream.

4. **You parse the field name/value pairs.**
   
   Now that the form data is available in the HttpServletRequest stream, you can parse it into field name/value pairs that correspond to the data fields used on the HTML form. The Java servlet APIs include the **getParameterNames** and **getParameterValues** methods to retrieve the list of field names and values from the input stream. After retrieving the name and value pairs, the values are available in enumerations within the servlet.
5. The servlet processes the database and other requests.

At this point, all of the data from the form is available to the servlet. The servlet can now run the functions that are required to service the request. For example, the servlet can run an SQL query against the iSeries database or use other Java classes in the AS/400 Toolkit for Java to work with other iSeries resources.

6. The response HTML is generated.

The servlet can start generating the response HTML to send to the browser at any point. Typically, the servlet generates HTML headers, followed by the actual form heading, then one or more lines of data, and finally a page footer.

The Java servlet APIs provide the HttpServletResponse output stream to transport generated HTML statements from the servlet back to the browser. Typically, you, as the servlet programmer, create well-formed HTML statements as simple strings, using concatenation as necessary to build a string of HTML tags and the response data. To actually send the HTML, you simply use the print or println methods on the HttpServletResponse stream object.

7. The HTTP Server sends the response HTML to the browser.

Because the HTML is written in the servlet, it is sent from HTTP Server for iSeries to the browser. The STDOUT file is used, which is common for all server-to-browser communication.

8. The resulting Web page is displayed at the browser.

Because the page is composed of standard HTML elements, there are no special requirements or security considerations for the browser.

**Benefits of using servlets**

If you have not yet started creating iSeries Web serving applications, seriously consider adopting Java servlets as the technique to use, even if you do not yet use Java. In fact, learning Java by working with servlets is ideal, since servlets are basically batch processes that do not deal with user-interface issues found in client-side programs.

A Java servlet is based on industry-standard Java and provides an alternative to Net.Data, CGI programming, and applets. Although using Net.Data (see “The Common Gateway Interface (CGI)” on page 5) and CGI (see “Net.Data: A scripting language” on page 7) programming techniques for Web serving is fine for iSeries users who have those skills, there is little to attract Java programmers to the iSeries server as a Web serving platform. With Java servlet support, the iSeries server now runs as a powerful Web serving platform.

With servlets, you can parse requests from HTML forms and use simple print and println methods to send response HTML back to the browser. There are only a few browser dependencies when you use servlets, while there are considerable differences between browsers when you run applets.

Java servlets are often faster than equivalent CGI programs. CGI programs have a reputation of being slow and unable to scale. The primary reason for this is the way they are loaded. CGI programs are loaded at each invocation in a process separate from the Web server. This requires a relatively large amount of resource, both from the Web server and the operating system. Also, many CGI programs (on other platforms) are written in the interpreted PERL scripting language. This interpretation adds significant overhead.

Java servlets were designed with performance and scaling in mind. They are loaded only once, either when the Web server is initialized or at first invocation. They are not unloaded unless this is explicitly done by the application server or system administrator. Also, servlets take advantage of Java’s built-in multi-threading model. This results in faster communication between tasks and threads.
Your servlets can optionally be started when the WebSphere Application Server is started, so that they are available and waiting for incoming requests. On the other hand, Net.Data and CGI programs are typically short-lived; there is little space to optimize their performance. Also servlets do not necessarily end when they are done serving a request. They can remain active for further requests. If you create an equivalent Net.Data or CGI application and a Java servlet, you typically see much better response time with the Java servlet.

Java servlets provide two simple input and output stream objects to get the data and to write the response, and two simple methods to parse field name and value pairs. Most of the work in CGI programming is concerned with getting the input data from the Web form, parsing the data into discrete field and value pairs, and writing response HTML back to the STDOUT file.

**Considerations of using servlets**

There are practically no considerations or potential disadvantages for using servlets. In e-business, the Java servlet is a far more useful business tool than the applet. A servlet is essentially the opposite of an applet. Think of it as a server-side applet in that servlets run inside the Web server or the application server in the way applets run inside the Web browser.

**Java Beans**

Software components are standard and reusable building blocks for software development. Basically, they encapsulate function and provide services based on a strict specification of their interface. Because of this specification, they can be used as “black boxes” (components whose internal state is hidden) and combined with other software components to build a complete application.

The Java Beans specification is the standard component model for the Java language and is the component model used by VisualAge for Java. It describes how Java classes should appear and behave to be treated as Java Beans. Beans can be visual components, such as a button or a list, that you use to build the user interface or view of your application. They can also be nonvisual components, such as a bank account, that typically represent the business logic or domain model of your application.

To build an application or an applet with a bean manipulator tool, such as the VisualAge for Java Visual Composition Editor, you typically drag and drop the beans you want to use into a working area and wire, or connect, them together. This visual programming approach is an extremely productive way to create applications.

Java Beans provide support for:

- **Portability**: Beans can be created and run on any Java platform.
- **Introspection**: The tool that you use to combine beans can automatically discover how a bean works.
- **Properties and customization**: Properties are a bean’s attributes. A developer using a bean can customize the appearance and behavior of a bean by changing its properties.
- **Persistence**: The state of a bean can be saved and then reloaded through the serialization function of the JDK 1.1 or later.

The key to understanding and using a bean is to understand its features, that is, the events, properties, and methods that it exposes. A bean exposes a feature by making it available to other beans. The features a bean exposes constitute its interface. You can use this interface to combine beans with each other, through a tool such as the Visual Composition Editor of VisualAge for Java.

You can find more information on Java Beans on the Web at:

http://java.sun.com/products/javabeans/
Separating the presentation from the logic with JavaServer Pages

CGI and Java servlets have two properties in common:

- Both provide the logic for Web applications to be run on the server.
- They need to construct the HTML code to be presented on the browser.

The latter item represents some level of complexity, since the programs need to “glue” those pieces of HTML describing the presentation (fonts, color, position... etc.) together with the dynamically produced data. While this does not sound extremely difficult from the programmer’s perspective, it creates two main challenges:

- Due to the fact that Web sites need to be attractive to the user, it becomes more and more common that the design of the visual appearance of a Web page is done by a different person (or department) from those who program the logic.
- During the writing of a servlet, the HTML code that defines the appearance of the page has to be included with the code in the form of text literals that most likely represent only a fraction of the entire HTML stream. This makes it difficult to design and test the page. Also most Web editors do not assist in designing the layout of the page.

The concept of JavaServer Pages helps overcome these challenges. JSPs allow you to embed Java code into HTML pages to execute them directly by the server when the page is served. JSP technology is similar to Microsoft’s Active Server Pages (ASP) technology, but JSP is standardized and portable. By using JSPs with servlets, you can separate business logic from its presentation and reuse the business logic (using Java Beans) more practically.

The servlet and JavaServer Pages programming model is based on the Model-View-Controller (MVC) model. In the MVC model, the data (model), the logic manipulating the data (controller), and the presentation of the data (view) are designed to be independent. If the view needs to change, the business logic and the data are not affected. If the data interface changes, the controller can be updated without affecting the view.

In the MVC model, the servlet receives a request from a client, accesses the data through a set of reusable components (beans or enterprise beans), and invokes a JavaServer Pages component to display the results of the request.

A JSP component is like a template for an HTML page, with slots for the dynamic content that varies on each request. These slots are filled at run time with dynamic data from the servlet, such as the user name’s or the current time. The servlet must query the dynamic data, package it, and pass it to the JSP component. The servlet stores dynamic data in a bean instance and places the bean instance somewhere the JSP component can access it. The JSP component retrieves the bean instance and inserts the dynamic data (typically bean properties) into the HTML page using special JSP tags.

When you access a JSP file, it is compiled by a JSP compiler into a servlet. The servlet is then loaded into memory and run. This compilation process only occurs the first time the JSP is accessed or after you change the source code.

Maintaining JSP files is easier if you keep the Java code included to a minimum. If you include complex Java code fragments in the JSP, you need to look in the view to find business logic. This is no better than editing a servlet to change the view of your application.
Using Enterprise JavaBeans to bring your business to the Web

Enterprise JavaBeans is the key technology to deal with distributed transaction and complex persistence issues in a Web application. Progressing from maintaining a simple Web presence, with a home page, to a dynamic presence, with an active Web site allowing ordering of products and services, contemporary businesses now face the next evolution of Web enablement. This is the need to integrate their Web-based systems with their other business systems. This linkage is the basis of the transactional stage of a company's Web enablement.

EJBs enable users of such application servers as WebSphere to achieve this integration. EJBs are critical to the operation of a transactional server.

EJBs extend Java's "Write Once, Run Anywhere" model of hardware and operating system independence to middleware and application server independence. It achieves this by separating the business application from the system services. Therefore, developers are not locked into using a particular vendor's middleware servers.

EJB technology complements and extends the existing Java architecture by providing an application server environment to handle system services that would otherwise be handled by the application code.

Enterprise JavaBeans, based on the Sun Microsystems Enterprise JavaBeans specification, are reusable software components to build distributed, transactional business applications using Java. EJBs contain business logic functions, which simplify the development of integrated solutions.

Some notable features of EJBs include:

- They make it possible to build distributed applications by combining components developed by you or by different software vendors. Defining business objects as components promotes reusability and development efficiency.

  An application developer can create new components, reuse existing components, or purchase components. By using existing components or purchasing components, the developer assembles the application rather than taking further time to develop and test new code.

- They make it easy to write applications. Application developers do not have to deal with low-level details of transaction and state management, security, persistence, multi-threading, resource pooling, and other complex low-level APIs.

  A programmer gains direct access to the low-level APIs. The majority of these details are managed by the EJB server. Therefore, developers can concentrate on producing business logic (the coding of their business rules, logic, and knowledge) rather than spending time on coding system-level services (such as transactional support, persistence, and so on).

- EJBs are developed once and then deployed on multiple platforms without recompiling or modifying source code.

- They allow interoperability with other Java and non-Java applications.

Enterprise JavaBeans and the iSeries server are a perfect match. The Enterprise JavaBeans component model logically extends the Java Beans concept. It is targeted at server-tier business logic development. It provides interfaces that insulate the programmer from the complexities and dependencies unique to a platform.

EJB technology provides a component model for server applications. It allows you to easily separate user interfaces from business logic.
The server-side business logic is packaged as Enterprise JavaBean components. Once they are written and deployed on a server, such as the iSeries server, client programmers can use them with very little knowledge of how the beans actually work. The client programmer only has to know what methods Enterprise JavaBeans support and how to call them.

EJB technology leverages existing applications currently running business environments. It also allows the extension of existing applications to provide new and additional functions of the business.

The EJB specification consists of two major units that reflect the critical separation between the business application and system services:

- Components: Entity beans and session beans
- Services: EJB server and EJB container

You can find more information on EJBs in Chapter 3 of IBM WebSphere V4.0 Advanced Edition Handbook, SG24-6176.

1.3 Developing and deploying Web applications

The previous sections describe the building blocks that you can use to create Web pages that present static information on a Web browser and include dynamically generated data and allows for interactions with the user. Creating dynamic Web pages, however, is normally not sufficient to enable e-business. To deploy the Internet for your business, you typically need a whole set of applications.

This section explains what a Web application really is and how it can be deployed and managed by the WebSphere Application Server.

1.3.1 What is a Web application

According to the IBM definition, a Web application is an application that is accessible by a Web browser. It provides some function beyond the static display of information, for example, by allowing the user to query a database. Although common components of a Web application include HTML pages, JSP pages, and servlets, it may also include other application components. Existing (sometimes called “legacy”) applications and databases that can be used to provide logic and data for the Web application are of special interest.

As described in 1.2, “Adding dynamic content to Web sites” on page 5, there are various techniques to create dynamic Web contents. Not all of the techniques are based on or are limited to Java technologies. For example, CGI programs on iSeries can be written with any of the program languages available under OS/400.

However, during the last two or three years, Java has been established as the development platform for Web applications, which in turn has caused many standards, tools, and techniques based on Java to be developed. This, together with the fact that Java is an easy-to-learn, fully object-oriented language, makes it the ideal choice to implement Web applications. See also 1.2.2, “Java-based technologies” on page 9.

Whenever we refer to a Web application in this book, we actually mean those applications based on Java standards.
1.3.2 The purpose and use of the WebSphere Application Server

Many Web applications today are based on such Java standards as servlets, JavaServer Pages, or Enterprise JavaBeans. They all require a certain run-time environment. For example, the application needs to have access to the appropriate classes from the Java Development Toolkit (JDK).

Such a run-time environment could be provided by the HTTP server, which also serves the static HTML documents or CGI based applications. In fact, the HTTP server for OS/400 V4R3 included servlet support. However, over time, the standards became increasingly more complex and powerful, and Web applications increased drastically in terms of how many applications exist and how many objects for each application need to be managed. This development made the idea of incorporating an application server into a Web server impractical.

An application server, as such, supports a new architectural model where the applications run on the server and the client's system requirements are minimal thanks to a Web browser. This is quite a powerful concept to know that a Web application can be accessed from virtually anywhere. Also, an application server provides the means to bring a host application to the Web. In short, by relying on an application server, you benefit from the following advantages:

- There is no need to support the client's software.
- Maintenance is easy because an application is deployed from a single place -- a server.
- An application is accessible from any client that supports a Web browser.
- Application development is streamlined by using the power of the Java 2 Enterprise Edition (J2EE) blueprint.

The IBM WebSphere Application Server

The IBM WebSphere Application Server is intended for organizations that want to take advantage of the productivity, performance advantage, and portability that Java provides for dynamic Web sites. It provides a framework for a consistent, strategic link between HTTP requests and business data and logic. It includes:

- Java run-time support for Java servlets
- Support for JavaServer Pages. A relatively new technology that provides a server-side scripting technique for generating Web pages. Where Java servlets typically require the skills of a Java programmer, JavaServer Pages can be created by authors who have some basic HTML programming skills.
- Support for Enterprise JavaBeans. These are distributed Java components for building enterprise-level applications. They provide security, distributed transactions, and production-level scalability for the Web applications, and simplify the process of developing such applications.
- High-performance connectors to many common back-end databases to reduce the coding effort required to link dynamic Web pages to real line-of-business data
- Application services for session and state management

WebSphere Application Server for iSeries

The WebSphere Application Server is available now in its fourth version. Each version, except the first, is or was available in more than one edition. The various editions differ by price and the level of the support they provide:

- **Standard Edition**, the free of charge (on iSeries) entry edition, provides run-time support for the servlets and JSPs (not available in V4.0).
- **Advanced Single Server Edition** adds to the capabilities of the Standard Edition the support for EJ Bs (available only in V4.0).
Advanced Edition allows you to manage multiple servers with support for advanced features, Workload Management (WLM), failover support, and scalability.

Enterprise Edition is not available on the iSeries server.

On the AS/400e and iSeries servers, IBM supports only the Standard and Advanced Editions.

**Versions, editions, modification levels and group PTFs**

As described in the previous section, the various editions of WebSphere Application Server allow you to choose how much you want to invest, based on the functions you need. For example, if you do not need Enterprise JavaBeans and only need to deploy a single application server, you may start with the WebSphere Application Server Version 3.5 Standard Edition.

Over time, several versions of the WebSphere Application Server have been made available, each implementing the latest technology and standards. At the time this redbook was written, Versions 3.5 and 4 are currently supported. Version 3.5 cannot be “upgraded” to Version 4, and you have to order a different product. See Table 1-1 for a list of the currently supported versions and editions of WebSphere Application Server for iSeries.

After June 2003, Version 3.5 will no longer be supported. Therefore, you may decide to start with Version 4 and the edition of your choice right away to avoid migrating later.

WebSphere Application Server is continuously enhanced and maintained during the lifetime of each version and edition. Therefore the same version and edition of WebSphere Application Server for iSeries exist in several modification levels. For the most current modification level for the product, you need to install a group PTF. The number for the group PTF is different for each version and edition of WebSphere Application Server. Table 1-1 lists the appropriate PTF numbers for each version.

For example, the minimum level of WebSphere Application Server supported by the WebFacing Tool is WebSphere Application Server Version 3.5.2 Standard Edition. If you have OS/400 V5R1 installed, you need to order 5733-AS3 and PTF SF99146. However, the modification level you get with the group PTF depends on when you order this PTF. In other words, if you ordered PTF SF99146 in April 2002 and applied it to 5733-AS3, you now have the WebSphere Application Server Version 3.5.6 Standard Edition. If you ordered and applied the same PTF a year earlier, for example in April 2001, you would be at Version 3.5.3 instead.

Therefore, it does not make any sense to verify the modification level with the Display PTF (DSPPTF) command. To help you identify the modification level of the WebSphere Application Server on your iSeries server, a data area with the same name as the PTF is stored in library QEBQDV4 or QJEBAES4 for WebSphere Application Server Version 4. For WebSphere Application Server Version 3, it is stored in the library QEBJ, with the date and modification level of the product.

**Important:** Do not use DSPPTF with group PTFs.

The easiest way to verify the modification level of your WebSphere Application Server is to enter the following command on any OS/400 command line:

```
WRKDTAARA QEJBADV4/*ALL
```

Or you can enter the following command:

```
WRKDTAARA QEJB/*ALL
```

2 WebSphere Application Server Version 3.5 and WebSphere Application Server Version 4 may be installed on the same iSeries server. Therefore, you may want to enter both commands.
You see the Work with Data Areas screen, which typically shows only a single data area with the name of the group PTF or none if no group PTF is installed yet. You enter option 5 (Display) in the field at the left of the data area name. Then you see a Display Data Area screen like the example in Figure 1-2.

![Figure 1-2](image)

The content of the data area shown at the bottom of the screen indicates that the second revision of PTF SF99241, dated 01 February 2002, has been applied to the product 5733-WA4, making it modification level 4.0.2.

Table 1-1 helps you identify the various editions and versions of the WebSphere Application Server that are currently available for the releases of OS/400. It also includes the number for the necessary group PTF and the minimum level of cumulative PTF package (CUM Pack).

<table>
<thead>
<tr>
<th>WebSphere Application Server for iSeries</th>
<th>OS/400 release</th>
<th>JDK\textsuperscript{a} 57xx-JV1</th>
<th>Prerequisite CUM Pack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 3.5 Standard 5733-AS3 SF99141 V4R4</td>
<td>1.2 (Opt. 3)</td>
<td>C1072440</td>
<td></td>
</tr>
<tr>
<td>Version 3.5 Standard 5733-AS3 SF99142 V4R5</td>
<td>1.2 (Opt. 3)</td>
<td>C2050450</td>
<td></td>
</tr>
<tr>
<td>Version 3.5 Standard 5733-AS3 SF99146 V5R1</td>
<td>1.2 (Opt. 3)</td>
<td>C2036510</td>
<td></td>
</tr>
<tr>
<td>Version 3.5 Advanced 5733-WA3 SF99137 V4R4</td>
<td>1.2 (Opt. 3)</td>
<td>C1072440</td>
<td></td>
</tr>
<tr>
<td>Version 3.5 Advanced 5733-WA3 SF99138 V4R5</td>
<td>1.2 (Opt. 3)</td>
<td>C2050450</td>
<td></td>
</tr>
<tr>
<td>Version 3.5 Advanced 5733-WA3 SF99147 V5R1</td>
<td>1.2 (Opt. 3)</td>
<td>C2036510</td>
<td></td>
</tr>
<tr>
<td>Version 4.0 Advanced 5733-WA4 SF99239 V4R5</td>
<td>1.3 (Opt. 5)</td>
<td>C1296450</td>
<td></td>
</tr>
<tr>
<td>Version 4.0 Advanced Single Server 5733-WS4 SF99240 V4R5</td>
<td>1.3 (Opt. 5)</td>
<td>C1296450</td>
<td></td>
</tr>
<tr>
<td>Version 4.0 Advanced 5733-WA4 SF99241 V5R1</td>
<td>1.3 (Opt. 5)</td>
<td>C1302510</td>
<td></td>
</tr>
<tr>
<td>Version 4.0 Advanced Single Server 5733-WS4 SF99242 V5R1</td>
<td>1.3 (Opt. 5)</td>
<td>C1302510</td>
<td></td>
</tr>
</tbody>
</table>

\textsuperscript{a} IBM Developer Kit for Java (JDK) is available as 5722-JV1 for OS/400 V5R1 and 5769-JV1 for earlier releases. The licensed program comes with several options of which you can install one or more depending on which version of JDK you need. This column shows you the necessary version and installation option for each version and edition of WebSphere Application Server for iSeries.
If you are using WebSphere Application Server V3.5, you also need to order and install a group PTF that upgrades the application server to version 3.5.2 or higher. For the latest information on the PTF numbers you need, see the WebSphere Application Server PTFs site:

http://ibm.com/servers/eserver/iseries/software/websphere/wsappserver/services/service.htm

For an overview of all versions of WebSphere Application Server for iSeries, see:

http://ibm.com/servers/eserver/iseries/software/websphere/wsappserver/

1.4 iSeries: A flexible application server for the Web

The iSeries platform supports a wide range of solutions that were built using many different application technologies or environments. As shown in Figure 1-3, it supports a number of distinct, yet interoperable application environments that share common system services:

► **Java**: Includes a Java Virtual Machine implementation, standard Java class implementations, and a suite of Java-based components for accessing iSeries data and program resources. iSeries also offers the WebSphere application development and run-time environment, which supports the Java standard.

► **Integrated Language Environment (ILE)**: Supports a number of programming technologies (RPG, COBOL, and C/C++).

► **OS/400 PASE**: A run-time environment for running AIX applications on iSeries.

► **Linux on iSeries**: Enables a native PowerPC Linux kernel to run in a secondary logical partition using either a shared or dedicated processor of OS/400. This is an additional server consolidation opportunity for iSeries.

► **Integrated xSeries Server** and **Integrated xSeries Adapter**: Make it possible to run and manage multiple Windows NT or Windows 2000 servers from a central iSeries server. This significantly simplifies management of those environments.

► **Lotus Domino**: Provides a scalable, reliable infrastructure as the basis for communication, decision making, and document management.

---

**Note**: This redbook does not cover the older versions 1.1, 2.0.2, 2.0.3.1, and 3.0.2 of WebSphere Application Server for iSeries, since most of the products described in the following chapters only support WebSphere Application Server Version 3.5.2 or later. If an older version is already installed on your AS/400 or iSeries server, make sure that you either completely uninstall it (if it is not used) or migrate it before you use a later version.
The iSeries aggressively supports the transformation of business applications to an e-business model, while minimizing disruption within the enterprise environment. It has business-proven values (reliability, security, scalability, and low cost of ownership) and supports the latest enabling technologies for e-business. In combination, these two qualities make the iSeries an excellent choice for extending existing applications and deploying new solutions.

Our focus, then, is on using different ways to transform an existing iSeries host business application with 5250 interfaces into a modern, distributed, thin-client application that can be integrated with new e-business applications.

---

**Figure 1-3** iSeries: The flexible server

The iSeries aggressively supports the transformation of business applications to an e-business model, while minimizing disruption within the enterprise environment. It has business-proven values (reliability, security, scalability, and low cost of ownership) and supports the latest enabling technologies for e-business. In combination, these two qualities make the iSeries an excellent choice for extending existing applications and deploying new solutions.

Our focus, then, is on using different ways to transform an existing iSeries host business application with 5250 interfaces into a modern, distributed, thin-client application that can be integrated with new e-business applications.
1.5 The IBM suite of host/server access products

IBM has continually provided many end-user-, server-access-related product choices for its customers. Recently it extended its product choices to include more Web-to-host e-business solutions. Some of these product solutions may appear to overlap with other IBM products. However, each solution has a different focus. Let's first review the focus of the products available from IBM.

As you read through the following sections, refer to Figure 1-4 to help you see the big picture of how all these products inter-relate. For example, look at the very bottom of Figure 1-4 to help you decide which product to use depending on the level of control you have over the client environment. WebSphere Application Server is required for many of these products, which brings a certain level of independence from the client, but raises the minimum system size (CPW and memory) for your iSeries server.

Figure 1-4  Comparison of IBM host/server access products
1.5.1 Methods to combine new Web applications with existing applications

In theory, the most modern, cleanest, and conceptually simplest approach to build Web application is to design a complete new application using object-oriented technologies and methods. However, in most cases, there are many reasons why you would not want to start completely from scratch:

- Developing a completely new application requires too much time and resources.
- Application developers do not have sufficient background with object-oriented design.
- There is a need to continue using the existing applications.

What are your options to integrate existing applications? If your application is sufficiently modularized, you may develop the browser interface and use either CGI programs (see “The Common Gateway Interface (CGI)” on page 5) or Java servlets (see “Using servlets to run Java code on the Web server” on page 12) to call the existing application modules and bypass the application’s user interface. While you can reuse your current applications, this option still requires the application developers to master the required Web technologies.

Another option is to use development tools that allow you to integrate the existing 5250-based screens with an HTML-based browser interface to minimize the changes within the application programs. To do so, you can choose from several different tools and methodologies ranging from:

- **Host On-Demand**: Provides the same look and feel as a 5250 terminal or emulator
- **WebFacing Tool**: Presents the same data together with GUI controls, graphics, and links
- **Host Publisher**: Allows you to completely decouple the 5250- (or 3270) layout from the browser, yet using the legacy screens as a data source

The following sections describe these tools in further detail.

**Web-to-host integration positioning**

IBM has aligned its strategic Web-to-host e-business solutions on a common technological foundation, centered around Java technology and IBM Software Strategy for e-business. The transition for an enterprise to go to a Web-based solution for applications can be divided into three stages of development as shown in Figure 1-5. IBM has products to provide the solution for each step of the way.
The stages in Figure 1-5 are explained in the following list:

- **Stage 1** involves implementing Web-to-host application integration solutions that open applications first to intranets, then to extranets, and last to the Internet, without requiring changes to the host applications. WebSphere host integration products and WebSphere Development Studio for iSeries provide the quickest way to Web-enable existing host applications requiring little or no new business logic. They provide solutions for different needs and complement each other to provide a total solution. The products include:
  - Host On-Demand
  - Host On-Demand with Screen Customizer
  - WebFacing Tool
  - Host Publisher
  - Web Interaction wizard (WebSphere Studio for iSeries)

- **Stage 2** begins by adding new business logic to existing Web-to-host application implementations. These include:
  - **Host Publisher**: These applications can be enhanced using such products as WebSphere Studio Site Developer Advanced.
  - **Web Interaction wizard**: These applications can be enhanced using such products as WebSphere Development Studio Client for iSeries (Development Studio Client).

- **Stage 3** involves developing new e-business applications. WebSphere Application Server and WebSphere Studio family of products provide a Java infrastructure for the development and execution of Java applications and servlets. It focuses on adding new business logic to existing applications or deploying totally new Web applications.

The staging of the transformation to e-business delivers a leading edge solution for both the integration of existing applications and the deployment of new applications on the Internet.
Bringing host applications to the Web without programming
This section discusses options that allow you to provide a browser interface to your existing applications without doing any kind of programming or application development activities.

IBM Host Integration family of products
The WebSphere Host Integration products support multiple host application environments, including the IBM iSeries, zSeries, and pSeries environments. The Host Integration product set focuses on:

- Providing the same end-user look and feel whether running on a Windows 32-bit operating system or another platform.
  
  Some examples of this are that IBM Personal Communications runs on Windows 32-bit systems, OS/2, and Windows 3.x. The Host On-Demand product runs in many additional desktop environments such as Linux, Sun Solaris, and so on.

- Providing a common and consistent interface to a variety of servers and host systems.

This family of products provides many tools to enable access to applications and data that reside on midrange servers, enterprise servers, and ASCII hosts. For example, they provide Host Access Beans for Java and the Java Interface for Host Access Class Library (HACL) that can be used to provide an enhanced user interface to existing back-end applications.

IBM has two end-user products that are designed to transform host applications into Web-based applications:

- WebSphere Host On-Demand (previously called IBM SecureWay Host On-Demand): This product runs on servers with JVM 1.1 or higher, and its applet can be downloaded to browsers with JVM 1.1 (such as Netscape or Internet Explorer). It is a browser-based 3270, 5250, and VT emulation and is primarily designed to meet the needs of intranet and extranet users.

  Host On-Demand is for users who are familiar with the original host application screens. It is also for users who are considered power users and require a full function customizable emulator. Host On-Demand is a good alternative when a user needs extended connection times. It also provides host-to-client file transfer and local print capability.

  \[Note: \] As opposed to Web applications created with WebSphere Host Publisher or the WebFacing Tool, Host On-Demand does not use any of the advanced graphical user interfaces, such as pull-down menus, images, links, etc., unless you enhance the user interface with the IBM Screen Customizer. It looks and behaves more like a real 5250 display station or emulation. It does not have the look and feel of a typical Web site, although it has to be started and run under the control of a Web browser.

  For more information on Host On-Demand, see 1.6.1, “Host On-Demand” on page 29. Also see the following resources:
  - IBM Web-to-Host Integration Solutions, SG24-5237 (Chapter 2)
  - IBM WebSphere Host On-Demand: Version 5 Enhancements, SG24-5989 (Chapter 12)

- iSeries Access for Web: This is a servlet that runs on OS/400 JVM and generates HTML output to a browser (such as Netscape or Internet Explorer). iSeries Access for Web provides a subset of the capabilities provided in Client Access Express for Windows or
IBM Host On-Demand. Its advantage is ease of deployment (no code to install or maintain on the end-user workstation).

iSeries Access for Web is designed for users who need to access the 5250 application on iSeries servers, access the DB2 UDB for iSeries data, work with OS/400 printers or printer output, and send and receive messages. Since 28 September 2001, additional function was added to this product and delivered as part of the iSeries Client Access Family product.

For more information on iSeries Access for Web, see:


Programming tools to bring 5250 or 3270 applications to the Web

IBM offers a variety of tools to enable host applications to run in a Web environment. The tools described in the following sections allow you to go beyond a one-to-one conversion and enhance the new Web application by taking advantage of the modern capabilities of Web browsers:

► IBM WebSphere Host Publisher: Runs on the OS/400, OS/390, AIX, Sun Solaris, and Microsoft Windows NT operating environment. It enables applications created with its Host Publisher studio to run unchanged in a WebSphere Application Server environment. You can externalize selected portions of an application to the Web, as well as consolidate pieces of multiple host applications into a single HTML page. The users of the application built with Host Publisher typically connect periodically for short periods of time and expect typical Web response times. This solution requires both a development and run-time investment. No source code is required.

This product is included in iSeries Client Access Family (5722-XW1) and in the WebSphere Host Integration Solution.

For more information, see 1.6.3, “Host Publisher” on page 32. You can also refer to the WebSphere Host Publisher site at:

http://www.ibm.com/software/webservers/hostpublisher

► WebSphere Transcoding Publisher: Enables customers to run their existing Web applications from hand-held information devices. It brings legacy data from the Web and dynamically converts formats and the presentation style of host data to a new breed of personal data assistants. WebSphere Transcoding Publisher runs on AIX, Linux, OS/400, Solaris, Windows 2000, and Windows NT. IBM WebSphere Transcoding Publisher V3.5, 11K8034/11K8036 is a stand-alone product.

For details, go to:

http://www.ibm.com/software/webservers/transcoding/

► IBM Screen Customizer: Converts host screens into graphical presentations. This solution requires a development investment, but does not require access to source code. This product is included in Host Access Client Package and WebSphere Host Integration Solution. It works through an emulator, such as the ones included in Personal Communications, Host On-Demand, and V4R5M0 and V4R4M0 Client Access Express (not available for V5R1M0 Express).

For more information, see the following references:

► IBM Web-to-Host Integration Solutions, SG24-5237 (Chapter 4)
► IBM WebSphere Host On-Demand: Version 5 Enhancements, SG24-5989 (Chapter 14)
► Screen Customizer Web site at:

http://www.ibm.com/software/network/screencustomizer
WebFacing Tool: Converts a 5250 host application into a Web application with only minor or no changes required to the host application source code. After using the WebFacing Tool to create a user interface for a Web browser, the same application can then support both the standard 5250 interface and the new Web GUI at the same time. This allows existing and power users to continue accessing their applications in the traditional manner and still provide a Web interface for casual and new users based on the same application.

This tool allows iSeries server developers to extend existing applications to the Web using their current skills. Some customization capabilities of the WebFacing Tool exist to enhance the Web interface, without needing to know Java, JavaScript, or even HTML. These capabilities include the possibility to define your own WebFacing style by using the WebFacing perspective of the WebSphere Studio Site Developer Advanced.

WebFacing Tool is an 5250-only solution and requires access to the Data Description Specifications (DDS) source code (as opposed to Host Publisher, which can also be used with 3270 or VT100 applications without the need of even seeing the source of the display panels). As opposed to Host On-Demand, WebFacing requires a development time investment.

The WebFacing Tool is not a product in itself. Instead, it is included with WebSphere Development Studio Client for iSeries, which comes as part of WebSphere Development Studio for iSeries (5722-WDS).

For more information about the WebFacing Tool, see 1.6.2, “WebFacing Tool” on page 31, and Chapter 10, “WebFacing Tool” on page 179. Also refer to the WebSphere Development Studio Client for iSeries Web page at:

http://ibm.com/software/ad/wdt400/webfacing/

See Chapter 2, “Creating applications for iSeries: A new set of tools” on page 39, for an in-depth description of WebSphere Development Studio for iSeries (5722-WDS) including WebSphere Studio Site Developer Advanced. You should also see the WebSphere Development Studio Client for iSeries Web site:

http://ibm.com/software/ad/wdt400

1.6 Web-to-host integration tools comparison

This section discusses more in depth four different IBM software components available to access the iSeries server applications from the client with a Web browser.

1.6.1 Host On-Demand

IBM WebSphere Host On-Demand is an IBM product that gives Web users access to host programs through a Web browser. It also includes a set of self-contained and portable core host access reusable components for direct manipulation of host application data. Figure 1-6 shows an overview of Host On-Demand.
Browser access to host systems is provided through the use of Java applets downloaded into the client Web browsers, although the applets can also be installed as stand-alone clients.

Database On-Demand, included with Host On-Demand, provides access to DB2 information stored on iSeries servers using a JDBC driver. It allows users to access iSeries databases by running SQL statements online. It uses the OS/400 JDBC client that has been integrated into Host On-Demand.

Host On-Demand includes an application development environment that contains Host Access Beans for Java, Host Access Class Library for Java, Host Access Controls for ActiveX, ECLApplets, and Open Host Interface Objects.

For more information about Host On-Demand, refer to these sources:

- Chapter 2 in IBM Web-to-Host Integration Solutions, SG24-5237
- IBM WebSphere Host On-Demand: Version 5 Enhancements, SG24-5989

**Host machine requirements**
The requirements for the host machine are:

- OS/400 V4R3 or higher
  
  Recent cumulative service is recommended. See also iSeries fixes, downloads, and updates on the Web at:
  
  http://as400service.rochester.ibm.com/

- IBM HTTP Server for iSeries (5722-DG1 for OS/400 V5R1, 5769-DG1 for previous releases)

- Developer Kit for Java (5722-JV1 for OS/400 V5R1, 5769-JV1 for previous releases)

- IBM Toolbox for Java (5722-JC1 for OS/400 V5R1, 5769-JC1 for previous releases)

- Qshell Interpreter (5722-SS1 for OS/400 V5R1; 5769-SS1 for previous releases): Option 30 is recommended

- TCP/IP Connectivity Utilities for iSeries (5722-TC1 for OS/400 V5R1, 5769-TC1 for previous releases (installed and configured))
Client machine requirements
The requirements for the client machine are:

- Windows NT (with Service Pack 5) or Windows 2000
- A Web browser:
  - Netscape Navigator 4.6 or higher
  - Microsoft Internet Explorer with IBM SP1 5.0 or higher

1.6.2 WebFacing Tool

The WebFacing Tool is not a product in itself, but rather is included with WebSphere Development Studio Client for iSeries, which comes as part of WebSphere Development Studio for iSeries (5722-WDS). You can use the WebFacing Tool with applications where DDS source code was used to create 5250 display screens.

The tool has user-friendly wizards that facilitate selecting your original application's DDS source, converting the source, and deploying the new browser-based interface to your program as a Web application. The conversion creates JavaServer Pages and Java Beans that substitute for your DDS code and make Web access possible. After your DDS code is converted, you can access the application through a browser or continue to use 5250 displays.

Basing the interface of your applications on JavaServer Pages allows for more flexibility in customizing its appearance. Before your DDS code is converted, you can use the Style wizard to change the look and feel of the pages that will be generated for you. Styles allow you to define attributes in your Web pages such as graphics, fonts, colors, and layouts. You can use one of the supplied styles or create your own. If you want to update the appearance of a previously converted project, simply run the WebFacing Tool again and select a new style.

When using the WebFacing Tool to convert 5250 interfaces to Web-based interfaces, the original application program remains the same. All the maintenance to that program is done on the host code. That is, you have to use the enhancements and technologies provided for the host language used.

Host machine requirements
To deploy and run your Web applications, you must ensure the following requirements are installed and configured on your host machine:

- OS/400 V4R5 or later
- WebSphere Application Server 3.5.2 or 4.0: Either the Standard or Advanced versions can be used. Support for Tomcat is planned to be available for OS/400 V5R2.

Note: You can only order WebSphere Application Server Version 3.5 or Version 4.0. You must install the group PTFs to upgrade it to Version 3.5.2 respectively or 4.0.1 or higher. See also Table 1-1 on page 21. For the latest PTFs available, go to:


- IBM HTTP Server (original or powered by Apache, 5769-DG1 for V4R5, and 5722-DG1 for V5R1)
- TCP/IP Connectivity Utilities (5769-TC1 for V4R5 and 5722-TC1 for V5R1)
- You must enable file sharing via NetServer so your Web application contents can be moved to the iSeries server (or publish via FTP).
Client machine requirements
Before you can use the WebFacing Tool to develop Web applications, you must ensure the following requirements are set up on your client machine:

- Windows 98, NT, ME, or 2000
- A Web browser, as follows:
  - Internet Explorer V5.0: For WebFacing Tool run-time development
  - Internet Explorer V5.0 or Netscape V4.7: For viewing online help in the development environment

1.6.3 Host Publisher
IBM WebSphere Host Publisher is a part of the IBM Host Integration software. Host Publisher enables enterprises to jump-start e-business applications from existing host applications and data. It does this using industry-standard HTML Web pages to support end users running non-Java browsers.

Host Publisher is Java-based and supports AIX, OS/390, Windows NT, OS/400, NetWare, and Sun Solaris operating systems.

There are two major components in IBM WebSphere Host Publisher:

- **Host Publisher Studio**: Provides easy-to-use tools to assist in the creation of Web-to-host integration projects.
- **Host Publisher Server**: Consists of IBM WebSphere Application Server and Host Publisher run-time components. It provides the run-time environment for executing Web applications created with the Host Publisher Studio.

Host Publisher Studio tools create Integration Objects that encapsulate the interaction and data retrieval with host applications. It also generates fully customizable HTML output with embedded JavaServer Pages tags to invoke the Integration Objects.

Host Publisher provides you with host access capabilities without needing to build Java applications and applets yourself. You can enhance the applications built by Host Publisher, for example, by adding graphics to the pages.

Host Publisher applications run on the server, unlike Java applets. The information is served to the clients as HTML. Because of this, there are no specific requirements placed on the client Web browsers, and therefore, it is appropriate for Internet, intranet, and extranet use.

Host machine requirements
The following software is required to be installed and configured on your iSeries server:

- OS/400 V4R4 or later
- WebSphere Application Server V3.5 (or later) Standard or Advanced Edition
- IBM HTTP Server (5769-DG1 for V4R5, 5722-DG1 for V5R1)
- JDK 1.1.8 (5769-JV1 for V4R5, 5722-JV1 for V5R1 Option 4 (JDK 1.2.2 is installed with Host Publisher Studio))
Client machine requirements
Before you can use the Host Publisher Studio to develop Web applications, you must ensure you set up your client machine with:

- Windows 95, 98, NT 4.0 (with Service Pack 4 or later), or 2000
- A Web browser

Host Publisher versus WebFacing Tool
At first glance, Host Publisher and WebFacing Tool have a similar approach in providing a Web browser user interface to the existing interactive applications for character-based terminals. However, there are indeed important differences between these two tools. The following sections discuss the most important differentiators.

WebFacing Tool needs DDS source code
Both tools interface with existing applications that do input and output operations from and to character-based display stations. Also both products allow the existing applications to remain unchanged, and users can continue to access them from their terminal or display station emulation.

Host Publisher uses an approach where a virtual image of each display format is accessed by Java Beans, so-called Host Access Objects, created by Host Publisher. These Java Beans can recognize the type of panel by identifying a character string at a certain position (as defined by the Host Publisher application developer) and return a subset of the output data to the browser via a JavaServer Page. The Java Bean can also accept data which was entered into an HTML form by the browser user and input it into the display panel, simulating a display terminal user.

You look at this as a 5250 (or 3270, or VT100) emulation program that actually runs on a server – the Host Publisher server. However, the actual output of the emulation program is never shown on any user’s display and only a buffer area is accessed by a Java Bean. Likewise, there is no real keyboard attached to the emulation program, but keystrokes are generated by a Java Bean.

The server part of the WebFacing Tool is directly hooked into the OS/400 workstation data management at the point where the variable output data from the application program is merged with the static information, for example, constant text, screen position, and appearance of the data from the display file (DSPF) object. Before the 5250 data stream is even generated, WebFacing presents the data to the Java Bean, which then returns it to the Web browser in HTML format. Likewise input data from the browser is delivered to Workstation Data Management as if it comes from a real or emulated 5250 device.

To create an application with the WebFacing Tool, you need to have access to the source members containing the Data Description Specifications for each single display format. If your application window includes a panel for which you do not have (or did not convert) a DDS source, the WebFacing application aborts with an error message. Such examples may include:

- Applications where you do not have the source code
- Function within your applications that presents an OS/400 screen, for example the Work with Spool Files (WRKSPLF) command
- Panel generated by the User Interface Manager (UIM) other than UIM help panels

Host Publisher in turn can work with any kind of display panels, regardless of whether the source code exists.
Host Publisher works with 5250, 3270, and other types of terminals
Because of its dependence on DDS, WebFacing can only work with dialog programs written for OS/400 (or System/38). Applications on a /390 system written for 3270 terminal or for VT100 terminals cannot be converted with the WebFacing Tool, but you can Web-enable them with Host Publisher.

WebFacing can create functioning code fully automatically
Provided no unsupported keywords are used for the DDS, the WebFacing Tool creates all necessary objects (HTML, JSPs, Java Beans, and XML files with configuration information) for you. After deploying the objects to an application server, they function immediately with a graphical interface (based on one of the templates provided with the product). However, the WebFacing application can be enhanced later with CODE Designer (see “Using CODE Designer Web Settings with your DDS source” on page 138) or by modifying a WebFacing Style.

In contrast, creating a Host Publisher application is not an automatic process. You need to go through a process called Macro Recording, where you literally have to “teach” the Host Integration Object (a Java Bean that is created during this process) how a terminal user would run the application. Although it is not extremely difficult to record the macros for the host integration object, it is time consuming and requires some learning.

After you create your host integration objects, you need to create two JSPs that interface with the Java Beans.

You can see an example that describes the necessary steps for creating a Web application with Host Publisher at the following Web site:

http://ibm.com/iseries/developer/java/topics/hostpubadapter.html

Host Publisher allows direct access to DB2 data
In addition to accessing one or more host applications, Host Publisher can also access DB2 data directly. In contrast, WebFacing always relies on accessing the data presented on the 5250 screen by the interactive application.

Combining the contents of multiple panels
For WebFacing, there is a one-to-one relationship between the 5250 screens and the panels presented to the browser. The application developer can hide certain parts of the 5250 screen for the Web user by enhancing the WebFacing application with CODE Designer (see “Using CODE Designer Web Settings with your DDS source” on page 138). However, the browser user needs to submit an HTML form for each 5250 screen that appears in the application window.

A Host Publisher application can be written in such a way that the information from multiple host panels (or even multiple applications on the same or a different host) can be presented on a single HTML form. It allows you to combine more than one host application to a single Web application, so that the user is not aware of this fact. With Host Publisher, the application developer decides which output data field should be presented to the Web user.

By default, WebFacing presents the contents of all output fields to the browser user. However, you can use CODE Designer to hide selected fields for the Web (see “Using CODE Designer Web Settings with your DDS source” on page 138).

Prerequisite application server
The version of Host Publisher strongly depends on the correct version of WebSphere Application Server. For example, Host Publisher 3.5 can only run under WebSphere Application Server Version 3.5 and Host Publisher 4 needs WebSphere Application Server Version 4 as a prerequisite.
Web applications created with WebFacing can run under WebSphere Application Server Version 3.5.2 or later. WebFacing applications can run under OS/400 V4R5 or later. You also need the correct OS/400 PTFs depending on which Service Pack of the WebSphere Development Studio for iSeries (5722-WDS) you are using. See the WebSphere Development Tools System requirements Web page for most recent information:

http://ibm.com/software/ad/wdt400/about/sysreq.html

**Supported browsers**
Applications created with the WebFacing Tool currently require the Microsoft Internet Explorer 5.0 or higher and cannot run on Netscape.

**Product packaging**
Initially called IBM SecureWay Host Publisher, the product is now available under the name WebSphere Host Publisher V3.5 (5648-E25) and WebSphere Host Publisher V4.0 (5724-B81). You can purchase it as part of the *Host Integration Solution* together with Host On-Demand and Screen Customizer.

Since October 2001, Host Publisher also comes as a part of the *Client Access Family for iSeries*. For OS/400 V5R2, it comes as part of the *iSeries Access Family (5722-XW1)*.

The WebFacing Tool is not sold as a separate product. It is part of the *WebSphere Development Tools for iSeries*, which is sold as separate product for OS/400 V4R5 and is now part of the WebSphere Development Studio for iSeries (5722-WDS). For OS/400 V5R2, WebFacing is now a component of the *WebSphere Development Studio Client for iSeries*, which replaces WebSphere Development Tools as a part of 5722-WDS. See Chapter 2, “Creating applications for iSeries: A new set of tools” on page 39, for more information on the WebSphere Development Studio Client for iSeries.

**More information**
You can find additional information about WebSphere Host Publisher on the Web at:

http://www.ibm.com/software/webservers/hostpublisher/

### 1.6.4 Web Interaction wizard

The Web Interaction wizard is a component of Developer Studio Client. This wizard is used to create the interaction between the Web pages and the ILE programs on the iSeries server. It is used to define input, output, and error Web pages and to identify the ILE program to call for a particular interaction. It allows the user to identify the parameters that will be passed to the ILE program and to map the input and output fields from the Web pages to the ILE program parameters.

A great advantage of using the Web Interaction wizard is that you can use the new Web technologies when maintaining the ILE programs containing your business logic.

**Host machine requirements**
The following software is required to be installed and configured on your host machine:

- OS/400 V4R4 or later
- WebSphere Application Server Version 3.5 or later for both development time and run time
- IBM HTTP Server (5769-DG1 for V4R5, and 5722-DG1 for V5R1)

**Client machine requirements**
Before you can use Developer Studio Client to develop Web applications, you must be sure to set up your client machine with:
1.6.5 Comparison table

Table 1-2 compares the solutions that are discussed in this section for converting existing iSeries host applications to Web-based applications.

Note: WebSphere Studio Site Developer Advanced is part of WebSphere Development Studio Client for iSeries. The WebFacing Tool is integrated into WebSphere Studio Site Developer Advanced.

<table>
<thead>
<tr>
<th>Host On-Demand</th>
<th>WebFacing Tool</th>
<th>Host Publisher</th>
<th>WebSphere Studio Site Developer Advanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presentation logic implemented on the server or the browser</td>
<td>Applets running on the browser</td>
<td>Servlets on server - Servlets on server - JavaScript on the browser</td>
<td>Depends on the implementation, but mostly servlets running on the application server</td>
</tr>
<tr>
<td>Needs application server</td>
<td>None</td>
<td>WebSphere Application Server (3.5.2 or later) or Tomcat</td>
<td>Versions for WebSphere Application Server and Host Publisher are highly dependent WebSphere Application Server</td>
</tr>
<tr>
<td>Enhances the appearance beyond the generated Web interface</td>
<td>With IBM Screen Customizer</td>
<td>Using WebFacing perspective or CODE Designer</td>
<td>Need JSP, JavaScript, or Java knowledge Using the Workbench with JSP and Java knowledge</td>
</tr>
<tr>
<td>Concurrent use of 5250 and Web interfaces</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes Needs implementation in host application</td>
</tr>
<tr>
<td>Application logic integration with other Web applications</td>
<td>Not easily possible</td>
<td>Only when modifying the generated HTML and JSPs</td>
<td>Through the use of the Integration Objects Possible through the use of the application logic on the client</td>
</tr>
<tr>
<td>Access to different database sources</td>
<td>No</td>
<td>Only as defined by the host application logic</td>
<td>Through the use of Host Publisher data integration objects Possible through the use of the application logic on the client</td>
</tr>
<tr>
<td>Programming effort to convert the host interface to a Web interface</td>
<td>None</td>
<td>None, unless DDS keywords are not supported by WebFacing</td>
<td>Time consuming to build integration objects and JSPs with Host Publisher Studio. Need Java, JavaScript, HTML to create advanced GUIs. Use the Web Interaction wizard to create Web pages</td>
</tr>
<tr>
<td>Supported host interfaces</td>
<td>- 3270 - 5250 - VT52 - VT100 - VT220</td>
<td>5250, DDS source needs to be available for each display format</td>
<td>- 3270 - 5250 - VT52 - VT100 - VT220 - DB2 - JDBC interface - Java applications</td>
</tr>
<tr>
<td>Host On-Demand</td>
<td>WebFacing Tool</td>
<td>Host Publisher</td>
<td>WebSphere Studio Site Developer Advanced</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------------------</td>
<td>---------------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>Presentation</td>
<td>HTML with applets, but same look and feel as an emulator</td>
<td>HTML, JSP, JavaScript</td>
<td>HTML, JSP, JavaScript</td>
</tr>
<tr>
<td>Supported browsers</td>
<td>Netscape 4.x or IE 5.x or above</td>
<td>MS Internet Explorer 5.0 or later</td>
<td>Netscape 4.x or IE 5.x or above</td>
</tr>
</tbody>
</table>
Creating applications for iSeries: A new set of tools

This chapter describes the new WebSphere Development Studio Client for iSeries and how it evolved from WebSphere Development Tools as part of WebSphere Development Studio for iSeries (5722-WDS). The chapter also highlights WebSphere Studio Site Developer Advanced, a member of the WebSphere Studio family, which comes as one component of WebSphere Development Studio Client for iSeries (Development Studio Client). Plus, this chapter looks at WebSphere Workbench, which resulted from the Eclipse project (see 3.1, “Eclipse: The start of a new era of tools” on page 56, for more on Eclipse).
2.1 IBM WebSphere Development Studio for iSeries

For OS/400 Version 5 Release 1, most of the previous separately sold program products for application development were bundled together into a single product called WebSphere Development Studio for iSeries (5722-WDS). Figure 2-1 shows the various WebSphere-related products as they apply to the iSeries server.

IBM WebSphere Development Studio for iSeries gives customers and solution providers rapid and cost-effective ways to increase the number of Web-enabled, e-business applications for the iSeries server. It marks a major shift in the packaging of IBM application development tools for the iSeries server. Rather than continuing to sell individual tools and compilers, this package consolidates all the key iSeries development tools, both host and workstation, traditional and new, into one attractively-priced iSeries offering. This product is a consolidated set of host and workstation tools optimized for a complete application development solution built on the iSeries server.

During the year 2001, the products WebSphere Studio and VisualAge for Java were merged together with a new application development workbench based on the Eclipse project. The new product is called WebSphere Studio Application Developer\(^1\).

Similarly, with OS/400 V5R2, WebSphere Development Tools for iSeries 5.1 have also been restructured and renamed to WebSphere Development Studio Client for iSeries 4.0. We discuss this in depth in 2.2, “WebSphere Development Studio Client for iSeries” on page 43. You can also refer to the WebSphere Development Studio Client for iSeries Web site at:

http://ibm.com/software/ad/wdt400/

\(^{1}\) At this time, there are several development tools based on the Eclipse platform.
WebSphere Development Studio Client for iSeries consolidates the key application development tools into one client-based package. The package includes the following workstation components:

- **WebSphere Studio Site Developer Advanced**, which replaces the former VisualAge for Java and WebSphere Studio for iSeries, includes the WebFacing Tool and other new functions
- **CODE**, which was formerly called CODE/400 (see Chapter 8, “Working with CODE” on page 127 for more details)
- **VisualAge RPG** (see Chapter 9, “Introduction to VisualAge RPG” on page 157, for details)
- **Distributed Debugger**

You can quickly develop and deploy traditional and e-business applications on your iSeries server with IBM WebSphere Development Studio Client for iSeries. This powerful suite of tools represents the next generation of WebSphere development tools for iSeries. It is the client component of the IBM WebSphere Development Studio product, which contains all of the host ILE compilers for RPG, C, C++, COBOL, and the Application Development Toolset. This client component of the overall package is designed to help you accomplish three primary programming goals:

- **Develop and maintain iSeries business logic**: You have several workstation components that help you to develop and maintain the ILE programs: CODE, Remote Systems Explorer (part of WebSphere Studio Site Developer Advanced), iSeries exploration, and command execution.
- **Create Web front ends to iSeries business logic**: Development Studio Client gives application developers, with traditional iSeries programming skills, the ability to quickly develop e-business applications without having to learn Java and other Web application skills. It provides a set of wizards, as part of the WebSphere Studio client software, that takes you through the steps required to create Web input and output pages.
- **Create GUI front ends to iSeries business logic**: With VisualAge RPG, you can develop and maintain client/server applications in a visual development environment on the workstation. You leverage your existing RPG skills to create graphical user interface (GUI) applications that you can deploy as native Windows applications, or as Java applets that can run on any Java-capable Web browser. These applications can access iSeries data and other iSeries objects.

The following sections briefly introduce the WebSphere Development Studio host components.

### 2.1.1 ILE RPG

ILE RPG is the compiler for RPG IV, the latest generation of RPG, enriched with the new functions you have wanted. With ILE, you have the power of a suite of compilers for multilanguage interoperability and faster call performance. Your investment in RPG applications is protected with full upward compatibility. A conversion aid helps you easily convert your RPG III applications to RPG IV language definition.

To learn about the V5R1 enhancements made for RPG IV, go to:


---

2 This product is based on the Eclipse platform.
You can also refer to the following IBM Redbooks about RPG programming on the iSeries:

- *Moving to Integrated Language Environment for RPG IV*, GG24-4358
- *AS/400 Programming with VisualAge for RPG*, SG24-2222

### 2.1.2 ILE C/C++

The ILE C/C++ Compiler supports program development on iSeries servers in both the C and C++ programming languages. The C compiler is a high-performance, high-function compiler for C development on the iSeries platform. It gives you the combination of function-rich C and the power of ILE. ILE offers fast calls and a uniform run-time model, which are definite benefits for call-intensive C.

With the C++ Compiler, you can develop object-oriented, mission-critical applications for the client, iSeries server, or both in a client/server solution. It is a single, powerful object-oriented C++ programming development environment that integrates cooperative processing with graphical user interface development.

To learn about the V5R1 enhancements made for ILE C/C++, go to:

http://www.ibm.com/software/ad/wds400/about/ile_ccpp.html

### 2.1.3 ILE COBOL

With the ANSI-85 functions of ILE COBOL, you can easily port code from other platforms to the iSeries server. Your investment in applications is protected. You can be confident that IBM will continue to support COBOL on iSeries so you can still use applications written in COBOL.

To learn more about the V5R1 enhancements made for COBOL, see the WebSphere Development Studio for iSeries Web site at:

http://www.ibm.com/software/ad/wds400/about/ile_cobol.html

### 2.1.4 Application Development ToolSet (ADTS)

ADTS for iSeries is a well-integrated set of host-based development tools you can use with confidence as you build your applications. It combines established tools, like Programming Development Manager (PDM), Screen Design Aid (SDA), and Source Entry Utility (SEU), with new utilities. There are two optional features in the package that are especially well suited for helping you to plan, manage, and develop applications in a multi-developer organization.

Here is a list of the ADTS components:

- Source Entry Utility (SEU)
- Screen Design Aid (SDA)
- Report Layout Utility (RLU)
- Data File Utility (DFU)
- Character Generator Utility (CGU)
- Advanced Printer Function (APF)
- Programming Development Manager (PDM)
- Host Access Programs for CODE and VisualAge RPG
- Interactive Source Debugger (ISDB)
- File Compare and Merge Utility (FCMU)
To learn more about ADTS, go to:
http://www.ibm.com/software/ad/wds400/about/adts.html

2.2 WebSphere Development Studio Client for iSeries

Figure 2-1 on page 40 describes the contents of WebSphere Development Studio for iSeries (5722-WDS). It initially included the WebSphere Development Tools 5.1, a rich set of Windows-based developer tools bundled together on a single DVD or five CD-ROMs. In addition to many iSeries- and OS/400-oriented tools, it contained such cross-platform products as VisualAge for Java and WebSphere Studio Professional Edition. They were enriched with plug-ins to support building applications for OS/400 and to integrate special object types that do not exist on other platforms.

In late 2001, the WebSphere Studio Professional Editions was enhanced with:

- An Integrated Java Development Environment, based on the Eclipse platform
- Relational Schema Center
- Pluggable repositories
- Site analysis tool
- J2EE 1.2 support

Putting all pieces together, a new product called WebSphere Studio Site Developer Advanced became available. It provides everything a Web developer needs to create, manage, and maintain dynamic Web applications. This easy-to-use developer solution is designed to meet the latest J2EE specifications and supports the latest Web standards including Java, JavaServer Page (JSP) components, HTML, DHTML, XML and Web services.

2.2.1 WebSphere Studio family

WebSphere Studio Site Developer Advanced is part of the WebSphere Studio family of products that were developed based on the Eclipse platform as a set of plug-ins that conform to the Workbench’s open standard APIs. These products are the follow-on technology for WebSphere Studio Advanced Edition V3 and VisualAge for Java Enterprise Edition V4.

The WebSphere Studio family includes:

- WebSphere Studio Site Developer Advanced
- WebSphere Studio Application Developer
- WebSphere Studio Application Developer Integration Edition
- WebSphere Studio Enterprise Developer

The products contain a different set of features as shown in Figure 2-2.
2.2.2 A new concept

Eclipse is an open source platform that was designed by IBM and released to the open source community. It is an open, portable, universal tooling platform that provides frameworks, services, and tools for building tools.

So why did IBM create Eclipse and donate it to the open source community? The reason was to ignite a new world of consistent application development tooling. This would allow each vendor to focus on their specific tooling without constantly reinventing the basic infrastructure that all application development tools need, such as project and team support, source editors, and debuggers. Think of Eclipse as the building of a new shopping mall. Eclipse is the building itself, while the vendors who write the tools are the individual stores. The more stores there are, and the more successful they are, the better the results are for everyone, especially the shoppers.
The Eclipse Software Development Kit (SDK) includes the Eclipse platform, a Java Development Tool (JDT), and the Plug-in Development Environment (PDE). The JDT and PDE are plug-ins to the platform. The Eclipse platform itself consists of several parts (see Figure 2-3):

- Platform run time
- Workbench that implements the graphical interface to Eclipse and its subcomponents JFace and the Standard Widget Toolkit (SWT)
- Workspace that acts as the repository for the projects
- Help system
- Version and Configuration Management (VCM) system

The Eclipse platform is built on a mechanism for discovering, integrating, and running plug-ins. A plug-in is the smallest unit of function that can be developed and delivered separately. Usually, a small tool is written as a single plug-in, whereas a complex tool has its functions split across several plug-ins. The JDT consists of five functional plug-ins and two documentation plug-ins.

The Eclipse consortium
The eclipse.org consortium was formed to deliver a new generation of application development tools. Industry leaders Borland, IBM, Merant, QNX Software Systems, Rational Software, RedHat, SuSE, TogetherSoft, and WebGain formed the initial eclipse.org board of directors and began work on the Eclipse open source project. You can find more details about this project on the Web at:

http://www.eclipse.org
All of the participating companies plan to release Eclipse platform compatible product offerings. In the Eclipse platform, code access and use is controlled through the Common Public License, which allows individuals to create derivative works that are royalty free and have worldwide re-distribution rights.

Eclipse is becoming a very attractive platform for tool developers. Independent Software Vendors (ISVs) can use the same APIs as IBM to create tools that will run on the Eclipse platform. It allows for a loose or tight integration of tooling with the Workbench and with other tools in the Workbench.

The Eclipse platform has become attractive to ISVs because it offers the infrastructure and integration points that allow all tools to be integrated into the Workbench. ISVs also like its seamless integration between the tools themselves.

ISVs can now integrate their tool into the environment using the same user interface as all the other tools use. That means that the way different objects are represented and maintained in the Workbench is consistent between different tools. This allows one tool to work with an object in the same way as another tool.

Another key integration point is the way objects are stored in the Workbench. Objects that are created and used by the tools are stored in the file system or in a repository using a common interface. This permits different tools to access objects in a consistent manner.

For more information, see Chapter 1 in the WebSphere Studio Application Developer Programming Guide, SG24-6585.

**WebSphere Studio Site Developer Advanced**

WebSphere Studio Site Developer is an integrated development environment (IDE) intended for Web developers who develop and manage complex Web sites. It is an easy-to-use toolset that minimizes the time and effort required to create, manage, and debug multiplatform Web sites. It is designed according to the J2SE and J2EE specifications and supports JavaServer Pages, servlets, HTML, JavaScript, and DHTML. It further includes tools for developing images and animated GIFs.

WebSphere Studio Site Developer enables Web developers to use their favorite content creation tools in conjunction with the built-in local and remote publishing capabilities. Using WebSphere Studio Site Developer, you can develop Web applications that use the following technologies:

- **JSPs**: A simple, fast, and consistent way to extend Web server functionality and create dynamic Web content. JSPs enable rapid development of Web applications that are server and platform-independent.
- **Servlets**: Server code that executes within a Web Application Server. WebSphere Studio Site Developer supports version 2.2 of the Servlet API.
- **Web services**: Self-contained, modular applications that can be described, published, located, and invoked over the Internet or within an intranet.

For more information, see the WebSphere Studio Site Developer Advanced Web site at: [http://www.ibm.com/software/ad/studiositedev/](http://www.ibm.com/software/ad/studiositedev/)

You should also refer to the WebSphere Studio Application Developer Programming Guide, SG24-6585.
2.2.3 Follow-on product for WebSphere Development Tools

The following components were included in WebSphere Development Tools for iSeries 5.1:

- WebFacing Tool (First Edition)
- WebSphere Studio, Professional Edition, with iSeries-unique features, which were sometimes called iSeries Affinity
- CoOperative Development Environment (CODE) previously called CODE/400
- VisualAge RPG

For a good place to start understanding the original contents of WebSphere Development Tools 5.1, see the following Redpapers:

- IBM WebSphere Development Tools for AS/400: An Introduction, REDP0503
- WebSphere Development Tools for iSeries: Generating Web Front Ends to Existing Applications, REDP0516

Consequently, after the appearance of the new WebSphere Studio Family, WebSphere Development Tools was restructured to include WebSphere Studio Site Developer Advanced in place of VisualAge for Java and WebSphere Studio Professional Edition. At the same time, the package was more tightly integrated with the other products of the WebSphere family. This was a reason to change the name to WebSphere Development Studio Client for iSeries Version 4.0.

Now there is the unusual situation that Version 4.0 is newer than Version 5.1. The reason is because WebSphere Development Studio Client for iSeries is now considered part of the WebSphere product family (which is currently at version 4), while WebSphere Development Tools was part of the OS/400 Version 5 products.

Figure 2-4 shows an overview of how WebSphere Development Tools was converted into the new WebSphere Development Studio Client for iSeries.
The CODE and VisualAge RPG components remained mainly unchanged. However, CODE can be launched from (but is not running under) the new WebSphere Studio Workbench Version 1.0, which was introduced with WebSphere Studio Site Developer. See 2.2.7, “CoOperative Development Environment (CODE)” on page 52, and 2.2.8, “VisualAge RPG” on page 52, for more information.

The other components of the product are implemented as plug-ins for the WebSphere Studio Workbench Version 1.0 – either as a cross-platform component of WebSphere Studio Site Developer or as an iSeries specific enhancement.

For an in-depth description of the cross-platform components, see WebSphere Studio Application Developer Programming Guide, SG24-6585.

Several iSeries-specific enhancements to WebSphere Studio and VisualAge for Java have already been implemented with WebSphere Development Tools. For example, the Enterprise Toolkit for iSeries (ET/400) contains special Java classes to access and develop iSeries applications. The iSeries Affinity component allowed you to develop iSeries components under WebSphere Studio.

All these iSeries extensions have also been integrated with WebSphere Studio Site Developer as part of Development Studio Client. The former iSeries Affinity is now called Web Tools - iSeries Extensions, where the Enterprise Toolkit for iSeries (ET/400) is now Java Tools - iSeries Extensions.

The WebFacing Tool already deployed the Eclipse-based workbench in WebSphere Development Tools Service Pack 3 and later. Therefore, you do not see a major change in the user interface when you migrate from WebSphere Development Tools 5.1 to Development Studio Client 4.0. However, since the WebFacing was constantly enhanced with each of the
five Service Packs during the year 2001, this effort continued when it was integrated with WebSphere Development Studio Client. It includes the ability to convert User Interface Manager (UIM) Help panels. See 2.2.4, “WebFacing Tool” on page 49, and Chapter 10, “WebFacing Tool” on page 179, for more information.

A new workbench perspective, the Remote Systems Explorer, is added to the environment, providing you with a variety of remote system tools. The term remote may be misleading here. It allows you, the application developer, to create and manage connections to one or more iSeries servers that you work with. This feature also allows multiple server connections to the same system, each with its own independent job environment. There is tight integration with the CODE Communications Console that allows CODE Editor, for example, to communicate through your new connections.


2.2.4 WebFacing Tool

The WebFacing Tool is introduced to help you add a Web browser interface to existing 5250 applications. The WebFacing Tool creates the Web interface by converting, at development time, the display file source into JavaServer Pages and Java databeans. These Java components are deployed to the WebSphere Application Server and, in combination with the WebFacing run time, provide a Web interface to the existing 5250 application.

No or little change is required to the application's business logic, and the same logic can support both a 5250 and Web interface. There is no dual maintenance. The conversion process is easy and cost-effective. There are no separate tool costs or run-time charges associated with the WebFacing Tool.

For more information about the WebFacing Tool, see 2.2.4, “WebFacing Tool” on page 49, and Chapter 10, “WebFacing Tool” on page 179. Also refer to the WebFacing Tool Web page at:

http://ibm.com/software/ad/wdt400/webfacing/

2.2.5 Web Tools - iSeries Extensions

WebSphere Studio is the premier IBM Web development tool. As described in 2.2.1, “WebSphere Studio family” on page 43, it has evolved from a single tool to a whole family of products tailored to different needs. One member of the family, the WebSphere Studio Site Developer Advanced, is included with WebSphere Development Studio Client for iSeries. But several iSeries related components have been added, such as display file-like Web palette parts including entry fields with validity checking and edit code support and subfile parts. They generate standard HTML and JavaScript.

There is also a wizard for generating a functional Web user interface to existing programs and service programs and a wizard to simplify publishing on the iSeries. It is easier to create Web applications that access iSeries data and applications with this version of WebSphere Studio than with any other Web development tool.

Web Tools - iSeries Extensions continue to provide all the functionality available in the classic WebSphere Studio for iSeries from within the Web Perspective of WebSphere Development Studio Client for iSeries. These are described in the following sections.
Design-time controls
Design-time controls (DTC) are Java Beans used within the Page Designer view. They are especially designed for use as visual Web front-end components to your iSeries business logic. DTCs are Web representations of I/O components in a display file. The following DTCs were ported from WebSphere Studio for iSeries:

- Check box
- Combo box
- Entryfield
- Label
- Listfile
- MLE
- Push button
- Radio button
- Subfile
- Table DTC

**Table400DTC**
Table DTC (Table400DTC) is new with WebSphere Development Studio Client for iSeries 4.0. It allows users to construct a table with multiple columns. It consists of input fields that can be mapped onto host program I/O parameters using the Web Interaction wizard.

The DTC is registered to the system and not the application. So it will be visible if any application tried to query what DTCs are available. Only the Eclipse version of the Web Interaction wizard is made to recognize this DTC.

**Web Interaction wizard**
The Web Interaction wizard guides you through a series of dialogs so you can bind your newly designed Web front end to your iSeries business logic.

**WebSphere Application Server for iSeries Deployment wizard**
This wizard allows the deployment of Web applications and WebFacing applications down to a WebSphere Application Server 4.0 for iSeries server.

The Deployment wizard uses the new Host Information Setup wizard, which is used to set up the following host-related run-time information for any servlets within the same Web project, for example, WebFacing:

- iSeries host name
- User ID
- Password
- Host program run-time library list

For WebSphere Development Tools 5.1, the iSeries-unique features of WebSphere Studio were previously called iSeries Affinity. Now with WebSphere Development Studio Client for iSeries, these are the Web Tools - iSeries Extensions. Refer to Chapter 11, “iSeries Web development tools in Development Studio Client” on page 207, for an in-depth discussion about this feature.
2.2.6 Java Tools - iSeries Extensions

The Java Perspective of the WebSphere Studio Workbench, delivered with WebSphere Studio Site Developer Advanced, replaces the functionality of VisualAge for Java that was part of WebSphere Development Tools 5.1. It contained iSeries-specific classes in a feature called Enterprise Toolkit for iSeries (ET/400). Since Java IDE is now integrated as Java Perspective in the workbench of WebSphere Studio Site Developer Advanced, the iSeries unique components were also moved to the workbench, now called Java Tools - iSeries Extensions. These include:

- Program Call wizard
- File Export/import wizards
- Remote Java Compile And Run
- Pre-supplied Java Beans
- Integration of the IBM Toolbox for Java into the WebSphere Workbench
- Java Tools for iSeries Examples

The following sections briefly explain these extensions.

Program Call wizard
This extension generates a Java Bean that contains the IBM Toolbox for Java code needed to call your iSeries programs from Java code. You can also use the output to create Web Services that call iSeries programs.

File Export/Import wizards
This extension provides the ability to import and export files to and from the integrated file system on an iSeries server into a Java Project.

Remote Java Compile and Run
This extension provides all the essential tools to remotely develop iSeries Java applications.

Pre-supplied Java Beans
The extension continues to provide useful Java Beans previously available in the ET/400 package. These beans include:

- Data File Utility beans: To extend the support of code to access one or more iSeries database files and to manipulate records within the files.
- Swing JFormatted bean: To let you convert iSeries fields and attributes and to provide edit code, edit word, formatting, and verification capabilities.
- Object List beans: To let you access iSeries object names (for example, libraries, files within a library, or user IDs on the system) and to allow you to set listing properties for selecting the desired type of object list.

Integration of the IBM Toolbox for Java into the WebSphere Workbench
This extension includes all help and examples associated with the Toolbox.

Java Tools for iSeries Examples
This extension provides a series of examples and samples for various tasks related to Java development for the iSeries using Java Tools. These examples are ported from the old ET/400 examples. Some new examples are created.
2.2.7 CoOperative Development Environment (CODE)

CODE is a client/server development environment for writing host or server code. It allows you to develop or maintain applications in ILE RPG, RPG/400, ILE COBOL, COBOL/400, ILE C, CL, ILE CL, DDS, and Java. It is an easy, cost-effective, and productive way to work with iSeries host applications on your Windows 98/ME/NT/2K/XP workstations.

You can use Client Access, Personal Communications, or TCP/IP to connect to the iSeries server. However, CODE doesn't require continuous connection to the iSeries. Many of the CODE features can function in a disconnected mode.

CODE includes a set of flexible workstation tools for developing and maintaining iSeries application programs. The workstation tools include the editor, screen, report and database designer, syntax checker, program verifier, project organizer, and a debugger front end. This combination of tools provides you with the productivity of the PC environment as well as the integrity of the iSeries server.

CODE includes the CODE Designer, a what you see is what you get (WYSIWYG) utility for designing display files and printer files. This new, powerful, and easy-to-use interface can make designing your DDS screens and printer files both fast and fun.

Another useful feature of CODE is the CODE Project Organizer. This utility allows you to access and manipulate iSeries objects with greater ease than PDM. With CODE Project Organizer, you can create a project and set up filters to gain quick GUI access to your frequently used OS/400 objects, members, application development manager projects, groups, and parts. Use the pop-up menus on these items to perform actions such as edit, compile, and debug. Use the CODE Actions window (notebook) to create and manage user-defined actions.

With Development Studio Client for iSeries, instead of using CODE Organizer, we recommend that you use the Remote Systems Explorer in the WebSphere Studio Workbench.

2.2.8 VisualAge RPG

VisualAge RPG is the premier IBM development tool, compiler, and run time for creating GUI applications that run on Windows or any GUI-capable client and provide seamless access to iSeries data and applications. The VisualAge RPG compiler is based on RPG IV.

VisualAge RPG features point-and-click simplicity for RPG programmers at the workstation. Use VisualAge RPG to capitalize on your existing RPG skills and start developing powerful e-business applications.

When built, your applications run on a workstation and can access iSeries host data and objects. VisualAge RPG integrated components allow application developers to preserve their current skills and easily develop OS/400 applications with graphical user interfaces. With VisualAge RPG, you can build an application from the top down. You start by focusing on the look and feel of the interface. Then, you tie all the parts together with workstation RPG logic that you write in the VisualAge RPG language. You can reuse RPG logic and display files (DSPFs) from an existing application.

You can now generate Java applications and Java applets from the VisualAge RPG source, making RPG a powerful e-business language. VisualAge RPG offers:

- Visual Development Environment
- RPG compiler on the workstation
- Programmable language-sensitive editor
2.3 More information

For more information, consult the article “What is the WebSphere Studio family of development tools?”, which you can find on the Web at:

New GUI framework

This chapter briefly introduces you to the new GUI framework called WebSphere Studio Workbench. WebSphere Development Studio Client for iSeries (Development Studio Client) is built on top of this framework.

After you read this chapter, you should understand the concept of the WebSphere Studio Workbench and be able to navigate through the new GUI. For more detailed information, refer to the online help.
3.1 Eclipse: The start of a new era of tools

The last ten years in the I/T industry were turbulent. The birth of Java, the adoption of the new application model, and the new set of the standards all contributed to this fact. The widespread of the Java language brought to life the new paradigm “Write Once, Run Anywhere”. As Java matured and the Internet became an integral part of our lives, the new ideas and proposals came from many I/T companies. These ideas tried to leverage the advantages of the Internet and the portability of Java applications. With time, this collective effort evolved into the specification called Java 2 Platform, Enterprise Edition (J2EE). In fact, this specification embraced many other specifications, such as JavaServer Pages (JSP) and Enterprise JavaBeans (EJB).

All these new developments required a new set of tools. While the tools were built by many vendors, IBM was one of the leaders among tool builders with its VisualAge for Java and WebSphere Studio products. However, adoption of the J2EE application model changed the way applications were built and packaged. A new application model was a challenge for many tool builders. It was hard to catch up with the frequent changes in the industry.

IBM developed a new approach to solve this problem. That was to build a tool framework that was flexible enough to quickly adjust to the new technologies. The result of this effort is known as Eclipse. To make Eclipse a universal framework, IBM donated it to the open source community and created the Eclipse consortium. More than 150 companies are part of this consortium.

Eclipse is a Java-based integrated development environment (IDE), which is fully extendable by users, customers, and vendors. The code base is written in Java and runs on Windows and Linux. To extend it, programmers write Java code in the form of “plug-ins”. You can download Eclipse, both the binary form and the source code itself, from the Eclipse Web site:

http://www.eclipse.org

IBM’s framework, which is built on Eclipse technology, is called WebSphere Studio Workbench. You can learn more about the WebSphere Studio Workbench in Chapter 1 in the WebSphere Studio Application Developer Programming Guide, SG24-6585.

3.2 A new approach to the application development process

New technologies have changed the way we develop an application. Wide acceptance of J2EE specification imposes the new requirements on the development tools. For example, J2EE defines such things as “platform roles” where the process of building, assembling, and deploying an application is split into multiple roles or responsibilities.

Consider the role in J2EE called Application Component Provider. The responsibilities for this role includes building an application or application component, such as HTML pages, Java Beans, or servlets. IBM has used this role-based approach in the design of WebSphere Studio Workbench.

This section looks at the implementation of this design in WebSphere Development Studio Client for iSeries.
3.3 Components of the Eclipse GUI framework

In general, during any development process, you work with resources, like source files, connections, images, and so on. A development tool gives you the way to look, access, and manipulate these resources in multiple ways. For example, a Java developer’s view of the source files is different from the view of a person responsible for building or assembling an application from the source files. By recognizing this fact, you expect a tool builder to present the project resources to you in a most convenient and productive way.

WebSphere Studio Workbench follows these expectations by providing the following GUI components:

- Workbench
- Perspectives
- Views
- Editors
- Controls

3.3.1 Perspectives

A perspective defines the layout of views in a workbench window and the available actions that can be performed on the resources. Multiple perspectives can exist in a workbench window. Each perspective targets a specific role or a set or responsibilities in the development process. Figure 3-1 shows a sample workbench window with multiple perspective opened. The Web perspective is active.

Let’s look at the very basic navigation techniques in WebSphere Development Studio Client for iSeries. First, you can switch between opened perspectives by clicking a corresponding button in the perspective bar (labeled “Different perspectives” in the figure). If you need to open another perspective, click the “Open perspective” button and select a perspective from the pop-up menu. Each perspective button has a distinct picture. Right-clicking any perspective button displays a menu from where you can select the “Close perspective” option.
There are several tabs under the Gallery and Tasks views. These tabs provide an easy way to switch between opened views (labeled as “Stacked views” in the figure). You can customize which views are shown in the perspective by using one of both of the following options:

- Closing any view by clicking the x button in the upper right-hand corner of a view
- Opening another view by selecting Perspective->Show View from the menu

You can also rearrange the location of the views in the workbench window. You can drag and drop any view in a workbench window. Simply watch the cursor shape as you move the view in the perspective:

- When the cursor changes its shape to a black arrow (▼), the direction of arrow shows the location to where a view will be dropped.
- When the cursor looks like a window (🗗), you can drop the view and it becomes a floating view (like a separate window).

Note: The cursor changes the shape to a black arrow only when you move it over the border of the workbench window.
When the cursor looks like stacked folders 📁, you can drop the view and it is stacked with the views underneath the cursor.

After you are satisfied with this arrangement, save the perspective by selecting from the menu **Perspective-> Save As.**

### 3.3.2 Views

Views provide a way to show the resources to a user. They also support multiple editors. Each view shows a subset of the resources available in the workbench, but depends on the purpose of a view. For example, the Gallery view provides the way to see all multimedia resources that come with the tool, and the Navigator view presents the existing projects in a tree-like manner. By its functionality, the Navigator view is very similar to Windows Explorer.

Besides a distinct way of presenting the resources, you can activate a view-specific menu by right-clicking a resource or anywhere in the view. WebSphere Studio Workbench presents only the menu options that are available for the selected resource. If you experience a situation where the instructions say, “Select from the pop-up menu...”, and you don’t see that choice, verify that you are working with the correct view.

Figure 3-2 shows the Remote Systems view. It has several buttons in the view's toolbar:
- Two action buttons create a new connection and refresh the view.
- Clicking the button with a black arrow activates a pull-down menu that is view specific.
- Clicking the x button results in closing the view.

![Figure 3-2 The Remote Systems view](image)

### 3.3.3 Editors

WebSphere Studio Workbench allows you to use internal or external editors. These components are implemented as plug-ins (internal editors) or as stand-alone applications (external editors) that can be started from within the workbench. For example, you can start the CODE tool from within Development Studio Client by double-clicking a DDS file member.

Depending on the type of a resource, there may be a specific editor to work with that resource. Different members of the WebSphere Studio Workbench family of tools have different sets of the available editors, for example an XML or Java editor. Each editor simplifies the work with a particular type of a resource. By default, the largest portion of the workbench window is reserved for the editors.

You can customize which editor is used with what type of a resource. You do this by going to the workbench properties and selecting **Window-> Preferences**, expanding the **Workbench** tree, and clicking **File Editors.**
3.4 Development Studio Client features

Earlier we discussed that Eclipse SDK only includes the base for developing a full-reach IDE. WebSphere Workbench adds several more plug-ins, or extensions, to the Eclipse platform. Finally, WebSphere Development Studio Client for iSeries, which is based on WebSphere Studio Site Developer Advanced, enriches the workbench with the array of advanced plug-ins. This makes Development Studio Client a premier platform for developing applications for iSeries.

3.4.1 Common features

We have found several features that are very useful during the development of the applications. They include:

- Builders
- Local history
- Team environment
- Compare editor

Builders

Builders represent a utility that takes the existing resource and creates a new resource or updates the existing resource. For example, Java builder produces a compiled version of a Java file. Depending on the selection, Java builder can compile either a single file, a package, or an entire project. A nice feature of this builder (as with VisualAge for Java) is the incremental build, in which a Java file is compiled on every modification. If there is any error during the build, it is displayed in the Task view and the line in error is marked with a small red icon.

Local history

A local history of a file is maintained by the tool. The default settings allow you to save the last 50 versions of a file for at least seven days. If you make a mistake and want to go back to an older version of a file, you don’t have to have a team environment installed. You can use your local history feature.

To change the default setting, select Window-> Preferences and expand Workbench tree. Then select Local History.

To use this feature, right-click any file and select Replace With-> Local History. The workbench shows you the list of available versions and the compare utility.

Team environment

Team environment is based on support of a repository. Neither WebSphere Studio Workbench nor WebSphere Studio Site Developer Advanced include a built-in repository. However, you can buy Rational ClearCase, or download and install another supported product (it is free) from Concurrent Versions System (CVS). You can find it on the Web at:

http://www.cvshome.org/

To learn how to install and configure CVS, see:


Rational ClearCase and CVS allow several developers to access and change the same resource without conflict.
Compare editor
The Compare editor provides a visual way to compare two files. They can be part of any project, local history, or repository. Figure 3-3 shows the compare editor for two Java files in two different projects.

To compare two files from the project tree, follow these steps:
1. Select the first file.
2. Press Ctrl and click the second file.
3. Right-click any of the highlighted files and select **Compare With→ Each Other**.

The Compare editor is quite sophisticated. As you cycle through the changes, it highlights the lines that are different as well as the part of the line (a method name, variable, and so on) that is different. The editor’s control buttons are located just above the file view area.
3.4.2 iSeries-specific features

In addition to the standard features of WebSphere Studio Site Developer Advanced, Development Studio Client adds several iSeries-specific plug-ins and tools. Some of the tools in this release are “loosely” integrated with WebSphere Studio Site Developer Advanced. CODE and VisualAge RPG can be launched from within WebSphere Studio Site Developer Advanced, but are implemented as stand-alone tools.

As a result of this, there are several iSeries-specific perspectives:

- Hosts
- iSeries Projects
- Remote Systems Explorer
- WebFacing
- iSeries Java development tools
- IBM Toolbox for Java

All these perspectives are explored in more detail in the following chapters.

3.5 Online help

Online help in Development Studio Client is implemented as one of the perspectives. IBM has done a great job in providing adequate help. The help contains the information about the workbench itself and the other plug-ins that are integrated into the platform.

This section briefly introduces the information that is available in the online help. It also provide some tips for optimal work with the help system.

First, open the Help perspective. All help information is divided into information sets. To select the information set, click the pull-down menu in the upper left corner of the perspective, as shown in Figure 3-4, and select the set you need.
Two of the available sets are relevant to this book: Site Developer Documentation and Development Studio Client. There are also several tabs at the bottom of the navigation view. For your convenience all help information is sorted, in case of Site Developer Documentation, based on the type of the help (reference, samples, tasks, and so on) or feature of the tool (Java development, Web development, and so on). There is a sophisticated search facility that presents information based on the relevance of the topic to the search criteria.

There is a quick help or, as it is called in the Development Studio Client help system, infopop. You can point or select any item in the workbench window and press F1. If the help is available for the selected item, it is displayed in the pop-up window. The information in the window contains some explanation and the links to the similar topics in the help system.

In some editors, for example the Java editor, there is a code-assist feature that helps you to access the name of the class, method, or variable without going to the class library reference information.
Part 2

Installation and configuration

This part explains the steps for installing and configuring the software products required to convert your 5250 interfaces and applications into a Web-based environment. The installation and configuration of the following applications are described:

- On the iSeries server:
  - WebSphere Development Studio for iSeries
  - HTTP Server (powered by Apache) for iSeries
  - WebSphere Application Server 4.0, Advanced Edition for iSeries

- On the PC client:
  - WebSphere Development Studio Client for iSeries
  - WebSphere Application Server Administrator's Console

After you set up your iSeries server and your PC client, you can perform the steps outlined in the following chapters.
Setting up the iSeries environment

This chapter explains the steps for installing and configuring the software products on iSeries. It explains the installation and configuration for the following products:

- On the iSeries server:
  - IBM HTTP Server for iSeries
  - IBM WebSphere Application Server for iSeries

- On the PC client:
  - IBM WebSphere Application Server Administrator’s Console

  Even though this program runs on a PC, it is the integral part of WebSphere Application Server 4.0, Advanced Edition for iSeries.
4.1 Overview of the required software components

Figure 4-1 shows a simple overview of the interaction between the different software components discussed throughout this redbook on both the iSeries server and the PC client.

![Diagram of software components](image)

Figure 4-1 HTTP server with WebSphere Application Server and tools

For the purposes of this redbook, OS/400 V5R1 was the release installed on our iSeries server. OS/400 V4R5 can be used to perform the same tasks, but some screens will look different than the ones presented here.

Table 4-1 shows the software components installed on the iSeries server and their licensed program numbers for OS/400 V5R1 and V4R5.

<table>
<thead>
<tr>
<th>Software product</th>
<th>Licensed Program Product for OS/400 V4R5</th>
<th>Licensed Program Product for OS/400 V5R1</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM WebSphere Application Server for iSeries</td>
<td>5733-WA4</td>
<td>5733-WA4</td>
<td>At a minimum, WebSphere Advanced Single Server Edition is required.</td>
</tr>
<tr>
<td>IBM HTTP Server for iSeries (original or powered by Apache)</td>
<td>5769-DG1 and group PTF SF99035</td>
<td>5722-DG1 and group PTF SF99156</td>
<td></td>
</tr>
<tr>
<td>IBM WebSphere Development Studio for iSeries</td>
<td>5769-WDS</td>
<td>5722-WDS</td>
<td></td>
</tr>
<tr>
<td>IBM Developer Kit for Java 1.3 or AS/400 Developer Kit for Java 1.3</td>
<td>5769-JV1 (Option 5)</td>
<td>5722-JV1 (Option 5)</td>
<td>This is a prerequisite for installing WebSphere Application Server for iSeries.</td>
</tr>
</tbody>
</table>

HTTP Server (powered by Apache) and WebSphere Application Server 4.0, Advanced Edition for iSeries, are used for the examples in the following chapters.
Figure 4-2 shows the sequence of the required steps to set up your iSeries environment. Note the references to the sections in this chapter that further explain the corresponding step.

4.2 Configuring IBM WebSphere Application Server for iSeries

This section explains how to install and configure the application server instance of WebSphere Application Server 4.0, Advanced Edition for iSeries.

You need to make sure your iSeries server meets the software and hardware prerequisites for installing WebSphere Application Server. Refer to the IBM WebSphere Application Server for iSeries documentation Web site for more information about minimum software requirements:

http://publib.boulder.ibm.com/was400/40/AE/english/docs/
4.2.1 Installing and configuring the WebSphere Application Server instance

You can install the IBM WebSphere Application Server for iSeries (Licensed Program 5733-WA4) in two different ways:

- Directly from the CD-ROM drive of your iSeries server
- Remotely from the CD-ROM drive of a Windows 32-bit operating system workstation connected to the iSeries server

Both installation methods give you the same options to install the base product (*BASE) and option 1:

- The base option (the Client Runtime and Development) supports the following functions:
  - Compiling servlets and enterprise beans
  - Creating enterprise bean JAR files
  - Running stand-alone enterprise bean client applications

- Option 1 (the WebSphere run-time environment) supports the following functions:
  - Serving servlets and JavaServer Pages files from the iSeries Web server
  - Deploying enterprise beans
  - Running enterprise beans

For the purposes of this redbook, we installed WebSphere Application Server directly from the CD-ROM drive of the iSeries server. This is the fastest of the two options.

You can install WebSphere Application Server directly on an iSeries by using the SETUP script in Qshell Interpreter or by running the Run Java (RUNJVA) command. If you are more familiar with UNIX programming, use Qshell Interpreter. If you prefer iSeries commands, use the RUNJVA command on the iSeries command line.

**Note:** To run either of these commands, you need a user profile with *ALLOBJ authority.

Make sure that the WebSphere Application Server 4.0, Advanced Edition for iSeries, CD-ROM is in the CD-ROM drive of your iSeries server.

Follow either the Qshell Interpreter or the OS/400 command line installation steps. In our examples, we used the default values for all parameters. The installation may take up to 75 minutes.

**Using the Qshell Interpreter**

Complete the following steps:

1. To start the Qshell Interpreter, on the OS/400 command line, type:
   
   ```
   STRQSH
   ```

2. Change directories to the root directory on the CD-ROM drive. Enter:

   ```
   cd /QOPT/WebSphere
   ```

3. Enter the following command:

   ```
   SETUP
   ```

   **Note:** Do not issue any other commands until the installation is completed. Qshell Interpreter runs interactively. Entering commands may cause the installation to stop prematurely.
After the installation is completed, you should see the following line on the screen:

Installation completed successfully.

**Using the OS/400 command line**

Enter the following command exactly as it is shown in one continuous line:

```sql
RUNJAVA CLASS(SETUP)
CLASSPATH('/QIBM/ProdData/Java400/jt400ntv.jar:/QOPT/WebSphere/OS400:/QOPT/WebSphere/OS400/INSTALL.JAR:/QOPT/WebSphere') PROP((os400.runtime.exec QSHELL) (java.compiler jitc) (java.version 1.3))
```

Although this command is displayed on more than one line, you must enter it as one continuous line on the OS/400 command line. Be sure to use the same capitalization as shown.

After the installation is finished, you should see a similar line on the screen:

Installation completed successfully.

**Loading the latest PTFs**

Before you continue with the configuration of the WebSphere Application Server, you must install the latest PTFs for WebSphere Application Server 4.0, Advanced Edition for iSeries.

For the latest WebSphere PTF information, go to the Web site at:


At the time this redbook was written, there was the group PTF SF99241. Run the following command to verify whether this PTF is installed on your system:

```bash
DSPDTAARA QEJBADV4/SF99241
```

You should see the following line on the Display Data Area screen:

Group PTF#: SF99241-02 5733WA4 V5R1 02/01/02 4.0.2

This `DTAARA` will only exist if the group PTF is installed.

### 4.2.2 Starting the WebSphere Application Server

Before you start WebSphere Application Server, make sure that TCP/IP is started. To start the WebSphere Application Server jobs, issue the following command on an OS/400 command line:

```bash
STRSBS QEJBADV4/QEJBADV4
```

This command starts the subsystem QEJBADV4 in library QEJBADV4. All WebSphere Application Server jobs run under the QEJBADV4 subsystem.

Two jobs make up the administrative server environment:

- **QEJBMNTR**: A monitor job
- **QEJBADMIN**: An administrative server job

The monitor job starts the administrative server and attempts to restart it when it ends abnormally. QEJBMNTR attempts to restart the administrative server the number of times specified by the `mntr.retryValue` value in the `/QIBM/UserData/WebASAdv4/default/properties/admin.properties` file. By default, the QEJBADV4 subsystem starts only these two jobs.
To ensure that the WebSphere Application Server is started successfully, follow these steps:

1. Issue the Work with Active Jobs (WRKACTJOB) command on an OS/400 command line by specifying QEJBADV4 for the subsystem parameter:

   `WRKACTJOB SBS(QEJBADV4)`

2. Scroll through the WebSphere Application Server jobs running under the QEJBADV4 until you can locate the administrative and monitor server jobs.

3. The status of the two administrative jobs should be:
   - QEJBADMIN should have a status of JVAW
   - QEJBMNTR should have a status of EVTW

When the WebSphere Application Server environment is ready for use, a message is written to the job log of the administrative server job, QEJBADMIN. The message indicates that the WebSphere administrative server is ready.

To determine if the WebSphere environment is ready, perform the following steps from an OS/400 command line:

1. Issue the Work with Active Jobs (WRKACTJOB) command on an OS/400 command line by specifying QEJBADV4 for the subsystem parameter:

   `WRKACTJOB SBS(QEJBADV4)`

2. Locate your administrative server job. The default administrative server job is named QEJBADMIN.

3. Type 5 (Work with Job) on the option line next to the job, and press Enter.

4. On the command line of the Work with Job display, specify option 10 (Display joblog, if active), and press Enter.

5. Scroll in the job log until you locate the message (see Figure 4-3):

   WebSphere administration server QEJBADMIN ready.

   If the message is not shown yet in the job log, press F5 to refresh the job log messages until the message appears. Depending on your iSeries server, it may take several minutes for the message to appear.

   The WebSphere Application Server environment has successfully started when this message appears.

```
Display Job Log

Job . . :   QEJBADMIN  User . . :   QEJB  Number . . . :   024179

Job 024179/QEJB/JOEJBADMIN started on 04/23/02 at 12:14:34 in subsystem QEJBADV4 in QEJBADV4. Job entered system on 04/23/02 at 12:14:34.
WebSphere server started with JDK 1.3.1.
Job 024094/QUSER/QSQSRVR used for SQL server mode processing.
Job 024093/QUSER/QSQSRVR used for SQL server mode processing.
Job 024181/QUSER/QSQSRVR used for SQL server mode processing.
WebSphere administrative server QEJBADMIN ready.
```

Figure 4-3  WebSphere Application Server administrative job QEJBADMIN job log

6. To display the port number on which the administrative server is listening for the Administrative Console, position the cursor on the message, and press F1. The message shown in Figure 4-4 appears.
4.2.3 Creating and starting additional administrative servers

To create a new administrative server, run the QSH script that creates the new server directories and sets up the correct authorities.

**Note:** Your OS/400 user profile must have *ALLOBJ authority to run the script.

Perform the following steps to create a WebSphere Application Server administrative server using the QSH interpreter:

1. Start the QSH interpreter by typing `qsh` on an OS/400 command line. The QSH Command Entry appears.
2. Issue the following two commands from the QSH command line:

   ```sh
   cd /QIBM/ProdData/WebASAdv4/bin
crtnewinst -instance WebF -bootstrap 999 -lsd 9999 -inhttp 9090 -exhttp 90
   ```

   The parameters are explained in Table 4-2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-instance</td>
<td>The required value <code>instance_name</code> specifies the name of the instance. The script creates the new administrative server instance in the <code>/QIBM/UserData/WebASAdv4/instance_name</code> directory.</td>
</tr>
<tr>
<td>-bootstrap</td>
<td>The required value <code>bootstrap_port</code> specifies the number of the TCP/IP port from which the client (such as the Administrative Console) connects to the administrative server instance. Specify an unused port number on your iSeries server. Port 900 is used by the default administrative server instance and should not be used for other instances. Use the <code>Work with TCP/IP Network Status (NETSTAT *CNN)</code> command to display a list of port numbers that are currently being used.</td>
</tr>
<tr>
<td>-lsd</td>
<td>The required value <code>lsd port</code> specifies the number of the TCP/IP port upon which the Location Service Daemon (LSD) service listens. Specify an unused port number on your iSeries server. Port 9000 is used by the default administrative server instance and should not be used for other instances. Use the <code>Work with TCP/IP Network Status (NETSTAT *CNN)</code> command to display a list of port numbers that are currently being used.</td>
</tr>
</tbody>
</table>
We used WebF as an example of an additional WebSphere Application Server administrative server name.

After the `crtnewinst` command runs, two messages result from creating the required directory structure and setting the proper authorities for the new WebSphere Application Server administrative server. They are displayed on the QSH screen as shown in Figure 4-5.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-inthttp</td>
<td>The optional value <code>internal_http_server_port</code> specifies the number of the TCP/IP port upon which the internal HTTP server listens. The internal HTTP server can be used directly or by external HTTP servers via the WebSphere plug-in. If you do not specify this parameter, the default value is 9080. Although this is an optional parameter, you should always specify the parameter if you do not want it to conflict with the default WebSphere instance, which uses port 9080.</td>
</tr>
<tr>
<td>-exthttp</td>
<td>The optional value <code>external_http_server_port</code> specifies the number of the TCP/IP port upon which the external http server listens. The external HTTP server must be configured for WebSphere, as well, to listen on this port through its administrative interface. If you do not specify this parameter, the default value is 80. The external HTTP server must also be configured to listen on the port specified for the -exthttp parameter.</td>
</tr>
</tbody>
</table>

We used WebF as an example of an additional WebSphere Application Server administrative server name.

After the `crtnewinst` command runs, two messages result from creating the required directory structure and setting the proper authorities for the new WebSphere Application Server administrative server. They are displayed on the QSH screen as shown in Figure 4-5.

### QSH Command Entry

```
$ cd /QIBM/ProdData/WebASAdv4/bin
$ >crtnewinst -instance WebF -bootstrap 999 -lsd 9999 -inthttp 9090 -exthttp 90
Trying to create WebSphere instance WebF
Instance WebF created successfully.
$```

*Figure 4-5  Creating a new WebSphere Application Server administrative server*

3. To start the WebF administrative server, issue the following command from the QSH command line:

```
strwasinst -instance WebF
```

Two jobs start for each administrative server:

- **A monitor job:** The system appends the string MNTR to the name of the administrative server, which in our case is WebF. The monitor job name becomes WEBFMNTR.

- **An admin job:** The system appends the string ADMIN to the name of the administrative server, which in our case is WebF. The admin job name becomes WEBFADMIN.

After the `strwasinst` command finishes starting the jobs, you see the message `Admin job 024869/QEJB/WEBFADMIN started` as shown in Figure 4-6.
To verify that the WEBFADMIN job is ready, perform the following steps:

1. Issue the Work with Active Jobs (WRKACTJOB) command on an OS/400 command line and specify the QEJBADV4 for the subsystem parameter:

   ```
   WRKACTJOB SBS(QEJBADV4)
   ```

2. Locate your administrative server job WEBFADMIN.

3. Type 5 (Work with Job) on the option line next to the job, and press Enter.

4. On the command line of the Work with Job display, specify option 10 (Display job log, if active), and press Enter.

5. Scroll in the job log until you locate the message:

   ```
   WebSphere administrative server WebFadmn ready.
   ```

6. Move the cursor to the line and press F1. You should see the help text, which is similar to the example in Figure 4-7. It displays the port number on which the administrative server of WebSphere Application Server is listening (999 in our example).

The next step is to install the required software on your PC client.

### 4.3 Installing and starting the IBM WebSphere Administrative Console on the PC client

The WebSphere Administrative Console allows you to administer WebSphere Application Server for iSeries. The Console is a stand-alone Java application that runs on a workstation (Windows NT/2000, AIX, or Solaris). It connects to the iSeries server where administrative data is stored. And, it is the essential tool to administer WebSphere Application Server 4.0, Advanced Edition for iSeries.
For the purposes of this redbook, we installed the WebSphere Administrative Console on a Windows 2000 workstation. Even the Administrative Console runs on a workstation.

### 4.3.1 Installing the WebSphere Administrative Console

To install the Administrative Console for WebSphere Application Server on your workstation, perform the following steps:

1. Insert the WebSphere Application Server 4.0 Advanced Edition for Windows NT CD in the CD-ROM drive of the Windows workstation on which you plan to run the console.

2. If Autorun is enabled, the Windows Install Shield program starts automatically. If not use Windows Explorer to navigate to your CD-ROM drive. Double-click the `setup.exe` file.

   **Note:** Ignore any messages that tell you to shut down your Web server.

3. Go through the first few initial windows (select the language and agree to the terms and conditions of using WebSphere Application Server).

4. Choose the **Custom Installation** option as shown in Figure 4-8 and click **Next**.

   ![Custom Installation Options](image)

   **Figure 4-8** Selecting the Custom Installation option

5. On the next window, select the components you want to install. Select **Administrator’s Console** and **IBM JDK** as shown in Figure 4-9. Make sure you deselect all other components but these two. Click **Next**.
6. On the next window, specify the name of the iSeries server where the WebSphere Application Server is installed. To specify the correct domain name, go to the OS/400 screen and execute the CFGTCP command. Select option 12 (see the example in Figure 4-10).

```
- Change TCP/IP Domain (CHGTCPDMN)
- Type choices, press Enter.
- Host name ............... 'AS03'
- Domain name ............ 'ITSOROCH.IBM.COM'
```

7. Go back to the installation window and type the host name and domain name (for example, AS03.ITSOROCH.IBM.COM). Make sure you type the host name as it appears in the Change TCP/IP Domain command because it is case sensitive (see Figure 4-11).
8. Continue with the windows that follow by accepting the defaults and clicking **Next**. Click **Finish** in the last window to complete the installation.

9. After the installation is completed, install the latest fixes for the Administrator’s Console. Starting from WebSphere Application Server V3.5.2, IBM distributes WebSphere Application Server for iSeries PTFs along with the corresponding fixes for the Administrator’s Console.

   Locate the `/QIBM/ProdData/WebASAdv4/ClientFixPacks/was40_ae_ptf_nt.zip` file on your iSeries server.

10. Unzip the file into a directory on your Windows workstation.

11. Run `install.bat` from the directory to which you unzipped the fixes.

12. Complete the installation process. You have to provide the following information:
   - For WebSphere Application Server installation directory, the default value is `C:\WebSphere\AppServer`. After you provide this information, the installation process may take several minutes before it prompts you for more information.
   - Answer **No** when the installation process asks whether you want to install fixes for the HTTP server.
   - Answer **No** when the installation process asks whether you want to install Connector Architecture for WebSphere.

### 4.3.2 Starting the WebSphere Administrative Console

To start the WebSphere Administrative Console on a Windows workstation, you can either use the Start menu shortcut that was created during installation on Windows 2000 or run a batch file from a command prompt.

**From the Start menu shortcut**

From the Start menu, click **Programs-> IBM WebSphere-> Application Server V4.0 AE-> Administrator’s Console**.

**Note:** With this method, Administrative Console starts on the default port 900.
From the command prompt
Open a command prompt, and enter the following two commands (assuming you chose the default install directory for WebSphere Application Server):

```
cd C:\WebSphere\AppServer\bin
adminclient AS03.ITSOROCH.IBM.COM 999
```

Note the following explanation:
- `WebSphere\AppServer\bin` is the directory where the WebSphere Administrative Console is installed.
- `adminclient` is the actual Administrator's Console program.
- `AS03.ITSOROCH.IBM.COM` is the host name of your iSeries server.
- `999` is the iSeries port number that you intend to use.

The amount of time that it takes for the WebSphere Administrative Console to start depends on the number of objects that you configured. It is not uncommon for this to take several minutes if you configured a large number of objects.

The Console Ready message is displayed in the message area at the bottom of the WebSphere Administrative Console window as shown in Figure 4-12.

![Figure 4-12  Administrative Console window](image)
4.3.3 Working with the WebSphere Application Server installation

Here are some hints and tips to help make sure your WebSphere Application Server installation and configuration is successful.

Changing WebSphere Application Server ports

If your HTTP server configuration uses a port other than the default (port 80), you must make sure this information is reflected in WebSphere Application Server. This is achieved by following these steps:

1. Start the Administrative Console.
2. In the WebSphere Administrative Console topology view, select **Virtual Hosts**.
3. Look at the **Host Aliases** information under the **General** tab of the virtual host. With the way we defined the port information, you should see the host aliases as shown in Figure 4-13. Only one virtual host (default_host) is defined in this example.

![Figure 4-13 Host Aliases information](image)

4. Port 90 is used for the access through the external HTTP server, and port 9090 is used by the built-in HTTP server. The asterisk (*) reflects the fact that the application server will accept requests for any host names for the specified ports.
5. You can add or remove entries from this table.
Starting and stopping the WebSphere Application Server
You can use the Administrative Console to start and stop application servers. If you attempt to stop an application server and it does not stop, you can stop it by using OS/400 commands. When you start or stop an application server, you also start or stop everything that runs within the application server.

Note: Make sure you understand the difference between the WebSphere Application Server and an application server:
- WebSphere Application Server is the name of the product.
- An application server is a run-time environment that supports different Java objects. A single WebSphere administrative domain can support multiple instances of an application server.

Starting an application server
To start an application server instance, follow these steps using the WebSphere Administrative Console:
1. In the topology tree, locate the application server that you want to start.
2. Right-click the selected application server.
3. In the resulting menu, click Start. It may take a few seconds for this operation to complete. A window opens that announces the successful start of the application server.

Stopping the application server
To stop an application server instance, follow these steps using the WebSphere Administrative Console:
1. In the topology tree, locate the application server that you want to stop.
2. Right-click the application server.
3. On the resulting menu, click Stop. It may take a few seconds for this operation to complete. A window opens that announces the successful stop of the application server.
4. If the stop operation fails, you can use the Force Stop option instead. Click Force Stop instead of Stop.

Note: Using the Force Stop option ends objects abnormally and does not clean up WebSphere resources as a normal Stop operation does. Use this option only after a normal Stop process fails.

Verifying the installation of WebSphere Application Server
You can verify the installation of WebSphere Application Server by accessing simple sample applications. To do this, point your Web browser to:

http://your.system.name:9090/webapp/examples/

See the example in Figure 4-14. Click any of the links and verify the results of the execution of the sample applications.
4.4 Setting up and configuring the IBM HTTP Server for iSeries

The HyperText Transfer Protocol (HTTP) is the foundation of the World Wide Web (WWW) as we know it today. Figure 1-4 on page 24 demonstrates the very important position of the Web server today in your e-business environment. It is a TCP/IP application protocol or a client server protocol that runs over any TCP/IP network.

HTTP allows servers and clients to interact and link with hypermedia objects that may be of any Multipurpose Internet Mail Extension (MIME) type, including text, audio/video, and graphic images. The formal connect process between the client and server, the handling and transferring of the data, and the subsequent disconnect are the heart of the HTTP protocol.

You must create a new HTTP server configuration (or modify an existing one) to contain the information that the HTTP server needs to route requests to the appropriate WebSphere Application Server methods.

4.4.1 Starting an administrative server instance

This section explains how to start an administrative HTTP server instance (*ADMIN). The administration server is an instance of the HTTP server that allows administration of certain iSeries functions using a Web browser (security, authority, and so on).
To start the *ADMIN server, you must:

- Verify that IBM HTTP Server for iSeries (5722-DG1) software is installed.
- Verify that the latest PTFs for the HTTP server for iSeries are applied.
- Verify that TCP/IP Connectivity Utilities for iSeries (5722-TC1) is installed.
- Start the administrative server instance.

**Verifying IBM HTTP Server for iSeries and TCP/IP Connectivity Utilities are installed**

Follow these steps to verify that the IBM HTTP Server for iSeries (5722-DG1) and the TCP/IP Connectivity Utilities (5722-TC1) are installed:

1. On the OS/400 command line, enter:

   ```
   GO LICPGM
   ```

   The Work with Licensed Programs screen appears.

2. Select menu option 10 (Display installed licensed programs). The Display Installed Licensed Programs screen is displayed as shown in Figure 4-15.

3. Scroll through the display using the Page Down key:
   - IBM HTTP Server for iSeries appears as licensed program 5722DG1 (for OS/400 V5R1) or 5769DG1 (for OS/400 V4R5). Its status must be *COMPATIBLE. Otherwise, you will have to install (or reinstall) it.
   - TCP/IP Connectivity Utilities appear as licensed program 5722TC1 (for OS/400 V5R1) or 5769TC1 (for OS/400 V4R5). Its status must be *COMPATIBLE. Otherwise, you have to install (or reinstall) it.

   ![Figure 4-15 List of installed products via GO LICPGM, option 10 to display](image-url)
Now that you have verified that your IBM HTTP Server for iSeries (5722-DG1) and your TCP/IP Connectivity Utilities (5722-TC1) are installed, you're ready to start the *ADMIN instance of IBM HTTP Server for iSeries.

**Verifying that the latest PTFs for the HTTP server for iSeries are applied**

For the list of the latest PTFs for the HTTP server for iSeries go to the HTTP Server: PTFs and Support Web site at:


Select the release you need and install the required PTFs. At the time this redbook was written, we had to install the group PTF SF99156 for V5R1.

**Starting the Administrative HTTP server instance**

In this step, you start an administrative server on the iSeries, making it possible for you to configure your HTTP server. You can start the *ADMIN server instance from Operations Navigator or from the OS/400 command line.

**From Operations Navigator**

Operations Navigator is the graphical point-and-click interface to the iSeries server and is part of the Client Access Express for Windows product. For more information on Client Access and Operations Navigator, see the Client Access product Web site at:

http://www.iseries.ibm.com/clientaccess/

**Note:** We expect that IBM Client Access Express for Windows (licensed program 5722-XW1) is installed and operational on both your iSeries server and your Windows client.

To start the *ADMIN instance from the operations Navigator, follow these steps:

1. Start Operations Navigator. From your Windows desktop, click **Start-> Programs-> IBM AS/400 Client Access Express-> AS/400 Operations Navigator.**
2. Double-click your iSeries server in the main tree view of Operations Navigator. You may be requested to sign on to your iSeries server.
3. Expand **Network-> Servers**, and click **TCP/IP**.
4. In the right panel of the Operations Navigator view (Figure 4-16), select **HTTP Administration**, right-click, and select **Start** to start HTTP Administration.
From an OS/400 command line

You can start the *ADMIN server from the OS/400 command line prompt. Follow these steps:

1. Start a 5250 session with your iSeries server.

2. Start the *ADMIN instance from the OS/400 command line by entering the following command:

   ```
   STRTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)
   ```

   The server takes a while to start. After a few minutes, you'll see the message HTTP server starting underneath the command line, at the bottom of the window.

   The *ADMIN job for the HTTP server is started. The next step is to create a new HTTP server configuration to support your work.
4.4.2 Creating a new HTTP server

To create a new HTTP server and its corresponding configuration file, follow these steps:

1. Start your JavaScript-enabled Web browser.
2. Enter the following URL:
   
   \[http://your_server_name:2001/\]

   Here \(your_server_name\) is your iSeries server host name. Throughout this redbook, we use \(as03.itsoroch.ibm.com\) as our server name. 2001 is the default port for the \(^{*}\)ADMIN server.

3. Enter your user name and password when prompted by the window shown in Figure 4-17. Your OS/400 user ID must have \(^{*}\)ALLOBJ authority.

![Figure 4-17 Logon window for the \(^{*}\)ADMIN HTTP server](image)
4. The AS/400 Tasks page appears in your browser as shown in Figure 4-18. We recommend that you set a bookmark for this page, so you can easily return to it.
5. Click **IBM HTTP Server for iSeries**. The IBM HTTP Server for iSeries page loads as shown in Figure 4-19.

![IBM HTTP Server for iSeries](image)

Figure 4-19   IBM HTTP for iSeries main page

6. Click **Configuration and Administration**. The Configuration and Administration page loads.
7. Click **Create HTTP Server** in the menu on the left part of the Browser window (see Figure 4-20).

![Figure 4-20 Create HTTP server main page](image)

8. Select the **HTTP server (powered by Apache)** radio button and click **Next**.

9. The next page of the Create HTTP Server wizard loads. Type your server name into the Server name field (we used **WebFacing** as a name). Click the **Next** button.

10. The next page of the Create HTTP Server loads. Select whether you want to create the new instance of the HTTP server based on the existing configuration. We select the **No**. Then click **Next**.
11. On the next page, you can specify the root directory for your new server instance. Enter the directory name /QIBM/UserData/WebASAdv4/WebF/Apache (see Figure 4-21). We select the root directory for our instance of HTTP server under the directory created for the instance of WebSphere Application Server in step 2 on page 73.

![Figure 4-21 Specifying a server root directory](image)

Click **Next**.

12. Specify the root directory for the HTML pages. This directory is called the *document root directory*. We specified /QIBM/UserData/WebASAdv4/WebF/Apache/htdocs. Click **Next**.

13. On the next page, specify the port number on which your instance of the HTTP server will listen for incoming requests. During the creation of a new WebSphere Application Server instance, we selected 90 for the HTTP port number for external requests (see step 2 on page 73). Type this number (90) and click **Next**.

14. On the next page, select the **Combined log file** radio button and click **Next**.
15. The wizard shows your summary page (see Figure 4-22). Click **Finish**.

![Create HTTP server summary page](image)

**Figure 4-22 Create HTTP server summary page**

16. The wizard presents the Congratulations page with the Configure and Manage buttons. To use the new instance of the HTTP server with WebSphere Application Server, you need to change its configuration. Click **Configure**.
17. The global settings screen for the WEBFACING server is displayed (see Figure 4-23). Under the Dynamic Content list, select **WebSphere Application Server**.

Figure 4-23  Global settings page for the HTTP server
18. Click the **WebSphere Version 4 Advanced** radio button and select **WebF** from the pull-down menu (see Figure 4-24). Click **OK**.

![Figure 4-24  Selecting the WebSphere Application Server instance](image)

You can view the directives by selecting **Display Configuration File** from the Configuration Files list on the global settings page. The configuration file for our instance of the HTTP server is updated with the following directives:

```
<Directory /QIBM/ProdData/WebASAdv4/WSsamples/>
  Allow from all
  Order allow,deny
  AllowOverride None
  Options None
</Directory>
Alias /WSsamples/ /QIBM/ProdData/WebASAdv4/WSsamples/
LoadModule ibm_app_server_http_module /QSYS.LIB/QEJBADV4.LIB/QSVTIHSAH.SRVPGM
WebSpherePluginConfig /QIBM/UserData/WebASAdv4/WebF/config/plugin-cfg.xml
```

### 4.4.3 Starting the HTTP Server (powered by Apache) instance

The HTTP Server (powered by Apache) runs in the QHTTPSVR subsystem, and each HTTP server instance starts multiple jobs. The WebSphere Application Server code that plugs into IBM HTTP Server for iSeries runs in the HTTP server job that communicates with the administrative server and one or more application servers.
You can start the HTTP server instance either from the HTTP Server Configuration and Administration forms or from an OS/400 command line.

**From the Configuration and Administration forms**

To start the HTTP Server from the browser-based client, complete the following steps:

1. Start the server using one of the following ways:
   - From the global settings page (see Figure 4-23 on page 92), click **Start**.
   - From the Manage HTTP Servers page (see Figure 4-25), select **WEBFACING** and click **Start**.

2. Scroll through the list of HTTP Servers created on your iSeries, in the right frame, until you locate the server you created and configured (for example, WEBFACING).

3. Select the radio button next to your HTTP Server.

4. Click the **Start** button at the bottom of the right frame. The Manage HTTP Servers page updates with the status of your HTTP server.

**From the OS/400 command line**

To start the HTTP server instance from the OS/400 command line, type:

```
STRTCPSVR SERVER(*HTTP) HTTPSVR(my_instance)
```
Here, \textit{my\_instance} is the name of your HTTP server instance (for example, WEBFACING).

If you change your HTTP server instance configuration, stop and then start your HTTP server instance. To stop your HTTP server instance, enter the following command from the OS/400 command line:

\texttt{ENDTCPSVR \texttt{SERVER(*HTTP) HTTPSVR(my\_instance)}}

\section*{4.5 Verifying the installation}

To verify the installation of the WebSphere Application Server 4.0, Advanced Edition for iSeries and the HTTP Server (powered by Apache) server, perform the following steps:

1. Start the WebSphere administrative server as explained in 4.2.3, “Creating and starting additional administrative servers” on page 73.

2. Start the WebSphere Administrative Console as explained in 4.3.2, “Starting the WebSphere Administrative Console” on page 78.

3. Expand the \textbf{WebSphere Administrative Domain} tree and select \textbf{Default Server}.

4. Click the \textbf{Start} button as shown in Figure 4-26.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure4-26.png}
\caption{Starting the application server}
\end{figure}

5. Start the instance of the HTTP Server (powered by Apache) as described in 4.4.3, “Starting the HTTP Server (powered by Apache) instance” on page 93.

6. Open a Web browser (we use Internet Explorer in our example).
7. Point your browser to the following URL:

   http://your_server_domain_name:90/WebSphereSamples/SingleSamples/HelloEJB/
   HelloEJB.jsp

   We access the HTTP server on port 90 (see step 13 on page 90). You should see the
   same page as the example shown in Figure 4-27.

![Figure 4-27 Verifying the installation of the products](image)

**Note:** The URL is case sensitive. Therefore, the capitalization must be consistent with the
above example.

The first time you start WebSphere Application Server, you must start the application
server before you start the HTTP server instance. The first time the WebSphere
Application Server starts, it performs initialization that affects your HTTP server instance. If
the HTTP server instance is already running, restart it after you start the WebSphere
Application Server to refresh the configuration.

### 4.6 Starting WebFacing jobs

The last step in the process of configuring your iSeries server is to start the WebFacing
server. The support for the WebFacing server comes as a PTF:

- SF66539 for V4R5
- SI02254 for V5R1

Make sure you have these PTFs installed. For up-to-date information about latest PTFs, see:

   http://www.ibm.com/software/ad/wdt400/about/sysreq.html

To run the WebFacing applications, start the WebFacing server. Perform the following steps:

1. Execute the appropriate following command on the iSeries server for your version:
   - V5R1:
     ```
     STRTCPVRVR SERVER(*WEBFACING)
     ```
   - V4R5
     ```
     STRWSVR
     ```

2. Run the WRKACTJOB command. You should see two jobs – QQFWFSVR and
   QQFVTSVR – under the QSYSWRK subsystem (see Figure 4-28).
3. Run the following command:

   DSPSYSVAL SYSVAL(QAUTOVRT)

4. If the value of QAUTOVRT is 0, change it to non-zero value by running the following command:

   WRKSYSVAL SYSVAL(QAUTOVRT)

**Note:** If QAUTOVRT is 0, then no browser sessions can be launched in place of virtual terminal (VT) jobs. The WebFacing run-time server relies on VT jobs. The value for QAUTOVRT determines the number of virtual terminal jobs that can be auto started.
Setting up the client environment

This chapter provides the detailed instructions about installing software components on your workstation. The following components are installed on your workstation:

- WebSphere Application Server Administrative Console
- WebSphere Development Studio Client for iSeries (Development Studio Client)
5.1 Installing WebSphere Application Server Administrative Console

The first component you need to install on your workstation is WebSphere Application Server Administrative Console. This is a stand-alone Java program that connects to the WebSphere administrative server that runs on iSeries. It is an integral part of WebSphere Application Server 4.0, Advanced Edition for iSeries.

The installation process for WebSphere Application Server Administrative Console is described in 4.3, “Installing and starting the IBM WebSphere Administrative Console on the PC client” on page 75.

5.2 Installing WebSphere Development Studio Client for iSeries

WebSphere Development Studio Client for iSeries is packaged and distributed on three CDs. The installation process is simple. The only thing you need to select is the installation directory.

Follow these steps to install WebSphere Development Studio Client for iSeries:

1. Insert disk one into the CD-ROM drive.

2. If the autorun feature is enabled on your workstation, the installation process will start automatically. If it is not the case, run the setup.exe file located in the root directory of disk one.

3. The first window prompts you to select one of four choices as shown in Figure 5-1. We recommend you read the installation notes.

4. After you read the installation instructions, click Install Product.
5. In the next window, click **Next**.
6. Read the agreement and click **Accept** (if you agree to the terms and conditions).
7. Select the installation directory (default is C:\WDSC), and click **Next**.
8. The Summary page appears next. Review the information and verify that it is correct. Click **Install** (see Figure 5-2).

![Figure 5-2 Installation summary page](image)

9. The installation process starts. Insert CD disk two and disk three when prompted. The installation process takes around 30 min.
10. Reboot your system after the installation.

**Note:** At the time this redbook was written, Service Pack 1 (SP1) was available.
Part 3 Working with iSeries applications on a workstation

This part introduces several tools to help you develop, test, and debug host applications in several languages. The main objective of this part is to show dedicated green-screen users that these workstation tools are much easier to use and more productive.

Part 3 covers:
- Description of the sample applications
- WebSphere Studio Site Developer Advanced iSeries plug-in to access resources on iSeries: The plug-in is called Remote Systems Explorer (see Chapter 7, “iSeries extensions (plug-ins) in Development Studio Client” on page 113).
- Cooperative Development Environment (CODE)
- VisualAge RPG
Introduction to the sample applications

This chapter presents the RPG Order Entry and Customer Master Inquiry applications by example.

The Order Entry application represents a commercial application, although it does not include all of the necessary error handling that a business application requires. The Customer Master Inquiry application is used to demonstrate how to use design-time control (DTC) and the Web Interaction wizard (Chapter 11, “iSeries Web development tools in Development Studio Client” on page 207).

In Chapter 10, “WebFacing Tool” on page 179, we use the RPG Order Entry application to Web-enable the application with the WebFacing Tool. The goal is to use the existing RPG application to service both the Web application and the host 5250 application.
6.1 Overview of the Order Entry application

This section provides an overview of the application and a description of how the application database is used.

6.1.1 The ABC Company

The ABC Company is a wholesale supplier with one warehouse and 10 sales districts. Each district serves 3,000 customers (30,000 total customers for the company). The warehouse maintains stock for the 100,000 items sold by the company. Figure 6-1 illustrates the company structure (warehouse, district, and customer).

![Figure 6-1 The company structure](image)

6.1.2 The ABC Company database

The company runs its business with a database. This database is used in a mission critical, online transaction processing (OLTP) environment. The database includes tables with the following data:

- District information (next available order number, tax rate, and so on)
- Customer information (name, address, telephone number, and so on)
- Order information (date, time, shipper, and so on)
- Order line information (quantity, delivery date, and so on)
- Item information (name, price, item ID, and so on)
- Stock information (quantity in stock, warehouse ID, and so on)

6.1.3 Application files

These are the files that comprise the application (they are located in the APILIB library):

- **ORDENTD (Parts Order Entry):** Display file
- **ORDENTR (Parts Order Entry):** Main RPG processing program
- **PRTORDERP (Parts Order Entry):** Print file
- **PRTORDERR (Print Orders):** RPG server job
- **SLTCUSTD (Select Customer):** Display file
- **SLTCUSTR (Select Customer):** RPG SQL stored procedure
SLTPARTD (Select Part): Display file
SLTPARTR (Select Part): RPG stored procedure

ORDENTR is the main RPG program. It is responsible for the main line processing. It calls two supporting RPG programs that are used to prompt for and select end user input. They are SLTCUSTR, which handles selecting a customer, and SLTPARTR, which handles selecting part numbers. PRTODERR is an RPG program that handles printing customer orders. It reads order records that were placed on a data queue and prints them in a background job.

6.1.4 Database table structure

The ABC Company database has eight tables:
- District
- Customer
- Order
- Order line
- Item
- Stock
- Warehouse
- History

The relationship among these tables is shown in Figure 6-2.

![Figure 6-2 Table relationships](image)

6.1.5 Customer transaction flow

The following scenario walks you through a customer transaction that shows the application flow. By understanding the flow of the iSeries application, you can understand the changes made to this application to support a graphical client.

Starting the application
First add APILIB to your library list and then start the application by running the following command:

```
CALL ORDENTR
```
When the Order Entry application is started, it displays the window shown in Figure 6-3.

![Figure 6-3 Parts Order Entry](image)

When the Parts Order Entry display appears, the user has two options:

- Type in a customer number, and press the Enter key.
- End the program by pressing either F3 or F12.

If the user does not know the customer number, they can press F4 to view a window that contains a list of available customers (Figure 6-4).

![Figure 6-4 Prompting for the customer](image)

The user presses F12 to remove the window and return to the initial screen, or they scroll through the items in the list until they find the customer they want. The user types 1 in the option field and presses the Enter key to indicate their choice. The selected customer information is entered into the initial screen (Figure 6-5).
Chapter 6. Introduction to the sample applications

109

Figure 6-5   Parts Order Entry

An additional prompt is displayed that allows the user to type a part number and quantity.

If the user does not know the part number, they press F4 to view a window that contains a list of available parts (Figure 6-6).

Figure 6-6   Select Part
The user presses F12 to remove the window and return to the initial screen, or they scroll through the items in the list until they find the part they want. The user types 1 in the option field and presses the Enter key to indicate their choice. The selected part is added to the initial window (Figure 6-7).

![Figure 6-7 Parts Order Entry](image)

The user may type 2 next to an entry in the list to change the order. When they press the Enter key, a window opens that allows the user to change the order line (Figure 6-8).

![Figure 6-8 Change Selected Order](image)
When the order is complete, the user presses F6 to update the database. Then, an order is placed on the data queue for printing.

The printed order is created by a batch process. It shows the customer details and the items, quantities, and cost of the order (Figure 6-9).

The printed order is made by the PRTORDERR (Print Orders) RPG server job.

You can find the sample code in the APILIB.savf file, which is available for download as part of the additional materials that accompany this redbook. The instructions for installing the file are outlined in Appendix A, “Additional material” on page 271.

6.2 Overview of the Customer Master Inquiry

Customer Master Inquiry is pure RPG/400 code where we use display files (DSPFs) for the data input.

6.2.1 Starting the application

Add AS2314 to your library list and call the main program from an iSeries command line:

    CALL MLG265

On the first window (Figure 6-10), enter the search criteria. This may include the ZIP code or the name.

![Figure 6-10  Customer Master Inquiry](image)
The second window (Figure 6-11) shows a subfile, which lists customers that match the search criteria.

Type x next to the customer you want to select and to see detailed information about them. Figure 6-12 shows the detailed description of the customer that was selected.

We use Customer Master Inquiry programs as a base code to demonstrate how to use WebSphere Development Studio Client for iSeries to create Web application with ILE RPG programs (Chapter 11, “iSeries Web development tools in Development Studio Client” on page 207).

The sample code is available in the AS2314.savf file of the additional downloadable material. See Appendix A, “Additional material” on page 271, for download and installation instructions.
iSeries extensions (plug-ins) in Development Studio Client

This chapter introduces a detailed description of using the Remote Systems Explorer (RSE). RSE helps you to manage the objects on your iSeries server. It also provides the seamless communication channel for other tools – CODE and VisualAge RPG.
7.1 Introduction to using Remote Systems Explorer

The Remote Systems Explorer is a workbench perspective to define your connections to an iSeries server through an RSE communications connection. It allows you to manage a set of files on the iSeries server through a filtering system, monitor jobs, and run commands. This chapter explains how RSE works and how you can set up connections.

7.2 Perspectives

A perspective defines the initial set and layout of views in the workbench window. One or more perspectives can exist in a single workbench window.

To open a perspective, follow these steps:

1. In the Site Developer window (Figure 7-1), click Open Perspective.

2. Click Other (Figure 7-2).

3. In the Select Perspective window (Figure 7-3), select Remote Systems Explorer, and click OK.
This opens the Remote Systems Explorer. In the toolbar at the left of the window, the RSE perspective icon appears. This icon opens the Remote Systems Explorer perspective. The Remote Systems Explorer shows the connections that are already defined. The New Connection and Local connections are displayed by default.
7.3 Profiles

Before you can use RSE to access your iSeries, you need a connection to your host iSeries. The first time you connect to your iSeries server, you need to define a profile. Profiles offer a way to group connections. As you complete the steps for your first connection, you can choose your personal profile, profiles, or a team profile that lets you share resources and information with other people. Personal profiles do not share resources with other people. The team profile is a default predefined profile in the RSE.

To create a parent profile in Remote Systems Explorer, follow these steps:
1. Open the New window. Click the Menu icon in the toolbar of the Remote Systems view and select New Profile. The New window (Figure 7-5) opens.

![Figure 7-5] Creating a new profile

2. Enter the new profile name.
3. Select the Make active check box to make this profile active.
4. Click Finish.

**Note:** The Remote Systems Explorer profiles in the workbench are unrelated to iSeries profiles.

Profiles created in the workbench can be used to hold connections. In the Remote Systems panel, only the connections for active profiles are shown. If you deactivate a profile, the connections that are held by that profile are not shown in the Remote Systems panel.

To deactivate a profile, follow these steps:
1. Click the Menu icon and select Select Active Profiles in the RSE menu.
2. The profiles that are active have a check mark. Click the profile to deselect it.

**Important:** If you delete a profile, you also delete the connections held by that profile.
7.4 Connections

Once you create a profile, you can set up connections to one or more separate iSeries servers. To define a new connection in Remote Systems Explorer, follow these steps:

1. Open the New Connection window using one of the following options:
   - In the Remote Systems panel, right-click and select Create Another Connection.
   - Click the Menu icon in the toolbar of the Remote Systems window and select New Connection.

   The New Connection window opens as shown in Figure 7-6.

2. Click the Parent profile drop-down list. To keep your files private, select a profile you created. To share your resources, select the Team profile. To create a parent profile, see 7.3, “Profiles” on page 116.

3. Enter the values for your connection in the window (Figure 7-7):
   - In the Connection name field, specify a name for the connection.
   - From the System type drop-down list, select an operating system.
   - In the Host name field, specify the host name for the machine. This can be the name of the machine or the IP address.
   - In the Default User ID field, specify your User ID on the machine.
   - In the Description field, specify how you want to describe this connection.
Click **Finish**. The connection that you just defined appears in the Remote Systems pane.

4. Connect to the remote system:
   a. Right-click **iSeries Files**, **iSeries Commands**, or **iSeries Jobs** and select **Connect**.
   b. Enter your password for this particular connection when prompted.

   When you successfully connect, the Connected property in the Properties view (located in the lower-left corner of the RSE perspective) is set to **yes**.

### 7.5 Subsystems and filters

Each connection in the Remote Systems Explorer exposes subsystems, filters, and items. This information is displayed in an easy-to-understand tree view.

#### 7.5.1 Subsystems

Every connection has subsystems to divide the iSeries in functional areas. Subsystems handle the task of retrieving remote information and executing commands.

To view the subsystems of a connection, expand the connection view (see Figure 7-8).
Each iSeries connection has four subsystems:

- **iSeries Files**
- **iSeries Commands**
- **iSeries Jobs**
- **Integrated File System (IFS) files**

**Note:** Subsystems in the Remote Systems Explorer are not the same as subsystems on the iSeries server.

You can expand the subsystems by clicking the plus (+) sign.

**iSeries Files Subsystem**

The iSeries Files Subsystem lets you manage the objects on your iSeries server. You can copy, rename, delete, and retrieve information about objects on the iSeries. You can also run programs on the iSeries and compile objects.

**iSeries Jobs Subsystem**

At any time, there are a number of jobs running on your iSeries host. You, or other members of your team, may all be performing actions at the same time. To keep track of jobs, Remote Systems Explorer provides a Job Subsystem that lets you:

- Hold jobs
- Release jobs
- End jobs
- Display the job log for active jobs

**iSeries Commands Subsystem**

The Commands Subsystem defines command filters, invokes programs, and commands on an iSeries host and displays output messages in the Commands view. The subsystem provides you with a number of predefined command sets. The help text of the iSeries Commands Subsystem explains how to define and run commands on your host.

**iSeries IFS Files Subsystem**

The RSE implements integrated file system support. The IFS provides a common interface to the different systems on your iSeries machine. The file system contains two default filters:
- **File systems**: Contains all of the file systems to which you have access
- **Root file systems**: Represents one component of file systems and lists all of the specific files and folders under the root of the IFS.

You can treat the IFS in the same way that you treat iSeries files, except the location of the files is different. Both the iSeries files and the IFS are displayed in a tree view.

### 7.5.2 Filters

Filters are an easy way to select libraries, objects, source members, commands, and jobs. Each filter can contain only one kind of item. The following example shows how to define a filter for CLP source members in the file QCLSRC in library MYLIB. To create a filter, follow these steps:

1. Expand the subsystem **iSeries Objects**, as shown in Figure 7-9.

![Figure 7-9 Expanding iSeries files](image)

2. Expand **Your members**. The New Member Filter window opens.
3. Enter the name of your new filter as shown in Figure 7-10.
4. Click **Add** to add the filter selections. In the Member Filter String window (Figure 7-11) that opens next, you can define the selections for your filter.

![Figure 7-11 Defining selections for the filter](image)

5. Click **Finish** to add the selections to your filter. To add more selections to your filter, repeat steps 4 and 5.

   **Note:** Each filter can contain only one type of object.

6. Click **Finish** to create the filter. The filter is added to the subsystem.

   A temporary log record is added to the collection for which you created a filter. The result of the filter is expanded in the panel, as shown in Figure 7-12.
7. The information in the Remote Systems Explorer panel is not updated automatically. Click the Refresh button to refresh the information.

**Filter pools**

If you have been using the RSE for some time, your workspace may contain too many filters to easily navigate. In this case, filters can be partitioned into pools, where one pool contains filters for many different elements. For example, one filter pool may contain filters for your accounts receivable program, while another contains filters for your payroll program.

Here is an example to show how filter pools work:

1. In the Remote Systems view, expand iSeries Files to see a collection of:
   - Filters for libraries
   - Objects
   - Members
   - A library list

2. Click the Menu button on the toolbar of the Remote Systems view (Figure 7-13), and select Show Filter Pools.
3. If the iSeries Files items are not expanded, click the plus (+) sign next to **iSeries Files**. You now see your filters listed under Default Filter Pool, as shown in Figure 7-14.

4. Right-click **iSeries Files** and select **New**. Now, you see the option to create a new filter pool or filter pool reference.

5. Right-click a filter pool and select **New**. The option to create a new library, object, or member filter now resides within a specific filter pool.
Filters can be moved between filter pools. To move filters between filter pools, follow these steps:

1. Expand the filter pools.
2. Right-click the filter you want to move and select **Move**, as shown in Figure 7-15.

3. In the Move window, expand the user profile (**MyProfile** in this example) that contains the filter pool you want to select.
4. Select the filter pool (**MyFilterPool** in this example), as shown in Figure 7-16.
5. Click **OK**. The filter is now transferred to the new filter pool.
**Figure 7-16  Selecting a filter pool**

*Note:* Notice that the filter pool in which the filter currently resides is not shown in the selection window (Figure 7-16).
Working with CODE

CODE includes a set of flexible workstation and host tools for developing and maintaining iSeries application programs. The workstation tools include the editor, screen, report and database designer, syntax checker, program verifier, and a debugger front end. Host components include the compiler, library system, and debugger back end.

This combination of tools provides you with the productivity of the PC environment as well as the integrity of the iSeries server.
8.1 What is code

Code is a client/server development environment for writing host or server code. It allows you to develop or maintain applications in ILE RPG, RPG/400, ILE COBOL, COBOL/400, ILE C, ILE CL, CL, DDS, and Java. It is an easy, cost-effective, and productive way to work with iSeries host applications on your Windows workstations.

The new integrated iSeries tools are not fully functional yet. Until all features are available in these new tools, users may have to use the classic tools for certain tasks.

8.2 Connections to iSeries

Through a tightly integrated communication design, the new remote systems framework is capable of launching the classic tools seamlessly. They provide you with such features as CODE Editor, CODE Designer, and the Distributed Debugger.

Connections are mainly defined in Remote Systems Explorer (RSE). These connection tools are accessed through Remote System view, which is part of the Remote Systems Explorer's perspective.

8.2.1 Managing system connections

RSE allows you to create and manage connections to multiple remote iSeries servers. This feature also allows multiple server connections to the same system, each with its own independent job environment. There is tight interaction with the CODE Communication Console, allowing CODE Editor, for example, to communicate through your new connection.

Connections are associated with one or more profiles allowing you to organize your connections into groups. Only connections with “active” profiles are displayed within the Remote Systems Explorer (see 7.1, “Introduction to using Remote Systems Explorer” on page 114).

These system connections can also be called as projects, which can contain only necessary information to manage application source code and objects.

8.2.2 Using the CODE Communication Console

CODE Communication Console is the front end of the Communication Daemon that must be active when you are using WebSphere Development Studio Client for iSeries and CODE.

The Communication Daemon and Console are started automatically when you start your workstation or when you select Start-> Programs-> WebSphere Development Studio Client for iSeries-> Communication-> Communication Daemon/Communication Console.

Communication Console is found in the Windows task bar. You can also launch it from the CODE Editor and the CODE Designer toolbar. To start console from task bar, right-click the daemon icon and select Properties (Figure 8-1).
On the Properties window, you can:

- Create a new server configuration
- Create a library list for the servers
- Adjust STRCODE Settings
- Change management settings
- See the error log

The server configurations are created automatically when WebSphere Development Studio Client for iSeries is used to connect iSeries servers, for example, from Remote Systems Explorer. However, if you are using CODE Editor or CODE Designer as a stand-alone application, you must create server configurations. Follow these steps:

1. Click the Add A New Server button in the Communication Properties window (Figure 8-2).
2. Type your iSeries server IP address or name.
3. Type the CODE server name.
4. Type your user ID and password.
5. Click OK.
8.3 Managing your environment with Remote Systems Explorer

Within each connection, subsystems define a particular functionality. Filters are used within each subsystem to provide you the ability to filter and view only objects in which you are interested. Four subsystems are provided by default, each containing useful predefined filters as shown in Figure 8-3. These are:

- iSeries Files
- iSeries Commands
- iSeries Jobs
- IFS Files

8.3.1 Subsystems and filtering

The iSeries Files Subsystem allows you to access your remote objects. You can create filters to provide a customized view of libraries, objects, and members. For a library filter, you can select from a set of predefined filters (*LIBL, *CURLIB, etc.) or specify your own filter string based on the name of the libraries.
8.4 Creating object filters to select only source physical files

Before you create an object filter, you must connect to your iSeries server first as explained in 7.4, “Connections” on page 117. When the connection is ready, create an object filter by following these steps:

1. Expand the **Your objects** tree in the Remote Systems Explorer view.

2. The New Object Filter window (Figure 8-4) opens. Enter the filter name for this filter pool and click **Add**.

Refer to Chapter 7, “iSeries extensions (plug-ins) in Development Studio Client” on page 113, for detailed information about how to manage your environment.
3. On the New window (Figure 8-5), select a library from the Library list. You can either type the library name in the box or click the **Browse** button to select correct library name from your iSeries server. If you select the Browse button option, you have to add a library to your library list filter or create a library filter.
4. Click **Add** to select object type and attributes.

5. The Add Object Type And Attribute window (Figure 8-6) opens. Select **File** for type and **PF-SRC** for attribute. Click **OK**.

![Figure 8-6 Object Type and Attribute window](image)

6. Click **Finish** on the New and New Object Filter windows to end the Object Filter creation.

7. Now you can open and edit the source code. Simply right-click the selected object, select **Open With**, and choose an appropriate editor (Figure 8-7).
8.5 Editing the source code

WebSphere Development Studio Client for iSeries can use classical tools such as CODE Editor, CODE Designer, or the new integrated LPEX editor. All tools can be launched from Remote Systems Explorer. This section focuses on the CODE tools.

8.5.1 CODE Editor

CODE Editor is a programmable editor that can create and edit many kinds of files, including programs, documentation, and data files. CODE is the best way to make changes to your source.

With CODE, your first compile is a clean compile. Using the program verifier, a component of the CODE Editor, all your compile errors can be fixed in the workstation environment without invoking a single host compile. The program verifier is the iSeries compiler on the workstation. Once a program is verified, a window opens that lists any compile errors. You can double-click an error (Figure 8-8) to see it displayed in the source and highlighted in the editor (Figure 8-9).
Because the CODE Editor is workstation based, it takes advantage of workstation features that are not readily available to the text-based editors, such as SEU. For example, a feature called *token highlighting* allows the CODE Editor to highlight and emphasize the different parts of each RPG specification by color. Figure 8-9 illustrates the variation of the token highlighting.

Token highlighting uses different colors for the comments, which helps in the readability of the code. This saves hours of debugging time, for example, in cases where invalid lines of code are commented out and the programmer who is debugging the code does not notice the asterisk (*) in column 7.

Figure 8-10 shows RPG/400 code, but not RPG IV code. One reason this code was chosen for the first example is to illustrate that you can start using the CODE Editor even if you have not yet moved all your code to RPG IV. CODE Editor supports many forms of RPG on the iSeries server, including RPG, RPG36, RPG38, and SQLRPG. It also supports many other iSeries language types, including CL, DDS, C, C++, COBOL, and Java.
Another reason for choosing RPG/400 code in this example is to illustrate how CODE Editor can help in your conversion to RPG IV. From a single pull-down menu option, CODE Editor converts either a selected portion of the source or an entire source member to RPG IV. By selecting Action-> Convert Selection or Action-> Convert RPG Source, you can easily convert source members visually and then save the resulting temporary workstation file back to an iSeries source member. Alternatively, you can include logic from an existing RPG/400 program. For example, you can include it into a new RPG IV program or module by cutting and pasting the portion of the logic you need from the temporary file and subsequently discarding it. The converted code is shown in Figure 8-11.
Notice that the results are the same as if we had issued a Convert RPG Source (CVTRPGSRC) command on the iSeries host. In this case, however, CODE Editor has created a temporary file on the workstation. You can decide now whether to save the converted source to a member on the system or to simply copy part of the logic to another source member written in RPG IV.

The source sequence numbers in Figure 8-11 are the same as in the SEU editor. SEU line commands (such as C, A, B, I) can be used in CODE Editor, just as they can be used in SEU. There are also tools such as the more workstation-oriented cut, copy, and paste, as illustrated on the toolbar. For programmers who prefer not to use the SEU-type line commands, you can remove the sequence numbers from the edit window by selecting View->Line numbers.

Prompting on RPG source statements works much like it does in SEU by pressing F4. However, many programmers find it less necessary to prompt in the CODE Editor because the CODE Editor edit panel understands the parts of the RPG specification, and the Tab key is programmed to jump from “token” to “token”. For example, on a C specification, the Tab key allows programmers to jump quickly and easily from Factor 1 to Operation Code to Factor 2 to Result and so on. And the format line in the CODE Editor changes automatically to the format of the source line where the cursor is currently positioned. Therefore, it is easy to tell exactly what part of the specification your cursor is on without prompting.

Other useful features of the editor that are not found in SEU include:

- Multiple levels of “undo” support
- Auto save of the source code to a local workstation file periodically
Much better visual presentation of the source code
An indented view of nested logic (Figure 8-12)

8.5.2 CODE Designer

CODE includes the CODE Designer, a what-you-see-is-what-you-get (WYSIWYG) utility for designing display files and printer files. This new, powerful and easy-to-use interface makes designing your DDS screens, reports, and physical files both fast and easy.

Note: Before you edit a source member with CODE Designer, you have to change your library list to correspond to the new environment, especially if you have field references to file. To do that, open CODE Communication Daemon properties and change the library list. After changing the library list, restart all CODE servers. When you start your editor again, a new library list is active.

Using CODE Designer Web Settings with your DDS source

You can use the Web Settings tab in CODE Designer to customize how your programs will look and function when accessed through a Web browser.

To open CODE Designer, use one of the following options:

- From the Windows desktop Start menu, click \Start-> Programs-> WebSphere Development Studio Client for iSeries-> IBM CODE-> CODE Designer.
From WebFacing perspective, right-click a DDS file and select **Open With-> Code Designer.**

Load the source code to CODE Designer. Now you can access Web Settings by clicking the icons in CODE Designer’s DDS Tree, which is located on the left-hand side of the CODE Designer window. You can also access Web Settings by selecting DDS objects from within CODE Designer’s Details and SCREEN tabs. If Web Settings are available for the selected object, the Web Settings tab is displayed in the bottom pane of CODE Designer (Figure 8-13).

![CODE Designer Web Settings](image)

**Figure 8-13  CODE Designer Web Settings**

**Available Web settings**

The available settings are:

- **Key labels**: Changes the text label of your command keys.
- **Key order**: Changes the command key order and visibility.
- **Display size**: Changes the size of the HTML table used in browser.
- **Visibility**: The Hide/Show field in the browser.
- **Location**: Changes the location of your fields in the Web pages.
- **Program-defined HTML**: Uses the field value as HTML.
- **User-defined HTML**: Changes the style class or overrides the constant text.
- **Appearance and Text**: Changes the style class or overrides the constant text.
- **Insert HTML**: Adds HTML to your Web page.
- **Options for VALUES**: Changes the label or text of the values settings.
> **Create Graphic**: Inserts a graphic into the Web page.
> **Create hyperlink**: Sets the field value as a hyperlink.

In this example, we use Visibility, Create graphic, and Options for VALUES.

**Visibility**
Visibility allows you to hide parts of the DDS code from the Web page. In this example, we hide part of the heading (1=Select) as shown in Figure 8-14.

![Figure 8-14 Hiding a field](image)

**Creating a graphic in your existing subfile**
Use the Create Graphic setting to display an image in place of your field. An example of using this setting is for a product database that includes images of inventoried items.

There are two ways to show a graphic in your Web page:

- Populate the File name field with the complete path to an image file on your workstation or with the complete URL to an image file (for example, http://www.ibm.com/image.gif)
- For fields in your program that return values that can be used as part of the image file name, or the full URL to an image file, add these values to the File name field. Select the field from the Choose a field list and click **Insert field value**.

In this example, we use the second method to bring a graphic into the Web page. In our application, field **IID** contains a part number like 000001 or 000002. To add images for each record, we perform the following steps:
1. Select a field from the first row of your subfile.
2. Select the **Web Settings** pane.
3. Click **Create graphic**.
4. Select the **Create graphic** check box.
5. Select a field from the Choose a field list.
6. Click the **Insert field value** button. Then the field name is inserted into the File name field.
7. Add the graphic extension.

Now we have complete graphic name &{IID}.gif (Figure 8-15). When the program is executed, the application server looks for such file names as 000001.gif and 000002.gif.

![Figure 8-15 Creating a graphic](image-url)

You must manually copy files, identified by using the Choose a field list, to the ..\webApplication\images\generated directory in your Web project. Refer to Figure 8-16.
The images in this directory are later published to the WebSphere Application Server when you deploy your project with the Export wizard.

**Options for VALUES**
The Options for VALUES setting is available for input fields that are specified with the VALUES keyword. The Options for VALUES Web setting allows you to change the label that is displayed to the user. To use this Web setting, select the **Option texts for VALUES** check box and enter a new label for the VALUES parameter (see Figure 8-17).
After Web-enabling the host application, the results of our changes appear as shown in Figure 8-18. The results of the Web settings modification are:

- Visibility: 1=Select is hidden.
- Options for Values: Selectable values has new label.
- Graphic: Graphic is added to subfile instead of field value.
8.6 Library list and compiling

To compile correctly from WebSphere Development Studio Client for iSeries, you must place the correct library list in your environment. Simply right-click one of the subsystems in Remote Systems Explorer and select Properties as shown in Figure 8-19.
On the Properties for IFS Files window (Figure 8-20), add a library to the library list. Click OK.
Another way to change the library list is to use the iSeries commands subsystem:

1. Expand the iSeries commands tree.
2. Right-click the Add library to library list command set.
4. In the Add Library List Entry (ADDLIBLE) window (Figure 8-21), type the library name in the Library field and select the List position:
   - *First
   - *Last
   - *After
   - *Before
   - *Replace
   Click OK.

![Add Library List Entry window](image)

Figure 8-21 Add Library List Entry window

Compilation is done as an interactive iSeries job. To compile a program as a batch job, you must change the command default. Follow these steps:

1. Right-click the source member and select Compile-> Work with commands. The Compile Commands window (Figure 8-22) opens.
2. Select one of the commands and click **Change**.

3. In the Command string field of the Change window (Figure 8-23), before the compile command, add:
   
   `SBMJOB CMD(`
   
   Then add parentheses () at the end of the command.

4. Change the SBMJOB command parameters by clicking **Prompt**. You can change actual command parameters by pressing F4.

   To change the library list for the compile command, you can choose one of these options:
   
   - Use the job description.
   - Add the library name or names into the Initial library list parameter (see Figure 8-24).
When compiling from the CODE Editor or CODE Designer, you can select how compilations are done on the iSeries server:

- From the CODE Editor, select **Actions-> Compile-> Prompt**.
- From CODE Designer, select **Tools-> Compile-> Prompt**.

Then the CODE Program Generator window (Figure 8-25) opens with the commands.
When you select Options-> Settings as shown in Figure 8-25, you can change:

- Action mode of compilation
- Interactive
- Batch
- Batch job parameters
- Job description
- Job queue
- Default ILE command
- CRTxxxMOD
- CRTBNDxxx

If you are using interactive mode for compilation, the library list settings are in the Properties of the Communication Daemon. To access Properties from CODE Editor, select Tools-> Servers-> Properties. Then the Communication Properties window (Figure 8-26) opens.
Figure 8-26 Communication Properties

Type the name of the library into the Library field and click Add. The library list setting becomes active when you restart the Communication Daemon.

8.7 Debugging ILE programs with Distributed Debugger

The IBM Distributed Debugger is a client/server application that enables you to detect and diagnose errors in code developed with OS/400 Integrated Language Environment (ILE) and Original Program Model (OPM) languages and Java. This client/server design makes it possible to debug applications running on an iSeries machine while the debugger user interface is running on a Windows workstation.

8.7.1 Starting the debugger

Follow this process to start a debug session:

1. Start the host server job in the QSYSWRK subsystem by using the STRDBGSRVR command (Figure 8-27).
2. Start the code server. From a 5250 screen, enter the STRCODE command. Make sure that the library list is set correctly before you enter the STRCODE command. Then you see the screen shown in Figure 8-28.

![Figure 8-28 STRCODE screen](image)

3. Start the CODE Editor and load the source code.

4. Start the Distributed Debugger. There are several ways to load the Debugger:
   - From the client command line, enter the following command:
     ```bash
     idebug -qengine=400 program_name
     ```
   - From the active CODE Editor window, follow these steps:
     i. Click the toolbar button 📐.
     ii. Click **Actions-> Debug** and select the appropriate option. As shown in Figure 8-29, we selected **Interactive application** for this example.
5. Select your server as shown in Figure 8-30. Click OK.

6. Enter the Call Program parameters (Figure 8-31). Click OK.
7. If the connection is not ready, enter your user ID and password in the Logon window (Figure 8-32).

8. Select the line where you want the debugger to stop. Press F9 to add a breakpoint.

9. To see values of the variables, highlight the variable, right-click, and select **Add to Program Monitor** as shown in Figure 8-33.
10. To run the program, click the arrow button in toolbar. Then the program will run to the next breakpoint and stop. Alternatively, you can right-click the line in question and select **Run To Location**. Figure 8-34 shows an example of how the debugging session looks.

On the lower left side of the window, you can see the values of your variables in the Monitors pane.
11. To terminate the Debugger, click the black square icon on the toolbar, select **Debug-> Terminate**, or press Shift+F5.

### 8.8 Getting help

The new integrated iSeries tools (CODE Editor, CODE Designer) are not fully integrated with WebSphere Studio Site Developer Advanced. The best way to find help is to open the Help perspective (see 3.5, “Online help” on page 62). As shown in Figure 8-35, you can access online help for CODE, VisualAge RPG, and IBM Distributed Debugger.

**Note:** With the latest service pack, CODE Editor and CODE Designer are partially integrated with WebSphere Studio Site Developer Advanced. They are invoked by default when you open RPG source or DDS files.
Figure 8-35  Development Studio Client Help
Chapter 9. Introduction to VisualAge RPG

This chapter explains how to use VisualAge RPG (VARPG), which is part of WebSphere Development Studio Client for iSeries (Development Studio Client). This chapter also introduces the graphical user interface designer.
9.1 VisualAge RPG features

VisualAge RPG is a new feature of the IBM WebSphere Development Studio Client for iSeries. This Visual Development Environment allows you to build client RPG applications with Windows GUI and transparent access to iSeries resources. You edit, compile, debug, and execute RPG right on the client. The VisualAge RPG compiler is the same code base as the host ILE RPG compiler, so you can easily move logic between the client and server. You can now generate Java applications and Java applets from the VisualAge RPG source, making RPG a powerful e-business language.

VisualAge RPG includes the following components:

- A Visual Development Environment
- An RPG compiler on the workstation
- A programmable language-sensitive editor
- TCP/IP support
- A Display file import facility
- Transparent PC-to-iSeries connectivity
- Source-level debugger
- RPG Goes Client/Server

There is a great and ever increasing demand for client/server applications in the user community serviced by iSeries servers. Users want the Windows/GUI interface, point-and-click navigation, integration with the workstation desktop, and multi-media integration. Developers also want to produce Java applications that can run virtually anywhere.

RPG is the dominant language in the iSeries community. RPG is on over 75 percent of all iSeries development systems. Most applications for the iSeries are written in RPG. iSeries users need the ability to move their valuable mission-critical legacy host applications to the client/server and to create new client/server applications. It is a time and money consuming process to retrain the programmers and rewrite the programs.

VisualAge RPG is designed to address these user needs. It’s easy to learn and easy to use for RPG programmers. The visual development environment allows you to build windows by dragging and dropping the visual parts from the extensive array of visual components.

If you’re converting an existing application, the Display file import facility converts your existing iSeries display files into their GUI equivalents. The logic behind the windows is written in RPG IV, which is the same RPG IV as on the iSeries server. It is also the same compiler as on the iSeries, but enhanced for the new GUI operations. It takes a short time to learn the new operations, so you can quickly be as productive in the new environment as you were with RPG IV.

9.2 VisualAge RPG Language Reference

The VARPG compiler supports an enhanced version of the RPG language. To use VARPG, you must understand the language fundamentals. You can obtain information about RPG IV from the iSeries Information Center at:

http://www.ibm.com/eserver/iseries/infocenter

Online information about VARPG is available in the Help perspective of WebSphere Studio Site Developer Advanced. To access the information, follow these steps:
1. From the Windows desktop, click **Start-> Programs-> IBM WebSphere Studio Site Developer Advanced-> IBM WebSphere Studio Site Developer Advanced**. The Help - site Developer window (Figure 9-1) opens.

2. Click the **Help** perspective to open it.

3. Select **Development Studio Client**.

4. At the bottom of the panel, click **Classic tools** as shown in Figure 9-2.

### 9.3 Programming with VARPG

This section discusses how to convert an existing DDS screen to VARPG. It describes the necessary steps for using a simple DDS source. In VARPG, you can import the screen and create your application based on this screen.

While RPG IV is a *procedural language*, VARPG is an *event-driven language*. Event-driven applications are fundamentally different from procedural applications. A procedural application, once it has been started, is always in control. It knows when input is expected and what the order of the different pieces of input is. An event-driven application is programmed so that it can handle any input at any moment. By this, the user is in control and not the program. Figure 9-3 shows the basics of an RPG IV procedural application.
VARPG is a event-driven language. The user interface is not only graphical, but the basics of this language differ from RPG IV. Figure 9-4 shows the structure of an event-driven model. The new concept of an Action subroutine that is invoked by events from the GUI makes the transition to event-driven programming relatively easy for RPG programmers.

A 5250 green screen is designed using fields, constants, and command keys. In the graphical design, VARPG contains parts, attributes, and events. Figure 9-5 shows how these terms fit into the design.
9.3.1 Starting the VARPG Designer

To start the VARPG GUI Designer, from the Windows desktop, click Start-> Programs-> IBM WebSphere Development Studio for iSeries-> VisualAge RPG-> GUI Designer. Now the VisualAge RPG workbench starts.

The VisualAge for RPG application development environment is based on a GUI project organizer. In this interface, your application is organized into projects. A project represents a complete set of objects (source code files, GUI objects, and so on).

9.3.2 The VARPG GUI Designer

You use the GUI Designer to create VARPG applications. The VisualAge RPG GUI Designer consists of these major components (see Figure 9-6):

- Menu bar
- Toolbar
- Project View
- Parts Palette
- Designer Canvas
The toolbar
The toolbar is a menu of icons as shown in Figure 9-7. It provides fastpath access to many of the menu items on the menu bar. To see which icon is which, move the mouse pointer over the icons. Then a short description of the icons function is displayed in the form of rollover help.

The Project View
The Project View is used to hold all the parts that you will create during your GUI Designer session. You can customize the Project View to display the parts of a project in different views (see Figure 9-8).
The Parts Palette

The Parts Palette (Figure 9-9) contains parts that you will use to create the graphical user interface of your application. Each part has attributes and events associated with it. By changing attributes and associating business logic with the events, you customize the behavior of the part according to your application logic.

Figure 9-8   Project View

Figure 9-9   Parts Palette
The Designer Canvas
The Designer Canvas (Figure 9-10) is the area where you design the GUI for your application. You select the visual and non-visual components and drop them to the canvas. By organizing them on the canvas, you create a what you see is what you get (WYSWYG) GUI interface for your application.

![Image of Designer Canvas](image)

Figure 9-10  Designer Canvas

9.4 Creating a simple GUI subfile application

This section explains how to create a simple GUI subfile application with VARPG. The application shows a list of customers displayed in a subfile. When the end user double-clicks a record in the subfile, a new window with detailed customer information is displayed. This example gives you an idea of how easy it is for an RPG programmer to create GUI applications with VARPG.

To create the first window with a subfile part, follow these steps:

1. Select the Subfile part icon on the Parts Palette (Figure 9-11).
2. Click the Designer Canvas at the location where you want to position the upper-left corner of the subfile.
3. Drag the subfile’s borders with your mouse to size it to the desired height and width.
4. Change the window heading:
   a. Double-click the Designer Canvas title bar. This opens the Window Part Properties window (Figure 9-12).
   b. Enter a new part name and title for the window.
5. Specify which database information should be displayed in the subfile, using the reference information from the iSeries server. You need to connect to the iSeries server. If you haven’t specified an iSeries server in VARPG before, follow these steps:
   a. On the GUI Designer menu bar, click Server-> Define TCP/IP servers.
   b. Click Add.
   c. Specify the iSeries server name or IP address and alias name in the window. Click OK.
   d. From the GUI Designer menu bar, click Server-> Define server logon.
   e. Click Add.
   f. From the Remote location drop-down menu, select the server.
   g. Specify a user ID and password for this server. VARPG uses this information:
      • When it accesses reference field information
      • When the compiler resolves external file descriptions
      • At run time when accessing iSeries objects
      If you don’t provide this information, VARPG displays a logon prompt everytime it needs to access the iSeries.
   h. Click OK.
   i. Click Close to close the Define Server logon window.

6. From the GUI Designer menu bar, select Server-> Define Reference Fields.
7. In the Define Reference Fields window, expand **iSeries Server-> APILIB-> CSTMR-> CSRCD** and double-click **CSRCD**.

8. Select the fields you want to display in the subfile. In this example, we select **CID, CFIRST, CLAST, and CCITY** (press the Ctrl key and keep pressing it while selecting the fields).

9. Move the cursor to the subfile part and click. These fields are now added to the subfile (see Figure 9-13).

![Image of Define Reference Fields window and subfile part]

**Figure 9-13 Adding fields to the subfile**


11. Double-click the subfile part to change its properties. The Subfile Part Properties window (Figure 9-14) opens. Follow these steps:
   a. On the General page, change the part name.
   b. Use the arrow buttons to slide the tabs, and then click the **Color** tab.
      i. Select the **heading color background** option.
      ii. Select the **Predefined colors** radio button and select **Blue** from the pull-down menu.
      iii. Select the heading color foreground.
      iv. Select the **Predefined colors** radio button and select **Yellow** from the pull-down menu.
   c. Click the **Font** tab.
      i. Select the **Select font** radio button.
      ii. Select **Comic Sans MS** and set the size to 10.
d. Click the Field List tab in the Subfile Part Properties window. Select CID and click Change. In the Subfile Field Part Properties window, complete these tasks:
   i. On the General page, add ID in the Heading 2 field.
   ii. Click the Style tab and change Column width to 100 pixels to fit the heading text into the column width.
   iii. Click OK.

  e. Click OK in the Subfile Part Properties window.

Figure 9-14 Changing the subfile part properties

You are now done with the GUI. Next, you need to write some RPG code to load the subfile with data. You need an event to trigger the loading. Otherwise, the subfile will appear empty at the application startup. You use the Create event of the window to read the data from the database file and write to the subfile:

1. To add the action subroutine for the Create event to the program, right-click the window title.

2. Select Events-> Create. The RPG editor is opened and the frame of an action subroutine is created in the source.

3. Enter the logic as shown in Figure 9-15.
4. Save the file.

5. In the GUI Designer, select Server-> Define iSeries information from the menu bar to specify the server on which your database file is located.

6. Build the project. Select Project-> Build-> Windows from the menu, or click the Build the Windows version of the project button on the toolbar.

7. Click OK in the new pop-up window that informs you that the build process was successful.

8. Click the Run the project button on the toolbar.

9. Your application starts and the window with the subfile part filled with data appears (see Figure 9-16).
10. You might say that filling the subfile with all records is unrealistic, so let’s add paging to the subfile. You need to add a certain number of records to the subfile to fill the list area of the subfile. The subfile part automatically creates a scrollbar when it is filled over its page size limits. When you scroll the subfile scrollbar to its end or click the arrow at the bottom of the scrollbar, the subfile issues a PageEnd event. You use the PageEnd event to add records to the subfile if it is scrolled to the bottom at run time. Follow these steps:

a. In the GUI Designer, in the design window, right-click the subfile and select Events-> PageEnd. The editor window opens and the PageEnd action subroutine is created.

b. In the source, create a FillPage user subroutine. This subroutine is invoked from the Create and PageEnd action subroutines.

c. Copy the code from the Create action subroutine into the FillPage subroutine. Add code to limit the number of records loaded at a time to 20 records (see Figure 9-17).

d. In the Create and PageEnd action subroutines, add a call to subroutine FillPage.
Figure 9-17   Adding the FILLPAGE subroutine

Now when you run the application, the subfile fills with 20 records. When you scroll the subfile scrollbar to the bottom, the PageEnd event fires and another 20 records are added to the subfile.

You might have noticed that you didn’t specify a subfile control record. The GUI subfile doesn’t have a control record, so it is much easier to deal with than the 5250 subfile. The GUI subfile allows input in cells like the 5250 subfile does. It allows you to change color attributes on a cell level and has many more features that you would expect from a GUI part.

11. Let’s add one more feature to your application. If the user double-clicks a record in the subfile, we want you to show an extra window with detail data for this selected customer. The subfile fires the Enter event when the user double-clicks. If you want to react to a single-click, you would use the Select event.

a. To create a new design window, select the Window with Canvas icon in the parts palette, and click in the empty space of the project window. A new window is created.

b. Double-click the Design Window title bar to change the name and title properties.

c. Click the Startup tab in the Properties window and deselect the Open immediately check box. This prevents the window from loading when the application starts. You will load this window when the Enter event from the subfile is fired.

d. To add the reference fields information for the customer detail data to this window, click Server-> Define reference fields from the GUI Designer menu bar.

e. In the Define Reference Fields window, expand iSeries Server-> APILIB-> CSTMR-> CSRCD and double-click CSRCD.
f. Select all fields in the record format.

g. Click the upper-left corner of the Design Window you just created. All selected fields from the Reference Field window are positioned on the Design window.

h. Resize the Design Window to make sure all fields fit into the frame. VARPG automatically sizes the window frame to fit all parts when you resize it in the designer (see Figure 9-18).

![Figure 9-18 Designing the Customer Details window](image)

i. Close the Customer Details window.

j. In the subfile Design window (Customer Info), right-click the subfile and select **Events-> Enter**. The Editor window opens with the Enter action subroutine being created.

k. Add code to this action subroutine to see the detail window. Use the new VARPG SHOWWIN operation code (see Figure 9-19).
Let's look at the code:

- The first statement uses the READS operation code, which is new in VARPG. It is equivalent to the READC operation code in RPGIV. It doesn't read changed records from the subfile, but it reads selected records.

- The CHAIN operation code is normal RPG processing. It accesses the data from the database.

- The SHOWWIN operation code creates the window that is specified, which is your detail window in this case.

- The WRITE operation writes the data from the RPG variables into the fields on the window. This is like writing a record format to a workstation file.

- The SETATR operation gives this window focus, so that it is on top of the other windows on the desktop.

- In the file specification, you need to change the file to keyed access and to add the keyword BLOCK(YES). This tells the compiler to block instead of using no-block, which is the default for keyed files (see Figure 9-20).
1. Save your source.

m. Click the **Build** button on the GUI Designer toolbar to build the application.

n. Click the **RUN** button on the GUI Designer toolbar to run the application.

o. Double-click a record in the subfile. The Customer Details window with the selected customer data appears as shown in Figure 9-21.
Writing subfile applications with VARPG can be quite easy as we’ve demonstrated here.

Keep in mind to write thin client applications. VARPG allows you to CALL server programs on the iSeries server and to pass data via parameters between the client VARPG program and the iSeries server program. Try to keep as much business logic as possible on the iSeries server.

If you are concerned about maintaining VARPG applications on a large number of client workstations, VARPG provides an option to install the Client applications on a Server, for example, on the IFS file system located on the iSeries.

You have the option to choose a compact installation of the VARPG application on your client workstations. This compact installation installs a shortcut on the client workstation. The shortcut points to the application on the server. Then the application is downloaded to the client workstation at startup. This approach has been adopted by many VARPG customers.

Finally, if you need to run your application on a non-Windows system, VARPG allows to create Java source from the VARPG source. This Java application then runs on any Java Virtual Machine (JVM). To create Java program from your VARPG application, follow these steps:

1. Click the Build the Java version of the project button (located next to the Windows Build button on the toolbar) in the GUI Designer.
2. After the built is completed, click the Run the Java version of the project button on the toolbar.

The application now runs in the JVM.

### 9.5 Additional information

Some operation codes are not supported in VARPG. The operation codes that are not supported are:

- ACQ
- DUMP
- EXFMT
- FORCE
- MHHZO
- MHLZO
- MLLZO
- NEXT
- REL
- SHTDN

For more information about programing with VARPG, consult the following Web sites:

http://www.software.ibm.com/ad/varpg/
http://www.ibm.com/software/ad/wdt400/course_downloads.html

You can also find information on VARPG in these Redbooks:

- AS/400 programming With VisualAge for RPG, SG24-2222
- AS/400 Applications: A Fast and Easy Way to Install, Set Up and Work with VRPG and CODE/400 (ADTS CS), SG24-4841

Plus, you can consult these VisualAge RPG manuals:

- *Programming with VisualAge RPG*, SC09-2449: Contains specific information about creating applications with VisualAge RPG. It describes the steps you have to follow at every stage of the application development cycle, from design to packaging and distribution. Programming examples are included to clarify the concepts and the process of developing VARPG applications.

- *VisualAge RPG Parts Reference*, SC09-2450: Provides a description of each VARPG part, part attribute, part event, part attribute, and event attribute. It is a reference for anyone who is developing applications using VisualAge RPG.


- *Java for RPG Programmers* by Phil Coulthard and George Farr: Introduces you to the Java language (and RPG IV) by comparing it to the RPG language. It is a good first step in your Java journey. It also includes an interactive CD tutorial on Java and VisualAge for Java, by MINDQ.

- *Experience RPG IV Tutorial* by Maha Masri, Heather Rogers, and Julie Santilli: This book is a handbook with questions and exercises to help you get hands-on experience with this exciting new version of RPG. It includes an interactive CD-ROM tutorial that teaches you RPG IV and ILE in a fun and step-by-step approach.

- *VisualAge for RPG by Example*, Bryan Meyers, Jef Sutherland: This book brings the RPG language to the GUI world and lets you use your existing knowledge to develop Windows applications. This rich, full-featured development environment provides all the tools necessary to build Windows applications using DB2 UDB for iSeries files and other OS/400 objects, including programs, data queues, and data areas.
Part 4

Bringing your existing applications to the Web

This part provides information about how you can Web-enable your iSeries applications with the WebFacing tool and iSeries Web development tools.
WebFacing Tool

The IBM WebFacing Tool is part of the IBM WebSphere Development Studio Client for iSeries (Development Studio Client). It allows you to quickly Web-enable existing iSeries applications to Web users with no or minimal modifications to the original application. It provides a simple mechanism for facing existing 5250 applications with HTML user-interfaces, as explained in this chapter. This allows users to interact with the same application from a Web browser and existing green screens. See Figure 4-1 on page 68 for an overview of how the WebFacing Tools works.

The WebFacing Tool consists of two parts:

- A display file (DSPF) to JavaServer Pages (JSP) conversion tool
- A run-time intercept to enable the iSeries application to be run as a Web-enabled application as well as a 5250 display device application

The WebFacing Tool allows users to convert existing 5250 display file source (DDS) to JSPs and associated Java Beans. The user interface is converted to JSPs only once, at development time. The WebFacing Tool is not a 5250 emulation or screen-scraper product. This approach provides significant performance improvements over the "screen-scraping" approach that converts 5250 data streams to HTML on the fly. The WebFacing Tool converts the DDS source. Therefore, it is not necessary to understand the program logic of the legacy software on the host.

The JSPs maintain the look and feel of the original application. Java Beans are used to communicate input and output data between the JSPs and the original iSeries application. After generating JSPs and Java Beans, they are packaged as a Web application and deployed to the WebSphere Application Server running on any platform.

At run time, users invoke the iSeries application from a browser or traditional 5250 display device. The invocation method is detected by the iSeries, and a switch is set that enables the workstation manager to identify whether the application was invoked from a Web browser or 5250 display device. Accordingly, the iSeries sends or receives the program data to or from the Web browser or to create a 5250 data stream for transmission to the 5250 display device.
When the WebFacing run time outputs data from the program, it converts it into a Java Bean and sends it to the appropriate JSP for displaying in a Web browser. When the WebFacing run time receives a Java Bean from the browser, it converts it into a program buffer that can be understood by the original iSeries program.

10.1 Sample application

We use the Order Entry sample application to demonstrate how to use the WebFacing Tool to convert the 5250 interface to a browser-based user interface. The sample application is described in Chapter 6, “Introduction to the sample applications” on page 105. The screens we convert are shown in Figure 6-3 on page 108 and Figure 6-4 on page 108.

10.1.1 Using the WebFacing Tool to convert DDS

This section takes you through the steps necessary to convert a 5250 interface to a browser-based user interface.

Creating a project

A project is the highest level of a unit within the IBM WebSphere Studio Site Developer. A project contains all the objects needed to develop an application. To create a project, follow these steps:

1. Start WebSphere Studio Site Developer Advanced on your workstation.
2. Click Open Perspective, as shown in Figure 10-1.

![Open Perspective](image)

3. Select WebFacing as shown in Figure 10-2, and click OK. The WebFacing perspective opens. The first time you use the Site Developer, the WebFacing Project panel is empty.
4. Click the **Open The New Wizard** icon as shown in Figure 10-3.

5. Select **WebFacing** in the left panel and **WebFacing Project** in the right panel as shown in Figure 10-4. Click **Next**.
6. In the next window, enter the project name and enterprise application project name. Optionally, you can modify the context root for the new Web application. Click **Next**.

7. In the next window, select the connection from the Connection drop-down menu. If no connection is defined, click **New** and proceed with the wizard.

8. In the Library filter drop-down box, type the name of the library that contains the file you want to convert. Click **Refresh list**.

9. Select the library (you should have created a connection to your iSeries prior to this step).

10. Click the **Refresh list** button. If you are not yet connected to the iSeries, a logon window opens. Enter the iSeries password and click **OK**. The libraries entered in the Library filter are retrieved from the iSeries server as shown in Figure 10-5.
11. Expand the library tree. The panel displays the source files in the library.

12. Expand the source file tree. The panel displays the source members of the display files in the source file.

13. Select the source members you want to convert.

14. Click the >> button. The selected source members are copied to the right panel, as shown in Figure 10-6.

**Note:** To select all source members, select the source file and click the >> button.

Click **Next**.
15. In the next window, select the UIM source members you want to convert, as shown in Figure 10-7. Click **Next**.
16. Specify the CL command to call on the iSeries. This is the CL command that starts the program that displays the screen. The name of the command label is only to identify the command.

17. Click **Add**. In our example, we want to start the ORDENTRCL program (Figure 10-8). You can select the **Sign-on with specified values** radio button and provide a user ID and password. With these settings, you don’t need to sign on every time you start your WebFacing application. However, you don’t want leave these settings in the production version of an application. Click **Next**.
18. Select the Web style you want to use. In the Site Developer, several styles are predefined. You learn how to create your own style in 10.1.3, “Creating your own styles” on page 190. In our example, we select style spen. Click Next.

19. Click Finish. The project is created and the references to the selected sources are maintained in the workbench.

Converting the display files

The WebFacing Conversion Tool is not a screen scraper. The sources of the display files are copied into the workbench. By right-clicking CL Commands or DDS, you can add and delete source members to and from your project.

To convert the project, follow these steps:
1. Right-click the project.
2. Select Convert from the pop-up menu. The contents of the project are now converted.

After the conversion process has completed, a conversion log is displayed in the right panel (Figure 10-9).

Note: In the right panel, more than one tab can be opened with conversion logs. Be aware that the tab with the correct conversion log is opened.
Java classes and JavaServer Pages are created based on the DDS source file. By expanding DDS, you see the display files that are converted. By expanding the display files and the record format, you see the generated Java classes and JSPs. You can open the source code editor for the Java classes and the JSPs by right-clicking the object and selecting Open or Open with. Select the editor you want to use. In the right panel, the source is opened in the editor you have chosen.

You can modify the source code. But when the project is converted again, the code is overwritten with the new generated sources. You can convert individual files by right-clicking the separate display files and selecting Convert. The WebFacing Conversion Tool converts only the selected display file.

Note: The conversion log reports that all display files are converted, even if you selected only one display file for conversion. The WebFacing Conversion Tool only converts the display files you selected.

10.1.2 Testing your application inside Development Studio Client for iSeries

Development Studio Client for iSeries includes WebSphere Application Server Advanced Single Server Edition. You can unit test your application within Development Studio Client for iSeries. Perform the following steps:

1. Start the WebFacing server on iSeries. It is one of the TCP/IP servers, so you can use either a 5250 or Operations Navigator interface.
2. Development Studio Client for iSeries opens the Server perspective (Figure 10-10). Expand **MyProject-> webApplication**.

3. Right-click **index.html** and select **Run on Server**.

![Figure 10-10   Starting the test environment](image)

4. The test environment starts. The tool creates a server configuration with the default values. This is the easiest way to create such a configuration. Wait for the index.html page to appear.

5. Click the **Server** tab at the bottom of the workbench window.

6. Right-click the server and select **Stop**.

7. Modify the server’s configuration to use the appropriate encoding. Expand **Server Instances** in the Server Configuration view.

8. Double-click **WebSphere v4.0 Test Environment** (see Figure 10-11).
9. The WebSphere configuration opens in the editor's view. Click the **Environment** tab.

10. Click **Add** to add the following property (Figure 10-12):
   - **Name**: client.encoding.override
   - **Value**: UTF-8

   Click **OK**.

![Add System Property](image)

*Figure 10-12  Setting the Java Virtual Machine (JVM) property*

11. Save the configuration information.

12. Right-click **index.html** and select **Run on Server**.

13. When **index.html** is displayed in the browser window, select the first link.

14. It takes a while to start the application before the first screen (Figure 10-13) appears. Explore the Order Entry application once it starts.
10.1.3 Creating your own styles

WebSphere Development Studio Client for iSeries allows you to modify the look and feel for the Web pages generated by the tool. You can modify the existing style or create the new one. To modify the existing style, follow these steps:

1. Switch to the WebFacing perspective.
2. Right-click the project name and select Properties. In the Properties for MyProject window (Figure 10-14), you should see the number of parameters you can modify.
3. Select **DDS field color**. You can customize the color for any DDS field color (Figure 10-14):

   a. Select a desired DDS color from the DDS field color drop-down list.
   
   b. In the Field color section of the properties window, click the .. button to the right of the Text box.
   
   c. Select desired color and click **OK**.
   
   d. Repeat this operation for the Text and Background color parameters of the Reverse image color - DSPATR(R)I section.

4. Select the **DDS field display attributes** property in the left part of the properties window.

5. Choose a font and font size for the DDS fields by clicking the appropriate .. button.

6. Select the **Window** property in the left part of the properties window.

7. Select **Title** in the Window areas drop-down menu.

8. Modify the appearance of the title by changing the color and the font in the Foreground section of the window (Figure 10-15).
9. In the Background section of the window, click the .. button to the right of the Image field.

10. Navigate to the directory
    C:\WDSC\WSSD\plugins\com.ibm.etools.iseries.webfacing\styles\metal\apparea and select `texture.gif`. Click OK. This specifies the background image of the page.

11. Repeat this operation for the Body windows area.

12. Select the **Subfile** property in the left part of the Properties window.

13. Modify the appearance of the subfile:
   
   a. Click the **Up arrow** from the Subfile areas drop-down list.
   
   b. Click the .. button to the right of the Image field.
   
   c. Select `UpArrow.gif` and click OK.
   
   d. Select the other areas of the subfile and change the color or the image (`DownArrow.gif`). The Style properties allow you to preview the changes in the Properties window (Figure 10-16).
14. Select the **Command keys** property.

15. Change the font and the font color in the Foreground section of the window.

16. Change the button’s image in the Background section of the window by clicking the .. button and navigating to the location of the GIF file for a command key button.

17. Select **Rollover** or **Button down** in the Command key states drop-down menu and change the appearance of the button (see Figure 10-17).
18. Click **OK**. The changes are saved to the project.

19. Test the application. Figure 10-18 shows the affect of our changes to the subfile.
Chapter 10. WebFacing Tool

10.1.4 The user ID and password the application runs under

The WebFacing server run time does not require that the application runs under the profile of the developer who does the conversion. The converted application can run under any OS/400 profile that has 5250 access. However, the WebFacing Tool has a way to save this information in the project’s properties.

To select the way your WebFacing application processes the signon parameters, follow these steps:

1. Go to the WebFacing perspective. Right-click the project and select Properties.
2. In the Properties window, select Project.
3. Select one of the following check boxes:
   - Retain user ID and password for subsequent prompted signon: Specify a user ID and password and click OK. This option is helpful when you develop the application and test it frequently. When you switch to the production environment, choose Prompt for signon.
– **Prompt for signon**: With this option, you are prompted for a user ID and password every time you run the application.

### 10.1.5 Exporting a WebFacing project

After the application is converted, you have to deploy it to the server. To deploy the Web application to the server, a Web Application Archive (WAR) file is used. A WAR file contains a group of HTML pages, JSPs, servlets, and other resources (as an option, you can include the source code in the WAR file). In addition to project resources, the WAR file includes a *Web deployment descriptor file*. This is an XML file that contains deployment information for a Web application. The Web deployment descriptor file (web.xml) provides information about the WAR file. In our example, we deploy the application to the host server using a file share. See 10.1.8, “Creating a file share in the iSeries integrated file system (IFS)” on page 203, which explains how to set up a file share.

The Web development tool provides facilities for importing and exporting WAR files, using the following wizards:

- **Import Resources from a WAR File**: Allows you to specify a Web or Enterprise Application project, along with some optional settings.
- **Export Resources to a WAR File**: Requires only an export location and some optional settings.

To export the objects of our sample application, we use the Export wizard:

1. Select the Web project you want to export.
2. Click **File-> Export** as shown in Figure 10-19.

![Figure 10-19 Selecting Export to deploy the project]

3. In the Export window (Figure 10-20), select **WAR file** and click **Next**.
4. Enter the name of the project and the destination WAR file. In our example, the destination WAR file is:

Q:\UserData\WebASAdv4\default\installableApps\MyProject.war

5. Click **Finish**.

The application is now packaged in the WAR file and exported to the destination folder on the iSeries.

### 10.1.6 Deploying a Web application with WebSphere Administrative Console

After a Web application is created and exported to the server, you need to deploy your Web application on the WebSphere Application Server. The Web application is deployed using the WebSphere Administrative Console. Follow these steps to deploy the WAR file:

1. To launch the WebSphere Administrative Console, open a Windows Command Prompt window and change to the WebSphere Application Server bin directory. For example, go to the directory C:\WebSphere\AppServer\bin, which is the installation directory that contains the `adminclient` command. Then run the command:

   ```
   adminclient iSeries hostname bootstrap port
   ```

   Note the following explanation:

   – *iSeries hostname* is the host name of your iSeries server.
   – *bootstrap port* is the port that you chose for your administrative instance.

   In our example, this command would be:

   ```
   adminclient asm03 999
   ```

   The port number to use is the bootstrap port you selected in the optional step for setting up a separate administrative instance.

   __Note:__ The host name for your iSeries server is case sensitive. To see what this value should be, run the CFGTCP command from an iSeries command line. On the Configure TCP/IP menu, choose option 12 (Change TCP/IP Domain) and make note of the Host name value. This is the value, without the quotation marks, that you must specify as your host name.

The Administrative Console then opens.
2. From the Administrative Console, select **Console-> Wizards-> Install Enterprise Application** (see Figure 10-21) to launch the Install Web Application wizard.

![Figure 10-21 Opening the wizard to install the enterprise application](image)

3. Select **Install stand-alone module (*.war, *jar)**.

4. Enter the path and WAR file name in the field path (optionally, you can use the **Browse** button to navigate to the WAR file). In our example (see step 4 on page 197), we enter:

   `/QIBM/UserData/WebASAdv4/default/installableApps/MyProject.war`

5. Enter the name of the application. In our example, enter **MyProject**.

6. Enter the name of the context root including a forward slash "/". In our example, we enter `/MyProject`.

   The context root is combined with the defined servlet mapping (from the WAR file) to compose the full URL that users type to access the servlet. For example, if the context root is `/gettingstarted` and the servlet mapping is `MyProject`, then the URL is:

   `http://host:port/gettingstarted/MyProject`

7. Click **Next** on the following windows until you reach the Selecting Application Servers window.

8. Select the correct server (in most cases, **Default Server**) and click **Next**.

9. On the next window, click **Finish**.

10. Wait until the Information dialog appears, as shown in Figure 10-22.
11. Add a new JVM property for the application server:
   a. Click the **JVM Settings** tab.
   b. Add the following property as shown on the JVM Settings page (Figure 10-23):
      - **Name:** client.encoding.override
      - **Value:** UTF-8

12. In the Administrative Console, right-click the node and select **Regen Webserver Plug-in** from the pop-up menu. This updates the Web server configuration with the information about the new application.

The Enterprise Application is now installed on the application server. To start the enterprise application, follow these steps:

1. Expand **WebSphere Administrative Domain**.
2. Expand the **Enterprise Applications**.
3. Right-click the application name and click **Start**. The Application Server for your enterprise application starts.
When the `start` command is completed, the Information dialog (Figure 10-24) opens and confirms a successful completion.

![Information dialog](image)

*Figure 10-24  Enterprise application started*

The message log of the Administrative Console also contains the start of the enterprise application as shown in Figure 10-25.

![Log of the Administrative Console](image)

*Figure 10-25  Log of the Administrative Console*

### 10.1.7 Launching the converted application in a browser

To launch your WebFacing Web application, ensure that the WebSphere Application Server and the Web application that you created for your WebFacing project are started.

![WebSphere Advanced Administrative Console](image)

*Figure 10-26  Default server*

You also need to start the WebFacing server on iSeries. It is one of the TCP/IP servers, so you can use either 5250 or Operations Navigator interface.

1. Type the following URL in a browser location field:

   `http://host_name:host_port/webapp_name/`
Note the following explanation:

- `host_name` is the name of your iSeries server, which in our example is `asm03`.
- `host_port` is the port that you assigned to your HTTP server instance if you created a separate instance for the WebSphere administrative server. If you are using the default instance, the port is 80, unless the default port number has changed. If you are using the default HTTP port 80, you can leave out the `:host_port` portion of the address you enter into the location field. Also, you can use WebSphere Application Server’s internal HTTP server to test your applications. The default internal port number is 9080.
- `webapp_name` is the name you supplied for your application. In our example, the name of the application is `Myproject`.

In our example, the URL would look like this:

```
http://asm03:9080/MyProject
```

The Web path name is case sensitive and must match the values specified when the Web application was created in the WebSphere Administrative Console. These addresses point to the index.html file that was created by the WebFacing Tool in the path `Q:\UserData\WebASAdv4\default\InstalledApps\MyProject.war`. You can also specify the index.html file directly as part of the address, for example:

```
http://asm03:9080/MyProject/index.html
```

Figure 10-27 shows the browser window for our sample application.

**Figure 10-27   Browser example application**

2. Click the first link. If you didn’t specify the sign-on information during the development of the WebFacing project, you are presented with the logon page. Enter your user ID and password and click **Logon**. The application starts on the iSeries host. The initial screen (Figure 10-28) is displayed in a browser.
3. In our example application, we also converted the prompt window. To launch this screen, press F4 (Prompt), and you see the Select Customer screen (Figure 10-29).
4. Refer to the 5250 interface in Figure 6-3 and Figure 6-4 on page 108. Compare it with the Web interface shown in Figure 10-29. You can use both interfaces to maintain the database.

10.1.8 Creating a file share in the iSeries integrated file system (IFS)

Deploying your project to the WebSphere Application Server on your iSeries server requires you to set up a file transfer method between your workstation and your server. This involves creating a file share in the iSeries IFS and mapping a drive letter on your workstation to the file share. By doing this, you can export files from your workstation to your server in the same way that you can copy files from one folder to another on your workstation using Windows Explorer.

This section provides some basic information about setting up a file share on the iSeries server and mapping a workstation drive letter to the file share. Links are provided for additional information.

WebFacing projects are exported to a WebSphere Application Server directory in the IFS. For example, if you created a WebSphere Application Server instance for your project, the IFS directory would be similar to this example: /QIBM/UserData/WebASAdv4/instance name.

In Operations Navigator, you can create a file share of the /QIBM directory. To do this, in Operations Navigator, drill down to the /QIBM directory in the IFS, right-click /QIBM and select Sharing-> New Share.
After you create a share, choose one of the following methods to map a drive letter to it:

- From the Windows Start menu:
  a. Click **Start-> Search-> For Files or Folders**.
  b. In the Search box, under Search for other items, click **Computers**.
  c. In the Computer Name field, enter the name of your iSeries server.
  d. When your system finds the server, double-click your server and supply a valid user name and password to access it. You should now be able to explore the server's integrated file system, which contains the share you created.
  e. The share is listed by the name you gave to it in the New Share window. Right-click the share and select **Map Network Drive**.
  f. Choose an available drive letter for the share, for example **Q**.

- Using Windows Explorer:
  a. Click **Tools-> Map Network Drive**. In the window that opens, choose a drive letter to map to and enter the following text in the path field:
     \iSeries server name
  b. Supply a valid user name and password.
  c. After you successfully sign on, choose the file share that you created with Operations Navigator.

- From a Windows command prompt window:
  You can use the Windows **net use** command to map a file share.

**Enabling support to access IFS from Windows Explorer**

Follow this process:

1. Ensure that NetServer is installed on your iSeries server. You can find information on NetServer at the following Web sites:

   NetServer is often installed by default on many systems, so you may not require an installation step.

2. Start NetServer. You can start it from an iSeries server command line by entering:
   STRTCPVR SERVER(*NETSVR)

3. Create a file share in the iSeries IFS so that a workstation drive letter can be mapped to it later. You can create file shares by using Operations Navigator. For more information about Client Access and Operations Navigator, refer to these Web sites:
   http://www.ibm.com/servers/eserver/iseries/oper_nav/

4. Map a workstation drive letter to the share. For example, if drive letter Q:\ is available on your workstation, map that drive letter to the /QIBM directory in the IFS. That way, any WebSphere Application Server files that are in an IFS directory path, such as /QIBM/UserData/WebASAdv4/default, can be accessed as Q:\UserData\WebASAdv4\default.
10.2 Security considerations

When you implement an iSeries-based WebSphere application, you need to consider security issues. Connecting to the Internet opens the system to many other systems on the network. This leaves you, as a business user of the Internet and even of your own intranet, vulnerable to the whims of hackers and industrial spies. You need to take action to prevent their activities.

Security is a baseline requirement for network computing. Privacy, authentication, authorization, and integrity are all important elements of any security strategy. They work together to help in your defense against threats. Security can be set up in the WebSphere Application Server, the iSeries and in the Web application.

Security is an important, complex, and broad subject and the topic of many books. For more details on the WebSphere security issues, refer to these Redbooks:

- IBM WebSphere V4.0 Advanced Edition Security, SG24-6520
- WebSphere J2EE Application Development for IBM @server iSeries Servers, SG24-6559
- WebSphere 4.0 for iSeries Installation and Configuration, SG24-6815

10.3 Hints and tips for working with the WebFacing Tool

This section offers several tips and hits for optimal use of the WebFacing Tool.

Performance

For the example in this chapter, we used a small and simple application to show how the WebFacing Tool works. In real life, you will convert much bigger and more complex applications. For testing purposes, we converted an application with 383 DDS sources containing 1,311 record formats. The conversion needed 1 hour and 55 minutes to convert all sources. Deploying the project to the IFS of the iSeries host needed 45 minutes.

This is not an official benchmark for conversions. The time needed for conversions depends on several circumstances, including:

- Availability of the network
- Line speed
- Configuration of the hardware

WebFacing Environment API

Although there is no need to change your existing application on the iSeries to access the application using a Web browser, you may want to know whether your application is accessed through the 5250 emulation or through a Web browser. The WebFacing Environment API enables you to check how your application is accessed. Use this API when you want to change the behavior of your program according to the type of access a user has.

The WebFacing Environment API is called QqfEnvironment and is part of the WebFacing server run time. The external procedure name QqfEnvironment is case sensitive. It is a procedure packaged in a service program called QQFENV that is located in the QSYS library. The API returns 1 if the application is running under WebFacing and 0 if it is running under 5250 emulation.

You can learn more about the WebFacing Environment API in the online help. Select WebFacing Tool-> Using the WebFacing Environment API.
This chapter explains how to use WebSphere Development Studio Client for iSeries (Development Studio Client) to convert a 5250 application into a Web-based application. It focuses on using design-time control (DTC), Web Interaction wizard, and Program Call beans.
11.1 Introduction to design-time control

DTC gives you the ability to define iSeries data fields, push buttons, radio buttons, combo boxes, lists, subfiles, and tables. These special features are used to inform WebSphere Development Studio Client for iSeries that these items are to communicate between the Web page and the iSeries host program.

This chapter focuses on using WebSphere Development Studio Client for iSeries to convert an iSeries host application, written in pure RPG, into a Web-based interface application. (The RPG uses display files for inputting data.) You can learn more about the iSeries host application in 6.2, “Overview of the Customer Master Inquiry” on page 111.

There are three steps to convert the iSeries host application to the Web with WebSphere Studio for iSeries:

1. Create JSPs using Development Studio Client and DTC.
2. Create interaction between host programs and the Web application using the Web Interaction wizard.
   The Web Interaction wizard creates the interaction between the Web pages and iSeries host programs. It is used to define the input, output, and error Web pages and to identify the host program to call for a particular interaction. It also allows you to identify the parameters passed to the host program and to map the input and output fields from the pages to the host program parameters.
3. Export the Web application to a Web Archive (WAR) file.

The application developed by WebSphere Development Studio Client for iSeries is defined as:

- One input browser page that provides input to the iSeries host program
- Two output browser pages to receive output from the iSeries host program

When a client sends a request to the Web application deployed in the WebSphere Application Server, a session between the client and the iSeries server is established. This session remains active until either the user signs off the application under the host program control, or the session manager determines that the browser user has become inactive. Figure 11-1 shows how the application generated by WebSphere Development Studio Client for iSeries will run.

You can find more information about design-time control in the WebSphere Development Studio Client for iSeries online documentation.
11.2 Developing the Web application

To build the Web application based on the 5250 customer inquiry application, you need to perform the following steps:

2. Define how the Web pages interact with the iSeries host programs via the Web Interaction wizard. See 11.2.2, “Using the Web Interaction wizard” on page 227.
3. Export Web application to a WAR file. See 11.2.5, “Exporting the Web application to production” on page 247.

11.2.1 Using design-time control

This section explains how to design the Web pages and define the input and output fields. Follow these steps to perform the task:

1. Start the WebSphere Development Studio Client for iSeries. Click Start-> Programs-> IBM WebSphere Development Studio Client for iSeries-> IBM WebSphere Studio Site Developer Advanced.
2. Open Web Perspective from the WebSphere Studio Site Developer Advanced toolbar. Click Perspective-> Open-> Other. On the Select Perspective window (Figure 11-2), select Web. Click OK.
3. Create a new project:
   a. Click **File-> New-> Web Project** as shown in Figure 11-3.

   If the Web Project option is not available, click **New-> Other**.

   b. From the New window (Figure 11-4), click **Web** in the left-hand panel and click **Web Project** in the right-hand panel.
4. Define the Web project (Figure 11-5):
   a. Type the name of the project in the Project name field. In our example, we use Cusinq.
   b. Leave **Use default location** selected as is.
   c. Enter an Enterprise Application project name. We use CusinqEAR for this example. We want to keep the data for a Web project separate from another Web project.
   d. Leave Context root with the same name as the project. The context root is the Web application root, which is the top-level directory of your application when it is deployed to a Web server.
   e. Click **Finish**.
5. Create input and output JSPs. Click **File -> New -> JSP File** as shown in Figure 11-6.

![Figure 11-5 Defining the Web project](image1)

![Figure 11-6 Creating a new JSP](image2)
6. On the Create a JSP File window (Figure 11-7), select the folder where you want the JSP to be stored and type the name for the JSP file. Click **Finish**.

![Create a JSP File](image)

*Figure 11-7  Creating a JSP file*
7. The Design window (Figure 11-8) opens automatically. If it does not, double-click CUSINQ.jsp.

Figure 11-8   JSP Design view
8. Complete the design of your page. We added some icons and a title to the page as shown in Figure 11-9. Now you are ready to begin defining the fields.

Figure 11-9 Example page design
9. Click **Insert-> Form and Input Fields-> Form** to insert a form in your page, as shown in Figure 11-10.

![Figure 11-10](image1)

**Figure 11-10** Adding a form

10. Inside the Form, type **Enter ZIP Code**.

11. With the cursor positioned right after the text you just typed, click **Insert-> Design-time Control**. The Design-time Control window (Figure 11-11) opens.

![Figure 11-11](image2)

**Figure 11-11** Design-time Control window
The Design-time Control window lists the iSeries DTCs that you can use to communicate with the host ILE program.

**Note:** Design-time controls must appear inside a form. Otherwise, the information associated with them is not sent to the iSeries server.

12. The iSeries DTCs include list box, combo box, and check box. Three of the more unique DTCs are:

- AS/400 Subfile DTC: Used to define subfiles
- AS/400 Entryfield DTC
- AS/400 Table DTC

Select the **AS/400 Entryfield DTC** to define the input entry fields for the Web page. The outline of the entry field appears within the form as shown in Figure 11-12.

13. Right-click the data entry box. Select **Control Properties** from the pop-up menu. The **AS/400 Entryfield DTC Properties** window opens as shown in Figure 11-13.
14. The Properties window allows you to define the Web page field characteristics. Note that we are defining the display characteristics of the field, and not whether the field is to be used for input or output. This is done later by using the Web Interaction wizard.

Since we are designing the input page for our sample application, we define the ZIP Code input field. You can define how the field should be edited and whether it should be a hidden field.

**Note:** You may also choose to import the definitions from the DB2 UDB for iSeries database. If you click the *DB Reference* button, you go through a series of displays and prompts that allow you to connect to the iSeries server and retrieve the characteristics of a data field (Figure 11-14).

Using the Events tab, you can define certain actions on this field to cause an associated JavaScript to execute. The developer must supply the desired JavaScript.

Click the **OK** button.
15. You have now defined how the Web page will look and the data fields that you want on the Web page. Now define the action that should be taken. Place the cursor after the entry field that was just defined and press Enter. The cursor is now positioned under the Enter ZIP Code text (make sure you are in the Form).
16. Define the Submit button that will cause the Web page to be submitted. Click **Insert-> Form and Input Fields-> Submit Button**. The Attributes window (Figure 11-15) opens.

In this window, assign a name of **Submit** to the button. The contents of the Label field will appear on the Web page. Click the **OK** button.

![Figure 11-15  Button attributes](image-url)
You are now finished designing the page. In the Design window (Figure 11-16), click the Source tab to review the HTML code that was generated.

Figure 11-16   Generated HTML code
18. Click the **Preview** tab (at the bottom of the HTML editor view) to review how the page will look in a browser (Figure 11-17).
For our sample application, we designed another page that corresponds to the second 5250 screen. This page contains a subfile definition as shown in Figure 11-18.

**Figure 11-18  Subfile JSP**

19. Create a normal JSP.
20. Add headings as we did in previous JSP creation.
21. Insert FORM or press Ctrl+O.
22. Insert the text **AS/400 Subfile DTC** into the form.

**Note:** It is very important to place Subfile DTC and all fields on the page into the form.
23. Modify the subfile DTC general properties as shown in Figure 11-19 using the values from the following list:

- **Subfile ID**: SFL1
- **Server service program library** and **object**: AS2314 CUSTINQSFL
  
  This program sets the initial zip value, creates the subfile user space, and fills the subfile with all records from the database.

- **Labels**: Enter labels of the subfile control buttons and the label of the selection column.

- **Subfile page size**: Enter the number of rows.

![Figure 11-19 AS/400 Subfile DTC Properties - General](image-url)
24. Click the **Parameters** tab. You can see the Parameters page in Figure 11-20.

25. Under Parameter properties, define parameters that are passed to your host program.
26. Click the **Fields Properties** tab (Figure 11-21), and define the subfile fields:

- ZIP to display the ZIP code (zoned decimal 5,0)
- CUST to display the Customer Number (zoned decimal 5,0)
- NAME to display the Customer Name (character 20)

The last page we designed is shown in Figure 11-22. This page displays the customer details. It corresponds to the 5250 screens shown in 6.2, “Overview of the Customer Master Inquiry” on page 111.
Chapter 11. iSeries Web development tools in Development Studio Client

227

27. Save the pages you designed.

The pages are now designed and saved. You are ready to proceed to the next step of publishing the Web page.

11.2.2 Using the Web Interaction wizard

The basic architecture provided by WebSphere Development Studio Client for iSeries includes an input Web page with input fields, an output Web page with several output fields, and an associated error page. The basic elements that we deal with and try to tie together are:

- Web pages and their input and output fields
- The parameters that are passed to and from the ILE program

In our example, we use the Web pages that we designed earlier to create two Web interactions:

- **getList Web interaction:** We use the **Cusinq.jsp** Web page as an input page with one input parameter (ZIP code). It also uses the **Cussfl.jsp** Web page as an output page with no parameters required because the subfile service program CUSTINQSFL is handling the displayed subfile.
**getDetail Web interaction:** We use the Cussfl.jsp Web page as an input page with the following parameters:
- **CUST:** Output, Zoned decimal 5,0 for the customer number field
- **NAME:** Output, Char 20, for the customer name field
- **ZIP:** Output, Zoned decimal 5,0 for the ZIP code field
- **SFLNAM:** Input, Char 10, for the subfile name (SFL1)

The getDetail Web interaction uses the *displaycustdetailResults.jsp* Web page as an output page.

Figure 11-23 shows the relationship between:
- Web pages CUSINO.jsp, CUSSFL.jsp, and displaycustdetailResults.jsp
- Web interactions getList.wit and getDetail.wit
- Web error pages getListError.jsp, getDetailError.jsp, and displaycustdetailError.jsp

![Diagram: Web interactions and Web pages](image)

**Figure 11-23  Web interactions and Web pages**

To create the Web interactions, follow these steps:

1. Select the **Cusing** project in the left pane of the WebSphere Development Studio Client for iSeries. Right-click and select **New-> Other-> Web-> Web Interaction**. Then the Web Interaction Wizard window (Figure 11-24) opens.
2. Click the Browse button to select the Destination folder. The Folder Selection window (Figure 11-25) opens.

3. Select the folders Cusinq-> webApplication. Click OK.

4. Back on the Web Interaction Wizard window, type the name for the Web interaction that you are about to define. We use getList as the name.

   Java package and Use error page are optional so you do not need to define these.
The *Invalidate session after the interaction occurs* option allows you to inform the session manager that the session should be terminated after this interaction. This is specified on the last interaction in a series of interactions. The getList interaction is not our last one, so do not select the check box. Click **Next**.

5. On the next window (Figure 11-26), specify which input and output pages you want to use or whether you want WebSphere Development Studio Client for iSeries to create the pages for you.

![Web Interaction Wizard](image)

**Figure 11-26  Input and output pages**

6. We want to use the pages we created earlier for input and output. To do this, click the **Use input pages** check box. Click **Add**.

7. From the Input page window (Figure 11-27), select `cusinq.jsp` for input and click **OK**.

8. On the Output page window, select the output page `Cussfl.jsp`. Click the **Preview** button if you want to preview the pages.
9. Click **Next** on the window shown in Figure 11-26.

10. On the next window (Figure 11-28), the program alias is the name of the Java program that will generate and call our ILE program. We use the same name as the ILE program.
The program object is the actual ILE RPG program name from the iSeries server. Click the **Browse** button to select the object (Figure 11-29) or type the program and library names into the fields. We clicked the Browse button. Click **OK**.

![Browse For Object](image)

**Figure 11-29 Browse For Object**

11. Click the **Add Program** on the input and output parameters window (Figure 11-28) to add the program to the program call definitions.

After you define the ILE program to call, define the parameters that are passed to the program. The number of parameters and their characteristics must match the ones defined in the ILE program. This refers to the parameters that you pass to the programs. The steps allow you to link these parameters with the input and output fields on the Web pages.

You can define data fields and structures that can be passed as parameters to the ILE programs. A *structure* is a collection of data fields that you can define and store for reuse in another interaction later. Only seven fields can be passed as parameters to the ILE program, and these fields may be structures.

**Note:** If you select Cusinqnew, and select Browse iSeries native files from the Source location pull-down menu, you can define the path to your ILE program source code. Then you can click the View button to browse your code for reference as you define the parameters.
12. Select **Cusinqnew** and click **Add Parameter**. Complete the fields in the right pane as shown in Figure 11-30.

The following Zips parameter is passed to the example ILE program CUSINQNEW. Its data type is set to **zoned decimal**, its length is set to 5, and Precision is set to 0.

![Figure 11-30 Adding a parameter](image)
You can also obtain the parameter from the database (Figure 11-31):

a. Select the program under Program call definitions and click **Add Parameter**.

b. Expand a connection-> your filter-> your library-> your record-> your file.

c. Select a field.

d. Click **Add**.

You can select multiple fields at one time.

![Figure 11-31 Adding a database field](image)

13. After you select all of your input parameters in window, specify the input and output parameters for your iSeries Host Program. Click **Next**.
14. On the next window (Figure 11-32), all input fields from the JSP page are in the left pane, and all of the input fields you defined in step 12 are in the right pane.

Figure 11-32  Map and link input parameters

a. Set the Save to Session parameter to true.

b. Select matching parameters from both sides and click the Link button to link the parameters. The icons next to the input parameter names represent the type of the field. Click Next.
15. On the next window (Figure 11-33), you do not need to link any of the parameters. Click Finish.

Figure 11-33 Mapping and linking output parameters

The WebSphere Development Studio Client for iSeries starts creating servlets, JSPs, beans, and Java code.

After you create the getList.wit Web interaction, you must create the getDetail.wit Web interaction:

1. Select File-> New-> Other.
3. The destination folder is the same as for the first Web interaction. The Web interaction name is getDetail. Click Next.
4. Select Cussfl.jsp for the input page and displaycustomerdetail.jsp for the output page. Click Next.
5. Click **Add Program** and type `getDetail` into the Program alias field. The program name is `GETDETAIL` and the library is `AS2314` (Figure 11-34).

![Image of Web Interaction Wizard](image)

**Figure 11-34**  `getDetail` Web interaction

6. Under the `getDetail` program, add the following parameters:
   - Input parameter `SFLNAM` (char 10) to hold the subfile name and pass the argument into the `GETDETAIL` ILE RPG program.
   - Output fields `CUST`, `NAME`, `ADDR`, `CITY`, `STATE`, and `ZIP`.

Do not forget to select the **Associate this program with the interaction** check box. Click **Next**.
7. Link the input parameters to the input fields (Figure 11-35). Click **Next**.

![Web Interaction Wizard](image)

**Figure 11-35**  *getDetail interaction: Input parameters*
8. Link the output parameters to the output fields (Figure 11-36). Click **Finish**.

![Web Interaction Wizard](image)

**Figure 11-36**  getDetail interaction: Output parameters

Now the getDetail.wit interaction is ready.

### 11.2.3 Testing application in Development Studio Client

You can unit test a Web application that communicates with an ILE program on an iSeries host without deploying the JSP files and servlets to WebSphere Application Server. To use this feature, you need to define information about the iSeries server that will be used for testing. You define the communication values with the **iSeries Host Information wizard**.
1. To open the wizard, right-click the Web project and select **iSeries host information setup** (Figure 11-37).

---

**Figure 11-37  iSeries host information setup**
a. Fill in your iSeries host name, iSeries User ID, and password.
b. Click **Add** to add the library names to the run-time library list. Type the library name and press Enter.
c. Click **Finish** (Figure 11-38).

*Figure 11-38  iSeries Host Information*
2. Now you can run the Web application. Running the Web application locally uses the default WebSphere test environment. Right-click **Cusinq.jsp** under Cusinq Web project in the Navigator view and select **Run on Server** (Figure 11-39).

![Figure 11-39 Selecting Run on Server](image)

The Server Tools feature automatically does the following tasks for you:
- Launches the Server Perspective window.
- Creates a server project with **Servers** as the default name.
- Creates the server instance with **WebSphere V4.0 Test Environment** as the default name.
- Creates the server configuration with **WebSphere Administrative Domain** as the default name.
- Sets the server instance to use the server configuration.
- Adds your project to the server configuration.
- Starts the server instance (this may take time to start depending on the server configuration settings).
- Opens the Debug view and the Source pane if breakpoints are set in the file.
- Displays the file in the Web browser (Figure 11-40).
Figure 11-40  Run Cusinq.jsp

There is a console view at the bottom of the workbench window where you can see all administrative and error messages during execution (see Figure 11-41). The following parts of the output in the console window are identified:

- The iSeries server job that handles all of your requests between the Web application and the host ILE program (a).
- The input value that transferred to the ILE RPG program (CUSINQNEW) (b).
- The host program call from Subfile Design-Time Control (c).
- Writing records to a subfile using Add a record to the subfile QdtsAppendSF API (d).

Figure 11-41  Console View
Type the ZIP code into the input field and click the **Get Customer List** button. The customer list looks like the example in Figure 11-42.
3. The last screen in our application is Customer detail (Figure 11-43). Select one of the records and click **Get Detail**.

![Customer Detail](image)

After the testing is completed, your next task is to create WAR file and export it into the application server as explained in 11.2.5, “Exporting the Web application to production” on page 247.

**11.2.4 Debugging a Web application with a host program call**

Debugging a Web application with an iSeries program call is more complex than normal ILE program debugging. Because all iSeries host program transactions are handled by a server job, you need to find out the system job and attach debugger into that batch job. You can find the batch job in the Console view (Figure 11-41 on page 243).

To debug Web application and host program calls, follow these steps:

1. Start iSeries Debug Server by using the Start Debug Server (STRDBGSVR) command on the iSeries server.

2. Start IBM Distributed Debugger. Click **Start-> Programs-> WebSphere Development Studio Client for iSeries-> IBM Distributed Debugger-> IBM Distributed Debugger.**
3. Find out the iSeries server job from the Console view. Copy the job information to the clipboard and paste the information into the Job to debug field (Figure 11-44).

4. Type the program library and program name into to the Specify the name of the program to add to debug field.

   Note: To make your source code visible in the debugger, you must compile the program must using the DBGVIEW(*SOURCE) parameter.

5. Select the Step into check box.
6. Click Attach.
7. Debugger asks you to start the program (Figure 11-45). Click OK.

8. Go back to the Main page of the Web application and enter the ZIP code again.
9. Click **Get Customer List**. The debugger view is started (Figure 11-46).

![Debugger view](image)

**Figure 11-46   Debugger view**

Now you can place breakpoints, view variables, and so on.

### 11.2.5 Exporting the Web application to production

When the Web pages are designed and the Web page data fields are defined, you are ready to export application into production system. This section shows how to create a WAR file. A WAR file is a packaged Web application. WAR files can be used to import a Web application into an application server. Follow these steps to export a Web application:
1. Right-click the Web application and select **Export WAR** (see Figure 11-47).

![Figure 11-47: Export WAR](image-url)
2. Select the Web project you want to export and the location for the WAR file (see Figure 11-48). Click Finish.

![Figure 11-48   WAR file name and location](image)

You can also export the source files to a WAR file.

**Note:** The path we use corresponds to WebSphere Application Server Version 4. That is the location where WebSphere Application Server Version 4 can find the WAR file.

You can find information about how to install WAR file to WebSphere Application Server 4.0 in 10.1.6, “Deploying a Web application with WebSphere Administrative Console” on page 197.

After you deploy the WAR file, you can access your Web application from a Web browser by entering the link:

http://yoursystemname:9080/Cusinq/Cusinq.jsp

Port number 9080 is a default internal HTTP port in WebSphere Application Server.

To run application in the production system including HTTP server, refer to Chapter 4, “Setting up the iSeries environment” on page 67.

### 11.2.6 Subfile DTC APIs

Your ILE program interacts with the subfile by coding to the subfile APIs. The data for a subfile is actually stored in a user space object (*USRSPC object type) on the host. This user space is created in the QTEMP library for the user profile that is executing the ILE program.
When you create a subfile DTC in one of your JSP files, one of the properties you set is the name of the service program. This name should be called each time the subfile is to be shown or when the user presses the Page Up or Page Down buttons that are generated by the subfile. That service program should provide several procedures that will be called by the subfile DTC.

Note that these procedure names are case sensitive. Each procedure is passed two parameters. The first parameter is a 10-character string that is the name of the subfile DTC as defined in the subfile DTC control properties settings. The second parameter is a blank delimited list of the exit parameters that were defined on the subfile DTC control properties. The list of procedures includes:

- **INIT**: This procedure is called the first time the JSP page containing the subfile DTC is about to be displayed. You could use this procedure to initially fill the subfile with all records or a page worth of records.
- **PGUP**: This procedure is called when the user presses the Page Up button associated with the subfile to view the previous page of records. If previous records exist in the user space, this procedure is not called and the subfile automatically displays the previous page.
- **PGDN**: This procedure is called when the user presses the Page Down button associated with the subfile to view the next page of records. If the next page of records exist in the subfile, this procedure is not called and the subfile automatically displays the next page.
- **CLUP**: This procedure is called when the session ends and the cleanup operation is called.

The subfile APIs that are available for processing a subfile DTC are:

- QdtsAppendSF: Adds a record to the subfile
- QdtsChainSF: Chains to a specific subfile record
- QdtsClearSF: Clears the subfile
- QdtsInitSF: Initializes the subfile
- QdtsReadSF: Reads a selected record
- QdtsUpdateSF: Updates a subfile record

The host service program, QDTSSFL.SRVPGM, is required when the user host program needs to invoke the WebSphere Development Studio Client for iSeries Subfile DTC API.

The Subfile DTC APIs save file is found in the Development Studio Client folder X:\WDSC\WSSD\plugins\com.ibm.etools.iseries.webtools\lib.

**Installing the QDTSSFL program**

To install the QDTSSFL program, perform these steps:

1. You must have a host ID with *ALLOBJ and *SAVSYS authorities. This is required for the installation steps.
2. FTP the qdtssfl.sav savefile from your workstation to iSeries host QGPL library.
3. Restore the QDTSSFL SRVPGM to QGPL library from the *SAVF file by running the following command from the iSeries command prompt:

   ```
   RSTOBJ OBJ(*ALL) SAVLIB(QGPL) DEV(*SAVF) SAVF(QGPL/QDTSSFL) RSTLIB(QGPL)
   ```

**Development environment for QDTSSFL**

Check the following conditions on the iSeries server before you run QDTSSFL:

- Ensure the QGPL library is in the library list.
- Bind the user program that makes the Subfile DTC API call to QGPL\QDTSSFL.SRVPGM.
11.2.7 Example code

The base code we are using is defined in 6.2, “Overview of the Customer Master Inquiry” on page 111, and the source code is included into additional material. But we cannot use the native 5250 RPG code. Only the main calculations and file definitions can be used in Web application. All screen input and output operations are handled by the DTC API.

CUSINQNEW

This program is called from CUSINO.jsp. By calling the CUSINQSF service program, we set the initial ZIP value as well as create the subfile user space in library QTEMP and fill the subfile with all records from the database. After compiling this program, it must be bound to the CUSINQSF service program using the CRTPGM command.

Example 11-1  CUSINQNEW

```
****************************************************************
* Define prototypes for procedures
F*---------------------------------------------------------------
DsetZip      PR      ExtProc('setZip')
D             PR      ExtProc('INIT')

* Program variables
*---------------------------------------------------------------*
DSFL1       S      10A    Inz('SFL1')
DPARMS      S      5A      Inz('55901')
DZips       S      5S 0

* Define input parameter
C  *Entry       Plist
C    Parm      Zips

  * Set the component name and initialize the subfile
C    CallP     setZip(Zips)
C    CallP     INIT(SFL1:PARMS)
C  
C  Return
```

CUSINQSF

Example 11-2 shows the source of Get Customer List service program. This service program handles all requests for the subfile SFL1. When you compile this program, you must bind it with the subfile service program QDTSSFL.

Note: For a run-time environment, ensure the QGPL library is in the library list.
Example 11-2  CUSTINQSFL

H NoMain
*-------------------------------------------------------------------*
* Define the logical file (view by ZIP )                           *
FMLGMSFL1  IF  E       K Disk
*-------------------------------------------------------------------*
* Define the prototypes for the procedures in the subfile     *
* handling service program, QDTSSFL                            *
DchainSF  PR  10U 0 ExtProc('QdtsChainSF')
D          Value
D rec          Like(Record)
D          10U 0 Value
D          10U 0 Value
*
DreadcSF  PR  10U 0 ExtProc('QdtsReadcSF')
D          Value
Drec        Like(Record)
Dlen        10U 0 Value
Drrn        10U 0 Value
*
DappendSF  PR  10U 0 ExtProc('QdtsAppendSF')
D          Value
Drec        Like(Record)
Dlen        10U 0 Value
Drrn        10U 0 Value
*
DupdateSF  PR  10U 0 ExtProc('QdtsUpdateSF')
D          Value
Drec        Like(Record)
Dlen        10U 0 Value
Drrn        10U 0 Value
*
DclearSF   PR  10U 0 ExtProc('QdtsClearSF')
D          Value
DinitSF    PR  10U 0 extproc('QdtsInitSF')
D          Value
D          Value
D          10U 0 Value
D          10U 0 Value
*-------------------------------------------------------------------*
* Define the prototypes for the procedures in this program      *
DPGUP      PR  10A
D          10A
D          5S 0
*
DPGDN      PR  10A
D          10A
D          5S 0
*
DINIT      PR  10A
D          10A
D          10A
*
DsetZip    PR          ExtProc('setZip')
D           5S 0
D           20
*
DFillSFL   PR  *-------------------------------------------------------------------*
* Define other program variables                              *
D rc        S  10U 0
D count     S  10U 0
D len S 10U 0 Inz(%Size(Record))
DPAGESIZE S 10U 0 Inz(3)

* Structure to define a subfile record
DRecord DS
D ZIP
D CUST
D NAME

DrSize S 10U 0 Inz(%Size(Record))
DFMT S 10A Inz('FMT01')
DZips S 5S 0
D
Dspc1 S 10A Inz('SFL1')
dspc s 10
dparms s 5

*---------------------------------------------------------------*
* FillSFL                                                      *
* This procedure will fill the subfile DTC by reading all      *
* records in the customer file that have the specified ZIP.    *
* The appendSF procedure is called to add the records to the   *
* subfile.                                                    *
*                                                            *
* Note that the subfile must have been previously initialized *
* and cleared by calling the INIT procedure.                  *
*                                                            *
*---------------------------------------------------------------*

PFillSFL B
C                   Eval      Count = 0
C     Zips          SetLL     CUSREC
C     Zips          ReadE     CUSREC

* Read all records of specified key and add them to the subfile
C                   DoW       NOT %EOF
C                   Eval      rc = appendSF(%Addr(spc): Record: len)
C                   Eval      Count = Count + 1

C     Zips          ReadE     CUSREC
C                   EndDo

PFillSFL E

*---------------------------------------------------------------*
* PGUP                                                      *
* Handle the page up request.                                *
*                                                            *
* This procedure will be called by the subfile DTC when       *
* user pressed the page up key and there are no previous      *
* records in the subfile.                                    *
*                                                            *
* Since the subfile filled with all records, this procedure   *
* just does a return.                                        *
*                                                            *
*---------------------------------------------------------------*

PPGUP B Export
DPGUP PI
D spc 10A
**PPGUP**

* Handle the page down request.

* This procedure will be called by the subfile DTC when
  user pressed the page down key and there are no more
  records in the subfile.

* Since the subfile filled with all records, this procedure
  just does a return.

---

**PPGDN**

* Handle the page down request.

* This procedure will also be called one time by the subfile
  DTC the first time the page with the subfile DTC is loaded.

* Parameters:
  - **spc** - The subfile (space) name (SFL1 in this case)
  - **parms** - Additional parameters as specified on the
    subfile DTC properties page. In this program
    the component name is passed.

---

**PINIT**

* This procedure is called by CUSINQNEW to initialize the
  subfile and fill it with records.

* This procedure will also be called one time by the subfile
  DTC the first time the page with the subfile DTC is loaded.

* Parameters:
  - **spc** - The subfile (space) name (SFL1 in this case)
  - **parms** - Additional parameters as specified on the
    subfile DTC properties page. In this program
    the component name is passed.

---

**Setup**

* If setup = *off
  Eval setup = *on
* Eval spc = spc
* Eval len = %Size(Record)
* LOV
  Eval rc = initSF(%Addr(spc):%Addr(Fmt):rSize)
* Eval rc = clearSF(%Addr(spc))
  CallP FillSFL
Example 11-3 shows the source code of the sample program GETDETAIL. It illustrates how to do a READC on an Affinity subfile DTC. If the user chooses a subfile record by selecting a check box (multiple select), or a radio button (single select) in a subfile record, the readcSF procedure returns the RRN of the selected record as well as the field data. If no record is selected, the procedure returns 0. To create this program, first compile it as a module. Then use the CRTPGM command to bind it to the subfile service program QDTSSFL.

Example 11-3   GETDETAIL

FMLGMSTL2  IF   E       K Disk
*---------------------------------------------------------------
* Prototype the readcSF procedure in the C library
DreadcSF      PR     10U 0 ExtProc('QdtsReadcSF')
D SFName      *       Value
D             *       Value
D              10U 0 Value
D              10U 0 Value
* Record structure for READC
DRecord       DS
D xZIP         5S 0
D xCUST        5S 0
D xNAME        Like(NAME)
D*xCITY       Like(CITY)
D*xADDR       Like(ADDR)
D*xSTATE      Like(STATE)
* Working variables
DSFName      S       10
Drc           S       10 0
DstartRRN     S       10U 0
Dsize         S       10U 0 Inz(%Size(Record))
* Define I/O parameters
C *Entry       Plist
DDS-source code

We include the source for DDS files used in the sample applications (see Example 11-4, Example 11-5, and Example 11-6).

Example 11-4  CUSMSTP

A* CUSTOMER MASTER FILE -- CUSMSTP
A  R CUSREC
A    CUST    5S  0    TEXT('CUSTOMER NUMBER')
A    NAME    20    TEXT('CUSTOMER NAME')
A    ADDR    20    TEXT('CUSTOMER ADDRESS')
A    CITY    20    TEXT('CUSTOMER CITY')
A    STATE    2    TEXT('CUSTOMER STATE')
A    ZIP    5S  0    TEXT('CUSTOMER ZIP CODE')
A    SRHCOD    3    TEXT('CUSTOMER NAME SEARCH CODE')
A    CUSTYP    1    TEXT('CUSTOMER TYPE')
A    ARBAL    10S  2    TEXT('ACCOUNTS RECEIVABLE BALANCE')
A  K CUST
11.3 Program Call beans

To invoke a program or procedure on your iSeries, you need Program Call beans. To create Java Program Call beans, you can use the Program Call wizard. The Program Call wizard is an iSeries extension of Java development tools. It creates Java Beans and Program Call Markup Language (PCML) files to call your iSeries programs or service programs. The Java Beans can also be used to create Web Services using Web Services Development Tools. The wizard prompts you for information regarding program or service program objects, along with the parameters for the objects, and then creates the desired Java Beans. The wizard helps you to define:

- One or more program objects or service program procedures
- The parameters needed by the program objects or procedures
- The structures used by the parameters

Based on your definition, the wizard creates a PCML file and Java Beans to remotely call your iSeries program or procedure. If you already have an existing PCML file that contains all the definitions, you can import the PCML file, and the wizard can generate Java Beans from it.

11.3.1 Creating Program Call beans

To define a Program Call bean, you need to have one or create a new Java or Web project. In our example, we use the Web project that we created in 10.1, “Sample application” on page 180. To launch the Program Call wizard, perform the following steps:

1. Select the project MyProject as shown in Figure 11-49.
2. Right-click the project and then select **New-> Other**, or click **File-> New-> Other** from the main menu.

3. In the left pane of the New window (Figure 11-50), expand **iSeries** and select **Java**. In the right pane, select **Program Call Bean** and click **Next**. The Program Call wizard launches.

4. To define a program or service program procedure that you want to call, click **Add Program**. The Program Call Wizard window (Figure 11-51) opens.
5. Enter the Java bean name.

6. Click **Browse** to select the program object. After you select a program to call, the Library field is filled automatically.

7. Select the Source location. By selecting the type of source file, a new window opens for you to select the source file.
8. Click **OK**. In the left panel, the new program call definition is added as shown in Figure 11-52.

9. To define a parameter required by the program or procedure, select the program in the left panel, click **Add Parameter**. The Add Parameter page is displayed.

   **Note:** If you want to define the information in a PCML file, click **Import PCML** to import part or all of the PCML file into the wizard.

10. Click **Next**. In the next window, you can specify the location where the Java Bean is generated.

11. Click **Finish**. In the Navigator window (Figure 11-53), you can see the Java objects that are created (expand the project and source folders).

![Figure 11-52  Program call definition](image)

- **Figure 11-52  Program call definition**

- **Figure 11-53  Program beans created**

![Figure 11-53  Program beans created](image)
11.3.2 Calling an iSeries program or procedure

The Program Call wizard generates Java Beans and a Program Call Markup Language (PCML) file to call any iSeries program object or service program procedure, such as one written in ILE RPG, ILE C, or OPM COBOL. The wizard generates two types of Java Beans:

- One type for remote calls to the specified iSeries program or procedure as explained in 11.3.1, “Creating Program Call beans” on page 257.
- One type for use by the Web Services wizard to create a Web Service, which can call the specified iSeries program or procedure to perform a task.

The generated Java Beans can pass data types that are supported by the IBM Toolbox for Java classes to iSeries programs or procedures. The data returned from the iSeries program or procedure is converted to the Java data types. If the generated Java Beans are for Web services, they can return data in XML format. The generated Java classes can be used to invoke an iSeries program from any Java code.

IBM Toolbox for Java is a set of Java classes that allow you to use Java programs to access data on your iSeries. You can use these classes to write client/server applications, applets, and servlets that work with data on your iSeries. You can also run Java applications that use the IBM Toolbox for Java classes on the iSeries Java Virtual Machine (JVM). Because the IBM Toolbox for Java uses communication functions built into Java, you do not need to use IBM iSeries Client Access Express to use IBM Toolbox for Java. The IBM Toolbox for Java is a licensed product on the iSeries (licensed program 5722-JC1 V5R1M0).
Extensible Markup Language (XML) is a widely accepted technology. It was designed as an interchange format for sending data between different applications on the Internet, especially for applications on different platforms. XML is a platform-independent language. It is also very useful to exchange data within applications.

This section discusses the XML language and its usage in WebSphere Development Studio Client for iSeries (Development Studio Client).
12.1 Why XML

The growth of the Internet is the result of a number of factors, including the lower cost of personal computers and better communications infrastructure. The biggest driver of Internet growth has been the World Wide Web (WWW), and the information, services, and content that can be accessed by anyone using a browser and a phone line. This growth can be linked to two technical standards that underpin the WWW:

- HyperText Markup Language (HTML)
- HyperText Transfer Protocol (HTTP)

Together these standards form a common language for the Internet. HTTP provides a universal communication protocol, and HTML together with a browser provides a universal user interface. Without these technologies, the Internet would remain a useful but unexciting tool for 20 years. Unlike previous hypertext systems, HTML is text-based, which makes it easy to work with, and an open standard, which means anyone could develop, build, and create HTML and HTML tools.

Many Web sites are now finding that they have evolved beyond the capabilities of HTML. HTML was originally designed to capture mainly textual documents with links to other mainly textual documents. It was also designed to describe the format of a document. The presentation of that format depended on the browser. As the Web expanded into new uses, more presentation information was added to Web pages, and more extensions were added to the HTML specification to support this.

Today’s Web contains many dynamic Web sites that provide “online services”. You can buy almost anything via the Web. The desire to manage large amounts of complex dynamic data, and present it, have created the requirement for a universal data format. This is provided by XML.

12.2 What is XML

XML is a markup language for documents and works well for transmitting data from server to browser. It also passes data between applications and is totally platform independent. Actually, XML is a part of a family of markup languages. XML offers a widely adopted standard way of representing text and data. XML is similar to HTML but has two important differences:

- XML is extensible. Tags can be defined as necessary, where HTML has a fixed set of tags.
- In XML, the form and contents of a document are separated. The tags generally define the structure and content of the data. The appearance is specified by a specific application or a stylesheet. An HTML document consists of tags that define the contents as well as the appearance of the text.

XML is a subset of the Standard Generalized Markup Language (SGML). XML is less formal than SGML and, therefore, easy to implement.

You can find more information about XML as a language on the Web at the XML Zone:

http://www.ibm.com/developerworks/xml/

Or you can refer to the online tutorial about XML on the developerWorks site at:

12.3 XML is important

More and more Web applications are built using XML. Many leading software companies support XML. XML is used for creating Web pages, as well as for data interchange between applications. XML is one of the key technologies for connecting new front-end applications with existing business logic and data. XML is a cross-platform method for describing data that can be integrated with a wide variety of other development methodologies. Both Microsoft’s and Netscape’s Web browsers support XML.

12.4 The benefits of XML

Some benefits of XML are:

- XML is an open standard and is supported by the World Wide Web Consortium (W3C).
- It is easy to read and understand.
- XML is extensible. It has no fixed set of tags. New tags can be created if necessary.
- XML is platform independent. It does not matter which operating system or platform you use. Therefore, XML is an outstanding language for transferring data between applications.
- XML separates the content from the presentation of your Web document. The presentation of an XML document can be controlled by eXtensible Style Language (XSL).

This list of benefits is not a complete summary of all the benefits of XML. But XML has become the universal data format for integrated e-business solutions.

12.5 XML in WebSphere Development Studio Client for iSeries

WebSphere Development Studio Client for iSeries provides a comprehensive visual XML development environment. The toolset includes components for building Document Type Definitions (DTD) (see 12.6.2, “DTD editor” on page 267), XML schemas (see 12.6.3, “XML Schema editor” on page 267), XML, XSL Transformations (XSLT), and mappings between XML and different back-end stores. This chapter gives an overview of the available tools in Development Studio Client.

To use the XML tools you must understand the language fundamentals. In Development Studio Client, the following tools are available:

- XML editor
- DTD editor
- XML Schema editor
- XSL trace editor
- XML-to-XML mapping editor
- XML and SQL query wizard
- Relational database (RDB)-to-XML mapping editor

To start the XML perspective in the WebSphere Development Studio Client for iSeries, perform the following steps:

1. Click the Open Perspective button in the toolbar.
2. Select Other.
3. Select XML and click OK.
A toolbar on top of the perspective is displayed as shown in Figure 12-1. The toolbar contains buttons that open the available editors. The buttons in the taskbar have the same function as the items under the menu options File-> New.

Figure 12-1   Toolbar

12.6 Editors

This section discusses the XML-related editors available in Development Studio Client.

12.6.1 XML editor

The XML editor is a tool for creating and viewing XML files. You can use it to create new XML files and open existing DTDs or existing XML schemas. You can also use it to edit XML files, associate them with DTDs or schemas, and validate them.

You can create XML files in three different ways:

- From scratch
- From an existing DTD
- From an existing XML schema

To create an XML file from scratch, perform the following steps:

1. Create a project to contain the XML file.
2. In the workbench, click File-> New-> Other-> XML-> XML File.
3. Click Next.
4. Select Create XML file from scratch.
5. Click Next.
6. Select the project or folder that will contain the XML file. In the File name field, type the name of the XML file. The extension of your XML file must be .xml.
7. Click Finish.

The XML file appears in the Navigator view and is opened in the XML editor. You can now edit it. If you close the XML editor, and want to later re-open the file you were working on, double-click the file in the Navigator view.

If you create an XML file from a DTD, the XML file will contain the root element of the DTD (that you selected when you created the file) and any elements or attributes contained in the root element. You can add elements, attributes, entities, and notations to the XML file. However, they must follow the rules established in the DTD that you used to create the XML file.

If you create an XML file from a schema, the XML file contains the root element of the schema (that you selected when you created the file) and any elements or attributes contained in the root element. It also contains information about the XML file name space and location. You can now add elements and attributes to the XML file.
12.6.2 DTD editor

A Data Type Definition is a file that contains a set of rules for a particular XML file. To validate an XML file, it must be accompanied by a DTD file.

The DTD editor is a tool for creating and viewing DTDs. You can use it to perform such tasks as:
- Creating new DTDs, either from scratch or from existing XML files
- Validating and editing DTDs
- Creating and deleting DTD elements, attributes, entities, notations, and comments
- Importing existing DTDs for structured viewing
- Generating an XML schema file from a DTD that you can further customize with the XML Schema editor
- Generating Java Beans from a DTD
- Generating an HTML form based on a DTD
- Creating an XML file from a DTD

12.6.3 XML Schema editor

XML Schemas (XSD) are an XML language for describing the constraints of the content of XML files. XSDs are an alternative for the document type definition (DTD). Where DTDs are adopted by the W3C, the XSD is currently in the recommendation phase of the W3C. A schema is functionally equivalent to a DTD, but is written in XML.

For more information about XML schemas, see:

http://www.w3c.org/XML/Schema

The XML Schema editor supports the following functions:
- Creates and deletes XML schema components
- Edits XSD
- Imports existing XSDs for structured viewing
- Generates a DTD from the XML schema
- Generates Java Beans for creating XML instances of an XML schema
- Generates XML files
- Generates relational table files
- Validates XSDs

12.6.4 XSL trace editor

The XSL trace editor enables you to visually step through an XSL transformation script. The XSL trace editor is a kind of playback mechanism that is designed to work together with the XML to XML mapping editor. You can use the XML to XML mapping editor to generate an XSL stylesheet by mapping information between a source and a target XML file.

The XSL trace editor only works on a one-to-one basis. You can only apply an XSL file to one XML file.

12.6.5 XML-to-XML mapping editor

The XML-to-XML mapping editor helps you map one or more source XML files to a single target XML file.
The tool accepts three types of input files:

- XML that includes either a DOCTYPE or an `xsi:schemaLocation` tag that indicates the corresponding DTD or XSD file. If an instance document validates successfully with the XML editor, you can use it as input to the XML to XML mapping editor.
- An XML Schema
- A Document Type Definition

The XML-to-XML mapping editor uses the Xalan processor to do transformations. Xalan is an XSLT processor that transforms XML documents into HTML, text, or other XML document types. The XSLT script that is generated has been tested against the Xalan processor.

For more information on the Xalan processor, refer to the following Apache Web site:

http://xml.apache.org/xalan-j/

12.6.6 XML and SQL query wizard

The XML and SQL query wizard (query builder) enables you to create an XML document as well as DTD files from the results of an SQL query. An XML file can be created with the results from an SQL query.

An optional query template file can be generated if you want to dynamically create XML documents from the SQL query at run time. The template file has the extension `.xst`.

To generate these files, use the XML from SQL wizard. The wizard launches from the Outline view. Right-click a query and select Generate new XML from the pop-up menu.

12.6.7 RDB-to-XML mapping editor

The RDB-to-XML mapping editor is a visual tool used for mapping relational tables or an SQL statement to a DTD. After you complete the mapping, you can generate a Document Access Definition (DAD) file that can be run by the DB2 XML Extender to compose or decompose other XML documents into DB2 data. The DAD file is an XML document that provides control over the mapping of XML documents to a DB2 database for both storage and retrieval.

There are two kinds of XML Collection DAD methods supported by the relational database to XML mapping editor:

- RDB_node
- SQL statement

The mapping is stored in a session file with the extension `.rmx`.

Prerequisite

The RDB-to-XML mapping editor supports the IBM DB2 Universal Database (UDB). To deploy the generated DAD script, you must have IBM DB2 UDB, Version 7.1 with Fix Pack 3, or higher.
Appendixes

This part contains helpful information for using the material (downloaded from the Web) that accompanies this redbook.
Additional material

This IBM Redbook refers to the additional material that can be downloaded from the Internet as described in this appendix.
Locating the Web material

The Web material associated with this Redbook is available on the Internet from the IBM Redbooks Web server. Point your Web browser to:


Alternatively, you can:

1. Open the IBM Redbooks Web site at:

http://ibm.com/redbooks

2. Click Redbooks Online in the frame in the left part of the window. This opens a submenu.

3. Select Additional materials from the submenu.

4. On the Additional Materials page, click Click here for the Redbooks FTP Server. This opens a page with multiple directories.

5. Open the SG246600 directory and download the files from this directory.

Using the Web material

The additional material that accompanies this Redbook includes the following files:

<table>
<thead>
<tr>
<th>File name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APILIB.savf</td>
<td>Order Entry RPG application</td>
</tr>
<tr>
<td>AS2314.savf</td>
<td>Customer Master Inquiry RPG application</td>
</tr>
<tr>
<td>Cusinq.war</td>
<td>Customer Master Inquiry Web application</td>
</tr>
</tbody>
</table>

System requirements for downloading the Web material

The following system configuration is recommended:

- **Hard disk space:** 10 MB minimum
- **Operating System:** WindowsNT or Windows 2000
- **Processor:** Pentium 400 MHz or higher
- **Memory:** 512 MB

How to use the Web material

Create a subdirectory (folder) on your workstation and unzip the contents of the Web material zip file into this folder.

Restoring the APILIB library

You need to FTP the APILIB.savf file to the iSeries server using the following example:

1. Create a save file in your library on iSeries using the Create Save File (CRTSAVF) command.

2. Transfer the downloaded save file to your iSeries library:

   ```
   c:\>ftp yoursystem
   Connected to yoursystem.yourdomain.com.
   220 Connection will close if idle more than 5 minutes.
   User (yoursystem.yourdomain.com:(none))=youruserid
   331 Enter password.
   Password=yourpassword
   230 youruserid logged on.
   ```
ftp>bin
200 Representation type is binary IMAGE.
ftp>put d:\yourdir\APILIB.savf /qsys.lib/yourlib.lib/apilib.savf
200 PORT subcommand request successful.
150-NAMEFMT set to 1.
150 Sending file to member APILIB in file APILIB in library yourlib.
250 File transfer completed successfully.
2758272 bytes transferred in 2.79 seconds. Transfer rate 987.21 KB/sec.
ftp>quit

3. Sign on to the iSeries server and restore the save file:
   RSTLIB SAVLIB(APILIB)DEV(*SAVF)SAVF(yourlib/APILIB)RSTLIB(APILIB)

You can find instructions for using the application in 6.1, “Overview of the Order Entry application” on page 106.

**Restoring the AS2314 library**

Use the same process as in previous example, but change the save file and library name from APILIB to AS2314. You can find instructions on how to use the application in 6.2, “Overview of the Customer Master Inquiry” on page 111.

**Installing the Customer Master Inquiry Web application**

Follow these steps to install the application:

1. From the WebSphere Studio Site Developer Advanced toolbar, click File-> Import. The Import window (Figure 12-2) opens.

![Figure 12-2 Importing a WAR file](image-url)
2. Select **WAR file.**

3. Click **Next.**

4. Browse to the downloaded WAR file (see Figure 12-3).

5. Type **Cusinq** for the Web Project name.

6. Type **Cusinq** for Context Root.

7. Type **CusinqEAR** into the Enterprise Application project name field.

8. Click **Finish.**

To test the Web application, refer to 11.2.3, “Testing application in Development Studio Client” on page 239.
Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 276.

- Moving to Integrated Language Environment for RPG IV, GG24-4358
- AS/400 Programming with VisualAge for RPG, SG24-2222
- Cool Title About the AS/400 and Internet, SG24-4815
- AS/400 Applications: A Fast and Easy Way to Install, Set Up and Work with VRPG and CODE/400 (ADTS CS), SG24-4841
- Unleashing AS/400 Applications on the Internet, SG24-4935
- IBM Web-to-Host Integration Solutions, SG24-5237
- IBM WebSphere Host On-Demand: Version 5 Enhancements, SG24-5989
- IBM WebSphere V4.0 Advanced Edition Security, SG24-6520
- HTTP Server (powered by Apache): An Integrated Solution for IBM @server iSeries Servers, SG24-6716
- IBM WebSphere V4.0 Advanced Edition Handbook, SG24-6176
- WebSphere J2EE Application Development for the IBM @server iSeries Server, SG24-6559
- WebSphere 4.0 Installation and Configuration on the IBM @server iSeries Server, SG24-6815

Other resources

These publications are also relevant as further information sources:

- Programming with VisualAge RPG, SC09-2449
- VisualAge RPG Parts Reference, SC09-2450
- VisualAge RPG Language Reference, SC09-2451
- Coulthard, Phil; Farr, George. Java for RPG Programmers. IBM Press, 2002. 193118206X.
Referenced Web sites

These Web sites are also relevant as further information sources:

- WebSphere Development Studio Client for iSeries home page
  http://www.ibm.com/software/ad/wdt400/
- WebSphere Development Studio Client for iSeries: CODE introduction - online course
  http://www.ibm.com/software/ad/wdt400/course_downloads.html
- iSeries Information Center
- iSeries Navigator home page
- XML zone on developerWorks
  http://www.ibm.com/developerworks/xml
- XML Schema home page
  http://www.w3c.org/XML/Schema
- Xalan-Java home page
  http://xml.apache.org/xalan-j

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.
Index

Numerics
3270 applications to the Web 28
5250 applications to the Web 28
5250 data streams to HTML 179
5250 display device 179
5722-DG1 5
5722-JC1 10
5722-JV1 21
5722-WDS 35
5722-XW1 35
5769-DG1 5
5769-JV1 21

A
ABC Company 106
database 106
Access for Web 27–28
Action mode of compilation 149
adding a record to the subfile 250
ADTS (Application Development ToolSet) 42
Advanced Printer Function (APF) 42
Affinity 47–48, 50
APF (Advanced Printer Function) 42
API (application programming interface) 6
Appearance and Text 139
applet 4, 10
benefits and considerations 11
compared to servlet 15
processing details 10
supporting browsers 10
Application Component Provider 56
application logic in a Web browser 10
application programming interface (API) 6
application server 5, 13
application testing 239
AS/400 Entryfield DTC 217
AS/400 Subfile DTC 217, 223
AS/400 Subfile DTC Properties 224
AS/400 Table DTC 217
AS2314 library 273
authentication 205
authorization 205
available Web settings 139
Display size 139
Key labels 139

B
BLOCK(YES) 173
breakpoint 153
builders 60
business to the Web 17
bytecodes 9

C
C/C++ 42
Call Program parameters 152
CGI
benefits 6
Common Gateway Interface 5
cmpared to Net.Data 8
compared to servlets 12, 14
considerations 6
documentation 7
persistent 6
CGI (Common Gateway Interface) 5
CGU (Character Generator Utility) 42
CHAIN operation code 173
chaining to a specific subfile record 250
Character Generator Utility (CGU) 42
classic tools 128
clearing the subfile 250
Client Access 261
Client Access Family for iSeries 35
client machine requirements
  Host On-Demand 31
  Host Publisher 33
  Web Interaction wizard 35
  WebFacing Tool 32
CLUP 250
COBOL 42
CODE 41, 48, 127
  Host Access Programs 42
code 128
CODE (CoOperative Development Environment) 52
CODE Communication Console 128
CODE Designer 52, 138–139
  Web settings 138
CODE Designer Web Settings with your DDS source 138
CODE Editor 128, 134
CODE Program Generator 148
CODE Project Organizer 52
code server 151
CODE/400 47
Common Gateway Interface (CGI) 5
Communication Daemon 128
compact installation 175
Compare editor 61
Compile Commands window 146
compiling a program 144, 146
connection 115, 117, 128, 153
Console view 246
Convert RPG Source 137
CoOperative Development Environment (CODE) 52
Create Graphic 140
Create hyperlink 140
Create subroutine 170
Creating object filter 131
CUM Pack for WebSphere Application Server 21
cumulative PTF package (CUM Pack) for WebSphere Application Server 21
Cusinq.jsp 227
CUSINGNEW 251
Cussfl.jsp 227–228
Customer Master Inquiry 111
Customer Master Inquiry Web application 273
CVTRPGSRC 137

D
DAD (Document Access Definition) file 268
Data Description Specifications (DDS) 29, 33, 187
Data File Utility (DFU) 42
Data File Utility bean 51
Database On-Demand 30
database table structure 107
DB2 data 34
DB2 Universal Database 268
DBGVIEW(*SOURCE) 246
DDS (Data Description Specifications) 29, 33, 187
DDS source 138
DDS source code 33
DDS Tree 139
Debug view 242
debugger 150
debugging
  a Web application with host program call 245
  ILE programs with Distributed Debugger 150
  with an ILE program 245
Designer Canvas 164
design-time control (DTC) 50, 207
destroy() method 12
Developer Kit for Java (JDK) 21
Development environment for QDTSSFL 250
Development Studio Client 47
  application testing 239
  extensions (plug-ins) 113
  features 60
  testing your application 187
  Web development tools 207
DFU (Data File Utility) 42
direct access to DB2 data 34
display device 6
display file (DSPF) 33, 179
display file conversion 186
Display PTF (DSPPTF) command 20
displaycustdetailResults.jsp 228
Distributed Debugger 150
DOCTYPE tag 268
Document Access Definition (DAD) file 268
document root directory 90
Document Type Definition (DTD) 265
doDelete servlet method 12
doGet servlet method 12–13
doPost servlet method 12–13
doPut servlet method 12
DSPF (display file) 179
DSPPTF (Display PTF) command 20
DTC (design-time control) 50, 207
DTD (Document Type Definition) 265
DTD editor 265, 267
dynamic Web site content 5

E
e-business 4
ECLApplets 30
Eclipse consortium 45, 56
Eclipse platform 43–44
Eclipse project 40, 56
Eclipse Software Development Kit (SDK) 45
eclipse.org 45
editing source code 134
editors 59
EJB (Enterprise JavaBeans) 17
EJB container 18
EJB server 18
Enterprise Application 198–199
Enterprise Application project name 274
Enterprise JavaBeans (EJB) 17
Enterprise Toolkit for iSeries (ET/400) 47–48, 51
entity bean 18
environment management with Remote Systems Explorer 130
error page 227
ET/400 47–48
ET/400 (Enterprise Toolkit for iSeries) 51
event-driven language 159
EXEC 5
Export Resources to a WAR File 196
Export wizard 142
exporting a Web application to production 247
Extensible Markup Language (XML) 263
Extensible Style Language (XSL) 265
extensions 113

F
FCMU (File Compare and Merge Utility) 42
File Compare and Merge Utility (FCMU) 42
File Export/Import wizards 51
file share 203
file system 120
FillPage subroutine 170
filter 118, 120
filter pool 123, 131

G
GET method 12
getDetail Web interaction 228
getList Web interaction 227
group PTF 20
  for WebSphere Application Server 21
GUI Designer 161

H
Help perspective 62
Host Access Beans for Java 30
Host Access Class Library for Java 30
Host Access Controls for ActiveX 30
<table>
<thead>
<tr>
<th>Access for Web</th>
<th>27–28</th>
</tr>
</thead>
<tbody>
<tr>
<td>comparison table</td>
<td>35</td>
</tr>
<tr>
<td>Host On-Demand (HOD)</td>
<td>27, 29</td>
</tr>
<tr>
<td>Host Publisher</td>
<td>28, 32</td>
</tr>
<tr>
<td>positioning</td>
<td>25</td>
</tr>
<tr>
<td>Screen Customizer</td>
<td>28</td>
</tr>
<tr>
<td>Transcoding Publisher</td>
<td>28</td>
</tr>
<tr>
<td>WebFacing Tool</td>
<td>29, 31, 49</td>
</tr>
<tr>
<td>WebSphere Development Studio</td>
<td>40</td>
</tr>
<tr>
<td>WebSphere Studio</td>
<td>35</td>
</tr>
<tr>
<td>Host Information Setup wizard</td>
<td>50</td>
</tr>
<tr>
<td>host applications to the Web</td>
<td>27</td>
</tr>
<tr>
<td>host integration</td>
<td>27</td>
</tr>
<tr>
<td>Host Integration family</td>
<td>27</td>
</tr>
<tr>
<td>Host Integration wizard</td>
<td>25</td>
</tr>
<tr>
<td>Host Integration Object</td>
<td>34</td>
</tr>
<tr>
<td>Host Integration Solution</td>
<td>35</td>
</tr>
<tr>
<td>host machine requirements</td>
<td>30</td>
</tr>
<tr>
<td>Host On-Demand</td>
<td>30</td>
</tr>
<tr>
<td>Host Publisher</td>
<td>32</td>
</tr>
<tr>
<td>Web Interaction wizard</td>
<td>35</td>
</tr>
<tr>
<td>Host On-Demand (HOD)</td>
<td>27, 29</td>
</tr>
<tr>
<td>host program call, debugging a Web application</td>
<td>245</td>
</tr>
<tr>
<td>Host program run-time library list</td>
<td>50</td>
</tr>
<tr>
<td>Host Publisher</td>
<td>25–26, 28, 32</td>
</tr>
<tr>
<td>comparing to WebFacing Tool</td>
<td>33</td>
</tr>
<tr>
<td>Host Publisher Server</td>
<td>32</td>
</tr>
<tr>
<td>Host Publisher Studio</td>
<td>32</td>
</tr>
<tr>
<td>host/server access products</td>
<td>24</td>
</tr>
<tr>
<td>HTML (HyperText Markup Language)</td>
<td>4, 264</td>
</tr>
<tr>
<td>HTTP (HyperText Transfer Protocol)</td>
<td>264</td>
</tr>
<tr>
<td>HTTP server</td>
<td>4</td>
</tr>
<tr>
<td>HTTP Server (powered by Apache)</td>
<td>5</td>
</tr>
<tr>
<td>HttpServletRequest</td>
<td>13</td>
</tr>
<tr>
<td>HttpServletResponse</td>
<td>14</td>
</tr>
<tr>
<td>HyperText Markup Language (HTML)</td>
<td>4, 264</td>
</tr>
<tr>
<td>HyperText Transfer Protocol (HTTP)</td>
<td>264</td>
</tr>
<tr>
<td>iSeries IFS file share</td>
<td>203</td>
</tr>
<tr>
<td>iSeries Files Subsystem</td>
<td>119, 130</td>
</tr>
<tr>
<td>iSeries Files subsystem</td>
<td>119</td>
</tr>
<tr>
<td>iSeries Jobs Subsystem</td>
<td>119</td>
</tr>
<tr>
<td>iSeries User ID</td>
<td>241</td>
</tr>
<tr>
<td>iSeries Web development tools</td>
<td>207</td>
</tr>
<tr>
<td>iSeries Commands for iSeries extensions</td>
<td>48, 113</td>
</tr>
<tr>
<td>iSeries Commands Subsystem</td>
<td>119</td>
</tr>
<tr>
<td>iSeries connections</td>
<td>128</td>
</tr>
<tr>
<td>iSeries Debug Server</td>
<td>245</td>
</tr>
<tr>
<td>iSeries IFS file share</td>
<td>203</td>
</tr>
<tr>
<td>iSeries IFS Files Subsystem</td>
<td>119</td>
</tr>
<tr>
<td>iSeries JOBS Subsystem</td>
<td>119</td>
</tr>
<tr>
<td>iSeries User ID</td>
<td>241</td>
</tr>
</tbody>
</table>

J

J2EE | 19, 43 |
Java | 5, 22 |
Java API | 10 |
Java applet | 10 |
Java Beans | 15 |
Java builder | 60 |
Java code on the Web server | 12 |
Java Development Toolkit (JDK) | 19 |
Java Perspective | 51 |
Java platform | 9 |
Java-based technology | 9 |
Java servlet processing on the iSeries server | 13 |
Java Tools - iSeries Extensions | 48, 51 |
Java Tools for iSeries Examples | 51 |
Java Virtual Machine (JVM) | 9, 261 |
javax.servlet.http | 12 |
javax.servlet | 12 |
JDK (Developer Kit for Java) | 21 |
JDK (Java Development Toolkit) 19
JFace 45
JSP 196
JSP (JavaServer Page) 16, 43, 179, 187
JSP component 16
JVM (Java Virtual Machine) 9

K
Key order 139

L
legacy applications 18
Library field 146
library list 50, 138, 144
Linux on iSeries 22
List position 146
local history 60
Location 139
Lotus Domino 22

M
Macro Recording 34
MAP 5
Microsoft 265
Model-View-Controller (MVC) 16
MVC (Model-View-Controller) 16

N
Net.Data 5, 7
  benefits 8
  considerations 8
  documentation 9
  processing details 8
Netscape 265
non-Java-based applications, database serving standards 5

O
Object List bean 51
Open Host Interface Objects 30
Operations Navigator 203
Options for VALUES 139, 142
Order Entry application 106
OS/400 PASE 22
output Web page 227

P
Page Designer view 50
PageEnd event 170
Parts Order Entry 108
Parts Palette 163
PASS 5
password 195, 241
PCML (Program Call Markup Language) 257
PDE (Plug-in Development Environment) 45
PDM (Programming Development Manager) 42
performance 205
PERL 5, 14
persistance 15
persistent CGI 6, 12
perspective 57, 114, 180
PGDN 250
PGUP 250
platform roles 56
plug-in 45
Plug-in Development Environment (PDE) 45
plug-ins 113
portability 15
POST method 12
pre-supplied Java Beans 51
privacy 205
procedural language 159
profile 116
program alias 231
Program Call beans 207, 257
program call definitions 232, 234
Program Call Markup Language (PCML) 257
Program Call wizard 51
program compilation 146
program object 232
Program-defined HTML 139
Programming Development Manager (PDM) 42
Project View 162
prompting on RPG 137

Q
QdtsAppendSF 250, 252
QdtsAppendSF API 243
QdtsChainSF 250, 252
QdtsClearSF 250, 252
QdtsInitSF 250, 252
QdtsReadcSF 252
QdtsReadSF 250
QDTSSFL program 250
QdtsUpdateSF 250, 252
QQFENV 205
QrtEnvironment 205
QSYSWRK subsystem 150
quick help 63
QzhbCgiParse API 5, 7

R
Rational ClearCase 60
RDB-to-XML mapping editor 265, 268
READC operation code 173
reading a selected record 250
READS operation code 173
Redbooks Web site 276
  Contact us  xi
Remote Java Compile and Run 51
Remote Systems Explorer 49, 114, 128
  connection 117
  filter 120
  filter pool 123
  managing your environment 130
perspective 114
profile 116
subsystems 118
Remote Systems Explorer (RSE) 113
Report Layout Utility (RLU) 42
restoring the APILIB library 272
restoring the AS2314 library 273
RLU (Report Layout Utility) 42
root file system 120
RPG 41
RPG IV 160
RSE (Remote Systems Explorer) 113
run-time library list 50, 241

S
Screen Customizer 28
Screen Design Aid (SDA) 42
SDA (Screen Design Aid) 42
security 205
server configuration 242
server instance 242
Server Perspective 242
Server Service program 224
service() method 12
servlet 4, 12
  benefits 14
  doDelete method 12
  doGet method 12–13
  doPost method 12–13
  doGet method 12
  init() method 12
  service() method 12
  supported by WebSphere Studio Site Developer 46
servlet life cycle 12
session bean 18
SETATR 173
SEU (Source Entry Utility) 42
SEU-type line commands 137
SGML (Standard Generalized Markup Language) 264
SHOWWIN operation code 173
source code 134
Source Entry Utility (SEU) 42
SQL 6
Standard Generalized Markup Language (SGML) 264
Standard Widget Toolkit (SWT) 45
Start Distributed Debugger 151
STRCODE 151
STRDBGSVR 150, 245
stream file 6
STRTCPSVR 204
structure 232
style
  Web 186
  WebFacing 29
Subfile DTC APIs 249
Submit button 220
subsystem 118
Swing JFormatted bean 51
SWT (Standard Widget Toolkit) 45

T
Table DTC 50
Table relationship 107
Table400DTC 50
TCP/IP 158
team environment 60
testing application without deploying 239
token highlighting 135
Toolbox for Java 51
Transcoding Publisher 28

U
UIM (User Interface Manager) 33, 49
UIM source member 184
updating a subfile record 250
URL 198
user ID 195
User Interface Manager (UIM) 33, 49
User-defined HTML 139

V
VALUES 142
variables 153
VARPG (VisualAge RPG) 158
VARPG GUI Designer 161
VARPG programming 159
versions of WebSphere Application Server 20
views 59
Visibility 139–140
Visual Composition Editor 15
Visual Development Environment 158
VisualAge for Java 15, 43, 47–48, 51
VisualAge for Java for iSeries 41
VisualAge RPG 41, 47–48, 52
  Host Access Programs 42
VisualAge RPG (VARPG) 158
VisualAge RPG Language Reference 158

W
W3C (World Wide Web Consortium) 265
WAR (Web Application Archive) file 196
WAR file 196–198
Web application 18, 200
  combining new and existing 25
  development, deployment 18
Web Application Archive (WAR) file 196
Web application to production 247
Web deployment descriptor file 196
Web development tools 207
Web Interaction wizard 26, 207–208, 227
Web server 4
  running Java code 12
Web services 46
Web settings 139
Web Settings tab 138
Web style 186
Web Tools - iSeries Extensions 48–50
web.xml 196
Learn a new way to build applications for the IBM @server iSeries Server

Bring your 5250 applications to the Web with the IBM WebFacing Tool

Catch up with the latest trends in application development

Whether you want to write your own Web applications or move existing applications to the Web, IBM lets you do this with a number of host access products. The newest addition is the IBM WebFacing Tool — part of the new WebSphere Development Studio Client for iSeries (Development Studio Client). The WebFacing Tool converts existing Data Description Specifications (DDS) into Java artifacts that are deployed in an application server. With the WebFacing Tool, existing iSeries applications require minor or no modifications and are accessible from the Web and iSeries display devices.

This IBM Redbook discusses WebSphere Development Studio Client for iSeries, the WebFacing Tool, and several more powerful tools, which include:

- **WebSphere Studio Site Developer Advanced**: The new integrated development environment (IDE) for developing applications. It offers the unique ability to add new features in the form of plug-ins.
- **Cooperative Development Environment (CODE)**: Workstation-based tool that supports the development of applications in many different host languages, including RPG and Java.
- **VisualAge RPG**: Allows you to create and maintain client/server applications on the workstation.
- **IBM Distributed Debugger**: Lets you detect and diagnose errors in code developed with OS/400 Integrated Language Environment (iLE) and Original Program Model (OPM) languages and Java.

For more information: ibm.com/redbooks