WebSphere Development Studio Client for iSeries Version 5.1.2

Learn a new way to build applications for the IBM eServer iSeries server

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

Find out about the latest trends in application development

Wilfried Blankertz
Maggie Biggs
Mohamed Esmat
Estela McCarty
Jerry Sigwing
Claus Weiss

ibm.com/redbooks
Note: Before using this information and the product it supports, read the information in “Notices” on page xi.

Second Edition (May 2005)

This edition applies to Version 5.1.2 of WebSphere Development Studio Client for iSeries Standard and Advanced Edition.

© Copyright International Business Machines Corporation 2003, 2005. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
# Contents

## Notices ................................................................. xi
## Trademarks .......................................................... xii

## Preface ................................................................. xiii
The team that wrote this redbook .................................. xiv
The team who wrote the first edition of this redbook .......... xv
Become a published author .......................................... xvi
Comments welcome .................................................... xvi

## Preface ................................................................. xiii

## Part 1. WebSphere Development Studio Client ........................................... 1

### Chapter 1. The iSeries Developer Roadmap ........................................... 3
1.1 The decision to modernize ................................................. 4
1.2 iSeries Developer Roadmap ............................................. 5
1.3 WebSphere Development Studio for iSeries and WebSphere Development Studio Client for iSeries .................................................. 12
1.4 Focus of this IBM Redbook .............................................. 13

### Chapter 2. Creating applications for iSeries: A new set of tools .......................... 15
2.1 IBM WebSphere Development Studio for iSeries ......................... 16
   2.1.1 ILE RPG ................................................. 17
   2.1.2 ILE C/C++ ............................................. 18
   2.1.3 ILE COBOL ............................................. 18
   2.1.4 Application Development ToolSet .................................. 18
2.2 WebSphere Development Studio Client for iSeries ....................... 19
   2.2.1 WebSphere Studio family ..................................... 20
   2.2.2 IBM WebFacing Tool ....................................... 25
   2.2.3 Web tools: iSeries extensions .................................. 25
   2.2.4 Java development tools ....................................... 26
   2.2.5 CoOperative Development Environment .......................... 27
   2.2.6 VisualAge RPG ........................................... 27

### Chapter 3. The Eclipse framework ........................................... 29
3.1 Eclipse: The start of a new era of tools ................................... 30
3.2 A new approach to the application development process .................... 30
3.3 Components of the Eclipse GUI framework ................................ 31
   3.3.1 Workbench ................................................ 31
   3.3.2 Projects .................................................. 32
   3.3.3 Perspectives ............................................... 32
   3.3.4 Views .................................................... 34
   3.3.5 Editors .................................................... 37
   3.3.6 Crash recovery ............................................ 40
3.4 Development Studio Client features .................................... 41
   3.4.1 Tools ..................................................... 41
   3.4.2 Common features .......................................... 42
   3.4.3 iSeries-specific features ................................... 48
3.5 Online help .......................................................... 48
3.6 Obtaining and installing fixes for WDSoc ................................... 51
   3.6.1 Which version or fixpack of WDSoc do you have installed? ...... 51
Chapter 4. Introduction to the sample applications

4.1 Overview of the Order Entry application
   4.1.1 The ABC Company ........................................... 62
   4.1.2 The ABC Company database ................................. 62
   4.1.3 Application files ........................................... 62
   4.1.4 Database table structure ................................. 63
   4.1.5 Customer transaction flow ................................. 63

4.2 Overview of the Customer Master Inquiry
   4.2.1 Starting the application ................................... 68
   4.2.2 Overview of the Customer Master Web application .... 70

Part 2. The Remote System Explorer and iSeries Projects

Chapter 5. Managing your iSeries development resources using RSE

5.1 Remote System Explorer ........................................ 75
5.2 Connecting to your iSeries host
   5.2.1 The first connection ........................................ 77
   5.2.2 Profiles ....................................................... 78
   5.2.3 Connections .................................................. 79
   5.2.4 TCP/IP ports required for WDSc .......................... 82
5.3 Subsystems
   5.3.1 iSeries Objects subsystem ................................. 84
   5.3.2 iSeries Commands subsystem .............................. 84
   5.3.3 iSeries Jobs subsystem .................................... 88
   5.3.4 IFS (Integrated File System) files subsystem ........ 92
5.4 Managing your iSeries objects
   5.4.1 Filters in the iSeries Objects subsystem .............. 97
   5.4.2 Working with iSeries Objects filters .................. 103
   5.4.3 Creating objects in the iSeries Objects subsystem ... 104
   5.4.4 Searching operations ..................................... 110
   5.4.5 Working with save files ................................. 113
5.5 Working with your library list
   5.5.1 Adding a library list entry ............................... 118
   5.5.2 Removing a library from the library list .............. 119
   5.5.3 Changing the current library ............................ 119
   5.5.4 Moving libraries up or down in the library list .... 120
   5.5.5 Moving libraries within the library list ............... 120
   5.5.6 Specifying initial libraries and commands ........... 120
5.6 iSeries table views
   5.6.1 Populating iSeries table views ......................... 121
   5.6.2 Working with items in a table view .................... 124
5.7 User actions
   5.7.1 Creating user actions .................................... 127
   5.7.2 Invoking user actions .................................... 129
   5.7.3 Named types ............................................... 129
5.8 Additional parameters with actions issued from a table view 131
5.9 Drag and drop functionality .................................. 137

Chapter 6. LPEX, the modern SEU plus much more

6.1 Editing source members in the RSE .......................... 139
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.2.1</td>
<td>Creating a dynamic Web project</td>
<td>441</td>
</tr>
<tr>
<td>13.2.2</td>
<td>Methods for designing Web application pages</td>
<td>452</td>
</tr>
<tr>
<td>13.3</td>
<td>Designing Web application pages</td>
<td>453</td>
</tr>
<tr>
<td>13.3.1</td>
<td>Creating a page template</td>
<td>454</td>
</tr>
<tr>
<td>13.4</td>
<td>Tying JSPs to RPG with the Web Interaction wizard</td>
<td>483</td>
</tr>
<tr>
<td>13.4.1</td>
<td>The iSeries Web Tools Run-time Configuration wizard</td>
<td>484</td>
</tr>
<tr>
<td>13.4.2</td>
<td>Creating the getList Web interaction</td>
<td>485</td>
</tr>
<tr>
<td>13.4.3</td>
<td>Creating the getDetail Web interaction</td>
<td>495</td>
</tr>
<tr>
<td>13.4.4</td>
<td>More complex interactions</td>
<td>501</td>
</tr>
<tr>
<td>13.4.5</td>
<td>Testing the application in Development Studio Client</td>
<td>502</td>
</tr>
<tr>
<td>13.5</td>
<td>Struts tools</td>
<td>509</td>
</tr>
<tr>
<td>13.5.1</td>
<td>View, controller, and Struts</td>
<td>510</td>
</tr>
<tr>
<td>13.5.2</td>
<td>WebSphere Studio and Struts</td>
<td>511</td>
</tr>
<tr>
<td>13.5.3</td>
<td>Struts-based Web applications</td>
<td>511</td>
</tr>
<tr>
<td>13.5.4</td>
<td>Struts actions</td>
<td>511</td>
</tr>
<tr>
<td>13.5.5</td>
<td>Struts cheat sheet</td>
<td>511</td>
</tr>
<tr>
<td>13.5.6</td>
<td>Struts configuration file</td>
<td>511</td>
</tr>
<tr>
<td>13.5.7</td>
<td>Struts form beans</td>
<td>512</td>
</tr>
<tr>
<td>13.5.8</td>
<td>Struts forwards</td>
<td>512</td>
</tr>
<tr>
<td>13.6</td>
<td>Using a Web diagram to generate a Web interaction</td>
<td>512</td>
</tr>
<tr>
<td>13.6.1</td>
<td>Web diagram</td>
<td>513</td>
</tr>
<tr>
<td>13.6.2</td>
<td>Representing an existing Web interaction in the Web diagram</td>
<td>514</td>
</tr>
<tr>
<td>13.7</td>
<td>Exporting the Web application to production</td>
<td>525</td>
</tr>
<tr>
<td>13.8</td>
<td>Subfile APIs</td>
<td>526</td>
</tr>
<tr>
<td>13.8.1</td>
<td>Example code</td>
<td>532</td>
</tr>
<tr>
<td>13.9</td>
<td>Program Call beans</td>
<td>538</td>
</tr>
<tr>
<td>13.9.1</td>
<td>Creating Program Call beans</td>
<td>538</td>
</tr>
<tr>
<td>13.9.2</td>
<td>Calling an iSeries program or procedure</td>
<td>541</td>
</tr>
</tbody>
</table>

Part 4. Testing and debugging

<table>
<thead>
<tr>
<th>Chapter 14.</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.1</td>
<td>Available debuggers</td>
<td>545</td>
</tr>
<tr>
<td>14.2</td>
<td>Prerequisites for debugging iSeries applications</td>
<td>546</td>
</tr>
<tr>
<td>14.2.1</td>
<td>Compile options</td>
<td>547</td>
</tr>
<tr>
<td>14.2.2</td>
<td>Setting debug preferences</td>
<td>547</td>
</tr>
<tr>
<td>14.2.3</td>
<td>Setting breakpoints prior to debugging</td>
<td>549</td>
</tr>
<tr>
<td>14.3</td>
<td>Integrated iSeries debugger</td>
<td>551</td>
</tr>
<tr>
<td>14.3.1</td>
<td>One-step debugging</td>
<td>551</td>
</tr>
<tr>
<td>14.3.2</td>
<td>Debugging with full prompt</td>
<td>556</td>
</tr>
<tr>
<td>14.3.3</td>
<td>How to debug</td>
<td>565</td>
</tr>
<tr>
<td>14.3.4</td>
<td>Testing and debugging an iSeries Web application</td>
<td>578</td>
</tr>
<tr>
<td>14.3.5</td>
<td>Debugging a Web application</td>
<td>588</td>
</tr>
<tr>
<td>14.4</td>
<td>IBM Distributed Debugger</td>
<td>592</td>
</tr>
<tr>
<td>14.4.1</td>
<td>Setting up a debug session</td>
<td>592</td>
</tr>
<tr>
<td>14.5</td>
<td>Profiling Java and J2EE applications</td>
<td>595</td>
</tr>
<tr>
<td>14.5.1</td>
<td>Introduction to profiling</td>
<td>597</td>
</tr>
</tbody>
</table>

Part 5. Working with VisualAge RPG

<table>
<thead>
<tr>
<th>Chapter 15.</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.1</td>
<td>VisualAge RPG features</td>
<td>611</td>
</tr>
<tr>
<td>15.2</td>
<td>VisualAge RPG language reference</td>
<td>612</td>
</tr>
<tr>
<td>15.2.1</td>
<td>Help topics for VisuaAge RPG in the Development Studio Client Help</td>
<td>613</td>
</tr>
</tbody>
</table>
15.2.2 Help topics in the GUI designer .................................................. 614
15.3 Programming with VARPG .......................................................... 615
  15.3.1 Starting the VARPG Designer ..................................................... 617
  15.3.2 The VARPG GUI Designer ....................................................... 617
15.4 Creating a simple GUI subfile application ..................................... 620
  15.4.1 Creating the user interface ....................................................... 620
  15.4.2 Saving the project ................................................................... 625
  15.4.3 Writing the logic to fill the subfile ............................................. 626
  15.4.4 Building and running the application ....................................... 627
  15.4.5 Enhancing the application ....................................................... 628
  15.4.6 Selecting a record from the subfile ......................................... 629
15.5 Building a thin client application with VARPG ............................ 633
  15.5.1 Changing the default setting in the GUI designer ..................... 633
  15.5.2 Building the first window ....................................................... 634
  15.5.3 Creating the second window .................................................... 639
  15.5.4 Creating the third window ....................................................... 642
  15.5.5 Adding code to run the application ......................................... 644
  15.5.6 Taking a look at the thin client application built so far .............. 651
  15.5.7 Further enhancing the thin client example ............................... 655
  15.5.8 The complete thin client application example ......................... 663
  15.5.9 The backend programs running under OS/400 or i5/OS ............. 671
15.6 RPGIV operation codes not available in VARPG ......................... 677
15.7 Additional information .............................................................. 677

Appendix A. Additional material ......................................................... 679
  Locating the Web material ............................................................. 679
  Using the Web material ................................................................. 679
  Installation instructions for the VARPG samples ........................... 680
    System requirements for downloading the Web material ................ 681
    How to use the Web material ....................................................... 681

Related publications .......................................................................... 685
  IBM Redbooks ............................................................................... 685
  Other publications .......................................................................... 686
  Online resources ........................................................................... 686
  How to get IBM Redbooks ............................................................. 688
  Help from IBM ............................................................................. 688

Index ................................................................................................ 689
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:

<table>
<thead>
<tr>
<th>™</th>
<th>™</th>
<th>™</th>
</tr>
</thead>
<tbody>
<tr>
<td>iSeries®</td>
<td>COBOL/400®</td>
<td>OS/2®</td>
</tr>
<tr>
<td>i5/OS™</td>
<td>Domino®</td>
<td>OS/390®</td>
</tr>
<tr>
<td>pSeries®</td>
<td>DB2 Universal Database™</td>
<td>OS/400®</td>
</tr>
<tr>
<td>xSeries®</td>
<td>DB2®</td>
<td>PowerPC®</td>
</tr>
<tr>
<td>z/OS®</td>
<td>DRDA®</td>
<td>Rational®</td>
</tr>
<tr>
<td>zSeries®</td>
<td>Integrated Language Environment®</td>
<td>Redbooks™</td>
</tr>
<tr>
<td>AFP™</td>
<td>IBM®</td>
<td>Redbooks(logo)™</td>
</tr>
<tr>
<td>AIX®</td>
<td>IMS™</td>
<td>RPG/400®</td>
</tr>
<tr>
<td>AS/400®</td>
<td>Language Environment®</td>
<td>SecureWay®</td>
</tr>
<tr>
<td>ClearCase®</td>
<td>Lotus®</td>
<td>System/38™</td>
</tr>
<tr>
<td>CICS®</td>
<td>MVS™</td>
<td>VisualAge®</td>
</tr>
<tr>
<td></td>
<td>Net.Data®</td>
<td>WebSphere®</td>
</tr>
<tr>
<td></td>
<td>NetServer™</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Notes®</td>
<td></td>
</tr>
</tbody>
</table>

The following terms are trademarks of other companies:

Intel, Intel Inside (logos), and Pentium 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.

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

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

With the advent of the Internet and growth of the number of Internet users, a new type of application emerged – the Web application. The beauty of the Web application is that it runs on a server and the client is simply a Web browser. This means that you can access such applications from anywhere in the world where there is access to the Internet.

Does this mean that you have to create an array of the new applications? What about the existing ones? These questions are right on target. The weak support for Web-enabling the existing applications is one of the biggest obstacles on the path to the wide acceptance of the Web application model.

IBM® iSeries™ customers know this situation. For many years, companies have created applications for the iSeries server and its predecessors that provide adequate support for their business processes. There is no reason to invest in new applications. Now, with the idea of the universal accessibility and automation of business processes (including business-to-business (B2B) capabilities), the concept of creating new applications is very appealing.

To take advantage of both worlds, IBM has created several host access products. One member of this family of products is the IBM WebFacing Tool. The IBM WebFacing Tool is packaged as part of WebSphere® Development Studio Client for iSeries (in short Development Studio Client or WDSc). It converts existing Data Description Specifications (DDS) into Java™ artifacts that are deployed in an application server, such as, WebSphere Application Server.

While the IBM WebFacing Tool is probably the most commonly known component of WDSc, there are many other tools combined into a single product. All those components together provide a complete workbench based on the Eclipse framework for the application developer for OS/400® or i5/OS™. This IBM Redbook discusses WebSphere Development Studio Client for iSeries V5.1.2 (WDSc). WDSc includes several powerful tools that are targeted to iSeries application developers:

- **WebSphere Studio Site Developer** (in WebSphere Development Studio Client Standard Edition for iSeries) or **WebSphere Studio Application Developer** (in WebSphere Development Studio Client Advanced Edition for iSeries): This is the new integrated development environment (IDE) for developing applications. The ability to add new features in the form of plug-ins makes this IDE unique. Anyone can develop a new plug-in and install it into the tool without creating a “plumbing” infrastructure. WebSphere Studio Site Developer and WebSphere Studio Application Developer include the development environment to create plug-ins.

- **iSeries-specific features**: These include such tools as the IBM WebFacing Tool and the iSeries Web development tools, which are installed as plug-ins.

- **CoOperative Development Environment (CODE)**: This is a workstation-based tool that supports the development of the applications in many different host languages, including RPG and Java.

- **VisualAge RPG**: If you are already an experienced RPG IV programmer, you can create graphical user interfaces to RPG programs very quickly in VisualAge® RPG.

- **Integrated iSeries debugger**: This tool 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. The debugger’s client/server design makes it
possible to debug applications running on an iSeries machine while the debugger user interface runs in the workbench IDE on a Windows® workstation.

**New content in the second edition of this book**

While the first edition of this redbook has been written based on WDSc 5.0, the second edition is an update in respect to the changes and new functions in WDSc 5.1.2. All significant changes or additions in the book are marked by a vertical bar at the left side of the page, except for Chapter 1, “The iSeries Developer Roadmap” on page 3 and Chapter 9, “Web Development Tools in WDSc” on page 241 which are both completely new.

**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.

**Wilfried Blankertz** is a Senior IT Specialist for iSeries Technical Sales in the IBM EMEA Central region located in Frankfurt, Germany. From 1995 to 1998, he was assigned to the ITSO, Rochester Center, where he co-authored more than 15 IBM Redbooks™ and taught IBM classes worldwide in all areas of OS/400® Groupware solutions and Systems Management. He also co-authored the first edition of this redbook. Before joining the ITSO, he worked as a Systems Engineer in IBM Germany, supporting customers with the AS/400® system and its predecessor systems for 30 years. He is also a Certified Lotus® Professional for Administration and for Domino® R5 Application Development. He owns several certifications for WebSphere application development for iSeries. You can reach Wilfried at WilBlank@de.ibm.com.

**Maggie Biggs** is a Senior Engineer with a large financial services firm and she currently focuses on WebSphere-related technologies. She has worked with the iSeries and all of its predecessors back to the System/34. In addition, she possesses strong multi-platform and multi-technology expertise and has worked in a variety of roles, including architecture, development, server administration, and management. She has also co-authored a redbook on Java and WebSphere performance management on the iSeries. You can reach Maggie at maggiebiggs@acm.org.

**Mohamed Esmat** is a Senior Software Engineer working within the Cairo Technology Development Center (CTDC) at IBM Egypt for the past three years, with more than six years of experience with the AS/400 platform, with predominantly developing and testing. He graduated from Ain Shams University in Cairo, Egypt 1998, and is currently pursuing a master's degree in the Low Power design field. He holds a Certified Oracle development and administration certification. You can reach Mohamed at mesmat@eg.ibm.com.

**Estela McCarty** is a Computer Consultant for the EMEA Custom Technology Centre located in La Gaude, France. She has been involved with Web-enabling solutions on the iSeries for the past four years, conducting technical workshops for customers and IBM Business Partners all over EMEA. She has been an iSeries Application Developer for nine years with IBM New Zealand. She is also a Certified IBM Solution Developer for WDSc for iSeries V5.0. You can reach Estela at EstelaMcCarty@fr.ibm.com.
Jerry Sigwing is a Staff Software Engineer with the Rochester iSeries Support Center. He was a Team Leader for the Languages & Utilities (LU) queue. He started with IBM in 1970 as a Customer Engineer in Nebraska and then went to the Boulder IBM Support Center in 1986 where he supported MVS™, VM, and VSE. After serving in a Staff position there, he moved to the Rochester iSeries Support Center in 1992 where he supports RPG, COBOL, CL, C, C++, and WebFacing on the AS/400. You can reach Jerry at sigwing@us.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 degree in Computer Science from the University in Hamburg, Germany, and a degree in Industrial Engineering. Claus worked as a Systems Engineer at 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 iSeries AD tools for the past 10 years specializing in Visual tools. Claus is a frequent speaker at COMMON and other iSeries conferences. Claus is also the author of the IBM Press book, Understanding the IBM WebFacing Tool. He also co-authored the first edition of this redbook.

The team who wrote the first edition of this redbook

Aleksandr V. Nartovich, a Senior IT Specialist in the IBM ITSO, Rochester Center, joined the ITSO in January 2001 after working as a Developer in the IBM WebSphere Business Components (WSBC) organization. During the first part of his career, Aleksandr was 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 contact Aleksandr by e-mail at: mailto:alekn@us.ibm.com.

John Dell’Oso, a Systems Development Architect with Elders Limited in Adelaide, Australia, has eight years experience working with the iSeries platform and 14 years experience overall in the IT industry. His main areas of expertise are application and systems development with languages such as RPG and C. Currently, his focus is on WebSphere, Java, and XML-based technologies and middleware design. You can contact John by sending e-mail to: mailto:jdell'oso@elders.com.au.

Pentti Rasanen, an IT Specialist from IBM Finland, joined IBM in 1999 and is working with the iSeries System Support Team. His areas of expertise include High Availability software support, application development tools, operating system technical support, and iSeries software support. Before joining IBM, he spent over 17 years working as a Software Developer and Designer on the AS/400 and System/38™ platforms. You can reach Pentti by sending e-mail to: mailto:pentti.rasanen@fi.ibm.com.

Thanks to the following people for their contributions to this project:

Lila Aravopoulos
Abe Batthish
Vadim Berestetsky
Harold Briscoe
Emily Bruner
Alison Butterill
Linda Chui
Dave Cheng
Mazen Faraj
John Fellner
Satish Gungabeesoon
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 introduces you to the new development tool, WebSphere Development Studio Client for iSeries (also called briefly Development Studio Client or WDSc). It covers the following topics:

- Why you need this type of tool
- The architecture of Development Studio Client
- The basics of the development environment, including perspectives, views, editors, etc.
- The sample applications used in this book
Chapter 1. The iSeries Developer Roadmap

Traditional applications continue to drive day-to-day business in many companies. Many of these applications follow a traditional style of design that is often called *spaghetti design code*. Yet, while these programs are the key to the successful operation of the business, they provide a distinct set of challenges. Integration of modern technologies into existing applications in their current form is difficult. Yet, investment in On Demand Business capabilities is essential for businesses to survive in today’s competitive marketplace.

Enterprise modernization is the ability to re-use existing traditional applications, as well as leveraging existing skills, while delivering new On Demand Business architectures. Modernization efforts can take various forms and provide various levels of return based on the investment made.

iSeries developers have experienced many difficulties in moving existing host applications to the Web and writing new applications according to Java 2 Platform, Enterprise Edition (J2EE) specifications. To simplify this process, IBM created the *iSeries Developer Roadmap*. An integral part of this road map is the WebSphere Development Studio Client for iSeries. This chapter introduces the iSeries Developer Roadmap and explains where the components of the IBM WebSphere Development Studio Client for iSeries (WDSc) and WebSphere Development Studio Client for iSeries Advanced Editions (WDSc AE) belong in this road map.
1.1 The decision to modernize

Enterprise modernization is the re-use of assets. Virtually all businesses need to modernize existing applications, which handle their mission-critical business processes. Businesses also have to consider the skills that are available and how best to use them. The alternative to leveraging these assets can often be too costly, too time consuming, or too unreliable.

Reusing components of an application that have already been through “customer testing” helps to mitigate the risk of creating a new application. Reusing code that has already been out in the market reduces potential quality issues since they have already been identified and hopefully corrected.

Applications that are good candidates for modernization fall into one or more of the following categories:

- Application interfaces are difficult to use.
- User workflows are outdated, time-consuming.
- Applications cannot be integrated into modern workflows.
- Mission-critical processes cannot easily be adapted to changing market conditions.

They are:

- Difficult to maintain
- Difficult to extend
- Offer little or no reuse of components

A good enterprise modernization strategy can help to solve these concerns. It also provides additional benefits such as modernizing programmers' skills or increasing the productivity of the programming staff.

Rewriting a critical application can be a costly venture. The learning curve to understand an existing application can be tremendous, especially when the original programmer may be long gone. And, after a lengthy cycle to rewrite an application, there are concerns of reliability and effectiveness. While ease of use and end user productivity may improve, the overall cost may be prohibitive. Reuse of existing assets, including utilization of existing development skills, can achieve the same tangible results in less time and with less expense.

Businesses must also cope with the rapid advancement of technology. With the technology “endpoint” in question, adopting a new technology that will potentially be obsolete in the near future can be a costly mistake. Not every company can afford to be a pioneer. Reuse of existing assets guarantees that a business will not make a wrong turn. And with the advent of Web services building on proven integration technologies, modernizing the enterprise is easier than ever before.

In today's market, when uncertain economic conditions force businesses to scrutinize all the facets of their operations, time to market and return on investment (ROI) become even more critical. Enterprise modernization in general becomes an even more attractive option. The economic advantages of reuse are undeniable. It's typically cheaper to reuse than to rewrite. And, componentization allows you to take advantage of where the technology is going.

IBM offers an enterprise modernization portfolio of products and services that provide a comprehensive, end-to-end solution. The offerings range from quick improvements to user interfaces that can be implemented opportunistically to more involved fundamental changes across multiple development teams.
1.2 iSeries Developer Roadmap

One strategy that IBM proposes for enterprise modernization strategy in an iSeries environment is the *iSeries Developer Roadmap*. This initiative from IBM lays out one strategy, which allows modernization of application assets, in small, consumable pieces. Each step builds on the previous one. Eventually these steps lead to the level of modernization that is required to fulfill the needs of the business (see Figure 1-1).

The left side of the diagram illustrates today’s development environment. Most traditional applications are designed to use a green-screen style of interface, written in RPG or COBOL as the main language. Developers use the *PDM suite* of tools for their development activities. These application programs are most likely the large multi-function, structured programs that were required for optimal performance on the S/36, S/38, and AS/400 in its early days.

The far right side of the chart represents a full J2EE implementation. The design is based on the J2EE framework as an Object Oriented (OO) design. It is programmed in Java and implemented using Enterprise JavaBeans (EJBs). The developers use desktop tools in a fully integrated environment. This application structure supports a distributed environment for deployment of the application components.

All of the columns in between represent a sequence of steps that can be executed if the needs of the business dictate a move to a more modern environment. Each step is both a rest stop on the road map and a final destination. For some applications and programmers, the road ends at one location. For others, the road may be longer and reach farther to the right.

The good news is that, regardless of the final destination, doing anything along the road map is modernizing. This may include improving, upgrading, or enhancing the development environment, the application interface, the application architecture, or the application framework and implementation. Any of these areas provide an return on the investment made to achieve the step.

Let’s explore the iSeries Developer Roadmap in more detail.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5250</td>
<td>5250 Web Portlets</td>
<td>5250 Web Portlets Web Services</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RPG/COBOL</td>
<td>RPG/COBOL</td>
<td>RPG/COBOL</td>
<td>RPG/COBOL Java DB2 and SQL</td>
<td></td>
<td>EJBs WDO</td>
</tr>
</tbody>
</table>

Figure 1-1  iSeries Developer Roadmap
Today’s environment

An enormous number of iSeries development shops still maintain, and in fact are developing thousands of lines of 5250 green-screen application code. They are writing and maintaining traditional styles of RPG and COBOL, large multi-functioned structured programs (see Figure 1-2). There is some movement to using modern compilers and the modular techniques of Integrated Language Environment (ILE). However, most shops have inherited a legacy of traditional programs. The developers in these shops are using green-screen development tools to build or enhance these green-screen applications.

The suite of tools that the majority of development shops have is called Application Development ToolSet (ADTS). ADTS is made up of several component products, such as Programming Development Manager (PDM), Source Entry Utility (SEU), Screen Design Aid (SDA), and Report Layout Utility (RLU). The base components of these tools are more than 20 years-old. While they were considered “state-of-the-art” when they were introduced, no features and functions have been added to them in the past few years. This makes it quite difficult for a developer to create modern applications using old development tools.

There is no question that companies have made a significant investment in the creation of these traditional applications. These are the programs that run the business. Companies have trained programmers and developers to use the green-screen development tools. However, for most of these applications and most of these programmers, it is time to update the architecture, update the skills, and move the environment into the twenty-first century.

Better tools

The first step in the road map simply replaces the traditional development tools with more exciting and modern tools to support the same code base (see Figure 1-3). This does not involve any change to the applications within the organization. These applications remain coded in RPG or COBOL, using data description specification (DDS) code as required.

The industry standard for development tools in today’s environment is the desktop. It enables developers to take full advantage of its graphic capabilities. Graphical editing, cut and paste, pop-up windows, and pull-down lists are available on the desktop, but not in a green-screen development tool.

The IBM developer desktop is based on Eclipse. The Eclipse project was initiated by IBM, but is supported by many application development tool vendors. Common behaviors for user interfaces, for team support, and for creation of components were outlined and published, along with a Java-based framework. This Eclipse framework is now available through open source and is used by tool vendors for creating components that look and behave similarly. The productivity benefits that can be realized from Eclipse are tremendous. A developer needs to learn only one set of behaviors. This skill is then transferable to all development environments, regardless of the language or task at hand. IBM calls its version of Eclipse the WebSphere Studio Workbench.

For more information about Eclipse, visit the Eclipse Web site at: http://www.eclipse.org

The WebSphere Development Studio Client for iSeries is the version of WebSphere Studio that is geared toward iSeries developers. This product includes Remote System Explorer.
RSE is the follow-on version of PDM, SEU, SDA, RLU, and the system debugger. RSE components are tightly integrated with each other and with the Integrated Development Environment (IDE) of IBM WebSphere Studio Workbench. By learning and using RSE, traditional programmers enjoy greater productivity than with host-based 3GL tools.

While working with RSE, programmers gain transferable skills that later apply to other Eclipse-based tools in WebSphere Development Studio Client. Learning RSE also opens opportunities to access the next generation of third-party tools that are plugged into the environment. RSE works with OS/400 files, commands, and jobs, as well as with integrated file system (IFS) files and Qshell commands. It also works with Linux files and commands that reside in their own logical partition (LPAR). That is, from your workstation, you can remotely access and edit files and run commands. RSE even works with files and commands in remote UNIX®, Microsoft® Windows, or Linux servers, and with local Microsoft Windows files and commands. Ultimately, as Java and Web services technologies are further adopted, this consistent support across file systems and command shells will be very important.

**Better user interface**

Tomorrow's business leaders are today's new employees, new customers, and new application users. They are conversant with modern technologies in a way never before experienced. They are Windows and mouse savvy. Their perception is that green-screen applications indicate an old and tired environment. This is true of both application users and programmers. Programming concerns can be relieved to some degree by implementing RSE from the previous step in the road map. That leaves the user interface of the business applications. IBM has several tools that assist in moving the user interface from a green screen into a more modern browser-based graphical user interface (GUI) (see Figure 1-4).

All of the refacing options from IBM require no or small changes to the underlying application code, whether it is in RPG, COBOL or another traditional language. This is a key consideration when trying to understand the potential ROI that is gained when modernizing the user interface. It is possible to simply reface the existing application, by re-using the entire application. This lowers the risk of taking a new application to market, by allowing the application code to remain intact. It is only the user interface that is being reworked.

While enhancing the user interface using the various refacing tools, consider the requirement of the application users to have access via a portal. All of IBM refacing tools create a portal-enabled solution that allows access to the applications, in the browser within a portlet.

Let's discuss the IBM alternatives to refacing.

**The IBM WebFacing Tool**

The IBM WebFacing Tool is part of the IBM WebSphere Development Studio Client for iSeries. It provides a quick way to Web-enable existing iSeries applications with minimal, if any, modifications to the original application. The IBM WebFacing Tool provides a simple mechanism to reface existing 5250 applications with Hypertext Markup Language (HTML) user interfaces. This allows users to interact with the same traditional application from either the Web browser or from green screens.

The IBM WebFacing Tool consists of two parts:

- A Display File (DSPF) to JavaServer Pages (JSP) conversion tool
- A runtime intercept to enable the iSeries application to run as a Web-enabled application as well as a 5250 display device application
The IBM WebFacing Tool is a source-code conversion technique that creates Java artifacts as a result of the processing display file source code. At execution time, when a request to run the existing program is made through a browser, the operating system (OS/400) sets an internal switch to track that the request to execute came from a browser, and not from a traditional green-screen environment. When the existing program does a write back to the screen, the status of the switch is interrogated and the output is sent back to the browser, not to the green screen.

This method requires that you have source code available. It results in the creation of a JSP for every record format in your application. If this is an application where you anticipate that having a real Java artifact provides benefit to future development efforts, then the IBM WebFacing Tool provides a good solution for you.

**IBM WebSphere Host Access Transformation Services**

IBM WebSphere Host Access Transformation Services (HATS) is a component of the Host Integration Solution for iSeries package. Various clip levels that can be acquired based on the number of concurrent users that are expected to use the transformation engine at one time.

HATS follows the architecture of a data stream interceptor. This means that it waits until the 5250 data stream has been created. Then an interceptor catches the data stream and transforms it into HTML pages based on pre-configured rules. It is possible to do further customization work on the output of the rules. This is a development effort and results in the creation of a JSP for that specific screen transformation.

The HATS rules-based design provides a good mechanism to enable an application quickly. However, if you require customization to enhance the affinity to other browser-based applications, then you must go into each application screen and define specific behaviors. A strong reason to use HATS is that, should you skip a screen during development work, or should a screen change and the new customized JSP not yet be available, HATS does not delete it. It simply uses the default rules until a new customized JSP is available.

**HATS Limited Edition**

Just for iSeries customers, HATS comes in a "lighter" version, called HATS Limited Edition (HATS LE). This version of HATS allows only limited customization. You may define default rules, but cannot use advanced functions, such as screen-image splitting or merging. Nor does it offer additional graphics capabilities, such as adding bar graphs or notebook tabs.

HATS Limited Edition is shipped with iSeries Access for Web and with the V5R3 version of OS/400 and IBM i5/OS.

**iSeries Access for Web**

iSeries Access for Web is the most recent addition to the iSeries Access Family (5722-XW1). It enables application users to access the iSeries server from a Web browser interface.

iSeries Access for Web is a set of Java servlets that run in an application server and execute various functions, such as work or print management.

Two main functions can be executed with iSeries Access for Web. The first is 5250 data stream translation to HTML. The second set of functions that can be performed is a subset of the functions available in iSeries Access for Windows.

iSeries Access for Web provides the following functions in addition to data stream translation:

- **Print**: Covers managing printers, printer outputs, printer shares, and output queues
- **Messages**: Covers message handling, such functions as display message, send message, and message queues
Jobs: Includes managing user jobs, viewing job logs, viewing printer output for completed jobs, and managing server jobs such as QZDASOINIT or the OS/400 TCP Database Server job

5250: Covers management and configuration of the 5250 interface

Database: Enables the creation and execution of Structured Query Language (SQL) queries against the DB2® Universal Database™ (UDB) for iSeries database to display and update tables

Files: Provides access to the files in your IFS; aside from browsing files, allows the upload and download of files directly to and from the IFS using the browser

Command: Allows the initiation of iSeries commands

Customize: Allows an administrator to tailor the authority of users to access various functions, such as the 5250 session

This feature is applied only at the application level on top of the object-level security. It allows the changes to the policy settings to customize the look and feel of the browser interface when connecting via iSeries Access for Web.

Other tasks can be performed, such as changing a user password.

**IBM WebSphere Portal Server**

IBM WebSphere Portal Server helps to improve employee productivity. It also helps to strengthen relationships with customers and trading partners by allowing users to interact in a personalized way with dynamic information, applications, processes, and people. In doing so, it also helps to increase satisfaction of customers, employees, and trading partners by extending and personalizing the user experience. This is a key component of the Business Portals value area.

IBM offers two editions of WebSphere Portal Server on iSeries:

- WebSphere Portal - Express for iSeries
- WebSphere Portal - Express Plus for iSeries

These editions include a portal framework to integrate the portal with applications and data sources. They offer a single, integrated point-of-access to interact with key applications, content, people, and business processes.

Another strong point for deploying WebSphere Portal solutions is security. Security features include:

- Security for managing portal membership
- User authentication
- Single signon across multiple applications and authorization for access to portal resources

WebSphere Portal also allows quick deployment of portals on a wide variety of wired and wireless computing devices that enable users to conveniently access portals where and when they need.
Better architecture

This stage in the road map represents a re-architecting of the existing application code. It is the first step where the application code is changed (see Figure 1-5).

This is invasive surgery. Traditional programs written in the style from the 80s and 90s are componentized to maximize the reusability and, therefore, the ROI. The key is to separate the user interface and business logic and isolate database access and printing. There are several key steps to review as application code is re-designed into modular style. Examples of such steps include:

- Moving to the modern compilers
- Moving to a modular style: ILE modules and procedures
- Moving database function to the database, for example referential integrity and triggers
- Designing and implementing a Web user interface, replacing green-screen display logic
- Possibly exposing some business logic to trusted trading partners through Web services

At present, IBM offers a few tools that can assist in achieving better architecture. The first is the Convert RPG Source command and the ILE COBOL compiler. These two vehicles help to move the source code of the traditional program into the most modern source form. While this converts to the more modern RPG and COBOL style, it does not solve the concern of identifying and separating user interface and business logic. This is a much more manual and labor intensive task. Some knowledge of the application structure is required.

The move to an optimized Web interface should be hand-crafted at this time. If you implemented the previous step of refacing using a tool that created a JSP interface, you can extend and enhance these JSPs to support the new separation of user interface and business logic. Or you can hand-craft them using the Web Tools component of WebSphere Development Studio Client.

Moving function to the database is a key component of better architecture. Most traditional programmers have implemented advanced database functions, but they have done so programmatically. For example, consider referential integrity. To maintain the integrity of the data, most programmers perform database checking via a read operation within their application code. Since V3R1, referential integrity can be coded into the database and the DB manager can handle the checking from wherever the request originated. Referential integrity, triggers, and stored procedures are all techniques that can be implemented as part of the move to modularity in the application code.

After the application code is re-architected into modules, it is possible to allow the exposure of some logic components through the creation and consumption of Web services. Web Services allows trusted trading partners to, in essence, share pieces of application code. One company requests the execution of a piece of code on another company's server. Parameters are passed back and forth, but the application code resides on the partners' machine. This technology is based on having discrete pieces of business logic or modular code.

The Web Tools of WebSphere Development Studio Client also provide Web Services wizards to allow the exposure of key pieces of logic to trusted trading partners. There are wizards to both create and consume Web services.

It is evident from the marketplace that modern applications will most likely be written in multiple languages. Languages will be chosen based on their applicability to do the job.
required. This means that for most applications, there will be a blending of language components, perhaps Java for the user interface, RPG for printing, and a combination of both for the business logic.

**Better portability**

This step of the road map is focused on the requirement to have an application developer create e-business solutions, based on Object Oriented concepts. In today's environment, portability is achieved by writing the application in the Java language. Business logic can then be ported and deployed to any server running a Java Virtual Machine (JVM). See Figure 1-6.

As noted in “Better architecture” on page 10, development shops that have been through this modernization effort, tell us that their applications may not be 100% Java when they are finished. They may still leave some functions in a more traditional language, such as RPG or COBOL, for performance reasons. However, as the code is now modularized, these functions are easily recognizable and can be customized for the next targeted platform.

Developers, in the portability stage, work in one development language and yet, target multiple platforms for execution. For application providers, this may serve the purpose of allowing them to target new markets. For all development shops, it means that the programmer talents can be shared across multiple environments. For companies that support multiple platforms, moving to the portability stage of the road map can realize significant savings in development cost.

The Java development tools, which are part of WebSphere Development Studio Client, can assist with this stage of the road map. These tools are consistent across the entire IBM WebSphere Studio product family. In the iSeries version of the Studio tools, WebSphere Development Studio Client, extensions were added to the base product to allow for easy affiliation with the iSeries platform. Some examples of these features include easy export of Java projects to the iSeries IFS and launching the Java compiler on the iSeries from the desktop tools.

For many development shops, this better portability stage is the final destination. After the application is portable, it moves to new platforms. This is the only requirement from the business. Other shops, may want to continue to the next stage.


Better scalability

The goal of better scalability is to create completely portable, distributed, and highly available applications which support high volume and high transaction rates in an e-business environment. The J2EE implementation with service-oriented architecture (SOA) and EJB is considered both the industry and IBM standard for cross-platform business applications (see Figure 1-7).

The use of these standards expands a developer's ability to extend applications and build new applications from previously constructed components. This allows a development shop to realize the full benefit of modern application architecture, components, modularity, re-use, and flexibility to respond to business requirements.

WebSphere Development Studio Client Advanced Edition for iSeries contains the EJB tools that are necessary to implement this stage. This product is not included in the base set of tools within the WebSphere Development Studio (5722-WDS) product. A development shop has to acquire licenses for each developer who needs this type of development environment.

For more information about the iSeries Developer Roadmap, see:
http://ibm.com/iseries/roadmap

1.3 WebSphere Development Studio for iSeries and WebSphere Development Studio Client for iSeries

Part of an enterprise modernization strategy must include a plan to update the tools that a developer uses to create modern applications. It is not possible to create a browser-based GUI using a green-screen development environment. Over the years, many tools have been available that allow the modernization of application assets and the integration of modern technologies. However, until recently a developer needed to manage and learn the behaviors of many tools to accomplish a modernization task: one tool for the maintenance of RPG and COBOL, one tool for creating a Web interface, one tool for Java development, and so on.

At V5R1, IBM chose to package many of the common development tools together into one package. This package was named WebSphere Development Studio for iSeries. This package is comprised of five server tools: RPG, COBOL, C, C++ and the ADTS set of green-screen development tools including PDM, SEU, SDA, and so on. WebSphere Development Studio for iSeries also included unlimited entitlement to the desktop suite of tools called WebSphere Development Studio Client for iSeries. This is a suite of programmer tools that load onto the programmer's desktop. Built on top of the Eclipse architecture, WebSphere Development Studio Client for iSeries is a fully IDE.

WebSphere Development Studio Client for iSeries contains components to assist with Java or Web development as well as offers perspectives for RPG and COBOL development, Extensible Markup Language (XML) creation, Web services development, and development environments for other modern technologies. The IBM WebFacing Tool is an integral part of this product.

In January of 2003, IBM announced an advanced version of the WebSphere Development Studio Client for iSeries product, called WebSphere Development Studio Client, Advanced Edition. It is a super-set of the base WebSphere Development Studio Client for iSeries.
product. This tool suite adds the ability to create EJBs and offers more advanced functions in the WebFacing perspective.

1.4 Focus of this IBM Redbook

The focus of this IBM Redbook is the WebSphere Development Studio Client for iSeries and the WebSphere Development Studio Client for iSeries Advanced Edition. Both products cover the "Better Tools" part of the iSeries Developer Roadmap.
Creating applications for iSeries: A new set of tools

This chapter describes the new WebSphere Development Studio Client for iSeries. It explains how it evolved from WebSphere Development Tools as part of WebSphere Development Studio for iSeries (5722-WDS).

In its second part, this chapter also highlights WebSphere Studio Site Developer (WSSD) and WebSphere Studio Application Developer (WSAD), the members of the WebSphere Studio family. They come as a component of WebSphere Development Studio Client for iSeries and WebSphere Development Studio Client Advanced Edition for iSeries (Development Studio Client) respectively.

This chapter also looks at WebSphere Workbench, which resulted from the Eclipse project (see 3.1, “Eclipse: The start of a new era of tools” on page 30, for more about Eclipse).
2.1 IBM WebSphere Development Studio for iSeries

Starting from OS/400 Version 5 Release 1, most previous program products that were sold separately for application development were bundled together into a single product called IBM WebSphere Development Studio for iSeries (5722-WDS). 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.

WebSphere Studio provides a flexible, portal-like integration of multi-language, multi-platform, and multi-device application development tools. These tools maximize developer productivity, increase return on investment (ROI), and improve overall time to value.

During the year 2001, WebSphere Studio and VisualAge for Java were merged together with a new application development workbench based on the Eclipse project. The new product was called WebSphere Studio Application Developer. Currently, there are several development tools based on the Eclipse platform.

Similarly, with OS/400 V5R3, WebSphere Development Tools for iSeries were restructured and renamed to WebSphere Development Studio Client for iSeries. It simply referred to the client-side tools. For more information, refer to the WebSphere Development Studio Client for iSeries Web site at:

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

WebSphere Development Studio Client for iSeries consolidates the key application development tools into one client-based package. The package in version 5.1.2 includes the following workstation components:

For the WebSphere Development Studio Client for iSeries V5.1.2

- WebSphere Studio Site Developer V5.1.2:
  - Eclipse-based IDE for creating Web, Web services, and Java applications.
  - Enterprise Generation Language (EGL) for generating Java applications from a 4GL procedural language.

- A powerful IDE for iSeries development.

- Wizards to help Java, Web, and Web services developers access iSeries data and applications.

- Cooperative Development Environment (CODE): formerly called CODE/400, the classic edit/compile/debug environment for current CODE users.

- VisualAge RPG (VARPG): development environment for creating event-driven GUI RPG applications (See Chapter 15, “Introduction to VisualAge RPG” on page 611, for details).

- IBM integrated iSeries debugger (Eclipse-based) for all languages.

- Both iSeries refacing tools, IBM WebFacing Tool and Host Access Transformation Services (HATS) toolkit.
For the WebSphere Development Studio Client Advanced Edition for iSeries V5.1.2

WebSphere Development Studio Client Advanced Edition for iSeries Version 5.1.2 provides all the capabilities of WebSphere Development Studio Client for iSeries Version 5.1.2 plus the following enhancements:

- WebSphere Studio Application Developer 5.1.2 to develop sophisticated, scalable, and fully J2EE-compliant application (including Enterprise JavaBeans).
- IBM WebFacing Tool advances enhancements to deliver portal support and to leverage and extend Web-enabled applications with Web application.
- Portal development tools
- iSeries Web and WebFacing Tools to support Enterprise Identify Mapping (EIM) based single sign on capability.
- iSeries Java Tools Java Connector Architecture (JCA) support.

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 Integrated Language Environment (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, 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 that takes you through the steps required to create Web input and output pages.
- **Create graphical user interface (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 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 the RPG IV language definition.

To learn about the V5R3 enhancements made for RPG IV, go to:

http://www.ibm.com/software/ad/wds400/about/ile_rpg.html
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 V5R3 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 V5R3 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

Application Development ToolSet (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, such as 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 to help 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)
2.2 WebSphere Development Studio Client for iSeries

Figure 2-1 shows the contents of WebSphere Development Studio for iSeries (5722-WDS).

WebSphere Development Studio Client comes in two editions:

- The first one named WebSphere Development Studio Client Advanced Edition for iSeries includes WebSphere Studio Site Developer.
- The second one named WebSphere Development Studio Client Advanced Edition for iSeries includes WebSphere Studio Application Developer.

All iSeries-specific tools in WebSphere Studio products are built as plug-ins. They include:

- Remote System Explorer and iSeries development tools (see Part 2, “The Remote System Explorer and iSeries Projects” on page 73)
- IBM WebFacing Tool (see Part 3, “Bringing your OS/400 applications to the Web” on page 203)
- iSeries Web development tools (see Chapter 13, “Building Web interactions with iSeries Web Tools” on page 431)
- iSeries Java development tools
- Integrated iSeries debugger (see Chapter 14, “Debugging and testing options” on page 545)

Customers can build their own plug-ins for iSeries and integrate them with Development Studio Client.
2.2.1 WebSphere Studio family

The WebSphere Studio product family is built on top of the Eclipse Workbench as a set of plug-ins that conform to the Workbench’s open standard application programming interfaces (APIs).

The WebSphere Studio family of products currently has the following members (Figure 2-2):

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

These products provide support for end-to-end development, testing, and deployment of Web and Java 2 Platform, Enterprise Edition (J2EE) applications. See Figure 2-3 that shows the relation between different WebSphere Studio components.
The WebSphere Studio product family provides integrated development tools for most e-business development roles including Web developers, Java developers, business analysts, architects, and enterprise programmers. The customizable, targeted, and role-oriented approach of the Workbench will be a common characteristic of future products in the WebSphere Studio family.

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 framework, services, and tools for building tools.

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 2-4):

- Platform runtime
- 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, Red Hat, 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 are 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 an 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**
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 Java 2 Server Edition (J2SE) and J2EE specifications and supports JavaServer Pages (JSPs), servlets, Hypertext Markup Language (HTML), JavaScript, and DHTML. It further includes tools for developing images and animated GIF files.

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 can also refer to the *WebSphere Studio Application Developer Programming Guide*, SG24-6585.

**WebSphere Studio Application Developer**
Application Developer is designed for professional developers of Java and J2EE applications, who require integrated Java, Web, Extensible Markup Language (XML), and Web services support.

It includes all of the features of Site Developer. Plus it adds tools for developing EJB applications and performance profiling and logging tools for both local and remote execution.

Developers can quickly build and test business logic and enhance the presentation artifacts with built-in Web creation tools inside the Application Developer IDE before deploying to a production server.

Using the performance profiling and tracing tools, it is possible to detect application performance bottlenecks early in the development cycle. Furthermore, the built-in test
environment for WebSphere Application Server and advanced tools for code generation help shorten the test cycle.

For more information, see the WebSphere Studio Application Developer Web site at:

http://www.ibm.com/software/awdtools/studioappdev/

**WebSphere Studio Application Developer Integration Edition**

Integration Edition includes all of the functionality in Application Developer, plus:

- Powerful graphical tools: Help you quickly and easily build custom application adapters to integrate your J2EE application with your back-end systems. Also help you save time and money by reusing existing resources.
- Visual flow-based tools: Increase developers’ productivity by allowing them to visually define the sequence and flow of information between application artifacts such as adapters, Enterprise JavaBeans components, and Web services.
- Wizards that help in building and deploying complex Web services out of adapters, EJB components, flows, and other Web services.
- Support for the full set of Enterprise services provided by WebSphere Application Server Enterprise Edition: This includes business rule beans, internationalization, and work areas that deliver additional integration capabilities, developer productivity, and business agility.

For more information, see the WebSphere Studio Application Developer Integration Edition Web site at:

http://www.ibm.com/software/awdtools/studiointegration/

**WebSphere Enterprise Developer**

Enterprise Developer includes all of the functionality in WebSphere Studio Application Developer Integration Edition. It also supports the following functions:

- Integrates transactional environments such as CICS® and IMS™
- Creates new Enterprise Information System (EIS) applications as part of a J2EE application
- Develops and maintains z/OS® applications
- Supports Java, COBOL, PL/I, and enterprise generation language (EGL) development
- Can implement struts-based MVC applications using connectors and EGL.

Another core technology that is integrated within Enterprise Developer WebSphere Studio Asset Analyzer (WSAA). This technology identifies application processes and connecting points. It also provides the ability to generate components from existing code.

Enterprise Developer provides a visual construction and assembly-based environment supporting the implementation of enterprise level applications, including support for the multiple developer roles and technologies required by those applications. Some examples of technologies supported are HTML, JSPs, servlets, EJBs, COBOL, EGL, PL/I, and connectors.

For more information, see the WebSphere Studio Enterprise Developer Web site at:

http://www.ibm.com/software/awdtools/studioenterprisedev/
2.2.2 IBM WebFacing Tool

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

No or little change is required to the application's business logic. 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 IBM WebFacing Tool.

For more information about the IBM WebFacing Tool, see Part 3, “Bringing your OS/400 applications to the Web” on page 203. Also see the IBM WebFacing Tool Web page at:
http://www.ibm.com/software/awdtools/wdt400/about/webfacing.html

2.2.3 Web tools: iSeries extensions

WebSphere Studio is the premier IBM Web development tool. As described in 2.2.1, “WebSphere Studio family” on page 20, it has evolved from a single tool to a whole family of products tailored to different needs. WebSphere Studio Site Developer is included with WebSphere Development Studio Client for iSeries and WebSphere Studio Application Developer is included with WebSphere Development Studio Client Advanced Edition for iSeries. 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 components

Web components (previously called Design-time Controls (DTCs)) are the fields and controls that appear on the Web pages of your Web application. You insert the Web components on the JSP pages that you design and then use the Web Interaction wizard to link them to parameters in your ILE program and Java Bean method.

The iSeries Web Components are:

- Button
- Check box
- Combo box
- Hyperlink
- Image
- Image button
- Label
- Radio button group
- Selection box
- Table
- Text area
- Text entry
These special features can be built with WebSphere Development Studio Client for iSeries and are used to communicate between the Web page and the iSeries host program.

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

### 2.2.4 Java development tools

Java development tools and iSeries Java development tools give you the ability to develop Java applications and write, compile, test, debug, and edit programs written in the Java programming language for Java applications development. Java development tools include an integrated environment that supports the complete cycle of Java program development.

The iSeries Java development tools include:

- **Program Call wizard**
  This extension generates a `JavaBeans` 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.

- **The iSeries Java Transform and Run view**
  This extension provides all the essential tools to remotely develop iSeries Java applications. You can remotely transform, run, and debug Java applications.

- **Pre-supplied JavaBeans**
  The extension continues to provide useful JavaBeans previously available in the IBM VisualAge for Java Enterprise Toolkit for AS/400 (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.
  - *Database Field List bean*: To retrieve a list of fields in a specified iSeries database file and record.

- **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.
2.2.5 CoOperative Development Environment

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, data description specification (DDS), and Java. It is an easy, cost-effective, and productive way to work with iSeries host applications on your Windows 98, ME, NT, 2000, or XP workstations.

You can use Client Access, Personal Communications, or TCP/IP to connect to the iSeries server. However, CODE does not require a 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 provides the productivity of the PC environment and integrity of the iSeries.

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, reports, and physical files both fast and fun.

Another useful feature of CODE is the CODE Project Organizer. With this utility, you can access and manipulate iSeries objects with greater ease than PDM. You can also 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 WebSphere Studio Workbench.

2.2.6 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 VARPG compiler is based on RPG IV.

VisualAge RPG features point-and-click simplicity for RPG programmers at the workstation. Use VARPG 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. VARPG integrated components allow application developers to preserve their current skills and easily develop OS/400 applications with GUIs. With VARPG, 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 VARPG 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 VARPG source, making RPG a powerful e-business language. VARPG offers:

- Visual Development Environment
- RPG compiler on the workstation
- Programmable language-sensitive editor
- Display file import facility
- Transparent PC-to-OS/400 connectivity
- Source-level debugger
The Eclipse framework

This chapter briefly introduces you to the new GUI framework called *WebSphere Studio Workbench*. WebSphere Development Studio Client for iSeries (WDSc) 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 IT 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 IT 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 Java Server 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

The IBM framework, which is built on Eclipse technology, is called WebSphere Studio Workbench. You can learn more about the WebSphere Studio Workbench in Chapter 1 of 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 the J2EE specification imposes the new requirements on the development tools. For example, J2EE can define 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 include building an application or application component, such as Hypertext Markup Language (HTML) pages, JavaBeans, or servlets. IBM has used this role-based approach in the design of WebSphere Studio Workbench.

This chapter 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 such resources as 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
- Projects
- Perspectives
- Views
- Editors
- Crash recovery

3.3.1 Workbench

When you start the WebSphere Development Studio Client for iSeries, you are prompted to select your workspace location (Figure 3-1). You may accept the default location, or you can browse the new location.

![Figure 3-1 Selecting the workspace location](image)

If you select the Use this workspace as the default and do not show this dialog box again option and later you need to change your default workspace location, add the -setworkspace parameter to your command. Your program invocation command should be:

```
"<Install Path>\WDsc.exe" -feature com.ibm.WDSClient -setworkspace
```

Where `<Install Path>` is the drive letter and path where you installed your WDSc client. This part should not be changed.

**Tip:** You can create several work spaces. By doing so, your work is organized properly and you can achieve better performance.

After you start the Workbench, you see a single window. The Workbench window displays one or more perspectives. A perspective contains views, such as the Navigator, and editors. More than one Workbench window can be open at any given time.
3.3.2 Projects

Projects are base component for building, version management, sharing, and organizing resources. A project can contain session and persistent properties, settings for environmental variables, and references to other projects.

The types of projects are:

- **Simple project**: Simple projects are generic projects that contain files and folders.
- **iSeries project**: iSeries projects provide you with a workspace on your local workstation that will contain local code for a project. You can work while disconnected with the iSeries Project Navigator, and occasionally perform synchronization to build your project and push your work to the server.
- **Java project**: Java projects contain Java code.
- **Web project**: Web projects contain the resources needed for Web application, including servlets, JSP files, Java files, static documents (for example HTML pages or images), and any associated metadata.
- **WebFacing project**: WebFacing projects enables you to quickly convert your DDS display file source members so that the user interface of your iSeries programs can run in a browser.
- **J2EE project**: J2EE projects contain Connector project, Application Client Project and Enterprise application projects.
  - The Connector is a J2EE standard extension mechanism for containers to provide connectivity to enterprise information systems (EISs).
  - The Enterprise application projects contain references to the resources needed for enterprise applications and can contain a combination of Web modules, Java Archive (JAR) files, EJB modules, and application client modules.
  - The Application client projects contain the resources needed for application client modules.
- **EJB project**: EJB projects contain the resources for EJB applications.
- **Eclipse Modeling Framework (EMF) project**: Eclipse Modeling Framework is a Java framework and code generation facility for building tools and other applications based on a structured model.
- **Server project**: Server projects describe the test environment for applications.
- **Plug-in Development Project**: Plug-in Development Projects help you to develop platform plug-ins while working inside the platform workbench
- **Enterprise Generation Language (EGL) project**: Enterprise Generation Language is a development environment and programming language that enables you to write full-function applications quickly, thereby freeing you to focus on the business problem your code is addressing, rather than on software technologies.

3.3.3 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 of responsibilities in the development process. Figure 3-3 shows a sample workbench window with multiple perspectives opened. The *Remote Systems* perspective is opened by default.

To open more perspectives, select **Window → Open perspective** and select a suitable perspective from the menu as shown in Figure 3-2.
Alternatively, you can click the Open perspective ( ) button (see Figure 3-3) and select a perspective.

Let's look at the 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 menu. Each perspective button has a distinct image associated with it. Right-clicking any perspective button displays a menu from where you can select the Close perspective option.
3.3.4 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 in Web Perspective provides the way to see all multimedia resources that come with the tool. The Navigator view presents the existing projects in a tree-like manner. By its functionality, the Navigator view is 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.
In our example, the Remote Systems perspective has five views:

- **Remote Systems view** (Figure 3-4): The Remote Systems view contains information about your current connections, filters, and much more.

![Remote Systems view](image)

**Figure 3-4** The Remote Systems view

- **Properties view** (Figure 3-5): The Properties view displays property names and basic properties of a selected resource.

![Properties view](image)

**Figure 3-5** Properties view

- **View of the Welcome page** (Figure 3-6): The Welcome page gives you basic information about the product.

![Welcome and Editor view](image)

**Figure 3-6** Welcome and Editor view
Tasks (Figure 3-7): The Tasks view contains line items for system-generated problems, warnings, and errors. You can add your own entries to the table, for example to build a list of to-do items, or tasks.

Outline view (Figure 3-8): The Outline view displays an outline of a structured file that is currently open in the editor area. It lists structural elements. The contents of the outline view are editor-specific. In our example, an ILERPG program is opened in the Editor view.

There are several tabs under the Remote Systems and Tasks views. These tabs provide an easy way to switch between opened views (labeled as “Stacked views” in Figure 3-3 on page 34).

If you need to open a view, which is not shown in your Workspace, select Window → Show view. Click a suitable view at the selection list (Figure 3-9) or browse for more views by clicking Other.
3.3.5 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 right-clicking a data description specification (DDS) file member and selecting Open with → CODE Editor.

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, such as an XML, Page Designer, 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 (Figure 3-10).

Tabs in the editor area indicate the names of resources that are currently open for editing. An asterisk (*) indicates that an editor has unsaved changes.

The gray border at the left margin of the editor area is the Editor information area. It may contain icons that flag errors, warnings, or problems detected by the system. Icons also appear if you created bookmarks, added breakpoints for debugging, or recorded notes in the Tasks view.

If you hover a mouse cursor over the editor tab, the temporary source location is displayed.
Figure 3-10 Editor area
You can customize which editor is used with what type of a resource. You go to the workbench properties. Select Window → Preferences. On the Preferences window (Figure 3-11), expand the Workbench tree and click File Associations.

![Figure 3-11 File Associations](image)

Figure 3-11 File Associations
The workbench allows you to open multiple files in multiple editors. Unlike views, editors cannot be dragged outside the workbench to create new windows. However, you can tile editor sessions within the editor area to view source files side by side (Figure 3-12).

To tile the editor view, open two or more files to editor view, select one of editor tabs. Holding down the left mouse button, drag the editor over the left, right, top, or bottom border of the editor area and release the mouse button.

For more information about the editors, see 6.1, “Editing source members in the RSE” on page 140.

### 3.3.6 Crash recovery

The workbench periodically saves a snapshot to reduce the risk of losing data due to crashes:

- Saved data is never lost since it is written to disk immediately (on save).
- Unsaved data in open editors may be lost, depending on the editor implementation.
- Bookmarks and tasks may be lost.
- If a crash occurs during Concurrent Versions System (CVS) synchronization, the workbench may be out of sync. You can check by performing the synchronize operation again.
- Previously-created projects are never lost.

To set the save interval, select **Window → Preferences**. On the Preferences window (Figure 3-13), click **Workbench**. Type a new value in the Workspace save interval field and click **OK**.
3.4 Development Studio Client features

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

3.4.1 Tools

The main tools of WebSphere Development Studio Client Advanced Edition for iSeries are:

- The workbench-based integrated development environment
- Remote System Explorer and iSeries development tools
- iSeries Web development tools
- iSeries Java development tools
- Struts environment support
- Web Services development tools
- XML development tools
- Server development tools
- Database development tools
- The IBM WebFacing Tool
3.4.2 Common features

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

- Importing and exporting files by dragging and dropping
- Copying, renaming, and moving
- Searching
- Bookmarks
- Rearranging views and editors
- Fast views
- Builders
- Local history
- Team environment
- Compare editor

Importing and exporting files by dragging and dropping
You can use your operating system's file system explorer to import or export a copy of a folder or file from the workbench to the file system.

Note: You must drag the resource or resources to the exact location in the Navigator view hierarchy where you want the resources to reside. You cannot simply drag and drop them onto a blank area in the Navigator view.

Copying, renaming, and moving
You can copy, rename, and move workbench resources using menu options in the main Perspective view (Figure 3-14).
**Search window**

**Chapter 3. The Eclipse framework**

**Searching**

You can search for text strings and files from the workbench. Click the Search (🔍) icon on the toolbar to open the Search window (Figure 3-15).

**Bookmarks**

Bookmarks are a simple way to navigate to resources that you frequently use. To create a bookmark, follow these steps:
1. Right-click the Editor information area (Figure 3-10 on page 38).
2. Select Add Bookmark. A bookmark marker is added to the Editor view.
3. Open the Bookmark view. Select Window → Show view → Bookmark. Then you see the Bookmarks window as shown in Figure 3-16.
4. Double-click the bookmark item and the source code opens.

![Bookmarks](image)

**Figure 3-16  Bookmark view**

**Rearranging views and editors**

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.

  **Note:** The cursor changes to a black arrow only when you move it over the border of the workbench window.

- When the cursor looks like a window (⿱), you can drop the view, and it becomes a floating view (like a separate window).
- When the cursor looks like stacked folders (⿲), you can drop the view and it is stacked with the views underneath the cursor.
- When the cursor look like a circle with a line through it (⊙), you cannot drop the view at the current position.

After you are satisfied with this arrangement, you can save the perspective by selecting from the workbench toolbar menu Window → Save Perspective As.

If you are not satisfied to the layout of the workbench, you can always reset it by selecting Window → Reset Perspective.

**Fast views**

Fast views are hidden views, which can be quickly made visible. A good reason for using fast views is to maximize your editor view:

1. Click the title bar of a view (in our example, Remote System view). Drag it to the shortcut bar at the far left of the window (see Figure 3-17).
2. Open the source file, right-click the title bar in the Editor view and select Maximize. Now you have a large area for your editor and can quickly switch views (Figure 3-17).
3. Click the Remote Systems view button on left side of the window. Your Remote Systems view (Figure 3-18) opens.
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 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 need to have a team environment installed. You can use your local history feature.

To change the default setting for the local history feature, select Window → Preferences. On the Preferences window, expand the Workbench tree and select Local History.

To use this feature, right-click any file in the Navigator view 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 the support of a repository. The workbench comes with built-in CVS support. You can download and install the CVS server from:
http://www.cvshome.org/

To learn how to use Team environment, refer to WebSphere Development Studio Client Advanced Edition for iSeries Online Help. To learn how to install and configure CVS, see:

CVS controls access to a resource by multiple users.

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 3-19 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 the Ctrl key 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 and the part of the line (a method name, variable, and so on) that is different. The editor's control buttons are located above the file view area.
### 3.4.3 iSeries-specific features

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

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

- **Remote Systems Explorer:** For more information, see Chapter 5, “Managing your iSeries development resources using RSE” on page 75, and Chapter 6, “LPEX, the modern SEU plus much more” on page 139.
- **iSeries Projects:** For more information, see Chapter 7, “Offline, better structured application development with iSeries Projects” on page 181.
- **WebFacing:** For more information, see Chapter 10, “Using IBM WebFacing Tool to create browser interface for a 5250 application” on page 255.

### 3.5 Online help

Online help in Development Studio Client is no longer a separate perspective as it was in WebSphere Development Studio Client version 4.0. It is opened in a separate window.

IBM has done a great job in providing sufficient help. The help contains the information about the workbench itself and the other plug-ins that are integrated into the platform. This section introduces the information that is available in the online help. It also provide tips for optimal work with the help system.

When you open WebSphere Development Studio Client for iSeries, the Welcome page is shown in the Editor view (Figure 3-20). You can activate the help window by selecting the online help link.
All help information is divided into *information sets*. To select the information set, click the **Content** button and select the set you need (Figure 3-21).
To synchronize the navigation frame with the current topic, click the Synchronize navigation button. This is helpful if you followed several links to related topics in several files and want to see where the current topic fits into the navigation path.

Two of the available topics are relevant to this book:

- Development Studio Client for iSeries
- iSeries programming information

The search facility is sophisticated and presents information based on the relevance of the topic to the search criteria (Figure 3-22).

There is a quick help or infopop, as it is called in the Development Studio Client help system. 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 links to the similar topics in the help system (Figure 3-23).

If you press F1 inside of the infopop, you see the help text of the infopop (Figure 3-24).
3.6 Obtaining and installing fixes for WDSc

New versions of WDSc, such as 5.1 or 5.1.2 have to be ordered from IBM. However, before a new version of the product becomes available, there are usually one or more fixpacks ("interim fixes") providing corrections and enhancements. Usually these fixpacks include all fixes from previous updates and can be accessed by the following two methods:

- Automated Update (Preferred Method)
- Manual download

Both methods allow you to download the fixpack from the Internet. The Automated Update is started from the WDSc workbench and it automatically downloads and installs the fix for you. The manual download is initiated from a browser using FTP to transfer a file to your workstation. After completing the file transfer, you can use that file to install the fixes on one or more workstations.

**Note:** You cannot use a downloaded fixpack to upgrade WDSc from one version to the next, such as upgrading from WDSc 5.1 to WDSc 5.1.2.

3.6.1 Which version or fixpack of WDSc do you have installed?

Before looking for fixpacks, you should understand which version of WDSc you have installed. You get the first overview by selecting Help → About WebSphere Development Studio Client for iSeries from the menu bar, which results in a panel similar to the one shown in Figure 3-25. Usually the fourth digit of a version number defines the installed fixpack. For example WDSc 5.1.2.2 means WDSc 5.1.2 with interim fixpack 2.

**Figure 3-25  About WebSphere Development Studio Client Advanced Edition for iSeries**
Note that WebSphere Development Studio Client for iSeries is composed of a number of products.

- WSWB - WebSphere Studio Workbench provides an open tool integration platform that allows tool builders to seamlessly integrate teams, assets, and other products across the development life cycle. It is based on Eclipse technology.
  
  For more details go to: http://www.developer.ibm.com/websphere/workbench.html

- WSSD - WebSphere Studio Site Developer is an easy to use and productive development environment for building and maintaining dynamic Web sites, Web services and Java applications.
  
  For more details go to: http://www.ibm.com/software/awdtools/studiositedev/

- WDSc - WebSphere Development Studio Client for iSeries extends the capabilities of the above two products
  
  For more details go to: http://www.ibm.com/software/awdtools/iseries/

Each component can have its own set of fixes.

Furthermore, when you look at the About panel shown in Figure 3-25, you see buttons labeled “Feature Details”, “Plug-in Details”, and “Configuration Details”. Plug-ins are a way to write code that extends the eclipse workbench. Features are a way to package multiple plug-ins into something meaningful. For example WDSc consists of around 20 plug-ins, but only 1 feature that packages them all together.

The overall product status can be found in the Update Manager perspective. To open this perspective select Help → Software Updates → Update Manager from the menu bar. Then expand the tree under WebSphere Development Studio Client for iSeries (or WebSphere Development Studio Client for iSeries Advanced Edition) in the Install Configuration view in the upper left pane of the perspective to see the versions of each component as shown in Figure 3-26.
Figure 3-26 Install Configuration in the Update Manager perspective

If you expand the first entry you will see the various places on your hard disk where features are installed. There should be three, the first one is actually the eclipse workbench, the second would be WSSD and the last is the iSeries extensions. Expand each of these and you will see the features installed at those locations and along with each feature the current version.

3.6.2 Using the automated update

The automated update will compare the current product installed on your PC with the available fixes on the IBM servers. Missing updates are then downloaded and installed.

To see whether there are updates for WDSn make sure your PC is connected to the Internet and perform the steps listed below. If you have to access the Internet through a firewall only
by means of a proxy server, verify the settings described in “Using a proxy to connect to the Internet for the automated update” on page 55 first.

1. Select Help → Software Updates → New Updates from the menu bar as shown in Figure 3-27.

![Figure 3-27 Updating WDSc - Check for Updates]

The workbench contacts the Web sites associated with all features installed under Eclipse on your workbench (not only WDSc) to discover what versions of those features are available. The potential upgrades are presented in an Available Updates page similar to the one shown in Figure 3-28.
2. Select the feature versions that you want to upgrade, and click Next.

3. Carefully review the license agreements for the upgraded features. If the terms of all these licenses are acceptable, check “I accept the terms in the license agreements.” Do not proceed to download the features if the license terms are not acceptable.

4. Feature versions can be digitally signed by the company that provides them. This allows the user to verify more easily that the features and plug-ins that are about to be downloaded and installed are coming from a trusted supplier.

5. Click Install to allow the downloading and installing to proceed.

6. Once all the features and plug-ins have been downloaded successfully and their files installed into the product on the local computer, a new configuration that incorporates these features and plug-ins will be formulated. Click Yes when asked to exit and restart the Workbench for the changes to take effect.

Using a proxy to connect to the Internet for the automated update
If your organization uses a firewall which only allows access to the Internet through a proxy server, the Eclipse workbench may not be able to connect to the IBM servers to check for and download fixes. To allow the workbench to take advantage of a proxy server perform the following steps:

1. Select Window → Preferences from the menu bar.
2. Click Install/Update in the left pane of the preferences dialog as shown in Figure 3-29.
3. Select the Enable HTTP proxy connection check box.
4. Enter the IP address or host name of your proxy in to the HTTP proxy host address field.
5. Enter the port number into the HTTP proxy host port field.
6. Click the Apply button.

There is also a second setting which allows you to define a proxy:

7. Click Internet in the left pane of the preferences dialog as shown in Figure 3-29.
8. Select the Enable proxy check box.
9. Enter the IP address or host name of your proxy in to the proxy host field.
10. Enter the port number into the HTTP proxy host port field.
11. In addition you can also select one of the following:

- Select the *Use SOCKS* check box or
- Select the *Enable proxy authentication* check box and enter a user Id and password to access the proxy.

**Other problems using the update manager**

In some cases problems have been reported with getting updates because of proxy server issues although the settings have been entered as described above.

The one problem is specifically with the Microsoft Internet Security and Acceleration Server (ISA). This problem is addressed in the following Microsoft Knowledge base article:

http://support.microsoft.com/default.aspx?scid=kb;en-us;295667
3.6.3 Installing new features with the update manager

In some cases, the automated update cannot be used at all. Those situations include:

- You have slow or unstable Internet connections and want to use a download manager with resume capability.
- You have difficulties accessing the live IBM server from behind firewalls in spite of the tips provided in the previous sections.
- You would rather download the fixes and install at a later time.
- You are a site administrator for a group of developers who do not have direct access to the Internet.

For these or similar cases, the fixpacks can also be downloaded the IBM Software Support Web site. To do so perform the following steps:

1. Open \http://ibm.com/software/support\ with a Web browser.
2. Select Downloads, the search for WebSphere Development Studio Client.
3. Follow the instructions to download the fixpack.
4. Unzip the file to a directory.
5. Open the “Instructions.html” file for install instructions as shown in Figure 3-31.

Figure 3-31 Instructions to manually install a WDSc Fixpack
To install the fixpack, you need to use the *Update Manager* that you can open by clicking **Help → Software Updates → Update Manager** as shown in Figure 3-32.

Then use the *Feature Updates* view in the lower left pane to navigate to the file containing your fixpack as shown in Figure 3-33.
However, as described above, make sure to follow the steps described in Instructions.html which comes as part of the instal package.

3.6.4 If problems occur after applying a fixpack

If you encounter problems with some work spaces after applying a fix perform the following steps:

1. Shutdown the WDSc workbench.
2. Open a command window and type:
   
   ```
   cd C:\Program Files\IBM\WebSphere Studio\Site Developer\V5.1.2\eclipse
   ```
   
   3. Then type the following command: (this will not affect projects in the workspace):
   
   ```
   eclipse.exe -initialize -data c:\workspace
   ```
   
   4. Start WDSc with the modified workspace.
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 iSeries Web Components that are replacing the ActiveX Design-time Controls (DTCs). iSeries Web Components can be used and rendered in the Page Designer. Web Components provide such controls as textField, label, dropDown, pushButton, and so forth.

Chapter 10, “Using IBM WebFacing Tool to create browser interface for a 5250 application” on page 255, uses the RPG Order Entry application to Web-enable the application with the IBM WebFacing Tool. The goal is to use the existing RPG application to service both the Web application and the host 5250 application.
4.1 Overview of the Order Entry application

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

4.1.1 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 4-1 The company structure](image)

4.1.2 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, etc.)
- Customer information (name, address, telephone number, etc.)
- Order information (date, time, shipper, etc.)
- Order line information (quantity, delivery date, etc.)
- Item information (name, price, item ID, etc.)
- Stock information (quantity in stock, warehouse ID, etc.)

4.1.3 Application files

These files comprise the application. They are located in the LIB6961 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.

### 4.1.4 Database table structure

The ABC Company database has eight tables:

- District
- Customer
- Order
- Order line
- Item
- Stock
- Warehouse
- History

Figure 4-2 shows the relationship among these tables.

![Figure 4-2 Table relationships](image)

### 4.1.5 Customer transaction flow

The following scenario takes 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

Follow these steps to test the sample application:

1. Add LIB6961 to your library list.
2. Start the application by running the following command:

   ```
   CALL ORDENTRCL
   ```

When the Order Entry application is started, it shows the Parts Order Entry display (Figure 4-3).
3. Type a customer number and press F4 to search for the customer (Figure 4-4).

If you cannot find a suitable customer from the database, you can create a new one:

a. Press F5 to open the customer listing.

b. From the customer listing, press F6. Then you see the Add Customer Details display (Figure 4-5).
c. Enter a customer number and press F4 to find the correct district (Figure 4-6).

d. Select the district and press Enter.

e. Enter the first name, initial, and last name.

f. Press F6 to add the new customer.
4. Select the customer and press the Enter key. Now you see the customer on the Parts Order Entry display (Figure 4-7).

![Figure 4-7](image1)

5. An additional prompt is displayed that allows you to type a part number and quantity. If you do not know the part number, press F4 to view a window that contains a list of available parts (Figure 4-8).

![Figure 4-8](image2)
6. Press F12 to close the window and return to the initial display. Or you can scroll through the items in the list until you find the part you want. Type 1 in the option field and press the Enter key to indicate your choice. The selected part is added to the initial display (Figure 4-9).

![Figure 4-9 Entering a parts order](image1)

7. Type 2 next to an entry in the list to change the order. Press Enter. A display opens that allows you to change the order line (Figure 4-10).

![Figure 4-10 Changing the order line](image2)
8. When the order is complete, press F6 to add a new order to the database. The 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 4-11). The printed order is made by the PRTORDER (Print Orders) RPG server job.

<table>
<thead>
<tr>
<th>File</th>
<th>PRTORDERP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control</td>
<td></td>
</tr>
<tr>
<td>Find</td>
<td></td>
</tr>
<tr>
<td></td>
<td>+.........1......2......3......4......5......6......7......8</td>
</tr>
</tbody>
</table>

```
ABC Company - Part Order
0002, RASANEN, Pentti S          Order Nbr: 3583
00001 Ave, ABC                    Order Date: 4-07-2003
Bldg 00001                        Del_01001_2345
Des Moines, IO 07091-2345
```

<table>
<thead>
<tr>
<th>Part Description</th>
<th>Quantity</th>
<th>Price</th>
<th>Discount</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>000002 Radio_Cntl_Skl_Plane</td>
<td>1</td>
<td>$90.00</td>
<td>.140</td>
<td>$90.74</td>
</tr>
<tr>
<td>000003 Cross_Cntl_Skl_Set</td>
<td>5</td>
<td>$93.00</td>
<td>.140</td>
<td>$64.46</td>
</tr>
<tr>
<td>000010 ITSU REDBOOK 5624-2152</td>
<td>1</td>
<td>$50.00</td>
<td>.140</td>
<td>$49.94</td>
</tr>
<tr>
<td>000015 Multi_Syst_Paper</td>
<td>1</td>
<td>$71.57</td>
<td>.140</td>
<td>$71.48</td>
</tr>
<tr>
<td>000021 Ton_Gallon_Hats</td>
<td>1</td>
<td>$56.64</td>
<td>.140</td>
<td>$56.57</td>
</tr>
<tr>
<td>000023 Square_Plastic_Calendar</td>
<td>1</td>
<td>$49.84</td>
<td>.140</td>
<td>$48.98</td>
</tr>
<tr>
<td>000028 Red_Brick_House</td>
<td>1</td>
<td>$45.36</td>
<td>.140</td>
<td>$45.30</td>
</tr>
</tbody>
</table>

Order total: $833.47

F3=Exit F12=Cancel F19=Left F20=Right F24=More keys

---

4.2 Overview of the Customer Master Inquiry

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

4.2.1 Starting the application

Follow these steps to test the sample application:

1. Add LIB6961 to your library list.
2. Call the main program from an iSeries command line:
   
   ```
   CALL MLG265
   ```
3. In the first Customer Master Inquiry display (Figure 4-12), enter the search criteria. This may include the ZIP code or the name.
4. The second Customer Master Inquiry display (Figure 4-13) shows a subfile, which lists customers that match the search.

<table>
<thead>
<tr>
<th>Select</th>
<th>&quot;X&quot;</th>
<th>Zip Code Number</th>
<th>Customer Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;X&quot;</td>
<td>55901</td>
<td>00004 Aleksandr Martovich</td>
</tr>
<tr>
<td></td>
<td></td>
<td>55901</td>
<td>00001 Bob Maatta</td>
</tr>
<tr>
<td></td>
<td></td>
<td>55901</td>
<td>00007 Camille Paquay</td>
</tr>
<tr>
<td></td>
<td></td>
<td>55901</td>
<td>00008 Janet Willis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>55901</td>
<td>00005 Pietti Rasanen</td>
</tr>
<tr>
<td></td>
<td></td>
<td>55901</td>
<td>00002 Petri Nuutinen</td>
</tr>
<tr>
<td></td>
<td></td>
<td>55901</td>
<td>00006 Wilfried Blankertz</td>
</tr>
</tbody>
</table>

ENTER - Continue  F3 - End Job  F4 - Restart Zip Code

Figure 4-13 Customer selection

Type an x next to the customer you want to select and to see detailed information about them. Figure 4-14 shows the detailed description of the customer that you 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. See Chapter 13, “Building Web interactions with iSeries Web Tools” on page 431.

The sample code is available in the LIB6961.savf file, which is part of the additional materials (see Appendix A, “Additional material” on page 679, for download and installation instructions).

### 4.2.2 Overview of the Customer Master Web application

The base code we use is defined in 4.2, “Overview of the Customer Master Inquiry” on page 68, but we cannot use the native 5250 RPG code. Only the main calculations and file definitions can be used in Web application. All subfile input and output operations are handled by the subfile API. Customer Master Inquiry application source code is included in the additional material.

Our iSeries Web application uses three different ILE RPG programs:

- Cusinqnew
- Custinqsfl
- Getdetail

**Cusinqnew**

Cusinqnew program is called from CUSINQ.jsp. By calling the INIT procedure, we create the subfile user space in library QTEMP and fill the subfile with all records from the database. Procedures are coded in Custinqsfl service program. Compile the Cusinqnew program by entering the following command:

```
CRTRPGMOD MODULE(LIB6961/CUSINQNEW) SRCFILE(LIB6961/WEBCOMP01) SRCMBR(CUSINQNEW)
OPTION(*EVENTF) DBGVIEW(*SOURCE)
```

Bind it to the CUSTINQSFL service program by entering the following command:

```
CRTPGM PGM(LIB6961/CUSINQNEW) MODULE(LIB6961/CUSINQNEW) BNDSRVPGM(LIB6961/CUSTINQSFL)
```
Custinqsfl

Custinqsfl service program handles all requests for the subfile SFL1. Compile this program by entering the following command:

```
CRTRPGMOD MODULE(LIB6961/CUSTINQSFL) SRCFILE(LIB6961/WEBCOMP01) SRCMBR(CUSTINQSFL)
OPTION(*EVENTF) DBGVIEW(*SOURCE)
```

And after a successful compilation, you must bind the program to the subfile service program QDTSSFL by entering the following command:

```
CRTSRVPGM SRVPGM(LIB6961/CUSTINQSFL) MODULE(LIB6961/CUSTINQSFL) EXPORT(*ALL)
BNDSRVPGM(LIB6961/QDTSSFL)
```

getDetail

The getDetail program illustrates how to perform a READC on an iSeries Table Web Component. If you choose 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 and the field data. If no record is selected, the procedure returns 0. To create this program, first compile it as a module by entering the following command:

```
CRTRPGMOD MODULE(LIB6961/GETDETAIL) SRCFILE(LIB6961/WEBCOMP01) SRCMBR(GETDETAIL)
OPTION(*EVENTF) DBGVIEW(*SOURCE)
```

Then enter this command to bind it to the subfile service program QDTSSFL:

```
CRTPGM PGM(LIB6961/GETDETAIL) MODULE(LIB6961/GETDETAIL) BNDSRVPGM(LIB6961/QDTSSFL)
```

The Custinqsfl file contains the Web Interaction project. We included this program, which reads 10 records from database and uses PGDN procedure to read next set of records. You can find the source code in the WEBCOMP02 source file. The member name is CUSINQSFL2. The program is a pure service program. All screen interactions are coded as procedures. Therefore, in the Web Interaction wizard, you must set Program type to *SRVPGM. You must also set the entry point name for reading records from database to GETLIST. The second Web Interaction uses GETDETAIL for entry point to get the selected record from the database.

Enter the following command to create the RPG module:

```
CRTRPGMOD MODULE(LIB6961/CUSINQSFL2) SRCFILE(LIB6961/WEBCOMP02) SRCMBR(CUSINQSFL2)
OPTION(*EVENTF) DBGVIEW(*SOURCE)
```

Then create the service program bounded with QDTSSFL service program with the command:

```
CRTSRVPGM SRVPGM(LIB6961/CUSINQSFL2) MODULE(LIB6961/CUSINQSFL2) EXPORT(*ALL)
BNDSRVPGM(LIB6961/QDTSSFL)
```

Also in additional material you can find the Web Project which is used in Web Interaction. File is called Customer Inquiry.war.

Important: When you import the WAR file, you must update the iSeries run-time configuration before testing application. Right-click Web Project and select Specify iSeries Web Tools run time configuration. Then enter your iSeries host name, user ID, password, and the runtime library list.
Part 2 introduces you to two workbench perspectives:

- The Remote System Explorer (RSE)
- iSeries Projects

The objective of this part is to demonstrate how the traditional "green-screen" iSeries application developer can use these tools to effectively manage, develop, and maintain their iSeries resources and applications in a workstation-based environment.

This part includes three chapters:

- Chapter 5, “Managing your iSeries development resources using RSE” on page 75.
- Chapter 6, “LPEX, the modern SEU plus much more” on page 139, focus on how to use the RSE to manage your iSeries resources and development activities.
- Chapter 7, “Offline, better structured application development with iSeries Projects” on page 181, acquaints you with the iSeries Projects perspective and the notion of disconnected development.
Managing your iSeries development resources using RSE

This chapter introduces you to the Remote System Explorer (RSE), the modern PDM. It discusses and demonstrates how to use this workbench perspective to manage your iSeries development resources in a productive manner.

This chapter covers the following topics:

- Remote System Explorer
- Connecting to your iSeries host
  - Profiles
  - Connections
- Subsystems
  - iSeries objects
  - iSeries commands
  - iSeries jobs
  - IFS files
- Managing your iSeries objects, including filters and filter pools
- Working with your library list
- iSeries table views
- User actions, including named types
- Drag and drop functionality

The chapter was written originally based on WDSc 5.0 and now updated with the enhancements for WDSc 5.1.2. All significant changes are marked by a vertical bar at the left side of the page.
5.1 Remote System Explorer

The Remote System Explorer is more than just a GUI. It is a workbench perspective that provides access to all development resources of your iSeries server. You can think of it as an enhanced and more flexible workstation version of the Programming Development Manager (PDM). It allows you to effectively manage and organize all the iSeries resources and applications needed by application developers in an easy and user friendly way.

When you first open the workbench, the RSE is open by default. If this is not the case, follow these steps to open the RSE perspective:

1. From the workbench menu, select Window → Open Perspective. You can also click the Open Perspective button on the toolbar on the left side of the workbench window.
2. Select Remote Systems Explorer. If this option does not appear on this submenu, try these steps:
   a. Select Other. The Select Perspective window opens.
   b. Select Remote Systems Explorer (default).
   c. Click OK.

By default, the RSE perspective consists of the following editors and views (see Figure 5-1):

- Remote systems
- A properties sheet
- An area for editors
- Outline
- A tabbed notebook containing:
  - Remote System Details
  - Tasks
  - iSeries Table View
  - iSeries Commands Log

**Note:** For more information about perspectives and views and how to work with them, see 3.3.3, “Perspectives” on page 32, and 3.3.4, “Views” on page 34.
5.2 Connecting to your iSeries host

Before you connect to your iSeries host via the RSE, you must define a profile and a connection.

5.2.1 The first connection

When you connect to your iSeries host for the first time after you install WebSphere Development Studio Client for iSeries or WebSphere Development Studio Client Advanced Edition for iSeries, you must follow the steps in this section.

To connect for the very first time, follow these steps:
1. Switch to the RSE perspective.
2. Request a new connection using one of the following methods:
   - In the Remote Systems view, expand the New Connection node and then expand the iSeries node.
   - Click the Menu button in the Remote Systems view and select New Connection.
3. The New – Name personal profile window (Figure 5-2) opens. Follow these steps:
   a. In the Profile field, the machine name of your workstation is displayed by default. You can keep this name, or you can overwrite it with a profile name of your choice (MyProfile for example).
   b. Click Next.
4. The New - Remote System Connection window opens as shown in Figure 5-4 on page 80.
   Complete steps 4 through 6 in 5.2.3, “Connections” on page 79, to complete creating your first connection.

For any new profiles that you may create in the future, follow the instructions in 5.2.2, “Profiles” on page 78. For any new connections you create, follow the instructions in 5.2.3, “Connections” on page 79.

5.2.2 Profiles

Profiles are RSE entities that help you to organize and share the information about RSE connections when you have a number of connections. Profiles can be private (previously called personal) or shared. Shared profiles are known as team profiles. Team profiles allow RSE resources, such as connections and filters, to be shared among members in a development team.

In order to enable team support, you need to have an Eclipse-enabled change management repository such as CVS, ClearCase®, or similar. The team profiles contain all the connections, filters, user actions and compile commands. Whenever these items are created, you are prompted for the profile to create them in. Whenever they are shown, the total from all active profiles are shown. By default, team members only have their own profile and the Team profile active.

To create a new profile in the RSE, follow these steps:

1. Switch to the RSE perspective.

2. Click the Menu ( ) button in the toolbar of the Remote Systems view. Select Work With Profiles. You can also click the Team tab in the remote systems view.

4. The New - Remote System Profile window (Figure 5-3) opens. Complete the following actions:
   a. Enter the new profile name in the Name field.
   b. Select the Make active box to make this profile active.
   c. Click Finish.

Profiles defined in the RSE perspective are unrelated to iSeries user profiles. These profiles are used to hold connections.

**Important:** Only connections for active profiles are displayed in the Remote Systems view.

To deactivate a profile, follow these steps:
1. Switch to the Team tab in the remote systems view.
2. Right-click any of the active profiles.
3. Active profiles have a check mark (✓) next to their name. Click Active to deselect it.

To delete a profile, follow these steps:
1. Switch to the Team tab in the remote systems view.
2. Right-click any profile and select Delete.

**Important:**
- Deleting a profile also deletes the connections held by that profile.
- Only active profiles can be deleted.

### 5.2.3 Connections

RSE connections allow you to access your iSeries source members and objects. You can create different connections with different names to the same host. Each connection has its own server job so you have multiple connections each with a different specifications such as library list, set of filters, etc.
Now that you have created a profile, you can create connections to one or more iSeries host systems.

To create a new connection in the RSE, follow these steps:

1. Switch to the RSE perspective.
2. Open a New connection window using one of the following options:
   - In the Remote Systems view, expand the **New Connection** node and then expand the **iSeries** node.
   - Click the **Menu** button in the Remote Systems view and select **New Connection**.
   The New - Remote System Connection window (Figure 5-4) opens.

3. Click the **Parent profile** drop-down list. To keep your resources private, select a profile you defined (see 5.2.2, “Profiles” on page 78). To share your RSE resources, select the **Team profile**. Note, connections created in your non-private profile do not have any default filters created, that is, the subsystems will have nothing underneath them until you explicitly create a filter.

   **Important:** Only active profiles are displayed in the Parent profile drop-down list.

4. Enter values for your connection as follows (see Figure 5-5):
   a. In the Connection name field, specify a name for your connection.
   b. Click the System type drop-down list and select the remote host operating system.

   **Note:** The System type drop-down list only shows if you selected the New Connection option from the Menu button in the Remote Systems view.
c. In the Host name field, specify the name of the iSeries host. This can either be the system name or the IP address.

d. Optionally enter some text in the Description field that describes your connection.

e. Select the **Verify host name** box. This step is optional and the box is checked by default. We recommend that you keep the default action. Selecting this option ensures that an error message is displayed at the time you define your connection, in the event that your host name is specified incorrectly.

![Remote System Connection](image)

**Figure 5-5** Entering data for the new connection

5. Click **Finish**. The connection that you defined appears in the Remote Systems view.

6. Connect to the remote iSeries host:
   a. Right-click either **iSeries Objects**, **iSeries Commands**, **iSeries Jobs**, or **IFS Files** and select **Connect**. These entities are known as **RSE subsystems** (see 5.3, “Subsystems” on page 83).
   b. Enter your user ID and password when prompted.

   When you successfully connect, the Connected property in the Properties view (located in the bottom left corner of the RSE perspective) indicates yes.

   **Tip:** At this stage, we recommend that you select the **Verify Connection** option by right-clicking any one of the subsystems. This action prompts you for your user ID and password to the iSeries host. Then it checks for any missing PTFs. However, to check for the most recent PTFs available, your workstation must have a connection to the Internet.

**Saving your iSeries sign-on information**

You can save your iSeries sign-on information within the RSE. This saves you from having to sign on to your iSeries host when you request a connection through an implicit action such as expanding a subsystem filter. For more information about subsystems and filters, see 5.3, “Subsystems” on page 83.
To save your sign-on information, follow these steps when you are prompted to provide your user ID and password via the Enter Password window (Figure 5-6):

1. In the User ID field, enter your iSeries host user profile.
2. Select the **Save user ID** check box. This associates your iSeries user profile with the RSE connection.
3. In the Password field, enter your iSeries host password.
4. Select the **Save password** check box. This saves and associates your password information to your user ID in the RSE.
5. Click **OK**.

![Figure 5-6  Saving iSeries sign-on information](image)

### 5.2.4 TCP/IP ports required for WDSc

WDSc uses TCP/IP to communicate between the local system and the remote iSeries server. The following tables list the TCP/IP ports that are required for the various functions in Development Studio Client. These are the default ports and may have been changed on your iSeries.

The ports listed in Table 5-1 are all for TCP/IP connections originating from Development Studio Client to the remote iSeries. You need the first set for Development Studio Client to function properly. If you don't require the last three functions then you don't need those ports open.

**Table 5-1  TCP/IP ports required for WebSphere Development Studio Client for iSeries**

<table>
<thead>
<tr>
<th>Development Studio Client Function</th>
<th>Required TCP/IP Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote System Explorer (basic functions for retrieving lists, editing, compiling and running).</td>
<td>449 (server mapper host server)</td>
</tr>
<tr>
<td></td>
<td>446 (DRDA® (record access) host server)</td>
</tr>
<tr>
<td></td>
<td>8470 (central host server)</td>
</tr>
<tr>
<td></td>
<td>8475 (remote command host server)</td>
</tr>
<tr>
<td></td>
<td>8476 (signon host server)</td>
</tr>
<tr>
<td>RSE - Integrated File System access</td>
<td>8473 (file host server)</td>
</tr>
<tr>
<td>RSE - Interactive job support (via the Start RSE Server (STRRSESVR) command)</td>
<td>8472 (data queue host server)</td>
</tr>
<tr>
<td>WebFacing (Runtime only)</td>
<td>4004 (WebFacing server)</td>
</tr>
</tbody>
</table>
There are two places in Development Studio Client where “call-backs” are used, where a TCP/IP connection originates from the remote iSeries back to Development Studio Client.

<table>
<thead>
<tr>
<th>Development Studio Client Function</th>
<th>Required Local PC Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSE - Interactive job support (via STRRSESVR command)</td>
<td>3001 (RSE communications daemon)</td>
</tr>
<tr>
<td>Integrated Debugger (WDSc V5.0 and V5.1 only)</td>
<td>8001 (Debug daemon) and 3001</td>
</tr>
</tbody>
</table>

Note: All ports (OS/400 and local PC) can be changed. To change the local PC ports use the workbench preferences pages:

- Remote Systems → Communications
- Debug → Debug Daemon

Ports and online help

The online help system uses a port to communicate with a local server (at localhost or 127.0.0.1). This port is generated and never conflicts with any other port in use, but a firewall may be configured to block all communication to localhost. The firewall would need to be reconfigured if this is the case in your environment and you want to access online help.

5.3 Subsystems

Subsystems in the Remote Systems view divide the iSeries server into functional areas. Each iSeries connection contains four subsystems:

➤ iSeries Objects
➤ iSeries Commands
➤ iSeries Jobs
➤ IFS Files

Similar to how the functions of PDM are structured, the first subsystem provides access to the libraries, objects, and source members on your iSeries or AS/400 server as explained in “iSeries Objects subsystem” on page 84. In addition, through the other three subsystem, you have the capability to work with commands, jobs and Integrated File System (IFS) files, which are all the artifacts an application developer on OS/400 or i5/OS is dealing with.

These subsystems are displayed as nodes in a tree in the Remote Systems view (Figure 5-7). To view the subsystems for a particular connection, expand the connection node by clicking the plus (+) sign.

![Figure 5-7 Expanding the connection node](image)
You can expand each subsystem node by clicking the plus (+) sign.

5.3.1 iSeries Objects subsystem

This subsystem allows you to manage objects on your iSeries host. You can create, copy, rename, delete, and retrieve information about objects. You can also run programs and compile source members.

Expanding this subsystem reveals a set of prompting filters that let you create your own filters for viewing libraries, objects and members. These prompting filters are analogous to the Work with Libraries Using PDM (WRKLIBPDM), Work with Objects Using PDM (WRKOBJPDM), and Work with Members Using PDM (WRKMBRPDM) OS/400 commands that most iSeries developers are familiar with. However, unlike the PDM, with RSE, you can assign names to your filters, save and reuse them later.

This subsystem contains two default predefined filters for viewing libraries in your Library list and User libraries. See Figure 5-8.

5.3.2 iSeries Commands subsystem

This subsystem contains predefined command sets that are ready to run when they are expanded in the Remote Systems view. Command sets can contain one or more commands. If the command set has more than one command, then all the commands are run in sequence when the command set is expanded. Output messages from commands are displayed in the iSeries Commands Log view, by default in the bottom pane of the RSE perspective.

Running command sets

This section explains how to run a command set, by executing one of the predefined commands that creates a source member. To execute a command set, follow these steps:

1. In the RSE view expand your connection by clicking the plus (+) sign.
2. Under your connection expand the iSeries Commands subsystem.
3. Click the plus (+) sign to expand the Create source member command. Alternatively, you can also right-click the command and select Run(Prompt). The Add Physical File Member (ADDPFM) window (Figure 5-9) opens.

If you are not connected to your iSeries host, you are prompted to enter your sign-on details at this point.

4. Enter your own values for the command parameters. Click OK to run the command. The output message from the command displays in the iSeries Commands Log view.

Notice that you can also select the Advanced check box for more command parameters (similar to F10 with a CL command prompt) or the All Parameters check box for all command parameters (similar to F9).

![Figure 5-9   Entering values for the ADDPFM command](image)

**Creating your own command set**

In addition to the predefined command sets, you can create your own command sets. We explain this procedure by creating a command set for the Add Binding Directory Entry (ADDBNDDIRE) command.

To create your own command set, follow these steps:

1. Either expand the Your command(s) node or right-click the iSeries Commands node and select New → Command Set. This action opens a new window for you to define your command set.

2. In the New - Command Set window (Figure 5-10), follow these steps:

   a. In the Command field, type the ADDBNDDIRE command or you can use the Browse button to display the available commands.

      You may also click the Prompt button to check the syntax of the command you entered and optionally provide parameters for that command.

   b. Select the Prompt when run check box. This option displays a prompt window for the command when it is run to allow the entry of values for the command parameters.

   c. Click the Normal radio button. This option specifies that the command should run in the Remote System server job on the iSeries host.
d. Click Next.

**Note:** You can also choose to run the commands in a batch or interactive job. If you run interactive commands, you must ensure that you associate an interactive job with your RSE connection:

1. Start the iSeries communications daemon if it is not already running. Click the Menu (▼) button from the Remote Systems view. Select **Start Communications Daemon**.

2. Open a 5250 emulator.

3. Run the CL command:

   ```cl
   STRRSESVR NAME(your-connection-name)
   ```

   Note, the connection name is case sensitive here and must exactly match the connection name, you used in your workbench. See also “Interactive” on page 177.

4. The New - New Command Set window opens (Figure 5-11). Enter the name of the command set in the Command set name field. In the example, we name this command set **Add Binding Directory Entry**.

   If you select the **Only create command set in this connection** check box, it is not available in connections other than the one you created it in.

   You can select either your own profile or the team profile to own this new command set. That way you can choose it to be shared by the team or not.

5. Click **Finish**.
6. At this point, if you are connected to your iSeries host, the command runs and you are prompted to enter values for the command parameters. For now, you can cancel the command prompt by clicking the **Cancel** button.

7. The command that was just created appears as a child node called Add Binding Directory Entry to the iSeries Commands subsystem. The command runs when you either:
   - Expand the command set node by clicking the plus (+) sign.
   - Right-click the command set node and select Run(Prompt).

### Working with command sets

You can change, rename, copy, move, and delete command sets. You can also move them up or down within the list under the subsystem. To execute these tasks, right-click the command set and then select the appropriate action from the menu. Select the **Change** task to add more commands to a command set.

You can find more information about command sets in the online help. From the Help Contents, select **Developing** → **iSeries server applications** → **Shells and commands in the Remote Systems view**.

### Starting and stopping the Remote Servers

Starting with WDS 5.1.2, the RSE view also allows you to start and stop the various host servers running under OS/400. To do so perform the following steps:

1. In the Remote Systems view, expand your connection and then right-click **iSeries Objects** or **iSeries Commands** and select **Remote Servers**. Then select any remote
server you want to change its state. In this example, we choose to stop the WebFacing server.

![Figure 5-12 Stopping the WebFacing Remote Server](image)

2. Select **WebFacing → Stop** as shown in Figure 5-12.

By looking at the context menu, you can tell whether the selected host server is currently active or not. If it is active, the **Start** option is greyed out, if is not active, the **Stop** option is greyed out. If both options are greyed out, your RSE is not connected to your iSeries server.

### 5.3.3 iSeries Jobs subsystem

This subsystem allows you to monitor and manage jobs running on your iSeries host. Expanding the **iSeries Jobs** node reveals six predefined filters that are created when you define a connection. These predefined filters are (see Figure 5-13):

- **Active jobs**: Shows all active jobs on your iSeries host. This is equivalent to running the Work with Active Jobs (WRKACTJOB) OS/400 command.

- **My active jobs** (previously called Your active jobs): Shows all active jobs running under the user profile you used in the currently active connection.

- **My jobs** (previously called Your jobs): Shows all your jobs regardless of their status.
Chapter 5. Managing your iSeries development resources using RSE

Figure 5-13   The expanded iSeries Jobs subsystem

- **My host server jobs**: Shows all jobs serving your user profile that start with QZRC initiated by the remote System Explorer and any server job using the remote command host server such as Java applications, client access, etc.

- **My WebFacing jobs**: Shows all jobs serving your user profile initiated by a WebFacing project.

- **My VisualAge RPG DDM jobs**: Shows all jobs serving your user profile initiated by VisualAge RPG.

Note, the latter three job filters show jobs serving your user profile. Unlike the **My jobs** filter or the **Work with User Jobs** (WRKUSRJOB) CL command, most likely do not show your user profile as part of the job name. You rather see names similar to 138567/QUSER/QZRCSRVS.

The reason for this is that many of these servers are implemented as pre-started jobs which are able to serve several different users before they end. To do so, they temporarily switch to the client's job user profile without starting a new job.

Using a 5250 screen, you need do run the **Display Job** (DSPJOB) CL command for each individual job to find out which user profile it is currently serving. This a major advantage of the RSE, since there is no single CL command to list all jobs serving a certain user profile.

The jobs shown in the iSeries Jobs subsystem are grouped by job subsystem and if a job is not associated with a subsystem it appears under “Other jobs”.

**Working with iSeries jobs**

To list the jobs on your iSeries host, expand the job filter. Individual job entries in the tree view are identified by their **job number**, **user**, and **job name**. After you expand the filter, you can perform such tasks as:

- **End jobs**
- **Hold jobs**
- **Release jobs**
- **Add jobs to job status view**
- **Display job logs for active jobs**

These actions are accessible through a pop-up menu when you right-click a job entry. See Figure 5-14.
Properties or attributes pertaining to a particular job are displayed in the Properties view. See Figure 5-15.

Tip: After you complete any tasks within the iSeries jobs subsystem, we recommend that you refresh the window. Right-click the iSeries jobs subsystem, the job filter, or the actual job itself. Then select Refresh (you can also click the Refresh button on the RSE toolbar). This ensures that updated information is displayed.
The Properties view only displays a minimal amount of information related to the job. You can find more details on a particular job by right-clicking the job and selecting Properties.

**Creating your own job filters**

In addition to the predefined job filters, you can create your own. This section explains how to do this by creating a filter that displays all active administration jobs that run in the IBM HTTP Server subsystem.

To create a job filter, complete the following steps:

1. Right-click the *iSeries jobs* subsystem.
2. Select **New → Job Filter**.
3. The New - Job Filter window (Figure 5-16) opens.

![Figure 5-16 The New - Job Filter window for creating job filters](image)

4. Enter values for the new job filter. At this point, you may choose to create a job filter using different values depending on what jobs you have on your system:
   a. In the Job name field, enter `ADMIN`.
   b. In the Job user field, enter `QTMHHTTP`.
   c. Deselect the **Job queue** and **Output queue** boxes since we are only interested in displaying active jobs.
   d. Click **Next**.
5. Another New - Job Filter window (Figure 5-17) for the job filter name opens. Follow these steps:
   a. In the Filter name field, enter the name for the job filter. In this example, we enter `HTTP Admin Jobs`.
   b. Click the **Owner profile** drop-down list and select a profile to own the filter.
   c. Click **Finish**.
The newly created job filter displays as a child node to the iSeries Jobs subsystem. You can now expand the filter to display the jobs. See Figure 5-18.

Job statuses are also indicated visually with icons to the left of the job entry. The icons are explained as follows:
- This icon means the job is active.
- This icon means the job is on the job queue.
- This icon means the job has ended.

**Working with job filters**
You can change, rename, copy, move, and delete job filters. You can also move them up or down within the list under the subsystem. To execute these tasks, right-click the job filter and select the appropriate action from the pop-up menu.

### 5.3.4 IFS (Integrated File System) files subsystem
Access to the iSeries IFS is supported via this subsystem. The IFS provides a common interface to the different file systems that are available on your iSeries host. Expanding the subsystem reveals three predefined filters (see Figure 5-19):
- **File systems**: Contains all of the file systems that are available on the iSeries server.
- **Root file system**: Displays all folders and files available in this file system. This is the most used file system of all those available besides the QSYS “traditional” library-based file system.

- **Home**: This is a filter that displays folders and files under the /home folder.

![Figure 5-19  The expanded IFS Files subsystem](image)

**Creating your own IFS filters**

Besides the predefined IFS filters, you can also create your own. We demonstrate this process by creating a filter that displays all files and folders in the path /QIBM/UserData/mqm.

To create your own IFS Files filter, follow these steps:

1. Right-click the **IFS Files** subsystem.
2. Select **New → Filter**. The New - File Filter window (Figure 5-20) opens.
3. Enter values for the new file filter. At this point, you may choose to create a file filter using different values depending on what is available in your IFS:
   a. In the **Folder** field, enter the path name for the filter to act upon. The easiest way to do this is to click the **Browse** button and select your path from there.
   b. Select your subsetting options. You can subset by file name or file type. Select the appropriate radio button representing these options and then enter your filter information directly. In the case of file types, click the **Select** button to display a list of file types for your selection. You can select more than one file type.
   c. Select the **Show files only** check box if you don’t want folders to appear in your filter. For this demonstration, we want this filter to display both folders and files, so we do not select this box.
   d. Click **Next**.
4. On the next New - File Filter window (Figure 5-21), in the Filter name field, enter the name for the file filter. In this example, we enter WebSphere MQ Files. Click the **Owner profile** drop-down list and select a profile to own the filter. Click **Finish**.

The newly created IFS file filter displays as a child node to the IFS Files subsystem. You can now expand the filter to display the folders and files specified by this filter. See Figure 5-22.
Working with folders and files
You can create, rename, copy, move, and delete folders and files. These tasks are accessible by right-clicking the folder or file and then selecting the appropriate action from the pop-up menu.

Opening files
To open files with an associated editor within the RSE perspective, you can choose one of the following options:

- Double-click the file. This opens the file using the associated editor. For example, if you double-click a Java file, then it opens the file with the Java editor.
- Right-click the file and select Open With. From the submenu, select the editor you want to open the file with.

Searching for folders and files
You can search for files (and folders) in a particular folder. To do this, follow these steps:

1. Right-click the folder. In this example, we search for “*.ini” files in the qmgrs folder that appears under the WebSphere MQ Files filter created in the “Creating your own IFS filters” on page 93.
2. Select Search. The Search window (Figure 5-23) opens.
3. Click the Remote Search tab. Enter your search pattern in the File name patterns field. You can also search for a certain string inside the file in the Search string field. Select the Search subfolders check box. Click the OK button.
The result of the search appears in the Remote Search view (Figure 5-24) as part of a tabbed notebook of views.

Working with file filters
You can change, rename, copy, move, and delete IFS file filters. You can also move them up or down within the list under the subsystem. Simply right-click the job filter and then select the appropriate action from the menu that appears.

Launching command shells
From the IFS Files subsystem, you can launch a command shell to the OS/400 Qshell environment. To launch a command shell, follow these steps:

1. In the Remote Systems view, right-click IFS Files.
2. Select Launch Shell.
3. The Remote Commands view (Figure 5-25) opens in a tabbed notebook of views. Enter your commands in the Command field and press Enter. The output of the command displays in the output area above the command entry field.
Figure 5-25 Remote Commands view

You can launch additional shells within the Remote Commands view. To launch additional command shells, click the Menu ( ) button on the Remote Commands view toolbar. Then select Launch Shell → your connection. The new command shell appears as an additional tab near the top of the Remote Commands view.

To terminate the selected command shell, click the Terminate the selected running shell ( ) button in the Remote Commands view toolbar.

To terminate and remove the command shell from the view, click the Terminate and remove the selected shell button in the Remote Commands view toolbar.

5.4 Managing your iSeries objects

This section discusses the use of filters and the different types of objects that you can create in the iSeries Objects subsystem within the RSE.

5.4.1 Filters in the iSeries Objects subsystem

The use of filters allows you to easily view and manage subsets of your iSeries objects. After you create a filter, you can save them under a unique name and reuse the filter. Four types of filters are available in the iSeries Objects subsystem as explained in the following sections.

Library filters
Library filters list a set of libraries from your iSeries host in the Remote Systems view.

To create a library filter, follow these steps:
1. In the Remote Systems view, expand your connection and then expand iSeries Objects.
2. Expand Work with libraries. You can also right-click iSeries Objects and select New → Library filter. Expanding Work with libraries corresponds with using the WRKLIBPDM OS/400 command.
3. The New - Library Filter window opens (Figure 5-26). Select a generic or specific library name from the Library drop-down list. Or, enter LIB*, for example, to list all libraries starting with "LIB". You can also browse to locate libraries by clicking **Browse**.

4. Click **Next**.

![New Library Filter window](image)

**Figure 5-26** New Library Filter window

5. On the next window, enter a name for your filter in the Filter name field. Select the profile you want to use. Use an individually created profile to keep your work private or select the team profile to share with others. Note that the option to select a profile only appears if you chose to create a new library filter from the iSeries Objects subsystem node. See step 2 on page 97.

6. Click **Finish**.

Your new library filter displays as a child node to the iSeries Objects subsystem. Expanding the newly created filter displays the libraries specified by this filter. Further expanding any library under the filter displays the objects in that library. See Figure 5-27.
Figure 5-27   Expanding the newly created library filter

Library list filters
Library list filters display the *LIBL library list from your iSeries host in the Remote Systems view. When expanded, it supplies the same actions as a library filter, with additional actions for manipulating the library list. If you are using your default profile, the wizard tells you that you already have a library list. Therefore, this action only applies if you are using a team profile or an additional personal profile.

To create a library list filter, follow these steps:

1. In the Remote Systems view, expand your connection and then expand iSeries Objects.
2. Right-click iSeries Objects and select New → Library List Filter. This action corresponds with the Edit Library List (EDTLIBL) OS/400 command.
3. The New - Library List Filter window opens. Enter a name for the filter in the Filter name field and switch the owner profile if necessary. Click Finish.

Your new library list filter displays as a child node to iSeries Objects subsystem.

Object filters
Object filters list a set of objects from your iSeries host in the Remote Systems view.

To create an object filter, follow these steps:

1. In the Remote Systems view, expand your connection and then expand iSeries Objects.
2. Expand Work with objects. You can also right-click iSeries Objects and select New → Object Filter. Expanding Work with objects corresponds with the Work with Objects Using PDM (WRKOBJPDM) command. The New - Object Filter window opens. For this demonstration, we create an object filter that displays all program objects in the library LIB6961.
3. In the New - Object Filter (Figure 5-28), complete these tasks:
a. Browse, specify, or select from the Library drop-down list the library that contains your objects.

b. Enter a generic or specific object in the Object field, or click **Browse** to select. The default is * (asterisk), and you can use up to ten characters.

c. To define **object type**, browse, specify, or select from the drop-down list, or click **More Types>>** to select from all known object types.

d. For **Object attribute**, browse, specify, or select from the drop-down list. Note that you can only enter a single **object type and attribute pair per filter**, until you change a filter to add more filter strings.

e. Click **Next**.

![New Object Filter window](image)

> **Figure 5-28**  **New Object Filter window**

4. On the next window, enter a name for your filter in the **Filter name** field. Select the profile you want to use. Use an individually created profile to keep your work private or select the Team profile to share with others. Note that the option to select a profile only appears if you chose to create a new object filter from the iSeries Objects subsystem node. See step 2 in this sequence. Click **Finish**.

Your new object filter displays as a child node to the iSeries Objects subsystem. Expanding the newly created filter displays the objects specified by the filter. In this case, it displays all program objects in library LIB6961. The object type and attribute also appear next to the object name in the list. See **Figure 5-29**.
Member filters

Member filters list a set of source and data members from your iSeries host in the Remote Systems view.

To create a member filter, follow these steps:

1. In the Remote Systems view, expand your connection and then expand **iSeries Objects**.

2. Expand **Work with members**. You can also right-click **iSeries Objects** and select **New → Member filter**. Expanding Work with members corresponds with the WRKMBRPDM OS/400 command. The New - Member Filter window opens. For this example, we create a member filter that displays all members in the RPGAPP source file in library LIB6961.

3. In the New - Member Filter window (Figure 5-30), complete these tasks:
   a. Browse, specify, or select from the Library drop-down list the library that contains your source members.
   b. Browse, specify, or select from the File drop-down list the file within the library that contains the appropriate source members.
   c. Enter a generic or specific member name in the Member field. The default is * (asterisk), and you can use up to 10 characters.
   d. To select a specific member type, enter the type in the Member type field, or click **More Types** to select from all known object types. This step is optional.
   e. Select the **Source members** and **Data members** check boxes accordingly to indicate whether you want to list source members, data members or both. You must select at least one. These options are only available if you specify a generic file name in the **File** field.
   f. Click **Next**.
4. On the next window, enter a name for your filter in the Filter name field. Select the profile you want to use. Use an individually created profile to keep your work private or select the Team profile to share with others. Note that the option to select a profile only appears if you chose to create a new member filter from the iSeries Objects subsystem node. See step 2 on page 101. Click Finish.

Your new member filter displays as a child node to the iSeries Objects subsystem. Expanding the newly created filter displays all the members specified by the filter. In this case, it displays all the source members in the RPGAPP source file located in library LIB6961. The member type appears next to the member name in the list. See Figure 5-31.
5.4.2 Working with iSeries Objects filters

You can change, rename, copy, move, or delete iSeries Objects filters. You can also move them up or down within the list under the subsystem. To perform these tasks, right-click the object filter and then select the appropriate action from the pop-up menu.

You *change* a filter when you want to alter the iSeries host items it displays in the Remote Systems view. When you change a filter, you can also add extra filter strings to the filter.

**Filter pools**

If you have been using the workbench for some time, it is quite possible that your workspace may contain too many filters to navigate easily. In this case, you can group filters into *filter pools*. You can group filters by project, release, connection, task, etc. For example, a filter pool may contain a combination of two library filters, one object filter, and six member filters. Or it might contain a series of object filters and nothing else.

Also filter pools allow you to show filters under only certain connections. For example, you can create a filter pool called “Human Resources application” and add related filters to this filter pool. Then you can choose to only show this filter pool under a specific iSeries connection.

*Note:* Filters exist in all connections by default.

To use filter pools, follow these steps:

1. Click the **Menu** button on the toolbar of the Remote Systems view, and select **Show Filter Pools**.

2. Expand **iSeries Objects**. Now you should see your filters listed under *Profile name Filter Pool*. See Figure 5-32.

![Figure 5-32 Showing filter pools](image)

3. Right-click **iSeries Objects** and select **New → Filter Pool**. From here, you can create a new filter pool under the iSeries Objects subsystem.

4. Right-click **Profile name Filter Pool** and select **New**. From here, you can create a new Library List, Library, Object, or Member filters within a specific filter pool.
5. To switch back to regular filters, click the Menu (▲) button on the toolbar for the Remote Systems view, and deselect Show Filter Pools.

The ability to use filter pools is available for all four subsystems. You can copy filters or move them from one filter pool to another.

You can also copy or move filter pools from one profile to another within the subsystem they reside.

5.4.3 Creating objects in the iSeries Objects subsystem

This section describes the built-in wizards within the RSE that enable the creation of the following iSeries objects:

- Data areas
- Data queues
- Message files
- Source physical files
- Members (not objects; for the purposes of this discussion, the term “object” will suffice)

The following sections explain how to create each object type.

Creating data areas

In the RSE, you can create data areas in libraries on your iSeries host. This function corresponds to the Create Data Area (CRTDTAARA) OS/400 command.

To create a data area, follow these steps:

1. In the Remote Systems view, expand your connection. Then expand iSeries Objects → Library list.

2. Right-click the library in which you want to create the data area. You can select any library within your subsystem that is part of a user created filter as well. We chose a library from the predefined library list filter. Select New → Data Area. The New - iSeries Data Area window (Figure 5-33) opens.

3. In the New - iSeries Data Area window, complete the following tasks:
   a. Enter the name of the data area you want to create in the Data area field.
   b. Select the type of data area you want to create using the radio buttons in the Type section of the window.
   c. Optionally add a description for the data area in the Text field.
4. Optionally click the Next button to specify further options for your data area, such as authority, length information, or the initial value.

5. Click Finish. The new data area displays as a child node to the library in which it was created in.

Basic information about the data area is displayed in the Properties view which is generally located underneath the Remote Systems view. Further information is available by right-clicking the data area and selecting Properties(B).

To change the value contained in the data area, right-click the data area and select Change. Enter the new value in the New value text box and specify any substring requirements. Click OK.

Creating data queues

In the RSE, you can create data queues in libraries on your iSeries host. This function corresponds to the Create Data Queue (CRTDTAQ) OS/400 command.

To create a data queue, follow these steps:

1. In the Remote Systems view, expand your connection. Then expand iSeries Objects → Library list.

2. Right-click the library in which you want to create the data queue. You can select any library within your subsystem that is also part of a user-created filter. We chose to use a library from the predefined library list filter. Select New → Data Queue. The New - iSeries Data Queue window (Figure 5-34) opens.

3. In the New - iSeries Data Queue window, complete the following tasks:
   a. Enter the name of the data queue you want to create in the Data queue field.
b. Alter the size of the data queue entry length in the Maximum entry length field if necessary.

c. Optionally add a description for the data queue in the Text field.

4. Optionally click the **Next** button to specify further options for your data queue, such as authority, force to auxiliary storage, sequence, queue size, and auto-reclaim.

5. Click **Finish**. The new data queue displays as a child node to the library in which it was created in.

Basic information about the data queue is displayed in the Properties view that is generally located underneath the Remote Systems view. Further information is available by right-clicking the data queue and selecting **Properties**.

---

### Creating message files

In the Remote System Explorer, you can create message files in libraries on your iSeries host. This function corresponds to the Create Message Files (CRTMSGF) OS/400 command.

To create a message file, follow these steps:

1. In the Remote Systems view, expand your connection. Then expand **iSeries Objects** → **Library list**.

2. Right-click the library in which you want to create the message file. You can select any library within your subsystem that is also part of a user-created filter. We chose to use a library from the predefined library list filter. Select **New** → **Message File**. The New - iSeries Message File window (Figure 5-35) opens.

3. In the New - iSeries Message File window, complete the following tasks:
a. Enter the name of the message file you want to create in the Message file field.
b. Optionally add a description for the message file in the Text field.

![Creating a message file](image)

Figure 5-35 Creating a message file

4. Optionally click the Next button to specify further options for your message file, such as authority, coded character set ID, or file size.

5. Click Finish. The new message file displays as a child node to the library in which it was created in.

**Working with message descriptions**

To *add* a message description to a message file, complete the following steps:

1. Right-click the message file and select **New → Message Description**. The Add Message Description window opens.
2. Enter details for the new message description.
3. Select the **Advanced(L)** check box to specify any advanced options.
4. Click the **OK** button to create the message description.

To display message descriptions in a message file, you can do either of the following tasks:

- Double-click the message file.
- Expand the message file.

The messages are displayed as child nodes to the message file in the Remote Systems view.

To *change* or *delete* a message description, right-click the message description and select the appropriate action from the pop-up menu.

**Creating source physical files**

In the RSE, you can create source physical files in libraries on your iSeries host. This function corresponds to the Create Source Physical Files (CRTSRCPF) OS/400 command.

To create a source physical file, follow these steps:
1. In the Remote Systems view, expand your connection. Then expand **iSeries Objects → Library list**.

2. Right-click the library in which you want to create the source physical file. You can select any library within your subsystem that is part of a user created filter as well. We chose to use a library from the predefined library list filter. Select **New → Source Physical File**. The New - iSeries Source Physical File window (Figure 5-36) opens.

3. In the New - iSeries Source Physical File window, complete the following tasks:
   a. In the File field, enter the name of the source physical file you want to create.
   b. In the Record length field, enter the record length for the source physical file.
   c. Optionally, in the Text field, add a description for the source physical file.

![Figure 5-36 Creating a source physical file](image)

4. Optionally click the **Next** button to specify further options for your source physical file, such as authority, coded character set id, or IGC data.

5. Click **Finish**. The new source physical file displays as child node to the library in which it was created.

**Note:** You can also create source physical files from the iSeries Commands subsystem. To do this, you follow these steps:

1. In the Remote Systems view, expand your connection. Then expand **iSeries Commands → Create source file**.

2. Specify or select parameters for the source file. Select the **Advanced** check box for extra parameters.

3. Click **OK**.

**Creating members**

In the RSE, you can create members in source physical files on your iSeries host. This function corresponds to the **ADDPFM OS/400 command**.
To create a member, follow these steps:

1. From the Remote Systems view, expand your connection. Then expand **iSeries Objects → Library list**.

2. Select and expand the library and source physical file where you want to create a member. You can select any library within your subsystem that is part of a user created filter as well. We chose to use a library from the predefined library list filter.

3. Right-click the source physical file and select **New → Member**. The New - iSeries Source Member window (Figure 5-37) opens.

4. In the New - iSeries Source Member window, complete the following tasks:
   a. In the Member field, enter the name of the member you want to create.
   b. In the Member type field, enter the member type, or select from the available items in the list.
   c. Optionally, in the Text field, add a description for the member.
   d. Click **Finish**.

![Figure 5-37 Creating a member](image)

The new member displays as a child node to the selected source physical file. The new member (empty) also opens in the default editor.
5.4.4 Searching operations

You can search for text strings on your iSeries server from the Remote System Explorer perspective.

**Search for text strings on the iSeries server**

This action corresponds to the *Find String Using PDM* (FNDSTRPDM) command in PDM, which would search for strings in one or more member of a source file. In addition to that, the search option in the Remote System Explorer can search multiple libraries and filters as well.

There are two ways to search:

- From a selection in RSE
- Or from the central search dialog

When you search from a selection, you navigate through your libraries and files until you find the item you want to search.

When you search from the central search dialog, you do not begin with any selection or containing item. Your search is as broad as you want it to be, and you can narrow the search by specifying certain libraries, files, or member names to search.

**Tip:** Before you do a Search operation in the source editor, save the file you are searching. The search function works from the most recently saved version of the file rather than from the contents that you see in the editor area. You do not need to save your file before you do a Find/Replace operation.

**To search for strings from a selection**

1. In the Remote Systems view, expand your iSeries connection until you reach the item that you want to search. Or, navigate through items displayed in the iSeries Table view.
2. Select the containing item by clicking on the item.
3. Right-click and select *Find string*.
4. In the *Find String* window, specify the string that you want to search in the Search string field. Select the *Case sensitive* check box if applicable.
5. The Scope area displays the targets that you have selected as shown in Figure 5-38.

---

**Note:** You can also create members from the iSeries Commands subsystem. To do this:

1. In the Remote Systems view, expand your connection. Then expand *iSeries Commands* → *Create source member*.
2. Specify or select parameters for the source member. Select the **Advanced** check box for extra parameters.
3. Click **OK**.
6. If applicable, specify columns to search with the Columns radio buttons. By default, **All columns** are searched.

7. Click **OK**. The matching strings display in the Remote Search view as shown in Figure 5-39.
To search for strings from the entire iSeries server

1. Click Search → iSeries from the workbench menu.
2. In the iSeries Search tab of the Search window, specify the string that you want to search in the Search string field.
3. Select the Case sensitive check box if applicable.
4. Use the Target area to specify the target that you want to search. The names can be generic.
5. If the file name is generic, two check boxes beneath this area are enabled: **Source members** and **Data members**. Source members is checked by default. Check Data members if you want you also want to search data members. Searching data members takes a longer amount of time compared to searching source members, which is why it is unchecked by default as shown in Figure 5-40.

---

Figure 5-39  Remote Search window showing the search results
6. If applicable, specify columns to search with the Columns radio buttons. By default, **All columns** are searched.

7. Click **OK**. The matching strings display in the Remote Search view.

**Tip:** The string searched for can be character or hexadecimal value.

### 5.4.5 Working with save files

Most i5/OS and OS/400 object types can be saved not only directly onto a tape or CD ROM but also into another object type called save file (object type *FILE with attribute SAVF). As a common practice, these save files can also be transferred to and stored on other platforms such as a Windows or Linux workstation. This format also allows you make OS/400 objects available for download from Web sites, for example as for redbook samples as described in “Restoring the LIB6961 library” on page 681.

Before WDScl 5.1.2 this was mostly done be sending the SAVF to the workstation by means of the File Transfer Protocol (FTP) or mapping the iSeries storage as a network drive to your workstation and using drag and drop in a file explorer or a copy command to transfer the saved objects to a binary file.

RSE allows you now to exchange OS/400 save files in a much easier way. The key capability of RSE to enable this, is the fact that you can copy and paste between different connections within your RSE workbench. That is, you may copy a save file from an OS/400 library into a directory on your local workstation (using the default “local” connection for example).
Download a save file from OS/400 to your PC
So to download a save file from you iSeries to your PC simply right-click an object with the extension .*file.savf and select Copy from the context menu (or select the object and press Ctrl-c). Then right-click the target directory in a local system connection and select Paste from the context menu (or select the object and press Ctrl-v).

Upload a save file from your PC to OS/400
Vice versa, you can upload the save file from you PC to an OS/400 library by copying it from local system connection to an iSeries connection.

Restoring objects or a complete library directly from a PC to OS/400
The biggest advantage appears, if you want to restore the content of a save file residing on your workstation. With RSE, you can do so in a single step! Just right-click any local file with extension .savf or .sav and select Restore to iSeries from the context menu as shown in Figure 5-41.

Note: As of WDSc 5.1.2.4, the extension of the file containing saved objects must be .sav or .savf in lower case letters. For all other file extensions, including .SAV or .SAVF (in uppercase), the Restore to iSeries item does not appear in the context menu. However, you can easily rename the file from the same context menu. Keep in mind though, that you cannot rename from LIB6961.SAVF to LIB6961.savf (which would result in error message “RSEG1009E You must rename LIB6961.SAVF”), but you can rename from LIB6961.SAVF to LIB6961.sav or from LIB6961.SAV to LIB6961.savf.
You see the Restore iSeries Save File dialog as shown in Figure 5-42, which gives you the choices to:

- Restore objects into an existing library or restore a complete library (if the Save Library (SAVLIB) command was used to fill the save file.
- To specify the name of a library and save file temporary store the save file for uploading.
- Specify the name of the library where the objects should be restored to. The default is *SAVLIB, but you can enter any other library name as shown by the mouse pointer in Figure 5-42.
- Select to see a prompt for the restore command (RST, RSTOBJ, or RSTLIB) in case you want to override any of the default parameters.

During the upload and restore process you can see the progress the in the bar at the bottom of the dialog similar to Figure 5-43.
Figure 5-43  Uploading save file

Note, if you selected to be prompted for the restore command, the prompt, similar to the one shown in Figure 5-44, appears after the files were actually loaded onto the iSeries. You may select the Advanced(1) check box to see more parameters or the All Parameters(2) to see all parameters.
After the restore process completed or ended abnormally, you can see the informational or error message from the corresponding job log in the *iSeries Commands Log* (by default in the lower left pane of the RSE perspective) as shown in Figure 5-45.
Working with zipped files
Also as a common practice, you may compress a save file into a .zip file. The Remote Systems view of the RSE perspective allows you to expand a .zip file and see which files or directory are stored in there. However, you cannot directly restore a save file to the iSeries without first extracting its content, although the context menu for save files within a ZIP file does show Restore to iSeries in WDSc 5.1.2.4.

5.5 Working with your library list
This section explains the actions that you can perform on your library list in the RSE perspective.

Important: When you perform the following actions with your library list, the changes only apply to the current session in the current communications server job.

5.5.1 Adding a library list entry
To add a library to your library list, follow these steps:
1. In the Remote Systems view, expand iSeries Objects.
2. Right-click the Library list filter and select Add Library List Entry. The Add Library List Entry window (Figure 5-46) opens.
3. In the Add Library List Entry window, complete the following tasks:
   a. In the Additional library field, enter a library name, or click Browse to navigate to a library.
   b. Select one of the radio buttons that specifies where to position the library relative to a reference library.
   c. From the Reference library drop-down list, select a reference library if applicable.
   d. Click OK.
5.5.2 Removing a library from the library list

To remove a library from your library list, follow these steps:

1. In the Remote Systems view, expand **iSeries Objects → Library list**.
2. Right-click the user library you want to remove and select **Remove from library list**.

**Note:** You can only remove a library that belongs to the **usr (user) section** of your library list.

5.5.3 Changing the current library

To change the current library, follow these steps:

1. In the Remote Systems view, expand **iSeries Objects**.
2. Right-click the **Library list** filter and select **Change current library**. The Change Current Library window (Figure 5-47) opens.
3. In the Current library field, enter a new library name, or click **Browse** to select a library. Then click **OK**.

**Tip:** An alternative method for adding a library list entry is to right-click a library in a predefined or user created filter and then select **Add To Library List**.
5.5.4 Moving libraries up or down in the library list

To move a library up or down in the library list, follow these steps:
1. In the Remote Systems view, expand iSeries Objects → Library list.
2. Right-click the user library you want to move. Select Move up in library list or Move down in library list.

5.5.5 Moving libraries within the library list

To move libraries within the library list, follow these steps:
1. In the Remote Systems view, expand iSeries Objects → Library list.
2. Right-click the user library you want to move and select Move Within Library List. The Move Library List Entry window (Figure 5-48) opens.
3. In the Move Library List Entry window, complete these tasks:
   a. From the Reference library drop-down list, select a reference library.
   b. Select one of the radio buttons that specifies where to position the library relative to the reference library.
   c. Click OK.

![Figure 5-48 Moving a library within a library list](image)

5.5.6 Specifying initial libraries and commands

In the RSE, you can specify libraries to be added to the initial library list, specify the current library at connect time, and specify to run a command every time you connect.

To specify initial libraries and commands, complete these steps:
1. In the Remote Systems view, expand your connection name.
2. Right-click any subsystem node and select Properties.
3. The Properties window (Figure 5-49) for the subsystem opens. Complete the following tasks:
   a. In the left side pane, click Initial Library List.
   b. Enter the name of the library to be added to the library list in the Library field. Click Add(B) to add the library into the list below. Do this for all libraries that you want to add to the library list.
c. After you added one or more libraries to the list and select one of them, the Remove button becomes active and allows you to remove the libraries from the list.

d. After you added more than one library, and select one of them, the Move Up and Move down buttons become active and allow you to position the libraries within the list.

e. In the Current library field, optionally specify or select from the drop-down list the current library (optional). The default is *USRPRF.

f. In the Initial command field, optionally enter a command to be executed every time the connection is made.

g. Click OK.

Figure 5-49 Specifying initial libraries and commands

These library list changes take place every time you connect the RSE connection. In addition to whatever is specified in the job description for the user profile that signs on. If you are frequently working with different environments, for example maintaining multiple applications, each having its own distinct library list, you can create multiple connections pointing to the same iSeries but using a different library list.

5.6 iSeries table views

iSeries table views are an alternative for displaying information within the RSE. Table views take the currently selected object in the iSeries Objects, iSeries Jobs, or IFS Files subsystems as input and display the contents in a table. With this view, you can see item properties at the same time since they are displayed as rows in a table.

You can perform the following tasks with the iSeries table view:

- Navigation
- Command prompting and running
- Direct editing of table entries
- Object and source subsetting and positioning
- View locking and unlocking
With the table view, you can list and sort libraries, objects, and members. You can also change, copy, rename, delete, edit, compile and run items in the view from the pop-up menu (by right-clicking the item). The iSeries table view is similar in functionality to PDM.

**Difference between iSeries Table and Remote Systems Details view**

There are two very similar but yet different type of views:

- The iSeries Table View
- The Remote Systems Details view

The *Remote Systems Details* view automatically appears in the lower right pane like the one shown in Figure 5-50 whenever you open the Remote System Explorer perspective or reset it to its original settings (*Window → Reset Perspective*).

![Figure 5-50 Remote Systems Details view](image)

By double-clicking a row in the Remote Systems Details view you can drill down to the subsystems, from there to the filters, objects and members. However, it does not only contain OS/400 objects, but also everything else, such as jobs, IFS files and IFS directories. The Remote Systems Details view also shows files of non-iSeries connections, for example the files on your local workstation.

The *iSeries Table View* is also located in the lower right pane, but does not contain any contents, unless you select objects to be displayed as described in “Populating iSeries table views” on page 122. It is specifically for displaying and working with objects and members from the QSYS file system and is designed for easy migration to the RSE for PDM users and support PDM actions and a command entry area, which the Remote Systems Details view does not.

The iSeries table view and the Remote Systems view share the same windows for the same functions. These functions are also synchronized. For example, if you rename an item in the table view, the change is reflected in the Remote Systems view.

The iSeries table view, by default, displays as part of a tabbed notebook together with other RSE views.

### 5.6.1 Populating iSeries table views

Before you can use any of the functions provided by table views, you must populate the table. To populate a table view from the Remote Systems view, right-click an appropriate item in the Remote Systems view and select **Show in Table**.
The table view displays as part of tabbed notebook of views. For this example, we populate a table view with items from a library called LIB6961. See Figure 5-51.

![Figure 5-51 Populating an iSeries table view](image)

Column titles in the table view change depending on the type of items being displayed. To view more information in the table view, click the iSeries Table Menu ( ) button on the toolbar and select Preferences.
In the right pane of the Preferences Table View dialog as shown in Figure 5-52, you may add or remove columns to be displayed in a Member or Object tables.

5.6.2 Working with items in a table view

Here are some tips for working with items in the table view.

- Click any of the column titles to sort by that attribute.
- To print the list of items in the view, click the Menu (ighbors) button on the toolbar and select Print.
- You can navigate through items in the view by double-clicking items that contain other items. For example, you can double-click source files to repopulate the view with the members in the source file. You can double-click IFS folders to display other folder and files. You can even double-click message files to display message descriptions contained within the message file. When you access editable items, such as a member or Windows file, you can double-click the item to open it in the default editor for that type of item. During the course of your session, if you populate table views, you can revisit each view in history by using the “forward” and “back” buttons on the table view toolbar. See Figure 5-53.

Figure 5-52  Table View Preferences

Figure 5-53  Table view navigation buttons
You can also navigate through your table views from history. Click the **Menu (Menu)** button on the table view toolbar and select **Work with**.

The view supports *Windows keyboard shortcuts*. For example, hold down the *Ctrl* key and click to select multiple items in the view. From here, you can then perform actions on the selected items.

To access PDM-like options, select the item or items in the table view. Then, right-click and select **PDM Options**. Select the option from the PDM submenu. Most PDM options from this menu are duplicates of options that can be selected from the parent pop-up menu, such as change, rename, and copy. The PDM Options submenu is included for consistency and to provide a sense of familiarity for PDM users.

### Subsetting objects and members in the table view

When dealing with long lists of items, it is generally helpful to subset and reorganize the items in the view.

To specify subsetted lists for objects or members, follow these steps:

1. In the populated iSeries table view, click the **Menu (Menu)** button from the table view toolbar and select **Subset**. Different Subset windows open depending on the types of items displayed in the table view. For members, see Figure 5-54. For objects, see Figure 5-55.

![Subset window for members](image)

**Figure 5-54** Subset window for members

![Subset window for objects](image)

**Figure 5-55** Subset window for objects

2. Subset your information by Name, Type, Attribute, or Text.
3. Click **OK**. The table displays the information specified in the Subset window.

After you subset a list, the title bar changes to reflect the new set of items. To revert back to the original display of all items in the table, click the **Menu (Menu)** button on the table view toolbar and select **Show All**.
Positioning objects and members in the table view
The positioning of items in the iSeries table view is similar to the “position to” and “position to type” functions in PDM.

To position to items in the table view, follow these steps:
1. In the populated iSeries table view, click the Menu ( ) button on the table view toolbar and select Position to. The Position To window (Figure 5-56) opens.
2. In the Position To window, complete these tasks:
   a. In the Name field, click the drop-down list and select *TOP or *BOTTOM. Or enter the name (or partial name) of the item that you want to position to.
   b. In the Type field (only for objects), leave the default as asterisk, or enter the type. If you do not specify the type, then the first type that is displayed or visible at the top of the view is used.
   c. Click OK.

![Figure 5-56 Position To window](image)

**Tip:** You can also invoke the Position To function by entering the first letter of the name that you want to position to in the table. In addition, if you press Shift+first letter repeatedly, the wizard cycles through all the names that start with that letter.

Direct editing of table view items
In the iSeries table view, you can directly edit properties of members and objects. This function corresponds to direct editing in PDM.

You can edit directly the following item properties:
- For objects, you can edit the Name and Text properties.
- For members, you can edit the Name, Type, and Text properties.

To directly edit item properties, follow these steps:
1. Click the row in the iSeries table view that contains the property you want to edit.
2. In the selected row, click the cell you want to edit.
3. Make the change directly in the cell and press Enter. If you press the Escape key instead of the Enter key while inside the cell, no change takes place. You may need to refresh (File → Refresh) the screen for the changes to be updated on the whole workbench views or simply unlock the table view.

Locking and unlocking table views
The iSeries table view toolbar contains a lock button that controls the correlation between the Remote Systems view and the iSeries Table tree view. See Figure 5-57.
If the lock is **disabled**, then whenever you click an object or library in the Remote Systems view, the associated contents of that item automatically populate the iSeries table view.

If the lock is **enabled** (the default state), then clicking various items in the Remote Systems view does *not* change the input to the iSeries table view.

To enable or disable the lock, click the button once to change its state.

### 5.7 User actions

User actions are host commands that you define to run against iSeries libraries, objects, jobs, and members. They can also be defined for folders and files in any remote UNIX, Windows, Linux, local, or IFS system.

Each user action consists of:

- A unique name.
- A command (that may have substitution variables): The user action specifies in which mode the command is run (normal, batch, or interactive). For more information about running modes, see “Creating your own command set” on page 85 and 6.3.2, “Running modes” on page 176.
- Substitution variables that are replaced with contextual information when the command is run.
- An optional comment used to describe the action in more detail.
- Various options that affect what happens when the command is run, for example, whether to prompt the command first.
- One or more file types to scope the resource types that apply to the action. For example, a command to start Screen Design Aid (SDA) would be scoped to members of type DSPF. The action is only shown for remote members that are of type DSPF.

#### 5.7.1 Creating user actions

User actions can be created from either the Remote Systems view or from the iSeries table view. In this section, we use the Remote Systems view. In the following procedure, we create a user action that adds service programs to a binding directory.

To create a user action, complete the following steps:

1. In the Remote Systems view, expand the **iSeries Objects** subsystem. Expand any filters if required.
2. Right-click any library, object, or member and select **User Actions → Work With User Actions**. The Work With User Actions window as shown in Figure 5-58 opens.
3. In the Work With User Action window, complete the following tasks:
   a. In the left side pane of the window, click **Object action** under the New node. Entry fields for specifying the user action appear on the right side pane of the window. If you are creating a user action that is to be applied to members, you click **Member action** at this point.
   b. In the Action name field, enter a name for the user action.
   c. Optionally, in the Comment field, enter a comment that may further describe the purpose of the action.
   d. From the Command drop-down list, select the execution environment in which this user action will run.
   e. In the text box underneath the Command text label, enter the command that this user action will run. At this point, you can also:
      - Click **Insert variable** to insert substitution variables into your command string.
      - Click **Browse** to display a list of available OS/400 commands. From here, you can select a command.
      - Click **Prompt** to prompt the command string entered in the text box. This prompt assists you in formatting the command string correctly. When prompting a
command, you can also click Help in the prompt window to receive further assistance on the command.

f. Select the appropriate boxes for your user action. These options affect what happens when the command is run. In this example, we selected Prompt first and Show action. With these selections, the command is prompted when the user action is run and the user action is displayed in the pop-up menu when applicable.

g. From the Defined Types list, select the object types to which this user action applies. Click the type in the list and then click Add to add the types to the Selected Types display area. In this example, we selected the ILE type which applies to *SRVPGM objects.

h. Click Create.

i. Click Close.

5.7.2 Invoking user actions

This section is based on the user action we created in 5.7.1, “Creating user actions” on page 127. To invoke a user action, follow these steps:

1. From the Remote Systems view (or an iSeries table view), right-click a service program object in the iSeries Objects subsystem and select User Actions. The user action displays in the next pop-up menu. See Figure 5-59.

2. Select the user action that appears. In this example, the user action displays as Add *SRVPGM to APP6961 BD. The command associated with this user action (ADDBNDDIRE) is prompted.

3. From the command prompt window, click OK to run the command.

4. Output messages from the command are displayed in the iSeries Commands Log view as part of the tabbed notebook of views.

5.7.3 Named types

In 5.7.1, “Creating user actions” on page 127, we specify object and member types to which the user action is scoped. In that section, we only chose from a list of pre-supplied types. However, we can also define our own named types.

Named types are collections of resource types that are either object or member types. Named types enable user actions to be scoped to one or more resource types.

Creating a named type

In 5.7.1, “Creating user actions” on page 127, we created a user action for the ADDBNDDIRE command. When creating this user action, a resource type of ILE was chosen. The user action created was intended to be used on service program objects only. However, the
predefined ILE resource type applies to modules, programs, and binding directories as well. In this section, we create a new named type that applies to service program objects only.

To create a new named type, follow these steps:

1. Right-click any of the RSE subsystems (except iSeries Commands) and select **Work With → Named Types**. The Work With Named Types window (Figure 5-60) opens.

2. In the left side pane of the window, there are three expandable nodes. They are:
   - **New**: Expanding this node reveals options for creating either **object types** or **member types**.
   - **Object**: Expanding this node reveals all the **named object types** defined. This includes predefined and user created named types. Selecting a named object type from the tree allows you to change details pertaining to the type. You can only edit user-defined named object types.
   - **Member**: Expanding this node reveals all the **named member types** defined. This includes predefined and user created named types. Selecting a named member type from the tree allows you to change details pertaining to the type. You can only edit user defined named member types.

   In the left side pane of the window, expand **New** and select **Object**.

3. Entry fields appear on the right side pane of the window. In the window, complete the following tasks:
   a. In the Name field, enter a unique name for the object type.
   b. In the Object type field, specify the object type. Click **Browse** to display a set of valid object types.
   c. In the Object attribute field, specify the object attribute. Click **Browse** to display a set of valid object attributes.
   d. Click **Add**. The object type/attribute pair appear in the Object Types display area.
   e. Repeat steps b through c as many times as required if you want to add more than one object type/attribute pair.
   f. Click **Create**.
   g. The Object node in the left side pane expands and displays the newly created named object type at the end of the list. This new type is now available as a resource when scoping user-defined actions.
5.8 Additional parameters with actions issued from a table view

In WDSc 5.1.2 you can specify additional parameters in the table view that will be used for action commands (user actions, PDM actions). This function exists in PDM, is heavily used there and was a frequent customer request.

In PDM (as well as many other “Work with” panels), before an option is executed, you can type parameters on the command line corresponding to options you typed in the list. These parameters are then added to the command when the option is performed. This function is now be available for actions initiated from the table view. The way it works is the same as in PDM, plus the ability to modify existing parameters.

This capability is very convenient, if you need to perform the same action on many objects or members. The following sections describe to sample scenarios.
Changing attributes of multiple file at once

Assume you need to change one or more attributes to the same value for more than one object. For example, you want to set the Maximum file wait time and the Maximum record wait time to 180 seconds for all physical and logical files in your library. You can do that easily from the RSE view by right-clicking each file and selecting Change(B) from the context menu.

However, doing so for more than two or three files, may become tedious and you might make mistakes by keying the wrong numbers. Using a table view can help here if you the following steps:

1. In the Remote Systems view, right-click the library and select Show in table from the context menu as shown in Figure 5-61. All objects of that library are shown now in the table view in the lower left pane of the workbench.

![Figure 5-61 Show objects in a table](image-url)
2. You can now either subset the objects as described in “Subsetting objects and members in the table view” on page 125 to see only physical and logical files or you can simply sort all objects by their type and attributes. To do the latter, simply click the header of the *Attribute* column and scroll down till you see the first logical (LF) or physical file (PF-DATA) as shown in Figure 5-62.

![Figure 5-62 Sort table view by object attribute](image)

3. Before you can change all files, you need to find out the names of the parameters to change. To do so right-click one of the files and selecting *Change...* from the context menu and the *Change Logical File* dialog appears.

![Figure 5-63 Change Logical File dialog](image)

4. In our example, we want to change the *Maximum file wait time* and the *Maximum record wait time* to 180 seconds. To see the name (as opposed to the description) of the parameters, select the Keywords check box as indicated by the mouse pointer in
Figure 5-63. The names of the parameters appear in the column between the parameter descriptions and the input field for the values of the parameters. As you can see in Figure 5-63, for this example the *Maximum file wait time* is called WAITFILE and the *Maximum record wait time* has the name WAITRCD.

5. Since there is no need to change only this particular file, click the **Cancel** button.

6. Back in the table view enter the parameters along with their values in parentheses into the command line as indicated by the mouse pointer in Figure 5-64.

![Figure 5-64 Specify additional parameters](image.png)

7. Now select all objects you want to change, right-click and **Change...** from the context menu.

Specifying additional parameters applies only to those actions that require executing a command, including user actions. The exception is the Paste and Copy actions which are not supported because they can be run across different views.

**Changing the owner of multiple objects**

Another good example to use this capability is the situation when you need to change the owner of many or all objects within a library. Since there is no predefined action to change the owner of an object, you need to create a user defined action by performing the following steps:

1. Right-click a library or object in the Remote Systems view or iSeries Table View and select **User Actions → Work With User Actions...** from the context menu and the **Work With User Actions** dialog appears.

2. In the left pane, under **New**, select **Object Action** as indicated by the mouse pointer in Figure 5-65.
3. On the right side of the pane, in the field *Action name:* enter the name you want to use for the new action, for example *Change Object Owner.*

4. In the field *Comment:* you may enter a more detailed description, for example *Change the owner of the selected objects.*

5. Enter the name of the command to be executed in the *Command:* field. In our example, it is the CHGOBJOWN command.

6. To make sure the names of the selected objects are used as parameters, you should insert variables. To do so click the *Insert Variable* button and select *&L - Object or Member Library Name* which inserts the name of the library for the selected object.

7. On the right of the &L in the *Command:* field, key in a slash (/) as the delimiter between library and object name.

8. Click the *Insert Variable* button again and select *&N - Name of selected resource* which inserts the name of the library for the selected object.

9. On the right of the &N in the *Command:* field, key in a space as the delimiter between object name and object type.

10. Click the *Insert Variable* button again and select *&T - Object or member type with asterisk* which inserts the name of the library for the selected object.
11. To make sure you used the correct syntax for the Change Object Owner (CHGOBJOWN) command, click the Prompt button and the Change Object Owner prompt as shown in Figure 5-66 appears.

![Change Object Owner Prompt](image)

As you can see in Figure 5-66, the New Owner parameter is required and needs to be inserted when the action is executed. If you want to provide a default value, such as QPGMR, you can enter it here. You can also use the substitution variable &U to insert the name of the user profile used for the RSE connection at the time when this user action is executed.

12. To see the names of the parameters, select the Keywords check box.

13. Click the Cancel button to close the prompt dialog.

14. If you want a command prompt to appear before the action is executed, select the Prompt first check box. Note, the command is prompted for each object, if run the action against multiple objects.

15. Click the Create button to create your new user action.

16. Click the Close button.

17. Enter the parameter NEWOWN(MyProfile), where MyProfile is the name of the new owner, into the command field as indicated by the mouse pointer in Figure 5-67.
18. Select one or more objects in the table view as shown in Figure 5-67, right-click, and select **User Actions → Change Object Owner** from the context menu. The command CHGOBJOWN is performed against all selected objects.

### 5.9 Drag and drop functionality

The RSE supports drag and drop functionality. Dragging and dropping within the RSE copies items between source and target locations. This functionality provides you with the ability to drag and drop:

- Objects and members between libraries and source files on the same iSeries host.
- Source files and members between libraries on different iSeries hosts.
- Between different views: For example, you can drag and drop items between the Remote Systems tree-view and the iSeries table view.
- Drag and drop also works for IFS files and directories.
- You can also drag and drop (or copy and paste) between different system types, it doesn't have to be between 2 iSeries systems. For example, you can drag a save file from your iSeries server to a directory on your local PC.
Chapter 6. LPEX, the modern SEU plus much more

This chapter builds on the concepts and techniques introduced in Chapter 5, “Managing your iSeries development resources using RSE” on page 75. It explains how to use the Remote System Explorer (RSE) for specific iSeries development activities such as the editing, compiling, running, and debugging of your applications. The Remote Systems LPEX Editor is considered to be the modern source entry utility (SEU).

This chapter covers the following topics:

- Editing source members in the RSE
  - Remote Systems LPEX Editor
  - Editor functions
- Compiling source members in the RSE
  - Run-time compile options
  - Working with compile commands
- Running programs from the RSE, including the running modes normal, batch, interactive
- Debugging programs in the RSE

Chapter 5, “Managing your iSeries development resources using RSE” on page 75, serves as an introduction to the RSE. We highly recommend that you read Chapter 5 before you continue with this chapter.

The chapter was written originally based on WDSc 5.0 and now updated with the enhancements for WDSc 5.1.2. All significant changes are marked by a vertical bar at the left side of the page.
6.1 Editing source members in the RSE

This section introduces you to some basic concepts for editing your source members in the default editor within the RSE perspective. The default editor is the Remote Systems Live Parsing Extensible Editor (LPEX) Editor. The Remote Systems LPEX Editor is a powerful language-sensitive editor that can create and edit many kinds of files, including programs, documentation, and text files.

However, we focus on using the Remote Systems LPEX Editor for editing iSeries source members, such as RPG, COBOL, Data Description Specification (DDS), and CL. We also explain and demonstrate some common editing related functions with this editor.

By definition, The LPEX editor is the default editor for source files in the Remote System Explorer, and can be used to create and edit many kinds of files, including program source files, documentation, and data files. In addition to basic editing functions, the LPEX Editor offers language parsing, location marking, elaborate search, and key stroke recording facilities.

6.1.1 Launching the Remote Systems LPEX Editor

To open source members using the Remote Systems LPEX Editor, follow these steps:

1. Navigate the filters in the Remote Systems view until you find the member or members you want to edit. You can also do this from an iSeries table view.

2. Double-click the member, or right-click the member, and select Open With → Remote Systems LPEX Editor. The editor launches and loads the member or members. See Figure 6-1.
In reference to Figure 6-1, the following list highlights some areas of the Remote Systems LPEX Editor:

- **Source member tabs**: As shown in Figure 6-1, more than one instance of the editor can be launched so that multiple source members can be opened. To edit a particular member, click the source member’s tab to place focus on that member.

  **Tip**: You can also use the keyboard shortcut, Ctrl+F6, to switch between editor instances.

- **Status line**: This line provides information about the current status of the editor. For example, the Row and Column numbers indicate the current position of the cursor in the editor area. This line also shows which mode the editor is in. When the editor is in Replace mode, the cursor appears as a solid block. Any text overlaid by the block is replaced by any new text that you type. When the editor is in Insert mode, the cursor appears as a thin vertical line. Any text that you type is inserted into the member at the cursor position, and existing text to the right of the cursor is shifted to the right. You toggle back and forth between the two modes, by pressing the Insert key.

- **Format line**: This line displays the grid line that shows marked fields based on the format type of the line being edited. Useful for RPG and DDS type source members.

- **Working area**: You click in this area to enter your text and perform other general editing tasks.
SEU commands area: This is where SEU commands are entered. These are the commands that most iSeries developers are familiar with when working with the SEU editor in OS/400 green-screen mode. To execute these commands, you click in the designated area, type the command, and press Enter.

Remote Systems LPEX Editor commands area: You enter editor default or user-defined commands in this area. To execute these commands, you click in the LPEX editor command area, type the command, and press Enter.

For example, when you enter the following command lines into the command area, two new lines are added to the file:

```
add 2
```

Tip: You can put the cursor at any location where you need more help and press F1.

6.1.2 Working with the Remote Systems LPEX Editor

LPEX is an editor provided with the workbench based on the Eclipse open source project that is not development language specific. However, it can be extended with parsers that provide this functionality. So when you are editing an RPG source member in the workbench you are really using the LPEX editor plus the ILE RPG parser. The parser is what handles the token highlighting, syntax checking, F1 help, content assist, and all other language specific features. Parsers are automatically used by LPEX based on the file type.

That is why in the preferences dialog you see a section for basic LPEX Editor Window → Preferences → LPEX Editor → Parsers to configure basic editor features as shown in Figure 6-2. And another section for language specific LPEX Editor Window → Preferences → Remote Systems → iSeries → LPEX Editor Parsers to customize language specific features as shown in Figure 6-3.

![Figure 6-2 Customizing basic editor features](image)
Figure 6-3  Customizing language specific features

The old CODE Editor was also frequently referred to as “LPEX”. That is why in the RSE it is referred to as “Remote Systems LPEX Editor” even though it is the same as the LPEX Editor.

Many basic and advanced functions are available with the Remote Systems LPEX Editor. These include:

- Cut, copy, and paste operations
- Block marking of lines, characters, or rectangles with copy, move, and delete operations
- Powerful find and replace functionality
- Unlimited undo and redo operations
- Token highlighting: Different language constructs are highlighted using different colors and fonts.
- SEU-like format line rulers to show the purpose of each column for column sensitive languages like RPG and DDS: These rulers can automatically update themselves to reflect the current specification.
- SEU-like source line prompting for RPG and DDS source members
- Sequence numbers that allow SEU-style commands in the prefix area
- Intelligent tabbing between columns for column-sensitive languages
- Automatic uppercase for languages that expect uppercase
- Commands to simplify text insertions and deletions for column-sensitive languages
- Filtered views to display specific source member content in the editor
- Indentation display of control structures for columnar languages
A show fields function for viewing field details of any database, workstation or printer files referenced in the source member

Source verification

Online language reference help

Syntax checking for RPG, Cobol and CL

You can find more information about these functions in the online help. From the workbench menu, follow these steps:

1. Click Help → Help Contents.

2. In the Help application window, expand Developing → iSeries server applications → The edit, verify, compile, run, and debug cycle in the Remote System Explorer → Editing → LPEX Editor → Working with text.

You can customize the Remote Systems LPEX Editor to your requirements. You can change Tab settings, tailor keyboard mappings to your requirements, and modify the general appearance and layout of the editor. For information about editor customization, see the online help. From the workbench menu, follow these steps:

1. Click Help → Help Contents.

2. In the Help application window, expand Developing → iSeries server applications → The edit, verify, compile, run, and debug cycle in the Remote System Explorer → Editing → LPEX Editor → Customizing the editor.

You can also issue editor specific commands in the commands area of the editor. For more information about LPEX editor commands and how to issue them, see the online help. From the workbench menu, follow these steps:

1. Select Help → Help Contents.

2. In the Help application window, choose either of the following actions:
   - Use these steps to reach the default editor commands help
     i. Expand Developing.
     ii. Expand iSeries server applications.
     iii. Expand The edit, verify, compile, run, and debug cycle in the Remote System Explorer.
     iv. Expand Editing.
     v. Expand LPEX Editor.
     vi. Expand Default editor commands.
   - Or you can follow these steps to reach the issuing editor commands help:
     i. Expand Developing.
     ii. Expand iSeries server applications.
     iii. Expand The edit, verify, compile, run, and debug cycle in the Remote System Explorer.
     iv. Expand Editing.
     v. Expand LPEX Editor.
     vi. Expand Working with text.
     vii. Expand Issuing editor commands.
6.1.3 Remote Systems LPEX Editor functions

This section discusses and demonstrates a cross-section of the functions that are available with the Remote Systems LPEX Editor.

Source line prompting

Source line prompting assists you in formatting your source code lines correctly. This function corresponds to the F4 prompting function within the SEU editor. This feature is available for RPG, DDS, and CL source members.

To prompt a source line in the editor, follow these steps:

1. Open the source member in the editor. See 6.1.1, “Launching the Remote Systems LPEX Editor” on page 140.
2. Click the source line you want to prompt.
3. Right-click and select Prompt or simply press F4.
4. The prompt window displays as the iSeries Source Prompter view. This view is usually part of a tabbed notebook of views. See Figure 6-4.

5. Enter, or select from the drop-down lists, appropriate values for the fields in the prompted line and press Enter.

After the iSeries Source Prompter view is active, the data within this view changes to reflect the source line the cursor is currently positioned on.

The prompt window is different for CL type members. When you prompt a line in a CL source member, a prompt for the CL command displays whereby you can enter in details for the command and then click OK to return to the editor.

Tip: In older editions of WDSc, you can invoke the iSeries Source Prompter via the keyboard shortcut Shift+F4. But now, you can invoke it simply via F4.

Syntax checking

As the title of this section suggests, this function checks your source code for correct syntax based on the source member type. Two variations of this function are available:

- Syntax check a source line
- Syntax check a member

To syntax check a source line, follow these steps:
1. Open the source member in the editor. See 6.1.1, “Launching the Remote Systems LPEX Editor” on page 140.

2. Click the source line you want to prompt.

3. Right-click and select **Syntax Check Line**.

4. If there is an error within the source line, a message is displayed directly below the offending line. See Figure 6-5.

5. Position the cursor on the line that has the error and make the necessary correction. When you move the cursor off the line, the error message is removed from the editor working area.

To syntax check the member, place focus on the member by clicking its tab. From the workbench menu, select **Source → Syntax Check All**.

**Tip:** The syntax checking function is invoked automatically by default for RPG, COBOL, DDS, and CL source members when you move the cursor off a source line. To disable the auto-checking feature, follow these steps:
1. From the workbench menu, select **Window → Preferences**.
2. In the left side pane of the **Preferences** window, expand **Remote Systems → iSeries → LPEX Editor Parsers**.
3. Select the source member type for which you want to turn this feature off.
4. In the right side pane, deselect the **Automatic syntax checking** check box.
Filter views
A filter view is a useful function that allows you to specify which sections of the source member you see in the editor. The filter criteria changes depending on the type of member being edited.

In the following scenario, we specify that the filter view only displays, in the editor, subroutines defined in an RPG member.

To specify a filter view, follow these steps:
1. Open the RPG source member in the editor. See 6.1.1, “Launching the Remote Systems LPEX Editor” on page 140.
2. Right-click anywhere in the editor working area and select Filter view → Subroutines.
3. The view in the editor changes, displaying only subroutine names within the source member. You can expand each subroutine by clicking the plus (+) sign to the left of the source line sequence number to reveal the source contained within the subroutine. See Figure 6-6.

Figure 6-6 Filter view of subroutines in an RPG member

4. To revert back to the view that displays all the source lines of the member, right-click in the editor working area and select Show all.

This example used the filter criteria “subroutines”. You can also filter on other items contained within the source member, such as:

- **By Date** - You may choose a date or a pair of dates to show only lines which have been changed before, after or between those dates.
- **Code** - All comment lines are hidden.
- **Comments** - Only comment lines are shown.
- **Control** - Only flow control statements, such as If, Do and so on, are shown.
- **Procedures** - Only procedures are shown.
Show fields
Show fields is another useful function that allows you to view field details of database, workstation, and printer files that are defined in RPG or CL source members. In this section, we invoke this function on an RPG source member.

To use the Show fields function, follow these steps:

1. Open the RPG source member in the editor. See 6.1.1, “Launching the Remote Systems LPEX Editor” on page 140.

2. Click a file specification line in the RPG member. Note, to be able to show the fields of an externally described file, you need to make sure that the library containing that file is part of the library list.

   **Tip:** To quickly add a library to the library list of your RSE server job, right-click the library in the RSE view and select **Add to Library List** ... from the context menu. You can add the library name at the start or at the end of the library list, or before, after or replacing another name you can specify.

3. Right-click and select **Show fields**.

4. The iSeries Field Table View opens and displays all fields defined in the file. Field details are displayed as rows in the table view. This view is usually part of a tabbed notebook of views. See Figure 6-7.

5. To view the field’s properties, right-click the field in the table and select **Field Properties**. The Field Properties window (Figure 6-8) opens to display further attributes associated with the field.
Online help
Two types of online help are available:

- Context-sensitive help
- Content assist

**Context-sensitive help**
To invoke context-sensitive help, click the item for which you need help and press the F1 key. The help documentation opens in a new window and you are taken to the section of the Help that is associated with the item you requested help for.
For example, clicking the EXFMT opcode in an RPG member and pressing F1 opens the help documentation to the page with details on this opcode. Figure 6-9 shows an example of context-sensitive help invoked for the EXFMT opcode.

**Content assist**

Content assist is a “short form” version of help. It can be used with RPG source members. This function can be used for opcodes, built-in functions, and user-defined tokens such as fields and subroutines.

To invoke content assist, click the token you need assistance with and then follow these steps:

1. Either press Ctrl+Spacebar, or from the workbench menu, select Edit → Content Assist.
2. A scrollable list of keywords that match the first letter of the token you selected is displayed. Select the keyword of interest to you. Then a description of the token is displayed in an adjoining box. Figure 6-10 shows an example of content assist invoked for the EXFMT opcode.
3. To return to the editor, either press the Esc key, or click in the editor working area.

![Figure 6-10: Content assist invoked for RPG opcode EXFMT](image)

Content assist can also be used to insert entries into your source member:

1. Type one or more characters of the item you want to insert into the source member.
2. Press Ctrl+Spacebar, or from the workbench menu, select Edit → Content Assist.
3. From the scrollable list, select the item you want to insert.
4. Press Enter to insert the item.

Note, code assist not only helps you to find the correct keywords according to the language syntax but also user-defined tokens, such as the names of variables, subroutines, and files which would be allowed to enter at this point. Figure 6-11 shows such an example for the ILE COBOL language, which is also supported for ILE RPG, but not for CL or non-ILE languages sources. For user-defined tokens, you must refresh the Outline view. For more information about the Outline view, see 6.1.4, “Outline views” on page 155.

**Content assist support for COBOL**

In WDS 5.1.2, content assist support for COBOL was also added. It works in all COBOL divisions, that is in the IDENTIFICATION, ENVIRONMENT, DATA, and PROCEDURE division for keywords and statements.
Figure 6-11 shows code assist for user-defined in ILE COBOL. Remember, you must refresh the Outline view in order to see user-defined tokens.

Source verify
The source verify function checks the validity of your source code without creating a compiled object on the iSeries host. Source verification can be performed on RPG, COBOL, and DDS source members. Verification can also be done offline.

To invoke the verify function, follow these steps:
1. Drill down through the Remote Systems view or iSeries Table view to find the source member you want to verify.
2. Right-click the source member and select either Verify or Verify (Prompt). The difference between Verify and Verify (Prompt) is that in Verify (Prompt) you are prompted to enter parameters for the verification process. See Figure 6-12. You can also select Source from the workbench menu and select Verify or Verify With Prompt. To invoke the verifier in this manner, the source member must already be opened in the Remote Systems LPEX Editor.
3. The source verifier launches and runs. Any errors found by the verifier are reported in the iSeries Error List view. Details, such as message ID, message text, severity, and source line number, are displayed for each error found. This view usually appears as part of a tabbed notebook of views similar to the one in Figure 6-13.

4. Double-click an error in the iSeries Error List view. The source member opens in the editor with the cursor positioned at the source line in error. An error message is also displayed directly beneath the offending source line. Correct the error and the message is removed.
5. Repeat the previous step for each error in the iSeries Error List view.

**Tip:** You can also invoke the source verifier using the keyboard shortcut Ctrl+Shift+V.

**Global verifier preferences**
Preferences for the source verifier can be set globally. In this example, we set a global preference that instructs the verifier to generate a listing after verification is completed. We set this preference for RPGLE source members.

1. From the workbench menu, select **Window → Preferences**.
2. In the left side pane of the Preferences window, expand **Remote Systems → iSeries → Program Verifiers → ILE RPG**.
3. In the right side pane of the Preferences window, select the **Generate listing** box.

Now, when a source verification is invoked on an RPGLE source member, a source listing is generated in the iSeries Listings view as shown in Figure 6-14.

![Image of iSeries Listings view]

To override any global settings temporarily, select the **Verify (Prompt)** option.

**Show indentation**
The Show indentation function applies to RPG source members. Invoking this function shows an indented view of any RPG language control structures such as IF, DOW, and FOR. This feature is useful for analyzing fixed format RPG source members that contain deeply nested control structure code.

To use the Show indentation function, follow these steps:
1. Open the RPG source member in the editor. See 6.1.1, “Launching the Remote Systems LPEX Editor” on page 140.
2. From the workbench menu, select **Source → Show Indentation**.

An indented view of the source member is displayed in the iSeries Indent view. See Figure 6-15.
Displaying and editing /COPY members and copy books
This function enables you to open any members that are referenced by RPG’s /COPY
compiler directive and COBOL’s COPY compiler directive. The referenced members are
opened in the Remote Systems LPEX Editor with their own tab.

To invoke this function on an RPG member, follow these steps:

1. Open the RPG source member in the editor. See 6.1.1, “Launching the Remote Systems
LPEX Editor” on page 140.
2. Position the cursor on a source line that contains the /COPY directive.
3. Perform either of the following steps:
   – Right-click and select /COPY Member → Edit to open the member in the editor in
     change mode.
   – Right-click and select /COPY Member → Browse to open the member in the editor in
     browse mode.

For COBOL members, select Copy Book and then Edit or Browse.

Saving your changes
When you edit a source member, the number of changes that you make during your editing
session are recorded in the status line. The changes you make are temporary until you decide
to save them. Members with unsaved changes are denoted by an asterisk (*) next to the
source member’s name tab. See Figure 6-16.

Saving your changes
When you edit a source member, the number of changes that you make during your editing
session are recorded in the status line. The changes you make are temporary until you decide
to save them. Members with unsaved changes are denoted by an asterisk (*) next to the
source member’s name tab. See Figure 6-16.
When you save your changes, both the asterisk and number of changes indicator are cleared from the window.

To save changes made to your source member, perform one of the following steps:

- Right-click in the editor work area and select Save.
- From the workbench menu, select File → Save.
- Press Ctrl+S.

If multiple editor instances are open, you can save all the members in one action. To do this, you can select File → Save All from the workbench menu, or simply press Ctrl+Shift+S.

**Closing editors**

To close an editor, perform one of the following actions:

- Click the Close (X) button in the member tab of any open editor.
- From the workbench menu, select File → Close.
- Press Ctrl+F4.

If multiple editor instances are open, you can close all instances in one action. To do this, select File → Close All from the workbench menu, or simply press Ctrl+Shift+F4.

### 6.1.4 Outline views

Although Outline views are not editing functions, they prove most useful when used in conjunction with the editor. Outline views display the outline of a structured member that is currently open in the editor area and lists the structural elements of the member.
These views are applicable to RPG and COBOL source members, as well as other types of sources in other perspectives, such as Java, HTML, Cascading Styles Sheets (CSS). By default, when you open a source member in the editor, the Outline view appears as a separate view in the RSE perspective. If the member type is appropriate, then an Outline view is shown. Otherwise the view is empty.

If the Outline view does not display due to it having been closed from a previous workbench session for example, you can restore the original appearance of the Remote Systems Perspective. To do so, select Window → Reset Perspective.

If you just want to open the Outline view without changing the size and position of the other panes, select Windows> Show View → Other. This will open the Show view window as shown in Figure 6-17. Expand Basic and choose Outline, then click the OK button. You can do so also in any other perspective, which does by default not show an outline, such as the WebFacing perspective.

The Outline view usually opens besides the editor area, but this may vary depending on your workbench preferences. Figure 6-18 shows the Outline view for an RPG source member.

![Figure 6-18 Editor pane and Outline view for an RPG member](image)

In Figure 6-18, the Outline view represents all the structural elements of the RPG source member in an easy to navigate tree. Expanding any of the elements reveals further child elements contained within.

For example, expanding a file element reveals a record format element. Expanding this element displays all the fields defined within the record format. Clicking any of the elements also positions the cursor to the definition of that element within the source member if applicable.

As of WDSc 5.1.2, the outline view not only shows where each element is defined, but also where its referenced. Again, clicking one of the circles in the outline, highlights the corresponding line in the editor pane as indicated by the mouse pointer in Figure 6-18.

Note, the line number shown at the right of the icon may not be the same as the one on the left of the highlighted line in the Editor pane. This happens if you inserted or deleted lines and decided not to resequence the line numbers under Window → Preferences → Remote
Systems → iSeries → LPEX Editor Parsers and deselected the Resequence lines at save check box.

To refresh the contents of the tree in the Outline view at any time during an editing session, click the Refresh button in the toolbar of the Outline view.

External description is not available

If your source member contains externally described files or data structures, you might see the text ‘External description is not available’ when you expand a file in the outline view. Very likely, this is caused by the fact that your RSE server job does not have the appropriate library name for that file in its library list.

Figure 6-19   External description is not available

To quickly add that library to the library list, right-click the library in the RSE view and select Add to Library List ... from the context menu. Then click the Refresh button in the toolbar of the Outline view.

Figure 6-20   Selecting Fast View
Using the Fast View

If you prefer to edit your source member in a maximized editor pane (by double-clicking the tab on top of the frame), you can still use the Navigator view. The trick is, to show it in a Fast View. To do so, right-click the bar on top of the Outline view and select Fast View from the context menu as shown in Figure 6-20. The result is that the outline view disappears and the icon representing it now appears on the left window bar underneath the icons for the open perspectives as shown in Figure 6-21.

![Figure 6-21 Outline as Fast View](image_url)

You can now maximize the editor pane. If you want to look at the Outline view, you single click the icon in the left bar and the outline overlaps the editor window until you click the button again.

6.1.5 Wizards in the Remote Systems LPEX editor

The Remote Systems LPEX editor also provides several wizards to make programming easier. Some of them are described in the following sections.

Creating an RPG procedure

A procedure is any piece of code that can be called with a bound call, in essence, the CALLP operation code. You can now an RPG program, main procedure, or subprocedure while editing any RPGLE member in the Remote Systems LPEX editor.

To create the procedure perform the following steps:
1. Expand your connection until you find the file in which you want to create the RPG procedure.
2. Double-click the file to open it in the Remote Systems LPEX editor.
3. With your cursor inside the editor view, right-click and select **New → RPG Procedure**. Or select **Source → New → RPG Procedure** from the workbench menu.
4. In the **New RPG Procedure** window select the procedure type from the Procedure type drop-down menu as indicated by the mouse pointer in Figure 6-22. Note that your selection automatically completes the External Name (EXTPROC) field.

   ![RPG Procedure Wizard](image)

   **Figure 6-22 RPG procedure wizard**

   You have the following choices:
   - Program (EXTPGM)
   - Main procedure (EXTPROC)
   - Subprocedure (EXTPROC)
   - Subprocedure with:
     - CLLE calling conventions (EXTPROC(*CL))
     - C calling conventions and parameter widening (EXTPROC(*CWIDEN))
     - C calling conventions and no parameter widening (EXTPROC(*CNOWIDEN))

5. Enter a valid ILE RPG name in the Procedure name field.
6. (Optional) Modify the external name in the External Name (EXTPROC) if you want to override the default value.
7. (Optional) Enter a descriptive purpose for the procedure. This description will appear in the code's header comments.

8. Select the Exportable for use with other code (EXPORT) check box if this procedure is exportable for use with other code. Note, however, that if you select this check box, the wizard places both the prototype and procedure code into the current file. After you finish the wizard, you need to manually move the prototype into a COPY file, so that procedures can call this prototype. You also need to enter a COPY statement into the procedure code, at the place where you removed the prototype code. Click Add to add a parameter to your procedure.

9. (Optional) Select the Return a value check box if you want to define a Return value. This action enables the Next button. Otherwise, click Finish.

If you selected Return a value, you also need to perform the following steps:

1. Click Next to go to the Return Value page of the RPG Procedure wizard.

2. Select a return value type from the Type drop-down list. The type that you select automatically completes a number of the other fields in this page of the wizard, and keeps some of the fields disabled if they are not applicable for the type you selected. You can modify any of the values in the enabled fields if you want to. Make sure to check the F1 help in the various fields of this wizard if you are not sure what to enter.

   Note: If the type you selected is date or time, the format label will be set to DATFMT (for date) or TIMFMT (for time). Then you can select the format and separator for your date or time label. If the type you selected is data, a Class Entry field is enabled where you can specify a valid Java class.

3. (Optional) Enter a description for the return value. This description will appear in the code's header comments. Click Finish.

Creating RPG D-Specifications

With WDSc 5.1.2, you can now use a New RPG D-Specification wizard to help you create a Definition Specification from any ILE RPG file in the Remote Systems LPEX editor.
A D-Specification defines items used in your program, such as standalone fields, named constants, and data structures with or without a subfield.

To create the D-Specification perform the following steps:

1. Expand your connection until you find the ILE RPG file in which you want to create the D-Specification.
2. Double-click the file to open it in the Remote Systems LPEX editor.
3. With your cursor inside the editor view, right-click and select **New → D-Specification** as shown in Figure 6-23 or select **Source → New → D-Specification** from the workbench menu.
4. In the New RPG D-Specification wizard as shown in Figure 6-24, you may specify a name and purpose for the D-Specification.
5. Select the type of D-Specification from the Type drop-down list. Your selection determines the content of the rest of the pages in the New RPG D-Specification wizard. Use the F1 help throughout the rest of the wizard for more information about each option.

![Figure 6-24](image)

**Figure 6-24**  First page of the D-Specifications wizard

You have the following choices:

- **Standalone Field**: to create a standalone field type on a field-type page or field-type keyword page, that contains field types and their keywords.
– **Named Constant**
– **Program-described data structure**: to create a Simple data structure keyword page or a Simple data structure subfield page, where the keywords and subfields are created separately.
– **Externally-described data structure**: to create an External data structure page or an External data structure subfield page, where the keywords and external subfields are created separately.
– **Program status data structure**: to create a Program status data structure subfield prefix page, and to specify the keywords and subfield prefixes for the Program status data structure.
– **File information data structure**: to create a File information data structure subfield prefix page, and to specify the keywords, subfield prefixes, and the File information data structure type.

6. Select from the three radio button choices underneath *Where to insert the new specification*:
– **At the current cursor location**: to simply insert the code at the current location in your source.
– **In the D-Specifications of the current procedure**: to append the code to the D-Specification of the procedure where your cursor is located.
– **In the global D-Specifications**: to insert the code right before the first found Input specification, which could be a C-Specification, O-Specification, P-Specification, “/free”, “***”, or the end of the file.

7. Click **Next** and the second page of D-Specifications wizard as in Figure 6-25 appears.

![Figure 6-25  Second page of D-Specifications wizard](image)
8. Complete the rest of the wizard, using the F1 help for tooltips and suggestions for all of the options as indicated by the mouse pointer in Figure 6-25.

**Creating an RPG Java Method Call**

Also with WDS 5.1.2, you can now create an RPG Java Method call from any ILE RPG file, when you use the Remote Systems LPEX editor. An RPG Java method call is a piece of ILE RPG code that calls various Java methods, so you can call Java methods from your RPG program. When you specify various Java class and method information, the method call creation wizard creates valid ILE RPG code that calls the Java methods from your RPG program.

The New Java method call wizard will change dynamically, depending on the selections that you make. Use the F1 help for more information about each option as you go through the wizard. To create the RPG Java method call perform these steps:

1. Expand your connection until you find the file in which you want to create the method call.
2. Double-click the file to open it in the Remote Systems LPEX editor.
3. With your cursor inside the editor view, right-click and select **New** → **Java Method Call**. Or select **Source** → **New** → **Java Method Call** from the workbench menu.
4. In the **New Java Method Call** wizard, specify the Java package and the class you want to use for the method call. Use the **Browse project**, **Browse local**, and **Browse remote** buttons to search for packages and classes. When you search for class files on your local system, you can click **Add JAR file** to find a file that contains the class you want to use. Use the F1 help for more information about each option.
5. Click **Next**.
6. Select the Java method that you want to use. The methods listed are the ones available from the class selected in the previous page of the wizard.
7. Click **Next**.
8. Select the Java method parameters that you want to use. For each parameter, you can generate an RPG field, specify its RPG name, type, length or dimension, and whether to pass the parameter by reference or to have a read-only reference. You can click each cell in the table to edit the value directly.

   If you selected a method that does not contain parameters, then the page will be blank, yet considered complete, with the Next button enabled. Click Next.

   **Note:** When you produce an RPG Java method call from ILE RPG, the ILE RPG code is mapped into Java code. For example, “char” in Java is mapped to “1C” in ILE RPG. See the related reference topic for more information about mapping between standard Java types and ILE RPG types.

   If the Java method you selected is not static, then:

   a. Specify the Java constructor you want to use to construct the Java object. (This page, and the next one, do not appear if you previously selected a static method, as no constructor is necessary.)
   b. Click **Next**.
   c. Specify parameters for the Java constructor selected. For each parameter, you can generate an RPG field, specify the name, type, length, dimension, and whether to pass the parameter by reference or to have a read-only reference.
   d. Click **Next**.
9. Specify the options for code generation. You can specify whether or not to generate an RPG object for your Java class, whether or not to create a method prototype, and where to place the generated RPG code.
   - If you decide to generate an RPG object, you need to specify an RPG name for the object and an RPG name for the object constructor.
   - If you decide to generate an RPG method prototype, you need to specify a prototype name.
   - If you decide to generate a method call code, you need to specify an RPG return field name.

10. You have two choices for specifying where you want to place the generated code:
   - Distributed appropriately: to generate D-Specifications in the D-Specification area of your code, and to generate calculations (the code to call the method) after the cursor.
   - All after the current line: to generate and append the code after the cursor.

11. Click Finish.

Converting ILE RPG code from fixed form to free form

You can convert fixed form ILE RPG code into free form ILE RPG code (C-specifications) in the Remote Systems LPEX editor.

To convert your source, with your ILE RPG source open in the Remote Systems LPEX editor, select Source ➔ Convert All To Free Form from the workbench menu.

Tip: You can use Edit ➔ Undo from the workbench menu, or press Ctrl+Z, to convert back the source if you want to make more changes.

If you only want to convert a section of your source (and not the entire file), select the source you want to convert, right-click and select Convert Selection To Free Form. In this instance, the selected source is replaced with the converted source.

Note: When you convert your source to a free format, the default preference for automatic indenting determines how your source is indented. To change this setting:

1. From the Remote Systems view, click the drop-down menu and select Preferences ➔ iSeries LPEX editor parsers.
2. In the preferences window, expand LPEX Editor Parsers and click ILE RPG. In the preferences area, verify that Automatic Indent is checked, and adjust the value in the Blanks (1-20) field.

### 6.1.6 WDSc 5.1.2 enhancements for the ILE RPG editor

Besides the fixed form to free form conversion wizard described in “Converting ILE RPG code from fixed form to free form” on page 164, WDSc 5.1.2 has brought some more enhancements for editing ILE RPG free form statements.

#### Automatic indenting when entering free form

On a free-form line, when Enter is pressed, automatically indent the position of the cursor for the new line if the current line has one of the following opcodes (line does not need to have semi-colon):

- DOU
- DOW
- FOR
- IF
Chapter 6. LPEX, the modern SEU plus much more

6.1.7 Customizing the automatic formatter

When editing free-form RPG, CL, CLP, CLLE, and CMD source with the Remote Systems LPEX editor, the source is automatically formatted. You can customize this formatter to suit your needs.

When you enter source into the editing area of CL file, the text is formatted when you go to a new line or you move your cursor to a different line. For example, if you begin your line of text with a command such as CRTCMD, the beginning of the label is automatically positioned to column 2. However, you can specify a different column position in the editor preferences.

Auto-indent is controlled by a preference setting as described in “Customizing the automatic formatter” on page 165. The indent value is specified in the ILE RPG Parser preference page. Valid range is 1 to 20 with a default of 2. The preference is on by default. The preference is disabled if the ‘Repeat previous specification type’ preference is off.

Auto closer when editing RPG sources

When Enter is pressed, the corresponding ENDxx statement is automatically add after the new line if the current line has one of the following opcodes (line does not need to have semi-colon):

- DO (fixed form only)
- DOU
- DOUxx (fixed form only)
- DOW
- DOWxx (fixed form only)
- FOR
- IF
- IFxx (fixed form only)
- BEGSR
- MONITOR
- SELECT

This applies to fixed and free form.

The case for the ENDxx statement is controlled by a ‘Style’ sub-preference to the Auto-closure preference. The possible values are ENDXX, EndXx, Endxx, endXx, and endxx. The default is ENDXX.

Auto-closure is controlled by a preference setting. The preference is off by default. The preference is disabled if the ‘Repeat previous specification type’ preference is off.

Syntax checking of embedded SQL statements in ILE RPG

Now, with WDS 5.1.2, SQL statements within ILE RPG sources are also syntax checked by the Remote Systems LPEX editor.
1. From the Remote Systems view, click the drop-down menu and select **Preferences → iSeries LPX editor parsers** or select **Window → Preferences** from the workbench menu and then select **Remote Systems → iSeries → LPX Editor Parsers** in the left pane of the Preferences dialog.

2. In the Preferences window, select **CL** (or any other language you want to customize the formatter for).

3. The check box for **Automatic formatting** should be selected by default. You can then edit the entries in the fields below to customize the formatter:
   - The value in the **Label position** field specifies the column position for the label.
   - The value in the **Command position** field specifies the column position for the command. This value must be greater than the Label position value, unless you select the Label above command check box, in which case, the command can appear directly beneath or even to the left of the label, on the next line.
   - The value in the **Continued line indent** field specifies the column when the text will be continued if it spans more than one line.
   - Check the **Label above command** check box if you want the label to appear above, rather than to the left of your command string.
   - Check the **One parameter per line** check box if only want one parameter to appear on each line.

![Figure 6-26 Customize the automatic formatter for CL source](image)

Figure 6-26 Customize the automatic formatter for CL source
6.2 Compiling source members in the RSE

This section describes the process for compiling source members in the RSE perspective. It also explains how to configure these compile commands to your requirements.

6.2.1 Compiling RPGLE source members

This section presents an example that focuses on compiling an RPGLE source member. Most of the concepts and procedures discussed in this section also apply to other iSeries member types.

This section also focuses on performing these compile-related tasks from the Remote Systems view. This is only our choice for purposes of the demonstration. You can easily perform these tasks from the iSeries table view as well.

All compile commands are accessible through the pop-up menu for source members. The pop-up menu is invoked by right-clicking the source member. Only compile commands that are applicable to the member type are shown in the subsequent submenu that opens when you select the compile option.

The compile process

You have two options when compiling your source members in the RSE:

- **Compile (no prompt):** You are not prompted to edit any parameters for the compile command before the compile starts.
- **Compile (with prompt):** Before the compile starts, you are prompted to edit parameters for the compile command.

The following example uses the “with prompt” option so that we can show the prompted compile command.

For members of type RPGLE, you can choose to create:

- A bound RPGLE program: This corresponds to the CRTBNDRPG OS/400 command.
- An RPGLE module object: This corresponds to the CRTRPGMOD OS/400 command.

To create a bound RPGLE program or RPGLE module object, follow these steps:

1. Navigate the filters in the Remote Systems view until you find the member you want to compile.
2. Right-click the member and perform one of the following actions:
   - Click **Compile (Prompt) → CRTRPGMOD** to create a module.
   - Click **Compile (Prompt) → CRTBNDRPG** to create a bound program. The Create Bound RPG Program (CRTBNDRPG) command interface window (Figure 6-27) opens.
3. Change or specify command parameters. You can select the **Advanced(5)** check box for further command options, or even check the **All Parameters(6)** check box to show all the parameters available for this command. Select the **Keywords(7)** check box to display parameter keywords.

**Note:** The number “(5)” after the word “Advanced” means that you can also select the Advanced(5) check box using the keyboard shortcut ALT+5. This applies also to letters appearing in parentheses behind any menu item.
4. Click **OK** to start the compile process. Messages issued by the compile command are displayed in the iSeries Commands Log view. See Figure 6-28.
Any errors found during the compile process are reported in the iSeries Error List view. This outcome is identical to that of the source verification process. See “Source verify” on page 151 for details.

5. From the iSeries Error List, double-click the error entry. The editor opens the member and positions the cursor on the offending line of source. Again, this procedure is identical to the one followed after performing a source verification. See “Source verify” on page 151 for details.

6. After a successful compile, refresh the Remote Systems view so you can see the newly created program or module object.

Creating programs and service programs from modules
Modules are the building blocks for ILE programs and service programs. The RSE options to create programs and service programs correspond with the OS/400 CRTPGM and CRTSRVPGM commands.

Creating programs
To create a program from a module object, follow these steps:

1. Drill down through your filters in the Remote Systems view until you find the first module required to create the program. You can use the iSeries Table View if you prefer. We find it convenient to choose the module that is to be the program entry procedure (PEP) module. This is the first module that is called after the program is activated.

2. Right-click the module and select Create → Program. The Create Program (CRTPGM) window opens.

3. Enter or select values for the command parameters. Click Advanced to display and specify more command parameters.

4. Click OK to create the program. Messages issued by the command are displayed in the iSeries Commands Log view.

Creating service programs
To create a service program from a module object, follow these steps:

1. Drill down through your filters in the Remote Systems view until you find the first module required to create the service program. You can use the iSeries Table View if you prefer.

2. Right-click the module and select Create → Service Program. The Create Service Program (CRTSRVPGM) window opens.

3. Enter or select values for the command parameters. Select the Advanced(L) check box to display and specify more command parameters, or even check the All Parameters check box to show all the parameters available for this command. Select the Keywords check box to display parameter keywords.

4. Click OK to create the service program. Messages issued by the command are displayed in the iSeries Commands Log view.

Run-time options for compiles
You can set run-time options for compiles either globally or specifically for a connection only. Specific connection settings override the corresponding global settings.

Setting global run-time options
To set global run-time options for compiles, follow these steps:

1. From the workbench menu, select Window → Preferences. The Preferences window shown in Figure 6-29 opens.
2. In the left side pane of this window, expand **Remote Systems → iSeries → Command Execution**.

3. In the right-hand side of the window, complete following tasks:
   a. In the Preferences for compiles and user action variables section:
      i. Enter or select the object library where compiled objects are placed. The default is the library where the source is stored (*SRCLIB).
      ii. Select the **Replace object** check box to specify whether a compiled object is replaced if one exists.
      iii. Select the **Compile in batch** check box to specify whether compiles are to run in batch. If batch is not selected, compiles run in the RSE communications server job.
   b. In the Preferences for batch compiles, commands, and user action variables section:
      i. In the Job description field, enter the name of the job description to be used with the batch job. The default is the job description specified in the user profile (*USRPRF).
ii. In the Job description library field, enter the name of the library where the job description resides. This field can only be changed if you specify a job description other than *USRPRF.

iii. In the Submit Job (SBMJOB) additional parameters field, specify additional parameters to append to the SBMJOB command for batch compiles.

4. Click **OK** to save your changes.

**Setting connection-specific run-time options**

To set connection-specific run-time options for compiles, follow these steps:

1. From the Remote Systems view, right-click your connection and select **Properties**. The Properties for your-connection-name window opens.

2. In the left side pane of this window, click **Subsystems**.

3. The options and fields in the Properties window look similar to those in the Preferences window discussed in “Setting global run-time options” on page 169.

   For connection-specific changes, click the toggle arrow (►) button on each option that you want to change so that the arrow on the button points to the right. When the toggle button is in this state, the corresponding option is activated which allows you to change its value. When the arrow on the toggle button points to the left, the corresponding option is disabled (grayed out) and the global setting is used. See “Setting global run-time options” on page 169.

4. Click **OK** to save your changes.

**6.2.2 Working with compile commands**

You can configure compile commands to suit your requirements. Configuration usually involves changing parameter values to these commands. This section demonstrates this procedure by changing two command parameters for the Create Bound RPG Program (CRTBNDRPG) command.

To change compile command parameters, follow these steps:

1. Drill down through your filters in the Remote Systems view until you find an RPGLE source member. You can use the iSeries Table View as well if you prefer.

2. Right-click the member and select **Compile** or **Compile (Prompt)**. Then select **Work With Compile Commands**. The Work With Compile Commands window as in Figure 6-30 opens.
3. In this window, complete the following tasks:
   a. Make any appropriate changes to the Parent profile field. If this is a change for your own personal profile, you will not need to change this field.
   b. Leave the Member type field set to \texttt{RPGLE}.
   c. In the Compile Commands list, select \texttt{CRTBNDRPG}. The Label field is populated with the value \texttt{CRTBNDRPG} (this cannot be changed) and the Command text box is populated with the OS/400 command string currently associated with the label.
   d. Click \textbf{Prompt}. The prompt window for the \texttt{CRTBNDRPG} command opens.
   e. In the prompt window, complete the following tasks:
      i. Click the \textbf{Default activation group} drop-down list, and select \texttt{*NO}.
      ii. Select the \textbf{Advanced} check box to display and specify more command parameters, or even check the \textbf{All Parameters} check box to show all the parameters available for this command. Select the \textbf{Keywords} check box to display parameter keywords.
      iii. Scroll down, click the \textbf{Debugging views} drop-down list, and select \texttt{*STMT}.
      iv. Click \textbf{OK} to save your changes.
The CRTBNDRPG command string in the Command text box reflects the changes you made in the command prompt window from the previous step. See Figure 6-31. You can also edit the command string directly in the text box if you prefer.

Figure 6-31  Work with Compile Commands window after changes are made

4. Click **Apply** to save your changes.
5. Click **Close** to close the window.

Changes made to compile commands are persistent and scoped to a profile. The preceding procedure is analogous to using the Change Command Defaults (CHGCMDDFT) command to change default values for command parameters. However, within the RSE, different profiles can have different default values for the same compile commands. If you need a different value for a particular parameter, which is rare, you can always select the “compile with prompt” version of the compile command and change the parameter in the prompt window before you run the command.

**Important:** Do not remove the *EVENTF value from the OPTION parameter of any compile command. This option instructs the compiler to create an “events” file. This file is used to populate the iSeries Error List view if errors are found during the compile process.
6.3 Running programs from the RSE

This section describes the process for running your programs from the RSE perspective. It also discusses the options available with regards to the mode you choose to run your programs in.

You have two options when running your programs from the RSE:

- **Run As(B):** You are not prompted to edit any command parameters before the program runs.
- **Run(Prompt)(F):** Before the program runs, you are prompted with a command window allowing you to specify or change command parameters.

In the following example, we use the “with prompt” option so that we can show an example of the prompted run command.

6.3.1 Running a program: An example

To run a program from the RSE, follow these steps:

1. Drill down through your filters in the Remote Systems view until you find a program (*pgm) object that you want to run. You can use the iSeries Table View if you prefer.

2. Right-click the program and select **Run(Prompt)(B)**. A submenu is displayed that offers four choices for modes in which to run the selected program. Select **iSeries application in RSE job** for now. See 6.3.2, “Running modes” on page 176, for more information about the different ways to run your programs.

    *My iSeries program for run* introductory window opens (See Figure 6-32). You can change this name easily through editing in the Name field.
3. Click the **Prompt** button and the Call Program window appears (See Figure 6-33). The Program and Library fields are already populated reflecting your program selection.

4. Enter any required program parameters in the Parameters field. Add, remove, and manipulate the parameter list using the **Add**, **Remove**, **Move up**, and **Move down** buttons.
5. Click **OK** to get back to Figure 6-32.

6. Click the **Run** button to run the program. Messages issued by the CALL command are displayed in the iSeries Commands Log view.

### 6.3.2 Running modes

As mentioned in the previous section, you can run your programs in one of four modes. These modes are:

- **iSeries application in RSE Job**
- **Batch**
- **Interactive**
- **Multi-threaded**

The following sections explain each mode.

#### iSeries application in RSE Job

If you choose to run your programs in normal mode, then the programs are run in the RSE communications server job. It is easier to monitor the status of your jobs with this option, although you tie up the communications server. If you have a long running job, this option may not be the best choice.

When you run your programs in the RSE communications server job, the OS/400 CALL command is used. You are prompted to enter parameters (if required) for your program if you choose the “run with prompt” option. See 6.3.1, “Running a program: An example” on page 174.

#### Batch

Running your programs in batch mode requires no initial setup and works as you would expect. When you run your programs in a batch job, the OS/400 SBMJOB command is used.
You are prompted to make any changes to the SBMJOB command if you choose the “run with prompt” option.

Messages issued from the SBMJOB command are displayed in the iSeries Commands Log view as shown in Figure 6-34.

**Figure 6-34   Messages issued by the SBMJOB command to the iSeries Commands Log**

**Interactive**

To run programs in an interactive job, you need a 5250 green-screen emulator. You must associate the emulator with a connection in the RSE communications server by issuing the Start RSE Server (STRRSESVR) command in the emulator:

```
STRRSESVR NAME(your-connection-name)
```

Your-connection-name is a connection that must already be defined in the RSE. After you type this command, a similar panel as in Figure 6-35 is shown in the emulator session. Note, that it is case sensitive and needs to be specified exactly as in your RSE perspective. You can also specify the following parameters, if you press F4=Prompt:

- A working library (WRKLIB) specifies the library where the STRRSESVR command creates a data queue to communicate with the RSE server. The default (*DFT) selects QGPL to be used.
- The remote location name (RMTLOCNAME). It specifies the TCP/IP hostname or IP address of the client machine where the remote systems explorer iseries communications...
daemon is running. The special value *PRV ensures to use the value form the last invocation of this command. *RESOLVE attempts to automatically determine the correct address and is recommended when using DHCP.

In some network environments, the client's address cannot be determined and you might receive error message RSE2001 ("Error connecting to Remote Systems Explorer"). In this case you may try to enter the IP address of your workstation with parameter RMTLOCLNAME.

- The TCP/IP port number (PORT). The default port number is 4300. If you want to use a different port, you also need to change the value for the RSE communications daemon port number under Preferences → Remote Systems → Communications.

**Important:** The connection name is case sensitive and needs to be specified exactly as in your RSE perspective.

If you attempt to run a program in an interactive job without performing the previous association, a message window opens with instructions on how to associate an interactive job with your RSE connection as shown in Figure 6-36.

![Figure 6-36 Message window for associating an interactive job with a connection](image)

When you run a program in an interactive job, a message is sent to the iSeries Commands Log (Figure 6-37). The message tells you to switch to the emulator where the STRRSESVR command was issued.

![Figure 6-37 Message issued after running a program in an interactive job](image)
Switching to the associated emulator session allows you to view output and interact with the program as shown in Figure 6-38.

![Figure 6-38 Interactive program running in the emulator session](image)

When you end your interactive program, the emulator session shows the display in Figure 6-35 again.

**Multi-threaded**

Programs that spawn new threads have to be started using this option. It is otherwise very similar to the other run modes.

### 6.3.3 Ad hoc commands

You can run ad hoc commands from the RSE in the iSeries Commands Log view (Figure 6-39). To run an ad hoc command from this view, follow these steps:

1. From the first drop-down list, select the mode in which the command is to run.
2. Enter your command string in the second entry field. After you enter a command string or partial command string, click **Prompt** to prompt the command. Then enter the rest of the command details in the prompt window.
3. Click **Run** or press Enter to run the command.
6.4 Debugging programs in the RSE

For information about the debuggers that are available and how to debug your programs in the RSE, see Chapter 14, “Debugging and testing options” on page 545.
Offline, better structured application development with iSeries Projects

This chapter provides an overview of the iSeries Projects perspective. It introduces the concept of organizing your iSeries development resources as off-line projects. In this mode of development, no connection to the iSeries host is required for a majority of your development tasks. iSeries Projects can also be used for more structured iSeries application development.

This chapter covers the following topics:

- The iSeries Projects perspective
  - Disconnected mode
  - iSeries connection and associated libraries
- Opening the iSeries Projects perspective
  - iSeries Project Navigator
- Creating an iSeries Project
  - Project containers, local source physical files and members
- Importing and viewing remote host objects
- Working with local source members
  - Offline source verify
  - Cached file descriptions
- Remote actions
  - Build styles: CL program, command, *NONE
  - Push operations, remote conflict detection
  - Generating compile code
  - Building the project, iSeries Build Status view
The chapter was written originally based on WDSc 5.0 and now updated with the enhancements for WDSc 5.1.2. All significant changes are marked by a vertical bar at the left side of the page.

7.1 What the iSeries Projects perspective is

A project is a container that enables top-level organization of your resources in the workbench. A project contains files and folders. Projects are used for:

- Building applications
- Version management
- Sharing and organizing resources

The primary focus of the iSeries Projects perspective is on disconnected iSeries development on a Windows platform. The intent is that you edit your iSeries resource locally. When you finish updating your files and members locally, you upload these resources to the iSeries host and submit a build of your application.

7.1.1 Working in disconnected mode

The iSeries project perspective allows for disconnected development. A live connection is required only when code updates to the remote host or application builds are needed. If you are working on a remote laptop or mobile computer, or are at home with a slower connection, you may choose to work in disconnected mode. With this option, you explicitly copy files to your PC, laptop, or mobile computer while you have a live connection, close the connection, work on the files at your leisure and then upload them back to the host after you finish.

Another advantage to working in disconnected mode is that you can verify source code for syntax and semantic errors without being connected. And you can submit a compile only when you are connected.

This iSeries Projects development life cycle requires two workbench entities:

- An iSeries connection
- An associated library

**iSeries connection**

You must associate an iSeries connection to an iSeries project. The connection allows you to remotely access the iSeries host. It also manages user ID and password verification to the remote host. For more information about iSeries connections, see 5.2.3, “Connections” on page 79.

**Associated library**

The associated library is a library on the remote iSeries host to which an iSeries project maps. This remote host library is the target of code uploads and application builds submitted from the iSeries project. It is expected that this library be treated as a “personal” developer library. Ultimately, there should be no need for any source in this library to be edited directly on the host.

This concept is in line with most change management procedures. That is “production” code is normally “checked out” into a developer’s own sandbox area. And this sandbox is where a developer makes changes to the code.
7.2 Opening the iSeries Projects perspective

Before you can start working with iSeries projects, you must open the iSeries Projects perspective in the workbench area.

To open the iSeries Projects perspective, follow these steps:

1. From the workbench menu, click Window → Open Perspective. You can also click the Open Perspective button on the toolbar on the left-hand side of the workbench.
2. Select iSeries Projects. If this option does not appear on this submenu, select Other...
The Select Perspective window opens. Select iSeries Projects from this window and click OK.

The iSeries Project perspective has three main views (see Figure 7-1):

- **iSeries Project Navigator**: The iSeries Projects Navigator shows all iSeries projects that currently reside in your workspace.

- **Remote Systems View**: The Remote Systems view enables browsing of various remote systems, including iSeries servers. This view is paramount to the operation of the Remote System Explorer (RSE) perspective. For more information about the RSE and how the Remote Systems view is used, see Chapter 5, “Managing your iSeries development resources using RSE” on page 75, and Chapter 6, “LPEX, the modern SEU plus much more” on page 139.

- **Properties View**: Displays various properties on the object in the current perspective. By default, Properties view appears in the lower left pane together with the Remote Systems view. By clicking the tab at the bottom of that pane you can bring each view in the foreground.

![Figure 7-1 The iSeries Projects perspective](image)
7.3 Creating an iSeries project

Creating an iSeries project provides you with a workspace on your local workstation that contains the resources for the project. The iSeries Project Navigator view allows you to work while disconnected. It uses occasional synchronization with a host to do pushes and builds.

7.3.1 Creating the project container

To create an iSeries project container, follow these steps:

1. Perform one of the following actions:
   – From the workbench menu, select **File → New → iSeries Project**.
   – Click the arrow button to the right of the Open the New Wizard ((JFrame) button and select **iSeries Project**.
   – Right-click anywhere in the iSeries Project Navigator view and select **New → iSeries Project**.

2. In the iSeries Project window (Figure 7-2), complete the following tasks:
   a. In the **Project name** field, enter the name of your iSeries Project.
   b. For **Project contents**, deselect the **Use default** box. This stores the contents of your project under the workspace directory to which the workbench currently points.
   c. Click **Next**.

**Note:** For more information about perspectives and views and how to work with them, see 3.3.3, “Perspectives” on page 32, and 3.3.4, “Views” on page 34.
3. In the iSeries Project Properties window (Figure 7-3), complete the following tasks:
   a. From the Connection drop-down list, select an existing iSeries connection. Or click New to create a new connection. For more information about creating connections, see 5.2.3, “Connections” on page 79.
   b. In the Associated library field, enter the name of the remote library to which this project maps to. Or click Browse to select the associated library from a list of libraries on the iSeries host.
   c. Click Next.
4. There are three *active build styles* to choose from. Build styles specify the set of actions performed when a build of the application is submitted to the iSeries host. For more information about build styles, see 7.6.1, “Build styles” on page 194, and 7.6.2, “Specifying build styles” on page 195. For now, select **CL Program** as shown in Figure 7-4.

5. Click **Finish**. The newly created project appears as a node in the iSeries Project Navigator view.
7.3.2 Creating a local source physical file

In 7.3.1, “Creating the project container” on page 184, we created a local project container to store the contents of the project. We can now create a local source physical file to hold source members. Creating a local source file creates a windows folder of the same name on your local system in the workspace.

To create a local source physical file, follow these steps:

1. Right-click the iSeries project node in the iSeries Project Navigator view and select New → iSeries Source Physical File. The New - iSeries Source Physical File window (Figure 7-5) opens.

2. In the New - iSeries Source Physical File window, complete the following tasks:
   a. Enter or select the project created in 7.3.1, “Creating the project container” on page 184.
   b. In the File name field, enter the name of the source physical file. If this source file does not exist on the host, it is created when any changes are pushed to the host or when a build of the project is submitted.
   c. Click Next.
3. In the New - iSeries Source Physical File Parameters window, specify any additional parameters. We specify a value of 112 in the Record length field and enter some text in the Description field.

4. Click Finish. The source file is created and appears as a child node to the iSeries project in the iSeries Project Navigator view (Figure 7-6).
7.3.3 Creating a local source member

In 7.3.2, “Creating a local source physical file” on page 187, we created a local source physical file. We can now create a local source member in the local source physical file. Creating a local source member creates a file that resides in the windows folder that maps to the source physical file.

To create a local source member, follow these steps:

1. Right-click the iSeries local source physical file in the iSeries Project Navigator view. Select New → iSeries Source Member. The New - iSeries Member window (Figure 7-7) opens.

2. In the New - iSeries Source Member window, complete the following tasks:
   a. In the Member name field, enter a name for the member. If this source member does not exist on the host, it is created when any changes are pushed to the host or when a build of the project is submitted.
   b. In the Source type field, enter or select the source type from the drop-down list.
   c. Click Next.
3. In the Description field, enter a description for the source member.

4. Click Finish. The source member is created and appears as a child node to the local source physical file in the iSeries Project Navigator view (Figure 7-8). The source type of the member maps to the file extension of the local file created.

![Figure 7-8 The local source member as a child node to the local source physical file](image)

5. After the source member is created, the default editor launches and loads the member.

### 7.4 Importing and viewing remote objects

In the previous section, we created local source files and members in the iSeries project container from scratch. Existing iSeries host objects can also be imported into an iSeries project. You may want to do this for a “maintenance” type of project for example.

**Tip:** You do not need to create an empty source physical file or member if you want to create a local copy of an existing source physical file or member. Simply perform the “import” action as explained in the following section. The local copy of all selected files or members are created in the iSeries project.

**Important:** If you import a source file or member from a library different than the library associated with the project, then, if you make any changes to the copy in the project and push changes the changes are not pushed back to the original member. Instead, they are pushed to the associated library.

For example: If you import member abc from library A into an iSeries project. The iSeries project is associated with library B. If you push changes to the member from the project they will be written to member abc in library B not in library A. In order to push changes back to library A you would need to create an iSeries project associated with that library.

To import remote host objects, follow these steps:

1. Right-click the project in the iSeries Project Navigator view and select **Import Remote Objects**.

2. In the Import Remote Objects window (Figure 7-9), expand the appropriate library filters to drill down to the source files and members you want to import. For more information about filters and how to create and use them, see Chapter 5, “Managing your iSeries development resources using RSE” on page 75.
3. Select the source files or members you want to import. You can select source members by first expanding the appropriate source file node. If you select specific source members to import, a local folder is created that maps to the source physical file in which the member or members reside on the host. You can make multiple selections by using standard Windows keyboard/mouse combinations such as Ctrl+Click and Shift+Click.

4. Click **OK**. The source files or members are downloaded to the local system. The tree in the iSeries Project Navigator view is updated with the new information.

![The Import Remote Objects window](image)

**Figure 7-9**  The Import Remote Objects window

**Tip:** You can also add remote source physical files and members to iSeries projects from the Remote Systems view:

1. Right-click the source physical file or member and select **Add to iSeries Project**.
2. In the Add to iSeries Project window, select the project.
3. Click **OK**.

### 7.4.2 Viewing remote objects

In the iSeries Project Navigator view, you can display both local and remote objects at the same time.

To view remote objects, follow these steps:

1. Right-click the project node in the iSeries Project Navigator view.
2. Select **Show Remote Objects**. This action displays all the remote objects that reside in the remote library specified as the associated library of the project.
The distinction between local and remote objects is made clear through the use of *decorators*. Decorators are small arrows on the icons for each source file and member (see Figure 7-10):

- ➤ : This decorator indicates that the resource exists only locally.
- ➤ : This decorator indicates that the resource exists only remotely, in the iSeries associated library.
- ➤ : This decorator indicates that the resource exists both locally and remotely.

![Figure 7-10  Showing local and remote resources](image)

### 7.5 Working with local source members

This section discusses a cross-section of tasks that can be performed on local source members.

#### 7.5.1 General management

You can copy, delete, move, and rename local source members. To gain access to these tasks, right-click the source member and select the appropriate action from the pop-up menu.

#### 7.5.2 Editing

All that has been said about editing remote source members in the RSE also applies to local source members. For more information about the LPEX editor and editing source members, see 6.1, “Editing source members in the RSE” on page 140.
7.5.3 Source verify (offline)

As stated in 7.1, “What the iSeries Projects perspective is” on page 182, the primary focus of the iSeries Projects perspective is on disconnected (or offline) development. With this in mind, you can also perform source verifications offline as well. Information pertaining to your resources are cached locally. This information is used by the source verifier to perform validations on your source code.

To work offline and use the local cache, follow these steps:

1. In the Remote Systems view, expand your connection and right-click the source physical file or member that you want to work with offline.
2. Select Make Available Offline. If an iSeries project does not exist for the library and connection that contains the source physical file or member, a new iSeries project is automatically created. By default, the project name will be iSeriesConnection_LibraryName. Since the project has been created using the library and connection values, the next time that the code is pushed to the server from the project, it will be pushed to the original library that it came from. If one or more iSeries projects are found, you will be asked to use an existing project or create a new one.
3. To see the iSeries project, select Window → Open Perspective → Other. In the Select Perspective list, select iSeries Projects.

To verify your local source members, follow these steps:

1. Double-click the source member to open it in the LPEX editor.
2. From the workbench menu, select Source → Verify or Source → Verify with Prompt.

Putting your connection into offline mode

You can also put an entire iSeries connection, and all the contents defined with that connection, info offline mode. This method of working offline prevents the Remote System Explorer from attempting to connect to the iSeries server if a request is made for remote information.

For example, if you are verifying a local source file or member in an iSeries project, the project would attempt to connect to the server if the required information was not available in the cache. This can be prevented by switching the connection to offline mode. There are two ways to do this:

1. Right-click your iSeries connection and select Work Offline.
2. Right-click the iSeries connection and select Properties. Then select the Work Offline check box in the properties dialog box.

Caching file descriptions

If your source members reference remote files, whether they be data, display, or printer files, you must cache information for these remote files and for offline source verification to work.

If you verify once while connected then all required external descriptions are automatically cached. If you want to update the cache or cache file descriptions for files you think you might need (but have not used them recently) then you can perform the following steps:

1. In the Remote Systems view of the RSE, drill down through your filters and find the file (*FILE) object or object that you want to cache.
2. Right-click the file object and select Cache File Descriptions. You must perform this action on the actual compiled file object, and not the associated source member. See Figure 7-11.
3. A Progress Information window opens. It indicates that information about the file object is being cached to the local system.

Figure 7-11 Choosing Cache File Descriptions for compiled file object

7.6 Remote actions

This section explains the processes for uploading (or pushing) an iSeries project to the remote host and building your project on the host. The actions associated with these processes are invoked from the Remote Actions menu option.

To access this menu option, follow these steps:

1. Right-click the project, any project resource, or anywhere in the iSeries Project Navigator view.
2. Select Remote Actions.

7.6.1 Build styles

A build style is a set of actions that provide push and build operations for your iSeries projects from the Remote Actions menu. You select one of the styles while creating a new iSeries Project. There are three IBM-supplied build styles:

- **NONE**: This build style contributes no actions. It is useful if you want to disable the menu items associated with a build style. This style is always available.
7.6.2 Specifying build styles

Each iSeries project has its own build style. Two types of active build styles are available:

- **Command**: This style provides operations to push (upload) resources of the project to the associated library. It also provides an operation to run a command you supply in a batch job. The command you supply can be any command. However, it is expected to take the source in the associated library and create programs, service programs, and files from it.

- **CL program**: This style also provides operations to push (upload) resources to the project's associated library. It compiles and runs two CL programs:
  - **COMPILE**: This program compiles source members into programs or modules. This program is created from the COMPILE.CLE source member in the source file specified in the configuration of the build style. This source member can be generated automatically or created and maintained manually. See the following section “Specifying a build style at the project level”.
  - **BIND**: This program creates programs and service programs after the COMPILE program has run successfully. This program is created from the BIND.CLE source member in the source file specified in the configuration of the build style. If you require this source member, then you must create and maintain it manually.

**Specifying a build style at the project level**

To specify a build style for an iSeries project, follow these steps:

1. Right-click the iSeries project and select Properties.
2. In the left side pane, select iSeries Build.
3. Select Command or CL Program from the list and click Configure.
   - If you select Command, complete these tasks:
     i. In the Build Command field, specify the command that is used to build the project.
     ii. Click OK.
     iii. Click OK.
   - If you select CL Program (see Figure 7-12), complete these tasks:
     i. In the Build Source File field, specify the source file that contains the members that will be uploaded to build the project.
     ii. Select Automatically generate COMPILE.CLE prior to each build to automatically generate the COMPILE.CLE file in the designated source file before you submit the build to the host. Leave this option deselected if you want to manually control the content of the COMPILE.CLE source member.
iii. Select **Automatically push all changed members prior to build** so any changed members are automatically created or updated in the associated remote library prior to the project build. In general, we recommend that you select this option.

iv. Click **OK**.

v. Click **OK**.

---

![Figure 7-12 CL Program Build Style Configuration window](image)

**Specifying a default build style**

You can also specify a default build style that applies to all new iSeries projects:

1. From the workbench drop-down menu, select **Window → Preferences**.
2. In the left side pane, expand **iSeries Projects** and select **Build and compile**.
3. Select your default build style from the list and click **Configure Templates**.
4. Follow the steps specified in “Specifying a build style at the project level” on page 195 for the default build style selected.
5. Click **OK**.

**Important:** Changes made to the default build style do *not* apply to existing iSeries projects. Only new projects inherit the default build style. As discussed in “Specifying a build style at the project level” on page 195, the default build style can be overridden at the project level.

---

### 7.6.3 Push operations

Push operations enable you to upload your project resources to the remote iSeries host. This ensures that the content of your local and remote source members are synchronized. Push operations are activated when either one of the Command or CL Program build styles are specified for the project. See 7.6.2, “Specifying build styles” on page 195.

There are two types of push operations:

- **Push changes**
- **Push selected**

**Push changes**

This operation pushes any *new or changed* resources found in the selected project. An item is changed if it was updated locally since the last time it was pushed by either a Push Changes
or a Push Selected action. The first time it is done, all resources are uploaded to the associated library. Each subsequent upload is incremental.

To invoke the push changes action, follow these steps:
1. Right-click either the iSeries project, any project resource or anywhere in the iSeries Project Navigator view.
2. Select Remote Actions → Push Changes. The Progress Information window opens to indicate the status of the upload of the changed project resources to the iSeries host as shown in Figure 7-13.

![Progress Information](image)

*Figure 7-13  Pushing iSeries project changes to the host*

**Push selected**
This action uploads all selected resources and their children to their corresponding iSeries source files and members in the associated library. It creates any resources that do not exist. It overwrites any changes that were made on the host system only if confirmed by the user. Use this action to force the contents of the associated library to match the workspace.

To invoke the push selected action, follow these steps:
1. Right-click the project or project resources you want to select for this action.
2. Select Remote Actions → Push Selected. The Progress Information window opens to indicate the status of the upload of the selected project resources to the iSeries host as shown in Figure 7-14.

![Progress Information](image)

*Figure 7-14  Pushing iSeries project selections to the host*

**Remote conflict detection**
When a remote source physical file or member is imported into an iSeries project, its last modified timestamp is cached internally for later comparisons. When you select Show Remote Objects at the iSeries Project level or the Push Selected or Push Changes action, you are warned if that object was modified on the remote host.
Conflicts on push operations can happen in two situations:

- There is no locally stored member timestamp for a member and the member exists on the host. This happens if any of the following scenarios occur:
  - The source member was created in the workspace without retrieving it from the host.
  - The member is renamed.
  - The member is copied or moved.
  - The host system is changed in the connection definition.
  - The connection is changed, causing the host system to change.
  - The associated library is changed.
  - The parent source file is renamed, copied, or moved.

- There is a locally stored member timestamp, the member exists on the host, and its timestamp is different than the stored one. This happens if any of the following scenarios occur:
  - The member on the host system is updated after the local source member is pushed.
  - The member on the host system is restored from a backup copy.

If there is a conflict, you are prompted with a Conflict Detected warning window (Figure 7-15):

- If you click Yes, you overwrite the source member on the host.
- If you click Yes to All, all source members that conflict for the duration of this push operation are overwritten without further warning.
- If you click No, this source member is not pushed to the host.
- If you click No to All, all source members that conflict during this particular push operation are skipped without further warning.
- If you click Cancel, the push operation is canceled.

Figure 7-15   Remote conflict detected

7.6.4 Generating compile code

The Generate Compile Code action is applicable to iSeries projects that have a CL Program build style configured. For more information about build styles, see 7.6.2, “Specifying build styles” on page 195.

Invoking this action generates (or overwrite as the case may be) a source member named COMPILE.CLLE in the source file specified in the build style configuration. This source member contains the commands that compile your project source members.

To invoke the Generate Compile Code action, follow these steps:

1. Right-click either the project, any project resource, or anywhere in the iSeries Project Navigator view.
2. Click Remote Actions → Generate Compile Code. The Progress Information window opens to indicate the current status of the action.
3. After the action is completed, the COMPILE.CLE member is created (or re-generated) in the source file specified in the build style configuration. The source file and member appear as child nodes to your project in the iSeries Project Navigator view.

4. Double-click the **COMPILE.CLE** member to view its contents in the default editor. See Figure 7-16.

![Figure 7-16 COMPILE.CLE source member](image)

**Managing COMPILE.CLE**

In many situations, the choice of compile commands used in the COMPILE.CLE source member may not be suitable in the context of your project. This is due to the method that the Generate Compile Code action applies in deciding which compile command to use. The command chosen is based on the *member type* of the project source member only. The last command run to compile a member of that type in the RSE is the command it uses in the COMPILE.CLE source member. For example, in the case of project source members with a type of RPGLE, if the last compile command invoked on a member of type RPGLE in the RSE was Create Bound RPG Program (CRTBNDRPG), then this command is used for all RPGLE project source members in the COMPILE.CLE member.

In many cases, this is not an acceptable scenario. You will, without doubt, have times when you require both bound programs and modules in your project and these objects require two different compile commands. This is the time where you need to manage the content of the COMPILE.CLE source member. We recommend that you refrain from using the Generate Compile Code action. We also recommend that you deselect the Automatically generate COMPILE.CLE prior to each build option in the build style configuration at the project level in this situation. See 7.6.2, “Specifying build styles” on page 195.

The functionality of the Generate Compile Code action was not fully developed at the time of writing this redbook. It is expected that more flexibility will be implemented for this action in a future release of WebSphere Development Studio Client for iSeries.

**ILE-based projects**

For Integrated Language Environment (ILE)-based projects, where modules are bound together to create programs and service programs, you must create a local source member...
named BIND.CLLE. This source member must be stored in the source file specified in the build style configuration. There is no action to create this member for you automatically. The BIND.CLLE member contains the commands that bind your project's module objects together to build the required programs or service programs.

7.6.5 Building your iSeries project

This section explains how to build your iSeries project on the remote iSeries host.

The build process

The high-level tasks performed when you request a build of your iSeries project associated with a CL Program build style can be summarized as follows:

1. Any project resources that changed since the project was last uploaded are pushed to the iSeries host. This only occurs if you selected the Automatically push all changed members prior to build option in the build style configuration. We recommend that you keep this option selected.

2. The build job is submitted to batch on the remote iSeries host. The associated library for the project is made the current library for the submitted job. The rest of the library list is as specified for the connection that is associated with the project.

3. The COMPILE.CLLE and BIND.CLLE source members are compiled to create the COMPILE and BIND programs in the QTEMP library. For more information about these source members, see 7.6.4, “Generating compile code” on page 198.

4. The COMPILE program is run. This program compiles your project's resources.

5. The BIND program is run. This program binds your compiled objects together to create the ILE programs and service programs for your project.

Submitting project builds

To build your iSeries project on the remote iSeries host, follow these steps:

1. Right-click the project, any project resource, or anywhere in the iSeries Project Navigator view.

2. Select Remote Actions → Submit Build. The Progress Information window opens to indicate the current status of the action.

3. After the build job is submitted on the remote host, the status of the job can be monitored from the iSeries Build Status view. See “The iSeries Job Status view” on page 200.

The iSeries Job Status view

This view displays all build jobs that are submitted to the remote host. When you submit a build job for a project using one of the Command or the CL Program build styles, the job is added to the Job Status view as shown in Figure 7-17.
Figure 7-17  The iSeries Build Status view

This view can be set up to update at regular intervals and monitor all iSeries Project build jobs that you submitted. The update interval is set via the refresh slider situated in the bottom left corner of the view. You can specify the refresh time in seconds. It takes some time to gather the information and it can only be gathered if a connection can be established to the iSeries host. Setting too small an interval may have an impact on performance. To set the refresh interval to “never refresh”, move the slider all the way to the right. To update the view, when the refresh interval is inactive, click the Refresh button in the top right corner of the view.

When the job finishes, and compile errors are found during the build, right-click the job and select Retrieve Errors to populate the iSeries Errors List view with the compile errors found. From the iSeries Errors List view, you can double-click an error and load the source member with the error or errors into the default editor and have the cursor positioned at the error. You can also remove jobs that finished from the iSeries Build Status View list. Simply right-click the job and select Remove. To end jobs that are queued or running, right-click the job and select Cancel. See Figure 7-18.

Figure 7-18  Invoking actions from the iSeries Build Status view

7.7  Working disconnected actions

This section summarizes the actions you can do and the actions you cannot do while working disconnected.

7.7.1  Actions applicable

The following list of activities can be performed off-line:

- Edit local and offline source files / members
- Syntax check RPG, COBOL, DDS
- Syntax check CL for cached commands
- Prompt RPG and DDS
- Prompt CL for cached commands
- Access language help for RPG, COBOL, DDS
- Access CL help for cached commands
- Verify RPG, COBOL, DDS for those members that are cached

### 7.7.2 Actions not applicable

You can perform the following actions only if you are connected to your iSeries server:

- Edit, design or compile host source members
- Run or debug host programs
- Run host commands
- Syntax check, prompt or F1 on CL commands not cached
- Verify members not previously verified
Part 3 covers a typical process of converting the iSeries host applications with the IBM WebFacing Tool. It explores some advanced features of the IBM WebFacing Tool and other Web development tools. These features help developers to customize the appearance of the converted application.

This part also shows the new Web diagrams feature. This is a visual development environment that allows you to create a visual diagram with application objects, such as JavaServer Pages (JSP) files, connections between objects, and actions. Ideally, you can create a prototype of your application and discuss the application logic without creating any source code. When you are satisfied with the design of the application, WebSphere Development Studio Client for iSeries generates the templates for all objects that are part of the diagram. Then, you add the logic to the generated objects.
Chapter 8. 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. These help to extend many aspects of an enterprise to implement superior business-to-business (B2B) and business-to-consumer (B2C) solutions.

Initially the term *WebSphere* was only used as the name for the IBM application server. It has 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 server 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 server. 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.
8.1 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.

8.1.1 What a Web server is

The primary software backbone of a Web presence is the Hypertext Transfer Protocol (HTTP) server, which is commonly known as a Web server. It is a Transmission Control Protocol/Internet Protocol (TCP/IP) application similar in concept to a network file server.

The key uses of the HTTP server are:

- **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.

HTML 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 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. They provide the dynamic data and transaction capabilities of the customer’s 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), which is no longer supported in i5/OS V5R3 or later.
- 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.

### 8.2 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.

#### 8.2.1 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®

**Common Gateway Interface**

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 Integrated Language Environment (ILE) RPG, ILE COBOL, ILE C, or Java.

**Understanding CGI processing**

The basic idea of CGI is based on the fact that the incoming Uniform Resource Locator (URL) from a browser may specify the name of an HTML file to be served and 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. In turn, it has to deliver the HTML stream to be formatted and displayed by the browser.

The following steps outline 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 `QzhbCgiParse` 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 Structured Query Language (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.

**Display device:** 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.

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 created by a person (a Web designer) and stored without modification in a text file. Under OS/400, such a text file typically resides as a stream file in a directory of the 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 AS/400 Programming*, GC41-5435, for more information about 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.

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

The most difficult part of the RPG CGI program is the code that parses the buffer returned from the QzhbCgiParse API. Again, after 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). It is shipped with OS/400 at no additional charge.

More information about CGI
You can find more information about 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, SG24-4835

Also, refer to Chapter 6 of HTTP Server (powered by Apache): An Integrated Solution for IBM @server iSeries Servers, SG24-6716, for more information about 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 AS/400 Programming, GC41-5435.

For more information, see the following Web site for sample CGI programs:
http://www.ibm.com/eserver/iseries/software/http/examples/

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 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 in depth here:

1. A Net.Data macro is requested on an incoming URL. The request is sent through the URL to IBM HTTP Server for iSeries.
2. IBM HTTP Server for iSeries 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 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 that describe 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 IBM HTTP Server for iSeries.
9. The resulting HTML is sent back to the browser by 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. That is if your primary goal is to complete the project without serious investment on internal or external programming skills acquisition.

More information about Net.Data
You can find more information about 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 about 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 can access or download the Net.Data manuals from the Net.Data home page at:

8.2.2 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 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. After the Java program is compiled into bytecodes on any platform that has a Java compiler, these bytecodes can run on any implementation of the JVM. 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. It 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 statement 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 applet process follows 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. After 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. After 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 browser uses 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 “Common Gateway Interface” on page 207), 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. It 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 208 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 provides 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 8-1):

- **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 (GET, 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.

Servlets are managed by the application server (see 8.3.2, “The purpose and use of the WebSphere Application Server” on page 220). They are loaded in memory upon the first client request or at server startup. Each client request is then served on a different thread.

![Figure 8-1  The servlet's life cycle](image)

**Comparing servlets and applets**

Java servlets have many advantages over Java applets for several 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 iSeries Servers*, SG24-6716.

If you are already familiar with CGI programming, covered earlier in “Common Gateway Interface” on page 207, 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 you click a Submit button. The request is sent, using the HTTP protocol, to HTTP Server for iSeries, which passes the request to the *application server*.

2. **The application server identifies the servlet and invokes it.**
   - See 8.3.2, “The purpose and use of the WebSphere Application Server” on page 220, for information about 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 `HttpServletRequest` 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 you retrieve 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 provides 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. You use 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 the 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. 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 “Common Gateway Interface” on page 207) and CGI (see “Net.Data: A scripting language” on page 209) 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 obtaining the input data from the Web form, parsing the data into discrete field and value pairs, and writing response HTML 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.

**JavaBeans**

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 JavaBeans 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 JavaBeans. 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.

JavaBeans 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.


**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 challenges:

- Due to the fact that Web sites need to be attractive to the user, it becomes more common that the design of the visual appearance of a Web page is done by a person (or department) different than 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 JavaBeans) more practically.

The servlet and JSP 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 JSP 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 JavaBeans 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. After 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 about EJBs in *EJB 2.0 Development with WebSphere Studio
Application Developer*, SG24-6819.

### 8.3 Developing and deploying Web applications

The previous sections describe the building blocks that you can use to create Web pages that
present static information about a Web browser, including dynamically generated data, and
that 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 you can use WebSphere
Application Server to deploy and manage it.

#### 8.3.1 What a Web application is

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 8.2, “Adding dynamic content to Web sites” on page 207, 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 8.2.2, “Java-based technologies” on page 211.

Whenever we refer to a Web application in this book, we actually mean those applications based on Java standards.

#### 8.3.2 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 can 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 Platform, 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: This is 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, JSP 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 fifth 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. The WebSphere V5 editions available on iSeries are:

- Express Edition provides run-time support for the servlets and JSPs (no EJB support).
- Base Edition adds support for EJBs to the capabilities of the Express Edition.
- Network Deployment Edition allows you to manage multiple servers with support for advanced features, Workload Management (WLM), failover support, and scalability.
**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 EJBs and only need to deploy a single application server, you may start with the WebSphere Application Server - Express for iSeries.

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 4 and 5 are currently supported (see Table 8-1 for a list of the currently supported versions and editions of WebSphere Application Server for iSeries).

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 8-1 lists the appropriate PTF numbers for each version.

To help you identify the modification level of the WebSphere Application Server on your iSeries server, you perform the following actions depending on your version and release:

- In V5R1, a data area with the same name as the PTF is stored in library QEJBQDV4 or QJEBAES4 for WebSphere Application Server Version 4. For Version 5, it is stored in the library QEJBAS5, with the date and modification level of the product.

  **Note:** The Display PTF (DSPPTF) command is not meaningful for 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 QEJBAS5/*ALL
```

  **Note:** WebSphere Application Server Version 4 and Version 5 may be installed on the same iSeries server. Therefore, you may want to enter both commands.

You see the Work with Data Areas display, 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 the Display Data Area display like the example in Figure 8-2.

```
Display Data Area
System: MYISERIES

Data area . . . . . . . : SF99241
Library . . . . . . . : QEJBADV4
Type . . . . . . . . . : *CHAR
Length . . . . . . . . : 50
Text . . . . . . . . . :

<table>
<thead>
<tr>
<th>Offset</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>'Group PTF#: SF99241-02 5733WA4 V5R1 02/01/02 4.0.2'</td>
</tr>
</tbody>
</table>
```

**Figure 8-2** Modification level of WebSphere Application Server after applying a group PTF
The content of the data area shown at the bottom of the display indicates that the second revision of PTF SF99241, dated 01 February 2002, was applied to product 5733-WA4, making it modification level 4.0.2.

In V5R2, use the Work with PTF Groups (WRKPTFGRP) command to check to the modification level, for example:

```
WRKPTFGRP PTFGRP(SF99245)
```

Table 8-1 helps you identify the various editions and versions of the WebSphere Application Server that are currently available for the releases of OS/400. It 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® 57xx-JV1</th>
<th>Prerequisite CUM Pack</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Version</strong></td>
<td><strong>Edition</strong></td>
<td><strong>Product number</strong></td>
<td><strong>Group PTF</strong></td>
</tr>
<tr>
<td>4.0</td>
<td>Advanced</td>
<td>5733-WA4</td>
<td>SF99239</td>
</tr>
<tr>
<td>4.0</td>
<td>Advanced Single Server</td>
<td>5733-WS4</td>
<td>SF99240</td>
</tr>
<tr>
<td>4.0</td>
<td>Advanced</td>
<td>5733-WA4</td>
<td>SF99241</td>
</tr>
<tr>
<td>4.0</td>
<td>Advanced Single Server</td>
<td>5733-WS4</td>
<td>SF99242</td>
</tr>
<tr>
<td>4.0</td>
<td>Advanced</td>
<td>5733-WA4</td>
<td>SF99148</td>
</tr>
<tr>
<td>4.0</td>
<td>Advanced Single Server</td>
<td>5733-WS4</td>
<td>SF99149</td>
</tr>
<tr>
<td>5.0</td>
<td>Express</td>
<td>5722-IWE</td>
<td>SF99270</td>
</tr>
<tr>
<td>5.0</td>
<td>Base</td>
<td>5733-WS5</td>
<td>SF99243</td>
</tr>
<tr>
<td>5.0</td>
<td>Network Deployment</td>
<td>5733-WS5</td>
<td>SF99244</td>
</tr>
<tr>
<td>5.0</td>
<td>Express</td>
<td>5722-IWE</td>
<td>SF99271</td>
</tr>
<tr>
<td>5.0</td>
<td>Base</td>
<td>5733-WS5</td>
<td>SF99245</td>
</tr>
<tr>
<td>5.0</td>
<td>Network Deployment</td>
<td>5733-WS5</td>
<td>SF99246</td>
</tr>
<tr>
<td>5.1</td>
<td>Express</td>
<td>5722-E51</td>
<td>SF99273</td>
</tr>
<tr>
<td>5.1</td>
<td>Base Edition</td>
<td>5733-W51</td>
<td>SF99276</td>
</tr>
<tr>
<td>5.1</td>
<td>Network Deployment</td>
<td>5733-W51</td>
<td>SF99278</td>
</tr>
<tr>
<td>5.1</td>
<td>Express</td>
<td>5722-E51</td>
<td>SF99274</td>
</tr>
<tr>
<td>5.1</td>
<td>Base Edition</td>
<td>5733-W51</td>
<td>SF99277</td>
</tr>
<tr>
<td>5.1</td>
<td>Network Deployment</td>
<td>5733-W51</td>
<td>SF99279</td>
</tr>
</tbody>
</table>
Table 8-2 shows the group PTFs for the various editions of WebSphere Application Server for i5/OS V5R3.

Table 8-2  WebSphere Application Server releases and group PTFs for i5/OS V5R3

<table>
<thead>
<tr>
<th>WebSphere Application Server for iSeries</th>
<th>i5/OS</th>
<th>JDK&lt;sup&gt;a&lt;/sup&gt;</th>
<th>Prerequisite CUM Pack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Edition</td>
<td>Product number</td>
<td>Group PTF</td>
</tr>
<tr>
<td>4.0</td>
<td>Advanced</td>
<td>5733-WA4</td>
<td>SF99290</td>
</tr>
<tr>
<td>4.0</td>
<td>Advanced Single Server</td>
<td>5733-WS4</td>
<td>SF99289</td>
</tr>
<tr>
<td>5.0</td>
<td>Express</td>
<td>5722-IWE</td>
<td>SF99272</td>
</tr>
<tr>
<td>5.0</td>
<td>Base Edition</td>
<td>5733-WSS</td>
<td>SF99287</td>
</tr>
<tr>
<td>5.0</td>
<td>Network Deployment</td>
<td>5733-WSS</td>
<td>SF99288</td>
</tr>
<tr>
<td>5.1</td>
<td>Express</td>
<td>5722-E51</td>
<td>SF99275</td>
</tr>
<tr>
<td>5.1</td>
<td>Base Edition</td>
<td>5733-W51</td>
<td>SF99285</td>
</tr>
<tr>
<td>5.1</td>
<td>Network Deployment</td>
<td>5733-W51</td>
<td>SF99286</td>
</tr>
</tbody>
</table>

a. IBM Developer Kit for Java (JDK) is available as 5722-JV1 for OS/400 V5R1 and later 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.

b. This is the level of the group PTF at the time of writing this redbook. Note, that you always receive the latest level of

For the latest information regarding the group PTFs you should always check the Preventative Service Planning - PSP Web page at:


8.4 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 8-3, it supports several distinct, yet interoperable, application environments that share common system services:

- **Java**: Includes a JVM 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**: Supports several programming technologies (RPG, COBOL, and C/C++).

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

---

**Figure 8-3** *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. Together, these qualities make the iSeries an excellent choice for extending existing applications and deploying new solutions.

Our focus 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.
8.5 The IBM suite of host and 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 8-4 to help you see the big picture of how all these products inter-relate. For example, look at the very bottom of Figure 8-4 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. It brings a certain level of independence from the client, but raises the minimum system size (Commercial Processing Workload (CPW) and memory) for your iSeries server.

![Figure 8-4 Comparison of IBM host and server access products](image-url)
8.5.1 Methods to combine new Web applications with existing applications

In theory, the most modern, cleanest, and conceptually simplest approach to build a Web application is to design a completely new application using object-oriented technologies and methods. However, in most cases, there are many reasons why you should not 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 “Common Gateway Interface” on page 207) or Java servlets (see “Using servlets to run Java code on the Web server” on page 213) 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
- **IBM 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 displays as a data source
- **Host Access Transformation Server (HATS)**: Makes 5250 and 3270 applications available through the Web browser while converting your host screens to a Web-like look and feel

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 8-5. IBM has products to provide the solution for each step of the way.

The stages in Figure 8-5 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 with 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
- IBM WebFacing Tool
- Host Publisher
- HATS
- 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 @server 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

  An example of this is IBM Personal Communications, which 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. 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 displays. 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 IBM 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 must be started and run under the control of a Web browser.

For more information about Host On-Demand, see 8.6.1, “Host On-Demand” on page 231. Also see the following resources:

- IBM Web-to-Host Integration Solutions, SG24-5237 (Chapter 2)
- IBM Host Access Client Package Update, SG24-6182 (Part 1)
- The WebSphere Host On-Demand Web site at: http://ibm.com/software/webservers/hostondemand

WebSphere Host Access Transformation Server (HATS) gives you all the tools you need to quickly and easily extend your legacy applications to business partners, customers, and employees. HATS makes your 3270 and 5250 applications available through the most popular Web browsers, while converting your host screens to a Web-like look and feel. HATS enables you to extend your host application to the Web within a single day of installing the software without any changes to your host. HATS is a zero footprint Web-to-host solution. The only software needed on the client is a Web browser.

For more information, see:

- 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.

For more information about 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 environments. 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 and 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 8.6.3, “Host Publisher” on page 233. 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 Host Access Client Package Update*, SG24-6182 (Part 3)
  - Screen Customizer Web site at:
  http://www.ibm.com/software/network/screencustomizer

- **IBM 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 IBM 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 IBM 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 Development Studio Client for iSeries.

  The IBM WebFacing Tool is a 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 needing to see the source of the display panels). As opposed to Host On-Demand, WebFacing requires a development time investment.

  The IBM 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 IBM WebFacing Tool, see 8.6.2, “IBM WebFacing Tool” on page 232, and Part 3, “Bringing your OS/400 applications to the Web” on page 203. Also refer to the WebSphere Development Studio Client for iSeries Web page at:

http://www.ibm.com/software/awdtools/wdt400/about/webfacing.html


8.6 Web-to-host integration tools comparison

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

8.6.1 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 8-6 shows an overview of Host On-Demand.

![Host On-Demand overview](image)

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:

- *IBM Web-to-Host Integration Solutions, SG24-5237 (Chapter 2)*
- *IBM Host Access Client Package Update, SG24-6182 (Part 1)*
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/](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 and later, 5769-JV1 for previous releases)
- IBM Toolbox for Java (5722-JC1 for OS/400 V5R1 and later, 5769-JC1 for previous releases)
- Qshell Interpreter (5722-SS1 for OS/400 V5R1 and later, 5769-SS1 for previous releases): Option 30 recommended
- TCP/IP Connectivity Utilities for iSeries (5722-TC1 for OS/400 V5R1 and later, 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

8.6.2 IBM WebFacing Tool
The IBM WebFacing Tool is not a product in itself, but rather is included with WebSphere Development Studio Client for iSeries (WDSc), which comes as part of WebSphere Development Studio for iSeries (5722-WDS). It comes also with WebSphere Development Studio Client Advanced Edition for iSeries, as separately marketed product. You can use the IBM 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 JavaBeans 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 are 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 IBM WebFacing Tool again and select a new style.
When using the IBM 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 higher
- 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)
- Enable file sharing via NetServer™ so your Web application contents can be moved to the iSeries server (or publish via FTP)

**Note:** You can install your business application on an iSeries server. You can also install both the HTTP server and the WebSphere Application Server server on a Windows NT, UNIX, or another iSeries server.

**Client machine requirements**
Before you can use the IBM 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 IBM WebFacing Tool run-time development
  - Internet Explorer V5.0 or Netscape V4.7: For viewing online help in the development environment

### 8.6.3 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 JSP 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, no specific requirements are placed on the client Web browsers. Therefore, it is appropriate for Internet, intranet, and extranet use.

You can find additional information about WebSphere Host Publisher on the Web at:
http://www.ibm.com/software/webservers/hostpublisher/

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

**IBM WebFacing Tool needs DDS source code**
Both tools interface with existing applications that perform input and output operations from and to character-based display stations. Also both products allow 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 JavaBeans, so-called Host Access Objects, created by Host Publisher. These JavaBeans 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 JavaBean can also accept data that 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. Only a buffer area is accessed by a JavaBean. Likewise, no real keyboard is attached to the emulation program, but keystrokes are generated by a JavaBean.
The server part of the IBM 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 generated, WebFacing presents the data to the JavaBean, which then returns it to the Web browser in HTML format. Likewise input data from the browser is delivered to Workstation Data Management as though it comes from a real or emulated 5250 device.

To create an application with the IBM 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 present an OS/400 display, for example the Work with Output Queue (WRKOUTQ) command
- A 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 IBM WebFacing Tool, but you can Web-enable them with Host Publisher.

**The IBM WebFacing Tool creates functioning code fully, automatically**

Provided no unsupported keywords are used for the DDS, the IBM WebFacing Tool creates all necessary objects (HTML, JSPs, JavaBeans, 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 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 must “teach” the Host Integration Object (a JavaBean that is created during this process) how a terminal user runs 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 JavaBeans.

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, the IBM WebFacing Tool always relies on accessing the data presented on the 5250 screen by the interactive application.
**Combining the contents of multiple panels**

For the IBM WebFacing Tool, 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. 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, the IBM WebFacing Tool presents the contents of all output fields to the browser user. However, you can use CODE Designer to hide selected fields for the Web.

**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 IBM 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 IBM 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, the IBM WebFacing Tool is now a component of 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 15, for more information about the WebSphere Development Studio Client for iSeries.
8.6.4 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 that contain 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, 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:

- Windows 95, 98, NT, or 2000
- A Web browser:
  - For development time: Microsoft Internet Explorer 4.0 or later
  - For run time: Netscape 4.7 or later, or Microsoft Internet Explorer 4.0 or later

8.6.5 Comparison table

Table 8-3 compares the solutions that are discussed in this section for converting existing iSeries host applications to Web-based applications.

<table>
<thead>
<tr>
<th>Feature</th>
<th>IBM WebFacing Tool</th>
<th>iSeries Access for Web</th>
<th>Host On-Demand</th>
<th>Host Publisher</th>
<th>HATS LE</th>
<th>HATS Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manage printers, spooled files, job information, messages, and files</td>
<td>N. Use iSeries Access for Web links as a workaround.</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y. Only via a 5250 interface</td>
<td>Y. Only via a 5250 interface</td>
</tr>
<tr>
<td>Bidirectional file transfer support</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Provides GUI interface to 5250 applications</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Can create new Java, HTML, and XML application function</td>
<td>Y</td>
<td>N. Has APIs and classes that can create an application interface</td>
<td>Y. Has to use Screen Customizer</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Feature</td>
<td>IBM WebFacing Tool</td>
<td>iSeries Access for Web</td>
<td>Host On-Demand</td>
<td>Host Publisher</td>
<td>HATS LE</td>
<td>HATS Full</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------</td>
<td>----------------</td>
<td>---------------</td>
<td>-------------------------------------------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>Requires Java application server; WebSphere is recommended to be eligible for IBM support</td>
<td>Requires application server can run on iSeries or xSeries or Linux environment</td>
<td>Requires application server can run on iSeries or xSeries or Linux environment</td>
<td>N</td>
<td>Y</td>
<td>Y. Application server can run on iSeries or xSeries</td>
<td>Y. Application server can run on iSeries or xSeries</td>
</tr>
<tr>
<td>Create interactive 5250 workloads</td>
<td>Y. Except for some iSeries models</td>
<td>Y. Only 5250 components</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Requires DDS source codes for display file to be converted</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N. Screen content is traced by Host Publisher Studio</td>
<td>N. Only when customization is required</td>
</tr>
<tr>
<td>Convert iSeries system screens and menus</td>
<td>N. Limited support for selected commands in WebSphere Development Studio Client (WDS) V5.0. Use iSeries Navigator, iSeries Access for Web or 5250 interfaces for system functions.</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Browser support Microsoft Internet Explorer 5.0</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Browser support Netscape 4.7 (AIX, Linux)</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Browser support Netscape 6.2 (Windows)</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y. HOD 8 may not need a browser</td>
<td>Y</td>
</tr>
<tr>
<td>Browser support Opera 5.0 (Linux)</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N. HOD 8 may not require a browser</td>
<td>N</td>
</tr>
<tr>
<td>Browser support Opera 6.0 (Windows)</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N. HOD 8 may not require a browser</td>
<td>N</td>
</tr>
<tr>
<td>SSL support</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Interface Portal ready</td>
<td>N. Can use iFrame portlet</td>
<td>N. Can use iFrame portlet</td>
<td>Y</td>
<td>Y</td>
<td>Y. Uses HOD portlet</td>
<td>N. Can use iFrame portlet</td>
</tr>
<tr>
<td>Modify screen flow</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y. Uses Screen Customizer</td>
<td>Y</td>
</tr>
</tbody>
</table>

---

3. Y. Except for some iSeries models
4. Requires DDS source codes for display file to be converted
5. N. Limited support for selected commands in WebSphere Development Studio Client (WDS) V5.0. Use iSeries Navigator, iSeries Access for Web or 5250 interfaces for system functions.
6. Interface Portal ready
7. Modify screen flow
<table>
<thead>
<tr>
<th>Feature</th>
<th>IBM WebFacing Tool</th>
<th>iSeries Access for Web</th>
<th>Host On-Demand</th>
<th>Host Publisher</th>
<th>HATS LE</th>
<th>HATS Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rules-based engine to convert screens 8</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N. Limited function available via Screen Customizer</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Requires WebSphere Studio based tool on the client PC</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Browser print support 9</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Local print support 10</td>
<td>N. Partial print support in WDSv5. Can use iSeries Access for Web link to access print function.</td>
<td>Y. Can convert legacy print to PDF</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Allows printer sharing</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Basic keypad support including function keys for application navigation</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>User can define specific workstation ID 11</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Customizable user profile based restrictions for user sessions 12</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>5250 support</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>3270, VT100 and VT220 support</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>SQL database access 13</td>
<td>N</td>
<td>Y</td>
<td>Y. Uses Database On-Demand as the SQL tool</td>
<td>N</td>
<td>N. Only through a 5250 interface</td>
<td>N. Only through a 5250 interface</td>
</tr>
<tr>
<td>Access to system messages</td>
<td>N. Uses a URL link to iSeries Access for Web</td>
<td>Y. Only through the 5250 interface</td>
<td>Y</td>
<td>Y. Only through 5250 interface</td>
<td>Y. Only through 5250 interface</td>
<td>Y. Only through 5250 interface</td>
</tr>
<tr>
<td>Merge or split application 14</td>
<td>Y. May require programming changes</td>
<td>N</td>
<td>Y. Can be done using Screen Customizer</td>
<td>Y. Handled by macro recorder</td>
<td>N</td>
<td>Y. Handled by macro recorder</td>
</tr>
<tr>
<td>Eligible for SW maintenance or SW subscription</td>
<td>Y</td>
<td>Y</td>
<td>Y. Via CSD</td>
<td>Y. Via CSD</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Feature</td>
<td>IBM WebFacing Tool</td>
<td>iSeries Access for Web</td>
<td>Host On-Demand</td>
<td>Host Publisher</td>
<td>HATS LE</td>
<td>HATS Full</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>--------------------</td>
<td>------------------------</td>
<td>----------------</td>
<td>---------------</td>
<td>---------</td>
<td>-----------</td>
</tr>
<tr>
<td>Support for Subfile rendering&lt;sup&gt;15&lt;/sup&gt;</td>
<td>Y</td>
<td>Y. Only through 5250 interface</td>
<td>N. Rendering is possible via Screen Customizer</td>
<td>Y. Uses a data macro but requires program. changes</td>
<td>Y. With some restrictions</td>
<td>Y. With some restrictions</td>
</tr>
<tr>
<td>Store and retrieve data in global variables</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Enter data on behalf of end users</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Convert text entry fields to valid value lists</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Create tabbed folders (using wizards)</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Create bar and line graphs (using wizards)</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Redirect users to other URLs</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Globally replace text</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Add business logic</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

1. Bi-directional file transfer support allows users to move files to and from the server.
2. This feature extends the user interface or application function to the client or browser.
3. WebFacing applications do not consume "interactive CPU utilization" on Models 800, 810, 825, 850, and 890. Depending on workload intensity, all other refacing tools require the higher cost Interactive Feature on older models or the Enterprise Edition on Models 800, 810, 825, 850, and 890. WebFacing applications running on older iSeries models also require the higher cost Interactive Feature.
4. The DDS files are used to create JSPs. Some programming may be required for display file conversion.
5. This feature brings the standard 5250 interfaces for system functions to the client or browser.
6. This feature gives the user the ability to install the application as a WebSphere Portal Server portlet.
7. Allows the user to make the interface more efficient by having less screen changes.
8. Screen conversions are configured based on predefined rules.
9. Allows the user to print the pages using the browser's print function.
10. Local print support allows the user to use PRINT function embedded in the application programs.
11. Assists the user in system management of functions performed by specific workstations.
12. Enables customization of user profiles preferences to allow or deny them access to certain functions in their browser sessions.
13. Allows the user to access SQL databases through the browser and not limited to the use of 5250 interface alone.
14. This feature allows the user to bypass or skip certain functions, modules or screens of application programs so they can jump directly to the other functions, modules, or screens.
15. The ability of the product to recognize and render 5250 list of records into Web pages.
Web Development Tools in WDSc

Most chapters in this redbook deal with functions specifically designed to aid programmers developing applications to run under OS/400 or i5/OS. This also includes Web applications which do have a platform independent “view” and “controller” part, whereas the “model” contains OS/400 based programs.

This part of the book describes how you can create Web applications with WebSphere Development Studio Client for iSeries (WDSc) and WebSphere Development Studio Client for iSeries Advanced Edition (WDSc AE). As stated above, many Web applications contain parts which are not specific to iSeries. This is the reason, why WDSc comes with WebSphere Studio Site Developer (WSSD) and WDSc AE contains WebSphere Studio Application Developer (WSAD) as we describe in 2.2.1, “WebSphere Studio family” on page 20.

Both products, WSSD and WSAD, contain an extensive set of Web Development Tools. Although there are other publications describing these products in depth, we felt it would be useful to provide an overview on the Web Development Tools in this chapter.

More in-depth information can be found in the Redbooks:

- \textit{WebSphere Studio Application Developer Version 5 Programming Guide}, SG24-6957
- \textit{WebSphere Studio 5.1.2 JavaServer Faces and Service Data Objects}, SG24-6361
- \textit{WebSphere Version 5.1 Application Developer 5.1.1 Web Services Handbook}, SG24-6891

You can also find more detailed information in the Online Help documentation of WDSc. Most of the material provided in this chapter can be found in the Online Help under Developing $\rightarrow$ Web applications $\rightarrow$ Web development tools.

\textit{Changes in the second edition of this redbook}

This chapter is completely new. Therefore no lines of text are marked as being new in this chapter.
9.1 Tools for Web development

WebSphere Studio is a development environment designed to make Web application creation simple and efficient. The Web tools that are part of WebSphere Studio allow you to work with Web technologies to create Web applications that range from simple, static Web sites, to fully dynamic Web applications that access data sources.

Web perspective

The Web perspective contains a selection of views and editors that are customized to be most useful to a Web developer. These views include reusable drag and drop components, link structures, access to page data, your project resources, code snippets, image galleries, and more. The editors range from standard source editors with content assist features, to Web Site Designer and Page Designer, editors that are fully-featured to help create, edit, and preview Web sites and the pages within those Web sites.

The following tools are the primary means for developing Web applications in the workbench.

Wizards

The wizards are the easiest way to add static or dynamic content to your Web pages. The basic wizards allow you to create and specify settings for Web projects, HTML files, and JSP files, and use the examples provided with the product. The advanced wizards help you retrieve information from common databases, use Java™ bean APIs to query or update databases, and generate output forms and servlets. With these features, you can do everything from a simple table lookup to more complex interactive applications.

The wizards provide detailed graphical user interfaces to guide you through the process of creating SQL queries, choosing Java methods and properties, and integrating them into the Web pages you are creating. You do not have to be an expert at SQL syntax or Java programming, because the wizards walk you through the process step-by-step, and then generate sophisticated JSP and servlet code to run your Web applications.

Web Site Designer

Web Site Designer is a high-level editor that allows you to quickly build complex Web sites. The navigation support enables you to begin your site by designing its organization, and then populating it with the appropriate Web pages. Web Site Designer also enables easy reorganization if you decide to restructure your Web site through a drag and drop site map.

You can toggle between two modes to design your site’s navigation. Web Site Designer has a Navigation view and a Details view to help you plan, organize, and create your overall Web site.

Page Designer

Page Designer is an advanced-function HTML and JSP editor that allows you to quickly build complex Web pages, both visually and textually. The dynamic element support enables you to include form elements, Java applets, embedded scripts, and JavaServer Pages (JSP) tags.

You can toggle among three modes to visually design pages, work with HTML, JavaScript™ or JSP content, and preview your pages. Content assist, which provides guided editing as you insert new tags, is available in the Source (text) page. To help you create the visual impact you want on your Web sites, the editor includes its own library of reusable graphics and two graphic programs for creating, editing, and animating image files.
Web deployment descriptor editor
The Web deployment descriptor editor helps you to define any of the deployment information that can exist in the Web application deployment descriptor included in a WAR file. Pages in this graphical editor provide controls to set numerous Web Application parameters related to servlets, filters, life-cycle listeners, mime mappings, parameters, references, security, and other general deployment settings.

In addition, the Web deployment descriptor editor provides an opportunity to configure WebSphere Application Server-specific bindings and extensions, as appropriate. When a new Web project is created, the generated web.xml file contains an appropriate DOCTYPE declaration. The Sun Microsystems Java Servlet 2.3 Specification includes the DTD for the web.xml file, along with examples.

Content assist
Content assist is a WebSphere Studio source editing tool that prompts you with a list of valid alternatives for completing the current line of code or inserting a macro. Content assist is sensitive to the context of the cursor position. It recognizes whether you are in an HTML area, a JavaScript area, or a JSP area. For HTML, content assist provides an appropriate list of proposals (tags, attributes, and attribute values) based on the context. Content assist can be invoked at anytime by pressing Ctrl+Space or selecting a menu item. You can also set a preference that automatically invokes content assist when special characters, such as ".", ",", or "=" are typed.

Content assist is available within the Page Designer Source page for adding HTML, WML, JSP, and JavaScript tagging. JavaScript content assist is available within a JSP and HTML files, and Java content assist is available within those JSP tags that support Java code, such as JSP scriptlet tags. Variations of content assist are also available within Java and XML source editing areas. See content assist documentation for specific source editors to understand the capabilities and limitations for specific code types.

9.2 The Web perspective
The Web perspective combines views and editors that assist you with Web application development. For example, this is where you typically edit Web project resources, such as HTML and JSP files, and deployment descriptors.

The following views are particularly helpful during Web application development:

Attributes
Provides tabbed pages that allow you to update attributes for tags selected in files open in the active Web editor. Changes to attribute value text fields are reflected in the edited file immediately when cursor focus is changed, or when you press the Enter key. In addition, changes to any of the controls in the Attributes view are immediately reflected in the edited file.

Gallery
A gallery contains a variety of catalogs of reusable files that can be applied to Web pages. The file types available include images, wallpaper, Web art, sound files, and style sheet files.

Thumbnails
Shows thumbnails of the images in the selected project, folder, or file. This view is especially valuable when used with the Gallery view to add images from the artwork libraries supplied by
WebSphere® Studio to your page designs. When used with the Gallery view, thumbnail also displays the contents of a selected folder. You can drag and drop from this view into the Project Navigator view or the Design or Source page of Page Designer.

**Quick Edit**
Allows you to edit small bits of code, including adding and editing actions assigned to tags. You can drag and drop items from the Snippets view into the Quick Edit view.

**Palette**
Contains expandable drawers of drag and drop objects. Allows you to drag objects, such as tables or form buttons, onto the Design or Source page.

**Snippets**
Contains expandable drawers of drag and drop code snippets. For instance, you can add JavaScript™ macros that become part of the user interface (UI), like a script to display the date and time when the page was last updated. If you are working in a JSP file, you can add common JSP code. You can drag and drop code snippets into the Design and Source pages, as well as the Quick Edit view.

**Page Data**
Allows you manage data from a variety of sources such as WebSphere Data Objects, JavaBeans™, and Web Services, which can be configured and dropped onto a JSP page.

**Styles**
Provides guided editing for cascading style sheets and individual style definitions for HTML elements.

**Colors**
Allows you to apply colors from a palette (or custom colors) to selected objects in the editing area.

**Servers**
Lists servers defined for the project and their status.

The other views in this perspective are common throughout the workbench. To add more views, select **Window → Show View** from the menu bar. You can close, resize, or move any of the views.

### 9.2.1 Project Navigator view

The Project Navigator view is customized for Web developers as an alternative to the Resource perspective’s Navigator view. The Project Navigator view is the default project view in the Web perspective.

All operations available within the Resource perspective’s Navigator view are also available in the Project Navigator view through Workbench or context menus. The following notable features are available in the Project Navigator view:

VCM version information can be toggled on and off using from the Preferences page (**Window → Preferences → Workbench → Label decorations**).

You can drag and drop files from Windows Explorer or the desktop into the Project Navigator view.
View filtering is supported by selecting Filters from the Project Navigator view Menu button. Resources can be filtered by name, project type or working set. Files beginning with a period are filtered out by default.

The view root can be changed by selecting Go Into from the project context menu.

The status line shows the full path of the selected resource.

Dragging a .java file from the Project Navigator view into a JSP file will insert a usebean tag, the same behavior that is exhibited when a .class file is dragged into a JSP file.

Errors and warnings on resources (including Java™, HTML/JSP, and Links Builder errors and warnings) are indicated with a red error or yellow warning next to the resource with the error, as well as the parent containers up to the project. This applies for all project types, not only Web projects.

Items available from the New cascading menu in the project context menu are context sensitive. All menus will have Project and Other options.

Organization of the Project Navigator view
The Project Navigator view shows a custom view of the Web project, consisting of the following top-level objects beneath the project node (based on default folder names):

Web Site Navigation - This configuration file only exists if you choose to use the Web Site Designer feature (a default setting) when you create a Web project. Double-click this file to open Web Site Designer, which includes a Navigation view and a Details view.

Web Deployment Descriptor - This file corresponds to the WebContent/WEB-INF/web.xml file. You can double-click to open the file in the deployment descriptor editor, or select Open With from its context menu to open the file with a different editor.

Java Resources - This node displays Java resources within the project. If the project contains a single Java source folder, the packages and classes (for example, servlets, beans) within the source folder will be shown directly beneath the Java Resources folder node. If the project contains multiple source folders, each source folder will appear beneath the Java Resources folder and can be expanded to show their packages and classes. A default folder named Java Resources is created when you create a Web project.

Note: Though the default name given to the folder is Java Resources, you can change the name by right-clicking the folder and selecting Rename in the Project Navigator view or through the Preferences page available by selecting Window → Preferences → Web Tools → New Project.

Libraries - This folder contains the library JAR files defined in the project properties. Three types of JAR files are shown:

- JAR files included in the project's WebContent/WEB-INF/lib directory
- JAR files external to the project, such as j2ee.jar and rt.jar
- Project libraries, which are special references to a Java project

When a Web project is exported, a JAR file is automatically created from the Java project to be used by the Web application during runtime. Libraries are shown in classpath order. By default, only the JAR files contained within the project are shown. You can also display external JARS and project libraries by selecting Show Referenced Libraries from the Project Navigator view’s Menu button.
**imported_classes folder** - This folder can be created during a WAR import, and contains class files that do not have accompanying source. The imported_classes folder is a Java classes folder; Java classes folders can also be created using the Web project Java Build Path properties page.

You can drag and drop class files from the Windows Explorer or desktop to the imported_classes folder in the Project Navigator view.

**Web content folder** - This folder contains items to be published to the server. By default, this folder will be named `WebContent` for newly created static and dynamic Web projects.

Note: Though the default name given to the folder is `WebContent`, you can change the name by right-clicking the folder and selecting Rename in the Project Navigator or from the Web page of the project’s Properties dialog. In a dynamic Web project, changing the folder name will update the Java build output directory. You can change the preference for the default folder name to be applied when creating new Web projects by selecting Window → Preferences → Web Tools → New Project.

**Theme** - The suggested directory for cascading style sheets and other style-related objects.

Non-published folders - Non-published folders contain items that are not published to the server, and include any folders that are not Java source or Web content folders.

### 9.2.2 Attributes view

The *Attributes view* displays specific information pertaining to the currently selected tag in an HTML or JSP file. Sometimes the display has just source code, but at other times it has other information. For example, the Attributes view shows a variety of information about an HTML table tag. Whereas the Source view shows the entire file, the Attributes view shows information about the current tag.

The Attributes view lets you edit JavaScript™ and JSP tags while Page Designer is open in the Design view. When you edit in the Attributes view, your changes take effect when cursor focus is changed.

If you make changes to JavaScript or JSP tags in the Source view of Page Designer, the changes take effect immediately in the Attributes view.

### 9.2.3 Palette view

The Web tools *Palette view* (Window → Show View → Other → Web → Palette) lets you quickly drag and drop items onto a file you are editing. You can use this view when you are editing a variety of files, such as JSP or HTML files, in Page Designer.

The Palette view contains a series of drawers that you can open. Each drawer contains related items that you can drag and drop into the active editor. For example, if you are editing a JSP page in Page Designer, you can open the JavaServer Pages drawer and drag and drop a JSP bean onto the JSP page. The drawers and their contents vary depending on the active editor.

You can customize the Palette view. For example, you can add additional items to the HTML drawer (right-click within the Palette view, choose Customize, and then select the additional HTML items to add). Or, you can increase the size of the icons (right-click within the palette and choose Use Large Icons).
Customizing the palette
The Web tools palette lets you drag an item from a drawer in the palette view and drop the item to the JSP or HTML file that you are editing. Follow these steps to customize the palette:

1. By default the Palette view opens when you are editing a JSP or HTML file in Page Designer. If the palette is not open, click **Window → Show View → Other → Web → Palette**.

2. Right-click inside the Palette view. This open the Customize menu, which provides the following options:
   - Click **Layout** to specify how you want to display the items in the drawer. Usually each item appears as an icon with some text. You have the following layout options:
     - **Columns** - the number of columns depends on the width of the particular view
     - **List** - items appear with minimal description
     - **Icons only** - each item appears as an icon with no text
     - **Details** - descriptions of each item are included
   - Click **Use Large Icons** to increase the size of the icons in each drawer. To restore the original size of the icons, click Use Large Icons again.
   - Click **Customize** to perform any of the following actions:
     - Move items within the drawer
     - Change the names and descriptions of items within the drawer
     - Specify a default drawer to be opened when the Palette view first appears
     - Hide items within a drawer
     - Change attributes for an item within a drawer
   - Click **Settings** to change the following options:
     - Change font settings
     - Change layout settings
     - Set drawer options, such as whether to close a drawer when another is opened
   - Click **Pinned** to keep the selected drawer open.

9.2.4 Snippets view
This documentation gives an overview of the Snippets view.

The Snippets view lets you catalog and organize reusable programming objects, such as HTML tagging, JavaScript™, and JSP code, along with files and custom JSP tags. The view can be extended based on additional objects that you define and include.

To view or collapse the objects in a specific drawer, click the drawer name.

The Snippets view has the following features:

- Drag-and-drop to Page Designer's Design page and various source editing pages: You can drag items from the view into the active editor and the text will be dropped into the document at the cursor location
- Double-click support: You can double-click an item and have it inserted at the current cursor position in the active editor
- User-defined drawers and items: You can define, edit, and remove items from view drawers as desired.
- Plug-in-defined drawers and items: Plug-in developers can contribute a default list of items to their own drawers.
Variables in insertions: By default, items will be edited using a dialog and, when inserted, you will be prompted for values for each of the variables.

Customization: You can select which drawers and items are shown in the Snippets view.

Custom insertion: Plug-in developers can customize the behavior of items so that when they are dropped during a drag-and-drop action, both the text that is inserted and the insertion location are strictly defined.

9.3 Source editors for markup languages

The source editors for markup languages are the editors represented by the Source tab in various editors that you can use to edit files coded with markup tags. Following are examples of the other editors:

- CSS Designer for editing cascading style sheets
- Deployment descriptor editor for editing web.xml files
- DTD editor for editing document type definitions
- JavaScript™ editor for editing JavaScript
- Page Designer for editing HTML and JSP files
- XML editor for editing XML files
- XML schema editor for editing XSD files
- XSL editor for editing XSL files

You can access a source editor by clicking the Source tab in one of the editors mentioned above or by right-clicking on a relevant file name in Navigator or Package Explorer view and then clicking Open With → Source Editor.

Each source editor provides a consistent interface regardless of the markup language with which it is associated. It provides capabilities such as find and replace, undo, redo, a spell checker, and coding assistance. It also highlights syntax in different colors. Following is a brief description of some of the source editors’ capabilities:

**Syntax highlighting**

Each keyword type and syntax type is highlighted differently, enabling you to easily find a certain kind of keyword for editing. For example, in HTML, element names, attribute names, attribute values, and comments have different colors; in JavaScript, function and variable names, quoted text strings, and comments have different colors.

**Unlimited undo and redo**

These options allow you to incrementally undo and redo every change made to a file for the entire editing session. For text, changes are incremented one character or set of selected characters at a time.

**Content assist**

Content assist helps you to insert JavaScript functions, HTML tags, or other keywords. Choices available in the content assist list are based on functions defined by the syntax of the language in which the file is coded.

**User-defined macros and snippets**

By using the Snippets view, you can access user-defined code snippets and (for all code types except JavaScript) macros to help you quickly add regularly used text strings.
Function selection
Based on the location of your cursor, the function or tag selection indicator highlights the line
numbers that include a function or tag in the vertical ruler on the left area of the Source page.

Pop-up menu options
These are the same editing options available in the workbench Edit menu.

9.4 The Web deployment descriptor editor

The Web deployment descriptor editor lets you specify deployment information for modules
created in the Web development environment. The information appears in the web.xml file.
Note that you use the Web deployment descriptor to set deployment descriptor attributes. You
do not use it to manipulate Web resource content.

The web.xml file for a Web project provides information necessary for deploying a Web
application module. It is used in building a WAR file from a project. Whenever you create a
new Web project, a minimal web.xml file is automatically created in WEB-INF under the
project's Web content folder.

The Web deployment descriptor editor is dynamic and includes many tabbed pages (views)
that represent various properties and settings in the deployment descriptor. For example, you
can click the Servlets tab to display the Servlets and JSPs page. From this page, you can add
or remove servlets and JSPs Web application. The Web deployment descriptor editor
includes the following tabbed pages, each of which you can edit:

- **Overview** - provides a quick summary of the contents in the Web deployment descriptor
  and lets you add, remove, or change the contents.

- **Servlets** - lets you create a new servlet, add an existing servlet or JSP file to the
deployment descriptor, or remove the selected servlet or JSP file from the deployment
descriptor.

- **Filters** - lets you create a new filter, add an existing filter to the deployment descriptor, or
  remove the selected filter from the deployment descriptor.

- **Listeners** - lets you create a new listener, add an existing listener to the deployment
  descriptor, or remove the selected listener from the deployment descriptor.

- **Security** - lets you define security roles and security constraints.

- **Environment** - lets you add or remove the name of a Web application environment
  variable to the deployment descriptor.

- **Reference** - lets you add or remove the name of an enterprise bean reference to the
  deployment descriptor. There are 5 types of references you can define on this page:
  - EJB
  - EJB Local (J2EE 1.3 only)
  - Resource
  - Resource Environment (J2EE 1.3 only)
  - JSP Tag Library

- **Pages** - lets you add or remove welcome and error page to the deployment descriptor.
  Also allows you to define the login-config element.

- **Parameters** - lets you add or remove servlet context initialization parameters for a Web
  applications. The context parameters apply to all servlets within the application.
MIME - lets you add or remove mappings between the extension types and MIME types (such as text/plain). File extensions are strings that describe an extension without the dot (.) delimiter (for example, txt).

Extensions - lets you specify settings for WebSphere® extensions (such as enabling reloading).

Source - lets you edit the web.xml source directly.

Editing the XML source directly is not the recommended method for editing the deployment descriptor. The best way to build deployment descriptor files is to edit the multiple tabbed pages. As you specify deployment information in these tabbed pages, the editor automatically incorporates the appropriate XML tagging.

In addition to the configuration information in the web.xml file, other deployment descriptors in a Web project include the following information:

Binding information is required by the application server to bind the deployment information specified in the application to a specific instance. For example, it may map a logical name of an external dependency or resource to the actual physical JNDI name of the resource. It also may map security role information to a set of groups or users.

IBM binding and extensions information (ibm-web-bnd.xmi and ibm-web-ext.xmi files) — additions to the standard descriptors for J2EE applications, Web applications, and enterprise beans. The extensions enable Enterprise Edition or legacy (older) systems to work in the current WebSphere Application Server environment. They are also used to specify application behavior that is vendor-specific, undefined in a current specification, or expected to be included in a future specification.

If you import a WAR file into an existing Web project, you can include the deployment descriptor files included in the WAR file as the Web project's new deployment descriptor. Any specific deployment information already defined in these files is used when deploying the updated Web application.

The web.xml file can be updated automatically to reflect changes to your Web project. For instance, when you use the New Servlet wizard to create a new servlet in a Web project, the wizard places the appropriate servlet entry into the web.xml file.

9.5 Web Site Designer

Web Site Designer is a Web authoring tool. With Web Site Designer, you can:

- Create a Web site whose pages have a consistent look and feel.
- Customize the look and feel by editing the page template.
- View the entire Web site, then add and delete pages using a tree view of the site structure.
- Automatically generate navigation links using special comment tags.

Web Site Designer allows you to visually create, import, and customize complete Web sites with a consistent look and feel using a tree view of the site structure.

How is Web Site Designer used?

Web pages can be created using a wizard and edited using Page Designer. However, Page Designer edits a single page at a time and is not intended to make changes to several pages at once. Web Site Designer is provided to edit the look and feel of the whole Web site. By using a page template, look and feel can be made consistent throughout a site.
The page template contains the Web Site Navigation (such as the navigation bar, navigation tab, navigation trail, site map, and navigation menu), that is created using the Site Designer tools. You can also view the Web site in a Navigation view to add new pages, delete pages and move pages in the site.

The pages in a Web site are organized in a hierarchy. Web Site Designer uses this structure information to generate navigation links and site maps automatically. This means that when the structure of a site changes, for example, when a new page is added, the navigation links are automatically regenerated to reflect the new Web site structure. This dynamic navigation is accomplished using special comment tags for the navigation bar and site map. These comment tags use the site structure information and other site-related information that Web Site Designer stores in an XML file called .website-config.

When you create a Web project, Web Site Designer support is a default feature of the project. The Web project contains a placeholder for the website-config file named Web Site Navigation. Double-clicking Web Site Navigation launches Web Site Designer and opens website-config in the Navigation view.

Web Site Designer consists of an editor and a Web Site Navigation drawer:

- **Web Site Designer editor.** You can view the whole site within Web Site Designer editor. You can add, delete, and rearrange the Web pages that compose the Web site using this tool. There are two views in the editor: the Navigation view and the Detail view. The Navigation view shows the structure of the Web site and the Detail view shows a list of all the Web pages used in the site structure.

- **Web Site Navigation drawer.** On the Palette view there is a Web Site Navigation drawer that contains Web site elements, such as different navigation bars and a site map. You can drag and drop these items onto your Web pages in Page Designer.

- **Site drawer.** On the Palette view there is a Site drawer that contains page and project elements.

You can use these Web Site Designer tools in conjunction with other Web tools, such as free layout mode and page templates. You can alter an existing page template or create your own and then apply the page template to all of the Web pages in your Web site to ensure a consistent design. When editing the page template, you can use free layout mode to make changes to the layout of each Web page. Instead of making a change, such as altering the size of the navigation bar, on each individual Web page, you can alter the page template instead. Then, you can apply that change to your entire Web site.

**What's new in Web Site Designer?**

Here are a few highlights of the enhancements that were made to Web Site Designer from the last release. For details about changes to specific tasks, refer to the online documentation.

- **Multiple Web project support.** Web Site Designer now supports multiple Web projects. You can link pages between your various projects automatically generating links.

- **Automatic site structure creation.** You can use the Convert to Web Site and Add Existing pages wizards to add sites or pages that you want Web Site Designer to manage. You can also use create a site structure automatically using the link crawling method. And you can apply page templates to multiple pages during your conversion.

- **Web Site Designer Editor enhancements.** You can view part of your site as a subtree of the larger site.

- **Application of template enhancements.** You can apply a page template by dragging and dropping from the Thumbnail view.
> **Detailed information view.** You can edit keywords, descriptions, and abstracts all at one time.

### 9.6 Page Designer

Page Designer allows you to work with HTML, XHTML, JSP, and Faces JSP files as well as with embedded JavaScript™ code. Within Page Designer, you can move among three views of each page: Design, Source, and Preview. Each page provides a different way for you to work with the file that you are editing. You switch pages by clicking on the tabs at the bottom of the editor. These pages work in conjunction with the Outline and Properties views, tool bar buttons, menu bar options, pop-up (right-click) menus, and palette components.

**Design**
The Design page is the WYSIWYG environment that enables you to create and work with a file while viewing its elements on the page. For example, you can see the graphics that you have inserted into the file and continually check the visual presentation of the Web page as you design it. This is in contrast to the Source page, which shows you the HTML or JSP source code for the file, but not the screen presentation. The Design page provides full visual editing capabilities, including the ability to drag and drop from the Palette, the J2EE Navigator view, the Thumbnails view, and Microsoft® Internet Explorer.

**Source**
The Source page enables you to view and work with a file’s source code directly. The Outline and Properties views both have features that supplement the Source page.

**Preview**
Preview shows you how the current page is likely to look when viewed in an external Web browser.

To preview any dynamic content (such as JSP tags), you should use the **Run on Server** option (WebSphere test environment) from the page’s pop-up menu in the Project Navigator view.

You can also use the **Tools → Launch External Web Browser** menu option to open the current file in the default Web browser.

### 9.7 Cascading Style Sheet (CSS) Designer

CSS Designer shows the source of a CSS file and enables you to edit styles with the help of syntax highlighting, content assist, and Preview function. The modifications made in CSS Designer are immediately applied to the Design page of Page Designer if the HTML file has a link to the CSS file.

WebSphere® Studio supports the following World Wide Web Consortium (W3C) cascading style sheet standards:

- **CSS1** (cascading style sheet level 1)
- **CSS2** (cascading style sheet level 2)
- **CSS Mobile Profile 1.0**

In addition, the WAP Forum standard **WCSS 1.0** (WAP CSS 1.0) is also supported.
CSS Designer provides a Source pane and a Preview pane with capabilities that are similar to the three pages of Page Designer, although the orientation and interaction between these panes is slightly different in CSS Designer.

**Source**
The Source pane enables you to view and work with a file's source code directly. The Properties and Styles views have features that supplement the Source page. Many of the functions related to creating new styles, applying styles, changing properties, and setting links to style sheets are available from both the CSS Designer Style menu and the Styles view's tool bar.

Individual styles in a style sheet are typically defined as follows:

```css
selector { property : value; }
```

Here is an example of an actual style definition for a paragraph that might appear in a .css file:

```css
P { font-style : italic; font-size : x-small;}
```

The Source pane has many of the standard WebSphere Studio source editing features, including syntax highlighting, unlimited undo and redo, content assist, element selection, and pop-up menu options.

**Preview**
The Preview pane shows you how the currently defined styles are likely to look when viewed in a Web browser. You can select an actual Web page to apply and display styles created by CSS Designer, or use the sample HTML file, which provides a minimal version of the defined styles.

The Panes menu in the CSS editor enables you to change the orientation and relative size of the Source and Preview panes.
Using IBM WebFacing Tool to create browser interface for a 5250 application

The IBM WebFacing Tool is included as part of the WebSphere Development Studio Client for iSeries (WDSc). It is installed automatically for you. This chapter provides an overview of the process involved in converting an iSeries application using the IBM WebFacing Tool.

Tip: If you already used the IBM WebFacing Tool, it may not be necessary to read this chapter entirely and you may want to proceed with Chapter 11, “Enhancing the WebFacing applications” on page 243. However, you can find some valuable tips and tricks in this chapter within shaded boxes like this one.

Some functions described in this chapter are only supported on WebSphere Development Studio Client for iSeries Advanced Edition (WDSc AE). You must purchase this feature separately from the WDSc Standard edition. Even if you received WDSc as part of 5722-WDS free of charge because of the Software Subscription, you must pay a fee to acquire WebSphere Development Studio Client for iSeries Advanced Edition.

Note, this redbook provides an introduction to the IBM WebFacing Tool. More elaborated topics are covered in Mastering the IBM WebFacing Tool, SG24-6331.

Changes in the second edition of this redbook

This chapter was originally written for WDSc 5.0. Now it has been updated to reflect the changes and enhancements added with WDSc 5.1 in 2003 and WDSc 5.1.2 in 2004, as well as several fixpacks between the major releases. Technical differences from the previous edition of this book are indicated by a change bar at the left of the text. Minor changes such as grammar corrections and slightly different phrasing are not marked.

The base for writing this chapter is WebSphere Development Studio Client Advanced Edition for iSeries Version 5.1.2.3 (called WDSc AE in the following), that is WDSc AE 5.1.2 with Fixpack 3 applied.
10.1 Prerequisites for creating a new WebFacing project

During the conversion process, you have to access your iSeries server. There are a few steps that you must perform on the iSeries server before you can convert a host application.

10.1.1 Downloading and installing the latest service pack for WDSc

You can download a WebSphere Development Studio Client for iSeries Service Pack and install it from the following site on the Web:

http://ibm.com/software/ad/wds400/support/

For more detailed information about installing fix packs refer to 3.6, “Installing Fixes for WDSc” on page 48.

10.1.2 Installing WebFacing PTFs on OS/400

You must install the WebFacing server PTFs on the iSeries server where your 5250 application will run.

For a list of the latest PTFs, go to the following Web site and click the Host PTF requirements link:

http://ibm.com/software/awdtools/wds400/sysreq/

Tip: The Development Studio Client Installation Launcher (C:\WDSc\install\poststrt.exe) comes up automatically when you boot your PC after you install the product. It also allows you to automatically connect to your iSeries server and check for the necessary PTFs. However, be sure to check the previous Web site for more up-to-date information about the required PTFs.

While it is always a good idea to install the latest cumulative PTF package, make sure that you have at least the minimum level installed as defined in the README file for the service pack you installed with WDSc.

Important: For each release of OS/400, there are at least three PTFs for the IBM WebFacing Tool that are not contained in any cumulative PTF package (cumpack). These PTFs are absolutely necessary for working with WebFacing.

10.1.3 Host servers

The Remote System Explorer (RSE) communications server in Development Studio Client requires the OS/400 host servers to be started for them to function properly. With OS/400 V5R1 or later, the host servers should start automatically. If you did not previously start your host servers, then when you try to access the system, you receive the message “Required host servers are not started”. To manually start the host servers on the iSeries server, run the following command:

STRHOSTSVR *ALL

10.1.4 Starting the WebFacing server

On the OS/400 system where your 5250 application is to run, you need to start the WebFacing server. This applies even if you test your WebFacing application with the local test environment under WDSc.
Make sure that you also load and apply the necessary PTFs as explained in 10.1.2, “Installing WebFacing PTFs on OS/400” on page 194, before you start the WebFacing server. If the WebFacing server was active before you applied new PTFs, you may have to end and start it again.

Start the WebFacing server or run-time environment on your iSeries server using iSeries Navigator (formerly called Operations Navigator for iSeries). Expand your iSeries server, then Network, then Servers and then TCP/IP. Within the list of servers in the right pane, you find the WebFacing server, which you can start or stop. You may also select to autostart the WebFacing server whenever TCP/IP is started.

You may also start or stop the WebFacing server through the Remote Systems Explorer perspective by expanding the iSeries server, then right-click iSeries Objects (or right-click iSeries Commands), select Remote Servers, then WebFacing. Depending on whether the WebFacing is currently running or not, you may either start or stop the server. If a connection to your iSeries has not yet been established, both options are greyed out. See also “Starting and stopping the Remote Servers” on page 87 for more information about using RSE.

You can also start and stop the WebFacing server using the following CL commands:

- To start the WebFacing server for OS/400 V5R1 and later, enter the following command on the OS/400 command line:
  
  STRTCPSVR SERVER(*WEBFACING)

- To stop the WebFacing server for V5R1 and later, enter the following command on the OS/400 command line:
  
  ENDTCPSPVR SERVER(*WEBFACING)

**Note:** To start or stop the WebFacing server for V4R5, you have to use the Start WebFacing Server (STRWFSVR) or the End WebFacing Server (ENDWFSVR) command. Both have been replaced for OS/400 V5R1 and later.

To determine whether the WebFacing servers started, you can enter the Work with Subsystem Jobs (WRKSBSJOB) command on an OS/400 command line and specify QSYSWRK as the subsystem name. Then look for QQFWFSVR, the WebFacing run-time server, and QQFVTSVR, the WebFacing virtual terminal server.

### 10.1.5 Number of virtual terminals (QAUTOVRT)

Check the system value QAUTOVRT. If this value is 0, then you cannot launch any browser sessions 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. If necessary, set QAUTOVRT to *NOMAX or to a value greater than zero.

You change this system value using the iSeries Navigator by expanding the tree under your iSeries server, and then expanding Configuration and Service → System Values. Double-click Devices in the right pane. The Devices System Values window opens. Under the Automatic Configuration tab, in the Maximum Number of Devices field, change the value as shown in Figure 10-1.
10.2 Creating your first WebFacing project

This section takes you through the basic steps for using the IBM WebFacing Tool to convert a 5250 application to a browser-based application. It shows you how to create a project, access the display files for conversion, and then convert the display files.

The IBM WebFacing Tool consists of four different components:

- **WebFacing perspective**: This component within the Eclipse-based WebSphere Studio Workbench shows all necessary objects to work with the IBM WebFacing Tool.

- **WebFacing Tool project wizard**: This component allows you to enter the necessary information needed to build a WebFacing application and store that information as a WebFacing project in your workspace.

- **WebFacing conversion wizard**: This component reads the information in the WebFacing project. You supply the information either by entering it into the WebFacing Tool project wizard or by changing certain properties at a later time.

The wizard then converts each record format defined in the data description specification (DDS) members to JavaServer Pages (JSPs), JavaScript, JavaBeans, Extensible Markup Language (XML) files, and Hypertext Markup Language (HTML) files. These files together form a Web application that is ready to be deployed to an application server such as WebSphere Application Server Version 3.5.5 or later.

- **WebFacing run-time server**: This is part of OS/400 V4R5 or later. It allows the WebFacing application to start and connect to the existing 5250 application.

**Note**: For OS/400 V4R5, WebFacing applications converted with WDSc V5, are not officially supported. The run time on OS/400 V4R5 systems should be fine for now, but there are no plans to update or service the WebFacing server on V4R5 systems.
The following sections show you how to use the components of the IBM WebFacing Tool without applying any enhancements to the generated applications. Chapter 11, “Enhancing the WebFacing applications” on page 243, offers more topics on using the IBM WebFacing Tool.

If you are not yet familiar with the new terms and concepts introduced with the Eclipse-based WebSphere Studio Workbench, such as perspective, project, or workbench, refer to Chapter 3, “The Eclipse framework” on page 27.

10.2.1 Getting started with the IBM WebFacing Tool

As mentioned earlier, the IBM WebFacing Tool is part of WebSphere Development Studio Client for iSeries (WDSc) or WebSphere Development Studio Client for iSeries Advanced Edition (WDSc AE). Therefore, you begin by starting the WDSc or WDSc AE. The following sections take you through the necessary steps based on a sample Order Entry application that you can download from the Web (see Appendix A, “Additional material” on page 547, for more information about downloading the sample application):

1. Start the IBM WebSphere Development Studio Client on your workstation. From the Windows desktop, click Start → Programs → IBM WebSphere Studio → Development Studio Client for iSeries (or Development Studio Client Advanced Edition for iSeries if the Advanced Edition is installed).

2. All projects you work with in WebSphere Studio workbench are stored in a workspace. This is a directory within the file system of your PC or a files server mapped to as a network drive to your PC. You can select which workspace to use, as a new function as of WDSc V5, when you start the workbench. If you are not sure which directory to select, you may leave the default as shown in Figure 10-2.

![Figure 10-2   Selecting the workspace](image)

Tip: All objects you deal with in the WebSphere Development Studio Client for iSeries workbench are stored in a directory on your PC. You can create a separate workspace if you want to isolate the work that you are doing for different projects or different releases of your WebFacing projects. The performance is better if you do not put all of your projects into a single workspace. It also helps to organize your WebFacing projects.

The “Use this workspace as the default” check box in Figure 10-2 lets you avoid this step in the future. However, if you decide to work with a different workspace in the future, you need to change the shortcut that is used to start the WDSc application by adding a -setworkspace parameter to the command:

"<Install Path>\WDSc.exe" -feature com.ibm.WDSClient -setworkspace

Where <Install Path> is the drive letter and path where you installed your WDSc client. This part should not be changed.
If you are using the WebSphere Studio workbench for the very first time with WebSphere Development Studio Client for iSeries (WDSc), you see the Remote Systems Explorer (RSE) perspective (see Chapter 5, “RSE, the modern PDM - more than just a GUI” on page 63, for more information about RSE). If you worked with the workbench before, the perspective, which was active at the time you closed the tool last time, is open.

**The WebFacing perspective**

All work with IBM WebFacing Tool needs to be done from the WebFacing perspective in your workbench. In previous releases of WDSc and WebSphere Development Studio for iSeries, you had to open that perspective manually as described in the following steps. For WDSc V5 or later, you can skip this step. The WebFacing perspective is automatically opened for you when you create the WebFacing project.

If this is the very first time you are working with WebFacing, you may proceed to “Creating your first WebFacing project” on page 199 now. However, if you already created a WebFacing project and accidently closed the perspective, follow the instructions here to open it again:

1. In the Development Studio Client window (Figure 10-3), click the **Open Perspective** icon or select from the menu **Window → Open Perspective → Other → WebFacing**.

   ![Figure 10-3 Opening a perspective](image)

   **Figure 10-3 Opening a perspective**

2. From the Open Perspective options, select **Other** (Figure 10-4).

   ![Figure 10-4 Selecting Other](image)

   **Figure 10-4 Selecting Other**

3. On the Select Perspective window (Figure 10-5), select **WebFacing** and click **OK**.

   ![Figure 10-5 Selecting WebFacing](image)

   **Figure 10-5 Selecting WebFacing**
Creating your first WebFacing project

To convert a 5250 DDS source members, you must create a WebFacing project in WebSphere Development Studio Client for iSeries. Part of creating the WebFacing project requires using a wizard, which prepares all the required components for the conversion process.

A project is the highest level of a unit within WebSphere Studio workbench. A WebFacing project contains all the objects needed to develop a WebFacing application. Follow these steps:

1. Create a project using one of following methods:
   - Click the Open New Project icon (Figure 10-6). In the left pane of the Select window select WebFacing then select WebFacing Web Project in the right pane and click the Next button.
   - Click the drop-down menu at the right of the Open New Project icon. Select WebFacing Web Project (if this option is available).
   - Click the drop-down menu at the right of the Open New Project icon. In the left pane of the Select window select WebFacing then select WebFacing Web Project in the right pane and click the Next button.
   - Select File → New → Project. In the left pane of the Select window select WebFacing then select WebFacing Web Project in the right pane and click the Next button.
   - Select File → New → WebFacing Web Project (if this option is available).
   - Select File → New → Other → WebFacing → WebFacing Web Project → Next.
– If you have the WebFacing perspective open, you can right-click anywhere in the WebFacing Projects or Navigator tabs and select New → Project → WebFacing → WebFacing Web Project and the WebFacing project wizard opens.

2. On the WebFacing Project window as shown in Figure 10-7, complete the fields as explained in the following steps:

a. Make sure the Create new J2EE 1.3 WebFacing project radio button is selected.

b. In the Project name field, enter the name of the new project. You should type a unique name in this field. Unique means that there must not be a project with the same name within your workspace. However, the new Web application most likely will have the same name, so it must also be unique among all Web applications. The project name cannot contain spaces.

c. To save your project to a location other than the default location (a new directory with the same name as the project within your workspace), and click Browse to select the preferred folder or key in the desired path.

d. Enter a name in the EAR Project field or use the default name. An Enterprise Application Resource (EAR) project is created if it does not already exist. For each new WebFacing Web project, we recommend that you use a unique EAR project. Note, in releases prior to WDSc V5.1.2, the EAR Project field was called the Enterprise...
Application project and the default was always DefaultEAR. In WDSCE the default is now the name of your WebFacing project with the letters EAR appended.

Important: For a Java 2, Enterprise Edition (J2EE) 1.3 project, do not specify an EAR file that was previously created for a J2EE 1.2 level project. The data compression support in WebFacing requires filters, which are not supported in J2EE 1.2. The test environment does not start properly if you have this configuration.

When you create a new WebFacing project, an entry is added to the application.xml, .modulemaps, and ibm-application-ext.xmi files in the META-INF folder in the Enterprise Application project folder.

e. The Context root field, by default, is populated with the same value you used for the Project name. When you deploy the finished WebFaced applications, you can deploy them as EAR files or Web Archive (WAR) files.

If you deploy your application using an EAR file, the value for Context root becomes part of the Uniform Resource Locator (URL) that is used to access your application. For example, depending on how you configured the WebSphere Application Server, if wf_app was used for your context root, the URL to access the application may be similar to http://myserver:8080/wf_app/. If you deploy your application as a WAR file, you can choose the value for context root at deployment time.

Note: The following options in older releases prior of WDSCE have been removed:

- The Create CSS file check box is no longer used and has been removed for WDSCE V5.
- In WDSCE V5.1.2, Struts support is now enabled for all WebFacing projects regardless whether you use WebSphere Development Studio Client for iSeries Advanced Edition or WebSphere Development Studio Client for iSeries. See 13.4, “Struts tools” on page 400, for more information about Struts.
- In releases prior to WDSCE V5.1.2 you could select the option to Generate JSP custom tags, if you are using the Advanced Edition of the product. This option has been removed in WDSCE V5.1.2.

f. Only if you have WDSCE AE installed, you may select to:

- Add system screen support (see 11.11, “Showing system screens with the WebFacing Tool” on page 408 for more information)
- Use single signon (see 11.1.1, “Securing your WebFacing application using single signon” on page 315 for more information).

g. Click the Next button.

In releases prior to WDSCE V5.1.2 the J2EE Setting dialog would have opened at this point. That panel has been removed in WDSCE V5.1.2 and all new WebFacing projects are built based on J2EE level 1.3.

However, if you plan to deploy your WebFacing application on WebSphere Application Server V4, you must use J2EE level 1.2. In order to run a WebFacing application in a WAS 4.0 environment, you must add WebFacing support to a previously created J2EE 1.2 Web project instead of using the New project wizard described here.

3. Select which DDS sources to convert in the Select display file source members to convert panel (Figure 10-8 on page 265). Follow these steps:

a. Specify from which OS/400 system the DDS source members should be retrieved. Select an existing connection from the drop-down list or click New to call the New
Remote iSeries System Connection wizard to describe how to access an iSeries server you have not used yet. For more information about connections, see 5.2.3, “Connections” on page 67.

b. In the Library filter drop-down box, type the name of the library that contains the file you want to convert or select one of the keywords *LIBL, *USRLIBL, or *CURLIB to see all libraries in your library list, the user part of it, or the current library assigned to your user profile. You may also type a generic name, such as LIB6*, to search all libraries with names starting with the characters LIB6.

c. You may further restrict the number of source files and members to be shown by specifying generic names in the File and Member fields. This is new with WDSc V5.

d. Click Refresh DDS list.

e. If you did not previously default to the OS/400 sign-on user profile and password (as described in “Saving your iSeries sign-on information” on page 69) or if you are not yet connected to the iSeries server, a logon window appears. Enter the OS/400 user profile name and password, and click OK. The libraries entered in the Library filter are retrieved from the iSeries server (Figure 10-8 on page 265).

Tip: If you see the error message “There are no libraries found for the library filter”, you may have entered a library that does not exist, or you may not be authorized to the library.

f. Expand the library tree. The panel displays all source files in the library or a subset of it according to the generic name you entered in the File field.
Figure 10-8  Selecting the DDS sources to be converted

g. Expand the source file tree. Typically the name would be QDDSSRC, but in our example it is RPGAPP. Depending on what you selected in the Member types field, the panel displays the source members for display files, MNUDDS or both types in the source file.

h. Select the source members you want to convert. In our example: ORDENTD, SLTCUSTD, and SLTPARTD.

i. Click the >> button to copy the selected source members to the panel on the right, as shown in Figure 10-8.

**Note:** To select all source members, select the source file and click the >> button. However, when you are new to using WebFacing, we recommend that you convert no more than ten or fifteen source members until you understand the entire process and have gathered experience for the time needed to perform each conversion.

**Tip:** If you accidentally selected a wrong member and want to remove it from the list, right-click its name and select Delete from the context menu.

j. Click Next.
4. On the Select UIM source members to convert panel Figure 10-9), select the UIM source members that you want to convert.

**UIM help source member selection:** Only User Interface Manager (UIM) help in the DDS can be converted by the WebFacing Tool. Use the Select UIM source members to convert page to browse your libraries for the UIM help source members that you want to convert.

If you used UIM to define panels (for example “Work with”-type lists) other than help panels, you cannot use the WebFacing Tool to convert those into a WebFacing applications. However, the WebFacing System Screen Support introduced with WDSc AE 5.1.2 not only supports panels provided with OS/400 but also user defined UIM panels. See 11.11, “Showing system screens with the WebFacing Tool” on page 408 for more information.

5. Click Next.

6. On the Specify CL commands panel (Figure 10-10), specify the CL command to call on the iSeries server. This is the CL command that starts the 5250 application. For our example, we used the following values:
a. For CL command, type CALL ORDENTR
b. For Command label, type Order Entry
c. For Invocation name, leave the default INV1

Note, the CL command may also contain variable parameters in the form PARM(&xxxxx). If you do so, the browser user will have the opportunity to provide the content of these parameters prior to starting the 5250 job. See 11.6, “Using a CL command with variable parameters” on page 265, for a detailed example.

d. Click Add.
e. You can select the Sign-on with specified values radio button and provide a user ID and password. With this setting, you don’t need to sign on every time you start your WebFacing application. For testing your new WebFacing application, it is convenient to use these fixed values.

When using the application in production, for security reasons, you may want Web browser users to enter the user profile name and password. In this case, leave the Prompt for signon option selected (see 11.1, “The user profile and password the application runs under” on page 244).
**Note:** When the WebFacing server starts the interactive job, the initial program (INLPGM) defined for the user profile is not called. You need to make sure the program to be called or the command to be executed is found in the library list.

You may also decide to specify the command name and program name qualified with the library, for example: QSYS/CALL LIB6961/ORDENTR. However, this does not necessarily mean the library list is set to find other objects accordingly, unless your application takes care of that.

See also “Setting the library list to run the 5250 application” on page 214.

For our sample application, regardless whether you use a fixed user profile or let it prompt at execution time, the user profile needs to specify LIB6961 as its current library, because the application does not contain any logic to add that name to the library list.

f. Click **Next**.

![Choose a Web Style](image)

**Figure 10-11 Choose a Web style panel**
7. Select the Web style you want to use. In Development Studio Client, several styles are predefined as shown in Figure 10-11. You can learn how to create your own style in 11.3, “Using buttons instead of command keys F1 through F24” on page 247.

**Note:** In WDSc V5.0.1, a new style called textbuttons has been added that is tailored for use with the “Replace text with button” conversion option. It has a PageBuilder.jsp that does not include the command keys area, since they are already included in the application area with this option.

The radio button Enable for Web Site Designer was introduced with WDSc 5.1 and allows you to use the Web Site Designer to further enhance your WebFacing application. See 9.5, “Web Site Designer” on page 250 for a brief description of the Web Site Designer.

8. Click **Next**.

The Complete WebFacing project panel (Figure 10-12) opens. On this panel, you create the project and have the option to convert the project either immediately or later.

![WebFacing Project panel](image)

**Figure 10-12  Creating a project**

9. Select one of the options and click **Finish**.
The project is created and the references to the selected sources are maintained in the workbench. Depending on the number and complexity of display files you choose, it takes several minutes to create the project and between 30 and 90 seconds per display file to convert the project. A progress bar is shown while the conversion process is active.

If you choose to convert your project at a later time, you can convert the project by right-clicking the project in the WebFacing perspective and selecting the Convert option from the context menu as shown in Figure 10-13.

![Starting the WebFacing conversion](image)

**Figure 10-13  Starting the WebFacing conversion**

### The WebFacing conversion log

After the conversion process completes, a conversion log is displayed in the right panel. You can maximize the log view by double-clicking the tab on top of the pane (as shown by the pointer in Figure 10-14).

Double-clicking the tab on top of the pane again restores it to its original size. The maximized view is shown in Figure 10-15 on page 272. This behavior applies to all panes in all perspectives of the workbench. See 3.3, “Components of the Eclipse GUI framework” on page 29, for more information about using the workbench.

**Tip:** If you accidentally closed the Conversion Log view, you can display it again, by right-clicking the project in WebFacing view and selecting View log from the context menu.

The upper left part of the conversion log shows the names of the display files which have been attempted to convert. The two icons on the left of each name shows whether the conversion and the compilation was successful:

- A blue check mark indicates a successful conversion.
- A yellow triangle with an exclamation mark means there is a warning.
- A red circle with white cross indicates that the conversion was not successful.

Click the name of a display file and all record formats appear on the right. Clicking one of the record formats shows each individual message in the bottom part of the view as shown in Figure 10-15 on page 272.
The conversion log consists of four different views:

- **DSPF Conversion Log**: This was described earlier. It opens by default.
- **Overview**: This view opens when you click the tab on the bottom left.
- **Referenced Keywords**: This view gives a summary of the keywords that were used in the DDS and which of those are fully, partially, or not supported by the IBM WebFacing Tool.
- **UIM Help Conversion Log**: This view shows similar information as the one for DSPF.

Multiple log files can be open at the same time in the editor view. You can select a specific log file by clicking a corresponding tab at the top of the editor view. The tabs at the bottom of the conversion log file select different views within the conversion log. Make sure you select the correct log file to analyze the results of the conversion process.

**Important**: Close the conversion log file after you analyze the results of conversion. If the conversion log file is open, WDSc may hang when it tries to update the files in the project after your changes. This was a problem on previous releases.
The WebFacing project

The WebFacing Conversion Tool is not a screen scraper. The source for the display files is converted to a set of XML, HTML, and JSP files.

Note: In WDSc V4, the WebFacing Conversion Tool also generated the source for JavaBeans. Starting with WDSc V5, the information is stored in XML files to describe the actions to be performed during execution time.

These files are saved in the WebFacing project. As described earlier, a project is a set of directories with a predefined structure within your workspace. After you create the project, you can see its contents in tree-form in the upper left pane of the workbench as shown in Figure 10-14 on page 271.

The upper left pane contains multiple views stacked upon each other. You can select which one to show by clicking the tab at the bottom of the view. With very early versions of WebFacing (WebSphere Development Studio for iSeries 5.1) the tab was on top of the pane. By default, the WebFacing Projects view is shown and contains only the WebFacing projects. When you click the Navigator tab, you see all projects in your workspace. The third view in this pane, called Server Configuration, came new with WDSc V5 and is explained in 10.4, “Testing your application within Development Studio Client” on page 276.
If the conversion log is still shown in the entire workbench window, double-click the tab on top to restore it with its original size into the right pane, so you can see the WebFacing projects view in the left pane again. You may collapse and expand the several branches of the tree there to see the files of various types that were created by the WebFacing conversion process.

**Adding or deleting components of the WebFacing project**

You can add and delete source members for conversion to and from your project by right-clicking DDS node in the WebFacing Projects view. Likewise you can add more CL commands by right-clicking the CL Commands node. Then you need to convert only the new components by choosing Convert from the context menu (as shown in Figure 10-13 on page 270) of the new element.

You can open the source code editor for the XML files (in place of the JavaBean source code in WDSv4) 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.

For performance reasons during project creation, conversion, and deployment of WebFacing applications, the XML files are stored in a JAR file called `DDSGeneratedData.jar`. It is stored in the `WEB-INF/lib` directory. This means that you cannot edit the XML files anymore unless you extract the individual files from the JAR file.

**Note:** The conversion log shows the last status of all display files that are converted. The WebFacing conversion tool only converts the display files that you selected.

**Viewing or modifying the generated code**

The upper left pane of the WebFacing perspective by default shows the WebFacing projects view as shown in Figure 10-14 on page 271 for example. The same pane contains two other views, which you can bring to the foreground by clicking on of the tabs at the bottom of the pane.

**Note:** If you have closed the WebFacing project view (or any other views mentioned in this book) accidently, you can restore the original layout of the WebFacing or any other perspective by selecting **Window → Reset Perspective** from the menu bar.

When you click the Navigator tab as in Figure 10-16, the Resource Navigator view is shown instead of the WebFacing projects view. It displays the actual contents of the project as opposed to the WebFacing projects view, which basically shows your selections to define this project.

The general structure of a J2EE conforming Web project is described in “Web project organization” on page 449. A WebFacing project also has the structure of a J2EE Web project with some additional directories. As you can see in Figure 10-16, the `webContent` folder contains the path `webfacing/jsp/DDSrecords/LIB6961/RPGAPP/ORDENTD/html`.

That is, for each converted display file there is a directory within `webfacing/jsp`. The path for the subdirectory is built with the components library (LIB6961 in our example), source file (RPGAPP), and source member (ORDENTD).
Within the “display file directory”, there are two JavaServer Pages (JSP) files for each record format with that display file, except for subfile records. The information related to subfile records is contained in the JSP for the subfile control record.

![Figure 10-16 Resource Navigator showing WebFacing project](image)

You can modify the generated code. However, when you convert the project or parts of it again, some of the code is regenerated and your changes may be overwritten. This applies in particular to content of `webfacing/jsp/DDSrecords`, described above.

A very common reason why you might need to reconvert one or more DDS members, is when you changed the DDS substantially. For example, if you added or removed a field or changed the size of a field. The best method to apply persistent changes to the generated JSPs is to add `Web Settings` to the DDS source as described in 11.9, “Customizing the WebFacing interface with CODE Designer” on page 361. See also chapter 7 and 8 of the redbook `Mastering the IBM WebFacing Tool`, SG24-6331, for more complex techniques to enhance the generated WebFacing application.

**Note:** The directory path is based on the library of source file and their members with the assumption that at execution time the display file object (*DSPF*) resides in the same library and has the same name as the DDS source members. If your display files reside in a different library or have different names at execution time, you can modify the default object mappings as described in 12.6, “DDS Object Mappings” on page 428.
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.

### 10.3 A look at the WebFacing project properties

Some important properties are associated with each project that can be modified to change the behavior of the conversion process. This section only shows you a few of the properties. To learn more about what can be done by setting these properties, see 11.3, “Using buttons instead of command keys F1 through F24” on page 247 or Section 7.5 of the redbook *Mastering the IBM WebFacing Tool*, SG24-6331, for a more complete discussion.

To access the properties, makes sure to open the WebFacing projects view (rather than the Resources Navigator view described in Figure 10-16 on page 274), right-click the WebFacing project and select Properties. Figure 10-17 shows the Properties window.

By default, the run-time properties for the entire project open. The properties shown here are important for running the new WebFacing application. They describe the name of the OS/400 system in the Host name field and the user profile along with its password to be used to start the 5250 application, if a fixed profile is selected.

Note, some of the parameters shown in Figure 10-17 apply only to the Advanced Edition of WDSce.

![WebFacing Project properties](image)
When changes in the WebFacing project properties take effect

When changes in the WebFacing project properties take effect depends on the type of property. In other words, it varies what you have to restart to see the result of your change.

- **Conversion property**: Changes take effect after reconversion.
- **Run-time properties**: Changes take effect after the Web application is restarted on the server, specifically any properties stored in the web.xml file.
- **Style properties**: Changes take effect when the browser reloads the CSS files, which is usually when a new browser instance is used, depending on your browser settings.

Setting the library list to run the 5250 application

To run an application on an iSeries server, the application typically demands to have one or more libraries in the job's library list. Necessary objects for an application, such as programs, DB2 tables, and others, exist within an OS/400 library and may interface with other applications and databases within different libraries on the iSeries server.

**Note**: You cannot use an initial program to set up the initial library list, because WebFacing does not invoke the initial program associated with a user profile.

A user profile, which is used to access the iSeries server, is always required to run the WebFacing application invoked by the browser link. It may be necessary to set up the user profile, so that a certain library list is available after the job was started under that profile.

There are several ways to achieve this initialization. A common technique to set a library list for an interactive (5250 based) job is to assign an initial program (INLPGM) to the user profile. However, when the WebFacing server starts the corresponding 5250 job, the initial program associated with the user profile is not called.

Alternative ways to set up the correct library list for your 5250 application are:

- Create a program to start the application, which includes setting the correct library list and other possible information needed by the application. Specify the qualified name of the program in the properties of the WebFacing application shown in Figure 10-10 on page 267.
- Create a job description (an OS/400 object of type *JOBD) and associate the job description to the user profile that is used to access the iSeries server. Make sure the library list in that job description contains the libraries for all applications to be started by this profile. If you decide the user should be prompted to enter their user profile name and password, make sure all these user profiles are associated with proper job description.
- If only a single library is needed, you may define it as the current library for the user profile.
- Add the libraries to the QUSRLIBL system value. In most cases, this technique is less flexible since it involves changing the default library list for all jobs and all users in the system, regardless of whether they need this library.

10.4 Testing your application within Development Studio Client

Development Studio Client includes multiple test environments (WebSphere Application Server V4 and V5, Tomcat, and so on). You can unit test your application within Development Studio Client, by performing the following steps:
1. Make sure the WebFacing server on your iSeries is started. See “Starting the WebFacing server” on page 194 for a description on how to do so.

2. In either the WebFacing view or the Navigator view, right-click your WebFacing project and select Run on Server from the context menu as shown in Figure 10-18.

3. The Server Selection window (Figure 10-19) opens. Specify the type of new server to use or create.

   **Note:** Unlike earlier versions of WDSc there many different versions of application servers listed here and you may have to expand a folder to see the appropriate server environment. Make sure not to select a Portal server (for using portlets see 10.7, “Creating a WebFacing Portlet project” on page 302 instead) and make sure to select server type having Test Environment in its description.

4. Click Finish. When you create your very first WebFacing project, the wizard automatically creates an application server in your local test environment for you.
One or more instances of the test environment may already be defined. Then you have the choice to either use an existing environment or have a new one created.

**Tip:** During our tests, we saw server environments unnecessarily being created. In some cases, this led to port conflicts when trying to start a new test environment. It turned out to be safer to always use an existing test environment rather than creating a new one in addition.

5. The test environment starts. The tool creates a server configuration with the default values. Depending on the memory size and processor speed of your workstation, and the size of the project, this may take several minutes to complete while you see information boxes on your workbench (see Figure 10-20).
Chapter 10. Using IBM WebFacing Tool to create browser interface for a 5250 application

279

Figure 10-20  Publishing to a test environment

Note: With WDSc V4, at this point, a new perspective, the Server Perspective, would open. A change has been implemented with WDSc V5 that includes certain views from the Server Perspective with the WebFacing Perspective. Now there is no need to open an additional perspective, resulting in less memory resources on your workstation.

You find the Server Configuration view in the upper left pane of the workbench after you start the WebFacing perspective. The Console and Servers views are located in the lower right pane, after you complete the Run on Server task at least once. To test the WebFacing application, a browser window opens in the upper right pane overlaying the Conversion Log view.

After some time, the information window closes. Notice that, at this point, the test environment did not completely start yet. When you take a closer look at your workbench, you notice that a new view, the Console view, in the lower right pane of the workbench opened and is brought to the front as shown in Figure 10-21.
While the built-in WebSphere Application Server starts, you see messages (typically in blue color) moving from bottom to top inside the Console view. You may have to scroll to the right to see the full text of the message. The last of the messages should be similar to this example:

```
5e315b47 WsServer      A WSVR0001I: Server server1 open for e-business
```

Any messages that appear red in color as you scroll up indicate that there is most likely some error with the configuration of your workstation or application.

6. After the application server completely starts, the index.jsp page opens within that pane.

**Important:** If you are planning to run the WebFacing application in a WebSphere Application Server 5.0 environment, no manual configuration is needed to provide UTF-8 encoding. This is because the WebSphere Application Server 5.0 run time allows for J2EE 1.3 function. This function is automatically detected in the WebFacing run time. However, if you added the WebFacing support to J2EE Level 1.2 Web project because you are still using WDSc V4, you must follow the steps outlined in the following section.

### Configuring UTF-8 support for older application server versions

The WebFacing Tool creates Web applications with multiple language support. Because languages use different character sets, data streams between the browser and WebSphere Application Server are UTF-8 encoded. For WebFacing J2EE 1.2 projects to work correctly, UTF-8 support must be configured in the WebSphere Application Server administrative
10.4.1 Unit testing your application

After you complete the steps given in the previous sections, you can test the new Web application locally within the WebSphere Studio workbench on your PC. The term locally refers to the Web application running on an application server in WDSm in the WebSphere test environment. It still has to connect to an iSeries server having the 5250 application installed in OS/400.

In the upper right pane of the WebFacing perspective, a new view opens. As shown in Figure 10-22, it contains an instance of Microsoft Internet Explorer displaying the index.jsp file that is created as part of the WebFacing application.
As you can see in Figure 10-22, the index.jsp file does look fairly attractive. You can learn how to enhance the visual appearance of this page using Page Designer in 11.7, “Enhancing the layout of the launch page” on page 347. Page Designer comes as part of the WebSphere Studio workbench (and therefore with WDSc).

Click the first hyperlink. This causes the WebFacing servlet running in your local application server to perform the following actions:

- If password prompting is selected, it shows the logon.jsp file. In previous version of WDSc the page was named logon.html.
- Accesses OS/400.
- Logs on with the given user profile and password. For our sample application, the user profile needs to specify LIB6961 as its current library, because the application does not contain any logic to add that name to the library list.
- Invokes the CL command (see Figure 10-10 on page 267).
- Invokes the first JSP file of the WebFacing application.

You can now test your application by clicking the buttons, entering data in any input fields, or pressing one of the command keys F1 through F24 instead of clicking a button.

It may take a while to start the application before you see the first page as shown in Figure 10-23. It takes longer the first time because the JSPs are page compiled, but subsequent requests are faster.
10.4.2 Ending the test environment

If you are done with the testing, you may stop the server and close the browser window by performing the following steps:

1. Click the **Servers** tab below the lower right pane.
2. Right-click the server and select **Stop** as shown in Figure 10-24

![Stopping the server in the test environment](image)

Figure 10-24   Stopping the server in the test environment

3. Close the browser window by clicking the **x** at the right of the tab above the browser pane shown in Figure 10-24.

**Tip:** After you finish testing a new or changed application using the local application server, we recommend that you stop the application server and close the browser window. This prevents problems with future changes of your application, because the old version is still in the cache. This also saves resources on your workstation.

10.5 Stepping through the sample application

If you followed the previous steps in this chapter, you have now successfully created a basic WebFacing application. That is, we used as many default settings as possible and did not attempt any additional enhancements yet.

The following sections demonstrate how to present this application to a browser user. They also compare the application to the original version on a character-based 5250 display station or emulation session. In Chapter 11, “Enhancing the WebFacing applications” on page 243, we explain what you can do beyond using just a *simple* WebFacing implementation.
10.5.1 Starting the application

First let’s look at the original 5250-based application and then compare what WebFacing created from it.

The 5250 user interface

When you want to start a traditional interactive application under OS/400, you always need to sign on with a valid user profile name and password to a workstation. The workstation must be associated to a workstation entry of an active subsystem to display the Sign On panel. After you sign on, you have several options to choose from to start an application:
  - Automatically start an initial program (INLPGM) defined in your user profile.
  - Enter an option of an automatically called menu (INLMNU).
  - Enter a CL command on an OS/400 command line as shown in Figure 10-25.

![Figure 10-25 Starting an application from an OS/400 command line](image)

After that, you typically see the first panel of your application.

Using a Web browser

When you want to access a Web application, you typically start by entering an URL into the address field of a browser. The same applies for a WebFacing application. You need to enter a URL in the form `http://hostname/application`. Here `hostname` is your Hypertext Transfer Protocol (HTTP) server, which also needs to be defined in the configuration of your application server. However, if you are using the test environment as described earlier in this chapter, `hostname` is `localhost:9080`.

The part `application` is the name of the Web application as defined in the application server. By default, this is the same name you specified as the WebFacing project name in the WebFacing Project panel (Figure 10-7 on page 262). Because of the way the WebFacing application is defined, a page with the name `index.jsp` opens automatically without needing to specify the name of that file.
New WebFacing launch page index.jsp with WDSc 5.1.2

After you select Run on Server as mentioned in 10.4, “Testing your application within Development Studio Client” on page 214, the browser window automatically opens the following URL:

http://localhost:9080/MyProject

The releases of the IBM WebFacing Tool provided with WDSc prior to version 5.1.2 generated a launch page called index.html to start the WebFacing application. Now, not only the name has changed to index.jsp, but also its layout and implementation.

Figure 10-26 shows the browser window for our sample application.

1. Click Launch. If you did not specify the sign-on information during the development of the WebFacing project, you see the logon page.

2. Enter the user ID and password and click Logon. If you specified the valid sign-on information as part of the WebFacing project, then no logon page is displayed.

At this point, the WebFacing run-time servlet, which is part of the generated Web application, contacts the WebFacing server running on OS/400. It passes along the name of the CL command to be called and the user profile and password. The WebFacing server, in turn, starts a new interactive job on the iSeries host and submits the requested CL command. In our example, the command is call ordentr.

10.5.2 The first application panel

After submitting the CL command to start the application, we look again at the application from a 5250 user’s view and then show the WebFacing browser interface.

The 5250 user interface

We are using a very simple application that allows the user to enter a customer number in the first panel, press F4 to prompt for a list of customers, or press F3 or F12 to end the application. Figure 10-27 shows the first panel of the sample application used in this book.
Using a Web browser

The initial page is displayed in the browser as shown in Figure 10-28. It is based on the first output of that job to a display file like the example in Figure 10-27.

Tip: To see more of the browser window while using the test environment, you may double-click the Web Browser tab as indicated by the mouse pointer in Figure 10-28.

Using command keys with the browser

Most 5250 applications take advantage of the command keys F1 through F24 to allow the user to request certain actions with a single key stroke. The IBM WebFacing Tool recognizes each command key that is activated for a display record format in the DDS. Then it converts it to a button for the browser user to perform the same function by clicking that button.

Tip: The browser user can either click the button or press the command key. When the focus is the address field of the browser, the command keys may not work or behave differently. For example, if F4 is pressed while the cursor is in the address field, a pull-down menu shows the last recently entered URLs instead of performing the function defined by the 5250 application.
See 11.3.1, “Command key recognition” on page 248, for information about how you can change the way the command keys are converted into buttons.

In our example application, we also converted the UIM help source files. This page (Figure 10-29) is launched by pressing the F1 function key with the cursor in the selected field or by clicking the Help button.

**10.5.3 Converting a window subfile**

In our example application, we converted a window subfile.

**The 5250 user interface**

The subfile opens when the user presses the F4 command key and types 1 in the Opt field to select a customer as shown in Figure 10-30.
Using a Web browser
To launch the customer selection subfile, click the prompt button on the page or press F4 and a window opens similar to the example in Figure 10-31.

In our example, the Opt field is defined to be restricted with the DDS keyword VALUES to allow only the values 0 and 1 to be keyed in. Because of this, the browser user has two choices to enter the value:

- Enter the number with the keyboard as a 5250 user would enter it.
- Click the option in the context menu that appears when you move the mouse pointer above the field.

"Using links instead of entering values into option fields" on page 312 shows a method to create a selection based on links. This is more commonly used with browser interfaces than the methods described earlier.

Scrolling in the subfile window
You may scroll in the subfile window by pressing the Page Up and Page Down keys or by clicking the shaded part of the scroll bar as shown in Figure 10-32. If you choose the latter
method, the numbers of the items to be displayed next are shown when you move the mouse
pointer above the area without clicking it. This is shown in Figure 10-32.

![Select Customer](image)

**Figure 10-32 Scrolling in subfile**

### 10.6 Running a WebFacing application in WebSphere

After you create, convert, and test your new WebFacing project in the WebSphere test, you
need to deploy it to an application server. WebSphere Application Server deployment
involves three general tasks:

1. Export your WebFacing project as a WAR or EAR file.

2. Install the application in WebSphere Application Server. Use the application installation
wizard (WebSphere Application Server 4.x and later) to install the WAR or EAR file.

3. Start the Web application.

Step 2 varies depending on which version of WebSphere Application Server you are using.
For example, WebSphere Application Server 4.x Advanced Edition uses a stand-alone
administrative console. WebSphere Application Server 4.x Advanced Single Server Edition
as well as WebSphere Application Server 5.0 use a browser-based administrative console.

The following sections outline the necessary steps for some versions of the WebSphere
Application Server.

#### 10.6.1 Transferring an application to the iSeries server

This process requires you to:

- Export an application from the WebFacing project to a WAR or EAR file.
- Transfer the WAR or EAR file, if needed, to the iSeries server.

**Exporting an application to a WAR or EAR file**

Perform these steps to export an application:

1. In any perspective of WDSc, click **File → Export**. The Export - Select window opens.
2. Select an **EAR file** or **WAR file** and click **Next**. In this example, we select a WAR file. See Figure 10-33.

![Export Window](image)

**Figure 10-33** *Selecting the file format for the export operation*

3. On the Export Resources to a WAR File window (Figure 10-34), complete the following steps:
   a. Select from which project you want to export the WAR file.
   b. Select the location to where you want to export the file. It can be a local or network drive.
   c. Select the **Export source files** check box if you want to include the source for your Web application in the WAR file.
   d. Select the **Overwrite existing files without warning** check box if you export the updated WAR file to the same location as with the previous export operations.
   e. Click **Finish**.
Transferring the WAR or EAR file, if needed, to the iSeries server

There are several ways to transfer a WAR or EAR file to your iSeries server, including:

- Using Remote System Explorer
- Mapping a network drive using Windows Explorer
- Mapping a network drive using iSeries Navigator

We discuss the first method by using RSE. Compared to the other two methods, it doesn’t require the iSeries NetServer to be started.

Transferring files by using RSE

You can use Remote System Explorer to transfer WAR and EAR files to your iSeries host. With Remote System Explorer, you can move files between your workstation and your server without using File Transfer Protocol (FTP) or NetServer file sharing.

The WebSphere Application Server Administrative Consoles for Versions 3.5.x and 4.x Advanced Edition require that WAR or EAR files, which it is going to install, are already on the server’s file system (IFS). The Administrative Consoles for Versions 5.x and 4.x Single Server Edition are browser based. WAR and EAR files can be installed directly from your workstation or from a path in your server’s file system if they are already transferred there.

To transfer files using Remote System Explorer, follow these steps:

2. Create a new connection if necessary. When initially creating your WebFacing project, you are prompted to create a connection to your iSeries server if one doesn’t already exist. You can use this connection if that is the server you want to deploy your WAR file to. If you want to create a new connection (for example, if the application server is installed on a different iSeries than where your original 5250 application is located), in the Remote System Explorer perspective, expand **New Connection → iSeries**. The Remote iSeries System Connection wizard opens. See 5.2.3, “Connections” on page 67, for more information.

3. In the Remote Systems view, expand **Local** to explore your local file system and locate the WAR file you exported in the step 3 on page 229.

4. Right-click the **WAR file** and select **Copy**. See Figure 10-35.

5. In the Remote Systems view, expand the connection for your iSeries server. For example, if your connection is called **MyConnection**, then expand **MyConnection → IFS Files → Root file system → the rest of the path**.

   You can choose any directory to store the WAR or EAR file in. A common practice is to use the path `/QIBM/UserData/WebAS51/Base/default/installableApps`, where `WebAS51` is the version of your application server and `default` the name of the instance.

6. Right-click the directory you want to transfer the file to and select **Paste**.

With your WAR or EAR file transferred to your iSeries file system, you can specify its location directly in the application server’s administrative console.

---

**Creating a file share in the iSeries IFS**

Deploying your project to the WebSphere Application Server on your iSeries server requires that you 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 files can be copied from one folder to another on your workstation using the
Windows Explorer.

This section provides basic information about setting up a file share on the iSeries server and
mapping a workstation drive letter to the file share.

WebFacing projects are exported to a WebSphere Application Server directory in the IFS. For
example, if you created a WebSphere Application Server V4 instance for your project, the IFS
directory is similar to /QIBM/UserData/WebASAdv4/instance name.

In iSeries Navigator, you can create a file share of your WebSphere Application Server
instance directory. To do this, follow these steps:

1. In iSeries Navigator, drill down to your instance directory. In our example, this is
   /QIBM/UserData/WebAS5/default.
2. Right-click this directory and select Sharing → New Share (Figure 10-36).
3. Create a new share directory. Complete the required parameters and click OK.

   Figure 10-36   Creating a new share directory

4. After a share is created, expand your connection name → File Systems in iSeries
   Navigator and click File Shares.
5. In the right-hand pane, right-click your file share and select Map Network Drive.
6. In the Map Network Drive window (Figure 10-37), specify sign-on parameters. Click OK.

Now you can use Windows Explorer to move a WAR or EAR file to IFS using a network drive.
10.6.2 Deploying to WebSphere Application Server

This section explains the deployment process for several versions of WebSphere Application Server.

Using the WebSphere Application Server Administrative Console

Follow these steps to install a Web application to WebSphere Application Server V5:

1. Use the application installation wizard to install the WAR file:
   a. Log on to the browser-based Administrative Console. After you install WebSphere Application Server on your server, the URL for the Administrative Console is:

   http://server_name:9090/admin/

   Here server_name is the name of your iSeries and 9090 is the default port for the Administrative Console.
   
   b. In the navigation tree on the left side of the Administrative Console, click Applications → Install New Application.
   
   c. If you want to install the WAR file from the hard drive of your workstation, select the Local path radio button and browse to where you have exported the WAR file. If you have transferred the WAR file to the IFS on your iSeries server, select the Server path radio button and enter the path to the WAR file manually. There is no browse capability.
   
   d. In the Context Root field, type the application’s context root. For example, if you used the same name as you did for your WebFacing project, such as MyProject, users can access your application through a browser with an address such as http://myservername:port/MyProject.
   
   e. Click Next and complete the steps of the application installation wizard.
   
   f. After you complete the installation steps, the console displays the Save to Master Configuration link. Click the Save to Master Configuration link. Then click the Save button.

2. Start the Web application.
   a. In the Administrative Console, click Applications → Enterprise Applications.
   
   b. Select the box next to the application that you just installed.
   
   c. Click the Start button.
WebSphere Application Server 5 Express

To deploy a WAR file to WebSphere Application Server Express (WAS Express) or WebSphere Application Server (WAS) V5.1 for iSeries you can also use the IBM Web Administration for iSeries, which is provided as part of the V5R3 version of the IBM HTTP Server (powered by Apache). It also comes with the V5R2 version of the IBM HTTP Server (powered by Apache), if you have HTTP group PTF SF99098 dated January 2004 or later installed.

To deploy a Web application WAS using IBM Web Administration for iSeries, follow these steps:

1. Make sure the Administrative server for iSeries HTTP Server is started.
2. Point your browser to the HTTP Admin graphical user interface (GUI) at:
   http://your iSeries server:2001
3. Enter a valid user ID and password and click OK.
4. Click IBM Web Administration for iSeries.
5. Click the Manage tab.
6. Click the Application Servers tab.
7. Select your WAS instance from the Server drop-down box in the middle of the header bar.
8. In the navigation tree in the left frame, click Install New Application.
9. On the Install New Application panel (Figure 10-38), follow these steps:
   a. Select the Application is contained in a WAR file radio button.
   b. Click Browse.

   ![Figure 10-38 Installing a new Web application](image)

10. A new browser window opens. Navigate to the WAR file that you want to install and click it. Click OK. The second window closes.
11. In the Context root field, specify the context root. We use /MyProject. Click Next.
12. Click **Next** again.

13. Select the virtual host to be used for the new application and click **Next**.

14. Verify the parameters in the summary page. If they're correct, click **Finish**. The application starts installing. The window show the Manage Installed Application pane.

15. Click **Refresh** periodically and wait until the status of your application changes to **Stopped**.

16. At this point, if your Express instance is stopped, click the **Start** button ( ). If the Express instance is started, select the radio button next to your new application in the Manage Installed Application pane and click the **Start** button.

---

**Deploying to WebSphere Application Server V4.0**

**Important:** If you want to deploy your application to WebSphere Application Server V4, make sure that you select J2EE 1.2 when you create a WebFacing project (see step 3 on page 201).

---

After a Web application is created and exported to the server, you need to deploy your Web application on the WebSphere Application Server. If you are not familiar with the WebSphere Application Server setup, see the WebSphere Application Server Information Center at:

http://publib.boulder.ibm.com/was400/40/AE/english/docs/

The Web application is deployed using the WebSphere Administrative Console. Follow these steps to deploy the WAR file (for WebSphere Application Server 4.0.1):

1. To launch the WebSphere Administrative Console, open a Windows Command Prompt window and change to the WebSphere Application Server bin directory. For example, enter the directory:
   
   C:\WebSphere\AppServer\bin

2. Run the command (WebSphere Application Server Single Server V4 is configured through a browser-based console):

   adminclient iseries hostname bootstrap port

   Note the following explanation:
   
   - `C:\WebSphere\AppServer\bin` is the installation directory containing the `adminclient` command.
   - `iseries hostname` is the host name or IP address of your iSeries server.
   - `bootstrap port` is the port that you chose for your administrative instance.

   The bootstrap port may be left blank, and the default value, 900, is assumed.

   In our example, we enter:
   
   adminclient as01b 999

   The port number to use is the bootstrap port that you selected in the optional step to set up a separate administrative instance.
The Administrative Console then opens (it may take a while).

3. From the Administrative Console, select **Console → Wizards → Install Enterprise Application** as shown in Figure 10-39 to launch the Install Web Application wizard.

![WebSphere Advanced Administrative Console](image)

**Figure 10-39** Opening the wizard to install the enterprise application

4. On the Specifying the Application or Module panel, complete these tasks:

   a. Select **Install stand-alone module (*.war, *jar)**.

   b. Enter the path and WAR file name in the field path (optionally, you can click the Browse button to navigate to the WAR file). In our example, we type:

      `/QIBM/UserData/WebASAdv4/default/installableApps/MyProject.war`

   c. Enter the name of the application. In our example, we enter `MyProject` (see Figure 10-40).

**Note:** The host name for your iSeries server is case sensitive. To see what this value should be, run the Configure TCP/IP (CFGTCP) command from an OS/400 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.

To find out the bootstrap port number, open a job log for the administrative job of your instance (the default job is QEJBADMIN in the QEJBADV4 subsystem). Look for the message `WebSphere application server serverName ready`. The second-level help text in this message lists the bootstrap port number.
d. Enter the name of the context root including a forward slash (/). In our example, we enter /MyProject.

The context root is the top level directory of the Web application, for deployment to the Web server. The context root is combined with the defined servlet mapping to compose the full URL that users type to access the servlet. For example, if the context root is /gettingstarted, the server is AS01B, and the port is 9080, then the URL is:

http://AS01B:9080/gettingstarted

**Tip:** By entering a unique value in the context root, duplicate values are avoided.

e. Click **Next**.

![Specifying the application or module](image)

**Figure 10-40  Specifying the application or module**

5. Continue clicking **Next** on the following panels until you reach the Selecting Application Servers panel (Figure 10-41). The Web module and virtual host combination must be unique.

6. On the Selecting Application Servers panel, select the correct server (in most cases **Default Server**) and click **Next**.
7. On the next panel, click **Finish**.

8. Wait until the Information dialog (Figure 10-42) opens. On this window, click **OK**.
9. Add a new JVM property for the application server (this is for multilanguage support). Click **WebSphere Administrative Domain → Nodes → Application Servers → Default Server.**

10. Complete the following steps in the right pane as shown in Figure 10-43:
   a. Click the **JVM Settings** tab.
   b. Add the following properties:
      - Name: `client.encoding.override`
      - Value: UTF-8

![Figure 10-43 Setting the JVM property](image_url)
11. In the Administrative Console (Figure 10-44), right-click the node and select **Regen Webserver Plugin**. This updates the Web server configuration with the information about the new application.

![Figure 10-44 Updating the Web server configuration](image)

**Note:** To make sure the new changes are incorporated, restart the server.

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.
4. Select **Start**. The application server for your enterprise application is started (if the default server has started).

When the start command is completed, a confirmation window appears as shown in Figure 10-45.

![Figure 10-45 Enterprise application started](image)
Also in the message log of the Administrative Console, the start of the enterprise application is logged, as shown in Figure 10-46.

![Log of the Administrative Console](image)

**Figure 10-46  Log of the Administrative Console**

### 10.7 Creating a WebFacing Portlet project

Rather than a self contained Web application as described in the previous part of this chapter, you can use WebFacing to create portlets to run under an IBM WebSphere Portal server. Portals let you organize Web information and access enterprise services through a single consolidated view of Web usage.

This function is available only in WebSphere Development Studio Client Advanced Edition for iSeries (WDSCE AE), V5.1.2 or later and requires the appropriate version of the IBM Portal Toolkit to be installed on your WDSCE workbench. It is provided as a separate CD with WDSCE AE and you need to install after you installed WDSCE.

When you create a new WebFacing project, you can create a new J2EE 1.3 struts WebFacing portlet project. You can also add a WebFacing portlet to an existing non-struts based portlet project.

For more information about IBM WebSphere Portal server on iSeries see WebSphere Portal Express and Express Plus V5 for the IBM iSeries Server, SG24-6096.

**Attention:** Portlet projects must be run on a portal server or portal test environment. They cannot be run on a Web application server.

### 10.7.1 The Portal Toolkit

The IBM Portal Toolkit provides the capabilities to customize and manage an enterprise portal through integration with IBM WebSphere Portal. With IBM WebSphere Portal you can execute business processes quickly across critical applications, collaborate with portal users inside and outside your e-business, improve employee productivity, cut costs and strengthen relationships with your customers and trading partners. The IBM Portal Toolkit allows an iSeries developer to create, test, debug, and deploy individual portlets and Web content to an IBM WebSphere Portal.

Using the IBM Portal Toolkit, iSeries developers can:
- Develop and debug portlets for IBM WebSphere Portal for iSeries and IBM WebSphere Portal Multiplatforms.
- Use templates to quickly and easily create portlets.
- Preview portlets during development to see how they look.
> Use Portlet wizard enhancements to support action handling, message handling, using Portlet Data, single sign-on function, cooperative portlets and multi-portlet applications development.

> Learn best programming practices from sample portlets.

**Installing Portal Toolkit**

Before you install Portal Toolkit, please read the installation notes on the CD. To install the Portal Toolkit, follow these steps:

1. Insert the Portal Toolkit CD.
2. Follow the on-screen instructions.

After you install the toolkit, the portlet developer and the documentation are integrated into WebSphere Development Studio Client for iSeries.

**10.7.2 Creating a WebFacing portlet project**

To create a WebFacing portlet project perform the following steps on a workstation which has WDSc AE 5.1.2 or later and the Portal Toolkit installed:

1. Open the New Project wizard as described in “Creating your first WebFacing project” on page 261. Click WebFacing and select WebFacing Portlet Project as shown in Figure 10-47.

![Figure 10-47 Select a WebFacing Portlet Project](image)

2. In the Create a WebFacing Portlet Project dialog as depicted in Figure 10-48, by default the Create a new J2EE 1.3 WebFacing portlet project radio button is selected. If you want to add a WebFacing portlet to an existing Web project, select Add a WebFacing portlet to an existing portlet application project and specify the project name.
3. Enter a unique name (sampleportlet in our example) in the Project name field as shown in Figure 10-48.

4. Enter a name for the EAR project in the EAR project field. You can either enter your own name or use the default name, which is the combination of your project name concatenated with the characters EAR (sampleportlet.ear in our example). An Enterprise Application project (EAR project) will be created for you if it does not already exist as shown in Figure 10-48.

5. Check the Add system screen support if you want to enable the display of iSeries system screens within your Web application. See 11.11, “Showing system screens with the WebFacing Tool” on page 408 for more information about this.

6. Check the Use single signon if you want the user to access multiple Web applications with a single user ID and password.

7. Click Next to display the Select Display File Source Members to Convert page.

8. Select which DDS sources to convert in the Select display file source members to convert panel as shown in Figure 10-8 on page 265. Follow these steps:
a. Specify from which OS/400 system the DDS source members should be retrieved. Select an existing connection from the drop-down list or click New to call the New Remote iSeries System Connection wizard to describe how to access an iSeries server you have not used.

b. In the Library filter drop-down box, type the name of the library that contains the file you want to convert or select one of the keywords *LIBL, *USRRLIBL, or *CURLIB to see all libraries in your library list, the user part of it, or the current library assigned to your user profile. You may also type a generic name, such as L186*, to search all libraries with names starting with the characters L186.

c. You may further restrict the number of source files and members to be shown by specifying generic names in the File and Member fields.

d. Click Refresh DDS list

9. If you did not previously default to the iSeries sign-on user ID and password, and if you are not yet connected to the iSeries server, a logon window appears. Enter the iSeries password, and click OK (see “Saving your iSeries sign-on information” on page 69). The libraries entered in the Library filter are retrieved from the iSeries server (Figure 10-8 on page 265).

10. Expand the library tree. The panel displays all source files in the library or a subset of it according to the generic name you entered in the File field.

11. Expand the source file tree. Typically the name would be QDDSSRC, but in our example it is RPGAPP. Depending on what you selected in the Member types field, the panel displays the source members for display files, MNUDDS or both types in the source file.

12. Select the source members you want to convert. In our example: ORDENTD, SLTCUSTD, and SLTPARTD.

Tip: If you accidently selected a wrong member and want to remove it from the list, right-click its name and select Delete from the context menu.

13. Click the >> button to copy the selected source members to the panel on the right. Figure 10-8 on page 265 shows the results.

14. On the Select UIM source members to convert panel (Figure 10-9 on page 266), select the UIM source members that you want to convert and click Next.

15. On the Specify CL commands panel (Figure 10-10 on page 267, specify the CL command to call on the iSeries server. This is the CL command that starts the 5250 application.

a. For CL command, type CALL ORDENTR

b. For Command label, type Order Entry

c. For Invocation name, leave the default of INV1

16. Click Add. In our example, we want to start program ORDENTR. You can add multiple program calls if required.

17. You can select the Sign-on with specified values radio button and provide a user ID and password. With this setting, you don’t need to sign on every time you start your WebFacing application. For testing your new WebFacing application, it is convenient to use these fixed values.

When using the application in production, for security reasons, you may want Web browser users to enter the user profile name and password. In this case, leave the Prompt for signon option selected (see 11.1, “The user profile and password the application runs under” on page 244).
Note: When the WebFacing server starts the interactive job, the initial program (INLPGM) defined for the user profile is not called. You need to make sure the program to be called or the command to be executed is found in the library list.

You may also decide to specify the command name and program name qualified with the library, for example: QSYS/CALL LIB6961/ORDENTR. However, this does not necessarily mean the library list is set to find other objects accordingly, unless your application takes care of that.

See also “Setting the library list to run the 5250 application” on page 214.

For our sample application, regardless whether you use a fixed user profile or let it prompt at execution time, the user profile needs to specify LIB6961 as its current library, because the application does not contain any logic to add that name to the library list.

18. Click **Next**

19. Select the Web style you want to use. In Development Studio Client, several styles are predefined as shown in Figure 10-49.

![Figure 10-49 Choosing a Web Style window](image)
The main difference in the styles that are available for portal applications is the look and feel of the command key buttons. This is because most of the look and feel for your WebFacing portlet will be based on the selected portal theme.

20. The Complete WebFacing project panel similar to the one shown in Figure 10-12 on page 269 opens. On this panel, you create the project and have the option to convert the project either immediately or later. Select one of the options and click Finish.

The project is created and the references to the selected sources are maintained in the workbench. Depending on the number and complexity of display files you choose, it takes several minutes to create the project and between 30 and 90 seconds per display file to convert the project. A progress bar is shown while the conversion process is active.

10.7.3 Testing your WebFacing Portlet application

Development Studio Client includes multiple test environments. You can unit test your application within Development Studio Client, by performing the following steps:

1. Make sure the WebFacing server on your iSeries is started. See “Starting the WebFacing server” on page 194 for a description on how to do so.

2. In either the WebFacing view or the Navigator view, right-click your WebFacing Portlet project and select Run on Server from the context menu as shown in Figure 10-18 on page 277.

3. The Server Selection window as shown in Figure 10-50 opens. Make sure to specify the WebSphere Portal Test Environment to use or create, rather than any of the other WebSphere servers listed here.

4. Click Finish. When you create your very first WebFacing project, the wizard automatically creates an application server in your local test environment for you.
5. The test environment starts. The tool creates a WebSphere Portal v5.0 Test Environment server configuration with the default values.

Figure 10-51 shows the browser window for our sample application.
Figure 10-51  Launch page in the browser window

6. Click Launch. If you did not specify the sign-on information during the development of the WebFacing project, you see the logon page.

7. Enter the user ID and password and click Logon. If you specified the valid sign-on information as part of the WebFacing project, then no logon page is displayed.

8. At this point, the WebFacing run-time servlet, which is part of the generated Web application, contacts the WebFacing server running on OS/400. It passes along the name of the CL command to be called and the user profile and password. The WebFacing server, in turn, starts a new interactive job on the iSeries host and submits the requested CL command. In our example, the command is **Call Ordentr**.

The initial page is displayed in the browser as shown in Figure 10-52.
Figure 10-52  Sample application first page shown on the browser
Chapter 11. Enhancing the WebFacing applications

This chapter explains how you can enhance the presentation and behavior of a WebFacing application after creating a basic WebFacing application, as discussed in Chapter 10, “Using IBM WebFacing Tool to create browser interface for a 5250 application” on page 255.

Some functions described in this chapter are only supported on WebSphere Development Studio Client Advanced Edition for iSeries (WDSc Adv). You must purchase this feature separately from the WDSc Standard edition. Even if you received WDSc as part of 5722-WDS free of charge because of the Software Subscription, you must pay a fee to acquire WebSphere Development Studio Client Advanced Edition for iSeries.

Changes in the second edition of this redbook

This chapter was originally written for WDSc 5.0. Now it has been updated to reflect the changes and enhancements added with WDSc 5.1 in 2003 and WDSc 5.1.2 in 2004, as well as several fixpacks between the major releases. Technical differences from the previous edition of this book are indicated by a change bar at the left of the text. Minor changes such as grammar corrections and slightly different phrasing are not marked.

The base for writing and testing the examples in this chapter is WebSphere Development Studio Client Advanced Edition for iSeries Version 5.1.2.4 (called WDSc AE in the following), that is WDSc AE 5.1.2 with Fixpack 4 applied.

Note, this redbook provides an introduction to the IBM WebFacing Tool. More elaborated topics are covered in Mastering the IBM WebFacing Tool, SG24-6331. However, since most sections of that book were and tested written before WDSc 5.1.2 was available, some of the most important enhancements of WDSc 5.1.2 are covered here. These are:

- “Securing your WebFacing application using single signon” on page 315
- “Enhancing the layout of the launch page” on page 347
- “Showing system screens with the WebFacing Tool” on page 408
11.1 The user profile and password the application runs under

The WebFacing server run time does not require the 5250 application to run under the user profile of the developer who performs the conversion. It can run under any OS/400 profile that has 5250 access. The IBM WebFacing Tool has the capability to either save user ID and password information in the project's properties or let the application prompt for user profile name and password during run time.

These settings exist for the entire project and can be overwritten for each CL command (as shown in Figure 11-1). To activate use profile and password prompting, follow these steps:

1. Open the WebFacing perspective.
2. By default in the upper left pane contains WebFacing, Resource Navigator and Server configuration views. Make sure the WebFacing view is active by clicking the tab at the bottom of the pane.

   Note: If you have closed WebFacing view (or any other views mentioned in this book) accidently, you can restore the original layout of the WebFacing or any other perspective by selecting Window → Reset Perspective from the menu bar.

3. Right-click the project and select Properties.
4. In the Properties window, expand the tree branch under Project.
5. Select the branch representing the CL command whose behavior you want to change.
6. Select one of the following options:
   - Deselect the Override project settings for this command check box for the CL command as shown in Figure 11-1. The behavior then falls back to what is defined in the project properties (Figure 11-2 on page 314).
   - Select the Override project settings for this command check box. Then choose one of the following options:
     - Select the Sign on with specified values option, and specify the user ID and password to be used. This option is convenient, for example, when you develop an application and test it frequently. You may also use it when a WebFacing application is a part of an Internet application where anonymous users have access to the application. In this case, you need to provide a low authority user profile for users coming in through a Web where no login is required or desirable.
     - Deselect the Sign on with specified values option: With this, you are prompted for a user ID and password every time you run the application by clicking the link for this CL command.
Chapter 11. Enhancing the WebFacing applications

When you click Project in the tree view of the properties box, you see the parameters defined at the project level. They provide the default values for all CL commands in this project. They are used unless the Override project settings for this command check box is selected in the properties for a particular CL command.

7. Newly introduced with WDSc 5.1.2 is the Authentication Tab on the right side of the dialog. As shown in Figure 11-2, choose one of the following options:

- **Enable single signon**
  
  This function is available only in WDSc AE, V5.1.2.

  Single signon enables users to access more than one application and multiple platforms using one user ID and password. If you enable single signon, you must also configure Enterprise Identity Mapping (EIM), Lightweight Directory Access Protocol (LDAP), WAS security, and security for your application. EIM is a mechanism for mapping, or associating, a person or entity to the appropriate user identities in various registries throughout the enterprise. To enable single signon for your WebFacing application:

  i. Select the Single signon radio button.

  ii. Click OK. The web.xml file is updated.

  Note: Selecting single signon disables other authentication options on the Authentication settings page.

- **Use OS/400 signon**
  The capability does not require an LDAP server to be set up and the WebFacing server uses an OS/400 user profile to start the interactive job. The browser user is either prompted to enter user profile and password or those values can be entered here in the properties.

After you selected the **Use OS/400 signon** radio button or if you are not using the Advanced Edition, you can choose one or more of the following additional options:

- **Prompt once for user ID and password**
  Use this option if you want to save the settings for user ID and password so that users do not need to enter user ID and password multiple times.

- **Specify signon values:**
  Specify the user ID and password to be used. With this option select, the use does not need to sign on to run your application. The values are stored in the web.xml file within the WebFacing project.

  With this option not selected, the user is prompted for a user ID and password every time you run the application by clicking the link for this CL command.

8. Click **OK** to save your changes and close the properties box. If you click **Cancel**, your changes are not saved.

Figure 11-2  Project-level properties

This change does not require you to convert the WebFacing application again. However, before testing in the workbench test environment, make sure to close the browser and stop the application server. This ensures that the cache is cleared.
11.1.1 Securing your WebFacing application using single signon

The single signon capability is available only in WDSc AE 5.1.2 or later.

Single signon enables users to access more than one application and multiple platforms using one user ID and password. For example, you can integrate secured WebFacing and Web Tools applications which a user can access without requiring separate IDs and passwords for each.

To use single signon for your applications, you need to perform the following tasks:

- Install and configure the Lightweight Directory Access Protocol (LDAP)
- Enable and configure WebSphere Application Server (WAS) security
- Set up security roles and constraints for your Web application
- Configure Enterprise Identity Mapping (EIM)
- Configure your applications to use EIM

To perform these tasks, you will need to install the iSeries Navigator or the IBM Directory Management Tool on a client PC. The following tasks use the iSeries Navigator, which is packaged with iSeries Access for Windows and can be installed from your iSeries server. See “Installing iSeries Navigator” in the iSeries Information Center for details on installing iSeries Navigator. Ensure that you install all of the networking components.

For more information about Configuring the Enterprise Identity Mapping (EIM) Identity Token Connection Factory for WebSphere Application Server see:

Installing and configuring LDAP

A Lightweight Directory Access Protocol (LDAP) server is available as part of OS/400 in the product Directory Services for OS/400. The server provides a network directory which can be accessed by network clients using the LDAP protocol. LDAP defines the transport and format of messages used by a client to access data in an X.500-like directory. Although LDAP does not define the directory service itself, a directory accessed using LDAP is typically called an LDAP directory.

The directory server allows access to a type of database that stores information in a hierarchical structure similar to the way that the OS/400 integrated file system is organized. The LDAP directory server model is based on entries which consist of one or more attributes, such as a name or address, and a type. These types typically consist of mnemonic strings, such as cn for common name or mail for e-mail address.

The directory server is installed with OS/400 using a default configuration. To re-configure the directory server, you must have *ALLOBJ and *IOSYSCFG authorities. To configure LDAP:

1. Using the iSeries navigator, expand Network → Servers → TCP/IP in the tree in the right pane.
2. Right-click IBM Directory Server (Directory for OS/400 V5R2) and select Configure from the context menu. If you have previously configured the directory server, select Reconfigure.
3. Use the Configure Directory Server wizard to configure the directory server. Note the settings you use, such as distinguished names (or suffixes), during configuration. Some of these values are required when you enable security for your applications.

When the wizard completes, your directory server is set up with a basic configuration. For more detailed information about configuring LDAP, see the iSeries Information Center, under Networking, select Directory Services (LDAP) and expand Get Started.
After you have configured your directory server, you can add entries to your LDAP directory using one of the following methods:

- Using the iSeries Navigator, right-click **IBM Directory Server** and select **Tools → Import**. This will import the directory entries from a text file.
- In OS/400, use the `ldapadd` command from Qshell.
- For OS/400 V5R2 or before, use the Directory Management Tool (DMT) to add entries manually.
- For i5/OS V5R3, instead of the DMT, a function to administer **IBM Directory Server for iSeries** has been implemented with the browser based iSeries Tasks, previously called HTTP Administrator function over HTTP port 2001. Note, that the **system application server instance** in WebSphere Express 5.0 needs to be installed.

**Enabling and configuring WAS security**

Before you can configure WebSphere Application Server (WAS) security in the **test environment**, you need to enable the administrative console within your WDSc workbench. To do so, follow these steps:

1. In the Server Configuration view (by default in the upper left pane of the WebFacing perspective), double-click your server configuration.
2. At the bottom of the upper right pane, click the Configuration tab and check **Enable administration console**.
3. Close and Save the configuration.
4. Restart the server by right-clicking your server configuration in the Server Configuration view and selecting **Restart** from the context menu.
5. In a browser window, enter `http://localhost:9090/admin` to open the WAS Administrative Console.
6. Enter a User Id in the Login panel and click **OK**. Note, this does not have to be any registered user at this point, since your test environment application does not have security enabled yet.
7. In the navigation bar on the left pane of the **Administrative Console** expand **Security → User Registries**.
8. Click LDAP to open the **LDAP Security Configuration** page to update the security settings for LDAP as shown in Figure 11-4.

This example uses LDAP, which is required for implementing single signon. If you are not implementing single signon, you can also select Custom or OS/400. If you select the OS/400 registry, authentication is performed using user profiles on your iSeries system.

9. Enter the following values for LDAP security similar to the examples shown in Figure 11-4:
   - **Server User ID** - enter a user ID registered for LDAP that WAS will use to access the LDAP server.
   - **Server User Password** - Enter the password.
   - **Type** - Select SecureWay.
   - **Host** - Enter the host name. You should use the full domain name.
   - **Port** - Use the default port of 389.
   - **Base Distinguished Name (DN)** - enter the base distinguished name of the directory service. (For example, dc=mysystem.mycompany.com.) This indicates the starting point when LDAP searches the directory service.
   - **Check Ignore Case.**
10. Click OK to apply the updates. The Global Security page as shown in Figure 11-5 is displayed.

12. Specify LDAP for the Active User Registry and click OK.
13. Save the changes and close the WAS Administrative Console.
14. Restart the server.
Setting up security for your application

Setting up security for your application requires editing the following tasks:

- Setting up security roles and constraints for your application by editing the Web Deployment Descriptor file `web.xml`.
- Gathering roles used for your application using the EAR Deployment Descriptor.

To define security roles and constraints perform the following steps:

1. Open the Resource Navigator (not the WebFacing view) as shown in Figure 11-6.
2. Expand the Web project folder.
3. Expand the `WebContent` folder.
4. Expand the `WEB-INF` folder.
5. Double-click `web.xml` to open it in the editor.
6. Click the **Security** tab on the bottom of the panel.

7. Click **Add** to insert a new security role.

8. This example uses the role **All application users** to specify all users in the LDAP directory on the iSeries server. Therefore enter the name **All application users** on the right of the security role icon as indicated by the mouse pointer in Figure 11-7.
9. Click the Security Constraints tab at the top of the panel and click Add to define the security constraints for this new security role. The Security Constraints box displays the new constraint as shown in Figure 11-8.
10. Select the new constraint in the **Security Constraints** box and click **Add** in the **Web Resource Collections** dialog as shown in Figure 11-9 to associate a set of Web resources to this constraint. This defines the Web resources that the associated security role will be authorized to access.

![Figure 11-9: Associate a new Web resource Collection to a constraint](image)

11. Enter a name for this set of constraints, for example **All resources**.

12. For this example, check **Get** and **Post** to specify the actions these users can perform. If you check nothing, all actions are allowed.

13. Click **Add** and enter */* for the URL pattern. This specifies that these particular users have access to all Web resources.

14. Click **OK** to add this definition.

15. Click **Edit...** under Authorized Roles to open the Define Authorization Constraint dialog as shown in Figure 11-10.
16. Check All application users to associate this role with the Web resource collections you just defined and click OK.

17. Save and close the Web Deployment Descriptor file web.xml.

18. In the Project Navigator, expand the EAR folder for your project and double-click the EAR Deployment Descriptor file application.xml to open it in the editor.

19. Click the Security tab and then click Gather... button.

20. Select All application users.

21. Check All authenticated users under WebSphere Bindings.

22. Save and close the EAR Deployment Descriptor file.

23. Restart the application.

Configuring EIM

Enterprise Identity Management (EIM) is a mechanism for mapping, or associating, a person or entity to the appropriate user identities in various registries throughout an enterprise. EIM enables administrators and application developers to more easily and efficiently manage multiple user registries across their enterprise. With multiple user registries, each user or entity within the enterprise requires a separate identity in each registry. The requirement for multiple user registries can grow into a large administrative problem that affects users, administrators, and application developers.

EIM enables you to create a system of identity mappings, called associates, between various user identities in various user registries for a person in your enterprise. It also provides a common set of APIs that can be used across platforms to develop applications that can use the identity mappings that you create to look up the relationships between user identities. You can use EIM in conjunction with network authentication service (NAS) to enable a single signon environment.

You can configure and manage EIM through iSeries Navigator. The iSeries server uses EIM to enable OS/400 interfaces to authenticate users using NAS. Configuring EIM involves the following steps:

1. Creating a domain in EIM.
2. Adding the domain to Domain Management.
3. Creating a Source User Registry definition in EIM.
4. Creating a User Identifier in EIM.
5. Creating associations in EIM for the User Identifier.

The following sections briefly describe each of the steps.

**Create a domain in EIM**

1. From the tree iSeries Navigator’s left navigation panel, select your system → Network → Enterprise Identity Mapping → Domain Management → domain → Configuration.
2. Right-click and select the configuration option to start the configuration wizard that will create an EIM domain and join your system to that domain.
   Now add the new domain to Domain Management.
3. From iSeries Navigator’s left navigation panel, select your system → Network → Enterprise Identity Mapping → Domain Management.
4. Right-click and select the add domain option to start the configuration wizard that will add the domain you created to domain management.
   The Identity Token connector requires a source user registry definition entry in EIM that represents the registry that WAS is using for authentication: either a local OS registry or an LDAP registry.
5. From iSeries Navigator’s left navigation panel, select your system → Network → Enterprise Identity Mapping → Domain Management → domain → User Registries.
6. Right-click and select Add Registry to start the configuration wizard that will add the registry to your domain.
   If your application server is hosted on an iSeries system and configured to use the local OS registry, use OS/400 as the EIM registry type.
   If your application server is configured to use the LDAP registry, select 1.3.18.02.33.14-caseIgnore as the EIM registry type.
   Note: The value 1.3.18.02.33.14-caseIgnore is the ObjectIdentifier-normalization form of the registry type whose principals are identified by the LDAP short name attribute. The wizard does not yet handle the descriptive name for this registry type. Support for a descriptive name will be provided in follow-on iSeries Navigator releases
   The Identity Token connector requires a user identifier entry (equivalent to an EIM identifier) in EIM that represents the user of the application.
7. From iSeries Navigator’s left navigation panel, select your system → Network → Enterprise Identity Mapping → Domain Management → domain → Identifiers.
8. Right-click and select New Identifier...
9. Enter an identifier name, such as the user's full name, and click OK.
   To support mapping from one user ID to another, you need to create associations in EIM.
10. Create a target association to represent the user profile on the target OS/400 system perform the following steps:
    a. From iSeries Navigator’s left navigation panel, select your system → Network → Enterprise Identity Mapping → Domain Management → domain → Identifiers.
    b. Double-click the application identifier for the user you want to associate.
    c. Click the Associations tab.
    d. Click Add.
11. Create a source association to represent the user ID that will be used when authenticating (logging in) to WAS:
   a. Click **Browse** and select the registry, for example, ldapreg.
   b. Type the user ID in the LDAP registry that you use to sign in.
   c. Select **SOURCE**.
   d. Click **OK** to add the new association.
12. Click **OK** to save the associations.

**Configuring your application to use EIM**
After you have configured EIM on your iSeries server, you can update the security and authentication settings for your application to use single signon. You will also need to import an external connector resource archive (RAR) file into your project.

1. From the Web or WebFacing Perspective, select your project and click **File → Import**.
2. Select **RAR file** and click **Next**.
3. In the Connector Import dialog, select **Browse** and navigate to
   `<WDSC>\iseries\eclipse\plugins\com.ibm.etools.iseries.webtools.ae_5.1.2\lib`
   The `<WDSC>` in the path is the drive and directory where you installed WDSc, usually `c:\WDSc`. The directory `com.ibm.etools.iseries.webtools.ae_512` may change in future releases of fixpacks.
4. Select the `eimIdTokenRA.rar` file and click **Open** to add it to the Connector Import dialog.
5. Click **Finish** to import the RAR file into the Web project.
   Next, you need to configure authentication settings for your application.
6. Since you are using the WebFacing Tool, configure authentication settings in the Run Time properties for your project. See step 7 on page 313 for more information about setting authentication options for WebFacing applications. For this example, check the Specify EIM resource reference field and enter `idTokenRootRR` as indicated by the mouse pointer in Figure 11-11.

7. Open the Resource Navigator (not the WebFacing view) as shown in Figure 11-6 on page 319.

8. Expand the Web project folder.
9. Expand the WebContent folder.
10. Expand the WEB-INF folder.
11. Double-click `web.xml` to open it in the editor.
12. Now click the References tab to configure the resource references.
13. At the top of this page, click the Resource tab and then click Add to specify the resource reference for the application. This is the same value as the one you specified when you enabled single signon in the project properties for your WebFacing application.
14. In this example, type `idTokenRootRR` for the resource reference and enter `eis/idTokenRoot` in the JNDI Name field under WebSphere Bindings.
15. Save and close the Web Deployment Descriptor file.

Now configure the server for Web security and the idToken JCA connector.
16. From the Server Configuration view, right-click Servers and select **New → Server and Server Configuration** to create a test server. Type a server name (for example, test sso), specify Test Environment for the server type, and click Finish.

17. Double-click the new server configuration in the **Server Configuration** pane and click the **Trace** tab.

18. Check **Enable trace**. Enter `com.ibm.jca.idtoken` at the beginning of the trace string.

19. Click the **Security** tab and then click **Add** to configure the JAAS authentication for the LDAP administrator.

20. Enter the user ID and password for this administrator and click **OK**.

21. Click the **J2C** tab.

22. Click **Add** in the Server Settings panel to add a new J2C resource adapter. The **Create Resource Adapter** dialog is displayed with the resource adapter name `eimIdTokenRAConnector` in the **Resource Adapter Name** field.

23. Click **OK** to close the dialog.

24. Select the new resource adapter name under **J2C Resource Adapters** and click **Add** under **J2C Connection Factories**. The Create Connection Factory dialog is displayed.

25. Enter a name for the connection factory.

26. Enter `eis/idTokenRoot` in the JNDI name field.

27. Set **Max connections** to 6.

28. Select the WAS administrator for the **Container-managed** and **Component-managed** authentication aliases and click **OK**.

29. Under Resource Properties, enter your iSeries host name as the value for `LdapHostName`, the name of your EIM domain (specified during EIM configuration) as the value for `EimDomainName`, and `ldapreg` as the value for `SourceRegistryName`.

   - **LdapHostName** - Enter your iSeries host name.
   - **EimDomainName** - Enter the name of your EIM domain, specified during EIM configuration.
   - **ParentDomain** - Enter the base distinguished name for the directory service. This value was also used during LDAP and WAS configuration.
   - **SourceRegistryName** - Enter the name of your LDAP registry, for example, `WASLDAP.RCHLAND.IBM.COM`.

30. Save and close the server configuration.

31. In the **Server Configuration** view, right-click the test server and select **Restart** to start it.

**Single signon and WebFacing portlet projects**

To use Single signon in a WebFacing application, running in an iSeries Portal server you must perform the following:

- Secure the iSeries portal server.
- Configure the supplied Identity Token resource (eimIdTokenRA.rar) in the WebSphere Application server.
- Secure the WebFacing application.

You secure the Portal server on the Secure Application Server and WebSphere Portal with LDAP wizard page. For information about configuring the Identity Token resource and securing the WebFacing application, and information about EIM configuration, refer to “Configuring EIM” on page 322.
Chapter 11. Enhancing the WebFacing applications

Note: If you are using iSeries Portal Server 5.1, the Create WebSphere Portal wizard has additional pages that will configure the Identity Token resource. When the user signs on to the portal, the user ID supplied is used to map that user ID to the user ID to be used on the iSeries to start the WebFacing application. Therefore, there must be a mapping in the EIM configuration to map this user ID to an appropriate iSeries user profile.

**Testing your application with single signon enabled**

After you have configured security for WAS and for your application, test the application using the test server configuration you created:

1. From the WebFacing or Web Perspective, right-click the main page of your application and select **Run on Server**.
2. Select the server you created to test your application and click **Finish** to publish the Web application to the server.

When the application starts, you will be prompted for a user ID and password. These values are those set up for LDAP security. If you call other Web applications that are configured in the same way, you should only be challenged for a user ID and password once.

### 11.2 Adding more source members to an existing project

You can add a new source to an existing project. Follow these steps:

1. Open the **WebFacing perspective**.
2. In the WebFacing Projects view, expand `project name` → DDS.
3. Select and right-click an object in the data description specification (DDS) directory and click **Add** (see Figure 11-12).
4. Select the source to be added.
5. Convert the project again (see step 7 on page 269).

![Figure 11-12  Adding a more DDS source members](image)
11.3 Using buttons instead of command keys F1 through F24

An important part of converting a 5250 user interface to a browser interface is replacement of the command keys F1 through F24 with browser buttons. There are several ways that you can control the way these buttons appear to a user:

- During the conversion process, the IBM WebFacing Tool analyzes the text constants defined in the DDS to find descriptions for command keys. It uses those constants to generate tables for the corresponding buttons. For example, if the DDS contains the constant F1=Help, the button to invoke the F1 function would have the label “Help”. See 11.3.1, “Command key recognition” on page 329, about how you can control the logic WebFacing uses to recognize text describing command key functions.

- Some command function keys, without corresponding text, may be enabled. In this case, you can provide a default label for each function key on a project level. To learn how to do this, see 11.3.2, “Command key button labels” on page 330.

- To modify the visual appearance, such as color and shape of the buttons, you can change the project properties as shown in “Appearance of command key buttons” on page 336.

- If you want to change the button label, the order in which the buttons appear, or you want to hide a certain function key from the browser user, you can use CODE Designer to do so. See “Modifying the appearance of the command key buttons” on page 377 for more information.

Advanced

Command key actions, a new concept with WDS Adv V5, allows you to override the command key definitions in the DDS or define your own buttons to invoke a WebFacing service or connect to a Web application uniform resource identifier (URI) or an external Uniform Resource Locator (URL).
11.3.1 Command key recognition

The command key recognition property is used to determine the text associated to a command key within the DDS. The text is then used as the text for the button when the DDS is converted. Figure 11-13 shows the Command key recognition patterns panel.

![Command key recognition properties](image)

This panel shows that, when a DDS is converted, the IBM WebFacing Tool looks for a match for `CF\text{xx}=<\text{Text}>`, `CA\text{xx}=<\text{Text}>` or `F\text{xx}=<\text{Text}>`, where `\text{xx}` is the command key number and `<\text{Text}>` is the label of the command key. `<\text{Text}>` is used as the value on the button that appears on the converted Web page. WebFacing searches the text strings within the DDS source to see if there is a match for a recognition pattern.

For example, if the text “F3=Exit F12=Previous” exists within a DDS source, then the text associated with the button for F3 is Exit and the text associated with the button for F12 is Previous.

If you use a different pattern to define command keys, then follow these steps:

1. Open the project’s properties.
2. Click **Command key recognition patterns** (see Figure 11-13).
3. In the Prefix field, select the identifying character or characters.
4. In the Separator field, enter the separating character or characters.
5. Click the **Add** button.
The details are added to the list box. The order in which the prefixes appear in the list box is the order that WebFacing uses to determine the text to use for the button. If you have multiple methods of defining function key text within a DDS, then change the priority order of the prefixes. Select a prefix and then click the **Move Up** or **Move Down** button. The order of the prefixes defines the search order from the DDS. In our example in Figure 11-13, the search is performed in the following order: F → CF → CA.

If no text is found for a command key, then the IBM WebFacing Tool sets the text to be the definition of the key within the DDS, for example, CF03 or CA12.

### 11.3.2 Command key button labels

This property allows you to define the default text that appears for a command key if no text is found within the DDS. If you always use F3 to exit the system, then you can set it up as *Exit* on this display. For any program where F3 is defined as a command key and for which no text is found that matches the command key recognition patterns, then *Exit* is used as text for the button. Figure 11-14 shows the text added for F3:

1. Click **Command key button labels**.
2. From the Command key drop-down menu, select the key (F3 in our example).
3. In the **Button label** field, select or type the text for the button label.
4. Click **Add**.
5. Repeat these four steps for other command keys.
6. When you are finished, click **OK**.

![Figure 11-14 Command key button labels properties page](image-url)
11.3.3 Defining new buttons with command key actions

An enhancement that came with WDSc V5 is command key actions. Command keys are displayed as labeled buttons on the generated Web pages. They are based on equivalent events defined in your DDS source, such as page refresh or help access. You can now define customized action buttons in your WebFacing application to invoke WebFacing services. These command keys override the existing ones and can connect to a Web application URI or an external URL.

The ability to override command keys is advantageous because you do not need to reconvert your application to implement the keys. When you define your own command key button, it overrides the standard function key in your converted WebFacing application, which originates from the iSeries host application. You can invoke the new action by clicking the command key in the application, or pressing the associated function key on your keyboard. You can specify that the button appears only on windows where the related function key is active. Or, you can specify that the button appears on all pages.

Using command key actions, you can link your Struts-based WebFacing service to a command key. You can develop these services manually. Or you can develop your integrated WebFacing service with the Struts builder. This is a visual development environment for building Struts applications. Then link your Struts-based WebFacing service to a command key in the Command key actions property sheet.

Note: Command key actions are only supported on WebSphere Development Studio Client Advanced Edition for iSeries (WDSc AE).

Extending your WebFacing application with command key actions

You can now define customized command keys in your WebFacing application to invoke a WebFacing service, a Struts-based WebFacing service, a JavaServer Page (JSP), or a static Hypertext Markup Language (HTML) page. These command keys override the existing function keys and can connect to a Web application URI or an external Uniform Resource Locator (URL). You have already defined a JSP or service using the Struts builder. In this case, you can use command key actions to invoke that object or service.

For more information about command keys and related WebFacing tasks, see 11.3.1, “Command key recognition” on page 329, and 11.9, “Customizing the WebFacing interface with CODE Designer” on page 361.

To define an overriding action for a command key, follow these steps:

1. In the WebFacing Projects view, right-click your project and select Properties.
2. Select Command key actions under Runtime.
3. Click Add.
4. In the Command Key field, select a function key (F1 to F24).
   - If you want to reference an internal URI, in the Action name field, specify or select a predefined WebFacing service. This also prefills the URI or URL field and the Button label field.
   - If you want to reference an external URL, in the Action name field, enter the action. Then, in the URI or URL field, enter the external URL.
5. In the Button label field, enter a button label for the service. If you selected a predefined WebFacing service, this field is prefilled but you can change it.
6. Select the **Enable only when the command key is active** check box to have your button appear selectively according to the related function key. Otherwise, leave the box deselected to have your button appear on all windows.

7. Click **OK**.

8. Redeploy your WebFacing application.

9. Restart the server.

**Notes:**

- The IBM WebFacing Tool does not give warnings if a function key is already in use.
- If you move any of the IBM-supplied services within your Web application, you need to manually update the URI link to point to the new location.

### 11.4 MNUDDS options

This property determines whether an MNUDDS is converted so that the options become hyperlinks or whether it still operates as a menu. By default, the menu options are converted to hyperlinks. Figure 11-15 shows the properties page for MNUDDS options.

![MNUDDS properties page](image)

**Figure 11-15  MNUDDS properties page**

For examples on converting and enhancing menus based on DDS see Chapter 8 of the redbook *Mastering the IBM WebFacing Tool*, SG24-6331, for example, the section titled “Enhancing record JSPs”.
11.5 Creating your own style

There are several ways to customize the appearance of the WebFacing application in a Web browser:

- For any WebFacing project, you select a Web style as explained in step 7 on page 269. This can be one of the six IBM-supplied styles or a user-defined style.
- If you want to edit the style that is used for the application area and command key elements, you can use the Style properties pages of the WebFacing project.
- To change the layout and the surrounding frame or the application area and command keys, you can modify the cascading style sheet (CSS) files.

Modifications to the style are based on the selected style and apply only to the current project. However, after you change the style for a particular WebFacing project, you can then save the new style and use it for other projects. See 11.5.3, “Working with WebFacing styles” on page 341 for details on how to do so.

WebSphere Development Studio Client for iSeries allows you to modify the look and feel for the Web pages generated by the tool. You can only modify a Web style for an existing WebFacing project.

11.5.1 Using WebFacing properties to modify the style

The easiest way to customize and enhance the appearance of your WebFacing application is through WebFacing properties.

![Properties for MyProject](image)

*Figure 11-16  The WebFacing project properties*
The properties dialog is basically a convenient and easy to use wizard which under the covers modifies several different files and style classes within cascading style sheets. Later in 11.5.2, “Presenting with style: Cascading style sheets” on page 337 we describe how to edit those files manually for more complex modifications.

To modify the style of a WebFacing project through WebFacing properties, follow these steps:

1. Switch to the WebFacing perspective.

2. In the WebFacing Projects view, right-click the project name and select Properties. In the Properties window (Figure 11-16), you should see the parameters that you can modify.

3. In the left part of the Properties window, select DDS field color. You can customize the color for any DDS field color:
   a. Select a desired DDS color from the DDS field color drop-down menu.
   b. In the Field color section of the Properties window, click the .. button to the right of the Text box.
   c. Select the desired color and click OK.
   d. Repeat this operation in the Reverse image color - DSPATR(RI) section, for the Text and Background color parameters.

4. In the left part of the Properties window, select DDS field display attributes.
   In the Display DDS field display attributes panel (Figure 11-17), choose a font and font size for the DDS fields by clicking the appropriate .. button.

![Properties for MyProject](Image)

*Figure 11-17 Changing the font*
5. In the left part of the Properties window, select the **Window** property. In the Window panel (Figure 11-19), follow these steps:
   a. In the Window areas drop-down menu, select **Title**.
   b. Modify the appearance of the title. Under the Foreground section, change the color and the font.
   c. In the Background section, click the .. button to the right of the Image field.
   d. Navigate to the folder where you keep your background images, select one, and click **OK**. This specifies the background image of the page.
   e. For Window areas at the top of the panel, select **Body** and repeat steps b through d.

![Properties for MyProject](image)

*Figure 11-18  Changing the Window properties*
6. In the left part of the Properties window, select the **Subfile** property. Modify the appearance of the subfile in Subfile panel (Figure 11-19):
   a. From the Subfile areas drop-down menu, select **Up arrow**.
   b. Click the .. button to the right of the Image field.
   c. Select **UpArrow.gif** and click **OK**.
   d. Select 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.

![Properties for MyProject](image)

**Figure 11-19**  Changing the subfile appearance

### Appearance of command key buttons

Use the Command keys panel to specify how the text and buttons for command keys defined in your DDS source will display in a Web browser. When converted, command keys are displayed as buttons on your Web pages. Users click these buttons to launch an event defined in your DDS source such as to refresh their page or to find online help. Complete these steps to change the appearance of the command keys:

1. In the left part of the Properties window, select the **Command keys** property.
2. From the Command key states drop-down menu, select one of three available key states.
3. In the Foreground section of the Command keys panel (Figure 11-20), change the font and the font color.
4. Under the Background section, change the button's image. Click the .. button and navigate to the location of the GIF file for a command key button.
5. From the Command key states drop-down menu, select **Rollover** or **Button down**, and change the appearance of the button.
6. Click **OK**. The changes are saved to the project.
7. Test the application. To see any changes, you may have to refresh the Web browser display.

11.5.2 Presenting with style: Cascading style sheets

There are several ways you can create and apply styles to Web pages, including using an external cascading style sheet and specifying styles within individual HTML files. Using style sheets is the preferred method because it allows you to have a consistent design throughout your Web site by separating the content (HTML files) and the design (the CSS). By allowing the sharing of a style sheet file, members of your Web development team can create Web pages in the same style.

The external style sheet is ideal when the style is applied to numerous pages. An author can change the look of an entire site by simply changing one file. Another advantage is that most browsers cache an external style sheet. This avoids a delay in page presentation after the style sheet is cached.

There are many ways to link style sheets to HTML as explained in the following sections. Each way has its own advantages and disadvantages.

**Linking**

You can link an external style sheet to an HTML document through the HTML LINK tag. External style sheets should *not* contain any HTML tags such as `<HEAD>` or `<STYLE>`. The style sheet should consist merely of statements or style rules. You can use a file consisting of the following text as an external style sheet:
The `<LINK>` tag is placed in the document `<HEAD>`. You can apply certain defined attributes to the `<LINK>` tag.

The optional TYPE attribute is used to specify a media type `text/css` for a cascading style sheet. This allows browsers to ignore style sheet types that they do not support:

```html
<link rel="stylesheet" type="text/css" href="styles/chrome/gradient.css">
```

The optional MEDIA attribute, which specifies the medium or media to which the style sheet should be applied, has possible values of:

- **all**: For all output devices
- **aural**: For speech synthesizers
- **braille**: For presentation on braille tactile feedback devices
- **print**: For output to a printer
- **projection**: For projected presentations
- **screen** (the default value): For presentation on non-paged computer screens
- **tty**: For character cell displays (using a fixed-pitch font)
- **tv**: For televisions

Multiple media can be specified through a comma-separated list or the value `all`.

**Note:** Netscape Navigator 4.x ignores any linked or embedded style sheets declared with all the MEDIA values, except screen.

The relationship between the linked file and the HTML document is defined by the REL attribute.

A **persistent style** is always applied when a style sheet is enabled. A persistent style is defined by REL = StyleSheet and the absence of the TITLE attribute. A **preferred style** is unique (only one preferred style can be specified) and is automatically applied. The preferred style is defined by the REL = StyleSheet and the included TITLE attribute.

The user can choose an alternate style to replace the preferred style. The alternate style is defined by REL = "Alternate StyleSheet". However, current browsers generally lack the ability to choose alternate styles.

You can use multiple style sheets to provide a single style:

```html
<Link REL=StyleSheet HREF="basics.css" TITLE="Gradient">
<Link REL=StyleSheet HREF="tables.css" TITLE="Gradient">
<Link REL=StyleSheet HREF="forms.css" TITLE="Gradient">
```

Three style sheets are combined into one “Gradient” style that is applied as a preferred style sheet. The same TITLE must be used to combine multiple style sheets into a single style.

**Embedding**

You can embed a style sheet in an HTML document through the HTML STYLE tag:

```html
<style TYPE="text/css" MEDIA=screen, projection>
<!-- BODY  { background: url(WebFace.gif) blue; color: black }
  EM  { background: red; color: black }
 .note { margin-left: 4em; margin-right: 4em }-->
</style>
```
The STYLE element is placed in the document HEAD. The required TYPE attribute is used to specify a media type the same as with the LINK element. The TITLE and MEDIA attributes may also be specified with STYLE.

**Tip:** Some older browsers do not recognize the STYLE element. They display its contents as though they were part of the BODY. This makes the style sheet visible to the user. You can prevent this by containing the STYLE element within an SGML comment, such as <!-- BODY etc. -->.

You should use an embedded style sheet when a single document has a unique style. When a style sheet is embedded in a Web document, only that particular page is affected by the style settings. If the document also has an externally linked style sheet or sheets and two elements are defined differently in the style sheets, the embedded style sheet takes precedence over the externally linked style sheet or sheets.

**Importing**

You can import a style sheet with the cascading style sheet’s @import statement. This @import statement may be used in a cascading style sheet file or inside the STYLE element:

```html
<STYLE TYPE="text/css" MEDIA="screen, aural, print">
<!-- @import url(http://www.research.ibm.com/images/v11/css/);
     @import url(/stylesheets/Gradient.css);
     DT { background: red; color: black }-->
</STYLE>
```

Other cascading style sheet rules may still be included in the STYLE element. All @import statements must occur at the start of the style sheet. Any rules specified in the style sheet itself override conflicting rules in the imported style sheets.

The order in which the style sheets are imported is important in determining how they cascade. For any elements that are duplicated, within more than one imported style sheet, the last imported style sheet is the one that sets the rule for the defined element.

**Cascading style sheet customization**

This section briefly illustrates the methods used to edit, link, and customize the style of the Web browser using cascading style sheets. Sections 7.3 and 7.4 of the redbook *Mastering the IBM WebFacing Tool*, SG24-6331 provide a more in-depth discussion on what you can do to customize WebFacing styles.

To create a new style or modify an existing style in the IBM WebFacing Tool, a project with an existing style must be created previously. Generally all new styles are created by modifying an existing style and saving the changes under a new name. The new style is available for use by all projects. In the **WebFacing Projects view**, follow these steps:

1. Expand your WebFacing project.
2. Right-click the **Style** folder.
3. Select **Style» Save as**.
4. Give the style a unique name.
5. Click **OK**.

You can then save and modify this user-defined style. This style is then available for subsequent project creation or for selection to replace the current project style.

To edit the style used for the command key elements and the application area, you can edit the Style properties pages or edit the apparea.css file directly using the tools supplied in the integrated development environment (IDE). This file is located in the Web perspective under
the `project name/WebContent/webfacing/styles/apparea` directory. See 11.5, “Creating your own style” on page 333. The changes made through the Style properties pages are applied to the corresponding style class names in the `apparea.css` file. When editing the `apparea.css` file directly, you need to ensure that the modifications are appropriate for the corresponding DDS elements.

The `chrome` directory is located in the Web perspective under the following directory:

`projectname/WebContent/webfacing/styles`

Note, this path has been changed with WDSc 5.1.2 by adding the directory `webfacing`. The `styles` directory contains the user-defined CSS file that controls the layout and frame surrounding the application area and the command keys. This file can be edited within the Web perspective as shown below.

The term “user defined” may be somewhat misleading, since most of the time you start with selecting one of the IBM supplied WebFacing styles, such as `avenue`, `corporate`, `fun` etc. But after that, you can modify the style sheet either indirectly as described with Figure 11-16 on page 333 through Figure 11-20 on page 337 or directly by editing the CSS file or other objects in the `styles` folder.

The properties for styles are stored in two separate directories within the `styles` folder inside the `WebContent` folder of the WebFacing project (see Table 11-1). The contents of these folders will not be overwritten by WebFacing later unless if you select to apply a different WebFacing style.

<table>
<thead>
<tr>
<th>Files that can be edited for style changes</th>
<th>Directory where files are stored</th>
<th>Part of Web page they apply to</th>
<th>Editable using properties pages (such as in Figure 11-16)?</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>apparea.css</code></td>
<td><code>\apparea</code></td>
<td>Application area</td>
<td>Yes</td>
</tr>
<tr>
<td><code>PageBuilder.jsp&quot;&lt;user-defined&gt;&quot;.css</code></td>
<td><code>\chrome</code></td>
<td>Command keys</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Frame and layout</td>
<td>No</td>
</tr>
</tbody>
</table>

**Opening CSS Designer**

Follow these steps to open your style sheet in CSS Designer:

1. In the WebFacing perspective open the WebFacing view,
2. Expand the project,
3. Expand the `Style` folder,
4. Expand the folder with the name of the WebFacing style, you selected, for example `gradient`.
5. Expand `apparea`.

Alternatively you can also use the Web perspective or the Navigator view within the WebFacing perspective:

1. Expand the project,
2. Expand `WebContent`,
3. Expand `WebFacing`
4. Expand `styles`
5. Expand apparea.

**Changing the Chrome**

You can change the layout of the frame (‘chrome area’) by editing the PageBuilder.jsp directly in the Web perspective under the directory:

`projectname/WebContent/webfacing/styles/chrome`

The path shown above has also been modified with WDS 5.1.2 by adding the directory webfacing.

After you changed the one of the style sheets or the PageBuilder.jsp, you should save the modified WebFacing style as described in the next section, so it can be used in other projects.

Section 7.3.3 of the redbook *Mastering the IBM WebFacing Tool*, SG24-6331 provides examples on modifying the chrome area.

### 11.5.3 Working with WebFacing styles

A *WebFacing style* is not a single object, but rather a collection of many files stored in the WebContent/webfacing/styles folder of your WebFacing project. The IBM WebFacing Tool allows you to manage all files necessary to describe the optical appearance of a WebFacing application as a single entity under one name. The following sections describe what you can do with WebFacing styles.

**Selecting a different style for an existing WebFacing project**

To replace a style for an exiting WebFacing application, right-click your project in the WebFacing Project view and select **Style → Select Style** from the context menu as shown in Figure 11-21.

![Figure 11-21  Selecting a new style for an existing project](image)
A dialog opens and asks whether you want to save the current style before replacing it by another style. If you click Yes, the Save as window opens. After you save the current style or if you click No, the Choose Webstyle window opens.

When you replace a style with a different one, the new style should become active immediately after restarting the application server in the test environment. There is an exception to this when you use the Replace text with button property, which is used by the textbuttons style. Setting the property on or off, or replacing with the textbuttons style, requires you to convert the WebFacing project.

**Saving the style to be used for other WebFacing projects**

After you modify one of the predefined styles as described in 11.5.1, “Using WebFacing properties to modify the style” on page 333 and 11.5.2, “Presenting with style: Cascading style sheets” on page 337, you may want to use it in the project where you originally worked with it and in other WebFacing projects. Then you may apply the style to new or existing projects.

Modifications to the style are based on the selected style and apply only to the current project. After you change the style for a particular WebFacing project, you can save the new style and use it for other projects, by performing the following steps:

1. Switch to the WebFacing perspective.
2. In the WebFacing Projects view, right-click your WebFacing project and select Style → Save as.
3. Specify the name for the new style and click OK. Now your style is available for other projects.
4. If you decide to select a new style for you project, right-click your project and select Style → Select Style.
5. You are asked whether you want to save the existing style. Click Yes or No. Clicking Yes saves any changes, and clicking No loses any changes to the existing style.
6. The new window appears with the list of available styles. Select one of the styles and click OK.

Note, by saving the WebFacing style you make it available to future WebFacing projects created on your workstation only. To make the new style available to other developers or simply to save it for recovery purposes, you should use the Export function explained in the following section “The WebFacing Styles view” on page 342.

**The WebFacing Styles view**

You can also use the WebFacing Styles view to manage your own and the predefined WebFacing styles. Click the WebFacing Styles tab at the bottom of the lower left pane of the WebFacing perspective. The WebFacing Styles view opens as shown in Figure 11-22.

The WebFacing Styles view shows all predefined styles. If you saved a modified style previously, you see it here with a different icon for example with MyProject_gradient indicated by the mouse pointer in Figure 11-22. From here, when you right-click the style, you can select from the following actions:

- **Copy as…**: Create a new style based on, but different, from the one you selected
- **Delete**: Delete a user-defined style
- **Rename**: Rename a user-defined style
- **Import**: Import a user-defined style from outside of your WDS webbench
- **Export**: Export a user-defined style to any directory on your workstation or file server
The export and import functions allow your user-defined WebFacing styles to be used by more than one application developer. When you save a new style to be used for other projects on your own WebSphere Development Studio Client for iSeries workbench, all necessary files that define the style are stored in the following directory:

C:\WDSC\iseries\eclipse\plugins\com.ibm.etools.iseries.webfacing_5.1.2.2\styles

Here, 5.1.2.2 is the current release or fixpack of WebSphere Development Studio Client for iSeries, which may change with future releases.

Figure 11-22   WebFacing Styles view

Obviously, this new style is not available to developers who are using other workstations. To make it available to other workstations, you may use the export function to store it on a server, repository, or diskette which can be accessed by other developers. They can then use the import function to use the style with their own projects.

11.6 Using a CL command with variable parameters

You specify the CL command in the Specify CL commands window of the WebFacing project wizard (for example in Figure 10-10 on page 267) or the Add CL commands wizard. At the same time, you can include variable parameters for the CL command to be provided by the browser user before the user clicks the link to start the WebFacing application.

To do so, replace one or more of the parameters following the PARM keyword with variable names starting with the ampersand sign (&) when defining a CL command either during creating of the WebFacing project or when a CL command later:

CALL LIB6961/ORDENTLANG PARM(&LANG)

Figure 11-23 shows the dialog for adding a new CL command.
Add a CL command with variable parameters

Note, the variable parameter does not need to be a CALL parameter. It can be any part of any CL command.

**Attention:** With WDSc V5, the online help and the text on the Specify CL commands window contain an error. The example shows CALL PROGRAM PARM('&PARMNAME'). If you follow that example and place single quotation marks (' ') around the parameter names, this function will not work. To correct this error, simply remove the single quotation marks as shown in the previous command example. This has been corrected in the documentation for WDSc v5.1.2.

As a result, the index.jsp file (previously called index.html as described in “Changing the launch page with Page Designer” on page 348) generated by the WebFacing conversion wizard contains an input field for each parameter field similar to the one shown indicated by
the mouse pointer in Figure 11-24. The user must enter the appropriate data before clicking the launch button.

As you can see in Figure 11-24, the index.jsp page, specially the input field to enter the content of the call variable, does not look very attractive. The reason for that is that it is expected that the links will be placed onto another Web page or portal of your company Web site. You can learn how to enhance the visual appearance of this page using Page Designer as explained in 11.7, “Enhancing the layout of the launch page” on page 347. Page Designer comes as part of WebSphere Studio workbench and therefore with WDSc.

However, not all modifications require to change the JSP. New with WDSc 5.1.2 is the fact that some of the constant text is stored in a properties file.

**Changing the text of the launch page title line**

The automatically created heading “Select program to start” serves its purpose while you are testing and developing one or more WebFacing applications. However, for a user of the application, it does not contain the necessary information (purpose of the application). Nor does it have an appealing look as a page heading. You can change the title line as explained in the following steps:

1. Open the navigator view.
2. Expand the tree under the JavaSource directory of your project as shown in Figure 11-25 to find the following file:
   JavaSource/com/ibm/as400ad/webfacing/runtime/rtmessage.properties
3. Open rtmessage.properties in the Properties File Editor by double-clicking it.
4. Replace the text Select program to start with Order Entry Examples ITSO Rochester.
5. Save the changes and close the Properties File Editor.
6. Stop the server in the test environment.

7. Test the application to see the result.

   Important: Although - or better because - you changed just a text file, you need to restart the Application Server to see the result of you modifications like in Figure 11-26.

Note, that you cannot change any attributes such as font, size, color or the position of the title line by modifying the rtmessage.properties file. In the following sections we explain how can can apply more complex enhancements to the launch page using Page Designer, one of the many tools provided with the WebSphere Studio Workbench which is included with WDSc.
11.7 Enhancing the layout of the launch page

The releases of the IBM WebFacing Tool provided with WDSc prior to version 5.1.2 generated a launch page called index.html to start the WebFacing application. Now, not only the name has changed to index.jsp, but also its layout and implementation.

The most obvious change is the fact that the same style sheet as for the application area for all other pages is used now for the launch page as well. Also, the anchor link to start the application has been replaced by a button (or whatever the WebFacing style defined for the command key buttons) and the second link which was used in the past to start the application in a new browser window has been completely removed.

Less obvious, but very important for understanding the following sections, is the fact that the internal implementation has been changed too. This means, although index.jsp is used to launch the application, the layout of that page is not directly defined within that file. Another file is actually included that defines the layout of the page. That file is called invocations.jsp and it resides in the folder WebContent/webfacing/services/invocation/html/ within the project.

Because of the different implementation, the instructions to enhance the appearance of the launch page that were provided in the previous edition of this redbook do not apply anymore. The following sections have been changed to reflect these differences introduced with WDSc 5.1.2.

Page Designer allows you to work with HTML files, JSP files, and embedded JavaScript. Within the Page Designer editor pane, you can move among three pages that provide different ways for you to work with the file that you are editing:

- The Design page is the what you see is what you get (WYSIWYG) environment that enables you to work with a file.
- The Source page enables you to view and work with a file’s source code directly.
- Preview shows you how the current page is likely to look when viewed in a browser.

You can switch between pages by clicking the corresponding tabs at the bottom of the editor pane. These pages work in conjunction with the workbench Outline and Properties views, tool bar buttons, menu bar options, and context (right-click) menus.

As shown in Figure 10-22 on page 281 and Figure 11-24, the index.jsp file contains only those elements that are functionally necessary to launch the WebFacing application. The following sections provide brief descriptions about the enhancements that you can make with Page Designer to improve the look of the index.jsp file.
11.7.1 Changing the launch page with Page Designer

To open your index.jsp file in Pager Designer, expand the tree under your WebFacing project in the WebFacing view only for one level. Right-click the file and select **Open With → Page Designer** as shown in Figure 11-27.

![Open Page Designer](image)

**Figure 11-27   Open Page Designer**

Note, you can also navigate to `index.jsp` in the navigator view or open the WebFacing project in the Web perspective. Double-clicking the file always opens the last editor used.

The three pages of Pager Designer open in the upper right pane showing different views of the index.jsp file. You can view each page by clicking one of the tabs at the bottom of the pane:

- **The Design page** is the WYSIWYG environment that enables you to create and work with a file while viewing its elements on the page. For example, you can see the graphics that you inserted into the file and continually check the visual presentation of the Web page as you design it. In Figure 11-28 you can see the same file we showed in Figure 11-24, but in the design view now.

This is in contrast to the Source page, which shows the HTML or JSP source code for the file, but not the screen presentation. The Design page provides full visual editing capabilities, including the ability to drag and drop from the Java 2 Platform, Enterprise Edition (J2EE) Navigator view, the Thumbnail view, and Microsoft Internet Explorer.

However, JavaServer Pages (JSP) usually contain dynamic contents besides static HTML tags. For that reason JSPs including the `index.jsp` from a WebFacing project often do not look very similar as they do at execution time. In particular this is due to the fact that the launch button and the input field is defined in `invocations.jsp` which is included.

You do see the representation of the JSP tag which causes `invocations.jsp` to be included, however as indicated by the mouse pointer in Figure 11-28.
The Source page enables you to view and work with a file's source code directly. The Outline and Properties views both have features that supplement the Source page. If you feel comfortable with entering or changing HTML code (or JSP), you can do so in the Source page. Then switch to the Design or Preview page to see how your page may look when completed. When it comes to move and rearrange design elements on the page, the Design view is certainly the better tool. Figure 11-29 shows the HTML code for the index.jsp file.

Both views, the source page and the design page are also synchronized. That is, if you click on a certain element of the design page and then switch to the source view, the source statement defining that element is highlighted as you can see by comparing Figure 11-28 and Figure 11-29. The same is true when switching from the source to the design page.
Note, the color coding of the various syntax elements in the Source view and the fact that the highlighted line corresponds to the position of the cursor in the Design view before you switched the views. This makes switching between views much more useful.
Example 11-1 shows the entire source code.

Example 11-1   HTML code for an index.jsp with input field

```html
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>

<HTML><HEAD>
<!-- Bring in the style definitions used by the current WebFacing application area style -->
<link rel="stylesheet" type="text/css" href="/webfacing/styles/apparea/apparea.css"/>

// Next bring in the .css file to make this page look a little less plain
// Since the WebFacing style used is not know just try and bring in any
// *.css file found in /webfacing/styles/chrome in this webapp
// This can be simplified to a single link to a desired .css file
// like the following:
<link rel="stylesheet" type="text/css" href="/webfacing/styles/chrome/myStyles.css"/>

String chromeDir = config.getServletContext().getRealPath("/webfacing/styles/chrome/");
java.io.File dir = new java.io.File(chromeDir);
String files[] = dir.list();
if (null != files)
for (int i = 0; i < files.length; i++)
{
  if (files[i].endsWith(".css"))
  {
    String hrefpath = "/webfacing/styles/chrome/" + files[i];
    %><link rel="stylesheet" type="text/css" href="/webfacing/styles/chrome/myStyles.css"/>
  %>
  
  // The "index.title" and "index.heading" keys below reference the strings with
  // those keys in the file in the JavaSource directory of your project at:
  // JavaSource/com/ibm/as400ad/webfacing/runtime/rtmessage.properties
%

<script language="javascript">
// Logoff and end host job if browser is closed
function onclose()
{
  var iX = window.event.clientX;
  var sW = 0 - screen.width;
  if (iX < sW)
  {
    document.LOGOFF.submit();
    alert("<bean:message key="index.closeAlert"/>");
  }
}%

</script>

<TITLE><bean:message key="index.title"/></TITLE></HEAD>

<%body onload="javascript:onclose();"%>
<H2 style="text-align:center;"><bean:message key="index.heading"/></H2>
<br/>
<jsp:include page="/webfacing/services/invocation/html/invocations.jsp" flush="true"/>
<html:form action="WebFacing" styleId="LOGOFF"/>
```
The Preview shows how the current page may look when viewed in Microsoft Internet Explorer, but the same restrictions for JSPs we described above for the Design page apply to the Preview page as well. To preview any dynamic content (such as JSP tags), you should use the Run on Server option (Unit Test Environment) from the page’s context menu in the J2EE Navigator view.

**Tip:** If a JavaServer Page is changed while an application is running, you can view the result of the modification immediately. That is, after saving the JSP in the Page Designer, you can switch to the browser window and just click the Refresh button. You do not even have to close the Page Designer.

However, the above does not apply to changes in property files (such as what we described in “Changing the text of the launch page title line” on page 345) or any style sheets (as in “Using WebFacing properties to modify the style” on page 333 and “Presenting with style: Cascading style sheets” on page 337); those require the application server to be restarted.

### The roles of index.jsp and invocation.jsp

As explained at the beginning of this section on page 347, prior to WDS 5.1.2, WebFacing generated index.html, a single static HTML page as a sample launch page for your WebFacing application. Now, not only the fact that a JavaServer Page (JSP) used is new, but also the implementation of having the content built by merging two files: index.jsp and invocations.jsp.

The reason for changing the implementation resulted from two conflicting objectives:

- The need to change the contents of the launch page during a WebFacing conversion because of certain modifications, for example after a new CL command has been added.
- A desire of application developers to enhance the launch page.

In the past, any modifications to index.html might have been destroyed by the WebFacing Tool. With WDS 5.1.2, you can now use a different approach by keeping in mind that invocations.jsp very likely can and will be overwritten, whereas index.jsp mostly remains as it is. Future versions of WebFacing may require the index.jsp be modified again. Therefore it is a good idea to keep a copy of your version of the index.jsp.

The idea is of modifying the launch page in a persistent way removing (or better “commenting out”) the reference to invocations.jsp from within index.jsp (see the line in bold starting with `<jsp:include` in Example 11-1 on page 350). Then copy the code from invocations.jsp to index.jsp. A variation of this approach is to copy the entire file invocations.jsp to a new file and modify the include statement within index.jsp.

Or you can even use an easier approach: Just move invocations.jsp from its original folder directly into the webContent folder (where index.jsp is also located). This method has several advantages:

- Moving the entire file is faster than copying and pasting source lines.
- If you use the Navigator view to move the file, the workbench automatically adjusts the reference to it. So, you do not need to change index.jsp manually.
If, during a future conversion step `invocations.jsp` needs to be rebuilt, you easily can get aware of that by checking the content of the `webfacing/services/invocation/html` folder: If a `invocations.jsp` file does exist, it has been newly created by WebFacing conversion.

After a new version of `invocations.jsp` has been created, you can use the compare function of the workbench, to find out which changes might need to be applied manually your modified version of that file.

The following examples show you how to use the latter method to enhance the launch page.

**Moving the `invocations.jsp`**

Unlike `index.jsp` you cannot find `invocations.jsp` in the WebFacing view. You either need to open the resource Navigator view, and drill down the folders `WebContent`, `webfacing`, `services`, `invocation` and `html` as shown in Figure 11-30. Alternatively you can also open the Web perspective and use the Project Navigator.

![Figure 11-30 Locating invocations.jsp in the Navigator view](image)

To move `invocations.jsp` file to a different folder, simply single click it name and drag it to the `WebContent` folder. The main reason for doing so is to avoid that WebFacing overwrites that file during future conversions. Performing the move as described in the workbench has an additional advantage: Any references to that file, in our case the include in `index.jsp`, can be automatically adjusted. If you want to take advantage of this function click **Yes** in the dialog shown in Figure 11-31.

![Figure 11-31 Fix links for moved or renamed files?](image)

If you test you application after moving the file, it should work the same way as it did before. It is a good to idea to verify if it does before going ahead and change the layout of the launch page.
Chapter 11. Enhancing the WebFacing applications

Note, although you moved this file to the WebContent folder, you still don’t see it in the WebFacing view.

**Content of the invocations.jsp**

The code in index.jsp is written in a way that it can be kept identical for all WebFacing projects, whereas invocations.jsp is dependent on which and how many CL commands you specified for your WebFacing project and whether variable parameters need to be entered by the user.

Example 11-2 shows the source code for our example of Figure 11-26 on page 346 with one variable parameter to be entered for a single CL command.

**Example 11-2 Source of the invocations.jsp for a single CL command with one variable parameter**

```html
<%@ page language="java" contentType="text/html; charset=UTF-8"%>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html"%>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean"%>
<table border="0" width="100%">
<!-- begin INV1 -->
<!-- /* sample programatic invocation
   <a href="WFInvocation.do?inv=INV1&LANG=langValue">Multilingual Order Entry</a>
*/ -->
<html:form action="WFInvocation.do" styleId="INV1" onsubmit="var rv=false;if(false!=document.wfflag){rv=true;}document.wfflag=false;return rv;">%>
<html:hidden name="inv" value="INV1" property="inv"/>
<tr>
<td width="50%" align="right"><html:submit styleClass="launchbutton buttonup wpsButtonText nav-h-normal"><bean:message key="invocations.launch"/></html:submit></td>
<td width="50%" align="left">Multilingual Order Entry</td>
</tr>
<tr>
<td width="50%" align="right">&amp;LANG</td>
<td width="50%" align="left"><input type="text" name="LANG"></td>
</tr>
</html:form>
<tr><td>&nbsp;</td></tr>
<!-- end INV1 -->
</table>
```

The markers <!-- begin INV1 --> and <!-- end INV1 --> within the source define the begin and end of the code for each CL command to be invoked - only one in our example, though.

**Enhancing the launch page**

Ways to enhance the layout of the launch page include:

- Changing constant text
- Changing the attributes of the table cells or rows
- Using different controls for input fields such as drop down lists

Unfortunately, as explained earlier on page 347, the Design page does not always work very well in showing JSPs in WYSIWYG form. Therefore we use the capabilities of the source editor most of the time.

**Tip:** Instead or in addition of using the Design page to verify your modification, you can also start your application in the test environment and leave it running while you change the source of the JSPs. After each change, save it (Ctl-s), switch to the browser and click the refresh button on the upper right.
We continue using the same application that we already enhanced in “Changing the launch page with Page Designer” on page 348. While Figure 11-26 on page 346 shows the full page, we just have a closer look at part of the page below the title line as depicted in Figure 11-32.

![Launch Multilingual Order Entry](image)

**Figure 11-32** Invocation button and input field for CALL parameter

Since any HTML page contains floating text, it is convenient to use a table to left or right adjust certain elements or center a group of fields on the page. For this reason all fields on the launch page are surrounded by a begin and end table tag - the fourth and the last line in Example 11-2.

As you can see in Figure 11-32, the application dependent part on the launch page (related to a single invocation of a CL command) consist of parts arranged in an invisible table with two rows and two columns. From upper left to lower right the cells contain:

- The launch button.
- The title of the button: **Multilingual Order Entry**, which was entered as description of the CL command when the WebFacing project was created or when the command was added. The same text is also used as page title for the application pages.
- The title of the input parameter field: &LANG.
- The input field.

You can improve the layout of that area in several ways. We start with changing the text constants.

**Changing text on the launch page**

Unlike the title text of the whole page which is stored in a properties file (see “Changing the text of the launch page title line” on page 345), the titles for each invocation (CL command) and the input fields are hard coded in the `invocations.jsp` file. To change the text describing the parameter input field follow these steps:

1. Open the Navigator view (not the WebFacing view) of the WebFacing perspective in the workbench.
2. Open the `invocations.jsp` file (which you moved into the `WebContent` folder on page 352) in the Page Designer editor by double-clicking it.
3. Open the Source page by clicking the Source tab at the bottom of the pane.
4. Locate the following statement:
   `<td width="50%" align="right">&amp;LANG</td>`
   The & sign is built by using the variable &amp; to avoid a conflict with the HTML syntax.
   The text is right aligned to show adjacent to the input field which is left aligned in the right column.
5. Replace the string &amp;LANG with a more meaningful text such as:
   *Select your language* <BR> *Bitte wählen Sie Ihre Sprache*
   The HTML tag `<BR>` makes sure that the second part of the text appears in a new line, still in the same cell of the table.
6. Save the file without closing it by pressing Ctrl-s or selecting **File → Save** from the menu bar.

7. If you followed our suggestion, to keep the application running in the test environment, you can now switch to the browser view and the refresh button on the upper right as indicated by the mouse pointer in Figure 11-33.

![Figure 11-33](image)

**Figure 11-33** New text for the input field

As you can see in Figure 11-33 the change became effective immediately. It is always a good idea to verify each modification separately. You can now continue with the next steps.

8. Switch back to the source editor view and locate the following statement:

```html
td width="50%" align="left">Multilingual Order Entry</td>
```

9. Replace the text **Multilingual Order Entry** with:

```html
and click the launch button <BR> und klicken Sie hier
```

10. Again, test the change as explained in step 7 above.

The idea of both changes is to show the instructions, the input field and the launch button in a logical sequence. To accomplish that, you need to rearrange the cells of the table. The two rows and two columns are currently defined by the statements shown in Example 11-3.

**Example 11-3** Table definition after changing text

```html
<tr>
  <td width="50%" align="right">Select your language <BR> Bitte wählen Sie Ihre Sprache</td>
  <td width="50%" align="left"><input type="text" name="LANG"></td>
</tr>
```
11. To move the top row with the launch button and its description to the bottom select all characters starting with the first <tr> and ending with the first </tr> (bold in Example 11-3). Then cut the marked text by pressing Ctrl-X or selecting **Edit → Cut** from the menu bar.

12. Create an empty line after the last </tr> by positioning the bind that tag and pressing Enter. Important: The empty line must be before the </html:form> tag.

13. Paste the lines you cut in step 11 into the empty line.

14. Because mistakes can happen easily when cutting and pasting code, this is a good point to verify the changes again as explained in step 7 on page 355.

15. To swap the columns of the second row, cut the contents of the line containing:

   `<td width="50%" align="left">and click the launch button <BR> und klicken Sie hier</td>`

16. Paste that line after the tag <tr> and before the line containing:

   `<td width="50%" align="right"><html:submit styleClass="launchbutton buttonup
wpsButtonText nav-h-normal"><bean:message key="invocations.launch"/></html:submit></td>`

17. When you test the changes again, you see that we are not completely done yet as shown in Figure 11-34.

   ![Figure 11-34](image)

   *Figure 11-34  Table layout after swapping rows and columns*

   The reason for the odd layout of the page shown in Figure 11-34 is the fact that after swapping the cells in the second row, the right adjust and left adjust attributes must also be changed for each cell.

18. Change the `align="right"` to `align="left"` and vice versa for the two cells in the second row.

**Using a drop-down box instead of the text input field**

Another lack in the layout of the launch page shown in Figure 11-34 is the fact that the user needs to know what values are allowed to enter as parameter and also key it in. This can be made easier by providing a drop-down menu to select possible choices rather than typing a character string.
To do so, we need to know the generated name of the input parameter. This name is based on the name of the variable that you provided in the Specify CL commands window of the WebFacing project wizard as shown in Figure 11-23 on page 344. In our example, the name is &LANG.

Assuming you have still the file invocations.jsp open in the Page Designer you can perform the following steps to change the input field a drop down menu.

1. Open the Source view of the invocations.jsp and position your cursor on the line containing the following code:

   `<td width="50%" align="left"><input type="text" name="LANG"></td>`

   That line defines a text input field with the name LANG. It has that name because we named the CL variable &LANG as explained above.

   In order to create a drop down box, you need to create a Selection list with the same name (LANG) and then delete the original input tag. To do so, you can type the appropriate HTML tags, if you know them, or you can use the facilities of Pager Designer to guide you. To do the latter continue with the following steps.

2. Position the cursor between `name="LANG">` and `</td>`,

3. Press Enter to move `</td>` to a new line and make sure the cursor is still positioned at the left of `</td>` as you can see in the upper right pane in Figure 11-35.

4. Select select Insert → Form and Input Fields → Drop-Down Menu from the context menu after a right-click or from the menu bar as shown in Figure 11-35 and the Insert Drop-Down Menu wizard appears.

![Image](image-url)
5. In the Insert Drop-Down Menu as shown in Figure 11-36, follow these steps:
   a. Enter the name of the input parameter. It must match the name of the existing input field - LANG in our example.
   b. Click **Add** to add the first value of input choices.
   c. In the Name column of the Options table, enter the text **English**. This is the selection name the user will see.
   d. Press the Tab key to jump to the Value column.
   e. Type **ENG** into the Value column. This is the value to be passed with the CL command.
   f. Press the Tab key to jump to the Selected column. Select **True**. This ensures that English is selected by default.
   g. Repeat step b through step e for the name **Deutsch** and the value **DEU** and leave **False** for the Selected column in that line.

   Figure 11-36 shows an example of the completed window.

6. Click **OK**.

7. When you test the your modification now (as we suggested earlier), you see that both, the input field as well as the new Drop-Down Menu appears on the page. To correct this, simply delete the tag:
   ```html
   <input type="text" name="LANG">
   ```
   Do not delete **<td width="50% align="left">** though.

8. Now you can save the `invocations.jsp` file again and test your modifications. The result should be similar as in Figure 11-37.
11.8 Creating a multiple language WebFacing application

You can choose from several techniques to allow users to run the same application with instructions in different languages. Regardless of the technique you choose, you must achieve the following two goals:

- Provide a way for the user to choose the appropriate language
- Allow the application to find the translated information according to the user's language

You can achieve the first goal by using a variable when the users starts the application. This is explained in 11.6, “Using a CL command with variable parameters” on page 343, and 11.7, “Enhancing the layout of the launch page” on page 347.

You can also select the language based on the user profile, but we do not cover that technique here.

A common technique to build multilingual applications on OS/400 is to place all language-dependent objects into separate libraries and then place the language library in front of the normal application library within the library list. The language-dependent objects
can be various types, such as display files, message files, printer files, or even programs, if they contain constants that need to be translated.

11.8.1 Separating language-dependent objects

Ideally, you do not want to duplicate any logic for the sake of translating an application. For our sample application, we chose to use a single library, LIB6961, for the complete application in the most common language, English in our case. Then we created copies of all display files and their corresponding DDS source in a new library for each language, for example LIB6961DEU for the German version.

Another option is to store all text constants in message files (*MSGF) and have the display files retrieve them during execution time. However, it is beyond the scope of this book to describe this technique in more detail.

To allow the user to select the appropriate language, you can use the technique to provide a variable parameter with the program call and simply write a CL program that adds the language library to the job's library list. Figure 11-38 shows such a CL program in the LPEX editor view. It can assign a language library for German, Finnish, and Italian.

![Figure 11-38   CL program ORDENTLANG](Image)

11.8.2 Language dependent objects in the WebFacing project

The DDS for the display files for each language must all be converted and included with the WebFacing project. See 11.2, “Adding more source members to an existing project” on page 327, to learn how to add more DDS members to an existing project.

Changing the DDS object mappings to use explicit library names

At this point, you may run into a potential conflict. For example, for each display, a file exists with the same name as for the other languages. You need to make sure that WebFacing run time also selects the correct object. You can do this via the DDS object mappings page in the WebFacing project properties:

1. Right-click your WebFacing project in the WebFacing Projects view and select Properties.
2. Under Run time, select DDS object mappings.
3. Click the **Replace &LIB with source library** button. This ensures that the original library for each DDS member is respected at run time and the same library is selected in which the display resides.

See 12.6, “DDS Object Mappings” on page 428, for more information about the object settings.

**Sample application in a multilingual environment**

Figure 11-39 shows the original sample application in the English language.

![Order Entry application in the English language](image)

When the user starts the application from the index.jsp launch page, a drop-down menu is provided to select the language as shown in Figure 11-37 on page 359. Figure 11-40 shows the result of selecting German.

![Order Entry application in the German language](image)

**11.9 Customizing the WebFacing interface with CODE Designer**

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. Programs that use DDS source to describe 5250 screens can be accessed with a Web browser after the DDS source is
converted with the IBM WebFacing Tool. Use Web Settings when you want to manipulate the Web presentation of individual screens and individual fields within screens. If you want to change the Web presentation of multiple screens or of an entire WebFacing project, use Style properties.

Web Settings enable you to change the way your pages appear in a browser. If you want to modify your pages after conversion, you can use a text editor or a Web design tool such as WebSphere Studio. An advantage of customizing your pages with Web Settings is that the instructions for the customizations are embedded as comments in your DDS source. Since Web Settings become part of your source, changes that you make are not lost if you later reconver a WebFacing project.

You can find more examples on using Web settings with Code Designer in Chapter 8 of the redbook Mastering the IBM WebFacing Tool, SG24-6331.

**Important:** After you add Web settings to your DDS source to be used by the IBM WebFacing Tool, do not change the DDS source with SDA anymore. If you do use SDA for updating the DDS source, the Web settings added by CODE Designer are damaged or removed. However, you can use any source editor, such as the Source Entry Utility (SEU) to change the DDS source, as long as you make sure not to modify comment entries starting with A*%%WB in columns 6 to 11. This has been corrected in V5R1 with PTF SI11184 and V5R2 with PTF SI15260.

### 11.9.1 A look at CODE Designer

CODE Designer is a feature of CODE (formerly known as the separate product CODE/400). You can use it to design display windows and reports. This feature is a complete design environment. It integrates such tools as the CODE Editor, the Error List, and the Program Generator so that you have quick and easy access to them as you design (see Figure 11-41).

You can think of CODE Designer as a modern version of the Screen Design Aid (SDA). Both tools allow you to develop OS/400 display files so that you can immediately see how the panels you designed appear on the user’s display. CODE Designer, however, uses all the advantages of a graphical user interface (GUI) for the programmer. Therefore, it has many advantages over SDA.

For example, when working with CODE Designer, you can see the image of the panel you are creating or changing and the resulting source code (DDS). At the same time, you can see a layout tree of the DDS source and many other views as shown in Figure 11-41.
11.9.2 Opening CODE Designer

To access CODE Designer, in the Development Studio Client for iSeries IDE, follow these steps:

1. Open the WebFacing perspective.
2. Expand the WebFacing project in the WebFacing Projects view. Make sure you are not in the Navigator view.
3. Expand the DDS node.
4. Right-click the node representing the display. That is the DDS source member you want to work with.
5. Select Open with.
6. Click **CODE Designer** as shown in Figure 11-42.

![Figure 11-42 Opening CODE Designer directly from the WebFacing perspective](image)

CODE Designer opens in a separate window similar to the one shown in Figure 11-41. It contains the selected DDS member.

**Note:** As of WDSc V5, CODE Designer is not yet integrated with the Eclipse-based Studio workbench.

Alternatively, you can also open CODE Designer from the Remote System Explorer perspective by performing the following steps:

1. Expand the connection icon for your server.
2. Sign on to your system.
3. Browse to the DDS member that you want to work with.
4. Right-click the member and select **Open With → CODE Designer**.

Help for CODE Designer is available from the CODE Designer Help menu.

### 11.9.3 Accessing Web settings

After a DDS source member is loaded into CODE Designer, you can access Web settings by clicking the icons in CODE Designer’s DDS Tree. The DDS Tree is located on the left-hand side of the CODE Designer window. Each icon in the tree represents a DDS object within the display file as shown in Figure 11-43.

Web settings exist for various levels in the hierarchy of DDS objects, such as the file, record, or field level. See Table 11-2 for a list of object types that have Web settings and the icon that represents it.

Web Settings are also accessible 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 as shown in Figure 11-43.
11.9.4 Web settings overview

Web settings are a feature of CODE Designer. When a DDS is open with CODE Designer, each record and field has a Web Settings tab. The type of record or field that is selected determines the Web settings that are available.

CODE Designer allows you to design DDS displays graphically. In CODE Designer, each DDS object is represented by an icon or button. For example, there are buttons so that you can easily create named fields or text constants for your DDS display. The Web Settings available for each DDS object vary depending on the object that you are working with. Table 11-2 shows the DDS objects and the Web settings that are available for each object. Web settings are stored as comments within the DDS and are identified with %%WB.

The input capable fields in Table 11-2 also encompass input and output fields.

Table 11-2 Available Web settings for a DDS object

<table>
<thead>
<tr>
<th>DDS object</th>
<th>Available Web settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Record</td>
<td>&quot;Key labels&quot; on page 366</td>
</tr>
<tr>
<td>Subfile Control Record</td>
<td>&quot;Key order&quot; on page 366</td>
</tr>
<tr>
<td></td>
<td>&quot;Insert into script&quot; on page 371</td>
</tr>
<tr>
<td></td>
<td>&quot;Display size&quot; on page 367</td>
</tr>
</tbody>
</table>
Web setting descriptions
The following sections describe each of the possible Web settings. Later in 11.9.5, “A sample application for WebFacing” on page 372, we give examples of how to use those settings.

Key labels
For function or command keys in your DDS source, you can specify the text labels to be used in your Web pages to represent the keys. To change a label, follow these steps:

1. Select a label from the Labels already defined list.
2. In the New Label field, enter the text you prefer for the key.
3. Click Change.

To remove a label that you defined, follow these steps:

1. Select the label from Override labels in the Web setting list.
2. Click Remove.

See “Modifying the appearance of the command key buttons” on page 377 for examples.

Key order
For function or command keys in your DDS source, you can specify keys to be visible on your Web pages and the order in which the keys are displayed. To change with this setting, follow these steps:

1. Select the Change order and visibility of command keys check box.
2. To make a key visible, from the Hidden list, select the key.
3. Click the arrow button to move it to the Display order list.

### DDS object

<table>
<thead>
<tr>
<th>DDS object</th>
<th>Available Web settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date field</td>
<td>“Visibility” on page 367</td>
</tr>
<tr>
<td>Time field</td>
<td>“Location” on page 367</td>
</tr>
<tr>
<td>Timestamp field</td>
<td>“Program-defined HTML” on page 367</td>
</tr>
<tr>
<td>Message constant</td>
<td>“User-defined HTML” on page 367</td>
</tr>
<tr>
<td>Date constant</td>
<td>“Appearance and Text” on page 368</td>
</tr>
<tr>
<td>Time constant</td>
<td>“Insert HTML” on page 368</td>
</tr>
<tr>
<td>User constant</td>
<td>“Appearance and Text” on page 368</td>
</tr>
<tr>
<td>System constant</td>
<td>“Insert HTML” on page 368</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DDS object</th>
<th>Available Web settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Named field (input capable)</td>
<td>“Visibility” on page 367</td>
</tr>
<tr>
<td>Continued field (input capable)</td>
<td>“Location” on page 367</td>
</tr>
<tr>
<td>Source reference field (input capable)</td>
<td>“Program-defined HTML” on page 367</td>
</tr>
<tr>
<td>Database reference field (input capable)</td>
<td>“User-defined HTML” on page 367</td>
</tr>
<tr>
<td>Database reference field by selection (input capable)</td>
<td>“Appearance and Text” on page 368</td>
</tr>
<tr>
<td>Text constant</td>
<td>“Insert HTML” on page 368</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DDS object</th>
<th>Available Web settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Named field (output)</td>
<td>“Visibility” on page 367</td>
</tr>
<tr>
<td>Continued field (output)</td>
<td>“Location” on page 367</td>
</tr>
<tr>
<td>Source reference field (output)</td>
<td>“Program-defined HTML” on page 367</td>
</tr>
<tr>
<td>Database reference field (output)</td>
<td>“User-defined HTML” on page 367</td>
</tr>
<tr>
<td>Database reference field by selection (output)</td>
<td>“Appearance and Text” on page 368</td>
</tr>
<tr>
<td>Text constant</td>
<td>“Insert HTML” on page 368</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DDS object</th>
<th>Available Web settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hidden field</td>
<td>“Send to browser” on page 372</td>
</tr>
</tbody>
</table>

a. The icon at the left of the object description is the same as you see in the CODE Designer workbench.
4. To hide a key, select the key from the Display order list and click the arrow to move it to the Hidden list.

If more than one key is available in the Display order list, you can change the order in which it is displayed, by selecting the key, then clicking the up or down buttons. If the Change order and visibility of command keys check box is not selected, defined keys are shown in numeric order.

**Display size**
The Display size setting is available if a secondary display size is selected for your display file. You set secondary display sizes through the properties window of the file node icon in the DDS Tree. Use the Display size Web setting to change the size of the HTML table that will be used to display your application in a browser.

**Visibility**
The options that are available for the Visibility setting are *Hide Field* or *Show Field*. If *Show Field* is chosen, the field is displayed on your Web pages.

**Location**
The options that are available for the Location setting are Change span and Change position. After your DDS source is converted with the IBM WebFacing Tool, your application is positioned on a Web page using an HTML table. The HTML table has the same number of rows and columns as your DDS display. That is, it is a 24 by 80 or 27 by 132. With Location settings, you can change how your fields are positioned on the HTML table after your source is converted.

- **Change span** allows you to indicate the number of rows and columns in the HTML table that are given to your field. A potential use for Change span is using a graphic in place of the text values that your program originally used to populate the field. For example, the original program may require seven columns for text for the field, but the graphic that you want to use only requires two columns.

- **Change position** allows you to indicate the row and column in the HTML table that indicate the starting position for your field.

**Program-defined HTML**
With the Program-defined HTML setting, select the *Use field value as HTML* box if the value returned by the field is HTML markup that you want to use in your page.

**User-defined HTML**
User-defined HTML allows you to specify HTML text or tags that can be displayed with or in place of the value for your field. To work with this setting, select the *Specify HTML* check box. In the entry area, enter the HTML tags you want to use. If you want to use the value for your field as part of the HTML, click the *Insert field value* button.

An example of how you can use this setting is if you want to display the value for your field in a bold font. To do this, use the `<BOLD>` tag with the `&{FIELD.value}` text.

With this example, edit the entry area so that it displays `<BOLD>&{FIELD.value}</BOLD>`.

Chapter 11. Enhancing the WebFacing applications 367
**Appearance and Text**

The options that are available for the Appearance and Text setting are:

- **Change style class** allows you to specify a style sheet class to be used with the field. As an example, you may want to display all numeric data with a particular font or color and define a style sheet class numeric for this purpose. If you have a customer number field in your program such as custno, you can select this option and enter the name for your numeric style sheet class.

- **Override field value** allows you to specify text that you want to display in place of the returned value for your field.

- **Override constant text** allows you to specify text that you want to display in place of the value set for a text constant.

- **Hide characters** allows you to indicate a range of characters in your field that you do not want to display. For example, if your screen displays a menu of options for a user to choose from and each option is listed numerically (such as: 1. User tasks - 2. Office tasks), you may want to hide the first two characters so that only the text for the option is displayed. In an example like this, you choose **Hide characters** and specify a range from 1 to 2.

**Insert HTML**

The Insert HTML setting provides a way to customize your fields by adding additional HTML. HTML can optionally be added before, inside, and after the HTML tag that is automatically generated for the field. For example, you may have an input field for employee numbers and, at your company, all employee numbers are prefixed with 64-. You can make the field easier to use by adding such HTML as:

- **Before**
  
  `<strong>64- prefix mandatory</strong>`

- **Inside**
  
  `value="64-"`

- **After**
  
  `<a href="help.htm">Click here for more information on this field</a>`

In this example, the HTML **before** the field provides some information about the field and uses the `<strong>` tag so that the information is emphasized with a bold font. Since the HTML generated by conversion uses the `<input>` tag for input fields, **inside** the `<input>` tag, we can add the HTML attribute `value="64-"` so that the field is pre-populated with the employee number prefix and your users are not required to enter it. **After** the field is the HTML for a link that users can click to get more information about the field.

Ensure that you thoroughly test the pages that you are altering with the **Insert HTML** setting. Changes that you make can conflict with the HTML that is generated by the WebFacing conversion. To work around this, convert your program without using Web Settings, look at the HTML or JSP source that is generated to see what changes can safely be made with Web settings, apply the Web settings, and then re-convert.

**Options for VALUES**

The Options for VALUES setting is available for input fields that are specified with the VALUES keyword. Fields that are specified with the VALUES keyword are displayed on Web pages as a single selection box. The options displayed in the selection box are the values given as parameters to the VALUES keyword in your DDS source.
The Options for VALUES Web setting allows you to change the label or text that is displayed to the user. For example, if one of the VALUES parameters that your program is expecting is "item8", you can change the text that is displayed in a browser to something more descriptive such as "8 gauge wire". While the text displayed to the user is "8 gauge wire", the value returned to your program is the same "item8" that the program expects.

To use this Web setting, select the Option texts for VALUES keyword check box. To set a new label for a VALUES parameter, select the value, enter the text that you want for the label in the New Label field.

**Create graphic**

Use the Create graphic setting if you want to display an image in place of your field. An example use for this setting is a product database that includes images of inventoried items.

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, you may enter http://www.ibm.com/image.gif. You can populate the File name field by entering a value directly or by using the Browse button to locate a local file. Local images that you specify are copied to the ..\Web Content\images\ directory of your WebFacing project during the WebFacing conversion. The images in this directory are published later to the WebSphere server when you deploy your project with the Export wizard.

*Width in pixels* and *Height in pixels* can be used if you know the dimensions of an image. Adding width and height information can sometimes decrease the time required to render Web pages that contain images.

If there are 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, you can add these values to the File name field by selecting the field from the Choose a field list and clicking Insert field value. For example, if you have &{FIELD1.value} that returns the value bicycle1, you can easily refer to a graphic bicycle1.gif that you may be working with on your workstation. In this example, populate the File name field from the Choose a field list and then add the .gif extension. You must manually copy files, to the ..\Web Content\images\ directory, that you identify by using the Choose a field list Create hyperlink.

Use the Create hyperlink setting so that your field is displayed as a hyperlink. Three different approaches are available for creating hyperlinks:

- Specify static URL
- JavaScript hyperlink
- Action hyperlink

If you want to override the browser's settings for hyperlink appearance, select the Override browser's hyperlink appearance with DDS appearance check box. If you use the Create hyperlink setting along with the Create Graphic setting, your field can be displayed as a graphic that also acts as a hyperlink.

There are some conditions where hyperlinks that you create with Web settings do not display in a browser, for example:

- The hyperlink is in a record that is not currently the top window or on the top layer.
- A function key that you associated with the link is not currently active.
- A referenced field that is part of the link is not in an active record or is not readable. For example, the record is protected by other records that use the PROTECT keyword.
- You override the browser's hyperlink appearance with DDS appearance.
Most browser's follow a convention for the color and appearance of hyperlinks on Web pages or have settings for hyperlink appearance that users can change. For example, unvisited links may be displayed in blue text, and visited links may be displayed in purple text. If you want the browser to display the hyperlink with the same appearance as is used for the field in your DDS, select the **Override browser's hyperlink appearance with DDS appearance** check box.

**Specify static URL**

If there is a specific URL that you want to be displayed for a field, select **Specify static URL**. The output value of your field is the text that is displayed as a hyperlink. The location loaded by the browser when a user clicks the link is the value that is entered in the URL field. If you want the link to launch a separate browser window when it is clicked, enter a name for the window in the Target field.

If there is a field value in your DDS that you want to be part of the URL, select a field from the Choose a field list and click the **Insert field value** button to add it to the URL field. An example use of the Insert field value is if there is a URL to which you want to add additional parameters.

For example, you may have a servlet at a location such as the following example:

http://mysite.com/mywebapp/servlet/MyServlet

The servlet is expecting arguments for a variables called parm1 and parm2. You can use the Insert field value to help construct the URL. In this example, your final URL may look like this example:

http://mysite.com/mywebapp/servlet/MyServlet?parm1=&{FIELD1.value}&parm2=&{FIELD2.value}

Here &{FIELD1.value} and &{FIELD2.value} are parameters you added using Insert field value. They represent values that change at run time.

**JavaScript hyperlink**

Enter the name of a JavaScript function that you want to be called when the link is clicked. For example, enter myFunction(). The function can be one that you entered using the Insert into script Web setting, a function that you defined in an external JavaScript file, or a function that is included with WebFacing in the webface.js file.

If you create your own external JavaScript files, place them in the usr directory of your WebFacing project. This directory is located in the path **project name**\Web Content\ClientScript\usr. Any scripts that you place in this directory are available to your WebFacing application at run time. If you have a short inline JavaScript function, you can enter the entire function in the JavaScript hyperlink field.

**Action hyperlink**

Action hyperlink creates a link that, when clicked, can position the cursor to a field, perform a submit, or a perform a combination of the two. The Position cursor to the field and Submit functions both have values that you can specify:

- **Position cursor to the field**

  When the link is clicked, the run-time value returned for cursor location is the location of the field you specify. You can select a field from the drop-down list or manually enter the name of a field.

  This value does not refer to the cursor as it is displayed by the browser. It refers to the WebFacing run-time value for current cursor location in the DDS record. If you also want a value returned for the field where the cursor is positioned, choose a field or enter a value in the Enter data field.
Entries in Position cursor to the field can take the following form:
- \&{FIELD}: Field in the current record.
- \&{RECORD.FIELD}: Field in the specified record.

Entries in the Enter data field can take the following form:
- \&{FIELD.value}: The value of the field in the current record.
- \&{RECORD.FIELD.value}: The value of the field in the specified record.

You can also manually enter values in the Enter data field.

Submit

If the Submit check box is selected, you can choose a Function key or JavaScript function call submit action for the hyperlink. For example, from the Function key drop-down list, you can choose ENTER, LOGOFF, or a function key that you defined in your DDS.

If you want to use a function key from another DDS record, manually enter the value for the function key. When a user clicks a link that you created for a function key, the program executes that function. In the JavaScript function call field, you can enter the name of a JavaScript function that you want to execute when the link is clicked.

The function can be one that you entered using the Insert into script Web setting, a function that you defined in an external JavaScript file, or a function that is included with WebFacing in the webface.js file.

If you create your own external JavaScript files, place them in the usr directory of your WebFacing project. This directory is located in the path project name\Web Content\ClientScript\usr. Any scripts that you place in this directory are available to your WebFacing application at run time.

You can combine the Position cursor to the field option with the Submit option. Do this if you want to set the value for a specific field and submit that value back to your program. To do this, follow these steps:

a. Select the Position cursor to the field check box.
b. Choose a field from the drop-down list (or enter a field).
c. In the Enter data field, choose a \{FIELD.value\} (or enter a specific value).
d. Select the Submit check box.
e. Select the Function key radio button.
f. Choose ENTER from the Function key drop-down menu.

For information about the entry format to use for Position cursor to the field and the Enter data field, see “Position cursor to the field” on page 370.

Insert into script

Use the Insert into script setting to customize the JSP files that are created for regular records or subfile control records. For example, if you have a subfile control record LISTCTL, the WebFacing conversion creates a corresponding JSP file LISTCTLJavaScript.jsp.

The code that you add in the Scripting field of the Insert into script setting is added to the file LISTCTLJavaScript.jsp when it is generated. This setting allows you to enter arbitrary JavaScript or JSP scriptlets into your converted programs. You can use this to initialize fields or perform any work required for the specific record format.

For example, you can add code that sets a field's value if its initial value is null:

```java
if(\&{CID}.value=="")
{
    \&{CID}.value="0001";
}
```
In this example, the code &{CID} refers to a field CID that was selected using the Insert field name button. Fields under the Insert field name button are enclosed with an ampersand (&) and two brackets ({}), for example &{FIELDNAME}.

Other examples include JSP code that determines the IP address of a WebFacing user and JavaScript that displays the time of day of the user’s system. You can use Insert into script along with the Send to browser setting. For example, you can use values determined by your scripts to populate fields for which you used the Send to browser setting.

If there is an existing field in your DDS that you want to use as part of your script, add it to your script using the Insert field name button. For example, you can select an H or hidden field and use it in your script so that a value can be assigned to it.

**Send to browser**

The Send to browser setting can be used for H or hidden DDS fields. This setting modifies the generated JSP files for your records. For example, if you have an H field in a record called LISTCTL, when this setting is used, an HTML hidden input form element is generated in the JSP file LISTCTL.jsp.

Hidden input form elements begin with the HTML code `<INPUT TYPE="hidden"`. Using Send to browser enables an H field to become part of the browser environment and for values to be assigned to it based on that environment. For example, the IP address of a user is determined by a script you added using the Insert into script setting.

**11.9.5 A sample application for WebFacing**

For this chapter, we use three displays as examples of what you can perform with Web settings. The application is a customer inquiry or update. It is part of an example library that you can download as explained in Appendix A, “Additional material” on page 679. The displays are a subfile selection display (Figure 11-44), a detail maintenance display (Figure 11-45), and a window selection display (Figure 11-46 on page 374).

After you download the example library, you can find the programs in the LIB6961 library. Add LIB6961 to your library list and enter the following command to run the program on the iSeries:

```call cstsflr```
## Customer Listing

Type options, press Enter  
2=Change   4=Delete   5=Display

<table>
<thead>
<tr>
<th>Opt Id</th>
<th>Dist W/H</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>1 0001</td>
<td>Wilfried BLANKERTZ</td>
</tr>
<tr>
<td>0002</td>
<td>1 0001</td>
<td>Pentti S RASANEN</td>
</tr>
<tr>
<td>0003</td>
<td>1 0001</td>
<td>Aleksandr NARTOVICH</td>
</tr>
<tr>
<td>0004</td>
<td>1 0001</td>
<td>Camille E PAQUAY</td>
</tr>
<tr>
<td>0005</td>
<td>1 0001</td>
<td>David FAIRRALL</td>
</tr>
<tr>
<td>0006</td>
<td>1 0001</td>
<td>Janet Y WILLIS</td>
</tr>
<tr>
<td>1001</td>
<td>1 0001</td>
<td>Annie R OAKLEY</td>
</tr>
<tr>
<td>1002</td>
<td>1 0001</td>
<td>Dean ASCHEMAN</td>
</tr>
<tr>
<td>1003</td>
<td>1 0001</td>
<td>Petri NUUTINEN</td>
</tr>
<tr>
<td>1004</td>
<td>1 0001</td>
<td>Neil WILLIS</td>
</tr>
<tr>
<td>1005</td>
<td>1 0001</td>
<td>Gary C Mullen-Schultz</td>
</tr>
<tr>
<td>1006</td>
<td>1 0001</td>
<td>Bob W MAATTA</td>
</tr>
</tbody>
</table>

F3=Exit   F6=Add   F12=Previous

---

## Add Customer Details

Enter Customer Details and Press Enter

Customer number . . . . . . . . . . . . . .  
District number . . . . . . . . . . . . . . +  
Warehouse id . . . . . . . . .  
First name . . . . . . . . . . . .  
Initial . . . . . . . . . . . .  
Last name . . . . . . . . . . . .

F3=Exit   F4=Prompt District   F6=AddCustomer   F12=Cancel

---

Figure 11-44 Customer list

Figure 11-45 Customer details
11.9.6 Initial appearance of a new WebFacing application

This section shows how WebFacing converts screens if no further enhancements are applied to them. Then we demonstrate several techniques to enhance the presentation and functions of the new Web application.

This WebFacing application uses the following source members from LIB6961/RPGAPP: CSTSFLD, CUSTDTLD, CSTDTLD, and SLTDSTD. If you convert the source with the IBM WebFacing tool you will see the results that follow.
The display files are converted using the `corporate1` style sheet. The display files have all command keys activated at the file level. The application handles whether the keys are active depending on the logic of the program. Figure 11-47 shows a converted subfile display file and Figure 11-48 shows the converted detailed page.
Figure 11-49 shows the converted district selection page.

![Customer Listing](image)

As you can see on the left-hand side of these pages, all the command keys are created as buttons on the Web page. DDS makes all the command keys active, and the IBM WebFacing Tool interprets these as being required. The text that exists on some of the buttons was retrieved from the DDS during conversion. The buttons that have the text CFxx have no text associated with them in the DDS.

If the DDS only defines the command keys that are available for this page, then buttons are created only for those command keys.

**Note:** If a command key’s availability is defined by an indicator within the DDS, then the button for the key appears on the Web page only when the indicator is on. If the indicator is off, then the button does not appear.

The command key’s availability may be defined by the application and not the DDS. For example, there is no indicator to condition CFxx or CAxx. In this case, the button always appears on the Web page if it is defined in the DDS.

11.9.7 Changing Web settings on a record format level

The following sections show examples of enhancing the presentation of an entire record format.
Modifying the appearance of the command key buttons
CODE Designer can modify the command keys that are to be displayed by the converted display file. In the WebFacing project, right-click the DDS that you want to modify as described in 11.9.2, “Opening CODE Designer” on page 363. As shown in Figure 11-50, all command keys are defined at the file level. We want to define the function keys to behave differently on each panel, so we must change the Web settings for each record format.

![Figure 11-50 DDS opened in CODE Designer](image)

When a record format is selected in the CODE Designer workbench, the Web settings are:
- Key labels
- Key order
- Insert into script
- Display size

Note: The Key labels and Key order Web settings are available on all record formats within a DDS, except the subfile record. However, the command keys of the last record format written by the program are in effect on the Web page. Therefore, you must only modify the Web settings for the record format that is going to be the last one written to the page. In the DDS used in this example, the header, footer, and message control formats do not change. Only the subfile control record must be changed since it is always the last record written.

For the sections covering Key order, Key label, and Insert into script, the subfile control record CSTSFLC is selected within CODE Designer.
**Key labels**

The Key labels Web setting allows the text that is associated with the command key to be overridden when the DDS is converted using the IBM WebFacing Tool. For example, “F12=Previous” is the text that appears on a green screen. When left as the default, the button appears with the text “Previous”. The label can be overridden to use the text “Back” instead by defining a Key label.

To do this, follow these steps:

1. In the Labels already defined list box, select **F12=**.
2. In the New Label entry box, enter **Back** and click the **Change** button.

“F12=Back” now appears in the Override labels in Web setting list box as shown in the bottom right-hand pane in Figure 11-51.

![Figure 11-51  Changing the command Key label](image)

**Key order**

The Key order Web setting defines the order of the command or function keys and which command or function keys are to be displayed. The DDS shown in Figure 11-50 has all the command keys active for this display file. This program uses F3, F6, and F12. Buttons are required only for these command keys. The function keys Page Up, Page Down, Home, and Print are not to be shown on the Web page.
To create buttons for the three command keys, follow these steps:

1. Select the **Change order and visibility of command keys** check box.

2. In the Command keys Hidden list box, click one of the command keys that is required, for example, CF03.

3. Click the > button. CF03 now appears in the Display order list box. Repeat this step for the other command keys. Figure 11-52 shows the F3, F6, and F12 command keys being selected.

You can also use this option to hide command key functionality from the Web page. If F7 was available on the 5250 version of a program and the Web page did not want to expose this functionality, then do not select the F7 key in this Web setting. Select only the command keys that you want to expose to the Web page. If a user presses F7 on the Web page, nothing happens because the command key is not defined.

The Command keys list box contains the command keys that are defined within the DDS. There is no ability to add new command keys to the Web page that would function within the iSeries application.

This Web setting also allows you to redefine the order of the command keys. By default, the IBM WebFacing Tool processes command keys in the order in which they are defined in the DDS. If the order is to be changed for the WebFacing version of the display, you select a
command key in the Display order list box and then click the Up or Down button to move the command key to the desired position.

Function keys can be processed in the same way as command keys. The function keys to display are shown in the Display order list box at the bottom of the Web Settings page.

**Insert into script**

The Insert into script Web setting allows you to add script to the generated JavaScript for the record. You can use this to initialize fields or perform any work required for the specific record format. You can use the contents of the fields in the record format within the JavaScript by selecting the field and clicking the Insert field name button.

The Insert field name button includes the field value into the JavaScript for the Web setting. At conversion, JavaScript is created that interprets the field name and creates a reference to the ID of the field that is generated by the IBM WebFacing Tool.

Figure 11-53 shows how you can initialize the option field in the customer subfile program by entering a 5 in the first entry in the subfile.

```
A                                      DSPSIZ(24 80 *DS3 27 132 *DS4)
```

**Display size**

If the DDS is defined with a secondary display size, then this Web setting is available. The Web setting allows you to specify which display size is to be used for generating the Web page. The display size defines the area that is available for fields to be placed on the Web page. If additional fields are to be used on the Web page, then it may beneficial to select the larger display size.

The following code within the DDS allows this setting to be displayed:

```
A                                      DSPSIZ(24 80 *DS3 27 132 *DS4)
```

Figure 11-54 shows the Web setting for Display size.
Chapter 11. Enhancing the WebFacing applications

How CODE Designer modified the DDS source

After the changes are saved, DDS is updated with comments to reflect the Web settings that changed. Figure 11-55 shows the new entries that are added to the DDS to store the Web settings that changed. The %%WB comments are added after the last field in the subfile control record.

```
A 8 23'Name'
A DSPATR(HI)
A*%%WB 9  REC 12=Back
A*%%WB 11  REC CF03 CF06 CF12
```

Figure 11-55 Changes to DDS to reflect Web settings

The changes to the DDS are made as comments. This allows the DDS to contain the settings required for the IBM WebFacing Tool without having to recompile any of the application code on the iSeries. It also enables the details to be retained for subsequent conversions of the display files using the IBM WebFacing Tool.

**Important:** After a DDS is changed by CODE Designer to include Web settings, the DDS can no longer be maintained using Screen Design Aid. If you attempt to modify the DDS using SDA, an error occurs advising you that it cannot be maintained due to WebFacing comments within the DDS.

However, recently a PTF was developed to address this issue. If you plan to continue using SDA, load and apply PTF SI11184 on OS/400 V5R1 or SI15260 on V5R2.

Reconverting the DDS

After you make changes to the command keys, you can reconvert the DDS using the IBM WebFacing Tool. You only need to convert the DDS member for the display file you just changed with CODE Designer. This saves time when working with large WebFacing projects.
Figure 11-56 shows the converted Web page after the command keys are modified in Web Settings.

Note: At run time, both the command keys and the buttons activate the required function. Pressing the command key highlights the appropriate button. Pressing an invalid command key, such as one that is not defined for the Web page, has no effect on the Web page. If the application has a command key defined and this is removed through the Key order Web setting, then the command key is not valid for the Web page.

11.9.8 Field-level Web settings

This section shows how you can use the Web settings for a field to hide or modify fields to change the look and feel of the display file. A field can be selected in CODE Designer by expanding the record formats. Each of the fields associated to the record format are displayed. Table 11-2 on page 365 shows the Web settings.

Figure 11-57 shows the Web settings for an input field.
Figure 11-58 shows the Web setting for an output field.

![Visibility Web setting](image)

Figure 11-58 Web settings for an output field

Figure 11-59 shows the Web settings for a hidden field.

![Location Web setting](image)

Figure 11-59 Web settings for a hidden field

**Visibility**

The Visibility Web setting allows you to determine whether a field is to be displayed on a Web page. It is the same as the hidden field concept in DDS. However, it allows you to override the value for fields that are defined as being display fields within the DDS.

Two options are available on the Visibility Web setting: the Hide field and the Show field. The default is Show field. If Hide field is selected, then no other Web settings are available for this field. The remaining Web settings are hidden, so they cannot be accessed. If changes are made to the other Web settings and Hide field is selected in the Visibility Web setting, then the other changes are removed. Figure 11-60 shows the Visibility Web setting with the hide field selected. All other Web settings are no longer available.

![Visibility Web setting](image)

Figure 11-60 Visibility Web setting

**Location**

The Location Web setting allows you to redefine where a field is positioned on a browser and the number of rows and columns that the field occupies on the page.

Two options are available for the Location Web setting:

- Change span
- Change position

**Change span** allows you to change the length that a field occupies on a display. When the Change span check box is selected, then the values of the rows and columns fields show the values from the DDS. These fields show how many rows and columns that the field occupies.
Changing the values in this field is typically done when a field is replaced with a graphic. See Figure 11-61. The graphic can be a smaller length than the original field it is replacing. By changing the span of the field, other fields can now occupy the space that would be used by the original field length without being concerned with overlay issues.

In Figure 11-61, the text “Type options, press Enter” is replaced with the IBM logo as a graphic. The text field was 25 characters long. The field size is reduced to 10 because the graphic takes only 10 columns within the browser.

*Change position* allows the field to be moved to a different position on the page when it is displayed on the Web page. When the Change position check box is selected, the values of the rows and columns fields contain the values from the DDS (see Figure 11-62).

There are some potential pitfalls with moving the position of a field. The pitfalls concern the overlaying of fields. If a field is moved to another position, it may overlay another field. One of the fields does not display because it is overridden by the other field. Similarly if the field being repositioned is currently overlaying another field, then the program may no longer get the correct value for the field that remains. The field can only be repositioned to the limits of the DDS size, for example, 24 x 80 or 27 x 132.

![Figure 11-61 Location Web setting showing Change span](image1)

Figure 11-61 shows the text field that contains the options text that was repositioned and now occupies space that was previously occupied by the text “Type options, press Enter” (as seen in Figure 11-56 on page 382).

In Figure 11-62, the text “2=Change 4=Delete 5=Display” moves to the previous line. This is overwritten and not displayed if the span of the graphic is not reduced (as shown in Figure 11-61).

![Figure 11-62 Location Web setting showing Change position](image2)
Figure 11-63 shows changing the Location settings (compare with Figure 11-56 on page 382).

Program defined HTML
The Program defined HTML allows the application to define HTML to be processed by the Web page. The contents of the field are interpreted as HTML when the page is processed. If the field does not contain valid HTML, then the Web page may not be created as expected. To activate this Web setting, you select Use field value as HTML.

For the customer listing application, this can be used to create a check box on the Web page that contains the list of district IDs on the customer detail program. The values contained in the list box need to be generated by the application by reading the appropriate file and creating the HTML to create the select box. Example 11-5 shows the HTML that needs to be generated. The application needs to generate this string and add the <option> tag details.

Example 11-5  HTML to create a select box for the district ID

```html
<select name='did' onclick="setFieldValue('11_CSTDTL1$_z_zDID$','this.form.did.options[this.form.did.selectedIndex].value')">
  <option label='Dist1' value='1'>MIDWEST</option>
  <option label='Dist2' value='2'>MID-EAST</option>
  <option label='Dist3' value='3'>FAR WEST</option>
  <option label='Dist4' value='4'>MID-WEST</option>
  <option label='Dist5' value='5'>SOUTH</option>
  <option label='Dist6' value='6'>WEST</option>
  <option label='Dist7' value='7'>CANADA</option>
  <option label='Dist8' value='8'>NORTH</option>
  <option label='Dist9' value='9'>EAST 2</option>
  <option label='Dist10' value='10'>MID-SOUTH</option>
  <option label='Dist11' value='11'>MID-SOUTH</option>
</select>
```
Details of the code changes required to create the HTML are shown in the CSTDTLR program in the downloaded example. A new field $DHTM is added to the DDS. This field is conditioned to only be available when the program is called via the WebFacing server. This is done by using the WebFacing environment API QqfEnvironment. Figure 11-64 shows the Program defined HTML Web settings for this field.

The field is defined as 1024 to contain the HTML statement for the check box. When left at this length, the subsequent fields on the Web page do not display because the field overwrites them. The Location Web setting is modified to change the span of the field from 1024 to 10 as shown in Figure 11-65. This allows all other fields to be displayed after the check box is implemented on the page.

Figure 11-66 shows how the converted Web page appears.
User defined HTML
User defined HTML allows you to override or modify the HTML tags generated for a field. Selecting Specify HTML allows you to enter HTML for this field. The Insert field value button inserts the value of the field into the HTML that is entered. This is seen in the entry box as &FieldValue.

The HTML entered here overrides any HTML that is normally generated for the field by the IBM WebFacing Tool. If the HTML that is entered does not include the field value, then the field is no longer available as part of the Web page. For example, if the option field on the customer subfile display is to be replaced by a check box, then the following HTML may be used:

<input type='checkbox' name='option'>

This HTML replaces the default HTML that WebFacing may create for the option field. By doing this, the option field is no longer available in the Web page, and the value is not sent between the iSeries and the Web page.

An example of why you may want user defined HTML is if the field is to appear in a bold font. The HTML to achieve this is:

<BOLD>&FieldValue</BOLD>

Figure 11-67 shows how using the bold font appears in the Web settings.

![Figure 11-67](image)

**Figure 11-67** Using user defined HTML to apply bold to the account field

Appearance and text
This Web setting allows you to change the format of the field, to override the values of the field, and hide parts of the field. Three options are available for Appearance and text:

- Change style class
- Override field value or Override constant text
- Hide characters

**Change style class**
This option is used to associate a style sheet class to a field. The style sheet class allows you to define the attributes for a field such as color or font. Then you can apply the class to various fields rather than updating individual fields with the same details.

Example 11-6 shows the style class being changed for the customer name field. For this example, a new style class called `customer_Name` is added to the apparea.css file. Figure 11-68 shows the class definition for `customer_Name`. The `wf_red` class is a style class that already exists within the cascading style sheet apparea.css. This class changes the color of the text to red.
Example 11-6  Customer name style class

.customer_Name {
  font-family : Verdana;
  font-size : 15pt;
}

See 11.5.2, “Presenting with style: Cascading style sheets” on page 337, for details about cascading style sheets.

Figure 11-68  Changing the style class

Figure 11-69 shows the resulting changes to the Web page. The customer name is now displayed in a 15 point font and is red.

Figure 11-69  Web page after style class changes are made
Override field value

This applies to fields that are not text constants. For text constant fields, this option is named Override constant text. The value entered in the entry box is used to output to the Web page regardless of the value returned from the program on the iSeries. The length of the override value cannot be greater than the field's length defined in the DDS.

Note: This option is available for output only fields. If the field is an input field with the values keyword or an input/output field with the values keyword in the DDS, then the option is not accessible in the Web setting.

If the field does not have the values keyword and is an input field, or an input/output field, in a subfile record, then the modified data tag is not set. This means that if the subfile is read using a read changed record, no record is returned if other changes are not made in the subfile.

Figure 11-70 shows the options field that is being overridden to provide a default value of 0001 for the customer ID field in the customer detail display file.

Override constant text

Text constants can be overridden to display different values than what is defined in the DDS. For non-text constant fields, this field is named Override field value. The text that is entered cannot be longer than the current length of the field.

Figure 11-71 shows the text of the valid options being changed so that 4=Delete now displays as 4=Remove.
**Hide characters**

The Hide characters option allows you to define a set of characters that are not displayed on the Web page. An example of this is a list of options within a DDS that contain such text as “1. first option”, “2. second option”.

You can use the Hide characters option to remove the number from the text strings. In this case, enter 1 for the From position and 3 for the To position. This hides the first three characters in the text string. That leaves the Web page showing “first option” and “second option” as the text strings.

In the example customer subfile program, you can change the heading from “Customer Listing” to “Customer List”. Figure 11-72 shows the heading text for the page being changed to remove the last three characters of the text.

**Note:** If the field is an input field with the values keyword or an input/output field with the values keyword in the DDS, then the option is not accessible in the Web setting.

![Figure 11-72 Changing the Hide characters option](image)

Figure 11-73 shows the Web page after the heading is changed. Compare this to Figure 11-63 on page 385 where the heading still indicates “Customer Listing”.

---

Archives
Insert HTML
This Web setting allows you to customize a field by adding additional HTML to the field. The Web setting has three options:

- **Before**
- **Inside**
- **After**

- **Before** inserts HTML tags before the WebFacing generated HTML for a field. **Inside** allows HTML to be added within the tags generated by WebFacing for a field. **After** allows HTML to be inserted after the HTML generated by WebFacing for a field.

For an input field, the HTML generated by the IBM WebFacing Tool uses the `<INPUT>` tag to describe the field. If the field is an output field, then the `<SPAN>` tag is used. The Example of generated HTML text box shows how the HTML appears as changes are made in the Before, Inside, and After entry boxes. See Figure 11-74. You can enter HTML tags into one, or all, of the options of this Web setting for this field.

**Before**
This option allows you to enter HTML text that appears before the HTML that defines the field in the Web page. As you enter text in the Before entry box, you see it is reflected in the example box beneath it.

An example of using this is to change a field, so that, when it is clicked, the F4 command key is submitted. In the detailed display in Figure 11-45 on page 373, the ability to perform a prompt on a field is signified by plus sign (+) next to a field. The DDS uses the RTNCSRLOC keyword to return the field name to the program. The program then invokes the appropriate prompt program. For the customer detail display file, the F4 prompt key is available for the
district field. The HTML shown in Figure 11-74 allows you to click the plus sign (+) to invoke the F4 command key.

The HTML allows you to move the cursor to the first position in the district field and then submits the F4 command key. The cursor is moved as the program needs to receive the field name as part of RTNCSRLOC. If the cursor is not repositioned, then no field name is returned since the cursor is positioned on the text field ‘+’.

![Figure 11-74 HTML inserted before the current field](image)

**Inside**
This option allows you to add additional HTML text within the `<INPUT>` or `<SPAN>` tag. As you enter text in the entry box, it is reflected in the Example of generated HTML text box beneath it.

For example, you can click the field to invoke the F4 command key. Figure 11-75 shows the HTML and how it appears as part of the generated HTML. This enables the user to click the district field, and then it submits the F4 command key.

![Figure 11-75 HTML inserted inside a current field](image)

**After**
The After option is similar in function to the Before option, except the HTML appears after the generated HTML for the field.

An example of this option is to add a URL to the field to access help text. Figure 11-76 shows the HTML that is used in this example.
Figure 11-76  HTML inserted after the current field

Figure 11-77 shows the updated Web page. The F4 command key is invoked by clicking either the district field or the plus sign (+) next to the district field. The District number field has a hyperlink to the URL next to the district field using text “Help text”.

Figure 11-77  Web page after the HTML changes

Options for VALUES

The Options for VALUES Web setting is only available for input fields that have the VALUES DDS keyword associated with them. This Web setting allows you to redefine what the user sees as choices for the field. When the VALUES keyword is used for a field and the cursor is passed over the field in the Web page, then a check box appears with the options that are available. The values shown may only be symbolic names and can be changed to be more meaningful for the user.

The subfile example in Figure 11-44 on page 373 has valid values of 2, 4, and 5. Figure 11-78 shows the Options for VALUES Web setting with the new text values for options 2 and 4.
Figure 11-78  Creating new labels in Options for VALUES

Figure 11-79 shows the result of changing these values to use descriptive text for the options. Options 2 and 4 show the text changed, whereas 5 is left as it is.

Create graphic

The Create graphic Web setting is used to replace the field with a graphic on the Web page. The graphic that is used can be a fixed name, such as picture.gif. Or it can be a variable name that uses the contents of the field to determine the graphic to be used.

If the graphic is a fixed file that is to be used, then enter the location and the name of the file in the File name field. Alternatively you can click the Browse button and search for the file. The File name field contains the complete path to the file that is selected. The file is copied to the project name\Web Content\images\generated\ directory, where project name is the name of the WebFacing project. When this project is published to the application server, the images are transferred to the server as part of the WAR file.
In Figure 11-80, the file used is C:\MyProject\IBMlogo.gif. When the display file is converted, the IBMlogo.gif file is copied to the project name\Web Content\images\generated\ directory.

If the graphic is to be a variable, based on the value of a field, then include the field as part of the File name field. To do this, select a field from Choose a field and click the Insert field value button. The field selected is added to File name. The contents of the field are used to select the name of the graphic to be used. If the graphics are contained in a GIF file, then you need to add .gif to the File name field after the field that is selected.

For example, if a district ID field is displayed, then it may have the values of Dist1, Dist2, or Dist3. The graphics that are associated with these are dist1.gif, dist2.gif, and dist3.gif. You must manually copy all graphic files that are to be used by this method to the project name\Web Content\images\generated\ directory. Figure 11-81 shows district ID on the customer listing subfile display being replaced by a graphic.

Using links instead of entering values into option fields
The Create hyperlink Web setting allows you to turn a field into a hyperlink to either access a URL or to set a field value and submit.

The Override browser's hyperlink appearance with DDS appearance option determines the appearance of the hyperlink. If you do not select the check box, then the default browser
appearance is used for the hyperlink. If you select the check box, then a style class is applied to the hyperlink to give it the appearance of a normal DDS field.

The Specify URL option allows a URL to be accessed when the link is clicked. The URL can either be a fixed URL or a variable URL that is determined by the contents of a field. If the URL is fixed, then enter the address in the URL field.

Figure 11-82 shows the URL being created as a fixed reference to http://www.ibm.com. In this case, you click the IBM logo, which opens the specified page.

If the URL is to be based on field contents, then select the field or fields from the Choose a field list box and click the Insert field value button. The fields are included in the URL field as \{field1\}. You can then add any additional details as required such as “.com”. Figure 11-84 shows how the district code on the customer subfile is changed so that when the graphic is clicked, the Web site is accessed.

The other option available is Set field value and submit. This allows you to create a hyperlink for a field on the Web page, so when you click this link, a value can be set for another variable. There must be one field that is input capable in the record format that is being modified. If there are no input capable fields, then you cannot select a field to receive the value. When the link on the Web page is clicked, then the field that is identified is updated with the value specified, and the page is submitted.
Figure 11-84 shows how you can change the window selection display (defined in Figure 11-46 on page 374) so that the selection field can be hidden and the code field is changed to a hyperlink. You must change the Visibility Web setting for the OPT selection field so that the field is hidden. When the district code is clicked in the selection Web page, it has the same effect as typing 1 in the selection field and pressing Enter on the 5250 application.

Send to browser as a hidden input element
This Web setting is available for hidden fields in a DDS record. The field, by default, is not made available to the Web page. If the option is changed to YES, then the field is created as part of the JSP, and its contents can be used within the Web page.
Figure 11-86 shows the Web setting for the hidden first name field being made available on the customer listing subfile page.

<table>
<thead>
<tr>
<th>Selected DDS</th>
<th>Web Settings</th>
<th>Comments</th>
<th>Error list</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 11-86  Allowing a hidden field to be sent to the browser

### 11.9.9 Adding extra logic to the Web browser with JavaScript

You can add extra logic to the Web pages using JavaScript as well as modify the DDS Web settings to customize the Web pages to be generated by the IBM WebFacing Tool.

By default, each WebFacing project contains the `webface5xxx.js` file. This file exists in the `projectname/WebContent/webfacing/ClientScript` directory. The `webface5xxx.js` file is part of the IBM WebFacing Tool. You should not modify this file because any changes made to it are overwritten when an upgrade is done to the tool.

If you want to create your own JavaScript for a project, then create a new file in the `projectname/WebContent/webfacing/ClientScript/usr` directory. The file can have any name you choose since the IBM WebFacing Tool checks this directory for files and includes them automatically. There can be multiple files in this directory. All of them are included as part of the application. For an example see “JavaScript functions” on page 401.

**Tip:** If you have a JavaScript utility file that you want to make available to all projects, then create the file in `x:\WDSc\iseries\eclipse\plugins\com.ibm.etools.iseries.webfacing_5.x.x\ProjectTemplate/Web Content/webfacing/ClientScript\usr`. Whenever a new project is created, this file is included as part of the project.

### 11.9.10 Adding JavaScript to an application

You can change the option field on the subfile display to use radio buttons and check boxes to show how JavaScript can be implemented within a WebFacing project. In this section, we use the original display files in the example library (downloadable from the Web) with none of the Web setting changes made in previous sections.

Figure 11-47 on page 375 shows how the display appears after the first conversion with the IBM WebFacing Tool. Figure 11-87 shows how the display appears after the changes are made. The options text at the top of the display is replaced with radio buttons that are linked so that only one of the buttons is active at any time. The option field on each subfile record is replaced by a check box that shows which records are to be processed. When you click Enter, then all the subfile records that have a check box marked are processed with the option of the radio button that is selected.
Web setting changes

To change the option field to use radio buttons and check boxes, various Web settings are changed for different fields.

The Visibility setting for the option entry field was changed to hide the field. The “Opt” column heading and “Type options, press Enter” were also hidden. See Figure 11-88.

Important: Before you create JavaScript for an application, make sure that you convert the application and view the generated source first. Details of the fields on the Web page exist in the <record format>.jsp and <recordformat>JavaScript.jsp files after the conversion. Details and field identifiers from these files are necessary when you create JavaScript to provide new functions.

The text that displays the options that are available for a user have the Web setting for User defined HTML changed as shown in Example 11-7.
Example 11-7  HTML to create a radio button for options

```html
<INPUT type='radio' name='options' value='2' onclick='setOptions(this.form.opt,
this.form.options, <%=zOrder%>);'> Change  
<INPUT type='radio' name='options' value='4'
onclick='setOptions(this.form.opt, this.form.options, <%=zOrder%>);'> Cancel Transaction  
<INPUT type='radio' name='options' value='5' onclick='setOptions(this.form.opt,
this.form.options, <%=zOrder%>);'> Display
```

Figure 11-89 shows the Web setting where the HTML was added.

The HTML defines a set of radio buttons and groups them under the name "options". This means that only one of these can be active at any time. When each button is clicked, the JavaScript function `setOptions` is called. The `setOptions` function is described in "JavaScript functions" on page 401.

The option entry field in the subfile record was hidden to preserve the HTML defining the field within the JSP generated by WebFacing. If the check box is created by changing the User defined HTML Web setting for the option field, then the option field no longer exists within the JSP and the associated value cannot be returned to the iSeries. If the check box is created as part of the Insert HTML Web setting, then the input field is still displayed on the Web page and the check box. To create the check box for the option field, the Web setting for the customer number field is changed.

The Insert HTML Web setting for the account number is changed as shown in Example 11-8. The HTML was added in the Before entry box.

Example 11-8  HTML to create check box for options

```html
<INPUT type='checkbox' name='opt' onclick="if (this.form.opt[<%=rrn - 1%>].checked) {
setOptions(this.form.opt, this.form.options, <%=zOrder%>);}
else {
clearFieldValue('document.SCREEN._<%=zOrder%>CSFLC$_z_zOPT$_<%=rrn%>'); }">
```
Figure 11-90 shows the Web setting to include the HTML for the check box.

The check box name "opt" is created that calls the JavaScript function `setOptions` when the box is selected and assigns the $$OPT field with the value of the selected radio button. If the box is not selected, then it calls the JavaScript function `clearFieldValue` to clear the value stored in the $$OPT field. The `setOptions` and `clearFieldValue` functions are described in "JavaScript functions" on page 401.

In the HTML tags, the `rrn` field is used to define which record in the subfile is being accessed. The subfile in this instance is numbered 1 through 6. `this.form.opt` is a numbered array that contains the values of the check boxes. This array is based at zero and goes through 5. The `zOrder` field is used to identify the layer in which the record is being shown on the display. The variable depends on how the display is constructed, which justifies the need to pass the value to the function.

## JavaScript functions

JavaScript functions were created to set a value of a field, clear a field, and to set the option entry field based on the radio button and check boxes that are selected. These functions were created in `project name\Web Content\webfacing\ClientScript\usr\myUtil.js`. Example 11-9 shows the contents of `myUtil.js`, that can be added to the `ClientScript\usr` folder.

### Example 11-9  JavaScript functions in myUtil.js

```javascript
/***************************************************************************
**  setFieldValue : Sets the value of a field to the value passed in and sets**
**  modified data tag on the field to true.                             **
**  This is based on the setFieldValueAndSubmit contained in**
**  webface5122.js, however it does not perform the submit.            **
**************************************************************************/
function setFieldValue(id,newValue)
{
    id2 = eval(id);
    id2.value = newValue;
    c_i_e_i_w_5122_wfInfoDB[formId][id2.id].mdt= true;
}

/***************************************************************************
**  clearFieldValue : Sets the value of a field to an empty string and sets**
**  modified data tag on the field to false.                          **
**************************************************************************/
function clearFieldValue(id)
{
    id2 = eval(id);
    id2.value = "";
```
The `setOptions` function reads through the check box array and determines whether a box is selected. If it is, then it determines which of the radio buttons are selected. If a radio button is selected, then the option field for the given subfile record is set to the value that is associated with the button. If no check box or radio button is selected, then there is no need to update the option field in the subfile. The order field is used to identify the layer that the record is being presented on the display.

When the page is submitted, by using the Enter key, the option field for each subfile record reflects the value of the selected radio button if the check box for the record is selected.

**Note:** The prefix `c_i_e_i_w_5122` for the JavaScript functions was introduced for Portal support and is the abbreviation for `com.ibm.etools.iseries.webfacing`, followed by the version number.
Accessing the button functions
The webface5xxx.js file contains three functions that can be overridden to provide access to the button functions for the command keys. The functions are:

- **bo_usr()**: This is used for button over or mouse over a button.
- **bu_usr()**: This is used for button up, for example, initializing a button or turning a button off.
- **bd_usr()**: This is used for button down, for example when the button is clicked.

Creating the JavaScript for these functions allows you to perform additional functions when the buttons are selected. It does not replace the functionality that exists when a button is clicked.

The webface5xxx.js file contains calls to these functions. Example 11-10 shows the calls to these functions.

Example 11-10  Webface.js file showing the calls to button functions

```javascript
function bo_ibm(myObj)
{
    var rv = false;
    try
    {
        rv = bo_usr(myObj);
    }
    catch(any_exp){ rv = false;}
    if (false == rv) {myObj.className='buttonover';}
}

function bu_ibm(myObj )
{
    var rv = false;
    try
    {
        rv =bu_usr(myObj);
    }
    catch(any_exp){ rv = false;)
    if (false == rv) {myObj.className='buttonup';}
}

function bd_ibm(myObj)
{
    var rv = false;
    try
    {
        rv = bd_usr(myObj);
    }
    catch(any_exp){ rv = false;)
    if (false == rv) {myObj.className='buttondown';}
}
```

11.10 Report Service: Viewing spooled files
A new function introduced with WDSc V5 is the capability to define an action button that allows the browser to display all spooled files in a WebFacing application. You define the reporting service through the use of command key override. You override one of the function
keys, such as F12, and instead, use the key to define your report service. These command keys override the existing function keys and can connect to a Web application URI or an external URL.

Report service is only supported on WebSphere Development Studio Client Advanced Edition for iSeries and your WebFacing project needed to be Struts enabled if it was created by a version of WDSc prior to 5.1.2. To display the content of individual spool (as shown in Figure 11-94 on page 407), you must have Option 8 of 5722-SS1, “OS/400 - AFP™ Compatibility Fonts”, installed. Also, DBCS users need to install font collection 5769-FN1 and set the QIGCCDEFT system value to its appropriate font.

Attention: With the first release of WDSc V5, report service only worked in the Internet Explorer browser (unlike the one that is part of the workbench IDE).

For more information about command keys and related WebFacing tasks, see 11.3.1, “Command key recognition” on page 329, and “Customizing the WebFacing interface with CODE Designer” on page 361.

Some of your command keys may already be associated with working spooled files in your DDS source. It is up to you to keep track of the functions allocated to command keys. Since system panels, such as the Work with Spooled Files (WRKSPLF), could not be converted by the IBM WebFacing Tool prior to WDSc AE 5.1.2, you can replace any command key that invokes the WRKSPLF command with a report service action key.

However, you can only display the spooled files with the report service. You can neither perform the same functions as those provided with WRKSPLF command nor can you show any OS/400 panels using Report Service. This is different from the System Screen Support introduced with WDSc AE 5.1.2 and described in 11.11, “Showing system screens with the WebFacing Tool” on page 408 later in this chapter.

To define a report service command key to work with your spooled files, follow these steps:
1. In the WebFacing Projects view of the WebFacing perspective, right-click your project and select Properties.
2. Under Runtime, select Command key actions. Click Add.
3. In the Add command key action window (Figure 11-91), follow these steps:
   a. In the Command Key field, specify or select a function key (F1 to F24).
   b. In the Action name field, select ReportService.
c. Leave the URI or URL field as *RetrieveListOfReports.do*.
d. Leave the Button label field as *List Reports* or override it with a label of your choice.
e. In the Target frame field, select *NEW* from the drop-down list if you want a new window to open when the button is clicked.
f. Leave the *Enable only when the command key is active* deselected, so that your report service button appears at all times.
g. Click *OK*.

**Important:** If you tested your WebFacing in the workbench test environment before implementing this change, make sure to close the browser window and end the server.

4. Test your WebFacing application in the test environment. You see the new button as shown in Figure 11-92.

![Order Entry Sample Application](image)

*Figure 11-92  Button for Report Service*
5. When you click the new button, a list of all spooled files for your user profile is displayed similar to the example in Figure 11-93.
6. You can click each spooled file to display its contents as a GIF file as shown in Figure 11-94.

Tip: If your system does not have Option 8 of 5722-SS1, “OS/400 - AFP Compatibility Fonts” installed, you cannot see the contents of the spool, but only a place holder as shown in Figure 11-95.
11.11 Showing system screens with the WebFacing Tool

Advanced

Until version 5.1.2 of WebSphere Development Studio Client, the IBM WebFacing Tool did not provide support for system screens. For example, if an application displayed a Work with Spool File panel as part of the application's screen flow, the user would get a runtime error and the application had to be re-started.

Version 5.1.2 of the Development Studio Client Advanced Edition introduces system screen support in the IBM WebFacing Tool. Specifically this support is for system screens that were created with User Interface Manager (UIM). This allows the application to display many system screens such as Work with All Spooled Files (WRKSPLF) as part of a converted WebFacing application.

**Note:** WebFacing System Screen Support allows you to show many different kinds of OS/400 panels including, but not limited to the WRKSPLF display. This is different from the Report Service described in 11.10, “Report Service: Viewing spooled files” on page 403.

This new feature is intended to support applications that include some system screens as part of the workflow of the application. It is not intended to support all screens that exist on the system. For example, the Application Development Tool Set (ADTS) panels were mainly created using DDS not UIM. Accessing these panels still causes a run-time error and the user needs to restart the WebFacing application to recover from these errors.

The Advanced Edition of the IBM WebFacing Tool provides support for any UIM panel regardless of whether this is a system panel or a user defined panel. It does not, however, provide support for DDS-based screens used for some panels of the OS/400 system or iSeries products.

To provide support for some of these DDS-based panels, the WebFacing team has pre-converted some DDS-based panels that have been identified as frequently used panels in a business application context. When displaying these pre-converted screens, no run-time error occurs since a corresponding Java Server Page (JSP) exists for record formats used in these panels.

11.11.1 Before you start using WebFacing system screen support

When using system screens in a WebFacing project, insure that the Display sign-on information (DSPSGNINF) parameter in the user profile is set to *YES or the system value QDSPSGNINF is ‘1’ and DSPSGNINF equals *SYSVAL. If QDSPSGNINF is set to ‘0’ or DSPSGNINF to ‘NO, attempting to use a system screen will result in an error message MCH0601 ‘Space offset is outside current limits for object’.

The error message MCH0601 is not directly visible to the browser user. Instead the application ends abnormally after the first system screen has been displayed and the panel shown in Figure 11-96 appears.
This a known restriction which may or may not be removed in the future.

11.11.2 Example: Work with All Spooled Files

To understand the system screen support in the IBM WebFacing Tool better, take a look at this example. This WebFacing sample application has the capability to launch a Work with All Spooled Files panel to allow the end user to work with orders that are ready to be printed. The application invokes the WRKSPLF command with the QCMDEXC interface. Here is the browser page the end user would see as shown in Figure 11-97.
Figure 11-97  Command Key button for Work with Spool Files system screen

Figure 11-98 shows the Work with All Spooled Files screen after clicking the button on the previous screen.
If options are used for selecting rows on this page, they cause a new panel to be shown as long as the panel is a UIM-based panel. The same is true for commands entered on the command line, as long as panels are displayed that are created with UIM, they are displayed in the browser.

### 11.11.3 Unsupported system panels

The WebFacing system screen support basically converts all UIM defined panels on the fly to HTML. However, not all displays provided by OS/400 or i5/OS have been created with UIM.

If the application tries to display a panel that is not UIM-based and has not been pre-converted, the user receives an error that a record data bean cannot be found, as shown in Figure 11-99.

---

**Figure 11-98  Work with All Spooled Files screen using WebFacing System Screen Support in WDS 5.1.2**

**Note:** If the corresponding 5250 panel provides a command line, then this Web page will provide a command line as shown in the sample page for Work Spool File.

This command line is available to invoke CL commands. In other words, the full capability of the original system screen is preserved. The ability to execute commands and access system objects depends on the user profile used invoke the WebFacing application.
This error invalidates the application's session and the user will need to re-start the application.

**Recovering from an error caused by an unsupported system panel**

If the user uses the Back button in the browser and then tries to continue in the application, the application will probably not work in the current browser session. At this point, we suggest the user closes the browser and re-starts the application in the application server. Then, in a new browser session, the user can invoke the application again and continue working. Bear in mind that the user should not access a non-supported panel in the WebFacing system screen environment.

**Important:** At this point, the user should not try to recover using the page back button.

Note: The instructions on how to recover from this error (to return to a state that allows the user to restart the application) are included in the message. Using the link in the error message provided returns the user back to the page that had been used to invoke this application.
11.11.4 Examples of supported panels

A list of “Work with”-commands supported by the WebFacing system screen support include:

- Work with Active Jobs (WRKACTJOB)
- Work with Job (WRKJOB)
- Work with All Job Queues (WRKJOBQ)
- Specify Journal Name (WRKJRN)
- Work with Libraries (WRKLIB)
- Work with Menus (WRKMNU)
- Display Messages (WRKMSG)
- Work with Message Descriptions (WRKMSGD)
- Work with Message Queues (WRKMSGQ)
- Work with Objects (WRKOBJ)
- Work with Object Locks (WRKOBJLCK)
- Work with All Output Queues (WRKOUTQ)
- Work with OUTQ Description (WRKOUTQD)
- Work with Printing Status (WRKPRTSTS)
- Work with Submitted Jobs (WRKSBMJOB)
- Work with System Status (WRKSYSSTS)
- Work with User Jobs (WRKUSRJOB)
- Work with User Profiles (WRKUSRPRF)
- Work with All Printers (WRKWTR)

These commands and most of their options should work in a WebFacing runtime environment when using system screen support.

Another example of supported system screens is the licensed program Query for iSeries (5722-QU1).

Currently not supported is the Start SQL Interactive Session (STRSQL) command provided with 5722-ST1, Data File Utility (DFU) and the Program Development Manager (PDM).

However, you can find equivalent replacements or even more functions for all three products in the WDSc workbench.

11.11.5 PTFs required

Make sure the latest PTFs are installed according to the IBM Web site:

http://www.ibm.com/software/awdtools/wdt400/sysreq/

If you experience problems with command prompt panels when using F10 to display additional parameters, you are probably missing an OS/400 system PTF.

11.11.6 National language support

The UIM-based system panels have national language support and display text in the correct language according to the job in which the WebFacing application is running. For pre-converted DDS system panels, the text is displayed in English unless the text on these panels was stored in message files by the panel developers.
Advanced techniques for using the IBM WebFacing Tool

This chapter explains some of the more advanced topics related to the IBM WebFacing Tool. Some functions described in this chapter are only supported on WebSphere Development Studio Client Advanced Edition for iSeries (WDSc AE). You must purchase this feature separately from the WDSc Standard edition. Even if you received WDSc as part of 5722-WDS free of charge because of the Software Subscription, you must pay a fee to acquire WebSphere Development Studio Client Advanced Edition for iSeries.

Changes in the second edition of this redbook
This chapter was originally written for WDSc 5.0. Now it has been updated to reflect the changes and enhancements added with WDSc 5.1 in 2003 and WDSc 5.1.2 in 2004, as well as several fixpacks between the major releases. Technical differences from the previous edition of this book are indicated by a change bar at the left of the text. Minor changes such as grammar corrections and slightly different phrasing are not marked.

The base for writing and testing the examples in this chapter is WebSphere Development Studio Client Advanced Edition for iSeries Version 5.1.2.4 (called WDSc AE in the following), that is WDSc AE 5.1.2 with Fixpack 4 applied.

Note, this redbook provides an introduction to the IBM WebFacing Tool. More elaborated topics are covered in Mastering the IBM WebFacing Tool, SG24-6331.
12.1 Current DDS keyword support

Not all data description specification (DDS) keywords for OS/400 display files can be converted by the IBM WebFacing Tool. However, since the first version of the IBM WebFacing Tool, several new keywords are supported with each service pack. A list of supported keywords is available in the online help of the product. You can find this list by performing the following steps:

1. In the workbench menu, click Help → Help Contents. A new window opens.
2. Click Reference in the left pane.
3. Expand the tree under User interface reference.
4. Expand the tree under iSeries WebFacing applications.
5. Then click DDS support tables. You see a window like the example shown in Figure 12-1.

You can use the DDS keyword survey tool to help evaluate the level of keyword support that the IBM WebFacing Tool will provide for your application. To download the DDS keyword survey tool, go to:

http://ibm.com/software/ad/wds400/
12.2 Using the WebFacing Environment API

Although there is no need to change your existing application on the iSeries server 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 application program interface (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 one if the application is running under WebFacing. It returns zero if it is running under 5250 emulation.

The following examples (from the WebFacing help contents “Using the WebFacing environment API”) show how to use this API. In the RPGLE sample (Figure 12-2), the external procedure QqfEnvironment is defined with the D Spec QQFENV. The QQFENV D Spec is given the same name as the service program and is defined as an integer since the procedure returns 0 or 1. A D Spec rc is also defined to hold the value 0 or 1 when Eval rc = QQFENV is performed. The RPG program then uses the value of rc to conditionally determine the behavior of the program and what is displayed on the DDS display.

```
RPGLE Sample
........1....2....3....4....5....6....7....8
FCHKENVFM CF E Workstn
  *
DQQFENV PR 10I 0 Exptrc('QqfEnvironment')
  *
DrC S 10I 0
  *
C Eval rc = QQFENV
C Eval FLD001 = rc
  *
C Dow NOT *IN03
  *
C If rc = 1
C Eval *IN01 = *on
  *
C Else
C Eval *IN01 = *off
C EndIf
  *
C Exfmt FMT01
C EndDo
  *
C Eval *INLR = *on
```

Figure 12-2 RPGLE sample

In the DDS sample in Figure 12-3, if the value for rc in the RPG (Figure 12-2) module is not 1, you see the message “Application is not running in the WebFacing environment”. If the value for rc is 1, you see the message “Application is running in the WebFacing environment”.

Chapter 12. Advanced techniques for using the IBM WebFacing Tool
When you create a program to use this API, follow these guidelines:

- Use the Create RPG Module (CRTRPGMOD) command to create a module with your RPG code that is calling the API. You must create an RPG module because it is using a procedure that is not in the program.

- When you create your program (using the Create Program (CRTPGM) command), use the BNDSRVPGM keyword to bind your RPG module with the QQFENV service program in QSYS.

```
DDS Sample
....+A*..1.....+2.....+3.....+4.....+5.....+6....+7.....+8
A*%%TS SD 20010924 150104 USERID REL-V4R4M0 5769-PW1
A*%%EC
A R FMT01
A*%%TS SD 20010924 150104 USERID REL-V4R4M0 5769-PW1
A CA03(03)
A 1 24'Testing WebFacing Environment API'
A DSPATR(HI)
A 10 'F3=Exit'
A COLOR(BLU)
A N01 5 13'Application is not running in the -
A WebFacing environment'
A 01 6 15'Application is running in the Webf-
A acing environment'
A COLOR(RED)
A FLD001 4S 00 7 39
A 7 22'QqfEnvironment: '
A
A*%%AGP SCREEN1 01
```

**Figure 12-3 DDS sample**

### 12.3 Programmatic invocation of WebFacing applications

WebFacing applications can be invoked programmatically from other Web applications. This provides a way to integrate WebFacing generated user interfaces with existing Web applications.

WebFacing applications are launched from Uniform Resource Locators (URLs). Typically, a URL is represented as a link that a user clicks to start the application. During WebFacing conversion, URLs are written to the index.jsp file. After the application is deployed, users click these links to start the application.

However, WebFacing URLs can also be constructed dynamically by other programs such as controller servlets. These URLs can be sent back to users as links to click on a Web page. More importantly, they can be used in the `forward()` and `sendRedirect()` methods in other Web applications.

Using dynamically determined WebFacing URLs with `forward()` and `sendRedirect()` methods means that a program like a controller servlet can perform such actions as determining the host to use, the WebFacing program to launch, what CL command to use for the WebFacing program, or other actions. After these actions are completed, the servlet can provide the appropriate WebFacing application, with its parameters already set, directly to a user.
With control of how WebFacing is invoked, it is also possible to use alternative methods of authentication. All user authentication can now be performed in a custom servlet before WebFacing is called. The authentication mechanism used must provide the WebFacing application with iSeries user credentials so that it can access OS/400 resources.

We take you through a simple example to determine the CL command to use to launch a program. The URL constructed by a controller servlet looks like this example:

`WFLogon?clcmd=call%20ordentr`

In this example, ordentr is the name of a program to launch. The value ordentr can be determined by a servlet and assigned to a variable such as iseriesProgram. Your servlet can construct the URL string using the value determined for iseriesProgram and assign it to a variable newURL using a line like this:

```java
newURL = "WFLogon?clcmd=call " + iseriesProgram;
```

newURL could then be used as the forward or redirect URL for your `forward()` or `sendRedirect()` methods.

In this example, the complete URL used by the browser, if sent as a redirect, looks like this example:

`http://<hostname>:<port>/<application>/WFLogon?clcmd=call%20ordentr`

The example shows the full URL beginning with `http://<hostname>:<port>/<application>/`. The value for newURL is the string after this:

`WFLogon?clcmd=call%20ordentr`

In an example like this, the first part of the URL, `http://<hostname>:<port>/<application>/`, represents the host, port, and context root for the application. If your controller servlet is in the same context root, it is not always required for the servlet to determine the entire URL. If necessary, you could code the servlet to construct a string for the fully qualified URL.

**Space character %20:** The characters `%20` in the URL represent a space character as encoded when sent to a browser. Space characters generally cannot be used explicitly in URLs. In the example where the URL string is being constructed and assigned to newURL, the space is present in the part of the string after `clcmd=call`. The reason for the space in the string is that the example represents a CL command call ordentr. In the URL string that is being constructed, it is not necessary to add `%20` directly. The server adds this encoding if necessary.

URL parameters that can be determined dynamically include:

- **clcmd:** CL command to launch the program.
- **host:** Host name where the original 5250 application is located.
- **port:** Port on which the WebFacing server is running.
- **userid:** User ID used to log onto the application.

**Note:** If a `forward()` method is used in your controller servlet, URL parameters are only sent within the application server tier (middle tier). Using `sendRedirect()` instead exposes URL parameters to the browser. `sendRedirect()` is less secure because such information as user IDs and passwords can be revealed in a browser’s location field or if a user views the properties for the page they are using.

- **password:** Password used to log onto the application. See note box above for user ID.
inv: WebFacing invocation file to use for launching the application.

Example URLs include:

- WFLogon?clcmd=call%20ordentr
  The CL command call ordentr is passed by the URL. Host, port, userid, and password in the web.xml file (WFapp.properties file for WDSv V4) are used. Prompt the user to logon only when userid or password are missing or prompt is specified. Print an error message if userid or password are incorrect.

- WFLogon?inv=cl172537
  Host, port, userid, password, and CL command are retrieved from the cl172537.invocation and web.xml files. Prompt only when userid or password is missing or prompt is specified. Print an error message if userid or password are incorrect.

- WFLogon?clcmd=call%20ordentr&inv=cl172537
  The CL command call ordentr is passed by the URL. Host, port, userid and password are retrieved from the cl172537.invocation and web.xml files. Print an error message if userid or password are missing or prompt is specified. Print an error message if userid or password are incorrect. Multiple parameters are separated by &.

- WFLogon?inv=cl172537&host=ISERIES1&port=4004&userid=WEBFACING&password=WEBFACING
  A CL command is retrieved from cl172537.invocation. Host, port, userid, and password are passed by the URL. Multiple parameters are separated by &.

- WFLogon?clcmd=call%20ordentr&host=ISERIES1&port=4004&userid=WEBFACING&password=WEBFACING
  The CL command call ordentr is passed by the URL. Host, port, userid, and password are passed by the URL. Multiple parameters are separated by &.

Here is a fully qualified example:

http://<hostname>:<port>/<application>/WFLogon?clcmd=call%20ordentr&host=ISERIES1&port=4004&userid=WEBFACING&password=WEBFACING

In this example, the strings <hostname> and <port> refer to the hostname and port for the application server where the WebFacing application is deployed. <application> is the context root for the deployed Web application. The example shows the following values passed by the URL:

- The CL command is call ordentr.
- The host where the 5250 application is located is ISERIES1.
- The port on which the WebFacing server is running is 4004.
- The userid is WEBFACING.
- The password is WEBFACING.

Multiple parameters are separated by &.

Servlet methods for calling a WebFacing application programatically
There are two servlet methods for calling a WebFacing application programatically. They are:

- forward(): This method is in the javax.servlet.RequestDispatcher class.
- sendRedirect(): This method is in the javax.servlet.http.HttpServletResponse class.

The most important differences between these two methods are:

- javax.servlet.RequestDispatcher's forward()
Chapter 12. Advanced techniques for using the IBM WebFacing Tool

12.4 Working with JSPs

A JavaServer Page (JSP) is a Hypertext Markup Language (HTML) page that contains additional code to execute application logic for the generation of dynamic content. Various technologies may be involved in applying this application logic, such as JavaBeans, Java Database Connectivity (JDBC) objects, Enterprise JavaBeans (EJB), and Remote Method Invocation (RMI) objects.

For example, you can use a JSP if you want to display static content, such as images and text, and dynamic information, such as variable database content, in your browser. Your JSP page may contain HTML code that displays the static text and graphics. It may also contain a method call to a JDBC object that accesses a database.

JSPs are an extension of the Servlet API. JSP pages have all of the benefits of servlets, including access to Java APIs.

Generally when using JSPs, Web authors can edit the fixed template portions of pages without affecting the application logic. Developers can similarly make logic changes at the component level without changing the individual fixed template portions of the pages that use the logic.

Use the online help, Web resources, and other books to master skills for creating and working with JSP files. The available resources include:

- **Online help:**
  - In WDS window, select from the menu Help → Help Contents.
  - Expand Developing.
  - Expand Web applications
  - Expand iSeries Web applications
  - Expand Creating dynamic Web project for iSeries Web development.

- **Access WebSphere Developer Domain at:**

- **Search for documentation or tutorials. You can find “Building dynamic Web sites” at:**

12.5 Performance considerations

The following sections discuss the performance of the conversion process and considerations when creating a project. It also describes the jobs that run on an iSeries when the conversion is performed and when the converted program is called.
For the purpose of the project and conversion sections of this chapter, the application that is
discussed contains approximately 3,800 display files with 25,000 record formats. The display
files range in size from 200 lines to 15,000 lines of DDS.

For 12.5.2, “Time to create the project and convert the display files” on page 423, the Order
Entry application was used for the performance tests.

We start with a discussion about what you can do to estimate the appropriate size of an
iSeries server to support running a Web application built with the IBM WebFacing Tool.

12.5.1 Using IBM @server Workload Estimator to plan a server configuration

The IBM @server™ Workload Estimator (previously called IBM Workload Estimator for
iSeries) has been used for several years. It aids in providing sizing recommendations for new
or existing iSeries servers to support several types of application workloads. These include
Lotus Domino, Java, or WebSphere. Since January 2003, the Workload Estimator also
supports the sizing of WebFacing applications.

Note: WebFacing Version 5.1.2 has been enhanced to support the dynamic transformation
of i5/OS™ system screens, and also to run WebFacing in WebSphere Portal Server. Sizing
workloads that will include these functions is not supported in Workload Estimator.

The estimator is a Web application that you can find on the Internet at:


Workload Estimator recommends the model, processor, interactive feature, memory, and
direct access storage device (DASD) necessary for a mixed set of workloads. To use the
Estimator, you select one or more workloads and answer a few questions about each
workload. Based on your answers, the Workload Estimator generates a recommendation and
shows the predicted central processing unit (CPU) utilization of the recommended system in
graphical format. You can then print the final results. Advanced users have the option to
provide more specific information for particular workloads, thereby providing a more accurate
estimate.

The WebFacing workload of the Workload Estimator forecasts resources consumed by an
iSeries in a WebFacing environment. You must use another tool to separately size any other
hardware required for clients and network resources.

The WebFacing workload specifically bases its projections from iSeries performance
measurements in IBM development laboratories using IBM WebFacing Tool and WebSphere
Application Server. A variety of performance workloads were measured to characterize the
resource consumption for this support. This workload is not intended to be used to
characterize other Web-enablement products or 5250 datastream transformation tools. The
workload that was measured to characterize this WebFacing workload is new. Therefore, do
not compare the results of this sizing with the results from any prior sizings.

There are three methods to size an iSeries with the WebFacing workload within the Workload
Estimator. Each method is differentiated by the point of entry to the Estimator. For any of
these methods, use the help text to assist you to make the appropriate selections.

Select the WebFacing workload on the workload selection panel. On the workload definition
panel, describe the transaction rate and the 5250 screens using the WebFacing support.
From there, you can choose to build a sizing with just the WebFacing processing (no iSeries
server application), or you can also include the 5250 iSeries server application in the sizing.
When you include the 5250 application in the sizing, you are also directed to the Existing workload definition panel to describe the existing 5250 application.

Select the Existing workload on the workload selection panel. On the workload definition panel, describe the existing 5250 iSeries server application that will use the WebFacing support. Make sure that you select yes for the question asking if this application will use WebFacing support. You are directed to the WebFacing workload definition panel to describe the transaction rate and the 5250 screens using the WebFacing support. Note that the result for this method gives the same result as the previous method. The order of the inputs was just in reverse order.

Use PM server iSeries (previously called Performance Manager/400 (PM/400)), which is not directly part of the Workload Estimator, to characterize your existing system with a 5250 iSeries server application. Select the option to use the Workload Estimator. At this point, you see panels within the Workload Estimator. Select the option to use WebFacing support. This directs you to the WebFacing workload panel to describe the transaction rate and the 5250 displays using the WebFacing support.

The WebFacing workload recommends a system with at least 300 Commercial Processing Workloads (CPWs) and 0.5 GB of memory. Regardless of the number of transactions you specify, the Workload Estimator does not recommend fewer CPU or memory resources than those minimum values. Workload Estimator also considers minimum configuration rules for the number of disk arms.

The calculations for CPW requirements are based on a variety of measurements. The memory and disk requirements are estimated. The actual memory and disk requirements can vary greatly depending on your specific application environment. Workload Estimator projects performance based on a specific set of measurements with specific benchmarks. Actual customer performance may vary significantly based on configuration and application design.

For additional information about performance, see the iSeries Performance Capabilities Reference, SC41-0607. See also section 2.1.2 of the redbook Mastering the IBM WebFacing Tool, SG24-6331 for examples.

### 12.5.2 Time to create the project and convert the display files

You can create a project with a single display file and a project with multiple display files. You can also see the time that it took to create the projects and convert the associated display files.

**Creating a project with a single display file**

The creation of a project that contains a single display file, with eight record formats, took less than one minute. The project was completed without converting the display file. Converting the display file and saving the object took approximately one minute.

**Note:** These tests were conducted in a lab environment with a local area network (LAN) connection between the PC and iSeries server. If you use a dial-up connection, for example, you should expect this time to increase.

The conversion time was measured with WDSc V4. However, the general approach to working with single or multiple files in the IBM WebFacing Tool is the same.

The timings in this section are given for reference only.
Creating a project with multiple display files

When we created a project with multiple display files, we added 3,800 display files and completed the project without converting the display files. The creation of the project took approximately 12 minutes. Subsequent conversion of all the display files within the project took approximately 16 hours.

When only one display file was converted, the conversion of the display file was performed in approximately one minute. Saving this converted file took approximately 10 minutes.

Note: As an indication of the amount of time for conversion, we found that the IBM WebFacing Tool converted up to 200 display files per hour. In some cases, this was as low as 50 display files per hour. This variable number depends on the size of the display files being converted, the line speeds, the iSeries load, and so on.

Workbench settings

When a DDS member is converted, the workbench verifies all other projects to determine whether they require updating. This can be a lengthy process if a large number of DDS members is defined within a project. You can turn off this option so that the workbench does not perform an automatic build whenever a resource is modified. Follow these steps:

1. Select Windows → Preferences from the tool bar.
2. Deselect the Perform build automatically on resource modification check box as shown in Figure 12-4. This means that manual builds are required on projects that depend on the resource that was just modified.
Considerations for creating a project

When you convert an application for the first time, consider the number of projects to create. If an application has only one entry point, then you are required to create one project. If individual programs are to be made accessible as separate Web pages, then you must create them as separate projects.

Consider the following example. A menu in an application has five options. There are two ways to reface this menu. The first way is to create a project that contains the menu DDS and the DDS for all the options within the menu. This means that you can access the completed application through the menu by using the browser.

The second method is to create a project for each menu option. In this case, you create five projects. When you create each project as an application within the application server, you can access them as individual Web pages. There is no need to reface the menu DDS,

Important: We recommend that you do not turn off this setting. If you decide to turn off this parameter, you must manually build your project before testing or exporting the application to production:

1. In the Navigator view, right-click your project.
2. Select Rebuild Project.
because it is not used as the entry point for the application. You can then access the programs by using the individual URLs that are created. You can embed these URLs into other Web applications to access the iSeries application.

Exporting the project
When a project is ready to be exported, a Web Archive (WAR) or Enterprise Application Resource (EAR) file is typically created directly onto the integrated file system (IFS) through a mapped network drive. In a situation where the project contains one display file, it took approximately three minutes to create the WAR file directly on the IFS. Larger projects take considerably longer. When we tried to create a WAR file for 3,800 display files, it took approximately 10 hours to create the WAR file directly to the IFS.

Tip: When you export the project, specify a folder on a local drive as the destination for the WAR file. After the WAR file is created, copy it to the IFS. By using this method, the WAR file for the project, with one display file, is created and copied in under 30 seconds. For the application with 3,800 display files, it reduces the time to approximately two hours.

12.5.3 Run-time jobs
After you create and export the project as an application within WebSphere Application Server, you can access it from a browser. There are several varying factors that can affect the performance of the run-time environment to allow indicative times to be discussed. These factors include:

- Network configuration
- Line speed
- Machine configuration and size

This section describes the various jobs that run on the iSeries server when a WebFacing application is invoked. Figure 12-5 shows the relationship between the run-time jobs.

Figure 12-5  WebFacing server jobs

QQFWFSVR: WebFacing server
QQFWFSVR is the WebFacing server. It receives requests to invoke a WebFacing program. This is a batch immediate job that runs in the batch subsystem QSYSWRK. You invoke the job by entering the following command:

```
STRTCPSVR *WEBFACING
```
This job receives the request from WebSphere Application Server when a program call is
initiated. Then it sends a request to the virtual terminal server to create a new instance of a
virtual terminal to run the requested program.

This is a multi threaded job to which each active Web session is assigned a thread in this job.
There is also a garbage collection job that runs as a thread. The garbage collection works on
cleaning up unused virtual terminal servers. The garbage collection terminates unused virtual
terminal servers but leaves three active servers even if they are not currently used. This is a
fixed number that you cannot change.

**QQFVTSVR: Virtual terminal server**
The virtual terminal server creates instances of virtual terminals to allow WebFacing
programs to run interactively without requiring a physical device to be attached to the session.
This is a batch immediate job that runs in the batch subsystem QSYSWRK. You invoke it by
entering the following command

```
STRTCPSVR *WEBFACING
```

This job creates a virtual terminal device to run the program defined in the Web application
that is being processed.

When the WebFacing servers are started, only one instance of the virtual terminal server is
started. A virtual terminal server job works with a maximum of 20 interactive jobs. When the
number of active Web sessions nears 20 sessions, a new virtual terminal server is spawned
to handle the next set of 20 sessions. This process continues as Web sessions are activated.

**Note:** With the latest WebFacing server PTFs for V5R2 SI12537 or PTF SI10731 for V5R1
(refer to the Informational APAR SE13580), the QQFVTSVR starts an extra job for the
performance improvements on the systems with many concurrent users. Before this PTF,
only one virtual terminal server ran to handle the requests. When the number of the virtual
terminal server reached 20, a new virtual terminal server was spawned.

Now one virtual terminal server runs when the WebFacing server starts and another virtual
terminal server when there is a request from the application. Before reaching 20 jobs, two
virtual terminal servers will run. This was changed to allow the virtual terminal servers to
handle all requests in a busy system environment.

**QQFINVOKER: Interactive job**
This job runs in the interactive subsystem QINTER. It uses the virtual terminal device created
by virtual terminal server. The QQFINVOKER job performs the function of an interactive
green-screen session. This job does not create a 5250 data stream, but instead creates the
data stream for the Web server that originates the request.

If you close the browser while the WebFacing application is still running, then the QQFxxxxx
interactive jobs are left active on the system. However if you terminate the application, the
invoker jobs go away. By default, the QQFxxxxx interactive job has a time-out value of 30
minutes. To change the session time-out setting on WebSphere Application Server, see:

```
http://publib.boulder.ibm.com/was400/51/english/info/rzai5/51/admin/help/uprs_rsession_manager.html
```

### 12.5.4 Tuning the performance of a WebFacing application

System settings, WebSphere settings, and the network connection all play a part in the
performance of a WebFacing application. The IBM WebFacing Tool itself play an important
role in how well your application performs. The tool has several new features that improve the
performance of a WebFacing application compared to the previous version of Development Studio Client.

For up-to-date information about improving your WebFacing application performance, read the white paper IBM WebFacing Tool Performance Update using WebSphere Application Server V5.0 at:


12.6 DDS Object Mappings

DDS Object Mappings provide WebFacing with a correlation between the display file that the WebFacing application uses and the display file source member that was converted by the IBM WebFacing Tool.

This mapping is described by rules of the form:

display-file = display-file-source-member

The rules can contain actual object names such as:

MYLIB/MYDSPF0BJ = /MYSRCLIB/MYSRCPF/MYDSPFMBR

Or they can contain replacement variables for the library (&LIB) and display file name (&DSPF).

If a replacement variable is used in the display-file part of the mapping, it matches any library or display file name. If a replacement variable is used in the display-file-source-member part of the mapping, it is replaced with the library or name of the display file being resolved.

See the following examples:

- &LIB/MYDSPF = /&LIB/QDDSSRC/MYDSPF
  
  This rule matches a write to a display file with the name MYDSPF, regardless in which library it is. If the application writes to display file MYDSPF in library PRODLIB, then this rule looks for the conversion output of display file source member MYDSPF from the PRODLIB/QDDSSRC file.

- &LIB/MYDSPF = /SRCLIB/QDDSSRC/MYDSPF
  
  With this rule, a write to display file MYDSPF in any library resolves to MYDSPF in SRCLIB/QDDSSRC, regardless of the library in which the display file itself was located. Using this form of the rule gives you the flexibility to move the compiled objects anywhere on the host and resolving them to the SRCLIB library source.

- PRODLIB/&DSPF = /PRODLIB/QDDSSRC/&DSPF
  
  With this rule, a write to any display file in the PRODLIB library maps to the corresponding source member in PRODLIB/QDDSSRC. If a WebFacing application writes to display file MYDSPF2 in library PRODLIB, this rule looks for the conversion output of display file source member MYDSPF2 from PRODLIB/QDDSSRC. This rule is useful if you want your vendor supplied display files in one library to be overridden by customized display files in another library. In this case, you want two such rules, one for the vendor library and one for the customized library.

- &LIB/&DSPF = /&LIB/QDDSSRC/&DSPF
  
  This rule maps any display file in any library to the source member with the same name as the display file in the QDDSSRC source file and in the same library as the display file. This is The IBM WebFacing Tools’s default rule when no other rule can apply.
To determine which rule to use, WebFacing examines each rule for which the display-file part of the name matches, for the display file being written to or read from. It then looks for conversion output that matches the display-file-source-member part of the rule. If no conversion output is found, it goes on to the next rule. The order in which the rules are searched is not necessarily the order in which they appear. The rules are searched based on their left-hand side values in the order shown here:

- Look for an exact match of library and DSPF object (that is MYLIB/MYDSPF).
- Look for a wildcard library and specific DSPF object (that is &LIB/MYDSPF).
- Look for the specific library and wildcard DSPF object (that is MYLIB/&DSPF).
- Look for a wildcard library and wildcard DSPF object (that is &LIB/&DSPF).

If there is still no match, then the search fails, and an error page is displayed that shows which DDS record could not be found.

Figure 12-6 shows the properties page for DDS object mappings. See 11.8.2, “Language dependent objects in the WebFacing project” on page 360, for an example of where it is necessary to change the default object mappings.

![Figure 12-6 DDS object mappings properties page](image)

**UIM Object Mappings**

UIM Object Mappings work under the same principles as the DDS Object Mappings. The exception is that UIM Object Mappings are for help panels.
12.7 Migrating between different IBM WebFacing Tool versions

If you have WebFacing projects from WebSphere Development Studio Client Version 4 or WebSphere Development Tools Version 5.1, you can migrate these to version 5 by using the import wizard. Follow these steps:

1. Select File → Import.
2. Choose WebFacing Projects and click Next.
3. Browse to the folder that contains the WebFacing projects that you want to migrate to version 5. Click Next.

Notice the following points:
- When you upgrade from version 4, the WebFacing projects is still available in your original version 4 workspace directory. Unless you changed the default location for the workspace, depending on the directory path you chose when installing version 4, you can find the workspace directory in a location such as X:\WDS\WSSD\workspace.
- When you upgrade from WebSphere Development Tools version 5.1, depending on the directory path you chose when installing version 5.1, you can find your WebFacing projects in a location such as X:\WDT\WebFacing\eclipse.

4. Any WebFacing projects that the wizard finds is listed in the WebFacing projects found pane. Select the projects that you want to import. The wizard creates an EAR file for the project. An EAR file is a Java 2 Platform, Enterprise Edition (J2EE) file. By default, the EAR file uses the name of your WebFacing project and adds _EAR to the name. If you want to use a different name, or if you are consolidating several projects into a single EAR file, you can change the name in the Enterprise Application Project field.

5. Click Finish. The project is migrated to the version 5 workspace.

6. To see the project in the WebFacing perspective, select Window → Open Perspective → WebFacing.

Notes:
- If your WebFacing project was already converted using the version 4 of IDE, it is not necessary to reconvert the project after the migration has completed.
- If your original project was migrated from WebSphere Development Tools Version 5.1, you must reconvert the project.
Building Web interactions with iSeries Web Tools

Unlike the discussion about the IBM WebFacing Tool, this chapter does not build upon 5250 applications. Instead, it integrates a new J2EE based Web application with traditional Integrated Language Environment (ILE) or Original Program Model (OPM) programs that run under OS/400 which can be called and parameters passed to and from them.

With the iSeries Web development tools of the WebSphere Development Studio Client for iSeries (WDSc) Version 5 you can create Web applications which interact with existing traditional applications and OS/400 objects without using a 5250 display file described with Data Definition Specifications (DDS). This chapter explains how to do this. It also describes using Web components (previously called Design-time Controls (DTCs)), the Web Interaction wizard, and Program Call beans.

Some functions described in this chapter are only supported on WebSphere Development Studio Client Advanced Edition for iSeries (WDSc AE). You must purchase this feature separately from the WDSc Standard edition. Even if you received WDSc as part of 5722-WDS free of charge because of the Software Subscription, you must pay a fee to acquire WebSphere Development Studio Client Advanced Edition for iSeries.

Changes in the second edition of this redbook

This chapter was originally written for WDSc 5.0. Now it has been updated to reflect the changes and enhancements added with WDSc 5.1 in 2003 and WDSc 5.1.2 in 2004, as well as several fixpacks between the major releases. Technical differences from the previous edition of this book are indicated by a change bar at the left of the text. Minor changes such as grammar corrections and slightly different phrasing are not marked.

The base for writing this chapter is WebSphere Development Studio Client Advanced Edition for iSeries Version 5.1.2.2 (called WDSc AE in the following), that is WDSc AE 5.1.2 with Fixpack 2 applied.
13.1 iSeries Web development tools

iSeries Web development tools help you create e-business applications that use a Web-based front end to interact with an OS/400 program on an iSeries server and with JavaBean methods.

**OS/400 program:** References to **OS/400 program** can be an OPM or ILE program or a procedure in service program created with any of the programming languages supported by OS/400 and i5/OS. Some publications only mention ILE programs in this context, but in fact, all types of programs are supported.

Many of the Web tools provided with WebSphere Development Studio Client for iSeries are exactly as inherited from WebSphere Studio Site Developer Advanced and WebSphere Studio Application Developer and not unique to iSeries. In addition, several tools in WDSc and WDSc AE are designed specifically for iSeries programmers to maximize productivity and leverage skills.

**Note:** WDSc V4 was based on WebSphere Studio Site Developer Advanced (WSSDa). In the meantime, WebSphere Studio Site Developer and WebSphere Studio Application Developer have been further enhanced. Now the term WSSDa is no longer used.

These iSeries-unique tools include:

- **Web Interaction wizard:** This wizard generates a Web interaction which connects to an existing OS/400 program object or an ILE procedure. The interaction accepts parameters from an input JavaServer Page (JSP) and passes them to the OS/400 program through iSeries run-time classes, which are also generated by the wizard. Then it sends the output of the program to an output JSP file. Finally, it generates a struts action class that ties all the pieces together.

  An interaction is defined by the communication that occurs between your Web pages and the business processes. In addition, you can use the wizard to define message handling, session, flow control, and other configuration options for the Web pages in your application.

  The wizard uses information entered by the application developer into a series of windows to add code to the JSP files and to update the struts-config.xml file. The wizard also uses this information to generate the iSeries classes that communicate between the Web pages and traditional OS/400 programs or JavaBeans that execute the business logic for your application.

  You can manually create the input and output JSP files before you use the Web Interaction wizard. Or you can let the wizard generate the input page, the output page, or both. In either case, you can insert specific fields and controls (generally referred to as **Web components**, previously called DTCs) on your pages to enhance their input and output functionality.

- **iSeries Web Tools Run-time Configuration wizard:** This wizard complements the Web Interaction wizard. It must be used to specify, per Web project, run-time information that all wizard-generated interactions use. This information includes sign-on information and library list setup.

- **iSeries Web components:** The functions of these components were formerly implemented as Design-time Controls (DTC). Early publications referred to the components as Visual Custom Tags (VCT). These are Web page widgets that are available in the Page Designer editor. They can be inserted into a Web page and generate Hypertext Markup Language (HTML) and JavaScript on your behalf, based on properties...
you specify. These properties are similar to display files, to leverage your skills, such as edit code.

The Web Interaction wizard also helps perform the following tasks:

- Choose the methods within a JavaBean to be invoked as input and output parameters.
- Define the host program definitions, such as the program, structure, and parameter, and associate them with the Web components on your Web pages without working directly with JSP and Java code.
- Link the Web component fields and controls on your Web pages to the parameters in your ILE program.

You can also test your application without deploying the JSP files and iSeries run-time classes to an iSeries server. The iSeries Web Tools Run-time Configuration wizard defines the iSeries host and the run-time configuration information, such as user profile, password and libraries, that is required to run the application in the WebSphere test environment and saves that information in the web.xml file.

The information saved in web.xml file becomes part of the Web application and is available to any Java 2 Platform, Enterprise Edition (J2EE) application server, not just the test environment. However, a deployer or administrator can modify this information. After you test your new application in the WebSphere test environment within Eclipse, deploy and install it on the WebSphere Application Server on the iSeries host. You can use the Run-time Configuration wizard again to define the configuration for that environment.

13.1.1 Web pages

Web pages perform the role of the user interface for your Web application. For iSeries Web applications, you can create the pages in the following ways:

- Design and create the input and output pages using the Page Designer editor.
- Generate the pages using the Web Interaction wizard.

In addition to the techniques described in our examples, you can also create Web pages based on a Page Template, use the Interaction wizard to generate Web pages based on a Page Template, or apply a Page Template to an existing page.

Designing your Web pages using the Page Designer editor

Page Designer is the default editor when editing the JSP files that are used as the input and output pages of your Web application. The JSP files are contained in your Web project. You can use the features of the editor to make the pages as sophisticated and advanced as you want. When you are ready, you use the Web Interaction wizard to:

- Define the parameters associated with the fields and controls on your pages.
- Define how you want to handle errors.
- Generate the run-time classes for the Web application and add a form attribute to the input JSP.

A new Page Designer is available in WebSphere Development Studio Client for iSeries V5. The Page Designer from WDSc V4 is now referred to as Page Designer Classic.

Using a page template

A new capability introduced with WDSc 5.1.2 is a Page Template File wizard. The wizard can create an iSeries Web Components Page Template, which you use to apply to a JSP file containing iSeries Web Components.
You can use a page template as a starting point to create a new HTML, XHTML, JSP, or Faces JSP page in Page Designer. Each new page looks like the page template and usually contains a section of the page for adding your own information. Page templates let you maintain a consistent look and feel for your entire Web site. For example, you can have consistent formatting across pages or use the same graphic on each page.

Also, if you use the Web Interaction wizard to generate input or output pages, you can choose a Page Template to be applied to those pages.

If you apply changes to the page template, the pages created from that page template are automatically updated when you save the updated page template. For example, if your company's logo changes, you need only add the new logo to the page template and then save it. All pages created from that page template are automatically updated with the new logo.

Page templates include the following features:

- Ability to create a page template and save it for future use.
- Ability to apply a page template to a page as you are creating the page.
- Option of inserting contents and page fragments into a page or template.
- Ability to merge the page template into a page so that any changes are not applied to the pages created from that template.
- Ability to apply changes you make to a page template to any pages created from that template.
- Ability to apply new page templates to pages created from different page templates.

Note, there are several circumstances that can cause a page template to not be applied to a Web page when using any of the page template wizards:

- Page template is not applied to a Web page that has a different encoding from that of the selected page template.
- Page template is not applied when either the selected page template or the Web page is described by XML Style JSP.
- Page template is not applied to a Web page that already has a page template associated with it.

Guide to Online Help: For more detailed information about this topic, open the online help function and perform the following steps:

1. Select Help → Help contents from the workbench menu.
2. In the left pane of the WebSphere Studio information center click Developing
3. Expand Web applications
4. Expand Web site design
5. Expand Web sites
6. Expand Editing the look and feel of a site
7. Expand and click Page templates.

You can find the WebSphere Studio information center also at: http://publib.boulder.ibm.com/infocenter/iadthelp/index.jsp

Generating Web pages using the Web Interaction wizard

When you use the Web Interaction wizard to generate the pages, the pages are saved as JSP files in your Web project. The wizard adds Web components to the files, and generates the iSeries run-time classes for you. The files provide a minimal amount of function and usability.
However, after the files are created, you can edit them to modify the existing Web components, to modify page display properties, to add images, and to update the text on the page and the labels on the buttons. If you change any of the links for the fields, or the host program name, then you can re-run the Web Interaction wizard to update the coding in the JSP files and to regenerate the iSeries run-time classes.

If you have created a page template, the Web Interaction wizard can also generate input and output pages based on that page template for you.

### 13.1.2 Web components

The iSeries Web components are the fields and controls that appear on the Web pages of your Web application. You can insert the Web components by dragging an icon from the palette into the JSP pages (see Figure 13-23 on page 467), that you design or by selecting Insert → iSeries Web Components (see Figure 13-36 on page 478). Then you use the Web Interaction wizard to link them to parameters in your ILE program and Java Bean method, as we describe in 13.4, “Tieing JSPs to RPG with the Web Interaction wizard” on page 483.

**Note:** The context menu options are specific to each particular WDSc view.

The iSeries Web components are:

- Form
- Simple Table
- Button
- Check Box
- Combo Box
- Hyperlink
- Image
- Image Button
- Label
- Radio Button Group
- Selection Box
- Table
- Text Area
- Text Entry

**Subfile DTC functions:** These functions are integrated with the Table Web component in WebSphere Development Studio Client for iSeries V5.

These iSeries Web components can be built with WebSphere Development Studio Client for iSeries. They are used to communicate between the Web page and the iSeries host program.

The following enhancements of the Web Components have been introduced with WDSc 5.1.2:

- Support for cascading style sheets. You can define your own external or internal style sheet, or inline styles, to override the default styles of iSeries Web Components. This gives you extra flexibility in customizing the look and feel of your Web pages.
- It is now easier to design your own JSP page with the enhanced Palette view, and entry field validation is provided in Attribute view to decrease the chance of entering erroneous data.
- A new Page Template File wizard is available to create an iSeries Web Components Page Template, which you can then use to apply to a JSP file to iSeries Web Components.
Web components instead of DTCs

Web components replace the ActiveX DTCs from WebSphere Development Studio Client for iSeries V4. Web components are implemented through Visual Custom Tags (VCT), which can be used and rendered in the Page Designer.

VCT is a JSP custom tag with a visualizer for the Page Designer design view. The original DTCs supported in the Page Designer Classic of WDSc V4 generate HTML code and JSP scriptlets. With the use of JSP custom tags, large amounts of Java code are reduced in JSPs by moving them into tag implementation classes. This is referred to as the JSP Custom Tag Library.

13.1.3 Web Interaction wizard

The Web Interaction wizard helps you define input and output pages, fields, and parameters. You also use the wizard to identify the host program or JavaBean to call for each interaction. An interaction is defined by the communication that occurs between your Web pages and the business processes.

You can use the wizard to define message handling for the Web pages in your application. The wizard uses this data to add a form attribute to the JSP files. The data also helps to generate the iSeries classes that communicate between the Web pages in your application and the ILE host programs or JavaBeans that perform the business processing for your application.

The following enhancements of the Web Interaction wizard have been introduced with WDSc 5.1.2:

- You can now use JSP files with plain HTML controls to build Web pages that interact with an iSeries program or procedure call in addition to iSeries Web Components.
- You can choose a Page Template for the generated input or output pages in the Web Interaction wizard.
- Message handling is improved such that you can now associate an iSeries Message file or a Java properties file with your Web application in the iSeries Web Tools Runtime Configuration wizard. This support enables your application to retrieve run-time messages from the properties or message file and display them on your Web pages.
- The flow controller feature is improved so that you can now connect multiple output pages from the Web Diagram editor and specify the flow condition in the Web Interaction wizard.
- You can now define a program or procedure to call when your Web application session times out.
- The flexibility of Java classes created by the Web Interaction wizard has been enhanced to include more methods and be organized into packages. These Java classes are generated from templates, and you can customize these templates as well.

13.1.4 iSeries Web Tools Run-time Configuration wizard

The iSeries Web Tools Run-time Configuration wizard is used to define host-related run-time information and to determine whether to use detailed trace or the JCA Program call connector. When this is done, you can unit test a Web application that communicates with an ILE program on an iSeries host without deploying the JSP files and generated run-time classes (action, form bean, and Program Call beans) to the host.

You use the wizard to identify the host where your ILE program is located and to define and modify your run-time library list. You also use it to specify whether you want to see the details for any errors that occur during the run time of your Web application.
13.1.5 Sample application used in this chapter

This chapter focuses on using WebSphere Development Studio Client for iSeries to add a Web interface to an existing iSeries application, written in pure RPG. Customer Master Inquiry, the original RPG program, uses 5250 display files to communicate with the user. This chapter shows how to create Web pages instead of the display file panels. The following sections show you how it appears to the user of a 5250 terminal or emulation.

Starting the application

To start the application, follow these steps:

1. Add LIB6961 to your library list and call the main program from an iSeries command line:

   CALL MLG265

2. On the first Customer Master Inquiry window (Figure 13-1), enter the search criteria. This may include the ZIP code or the name.

3. The second Customer Master Inquiry window (Figure 13-2) shows a subfile, which lists customers that match the search criteria. Type x next to the customer whose information you want to see.

   ![Figure 13-1 Customer Master Inquiry](image)
Figure 13-2  Customer subfile

The Customer Master Inquiry window in Figure 13-3 shows the detailed description of the customer that was selected.

Figure 13-3  Customer detail information
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 that communicates with ILE RPG programs. The sample code is available in the LIB6961.savf file of the additional downloadable material. See Appendix A, “Additional material” on page 679, for download and installation instructions.

Adding a Web interface to the application
To convert the iSeries host application to the Web application with Development Studio Client for iSeries, follow these steps:

1. Create JSPs using Development Studio Client and Web components.
2. Create an 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) or Enterprise Application Resource (EAR) file.

Alternatively, you can use the Web Interaction wizard to create the JSPs for you and then modify them. Or you can also use a Web diagram to generate the Web interaction.

The sample 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 an interaction or the session manager determines that the browser user is inactive.

To enable the session manager to make this determination, the \textit{Invalidate session after interaction occurs} check box must be selected on the Specify a Name and Location for your Web Interaction window (Figure 13-79 on page 523) for an interaction. When a session ends, the associated server job (QZRCSRVS running under the QUSER user profile) is also ended.

Figure 13-4 shows how the application generated by WebSphere Development Studio Client for iSeries runs.
You can find more information about iSeries Web development tools in the online documentation as explained here:

1. In the workbench menu, click Help → Help Contents and a new Window opens.
2. Click Product Overview in the left pane.
3. Expand Development Studio Client.
4. Expand Overview of IBM WebSphere Development Studio Client for iSeries.
5. Expand Tools overview.
6. Click iSeries Web development tools.

### 13.2 Developing the Web application

The steps to build the Web application based on the customer inquiry application are summarized as follows:

1. Create a Web project.

   Web projects contain the Web resources that you create and maintain for Web applications. Web projects typically contain servlets, JSP files, Java objects, static documents (for example, HTML pages or images), and any associated metadata. See 13.2.1, “Creating a dynamic Web project” on page 441.

2. Design Web application pages.

   The Web application pages are JSP files that you create or that are created for you by the Web Interaction wizard. Design the page and define the iSeries host program input using Web components.

3. Creating a Web interaction.

   The Web Interaction wizard adds code to your files and generates iSeries classes. This enables the communication between your Web pages and the ILE host programs or
JavaBeans that perform the business processing for your application. See 13.4, “Tieing JSPs to RPG with the Web Interaction wizard” on page 483.


The following sections explain the steps in more detail for our sample application.

### 13.2.1 Creating a dynamic Web project

The following sections guide you through all steps needed to create a dynamic Web application. explains how to design the Web pages and define the input and output fields. Follow these steps:

1. Start the WebSphere Development Studio Client for iSeries. Click Start → Programs → IBM WebSphere Studio → Development Studio Client {Advanced Edition} for iSeries 5.1.2. Whether you choose Development Studio Client Advanced Edition or not depends on whether you have it installed.

2. Select or type the path for the workspace you want to use.

3. Create a new project. Select File → New → Project as shown in Figure 13-5, or click the triangle at the right of the Open New wizard icon or press Ctrl+N.

![Figure 13-5 Creating a new dynamic Web Project](image)
4. In the left pane of the New - Select window (Figure 13-6), select Web and click Dynamic Web Project in the right-hand pane.

5. Click Next.

Note: In WDSc 5.0 and earlier, you created a “Web Project” and an option within the wizard dialog (Figure 11-7 on page 340 of the first edition of this book) decided whether it was static (just containing HTML only) or dynamic. Now, there are two different project types: static or dynamic Web project. Also note that some of the following dialogs have been changed by moving some options to different panels and using a different terminology occasionally.
6. In the New Dynamic Web Project panel (Figure 13-7), enter the following information:
   a. Type the name of the project in the **Project name** field. In our example, we use `cusing`.
   b. Keep the default **Project location** entry. It should be the workspace you selected when you started WDSc.
   c. Important: Make sure the **Configure advanced options** check box is selected.
   d. Click **Next**.

Figure 13-7   Naming the Dynamic Web Project
7. On the J2EE Settings Page panel (Figure 13-8), follow these steps:

   a. Enter an EAR project name.

      Dynamic Web projects are always imbedded in Enterprise Application projects. The wizard that enables you to create a Web project requires that an Enterprise Application (EAR) project exist, or will create one for you. Creating a Web project will update the application.xml deployment descriptor of the specified Enterprise Application project to define the Web project as a module element.

      ![Image of New Dynamic Web Project dialog box](image)

      **Figure 13-8  Defining J2EE settings**

      Any new or existing Enterprise Application project must be associated with your new Web project to facilitate deployment to an application server. The default for the new EAR project name is DefaultEAR. However, if you use this name, the files for multiple Web projects that you create may be contained in the same EAR project.

      If you want to keep the data for one Web project separate from other Web projects in your workspace, you can choose to click New to enter another EAR project name or you can type over and enter a specific name for the EAR file. We choose to type cusinqEAR in the New project name field for this example.

   b. Accept the defaulted-in Context root to ensure that it has 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.

   c. Select the 1.3 J2EE level, unless you plan to deploy the Web application on WebSphere Application Server V4 or older. However, some of the functions described in the following sections may not work if you select J2EE level 1.2.

   d. Click Next.
8. On the Features Page panel (Figure 13-9), follow these steps:
   a. Select **Add Struts support**.
   b. Ensure that **Default style sheet (CSS file)** is checked, so a CSS file is generated for your project.
   
   ![New Dynamic Web Project dialog](image)
   
   **Figure 13-9**  Selecting project features
   
   c. Select **Include the iSeries Web components Tag Library**. This Tag Library contains visual tags that help you create input and output pages to use with the Web Interaction Wizard. It is necessary to enable many of the functions described in the following sections.
   
   **Tip:** When you click the text (not the check box) for any of the **Web project features**, listed in the left pane, you see a short description in the right pane of the dialog.
   
   d. For most Web projects, you do not need to enter more and you may click **Finish** here. However, to see the additional panels, click **Next** now.
e. The Struts Settings dialog as shown in Figure 13-10 allows you to override default settings for Struts. Different from previous versions of WDSc, a Web diagram is created automatically with the project unless you deselect the *Create a Web diagram for the project* option here.

![Struts Settings dialog](image)

*Figure 13-10  Struts Settings dialog*
f. Click **Next** to see the *Select a Page Template for the Web Site* dialog.

![Image of Select a Page Template for the Web Site dialog]

**Figure 13-11  Select a Page Template for the Web Site**

This dialog allows you to select one of the sample Page templates or import an existing user-defined one from a different project or workspace. See “Creating a page template” on page 454 how to create a user-defined Page template.

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

It takes a couple of seconds to create the project as a directory structure underneath your workspace directory.

If you did not have the Web perspective open, when you started the process to create a new project (as explained in step 3 on page 441), the dialog shown in Figure 13-12 is displayed now. It allows you to control whether or not the Web perspective is opened automatically. Furthermore, by selecting the *Do not show this message again* check box, you can decide to either switch to the appropriate perspective or not without being prompted in the future.
You see the cusinq project and the cusinqEAR project in the Project Navigator view in the upper left pane of your workbench. In earlier versions of WDSc, the Project Navigator view was called Project Navigator.

Also new with WDSc 5.1.2 is the fact that a Web diagram with the name diagram.gph has been created for you because you added struts support to your project. The diagram is automatically opened in the editor pane.

Unlike in previous versions of WDSc (or WSSD or WSAD) the empty Web diagram does not appear as a white space in the editor pane. It rather shows a short description of what you can do with a diagram as indicated by the mouse pointer in Figure 13-13. This description disappears, once you have added some content to the diagram.
You can use this Web diagram to visually represent a Web interaction as we explain in a later section “Web diagram” on page 513. However, we first show you examples describing how to create Web pages and an Web interaction without involving a Web diagram.

**Web project organization**
The J2EE model, and more specifically, the Sun Microsystems Java™ Servlet 2.3 Specification, defines a Web application directory structure that specifies the location of Web content files, class files, class paths, deployment descriptors, and supporting metadata. The Web project hierarchy mirrors that of the Web application created from a project.

The main project folder contains all development objects related to a Web application. The WebContent folder contains the elements of the project necessary to create a Web application. This folder structure maps to the Web application archive (WAR) structure defined by Sun. The following default elements are located in the Web project folder hierarchy:

After you created a Web project, as described in the previous sections and show in the Web perspective, you see the Project Navigator view in the upper left pane as shown on the left side of Figure 13-14. In the Project Navigator view, Web projects are filtered into folder nodes to customize the display of Web resources for easy management during development.

In other words, the Project Navigator view looks similar to the directory structure of the Web project, but is not exactly the same as you can see when you compare the three navigators shown in Figure 13-14.
To see the structure of the Web project, you can open the Resource Perspective (Select Window → open Perspective → Other → resource) or the Navigator view (Select Window → Show View → Other → Basic → Navigator). The following default elements are located in the Web project folder hierarchy:

- **JavaSource** - The JavaSource folder contains the source code of Java classes. Note, this is not the same as the Java Resources node in the Navigator view.

- **WebContent** - The WebContent folder contains the contents of the WAR file that will be published to the server. It contains all the Web resources, including HTML files, JSP files, and graphics needed for the application. Any files not under WebContent are considered development resources (for example .java and .sql files) and will not be deployed when the project is unit tested or published.

- **WebContent/theme** - The WebContent/theme folder contains cascading style sheets and other style-related objects.

- **WebContent/WEB-INF** - The WebContent/WEB-INF folder contains the supporting Web resources for a Web application, including the web.xml deployment descriptor file and the classes and lib directories. The web.xml file contains general information about the Web application, including servlet mappings, security information, and the welcome page. The
ibm-web-bnd.xml file contains bindings to references used at the Web module level as defined by IBM. The ibm-web-ext.xml file is used by WebSphere Application Server to support additional options beyond the J2EE specification such as reloading intervals and the default error page. The struts-config.xml file is the deployment descriptor for a Struts application to initialize its own resources, such as: ActionForms, to collect input from users, ActionMappings, to direct input to server-side Actions, and ActionForwards, to select output pages.

In the course of development, the web.xml and the struts-config.xml files are updated automatically to reflect changes to your Web project. For example, when the Web Interaction wizard is used to create a Web interaction in a Web project, it automatically places the appropriate servlet entries into the web.xml file. When you are testing your application in the WebSphere Test Environment, and the web.xml file has been updated, the change is not always recognized. In this case you need to restart the project. To do this go to the Server view, right-click the server, select Restart Project and then select the project. Watch the console view for a message indicating that the application has been started.

- **WebContent/WEB-INF/classes** - The WebContent/WEB-INF/classes folder contains servlets, utility classes, and the Java compiler output directory. The classes in this directory are used by the application class loader to load the classes. Folders in this directory will map package and class names. The .class files are automatically placed in this directory when the Java compiler compiles the source files from the source directory. Any files placed directly in this directory will be deleted by the Java compiler when it runs.

- **WebContent/WEB-INF/lib** - The WebContent/WEB-INF/lib folder supports .jar files that your Web application references. Any classes contained in these .jar files will be available for your Web application.

**Project Navigator view**

The Project Navigator view is customized for Web developers as an alternative to the Resource perspective's Navigator view. The Project Navigator view is the default project view in the Web perspective.

All operations available within the Resource perspective’s Navigator view are also available in the Project Navigator view through Workbench or context menus.

- **Web Site Navigation**: This editor is for creating the navigation structure of a Web site and can be used to automatically create navigation links on all pages. This configuration file only exists if you choose to use the Web Site Designer feature (a default setting) when you create a Web project. Double-click this file to open Web Site Designer, which includes a Navigation view and a Details view.

- **Web Deployment Descriptor** - This file corresponds to the WebContent/WEB-INF/web.xml file. You can double-click to open the file in the deployment descriptor editor, or select Open With from its context menu to open the file with a different editor.

- **Java Resources** - This node displays Java resources within the project. If the project contains a single Java source folder, the packages and classes (for example, servlets, beans) within the source folder will be shown directly beneath the Java Resources folder node. If the project contains multiple source folders, each source folder will appear beneath the Java Resources folder and can be expanded to show their packages and classes.

- The node Libraries underneath Java Resources contains the library JAR files defined in the project properties. Three types of JAR files are shown:
  - JAR files included in the project's WebContent/WEB-INF/lib directory
  - JAR files external to the project, such as j2ee.jar and rt.jar
  - Project libraries, which are special references to a Java project
When a Web project is exported, a JAR file is automatically created from the Java project to be used by the Web application during runtime. Libraries are shown in classpath order. By default, only the JAR files contained within the project are shown. You can also display external JARs and project libraries by selecting Show Referenced Libraries from the Project Navigator view’s Menu button.

- **WebContent**: This folder contains items to be published to the server. By default, this folder will be named `WebContent` for newly created static and dynamic Web projects. The `WebContent` folder contains the contents of the WAR file that will be deployed to the server. It contains all the Web resources, including HTML files, JSP files, templates, graphics, and javascripting files needed for the application. Any files that are not under `WebContent` are considered development resources (for example Java and Structured Query Language (SQL) files) and are not included into a WAR or EAR file when the project is unit tested or published.

**The Struts Explorer view**

The Struts Explorer view provides you with a way to navigate through the various referenced parts of a Web application from a starting point such as a Web page (JSP or HTML file), an action mapping, a global forward, or, in Version 1.1 of the Struts standard, a global exception. Parts of a Web application can be JSP files, links, action mappings, form beans, and so forth. Some of them are resources; some are contained within a resource.

With the Struts Explorer view, you can get a general idea of how the different parts of a Web application are linked together and how different parts reference each other. This view shows the possible execution path of the Web application from Web pages through links, actions, forwards, and for Struts 1.1, exceptions. In the case of modules, for each Web page in the module, the view assumes a module execution context of that page so that as you follow the execution path, it is resolved correctly. Additionally, the viewer provides a flat listing of the Struts parts that are configured in the Struts configuration file, per module for Struts 1.1 Web applications. This way, you can easily access these parts to be used with the Web diagram editor.

The view also provides additional information about a part to facilitate Web application development. Among the information displayed are errors that may cause a Web application to run incorrectly and data parameters that a part requires.

### 13.2.2 Methods for designing Web application pages

Web pages, that is JavaServer Page (JSP) files, perform the role of the user interface for your Web application. For iSeries Web applications, you can create pages in the following ways:

- Design and create input and output pages using the Page Designer editor.
- Generate pages using the Web Interaction wizard.

You can use each of the methods listed to achieve the same results. In either case, you can insert specific fields and controls (referred to as iSeries Web components), on your pages to enhance their input and output functionality.

The following paragraphs outline both methods. Then in “Designing Web application pages” on page 453, we describe an example using the first method by designing the Web pages from scratch. Later, in “Creating the getDetail Web interaction” on page 495, we also describe an example for the second method, letting the Web Interaction wizard generate an output page.
Designing your Web pages using the Page Designer editor

Page Designer is the default editor you use when editing the JSP files that are used as the input and output pages of your Web application. The JSP files are contained in your Web project. You can use the features of the editor to make the pages as sophisticated and advanced as you want. When you are ready, use the Web Interaction wizard to:

- Define the parameters associated with the fields and controls on your pages
- Define how you want to handle errors and other configuration options
- Generate run-time classes and update the struts-config.xml and web.xml files

Generating the Web pages using the Web Interaction wizard

Instead of creating new JSP pages from scratch, you can use the Web Interaction wizard to generate the pages. The pages are saved as JSP files in your Web project. The wizard adds Web component fields and controls to the files. It also generates iSeries run-time classes.

When generated for the first time, the files provide a minimal amount of function and usability. However, after the files are created, you can edit them to modify the existing Web components, to modify page display properties, to add images, and to update the text on the page and the labels on the buttons. If you change any of the links for the fields, or the host program name, then you can re-run the Web Interaction wizard to add or change a form attribute in the JSP files and to regenerate the iSeries run-time classes.

13.3 Designing Web application pages

The following sections assume that you can start creating your Web application pages by building a master JSP containing elements used by all pages, such as your company name and logo. Now, with WDSce 5.1.2, you can use a Page Template (a .jtpl file) to do so. This is an easier approach compared to using a JSP fragment file we used for our example in the first edition of this redbook.

You design the style, content, and structure of the files using the Page Designer editor by following these steps:

1. Create a page template file.
2. Customize the file.
3. Edit a style sheet.
4. Apply the style sheet to your master JSP file.
5. Create additional JSP files using the master JSP file.

Guide to Online Help: For more detailed information about this topic, open the online help function and perform the following steps:

1. Select Help → Help contents from the workbench menu.
2. In the left pane of the WebSphere Studio information center click Developing
3. Expand Web applications
4. Expand iSeries Web applications
5. Expand and click Designing Web application pages.

You can find the WebSphere Studio information center also at:
http://publib.boulder.ibm.com/infocenter/iadthelp/index.jsp

You may also refer to Chapter 7 in the redbook WebSphere Studio Application Developer Version 5 Programming Guide, SG24-6957, for more information about Page Designer and other topics not limited to iSeries. Some more recent information not covered in this book can...
13.3.1 Creating a page template

In order to achieve the same look and feel for all Web pages within your application, you can use a new capability introduced with WDSc 5.1.2: the New Page Template File wizard. Generally, you can create a page template in two ways:

- Create a new page template.
- Create a page template from an existing Web page file.

In our example, we choose to use the first option.

Create a new page template

After you created a Web project as described in “Creating a dynamic Web project” on page 441, you may start with describing the common characteristics of all pages through a page template by performing the following steps.

1. Expand the project cusinq in the Project navigator within the Web perspective and right-click WebContent or select File → New → Web → Page Template File from the menu. The New Page Template File wizard as shown in Figure 13-15 opens.

![New Page Template File wizard](image)

2. Leave the default /cusinq/WebContent for the folder to store the new file.
3. Enter a file name Master_Template for the new page template file in the File Name field.
4. In the Markup Language field, choose HTML as the markup language.
5. In the Model field, choose Template containing iSeries Web Components Tag Library as a model for the new template.
6. Select the **Configure advanced options** check box and click **Next**.

7. Leave the entries on the Tag Libraries page showing iwcl as prefix and /WEB-IBF/IWCLTagLib.tld as is, and click **Next**.

8. The Master.css style sheet and the iSeriesWebComponents.css style sheet files are listed by default in the Style Sheet field of the next dialog shown.

   This indicates that the style sheets are linked to the new page template file. All JSP files that are created using this page template file will also contain links to these style sheets. The style definitions in the style sheet files are applied to the JSP files and to the iSeries Web components that are added to those files, respectively.

9. Click **Finish** to create the page template file.

After a few seconds, the new page template Master_Template.jtpl is created under the WebContent folder, opens in the Page Designer editor as shown in Figure 13-16 and is ready to be modified to match the desired style of your Web application.

---

You can now add content to the newly created page template. The content is the part of the template that appears in each page that uses the template. Here are several ways to create content on the page:

1. Enter text directly on the page and change attributes as needed.

2. Drag and drop components from the palette. The choices on the palette differ depending on what kind of page template you created. For example, if you have created a Faces
components page template, you can drag and drop components from the Faces components drawer on the palette.

3. Open the Page Template drawer on the palette and drag and drop a page fragment to the page. A page fragment lets you include part of a page from another file (similar to an SSI include or a JSP include). Changes in the page fragment are also applied immediately to all the pages created from the template.

A content area, depicted by the mouse pointer in Figure 13-16, is automatically included in the file. This content area of the page template is reserved for the content that you add to a JSP file that uses this new page template file. In “Designing the layout of the new Page Template file” on page 456 we show how you can customize your new page template.

Restriction: To avoid problems in running the iSeries Web Interaction wizard, do not insert a form or an iSeries Web component in a page template file that you intend to apply to the input and output JSP pages of an iSeries Web application.

Create a page template from an existing file
Alternatively, if you already had created one Web page and want to use its look and feel for all other pages in this in the project, you should perform the following steps:

1. In Page Designer, open the existing HTML or JSP file that you want to be a page template.
2. Make sure to insert a content area in the appropriate position on the page. To create a content area, open the Page Template drawer on the palette, drag and drop a content area to the page, and then add content to the content area.

Designing the layout of the new Page Template file
The Page Designer editor is the default editor for JSP and HTML files as well as page templates. It provides the capability to apply a wide range of design elements to your files. For information about these design elements, you can search the online information for Page Designer (see the Guide to Online Help box on page 459). A small set of the available design elements is used in the following examples:

1. Delete the Place content here text that appears on the page in the editor view by either pressing the Delete key several times or marking the entire string with the mouse and pressing the Delete key once.
2. In the white space of the Page Designer editor, type ABC Office Supplies.
3. If the attributes view does not appear automatically in the lower left pane, right-click in the white space next to the text you entered and select Attributes.
4. In the lower left pane, you see now the Attributes view. In the Paragraph field, select **Heading 1**. In the Alignment field, select **Center** as indicated by the mouse pointer in Figure 13-17. Notice that the size and position of the text “ABC Office Supplies” changes immediately.

5. You may want to add some graphics to enhance the look of your application. In the middle left pane of your workbench, click the **Gallery** tab.

6. In the Gallery, expand **Image** and click the **Illustration** folder as indicated by the mouse pointer in Figure 13-18.
Figure 13-18 Opening the Illustration folder in the Gallery View

7. The Thumbnail view opens in the lower left pane. You may scroll through the window to see all the images in that folder or double-click its title bar to maximize the window.
8. Select an image and drag it to the left of the text “ABC Office Supplies” and a second one to the right as shown in Figure 13-19.

9. Select File → Save from the menu bar or press Ctrl+S to save your new page template file.

10. Close the file by selecting File → Close from the menu, pressing Ctrl+F4, or clicking the X button in the tab on top of the editor pane.

At this stage, the page contains a centered company name heading and two graphics. The text font, size, and color, and the background color, are all default values applied by the editor.

Guide to Online Help: For more detailed information about the capabilities of Page Designer, open the online help function by performing the following steps:

1. Select Help → Help contents in the workbench menu:
2. Click Developing
3. Expand Web applications-
4. Expand Web page design
5. Expand Pages
6. Expand Page Designer
7. Click Working in the Design page.
Editing a style sheet

You can change style definitions to enhance the design of your JSP files in several ways:

- Edit the style definitions in the `Master.css` style sheet file
- Modifying style sheet definitions for iSeries Web components
- Creating style definitions within your JSP files
- Adding inline style definitions for the iSeries Web components in your JSP files

If you use a template file when you create your JSP files, any changes that you make to the style sheet files used by the template file are automatically applied to the JSP files.

In our example, we show the first method to edit the style sheet to change the default values that were applied to the JSP file:

1. In the Project Navigator view, expand the `WebContent` and then `theme` folder. Double-click `Master.css` to open the file in the `CSS editor`.

   The source view for the file appears in a frame on the right within the editor pane. The Preview area appears in a frame on the left of the source view and the `Styles View` opens at the lower left pane of the workbench.

   If you click any tag definition in the source view or the Styles view, the preview area displays the effect of the style definition for that tag. At the same time, the Styles View respectively the Source editor is synchronized to the tag selected. In fact, the Styles View behaves similar to the Outline View for programming languages as described in 6.1.4, “Outline views” on page 155.

2. Click `BODY` in the source view of the style sheet to highlight the line. Right-click and select `Edit` to open the Set Style Properties window.

3. Change the font definition for `BODY`:
   a. Click `Fonts` in the left frame.
   b. Under Font family, click `Times New Roman` and click `Remove`.
   c. Click `sans-serif` in the right field and click `Add`.

4. Change the background for `BODY`:
   a. In the left pane of the “Set Style properties --- BODY” dialog, select `Background`.
   b. Type `#f8f7cd` in the Color field as shown in Figure 13-20. This number describes the Red-Green-Blue (RGB) value that consists of `#` followed by six hexadecimal numbers. An RGB value indicates a single color with combination of red, green, and blue.
   c. Click `OK` to apply the style definition changes to the body tag and to view the effect in the preview area.

   Tip: Instead of typing the RGB value of the color, you may also do one of the following:
   - Click the colored button next to Color, and select a color from the Color dialog box.
   - Click the `eyedropper` icon on the right and then click anywhere on the screen where you want to pick up the color. The Color button and field will be updated with that color and RGB value.
5. Change the font definition for H1:
   a. In the Styles view, click H1 and the corresponding line in the Source view is highlighted. Also the current definition of the Heading 1 style is applied to the example shown in the preview area.
   b. Right-click and select Edit to open the Set Style Properties window.
   c. Under Font family, select Times New Roman and click Remove.
   d. In the right field, click sans-serif and click Add.
   e. Click OK to apply the changed font definition and to view the result in the preview area.
6. Repeat step 5 on page 461 to change the font definition for the H2 tag.
7. Add a style definition for the paragraph (P) tag:
   a. Right-click in the Styles, preview or source view of the style sheet and select Add.
   b. In the Set Selector of the New Style window, select the HTML tag radio button.
   c. In the HTML tag name area, locate and click P or type the letter P for the paragraph tag. Click OK.
   d. In the Add Style window, complete these tasks:
      i. Select and add sans-serif for the font family.
      ii. In the Color field, type #6666CC.
      iii. In the size list, select 10.
      iv. Click OK to apply the new definitions to the style sheet.
8. Add a style definition for FORM:
   a. Right-click in the source view of the style sheet and select Add.
   b. In the Set Selector of the New Style window, select the HTML tag radio button. In the
      HTML tag name area, click FORM for the form tag. Click OK.
   c. In the Add Style window, complete these tasks:
      i. Select and add sans-serif for the font family.
      ii. In the Color field, type #6666CC.
      iii. In the size list, select 10. Click OK to apply the new definitions to the style sheet.

9. Add a style definition for the layout box:
   a. Right-click in the source view of the style sheet and select Add.
   b. In the Set Selector of the New Style window, select the HTML tag radio button. In the
      HTML tag name area, click DIV for the layout box tag. Click OK.
   c. In the Add Style window, complete these tasks:
      i. Select and add sans-serif for the font family.
      ii. In the Color field, type #6666CC.
      iii. In the size list, select 9.
      iv. Click OK to apply the new definitions to the style sheet.

10. Select File → Close or press Ctrl-F4 to close the style sheet and save your changes when
    prompted to do so.

---

Guide to Online Help: For more detailed information about changing font, color and other
character attributes, open the online help function. Select Help → Help contents in the
workbench menu and perform the following steps:

1. Click Developing
2. Expand Web applications-
3. Expand Web page design
4. Expand Pages
5. Expand Creating and editing Web pages - overview
6. Click Changing character attributes.

For more information about style sheets:

1. Click Developing
2. Expand Web applications-
3. Expand Web development tools
4. Click CSS Designer.
5. After reading the Cascading Style Sheet (CSS) Designer page, go to the bottom of the
   page and click any of the items listed under Related tasks.

You can also use the information search function in the online help and enter “style sheet”
as your search argument

---

Creating the first input page
Now you are ready to create a JSP containing the input field for the first Web interaction.
Follow these steps:

1. Expand the project within the Project Navigator View in the upper left pane of the Web
   perspective and right-click WebContent.
2. Select New → JSP File from the context menu.
3. In the File name field of the New JSP file dialog that opens as shown in Figure 13-21, type a name for the JSP file, for example, cusinq. A file extension is not required because the wizard automatically applies .jsp to the file name.

![New JSP file dialog](image)

**Figure 13-21** New JSP file dialog

4. In the Folder field, ensure that the path /cusinq/WebContent appears.

5. Select the Create from page template check box as indicated by the mouse pointer in Figure 13-21.

6. Click Next to go to the Page Template File Selection dialog.

7. Select the User-Defined Page Template radio button.

8. In the Thumbnail area on the page, select the template file Master_Template.jtmpl you created in “Create a new page template” on page 454 and click Finish. The new JSP file is created and opens in the editor view.

9. Select File → Save to save the new file with the design inherited from the template file you created.

Any content that you defined in the template file appears in the new JSP file, and the content area from the template also appears in the JSP file. You use this content area to add text, a form, and iSeries Web components or standard HTML components that are specific to this page. The new file also reflects any style sheet changes that you made in “Editing a style sheet” on page 460.
Adding a unique heading and user instructions

Now you need to define the elements unique to each page, especially those to enter user input or display variable output data. First enter the constant text to identify the page and instruct the user what to do:

1. Remove the text Default content of bodyarea and replace it with the following text:
   
   Customer Master Inquiry

2. In the bottom left corner of the workbench, click the Attributes tab. The Attributes view allows you to modify the paragraph attributes of the area where your cursor is currently positioned. Select Heading 2 for paragraph and Center for Alignment. Make sure that the shaded box still surrounds the text you entered in the full width of the window.

3. Click with your mouse pointer under the heading you just added and enter the text (for reference, see Figure 13-22 on page 465):
   
   Enter the ZIP code into the field and click the Submit button.

Inserting a form on the JSP

Any input field in an HTML or JSP file must be inside a form. Therefore, you must add a form to your page:

1. Place the cursor below the statement Customer Master Inquiry on the page.

2. Right-click and select Insert → Form and Input Fields → Form or you can simply press Ctrl+O to insert a form in your page.
   
   An new possibility to add a form, is to click the Palette tab in the frame on the right of the Web perspective, click the iSeries Web Components button, select Form from the list, and drag and drop it on the JSP page as shown in Figure 13-22.

The form is required for Web interface purposes. All input fields and Web application controls used by the Web application to communicate to a host program or a JavaBean, except the Table Web component, must be contained within a form. An exception is the Table Web component, which needs to be outside of the form to be interpreted. You can only have one form on a JSP page.
Adding Web components to the JSP page

When you create JSP pages for a Web application, you can add iSeries Web components and (new with WDS 5.1.2) also standard HTML components to the pages for the fields and controls that are used in the Web interaction. By using iSeries Web components, you can:

- Provide convenient ways for the user to enter input data and pass it to the iSeries application.
- Format the output data from that application.

Alternatively, standard HTML components can also be used for this purpose. However, the iSeries Web components provide a higher level of convenience similar to what application developers are used to when they define display files with OS/400 Data Definition Specifications (DDS). Table 13-1 shows the iSeries Web components and the equivalent standard HTML components that are supported for Web interactions in iSeries Web applications.

Table 13-1 Comparing iSeries Web components and the equivalent standard HTML components

<table>
<thead>
<tr>
<th>iSeries Web component</th>
<th>HTML component supported for Web interactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button</td>
<td>Submit Button</td>
</tr>
<tr>
<td>Check Box</td>
<td>Check Box</td>
</tr>
<tr>
<td>Combo Box</td>
<td>Drop-Down Box</td>
</tr>
</tbody>
</table>
You can add Web components to JSPs in the following ways:

- Drag and drop the Web component onto the Design view of the JSP page.
- Drag and drop the Web component onto the Source view of the JSP page.
- Select Insert → iSeries Web Components to add the Web component to the JSP page.
- Click the Insert a Web Component icon () on the tool bar to insert the Web component to the JSP page.

The following steps explain this process by using the first method mentioned above to drag and drop onto the Design view of the JSP page:

1. In the right pane of in the Web perspective, click the Palette tab at the bottom as indicated by the mouse pointer in Figure 13-22 on page 465. From the displayed tabs, click the iSeries Web Components tab to display the available Web components.
2. Click the iSeries Web Components tool bar button as shown in Figure 13-23 and drag and drop Text Entry into the form on the JSP page and a frame depicting the text entry field similar to the one in Figure 13-24 appears.
3. The Attributes view appears in the lower left pane of the Web perspective. You can now define the behavior and appearance of the text box by setting one more options as shown in the following example.
Depending on the size of your display, you may want to enlarge the pane to avoid scrolling as we did in Figure 13-25. You can also maximize the dialog by double-clicking its title.

a. Click the **General** tab.
   i. In the Name field, type **ZIP**.
   ii. In the Label field at the bottom of this pane (scroll down or maximize the view by double-clicking title bar), type **Zip code**.
   iii. From the Label position pull-down list, select **Left**.

b. Click the **Data** tab and set the data attributes as shown in Figure 13-26:
   i. For Type, select **Numeric**.
   ii. For length, type 5.
   iii. For Decimal places, type 0.

![Figure 13-26 Defining data attributes for the Text Entry Web component](image)

Alternatively, you may also choose to import the field characteristics from a DB2 UDB for iSeries table or OS/400 file description. If you click the **Get data attribute from iSeries database** radio button and then the **Get** 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 as shown in Figure 13-27.
c. Click the **Events** tab in the Attribute window as shown in Figure 13-28 to define certain actions on this field to cause an associated JavaScript to execute. The developer must supply the desired JavaScript.

![Figure 13-28 Text Entry Events Attributes](image)

![Figure 13-27 Adding DB2 UDB for iSeries reference field](image)

d. Click the **Styles** tab as shown in Figure 13-29 to modify fonts, colors, and other attributes, if they are different from the style sheet.
You have now defined how the Web page will look and the data fields that you want on the Web page.

4. 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 the cursor is located inside the form).

5. With the cursor to the right of the text entry field, press Enter one or two times to move the cursor down within the form while creating empty lines.

6. Click the iSeries Web Components tab within the panel in right pane.

7. Drag and drop the Button button from the palette view (shown in Figure 13-36 on page 478) to the new location on the JSP page or simply double-click it. The Attributes view for the iwcl:WButton component should open in the bottom pane left of the Web perspective. If it does not, click the Attributes tab in that frame.

8. On the General page of the Attributes window (Figure 13-30):
   a. Type submit (all lowercase) in the Name field.
   b. In the Label field, type Submit.
   c. For Button type, ensure that Submit is selected.
9. In the Editor Design view click in the area on the right of the Submit button and press the Spacebar to insert a blank.

10. Drag and drop the Button component again from the iSeries Web Components menu onto the JSP page next to the new Submit button.

11. In the Attributes view (Figure 13-31), in the Name field, type reset (all lowercase). In the Label field, type Clear. Select Reset in the Button type field to apply these values to the Web component.
You have now finished designing the page. Figure 13-32 shows you the Design view of cuisinq.jsp in Page Designer.

![Design view of cuisinq.jsp in Page Designer](image)

12. In the editor view, click the **Source** tab at the bottom.

13. Double-click the **cuisinq.jsp** tab on top of the pane to maximize the window to review the HTML code that was generated. With the cursor anywhere inside the source, right-click and choose **Format → Document** to view the source with indentations similar to Figure 13-33.

**Note:** The Design view in Figure 13-32 as well as the Preview shown in Figure 13-34 on page 475 show an advantage of using the Page template compared to the technique involving a JSP fragment we showed in the previous edition of this book. Not only all style settings are automatically inherited from the template, there also shown in either view whereas an included JSP fragment would not be visible here.
14. Click the Preview tab (at the bottom of the HTML editor view) to review how the page will look in a browser as shown in Figure 13-34.
15. Double-click the **cusinq.jsp** tab on top of the pane, to restore the original size of the pane and make all other frames visible again.

16. Click **File → Close** or press Ctrl+F4 to close the **cusinq.jsp** file and select to save your changes at the prompt.

**Creating an output JSP file**

For our sample application, we designed another page that corresponds to the second 5250 screen. The purpose of this page is to display information for each customer that has the same ZIP code as the user entered into the input JSP cusinq.jsp. Since the output consists of a variable number of records, you can use a technique similar to a **subfile** in a 5250 display, such as the one shown in Figure 13-2 on page 438.

With WebSphere Development Studio Client for iSeries Version 4, such a capability was provided with a **subfile DTC**. Now with WebSphere Development Studio Client for iSeries V5 and later, the same function is implemented with the **Table Web component**. The following steps explain how you can create an output JSP that contains a Table Web component:

1. Expand the project within the Web perspective and right-click **WebContent**.
2. Select **New → JSP File**.
3. In the File name field on the window that opens, type a name for the JSP file, for example, cusstfl. In the Folder field, ensure that the path /cusiing/WebContent appears. Also, verify that the Create from page template check box is selected.

4. Click Next and the Page Template File selection dialog is shown.

5. Select the User-Defined Page Template radio button.

6. In the Thumbnail area on the page, select the template file Master_Template.jtpl you created in “Create a new page template” on page 454 and click Finish. The new JSP file is created and opens in the editor view.

7. Select File → Save to save the new file with the design inherited from the template file you created.

8. Make sure the Design view is active, by clicking the Design tab on the bottom left of the editor pane.

   **Tip:** The Preview view may look very similar to the Design you, but you cannot change anything. So, if you cannot perform the following steps because the screen is write protected, very likely you did not switch to the Design view.

9. Remove the text Default content of bodyarea and replace it with Customer Master List.

10. In the Paragraph field within the Attribute view, select Heading 2.

11. In the Alignment field, select Center.

   To allow the user to easily return to the previous page, you can insert a link now.

12. Position the cursor below the page heading. Right-click and select Insert Link.

13. In the Insert Link dialog, complete these tasks:
   a. Click Browse next to the URL field.
   b. Select File.
   c. Expand the WebContent folder.
   d. Select the cusiing.jsp file. Click OK.
   e. Click OK.
   f. In the Link text field, enter Return to enter another Zip code.
   g. Click OK.

   At this point the cstsfl.jsp should look like the page shown in Figure 13-35.
The purpose of this JSP is to display all selected customers similar to how this was done in the 5250 application using a subfile as shown in Figure 13-2 on page 438. To achieve this on a Web page, you can use a Table Web component.

In WebSphere Development Studio Client for iSeries V4, there was a Table DTC and a Subfile DTC. In WebSphere Development Studio Client for iSeries V5, the functionality of both DTCs merged into a single Table Web component. The following steps explain how to insert a Table Web component into your output page by using the menu.

1. Make sure your cursor is positioned under the link Enter other Zip code as shown in Figure 13-35 and select Insert → iSeries Web Components from the menu as shown in Figure 13-36 or click the Insert an iSeries Web component icon on the tool bar to open the iSeries Web Components dialog.
2. In the iSeries Web Components dialog, select the Table icon as shown in Figure 13-37 and click OK.

Note, this is just an alternative method of inserting an iSeries Web Component. You could also use the Palette tab as shown in Figure 13-23 on page 467.
3. A graphic depicting a table with just a single cell appears at the position of your cursor in the Design pane. At the same time, the Attributes iwc1:WTable editor opens in the bottom left pane of the Web perspective. If it does not, click the Attributes tab in that frame.

Now, you need to enter information about content and appearance of the table and its cells.

4. In the Attributes editor, complete these tasks:
   a. Click the General tab as shown in Figure 13-38.
      i. In the Name field, enter SFL1.
      ii. For Selection mode, select Single from the drop-down list.
The Table Web component can be used to display database records similar to the way in which they are displayed in a subfile on the host. To display database records in a Table Web component, you need to:

- Specify the name of the service program called by the table.
- Code specific procedures that interact with subfile APIs on the host.
- Use the appropriate subfile APIs to provide the data to be display in the Table Web component.

See 13.8, “Subfile APIs” on page 526, for more information about using the APIs.

b. Click the **Parameters** tab (Figure 13-39).
   
i. Select the **Get data from an iSeries service program** check box.
   
ii. In the Library field, type LIB6961.
   
iii. In the Object field, type CUSTINQSFL.

   See “CUSTINQSFL Get Customer List ILE RPG service program” on page 533 for details on this program.

iv. Select the **Writes entire subfile** radio button.

v. In the Service program parameters table, click the **Add** button and enter the characteristics of the parameter to be passed to the program:

   - **Name**: ZIP
   - **Data type**: Zoned decimal
   - **Length**: 5
   - **Decimal**: 0

![Figure 13-39 Parameters for Table Web components](image)

c. In the Attributes window, click the **Columns** tab as shown in Figure 13-40. Then define each of the subfile fields. Click the **Add** button for each field:

   i. For ZIP, specify a **Length** of 5 and **Heading text** of **Zip code**.
**Note**, you may either have to maximize the attributes pane or click the scroll bar at the bottom of the pane to see the right most columns in particular the Properties column.

ii. Click into the Properties field and a little button appears on the right within the field as indicated by the mouse pointer in Figure 13-40. Click the button and the Column Properties dialog opens as shown in Figure 13-40. Define the Type as **Zoned decimal** with 0 Decimal places and click **OK** to close the column Properties dialog.

![Column Properties of iSeries Table Web Component](image)

iii. For the field CUST, specify a Length of 5, a Heading text of **Customer number** and define the Type as **Zoned decimal** with 0 Decimal places in the Column Properties window and click **OK** to close the column Properties dialog.

iv. Define the field NAME with a Length of 20, a Heading text of **Customer name**. Define the Type as **Character** and click **OK** to close the column Properties dialog.

At this point, the updated cussf1.jsp should look like the page in Figure 13-41.
The cussfl.jsp is not only an output page. It also allows the user to select one customer to display a customer detail page. Because information about the selected customer needs to be returned to a program, you must define a form and a submit button.

The selected record is provided by the Table Web component. That is why you had to select Single in the Selection mode drop-down box in Figure 13-38 on page 479.

1. Position the cursor at the right of the table and insert a form by pressing Ctrl+O.
2. Position the cursor inside the form. Right-click and select Insert → Form and Input Fields → Submit Button.
3. In the Insert Submit Button window (Figure 13-42), in the Name field, type submitSelection. In the Label field, type Show Detail.
4. Click OK.
There is no functional difference between the iSeries Web Component button and the Submit Button. However, we recommend that you use the iSeries Web Component button. This way, when the JSP is rendered at run time, all the fields and controls have a uniform style.

5. Save the new output page by pressing Ctrl-S and close the Page Designer by pressing Ctrl-F4.

A new input page and an output page are now designed and saved. You can now proceed to the next step of creating the Web interactions to link to ILE or non-ILE programs running under OS/400 or i5/OS.

Guide to Online Help: For detailed information about Page Designer, open the online help function. Select Help → Help contents in the workbench. Then click Developing and expand Web applications → Web development tools → Page Designer → Working in the Design page.

13.4 Tieing JSPs to RPG with the Web Interaction wizard

The Web Interaction wizard provided by WebSphere Development Studio Client for iSeries can create an interaction that allows you to exchanges data between a traditional OS/400 program, an input JSP, one or more output JSPs, and an associated error page. The basic elements that we deal with and tie together are:

- Web pages (JSP) and their input and output fields
- Parameters that are passed to and from the OS/400 program. In our examples, we used ILE RPG programs. You may also use any other programming language including Original Program Model (OPM) programs.

The Web pages that we designed earlier in this chapter can be used to create two Web interactions:

- The **getList Web interaction** uses the `cusing.jsp` Web page as an input page with one input parameter (ZIP code). The `cussfl.jsp` Web page serves as an output page with no parameters required because the subfile service program CUSTINQSFL (see Example 13-3 on page 533) is handling the displayed subfile.
- The **getDetail Web interaction** uses 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. At this point, the latter JSP does not exist yet, since we let the interaction wizard create it (see “Creating the `getList` Web interaction” on page 485).

Figure 13-43 shows the relationship between:
- Web pages `cusing.jsp`, `cussfl.jsp`, and `displaycustdetailResults.jsp`
- Web interactions `getList.wit` and `getDetail.wit`

When you create the JSP files that are used with the Web Interaction wizard, you must meet the following conditions:
- All Web components must be inserted into a form on the JSP page, except for the Table Web component. The Table Web component must be outside of the form.
- Multiple forms are not supported. You can only have one form on a JSP page.

13.4.1 The iSeries Web Tools Run-time Configuration wizard

The goal of the iSeries Web interaction wizard is to connect a Web application to a program running within an i5/OS or OS/400 job. To use this capability, you need to define information about the iSeries server that will be used for testing. You define the communication values with the iSeries Web Tools Run-time Configuration wizard by performing the following steps:

1. To open the wizard, right-click the `cusing` project in the Project Navigator view and select Specify iSeries Web Tools run-time configuration from the context menu.

2. On the iSeries Web Tools Run-time Configuration windows, follow these steps:
   a. Type your iSeries Host name, OS/400 User profile, and Password.

   Optionally you can select to be prompted for those values at run time or to Enable password encoding.
Only with WDSO Advanced Edition Version 5.1.2 or later, instead of hard coding the user profile and password, you can also decide to do one of the following:

- Use the J2EE Connector Architecture (JCA) connector and Java Authentication and Authorization Service (JAAS) credentials,
- Use Configure Enterprise Identity Mapping (EIM) based single sign on.

b. Click Next.

c. Add the library LI6961 to the Run-time library list and make sure to select the **Display detailed runtime errors** check box since you are testing a new application as shown in Figure 13-44.

d. Click Finish.

Figure 13-44  **iSeries Web Tools Run-time Configuration wizard**

### 13.4.2 Creating the getList Web interaction

To create the first Web interaction, follow these steps:

1. Select the **cusinq** project in the left pane of the WebSphere Development Studio Client for iSeries and do one of the following to open the Web Interaction wizard:
   - Right-click and select **New** → **Other** then select **Web** and **Web Interaction**,
   - Select **File** → **New** → **Other** from the menu bar then select **Web** and **Web Interaction**,  
   - Click the **Create or modify a Web interaction** icon on the tool bar.

2. The Web Interaction wizard dialog opens as shown in Figure 13-45. Type the name for the Web interaction that you are about to define. We use **getList** as the name.

   Leave the defaults for **Java package prefix** and **Use error page**.
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.

3. Click **Next**.

![Web Interaction wizard](image)

### Specify the Input and Output Pages for your Web Interaction

On the *Specify the Input and Output Pages for your Web Interaction* dialog as shown in Figure 13-46, 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.

1. We want to use the pages we created earlier for input and output. To do this, make sure **Use input pages** radio button is selected and click **Add**.

2. On the Input page window, select `cusing.jsp` for input and click **OK**.

3. Make sure **Use output page(s)** radio button is selected and click **Add**.

4. On the Output page window, select the output page `cussfl.jsp`.

5. Click the **Preview** button if you want to make sure you selected the correct page.

6. Click **Next**.
Now you need to define which program or procedure the interaction should use to perform the business logic. In addition the wizard needs to know which the input data that program expects to receive and which output parameters it returns. There three ways to provide this information:

- Manually key in the necessary information for the program and each parameter.
  To avoid mistakes, you may browse all programs on you iSeries instead of typing its name manually. If a parameter or structure has the same characteristics as a field or record format in an externally defined file, you can also browse the file object and select the definition.
- Import an existing Program Call Markup Language (PCML) file.
  The PCML file could have been entered manually or (more likely) generated by the ILE RPG or ILE Cobol compiler or Program Verifier. You can also use an existing PCML file that was previously generated by the Program Call wizard.
- Use the Import Source option to import a PCML file directly from ILE RPG or ILE COBOL source code. This capability was newly introduced with WDSc 5.1.2.

In all cases, two identical PCML files are created and stored in the root and in folder Java Resources within your project. In addition, an MPCML file is also created in the project root. PCML is a tag language that helps you call RPG or Cobol programs, from Java with less Java code.

The names for the PCML file and the MPCML file generated by the Web Interaction wizard are the same as the Web project name. For example, if the Web project name is cusinq, then the PCML file that is generated is cusinq.pcml. Similarly, the name of the MPCML file is

Figure 13-46  Specifying input/output pages for Web Interaction
You do not need the .mpcml file when you deploy your application or services. It is used only in the workbench.

**Note:** If you create a new Web project and then import all of the files, including the PCML and MPCML files, from another Web project, the name of the PCML and MPCML files no longer match the name of the new Web project. As a result, the Web application fails to run correctly. You must rename the PCML and MPCML files to match the Web project name to resolve this problem.

Likewise, if you rename an existing Web project, be sure to rename the PCML and the MPCML files to match the new Web project name.

The following section describes how you can provide that information using the Import Source option as the easiest alternative.

**Specify the input and output parameters for your iSeries host program**

In the next window as shown in Figure 13-47, you specify the OS/400 program to be called along with a definition of each parameter which must be passed.
As of WDSc 5.1.2 you can now import the definitions for the program name and its call parameters from the source member rather than having to describe each parameter manually. To do so, instead of manually typing the program name (as in Figure 13-47), perform the following steps:

1. In the Import Source dialog, make sure the Use an iSeries program or procedure button is selected.

2. Click the Import Source button at the lower left of the dialog.

3. Select Remote File System from the Source file pull-down menu as indicated by the mouse pointer in Figure 13-48 and click OK.

4. In the Browse for File dialog, you may either create a new one or select an existing connection to your iSeries system. To do the latter, expand the node for that connection.

5. If the source for your program resides in a traditional OS/400 library (as opposed to a IFS stream file), make sure to expand the tree under /QSYS.LIB.

6. Then expand the library, LIB6961 in our example, and then the source physical file (WEBCOMPO1).

7. Click the desired member, CUSINQNEW.MBR in our example and click OK.

   Note, that it may take several seconds, till the Import Source dialog shows the selected program (CUSINQNEW) in the left pane and the Program Call Markup Language (PCML) tree view in the right pane as shown in Figure 13-49.
8. Select the program and click the **OK** button.

   The program name along with all parameters (in this example only the field ZIPS) are copied into the Web interaction wizard dialog.

   **Note:** In order to be able to create the PCML for your source program, the workbench must be able to compile or verify the source code. One reason for not being able to compile the program successfully can occur if the program contains externally described files and the appropriate library is not part of the library list of your server job. See 5.5.1, “Adding a library list entry” on page 118 how to add one or more entries to the library list.

   The program alias is the name of the Java bean to be generated, which calls your RPG program **CUSINQNEW**. In this example, we use the same name for the Java bean as for the RPG program.

9. Back in the *Specify the input and output parameters for your iSeries host program* panel, single click the program name in the left pane.

10. Make sure the correct library name appears in the right side of panel as indicated by the mouse pointer in Figure 13-50.

    **Important:** With our tests, we saw some cases, where QSYS was used instead of the application library. Make sure to correct that.

11. Select the **Associate this program with the interaction** check box.

12. Click the **OK** button.
13. In the left pane of the panel, expand the tree under the program name.

14. Single click the parameter and the definition of this parameter appears on the right as shown in Figure 13-51.

15. Select Input from the Usage field pull-down menu.

16. Click the OK button.

Our example uses only one call parameter. If there are more parameters, you need to repeat the above steps 13 through 16 for each field and describe which field is used for input, output, or both.
Click the **Next** button.

**Guide to Online Help:** For detailed information about this topic, open the online help function by performing the following steps:

- a. Select **Help** → **Help contents** in the workbench.
- b. Click **Developing**.
- c. Expand **Web applications**.
- d. Expand **iSeries Web applications**.
- e. Expand **Creating iSeries Web interactions**.
- f. Expand **Using JSP files for a Web interaction**.
- g. Under Related references in the right pane click **iSeries Web Interaction wizard pages**.
- h. Click **Program call definitions**.

**Map and Link the Input Parameters to the Input Fields**

At this point you have defined the input and output JSPs to be used as well the business logic to be called and the parameters passed to and from the program or procedure. Now you need to map the input and output fields of the JSPs to the appropriate call parameters.
In the *Map and Link the Input Parameters to the Input Fields* dialog as shown in Figure 13-52, you can see input fields from the JSP page in the left pane. All of the input parameters of the RPG program you defined in “Specify the input and output parameters for your iSeries host program” on page 488 appear in the right pane. The purpose of this dialog is to relate these two sets of parameters, by performing the following steps:

1. While the ZIP parameter in the left pane is selected, set the value of the *Save to Session* property in the lower-left pane to *true*.

   Note, the *Save to Session* Property is very important to make sure the content of this parameter is passed on to the next interaction. See also “Save to Session property not set” on page 508.

2. Select matching fields and parameters (in our case only the ZIPS field) 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 and the fact whether they are linked or not.

3. Click *Next*.

*Figure 13-52  Map and link input parameters*
On the Map and Link the Output Parameters to the Output Fields page (Figure 13-53), you do not need to link any parameters.

9. Click Finish and the Web Interaction wizard starts generating run-time classes, updates the run-time descriptor files, and updates the JSP files.

The wizard uses information you entered to update the action attribute of the form within the input JSP to the interaction URL and the onsubmit attribute to the Java scriptlet return isSubmitOK(), for example:

```<FORM action="/cusing/getList.do" onsubmit="return isSubmitOK();">```
The wizard also updates the strut-config.xml file and generates the run-time classes that communicate between the Web pages and the ILE program you specified in Figure 13-47 on page 488.

**Tip:** Each Web interaction is represented by a WIT file in your Web project. To modify a Web interaction, double-click the WIT file to open the Web Interaction wizard and modify the settings for the interaction. You see the same panels as shown in Figure 13-45 on page 486 through Figure 13-53 on page 494.

### 13.4.3 Creating the getDetail Web interaction

After you created the getList Web interaction, you can now create the getDetail Web interaction. Two facts are worth mentioning here:

- This interaction uses the same page, cussfl, as input page, which was used as output page in the getList Web interaction described previously.
- At this point, we did not create an output page for the getDetail Web interaction yet. For the sake of showing different methods, we leave it to the interaction wizard to create one for us.

#### Add LIB6961 to the library list

Similar as in “Creating the getList Web interaction” on page 485, you can use the Import Source later in this section to import the parameter definitions for the RPG program GEDTDETAIL. In order to so, a successful program verification needs to be performed. For the GEDTDETAIL program, this is only possible, if you add library LIB6961 to your library list *before* you start the interaction wizard. To do so:

1. Open the Remote System Explorer (RSE) perspective.
2. Right-click Library List and select Add Library List Entry.
3. Enter LIB6961 into Additional Library field.

#### Start the interaction wizard

To Create the getDetail Web interaction, perform the following steps:

1. Click the WebContent node in your Web project. Then select File → New → Other from either the menu bar or the context menu.
2. On the Select window, select Web in the left and then Web Interaction in the right pane.
3. Click Next.
4. On the Specify a Name and Location for your Web Interaction window, the destination folder is the same as for the first Web interaction. The Web interaction name is getDetail.
5. Click Next.
6. On the Specify the Input and Output Pages for your Web Interaction window as shown in Figure 13-54, select cussfl.jsp for the input page.

7. Click the Generate output Page radio button.

   Note, in this step of the example you use a different approach as you did for the getList interaction in Figure 13-46 on page 487: So far, you first created the input and output pages before running the Web Interaction wizard. Now, as shown in Figure 13-54, you request that the output page is generated by the wizard by clicking the Generate Output Page radio button. The output page that is to be generated appears in the field on the left of the dialog.

   The page name is a combination of the Web interaction name (getDetail) plus Results.jsp. Therefore the output page that is created for you has the name getDetailResults.jsp, which does not conform with our original design of the application (displaycustdetailResults.jsp as shown in Figure 13-43 on page 484). To rename the JSP, including all references to it, you can single click getDetailResults.jsp and then click the Rename(C) button. A dialog allows you to specify the desired name for the output page.

8. Since we want the new page to inherit the style attributes and layout defined with the page template created in “Create a new page template” on page 454, make sure to select the Generate input/output page(s) using the page template check box.

9. Click Next.
10. In the *Specify the Input and Output Parameters for your iSeries Host Program* dialog, you can use the Import Source technique described with Figure 13-48 on page 489 again to describe the parameters for the GETDETAIL program automatically.

Note, in case the Import Source is not successful, the reason might be that library LIB6961 is not part of your library list. Close the interaction wizard and see “Add LIB6961 to the library list” on page 495 for more information.

**Important:** Before clicking the **Add Program** button, this panel already shows an Edit Program dialog with the CUSINQNEW program details you added earlier for the getList interaction. Make sure that you do *not* change that definition. Both programs, GETDETAIL and CUSINQNEW should now appear in the left pane.

11. After you imported the information from the source file, single click the GETDETAIL program in the left pane, select the **Associate this program with the interaction** check box in the right and click **OK**.

12. Expand the GETDETAIL program in the left pane, verify the parameters and change the usage field right pane:
   - The Subfile name parameter (SFNAME, data type character with length 10):
     - Select **input** in the Usage field.
     - Click **OK**.
   - For customer number (character with length 5):
     - Select **output** in the Usage field.
     - Click **OK**.
   - For **NAME** (character with length 20):
     - Select **output** in the Usage field.
     - Click **OK**.
   - For the **ADDR** parameter (character with length 20):
     - Select **output** in the Usage field.
     - Click **OK**.
   - For City (character with length 20):
     - Select **output** in the Usage field.
     - Click **OK**.
   - For State (character with length 2):
     - Type **output** in the Usage field.
     - Click **OK**.
   - Zip code (zoned decimal with length 5):
     - Select **output** in the Usage field.
     - Click **OK**.

13. Click **Next**.
For details see the online help and expand the following topics:

1. Reference.
2. User Interface reference.
3. iSeries Web Applications.
4. iSeries Web Interaction wizard pages.
5. Program call definitions.
6. iSeries program or procedure.

Map and Link the Input Parameters to the Input Fields

1. On the Map and Link the Input Parameters to the Input Fields panel, link the input parameter SFLNAM to the input field SFL1 (Figure 13-55).

   ![Map and Link the Input Parameters to the Input Fields panel](image)

   **Figure 13-55** Get Detail input parameters

2. Click Next.

   **Note:** The Map and Link the Output Parameters to the Output Fields dialog does not appear at this point. The reason is, because you choose to let the Interaction Wizard generate the output page in Figure 13-54 on page 496. Instead you see the Design the Result Form panel as shown in Figure 13-56.

   However, if you later edit the Web interaction after it has been successfully created, the Design the Result Form panel never appears again and the Map and Link the Output Parameters to the Output Fields dialog is shown instead, because at that point the output page does exist.
Design Result Form dialog
The Design Result Form dialog appears now, because you had decided to let the wizard generate the output page for you.

1. In the pane on the lower left of the Design Result Form dialog, you can modify the page title, colors, and the labels for each field to display. Change the Page Title from Result Form to Customer Details (Figure 13-57).

Figure 13-56  Result Form

Figure 13-57  Result form title change
2. To change the field labels, select a field and click the **Fields** tab below the pane.

3. Click **Next**.

4. Since you selected in Figure 13-54 on page 496 to base the generated output page on a page template, the Specify Page Template dialog appears now. Click the **User Defined Page Template** radio button and select **Master_Template.jtpl**.

5. Click **Next**.

6. The **Map Page Elements to the Page Template** panel as shown in Figure 13-58, allows you to indicate how the page template should display the contents of the file using the page template.

7. Click **Next**.

8. Now, the **Select Web Pages**, allows to specify the JSP pages to which the page template applies, if there were more than one.

9. Click **Finish**.

Now the getDetail interaction is ready. You should see its configuration file **getDetail.wit** as part of the project.
Adding links to other pages
We also want to give the user an opportunity to go back to the customer list to enter a new ZIP. Add a link to return the user to the Zip code entry page. Use the same technique as explained in step 13 on page 476.

13.4.4 More complex interactions
As of WDSc 5.1, you can also use multiple input and output pages for a Web interaction. The multiple pages are specified when you select the options *Use input pages and *Use output pages on the wizard page Specify the Input and Output Pages for your Web Interaction by clicking the Add button multiple times.

Using multiple output pages and the Flow controller option
To specify that you want different output pages to appear based on the logic in your program, you define an output parameter as a flow controller. You can select the Flow Controller option on the Map and Link the Output Parameters to the Output Fields page of the Web Interaction wizard. To open the Flow Control Specification window, click the Value field and then click the button that appears in the field.

You use the Flow Control Specification window to map parameter values to JSP names. During run time, the returned value for these parameters are compared to the values that you specified in this window and causes the run time to display the corresponding page.

If Use error page is selected on the first page of the Web Interaction wizard, then *OTHER is defined in the Value field and error.jsp is in the Output JSP field in the table within the Flow Control Specification window. This means that if the output parameter returns any values not specifically defined in the Flow Control Specification window, then the error.jsp page displays in the browser.

If *OTHER appears in the table, it cannot be deleted or moved up or down in the list. Any new items that you add are placed above *OTHER in the list.

If Use error page was not selected, then the table in the window is empty.

Click New to specify a value and define a JSP page to display when that value is returned as the flow controller from your program. Each value in the Values column must be unique. You can change the JSP files defined in the Output JSP column, including the JSP file defined for *OTHER.

If you remove a JSP file from your Web project that had been defined as an output JSP file for flow control purposes, the file name still appears in the table in the Flow Control Specification window, but you cannot select it.

Click Delete to remove a selected value and output JSP page from the table.

Use Move up and Move down to rearrange the order of the values and output JSPs in the table.

The following RPG code snippet is an example of using a flow controller. The code checks that a record was selected in a subfile. If it is not selected, an error page is displayed. If it is selected, then the next page is displayed:

```rpg
PGetCustRec       B                   export
D GetCustRec      pi
D forward                            20A
D userinfo                            likeDS(CustInfo)
*```

Chapter 13. Building Web interactions with iSeries Web Tools  501
* Read the selected record from the CUSTLIST subfile
C       Eval   rc=readcSF(CustList):
C           %Addr(CustRec):
C           %Size(CustRec):
C           1)

* If user didn't select anything
C   If     rc < 1
C   Eval   forward = "NOSEL"
C   Else
C   Eval   forward = "OK"
C   EndIf

* If user selected a record ...
C   If     rc > 0
C     . . .

Using multiple input pages
You can use pages in a parallel, independent organization, and link program parameters to
the same fields on multiple pages. Parallel use of pages implies that the pages are
independent of each other. The independent pages can contain fields that are linked to the
same input parameters in a host program. However, only one of the pages can be used to
input data to the host program for processing at any one time. The results of the processing
are returned to the output pages defined in the interaction.

Consider the following example:

- Use input pages input1.jsp, input2.jsp, and host program pgm1.
- Page input1.jsp contains input fields in1, in2, and a Submit button.
- Page input2.jsp contains input fields in3, in4, and a Submit button.
- The Submit button on each page calls pgm1 with the action class generated by the Web
  Interaction wizard.
- Program pgm1 has input parameters inhost1 and inhost2.
- Parameter inhost1 is linked to in1 in input1.jsp, and to in3 in input2.jsp.
- Parameter inhost2 is linked to in2 in input1.jsp, and to in4 in input2.jsp.

When the application runs, either input1.jsp or input2.jsp displays in the browser.

- If input1.jsp is displayed, the user enters values for in1 and in2, and clicks Submit to
  initiate the call to pgm1 and passes the data that was entered to parameters inhost1 and
  inhost2 for processing.
- If input2.jsp is displayed, the user enters values for in3 and in4, and clicks Submit to
  initiate the call to pgm1 and to pass the data that was entered, to parameters inhost1
  and inhost2 for processing.

In this parallel use of independent pages, the same program parameter can be linked to more
than one field if the field is on a different page. In this way, more than one input page can be
defined in the Web Interaction wizard to call the same host program.

13.4.5 Testing the 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 iSeries run-time classes to WebSphere Application
Server. Now you can run the Web application. Running the Web application locally uses the
default WebSphere test environment.
1. Right-click cusinq.jsp (not the cusinq project) under the cusinq Web project in the Project Navigator view and select Run on Server. The Server Tools feature automatically performs the following tasks:

- Launches the Server Perspective window.
- Creates a server project with Servers as the default name.
- Starts the application from an instance of Internet Explorer embedded in the Eclipse workbench.

![Server Selection dialog](image)

Figure 13-59  Server Selection

- Creates the server instance with WebSphere V5.1 Test Environment as the default name. You may select a different name or select to run it on an existing server by responding to the Server Selection dialog as shown in Figure 13-59.
- Creates the server configuration (or uses an existing one, if you test the same project again or you decided in Figure 13-59 to use another project’s server).
- Sets the server instance to use the server configuration.
- Publishes your project to the server configuration.
- Starts the server instance (this may take time to start depending on the server configuration settings).
Be patient, it may make take several minutes, even after the last information box disappeared, until you see the first page of your application (Figure 13-60). We recommend that your workstation has at least 768 MB or 1 GB (preferred) of memory to use the test environment efficiently.

2. You may need to click the input field, because focus is often on the address line of the integrated Web browser. Enter the value 55901 into the input field, if you are using the sample library for this book. Click Submit.

3. The CUSINGNEW ILE RPG program and the getList Web interaction have not been involved yet. Figure 13-60 shows a static HTML page with an empty input field. After you clicked the Submit button, the getList Web interaction called the CUSINGNEW ILE RPG program and passed the ZIP code as input parameter.

Provided no error occurred, you see the output page cussfl.jpg as shown in Figure 13-61. We maximized the view within the workbench by double-clicking the tab on top, so that all parts of the page can be seen.

4. Restore the browser window to its original size by double-clicking the tab on top again.
5. In the lower right pane, you see the Console view (Figure 13-62 on page 506), which shows all informational and error messages during execution. You can double-click the tab on top of it to enlarge the view. You may still need to scroll to the right to see messages in their full length.

The following list identifies parts of the output messages in the console window.

- **A**: The input value received from cusinq.jsp to be passed later to the ILE RPG program (CUSINQNEW).
- **B**: The iSeries server job that handles all of your requests between the Web application and the host ILE program.
- **C**: Invoking the program call JavaBean CUSINQNEW.
- **D**: The host program call CUSTINQSFL from Table Web component.
- **E**: Writing records to a subfile using Add a record to the subfile QdtsAppendSF API.
Figure 13-62  Informational messages in the console
6. The last window in our application is Customer Details (Figure 13-63). Select one of the records and click **Show Detail**.

![Customer Details](image)

After testing is completed, your next task is to export the EAR file to the application server as explained in 13.7, “Exporting the Web application to production” on page 525.

**Examples for problem determination with the Web interaction wizard**

Since nobody is perfect, there is always a possibility that either a function of the product is not working correctly or you have made a mistake using it. The following section illustrate some common mistakes with the Web interaction wizard.

**Wrong parameter name in Table Web component**

Example 13-1 captures an error situation, where the parameter name specified for the Table Web component in Figure 13-39 on page 480 does not match the one expected by the ILE RPG program. Note however, that the runtime errors printed to the console have been updated and no longer refer to Subfile DTC. Therefore with WDS 5.1.2, the messages look slightly different from what is shown in Example 13-1.

**Example 13-1 Wrong parameter name in Table Web component**

```text
73bb207 SystemOut 0 starts preprocessing action [getListAction]
73bb207 SystemOut 0
  <br>  Form GetListInputForm data { <br>
  [ZIP] = {[55901]}<br>  }<br>
73bb207 SystemOut 0 iSeries host job id [033522/QZRCSRVS/QUSER].
73bb207 SystemOut 0 invoking host program alias [CUSINQNEW]
73bb207 SystemOut 0 postprocessing action [getListAction]
73bb207 SystemOut 0
  <br>  Form GetListOutputForm data { <br>
  }<br>
```
You can find more information about debugging Web applications in 14.3.4, “Testing and debugging an iSeries Web application” on page 578.

**Save to Session property not set**

Another common mistake occurs, if you neglect to set the *Save to Session* property for a field which needs to be passed on to the next interaction as described in step 1 on page 493. In our example, the error materializes in a way that an empty subfile table is shown with no apparent error message at the first glance as you can see in Figure 13-64.

![Figure 13-64 Save to Session property not set](image-url)
However, if you examine the console messages, you find the following error message:

306357:13:12.242 0 >** Subfile Error ** : the value for parameter name, [ZIP] cannot be found in the associated session variable, [ZIP].

check the name of the session variable name defined in 'Parameters' page of Subfile Table and the corresponding session variable name of the input field defined Web Interaction Wizard.

13.5 Struts tools

Struts is a set of Java classes and JSP tag libraries that provide a conceptual framework for developing Web applications. The Struts technology is open source and was developed as part of the Apache Software Foundation's Jakarta project.

The following sections present an overview of Struts and outline how WebSphere Studio supports the technology. You can find more information about struts on the Web at:


Model 2 and Struts

You use Struts to simplify the development of the applications. Struts-based applications are divided into three functional areas:

- **Model**: The model is the business logic, which in most cases involves access of data stores like relational databases. The development team that handles the model may be expert at writing DB2 COBOL programs, or EJB entity beans, or some other technology appropriate for storing and manipulating enterprise data.

- **View**: The view is the code that presents images and data on Web pages. The code comprises JSPs and the JavaBeans that store data for use by the JSPs.

- **Controller**: The controller is the code that determines the overall flow of events.

This *Model-View-Controller* division (sometimes called *model 2*) has several benefits:

- You can isolate development effort to some extent, so that implementation changes in one part of the Web application do not require changes to another. The developers responsible for writing the business logic can work independently of the developers who are responsible for the flow of control. Also Web-page designers can work independently of the developers.

- You can more easily prototype your work. For example, you may:
  a. Create a prototype Web application that accesses several workstation-based programs.
  b. Change the application in response to user feedback.
  c. Implement the production-level programs on the same or other platforms.

- Outside of the work you do on the programs, your only adjustments are to configuration files or name-server content, not to other source code.

- You can more easily migrate legacy programs, because the user interaction is the same wherever those programs run and however they interact with data stores.

- You can maintain an environment that comprises different technologies across different locations.
As shown in Figure 13-65, the Web server at run time contains both the view and controller components of a model 2 Web application. A third tier, which is usually outside of the Web server, contains the model. The diagram also references Struts-specific components that are described in the next section.

13.5.1 View, controller, and Struts

The Struts contribution to the view is two-fold:
- Struts provides the Java class org.apache.struts.action.ActionForm, which a Java developer subclasses to create a form bean. At run time, the bean is used in two ways:
  - When a JSP prepares the related HTML form for display, the JSP accesses the bean, which holds values to be placed into the form. Those values are provided from business logic or from previous user input.
  - When user input is returned from a Web browser, the bean validates and holds that input either for use by business logic or (if validation failed) for subsequent redisplay.
- Struts provides numerous, custom JSP tags that are simple to use, but are powerful in the sense that they hide information. Page Designer does not need to know much about form beans, for example, beyond the bean names and the names of each field in a given bean.

The Struts contribution to the controller is as follows:
- The Struts controller servlet handles run-time events in accordance with a set of rules that are provided at deployment time. Those rules are contained in a Struts configuration file and specify how the servlet responds to every outcome received from the business logic. Changes to the flow of control require changes only to the configuration file. The controller servlet is an instantiation of class org.apache.struts.action.ActionServlet and is hereafter called ActionServlet.
- Struts also provides the Java class org.apache.struts.action.Action, which a Java developer subclasses to create an action class. At run time, the ActionServlet is said to perform actions. This means that the servlet invokes the perform method of each of the instantiated action classes. The object returned from the perform method directs ActionServlet as to what action or JSP to access next.

We recommend that you promote reuse by invoking business logic from the action class rather than including business logic in that class.
Struts does not contribute directly to model development. However, the Struts configuration file provides an elegant way to control the circumstances under which the model components are invoked.

13.5.2 WebSphere Studio and Struts

Struts application development tools that are provided by WebSphere Studio make it easy for you to build and manage a Struts-based Web application. WebSphere Studio provides:

- The ability to set up a Struts project so that tag libraries and other Struts-related resources are located properly. As a result, you can reference those resources without fail as you develop your application.
- Wizards to create form beans and action classes so that you have a head start in developing the logic that is specific to your application.
- A specialized editor to create and modify the Struts configuration file.
- Struts support for validation and editing, for example, by helping you to use Struts tag libraries.
- A visual assembly tool, which has the following benefits:
  - Helps architects to design the flow of a Struts-based Web application and to communicate that design to other professionals
  - Embeds Struts code in several components
  - Provides quick access to resource-appropriate editors and wizards
  - Separates team responsibilities for greater productivity and focus

13.5.3 Struts-based Web applications

A Struts-based Web application, also known as a Struts application, is a Web application that has Struts support added. Struts applications use a Struts framework to implement a model-view-controller design approach to building Web applications.

In WebSphere Studio, each Struts application has an ActionServlet configured in the application's deployment descriptor (web.xml file). For a Web application to be a Struts application, the ActionServlet must not only be configured but also must be used somewhere in the application.

13.5.4 Struts actions

A Struts action is an instance of an Action class subclass that implements a portion of a Web application and returns a forward. An action mapping is a configuration file entry that associates an action name with an action, a form bean, and a local forward.

13.5.5 Struts cheat sheet

A cheat sheet is a program that helps you accomplish a complex task that involves a variable series of subtasks. The Struts cheat sheet creates a Struts-based Web application.

13.5.6 Struts configuration file

A Struts configuration file contains information about data sources, form beans, global forwards, and action mappings.
Struts configuration files are in XML format. In WebSphere Studio, the XML editor has been enhanced to use the Document Type Definition (DTD) configuration file.

13.5.7 Struts form beans

A form bean is an instance of an ActionForm class subclass that stores HTML or JSP form data from a submitted client request or that stores input data from a link that a user clicked. A form bean is a type of JavaBean. An HTML or JSP form comprises fields in which the user can enter information.

When a browser submits an HTML form, the Struts ActionServlet completes these actions:
1. Looks at the field names from the HTML form
2. Matches them to the properties' names in the form bean
3. Automatically calls the setter methods of these variables to put the values retrieved from the HTML form

In addition, if you implement a validate method and set the validate flag in the corresponding action mapping entry in the Struts configuration file, the ActionServlet invokes the validate method.

13.5.8 Struts forwards

A Struts forward is an object that is returned by an action and that has two fields: a name and a path (typically the URL of a JSP). The path indicates where a request is to be sent. A forward can be local (pertaining to a specific action) or global (available to any action).

13.6 Using a Web diagram to generate a Web interaction

You can use a Web diagram to visually represent a Web interaction. Then you can invoke the Web Interaction wizard to generate the necessary iSeries run-time classes for the interaction. This can be done in several ways:

- Create a representation of a Web interaction in the Web diagram by adding JSP file nodes and an action icon to the free-form surface. Draw connections to or from the nodes and the action icon. Then invoke the Web Interaction wizard to create the JSP files and to define and generate the Web interaction.
- Create the Web pages (JSP files) in your Web project, and drag and drop the files on to the diagram free-form surface. Add an action icon to the diagram and connect the JSP files represented in the diagram to or from the action icon. Then invoke the Web Interaction wizard to generate the Web interaction. Upon completion, the states of the parts represented in the Web diagram are updated and information for the interaction is added to them.
- Create the Web interaction before using the Web diagram. Drag and drop the input and output JSP files and the action class file onto the diagram. Since these files are realized, or rather that they already exist in the Web project, the representation of the JSP files and the action icon are not gray. The connections between the JSP files and the Action icon are solid lines. You can now invoke the Web Interaction wizard to update the interaction with any changes that you made.

This section explains what a Web diagram is. It also explains how you can create a Web diagram to represent a Web application that already exists based on the examples we showed earlier in this chapter. Then, it shows how you can start designing a new application from scratch within a Web diagram.
13.6.1 Web diagram

A Web diagram is a file that helps you visualize the flow structure of a Struts-based Web application. See 13.5, “Struts tools” on page 509, for more information about Struts. Because of the indirectness involved with a Struts application, the ability to visually see the application’s flow can help you to better understand the application.

For example, by just looking at the JSP source, you cannot see what Struts actions are invoked by a Struts tag library link. To access this information, you must follow the trail of the Struts action mapping from the JSP link to the deployment descriptor, to the action servlet, to the Struts configuration file. From the Struts configuration entry for the action mapping, you can find the action. In a large, complex Struts application, following and understanding these logic flows can be error prone and tedious.

The Web diagram editor offers the ability to reflect changes of underlying resources in real time. For example, on the editor’s free-form surface (FFS), you can create a node for a Web page named MyJSP.jsp that does not yet exist. A node that appears on the FFS before the resource that it represents exists is called unrealized.

If you then create the JSP outside of the diagram editor, the diagram editor is then informed of the JSP creation and reflects the change in its diagram. Similarly, if a realized part shown in the diagram has its underlying resource deleted, the diagram editor is informed of this deletion and reflect the change in the diagram.

Figure 13-66 shows the Web diagram of the fully-completed sample application that we described in previous sections. The parts of the diagram are nodes and connections.

![Web diagram of the sample application](image-url)
Web diagram nodes
The Web diagram editor displays the following icons for Struts nodes:

- **Action mapping** 🌈 represents an Action mapping defined within a Struts configuration file. The Struts configuration file must be defined within the same Web application as the Web diagram being edited.
- **Form bean** 🍀 represents a form bean defined within a Struts configuration file.
- **JavaBean** 🍀 represents a bean defined within a Struts configuration file.
- **Web page** 🌇 represents a JSP or an HTML page. The Web page must be defined within the same Web application as the Web diagram being edited.
- **Web application** 🏕️ represents a Struts-based Web application.
- **Struts module** 🍃 represents a Struts module and the intended entry point for that module.

Web diagram connections
A connection describes a relationship between two nodes. Different types of connections are identified in the Web diagram by color or shapes. You may change the color or shape by selecting Window → Preferences in the menu bar to open the Preferences window. In the window on the left side of the Preferences window, expand Web Tools → Struts Tools → Web Diagram Editor. The default colors are:

- **Black**: General connection
- **Red**: Action Mapping Input connection
- **Green**: Procedural data reference connection
- **Blue**: Declared data reference connection
- **Dotted Line**: Unrealized connection
- **Solid Line**: Realized connection

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

You can also interact directly with the parts that are represented within a diagram. For example, you can invoke the JSP editor directly from the diagram editor.

13.6.2 Representing an existing Web interaction in the Web diagram
If you previously created a Web interaction in your Web project, you can use a Web diagram to represent that interaction and invoke the Web Interaction wizard to update the interaction. First, you must create a new Web diagram in your Web project.

Creating an empty Web diagram
Perform the following steps to create the Web diagram. Note, however, that with WDSc 5.1.2 or later an empty Web diagram called diagram.gph is created automatically when you create a dynamic Web project with struts support.

1. In the Project Navigator view, right-click the project name and select New → Other.
2. In the left pane of the wizard, expand Web and click Struts.
3. Select Web Diagram in the right pane and click Next.
4. Enter a name, say cust_inq for the diagram in the File name field (the GPH file extension is automatically applied to the name). Click Finish to create the file in your project. The file opens on a completely empty FFS in the editor view.

**Adding Web pages and interactions to the diagram**

Perform these steps to add the Web pages and interactions:

1. In the Project Navigator view, drag and drop the JSP files that are used as the input (custinq) and output (cussfl) pages of the interaction onto the FFS of the Web diagram. Since the JSP files exist in the Web project, they appear in the diagram as realized. That is, the icons appear in color.

![Figure 13-67  Creating a new Action Mapping Node](image1)

2. Right-click the free-form surface and select New → Action Mapping Node, as shown in Figure 13-67. Notice how the mouse pointer changes its shape as shown in Figure 13-68.

![Figure 13-68  Positioning the mouse to create a new Action Mapping Node](image2)
3. Move the mouse pointer where you want the icon for the new Action Mapping Node to be placed, for example, between the input page and the output page as in Figure 13-68. Then click the free-form surface to create an action icon as shown in Figure 13-69 on page 516.

4. Enter the name of the Web interaction by replacing the text “/action” in the highlighted area under the new icon as in Figure 13-69 and press Enter. For example, the Web interaction is represented by a WIT file in your Web project. If the Web interaction file is getList.wit, enter getList for the name of the action icon.

Since the interaction represented by the action icon already exists in the Web project, the action icon is realized. This is indicated by the fact that the originally black and white icon turns green immediately after you enter the correct name of an existing interaction.

5. Right-click the JSP node that represents the input page (cusinq.jsp) and select Connection. Place the mouse pointer on the action icon and click to draw the connection line (Figure 13-70).

6. Right-click the action icon and select Connection. Place the mouse pointer on the JSP icon that represents the output page (cussfl.jsp) and click to draw a connection line.

7. To invoke the Web Interaction wizard, right-click the action icon and select Open iSeries Web Interaction wizard. The wizard pages are updated with the data in the Web diagram and the fields and controls within the form in the JSP files.

8. As you progress through the pages of the wizard, update the options used in the interaction. Then click Finish to regenerate the Web interaction.

Repeat these steps for all Web pages and interactions that you want to appear on this diagram. This can be all pages and interactions or just a subset, depending on the size and complexity of your project. However, if the project is not too big, you can use an automated process generate the connections, as described in the following section.

**Allowing the diagram to draw connections automatically**

Instead of dragging all pages to the FFS, creating the Action Mapping Nodes, and drawing each connection between them, you can automate the process with the draw action. By using the draw all action, you can draw the entire Struts flow of a Web application if you start with the starting point of the application, which is the main menu page as described in the following steps:
1. In the Project Navigator view, drag and drop the JSP file, which serve as the starting point for the entire application, onto the FFS of the Web diagram. In our sample application, this is the cusinq.jsp file.

2. Right-click the `cusinq.jsp` icon and select Draw → Draw All from the context menu. Starting from this input page, all realized connections and nodes are drawn as shown in Figure 13-71.

![Figure 13-71 Drawing all connections](image)

As you can see in Figure 13-71, some of the connections overlay each other. This may prevent you from clearly understanding the structure of the application. The strength of the FFS is the possibility to move the nodes over the surface, while the connections follow the movement maintain the relationship between the nodes. You can use that to rearrange the diagram.

3. Drag the `cussfl.jsp` and the `displaycustdetailResults.jsp` files so that all three icons representing the Web pages are arranged in a vertical line on the left of the diagram as shown in Figure 13-72.

![Figure 13-72 Rearranging the Web pages on the FFS](image)

4. In a similar fashion, move the two Action Mapping Nodes to the center and the Form Bean Nodes to the right of the diagram as shown in Figure 13-73.
If you move the cussfl.jsp node a little to the right, as shown in Figure 13-74, you have a diagram with no overlapping or hidden connections.

The diagram in Figure 13-74 shows the relationship between all major components of your Web application at a glance. You can use the diagram as a starting point to do development work for your application.

**Using a Web diagram to extend an existing application**

A useful feature of the Web diagram editor is the ability to reflect changes of underlying resources in real time. For example, on the editor's FFS, you can create a node for a Web page named MyJSP.jsp that does not exist yet. A node that appears on the FFS before the resource that it represents exists is called *unrealized*.

If you create the JSP outside of the diagram editor, the diagram editor is informed of the JSP creation and reflects the change in its diagram. Similarly, if a realized part shown in the diagram has its underlying resource deleted, the diagram editor is informed of this deletion and reflects the change in the diagram.

You can also interact directly with the parts that are represented within a diagram. For example, you can double-click the icon to invoke the JSP editor directly from the diagram editor.

You can open the iSeries interaction wizard by right-clicking an Action Mapping Node and selecting Open iSeries Web Interaction wizard from the context menu.
Creating new components
To create a new Web page, interaction, Java Bean, or other elements, right-click the surface and select New followed by the appropriate option as shown in Figure 13-75. An unrealized node is placed on the surface. You can define the connections to the existing nodes that you are planning to implement as described in step 5 on page 516.

![Figure 13-75 Creating new diagram components](image)

To demonstrate this, we add another interaction to our example application with the purpose of formally ending the application.

Ending a Web application with a ‘No program call’ Web interaction
You can create a Web interaction that does not require communication with an ILE program or JavaBean. This type of interaction can be used to end a Web application. For example, after a user completes a business process through the Web interface, the last operation may be to log out of the application.

As a result, when you select No program call in the Web Interaction wizard, you do not see the wizard page used to define input and output parameters.

No program call: The No program call option is only available if you are defining an interaction that uses input and output pages that are already created. If you select either Generate input JSP or Generate output JSP on the second page of the wizard, Specify the Input and Output Pages for your Web Interaction, then this option is not available.

In this example, we use the Web diagram to add a No program call Web interaction to the sample application we described in the previous sections. The purpose of this interaction is to end the session in a controlled manner. When a session ends, the associated server job (QZQCSRVS running under the QUSER user profile) is also ended. The host program states are automatically reset. You do this by selecting the Invalidate session after interaction occurs option on the Specify a Name and Location for your Web Interaction window as shown in Figure 13-45 on page 486 and Figure 13-79 on page 523.

As you can easily see in Figure 13-74, displaycustdetailResults.jsp serves only as an output page and has links to the other two pages of the application to restart the inquiry. Currently it does not serve as an input page to any Web interaction. We want to add a Logoff button connecting to a No program call Web interaction, which invalidates the session and sends an output page without involving any OS/400 program.

Adding a Logoff button to the results page
All your development can now be started from the Web diagram you created in the previous steps. To add the required button to the displaycustdetailResults.jsp page, you can use Page Designer. To launch it, perform the following steps:

1. Double-click the icon for the displaycustdetailResults page on the FFS. Page Designer opens in the same pane of your workbench.
2. Make sure that you see the Design view. In case the Preview or Source view opened, click the Design tab on the right below the pane.

3. Right-click in the area just above the table. Select Insert → Form and Input Fields → Form, or press Ctrl+O to insert a form in your page, as shown in Figure 13-22 on page 465.

4. Right-click within that form and select Insert → Paragraph → Normal from the context menu. This allows you to center the button on the page.

5. Right-click within that paragraph and select Attributes. The Attributes [P] view opens in the lower left pane.

6. In the Alignment field, select Center.

7. Expand the iSeries Web Components tab on the right panel.

8. Drag the Button component and drop it into the paragraph you created in step 3. The Attributes [iwcl:WButton] view should open in the lower left pane. If it does not, right-click the newly created button and select Attributes.

9. In the Attributes [iwcl:WButton] view, in the Name field, type Logoff. In the Label field, type Log off from Customer Inquiry. Select Submit for Button type as shown in Figure 13-76.

10. Save your result JSP. Press Ctrl+S or select File → Save displaycustdetailResults.jsp from the menu bar.

Creating a No program call in the Web Interaction
Now we develop the interaction to be called, when the users clicks the Logoff button:
1. Right-click the white space in the FFS and select **New → Action Mapping Node**. Note, how the mouse pointer changes its shape as shown in Figure 13-69 on page 516.

2. Move the mouse pointer where you want the icon for the new Action Mapping Node to be placed. For example, you may move it under the getDetail node and to the lower left of displaycustdetailResults.jsp. Click the free-form surface to create an action icon at that point.

3. Enter `nopgm` as the name of the Web interaction to replace the text “/action” in the highlighted area under the new icon and press Enter.

   At this point, you have created an icon that only represents a Web interaction that still needs to be created. You have yet to create the Web interaction itself. The Web diagram indicates this by showing the icon in the color gray, the default for **unrealized nodes**.

   Before you actually create the interaction using the iSeries Web Interaction wizard, you should first create the output page and the connections between all tree components.

### Creating the `endResults.jsp`

You also need to create an output page to be displayed as a confirmation that the user has closed this application:

1. Right-click the white space in the FFS and select **New → Web Page**. Give it the name `endResults.jsp`.

2. Double-click the new **unrealized** Web page node. The New JSP File wizard opens as shown in Figure 13-77.

![New JSP File wizard](image)

   **Figure 13-77** New JSP File wizard

3. Select the option **Create from page template**.

4. Leave the settings as they appear. Click **Finish**.

5. The new JSP is created and opens in the Page Designer. Replace the text “Place content here” with “You are now logged off from the Customer Inquiry program.”
6. Select **Heading 2** and **Center** in the Attributes [P] view.

7. Save your result JSP. Press Ctrl+S or select **File → Save endResults.jsp**.

8. Close the Page Designer and return to the Web diagram.

The new JSP is now recognized as *realized*. Therefore the icon shows multiple colors. Next you must define the connections between the new nodes:

1. Right-click the JSP node that represents the displaycustdetailResults.jsp input page and select **Connection**. Place the mouse pointer on the **nopgm** action icon and click to draw the connection line (see Figure 13-78).

2. Right-click the **nopgm** action icon and select **Connection**. Place the mouse pointer on the JSP icon that represents the endResults.jsp output page and click to draw a connection line as shown in Figure 13-78.

![Figure 13-78 Creating a connection between Web page and action icon](image)

**Realizing the Web interaction using the wizard**

At this point, you have the input and output pages. However, the Web interaction is still not realized and the connection shows up as dotted lines indicating they are not realized yet. The nodes and connections on the Web diagram can help the Web Interaction wizard to create all necessary objects:

1. To invoke the Web Interaction wizard, right-click the action icon and select **Open iSeries Web Interaction** wizard. The wizard pages are updated with the data in the Web diagram and the fields and controls within the form in the JSP files.

2. Since this interaction should terminate the entire session, deselect the **Use error page** and select **Invalidate session after interaction occurs** on the Specify a Name and Location for your Web Interaction window as shown in Figure 13-79.
3. Click **Next** twice.

4. On the No Program Call panel (Figure 13-80), select **No program call**.

5. Click **Finish** to generate the Web interaction, since there are no parameter mappings to be defined in the next pages.

After the wizard finishes, you can see, in the Web diagram in Figure 13-81, that all components are now changed to the **realized** state. The new interaction is ready to be tested as explained in 13.4.5, “Testing the application in Development Studio Client” on page 502.
Using a Web diagram to create new applications

You can create a Web diagram based on an existing Web application. You can also build a new application from scratch by placing the nodes on the empty FFS as explained here:

1. Create a new Web project as explained in 13.2.1, “Creating a dynamic Web project” on page 441.
2. Optionally create a page template as explained in “Creating a page template” on page 454.
3. Create an empty Web diagram as explained in “Creating an empty Web diagram” on page 514.
4. Create the Web pages and interaction. Right-click the surface and select New followed by the appropriate option as shown in Figure 13-75 on page 519. Enter the same names for the Web pages and interactions as shown in Figure 13-81.
5. Create the connections between the nodes. Right-click the originating node and select Connection. Place the mouse pointer on the target icon and click to draw the connection line. You draw the following connection lines:

   - From cusinq.jsp to getList
   - From getList to cusssfl.jsp
   - From cusssfl.jsp to getDetail
   - From getDetail to detailResults.jsp
   - From detailResults.jsp to noPgm
   - From noPgm to endResults.jsp

The result should look similar to Figure 13-82.
Now that you have laid out the structure of the entire application, you can use the interaction wizard to generate the Web interactions and Web pages with a basic layout. After that you, can use Page Designer to finish the visual and functional appearance of the JSPs.

13.7 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 an EAR file, which is a packaged Web application.

Application assembly and J2EE applications

Application assembly is the process of creating an EAR file that contains all files related to an application and an XML deployment descriptor for the application. This configuration and packaging prepares the application for deployment onto an application server.

EAR files are comprised of the following archives:

- Enterprise bean (JAR) files (known as EJB modules)
- Web application (WAR) files (known as Web modules)
- Application client (JAR) files (known as client modules)
- Resource adapter (RAR) files (known as resource adapter modules)

Ensure that modules are included in an EAR file so that they may be deployed onto the server. The exceptions are WAR modules, which you can deploy individually. Although WAR modules can contain regular JAR files, they cannot contain the other module types described previously.

For detailed instructions about exporting a Web application from WebSphere Development Studio Client for iSeries and deploying it to WebSphere Application Server, see 10.6, “Running a WebFacing application in WebSphere” on page 289.
13.8 Subfile APIs

You can use the Table Web component to display database records similar to the way in which they are displayed in a subfile on the host. Prior to WebSphere Development Studio Client for iSeries V5, the subfile DTC was used to implement a similar function.

The OS/400 program specified in Figure 13-39 on page 480 interacts with the Table Web component 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 executes the ILE program.

Subfile API: Although the subfile DTC was functionally replaced in WebSphere Development Studio Client for iSeries V5 by the Table Web component, the corresponding APIs are still called a subfile API.

Specifying the name of the service program called by the table

When you create a Table Web component in one of your JSP files, one of the attributes you set is the name of the service program that the table calls. This program 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.

To specify the name and location of the service program on the host, select the Get data from an iSeries service program check box as shown in Figure 13-39 on page 480. Then in the Library and Object fields, enter the library and program name respectively.

Code procedures in the service program to interact with subfile APIs

The service program should provide several procedures that are called by the Table Web component.

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 Table Web component as defined in the Table Web component attributes. The second parameter is a blank delimited list of the exit parameters that were defined on the Table Web component attributes. The list of procedures includes:

- **INIT**: This procedure is called the first time the JSP page that contains the Table Web component is about to be displayed. You can use this procedure to initially fill the table with all records or a page worth of records.

- **PGUP**: This procedure is called when the user clicks the Page Up button associated with the table 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 clicks the Page Down button associated with the table to view the next page of records. If the next page of records exists 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.

Using the appropriate subfile APIs to manipulate the table data

Subfile APIs in your ILE service program manipulate the records in the user space object that are displayed in the Table Web component on a JSP page.
The subfile APIs that are available for processing a Table Web components 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

**Relationship of page up and down buttons in the table to subfile APIs**

When you select the Get table data from an iSeries service program option on the Parameters tab in the table attributes view (as shown in Figure 13-39 on page 480), the radio buttons for service program behavior are available for selection.

The default option, *Writes entire subfile* (previously called *Reads all records from the database*) indicates that a footer section is displayed in the table for paging purposes. Page up and page down buttons appear in this footer. Paging information also appears in the form of page x of y, for example Page 5 of 10. This option also indicates that, regardless of the number of records that are read into the user space for display in the table, the records represent the entire set of records that are displayed in the table. If you use the INIT procedure to initially fill the table with all records, the PGUP and PGDN procedures are not required.

Conversely, the *Writes a page at a time* (previously called *Reads one page of records at a time from the database*) option indicates that a footer section is not displayed in the table for paging purposes. Page up and page down buttons are added separately to the bottom of the table. This option also indicates that the contents of the user space can be cleared, updated, appended to, and chained to, depending on the subfile APIs that are used in the service program. As a result, the total number of pages cannot be predetermined and the page x of y format is not provided. This option also requires that the INIT, PGUP, and PGDN procedures are coded in the service program to control paging in the Table Web component.

**Service program QDTSSFL**

The host service program QDTSSFL.SRVPGM is required when the user host program needs to invoke the WebSphere Development Studio Client for iSeries subfile API.

In previous releases, the subfile APIs save file was distributed as a part of the Development Studio Client and needed to be installed on your iSeries server. With in WDSC 5.1.2, you need apply PTF SI09791 in OS/400 V5R1, SI09790 in V5R2. No special action must be taken for i5/OS V5R3, since this service program is included in library QSYS.

**Attention:** If you do install the qdtssfl.sav provided with previous releases of WDSc, some functions of the Web application may not work as intended. For example, Subfile APIs cannot be use to send data to the table Web component and it displays an empty table.

**Development environment for QDTSSFL**

Check the following conditions on the iSeries server before you run QDTSSFL:

- Ensure that the QSYS library is in the library list.
- Bind the user program that makes the subfile API call to QSYS\QDTSSFL.SRVPGM.

For example use the Create RPG Module (CRTRPGMOD) command to create an RPG module:

```
CRTRPGMOD MODULE(GETDETAIL)
```
Then use Create Service Program (CRTSRVPGM) command to create this as a service program. It must be bound with service program QDTSSFL which performs the actual subfile operations:

CRTPGM PGM(GETDETAIL) BNDSRVPGM(QGPL/QDTSSFL)

Guide to Online Help: For detailed information about this topic, open the online help function by performing the following steps:

1. Select Help → Help contents in the workbench.
2. Click Reference.
3. Expand User Interface Reference.
4. Expand iSeries Web applications.
5. Expand Creating iSeries Web components.
7. Click Subfile APIs, or
8. Click Subfile example for an RPG example.
9. Click COBOL program example using subfile APIs for an Cobol example.

Subfile API operations and names
The following sections describe each of the subfile APIs. They include the syntax, parameters, program prototype, and an example for each subfile API.

Adding a record to the subfile: QdtsAppendSF
Use this API to add a record to the subfile. Records are added to the end of the subfile.

The syntax is:

```
CL0N01Factor1+++++++Opcode&ExtExtended-factor2++++++++++++++++++++++++++++++
Eval   uRC=QdtsAppendSF(spaceName: pRcd: length)
```

The parameters are:
- uRC: 0=Success, -1=Fail
- spaceName: Address of the space name
- pRcd: Address of the subfile record to be added
- length: The length of the subfile record

The prototype is:

```
DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++++
D  append          PR        10U 0 Extproc(QdtsAppendSF)
D   spaceName 10A Const
D     pRcd                          *   Value
D     len                         10U 0 Value
```

Here is an example:

```
DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++++
D initSF          PR            10U 0 Extproc(QdtsAppendSF)
D  spaceName 10A Const
D     pRcd                          *   Value
D     len                         10U 0 Value
D spaceName       S                   Inz('SFL1')
D Record          DS
D  Name                         20A
D  City                         30A
D  uRC            S             10I 0
* 
CL0N01Factor1+++++++Opcode&ExtExtended-factor2+++++++++++++++++++++++++++++++```
Eval Name='Larry'
Eval City='Pickering'
CallP append(%Addr(SFName: %Addr(Record): %Size(Record))

Chaining to a specific subfile record: QdtsChainSF
Use this API to retrieve a specific record from the subfile by passing the relative record number of the record you want.

The syntax is:
CL0N01Factor1+++++++Opcode&ExtExtended-factor2+++++++++++++++++++++++++++++
Eval uRC=QdtsChainSF(spaceName: pRcd: length, uRRN)

The parameters are:
- uRC: 0=Success, -1=Record not found
- spaceName: Address of the space name
- pRcd: Address of the structure to receive the subfile record
- length: The length of the subfile record
- uRRN: The relative record number of the subfile record to chain to (read)

The prototype is:
DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++
D chain           PR        10U 0 Extproc(QdtsChainSF)
D  spaceName 10A Const
D   pRcd                          *   Value
D   len                         10U 0 Value
D   rrn                         10I 0 Value

Here is an example:
DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++
D chain           PR            10U 0 Extproc(QdtsChainSF)
D  spaceName 10A Const
D   pRcd                          *   Value
D   len                         10U 0 Value
D SFName       S                   Inz('SFL1')
D Record          DS
D  Name                         20A
D  City                         30A
D uRC             S             10I 0
D uRRN            S             10I 0
*
CL0N01Factor1+++++++Opcode&ExtExtended-factor2+++++++++++++++++++++++++++++
Eval uRRN = 3
Eval uRC=chain(%Addr(SFName: %Addr(Record): %Size(Record): uRRN)

* Name = Donna (possibly)
* City = Guelph (possibly)
* uRRC = 3 (possibly)

Clearing the subfile: QdtsClearSF
Use this API to remove all records from the subfile. You can call this API if you are filling the subfile one page at a time.

The syntax is:
CL0N01Factor1+++++++Opcode&ExtExtended-factor2+++++++++++++++++++++++++++++
Eval uRC=QdtsClearSF(spaceName)

The parameters are:
- uRC: 0=Success, -1=Fail
- spaceName: Address of the space name

The prototype is:

DName++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++++ D clear PR 10U 0 Extproc(QdtsClearSF) D spaceName PR 10A Const D uRC S 10I 0 D SFName S 10A Inz('SFL1') *

CL0N01Factor1+++++++Opcode&ExtExtended-factor2+++++++++++++++++++++++++++ Eval uRC=clear(%Addr(SFName))

Initializing the subfile: QdtsInitSF

Use this API to initialize the subfile for first time use. The call to this API is typically placed in the INIT procedure that is called the first time that the JSP containing the subfile is to be displayed.

The syntax is:

CL0N01Factor1+++++++Opcode&ExtExtended-factor2+++++++++++++++++++++++++++ Eval uRC=QdtsInitSF(spaceName: pReserved: length)

The parameters are:
- spaceName: Address of the space name
- pReserved: Address of any 10 character string (unused by the API but must be coded)
- length

The prototype is:

DName++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++++ D initSF PR 10U 0 Extproc(QdtsInitSF) D spaceName PR 10A Const D pReserved * Value D len 10U 0 Value

Here is an example:

DName++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++++ D initSF PR 10U 0 Extproc(QdtsInitSF) D spaceName PR 10A Const D pReserved * Value D len 10U 0 Value D spaceName S Inz('SFL1') D rsvd S 10A D len S 10U 0 D uRC S 10I 0 *

CL0N01Factor1+++++++Opcode&ExtExtended-factor2+++++++++++++++++++++++++++ Eval uRC=initSF(spaceName: %Addr(rsvd): len)

Reading a selected record: QdtsReadcSF

Use this API to read a record that was selected in the subfile. Records in a subfile are selected by selecting the radio button or check box that is associated with each subfile record.
The syntax is:

```
CL0N01Factor1+++++++Opcode&ExtExtended-factor2+++++++++++++++++++++++++++++
    Eval uRRN=QdtsReadc(spaceName: pRcd: length: ustartRRN)
```

The parameters are:

- **uRRN**: The relative record number of the selected subfile record
- **spaceName**: Address of the space name
- **pRcd**: Address of the structure to receive the subfile record
- **length**: The length of the subfile record
- **ustartRRN**: The record from which to start searching

The prototype is:

```
DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++
D readc           PR            10U 0 Extproc(QdtsReadcSF)
D  spaceName 10A Const
D   pRcd                          *   Value
D   len                         10U 0 Value
```

Here is an example:

```
DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++
D readc           PR            10U 0 Extproc(QdtsReadcSF)
D  spaceName 10A Const
D   pRcd                          *   Value
D   len                         10U 0 Value
D SFName       S                   Inz('SFL1')
D Record          DS
D  Name                         20A
D  City                         30A
D urrn            S             10I 0
D uRRN            S             10I 0
```

```
* CL0N01Factor1+++++++Opcode&ExtExtended-factor2+++++++++++++++++++++++++++++
    Eval uRRN=readc(%Addr(SFName:%Addr(Record):%Size(Record):)
* Name = Donna (possibly)
* City = Guelph (possibly)
* uRRN = 3 (possibly)
```

**Updating a subfile record: QdtsUpdateSF**

Use this API to update an existing record in the subfile. It is not necessary to read the record before you update it.

The syntax is:

```
CL0N01Factor1+++++++Opcode&ExtExtended-factor2+++++++++++++++++++++++++++++
    Eval uRC=QdtsUpdateSF(spaceName: pRcd: length, uRRN)
```

The parameters are:

- **uRC**: 0=Success, -1=Record not found
- **spaceName**: Address of the space name
- **pRcd**: Address of the record structure
- **length**: The length of the subfile record
- **uRRN**: The relative record number of the subfile record to update

The prototype is:

```
DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++
D chain           PR            10U 0 Extproc(QdtsUpdateSF)
D  spaceName 10A Const
```
Here is an example:

```
DName+++ETDsFrom+++To/L+++IDc.Keywords+++D  
D  chain           PR            Extproc(QdtsUpdateSF)
D  spaceName 10A Const
D  pRcd                          
D  len                         
D  rrn                         
  
Eval   Name = 'Donna'
Eval   City = 'Guelph'
Eval   uRRN = 3
Eval   uRC=chain(%Addr(SFName: %Addr(Record): %Size(Record): uRN)
```

### 13.8.1 Example code

The base code we use is defined in 4.2, “Overview of the Customer Master Inquiry” on page 68. The source code is included in additional material in the library LIB6961 source file WEBCOMP01. We cannot use native 5250 RPG code. Only the main calculations and file definitions can be used in Web application. All display input and output operations are handled by the subfile API.

**CUSINQNEW Customer Inquiry ILE RPG program**

This program is called from Cusinq.jsp. By calling the CUSTINQSFL service program, you set the initial ZIP value. You also create the subfile user space in library QTEMP and fill the subfile with all records from the database. After compiling this program, you must bind it to the CUSTINQSFL service program using the Create Program (CRTPGM) command.

**Example 13-2  CUSINQNEW**

```
****************************************************************
*                                                              *
* CUSINQNEW                                                    *
* This program will be called by the CUSINQ.jsp page           *
* before the CUSSFL.jsp page with the Subfile DTC is           *
* displayed.                                                   *
* Or by VARP program REDBOOK1 *                                *
* By calling the CUSTINQSFL service program, this program will* *
* set the initial zip value as well as create the subfile      *
* user space and fill the subfile with all records from        *
* the database.                                               *
* After compiling this program it must be bound to the          *
* CUSTINQSFL service program using the CRTPGM command.         *
****************************************************************
*                                                              *
* Define prototypes for procedures
```
CUSTINQSFL Get Customer List ILE RPG service program

Example 13-3 shows the source of Get Customer List service program CUSTINQSFL. 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.

Example 13-3 CUSTINQSFL

```
H NoMain

*---------------------------------------------------------------
* Define the logical file (view by ZIP )
FMLGMSTL1 IF E K Disk
*---------------------------------------------------------------
* Define the prototypes for the procedures in the subfile
* handling service program, QDTSSFL
D/copy LIB6961/WEBCOMPO1,SUBFILEPR2
*---------------------------------------------------------------
* Define the prototypes for the procedures in this program
D/COPY LIB6961/WEBCOMPO1,R6961PROTO
*---------------------------------------------------------------
* Define other program variables
D rc S 10I 0
D count S 10U 0
D len S 10U 0 Inz(%Size(Record))
*---------------------------------------------------------------
* Structure to define a subfile record
D Record DS
```
D ZIP 55 0
D CUST 55 0
D NAME 20A
 *
D Reserved 5 10A
D
 *
*------------------------------------------------------------------------*
* * 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 *
* *
*------------------------------------------------------------------------*
P FillSFL B EXPORT
D FillSFL PI
D Subfile 10A
D Zips 55 0
 *
C  Eval Count = 1
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(Subfile : %ADDR(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 Table Web component *
* 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. *
* *
*------------------------------------------------------------------------*
P PGUP B Export
D PGUP PI
D Subfile 10A
D Zips 55 0
 *
C  Return
 *
P PGUP E
Handle the page down request.

This procedure will be called by the Table Web Component 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

-----------------------------------------------------------

P PGDN            B                   Export
D PGDN            PI
D Subfile                      10A
D Zips                          5S 0
*  
C                   Return
  
P PGDN            E

-----------------------------------------------------------

INITFirst

This procedure is called by CUSINQNEW to initialize the subfile, and fill it with records

Parameters:

Subfile - the subfile (space) name (SFL1 in this case)
Zips - additional parameters as specified on the subfile DTC properties page. In this program the component name is passed.

-----------------------------------------------------------

P INITFirst       B                   Export
D INITFirst       PI
D Subfile                      10A
D Zips                          5S 0
D Setup                         1
*  
C                   If        setup = *off
C                   Eval      setup = *on
C                   Eval      len = %Size(Record)
C                   Eval      rc = initSF(Subfile: %ADDR(Reserved) :len)
C                   EndIf
C                   Eval      rc = clearSF(Subfile)
C                   CallP     FillSFL(Subfile:Zips)
C                   Return
  
P INITFirst       E
GETDETAIL

Example 13-4 shows the source code of the sample program GETDETAIL. It illustrates how to perform a READC on an Table Web component. The user may choose a subfile record by selecting a check box (multiple select), or a radio button (single select) in a subfile record. Then, the readcSF procedure returns the relative record number (RRN) of the selected record and 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 13-4  GETDETAIL

******************************************************************************
*                                                                     *
* GETDETAIL                                                         *
*                                                                     *
* Sample program to illustrate how to do a READC on an Table Web*     *
* Component                                                          *
*                                                                     *
* If the user has selected a subfile record by clicking on a         *
* check box (multiple select), or a radio button (single select)*    *
* in a subfile record, the readcSF procedure will return the         *
* RRN of the selected record as well as the field data.              *
*                                                                     *
* If no record has been selected, the procedure returns a 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                                                   *
*                                                                     *
* Parameters:                                                     *
*    Subfile   - the subfile name (SFL1)                           *
*    others    - problem info.                                      *
*                                                                     *
******************************************************************************

* Define the 'by CUST' logical file
FMLGMSTL2 IF E           K Disk

* Prototype the readcSF procedure in the C library
/copy LIB6961/WEBCOMPO1,SUBFILEPR2

D*readcSF PR 10U 0 ExtProc('QdtsReadcSF')
D* Subfile * 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)

* Working variables
DSsubfile S 10 Inz('SFL1')
Drc S 10I 0
DstartRRN S 10U 0
Dsize S 10U 0 Inz(%Size(Record))

* Define I/O parameters
Chapter 13. Building Web interactions with iSeries Web Tools

 DDS source code

We include the source for DDS files used in the sample applications (see Example 13-5, Example 13-6, and Example 13-7).

Example 13-5  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
### Example 13-6  MLGMSTL1

```
A**************************************************************************
A* FILE NAME  : MLGMSTL1                                            *
A* DESCRIPTION: LOGICAL VIEW OF CUSTOMER MASTER FILE (CUSMSTP)      *
A* BY ZIP CODE (ZIP) AND NAME (NAME)                               *
A**************************************************************************
A          R CUSREC                    PFILE(CUSMSTP)
A          K ZIP
A          K NAME
```

### Example 13-7  MLGMSTL2

```
A**************************************************************************
A* FILE NAME  : MLGMSTL1                                            *
A* DESCRIPTION: LOGICAL VIEW OF CUSTOMER MASTER FILE (CUSMSTP)      *
A* BY ZIP CODE (ZIP) AND NAME (NAME)                               *
A**************************************************************************
A          R CUSREC                    PFILE(CUSMSTP)
A          K CUST
```

### 13.9 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 JavaBeans and PCML files to call your iSeries programs or service programs. You can also use the JavaBeans 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. Then it creates the desired JavaBeans. 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 JavaBeans 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. Then, the wizard can generate JavaBeans from it.

#### 13.9.1 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 13.2.1, “Creating a dynamic Web project” on page 441. To launch the Program Call wizard, perform the following steps:
1. Select the project **MyProject** as shown in Figure 13-83.

![Figure 13-83   Selecting the project](image)

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 13-84), expand **iSeries** and select **Java**. In the right pane, select **Program Call Bean** and click **Next**.

![Figure 13-84   Selecting the Program Call bean](image)

4. The Program Call wizard launches. To define a program or service program procedure that you want to call, click **Add Program**.

5. On the Program Call wizard window (Figure 13-85), follow these steps:
   a. Enter the JavaBean name.
   b. Click **Browse** to select the program object. After you select a program to call, the Library field is filled automatically.
   c. Select the Source location. By selecting the type of source file, a new window opens for you to select the source file. Click **OK**.
In the left panel, the new program call definition is added as shown in Figure 13-85.

d. 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. Add the required parameters for the program.

**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.

e. Click Next.

![Figure 13-85  Program call definition](image1)

6. In the next window, you can specify the location where the Java Bean is generated. Click Finish. In the Navigator window (Figure 13-86), you can see the Java objects that are created (expand the project and source folders).

![Figure 13-86  Program beans created](image2)
13.9.2 Calling an iSeries program or procedure

The Program Call wizard generates JavaBeans and a 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 JavaBeans:

- One type for remote calls to the specified iSeries program or procedure as explained in 13.9.1, “Creating Program Call beans” on page 538
- 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 JavaBeans 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 JavaBeans 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.

**Note:** IBM Toolbox for Java is a set of Java classes that allow you to use Java programs to access OS/400 objects. 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).
Testing and debugging

Several tools are available to debug an application in WebSphere Development Studio Client for iSeries. Part 4 discusses the available tools and techniques for debugging an application.
Debugging and testing options

This chapter contains information about how to debug and test your applications using the various debugging options and profiling tools in WebSphere Development Studio Client for iSeries.
14.1 Available debuggers

IBM WebSphere Development Studio Client for iSeries offers the following debuggers:

- The Integrated iSeries debugger
- The Java debugger for iSeries
- The WebSphere Application Server debug adapter
- The IBM Distributed Debugger
- The Compiled language debugger
- The JavaScript debug adapter

The Integrated iSeries debugger

The Integrated iSeries debugger helps you to find errors in code that is developed using OS/400 or i5/OS Integrated Language Environment (ILE) or Original Program Model (OPM) languages. This debugger enables you to debug applications that are running on the iSeries host while your interface is running in the workbench at your workstation.

The Integrated iSeries debugger supports any program written in the following languages:

- ILE RPG
- ILE COBOL
- ILE CL
- C
- C++
- RPG
- COBOL
- CL
- DB2 and SQL stored procedures

You use this debugger, in most cases, during iSeries application development.

The Java debugger for iSeries

The Java debugger for iSeries enables you to detect and diagnose errors in Java applications that are running either locally on your workstation or remotely on the iSeries.

The WebSphere Application Server debug adapter

This tool allows you to debug Web objects that are running in WebSphere Application Server. These objects include Enterprise JavaBeans (EJBs), JavaServer Pages (JSPs), and servlets. The debug adapter is supported for WebSphere Application Server versions 4.02 and later. However, it only supports step-by-step debugging for WebSphere Application Server versions 5.0 and later.

The IBM Distributed Debugger

Use this tool when application development is done with CODE.

The Compiled language debugger

You can use this debugger when you develop compiled language application for workstations.

The JavaScript debug adapter

The WebSphere Application Server debug adapter also allows you to debug server-side JavaScript.
14.2 Prerequisites for debugging iSeries applications

Before you start debugging an iSeries application, you need to perform some or all of the following tasks:

1. Compile a program with debug data.
2. Change the debugger settings.
3. Ensure that your iSeries user profile has the appropriate authorities.
4. Start a debug session using one of the debugging methods: one-step or full prompt.

14.2.1 Compile options

Programs must be compiled with appropriate debug information before compiled program objects can be used with the debugger. For example, ILE RPG, COBOL, and CL programs must be compiled with the *SOURCE, *LIST, *STMT, or *ALL option. The default is *STMT (Statement view) when compiling from the iSeries display. The default is *SOURCE when compiling in WebSphere Development Studio Client for iSeries (WDSc). For more information about compiling each language, see your compiler documentation.

14.2.2 Setting debug preferences

Before using debugging functions, you should set the global debug preferences:

1. From the workspace toolbar, select Window → Preferences.
2. On the Preferences window (Figure 14-1), expand the Debug tree.

The first window is for general debugging settings:

- **Build before launching:** If the workspace requires building, an incremental build is performed prior to launching an application. This is mainly for Java debugging.
- **Remove terminated launches when a new launch is created:** If selected, all terminated applications in the Debug view are automatically cleared.
- **Reuse editor when displaying source code:** When this option is on, the debugger reuses the editor that it opened to display source from different source files. This prevents the debugger from opening an excessive number of editors. When this option is turned off, the debugger opens a new editor for each source file that must be displayed.
- **Save dirty editors before launching:** This option controls whether to prompt the user to save any dirty editors before an application is launched.
- **Default perspective for debug:** These settings controls the perspective which is opened when an application is launched in debug mode.
- **Default perspective for Run:** This setting controls the perspective that is switched to when an application is launched in run mode. By default, each launch configuration inherits this setting. However, a launch configuration may also override this setting by specifying an explicit perspective in the Common tab.

Other debugging preferences are:

- Console
- Debug Active Groups
- Debug Daemon
- iSeries Debug
- Launch History
- SQL Stored Procedure Debugger
- Variable Views
From these settings, iSeries Debug is the most important (Figure 14-2).
When you use the one-step debugging method, the debug session is launched by default with the Update production Files check box not selected. When you select this option in the debug preferences, the debugging session can update files. When using the full prompt debugging method, you can select whether the option is on or off.

If your workstation has multiple IP addresses and you still want to use one-step debugging, select the Specify host name of your workstation option. Then in the Host name of your workstation field, specify your workstation’s host name or IP address.

### 14.2.3 Setting breakpoints prior to debugging

You can set breakpoints before you start debugging by setting a source breakpoint in the editor or by setting a Service Entry Point (SEP).

**Set a Source Breakpoint**

To set the source breakpoint perform the following steps:

1. Open the source code.
2. Right-click the line number area on the left side of the editor and select Add Breakpoint (Figure 14-3).

**Note:** You can also double-click in the prefix area of an executable line to set a breakpoint.

![Figure 14-3 Adding a breakpoint](image)
3. The Breakpoint definition window opens (Figure 14-4). To set the breakpoint, the debugger needs to know to which program the source belongs. The Breakpoint window collects that information.
   a. Enter the library name. You can also select it from the pull-down list, or browse for it by clicking Browse next to the Library drop-down list.
   b. Select the type of object, Program or Service program. You can enter the program or service program name, select it from the pull-down list, or browse for it. The module name is only required if it is different from the program/service program name.
   c. Click OK. A breakpoint marker appears in the prefix area of the line in the editor (Figure 14-5). If there is already an active debug session, the breakpoint is added to the debugger's Breakpoints view immediately and there is no marker in the editor.

![Figure 14-4 Breakpoint definition](image)

After you have an active debug session, you can modify breakpoints that are created before debugging from the Breakpoints view. You do this in the same manner as line breakpoints that are created during a debug session. See Figure 14-5.

![Figure 14-5 Breakpoint marker](image)

**Note:** You see the window the first time you set a breakpoint in the editor, with or without a debug session. If a debug session is active, you don’t get the breakpoint marker in the editor. The breakpoint is set directly in the debugger instead. In this case, you get the Breakpoint window each time you set a breakpoint.

**Service Entry Point (SEP)**

The SEP is a breakpoint that is triggered when the first line of a specified program or procedure is executed for a specific user in a job that is not currently under debug.

To set a Service Entry Point:

1. In the Remote Systems view, select the program or service program that you want to debug.
2. Right-click and select the option Service Entry Point and select option Set.
Figure 14-6  Set a Service Entry Point

This sets a breakpoint in the program for the user profile in the RSE connection used for selecting the breakpoint. This breakpoint is triggered when the program is executed in a job running under this user profile.

You can use the **Show View** option under the **Service Entry Point** option to display the active Service Entry Points for your user profile on the iSeries server.

### 14.3 Integrated iSeries debugger

Using the Integrated iSeries debugger is the best way to debug an application in an iSeries environment. It is fully integrated into the WebSphere Development Studio Client for iSeries workbench. You can use it to debug code developed with OS/400 and i5/OS ILE and OPM languages.

There are two methods to start an Integrated iSeries debugging session:
- One-step debugging
- Debugging with full prompt

#### 14.3.1 One-step debugging

One-step debugging method offers a quick way to debug iSeries applications. To use one-step debugging, right-click the program object from the Remote System Explorer (RSE) perspective and choose **Debug As**. Select one of the actions from the pop-up menu:
- Batch
- Interactive
Multi-threaded

You can also select the object, click the down arrow of the debug button ( ), and select **Debug As**. You then see the list of debugging methods as shown in Figure 14-7 and can select the appropriate method:

- iSeries Batch Application
- iSeries Interactive Application
- iSeries Job
- iSeries Multi-Threaded Application

![Debug As window](image)

Figure 14-7 Debug As window

When performing one-step debugging, the debug session:

- Stops at the first debuggable line of the selected program.
- Ends when the program runs to completion.

**Note:** Selecting a *SRVPGM or any other object type in the Remote System Explorer results in a prompt.

When you use either of the debugging methods to debug applications, the session information is saved for later use.

**Debugging an interactive 5250 application**

When you start debugging interactive applications, you must start a 5250 emulator session. Within that session, start the IBM RSE Communications Server with following command:

```
STRRSESVR NAME(your-connection-name)
```

The parameter NAME is your connection name from the RSE. Note, that it is case sensitive and needs to be specified exactly as in your RSE perspective. You can also specify a working library, remote location name, and a TCP/IP port number, if you press F4=Prompt. See also “Interactive” on page 177.

**Note:** If you start debugging without starting the RSE communication server first, you are informed to start it as shown in the message in Figure 14-8.
Figure 14-8  RSE Communication Server not active

Start a debugging session in Development Studio Client. Select and right-click a program object from the RSE perspective. Select **Debug As → Interactive** (Figure 14-9). Alternatively, you can select a program object, click the down arrow of the debug button , and select **Debug As → iSeries Interactive Application** (see Figure 14-7).

Figure 14-9  Debug as Interactive

The debugging session starts by calling the selected object and steps into the first debuggable line. Now you can start debugging your program. For more information about debugging, see 14.3.3, “How to debug” on page 565.

**Debugging a batch application**

To use one-step debugging to debug a batch application, right-click a program object from the RSE perspective and choose **Debug As**. Select **Batch** from the pop-up menu. You can also select the object, click the down arrow of the debug button , and select **Debug As → iSeries Batch Applications**.

If you choose to debug the application, then WebSphere Development Studio Client launches the application by issuing the following command:

```
SBMJOB CMD(CALL PGM(ProgramName)) HOLD(*YES)
```
The debugger releases the batch job and the debug session begins (Figure 14-22 on page 566).

**Note:** Other parameters are default values of the command. A job is launched in the subsystem determined according to your user profile settings. If you want to use a different command or your program requires parameters, you have to use a full prompt.

**Debugging iSeries Job**

Debugging iSeries Job is useful when the user has a problem with the application. A common scenario for debugging a job is an interactive application that causes problems when you are working with a specific screen. You want to start debugging at that point, not from the beginning.

To do this, start the debugger when you are in the RSE. Right-click a job and select **Debug As → iSeries Job** (Figure 14-10).

![Figure 14-10 Debug iSeries Job](image)

When the debugger is ready, a message is displayed. It instructs you to click OK and start your application (Figure 14-11).

![Figure 14-11 iSeries Debug Message](image)
In our case, we debug the Order Entry application. Before we issue the Debug As command, the program waits for a user action with the 5250 screen (Figure 14-12).

![Order Entry application](image)

We press F4 to prompt for a customer. The debugging session steps into the program and stops at the next debuggable statement. The Debug perspective opens (Figure 14-13).

Proceed with debugging the application. For more information about how to set up a breakpoint and monitor variables, see 14.3.3, “How to debug” on page 565.
14.3.2 Debugging with full prompt

Debugging with full prompt allows you to create launch configurations before you start to debug an application. Launch configurations are sets of parameters that you define for your debugging environment. For example, you define what to debug, how to start the program, and so on.

To start creating launch configurations, click the down arrow of the debug button ( ) and then select **Debug**. The Launch Configuration window (Figure 14-14) opens. Another way is to select **Run → Debug** to open the Launch Configuration window.

To set up some configuration parameters, choose one of these options:

- Select the appropriate entry and click **New**.
- Right-click the entry and select **New** from the pop-up menu.
- Double-click the entry.

Alternatively you can select **Debug (Prompt)** from the pop-up menu of a program or service program in the RSE view.
Debugging an interactive 5250 application with full prompt

Start Launch Configurations from the pop-up menu in the RSE. Select **Debug (Prompt) → Interactive**, or select **Run → Debug** and the Launch Configurations window (Figure 14-15) opens.

![Figure 14-14  Launch Configurations window](image)
What To Debug page
You must set the following options for debugging:

- **Name**: Change the default name New configuration (n) to something more meaningful.
- **Connection**: Must be the Connection name that was used in the STRRSESVR command.
- **Library**: Enter your object library name.
- **Program or Service Program**: Select the type of program.
- **Program or Service program name**: Enter your program or service program name.
- **Step into**: If Step into check box is not selected, the debugger stops at the first encountered breakpoint. If you select this check box and you enter the program or service program name, the debugger steps into the specified program object. Then you can add breakpoints before your program actually starts.
- **Terminate debug session on program completion**: Select this option if you want the debug session to terminate when the program ends. This option is not available for service programs.
- **Update production files**: If this check box is selected, database files in production libraries are allowed to be updated.

How To Start page
On this page, you describe how to start the application (Figure 14-16):
1. Enter the command to start the application in the box. If you completed the What To Debug page, or if the debug session started from RSE or the iSeries Project perspective, the command is automatically created for you.

2. Prompt the command to get the correct parameters.

**Restriction:** You can’t use the Submit Job (SBMJOB) command to invoke your application. Use Debug Batch Application for launch configuration to start batch debugging. You can learn more about this in “Debugging a batch application with full prompt” on page 562.
**Advanced page**

On the Advanced page of the interactive launch configuration, specify the integrated file system (IFS) search path (Figure 14-17):

- **IFS source search path**: If your source code is placed into the IFS, enter the path where the source code can be found. When searching for source, the debugger looks first in your workbench workspace. If source cannot be found there, then the debugger searches the IFS search path that you set.

- **Automatically detect host name of your workstation**: The debugger should determine the TCP/IP host name of your workstation. If you have difficulties starting the debug session, select the Specify host name of your workstation option to define the host name.

- **Specify host name of your workstation**: If you select this option, in the Host name of your workstation field, specify your workstation IP address. This entry is required if your workstation has multiple IP addresses.

![Image of Advanced page of launch configurations](image-url)

*Figure 14-17  Interactive debug: Advanced page*
**CISCO VPN software:** If you are using CISCO VPN software, you need to supply your IP address manually in the iSeries debug Launch Configurations page. This applies when you debug your application as iSeries interactive, iSeries job, iSeries batch, or iSeries multithreaded. To specify your IP address, follow these steps:

1. Click the down arrow next to the debug icon and select **Debug**.
2. In the Launch Configurations window, select **iSeries: Debug Interactive Application**.
3. Click **New**.
4. Click the **Advanced** tab.
5. Under Workstation host name address, by default, the Automatically detect host name of your workstation options is selected. Instead, select the **Specify host name of your workstation** radio button.
6. Enter the IP address of your workstation.
7. Continue with the configuration as you normally do.

Set this entry in the **Preferences**, but change it when the address changes.

**Common page**
Click the Common tab to specify who can use the saved launch configurations (Figure 14-18). Note the following sections on this page:

- **Type of launch configuration:** If you want to store the launch configuration locally and have only one person use it, ensure that you select the **Local** radio button. If you specify to store the launch configuration as a file and have a team use it, select the **Shared** radio button. In **Location of shared configuration field**, enter the location of the file. You may also click **Browse** to find the location.
Interactive debug: Common

- **Perspective to switch to or open when launched**: Select the perspective that you want to open after the launch.
- **Display in favorites menu**: If you want the launch configuration to appear in the favorites menu, select the **Debug** check box. You can find the favorites menu under the **Debug** button or when you select **Debug → Debug History**.

When you finish the **Launch Configurations** settings, click **Apply** to save your configuration. To start debugging the interactive application, click the **Debug** button. Then the debugging perspective opens (Figure 14-22 on page 566).

**Debugging a batch application with full prompt**

Debugging with a full prompt uses **Launch Configurations** to specify the information required for your debug session. Launch configurations are saved for later use.

To create launch configurations, click the down arrow of the debug button and select **Debug** or select **Run → Debug**. The **Launch Configuration** window opens (Figure 14-14).

**Note**: If you used the debugger earlier, clicking the debug button causes the previous debug session to start.
To create a new configuration, choose one of the following options:

- Select the iSeries: Debug Batch Application entry and click the New button.
- Right-click the iSeries: Debug Batch Application entry and select New.
- Double-click the iSeries: Debug Batch Application entry.

Alternatively you can select Debug (Prompt) → Batch from the pop-up menu of a program or service program in the RSE view.

Enter the required information for iSeries Batch Debugging (Figure 14-20). The configuration names My iSeries program and My iSeries job are used by the debugger to create launch configurations for one-step debugging. Change the name if you want to save the Launch configuration for later use.
When using batch debugging, the command used to launch the application is the Submit Job (SBMJOB) command. The Command to start application entry field is prefilled when you complete the What To Debug page of Launch Configurations. You can edit the command to fit your environment. For example, you can enter a user command that submits the job to your default batch subsystem. To edit the command, type over it or click the Prompt button and enter your new parameters (Figure 14-21).

When you are finished with the prompt, click OK. The new parameters appear in the How To Start page. Click Apply to save the information and click Debug if you want to start debugging.

If you click Debug, your launch configuration is saved. You don’t have to click Apply first. You need to click Apply only if you want to create and save a launch configuration.
14.3.3 How to debug

When the debugging session is started by one-step or full prompt method, you operate the debugging actions in the Debug perspective. This is also the main window to see the results (Figure 14-22).
Debug perspective

The debug perspective contains several views that allow you to manage your debug session; view information about your programs, modules, procedures or functions; monitor variables, work with breakpoints; and view your source. The Debug view allows you to manage the debugging process with commands (Figure 14-23).
Commands can be activated from the Debug view toolbar, or by right-clicking, or by selecting Run from the perspective toolbar. The icons on the Debug view toolbar include:

- **Disconnect**: Disconnect from the remote target.
- **Relaunch**: Relaunch the selected debug target.
- **Remove All Terminated Launches**: Clear all terminated debug targets.
- **Resume**: Run the program to the next breakpoint.
- **Step Over**: The current statement is executed without stopping in any program or procedure called within the statement (unless a breakpoint is hit). The program stops at the next statement.
- **Step Into**: The current statement executes and the program stops at the next line encountered for which debug information is available.
- **Step Return**: To return to the previous call level.
- **Suspend**: Hold the job.
- **Terminate**: End the debugging session.
- **Terminate All**: End all debugging sessions.
- **Terminate and Remove**: End and delete the debugging session.

Under the Debug view pull-down menu (down arrow) you find this option:

- **Show Qualified Names**: Toggle this option to display or hide qualified names.

**Breakpoints**

iSeries Integrated Debugger has two kinds of breakpoints: Line and Watch. **Line breakpoints** are the most common way to stop a program before you execute a certain line of code and find out the field values in a current state. **Watch breakpoints** are useful when you want to stop if the selected field value is changed. Breakpoints are saved for later use. Rather than deleting a breakpoint, you can disable it so that it does not stop program execution. When you disable a breakpoint, it remains in the Breakpoints view.

**Adding a line breakpoint**

To add a line breakpoint, follow these steps:

1. Perform one of the following actions on an executable line:
   - Double-click in the prefix area (the small left margin of the source view).
   - Right-click in the prefix area and select Add Breakpoint.
   - Right-click in the source and select Add Breakpoint.
   - Right-click the Breakpoints view and select Add Breakpoint → Line (Figure 14-24).
Figure 14-24 Adding a line breakpoint

The Add a Line Breakpoint wizard (Figure 14-25) opens.

2. Program name, Module name, and Source are pre-filled with the information from the currently selected stack entry as shown in Figure 14-25. If a line number is selected in the source, it is placed into the Line entry field.

Click Next if you want to change optional parameters or you want to specify a condition for the breakpoint.

Figure 14-25 Add a Line Breakpoint wizard

3. The Optional parameters panel (Figure 14-26) opens. The optional parameters include:
   - **Thread**: Enter the thread number to set the breakpoint in. Unless you are setting the breakpoint for a multi-threaded application, 1 or Every are the only valid entries.
   - **Frequency**: Enter the frequency when to stop on a breakpoint and when to skip it. The debugger keeps track of how many times each breakpoint is encountered. The fields in this section tell the debugger on which encounter of a breakpoint the debugger should first stop, how often it should stop, and on which encounter the debugger should no
longer stop. In the From field, enter the first breakpoint encounter on which you want the debugger to stop. In the To field, enter the last breakpoint encounter on which you want the debugger to stop. In the Every field, enter the frequency with which you want the debugger to stop on this breakpoint.

- **Expression**: Enter an expression into this field. The execution of the program stops at the breakpoint only if the condition specified in this field tests true. For example, if you debug an RPG program, you can type `CustId='0003'`. The program stops here when the value of the field `CustId` is 0003.

Click **Finish** to add a breakpoint. Then you see the new breakpoint in the Breakpoints view (Figure 14-27).

![Figure 14-26 Optional parameters](image)

**Adding a Watch breakpoint**

To add a Watch breakpoint, follow these steps:

1. Select the variable to be watched in the source. Right-click and select **Add Watch Breakpoint**. Or right-click in the Breakpoints view and select **Add Breakpoint** → **Watch**.

2. The Add a Watch Breakpoint wizard opens. If you selected the variable in the source, the Address or Expression field is already filled in. Otherwise enter the name of the variable you want to watch. Leave the Number of bytes field set to zero to watch the entire length of the variable.

Click **Next**.

3. Change any optional parameters, and click **Finish**. The breakpoint is added to the Breakpoints view (Figure 14-27).

**Note**: The Expression parameter is not supported in Watch breakpoints.
4. When your debugging session detects that the content of your watch is changed, the Debugger Message window (Figure 14-28) opens.

Click OK. The debugger display the line right after the location that changed the value of the watched variable.

To enable or disable a breakpoint, right-click the breakpoint in the prefix area or the Breakpoints view and select Enable or Disable (Figure 14-29).

**Service entry points**

In addition to setting a Service Entry Point in the RSE view, you can also set it in the Debug perspective while debugging an application. You use service entry points when you want to debug an application but you don't know the job its running in. This could be for example a program invoked by a Toolbox call or a WebFacing job. You can use service entry points for:
Applications that are invoked using a server job, for example a Toolbox program call
Programs that are spawned by other programs
Any program that is not currently running or under debug

To set a service entry point, open the source of the application in the Debugger editor. Right-click in the editor or in the editor prefix area directly to the left of the line where you want to add the service entry point. Select Add Service Entry Point (Figure 14-30).

![Figure 14-30   Adding a service entry point](image)

This invokes the Add Service Entry Point window (Figure 14-31), which displays the program, module, source file, and line number of the service entry point that will be created. In this window, specify the user profile for which the service entry point will be activated. By default, the user profile is set to "CURRENT (the user profile for the current debug session).

![Figure 14-31   Add Service Entry Point](image)

When a service entry point is set, it is triggered when the application not currently under debug is called. This feature is useful especially when you develop Web applications that use host programs. For more information about how to use a service entry point, see “Debugging Web interaction programs” on page 584.

Important: Service entry points (SEP) are only supported on OS/400 V5R2 and later as well as in i5/OS environments. To set a SEP, from a debug session (it does not work from the editor) use the source pop-up menu or prefix area pop-up menu. When the program is called for the specified user ID in any job on the iSeries, the service entry point is hit and a new debug session is started for that job.

Debugger editor view

When you launch a debug session, the debugger uses the Debugger editor to display source code (Figure 14-32). Source is opened in the editor in browse mode. You cannot modify it.
To modify a file that is in the Debugger editor, right-click inside the editor. Select **Edit** from the pop-up menu (Figure 14-33).

The source code is opened in the same window but in a different tab. Selecting the tab will also allow you to populate the Outline view by pressing the **Refresh** button (Figure 14-34).

The Outline view is a good way to examine global definitions and main procedures of the source. For more information, see 6.1.4, “Outline views” on page 155.

Back in the Debug editor, when you need to know the value of a field without adding it to the Monitors view first, move the cursor on top of the field. The value is shown in a separate box (Figure 14-35).
From the debug editor, you can work with several other debug functions. The prefix area pop-up menu allows you to edit, remove, and enable/disable breakpoints and run the program to a selected location. Run To Location, Monitor Expressions and Monitor Storage, and Change text file are also available from the source pop-up menu. Depending on the debug option used to compile the program, Switch view between *Source, *Listing, and *Statement is available.

When you finish setting breakpoints, you need to resume the program execution so you can examine how the program is working. You can resume it in any of the following ways:

- Click the Debug view's Resume button .
- Select Resume from the Debug view's pop-up menu.
- Select Run $\rightarrow$ Resume from the menu bar.
- Press F8.
- Run to location.

The program stops at the next encountered breakpoint, unmonitored exception, or it runs to completion.

When the program is stopped, you can use the step operations to execute the program statement-by-statement. You can use the Step Into or Step Over button from the debug view toolbar, the debug view pop-up menu, the Run menu, or use F5 for Step Into and F6 for Step Over.

You may also select Run To Location to stop at a specific line in your program:

1. Make sure the line to run to is visible in the editor.
2. Ensure that the editor cursor is at the line and run the program to the cursor. Right-click the line and select Run to Location from the pop-up menu in the prefix area or the source.

**Run to Location:** This option does not run to the specified location if a breakpoint or unmonitored exception is encountered, or if the program completes without executing the specified location.

To leave the current call level and move back to the previous level use Step Return.

**Variables view**

The Variables view (Figure 14-36) always shows the local variables of the currently selected stack frame. You should use this view if you are debugging C, C++. When you debug RPG,
COBOL, CL, ILE RPG, ILE COBOL, or ILE CL, use the Monitors view (Figure 14-39 on page 575).

**Figure 14-36** Variables view

**Monitors view**

The Monitors view (Figure 14-39) allows you to see the contents of the field.

To add a field to the Monitors view, perform one of the following actions:

- Double-click the variable in the source, right-click it, and select **Monitor Expression**.
- Click the **+** button from the Monitor Expression view toolbar. Enter a field name and click **OK** (Figure 14-37).

**Figure 14-37** Adding a field to the Monitor Expression view

- Right-click in the Monitors view and select **Monitor Expression** (Figure 14-38). If a variable is selected in the source when the Monitor Expression window is brought up, this variable is entered into the entry field.
Chapter 14. Debugging and testing options

The Monitor Expression window is particularly useful when you want to monitor one specific element of an array or structure. You can enter, for example, ar1[15]. When the value of the field is changed during debugging session, the field information is changed to the color red in the Monitors view (Figure 14-39). A small delta symbol is added to the entry in the Monitors view.

If you need to change a field value, double-click the variable. The input field is opened (Figure 14-40). Type a new value and press Enter.
Modules (Programs) view
The Programs view allows you to navigate programs, service programs, ILE modules, and OPM programs. Any program, where execution stops, is automatically added to the Programs view (Figure 14-41).

![Figure 14-41 Programs view](image)

To add a program to the debug session, click the button from the toolbar, enter the program name, and select the type of the program from Add Program window (Figure 14-42).

![Figure 14-42 Add Program](image)

When you view programs, clicking filters out modules without debug data and leaves only the programs with debug data.
Storage monitor view

A Storage monitor allows you to view storage contents. If the program is compiled with the *TERASPACE storage model, you can alter the content of the storage.

To add a variable to the storage view. Right-click Storage view and select Monitor Expression. Enter the field name and click OK. Ensure that the Storage Monitor radio button is selected. Now you can see the storage content from Storage view (Figure 14-43).

![Figure 14-43  Storage view](image-url)
14.3.4 Testing and debugging an iSeries Web application

When you finish developing an iSeries Web application, the best way to test it is to use the
WebSphere Development Studio Client for iSeries test environment:

1. When you test the Web application the first time, right-click the first Web component and
   select Run on Server from pop-up menu.

2. The Server selection window as shown in Figure 14-44 opens. The selection of servers
   shown in the window depends on how you selected environments at the installation time of
   the WebSphere Development Studio Client for iSeries.

   The possible environments are:
   - WebSphere Portal v5.0 Test Environment (only if the Portal Tool Kit is installed)
   - WebSphere Version 5.1
   - WebSphere Version 5.0
   - WebSphere Version 4.0
   - Apache Tomcat Version 4.1
   - Apache Tomcat Version 4.0
   - Apache Tomcat Version 3.2
   - Other
Note, that starting with WDSc 5.1.2 the different versions of supported servers are shown in folders. So you first need to expand the folder of the appropriate server version, for example, WebSphere V5.1, and then select the type of server you want to use, for example, ‘Test Environment’ or ‘Express test Environment’ as shown in Figure 14-45.

![Figure 14-45 Create a new test server](image)

Note that the server configurations not containing the word Test, for example Express Server and Server in Figure 14-45, allow you to remotely attach to an application server, such as WebSphere Application Server running on the iSeries. You can use Other to configure a J2EE Publishing Server, a Static Web Publishing Server, or a TCP/IP Monitoring Server.

**Note:** Apache Tomcat version 3.2 supports only J2EE 1.2 Web modules.

3. Select your appropriate server and click Next.
4. Set the HTTP port number. Leave this value unchanged unless you have a reason to change it. Click Finish to start testing.

During the startup procedure, several windows open (Figure 14-46). The test environment starts (Figure 14-50 on page 582). Figure 14-46 shows the Publishing information window.

![Figure 14-46 Startup publishing](image)

5. If you already started the Server, you can publish the Web application without starting the server again. Click the Servers tab from the notebook (Figure 14-47).
6. Right-click the selected server and click **Publish**.

When the publishing is done, the status window is shown. If you click the Details button, you see the detailed information shown in Figure 14-49.
Figure 14-49   Publishing information
When the application is started, the requested page is shown (Figure 14-50). The test environment contains the main browser view and several views in the notebook at the bottom of the window.

Figure 14-50   Test Environment
From the notebook view, you can browse the application information. When testing the iSeries Web application, remember to enter all necessary information in the iSeries Web Tools Run-time Configuration window (Figure 14-51).

Enter your iSeries host name, user ID and password, and runtime library list. Select the **Display detailed runtime errors** check box if you want to see the details for any errors that occur during the run time of your Web application. With this selection, you can browse the field values transferred to JSPs.

![iSeries Web Tools Run-time Configuration panel](image)

*Figure 14-51  iSeries Web Tools Run-time Configuration panel*
You can also see the iSeries Server job name that is connected to your iSeries Web Application as shown in the Console view (Figure 14-52). This is useful for tracing and debugging development errors that may occur.

**Note:** Ensure that an error page is specified on the first page of the Web Interaction wizard, Specify a Name and Location for your Web Interaction.

The areas marked in Figure 14-52 indicate:

1. The input parameter passed to the first iSeries Web Integration program.
2. The iSeries Server job that is connected to your Web application.

![Console view](image)

**Figure 14-52** Console view

### Debugging Web interaction programs

When you need to debug a specific application, you may want to debug a job. There may be other reasons:

- You found a problem at a particular point in your program and you do not want to step through the program or set breakpoints. In this situation, you can run your program, and during the program, pause shortly before the anticipated failure (for example, while the program is waiting for keyboard input). You can then provide the input and debug from that point on.

- You are developing or maintaining a program that hangs periodically and you want to find out why it is hanging. In this situation, you can debug the job, and look for infinite loops or other problems that may cause your program to hang.

In our case, we continue to debug our Web application further.
One-step debugging

When you need to debug the iSeries Web Application further or you need to debug the iSeries job, go to the RSE perspective. Expand **iSeries Jobs** → **Your active jobs**. Right-click your **iSeries Server Job** and select **Debug As** → **iSeries Job** (Figure 14-53). The job is the same as when we tested the Web application and looked the Console view.

![Debug iSeries Job](image)

Figure 14-53  Debug iSeries Job

The debugger prompts you to invoke your iSeries application.
Debugging iSeries job with full prompt

When you are debugging with full prompt, you can set optional debugging parameters and control the debugging session in greater detail.

Start full prompt debugging:

1. Select Run → Debug and select iSeries: Debug Job → New (Figure 14-54).

   ![Launch Configurations](image)

   **Figure 14-54  Debug iSeries job**

   2. Complete the appropriate fields and click Debug.

   3. Debugger prompts you to invoke your iSeries application (Figure 14-55). Click OK.

      ![iSeries Debug Message](image)

      **Figure 14-55  iSeries Debug Message**

   4. Go back to main page of the Web application and enter your input again. The program starts and the debug perspective is started.
5. When using one-step debugging, the debug session steps into the program and stops. To add breakpoints go to the Debug editor view, select a line, and add a new breakpoint (Figure 14-56). For more information about how to add breakpoints, see 14.3.3, “How to debug” on page 565.

![Figure 14-56 Adding a breakpoint](image)

6. After you added one or more breakpoints, right-click in the editor view and select Run To Location.

7. Our Web application uses several service programs and modules. To debug all of them, we need to add a program to our debug session. In the notebook view, click the Modules tab and click the **+** icon. The Add Program window (Figure 14-57) opens.

![Figure 14-57 Add Program](image)

8. Complete the required parameters and click OK.

9. In the Modules view, you can see the added program. When you double-click the line with the library name/source file (source member), the source code opens in the Editor view (Figure 14-58).
10. To edit the source code, double-click the editor view and select Edit (Figure 14-59).

11. Source code is opened in edit mode in the debug session editor view.

12. Continue debugging until you find the error. The Debug session terminates automatically if you select the Terminate debug session on program completion check box in Launch Configuration. You can terminate the session manually by selecting Run → Terminate, right-clicking Debug view and selecting Terminate, or clicking the button on the Debug view toolbar.

14.3.5 Debugging a Web application

Debugging a Web application is executed in the same way as any application debugging using the Integrated iSeries debugger. The big difference when compared to other debugging methods is that you need more memory on your workstation. Follow these steps:

1. Right-click your Web application in the Web perspective and select Debug on server. The Step-by-Step Debug window (Figure 14-60) opens. The debugging session is started as step-by-step debugging so you are informed every time to step into the method or skip the method to be debugged.

   You can select the Disable step-by-step mode check box and continue the Web application to the first debuggable statement. You see this window because the application has been started as one-step debugging method.
Figure 14-60  Step-by-step debug

2. When the application reaches the first debuggable statement, the Debug perspective (Figure 14-61) opens. The Debug perspective has the same views as when debugging legacy applications, but the content of the views is slightly different.
In the Editor view (Figure 14-62), you can see the source code of your application. You can add bookmarks, breakpoints, and tasks.

![Editor view](image1.png)

**Figure 14-62 Editor view**

In the Debug view, you can see application run time attributes as thread information. When the application stops, the current thread is expanded and you can see the content of the thread in view. When you click inside of the thread information, you can open the source code of the selected method, if the source code is not available, the Debugger Source Lookup window (Figure 14-63) opens.

![Debug Source Lookup](image2.png)

**Figure 14-63 Debug Source Lookup**

You can select **Use default source lookup path** to search the source file or you can add file by clicking the **Add JARs** button. If the requested line contains a class file and the source code is not present, you are informed in the editor view (Figure 14-64).
Chapter 14. Debugging and testing options

Figure 14-64  Class file editor

Now you can attach the source file by clicking the **Attach Source** button.

In the Variables view, you can see the variables of current selected thread item (Figure 14-65).

Figure 14-65  JSP variables view

In the Outline view, you can see the structure of the current object (Figure 14-66).

Figure 14-66  JSP outline view
The Console view shows the output of the execution of your program. It allows you to enter input for the program.

You can continue the program execution as explained in “Debug perspective” on page 566.

### 14.4 IBM 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 or i5/OS ILE and OPM languages. 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.

**Note:** You should only use the Distributed Debugger for iSeries if you are developing with CODE or debugging Web objects that are running on WebSphere Application Server. You should use Object Level Trace when developing for WebSphere Application Server.

#### 14.4.1 Setting up a debug session

Before you can start debugging an iSeries application, you may need to perform some or all of the following tasks:

- Compile a program with debug data.
- Start the debug server.
- Start a debug session.
- Set the debugger port.
- Change the debugger settings.
- Ensure that your AS/400 user profile has the appropriate authorities.

**Compile options**

Programs must be compiled with appropriate debug information before compiled program objects can be used with the debugger. For example, ILE RPG, COBOL, and CL programs must be compiled with the *SOURCE, *LIST, *STMT, or *ALL option. The default is *STMT (Statement view). For more information about compiling each language, see your compiler documentation.

**Setting and specifying the debugger port**

By default, the debug server is set up to listen for connection requests on TCP/IP port 3001. If port 3001 is being used by another application, you can designate another port:

1. Enter the Work Server Table Entry (WRKSRVTBLE) command on the iSeries debug host. Change the port called QDBGGSVR to the new value.
2. When starting the Distributed Debugger, in the Port field of the Attach window or Load program window, depending on the method of start-up that you chose, specify the new port. The port specified in this field must match the port specified on the iSeries WRKSRVTBLE command.

**Note:** Before you change the port, end the debug server by entering the End Debug Server (ENDDDBGGSVR) command. Then change the port and start the debug server again. The valid port number range is 1 to 64,767.

**Starting the debug server**

Before you can use the debugger, you must start the debug server on the iSeries:
1. Enter the Start Debug Server (STRDBGSVR) command on an iSeries command line and press Enter. The server job is started under the QSYSWRK subsystem (Figure 14-67).

```
<table>
<thead>
<tr>
<th>Opt</th>
<th>Subsystem/Job</th>
<th>User</th>
<th>Type</th>
<th>CPU %</th>
<th>Function</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>QSSRVR</td>
<td>QUSER</td>
<td>PJ</td>
<td>0.0</td>
<td></td>
<td>CNDW</td>
</tr>
<tr>
<td></td>
<td>QSSRVR</td>
<td>QUSER</td>
<td>PJ</td>
<td>0.0</td>
<td></td>
<td>CNDW</td>
</tr>
<tr>
<td></td>
<td>QSSRVR</td>
<td>QUSER</td>
<td>PJ</td>
<td>0.0</td>
<td></td>
<td>CNDW</td>
</tr>
<tr>
<td></td>
<td>QSSRVR</td>
<td>QUSER</td>
<td>PJ</td>
<td>0.0</td>
<td></td>
<td>CNDW</td>
</tr>
<tr>
<td></td>
<td>QSSRVR</td>
<td>QUSER</td>
<td>PJ</td>
<td>0.0</td>
<td></td>
<td>CNDW</td>
</tr>
<tr>
<td></td>
<td>QSYSJOB</td>
<td>ITSCID87</td>
<td>BCH</td>
<td>0.0</td>
<td>PGM-QSSROUTE</td>
<td>SELV</td>
</tr>
<tr>
<td></td>
<td>QCPRT</td>
<td>QUSER</td>
<td>BCH</td>
<td>0.0</td>
<td></td>
<td>DEQW</td>
</tr>
</tbody>
</table>
```

Figure 14-67  Debugging the server

2. Start the code server. From a 5250 screen, enter the STRCODE command (Figure 14-68).

Figure 14-68  STRCODE
Starting a debug session

Start the Distributed Debugger:

1. Load the Debugger using one of the following options:
   - From the client command line, enter the following command:
     ```
     idebug -qengine=400 program_name
     ```
   - From the active CODE Editor window, click the debugger button. Or, from the toolbar, select Actions → Debug and select the appropriate option (Figure 14-69). We selected Interactive application for this example.

![Figure 14-69 Starting debugger from CODE editor](image)

2. Select your server (Figure 14-70). Click OK.

![Figure 14-70 Selecting a server](image)

3. On the Call Program window (Figure 14-71), enter the necessary information. Click OK.
4. If the connection is established, in the Debugger iSeries Logon window, enter your user ID and password. The debugging session starts (Figure 14-72).
The Debugger stops at the first debuggable line in code. You can now add breakpoints, add more programs to debugging session, and monitor variables. To add a breakpoint, right-click the selected line and select **Set breakpoint** or press F9 at the selected line.

When you need to add more programs to the debugging session, select the **Programs** tab from notebook view and then select **Programs → Add Program** from toolbar. The Add Program window (Figure 14-73) opens.

![Add Program](image)

**Figure 14-73  Add Program**

Click the **Programs** tab. The new entry is shown in the Programs view (Figure 14-74).

![Programs view](image)

**Figure 14-74  Programs view**

To monitor the values of the variables, right-click the selected variable and select **Add to Program Monitor**.

Continue to run the program by clicking the **go** button in the toolbar. Then the program runs to the next breakpoint and stops. Alternatively, you can right-click the line in question and select **Run To Location**.

### 14.5 Profiling Java and J2EE applications

Perhaps you have completed debugging and testing and your Java or J2EE application has been deployed. Once deployed, you notice a problem during the runtime (for example a memory leak), but you cannot access the production servers to examine what might be wrong.
To resolve the concern, you might use the Profiling tools found in the WebSphere Development Studio Client for iSeries to either profile your Java or J2EE application on a local development workstation or remotely against a staging server.

### 14.5.1 Introduction to profiling

Profiling is the task of examining the runtime attributes of a Java or J2EE application. This information is critical for resolving production-related problems within your Java or J2EE application. If you do not have access to appropriate profiling information, you will most likely spend an unnecessary amount of time trying to determine the source of your production problem.

There are generally two things we try to discover during profiling:

- Execution time
  - Where the execution time is spent. This helps identify time consuming methods.
- Memory utilization
  - Which objects are filling up the memory.

When you profile an application, there are a couple of things to remember. The behavior of the application code can change based on the available data. If your production system has a table with 1 million records but your development database only has 100 records, it may result in different runtime characteristics.

For example, most computers today have sufficient memory and processing power to easily deal with 100 records. The result may be that you will not see problems with caching or a performance problem in an algorithm which only happens if you have more than 100,000 records.

Another thing to consider when profiling is the amount of interaction with the application. For a Web application, this is related to the number and diversity of requests. If you only make a single request to a component, you may end up with misleading data. One example of this is when using singletons and various pools, where initialization is expensive but subsequent requests are cheaper. If you only record one request, you will only see the expensive initialization.

### Profiling Architecture

The profiling facilities provided with the WebSphere Studio family of development tools, including WebSphere Development Studio Client for iSeries, are designed to work in a client/server architecture. The profiling architecture is illustrated in Figure 14-75.

![Figure 14-75 Profiling: Architecture](image)

On the client-side, the profiling tool runs inside the Integrated Development Environment (IDE), including WDSec. The profiling tool provides the developer with various views into the application. The Agent Controller runs on the server-side and it manages one or more agents running within the Java Virtual Machines (JVM) on the server.
During development, you might use the profiling tool within the test environment on your local development machine. Then, during user acceptance or scalability testing, you might profile your application once again on the iSeries server where the application is deployed for staging.

The profiling capabilities include support for both local and remote profiling. With the profiling tools installed on the client-side, you merely need to make sure that the Agent Controller is available on the server.

**Tip:** Profiling is ideal while prototyping, during development, and throughout the remainder of the development life cycle prior to reaching production. Your system administrator will likely not recommend profiling in production unless absolutely necessary and we agree with this recommendation.

The Agent Controller is available on the distribution media that came with your WebSphere development environment (for example WDSc).

**Profiling Applications**
In the next sections, let's examine more closely how to profile a Java or J2EE application.

**Starting the server for profiling**
Before you can collect profiling information for your application, the Application Server must be started in profiling mode. Assuming the Agent Controller is installed on the server:

1. Access the **Server Perspective** within the development environment.
2. Select the server where you want to start a profiling session.
3. Right-click the server name to access the context menu.
4. Select **Profile** from the context menu as shown in Figure 14-76.

![Figure 14-76  Profiling: Selecting profile from the context menu](image)

This launches the server with profiling enabled and automatically opens the wizard for connecting to the Agent Controller.

**Profiling wizard**
After a successful start of the server, the Profiling wizard opens. The next step is to select the agent, as shown in Figure 14-77.
Chapter 14. Debugging and testing options

Figure 14-77  Profiling: Selecting the agent

The available agents are:

- **Java Profiling Agent**
  - This is a standard Java profiler. It is deployed with the agent controller and collects information about the execution of a Java program.

- **J2EE Request Profiler**
  - This is made available by the Application Server. It has knowledge of J2EE and allows profiling information to be collected intelligently across containers and application servers.

The second page of the profiling wizard allows you to specify the destination for the collected profiling data, as illustrated in Figure 14-78.
On the third page of the wizard you can specify the filters that will be used during profiling and this is shown in Figure 14-79.

Filters can help to limit the data collected. The filters are applied top-down, so you can have a filter which includes itso.ad.business before a filter which excludes itso.ad.*. The result of this
is that no information is collected for classes in itso.ad and subpackages, but information is collected for classes in itso.ad.business and subpackages.

There are three predefined filter sets:

- **Default**
  - This excludes standard Java interface and implementation packages.

- **WebSphere J2EE**
  - This excludes some WebSphere packages in addition to the Java interface and implementation packages.

- **WebSphere Studio**
  - This excludes the packages defined for the WebSphere Studio Workbench.

The fourth page of the wizard allows you to specify profiling options, as depicted in Figure 14-80.

![Figure 14-80 Profiling: Selecting profiling options](image)

The options you select influence the amount of information collected from profiling:

- Select both *My application uses too much memory* and *My application is too slow*.
- Click *Show details* for both options.
- Show instance level information.
- Select *Show execution flow graphical details*.
- Click Next to continue the wizard.

The fifth and final page of the wizard asks for profiling limits, as show in Figure 14-81.
Using these limits you can have profiling automatically stop after a number of invocations or a certain amount of time has passed.

Leave both options deselected and click Finish. A window with profiling tips is displayed. Click OK to close it.

**Profiling perspective**

Now that the server is ready for profiling, access the development environment on your workstation. Open the Profiling and Logging perspective within WDScl. Right-click your application's project in the Profiling and Logging perspective's Navigator. Select the option to Profile on Server, as shown in Figure 14-82.
Then in the Profile Monitor view, right-click and select the option to start monitoring, as shown in Figure 14-83.

Figure 14-83  Profiling: Start monitoring

Begin exercising your application. After you have executed some parts of your application, you can examine the collected profiling information. **Right-click** <monitoring> in the Profiling Monitor and select Refresh Views. You will then be able to begin analyzing the collected data.

**Analyzing timing information**

After collecting the data, we can look at the runtime performance of the application.

**Method statistics**

**Right-click** <monitoring> Profiling and **Open With → Method Statistics**. This opens the Method Statistics view, as depicted in Figure 14-84.
The timing information in this view is relevant when determining which methods are taking the most execution time. The unit of time shown in this view is a second.

**Method invocation**

When you have determined which methods may have a performance problem, it is interesting to discover the reason:

- Select the method and Show Method Invocation (see Figure 14-85).
  - This opens the Method Invocation view, which contains a graphical presentation of the method invocation.
- Click in the view and select Show Invocation Table from the context menu.
  - This opens the Method Invocation Table view.
Chapter 14. Debugging and testing options

Figure 14-85 Profiling: Select Method Invocation

The Method Invocation Table view displays each method invocation. The cumulative time presents the time spent executing the method and any methods called. You can expand the method invocation by pressing the small + in front of the method invocation. This way you can examine the timing information for methods called from the selected method.

Examining memory utilization

Excessive memory utilization can quickly turn into a bad production problem, especially if it results in memory being paged out (swapped to disk).

Instance statistics

In the Profiling Monitor, right-click <monitoring> Profiling and select Open With → Instance Statistics. This action will open the Instance Statistics view as illustrated in Figure 14-86.

Figure 14-86 Profiling: Open the Instance Statistics view

The Instance Statistics view shows statistics related to instances of classes during runtime. The column Total Instances shows the number of instances that has been created during the time the profiler has been running. The column Live Instances shows the number of instances currently allocated in memory. The columns Total Size and Active Size shows the amount of memory used in bytes.
**Garbage collection**

Try selecting Run Garbage Collection from the context menu (*right-click*) of <monitoring> Profiling. If the state is *<attached>* Profiling, you must first select Start Monitoring from the context menu, because the agent must be running to perform this action. Then select Profile → Refresh Views from the main menu. The result will most likely be that the number of live instances have decreased for at least a couple of classes.

**Object references**

It is also possible to examine the object references in the running application. Select Collect Object References from the <monitoring> Profiling context menu by *right-clicking*. Select Profile → Refresh Views. In the Instance Statistics view, select the Show Object References from the context menu. This opens the Object References view where you can examine the references to and from objects.

**Execution flow**

By *right-clicking* on <monitoring> Profiling, select Open With → Execution Flow from the context menu to open the Execution Flow view. The Execution Flow view shows the interactions between classes in a graphical format. The graphical view is practical when you are trying to determine which method is taking the most processing time.

You can use the zoom icon to enlarge an interesting area. The larger the graphical view, the more details are displayed. Each vertical bar represents a class. When you point the cursor to a vertical bar, the class name and the method that is invoked are displayed at the bottom. You can see the method names between the vertical bars. On the right-hand side, the time line is displayed.

*Right-click* and select Show Execution Table from the context menu. The Execution Flow Table view shows the detailed flow of execution from one class to another.

**Other available views**

There are several other views available to help you with profiling:

- **Package Statistics**
  - Shows statistics for packages. It is possible to expand a package to see the statistics for the contained classes. This view shows object instances and can be used for memory profiling.

- **Class Statistics**
  - Shows statistics for classes. The classes can be expanded to see the called methods. This view shows object instances.

- **Class Interactions**
  - Shows a sequence diagram of the class interactions during execution. This can help you to better understand the execution of the application.

- **Object Interactions**
  - Shows a sequence diagram of the object interactions during execution. This is much like the Class Interaction view except that individual objects are shown.

- **Thread Interactions**
  - Shows the interaction between threads in the application. This view can help you profile multi-threaded applications.
Process interactions
- Shows the interaction between processes (JVMs). This view is only available from the Host object in the Profiling Monitor view. If you cannot see any Host objects in the Profiling Monitor, click the down arrow on the titlebar and select Hosts.

Note: You must use the J2EE Request Profiler to collect this information.

Host interactions
- Shows the interaction between hosts (computers). This view is only available from the Monitor object in the Profiling Monitor view. If you cannot see any Monitor objects in the Profiling Monitor, click the down arrow on the titlebar and select Monitors.

Note: You must use the J2EE Request Profiler to collect this information.

If you want to use the Class Interactions or Object Interactions view while profiling applications running on application server, you should consider using the J2EE Request Profiler because it hides a large part of the classes or objects that are not relevant to your application code.

Stopping the profiling session
When you have finished profiling your application, you can end the session by simply stopping the server. If you want to keep the server running, perhaps to continue profiling later, you can also right-click and select <monitoring> Profiling and Pause Monitoring or Detach from Agent.

To save the profiling data in the ProfileProject select Save. You can reopen the statistics views later again for further analysis.
Part 5

Working with VisualAge RPG

Part 5 covers the basic tasks in using the VisualAge RPG tool. It can be very useful for RPG programmers who want to start building GUI applications.
Introduction to VisualAge RPG

This chapter explains how to use VisualAge RPG (VARPG), which is part of WebSphere Development Studio Client for iSeries (WDSc). This chapter also introduces the graphical user interface (GUI) designer.

The chapter was written originally based on WDSc 5.0 and now updated with the enhancements for WDSc 5.1.2. All significant changes are marked by a vertical bar at the left side of the page.
15.1 VisualAge RPG features

VisualAge RPG is a 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 directly on the client. The VisualAge RPG compiler is the same code base as the host ILE RPG compiler. You can move some logic between the client and server. You can 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

There is a great and ever increasing demand for applications with a GUI 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.

RPG is the dominant language in the iSeries community. RPG is on over 75% of all iSeries development systems. Most applications for the iSeries are written in RPG. iSeries users need the ability to enhance their valuable mission-critical legacy host applications to add a powerful GUI to these applications. It is a time and money consuming process to retrain 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 VARPG, which is similar to RPG IV on the iSeries server. It is based on 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.

WebSphere Development Studio Client Versions 5.0 to 5.1.2 added many enhancements to the VisualAge RPG product. The VARPG compiler is now compatible with the V5R2 RPGIV language definitions. This means all the enhancements from RPGIV in V5R1 and V5R2 are now available in VARPG.

Here are some of the highlights of these enhancements:

- Free form RPG, with specific improvements to the syntax that deals with GUI attributes
- Full support for qualified data structures
- Capability for error handling using the MONITOR operations code
- Data structure support for Input and Output fields for externally described files
- New built-in functions for conversion from character to numeric
- New Control Specification keyword SIGNON to allow more flexible authentication behavior.
- Enabling of prototyped CALLP operation code to call iSeries server programs.
This is only a partial list. The VARPG compiler now includes support for all RPGIV language constructs at the V5R2 language level.

15.2 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 two places:

- The Help in WebSphere Development Studio Client (contains a Search engine)
- The Help in the VARPG GUI designer (VARPG V5.1.2 contains an embedded Search engine)

15.2.1 Help topics for VisualAge RPG in the Development Studio Client Help

To access the information center in the Development Studio Client workbench, follow these steps:

1. From the Windows desktop, click Start → Programs → IBM WebSphere Studio → Development Studio Client for iSeries 5.x (or Development Studio Client Advanced Edition for iSeries 5.x). The workbench opens.
2. Select the Help menu option from the workbench.
3. Select Help contents.
4. From the Contents view of the Help window, select Developing and expand the nodes ...
   with classic iSeries Tools → VisualAge RPG as shown in Figure 15-1.

To use the help in the GUI designer, select the HELP option and select the manual you are interested in from the pull-down menu. These are the online manuals available for VARPG:

- Language reference: This is organized similar to the RPGIV language reference manual. It shows the differences between these two language environments.
- Parts reference: This reference lists all the GUI parts with their attributes and events and event attributes. It allow you to exploit the GUI capabilities of VARPG.
- Programming with VisualAge RPG: This book is a guide for using VisualAge RPG to develop client/server applications. It describes the steps 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.
GUI Designer: This guide explains many of the features of VARPG. It has examples of how to write code for the different GUI parts available. This guide is essential to get started with VARPG.

IPF: The Information Presentation Facility (IPF) lets you create and manage online help files for your application. With VisualAge RPG, the online help that you create will be native Windows help. These guides provide information about the built-in VARPG help support system. It may be more appropriate to use the native Windows help instead of IPF. Implementing native Windows help support needs a bit more work since it is not as closely integrated into VARPG, but it provides the end user with the Windows typical help environment. The language reference guide has tips on using the native windows help system in VARPG.

15.2.2 Help topics in the GUI designer

You can also access help topics from the GUI designer, by selecting Help and then on of the following topics from the menu bar:

Programming with VisualAge RPG: This book is a guide for using VisualAge RPG to develop client/server applications. It describes the steps 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.
Chapter 15. Introduction to VisualAge RPG

15.3 Programming with VARPG

The VARPG environment has different elements. There are also basic differences between the procedural 5250 programming style and the GUI event-driven style.

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, after it is 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 15-2 shows the basics of an RPG IV procedural application.
VARPG is an event-driven language. The user interface is not only graphical, but the basics of this language differ from RPG IV. Figure 15-3 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.

![Diagram of event-driven model](image)

**Figure 15-3** Event-driven model

A 5250 green screen is designed using fields, constants, and command keys. In the graphical design, VARPG contains parts, attributes, and events. Figure 15-4 shows how these terms fit into the design. The VARPG Entry field part replaces the 5250 input/output fields, and the Push button part replaces the command keys in a 5250 panel. The GUI Entry field, for example, has many more attributes than its 5250 counterpart because the GUI provides a richer environment. Each of the VARPG parts has many attributes to provide the capabilities of a rich user interface. The major difference to the 5250 user interface environment is the fact that all these GUI parts also emit events. A VARPG programmer can easily select any event that a specific part supports and write RPG logic that is invoked when this event is triggered by the user interface at run time. When the VARPG-specific action subroutine is invoked by an event,
the GUI designer has an easy way to link GUI events to action subroutines. Action subroutines are similar to “normal” RPG user subroutines. They are invoked by GUI events instead of the EXSR operation code.

To create a GUI application, you use the VARPG GUI designer. It allows to create the user interface of your application easily. It is also the launchpad for the source editor and the VARPG compiler. It actually functions as the programmer workbench for the VARPG environment.

15.3.1 Starting the VARPG Designer

Before you work with VisualAge RPG, you need to make sure that at the end of installing WDSc, you have run the option ‘Complete the installation of CODE and VisualAge RPG’. If you are not sure you can invoke this option any time from the Windows desktop.

Start → Programs → IBM WebSphere Studio → Development Studio Client for iSeries 5.x (or Development Studio Client Advanced Edition for iSeries 5.x) → Complete the installation of CODE and VisualAge RPG.

To start the VARPG GUI Designer, from the Windows desktop, click

Start → Programs → IBM WebSphere Studio → Development Studio Client for iSeries 5.x (or Development Studio Client Advanced Edition for iSeries 5.x) → VisualAge RPG → GUI Designer.

Or on your desktop, double-click the VisualAge RPG project folder. In the project window that opens, double-click the New GUI project icon.

Note: VARPG provides many sample applications. To work with these samples, open the Samples folder in the VisualAge RPG projects folder. Select any of the sample VARPG projects. To open the GUI designer for an existing project, right-click the project icon and select the Edit option.

Now the VisualAge RPG GUI designer (workbench) starts.

The VisualAge for RPG application development environment is based on GUI projects. In this environment, your applications are also organized into projects. A project represents a complete set of objects (source code files, GUI objects, and so on).

15.3.2 The VARPG GUI Designer

Use the GUI Designer to work with VARPG applications. The VisualAge RPG GUI Designer as shown in see Figure 15-5 provides the typical GUI features and in addition these components:

- Project View
- Parts Palette
- Designer Canvas
The Project View

The Project View, similar to the example in Figure 15-6, shows all the parts contained in your user interface for the application you are working with.
The project view in this example shows that the application consists of three windows. The window named WIN2 is expanded. A canvas part allows you to position multiple individual parts inside the window frame. The canvas contains a subfile. The subfile itself has three columns. The subfile can be filled by writing to it using the WRITE operations code, which is similar to the DDS subfile. A pop-up menu is also part of the window. The menu has four menu items.

Double-clicking the window icons in the tree view opens them in the designer. Double-clicking individual parts on the design window opens the properties notebook for the selected part.

The project view is the programmer’s workbench for the user interface.

The Parts Palette

The Parts Palette (Figure 15-7) contains the parts that you use to create the GUI of your application. Attributes and events are associated with each part. By changing attributes and associating business logic with the events, you customize the behavior of the part according to your application logic.

![Parts Palette](image)

To add a specific part to the design window, click the part on the parts palette, then click the location in the design window where you want to place the part. An instance of the part is placed on the design window. You can now work with this part in the design window to change its properties and select events that should invoke specific action subroutines. To work with a part on the design window, right-click it and select the desired action from the pop-up menu.

The Design window

The Design window (Figure 15-8) is the area where you design the GUI for your application. The area inside the frame is normally the Canvas. You select the visual and non-visual parts on the parts palette and drop them on the canvas. By organizing them on the canvas, you...
create a 'What You See is What You Get' (WYSWYG) user interface for your application. You can also create a window without Canvas. It only allows you to place one part in the window frame, which can be useful. For example, for the image part, when placing it on a window without canvas, it automatically occupies the entire window frame.

![Design window](image)

**Figure 15-8  Design window**

### 15.4 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.

We use the sample library **LIB6961**, it needs to be installed on your iSeries to follow the examples in this chapter. See Appendix A, “Additional material” on page 679, for download instructions.

#### 15.4.1 Creating the user interface

To create the first window with a subfile part, follow these steps:

1. Start the GUI designer to create a new GUI project. The GUI designer opens with a default design window.

2. Select the Subfile part icon on the Parts Palette as shown in Figure 15-9.

   **Tip:** If you “hover”, that is move the mouse pointer without clicking over any of part icons, the description of the part appears in the Status bar at the bottom of the Parts Palette.

3. Click within Design window at the location where you want to position the upper-left corner of the subfile. The subfile frame is created on the window canvas.
4. Drag the subfile’s borders with your mouse to size it to the desired height and width.

![Subfile part icon](image)

**Figure 15-9 Creating a subfile part**

5. To change the window heading perform the following steps:
   
a. Double-click the Design window title bar as indicated by the mouse pointer in Figure 15-9 and the Window Part Properties dialog similar as in Figure 15-10 opens.

   Be careful that you double-click the Window Title Bar, if you double-click the Canvas, the Part Properties dialog for the Canvas will be shown instead of the properties for the window.

   b. Enter Customer as the new Part name.
   
c. Enter a title for the window, Customer info in our example.
   
d. Click OK to close the Window Part Properties dialog.
The subfile at the moment is just an empty frame. You need to specify what data to display in the subfile at run time. Most likely you already have the field definitions for the data to be displayed in your database on the iSeries. The GUI designer allows you to specify this reference information for the columns in the subfile.

Here are the steps to specify which database information should be displayed in the subfile, using the reference information from the iSeries server. In order to do this, you need to connect to the iSeries server.

If you have not specified an iSeries server in VARPG before, follow these steps:

1. On the GUI Designer menu bar, click Server → Define TCP/IP servers.
2. Click Add.
3. Specify the iSeries server name or IP address and an alias name in the window.
4. Click OK.
5. Click Close.
6. From the GUI Designer menu bar, click Server → Define server logon.
7. Click Add.
8. From the Remote location drop-down menu, select your iSeries server.
9. 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 every time it needs to access the iSeries.

10. Click **OK**.
11. Click **Close** to close the Define Server logon window.

Now you are ready to specify the reference fields on your iSeries:

1. From the GUI Designer menu bar, select **Server → Define Reference Fields**.
2. In the *Define Reference Fields* dialog, expand **Server → Name of your server** → **LIB6961 → CSTMR** and double-click **CSRCRD**. In the right pane of this dialog you see all fields of this record format.
3. Select the fields you want to display in the subfile. In this example, you select **CID**, **CFIRST**, **CLAST**, and **CCITY**. You can select multiple fields by holding the Ctrl key and clicking each field.
4. Move the mouse cursor to the subfile part within the canvas of Design window and click inside of it. The selected fields are now added to the subfile as shown in Figure 15-11.
5. Click the **Close** button in *Define Reference Fields* dialog to close that window.

![Figure 15-11 Adding fields to the subfile](image)

To make the subfile look nicer, you may change the heading colors by double-clicking the subfile part on the canvas to change its properties. The *Subfile Part Properties* dialog as in Figure 15-12 opens. Follow these steps:

1. On the General page, change the part name to **CUSTPART**.
2. Use the arrow buttons on the upper right of the dialog to slide the tabs, and then click the **Color** tab.

3. On the Color page, complete these tasks:
   a. Select the **Heading color background** option from the **Apply to** pull-down menu.
   b. Select the **Predefined colors** radio button and select **Blue** from the pull-down menu.
   c. Select the **Heading color foreground**.
   d. Select the **Predefined colors** radio button and select **Yellow** from the pull-down menu.

4. Click the **Font** tab.

5. On the Font page, complete these tasks:
   a. Select the **Select font** radio button.
   b. Click the **Change Font** button.
   c. Select **Comic Sans MS** and set the size to 10.

6. In the **Subfile Part Properties** dialog, click the **Field List** tab. Select **CID** and click **Change**.

7. In the **Subfile Field Part Properties** dialog, complete these tasks:
   a. On the General page, in the Heading 2 field, add **ID**.
   b. Click the **Style** tab.
   c. Change **Column width** to 100 pixels to fit the heading text into the column width.
   d. Click **OK**.

8. Click **OK** in the **Subfile Part Properties** dialog.
15.4.2 Saving the project

Before you write the logic, save the project:

1. In the GUI Designer, select Project → Save.
2. In the Save window (Figure 15-13) that opens, complete these tasks:
   a. Specify the project name CUSTLIST.
   b. Click the Source file name entry field (all remaining fields are filled using the project name as a base).
   c. Click OK.
Now write the logic to fill the subfile.

### 15.4.3 Writing the logic to fill the subfile

You are done with the user interface. You need to write RPG code to fill the subfile with data and you need use an event to get the data from the iSeries server and fill the subfile with his data. Otherwise, the subfile appears empty at application startup. A very convenient event to use when a VARPG application starts is the Create event of the window. You use this event to read the data from the database and write it to the subfile.

In the GUI run-time environment, each part fires a Create event when it is created, the window Create event is particularly useful to load initial values to any part on the window. The window Create event is fired after all parts on the window are created, so the parts are ready to be worked with when this event is fired:

1. To add the action subroutine for the Create event, right-click the window title.
2. Select Events → Create, and the VisualAge RPG editor similar to the one shown in Figure 15-14 opens. The frame of an action subroutine is created in the source editor window.
3. Enter the logic as shown in Figure 15-14 and do not forget to add the F specification outside the action subroutine.
15.4.4 Building and running the application

Perform these steps to test your application:

1. Save the file.
2. In the GUI Designer, select Server → Define iSeries information to specify the server on which your database file is located by performing these steps:
   a. On the Servers page: Make sure the correct iSeries server is selected (the one that contains library LIB6961)
   b. On the Files page: Use the Change button to change the Remote File information to change the library name for file CSTMR from *LIBL to LIB6961. If you have changed the iSeries environment and LIB6961 gets added to the library list of your jobs when you sign on. You need to make sure that the file CSTMR can be found by the VARPG compiler and later by the VARPG runtime.
3. Build the project. Select Project → Build → Windows, or click the Build the Windows version of the project button on the toolbar.
4. In the new pop-up window that informs you that the build process was successful, click OK. If you have any compile errors, check the error list that is displayed. To fix an error double-click the error in the error list, that will position you in the source editor at the line where the compiler encountered the error. Fix the problem and build the project again until all errors are fixed.
5. Click Run the project on the toolbar. Your application starts and the window with the subfile part, filled with data, appears (see Figure 15-15).
15.4.5 Enhancing the application

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 one page of the list area of the subfile. The subfile part automatically creates a scroll bar when it is filled over its page size limits. When you scroll the subfile scroll bar to its end or click the arrow at the bottom of the scroll bar, the subfile issues a PageEnd event.

You use this PageEnd event to add records to the subfile if it is scrolled to the bottom at run time by following these steps:

1. if the design window is closed, go to the project view and in the tree view double-click the window icon.

2. In the GUI Designer, in the design window right-click the subfile part on the design window and select Events → PageEnd. The Editor window as shown in Figure 15-16 opens, and the PageEnd action subroutine is created.

3. In the source editor, create a FillPage user subroutine. This subroutine is invoked from the Create and PageEnd action subroutines.

4. 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 19 records. (You can use SEU line commands cc and a to do the copy).

5. In the Create and PageEnd action subroutines, add an EXSR op code to invoke subroutine FillPage.

6. Make sure you specify variable COUNT as a numeric of length 2 with 0 decimals in the D specifications as shown in Figure 15-19 on page 631.

7. Save and build the application.
Figure 15-16   Adding the Fillpage subroutine

Now when you run the application, the subfile fills with 19 records. When you scroll the subfile scroll bar to the bottom, the PageEnd event fires and another 19 records are added to the subfile.

You may 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 into 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.

15.4.6 Selecting a record from the subfile

Let's add one more feature to your application. If the user double-clicks a record in the subfile, it shows 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 use the Select event.

1. 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 view. A new window is created.

2. Double-click the Design Window title bar and change the name to Win2 and the title to Customer detail in the Window properties dialog.

3. Click the Startup tab in the Window properties dialog and deselect the Open immediately check box. This prevents the window from loading when the application starts. You load this window when the Enter event from the subfile is fired.

4. Click OK to close the Window properties dialog.

5. To add the reference field information for the customer detail data to this window, from the GUI Designer menu bar, click Server-> Define reference fields.
6. In the Define Reference Fields window, expand **iSeries Server-> LIB6961-> CSTMR** and double-click **CSRCD**.

7. Select all fields in the record format, by selecting the first field in the list. Then, holding down the Shift key, selecting the last field, to select all fields.

8. Click the upper-left corner of the Design Window **WIN2** you just created. All selected fields from the Reference Field window are positioned on the Design window.

9. 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 as in Figure 15-17.

10. Rename the CDATA field to CDATA1. The GUI only allows fields with a maximum length of 255, but the field length in the database is 500. Therefore, you must redefine this field.

11. Click the **X** button in the window title, to close the Customer Details window.

12. 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.

![Figure 15-17 Designing the Customer Details window](image)
13. Add code to this action subroutine to see the detail window. Use the VARPG \texttt{SHOWWIN} operation code as shown in Figure 15-18.

\begin{verbatim}
* C CUSTPART BEGACT ENTER CUSTOMER
C C READS CUSTPART
C C CHAIN(E) CSTMR
C C SHOWWIN(E) 'WIN2'
C C eval cdata1=cdata
C C WRITE 'WIN2'
C C eval WIN2.WIN2.Focus = 1
C C ENDACT
\end{verbatim}

\textbf{Figure 15-18  Coding the Enter event}

Let's look at the code:

- The first statement uses the \texttt{READS} operation code, which is VARPG-specific and does not exist in RPG IV. It is equivalent to the \texttt{READC} operation code in RPGIV. It does not read changed records from the subfile, but it reads selected records.

- The \texttt{CHAIN} operation code is normal RPG processing. It accesses the data from the database.

- The \texttt{SHOWWIN} operation code loads a window into memory, which is your detail window in this case.

- The \texttt{WRITE} operation code 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 second \texttt{EVAL} operation code gives this window focus, so that it is on top of the other windows on the desktop. You are using a qualified Attribute name instead of the \texttt{SETATR} operation code.

- In the file specification, you need to change the file to \textit{keyed access} and to add the keyword \texttt{BLOCK(YES)}. This tells the compiler to block for performance reasons instead of using no-block, which is the default for keyed files (see Figure 15-19).

\textbf{Figure 15-19  Changing the file specification}

14. Save your source.

15. On the GUI Designer toolbar, click \textbf{Build} to build the application.

16. On the GUI Designer toolbar, click \textbf{Run} to run the application.
17. Double-click a record in the subfile. The Customer Details window (Figure 15-20) with the selected customer data opens.

![Customer Details Window](image)

**Figure 15-20  Testing the code**

Writing subfile applications with VARPG can be quite easy, as we have demonstrated here. The type of application created here is called a **fat client application**. That is because all logic runs on the client and the server is used for database access only.

The approach described in the next sections, allows you also to write thin client applications with VARPG by using the call interfaces to the iSeries so most of your business logic runs on the iSeries server. Only user interface-related logic runs on the client.
15.5 Building a thin client application with VARPG

The following section shows how to write a thin client VARPG application that uses similar programs or service programs as the sample Web application described in 13.2, “Developing the Web application” on page 440. This technique allows you to re-use business logic for both client/server-rich GUI applications and Web browser applications.

The Web tools sample application consists of three Web pages. To create a similar GUI application, you design three windows in VARPG for this sample application. This time, to make this application thin on the client, you do not use direct file access to the iSeries database. Data is accessed by calling procedures in service programs on the iSeries. Data structures defining the data are passed as parameters between the client and the server programs.

The following section describes step by step how to create the VARPG project for this example. Later, in 15.5.9, “The backend programs running under OS/400 or I5/OS” on page 671, we explain the ILE RPG programs used to communicate with the VARPG thin client.

15.5.1 Changing the default setting in the GUI designer

Change the user options in the VARPG GUI designer to simplify the environment for the programmer:

1. Select Options → User Preferences. The User preferences dialog as shown in Figure 15-21 opens.

![Figure 15-21 User preferences window]
2. Select the **Dialog background** check box. This gives you, by default, a gray background in your windows.

3. Select the **MS Sans Serif** for Font name. Set Font size to 10 to make the text on the windows look better by default.

4. Under Define Reference fields dialog, you may want to select **Use cache**, which makes the builds faster.

5. Under Build process, deselect **Save confirmation** and **Success notification**. This avoids showing the dialog boxes when using the VARPG build option that ask whether you want to save first. By deselecting these options, it always saves first without asking. Then you only get a build confirmation message when the build fails, not when it is successful.

6. Under iSeries server, specify your default iSeries server name and select **TCP/IP** as the communications protocol. This information is then automatically added to a new project.

7. Click **OK** and you are ready to start.

### 15.5.2 Building the first window

Let's build the user interface and start with creating a new VARPG project by performing these steps:

1. In the GUI designer, use **Project-> New-> GUI project** to create the new project. It creates one Design window automatically. This window should now have a gray background. The finished window should look like the example shown in Figure 15-22.

![Figure 15-22 First application window](image)

2. Double-click the window heading and properties dialog for that window opens.

3. Change the **Name** attribute to `win1`.

4. Change the **Title** attribute to `Customer Master Inquiry`.

5. Add a Static text part in the middle of the Window for your company name by selecting the Static Text icon from the parts pallette.
6. Double-click the **Static text part**. In the **Static Text Part Properties** dialog which opens, perform the following steps.
   a. Change **Text** attribute to your company name.
   b. Click the **Font** tab. Click the **Select font** option and select a visually appealing font and increase the size (Figure 15-23).

   ![Figure 15-23 Properties dialog Font page](image1)

   c. Click the **Color** tab and select **Foreground** from the **Apply to** pull down menu.
   d. In the same dialog select **Blue** from the **Predefined colors** pull down menu.

   ![Figure 15-24 Properties dialog Color page](image2)

   e. Click **OK**.

7. Size the **Static Text** Part on the design window so all text shows.
8. Place two Image parts on both sides of the static text.
9. Double-click the Image part and the *Image Part Properties* window opens as shown in Figure 15-25.

10. Specify nice images to show on this window. We used images from the WDSc workbench. Later, after saving the project and the project directories are created, copy the images into the project’s RT_WIN32 directory so the VARPG runtime can find them.

![Image Part Properties window](image)

*Figure 15-25  Properties page to specify image name*

11. Add a Static Text Part for the entry field heading to the design window.

12. Double-click the part to view the *Static Text Part Properties* window as in Figure 15-26.

13. Change Text field to *Enter ZIP code* and click OK to apply the change.

14. Add an Entry field part and position it next the “Enter ZIP code” text by dragging the Entry field icon from the parts palette to the canvas.

![Static Text Part Properties window](image)

*Figure 15-26  Static Text Part Properties window*

15. Double-click the Entry field. The *Entry Field* Part Properties window (Figure 15-27) opens. Change its properties.
### Chapter 15. Introduction to VisualAge RPG

15. **Introduction to VisualAge RPG**

   a. Change its **part name** attribute to ZIP on the General page of the properties notebook.
   b. Select the **Data** page in the properties dialog. Make the following changes:
      - **Type**: **Numeric**
      - **Length**: 5
      - **Decimals**: 0
   c. Click the **Editing** tab. On this page, specify Z for **Edit Code**.
   d. Click **OK** to apply the changes to the **Entry field** part.

![Figure 15-27 Entry Field Part Properties page for data definition](image)

16. Add two **Push button** parts ( ) to the design window. Position them side by side at the bottom of the window.

17. Change the properties of the first button:
   a. Change **Part name** to **PSBSEARCH**.
   b. Change **Label** to **List customers**.

![Figure 15-28 Push button properties](image)
18. Click the **Style** page.
   a. Select the **Default** check box. This causes a Press event for this push button to be generated when the Enter key is pressed at runtime.
   b. Click **OK** to apply the changes.

![Push button part properties, Style page](image1)

Figure 15-29 Push button properties, Style page

19. Double-click the second push button to open its properties window.
   a. Change the **Name** to PSBCREATE.
   b. Change the **Label** to **Create new customer**.
   c. Click **OK** to apply the change.

20. Add a **Status bar part** to the window.

21. Double-click it. On the properties window, complete these tasks:
   a. Change **Name** to **Status**.
   b. Click **OK** to apply the change.

The final window should look as shown in the example in Figure 15-22 on page 634. It contains some decorative artwork and has an entry field that allows you to specify a ZIP code. When you click PSBSEARCH, the database is accessed and all customers with this ZIP code are listed in a second window that you create next.

Clicking the **Create customer** button allows the user to add new customers to the database. The status bar shows the results of the search and how many customers were found for a search request. The code to do this is added after all windows are designed.
15.5.3 Creating the second window

Figure 15-30 shows the second window to be added to this application.

Figure 15-30  Second window with subfile and pop-up menu

Follow these steps to create the second window:

1. Click the Window part in the parts palette and then click the Designer project view. A new window is created.
2. Double-click its frame and a Properties window opens.
3. On the Properties window, complete these tasks:
   a. Change Name to WIN2.
   b. Change Title to Customer Master List.
4. Click the Startup tab.

Figure 15-31  Window properties, Startup page
5. On the **Startup** page (Figure 15-31), complete these tasks:
   a. Deselect the **Visible** check box.
   b. Deselect the **Open immediately** check box.
   
   **Note:** These deselections keep the window from loading at startup of the application. When it is loaded programmatically, it is invisible until the Visible attribute is set to 1.
   
   c. Click **OK** to confirm the changes.

6. Add a **Subfile** part ( subfile ) to the window.

7. To add some fields to the subfile, complete these tasks:
   a. In the GUI Designer, select **Server-> Define reference fields**.

   ![Define Reference Fields window](image.png)

   **Figure 15-32**  *Define Reference Fields window*

   b. In the Define Reference Fields window (Figure 15-32) that opens, select the **iSeries server** and the **LIB6961 library**.

   c. In the library, expand **CUSTMSTL**.

   d. Double-click the **CUSREC record format**. All fields in **CUSREC** display in the field list.

   e. Select the **CUST, NAME, and ZIP fields** (hold down the Ctrl key for multiple selections).

   f. Go back to the design window and click the **Subfile part** to add the fields. The fields display as subfile columns.

8. Double-click the **Subfile** part to change its properties:
   a. Change the **name** to **SUBZIP**.

   b. Select the **Field List** page in the subfile properties dialog.

   c. Select the **ZIP field** in the field list and click **Move up** twice so this field is located at the top of the list.
d. Click **Change** on this page to change the **ZIP column** properties:
   i. Change **Name** to ZIPC.
   ii. Change **Heading** to ZIP code.
   iii. Select the **Style** page and change **Column width** to 99.

e. Click **OK** on the Field properties dialog (not the subfile properties dialog).

9. In the Subfile properties dialog, on the **Field list** page, select the **CUST** field and click **Change** to change the CUST column properties:
   a. Change the **Name** to CUSTNO.
   b. Change the **Heading** to Number.
   c. Select the **Style** page and change the **Column width** to 93.
   d. Click **OK** in the Field properties dialog (not the subfile properties dialog).

10. In the Subfile properties dialog, on the **Field list** page, select the **NAME** field and click **Change** to change the NAME column properties:
    a. Change **Name** to NAMES.
    b. Select the **Style** page and change **Column width** to 288.
    c. Click **OK** on the Field properties dialog (not the subfile properties dialog).

11. Click **OK** on the **Subfile** properties dialog to apply all changes to the subfile part.

12. Select the **Subfile** part and size it so the horizontal scroll bar disappears (all data fits into it).

13. Size the height of the **Subfile** part so that 10 to 15 rows can be displayed.

14. Move the **Subfile** part to the upper left corner of the window and size the window so it contains only the Subfile, with no empty space around it.

15. Go to the parts palette and select the pop-up menu part.

16. Click the design window title bar to place a pop-up menu on this window.

**Note:** Pop-up menus appear at runtime at the cursor location when you right-click, so the location at design time is irrelevant.

Menus are dealt with differently than other GUI parts. They can only be changed in the project view's tree list, not in the design window as you can see in Figure 15-33.
17. Add three more menu items to the pop-up menu by selecting the Menu item part in the Parts Palette, and click the pop-up menu in the project view.

   Note. Do not use the pop-menu on the design window. Do this three times. Then you should have four menu items on the pop-up menu as shown in Figure 15-33.

18. Double-click the pop-up menu node in the project view and change its name to POPSUB.

19. Double-click each of the four menu items and change them to:

   Name: MNICREATE  Label: Create new customer  
   Name: MNEDIT     Label: Edit customer detail  
   Name: MNIBROWSE  Label: Browse customer detail  
   Name: MNDELETE   Label: Delete customer record

   This window is now completed. One more to go.

### 15.5.4 Creating the third window

Now, you can proceed to add a third window to the project view similar to the one shown in Figure 15-36 on page 644 by performing the following steps:

1. Add a new Window.
2. Double-click its frame or title bar and change its name to WIN3.
3. Change its title to Customer detail.
4. Go to the Startup page and deselect the Open Immediately check box.
5. Add the fields containing the detail information to the window. In the Designer, select Server-> Define reference fields.
6. In the define reference fields dialog, select the iSeries server and the LIB6961 library.

7. In the library, expand the MLGMTL2 file.

8. Double-click record format CUSREC. All fields in CUSREC display in the field list.

9. Select all fields by selecting the CUST field, then holding down the Shift key, and selecting the ARBAL field.

10. Go back to the design window and click the upper left corner.

11. All fields and headings are added to the window. They should all be still selected. You can now move them around so they are well located in the window frame.

12. Deselect the labels by holding down the Ctrl key and single clicking each label.

13. Click the selected entry fields and move them to the right so that all labels are completely visible.

14. You need to expand the label static text parts to the right to make the text completely visible, since the headings are pretty long.

15. Resize the Window by stretching its frame.

16. Add a push button part to the bottom of the window.

17. Change its name to PSBOK and its label to OK.

You are done with the GUI design. Now you must save the application. On the Save as Application window, complete these steps:

1. Save the project as CUSTMAST:
a. In the Application name field type CUSTMAST.

b. Click the Source file entry field. All fields are filled with defaults, based on the application name.

c. Click OK to save.

Figure 15-36 shows the third window you just created.

15.5.5 Adding code to run the application

Now you add some logic to the application. The Web application example (see 13.2, “Developing the Web application” on page 440), on which this application is based, shows a list of customers based on the ZIP code entered in the first page. From the subfile containing this customer list, you can select a customer and get detail information.
It calls RPG programs and procedures on the iSeries server to find the customers in the database and add them to the subfile. The logic to show the detail customer information is implemented with RPG programs on the iSeries server. In our example, the VARPG client program re-uses most of the logic used in the Web application discussed earlier in this Redbook by calling the same server programs on the iSeries.

The subfile in the Web application is implemented by using a user space to collect the data to show on the Web page. The Table Web component then reads the data from this user space and displays it. In the VARPG application, you can use this same user space. You invoke an additional procedure to read data from the user space and fill the VARPG subfile.

**Setting up for free form RPG action subroutines**

In this exercise, you use some of the new feature of the VARPG language that IBM introduced in Development Studio Client Version 5.0. One of these features is free form RPG. To demonstrate this new capability we choose free form RPG for the following examples. However, if you prefer to use the fixed form just, you may as well use equivalent operation codes for fixed form RPG to achieve the same results.

In order for the source editor to add a free form-based action subroutine instead of fixed form, it needs to find an */end-free* compiler directive at the end of the C specs. Let’s add this first:

1. In the GUI designer, select Project-> Edit source code.
2. Add the */free* and the */end-free* directives at the end of the source as shown in Figure 15-37. The */end-free* directive must be located at the end of the C specs before any O or P specs.
3. Save the source.

![Figure 15-37 Editor with /end-free directive added](image-url)
Positioning the cursor on the entry field

To position the cursor in the ZIP code entry field at runtime, you can write an action subroutine that sets focus to this entry field when the window is loaded. In the GUI designer perform the following steps:

1. Right-click the WIN1 window frame and select Event-> Create.
2. An Action subroutine for this event is added.
3. Add the following one line of code between the BEGACT and ENDACT operation codes:

   ```
   BEGACT WIN1_WIN1_CREATE;
   win1.zip.focus=1;
   ENDACT;
   ```

We suggest that you use the qualified dotted notation, such as for attributes to set or get attribute values instead of the older style fixed form operation codes SETATR and GETATR or their equivalent built-ins.

Getting data from the server

Now build the action subroutine to react to the press event from the PSBSearch push button:

1. Double-click the WIN1 icon in the project view to display the design window.
2. Right-click the List customers button (PSBSearch) and select Event → Press and an Action subroutine is added.
3. Add the code in Example 15-1 to the VARPG source.

Example 15-1   Code to add to the action subroutine

```
This code shown in Example 15-1 performs the following actions:

- Reads the ZIP code from WIN1.
- Initializes variable pgmname1 with the library and program name to call on iSeries.
- Calls a procedure that invokes an Series program and passes the library, program, and ZIP code to the procedure.
  
  Note, the CALLP operation code as shown in Example 15-1 only works with VARPG V5.1.2 or higher.
- The iSeries program uses the ZIP code and fills a user space called SFL1 with the resulting data for this query (all customers in the database located at this ZIP code).
- Invokes subroutine Getdata.
- Getdata loads WIN2, which contains the subfile part. The monitor is there in case the window is already loaded and avoids getting a runtime exception.
- Three variables are initialized:
  - The customer number (custno) initialized to 0001, because customer number 0 indicates end of data in the user space.
  - A counter (count) to keep track of the number of records that were found.
  - The name of the user space (subfile) containing the data on the iSeries.
- Inside the DOW loop this subroutine receives data from a procedure on the iSeries. The procedure gets records out of user space SFL1 that was filled with the previous program call. The loop ends when there is no more data in the user space.
- The subfile SUBZIP is filled with the data returned from the iSeries procedure.
- The count is increased by one. It goes back to the start of the DO loop.
- After the loop ends, the code adjusts the count variable.
- WIN2 is made visible showing the subfile with all data received, but only if a customer is found for this ZIP code.
- The Status bar of WIN1 is updated with text indicating how many customers were found for the search.
You need to define a couple of variables used in this code just add the D specifications shown in Example 15-2.

**Example 15-2** Defining variables for the thin client example

```
..1........2........3........4........5........6.....
\* d* counters used to keep track of records loaded into user space
\* d count   s    10u  0 \inz(0)
\* d counti  s    10i  0 \inz(0)
\* d Variable to store user space name
\* d subfile  s     10
\* d Name of qualified library and program to call
\* d pgmname1  s    22
\* d This convinces the compiler to accept the fact that zip is a zoned field:
\* d zip     s     5s  0
..1........2........3........4........5........6.....
```

That’s it. You are done with the action subroutines. Let’s look at the VARPG procedures that call the iSeries program or service program.

**Calling an iSeries program in V5.1.2 or higher**

With VARPG prior to WDS 5.1.2 the call to a server program was implemented with the CALL operation code and rather than CALLP and had to be coded in fixed form. Examples for that have been documented in the first edition of this redbook. To keep the example as simple as possible, the following sections only describe the new way of calling a server program on iSeries.

In WDSc V5.1.2 VARPG added a new enhancement to allow prototyped calls to iSeries programs, this allows to use a CALLP operation code for iSeries program invocation. If you already use VARPG V5.1.2 or higher you can code the program call in the following way. There is no need for the wrapper procedure CALLPGM anymore. Example 15-3 shows the prototype needed and to complete the code shown in Example 15-1 on page 646 for action subroutine WIN1_PSBSEARCH_PRESS.

**Example 15-3** Call iSeries program in VARPG V5.1.2 or higher

```
..1........2........3........4........5........6.....
\* d* Prototype for program call to iSeries Linkage and EXTPGM keywords have
\* d to be specified
\* DNEWCALL PR    LINKAGE(*SERVER)
\* D extpgm(pgmname1)\n\* D pgmname1 is the variable to contain library and program name
\* D pgmname1  s    22
..1........2........3........4........5........6.....
```

The Linkage keyword indicates that this call is to an iSeries server. The CALLP passes the ZIP code to the iSeries program. The iSeries program creates a user space and then fills the user space with data from the database for each customer that is located in ZIP code area.

**Calling a procedure in a service program**

Now, the next procedure, CallServPGM, invoked to retrieve the data out of the user space calling a procedure in a *Service program* on the iSeries is shown in Example 15-4.
Example 15-4  CallServPGM

```
*p callServPGM  B
  d subfile  10  
  d count  10u 0  
  d strucl  like(struc)  
  d QZRUCLSP  c  linkage(*server)  
  d  const('QZRUCLSP')  
  d pgmname  s  20  inz('GETDETAILVLIB6961')  
  d
  d* parameters needed for call system API QZRUCLSP
  d* second parameter procedure name (case sensitive)
  d* structure to specify procedure name to be invoked in service program
  d* structure is needed to be able to delimit procedure name with hex 0
  d* and avoid character translation when sending to iSeries
  d* structure for second procedure
  d expstruc2  ds
    d expname2  4  inz('GetR')  Need hex zero
    d exp1  5i 0 inz(0)
  d* third parameter what kind of return value is passed back from procedure
  d rtvalue  s  10i 0 inz(0)
  d* fourth parameter specify what kind of parameters to be passed to procedure
  d* this one is for GetR procedure
  d parmform2  s  10i 0 dim(3)
  d* fifth parameter number of parameters to be passed to procedure
  d* this one for GetR procedure
  d numparm2  s  10i 0 inz(3)
  d* defines error return parameters for QZRUCLSP API
  d numrec  s  10i 0 inz(0)
  d errorstr  ds
    d er1  10i 0 inz(255)
    d er2  10i 0 inz(255)
  d errorcod  255
D*******************************************************************
C  C eval  parmform2(1)=2
C  C eval  parmform2(2)=2
C  C eval  parmform2(3)=2
C  C call  QZRUCLSP
C  C parm  pgmname
C  C parm  expstruc2
C  C parm  rtvalue
C  C parm  parmform2
C  C parm  numparm2
C  C parm  errorstr
C  C parm  numrec
C* These are the real parameters 3 as specified in variable numparm2
C  C parm  subfile
C  C parm  strucl
C  C parm  count
C  C RETURN
P     E
.*1.2.3.4.5.6.
This looks complicated, but it really isn't. To call a procedure in a Service program without binding it, you need to use the system API QZRUCCLSP. This requires certain parameters so it can do the job and invoke a procedure in a service program directly.

You use seven parameters to describe the procedure you want to call and its interface and then at the end you pass the parameters the procedure expects. It requires the following information:

- **Parameter 1 (pgmname):**
  The name of the service program and the library in which this program is located.

- **Parameter 2 (expstruc2):**
  The name of the procedure to invoke, which is `GetR` in this case. This name has to be null terminated (the procedure name is case-sensitive).

- **Parameter 3 (rtvalue):**
  The data type of the return value of the procedure you invoke, if this procedure returns a value.

- **Parameter 4 (parmform2):**
  The data type of the parameters being passed to the procedure you invoke. For each parameter, you need to specify which type it is. The procedure `GetR` expects 3 parameters, so this is an array with three elements.

- **Parameter 5 (numparm2):**
  The number of parameters being passed. This API allows up to seven parameters. If you have more, use a data structure, which only counts as one but can contain multiple fields. For procedure `GetR` you have to specify three for the number of parameters it expects.

- **Parameter 6 (errorstr):**
  A data structure for error return codes and error messages. This may contain valuable information in case the CALL to the procedure failed.

- **Parameter 7 (numrec):**
  A parameter to indicate whether a return value is passed back by the procedure.

- **Parameters 8, 9, and 10, the “real” parameters to be passed.** In this case, there are three parameters:
  - The user space name (`subfile`)
  - The structure containing the values to be displayed in the subfile (`struc1`)
  - The variable containing the record number for the record to get from the subfile (`count`)

Now let us look at the prototype for this procedure in Example 15-5. Then you are ready to run the VARPG program.

**Example 15-5  The prototype**

```
.*..1.........2........3........4........5........6.....
dCallServPGM      PR
  d                       10
  d                       10u 0

dstruc1               like(struc)
d********************************************
d struc           ds
  d ZIPC                         5s 0
  d custno                       5s 0
  d names                        20
.*..1.........2........3........4........5........6.....
```
The CallServPGM prototype describes the variables to be passed to this procedure:

- The variable containing the user space name.
- The subfile record number to get.
- The structure strucl containing the data from the user space.

The structure being passed consists of the ZIP code, the customer number, and the customer name. These are the values that you need to fill the subfile columns.

### 15.5.6 Taking a look at the thin client application built so far

If you followed the example given so far, the VARPG code should like the one shown in Example 15-6. Note that format lines (such as the first line in Example 15-6) have been included to make identification of the correct column for fixed form statements easier. You should not type lines should into the editor, but rather enter the character f into the prefix area.

**Example 15-6  VisualAge RPG code for the basic thin client example**

```rpg
DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++Comments+++++++
D NEWCALL PR LINKAGE(*SERVER)
D extpgm(pgmname1)
D 5s 0
D*******************************************************************************
d* only use in V5.1 or lower, not used in 5.1.2
d* CallPGM PR
d 10A
d 10A
d 5s 0
d* Program S 10A
d* Library S 10A
d*******************************************************************************
D* Procedure to Call procedure in Service Program to get data out of user space
d CallServPGM PR
d 10
d 10u 0
d struc1 like(struc)
d* counters used to keep track of records loaded into user space:
d count s 10u 0 Inz(0)
d counti s 10i 0 Inz(0)
d* variable to store user space name:
d subfile s 10
d* name of qualified library and program to call:
d pgmname1 s 22
d* That convinces the compiler to accept the fact that zip is a zoned field:
d zip s 5s 0
d*******************************************************************************
d struc ds
d ZIPC 5s 0
d custno 5s 0
d names 20
DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++Comments+++++++
/FREE
//*******************************************************************************
// Window . . : WIN1
// Part . . . : WIN1
// Event . . : CREATE
//
// Description: See “Positioning the cursor on the entry field” on page 646
BEGACT WIN1_WIN1_CREATE;
win1.zip.focus=1;
ENDACT;

//******************************************************************
// Window . . :  WIN1
// Part . . . :  PSBSEARCH
// Event . . :  PRESS
//
// Description: See “Getting data from the server” on page 646
// ************************************************************************

BEGACT WIN1_PSBSEARCH_PRESS;
// Read zip code data from window
read 'win1';

// This is for V5.1.2 or higher if you use earlier versions un-comment the procedure
// Program='CUSINQNEW';
// Library='LIB6961';
// to CallPGM
// CALLP CallPGM (Program:Library:ZIP);
// pgmname1='LIB6961/CUSINQNEW';
// Callp NewCall(ZIP);
exsr getdata;
ENDACT;

************************************************************************
begsr getdata;
// load window if already loaded clear subfile
monitor;
showwin 'win2';
on-error;
clear subzip;
endmon;
// initialize variables
custno=00001; //when customer number returned is 0, end of user space reached
count=1; // counter, count how many customers are found for zip code
counti = 1; // Second counter, count gets changed when no more records found
subfile= 'SFL1'; // user space name
// Loop to get all records out of user space
dow count > 0;
CALLP CallServPGM (subfile:count:struc); // get data from user space
// if there is a valid customer add to the subfile
if count > 0;
write subzip; // write record to subfile
count +=1; // add 1 to count
counti = count;
endif;
endo;
// update status bar on first window
counti -=1; // re-adjust counti
If counti > 0
win2.win2.visible=1;
win1.status.sblabel = %char(counti) + ' Customers found in zip area ' + %char(zip);
else;
  // if there are no customers found don't show subfile
  win1.status.sblab=' No customers found in zip area '+%char(zip);
  clswin 'win2';
endif;
endsr;
/END-FREE

PName+++++++++++B...................Keywords+++++++++++++++++++++++++++++Comments++++++++
pCallServPGM      B
  d                  PI
  dsubfile                        10
  dcount                          10u 0
  dstruc1                        10u 0
  dstruc1                               like(struc)

  d struc ds
  d   ZIPC                         5s 0
  d   custno                       5s 0
  d   names                       20
  dQZRUCCLSP        c                   linkage(*server)
    d                                     const('QZRUCCLSP')
  d
  dpgmname          s             20    inz('GETDETAILVLIB6961')
  d* parameters needed for call system API QZRUCCLSP
  d* second parameter procedure name (case sensitive)
  d* structure to specify procedure name to be invoked in service program
  d* structure is needed to be able to delimit procedure name with hex 0
  d* and avoid character translation when sending to iSeries
  d* structure for second procedure
  dexpstruc2               ds
  dexpname2                        4    inz('GetR')
  dexp1                            5i 0 inz(0)                               Need hex zero
  d* parameters needed for call system API QZRUCCLSP
  d* third parameter  return value what kind of return value is passed back from procedure
  drtvalue          s             10i 0 inz(0)
  d* fourth parameter specify what kind of parameters to be passed to procedure
  d* this one is for GetR procedure
  dparmform2        s             10i 0 dim(3)
  d*  fifth parameter number of parameters to be passed to procedure
  dnumparm2         s             10i 0 inz(3)
  d*  defines error return parameters for QZRUCCLSP API
  dnumrec           s             10i 0 inz(0)
  derrorstr         ds
    d er1                           10i 0 inz(255)
    d er2                           10i 0 inz(255)
  d errorcod                     255
D
D

CSRNO1Factor1++++++Opcode(E)+Extended-factor2+++++++++++++++++++++++++++++Comments+++
Building and running the application

Perform these steps to complete the example:

1. Click **Build** to create the application.

2. Click **Run** to run the application. The library that comes with this redbook, LIB6961, contains the programs and procedures needed to work with this VARPG program. This library must be in the library list for the job that is created to access the RPGIV programs. Make sure that the user profile you use adds library LIB6961 to the library list.

3. In the Customer Master Inquiry window shown in Figure 15-38, enter 55901 in the ZIP code entry field.

4. Click the **List customers** button or press the Enter key.

![Customer Master Inquiry](image)

*Figure 15-38 First window with Search criteria*

The window with the subfile similar to Figure 15-39 appears if any customers are found.
Notice the status bar in WIN1 (Figure 15-40) has the summary information.

5. Click the X button on the Window title bar of the Main window to exit the application.

15.5.7 Further enhancing the thin client example

Now that the basic function works, let's add some easy enhancements.

Sorting subfile data when clicking a column heading

Write code to sort the data by subfile column when the user clicks the column headings, you use the column select event (COLSELECT) that gets fired when the user clicks the subfile heading. In the GUI designer:

1. Double-click WIN2 in the project view to open it in a design window.
2. Right-click the subfile part in WIN2 and select Event-> COLSELECT.
3. In the action subroutine, add the following code (the BEGACT and ENDACT statements have already been added for you automatically):

   ```
   BEGACT WIN2_SUBZIP_COLSELECT;
   win2.subzip.sortasc = %colnumber;
   ENDACT;
   ```

4. Add a definition for variable %COLNUMBER in the D spec area, define it as 5 0. The variable %colnumber is an event attribute for the COLSELECT event.

   ```
   DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++++++
   d %colnumber s 5 0
   ```

   It contains the column identifier of the column heading that was under the mouse cursor when the click happened.
5. Build and run the application.

6. Try this feature:

   When you click the **Name** heading in the subfile, it should now display the data sorted by name. If you click the **Number** heading, the data should appear sorted by customer number.

7. Exit the application using the Window **X** button.

**Using a pop-up menu to work with subfile records**

Now you can activate the pop-up menu. You use the popup event of the subfile part to make a **popup menu** visible. In the GUI designer:

1. Double-click **WIN2** in the project view, to open this window as a design window.
2. Right-click the **subfile** part in WIN2 and select **Event-> POPUP**.
3. In the action subroutine, add the following code:

   ```
   BEGACT WIN2_SUBZIP_POPUP;
   win2.subzip.selectitem=%index;
   win2.popsub.visible=1;
   ENDACT;
   ```

4. Define the variable **%index** in the D specification area as 5 0.

   ```
   DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++
   d %index      s          5 0
   ```

   The event attribute **%index** returns the number of the row at which the cursor was positioned when the right-click occurred.

This code selects the subfile row the cursor was on when the user right-clicked the subfile. Then it displays the pop-up menu you defined before for this window.

**Creating the logic for browsing customer data**

Now you must code what should happen when the user selects one of the four menu items that you created for this **pop-up menu**.

Let's code the browse action first:

1. In the project window, expand **WIN2** and expand the **pop-up menu**. Make sure you use the tree view in the project window, menus are not supported in the design window.
2. Right-click the pop-up menu item **MNIBROWSE** and select **Event-> MENUSELECT**.

   In the action subroutine, add the code shown in Example 15-7.

**Example 15-7   Code to add to the action subroutine**

```
BEGACT WIN2_MNIBROWSE_MENUSELECT;

  taskhist= 'getRecord'; // Browse selected
  reads subzip; // read selected record from subfile
  Cust=custno; // call service program to get customer detail data
  CALLP CallServPGMA (strucdet : taskhist);

  monitor;
  showwin 'win3'; // Show window with detail data
  on-error;
    // if window already loaded give this window focus so it show on top
    %setatr('win3': 'win3': 'focus')=1;
  endmon;
  write 'win3'; // Write data to detail window
```
// Since it is browse mode only, disable entry fields and color yellow
CALLP fldenable(0 : *yellow);
ENDACT;

The code shown in Example 15-7 performs the following actions:

- Sets the variable taskhist to indicate that the browse option was selected. This value is the name of the procedure to be invoked on the iSeries.
- Reads the selected record from the subfile.
- Calls a procedure to get the detail data for this customer from the iSeries server. Call is to a VARPG procedure, which will then issue the CALLP to a Service Program which invokes the procedure identified in parameter taskhist.
- Loads the third window WIN3 that you created in a prior step.
- Writes the data to WIN3.
- Disables all entry fields on WIN3, since this is browse mode only. It also changes the background of these entry fields to the color yellow to indicate that they are not input capable. To support this function, you create the fldenable procedure in the next steps.

3. Add a the following D specification for variable taskhist:

   DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++
   d taskhist        s              9

Procedure fldenable has two parameters:

- The value for setting the enabled attribute for the entry fields
- The color name for setting the background color of the entry fields

4. Enter the code for the procedure as shown in Example 15-8.

   Example 15-8  Procedure fldenable
   p fldenable       B
   d                 PI
   d enable                         5U 0 Value
   d color                          5I 0 Value
   d*
   /free
   win3.cust.enabled = enable;
   win3.name.enabled = enable;
   win3.addr.enabled = enable;
   win3.city.enabled = enable;
   win3.state.enabled = enable;
   win3.zip.enabled = enable;
   win3.srhcod.enabled = enable;
   win3.custyp.enabled = enable;
   win3.arbal.enabled = enable;
   // To indicate entry (not) allowed, change color of fields
   win3.cust.backcolor = color;
   win3.name.backcolor = color;
   win3.addr.backcolor = color;
   win3.city.backcolor = color;
   win3.state.backcolor = color;
   win3.zip.backcolor = color;
   win3.srhcod.backcolor = color;
   win3.custyp.backcolor = color;
   win3.arbal.backcolor = color;
The procedure fldenable simply takes the parameters for the enabled and backcolor attributes and applies them to all entry fields on the window. Don’t forget to add the Prototype for this procedure.

Example 15-9

```plaintext
d fldenable PR
  d enable 5u 0 value
  d color 51 0 value
```

The procedure CallServPGMA shown in Example 15-10 is almost the same as the CallServPGM procedure previously described in Example 15-4 on page 649 and used to call a service program. Only the number of parameters passed to the iSeries procedure is different instead of three parameters you use one parameter, this parameter is the data structure containing all customer data.

Example 15-10  CallServPGMA

```plaintext
pCallServPGMA B
  d struc2 like(strucdet)
  d proname 9 value
  d
  dQZRUCLSP c linkage(*server)
  d const('QZRUCLSP')
  d
  dpgmname s 20 inz('GETDETAILVLIB6961')
  d
d* second parameter procedure name (case sensitive)
d* structure to specify procedure name to be invoked in service program
d* structure is needed to be able to delimit procedure name with hex 0
d* and avoid character translation when sending to As/400
d* structure for second procedure
dexpstruc2 ds
dexpname2 9 inz
  d* procedure name gets passed as parameter
dexp1 5i 0 inz(0) Need hex zero
  d* parameters needed for call system API QZRUCLSP
d* third parameter return value what kind of return value is passed back from procedure
drtvalue s 10i 0 inz(0)
d* fourth parameter specify what kind of parameters to be passed to procedure
d* this one is for 1 parameter, instead of 3 in the previous
dparmform2 s 10i 0 dim(1)
  d*
d* fifth parameter number of parameters to be passed to procedure
dnumparm2 s 10i 0 inz(1)
  d*
d* defines error return parameters for QZRUCLSP API
dnumrec s 10i 0 inz(0)
  derrorstr ds
d er1 10i 0 inz(255)
d er2 10i 0 inz(255)
```
The only difference from the procedure CallServPGM, you used before to call a service program, is the fact that you provide the procedure name as a parameter and that the number of parameters that the procedure expects is one.

The structure you pass is an externally described structure based on the record format of the file that is accessed by the procedure on the iSeries:

```
DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++
D strucdet    e ds                  extname(MLGMSLT2:CUST)
D strucbackup     ds                  likeds(strucdet)
```

Add a prototype for CALLSERVPGMA

```
D CallServPGMA    PR
D struc2                              like(strucdet)
```

5. Build and run the application.

6. Right-click the subfile and select Browse customer detail as shown in Figure 15-41.

```
Figure 15-41 Pop-up menu for subfile
```

The Customer Master List window (Figure 15-42) opens.
7. You might notice that the OK button does not do anything. To fix this, cancel the application by clicking the X button on the main window title bar.

8. In the GUI designer, bring up WIN3, right-click the OK push button and add an action subroutine for the PRESS event. Add the following code:

```
BEGACT WIN3_PSBOK_PRESS;
// take the action that was selected in the menu Create/Browse/Edit select;
// browse
when taskhist = 'getRecord';
clswin 'win3';
endsl;
ENDACT;
```

At the moment the select operation code is not very useful but as you add more function, you will need different logic depending what task the enduser selected from the popup menu. In the next task you will add the Edit customer data option and add another When operation code.

For browse mode, the window is simply closed by using the CLSWIN operation code.

Now you can implement the other pop-up menu options.

**Creating logic to edit customer data**

The edit option works similar to the browse option, but you will allow editing of data on the customer detail window and then write the changed values back to the iSeries database. In the GUI designer:

1. In the project window, expand WIN2 and expand the POPSUB pop-up menu in the tree view.
2. Right-click MNIEDIT from the pop-up menu and select Event-> MENUSELECT.
3. In the action subroutine, add the code shown in Example 15-11.
Example 15-11  Code to add to the action subroutine

BEGACT WIN2_MNIEdit_MENUSELECT;
// Edit selected

taskhist= 'edtRecord';
//read selected record from subfile
reads subzip;
// call service program to get customer detail data
Cust=custno;
CALLP CallServPGMA (strucdet : 'getRecord');
// Show window with detail data
monitor;
showwin 'win3';
on-error;
// if window already loaded give this window focus so it show on top
%setatr('win3':'win3':'focus')=1;
endmon;
// Write data to detail window
write 'win3';
// Since it is edit mode enable entry fields and color white
CALLP fldenable(1 : *white) ;
ENDACT;

The code is similar to the code you added in the browse menu option. The procedure you invoke on the iSeries is again getRecord, but the taskhist is changed to indicate that it is an edit action. To enable the entry fields on the customer detail window and to change the background color to white, you call procedure fldenable with the corresponding parameter values.

You need to add logic to the action subroutine for the PRESS event from the OK button on WIN3.

4. Add the When logic for edit mode (Example 15-12) to the Select structure (you already added the logic for browse mode) in the previous task.

Example 15-12  When logic

// take the action that was selected in the menu Create/Browse/Edit
BEGACT WIN3_PSBOK_PRESS;

 select;
 // browse
 when taskhist = 'getRecord';
 clswin 'win3';
 // edit
 when taskhist = 'edtRecord';
 strucbackup = strucdet;
 READ 'win3';
 if strucbackup <> strucdet;
 CALLP CallServPGMA (strucdet : taskhist);
 Endif;
 clswin 'win3';
 endsl;
ENDACT;

The original data is compared to the new window data. If no changes were made, the window is closed. Otherwise procedure edtRecord is called on the iSeries to update the database with the new data.
Creating logic to delete an existing customer

The delete an existing customer option works differently since the data does not have to appear in WIN3. In the GUI designer:

1. In the project window, expand WIN2 and expand the POPOSUB pop-up menu.
2. Right-click MNDELETE from the pop-up menu item and select Event-> MENUSELECT.
3. In the action subroutine, add the following code:

   BEGACT WIN2_MNDELETE_MENUSELECT;
   // Delete selected call procedure to delete customer record
   READS subzip;
   Cust = CUSTNO;
   CALLP CallServPGMA (strucdet : 'delRecord');
   ENDACT;

   The selected record is read from the subfile and procedure delRecord on the iSeries server is called. If you want, you can easily add a message box that asks the enduser, “Do you really want to delete this record” and depending which button is pressed on the message box go ahead with the delete action or cancel it. You can use the DSPLY operation code to display a message box.

Creating logic to create a new customer

The create new customer option works similar to the edit option. In the GUI designer:

1. In the project window, expand WIN2 and expand the POPOSUB pop-up menu.
2. Right-click MNICREATE from the pop-up menu and select Event-> MENUSELECT.
3. In the action subroutine, add the following code:

   BEGACT WIN2_MNICREATE_MENUSELECT;
   // Create selected, just prompt for new data
   taskhist= 'newRecord';
   monitor;
   showwin 'win3';
   on-error;
   %setatr('win3':'win3':'focus')=1;
   endmon;
   clear 'win3';
   // Since it is create mode enable entry fields
   CALLP fldenable(1 : *white) ;
   win3.cust.focus=1;
   ENDACT;

   4. The window WIN3 is shown and cleared. Now the action subroutine for the OK button needs to be changed. Add another When operation code to the Select structure. Only the additional code is shown here:

   // create
   when taskhist = 'newRecord';
   READ 'win3';
   CALLP CallServPGMA (strucdet : taskhist );
   clswin 'win3';

   For this option, procedure newRecord on the iSeries server is invoked.

   When you created WIN1, you added a second push button PSBCREATE to this window to allow the enduser to create a new customer directly from this window. In order to make this push button work, you need to connect the PRESS event to the MNINew action subroutine from the popup menu on window WIN2.

5. In the GUI designer, select Project-> Edit Action subroutine and the Action subroutines dialog as shown in Figure 15-43 appears.
6. In the Action subroutine dialog, complete the following tasks:
   a. Select WIN1 in the Windows list.
   b. Select PSBCREATE in the Parts list.
   c. Select PRESS in the events list.
   d. Select WIN2_MNICREATE_MENUSELECT in the Action subroutine list.
   e. Click the Create link button.

   This links the PRESS event to the already existing MNICREATE action subroutine and invokes this subroutine for both events.

7. Build and run the application.

   To improve the application, you could clear the subfile using the CLEAR operation code and then refresh the subfile content by invoking procedure CALLPGM and running subroutine GETDATA after your delete, create, or edit action is executed. This would update the subfile with the new database information.

15.5.8 The complete thin client application example

If you followed the example given so far, the VARPG code should like the one shown in Example 15-6. Note that format lines (such as the first line in Example 15-6) have been included to make identification of the correct column for fixed form statements easier. You should not type lines should into the editor, but rather enter the character f into the prefix area.

Example 15-13 Example code for the complete thin client application

```
*********************************************************************
*                                                                  *
* Program ID . . : CUSTMAST.VPG (or CUST6961.VPG)                   *
*                                                                  *
* Description . : See                                              *
*   “Building a thin client application with VARPG” on page 633*    
```
* of the redbook
* WebSphere Development Studio Client for iSeries V5.1.2 *
* Function . . . : *
* Messages . . . : *
* Files . . . . : *
* Input . . . . : *
* Output . . . . : *

*********************************************************************
*
H
H*EXE
H*NOMAIN
DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++Comments++++++++
D NEWCALL PR LINKAGE(*SERVER)
D extpgm(pgmname1)
D 5s 0
D**************************************************************************
d* only use in V5.1 or lower, not used in 5.1.2
d*CallPGM PR
d* 10A
 d* 10A
 d* 5s 0
d*Program S 10A
d*Library S 10A

d**************************************************************************
D* Procedure to Call procedure in Service Program to get data out of user space
D CallServPGM PR
D 10
D 10u 0
D struct1 like(struc)
D CallServPGMA PR
D struct2 like(strucdet)
D 9 value
D fieldenable PR
D enable 5u 0 value
D color 5i 0 value

d**************************************************************************
d %colnumber s 5 0
d %index s 5 0
d* counters used to keep track of records loaded into user space:
d count s 10u 0 Inz(0)
d counti s 10i 0 Inz(0)
d* variable to store user space name:
d subfile s 10
D name of qualified library and program to call:
d pgmname s 22
D* That convinces the compiler to accept the fact that zip is a zoned field:
d zip s 5s 0
d**************************************************************************
D struct ds
D ZIPC 5s 0
D custno 5s 0
D names 20
DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++Comments++++++++
DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++Comments++++++++++
d strucdet e ds extname(MLGML2:cust)
d strucbackup ds likeds(strucdet)
d taskhist s 9
/FREE

//******************************************************************
// Window . . : WIN1
// Part . . . : WIN1
// Event . . : CREATE
//
// Description: See “Positioning the cursor on the entry field” on page 646
//
//******************************************************************

BEGACT WIN1_WIN1_CREATE;
  win1.zip.focus=1;
ENDACT;

//******************************************************************
// Window . . : WIN1
// Part . . . : PSBSEARCH
// Event . . : PRESS
//
// Description: See “Getting data from the server” on page 646
//
//******************************************************************

BEGACT WIN1_PSBSSEARCH_PRESS;
  // Read zip code data from window
  read 'win1';

  // This is for V5.1.2 or higher if you use earlier versions un-comment the procedure
  // Program='CUSINEW';
  // Library='LIB8961';
  // to CallPGM
  // CALLP CallPGM (Library:Program:ZIP);
  pgmname1='LIB8961/CUSINEW';
  Callp NewCall(ZIP);
  exsr getdata;
ENDACT;

 ['/*************************************************************/
  begsr getdata;
  // load window if already loaded clear subfile
  monitor;
  showwin 'win2';
  on-error;
  clear subzip;
  endmon;
  // initialize variables
  custno=00001; //when customer number returned is 0, end of user space reached
  count=1; // counter, count how many customers are found for zip code
  counti = 1; // Second counter, count gets changed when no more records found
  subfile= 'SFL1'; // user space name
  // Loop to get all records out of user space
  dow count > 0 ;
  CALLP CallServPGM (subfile:count:struc); // get data from user space
  // if there is a valid customer add to the subfile
if count > 0;
write subzip; // write record to subfile
count +=1; // add 1 to count
counti = count ;
endif;
enddo;
// update status bar on first window
counti -=1; // re-adjust counti
if counti > 0 ;
win2.win2.visible=1;
win1.status.sblabel = %char(counti) + ' Customers found in zip area ' + %char(zip);
else;
// if there are no customers found don't show subfile
win1.status.sblabel= ' No customers found in zip area ' + %char(zip);
clos 'win2';
endif;
endsr;

/*---------------------------------------------*/
/* Window .. : WIN2                             */
/* Part . . . : SUBZIP                           */
/* Event  . . : COLSELECT                        */
/* Description: See “Sorting subfile data when clicking a column heading” on page 655 */
/*---------------------------------------------*/

BEGACT WIN2_SUBZIP_COLSELECT;
win2.subzip.sortasc = %colnumber;
ENDACT;

/*---------------------------------------------*/
/* Window .. : WIN2                             */
/* Part . . . : SUBZIP                           */
/* Event  . . : POPUP                            */
/* Description: See “Using a pop-up menu to work with subfile records” on page 656 */
/*---------------------------------------------*/

BEGACT WIN2_SUBZIP_POPUP;
win2.subzip.selectitem=%index;
win2.popsub.visible=1;
ENDACT;

/*---------------------------------------------*/
/* Window .. : WIN2                             */
/* Part . . . : MNIBROWSE                        */
/* Event  . . : MENUSELECT                       */
/* Description: “Creating logic to create a new customer” on page 662 */
/*---------------------------------------------*/

BEGACT WIN2_MNIBROWSE_MENUSELECT;
taskhist= 'getRecord'; // Browse selected
reads subzip;                         // read selected record from subfile
Cust=custno;          // call service program to get customer detail data
CALLP CallServPGMA (strucdet : taskhist);
monitor;
showwin 'win3';                           // Show window with detail data
on-error;
// if window already loaded give this window focus so it show on top
%setatr('win3':'win3':'focus')=1;
endmon;
write 'win3';                             // Write data to detail window
// Since it is browse mode only, disable entry fields and color yellow
CALLP fldenable(0 : *yellow) ;
ENDACT;

//******************************************************************
// Window . . :  WIN3
// Part . . . :  PSB0000052
// Event  . . :  PRESS
//
// Description: See “Creating the logic for browsing customer data” on page 656
// and “Creating logic to edit customer data” on page 660
// and “Creating logic to create a new customer” on page 662
//******************************************************************
BEGACT WIN3_PSB0000052_PRESS;
// take the action that was selected in the menu Create/Browse/Edit
select;
// browse
when taskhist = 'getRecord';
clswin 'win3';
// edit
when taskhist = 'edtRecord';
strucbackup = strucdet;
READ 'win3';
if strucbackup <> strucdet;
CALLP CallServPGMA (strucdet : taskhist);
Endif;
clswin 'win3';
// create
when taskhist = 'newRecord';
READ 'win3';
CALLP CallServPGMA (strucdet : taskhist );
c1sw1 'win3';
ends1;
ENDACT;

//******************************************************************
// Window . . :  WIN2
// Part . . . :  MNIEDIT
// Event  . . :  MENUSELECT
//
// Description:
//
//******************************************************************
BEGACT WIN2_MNIEDIT_MENUSELECT;
// Edit selected
taskhist= 'edtRecord';
//read selected record from subfile
reads subzip;
// call service program to get customer detail data
Cust=custno;
CALLP CalIServPGMA (strucdet : 'getRecord');
// Show window with detail data
monitor;
showwin 'win3';
on-error;
// if window already loaded give this window focus so it show on top
%setatr('win3':'win3':'focus')=1;
endmon;
// Write data to detail window
write 'win3';
// Since it is edit mode enable entry fields and color white
CALLP fldenable(1 : *white) ;
ENDACT;

//******************************************************************
// Window . . : WIN2
// Part . . . : MNIDELETE
// Event . . : MENUSELECT
//
// Description: See “Creating logic to delete an existing customer” on page 662
//
//******************************************************************
BEGACT WIN2_MNIDELETE_MENUSELECT;
// Delete selected call procedure to delete customer record
READS subzip;
Cust = CUSTNO;
CALLP CalIServPGMA (strucdet : 'delRecord');
ENDACT;

//******************************************************************
// Window . . : WIN2
// Part . . . : MNICREATE
// Event . . : MENUSELECT
//
// Description: “Creating logic to create a new customer” on page 662
//
//******************************************************************
BEGACT WIN2_MNICREATE_MENUSELECT;
// Create selected, just prompt for new data
taskhist= 'newRecord';
monitor;
showwin 'win3';
on-error;
%setatr('win3':'win3':'focus')=1;
endmon;
clear 'win3';
// Since it is create mode enable entry fields
CALLP fldenable(1 : *white) ;
win3.cust.focus=1;
ENDACT;
/END-FREE
p fldenable B
d enable 5U 0 Value
d color 5I 0 Value
d*
/free
  win3.cust.enabled = enable;
  win3.name.enabled = enable;
  win3.addr.enabled = enable;
  win3.city.enabled = enable;
  win3.state.enabled = enable;
  win3.zip.enabled = enable;
  win3.srhcod.enabled = enable;
  win3.custyp.enabled = enable;
  win3.arbal.enabled = enable;
//  To indicate entry (not) allowed, change color of fields
  win3.cust.backcolor = color;
  win3.name.backcolor = color;
  win3.addr.backcolor = color;
  win3.city.backcolor = color;
  win3.state.backcolor = color;
  win3.zip.backcolor = color;
  win3.srhcod.backcolor = color;
  win3.custyp.backcolor = color;
  win3.arbal.backcolor = color;
RETURN;
/end-free
p CallServPGM B
d subfile 10
dcount 10u 0
dstruc c linkage(*server)
***********************
d struc ds
d ZIPC 5s 0
d custno 5s 0
d names 20
dQZRUCLSLP c linkage(*server)
d const('QZRUCLSLP')
d
dpgmname s 20 inz('GETDETAILVLIB6961')
d* parameters needed for call system API QZRUCLSLP
d* second parameter procedure name (case sensitive)
d* structure to specify procedure name to be invoked in service program
d* structure is needed to be able to delimit procedure name with hex 0
  d and avoid character translation when sending to iSeries
  d* structure for second procedure
dexpstruc2 ds
dexpname2 4 inz('GetR')
dexp1 5i 0 inz(0) Need hex zero
d* parameters needed for call system API QZRUCLSLP
d* third parameter return value what kind of return value is passed back from procedure
drtvalue s 10i 0 inz(0)
d* fourth parameter specify what kind of parameters to be passed to procedure
d* this one is for GetR procedure
dparmform2 s 10i 0 dim(3)
d* fifth parameter number of parameters to be passed to procedure
dnumparm2 s 10i 0 inz(3)
d* defines error return parameters for QZRUC LS API
dnumrec s 10i 0 inz(0)
derrorstr ds
d er1 10i 0 inz(255)
d er2 10i 0 inz(255)
derrorcod 255

c                   eval      parmform2(1)=2
c                   eval      parmform2(2)=2
c                   eval      parmform2(3)=2
C                   call      QZRUC LS
C                   parm                    pgmname
C                   parm                    expstruc2
C                   parm                    rtvalue
C                   parm                    parmform2
C                   parm                    numparm2
C                   parm                    errorstr
C                   parm                    numrec
C                   parm                    subfile
C                   parm                    struc1
C                   parm                    count
C*  These are the real parameters as specified in variable numparm2
C                   parm                    error
C                   parm                    disp msgl
C                   RETURN
P                 E
pCallServPGMA     B

CSRNO1Factor1+++++++Opcode(E)+Extended-factor2+++++++++++++++++++++++++++++++++++++++++++++++++++++
C* These are the real parameters as specified in variable numparm2
C                   eval      parmform2(1)=2
C                   eval      parmform2(2)=2
C                   eval      parmform2(3)=2
C                   call      QZRUC LS
C                   parm                    pgmname
C                   parm                    expstruc2
C                   parm                    rtvalue
C                   parm                    parmform2
C                   parm                    numparm2
C                   parm                    errorstr
C                   parm                    numrec
C                   parm                    subfile
C                   parm                    struc1
C                   parm                    count
C                   errorcod      disp msgl
C                   RETURN
P                 E
pCallServPGMA     B

DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++Comments++++++++
d struc2 like(strucdet)
d proname 9 value
C                   linkage(*server)
d                   const('QZRUC LS')
d                   const('QGETDETAILVLIB6961')
d* second parameter procedure name (case sensitive)
d* structure to specify procedure name to be invoked in service program
C                   linkage(*server)
d                   const('QZRUC LS')
d                   const('QGETDETAILVLIB6961')
d* and avoid character translation when sending to As/400
C                   linkage(*server)
d                   const('QZRUC LS')
d                   const('QGETDETAILVLIB6961')
d* structure for second procedure
C                   linkage(*server)
d                   const('QZRUC LS')
d                   const('QGETDETAILVLIB6961')
d* parameters needed for call system API QZRUC LS
C                   linkage(*server)
d                   const('QZRUC LS')
d                   const('QGETDETAILVLIB6961')
d* third parameter return value what kind of return value is passed back from procedure
drtvalue s 10i 0 inz(0)
d* fourth parameter specify what kind of parameters to be passed to procedure
d* this one is for 1 parameter, instead of 3 in the previous
dparmform2 s 10i 0 dim(1)
15.5.9 The backend programs running under OS/400 or I5/OS

The following section briefly describe the backend programs written in ILE RPG which run as counter part to the VARPG thin client under OS/400 or I5/OS.

Program CUSINQNEW

Let's have a look at the CUSINQNEW program first, this is the program that gets called when the Search button is pressed. It binds CUSTINQSFL. Example 15-14 shows the source code of the program.

Example 15-14  Program CUSINQNEW

D/copy LIB6961/WEBCOMP01,R6961PROTO
* *
* Program variables
*------------------------------------------------------------*
DSubfile   S    10A   Inz('SFL1')
DZips      S    5S    0
DSetup     S    1    Inz(*off)
*------------------------------------------------------------*
* Define input parameter
C  *Entry    Plist
C  Parm     Zips
* *
* Set the component name and initialize the subfile
The code calls procedure INITFirst, which is located in Service program CUSTINQSL. Procedure INITFirst creates the userspace that stores the customer records found for the ZIP code requested, INITFirst then invokes another procedure to fill the userspace.

Service program CUSTINQSL
Example 15-15 shows the CUSTINQSL service program.

Example 15-15 Program CUSTINQSL

```
H NoMain
*---------------------------------------------------------------
* Define the logical file (view by ZIP)
FMLGMSTL1 IF E K Disk
*---------------------------------------------------------------
* Define the prototypes for the procedures in the subfile
* handling service program, QDTSSFL
D/copy LIB6961/WEBCOMP01,SUBFILEPR2
*---------------------------------------------------------------
* Define the prototypes for the procedures in this program
D/COPY LIB6961/WEBCOMP01,R6961PROTO
*---------------------------------------------------------------
* Define other program variables
D rc S 10I 0
D count S 10U 0
D len S 10U 0 Inz(%Size(Record))
*---------------------------------------------------------------
* Structure to define a subfile record
D Record DS
D ZIP 5S 0
D CUST 5S 0
D NAME 20A
D Reserved S 10A
P INITFirst B Export
d INITFirst pi
  d Subfile 10A
  d Zips 5S 0
  d Setup 1
d*
  c EVAL subfile='SFL1'
  c If setup = *off
  c Eval setup = *on
  c Eval len = %Size(Record)
  c *LOVAL Setll CUSREC
  c Eval rc = initSF(Subfile:
  c %ADDR(Reserved) :len)
  c Eval rc = clearSF(Subfile)
  c CallP FillSFL(Subfile:Zips)
  c EndIf
  c Return
*---------------------------------------------------------------
P INITFirst E
```
The user space that contains the subfile data is created with the name SFL1. The initSF procedure is supplied by IBM, it creates the user space. Procedure ClearSF removes any data in the user space.

The FillSFL procedure is also part of the CUSTINQSFL service program (see Example 15-16).

Example 15-16  Procedure FillSFL

```
P FillSFL      B                   EXPORT
D FillSFL     PI
D Subfile     10A
D Zips        5S 0
*
C             Eval      Count = 1
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(Subfile :
C                     %ADDR(Record) :len)
C           Eval      Count = Count + 1
C Zips        ReadE     CUSREC
C           EndDo
*
PFillSFL      E
```

This procedure reads through the customer file using the ZIP code that was specified in the user interface as a key. It fills the user space SFL1 with the data, using the appendSF procedure that is part of the QDTSSFL service program.

If no more records matching the ZIP code are found, the procedure returns. The data in the user space is ready to be displayed in the subfile on WIN2.

This is the logic on the iSeries server that is executed with the call to program CUSINQNEW from the VARPG action subroutine when the PRESS event of the SEARCH push button is triggered.

Service program GETDETAILV

The VARPG program then calls procedure GetR in Service program GETDETAILV shown in Example 15-17.

Example 15-17  The F and D specs for GETDETAILV

```
H NOMAIN
* Define the 'by CUST' logical file
FMLGMSTL2  UF A E           K DISK
*---------------------------------------------------------------
* Define the prototypes for the procedures in the subfile
* handling service program, QDTSSFL
D/copy LIB6961/WEBCOMP01,SUBFILEPR2
D***********************************************************
D* D/copy LIB6961/WEBCOMP01,R6961PRGET
D***********************************************************
* Record structure for READC
D Record     DS
D xZIP       5S 0
```
The Procedure `GetR` shown in Example 15-18, uses the chainSF procedure in QDTSFSL to read a subfile record. The record number is specified in the variable count. Remember, there is a DOW loop in the VARPG program that invokes `GetR` to read a record from the user space until there is no more data in the user space SFL1. Each record is written to the subfile, this is how you fill the VARPG subfile. It is a little different from the Web tools subfile, which accesses the user space directly.

Example 15-18  The `GetR` procedure

```plaintext
P GetR B EXPORT
D GetR PI
D SUBFILE 10 like(record)
D record1 like(record)
D count IOU 0
D***************************************************************
D return1 s IOI 0
C Eval return1 = chainSF(SUBFILE : %addr(record1) : size : count)
C If return1 = -1
C Eval count= 0
C Endif
C return
C*
P GetR E
```

The rest of the procedures access the customer master file and read, update, delete, and write records depending on the menu options the user selects from the VARPG pop menu. Here are the four procedures:

- **getRecord** procedure (Example 15-19): This procedure reads a record based on the customer number. It puts the record data into data structure Recordcont, which is passed back to the calling program.

Example 15-19  Procedure `getRecord`

```plaintext
P getRecord B Export
D getRecord PI
D detaildata Likeds(currRecord)
D***************************************************************
D Recordcont DS Likerec(cusrec:*input)
D***************************************************************
D currcust s 5s 0
C* get key info and chain to file
C eval currcust=detaildata.cust
C currcust Chain(N) CUSREC Recordcont 99
C C* move data to structure to return to caller
C If *in99 = *off
```
Chapter 15. Introduction to VisualAge RPG

Example 15-20  Procedure newRecord

```
P newRecord       B                                Export
D newRecord       PI
D detaildata     Likeds(currRecord)
D
D*****************************************************************
D Recordcont      DS                  Likerec(cusrec:*output)
D*****************************************************************
D currcust        s              5s 0
C* get key info and chain to file
C                   move      detaildata    Recordcont
C                   Write     CUSREC        Recordcont             99
C
C* Problem encountered
C                   If        *in99 = *on
C                   Eval      detaildata.cust=99999
C                   EndIf
C*
C                   Return
C*                     
P newRecord       E
```

Example 15-21  Procedure delRecord

```
P delRecord       B                                Export
D delRecord       PI
D detaildata     Likeds(currRecord)
D
D*****************************************************************
D currcust        s              5s 0
C* get key info and chain to file
C                   eval      currcust=detaildata.cust
C                   Delete    CUSREC                             99
C                   Return
P delRecord       E
```

**newRecord** procedure (Example 15-20): Procedure newRecord adds a record to the database, writing the content of data structure Recordcont to the database. Recordcont is filled with the content of data structure detaildata, which is passed in from the VARPG program.

**delRecord** procedure (Example 15-21): Procedure delRecord uses the customer number passed in to delete the customer record.
**edtRecord** procedure (Example 15-22): Procedure edtRecord reads the record to be updated. Then it writes the new data to the record by using *OUTPUT structure Recordcont.

### Example 15-22 Procedure editRecord

```plaintext
P edtRecord          B                                Export
D edtRecord          PI                                Likeds(currRecord)
D detaildata                          Likeds(currRecord)
D
D******************************************************************************
D Recordcont          DS                  Likerec(cusrec:*INPUT)
D Recordcont1         DS                  Likerec(cusrec:*OUTPUT)
D******************************************************************************
D currcust        s              5s 0
C* get key info and chain to file
C                      eval      currcust=detaildata.cust
C  currcust        Chain       CUSREC        Recordcont    99
C                      move      detaildata    Recordcont1
C* Update the record with new data
C                      Update    CUSREC        Recordcont1    99
C* Problem encountered
C                      If         *in99 = *on
C                      Eval      detaildata.cust=99999
C                      EndIf
C*
C  Return
C*
P edtRecord          E
```

**Note:** This note applies to re-using this Service Program in the iSeries Web Tooling exercise. Procedure GetSelected, that is also part of Service Program GETDETAILV, is not shown in this chapter. You could use procedure GetSelected in Service Program GETDETAILV to get the detailed customer information, instead of using program GETDETAIL in the Web Tools exercise.

To re-use the GETDETAILV service program from a Web user interface, you need to invoke procedure GetSelected in Service program GETDETAILV when using the interaction wizard. The signature for procedure GetSelected is the same as the signature for program GETDETAIL. This change would allow you to use one service program to serve the WebTools front end as well as the VARPG frontend.

These are all the programs needed on the iSeries server. You are done with this application. If you provide procedures in Service programs, you can also easily create a Web service interface for these procedures using the wizards in WDSC. Writing the server side code in procedures that are then accessible from any platform gives you the highest flexibility and chances of re-usability in your environment.

Keep in mind that you must write thin client applications in order to achieve re-usability. VARPG allows to CALL server programs on the iSeries server and to pass data via parameters between the client VARPG program and the iSeries server program. 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 integrated file system (IFS) 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. The application is downloaded to the client workstation at startup. This approach has been adopted by many VARPG customers to avoid having to maintain applications on multiple clients.

If you need to run your application on a non-Windows system, VARPG allows to create Java source from the VARPG source. These Java applications then run on any Java Virtual Machine (JVM). To create Java programs from your VARPG application, follow these steps:

1. Click Build the Java version of the project (located next to the Windows Build button on the toolbar) in the GUI Designer.
2. After the build is completed, click Run the Java version of the project on the toolbar.

The application now runs in the JVM.

15.6 RPGIV operation codes not available in VARPG

Some operation codes are not supported in VARPG. These are listed here.

- ACQ
- DUMP
- EXFMT
- FORCE
- MHHZO
- MHLZO
- MLHZO
- MLLZO
- NEXT
- REL
- SHTDN

15.7 Additional information

For more information about programming with VARPG, consult the following Web sites:


The following news group is used by many VARPG users and provides excellent help from IBM experts as well as from the VARPG user community:

VARPG news group: news://news.software.ibm.com/ibm.software.varpg

For German customers, there is a very active German VARPG users group available. To enroll visit this web site: http://de.groups.yahoo.com/group/VARPGDE/ or mailto:VARPGDE@yahoogroups.de.

You can also find information on VARPG in these IBM 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 explains 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.
Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/SG246961

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the Additional materials and open the directory that corresponds with the redbook form number, SG246961.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

<table>
<thead>
<tr>
<th>File name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lib6961v52.savf</td>
<td>Sample library for running the sample applications described in this redbook. This library is for V5R2 of OS/400 or V5R3 of i5/OS.</td>
</tr>
<tr>
<td>lib6961v51.savf</td>
<td>Sample library for running the sample applications described in this redbook. This library is for V5R1 of OS/400. Several objects in this library will not be restored because the sample application described in Chapter 15, “Introduction to VisualAge RPG” on page 611, requires V5R2 version of OS/400.</td>
</tr>
<tr>
<td>Cusinq.war</td>
<td>Web project for the Web Integration explained in Chapter 13, “Building Web interactions with iSeries Web Tools” on page 431.</td>
</tr>
</tbody>
</table>
Installation instructions for the VARPG samples

Use the VARPG component install utility to install the two VARPG samples on your PC. To invoke the component install utility, use the Start menu on your desktop and follow these steps:

To restore project LIST6961 perform the following steps:

1. Extract the file LIST6961.zip into a new directory, for example C:\LIST6961.
2. From the Windows Start menu select Start → Programs → IBM WebSphere Studio → Development Studio Client for iSeries Vx.x.x → VisualAge RPG → Component Install Utility
3. On the Component Install Dialog specify the following:
   - Install from directory C:\LIST6961
     C:\LIST6961 being the drive where you extracted the ZIP file.
   - Target Folder/project c:\wdsc\projects\VARPG Projects
     c:\wdsc being the install drive and directory of the VARPG product directory.
   - Source directory c:\wdsc\LIST6961
     c:\wdsc being the install drive and directory of the VARPG product directory
4. Click the Ok button. A message box should confirm the successful installation.
5. Stay on the Component Install dialog to install the second VARPG project and skip the first step below.

For project CUST6961:

1. Extract the file CUST6961.zip into a new directory, for example C:\CUST6961.
2. From the Windows Start menu select Start → Programs → IBM WebSphere Studio → Development Studio Client for iSeries Vx.x.x → VisualAge RPG → Component Install Utility
3. On the Component Install Dialog specify the following:
   - Install from directory C:\CUST6961
     C: being the drive where you extracted the ZIP file.
   - Target Folder/project c:\wdsc\projects\VARPG Projects
     c:\wdsc being the install drive and directory of the VARPG product directory.
   - Source directory c:\wdsc\CUST6961
     c:\wdsc being the install drive and directory of the VARPG product directory
4. Click the Ok button. A message box should confirm the successful installation.
5. Click the Cancel button on the Component Install dialog.

You are finished installing the VARPG sample projects. If you now double-click the VisualAge RPG Projects icon on your desktop, the two new project should appear in the project list. Right-click the one you want to work with and select the edit option and the GUI designer comes up with the selected VARPG project loaded.
System requirements for downloading the Web material

The following system configuration is recommended:

**Hard disk space:** 190 MB minimum  
**Operating System:** Windows 2000 Professional  
**Processor:** Pentium® III or higher  
**Memory:** 512 MB

How to use the Web material

To use the Web material, follow the instructions in the following sections.

**Restoring the LIB6961 library**

You need to send by File Transfer Protocol (FTP) the lib6961v52.savf file (or the lib6961v51.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 yoursyste
   Connected to yoursyste.yourdomain.com.
   220 Connection will close if idle more than 5 minutes.
   User (yoursyste.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\lib6961v52.savf /qsys.lib/yourlib.lib/lib6961.savf
   200 PORT subcommand request successful.
   150-NAMEFMT set to 1.
   150 Sending file to member LIB6961 in file LIB6961 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(LIB6961)DEV(*SAVF)SAVF(yourlib/LIB6961)RSTLIB(LIB6961)
   ```

**Installing the Customer Master Inquiry Web application**

Follow these steps to install the application:

1. From the WebSphere Development Studio Client for iSeries tool bar, click **File → Import.**

2. On the Import window (Figure A-1), select **WAR** and click **Next.**
3. On the Import Resources from a WAR File window (Figure A-2), complete the following steps:
   a. Enter **Cusing** for the Web Project name.
   b. Enter **Cusing** for Context Root.
   c. Enter **CusingEAR** into the Enterprise Application project name field.
   d. Click **Finish**.
Figure A-2  Import Resources from a WAR File

4. Open the Web perspective.
   a. Right-click your new Web project and select Specify iSeries Web Tools runtime configuration.
   b. In the new dialog window, enter your iSeries system name, user ID, and password.
   c. Click Finish.

When you import the second example for Web Interaction, use the name Customer Inquiry instead of Cusinq.

Important: During the testing phase of these sample applications, we discovered a bug in the Web tools. To run the sample application correctly, install Fix Pack 5.0.1. For the installation instructions for the Fix Pack 5.0.1, see the following Web site:

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 about ordering these publications, see “How to get IBM Redbooks” on page 688. Note that some of the documents referenced here may be available in softcopy (PDF) only at http://ibm.com/redbooks.

- **Maximum Performance with WebSphere Application Server V5.1 on iSeries**, SG24-6383
- **Mastering the IBM WebFacing Tool**, SG24-6331
- **WebSphere Studio V5 Architecture and Overview**, REDP-3742
- **Patterns: Self-Service Application Solutions Using WebSphere V5.0 for iSeries**, REDP-3670
- **iSeries Access for Web and HATS Limited Edition: V5R2 Hot Topics for IBM @server iSeries Browser Users**, SG24-7005
- **IBM @server i5 and iSeries System Handbook**, GA19-5486
  Note: This book is frequently updated to match the current hardware and software releases. Therefore make sure to check http://ibm.com/redbooks for the latest edition.
- **WebSphere for the IBM @server iSeries Server Buying and Selling Guide**, REDP-3646.
  Note: This paper is frequently updated to match the current software releases. Therefore make sure to check http://ibm.com/redbooks for the latest edition.
- **Student Edition: WebSphere Development Studio Client for iSeries V5.0**, SG24-7086
  Note: This redbook is targeted to university and college students who are studying computer and information technology. It is based on the original edition of WebSphere Development Studio Client for iSeries V5.0, SG24-6961, published August 2003. The Student Edition contains additional exercises and quizzes as well as an on-going study.
- **WebSphere Portal Express and Express Plus V5 for the IBM @server iSeries Server**, SG24-6096
- **AS/400 Programming with VisualAge for RPG**, SG24-2222
- **Moving to Integrated Language Environment for RPG IV**, GG24-4358
- **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 Host Access Client Package Update**, SG24-6182
- **WebSphere Studio Application Developer Programming Guide**, SG24-6585
WebSphere Application Server V5 for iSeries: Installation, Configuration, and Administration, SG24-6588

HTTP Server (powered by Apache): An Integrated Solution for IBM @server iSeries Servers, SG24-6716

EJB 2.0 Development with WebSphere Studio Application Developer, SG24-6819

WebSphere Studio Application Developer Version 5 Programming Guide, SG24-6957

WebSphere Application Server - Express V5.0 for iSeries, REDP-3624

Other publications

These publications are also relevant as further information sources:

HTTP Server for AS/400 Programming, GC41-5435

Programming with VisualAge RPG, SC09-2449

VisualAge RPG Parts Reference, SC09-2450

VisualAge RPG Language Reference, SC09-2451

iSeries Performance Capabilities Reference, SC41-0607


VisualAge for RPG by Example, by Bryan Meyers and Jef Sutherland. 29th Street Press, April 1998, ISBN 1882419839

Online resources

These Web sites and URLs are also relevant as further information sources:

If you do not have WDSc installed on your workstation, you can also access the online help at:

http://publib.boulder.ibm.com/infocenter/iadthelp/index.jsp

WebSphere Development Studio for iSeries

http://www.ibm.com/software/awdtools/wds400/

The IBM WebFacing Tool

http://www.ibm.com/software/awdtools/wdt400/about/webfacing.html

Technical resources for the IBM WebFacing Tool:

http://ibm.com/servers/enable/site/ebiz/webfacing/start.html

IBM Web application development tools for iSeries

http://www.ibm.com/software/awdtools/iseseries/

Technical support and resources for iSeries and AS/400

http://www-912.ibm.com/
WebSphere Studio Site Developer
http://www.ibm.com/software/ad/studiositedev

WebSphere Studio Application Developer
http://www.ibm.com/software/awdtools/studioappdev

The eclipse.org Web site
http://www.eclipse.org

The Concurrent Versions System home page
http://www.cvshome.org

IBM WebSphere Application Server Version 4.0 Advanced Edition for iSeries
http://publib.boulder.ibm.com/was400/40/AE/english/docs

The IBM @server Workload Estimator

The Struts home page
http://jakarta.apache.org/struts

iSeries Information Center
http://www.ibm.com/eserver/iseries/infocenter

WebSphere Host On-Demand
http://ibm.com/software/webservers/hostondemand

iSeries Access for Web

WebSphere Host Publisher
http://www.ibm.com/software/webservers/hostpublisher

WebSphere Transcoding Publisher
http://www.ibm.com/software/webservers/transcoding

**Online Tutorials**
Several self study tutorials covering WDSc are available on the Internet:

- You can register for the Web-based Training Course SW738 *Introducing IBM WebSphere Development Studio Client for iSeries* at:

- Another Web-based Training Course SW712 *IBM WebSphere Development Studio Client V5.1 - Application Development with Struts* is located at:

- You can find an *An A – Z Hands-on Guide to IBM WebFacing Tool V5.1.2 Advanced Edition Lab* at:
  http://ibm.com/servers/enable/site/education/abstracts/3e06_abs.html

- The IBM WebFacing Tool V5.1 - Advanced Customization:
  http://www-1.ibm.com/servers/enable/site/education/abstracts/webfacing_abs.html

- There are also some tutorials provide with the online Help under *Tutorials and samples*. If you do not have WDSc installed on your workstation, you can also access the online help at:
  http://publib.boulder.ibm.com/infocenter/iadthelp/index.jsp
How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services
Index

Symbols

%%WB 362, 381
%colnumber 655
&DSPF 428
&L 135
&L - Object or Member Library 135
&LIB 428
&N 135
*%%WB 362, 381
*CURLIB 264, 305
*DSPF 274
*JOBD 276
*LIBL 264, 305
*LIST 547
*MSGF 360
*OTHER 501
*SADV 113
*SSOURCE 547
*SRVPGM 71
*STMT 547
*USRLIBL 264, 305
.modulemaps 263
.sav 114
.savf 114
.website-config 251
.wit 495
.zip 118
/COPY members 154
/end-free 645
/free 645
<LINK> tag 338

Numerics

127.0.0.1 83
3270 applications to the Web 229
5250 applications to the Web 229
5250 user interface 285, 287
5722-DG1 207
5722-E51 223–224
5722-IJE 224
5722-JC1 212
5722-JV1 224
5722-WDS 236
5722-XW1 236
5733-W51 223–224
5733-W44 223–224
5733-WS4 224
5733-WS5 223–224
5769-DG1 207
5769-FN1 404
5769-JV1 224
5769-DG1 207
5769-FN1 404
5769-JV1 224

A

A*%%WB 362
Access for Web 229
action bean 436
action class 510
Action hyperlink 369–370
action mapping 511, 514
Action Mapping Node 516, 521
ActionForm 451, 512
ActionForwards 451
ActionMappings 451
ActionServlet 510–512
active build styles 186
ActiveX Design-time Controls 61
ad hoc commands 179
Add Bookmark 44
Add Breakpoint 567
Add Physical File Member (ADDPFM) 85
Add Program 576, 587, 596
Add to Library List ... 148, 157
Add to Program Monitor 596
adding a record to the subfile 527
Adding JavaScript to an application 398
ADDPFM 85
adminclient 296
administration console
   enable in test environment 316
Administrative Console 316
ADTS (Application Development ToolSet) 18
Advanced Printer Function (APF) 18
AFP Compatibility Fonts 404, 407
After 392
Agent Controller 597
All authenticated users 322
alternate style 338
Apache 295
Apache Software Foundation 215, 509
Apache Tomcat Version 3.2 578
Apache Tomcat Version 4.0 578
Apache Tomcat Version 4.1 578
APF (Advanced Printer Function) 18
API (application programming interface) 208
apparea 340
apparea directory 340
apparea.css 340
Appearance and Text 368
appendSF 673
applet 206, 212
   benefits and considerations 213
   compared to servlet 217
   processing details 212
   supporting browsers 212
application client files 525
application client project 32
Application Component Provider 30
Application Developer 20
   Integration Edition 20, 24
overview 23
application logic in a Web browser 212
application programming interface (API) 208
application server 207, 215
application testing 502
application.xml 263, 444
application-ext.xml 263
Attributes view 246, 468
Authentication Tab 313
Auto-indent 165
automatic formatter 165
Automatic formatting 166
Automatic site structure creation 251
automatically detect host name of your workstation 560
autostart the WebFacing server 257

B
background color 334
Base Distinguished Name (DN) 317
batch application 553, 562
batch mode 176
bd_usr() 403
BEGACT 646
BEGSR 165
BIND.CLE 200
BLOCK(YES) 631
bo_usr() 403
bookmark 43
bootstrap port 296
breakpoint 567
   enable or disable 570
   prior to debugging 549
browse 661
Browse local 163
Browse project 163
Browse remote 163
bu_usr() 403
Build before launching 547
build style 194, 201–202
   *NONE 194
   CL program 195
   command 195
   specifying 195
   specifying a default 196
   specifying at project level 195
builder 46
business rule bean 24
business to the Web 218
business-to-business (B2B) xiii
button functions 403
Button Web component 435
By Date 147
bytecodes 211

c_i_e_i_w_5122_ 402
   c_i_e_i_w_5122  402
   CALL 648
   CALLP 648
   CALLPGM 648, 663
   CallServPGM 648, 651, 658
   CallServPGMA 658–659
canvas 619
Cascading Style Sheet Designer 252
cascading style sheet 333, 337, 435
   customization 339
   importing 339
   rule 339
CAXx= 329
CFxx= 329
CGI
   benefits 208
   Common Gateway Interface 207
   compared to Net.Data 210
   compared to servlets 213, 216
   considerations 209
   documentation 209
   persistent 208
CGI (Common Gateway Interface) 207
CGU (Character Generator Utility) 18
CHAIN 631
chaining to a specific subfile record 527
Change Command Defaults (CHGCMDDFT) command 173
Change Object Owner (CHGOBJOWN) 136
change position 367, 384
change span 367, 383
change style class 368
Change(B) 132
Character Generator Utility (CGU) 18
cheat sheet 511
Check Box Web component 435
Check PTF 256
CHGCMDDFT command 173
CHGOBJOWN 135–136
chrome 340–341
chrome directory 340
CICS 24
CISCO VPN software 561
CL command 282
   specify 343–344
   variable parameters 267
CLEAR 663
clearFieldValue 401
clearFieldValue function 401–402
clearing the subfile 527
ClearSF 673
Client Access 541
Client Access Family for iSeries 236
client machine requirements
   Host On-Demand 232
   Host Publisher 234
   IBM WebFacing Tool 233
   Web Interaction wizard 237
client modules 525

690  WebSphere Development Studio Client for iSeries V5.1.2
client.encoding.override 300
CLUP 526
COBOL 18
    content assist 150
CODE 16, 27
    Host Access Programs 18
Code
    filter view 147
CODE (CoOperative Development Environment) 27
    CODE Designer 27, 362
        open from Remote System Explorer 364
        open from WebFacing project 363
CODE Editor 594
    code procedures in service program to interact with sub-file APIs 526
CODE Project Organizer 27
Color
    changing background 460
COLSELECT 655
Combo Box Web component 435
command key 286, 329–330
    actions 331
    button labels 330
    elements 339
    recognition 276, 428
    recognition patterns 330
command keys with the browser 286
Command position 166
command sets 84
    creating 85
    working with 87
command shells 96
Comments
    filter view 147
Common Gateway Interface (CGI) 207
compact installation 677
Compare editor 47
compile commands 171
compile options 547, 592
Compiled language debugger 546
compiling source members 167
    CHGCMDFT command 173
    creating programs 169
    creating service programs 169
    CRTBNDRPG command 167, 171
    CRTPGM command 169
    CRTSRPGM command 167
    CRTSRVPGM command 169
    RPGLE 167
    run-time compile options 169
    working with compile commands 171
component 519
Configure Directory Server 315
Configure Enterprise Identity Mapping (EIM) 485
conflict detection 197
Connection profile 558
connections 79
    Connection name 80
    creating 80
    System type 80
Verify host name 81
    verifying 81
    Web diagram 513
connection-specific run-time options 171
connector resource archive (RAR) 324
Console view 279
content area 456
Content assist 243
content assist 150
context menu 270
context root 298
context-sensitive help 149
Continued line indent 166
Control
    filter view 147
Controller 509
conversion 258
conversion property 276
conversion wizard 258
Convert RPG Source command 10
converting an iSeries application 255
CoOperative Development Environment (CODE) 27
copy 42
    copy books 154
    corporate1 style sheet 375
count 650
crash recovery 40
Create Bound RPG Program (CRTBNDRPG) command 171
Create event 626
Create from page template 463
Create graphic 369
Create hyperlink 369
Create source member 85
Create subroutine 628
creating a Web page style 333
Creating a WebFacing Portlet project 302
creating objects
    ADDPFM command 108
    CRTDTAARA command 104
    CRTDTAQ command 105
    CRTMSGF command 106
    CRTSRCPF command 107
    creating your own IFS filters 93
    creating your own job filters 91
    CRTBNDRPG 172
    CRTBNDRPG command 171
CSS
    Designer 252
    CSS Designer 340
    CSS editor 460
    CSS file 276
    WebFacing 263
css1 252
css2 252
cstsf1.jsp 476
CUM Pack for WebSphere Application Server 223
cumpack 256
current library 276
cusinq 454, 463
pool 103
filter sets 601
Filters 249
Find String Using PDM (FNDSTRPDM) 110
first application panel 285
fixed template 421
fixpacks 51
fdenable 657–658, 661
Flow Controller 501
Flow controller 501
FNDSTRPDM 110
folders 95
folders and files 95
Font family 461
FOR 164–165
form 435, 464
form bean 436, 510, 512, 514
Format Document 473
forward() 418
free form ILE RPG code 164
free-form surface (FFS) 513, 515–516, 518
Frequency field 568
FTP 113
full prompt 556
full prompt to debug an iSeries job 585
function key text 330
Fxx 329

G
gallery 243
garbage collection 427
Generate Compile Code 198
Generate JSP custom tags 263
Generate output JSP for No program call 519
GET method 214
GETATR 646
GETDATA 663
GETDETAIL 497, 536, 676
getDetail 71, 495
getDetail Web interaction 483
getDetail.wit 484, 500
getDetailResults.jsp 496
GETDETAILV 676
getList 485
getList Web interaction 483
getList.wit 484
GetR 650, 674
getRecord 661, 674
GetSelected 676
global run-time options 169
group PTF 222
for WebSphere Application Server 223
GUI Designer 617

H
HATS 227
Height in pixels 369
HELP 287
hidden DDS fields 372
hidden input element 397
Hide characters 368, 390
Hide Field 367
Host Access Beans for Java 231
Host Access Class Library for Java 231
Host Access Controls for ActiveX 231
Host Access Objects 234
Host Access Transformation Server 227
host applications to the Web 228
Host Information wizard 484
host integration 228
Access for Web 229
comparison table 237
Host On-Demand (HOD) 228, 231
Host Publisher 229, 233
IBM WebFacing Tool 25, 230, 232
positioning 227
Screen Customizer 230
Transcoding Publisher 230
WebSphere Development Studio 16
WebSphere Studio 237
Host Integration family 228
Host Integration Object 235
Host Integration Solution 236
host machine requirements
Host On-Demand 232
Host Publisher 234
IBM WebFacing Tool 233
Web Interaction wizard 237
Host On-Demand (HOD) 227–228, 231
Host Publisher 227–229, 233
comparing to IBM WebFacing Tool 234
Host Publisher Server 233
Host Publisher Studio 233
host servers 256
starting from RSE 87
host/server access products 226
How To Start page 558
HTML (HyperText Markup Language) 206
HTML code 349
HTML file 337
HTML LINK element 337
HTTP proxy host address 55
HTTP server 206
HTTP Server (powered by Apache) 207
HttpServletRequest 215
HttpServletResponse 216
hyperlink 332
Hyperlink Web component 435
HyperText Markup Language (HTML) 206

I
IBM Developer Kit for Java (JDK) 224
IBM Directory Management Tool 315
IBM Distributed Debugger 546, 592
IBM eServer iSeries Client Access 541
IBM eServer iSeries Client Access Express 541
IBM Host Integration family of products 228
IBM HTTP Server for iSeries 207
IBM RSE Communications Server 552
IBM SecureWay Host On-Demand 228
IBM SecureWay Host Publisher 236
IBM Toolbox for Java 26, 541
IBM Toolbox for Java (5722-JC1) 212
IBM WebFacing Tool 25, 203, 227, 230, 232, 255
advanced techniques and considerations 415
compared to Host Publisher 234
getting started 259
migrating between versions 430
project wizard 258
IBM WebSphere Application Server 221
IBM WebSphere Studio Site Developer Advanced 259
ibm-web-ext.xml 451
Identity Token 327
Identity Token connector 323
IF 164–165
IFS (integrated file system) 208
IFS files subsystem 92
creating your own IFS filters 93
launching command shells 96
predefined filters 92
search for files and folders 95
working with folders and files 95
IFS filter 93
IFS source search path 560
IFxx 165
ILE 199
debugging 546
ILE (Integrated Language Environment) 224, 432, 592
ILE C 207
ILE COBOL 207
ILE COBOL compiler 10
ILE RPG 17, 207
Image Button component 435
Image Web component 435
Import Source option 487
imported style sheet 339
IMS 24
in Remote Systems Explorer 83
indent JSP source 473
indentation 153
index 351, 353
source view with input field 349
index.jsp 281, 344, 348, 351, 353, 418
infopop 50
information center 434, 453
Information Presentation Facility (IPF), IPF 614
information set 49
INIT 526
init() method 214
INITFirst 672
initial program 276, 284
initializing the subfile 527
INLPGM 268, 276, 284, 306
Insert Drop-Down Menu 357–358
Insert HTML 368, 400
Insert into script 371
Insert Link 476
Insert Variable 135
Inside 392
integrated file system (IFS) 208
creating a file share in the iSeries 292
Integrated iSeries debugger 546, 551
Integrated Language Environment (ILE) 199, 224, 432, 551, 592
debugging 546
Integrated xSeries Adapter 225
Integrated xSeries Server 225
Integration Edition
see Application Developer Integration Edition
Integration Objects 233
interaction 436
interaction wizard in Web diagram 518
interactive 5250 application 552
interactive job 208, 427
interactive mode 177
Interactive Source Debugger (ISDB) 18
interim fixes 51
introspection 217
Invalidate session 439, 486
Invalidate session after interaction occurs option 519
invocations.jsp 347–348, 351–354, 357
invoking your iSeries application 585
ISDB (Interactive Source Debugger) 18
iSeries Access Family (5722-XW1) 236
iSeries Access for Web 229
iSeries application 585
iSeries applications 547
iSeries Build Status view 200
iSeries Commands Log 117
running ad hoc commands 179
view 84
iSeries Commands subsystem 84
command sets 84
iSeries communications daemon 86
iSeries Developer Roadmap 3, 5
iSeries extension 25
iSeries job
debugging 585
working with 89
iSeries Jobs subsystem 88, 92
creating your own job filters 91
job statuses 92
predefined filters 88
viewing jobs 90
working with job filters 92
WRKACTJOB command 88
iSeries Navigator 257, 315
iSeries Objects subsystem 84
changing the current library 119
creating data areas 104
creating data queues 105
creating members 108
creating message files 106
creating objects 104
creating source physical files 107
filter pools 103
filters 97
managing your iSeries objects 97
predefined filters 84
specifying initial libraries and commands 120
working with filters 103
working with your library list 118
WRKLIBPDM command 84
WRKMBRPDM command 84
WRKOBJPDM command 84
iSeries objects, removing a library from the library list 119
iSeries Projects 182
associated library 182
build process 200
build styles 194, 201–202
building 200
caching file descriptions 193
creating 184
creating a local source member 189
creating a local source physical file 187
creating the project container 184
generating compile code 198
ILE based 199
importing remote host objects 190
iSeries Build Status view 200
iSeries connection 182
managing COMPILE.CLE 199
offline source verify 193
perspective, definition 182
perspective, opening 183
push changes 196
push operations 196
push selected 197
remote actions 194, 201
remote conflict detection 197
submitting project builds 200
viewing remote objects 191
working in disconnected mode 182
working with local source members 192
iSeries server, creating a file share in the IFS 292
iSeries sign-on information 81
iSeries Table View 122
iSeries table views 121
access PDM-like options 125
direct editing of table items 126
locking and unlocking 126
navigate views 124
populating views 122
positioning the list 126
print item list 124
sorting items 124
subsetted lists 125
working with items 124
iSeries Tasks 316
iSeries Web Components 432, 455, 465
iSeries Web Components Page Template 433
iSeries Web Components Tag Library 445
iSeries Web development tools 432
iSeries Web Integration 584
iSeries Web Tools Run-time Configuration wizard 485, 583
iSeriesWebComponents.css 455
iwcl 455
WButton 471
WTable 479
IWCLTagLib.tld 455
J
J2C 326
J2EE 221
J2EE applications 525
J2EE Connector Architecture (JCA) 485
J2EE level 444
J2EE Request Profiler 599
jee.jar 451
JAAS 485
Java project 509
Java Tomcat for iSeries support 215
JAR 525
Java 207, 224
Java API 212, 421
Java applet 212
processing on the iSeries server 212
Java Authentication and Authorization Service (JAAS) 485
Java builder 46
Java code on the Web server 213
Java Database Connectivity (JDBC) 421
Java debugger for iSeries 546
Java Development Toolkit (JDK) 220
Java development tools 26
Java Method Call wizard 163
Java package prefix 485
Java platform 211
Java Profiling Agent 599
Java project 32
Java Resources 451, 487
Java servlet processing on the iSeries server 215
Java Tools for iSeries examples 26
Java Virtual Machine (JVM) 211, 541
Java Virtual Machines (JVM) 597
Java-based technology 211
JavaBean 217, 421, 436, 514
JavaScript 371, 398
JavaScript debug adapter 546
JavaScript function call 371
JavaScript hyperlink 369–370
javascript user-defined function 401
JavaServer Faces 454
JavaServer Page (JSP) 218, 421, 432–433, 452
separating the presentation from the logic 217
JavaServer Pages (JSP)
for WebFacing 274
JavaSource 450
javax.servlet 214
javax.servlet.http 214
JCA 485
JCA Program call connector 436
JDBC (Java Database Connectivity) 421
JDK (Developer Kit for Java) 224
JDK (Java Development Toolkit) 220
JFace 21
job description 276
job filter 91
Job Status view 200
JSP 432–433, 452
component 218
for WebFacing 274
tag libraries 509
JSP (JavaServer Page) 218, 421
JSP Custom Tag Library 436
JSP custom tags 263
JSP fragment 453
JSP scriptlets 371
jtpl 453, 455
JVM 597
JVM (Java Virtual Machine) 211
JVM property 300

K
key label 366
key order 366

L
Label above command 166
Label position 166
Label Web component 435
languages, multiple 359
launch configuration 556
launch page 351
for WebFacing 285, 347
launching command shells 96
terminate 97
terminate and remove 97
Layout
of the palette view 247
LDAP 313, 315
configuring 315
port 317
ldapadd 316
LDAPHostName 326
legacy applications 220
lib 451
Lib6961 268, 282, 306
Libraries 451
library 264, 305
for display file 274
list to run the 5250 application 276
on the iSeries server 276
library filters 97
library list 118, 276, 359
adding to 118
library list filter 99
Lightweight Directory Access Protocol (LDAP) 313, 315
line breakpoint 567
line number
not the same 156
LINK element 337, 339
link style sheets 337
links to other pages 501
Linux on iSeries 225

list of spooled files 406
Listeners 249
Live Parsing Extensible Editor (LPEX) 140
local 113
local history 46
localhost 83
location 367, 383
login-config 249
logon.html 282
logon.jsp 282
look and feel 333
Lotus Domino 225
LPEX (Live Parsing Extensible Editor) 140
LPEX editor 140, 360
/COPY members 154
closing 155
copy books 154
filter views 147
functions 145
launching 140
online help 149
saving changes 154
show fields 148
show indentation 153
source line prompting 145
source verify 151
syntax checking 145

M
Macro Recording 235
MAP 207
Map Page Elements to the Page Template 500
Master.css 455, 460
Master_Template 454–455
Master_Template.jtpl 500
Master_template.jtpl 457
Maximum file wait time 132–134
Maximum record wait time 132–134
MCH0601 408
MEDIA attribute 338
MEDIA values 338
member filter 101
WRKMBRPDM command 101
members in source physical file 108
memory 597
Memory utilization 597
MENUSELECT 656, 662
message descriptions 107
message files 106, 360
working with message descriptions 107
META-INF 263
Method Invocation Table 605
migrating WebFacing projects 430
migration between different IBM WebFacing Tool versions 430
MIME 250
MLG265 437
MLGMSTL2 659
MNIBROWSE 656
MNICREATE 662–663
MNDELETE 662
MNINEW 662
MNUDDS options 332
Model 509
model2 509
Model-View-Controller (MVC) 218, 509
modes of execution 176
modify style 334
Modules view 576
MONITOR 165
Monitor Expression view 574
Monitors view 572, 574
move 42
MPCML 487
multilanguage support 300
multilingual applications for WebFacing 359
multiple display files 424
multiple input pages 502
multiple language WebFacing 359
multiple output pages 501
multiple style sheet 338
Multiple Web project support 251
Multi-threaded 179
MVC (Model-View-Controller) 218
My jobs 89
MYLIB/MYDSPFOB 428

N
Named Constant 162
named type 129–130
    creating 129
    removing 131
NAS 322
National language support 413
Navigator 450
Net.Data 207, 209
    benefits 210
    considerations 210
    documentation 211
    processing details 210
network authentication service (NAS) 322
New Page Template File wizard 454
new source to an existing project 285
newRecord 662, 675
No program call Web interaction 519
nodes 514
    Web diagram 513
non-Java-based applications 207
numparm2 650
numrec 650

O
Object Action 134
object filter 99
    WRKOBJPDM command 99
Object List bean 26
    object mappings 274
    objects subsystem
        adding a library list entry 118
One parameter per line 166
ON-ERROR 165
one-step debugging 551, 585, 588
online help 48, 83
onsubmit attribute 494
Open Host Interface Objects 231
Open Page Designer 348
Operations Navigator 257
OPM 431–432, 483, 592
    debugging 546
Options for VALUES 368
org.apache.struts.action.Action 510
org.apache.struts.action.ActionServlet 510
Original Program Model (OPM) 431–432, 483, 551, 592
    debugging 546
OS/400 PASE 224
OS/400 program 432
OTHER 165
Other jobs 89
Outline View 460
Outline view 36, 155, 572, 591
Override constant text 368, 389
Override field value 368, 389
Override field value field 389
override label 366
Override project settings 313

P
Page Designer 347, 433, 453, 455
    Changing index.html 348
    Classic 433
    editor to design Web pages 433
Page Down 526
Page Down button 527
page presentation 337
Page Template 433, 453
    adding content 456
    page template 251, 435
Page Template File Selection 463
Page Template File selection dialog 476
Page Template File wizard 433, 454
Page templates 434
Page Up 526
Page Up button 527
PageBuilder.jsp 341
PageEnd event 628
Palette view 246
Parameters for Table component 480
ParentDomain 326
PARM 344
parmform2 650
Part name 621
Parts Palette 619
PASS 207
password 267, 305, 312
password encoding 484
PCML 487
PCML (Program Call Markup Language) 538
PDE (Plug-in Development Environment) 21
PDM 413
PDM (Programming Development Manager) 18
PDM actions 122
PEP (program entry procedure) module 169
performance
exporting the project 426
profiling 23
workbench settings 424
PERL 207, 216
persistence 217
persistent CGI 208, 214
persistent style 338
personal profile 78
perspective 32
PGDN 71, 526
pgmname 650
PGUP 526
platform roles 30
plug-in 22
Plug-in Development Environment (PDE) 21
pool, filter 103
POPSUB 660, 662
POPUP 656
pop-up menu 641
PORT 178
port
for LDAP 317
portability 217
portal applications 307
portal server 302
Portal Test Environment 307
Portal Toolkit 302
Portals 302
Portlet 302, 307
ports
TCP/IP 82
Position cursor to the field 370
POST method 214
pRcd 528–529, 531
pre-defined filters 92
preferred style 338
pReserved 530
PRESS 660, 663
PRESS event 662
pre-supplied JavaBeans 26
Preventative Service Planning (PSP) 224
Preview page 347, 351
Print Orders (PRTORDERR) RPG server job 68
problem determination 507
procedural language 615
profile 78, 312
creating 78
deactivate 79
deleting 79
personal 78
shared 78
team 78
Profiling
Class Interactions 606
Class Statistics 606
Execution flow 606
Garbage collection 606
Host interactions 607
Instance statistics 605
Method invocation 604
Method statistics 603
Object Interactions 606
Object references 606
Package Statistics 606
Process interactions 607
Stopping the profiling session 607
Thread Interactions 606
Profiling and Logging perspective 602
Profiling applications 596
profiling architecture 597
profiling mode 598
Profiling perspective 602
profiling wizard 598
program 169
program alias 490
Program Call bean 431, 436, 538
Program Call Markup Language (PCML) 487, 538
Program Call wizard 26
program defined HTML 385
Program Development Manager (PDM) 413
program entry procedure (PEP) module 169
Program status data structure 162
program-defined HTML 367
Program-described data structure 162
Programming Development Manager (PDM) 18
Programs view 576
project 261
adding new source to existing 285
creating new 261
WebFacing 272
project creation
considerations 425
multiple display files 424
single display file 423
Project Navigator 448
Project Navigator view 449, 451
project size 425
Project View 618
prompt 288
Prompt first 136
prompting filters 84
Properties dialog 481
properties page 329
Properties view 35
PROTECT keyword 369
proxy 55
proxy host 56
PSBCREATE 662
PSP 224
PTF
automatic check 256
cumpack 256
V5R1, SI07523 (SDA) 381
V5R2, SI07526 (SDA) 381
PTF CUM pack for WebSphere Application Server 223
PTFs 413
publishing information 579
Push Changes 196
Push changes 196
Push operations 196
Push Selected 197
Push selected 197

Q
QAUTOVRT 257
QDBGSVR 592
QDDSSRC 428
QDSPSGNINF 408
QdtsAppendSF API 505, 527–528
QdtsChainSF 527, 529
QdtsClearSF 527, 529
QdtsInitSF 527, 530
QdtsReadcSF 530
QdtsReadSF 527
QDTSSFL 673–674
QDTSSFL development environment 527
QDTSSFL program 533, 536
QDTSSFL.SRVPGM 527
QdtsUpdateSF 527, 531
QIGCCDEFNT 404
QQFENV 417
QqlEnvironment 386, 417
QqINVOKER 427
QQFVTSVR 257, 427
QQFWSFVR 257, 426
Qshell 316
QSYSWKR 593
Query for iSeries (5722-QU1) 413
Quick Edit 244
quick help 50
QUSRLIBL 276
QzshbCgiParse API 208–209
QZRCSRVS 439, 507, 519
QZRUCCLSP 650

R
Radio Button Group Web component 435
RAR 324
RAR file 525
REDC 631
readcSF 536
reading a selected record 527
READS 631
Reads all records from the database option 527
Reads one page of records at a time from the database option 527
realized JSP 522
realized node in Web diagram 516
rearranging views and editors 44
recognition pattern 329
Reconverting the DDS 381
Recovering from an error caused by an unsupported system panel 412
Redbooks Web site 688
    Contact us xvi
Reference 249
Referenced Keywords 271
REL attribute 338
Relaunch icon 567
remote conflict detection 197
Remote Java Compile and Run 26
Remote Method Invocation (RMI) 421
Remote Servers 87
Remote System Details 76
Remote System Explorer (RSE) 6, 76
    ADDBNDDIRE command 85
    compiling source members 167
    connections 79
    debugging programs 180
    drag and drop functionality 137
    editing source members 140
    iSeries table views 121
    Outline views 155
    perspective, opening 76
    Profiles 78
    running programs 174
    Subsystems 83
    User actions 127
    views, default 76
Remote System Profile 79
Remote Systems Details 122
Remote Systems LPEX Editor 140, 142
    functions 145
Remote Systems view 35, 81
Remove All Terminated Launches icon 567
Remove terminated launches when a new launch is created 547
rename 42
Repeat previous specification type 165
Report Layout Utility (RLU) 18
Report Service 404
Resequence lines at save 157
Reset Perspective 156, 273, 312
resource adapter module 525
resource archive (RAR) 324
Resource Navigator view 273
Resource Perspective 450
Restore to iSeries 114, 118
Resume icon 567
Reuse editor when displaying source code 547
reverse image color 334
RGB 460
RLU (Report Layout Utility) 18
RMI (Remote Method Invocation) 421
RMTCOCNAME 177
Roadmap 3
RPG 17
RPG IV 616
RPG procedure
    creating 158
rrn field 401
RSE
    communications daemon port number 178
RSE (Remote System Explorer) 76
RSE subsystem 81
<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSE2001</td>
<td>178</td>
</tr>
<tr>
<td>RSEG1009E</td>
<td></td>
</tr>
<tr>
<td>You must rename</td>
<td>114</td>
</tr>
<tr>
<td>rt.jar</td>
<td>451</td>
</tr>
<tr>
<td>RT_WIN32</td>
<td>636</td>
</tr>
<tr>
<td>rtmessage.properties</td>
<td>345–346</td>
</tr>
<tr>
<td>rtvalue</td>
<td>650</td>
</tr>
<tr>
<td>Run (no prompt) option</td>
<td>174</td>
</tr>
<tr>
<td>Run (with prompt) option</td>
<td>174</td>
</tr>
<tr>
<td>Run on Server</td>
<td>252, 277, 578</td>
</tr>
<tr>
<td>Run To Location</td>
<td>573</td>
</tr>
<tr>
<td>running ad hoc commands</td>
<td>179</td>
</tr>
<tr>
<td>running modes</td>
<td>178</td>
</tr>
<tr>
<td>CALL command</td>
<td>176</td>
</tr>
<tr>
<td>SBMJOB command</td>
<td>176</td>
</tr>
<tr>
<td>running programs</td>
<td></td>
</tr>
<tr>
<td>CALL command</td>
<td>176</td>
</tr>
<tr>
<td>from RSE</td>
<td>174</td>
</tr>
<tr>
<td>in batch mode</td>
<td>176</td>
</tr>
<tr>
<td>in interactive mode</td>
<td>177</td>
</tr>
<tr>
<td>modes of execution</td>
<td>176</td>
</tr>
<tr>
<td>run-time compile options</td>
<td>169</td>
</tr>
<tr>
<td>Run-time Configuration wizard</td>
<td>432–433, 436</td>
</tr>
<tr>
<td>run-time properties</td>
<td>276</td>
</tr>
<tr>
<td>sampleportlet</td>
<td>304</td>
</tr>
<tr>
<td>Save dirty editors before launching</td>
<td>547</td>
</tr>
<tr>
<td>Save Files</td>
<td>113</td>
</tr>
<tr>
<td>Save Library (SAVLIB)</td>
<td>115</td>
</tr>
<tr>
<td>Save to Session</td>
<td>493, 508</td>
</tr>
<tr>
<td>SAVF</td>
<td>113</td>
</tr>
<tr>
<td>SAVLIB</td>
<td>115</td>
</tr>
<tr>
<td>Screen Customizer</td>
<td>230</td>
</tr>
<tr>
<td>Screen Design Aid (SDA)</td>
<td>18, 362, 381</td>
</tr>
<tr>
<td>SDA</td>
<td>362</td>
</tr>
<tr>
<td>SDA (Screen Design Aid)</td>
<td>18, 362, 381</td>
</tr>
<tr>
<td>search</td>
<td>43</td>
</tr>
<tr>
<td>searching for files and folders</td>
<td>95</td>
</tr>
<tr>
<td>Security</td>
<td>249</td>
</tr>
<tr>
<td>Security Constraints</td>
<td>321</td>
</tr>
<tr>
<td>SELECT</td>
<td>165</td>
</tr>
<tr>
<td>select</td>
<td>660</td>
</tr>
<tr>
<td>Select a Page Template for the Web Site</td>
<td>447</td>
</tr>
<tr>
<td>SELECT list</td>
<td>358</td>
</tr>
<tr>
<td>Select UIM source members to convert</td>
<td>266</td>
</tr>
<tr>
<td>Select Web Pages</td>
<td>500</td>
</tr>
<tr>
<td>Selection Box Web component</td>
<td>435</td>
</tr>
<tr>
<td>Selection list</td>
<td>357</td>
</tr>
<tr>
<td>Send to browser</td>
<td>372</td>
</tr>
<tr>
<td>Send to browser as a hidden input element</td>
<td>397</td>
</tr>
<tr>
<td>sendRedirect()</td>
<td>418</td>
</tr>
<tr>
<td>SEP</td>
<td>549–550</td>
</tr>
<tr>
<td>server configuration</td>
<td>278, 308, 503</td>
</tr>
<tr>
<td>Server Configuration view</td>
<td>279</td>
</tr>
<tr>
<td>server instance</td>
<td>503</td>
</tr>
<tr>
<td>Server Perspective</td>
<td>503</td>
</tr>
<tr>
<td>WDSC V5 versus WDSC V4</td>
<td>279</td>
</tr>
<tr>
<td>server project</td>
<td>32</td>
</tr>
<tr>
<td>Server Selection window</td>
<td>307</td>
</tr>
<tr>
<td>Server server1 open for e-business message</td>
<td>280</td>
</tr>
<tr>
<td>Servers view</td>
<td>279</td>
</tr>
<tr>
<td>Service Data Objects</td>
<td>454</td>
</tr>
<tr>
<td>Service Entry Object (SEP)</td>
<td>549–550</td>
</tr>
<tr>
<td>Service program</td>
<td>648</td>
</tr>
<tr>
<td>service program</td>
<td>169, 432</td>
</tr>
<tr>
<td>Table Web component</td>
<td>527</td>
</tr>
<tr>
<td>service program name called by table</td>
<td>526</td>
</tr>
<tr>
<td>service() method</td>
<td>214</td>
</tr>
<tr>
<td>servlet</td>
<td>206, 213</td>
</tr>
<tr>
<td>benefits</td>
<td>216</td>
</tr>
<tr>
<td>doDelete method</td>
<td>214</td>
</tr>
<tr>
<td>doGet method</td>
<td>214–215</td>
</tr>
<tr>
<td>doPost method</td>
<td>214–215</td>
</tr>
<tr>
<td>doPut method</td>
<td>214</td>
</tr>
<tr>
<td>init() method</td>
<td>214</td>
</tr>
<tr>
<td>service() method</td>
<td>214</td>
</tr>
<tr>
<td>supported by WebSphere Studio Site Developer</td>
<td>23</td>
</tr>
<tr>
<td>Servlet API</td>
<td>421</td>
</tr>
<tr>
<td>servlet life cycle</td>
<td>214</td>
</tr>
<tr>
<td>Servlets</td>
<td>249</td>
</tr>
<tr>
<td>session bean</td>
<td>220</td>
</tr>
<tr>
<td>Set Style Properties</td>
<td>461</td>
</tr>
<tr>
<td>SETATR</td>
<td>631, 646</td>
</tr>
<tr>
<td>setFieldValue function</td>
<td>402</td>
</tr>
<tr>
<td>setOptions</td>
<td>401–402</td>
</tr>
<tr>
<td>setOptions function</td>
<td>400</td>
</tr>
<tr>
<td>setting breakpoints prior to debugging</td>
<td>549</td>
</tr>
<tr>
<td>-setworkspace</td>
<td>31, 259</td>
</tr>
<tr>
<td>SEU (Source Entry Utility)</td>
<td>18, 362</td>
</tr>
<tr>
<td>SFL1</td>
<td>498, 673</td>
</tr>
<tr>
<td>SFLNAM</td>
<td>498</td>
</tr>
<tr>
<td>SGML comment</td>
<td>339</td>
</tr>
<tr>
<td>shared profile</td>
<td>78</td>
</tr>
<tr>
<td>sharing</td>
<td></td>
</tr>
<tr>
<td>command set</td>
<td>86</td>
</tr>
<tr>
<td>Show Field</td>
<td>367</td>
</tr>
<tr>
<td>Show in table</td>
<td>132</td>
</tr>
<tr>
<td>Show Qualified Names</td>
<td>567</td>
</tr>
<tr>
<td>Show Qualified Names icon</td>
<td>567</td>
</tr>
<tr>
<td>Show View</td>
<td>551</td>
</tr>
<tr>
<td>SHOWWIN</td>
<td>631</td>
</tr>
<tr>
<td>simple project</td>
<td>32</td>
</tr>
<tr>
<td>Simple Table</td>
<td>435</td>
</tr>
<tr>
<td>single file display</td>
<td>423</td>
</tr>
<tr>
<td>Single signon</td>
<td>313</td>
</tr>
<tr>
<td>single signon</td>
<td>263, 313</td>
</tr>
<tr>
<td>with WebFacing portlet</td>
<td>326</td>
</tr>
<tr>
<td>Site Developer</td>
<td>20</td>
</tr>
<tr>
<td>Snippets</td>
<td>247</td>
</tr>
<tr>
<td>source breakpoint</td>
<td>549</td>
</tr>
<tr>
<td>Source Entry Utility (SEU)</td>
<td>6, 18, 139, 362</td>
</tr>
<tr>
<td>source member</td>
<td>140, 428</td>
</tr>
<tr>
<td>Source page</td>
<td>347, 349</td>
</tr>
<tr>
<td>source physical file</td>
<td>107</td>
</tr>
<tr>
<td>Source view</td>
<td>350</td>
</tr>
<tr>
<td>source, adding new to existing project</td>
<td>285</td>
</tr>
<tr>
<td>SourceRegistryName</td>
<td>326</td>
</tr>
<tr>
<td>spaceName</td>
<td>528–531</td>
</tr>
<tr>
<td>spaghetti design code</td>
<td>3</td>
</tr>
</tbody>
</table>
Specify CL commands 344
Specify CL commands window 343
specify host name of your workstation 560
Specify static URL 369–370
specifying styles 337
spooled files list 406
SQL 165, 208
SQL statements
   filter view 148
Standard Widget Toolkit (SWT) 21
Start SQL Interactive Session (STRSQL) 413
Step Debug icon 567
Step Over icon 567
step-by-step debugging 588
Storage monitor view 577
STRCODE 593
STRDBGSRVR 593
stream file 208
STRHOSTSRVR *ALL 256
STRRSSRVR 177, 552
   TCP/IP port 82
STRRSSRVR command 86, 178
STRSQL 413
STRTCPSSVR 257
strucdet 659
struc 650
strut-config.xml 453, 495
Struts 509
   action 511
   ActionServlet 512
   Add support to a Web project 445
   and WebSphere Studio 511
   application 511
   cheat sheet 511
   configuration file 511
   form bean 512
   forward 512
   module 514
   nodes 514
Struts enabled 331, 404
Struts Explorer 452
Struts Settings 446
Struts-based Web application 511, 513
Struts-based WebFacing service 331
struts-config.xml 432, 451, 453
STRWFSRVR 257
style
   modify 334
   Web page 333
   WebFacing 230, 333
STYLE element 339
Style properties 276, 333
style sheet 337
Styles 340, 460–461
Styles view 342
subfile 274, 475, 650
subfile API 526
   operations, names 528
   to manipulate table data 526
subfile DTC 26, 435, 475, 477, 526
Subfile Error 509
Subfile part 620
Submit 371
Subsystems 83
subsystems 83
   filter pools 103
   IFS files 92
   iSeries Commands 84
   iSeries Jobs 88, 92
   iSeries Objects 84
   prompting filters 84
Supported panels 413
survey tool 416
Suspend 567
Suspend icon 567
Swing JFormatted bean 26
SWT (Standard Widget Toolkit) 21
system application server instance 316
System Screen Support 263, 266, 408

T
Table DTC 477
Table Web component 435, 464, 475, 477, 526, 536
tag libraries 509
Tag Library 445
taskhist 657, 661
Tasks view 36
TCP/IP support in VisualAge RPG 612
team environment 47
team profile 78
Template containing iSeries Web Components Tag Li-
brary 454
Terminate All icon 567
Terminate and Remove icon 567
Terminate icon 567
test environment 277
   enabling security 316
testing application without deploying 502
Testing your Webfacing Portlet application 307
text 334
Text Area Web component 435
Text Entry Web component 435, 467
textbuttons 269
textbuttons style 342
theme 450, 460
this.form.opt 401
Thread field 568
Thumbnail view 458
thumbnails 243
TIMFMT 160
TITLE attribute 338
Toolbox for Java 26
Transcoding Publisher 230
TYPE attribute 338

U
UIM 266, 408
UIM (User Interface Manager) 235, 266, 305
UIM help 287
UIM object mappings 429
unit testing your application 281
unrealized node 521
in a Web diagram 518–519
Update Manager 59
updating a subfile record 527
uRC 528–531
URL 298
  WebFacing application 284
uRRN 529, 531
Use error page 485
Use OS/400 signon 314
Use SOCKS 57
usebean 245
User Actions 134
user actions 127, 134
  creating 127
  invoking 129
  named types 129
User defined HTML 367, 387
user ID 267, 305
User Interface Manager (UIM) 235, 266, 408
user profile 312
User-Defined Page Template 463, 476
user-defined tokens 150
ustartRRN 531
UTF-8 280, 300

V
VALUES 288
variable parameters 343
Variables view 573
VARPG (VisualAge RPG) 612
VARPG GUI Designer 617
VARPG programming 615
VCT 432, 436
version migration for IBM WebFacing Tool 430
versions of WebSphere Application Server 222
view 34, 44, 509
  iSeries table 76
  WebFacing projects 272
View log 270
Views
  *Listing 573
  *Source 573
  *Statement 573
  iSeries Build Status 200
  iSeries Commands Log 76
  Outline 76, 155
  Property Sheet 76
  Remote Systems 76
  Tasks 76
virtual terminal server 257, 427
visibility 367, 383
Visual Composition Editor 217
Visual Custom Tags (VCT) 432, 436
Visual Development Environment 612
VisualAge for Java 217
VisualAge RPG (VARPG) 16, 27, 612

Host Access Programs 18
VisualAge RPG Language Reference 613

W
WAITFILE 134
WAITRCD 134
WAR 249
WAR file 263, 289, 439, 525
watch breakpoints 567
WCSS 252
WDSc xiii
WDSC (WebSphere Development Studio Client) 255
Web Administration for iSeries 295
Web application 220, 514, 525
  assembly 525
  combining new and existing 227
  debug 588
  development, deployment 220
  page design 453, 459
to production 525
Web Archive Resource (WAR) 263, 439
Web authors 421
Web browser 286
Web component 25, 431, 435–436, 440
  adding to JSP pages 465
  Button 435
  Check Box 435
  Combo Box 435
  Hyperlink 435
  image 435
  Image Button 435
  Label 435
  Radio Button Group 435
  Selection Box 435
  Table 435, 526, 536
  Text Area 435
  Text Entry 435
Web Components 432, 455
  enhancements with WDSc 5.1.2 435
  Web Components Page Template 433
Web Components Tag Library 445
Web content 337
Web deployment descriptor 249, 451
Web design 337
Web development tools 432
Web diagram 439, 513
  connections using colors and shapes 514
  empty 514
  nodes 514
Web Interaction
  enhancements with WDSc 5.1.2 436
Web interaction 432, 495
  in Web diagram 514
  programs 584
Web Interaction wizard 25, 228, 431–436, 439, 453, 483
Web interaction wizard 507
Web module 525
Web page 433, 514
  styles 333
  using Page Designer editor 433
WIN1  662
WIN1_PSBSEARCHPRESS  648
WIN2  660
WIN3  660
window subfile  287
wizards  242
Work with All Spooled Files (WRKSPFLF)  408
Work with All Spooled Files example  409
Work with Objects Using PDM (WRKOBJPDM)  99
Work with Spool File  408
Work with Spooled Files (WRKSPFLF)  404
Work with user actions  134
Work with User Jobs (WRKUSRJOB)  89
workbench  31, 260
working with file filters  96
working with folders and files  95
working with your library list  118
Workload Estimator  422
workspace  258–259, 441
         location  31
         set as the default  259
World Wide Web Consortium (W3C), W3C  252
WRITE  631
WRKLIB  177
WRKSPFLF  404, 408
WRKSRVTBLE  592
WRKUSRJOB  89
WSAA  24
WSSDa  432
WSVR0001I message  280
WYSIWYG  347

X
X.500  315
XML editor  512

Y
You must rename  114

Z
z/OS  24
WebSphere Development Studio Client for iSeries Version 5.1.2

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

Find out about the latest trends in application development

While the IBM WebFacing Tool is probably the most commonly known component of WDSc, there are many other tools combined into a single product. All those components together provide a complete workbench based on the Eclipse framework for the application developer for OS/400 or i5/0S. This IBM Redbook discusses WebSphere Development Studio Client for iSeries V5.1.2 and WebSphere Development Studio Client Advanced Edition for iSeries V5.1.2. The Development Studio Client includes several powerful tools that are targeted to iSeries application developers:

- WebSphere Studio Site Developer or WebSphere Studio Application Developer
- iSeries-specific features, such as:
  - Remote Systems Explorer
  - IBM WebFacing Tool
  - Web Interaction Wizard
  - CoOperative Development Environment (CODE)
  - VisualAge RPG
  - Integrated iSeries debugger

For more information: ibm.com/redbooks