

IBM WebSphere Studio
Application Developer Version 5.0.1



Migration Guide

Note!

Before using this information and the product it supports, be sure to read the general information under Notices.

Fourth edition (April 2003)

© Copyright International Business Machines Corporation 2000, 2003. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Chapter 1. WebSphere Studio Application Developer Version 5.0.1 Migration Guide	1
--	----------

Chapter 2. Targeting WebSphere Application Server Version 4.0.x versus Version 5 versus Version 5 Express. . . .	3
---	----------

Chapter 3. Migrating from WebSphere Studio Application Developer Version 4.0.x	5
3.1 Differences between WebSphere Studio Application Developer Version 4.0.x and Version 5.	5
3.2 WebSphere Application Server changes and Servlet/JSP conversion tools	6
3.3 Internal changes from Version 4.0.3	6
3.3.1 Circular project dependencies will not build by default	6
3.3.2 Version 5 Web projects are source location compatible with Version 4.0.3.	6
3.3.3 WebSphere Studio Application Developer Web project structures	7
3.3.4 Static versus J2EE Web Projects	7
3.3.5 WebSphere Studio Application Developer EJB 1.1 and Application Client 1.2 project structures	7
3.3.6 Page Designer "Classic"	8
3.3.7 Page Designer enhancements	8
3.3.8 Page Designer "Classic" versus Page Designer functions	9
3.3.9 HTML and JSP distinctions.	9
3.4 Migrating projects using a software configuration management (SCM) system	9
3.4.1 Migrating projects using CVS or Rational ClearCase	9
3.4.2 Post-Migration removal of EAR and Server Configuration absolute path references	11
3.4.3 Migrating projects using other SCMs	11
3.5 Migrating by exporting and importing your projects.	11
3.6 Migrating projects using an existing Version 4.0.x workspace.	12
3.6.1 Post-Migration removal of EAR and Server Configuration absolute path references	12
3.6.2 Known problems and limitations	12
3.7 Migrating J2EE project structures and/or J2EE specification levels	13

Chapter 4. Migrating from WebSphere Studio Application Developer Version 5 Early Availability or Beta Versions	15
---	-----------

Chapter 5. Migrating from WebSphere Studio "Classic" to WebSphere Studio Application Developer	17
5.1 Creating a new single-server stage for migration	18
5.2 Creating a Web configuration descriptor file	18
5.3 Exporting a migration JAR file.	18
5.4 Importing the migration JAR file into WebSphere Studio Application Developer	18
5.5 Testing your migrated application on a test server	19

Chapter 6. Migrating from VisualAge for Java to WebSphere Studio Application Developer	21
6.1 Differences between VisualAge for Java and WebSphere Studio Application Developer	21
6.2 Migrating from VisualAge for Java	22
6.2.1 Exporting your Java files and project resource files from VisualAge for Java	22
6.2.2 Starting WebSphere Studio Application Developer and creating new projects to contain your code	23
6.2.3 Importing your Java and resource files into WebSphere Studio Application Developer	23
6.2.4 Using the web.xml editor to ensure that servlets are correctly defined (Web project only)	23
6.2.5 Migrating project and workspace settings	24
6.2.6 Setting up your WebSphere V4 test environment and testing your migrated application(s).	25
6.2.7 Deploying your applications from WebSphere Studio Application Developer to remote WebSphere Application Server	26
6.2.8 Sharing WebSphere Studio Application Developer project settings between multiple developers (post-migration)	26
6.3 Team support in WebSphere Studio Application Developer	26

Chapter 7. Migrating enterprise beans from VisualAge for Java to WebSphere Studio Application Developer	27
7.1 VisualAge for Java EJB Export Tool (migrating map/schema from EJB 1.0 to EJB 1.1).	27
7.1.1 Items migrated	28
7.2 VisualAge for Java Version 3.5.3 EJB 1.0 JARs versus VisualAge for Java Version 4.0 EJB 1.1 JARs	28
7.3 Moving multiple VisualAge for Java EJB groups into WebSphere Studio Application Developer EJB projects.	29
7.4 Migrating your enterprise beans	29
7.4.1 Exporting your enterprise beans.	29
7.4.2 Importing your enterprise beans into WebSphere Studio Application Developer	29

7.4.3 Generating deploy code and data source binding information	30
7.4.4 Creating a server configuration and instance	30
7.4.5 Adding the JDBC data source to the WebSphere 4.0 server configuration	30
7.4.6 Testing the enterprise beans with the EJB test client	31
7.5 Known problems and workarounds	31
7.6 Locating EJB information	31
7.7 Migrating EJB access beans	32
7.8 Migrating custom finder helpers	32

Chapter 8. Migrating from EJB 1.0 to EJB 1.1 or to EJB 2.0 35

8.1 Migrating code from EJB 1.0 to EJB 1.1	35
8.2 Converting projects from EJB 1.x to EJB 2.0	36
8.3 Migrating code from EJB 1.x to EJB 2.0	37
8.4 Migrating a JMS listener application to use message-driven beans	38

Chapter 9. Migrating from VisualAge for Java Visual Composition Editor to Visual Editor for Java 41

9.1 Saving enhanced design-time metadata from VisualAge for Java	41
9.2 Completing the migration (importing into WebSphere Studio)	42

Chapter 10. Converting from VisualAge for Java Persistence Builder to EJB 2.0 43

10.1 Migrate and export the Persistence Builder Model as an EJB 1.1 JAR	43
10.2 Import EJB 1.1 JAR into EJB 1.1 project, set Access and Isolation values	43
10.3 (Optional) Convert client application and get server side running, all at EJB 1.1 level	44
10.4 Convert to EJB 2.0 project	44
10.5 Convert EJB 1.1 code to EJB 2.0 and set application profile declarations	44
10.6 Convert Persistence Builder client application into EJB client application	45

Chapter 11. Migrating from WebSphere Studio Application Developer Version 5 to Version 5.0.1 47

11.1 Migrating Web projects Version 5 to Version 5.0.1	47
11.2 Converting Web projects to Struts 1.1 Beta 3	47

Chapter 12. Build setup (library, JARs, dependent project JARs, Ant builds) . . 49

12.1 Java library JARs and third-party external JARs	49
12.1.1 Recommended way to use a third-party JAR within a Web project.	49
12.1.2 Recommended way to use a third-party JAR for use with multiple EJB or Web projects.	49
12.1.3 References between Web projects and other EJB projects	50

12.1.4 Alternative way to use external JAR files (global build and server classpath)	50
12.2 Optimizing multi-project builds using Dependent Project JARs	51
12.3 Automated production builds using Ant	51

Chapter 13. Migration examples 53

13.1 Example: VisualAge for Java JSP/servlet sample (LeapYear)	53
13.1.1 Exporting your files from VisualAge for Java	53
13.1.2 Creating a new WebSphere Studio Application Developer Web project	54
13.1.3 Importing the Java and project resource files into the WebSphere Studio Application Developer project	54
13.1.4 Defining any servlets and make any restructured application changes	55
13.1.5 Creating a WebSphere Studio Application Developer server project	55
13.1.6 Testing the migrated LeapYear application	56
13.2 Example: Enterprise beans, VCE, and database samples (HelloWorld session bean and Increment enterprise bean)	56
13.2.1 Exporting the client Java source from VisualAge for Java	57
13.2.2 Exporting the EJB group from VisualAge for Java into an EJB 1.1 JAR	57
13.2.3 Creating a new WebSphere Studio Application Developer EJB project.	58
13.2.4 Importing the EJB 1.1 JAR file into WebSphere Studio Application Developer EJB project	58
13.2.5 Generating and deploying RMIC stub and tie code.	58
13.2.6 Specifying the data source binding information	58
13.2.7 Creating a new WebSphere Studio Application Developer project	58
13.2.8 Creating a WebSphere Studio Application Developer server project	59
13.2.9 Specifying your data source, your EAR project, and then start the server	60
13.2.10 Starting DB2 and connecting to sampleDB	60
13.2.11 Testing the migrated HelloWorld client.	60
13.2.12 Testing the migrated Increment client	61
13.2.13 Testing the migrated Increment and HelloWorld EJBs using the EJB Test Client	61
13.3 Example: WebSphere Studio "Classic" Version 4.0 Web Application (YourCo)(Windows)	61
13.3.1 Starting WebSphere Studio "Classic" Version 4.0 and creating a new Migration stage	62
13.3.2 Creating a Web configuration descriptor file	62
13.3.3 Creating a migration file	62
13.3.4 Starting WebSphere Studio Application Developer and importing the WAR file	63
13.3.5 Creating a WebSphere Studio Application Developer server project	63
13.3.6 Testing the migrated YourCo application	64

13.4 Example: Migrating "Example: VisualAge for Java JSP/servlet sample " EJB 2.0 (Increment enterprise bean and HelloWorld session bean).	64
13.4.1 Create a new EJB 2.0 project and Enterprise Application 1.3 project	64
13.4.2 Import the VisualAge for Java EJB 1.1 JAR into EJB 2.0 project	65
13.4.3 Migrate code from EJB 1.0 to EJB 1.1	65
13.4.4 Migrate code from EJB 1.1 to EJB 2.0	65
13.4.5 Generating and deploying RMIC stub and tie code.	66
13.4.6 Specifying the data source binding information	66
13.4.7 Creating a new WebSphere Studio Application Developer Java client project	67

13.4.8 Creating a server project with an WebSphere Application Server Version 5.	68
13.4.9 Specify your data source, your EAR project, and then start the server	68
13.4.10 Testing the migrated HelloWorld client.	69
13.4.11 Testing the migrated Increment client	69
13.4.12 Testing the migrated Increment and HelloWorld EJBs using the EJB Test Client	69

Chapter 14. Further reading. 71


Notices 73

Programming interface information	75
Trademarks and service marks	75

Chapter 1. WebSphere Studio Application Developer Version 5.0.1 Migration Guide

In this version of IBM^(R) WebSphere^(R) Studio Application Developer Version 5.0.1, you can migrate code from VisualAge for Java, or WebSphere Studio "Classic" or WebSphere Studio Application Developer Version 4.0.x. or WebSphere Studio Application Developer Version 5 Beta.

This guide is organized into the following chapters:

- "Chapter 2. Targeting WebSphere Application Server Version 4.0.x versus Version 5 versus Version 5 Express" on page 3
- "Chapter 3. Migrating from WebSphere Studio Application Developer Version 4.0.x" on page 5
- "Chapter 4. Migrating from WebSphere Studio Application Developer Version 5 Early Availability or Beta Versions" on page 15
-  "Chapter 5. Migrating from WebSphere Studio "Classic" to WebSphere Studio Application Developer" on page 17
- "Chapter 6. Migrating from VisualAge for Java to WebSphere Studio Application Developer" on page 21
- "Chapter 7. Migrating enterprise beans from VisualAge for Java to WebSphere Studio Application Developer" on page 27
- "Chapter 8. Migrating from EJB 1.0 to EJB 1.1 or to EJB 2.0" on page 35
- "Chapter 9. Migrating from VisualAge for Java Visual Composition Editor to Visual Editor for Java" on page 41
- "Chapter 10. Converting from VisualAge for Java Persistence Builder to EJB 2.0" on page 43
- "Chapter 11. Migrating from WebSphere Studio Application Developer Version 5 to Version 5.0.1" on page 47
- "Chapter 12. Build setup (library, JARs, dependent project JARs, Ant builds)" on page 49
- "Chapter 13. Migration examples" on page 53
- "Chapter 14. Further reading" on page 71

Information about using WebSphere Studio Application Developer can be found in the *Getting Started* guide and the online help. Read the *Installation guide* prior to installing WebSphere Studio Application Developer. After you have successfully installed WebSphere Studio Application Developer, read the *Getting Started* guide and complete the *Getting Started* tutorials. The tutorials will introduce you to the workbench, Java^(TM) development, and Web services. After you have completed the tutorials, read this guide to migrate your application resources into WebSphere Studio Application Developer.

This guide is available in both HTML and Acrobat PDF versions, in the /readme directory. Both versions contain the identical information. You can open migrate.html in any Web browser. To open migrate.pdf, you must have installed the Acrobat Reader software, which you can download from www.adobe.com/products/acrobat/readstep2.html.

This *Migration Guide* uses Windows^(R) conventions throughout. For example, "WS_Installdir\" in Windows is equivalent to "WS_Installdir/" in Linux^(TM). For future updates to this guide, refer to www.ibm.com/websphere/developer/zones/studio/transition.html.

Chapter 2. Targeting WebSphere Application Server Version 4.0.x versus Version 5 versus Version 5 Express

One of the significant features of WebSphere Studio Application Developer Version 5 is that it can target both WebSphere Application Server Version 4.0.x or Version 5 or both. This *Migration Guide* focuses on getting application programs migrated into WebSphere Studio Application Developer Version 5 tool, and not on differences in administration, operation, or features/functions between the two WebSphere Application Server versions. So, where this guide has examples with instructions in configuring and running the internal WebSphere Test Environment (an embedded non-production copy of the WebSphere Application Server) it contains instructions for WebSphere Application Server Version 4. WebSphere Application Server Version 5 is very similar, but there are slight differences in configuring data sources, and so on, and there is full online documentation available. There is also a WebSphere Application Server - Express test environment (the only test environment included in the WebSphere Application Server - Express development product), which is essentially equivalent to WebSphere Application Server Version 5 but without support for EJBs.

For information on differences between WebSphere Application Server Version 4 and Version 5 functions, please refer to the WebSphere Application Server Version 5 online InfoCenter documentation. Most applications developed and tested on WebSphere Application Server Version 4 will run unchanged on Version 5. However, if the application uses newer specification level features/functions (J2EE 1.3, Servlet 2.3, Java ServerPages (JSP) 1.2, Enterprise JavaBeans^(TM) (EJB) 2.0, Web J2EE 1.3, and so on) then that application will only work on WebSphere Application Server Version 5.

Note that some wizards in WebSphere Studio Application Developer create J2EE projects or resources, and they default to J2EE 1.3. To change the default to J2EE 1.2, select **Window > Preferences > J2EE > Highest J2EE version used for development**.

Chapter 3. Migrating from WebSphere Studio Application Developer Version 4.0.x

This chapter covers migrating from WebSphere Studio Application Developer Version 4.0.x to Version 5. All the migration information in this chapter is applicable for both Windows and Linux operating system.

There are two supported methods that you can use to migrate your projects from WebSphere Studio Application Developer Version 4.0.x to Version 5. Each of these methods is described in greater detail, below:

- Using a software configuration management (SCM) system such as Concurrent Versioning System (CVS) or Rational^(R) ClearCase^(R). This is the recommended method.
- Exporting your projects from Version 4.0.x and then importing them to this edition. This method migrates everything except project build path information.
- Using your existing Version 4.0.x workspace. This is not supported for reasons explained later.

Note that migrating from Version 4 to Version 5 does *not* automatically change the project J2EE level since Version 5 can still build and deploy to WebSphere Application Server Version 4. Changing the J2EE level of Web projects is simply a matter of selecting the project in the Navigator View and then **Properties > Web > J2EE Level**. Changing an EJB Project level is more involved. Refer to “Chapter 8. Migrating from EJB 1.0 to EJB 1.1 or to EJB 2.0” on page 35. All J2EE project types, including Web projects, can be migrated using the J2EE Migration Wizard available in WebSphere Studio Application Developer. To access the J2EE Migration wizard, right-click on the EJB project and then select **Migrate > J2EE Migration Wizard**. For more information about the different wizard options, press F1 while in the J2EE Migration wizard.

3.1 Differences between WebSphere Studio Application Developer Version 4.0.x and Version 5

The following is a partial list of enhancements since Version 4.0.x:

- WebSphere Studio Application Developer Version 5 can generate code for either WebSphere Application Server Version 4.0 or Version 5, and includes both WebSphere Application Server Version 4.0.4 and Version 5 Test Environments.
- The enterprise applications archives (EARs) J2EE level has changed from 1.2 to 1.3 for WebSphere Application Server Version 5 projects.
 - J2EE 1.2 EARs will run on either WebSphere Application Server Version 4.0.x or WebSphere Application Server Version 5.
- The Enterprise Java Beans (EJB) specification level has changed from 1.1 to 2.0 for WebSphere Application Server Version 5 EARs.
 - EJB 1.1 projects are still supported, and may be part of either WebSphere Application Server Version 4.0.x J2EE 1.2 EARs or Version 5 J2EE 1.3 EARs.
- The Web applications (WARs) J2EE level has changed from 1.2 to 1.3 for WebSphere Application Server Version 5 projects.
 - The JSP level has changed from 1.1 to 1.2 and the Servlet level has changed from 2.2 to 2.3

- J2EE 1.2 Web projects (WARs) are still supported, and may be part of either WebSphere Application Server Version 4.0.x J2EE 1.2 EARs or Version 5 J2EE 1.3 EARs.
- In Version 5, you can create static Web projects as well as J2EE Web projects. In a static Web project you will only be able to create content served by a traditional HTTP server (HTML, JavaScript^(TM), images, text and so on).
- The underlying workbench, which is based on the Eclipse open-source project, has changed from Version 1.0 to Version 2.0.
 - There is a new and much improved Java builder
 - There is a new and much improved VCM interface for SCM vendors
- Page Designer "Classic" and Page Designer are both available. For more information, see the "3.3 Internal changes from Version 4.0.3".

3.2 WebSphere Application Server changes and Servlet/JSP conversion tools

The WebSphere Application Server InfoCenter
[\[www.ibm.com/software/webservers/appserv/doc/v40/aes/infocenter/index.html\]](http://www.ibm.com/software/webservers/appserv/doc/v40/aes/infocenter/index.html)
 has the following information:

- Differences between WebSphere Application Server Version 3.5 and 4.0
[\[www.ibm.com/software/webservers/appserv/doc/v40/aes/infocenter/was/03.html\]](http://www.ibm.com/software/webservers/appserv/doc/v40/aes/infocenter/was/03.html).
- WebSphere Application Server Version 5.0 Migration Guide with information on the differences between WebSphere Application Server 4.0 and 5.0
[\[www.redbooks.ibm.com/pubs/pdfs/redbooks/sg246910.pdf\]](http://www.redbooks.ibm.com/pubs/pdfs/redbooks/sg246910.pdf).

The WebSphere Application Server downloads page
[\[www14.software.ibm.com/webapp/download/product.jsp?s=p&id=TDUN-49EVRT&type=s&dt=DIAGNOSTIC+TOOL\]](http://www14.software.ibm.com/webapp/download/product.jsp?s=p&id=TDUN-49EVRT&type=s&dt=DIAGNOSTIC+TOOL) has tools to help convert your application:

- MigrateWC takes a .91 or 1.0 JSP and converts it to a 1.1 JSP. It also takes a 2.1 Servlet and converts it to a 2.2 Servlet.
- XMLconvert converts XML configuration files from Release 3.02x or Release 3.5x to Release 4.0 format.

3.3 Internal changes from Version 4.0.3

3.3.1 Circular project dependencies will not build by default

If your projects have circular dependencies, Version 5 reports a build error. You can go into **Window > Preferences > Java > Compiler**, select the **Other** tab, and clear the **Stop building when an invalid class path is detected** check box. Note that this will no longer cause the build to stop, but there will still be one or more build 'circular dependency' errors shown on the Task view (even when the build otherwise is successful). In this case, you may change these errors to warnings by selecting the **Other** tab, and then changing the preference in the **Circular Dependencies** drop-down.

3.3.2 Version 5 Web projects are source location compatible with Version 4.0.3

In WebSphere Studio Application Developer Version 5, there are internal project structure changes from Version 4.0.3. A Version 5 J2EE 1.2 Web WAR, when exported with Java source, will import into WebSphere Studio Application

Developer Version 4 and the source code folder will be automatically converted into the right name and will build fine. The Web project still executes correctly on WebSphere Application Server Version 4 similarly when a Version 4 project is imported into Version 5, because the source code folder is automatically converted to the correct name.

Note: The above is not true if the Web projects are shared between the Version 5 and Version 4 through a software configuration management (SCM) system. The Version 4 projects need to be migrated to Version 5 project structure and cannot be loaded back into Version 4 from a SCM system once migrated.

3.3.3 WebSphere Studio Application Developer Web project structures

The internal Web project structure in WebSphere Studio Application Developer Version 5 is different than it was for WebSphere Studio Application Developer Version 4.0.x. This difference is not related to J2EE 1.2 versus J2EE 1.3, but rather it is a tool usability change.

In Version 4, Web projects were J2EE Web projects by default and they appeared in the Navigator view with a **source** folder and a **webApplication** folder. In Version 5, if you create a J2EE Web project, then it will appear with a **Java Source** folder instead of a **source** folder and a **Web Content** folder instead of a **webApplication** folder.

However, if a Version 4 Web project is saved into an SCM repository and then loaded into Version 5, it will retain the old structure with the **source** and **webApplication** folders. Either structure will build correctly in Version 5.

3.3.4 Static versus J2EE Web Projects

In Version 5, you can create static as well as J2EE Web projects.

Static Web projects contain only static resources like HTML, Java Scripts, images, text and so on, and no dynamic content in them. Static Web projects can run and be served by a traditional HTTP Web server and do not need a Web Application Server.

J2EE Web projects contain dynamic J2EE resources such as servlets, JSPs, filters, and associated metadata, in addition to static resources. When you create J2EE Web projects, you can include cascading style sheets and JSP tag libraries, so that you can begin development with a richer set of project resources. J2EE Web projects are always imbedded in Enterprise Application projects and run only on Web Application Servers.

3.3.5 WebSphere Studio Application Developer EJB 1.1 and Application Client 1.2 project structures

The EJB 1.1 (J2EE 1.2) and Application Client 1.2 (J2EE 1.2) internal project structure in WebSphere Studio Application Developer Version 5 is different than it was for WebSphere Studio Application Developer Version 4.0.x. This difference is *not* related to J2EE 1.2 versus J2EE 1.3, but rather is a tool usability change.

3.3.5.1 EJB 1.1 project structures

The default in Version 5 for new EJB 1.1 projects is to have the resulting compiled binary classes in the same `\ejbModule` directories as the source. However, the old Version 4.0.x structure (with source directories under `\ejbModule` and the compiled

binary classes in the \bin directory) is retained in Version 5 for EJB 1.1 projects originally created in Version 4.0.x and then saved into an SCM repository and then reloaded into Version 5 (this avoids having to restructure the SCM directories and to avoid having to change automated build scripts). Either structure will build correctly in Version 5.

3.3.5.2 Application Client 1.2 project structures

The default in Version 5 for new Application Client 1.2 projects is to have the resulting compiled binary classes in the same \appClientModule directories as the source. However, the old Version 4.0.x structure (with source directories under \appClientModule and the compiled binary classes in the \bin directory) is retained in Version 5 for Application Client 1.2 projects originally created in Version 4.0.x and then saved into an SCM repository and then reloaded into Version 5 (this avoids having to restructure the SCM directories and to avoid having to change automated build scripts). Either structure will build correctly in Version 5.

Note that if a Version 4.0.x EJB 1.1 or Application Client 1.2 project is exported as an JAR and then imported into a new Version 5 EJB 1.1 or Application Client 1.2 project, it will have the *new* common source/binary structure.

3.3.6 Page Designer "Classic"

WebSphere Studio Application Developer Version 4 shipped Page Designer "Classic" for Windows operating system. Version 5 contains a new Page Designer, (which is the default for both Windows and Linux operating systems) but also includes Page Designer "Classic" that was shipped with Version 4 for Windows only as an optionally installable component. If you wish to have Page Designer "Classic" bound to HTML/JSP as the default editor, then install Page Designer "Classic" and change your workbench preference.

For Page Designer "Classic" installation instructions, refer to the *Installation Guide*.

To change the default HTML/JSP editor:

1. Select **Window > Preferences > Workbench > File Associations**.
2. In the **File types** field, select ***.html** or ***.jsp** or both or whatever you would like to assign to Page Designer "Classic".
3. In the **Associated editors** field, select **Page Designer Classic** and click **Default**.

If you do *not* want to change the default to become Page Designer "Classic", you can still select an HTML or JSP file, click on it, and then select **Open With > Page Designer Classic**.

3.3.7 Page Designer enhancements

Page Designer now has some enhancements which are not in Page Designer "Classic":

- XHTML is supported for PvC application development
- HTML/JSP tag attribute can be viewed/edited in attributes view
- Link to Servlet available in some dialogs
- Move and copy a table within page by dragging and dropping
- Enhanced JSP tag visualization within Design Page
- Enhanced tag library palette within dialog to insert custom JSP tags
- CSS Preview window in CSS Designer to preview the applied CSS effect

- HTML/JSP thumbnail view

3.3.8 Page Designer "Classic" versus Page Designer functions

Page Designer "Classic" has some HTML editing functions which are not in the current Page Designer:

- JSP extensions for Design-time Control, Dynamic Elements, Author Time Visual
- WebSphere enhancements (Dynamic Table Extensions for DB navigation, Table alternating coloring)
- Dynamic HTML support of Roll over effect, Event/action
- Script editing (JavaScript or VBScript) in Design View or JSP Scriptlet library
- Tag Checker, Accessibility Checker
- WTP Annotation Support
- Insertion of ActiveX controls
- Image Pasting with logo/photo frame creation
- Page editing assist for Ruler or various wizards for component insertion
- URL editor, Heading editor, Table of Contents generation
- WYSIWYG View Bi-Directional (BiDi) language support
- Color gallery prepared by professional designers

3.3.9 HTML and JSP distinctions

- In Version 4.0.x, HTML files and JSP files were treated identically by Page Designer. For example, you could have JSP tags in an HTML file. This is no longer true: in this release, there is a distinction between JSP and HTML files, so you can no longer have JSP tags in an HTML file.
- The previous distinction affects encoding of non-English JSP files. In versions prior to this version, HTML encoding rules were used, even for JSP files, to determine the encoding named in a file. That is, the content type attribute of the meta tag was looked at (`<META http-equiv="Content-Type" content="text/html; charset=UTF-8">`). In this version, this was changed to use JSP encoding rules to determine the encoding named in a JSP file. That is, the page directive of the JSP file is looked at (`<%@page contentType="text/html; charset=UTF-8"%>`). For HTML files, encoding is unchanged from previous versions.

3.4 Migrating projects using a software configuration management (SCM) system

3.4.1 Migrating projects using CVS or Rational ClearCase

This is the recommended way to move workspaces from Version 4.0.x to WebSphere Studio Application Developer Version 5. This is the only method that migrates all of your information, including project build path information.

1. As a backup precaution, save all your Version 4 projects into your SCM repository. Then commit (release) any pending changes.
2. Still working in Version 4, save your work again into a new Version 5 branch (stream). This is the branch that you will use when working with the Version 5.
3. Install the Version 5.
4. Close WebSphere Studio Application Developer Version 4 and start WebSphere Studio Application Developer Version 5.

Tip: In Version 4, the workspace directory was located in the installation directory, by default. In Version 5, this default has changed to a directory called

workspace in the My Documents directory. If you wish to override the location where you work is stored, use the `-data` option on the `wsappdev.exe` command when you start the workbench.

Note: Do not use `-data` to point to an existing Version 4 workspace since that is a different unsupported approach to migration. (For more information, refer to the "Migrating projects using an existing Version 4.0.x workspace".)

5. Disable **Windows > Preferences > Workbench > Perform build automatically on resource modification** (to avoid build errors as individual dependent projects are loaded).
6. **For CVS:** Load all of the projects that you want to work with from the SCM repository into WebSphere Studio Application Developer Version 5.
For ClearCase: Use a clean Version 5 workspace, then for each project to be loaded, select **File > Import > Existing WebSphere Studio 4.x ClearCase Project into Workspace**.
7. Restore your desired setting for **Windows > Preferences > Workbench > Perform build automatically on resource modification**.
8. Change the **source** folder name from **source** to **JavaSource** and **webApplication** folder to **WebContent** for the Web projects if a full build is needed. Otherwise, the old folder structure is retained and the web projects will *not* be fully rebuilt.
9. Do a full rebuild (**Project > Rebuild all**), and save the resulting projects back into your repository in your new Version 5 stream. (Do *not* mix these resources with your ongoing Version 4 stream.)

Note: These projects are now Version 5 projects and *cannot* be used by WebSphere Studio Application Developer Version 4.0.x.

Post migration considerations:

- Version 4 CVS files were stored as binary (no CarriageReturn/LineFeed translations). If you work in a mixed (DOS/Windows plus UNIXTM/Linux) platform environment, you might wish to now mark source files as text (using **Team > CVS > Change ASCII/Binary Property**) and resave them in CVS.
- Version 4 Web projects from a CVS repository require **Window > Preferences > Team > CVS > Prune empty directories** setting to be disabled (the default is that it is enabled). If it is *not* disabled, and you load a Web project with an empty **source** folder (like the MyHomePage Web example), then you will get the following errors at check in:

The project was not built since it is involved in a cycle or has classpath problems.
Missing required source folder: /MyHomePageExample403/source.
- For Web projects saved and loaded from a ClearCase repository, you need to have a file checked out before you can open it in the editor. If it is not checked out you receive errors error activating this view (Null pointer exception in logs for Page Designer). With web.xml editor, editing a web.xml file you need to have web.xml, ibm-web-bnd.xml, and ibm.web-ext.xmi checked out. (There are indications that you need these files to be checked out on the status line, which states that they are read only, but it is easily missed.)
- If your projects have circular dependencies, Version 5 reports a build error. You can go into **Window > Preferences > Java > Compiler**, select the **Other** tab, and clear the **Stop building on build path errors** check box.
- The `.vcm_meta` (or `.cc_meta`) files could now be deleted from the Version 5 project because they are not used by Version 5 (it uses a new `.project` file

instead) and because you are using a new repository branch (stream) for these Version 5 projects. Note that these files are still needed in the ongoing Version 4 branch (stream).

3.4.2 Post-Migration removal of EAR and Server Configuration absolute path references

Version 4 EAR IBM application extension files and server configuration files contained absolute path references. After you have migrated them into Version 5, you need to open them with their editor (which will automatically change their old absolute path references into new relative references).

1. For each EAR project, in a Navigator View, right-click **META-INF/application.xml** > **Open with** > **Deployment Descriptor Editor**.

- a. A dialog window pops up with the message:

The IBM extensions file contains deprecated absolute paths.
This can be auto-corrected and should be saved. This will
remove the paths from the file, and only needs to be done once.
Would you like to autocorrect?

- b. Click **Yes**.
- c. Save and then close the editor window.

Note: Alternatively, you can use the J2EE Migration wizard to migrate the project structure only for an EAR project. To access the J2EE Migration wizard, right-click on the EAR project and then select **Migrate > J2EE Migration Wizard**. For more information about the different wizard options, press F1 while in the J2EE Migration wizard.

2. For each Server configuration, in a Server Perspective, Server Configuration View, right-click on the server, and then select **Open**.
 - a. You will get a similar autocorrect dialog.
 - b. Click **Yes**.
 - c. Save and then close the editor window.

3.4.3 Migrating projects using other SCMs

There are other SCM vendors who provide SCM plug-ins for WebSphere Studio Application Developer. Please browse the list of vendors at www.ibm.com/software/ad/studioappdev/partners/scm.html. As part of their *Ready for WebSphere Studio Software* [www.developer.ibm.com/websphere/ready.html] validation, all SCM vendors who provided a Version 4 plug-in will have ensured that the preceding migration steps (save from Version 4 to SCM repository, load from repository into Version 5) also work for their systems.

3.5 Migrating by exporting and importing your projects

1. In WebSphere Studio Application Developer Version 4.0.x, export your projects to a WAR file, an EAR file, or a JAR file (**File > Export**).
2. In WebSphere Studio Application Developer Version 5, import your WAR file, an EAR file, or a JAR file (**File > Import**).

Note: This is not a full migration since no project build path information is maintained.

3.6 Migrating projects using an existing Version 4.0.x workspace

This approach is partially supported, and will result in an incomplete migration. User interface settings, debug settings, and most preferences are all lost. Project names, project source files, and project Java build path (class path) are retained, but nothing else is guaranteed. This approach should only be used if no supported SCM system is being used and if it is critical to retain project build path information, which is lost when you import projects that were exported from Version 4. You can use the existing Version 4.0.x workspace by doing the following:

1. Commit (release) any pending changes to the repository.
2. Close all perspectives and shutdown WebSphere Studio Application Developer Version 4.
3. Back up the contents of *workspace_directory*, where *workspace_directory* is the fully qualified directory name that contains the Version 4.0.x workspace. By default, the Version 4.0.x workspace subdirectory is located in the same directory where the product is installed. You will need this backup if you ever want to work with WebSphere Studio Application Developer Version 4.0.x again. Once you have pointed to a Version 4.0.x workspace from a Version 5 IDE, you can no longer go back to using that workspace in WebSphere Studio Application Developer Version 4.0.x.
4. Install WebSphere Studio Application Developer Version 5.
5. When you start WebSphere Studio Application Developer Version 5 with a Version 4.0.x workspace from a command prompt (that is, use the `-data` option to specify a fully qualified path to the Version 4.0.x workspace directory on the `wsappdev.exe` command), that will cause an upgrade of the .metadata information.
6. When prompted to confirm that you wish to convert to the new user interface format, click **OK**.
7. Before doing any rebuilds or validating any projects that are in the workspace, select all of the projects in the Navigator view within the Resource perspective and then select **Refresh** from the pop-up menu. This will ensure that all files are synchronized with their appropriate metadata.
8. Open any closed projects (see known problems below).
9. Verify your class path variables (see known problems below).
10. Some builders and validators have been added, removed, or modified in this Version 5. To ensure that the correct errors and warnings are shown, you must rebuild all the projects by selecting **Project > Rebuild All**, and then select **Run Validation** for each Java project.
11. Some user preferences might be maintained, but many others will not be. Check your preference settings in Version 5 to be sure that they are as you want them.

3.6.1 Post-Migration removal of EAR and Server Configuration absolute path references

The post-migration instructions described in “3.4.2 Post-Migration removal of EAR and Server Configuration absolute path references” on page 11 also apply here.

3.6.2 Known problems and limitations

The following problems may occur if you attempt to migrate by opening a Version 4.0 workspace in WebSphere Studio Application Developer Version 5.

3.6.2.1 Incorrect value in the JRE_LIB class path variable

To reset your JRE_LIB class path variable to a valid location, follow these steps. *Do this even if the value seems correct* when you first open the Preferences window.

1. Select **Window > Preferences > Java > Installed JREs**.
2. In the list, select the check box for the default JRE location that you wish your JRE_LIB set to.
3. Choose **Edit**, and then click **OK** to close the Edit JRE dialog box.

If you do not do this, the value for JRE_LIB might be incorrect, causing many build errors in Java files.

As a general check, verify the value of all your other class path variables.

3.6.2.2 For previously SCM shared projects, the Team menu contains Share Project

Team support has changed significantly between Eclipse 1.0 and 2.0. The method of sharing projects with the repository has changed as well.

- If you select the **Team > Share Project** option then a wizard will guide you through the migration process. When you are finished, your project will be shared and the Synchronize view will open. You will see conflicting changes on every file. This is due to changes in the way sync information is stored between Eclipse 1.0 and 2.0.
- If you do not have any outgoing changes (which you should not have if you committed all your outgoing changes before upgrading as recommended above), then you can simply select the project in the Synchronize view and select **Override and Update** which will load the current contents from the server.
- If you do have outgoing changes, you can pull down the triangle menu in the Synchronize view and select **Compare File Contents**. After some work, the Synchronize view will show you only the files which are actually different. You can then use the Synchronize view to resolve these conflicts.

3.6.2.3 Projects created outside the workspace directory

By default, projects are created in the workspace directory. If you override the default to create projects elsewhere, open all of your projects now before closing the workbench. This will allow the .project file for that project to be written in the proper location. Failure to open a closed project whose directory is outside of the workspace will result in a project that masks the actual project, with only a .project file existing within it.

3.6.2.4 JSP breakpoints must be reset

You will need to remove any JSP breakpoints that you have, and reset them within the migrated Version 5 workspace.

3.7 Migrating J2EE project structures and/or J2EE specification levels

As mentioned above, the EJB project and Web project internal structures are different between WebSphere Studio Application Developer Version 4 and Version 5. New projects in Version 5 will always use the new internal structure. Although the old Version 4 structures will continue to work in Version 5, you can optionally convert any J2EE EAR, EJB, WAR, or application client project to the new structure in Version 5 using the J2EE Migration Wizard. The J2EE Migration Wizard allows you to perform the following migration activities:

1. Migrate the old Version 4 project structure to Version 5 project structure.
2. Migrate the J2EE 1.2 specification level to J2EE 1.3 specification level for J2EE EAR, EJB, Web, or application client project.

3. Convert CMP 1.x beans to CMP 2.x and add Local Client Views for EJB projects.

To access the J2EE Migration wizard in Version 5, follow the steps below:

1. Select the project.
2. Right-click on it and then select **Migrate > J2EE Migration Wizard**. Follow the steps in the wizard to guide you through migration.
3. If your project is under source control, then save the restructured project in your SCM.

For more information about the different wizard options, either press F1 while in the J2EE Migration Wizard or read the online help section title "Migrating application modules from J2EE 1.2 to J2EE 1.3".

Chapter 4. Migrating from WebSphere Studio Application Developer Version 5 Early Availability or Beta Versions

WebSphere Studio Application Developer Version 5 Beta was a limited availability Beta, and WebSphere Studio Application Developer was available to Passport Advantage^(TM) subscribers.

No migration is required. Either Version 5 Early Availability or Version 5 Beta workspaces may be used (or shared) directly with Version 5 General Availability. Version 5 Beta projects in Software Configuration Management (SCM) systems (CVS or Rational ClearCase) may be used (or shared) directly with Version 5 General Availability.

Note: Sharing Version 5 Early Availability or Version 5 Beta workspaces with Version 5 General Availability is *not* recommended. A one time step up to Version 5 General Availability from Version 5 Early Availability or Version 5 Beta and continue future work in General Availability is the recommended approach.

Chapter 5. Migrating from WebSphere Studio "Classic" to WebSphere Studio Application Developer

This chapter documents how to migrate from WebSphere Studio Version 4.0 (both Advanced and Professional Edition) to the WebSphere Studio Application Developer. Migrating from WebSphere Studio "Classic" Version 4.0 to WebSphere Studio Application Developer Version 5.0 involves the following steps:

1. Create a new single-server stage for migration.
2. Create a Web configuration descriptor file.
3. Export a migration JAR file.
4. Import the migration JAR file into WebSphere Studio Application Developer.
5. Set up your server and test your migrated application.

Note: The following instructions are for migrating from WebSphere Studio Version 4.0. If you want to migrate from an earlier version of WebSphere Studio, you should migrate to WebSphere Studio 4.0 first, then migrate to WebSphere Studio Application Developer.

The advanced publishing feature (mapping files to publishing stages) and the Page Detailer feature (analysis of web pages) of WebSphere Studio "Classic" is not available in WebSphere Studio Application Developer. Some other features from the Version 4.0.x CD media pack are also no longer available. For example, the Page Detailer feature for analysis of web pages, the HotMedia^(R) feature for rich media types, the Voice XML editor (moved to WebSphere Everyplace^(TM) Toolkit and Portal Toolkit), DataBaseWizard for pervasive devices.

You should be aware of the following limitations before you migrate any of your WebSphere Studio data:

- WebSphere Studio Application Developer uses an XML-based SQL editor, so your .sql files cannot be used in it.
- Project publishing information and stage information cannot be migrated.
- WebSphere Studio server configuration information cannot be migrated.
- Version control information cannot be migrated.

During the migration process outlined below, WebSphere Studio creates a JAR file that contains all of your project files, publishable and source, for a single server. All the files visible in the Publishing view for the default server will be packaged into the JAR file. You can then import the JAR file into WebSphere Studio Application Developer.

When you migrate existing projects, all the project publishing information and the stage information are lost during the migration. If your stage has multiple servers, only files published to the default server are kept. Therefore, for the purpose of migration, you will create a new stage that has only one server.

5.1 Creating a new single-server stage for migration

If you have more than one server in your current stage, create a new stage called Migration with only one server by following these steps:

1. Click **Project > Customize Publishing Stages**.
2. Type Migration in the **Stage name** field.
3. Click **Add**.
4. Click **OK**.
5. Click **Project > Publishing Stage** and select **Migration** from the list of available stages.
6. While in the publishing view, click **Insert > Server**.
7. Type a server name, such as localhost.
8. Changing the server or changing the publication stage does not propagate the servlet mapping information for WebSphere Application Server Version 4.0. Go to the Publishing view, and, for each servlet, click **Properties > Publishing > Servlet Mapping** and then copy the actual servlet mapping.

5.2 Creating a Web configuration descriptor file

1. While in the project file view, click **Project > Create Web Configuration Descriptor File**.
2. Select all required servlets.
3. Select all required Tag Library Descriptor (TLD) files.
4. Click **Create**.

The default Web configuration descriptor file name is *serverName_web.xml*, localhost_web.xml in this scenario. Unless you specified a different location, the .xml file is saved in the WEB-INF directory.

5.3 Exporting a migration JAR file

1. While in the project file view, select server **localhost** and click **Properties > Publishing > WebApp Web Path** and enter a web path (context root), such as myWebPath. This will also be used as the WebSphere Studio Application Developer project name.
2. While in the project file view, select **Project > Create Migration file**.
3. Verify that **localhost** is the selected server.
4. Verify that **localhost_web.xml** is the selected Web configuration descriptor file.
5. Click **OK**.
6. The default JAR file name is serverName.jar, localhost.jar for this scenario. Rename the file if desired.
7. Save the JAR file.

5.4 Importing the migration JAR file into WebSphere Studio Application Developer

1. Start WebSphere Studio Application Developer.
2. Create a Web project (**File > New > Project > Web Project**).
3. In the **Project name** field, type the name of your Web project. This should be the same name you specified in step 1 of the preceding "Exporting a migration JAR file".

4. Specify the name of a new or existing EAR project that will contain the new Web project for purposes of deployment.
5. In the **Context Root** field, type the Webapp Web Path name you specified when you created the migration JAR file in WebSphere Studio. Click **Finish**.
6. In the Navigator view, select the Web project you just created.
7. Import the JAR file.
 - a. Click **File > Import**.
 - b. Click **WAR file**. Click **Next**. You *must* import the JAR file using the WAR file option; otherwise it will not work properly.
 - c. Enter the path to localhost.jar in the **WAR File** field or click **Browse** to search for it. (You can only browse for a .WAR name, not a .JAR name.)
 - d. Select the existing Web project that you created. The **Context Root** field is automatically populated with the value you specified earlier.
 - e. Click **Finish**. A dialog appears asking "Resource WEB-INF/web.xml already exists. Would you like to overwrite it?".
 - f. Select **Yes** and WebSphere Studio Application Developer unpacks *localhost.jar*.
8. You may have several unresolved references or missing import files. These will appear in the Tasks view. To fix this, you must change the Java build path for the Web project:
 - a. Right-click the project and click **Properties > Java Build Path**.
 - b. Click the **Libraries** tab. Click **Add External JARs**.
 - c. Import any JARs that you need from the following directories:
 - WS_Installdir/runtimes/aes_v4/lib and
 - WS_Installdir/runtimes/base_v4/lib
9. In the Navigator view, right-click the project and select **Rebuild Project**.

5.5 Testing your migrated application on a test server

You are now ready to test your application. To test it on the default test server, follow these steps:


1. Right-click the EAR project.
2. Select **Run on Server**

To test your application on other server run-time environments, refer to the online help for the Server Tools feature.

Chapter 6. Migrating from VisualAge for Java to WebSphere Studio Application Developer

This chapter documents how to migrate from VisualAge^(R) for Java Professional Edition or VisualAge for Java Enterprise Edition to WebSphere Studio Application Developer.

Note: The instructions given in this chapter are for migrating from VisualAge for Java Version 3.5.3 or 4.0 for Windows. If you want to migrate from an earlier version of VisualAge for Java to WebSphere Studio Application Developer, you should first migrate from your earlier version of VisualAge for Java to Version 4.0 for Windows (if you have enterprise beans) or Version 3.5.3 or 4.0 for Windows (without enterprise beans), before migrating to WebSphere Studio Application Developer. Version 4.0 includes a special tool (the EJB Export Tool) for exporting enterprise beans, which you need in order to migrate EJB applications to WebSphere Studio Application Developer. All other data can be migrated from either Version 3.5.3 or Version 4.0.

Note:  Instantiations, Inc., an IBM Business Partner, distributes a product, called CodePro Studio that provides productivity enhancements to VisualAge for Java and WebSphere Studio Application Developer, including migration and co-existence facilities. To help VisualAge for Java customers begin their migration, Instantiations is offering a free, unlimited use VisualAge for Java to WebSphere Studio Application Developer export facility as part of their time-limited evaluation copy of CodePro Studio. You can download the evaluation copy from www.instantiations.com/vaj-migrate. For further information on Instantiation's advanced migration and co-existence support including full bi-directional export/import of files, creation of export/import sets, project synchronization and task automation, please browse Instantiations, Inc. www.instantiations.com/codepro/ws.

6.1 Differences between VisualAge for Java and WebSphere Studio Application Developer

The following is a partial list of changes from VisualAge for Java:

- The Enterprise Java Beans (EJB) specification level has changed from 1.0 to 1.1 (EJB 2.0 is also supported for applications that will be deployed to WebSphere Application Server Version 5).
- For Web applications, the JSP level remains at 1.1 (1.2 for WebSphere Application Server Version 5 applications).
- For Web applications, the Servlet level remains at 2.2 (2.3 for WebSphere Application Server Version 5 applications).
- The level of the Java 2 platform that is supported has changed from 1.2 to 1.3. (The compiler can target 1.4 code generation, but the WebSphere Application Server run-time environment is still 1.3.)
- The Visual Composition Editor has been replaced by the Visual Editor for Java.
- VisualAge for Java version control and the proprietary source code repository have been replaced by support for software configuration management (SCM) plug-ins.

- The VisualAge for Java Tools API has been replaced by the WebSphere Studio Workbench plug-in architecture.
- The VisualAge for Java XML tools have been replaced by WebSphere Studio Application Developer XML tools.
- The VisualAge for Java project concept has been replaced by multiple types of WebSphere Studio Application Developer projects.
- Convertors in VisualAge for Java EJB access beans (*not* Data access beans) are not available in WebSphere Studio Application Developer EJB access beans. Use EJB convertors/composers in the underlying EJBs instead. For more information about EJB convertors/composers, refer to the online help documentation.

6.2 Migrating from VisualAge for Java

The following steps outline how to migrate from VisualAge for Java. Details on how to perform these steps are provided below:

1. “6.2.1 Exporting your Java files and project resource files from VisualAge for Java”
2. “6.2.2 Starting WebSphere Studio Application Developer and creating new projects to contain your code” on page 23
3. “6.2.3 Importing your Java and resource files into WebSphere Studio Application Developer” on page 23
4. “6.2.4 Using the web.xml editor to ensure that servlets are correctly defined (Web project only)” on page 23
5. “6.2.5 Migrating project and workspace settings” on page 24
6. “6.2.6 Setting up your WebSphere V4 test environment and testing your migrated application(s)” on page 25
7. “6.2.7 Deploying your applications from WebSphere Studio Application Developer to remote WebSphere Application Server” on page 26
8. “6.2.8 Sharing WebSphere Studio Application Developer project settings between multiple developers (post-migration)” on page 26

6.2.1 Exporting your Java files and project resource files from VisualAge for Java

There is no support for the bulk migration of versioned projects and resources from the VisualAge for Java repository. You can only migrate projects and resources that are in your VisualAge for Java workspace. If you want to migrate a versioned copy of a project or resource into WebSphere Studio Application Developer, you must bring it into your VisualAge for Java workspace and then migrate it.

Note: If your project contains more than one kind of data (for example, enterprise beans and Java source code files), you should split up your data into different JARs based on their type.

Export your projects to a JAR file by following these steps:

1. If the projects that you want to export are not currently in your VisualAge for Java workspace, add them to the workspace now.
2. In the VisualAge for Java Workbench window, select your project(s), right-click, and click **Export**.
3. Select the **Jar file** radio button and click **Next**.
4. Specify the name of the JAR file.

5. Select the **.java** check box to export your Java files and the **resources** check box to export your resource files.
6. Fill in the other fields as required. Refer to the VisualAge for Java online help for more information on how to perform this task.

6.2.2 Starting WebSphere Studio Application Developer and creating new projects to contain your code

Start WebSphere Studio Application Developer, then create the appropriate projects. The following is a set of general migration guidelines to help you decide which kind of WebSphere Studio Application Developer project you should import your files into:

- If your code is part of a Web application, you should import the code into a Web project:
 - Import all Java files into the Web project **JavaSource** directory (the proper hierarchy based on their package statements will automatically be created by WebSphere Studio Application Developer)
 - Import all resource files into the Web project **WebContent** directory.
- If your code is straight Java, (for example, an application that will run stand-alone) you should import the code into a Java project.
- If your code is enterprise beans, you should import the code into an EJB project.

Note: The preceding is only a general set of guidelines to help you decide which kind of WebSphere Studio Application Developer projects you should use. We recommend that you read the WebSphere Studio Application Developer online help and become familiar with the different kinds of WebSphere Studio Application Developer projects before you create any projects or import any code.

6.2.3 Importing your Java and resource files into WebSphere Studio Application Developer

1. Open WebSphere Studio Application Developer and switch to the Resource perspective.
2. Click **File > Import > Zip file**. Click **Next**.
3. Browse to the appropriate JAR file.
4. Select the files you want to import and the project or folder you want to contain your files.

Note: When you import your files into WebSphere Studio Application Developer, you should ensure that they go in the appropriate directory. We recommend that you read the WebSphere Studio Application Developer online help and become familiar with the different kinds of WebSphere Studio Application Developer projects before importing your code. This will help you determine which folders should contain which kind of code.

6.2.4 Using the web.xml editor to ensure that servlets are correctly defined (Web project only)

If your application uses servlets, then you need to define the servlet-URL mappings in the web.xml file. Follow these steps:

1. In the Web perspective, open the web.xml file, which is located in the Web Content/WEB-INF subdirectory of your Web project.
2. Click the **Servlets** tab.

3. Click **Add**, and select the **Servlet** radio button.
4. Type the servlet name and click **OK**.
5. Click **Browse** to change the **Servlet class** value to the appropriate package name.
6. (Optional) The display name is a short name used to identify the servlet. In the **Display name** field, type a short name for the servlet.
7. A URL mapping defines a servlet and a URL pattern. Click the **Add** button located next to the **URL mappings** field, then type the name of the URL mapping.
8. Save the changes (**File > Save web.xml**) and close the web.xml file.

6.2.5 Migrating project and workspace settings

You must record the following VisualAge for Java settings and set them up in WebSphere Studio Application Developer:

- Project class path
- Resource associations
- Code formatting
- EJB server configuration
- WTE configuration
- Java files and project resource files

Project class path

In VisualAge for Java, you set your project class path in the Resources pages of the Options window (**Window > Options > Resources**). After you have migrated your projects into WebSphere Studio Application Developer, you can set up your project's class path in the project's Properties window (Right-click the project and select **Properties > Java Build Path**. Click the **Libraries** tab.) You can also set class path variables in the Preferences window (**Window > Preferences > Java > Classpath Variables**.)

Resource associations

If you set up an association between a file type and an executable, you can open a file that sits outside the workbench from within it.

In VisualAge for Java, you set up your resource associations in the Options window (**Window > Options > Resources > Resource Associations**). After you have migrated your resource files to WebSphere Studio Application Developer, you can set up your resource associations using the Preferences window (**Window > Preferences > Workbench > File Associations**).

Code formatter

In VisualAge for Java, you set up your code formatting options in the Formatter page of the Options window (**Window > Options > Coding > Formatter**). After you have migrated your code to WebSphere Studio Application Developer, you can set up your code formatting in the Preferences window (**Window > Preferences > Java > Code Formatter**).

EJB server configuration

In VisualAge for Java, you set up your EJB server configuration in the Properties window for the EJB server. (In the EJB page, select **EJB > Open To > Server Configuration**. Select the server, right-click it, and click **Properties**). After you have migrated your enterprise beans to WebSphere Studio Application Developer, you can set up your server configuration in the Server Configuration view (In the **Server** perspective, open the **Server Configuration** view. Right-click the server and click **Open**. Click the **Data source** tab.)

WTE configuration

In VisualAge for Java, your WebSphere Unit Test Environment and WebSphere Application Server run time settings are in various property files in the following directory: *VisualAgeInstallDir\ide\project_resources\IBM WebSphere Test Environment\properties*, where *VisualAgeInstallDir* is your product installation directory.

If, for example, you have enabled URL rewriting in the session.xml property file by changing the property to true as shown below, `<url-rewriting-enabled>true</url-rewriting-enabled>` you can configure this property in the WebSphere Studio Application Developer Version 4.0 Test Environment. (In the Server perspective, open the Server Configuration view, right-click the server you want to work with and click **Open**. Click the **Web** tab and select the **Enable URL rewrite** check box).

Java files and project resource files

The property file default.servlet_engine contains the `<root-uri>` context root of the VisualAge for Java web application(s). When creating a Web Project in WebSphere Studio Application Developer the **Create a Web Project** dialog contains a **Context root** field for this data.

Windows Web application settings in files such as *VisualAge\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\servlets\default_app.webapp* that you have customized yourself should be migrated to *your_Web_project\Web Content\WEB-INF\web.xml* file in WebSphere Studio Application Developer. For example, if you have changed servlet names and servlet paths in the default_app.webapp file, you would make the corresponding changes in your web.xml file.

Linux Web application settings in files such as *VisualAge\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\servlets\default_app.webapp* that you have customized yourself should be migrated to *your_WebSphere_Studio_Web_project/Web Content/WEB-INF/web.xml* file in WebSphere Studio Application Developer. For example, if you have modified servlet names and servlet paths in the default_app.webapp file, you would make the corresponding changes in your web.xml file.

6.2.6 Setting up your WebSphere V4 test environment and testing your migrated application(s)

If the application is a Java project, then you just use the normal WebSphere Studio Application Developer **Run** or **Debug** support for Java projects to test it.

If the application uses WebSphere Application Server (Web projects and EJB projects), then test it using the built-in WebSphere Application Server. This requires that a default test server be created and started. For an EJB project, right-click the EJB project and select **Run on Server** to run the EJB Test Client. For a Web project, right-click on the main HTML page, and select **Run on Server** to launch the web browser.

For information on testing other types of projects, refer to the online help.

6.2.7 Deploying your applications from WebSphere Studio Application Developer to remote WebSphere Application Server

If you are using the WebSphere Application Server as your run-time environment, deploy your application using the Server Tools feature of WebSphere Studio Application Developer.

6.2.8 Sharing WebSphere Studio Application Developer project settings between multiple developers (post-migration)

WebSphere Studio Application Developer projects (and their associated settings) can be shared between developers. To do this, save a project into the WebSphere Studio Application Developer software configuration management (SCM) server, then extract it onto another team member on the WebSphere Studio Application Developer.

6.3 Team support in WebSphere Studio Application Developer

For information on team support in WebSphere Studio Application Developer Version 4.0, refer to www.ibm.com/websphere/developer/library/techarticles/0108_karasiuk/0108_karasiuk.html

There is also information in the Installation guide and online help about team support in WebSphere Studio Application Developer.

Chapter 7. Migrating enterprise beans from VisualAge for Java to WebSphere Studio Application Developer

This chapter provides instructions on how to migrate your VisualAge for Java enterprise beans to WebSphere Studio Application Developer. Migrating VisualAge for Java enterprise beans to WebSphere Studio Application Developer involves the following steps:

1. Export VisualAge for Java enterprise beans as a 1.1 JAR file using the VisualAge for Java EJB Export tool.
2. Import the EJB 1.1 JAR file into WebSphere Studio Application Developer.
3. Generate deploy code and data source binding information.
4. Create a server configuration and instance.
5. Add data source information to your configuration.
6. Test the enterprise beans with the new EJB test client.

This EJB migration chapter also contains information about:

- Items migrated
- Code changes due to EJB 1.0 specification versus EJB 1.1 specification
- VisualAge for Java Version 3.5.3 EJB 1.0 JARs versus VisualAge for Java Version 4.0 EJB 1.1 JARs
- Moving multiple VisualAge for Java EJB groups into WebSphere Studio Application Developer EJB projects

7.1 VisualAge for Java EJB Export Tool (migrating map/schema from EJB 1.0 to EJB 1.1)

VisualAge for Java Version 4.0 includes a special tool (the EJB Export Tool) for migrating enterprise beans, therefore, all enterprise beans must be migrated from that version to WebSphere Studio Application Developer. Note that the tool migrates VisualAge for Java EJB 1.0 map/schema metadata into EJB 1.1 map/schema XML files, but does *not* change the code in any way (it is still EJB 1.0 code).

Note: It is recommended that all of your database tables contain a table qualifier prior to exporting using the VisualAge for Java Version 4.0 EJB Export Tool. Most databases require a table to have a qualifier and the WebSphere Studio Application Developer workbench does not handle empty qualifiers when one is expected for the database vendor.

There is an Export tool for EJB 1.1 FixPak (be sure to get the latest version 4.1.5 or later), available from the VisualAge Developer's Domain (see www7.software.ibm.com/vad.nsf/Data/Document4624) which fixes certain defects with the tool. One defect the FixPak fixes is a problem the EJB tool has if a group has relationships with two or more ForeignKeys mapping to the same two or more tables. Refer to the FixPak's README file for more information about the defects the FixPak fixes.

7.1.1 Items migrated

The following items are migrated when you migrate your enterprise beans:

- Entity beans (CMP, BMP)
- Session beans (both stateful and stateless)
- Finder helpers (method and where custom finders)
- Inheritance (with multi-levels)
- Associations
- Mappings and schemas
- Deployment and control descriptions

7.2 VisualAge for Java Version 3.5.3 EJB 1.0 JARs versus VisualAge for Java Version 4.0 EJB 1.1 JARs

In VisualAge for Java Version 3.5.3 you can generate an undeployed EJB 1.0 JAR, or generate a deployed EJB 1.0 JAR (with deployment code, stubs, tie code, and so on). You can import the EJB 1.0 JAR into the WebSphere Application Server Version 4.0 Application Assembly Tool (AAT) and use that to generate deployment and control descriptors and to run the EJB Deploy Tool to generate deploy code, stubs, and so on into an EJB 1.1 format JAR (enterprise applications archives). The net result of the previous steps is that you can have a deployed JAR that can be run in WebSphere Application Server Version 4.0, and hence can also be run in WebSphere Studio Application Developer (since its WebSphere test environment is a WebSphere Application Server Version 4.0 server). However, run-time execution does not imply ongoing development capability.

You can include the previously generated Java code into the corresponding EJB JARs and import that into WebSphere Studio Application Developer, and WebSphere Studio Application Developer would parse and understand the EJB session support, security information, finder information, and assembly descriptor (const methods, and so on). However, extensions (class inheritance and ForeignKey associations) and the map and schema information are all lost. They can only be retained if you use the VisualAge for Java Version 4.0 EJB Export Tool, to create an EJB 1.1 JAR containing the map and schema XML information and extension XML information. As well, the names of generated stub classes are different in 1.0 than 1.1 enterprise beans. The net result is that neither an EJB 1.0 JAR nor an EJB Deploy Tool EJB 1.1 JAR retains the basic design metadata to allow them to be used for ongoing development in WebSphere Studio Application Developer.

The VisualAge for Java Version 4.0 EJB Export Tool produces an EJB 1.1 JAR that contains all the EJB design metadata, including the map and schema information and the extensions (class inheritance and ForeignKey associations), so that you can immediately continue to develop within WebSphere Studio Application Developer and regenerate deployment code anytime that is required. This is the only supported method of migrating VisualAge for Java enterprise beans into WebSphere Studio Application Developer; and the use of the VisualAge for Java Version 4.0 EJB Export Tool is the only method that will work for ongoing EJB development within WebSphere Studio Application Developer.

7.3 Moving multiple VisualAge for Java EJB groups into WebSphere Studio Application Developer EJB projects

In VisualAge for Java, you may have multiple EJB groups, with each group being used for intragroup inheritance or associations. Typically, you would export each VisualAge for Java EJB group (using the VisualAge for Java 4.0 EJB Export tool) into a matching JAR, and then import that JAR into a matching WebSphere Studio Application Developer EJB project.

WebSphere Studio Application Developer EJB projects correspond to EJB modules. Where intragroup inheritance or associations are not required, you may wish to keep the EJB groups as separate EJB projects - you can close individual projects and hence keep the overall memory requirements down. WebSphere Studio Application Developer has the concept of enterprise-level grouping, using .WAR and .EAR files, so several EJB groups can be logically combined that way, and then deployed as if they were a single unit.

If you import multiple EJB groups into the same EJB project, then any XML in them may not be correctly merged (the last imported group overwrites).

7.4 Migrating your enterprise beans

The following steps outline how to migrate your enterprise beans. Details on how to perform these steps are provided below:

1. Export them as a 1.1 JAR using the VisualAge for Java Version 4.0 EJB Export tool.
2. Import them into WebSphere Studio Application Developer.
3. Generate deploy code and data source binding information.
4. Create a WebSphere Studio Application Developer server configuration for your test server, if you do not already have one.
5. Add data source information to your server configuration.
6. Test the enterprise beans with the new EJB test client.

7.4.1 Exporting your enterprise beans

1. In VisualAge for Java Version 4.0, add the **Export Tool for Enterprise Java Beans 1.1** feature to the workspace.
2. In the EJB page of the Workbench, right-click the EJB group that you want to export, and click **Export > EJB 1.1 JAR**.
3. Fill in the fields as necessary (ensure that the **.java** check box is selected), select your database, and click **Finish**.

7.4.2 Importing your enterprise beans into WebSphere Studio Application Developer

1. In WebSphere Studio Application Developer, create a new EJB 1.2 project and a new enterprise applications archives project. You will automatically be switched to the J2EE perspective.
2. Select **File > Import > EJB JAR file**. Click **Next**, then select your JAR file, your EJB project, and your .EAR file. Click **Finish**.
3. If you have any errors (they will be listed in the **Tasks** view), refer to “7.5 Known problems and workarounds” on page 31 below to troubleshoot them.

4. After you have imported your EJB files, follow the instructions in “7.6 Locating EJB information” on page 31 to help you find out where to find your EJB and method properties, your schemas, maps, and so on.

7.4.3 Generating deploy code and data source binding information

1. In the J2EE Hierarchy view, expand the **EJB Modules** folder and select the newly imported EJB JAR.
2. From the EJB JAR’s pop-up menu, click **Generate > Deploy and RMIC Code**. Select to generate code for all your beans.
3. From the EJB JAR’s pop-up menu, click **Open With > Deployment Descriptor Editor**.
4. In the EJB deployment descriptor, click the **Overview** tab.
5. Scroll down to the WebSphere Binding section, and type the WebSphere binding JNDI name and the DataSource binding JNDI name. Record this data source name as you will use it later when you configure your data source in the test server instance.
6. Save your changes and close the EJB deployment descriptor.

7.4.4 Creating a server configuration and instance

1. Switch to the Server perspective.
2. Click **File > New > Server Project**.
3. In the **Project name** field, type **ServerTest**. Click **Finish**.
4. In the Navigator view, select **ServerTest**. From its pop-up menu, click **New > Server and Server Configuration**.
5. Type the name of the server, for example, **MyServer**, and select **ServerTest** from the Folder list.
6. In the **Server instance type** field, expand **WebSphere Version 4.0** and select **Test Environment**. In the **Template** field, click **None**.
7. Click **Next**. Specify a server port number of 8080.
8. Click **Finish**.

A new server configuration and instance will be created.

7.4.5 Adding the JDBC data source to the WebSphere 4.0 server configuration

1. In the Server Configuration view, expand the server item and select your server.
2. Right-click it, and from your pop-up menu, click **Open**.
3. Click the **Data source** tab.
4. In the JDBC Driver List, select the appropriate database driver (for example, **Db2JdbcDriver** if you are using **DB2^(R)**) and click **Edit**.
5. Verify that the **Class path** field contains the correct path to **db2java.zip**. Click **OK**.
6. If you are not using **DB2**, add a new driver - ensure that you use the J2EE JDBC driver that the Database vendor provides.
7. Select your JDBC driver, and click the **Add** button that is to the right of the **Data source** field.

8. In the **Add a data source** dialog, type a data source name. In the **JNDI name** field, type the same name you used in step 5 of "Generating deploy code and data source binding information" section. Enter the name of your database and click **OK**.
9. Save your changes.

7.4.6 Testing the enterprise beans with the EJB test client

1. In the Server Configuration view, expand the server item and select your server. Right-click it, and click **Add Project**. Select the .EAR project that contains your EJB module.
2. In the Servers view, select your server instance. Right-click it and, from its pop-up menu, click **Start**.
3. The project will be published and the server will start. After a few moments, click the **Console** tab to view the console window. You should see a message saying the Default Server is open for e-business. Scroll through the console to verify the EJB JAR has been started.
4. In J2EE Hierarchy view, expand the **EJB Modules** folder, select the enterprise bean that you want to test, right-click it, and click **Run on server**.

7.5 Known problems and workarounds

- When migrating the method finder helpers, the finder helper interfaces will disappear, as they have moved into an XML description file. This will generate a problem since your finder helper object usually implements this interface; the implementation must be removed.
- If you use inheritance in your enterprise beans, and you have built your own custom finders, they may break, since the mapping of fields in the generated code is different in WebSphere Studio Application Developer. To fix this, go to the EJSCMPxxxxxx generated class, find the select statement generated for the findbyPrimaryKey, and use it as a template.
- If you set up the Preferences page (in WebSphere Studio Application Developer) to stop an automatic build from being done every time you save changes, you must perform the following steps to generate the EJB deployed code and RMIC stubs:
 1. Select your EJB project, right-click and select **Generate > Deploy and RMIC code**.
 2. After you get an error (because the RMIC compiler can only see compiled code and since the generated Java classes were not compiled yet it will give an error), switch to Java perspective and build the project.
 3. Repeat step 1. You should not receive any errors this time.
 4. Repeat step 2, as you want to compile your newly generated stubs so you do not receive run-time errors when you deploy your enterprise beans.

7.6 Locating EJB information

The following table contains a list of EJB items and where to find your enterprise beans after you have migrated them to WebSphere Studio Application Developer:

Table 1. EJB items and their location table

Item	Location
Enterprise beans (fields, classes)	Expand the ejbModule folder in the J2EE Hierarchy view and select the EJB Jar you want to work with and double-click on it.
Enterprise beans (finder classes, access beans, generated classes).	Expand the ejbModule folder in the Navigator view and go to the com subdirectory (or your package structure). They are located in this subdirectory.
Association information	Expand the ejbModule folder in the J2EE Hierarchy view and select the EJB Jar you want to work with and double-click on it. In the deployment descriptor, click the Overview tab.
Finder helper description	Expand the ejbModule folder in the J2EE Hierarchy view and select the EJB Jar you want to work with and double-click on it. In the deployment descriptor, click the Beans tab.
Mapping /schema description	Expand the ejbModule folder in the J2EE Hierarchy view and select the EJB Jar you want to work with and double-click on it. From its pop-up menu, click Generate > EJB to RDB mapping .
Transaction demarcation	Expand the ejbModule folder in the J2EE Hierarchy view and select the EJB Jar you want to work with and double-click on it. In the editor, click the Beans tab.
Isolation Levels or find for update or read me methods marking	Expand the ejbModule folder in the J2EE Hierarchy view and select the EJB Jar you want to work with and double-click on it. In the deployment descriptor, click the Access tab.
EJB environment variables	Expand the ejbModule folder in the J2EE Hierarchy view and select the EJB Jar you want to work with and double-click on it. From its pop-up menu, click Open With > EJB Editor . In the editor, click the Beans tab.

7.7 Migrating EJB access beans

When enterprise beans are exported from VisualAge for Java, Enterprise Edition, Version 4.0 using the Export Tool for Enterprise Java Beans 1.1, the metadata for any associated Java bean wrapper and copy-helper access beans will also be exported. Rowset access beans are not supported by WebSphere Studio Application Developer, so the metadata for these access beans will not be exported.

Note: Migrated access beans with methods that have arrays as arguments or return types may cause problems for the access bean tools in WebSphere Studio Application Developer.

7.8 Migrating custom finder helpers

When you export enterprise beans from VisualAge for Java, Enterprise Edition, Version 4.0 using the Export Tool for Enterprise JavaBeans 1.1, or when 1.0 JAR files are deployed with the Deployment Tool for Enterprise JavaBeans (that is, the EJB Deploy Tool), the finder helper interfaces are migrated to the extension document. If the JAR file is an EJB JAR file, the metadata is also migrated to the extension document from the finder helper interfaces. However, migration of the finder helper interfaces to the extension document only occurs if the finder descriptors in the JAR file are not found in the extension document. If the enterprise beans are exported using the Export Tool for Enterprise Java Beans 1.1, the redundant classes are filtered from the exported JAR. If the enterprise beans

are not exported using the Export Tool for Enterprise Java Beans 1.1 and are imported along with the redundant classes, the classes are simply ignored.

Chapter 8. Migrating from EJB 1.0 to EJB 1.1 or to EJB 2.0

This chapter provides instructions on how to migrate your enterprise beans from EJB 1.0 to EJB 1.1 or to EJB 2.0. The main steps are:

- Migrate code from EJB 1.0 to 1.1
- Convert projects from EJB 1.x to EJB 2.0
- Migrate code from EJB 1.x to EJB 2.0
- Migrate a JMS listener application to use message-driven beans

8.1 Migrating code from EJB 1.0 to EJB 1.1

If you are migrating EJB 1.0 code to WebSphere Studio Application Developer, there are some EJB 1.1 specification changes that may affect your code. You may see some of the following validation errors or warnings due to EJB 1.1 changes:

- An enterprise bean cannot use the `javax.jts.UserTransaction` interface (use the new `javax.transaction.UserTransaction` interface instead)
- An entity bean cannot use the `UserTransaction` interface (this is not allowed in 1.1)
- An entity bean must define `FinderException` in their throws clause of finder methods
- An enterprise bean cannot throw `java.rmi.RemoteException` (it can throw `javax.ejb.EJBException` instead) but `RemoteException` must still be defined in EJB home and remote interfaces (as required by RMI).
- You cannot use the `finalize` method
- `java.security.Identity` is deprecated
- Enterprise bean methods `getCallerIdentity()` and `isCallerInRole(Identity)` are deprecated (use `getCallerPrincipal()` and `isCallerInRole(String roleName)` instead)
- `getEnvironment` is deprecated. Open `ejb-jar.xml` with EJB deployment descriptor, **Beans** tab, and view environment variables (such as `TARGET_HOME_NAME`)

```
String homeName=aLink.getEntityContext().getEnvironment().getProperty("TARGET_HOME_NAME");
if (homeName == null) homeName = "TARGET_HOME_NAME";
```

becomes:

```
Context env = (Context)(new InitialContext()).lookup("java:comp/env");
String homeName = (String)env.lookup("ejb10-properties/TARGET_HOME_NAME");
```

- CMP enterprise beans `ejbCreate(...)` methods should return the bean's primary key class (instead of void).
- New `HomeHandle` interface, and `EJBHome` interface requires new `getHomeHandle()` method:

```
public javax.ejb.HomeHandle getHomeHandle() { return null; }
```

You should consider enhancements to match EJB 1.1 changes such as:

- Entity bean primary keys can now be `java.lang.String` objects
- Entity bean finder methods should define `FinderException` in their throws clause
- Entity bean finder methods return `java.util.Collection`
- Check the format of your JNDI names (and local namespaces are now supported)

- There are now security roles, and isolation levels are now defined at the method level (were defined at EJB level for EJB 1.0)

You should review the EJB 1.1 changes in general, to see what other application changes are appropriate. For detailed information, refer to the following publications and Web sites:

- Enterprise Java Beans Specification, Version 1.1 - Section 1.3, Application compatibility (page 16). You can access this publication at: java.sun.com/products/ejb/docs.html
- The IBM WebSphere InfoCenter which is at the following URL: www.ibm.com/software/webservers/appserv/doc/v40/aes/infocenter/index.html. Refer to the following sections:
 - Transitioning to Version 4.0 (Click **Application Server AE > Migration > Transitioning to Version 4.0** and read the Role-based security section.)
 - Migrating to supported EJB specifications (Click **Application Server AE > Migration > 3.3 Migrating APIs and specifications > 3.3.1 Migrating to supported EJB specifications.**)
 - Migrating from Version 3.x (Click **Application Server AE > Migration > 3.2 Migrating from previous products versions > 3.2.2 Migrating from Version 3.x.**)
- *WebSphere Version 4 Application Development Handbook* - Chapter 11, Migrating 1.0 EJBs to 1.1 EJBs (page 267). You can access this publication at: www.redbooks.ibm.com/pubs/pdfs/redbooks/sg246134.pdf (Click **Chapter 11 > Developing Web Applications with VisualAge for Java > Developing EJBs in VisualAge for Java** and read the section **Migrating 1.0 EJBs to 1.1 EJBs**).
- *Programming J2EE APIs with WebSphere Advanced* - Chapter 5, EJB 1.1 code changes (page 113). You can access this publication at: www.redbooks.ibm.com/pubs/pdfs/redbooks/sg246124.pdf (Click **Chapter 5 > 5.8 EJB-JAR** and read the section **EJB 1.1 code changes**)
- *EJB Development with VisualAge for Java for WebSphere Application Server* - Appendix B. You can access this publication at: www.redbooks.ibm.com/pubs/pdfs/redbooks/sg246144.pdf

8.2 Converting projects from EJB 1.x to EJB 2.0

You can migrate your existing 1.1 project to a 2.0 project, or you can keep both by creating a new 2.0 project and then import the existing 1.1 project JAR file into it.

- In the J2EE Hierarchy view, right-click on the 1.1 project and then select **Migrate > J2EE Migration Wizard**. For more information about the different wizard options, press F1 while in the J2EE Migration wizard or read the online help section title "Migrating application modules from J2EE 1.2 to J2EE 1.3".
- Or, export the existing 1.1 project as an EJB JAR, create a new 2.0 project, and then **File > Import > EJB JAR**.

Although the project is an EJB 2.0 project, the existing (or imported) EJB 1.1 Container Managed Persistence (CMP) beans remain EJB 1.1 beans. That is, the CMP beans are not converted to EJB 2.0.

The J2EE Migration wizard migrates the Enterprise Beans within an EJB 2.0 project from 1.x into 2.0. (If you choose to migrate your 1.1 CMP beans to 2.0, all beans in the 2.0 project must be migrated. However, you can selectively choose to add local client views to these migrated 2.0 beans.)

- It will maintain the existing EJB 1.1 inheritance in the EJB 2.0 project.

- It will migrate EJB 1.1 (proprietary) relationships into EJB 2.0 (standard) relationships, plus other benefits.

Note: If you have any mapped associations, EJB 2.0 associations will be created for the associations themselves, but the role maps for that association will become invalid. If you run validation, you will see an error occurs. Make sure you simply open the Mapping Editor first, save the map, and the role maps will be removed. You may then run validation again and re-map the roles.

8.3 Migrating code from EJB 1.x to EJB 2.0

Note: EJB 2.0 beans are only supported in an EJB 2.0 project (although a 2.0 project also supports 1.1 beans).

1. For any CMP 1.x bean, replace each CMP field with abstract `getXXX` and `setXXX` methods. (Then the bean class needs to be abstract.)
2. For any CMP, create an abstract `getXXX` and `setXXX` method for the primary key.
3. For any CMP 1.x finder, create an EJBQL (EJB Query Language) for each finder.

Note: EJB Query Language has the following limitations in WebSphere Studio Application Developer Version 5:

- EJB Query Language queries involving EJBs with keys made up of relationships to other EJBs will appear as invalid and cause errors at deployment time.
 - The IBM EJB Query Language support extends the EJB 2.0 spec in various ways, including relaxing some restrictions, adding support for more DB2 functions, and so on. If portability across various vendor databases or EJB deployment tools is a concern, then care should be taken to write all EJB Query Language queries strictly according to instructions described in Chapter 11 of the EJB 2.0 specification.
4. For any CMP 1.x finder, return `java.util.Collection` instead of `java.util.Enumeration`.
 5. For any CMP 1.x bean, change all occurrences of `this.field = value` to `setField(value)` in `ejbCreate()` and elsewhere throughout the code.
 6. Update your exception handling (rollback behavior) for non-application exceptions:
 - Throw `javax.ejb.EJBException` instead of `java.rmi.RemoteException` to report non-application exceptions.
 - In EJB 2.0 and 1.1, all non-application exceptions thrown by the instance result in the rollback of the transaction in which the instance executed, and in discarding the instance.
 - In EJB 1.0, the container would not rollback a transaction and discard the instance if the instance threw the `java.rmi.RemoteException`.
 7. Update your Exception handling (rollback behavior) for application exceptions:
 - In EJB 2.0 and 1.1, an application exception does not cause the container to automatically rollback a transaction.
 - In EJB 1.1, the container performs the rollback only if the instance have invoked the `setRollbackOnly()` method on its `EJBContext` object.
 - In EJB 1.0, the container was required to rollback a transaction when an application exception was passed through a transaction boundary started by the container.

8. Update any CMP setting of application specific default values to be inside `ejbCreate` (not using global variables, since EJB 1.1 containers set all fields to generic default values before calling `ejbCreate` which will overwrite any previous application specific defaults). This approach will also work for EJB 1.0 CMPs.

To access the J2EE Migration wizard, right-click on the EJB project and then select **Migrate > J2EE Migration Wizard**. For more information about the different wizard options, press F1 while in the J2EE Migration wizard.

8.4 Migrating a JMS listener application to use message-driven beans

In WebSphere Application Server Version 4.0, IBM had a proprietary implementation of Message Beans with the JMS Listener similar to the Message Driven Beans in EJB 2.0 specification. Customers using Message Beans with the JMS Listener in WebSphere Application Server Version 4.0 can migrate to the EJB 2.0 Message Driven Beans using the **mb2mdb.bat** program available in WebSphere Application Server Version 5 run-time environment that comes with WebSphere Studio Application Developer. This program is located in the `...\WS_install\dir\runtimes\base_v5\bin` folder (where *WS_install\dir* is the directory where WebSphere Studio Application Developer is installed).

The **mb2mdb** command line utility takes as its input either a deployed MessageBean.jar module or a deployed Enterprise Application (.ear file) that contains a message bean, along with the JMS listener configuration XML file that defines the WebSphere Application Server Version 4.0 message beans. The result is a new .jar or .ear module that can then be deployed directly into a WebSphere Application Server Version 5 application server.

To display the usage help for the migration utility, typing the following command on the command line: `mb2mdb`

To migrate a WebSphere Application Server Version 4.0 enterprise application that uses message beans to use EJB 2.0 message-driven beans, type the **mb2mdb** command with the following syntax on the command line:

```
mb2mdb inputMB.jar-ear jmsListenerConfig.xml workingDirectory outputMDB.jar-ear options
```

where:

- *inputMB.jar-ear* is the name of the deployed WebSphere Application Server Version 4.0 .jar or .ear file containing a stateless session message bean.
- *jmsListenerConfig.xml* is the name of the XML configuration file used to configure the WebSphere Application Server Version 4.0 JMS listeners.
- *workingDirectory* is the name of a new or existing directory that is used to generate the new message-driven bean and package the outputMDB.jar or outputMDB.ear file. **Note:** By default, the tool clears the working directory after it has completed. If you want to preserve the contents of the working directory, you must specify the **-keep** option.
- *outputMDB.jar-ear* is the name of the output .jar or .ear file for the migrated message-driven bean application.
- *options* is a set of optional parameters that you can use to control the `mb2mdb` utility. The following table lists the valid options.

Table 2. List of valid options.

Options	Description
-keep	Prevents the tool from clearing out the working directory after completion.
-verbose	Causes the tool to display informational messages as to the progress of the migration and its parameters.
-map listenerHome=bindingHome	<p>Provides a mechanism to map between the JNDIHomeName specified for a listener in the JMS listener configuration XML file and the default binding home name specified in the inputMB.jar-ear file.</p> <p>If the jmsListenerConfig.xml file contains a deployed EJB home JNDI name that is different to the default binding within the inputMB.jar-ear, use this option to map between the two names.</p> <p>This enables you to install the output .jar or .ear file for the message-driven bean into an application server and bind the bean with a different JNDIHomeName than is specified in the bean's bindings.xml.</p>

The result of this task is a new .jar or .ear file for a message-driven bean that can then be deployed directly into a WebSphere Application Server Version 5.0 application server.

To successfully install the .jar or .ear file:

1. Bind the message-driven bean against a listener port defined to the message listener service of the application server.
2. Use the WebSphere Application Server administrative console to define the listener port, which defines the JMS connection factory and destination that a message-driven bean bound to it listens on.

Chapter 9. Migrating from VisualAge for Java Visual Composition Editor to Visual Editor for Java

This chapter provides instructions on how to migrate applications created in the Visual Composition Editor feature of VisualAge for Java to the Visual Editor for Java within WebSphere Studio Application Developer:

- Saving enhanced design-time metadata from VisualAge for Java
- Completing the migration (importing into WebSphere Studio)

9.1 Saving enhanced design-time metadata from VisualAge for Java

This step is optional but is highly recommended (especially if your application has any connections) for the following reasons.

- VisualAge for Java did not save information about the placement of top-level beans (that is, beans that are not contained inside other beans) on the free-form surface. By contrast, WebSphere Studio always saves this information as a comment on the line of code that declares the bean and restores the bean to that position on the canvas every time you open the visual editor. You have the option to save this design-time information in VisualAge for Java before migration. If you do not save it, when you first open your .java files in the new Visual Editor for Java the editor calculates a default position for top-level beans (also known as free-form parts), which you can easily change by means of a drag and drop operation.
- Although the new Visual Editor for Java in WebSphere Studio Application Developer does not support connection design, it is possible that equivalent function might be added in the future. (**Note:** This is not a product commitment; it is just preparation for a future possibility.)

To save the enhanced metadata information prior to migration:

1. Go to the VisualAge for Java Developer Domain [www7.software.ibm.com/vad.nsf/data/document4293] and download the "IBM VCE Code Generation and Export Utility".
2. Following the tool readme, add the tool into VisualAge for Java and then stop and restart VisualAge for Java.
3. Version the current application code into the VisualAge for Java repository (so you can return to this version in case of any ongoing VisualAge for Java development).
4. For each of your graphical applications within VisualAge for Java, select one or more of the graphical programs (typically XxxxxView), right-click, and then do the following:
 - a. Click **VCE Code Generation/Export**, and leave selected the **Export to a directory after code regeneration** option.
 - b. Click **Finish**.
 - c. Leave **Directory** as the selected export destination, click **Next**.
 - d. Select your target directory, clearing the **.class** option and selecting the **.java** option (since you want the source code), and then click **Finish**.
 - e. Optionally, reload your VisualAge for Java code with its previous version (right-click and then select **Replace with > Previous edition**).

9.2 Completing the migration (importing into WebSphere Studio)

Your classes are now ready to be imported into WebSphere Studio. Refer to the earlier chapter “Chapter 6. Migrating from VisualAge for Java to WebSphere Studio Application Developer” on page 21. Once your previous Visual Composition Editor source programs have been imported into WebSphere Studio Application Developer, you can maintain them in the Visual Editor for Java.

Chapter 10. Converting from VisualAge for Java Persistence Builder to EJB 2.0

This chapter provides an overview of how to convert your VisualAge for Java Persistence Builder applications to EJB 2.0 on WebSphere Studio Application Developer Version 5.0.

- Migrate and export the Persistence Builder Model as an EJB 1.1 JAR
- Import EJB 1.1 JAR into EJB 1.1 project, set Access and Isolation values
- (Optional) Convert client application and get server side running, all at EJB 1.1 level
- Convert to EJB 2.0 project
- Convert EJB 1.1 code to EJB 2.0 and set Application Profile Declarations
- (If not already done) Convert Persistence Builder client application into EJB client application

10.1 Migrate and export the Persistence Builder Model as an EJB 1.1 JAR

1. Use VisualAge for Java 4.0 tool to migrate Persistence Builder Model into an EJB 1.0 Group (**OnlineHelp > Tasks > AddingPersistence > MigratingPersistenceBuilderModel to EJB**)
2. Use VisualAge for Java EJB 1.1 Export Wizard to export an EJB 1.1 JAR

10.2 Import EJB 1.1 JAR into EJB 1.1 project, set Access and Isolation values

1. Import EJB 1.1 JAR into WebSphere Studio Application Developer EJB 1.1 project.
Note: EJB 2.0 primarily provides server-side performance improvements. You should be able to get your Persistence Builder application converted and running using EJB 1.1 if you wish (and this might be a nice project stepping-stone before undertaking the EJB 2.0 conversion in order to achieve the performance enhancements). Alternately, you can convert your server-side code to EJB 2.0 first, and then get your client and overall application running.
2. Per Bean within your EJB 1.1 project, set your ReadOnly Access Intents (**Read**, or **Update**) and Isolation Levels (**Repeatable read**, or **Read committed**, or **Read uncommitted**, or **Serializable**) - these are not exported out of Visual Age for Java and must be manually set.
 - In the J2EE View under EJB Modules, select the EJB Module you are working with and double-click on it. In the EJB Deployment Descriptor on the **Access** tab, in the **Access Intent for Entities 1.x** section **Add**, or **Edit**, or **Remove** (as appropriate) in the **Isolation Level** section **Add**, or **Edit**, or **Remove** (as appropriate).

10.3 (Optional) Convert client application and get server side running, all at EJB 1.1 level

See details in “10.6 Convert Persistence Builder client application into EJB client application” on page 45.

10.4 Convert to EJB 2.0 project

Convert EJB 1.1 project into EJB 2.0 project by selecting your EJB project, then right-clicking **Migrate > J2EE Migration Wizard**.

The EAR project that contains the above EJB 1.1 module project also needs to be migrated to J2EE 1.3 specification level. Both the EJB project and EAR project can be migrated at once by selecting the EAR project in the J2EE Migration wizard.

For more information about the different wizard options, press F1 while in the J2EE Migration wizard or read the online help section titled “Migrating application module from J2EE 1.2 to J2EE 1.3”.

10.5 Convert EJB 1.1 code to EJB 2.0 and set application profile declarations

1. Use the J2EE (EJB Bean) Migration wizard to convert from 1.x CMPs into 1.1 CMPs into 2.0 CMPs
 - Refer to the earlier chapter “8.3 Migrating code from EJB 1.x to EJB 2.0” on page 37, including additional (manual) changes.
 - This will also migrate your **Access Intents** from **Entity 1.x** to **Entity 2.x** format
 - This will also migrate any 1.1 (proprietary) relationships into 2.0 (standard) relationships - using Local Interfaces.
 - This also migrates any relationship accessor access intents.
 - Client code (including existing session beans) will have to change to match the new method signatures.
2. Manually adjust your Access Intents
 - Entity 2.x Access Intents are more powerful (**Optimistic/Pessimistic Read/Update**, **Pessimistic Update Exclusive**, and so on).
 - If you an optimistic type (wsOptimisticUpdate or wsOptimisticRead), then you can also select the **Read Ahead Hint** check box. This adds the ability to Preload (ReadAhead) related enterprise beans across relationships in a single query.
 - Per Bean within your EJB 2.0 project, adjust your migrated Access Intents.
 - a. For each EJB project, right-click **Open with > EJB Deployment Descriptor Editor > Access** tab.
 - b. Then click **Access Intent for Entities 2.x > Add**, or **Edit**, or **Remove** (as appropriate).

10.6 Convert Persistence Builder client application into EJB client application

Important note: Persistence Builder provides the capability to create stand-alone applications (as well as distributed applications). There is no facility for stand-alone EJB applications, the distributed alternatives are:

- Install a WebSphere Application Server and run everything on one machine.
- Install the WebSphere Application Server "light server" to provide the client-side environment to access remote servers.
- Do not use EJBs at all (use JDBC, or the new DataBase Access Beans for RowSet JDBC access, or other approaches).

Additional note: Persistence Builder provided Visual Composition Editor (VCE) palette items for visually building client applications. There are no corresponding components within Application Developer. Alternatives are:

- Convert the client-side Java program by removing the use of the Persistence Builder visual components.
 - Convert the client-side Java program into a web browser-based program accessing server-side servlets.
1. Use Session Beans to abstract layer(s) of business logic above Entity Beans
 - **Important reference:** Refer to "Rules and Patterns for Session Facades" [www.ibm.com/websphere/developer/library/techarticles/0106_brown/sessionfacades.html].
 - If your Persistence Builder applications used the recommended Persistence Builder "Task Wrapper Pattern" [www7b.software.ibm.com/vadd-bin/httpd1?1/vadc/techarticle/0010_lanuti.pdf]. Then your application servlets and JSPs do not directly use the business objects nor associated transactions, instead all persistence and transaction exceptions are encapsulated and the data flowing into and out of the TaskWrapper is typically only serializable objects. Such a "Task Wrapper" application should convert fairly easily into EJBs plus Session Beans.
 - The Session Beans now control transaction demarcation. They use EJB Container Managed Persistence (CMPs) with container managed transactions ('TRANSACTION_REQUIRED').
 - No RollBacks are available to the servlet/client calling the Session Bean, so they must have recover logic.
 - Manually convert any Persistence Builder **ConflictResolution** code into **EJB TransactionRolledBack** exception handling.
 - Where Persistence Builder provided nested transactions, servlets/clients now would use multiple Session Beans with 'REQUIRES_NEW' and with recover/compensation logic in case of errors.
 - **Important Reference:** "WebSphere Application Server, Development Best Practices for Performance and Scalability" [www-4.ibm.com/software/webservers/appserv/ws_bestpractices.pdf].
 2. Manually convert persistence builder business logic from persistence builder model to use new Session Beans.
 - Persistence Builder clients used to use MyobjectImpl implementation classes. New (converted) EJB clients (or more typically their new application session beans) will now use the EJB MyObject classes (remote interface, and hence indirectly the MyobjectBean object).

- If the developer had any business methods in the previous implementation classes, then they need to be copied into the new EJB bean (and promoted to the remote interface), including any required exception handling.
3. Manually convert from Persistence Builder Lite Collections to EJB 2.0 Dependent Values.
- Alternately, single-value finders can return a single field - which may be suitable in some cases.
 - Alternately, a Session Bean can directly use JDBC - which may be suitable in some cases.

Chapter 11. Migrating from WebSphere Studio Application Developer Version 5 to Version 5.0.1

This chapter covers the following migration topics:

- Migrate Web projects Version 5 to Version 5.0.1
- Convert Web projects to Struts 1.1 Beta 3

11.1 Migrating Web projects Version 5 to Version 5.0.1

In Version 5.0 (General Availability), the folder names are **Java Source** and **Web Content**. In Version 5.0.1, the default folder names for new Web projects are configurable through a preference page (**Window > Web Tools > New Project**). The default default names are now **JavaSource** and **WebContent**. These default names will be used for new Web projects only. Web projects created in Version 4.0.x and Version 5.0 will continue to function using the old names. The same is true for Static Web projects.

If you opt to change the source folders names for 4.0.x and 5.0 projects in Version 5.0.1, use the **Rename** pop-up menu action in the Navigator view. The **Rename** pop-up menu action renames the folder names and fixes the Java build path for the 4.0.x and 5.0 Web projects.

For the **JavaSource** folder, the **Rename** pop-up menu action works in J2EE Navigator View and Package Explorer view. For the **WebContent** folder, the **Rename** pop-up menu action works in Resource Navigator view, J2EE Navigator View and Package Explorer view.

If a Version 4 Web project is saved into an SCM repository and then loaded into Version 5.0.1, it will retain the old structure with the **source** and **webApplication** folders. Either structure will build correctly in Version 5.0.1.

Note: If the users opt to rename **Java Source** and **Web Content** folder names in Version 5.0.1, then they have to manually update any automated build scripts they have to change them to use the new folder names.

11.2 Converting Web projects to Struts 1.1 Beta 3

In WebSphere Studio Application Developer Version 5.0.1, the Struts tools run-time has stepped up from Version 1.1 Beta 2 to Version 1.1 Beta 3. In WebSphere Studio Application Developer Version 5.0 (General Availability) when you create a Web project you have the option to add Struts support to your project. You can choose either Struts 1.0.2 or Struts 1.1 Beta 2. In WebSphere Studio Application Developer Version 5.0.1, the latter choice is replaced by Struts 1.1 Beta 3. If you created Struts 1.1 Beta 2 Web projects in WebSphere Studio Application Developer Version 5.0, you might wish to convert it to Struts 1.1 Beta 3 but this is not required as Struts 1.1 Beta 2 is still supported in WebSphere Studio Application Developer Version 5.0.1. If you do have Struts 1.1 Beta 2 Web projects that you wish to convert to Struts 1.1 Beta 3, you will need to do the following:

1. Load your Struts 1.1 Beta 2 projects into a WebSphere Studio Application Developer Version 5.0.1 workspace.

2. Create a new Struts 1.1 Beta 3 Web project named **beta3**. This provides convenient access to the Struts 1.1 Beta 3 artifacts we will need while we are converting our real projects. You can delete this project when you are done.
3. For each Struts 1.1 Beta 2 project that you want to convert to Struts 1.1 Beta 3, do the following:
 - a. Delete the following .jar files from your project's Web Content/WEB-INF/lib directory: commons-*.jar and struts.jar.
 - b. Copy the following .jar files from beta3/WebContent/WEB-INF/lib directory to your project's Web Content/WEB-INF/lib directory: commons-*.jar and struts.jar.
 - c. Delete the following .tld files from your project's Web Content/WEB-INF directory: struts-*.tld.
 - d. Copy the following .tld files from beta3/WebContent/WEB-INF directory to your project's Web Content/WEB-INF directory: struts-*.tld.

Chapter 12. Build setup (library, JARs, dependent project JARs, Ant builds)

This chapter covers the following migration topics:

- Java library JARs, and third-party external JARs
- Optimizing multi-project builds using dependent Project JARs
- Automated production builds using Ant

12.1 Java library JARs and third-party external JARs

For detailed explanations of what is involved, see the article on J2EE Class Loading (www.ibm.com/websphere/developer/library/techarticles/0112_deboer/deboer.html) (J2EE modules and class paths) and on developing J2EE utility JARs (www.ibm.com/websphere/developer/library/techarticles/0112_deboer/deboer2.html) (Java JARs in J2EE modules). These provide excellent technical background and advice.

12.1.1 Recommended way to use a third-party JAR within a Web project

The recommended way to use a third-party JAR file within a Web project is to import it (keeping it as a JAR file) into the library folder of your Web project. This is the only J2EE-defined, portable way to use a JAR file, and will ensure that you do not have to make any changes when deploying to another server.

To use an external JAR file in a single Web project, follow the steps below. If you need to use the JAR file in an EJB project or in multiple Web projects, follow the steps in the “12.1.2 Recommended way to use a third-party JAR for use with multiple EJB or Web projects” instead.

1. Select **File > Import > File System**. Click **Next**. You must select **File system**, not **Zip file**, in order to ensure that the JAR file is not expanded when it is imported.
2. Click **Browse** and locate the JAR file directory.
3. Import it into your *WebProject*/WebContent/WEB-INF/lib folder, where *WebProject* is the name of your Web project.
4. Click **Finish**. The JAR file will be automatically added to the Java build path, and no further changes are required at run time.

12.1.2 Recommended way to use a third-party JAR for use with multiple EJB or Web projects

The recommended way to use a third-party JAR file with two or more EJB or Web projects is to import it (keeping it as a JAR file) into your Enterprise Application (EAR) project. This is the only J2EE-defined, portable way to use a JAR file, and will ensure that you do not have to make any changes when deploying to another server.

To use an external JAR file with multiple EJB or Web projects, follow the steps below. If you only need to use the JAR file in a single Web project, follow the steps in the previous section.

1. Select **File > Import > File System**. Click **Next**. You must select **File system**, not **Zip file**, in order to ensure that the JAR file is not expanded when it is imported.
2. Click **Browse** and locate the JAR file directory.
3. Import the JAR file into the Enterprise Application project that contains the EJB or Web projects.
4. Click **Finish**. The JAR file will automatically be added to the Java build path, and no further changes are required at run time.
5. Follow the steps in the following next section to add the JAR to the module dependencies of the Web or EJB project.

12.1.3 References between Web projects and other EJB projects

If you have a Web or EJB project that depends on another EJB project, you must take the following steps to make sure that the project will be visible at run time.

1. Select the Web or EJB project, right-click it, and select **Properties > Java Build Path > Projects** tab.
2. Select the projects that the Web or EJB project requires at run time.
3. Click **Finish**.

This changes the Web or EJB project manifest file to contain an explicit reference to the required EJB project. It also modifies the Java build path to include the references to EJB and Web projects. Using this technique, the WebSphere Application Server does not need any specific setting for module visibility (deprecated).

If you have Web or EJB project references to JAR files, you must take the following steps to make sure that the JAR files are on the Java Build path of the project.

1. Select the Web or EJB project, right-click it, and select **Properties > Java JAR Dependencies**.
2. Select the JARs from the **Available Dependent Jars** list for the Web or EJB project required at run time.
3. Click **OK**.

12.1.4 Alternative way to use external JAR files (global build and server classpath)

You may also leave the JAR file outside of WebSphere Studio Application Developer and add it to both the Java build path and the server instance's class path. This is not recommended, because your application will not be easily portable. When you move to a different server, you will always have to update the server's class path. As well, you need to ensure that your class files do not conflict with other versions of similar class files already on the server classpath (and needed by the server or its other applications). If you do decide to take this approach, take the following steps:

1. Add the external JAR file to the Java build class path of the project that requires the JAR file.
 - a. Select the project, right-click it, and select **Properties** from its pop-up menu.
 - b. Click **Java Build Path**.
 - c. Click the **Libraries** tab.
 - d. Click **Add External JARs**. Select the JAR file, and click **Open**.

- e. Click **OK**.
- 2. Add the external JAR file to the server instance's class path
 - a. Open the Server Configuration view, and expand the **Server** folder.
 - b. Select the server instance that the project is deployed on. Right-click it and click **Open**.
 - c. Click the **Paths** tab.
 - d. Within ws.ext.dirs, click **Add External JARs**. Select the JAR file, and click **Open**. Note that ws.ext.dirs is used for application JAR files, and the CLASSPATH is used for server JAR files.
 - e. Close the server instance and save your changes.

12.2 Optimizing multi-project builds using Dependent Project JARs

WebSphere Studio Application Developer's powerful autobuild feature can slow down build performance during complex, multiproject builds. There are a number of ways to control these autobuilds (dependent files, active and inactive projects, and projects in source or JAR format) but these options can get quite complicated. There is an article that explains the options and how to optimize your build performance. See the WebSphere Developer Domain article "Optimizing complex builds in WebSphere Studio Application Developer" (www.ibm.com/websphere/developer/library/techarticles/0204_searle/searle.html).

12.3 Automated production builds using Ant

You can use Ant with WebSphere Studio Application Developer to automate your production builds. There is a multi-part article that explains the following things:

- What is Ant
- How to run Ant both inside and outside WebSphere Studio Application Developer
- How to use Ant for production builds of J2EE elements (EJBs, WARs, EARs, and so on)
- How to extend Ant with new WebSphere Studio Application Developer build tasks

See the WebSphere Developer Domain article "Using Ant with WebSphere Studio Application Developer" (www.ibm.com/websphere/developer/library/techarticles/0203_searle/searle1.html).

Chapter 13. Migration examples

This chapter contains migration examples to help you learn more about migrating from VisualAge for Java and WebSphere Studio "Classic" to WebSphere Studio Application Developer.

- VisualAge for Java JSP/servlet sample (LeapYear)
- VisualAge for Java EJB, VCE, and database samples (HelloWorld session bean and Increment enterprise bean)
- WebSphere Studio "Classic" Web application sample (YourCo)
- Migration from EJB 1.x to EJB 2.0 (HelloWorld session bean and Increment enterprise bean)

13.1 Example: VisualAge for Java JSP/servlet sample (LeapYear)

Description

This is the FindTheLeapYears sample provided with VisualAge for Java Version 4.0. Information about it can be found in the VisualAge for Java online help (**Samples > JSP/Servlet Development Environment**).

Migration overview

You will follow the steps below to migrate the sample from VisualAge for Java to WebSphere Studio Application Developer. These steps are discussed in more detail below:

1. "13.1.1 Exporting your files from VisualAge for Java".
2. "13.1.2 Creating a new WebSphere Studio Application Developer Web project" on page 54.
3. "13.1.3 Importing the Java and project resource files into the WebSphere Studio Application Developer project" on page 54.
4. "13.1.4 Defining any servlets and make any restructured application changes" on page 55.
5. "13.1.5 Creating a WebSphere Studio Application Developer server project" on page 55.
6. "13.1.6 Testing the migrated LeapYear application" on page 56.

13.1.1 Exporting your files from VisualAge for Java

1. Open VisualAge for Java.
2. Select the **IBM JSP Examples** project.
3. Right-click the project and select **Export**. Select the **Directory** radio button and click **Next**.
4. Type the name of the directory you want to export the files to.
5. Clear the **.class** check box. You do not need to export these files as you will rebuild the project in WebSphere Studio Application Developer and re-create these files.
6. Select the **.java** check box and click **Details**. Select only the LeapYear files and click **OK**.
7. Select the **resource** check box and click **Details**.



8. Select **LeapYearInput.html** and **LeapYearResults.jsp**, which are located in the following directory: **IBM WebSphere Test Environment\hosts\default_host\default_app\web\JSP\sample3**.
9. Click **OK**.
10. Clear the **Create Manifest file** check box (you do not need to create a manifest file).
11. Click **Finish**.
12. Close VisualAge for Java.

13.1.2 Creating a new WebSphere Studio Application Developer Web project

1. Start WebSphere Studio Application Developer.
2. Create a new Web project (**File > New > Project > Web > WebProject**) called LeapYear.
3. Ensure that **J2EE Web project** is selected, and then click **Next**.
4. Select **New**.
5. Change the **Enterprise Application project name** to LeapYearEAR and select **J2EE level 1.2**. You could put the Web project into any existing Enterprise Application (EAR) project, but for this example, you will put it in LeapYearEAR.
6. Leave LeapYear in the **Context root** field.
7. Click **Finish**.

13.1.3 Importing the Java and project resource files into the WebSphere Studio Application Developer project

Import the Java source files into the LeapYear project source directory by following these steps:

1. In the Web perspective, expand LeapYear and select the **JavaSource** directory.
2. Click **File > Import > File system** and click **Next**. Browse to the directory you exported your files to and click **OK**.
3. You only want to import the Java source files into the **JavaSource** directory, so, in the Import dialog, expand your export directory and select only the **com** subdirectory (it contains the three Java source files).
4.  Click **Finish**. This creates LeapYear\JavaSource\com\ibm\ivj\wte\samples\leapyear\LeapYearXXXX.java files. Java classes are automatically compiled into LeapYear\WebContent\WEB-INF\classes.
5.  Click **Finish**. This creates LeapYear/JavaSource/com/ibm/ivj/wte/samples/leapyear/LeapYearXXXX.java files. Java classes are automatically compiled into LeapYear/WebContent/WEB-INF/classes.

Import the resource files into the LeapYear project under the **WebContent** directory by following these steps:



1. In the current Web perspective, expand the LeapYear project and select the **WebContent** directory.
2. Select **File > Import > File system** and click **Next**. Browse to the directory you exported your files to, expand your export directory to the **sample3** subdirectory, then click **OK**.

3. You only want to import the resource files into the **WebContent** directory, so, in the Import dialog, select the **sample3** subdirectory, which contains the .jsp and .html files.
4. Click **Finish**. The files are imported into the **WebContent** directory.

13.1.4 Defining any servlets and make any restructured application changes

1. You now need to create a servlet. Select the LeapYear project and expand it (**Leap Year > WebContent > WEB-INF**) to the web.xml file. Open the web.xml file.
2. Click the **Servlets** tab at the bottom of the page.
3. Click **Add**.
4. Ensure that the **Servlet** radio button is selected.
5. Select the class **LeapYear**, then click **OK**.
6. Select **URL Mapping > Add**, then type LeapYear.
7. Save the changes (**File > Save web.xml**) and close the web.xml file.

You now need to make any application changes due to the slightly changed source/application structure:

1. There will be two errors listed in the Tasks view. One is in LeapYearInput.html and one is in LeapYearResults.jsp.
2. Open the LeapYearResults.jsp file. Replace /JSP/index.html with LeapYearInput.html.
3. Open the LeapYearInput.html file. Replace /servlet/com.ibm.ivj.wte.samples.leapyear.LeapYear with LeapYear.
4. Save your changes and close the LeapYearResults.jsp and LeapYearInput.html files.
5. To avoid a run-time error, open the LeapYear.java file, which is located in the following subdirectory:
 -  `JavaSource\com\ibm\ivj\wte\samples\leapyear`
 -  `JavaSource/com/ibm/ivj/wte/samples/leapyear`
6. Go to line 118 and change `getRequestDispatcher` from `"/JSP/Sample3/LeapYearResults.jsp"` to `"LeapYearResults.jsp"`
7. Save your changes and close LeapYear.java.

At this point the sample has been migrated into WebSphere Studio Application Developer. All that remains is to create a WebSphere Studio Application Developer server project and test the sample in the WebSphere test environment.

13.1.5 Creating a WebSphere Studio Application Developer server project

1. Click **File > New > Project > Server > Server Project**. Click **Next**. In the **Project name** field, type newServer and click **Finish**. You will automatically be switched to the Server perspective.
2. Right-click **newServer** and click **New > Server and Server Configuration**.
3. In the **Server name** field, type WSTestEnv. In the **Server instance type** field, select **WebSphere V4.0 Test Environment**. Click **Finish**.

Now, you need to specify your EAR project to the server configuration:

1. In the Server Configuration view, expand the server items and click **WSTestEnv**.
2. Right-click it and click **Add > LeapYearEAR**.

13.1.6 Testing the migrated LeapYear application

1. Select the LeapYearInput.html file.
2. Right-click the HTML file, and from its pop-up menu, click **Run on Server**.
3. Wait while the server starts. Watch the **Console** page (click the **Console** tab in the Servers view) until the message "Server Default Server open for e-business" appears.
4. When a browser opens, type 2001 in the **Start Year** field, then click **Submit**.
5. The Console view shows the message LeapYear: init. Wait until you see the list of leap years, then select **WSTestEnv** in the Servers view. Right-click it and click **Stop**.

13.2 Example: Enterprise beans, VCE, and database samples (HelloWorld session bean and Increment enterprise bean)

Description

You will work with two EJB Development samples, Hello World and Increment. To access these samples open the online help, and select **Samples > EJB Development**.

You must work with VisualAge for Java Version 4.0 for this example.

You do not have to migrate these samples together. If you are not currently working with DB2, you may not wish to migrate Increment. In this case, you can ignore any steps that only apply to the Increment sample.

Before you begin

- Be sure the applications run in VisualAge for Java (that is, the DB2 setup for the Increment sample is complete)
- Be sure to stop the VisualAge for Java WebSphere test environment EJB server and Persistent Name Server (so they will not conflict with WebSphere Studio Application Developer).

Migration steps

You will follow the steps below to migrate the samples from VisualAge for Java to WebSphere Studio Application Developer. These steps are discussed in more detail below:

1. "13.2.1 Exporting the client Java source from VisualAge for Java" on page 57.
2. "13.2.2 Exporting the EJB group from VisualAge for Java into an EJB 1.1 JAR" on page 57.
3. "13.2.3 Creating a new WebSphere Studio Application Developer EJB project" on page 58.
4. "13.2.4 Importing the EJB 1.1 JAR file into WebSphere Studio Application Developer EJB project" on page 58.
5. "13.2.5 Generating and deploying RMIC stub and tie code" on page 58.
6. "13.2.6 Specifying the data source binding information" on page 58.

7. "13.2.7 Creating a new WebSphere Studio Application Developer project" on page 58.
8. "13.2.8 Creating a WebSphere Studio Application Developer server project" on page 59.
9. "13.2.9 Specifying your data source, your EAR project, and then start the server" on page 60.
10. (Optional) "13.2.10 Starting DB2 and connecting to sampleDB" on page 60.
11. "13.2.11 Testing the migrated HelloWorld client" on page 60.
12. "13.2.12 Testing the migrated Increment client" on page 61.
13. "13.2.13 Testing the migrated Increment and HelloWorld EJBs using the EJB Test Client" on page 61.

13.2.1 Exporting the client Java source from VisualAge for Java

1. Start VisualAge for Java Version 4.0.
2. Select the **IBM EJB Samples** project
3. Right-click the project and click **Export**.
4. Select the **Directory** radio button and click **Next**.
5. Type the name of the directory you want to export your files to.
6. Clear the **.class** check box. (You will rebuild the .class files in WebSphere Studio Application Developer to ensure that you have successfully migrated the samples.)
7. Select the **.java** check box. Click **Details** and clear the **IBM EJB Samples** check box.
8. Select the **HelloClient** and **IncrementClient** check boxes and click **OK**.
9. Clear the **.resource** check box (unless you want the DB2 .clp setup scripts).
10. Click **Finish**.

13.2.2 Exporting the EJB group from VisualAge for Java into an EJB 1.1 JAR

1. Open map browser to load map for IBM EJB samples.
2. Click the **EJB** tab. In the Enterprise Beans pane, right-click **IBMEJBSamples** and click **Export > EJB 1.1 JAR**.
3. Select a directory to export your JAR file to.
4. In the **JAR file** field, type the name of the JAR file you want to create.
5. In the **Select a target database** list, select **DB2 for NT, V7.1**.
6. Select the **.class** check box. (You will rebuild the files in WebSphere Studio Application Developer as a verification of the migration, but this must be selected in order to proceed.)
7. Select the **.java** check box. (You need the four HelloWorld and the five Increment EJB files.)
8. Click **Finish**.
9. If you were running the samples, then stop the VisualAge for Java Persistent Name Server, EJB server, and WTE.
10. (Optional) Exit VisualAge for Java.

13.2.3 Creating a new WebSphere Studio Application Developer EJB project

Note: This section is optional. If you do not have an EJB and EAR project, they will automatically be created when you import the EJB 1.1 JAR file.

1. Start WebSphere Studio Application Developer.
2. Create a new EJB project (**File > New > Project > EJB > EJB Project > Create 1.1 > Next**).
3. In the **Project name** field, type EJBSamples.
4. Select **EAR new**.
5. In the **EAR project name** field, type EJBSamplesEAR.
6. Click **Finish**. You could put this EJB project into any existing Enterprise Application (EAR) project you currently have, but for this example, create a new EAR project called EJBSamplesEAR. The new projects are created and appear in the J2EE perspective.

13.2.4 Importing the EJB 1.1 JAR file into WebSphere Studio Application Developer EJB project

1. Click **File > Import > EJB JAR file**. Click **Next**.
2. In the **EJB JAR file** field, browse for the location of the EJB 1.1 JAR you created.
3. In the **EJB Project** field, select existing EJB project **EJBSamples**. This selects the corresponding EAR project **EJBSamplesEAR**.
4. Click **Finish**. You will receive four or five EJB 1.1 warnings. You can ignore them.

13.2.5 Generating and deploying RMIC stub and tie code

1. In the J2EE perspective, click on the **J2EE Hierarchy** tab, expand the EJB Modules folder, and select **IBMEJBSamples**.
2. Right-click it and click **Generate > Deploy and RMIC Code**.
3. Select the **Hello World** and **Increment beans**. Click **Finish**.





13.2.6 Specifying the data source binding information

1. In the J2EE Perspective, expand EJB Modules, and select **IBMEJBSamples**.
2. Right-click the module and select **Open With > Deployment Descriptor Editor**.
3. In the EJB deployment descriptor, click the **Overview** tab.
4. Scroll down to the **WebSphere bindings** section, and type jdbc/sampledb in the **DataSource JNDI name** field.
5. If necessary, type a default user ID and password.
6. Close the EJB deployment descriptor (you will be prompted to save your changes).


13.2.7 Creating a new WebSphere Studio Application Developer project


1. Create a new Java project (**File > New > Project > Java > Java Project**).
2. In the **Project name** field, type EJBClients. Click **Next**.
3. Click the **Projects** tab. Select the **EJBSamples** check box.
4. Click **Finish**.

You are now going to import the Java code into a Java project:

1. In the Java perspective, open the Navigator view (**Window > Show view > Navigator**).
2. In the Navigator view, click the EJBClients project.
3. Select **File > Import > File system** and click **Next**. Browse to the directory that you exported your client code to.
4. Expand your export directory and find your files (they will be in a subdirectory under **com**). Select the folders that contain your HelloClient and Increment Client files.
5. Click **Finish**.
6. There will be two errors in the Tasks view (if you did not add the EJBSamples project to the class path, you will have several errors). One occurs because references to javax.ejb.CreateException, which can be found in j2ee.jar, exist. The other occurs because of a reference to com.ibm.ivj.ejb.runtime.AbstractSessionAccessBean, which can be found in ivjeb35.jar.
7. Select **EJBClients** and right-click it. Click **Properties > Java Build Path**.
8. If you did not link to the EJBSamples project, select the **Projects** tab and select the **EJBSamples** check box. There should now only be two errors left.
9. Click the **Libraries** tab and click **Add External JARs**.
10.  Browse to the *WS_InstallDir*\runtimes\aes_v4\lib directory, where *WS_InstallDir* is your product installation directory.
11.  Browse to the /opt/IBM/WebSphere Studio/runtimes/aes_v4/lib directory.
12. Select the j2ee.jar file. Click **Open**, and then click **Add External JARs**.
13.  Browse to the *WS_InstallDir*\runtimes\aes_v4\lib directory.
14.  Browse to the /opt/IBM/WebSphere Studio/runtimes/aes_v4/lib directory.
15. Select the ivjeb35.jar file. Click **Open**, then click **OK**. There should now be no errors in the Tasks view, except the EJB 1.1 warnings.

Note:

-  In order for the IncrementClient to access the IncrementBean, you must set the portNumber and the hostName variables. Do this by changing lines 35 and 36 of IncrementClient.java to:

```
private String portNumber = "2809";  
private String hostName = "localhost";
```
-  HelloClient.java will run on Linux with no changes but only if logged in as "root" as the current Access Bean support does not allow to change the port number that the HelloWorldAccessBean uses in HelloClient.java. When logged in as root, any port number can be used.

At this point the samples have now been migrated into WebSphere Studio Application Developer. All that remains is to test them in the WebSphere test environment.

13.2.8 Creating a WebSphere Studio Application Developer server project

1. Click **File > New > Project > Server > Server Project** and click **Next**. In the **Project name** field, type EJBServer and click **Finish**.

2. The Server perspective will automatically open. In the Navigator view, select **EJBServer** and right-click it.
3. Click **New > Server and Server Configuration**.
4. In the **Server name** field, type **MyEJBServer**. In the **Server type** field, expand **WebSphere Version 4.0**, select **Test Environment**, and then click **Finish**.

13.2.9 Specifying your data source, your EAR project, and then start the server

1. In the Server Configuration view, expand **Servers**, and then select **MyEJBServer**.
2. Right-click it, and from your pop-up menu, click **Open**.
3. Click the **Data source** tab.
4. In the JDBC Driver List, select the **Db2JdbcDriver** and click **Edit**.
5. Verify that the **Class path** field contains the correct path to **db2java.zip**. Click **OK**.
6. Select your JDBC driver, and click the **Add** button to the right of the **Data source** field.
7. In the **Add a data source** dialog, type **EJB Sampledb** in the **Name** field. In the **JNDI name** field, type **jdbc/sampledb**.
8. In the **Database name** field, type **sampledb**. Click **OK**.
9. Save your changes and close the server configuration editor.

Now, you need to specify your EAR project to the server configuration and start the server:

1. In the Server Configuration view, click **Servers > MyEJBServer**.
2. Right-click it and click **Add > EJBSamplesEAR**.
3. In the Server view, right click on **MyEJBServer**, then click **Start**.

13.2.10 Starting DB2 and connecting to sampleDB

Note: DB2 must be running with the sampleDB created and available (as per VisualAge for Java sample). This is only applicable if you are migrating the Increment sample.

Optional: You can perform the following steps to verify your DB2 setup:

1. Switch to the Data perspective.
2. Right-click in the DB Servers view, right-click and select **New Connection**.
3. In the **Connection Name** field, type **sampleDB connection**
4. In the **Database** field, type **SAMPLEDB**.
5. If necessary, type a default user ID and password.
6. In the **JDBC Driver** field, select **IBM DB2 App Driver**.
7. In the **Class Location** field, specify the location of the **db2java.zip** file.
8. Click **Finish**.
9. In the DB Servers view, expand **sampleDB connection** to ensure that the **Tables** subdirectory contains **MYHOST.INCREMENTS**.

13.2.11 Testing the migrated HelloWorld client

1. In the Java perspective, select the **EJBClient** project.

2. Click the **Run** from the Run menu, and then select **Java application** from the Launch Configurations list.
3. Click **New** to create new configuration file.
4. In the main class, select **HelloClient**.
5. On the **JRE** tab, select **WAS V4 JRE** and then select **Run**.
6. In the client window **Input** field, type **XX YY** and click **Send**.
7. The value of the **Output** field becomes you said: **XX YY**. Close the window to terminate the client.



You have successfully migrated and tested the HelloWorld sample.

13.2.12 Testing the migrated Increment client

1. In the Java perspective, select the EJBClients project.
2. Click the **Run** from the Run menu, and then select **Java application** from the Launch Configurations list.
3. In the main class, select **IncrementClient**.
4. On the **JRE** tab, select **WAS V4 JRE** and then select **Run**.
5. The Console view contains the following content:

```
Obtaining initial context., Looking up Increment enterprise bean.,
Obtaining IncrementHome object.,
Looking for increment object named: TEST,
Object named: TEST has count: 0,
Now, object named: TEST has count: 1,
Now, object named: TEST has count: 2,
Now, object named: TEST has count: 3,
Now, object named: TEST has count: 4,
Now, object named: TEST has count: 5
```

13.2.13 Testing the migrated Increment and HelloWorld EJBs using the EJB Test Client

1. In the J2EE Hierarchy view of the J2EE perspective expand the **EJB Modules** folder, right-click **IBMEJBSamples**, select **Run on server**. The EJB Test Client will start running.
2. In the **JNDI Explorer** page, click **HelloWorld** to test the HelloWorld EJB.
3.  In JNDI Explorer, expand **com\ibm\ivj\ejb\samples\increment** and then click **Increment** to test the Increment EJB.
4.  Expand **com/ibm/ivj/ejb/samples/increment** and click **Increment** to test the Increment EJB.

You have successfully migrated and tested the Increment and HelloWorld samples.

In the Servers view of the Server perspective, select **MyEJBServer**, right-click it, and then click **Stop**.

13.3 Example: WebSphere Studio "Classic" Version 4.0 Web Application (YourCo)(Windows)

Description

You must work with WebSphere Studio "Classic" Version 4.0.x for this example.

The sample you are going to work with is the **YourCo** sample. To access this sample open the online help (**Help > WebSphere Studio 4.0 > How to > Work with the samples > Overview**). To load this sample, follow the instructions in **Opening the Studio Samples** (for WebSphere Application Server 4.0) and load **YourCo.war**.

Note: The migrated application will execute in WebSphere Studio Application Developer, but WebSphere Studio Application Developer does not currently provide all the web page design and development features of WebSphere Studio, Professional or Advanced Editions.

Before you begin

- Ensure that the YourCo sample application is loaded in WebSphere Studio "Classic".
- Stop any instances of the WebSphere Application Server (so it will not conflict with WebSphere Studio Application Developer)

Migration steps

To migrate this sample from WebSphere Studio "Classic" to WebSphere Studio Application Developer, you will follow the steps below. Each step is described in more detail below.

1. (Optional) "13.3.1 Starting WebSphere Studio "Classic" Version 4.0 and creating a new Migration stage".
2. "13.3.2 Creating a Web configuration descriptor file".
3. "13.3.3 Creating a migration file".
4. "13.3.4 Starting WebSphere Studio Application Developer and importing the WAR file" on page 63.
5. "13.3.5 Creating a WebSphere Studio Application Developer server project" on page 63.
6. "13.3.6 Testing the migrated YourCo application" on page 64.

13.3.1 Starting WebSphere Studio "Classic" Version 4.0 and creating a new Migration stage

(Optional) Normally, you would create a new stage for a migration, but for the purposes of this example, you use the Test stage included with WebSphere Studio "Classic". Using the Test stage will save you from having to manually edit many servlet mappings in step 8.

For information on how to create a new stage for migration, refer to the "Migrating from WebSphere Studio "Classic" to WebSphere Studio Application Developer".

13.3.2 Creating a Web configuration descriptor file

1. In the project file view, click **Project > Create Web Configuration Descriptor File**, and accept the default value `WEB-INF\localhost_web.xml`.
2. Select all required servlets (any files that are not named xxxxBean).
3. There are no Tag Library Descriptor (TLD) files for this sample.
4. Click **Create**.

13.3.3 Creating a migration file

1. While in the project file view, select server *localhost* and click **Properties > Publishing > WebApp Web Path** and type a web path (context root)

- “newStudioSample”. (Setting a Web path will remain the recommended approach in the final WebSphere Studio Application Developer product).
- 2. While in the project file view, select **Project > Create Migration file**.
- 3. Verify that **localhost** is the selected server.
- 4. Verify that **localhost_web.xml** is the selected Web configuration descriptor file.
- 5. Click **OK**.
- 6. The default JAR file name is `X:\Studio40\projects\YourCo\localhost.jar`, where `X` is your WebSphere Studio “Classic” installation directory.
- 7. Click **Save**.
- 8. Close WebSphere Studio “Classic”.
- 9. Rename the `localhost.jar` file to `localhost.war`.

13.3.4 Starting WebSphere Studio Application Developer and importing the WAR file

- 1. Start WebSphere Studio Application Developer.
- 2. Click **File > Import > WAR file > Next**.
 - Note:** You *must* import the JAR file using the WAR file option, otherwise it will not work properly.
- 3. Type the path to `localhost.war` in the **WAR File** field or click **Browse** to search for it.
- 4. In the **Web Project** field, select **New**, and then type `newStudioSample`
- 5. In the **EAR project name** field, select **New**, and then type `newStudioSampleEAR`
- 6. Click **Finish**. WebSphere Studio Application Developer will unpack `localhost.war`.
- 7. You will have many unresolved references or missing import files. They will appear in the Task view.
 - a. `com.ibm.db` requires `databaseans.jar`,
 - b. `com.ibm.webtools.runtime` requires `webtlsrn.jar`,
 - c. `com.ibm.ejs.ns.jndi` requires `ns.jar`,
 - d. `com.ibm.webshpere.advanced.cm.factory` requires `cm.jar`,
 - e. `com.ibm.ejs.models.base.extensions.webappext.ServletExtensions` requires `ws-base-extensions.jar`

To fix this, you must modify the Java build path for the Web project.

- a. Right-click the project and click **Properties > Java Build Path**.
- b. Click the **Libraries** tab. Click **Add External JARs**.
- c. Import the following JAR files: **databaseans.jar**, **webtlsrn.jar**, **ns.jar**, **cm.jar**, and **ws-base-extensions.jar** from this directory:
`MyInstall\runtimes\aes_v4\lib`
- d. Twenty-four warnings will remain. You do not need to deal with them.
- 8. Right-click the **newStudioSample** project and click **Rebuild Project**.

At this point the sample has been migrated into WebSphere Studio Application Developer. All that remains is to create a WebSphere Studio Application Developer Server project and test the sample in the WebSphere Test Environment.

13.3.5 Creating a WebSphere Studio Application Developer server project

- 1. Switch to the Server perspective.

2. Click **File > New > Project > Server > Server Project**. Click **Next**. In the **Project name** field, type **newServer** and click **Finish**.
3. In the Navigator view, right-click **newServer** and click **New > Server and Server Configuration**.
4. In the **Server name** field, type **WSTestEnv**. In the **Server instance type** field, select **WebSphere V4.0 > Test Environment**. Click **Finish**.

Now, you need to specify your EAR project to the server configuration:

1. In the Server Configuration view, click **Servers > WSTestEnv**.
2. Right-click it and click **Add > newStudioSampleEAR**.

Note: (Optional) Right-click **newStudioSample** project, select **Properties > Server Preference > Always run on the following server**, select **WSTestEnv**, then click **Apply > OK**. (This step is only necessary if you have other servers.)

13.3.6 Testing the migrated YourCo application

1. Select the **YourCoIntro.html** file, which is located in the following directory in your **newStudioSample** project: **WebContent\StudioSamples**
2. Right-click **YourCoIntro.html**, and from its pop-up menu, click **Run on Server**, and then select **WSTestEnv**.
3. Wait while the server starts. Watch the **Console** page (click the **Console** tab in the Servers view) until the message **Server Default Server open for e-business** appears.
4. If you have not already run this sample in WebSphere Studio "Classic", then you need to configure the database by clicking **Database Configuration**.
5. When a browser opens, scroll down and click **Run This Sample**.
6. Wait until the browser **Welcome** page appears, then click **Employee Center**.

Note: The first time you run this application, you will receive the following errors in the Console page: **DataSource not found. Try to construct a new datasource name: jdbc/yourco** **DataSource not found. Try to construct a new datasource name: jdbc/studio**. These errors are self-correcting. You can ignore them.

7. When you are done, close the browser window and the Web Browser view, then in the **Server Control Panel** right-click **WSTestEnv** and click **Stop**.
8. (Optional) Close WebSphere Studio Application Developer.

13.4 Example: Migrating "Example: VisualAge for Java JSP/servlet sample " EJB 2.0 (Increment enterprise bean and HelloWorld session bean)

This example, migrates the "Example: VisualAge for Java JSP/servlet sample" from EJB 1.x to EJB 2.0. Note that this example requires that you have completed "Example: Enterprise beans, VCE, and database samples" and that you have DB2 setup and running for that sample.

13.4.1 Create a new EJB 2.0 project and Enterprise Application 1.3 project

1. Select **File > New > EJB Project > Create EJB 2.0 project**, and then click **Next**.
2. In the **EJB Project Name** field, type **EJBSamplesV2**
3. Select the new EAR project.

4. In the **EAR Project Name** field, type `EJBSamplesV2EAR`, and then click **Finish**.

13.4.2 Import the VisualAge for Java EJB 1.1 JAR into EJB 2.0 project

Import the VisualAge for Java EJB 1.1 JAR file in the "Example: Enterprise beans, VCE, and database samples" section to an EJB 2.0 project.

1. Import the EJB 1.1 JAR (or the EJB 1.1 project) into the EJB 2.0 project:
 - a. Click **File > Import > EJB JAR** file and then click **Next**.
 - b. In the **EJB JAR file** field, browse for the location of the EJB 1.1 JAR that you created from VisualAge for Java.
 - c. Select the existing EJB project.
 - d. In the existing EJB Project name, select **EJBSamplesV2**.
 - e. Click **Finish**.
 - f. You will be asked if you wish to overwrite META-INF/MANIFEST.MF. Click **Yes**.
 - g. You will receive four EJB 1.1 warnings. They will be fixed in the next section.
2. To change the EJB module display name to make it different than your previous EJB 1.1 module name, follow these steps:
3.
 - a. In the Navigator view, expand **EJBSamplesV2 > ejbModule > META-INF** and double-click to open `ejb-jar.xml` with the EJB Deployment Descriptor Editor.
 - b. In the **Overview** tab, change the display name to `IBMEJBSamples-V2`.
 - c. Save and close that EJB deployment descriptor editor window.

13.4.3 Migrate code from EJB 1.0 to EJB 1.1

1. Edit `IncrementBean.java` `ejbCreate()` line 57 to throw `javax.ejb.EJBException` instead of `java.rmi.RemoteException`.
2. Edit `IncrementBean.java` `ejbCreate()` line 57 to return `IncrementKey` instead of `void`.
3. Edit `IncrementBean.java` `ejbCreate()` to add "return null;" as new last line 61.
4. Edit `IncrementBean.java` to add new lines 63 to 65 implementing new `ejbPostCreate()` method:

```
public void ejbPostCreate(IncrementKey argPrimaryKey)
    throws javax.ejb.CreateException, javax.ejb.EJBException {
}
```
5. Save and close `IncrementBean.java` (all of the previous `IncrementBean` warnings should go away).
6. Edit `HelloWorldBean.java` `ejbCreate()` line 23 to throw `javax.ejb.EJBException` instead of `java.rmi.RemoteException`.
7. Save and close `HelloWorldBean.java` (the previous `HelloWorldBean` warnings should go away).

13.4.4 Migrate code from EJB 1.1 to EJB 2.0

Note: The bean migration to EJB 2.0 is not required, since EJB 1.1 beans work correctly in an EJB 2.0 project. However, there is a tool to assist in migrating EJB 1.x beans into EJB 2.0. To use the J2EE Migration wizard,

right-click on the EJB project and then select **Migrate > J2EE Migration Wizard**. For more information about the different wizard options, press F1 while in the J2EE Migration wizard.

The steps below outline the general approach should you choose to undertake this yourself.

1. Use the J2EE Migration wizard to do this initial bean migration from 1.1 to 2.0. For more information about the different wizard options, press F1 while in the J2EE Migration wizard or read the online help section titled "Migrating application modules from J2EE 1.2 to J2EE 1.3".
2. Open IncrementBean.java with the Java Editor.
 - a. In the ejbCreate method, comment out the primaryKey operation and add a set invocation (line 60, new line 61)

```
// primaryKey = argPrimaryKey.primaryKey;
setPrimaryKey(argPrimaryKey.primaryKey);
```
 - b. Modify increment method to get/increment/set the count field

```
public int increment()
{
//    return ++count;
    int c = getCount();
    c += 1;
    setCount(c);
    return c;
}
```
 - c. You may see an error "The method _initLinks is undefined for the type com.ibm.ivj.ejb.samples.increment.IncrementBean". If this is the case, go to the line where the error occurs and delete the method call _initLinks().
 - d. Save and close the Java Editor window for IncrementBean.java.

13.4.5 Generating and deploying RMIC stub and tie code

1. In the J2EE perspective, J2EE Hierarchy view, expand the **EJB Modules** folder, and select **IBMEJBSamples-V2** (or in the J2EE perspective, J2EE Navigator view, select the **EJBSamplesV2** project).
2. Right-click it and then click **Generate > Deploy and RMIC Code**.
3. Select the **Hello World** and **Increment** beans.
4. Click **Finish**.

13.4.6 Specifying the data source binding information

1. In the J2EE Perspective, J2EE Hierarchy view, expand EJB Modules and select **IBMEJBSamples-V2** (or in the J2EE perspective, Navigator view, select the **EJBSampleV2/ejbModule/META-INF/ejb-jar.xml** file in the EJBSamplesV2 project).
2. Right-click the module and select **Open With > Deployment Descriptor Editor**.
3. In the EJB deployment descriptor, click the **Overview** tab.
4. Scroll down to the WebSphere Bindings section. In the JNDI CMP Factory Connection Binding section, type jdbc/sampledb in the **DataSource JNDI name** field. For the Container authorization type, select **Per_Connection_Factory**.
5. In the EJB deployment descriptor editor, click the **Beans** tab.
6. Select the **Increment bean**.

7. In WebSphere Binding section, type `com/ibm/ivj/ejb/samples/increment/Increment` in the **JNDI name** field.
8. Type `jdbc/sampled` in the **CMP Container Factory JNDI Name** field.
9. Select **Per_Connection_Factory** for the Container authorization type.
10. Select the **HelloWorld bean**.
11. In WebSphere Binding section, type `HelloWorld` in the **JNDI name** field.
12. Save then close the EJB deployment descriptor editor.

13.4.7 Creating a new WebSphere Studio Application Developer Java client project

1. Create a new Java project (**File > New > Project > Java > Java Project**).
2. In the **Project name** field, type `EJBSamplesClientsV2`. Click **Next**.
3. Click the **Projects** tab. Select the **EJBSamples-V2** check box.
4. Click **Finish**.

You are now going to import the Java code into the Java project:

1. In the Java perspective, open the Navigator view (**Window > Show view > Navigator**).
2. In the Navigator view, click the **EJBSamplesClientsV2** project.
3. Select **File > Import > File system** and click **Next**. Browse to the directory that you exported your VisualAge for Java client code to.
4. Expand your export directory and find your files (they will be in a subdirectory under `com`). Select the folders that contain your `HelloClient` and `Increment Client` files by selecting **com**.
5. In the **Destination folder** field, type `EJBSamplesClientsV2` (or browse to that folder).
6. Click **Finish**.
7. There will be two errors in the Tasks view (if you did not add the `EJBSamplesV2` project to the class path, you will have several errors). One occurs because references to `javax.ejb.CreateException`, which can be found in `j2ee.jar`, exist.
8. Select **EJBSamplesClientsV2** and right-click it. Click **Properties > Java Build Path**.
9. If you did not link to the `EJBSamplesV2` project, select the **Projects** tab and select the **EJBSamplesV2** check box. There should now only be two errors left.
10. Click the **Libraries** tab, click **Add Variable**, then select **WAS_50_PLUGINDIR**. Click **Extend** then select the `lib/j2ee.jar` file. Click **OK**, then click **OK** again. There are still two errors in the Tasks view (a new one due to `com.ibm.ivj.ejb.runtime.AbstractSessionAccessBean`, which can be found in `ivjeb35.jar`).
11. Repeat the previous **Properties > Java Build Path > Libraries** step to add the `lib/ivjeb35.jar` file. There should now be no errors in the Tasks view.
12. Select the **EJBSamplesClientsV2** project and repeat the **Properties > Java Build Path > Libraries** step to add the `namingclient.jar` file.
13. Select **Add External JARS** and browse to `WS_InstallDir\wstools\eclipse\plugins\com.ibm.websphere.V5_5.0.2\implfactory.jar` to the classpath in order for the application to run.
14. Click **Open** and then **OK** to exit the Properties editor.

At this point the samples have now been migrated into WebSphere Studio Application Developer. All that remains is to test them in the WebSphere test environment.

13.4.8 Creating a server project with an WebSphere Application Server Version 5

1. If you have not previously created a server project, click **File > New > Project > Server > Server Project** and then click **Next**.
2. In the **Project name** field, type **MyEJBServerV5** and then click **Finish**.
3. The Server perspective will automatically open. In the Navigator view, select **MyEJBServerV5** and right-click it.
4. Click **New > Server and Server Configuration**.
5. In the **Server name** field, type **MyEJBServerV5**.
6. In the **Server instance type** field, expand **WebSphere V5.0**, select **Test Environment**, and then click **Finish**.

13.4.9 Specify your data source, your EAR project, and then start the server

1. In the Server Configuration view, expand the servers and select **MyEJBServerV5**.
2. Right-click it, and from your pop-up menu, click **Open**.
3. Click the **Data source** tab.
4. Select the **DefaultDb2JDBCDriver** in the **JDBC Provider List**.
5. Beside the **Data source defined In JDBC driver selected above** list, click **Add**.
 - a. Select **Version 5.0 data source**, and then click **Next**.
 - b. In the **Name** field, type **ejb samples**
 - c. In the **JNDI name** field, type **jdbc/sampledb**
 - d. In the **DataSource Helper Class Name** field, retain **com.ibm.websphere.rsadapter.DB2DataStoreHelper**
 - e. Select ☒ **Use This DataSource In Container Manager Persistence (CMP)**
 - f. Click **Finish**.
6. Select this new EJB samples in the **Data source defined** list.
7. In the **Resource properties defined in the data source selected above** list, select the **databaseName** entry, and then click **Edit**.
 - a. In the **Name** field, retain **databaseName** (note the upper case N).
 - b. In the **Type** field, retain **java.lang.String**
 - c. In the **Value** field, change to **SAMPLEDB**.
 - 1) Edit and then type **SAMPLEDB**
 - 2) Do *not* select the **Required** check box.
 - 3) Click **OK**.
8. Save your changes and close the server configuration editor.

Now, you need to specify your EAR project to the server configuration and start the server:

1. In the Server Configuration view, click **Servers > MyEJBServerV5**.
2. Right-click it and then click **Add > EJBSamplesV2EAR**.

3. (Optional) In Navigator view, right-click **EKBSamplesV2** project, select **Properties > Server Preference > Always run on the following server**, select **MyEJBServerV5**, and then click **Apply > OK**. (This step is only necessary if you have other servers.)
4. In the Server view, right-click on **MyEJBServerV5**, and then click **Start**.

13.4.10 Testing the migrated HelloWorld client

1. In the Java perspective, select the **EJBSamplesClientsV2** project.
2. Click the Run triangle (pull-down), and select **Run...** then select **Java Application**, then click **New**.
3. In the **Name** field, type **HelloWorldV2onV5**
4. In the **Main** tab, leave **Project** as **EJBSamplesClientsV2**.
5. Beside Main class, click **Search**, select **HelloClient**, and then click **OK**.
6. In the **JRE** tab, select **WebSphere V5 JRE** from the pull-down menu, click **Apply**, and then click **Run**.
7. In the client window **Input** field, type **XX YY** and then click **Send**. The value of the **Output** field becomes you said: **XX YY**.
8. Close the window to terminate the client.

You have successfully migrated and tested the HelloWorld sample.

13.4.11 Testing the migrated Increment client

1. In the Java perspective, select the **EJBSamplesClientsV2** project.
2. Click the Run triangle (pull-down), and select **Run**, select **Java Application**, and then click **New**.
3. In the **Name** field, type **Increment2onV5**
4. In the **Main** tab, leave **Project** as **EJBSamplesClientsV2**
5. Beside Main class, click **Search**, select **IncrementClient**, click **OK**.
6. In the **JRE** tab, click the triangle (pull-down), select **V5 JRE**, click **Apply**, and then click **Run**. The Console view contains the following content:

```
Obtaining initial context., Looking up Increment enterprise bean.,

Obtaining IncrementHome object., Looking for increment object named: TEST,
Object named: TEST has count: 0, Now, object named: TEST has count: 1,
Now, object named: TEST has count: 2, Now, object named: TEST has count: 3
Now, object named: TEST has count: 4, Now, object named: TEST has count: 5
```

13.4.12 Testing the migrated Increment and HelloWorld EJBs using the EJB Test Client

1. In the J2EE Hierarchy view of the J2EE perspective, expand the **EJB Modules** folder, right-click **IBMEJBSamplesV2**, and select **Run on server**. The EJB Test Client will start running.
2. In the Unit Test Client home page, click **JNDI explorer**.
3. In the JNDI Explorer page, click **HelloWorld** to test the HelloWorld EJB.
4. In the JNDI Explorer page, expand **com\ibm\ivj\ejb\samples\increment** and click **Increment** to test the Increment EJB.

You have successfully migrated and tested the Increment and HelloWorld samples.

In the Servers view of the Server perspective, select **MyEJBServerV5**, right-click it, and then click **Stop**. In the Debug view, select all the terminated processes, right-click them, and then select **Remove all Terminated**.

Chapter 14. Further reading

Up-to-date information on migration and other topics is available at www.ibm.com/websphere/developer/zones/studio/transition.html

The following publications and Web pages provide general information which you may find helpful when working with WebSphere Studio Application Developer:

- Enterprise JavaBeans Specification, Version 1.1 and Version 2.0:
java.sun.com/products/ejb/docs.html
- JSR-000053 Java Servlet 2.3 and JavaServer Pages 1.2 Specifications:
java.sun.com/aboutjava/communityprocess/first/jsr053/index.html
- The IBM WebSphere Application Server Version 4 InfoCenter:
www.ibm.com/software/webservers/appserv/doc/v40/aes/infocenter/index.html
- *WebSphere Version 4 Application Development Handbook*:
www.redbooks.ibm.com/pubs/pdfs/redbooks/sg246134.pdf
- *Programming J2EE APIs with WebSphere Advanced*:
www.redbooks.ibm.com/pubs/pdfs/redbooks/sg246124.pdf
- *EJB Development with VisualAge for Java for WebSphere Application Server*:
www.redbooks.ibm.com/pubs/pdfs/redbooks/sg246144.pdf
- *EJB 2.0 Development with WebSphere Studio Application Developer*:
www.redbooks.ibm.com/redpieces/pdfs/sg246819.pdf
- *WebSphere Application Server Version 3.5 to 4.x - Migration Hints & Tips*:
www7b.software.ibm.com/wsdd/library/techarticles/0208_wright/wright.html
- *WebSphere Studio Application Developer Service Portal*:
www.ibm.com/software/ad/studioappdev/support/
- *WebSphere Studio Application Developer FAQ Frequently Asked Questions*:
www.ibm.com/support/search.wss?rs=457&tc=SSBRLP&dc=D800
- *WebSphere Application Server Service Portal*:
www.ibm.com/software/software/webservers/appserv/support.html
- *WebSphere Application Server FAQ Frequently Asked Questions*:
www.ibm.com/support/search.wss?rs=180&tc=SSEQTP&dc=D800

Further reading which may be of interest:

- An article on Using Ant with WebSphere Studio Application Developer (including J2EE project builds/exports):
www.ibm.com/websphere/developer/library/techarticles/0203_searle/searle1.html
- An article on Optimizing complex builds in WebSphere Studio Application Developer:
www.ibm.com/websphere/developer/library/techarticles/0204_searle/searle.html
- An article on J2EE Class Loading (J2EE modules and class paths) in WebSphere Studio Application Developer:
www.ibm.com/websphere/developer/library/techarticles/0112_deboer/deboer.html
- An article on developing J2EE utility JARs (Java JARs in J2EE modules) in WebSphere Studio Application Developer:
www.ibm.com/websphere/developer/library/techarticles/0112_deboer/deboer2.html
- An article on team support in WebSphere Studio Application Developer:
www.ibm.com/websphere/developer/library/techarticles/0108_karasiuk/0108_karasiuk.html

- An article on Migrating Enterprise Access Builder Components from VisualAge for Java to WebSphere Studio Application Developer:
www.ibm.com/websphere/developer/techjournal/0201_minocha/minocha.html
- An article on EJB application design using the Session Facade to talk to CMPs
www.ibm.com/websphere/developer/library/techarticles/0106_brown/sessionfacades.html
- An article on WebSphere Application Server Best Practices:
www.ibm.com/software/webservers/appserv/ws_bestpractices.pdf
- An article on WebSphere Best Practices zone:
www.ibm.com/websphere/developer/zones/bp/
- WebSphere Developer Domain main Web page:
www.ibm.com/websphere/developer
- WebSphere Developer Domain Technical Articles:
www.ibm.com/websphere/developer/techjournal/
- Information about the WebSphere Studio family, and the features and directions of WebSphere Studio Application Developer:
www.ibm.com/websphere/developer/zones/studio/transition.html
- All about the IBM WebSphere Studio Family of Development Tools:
www.ibm.com/websphere/developer/library/techarticles/0108_studio/studio_beta.html
- External Application Developer newsgroup:
news://news.software.ibm.com/ibm.software.websphere.studio.application-site-developer
- External workBench (Eclipse) newsgroup:
news://news.software.ibm.com/ibm.software.websphere.studio.workbench
- External WebSphere Application Server newsgroup:
news://news.software.ibm.com/ibm.software.websphere.application-server
- An article on deploying a J2EE Application from WebSphere Studio Application Developer to WebSphere Application Server:
www.software.ibm.com/vad.nsf/Data/Document3584
- Application Developer Software Configuration Management (Source Code Management) vendors:
www.ibm.com/software/ad/studioappdev/partners/scm.html
- Migrating applications to Application Developer from competitors development tools: www.ibm.com/websphere/developer/zones/studio/migration.html
- Migrating VisualCafé WebLogic applications to Application Developer (still deploying to WebLogic):
www.ibm.com/websphere/developer/library/techarticles/0209_searle/searle1.html
- Eclipse.org: www.eclipse.org
- WebSphere Developer Domain Plugin Central:
www.ibm.com/websphere/developer/downloads/plugin/
- Eclipse workbench plug-ins (not part of Eclipse.org): www.eclipse-workbench.com/jsp/plugins.jsp
- Eclipse plug-ins (not part of Eclipse.org): www.eclipse-plugins.2y.net/eclipse/plugins.jsp

Notices

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

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

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

*IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.*

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan*

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

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

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

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

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

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

*Lab Director
IBM Canada Ltd. Laboratory
8200 Warden Avenue
Markham, Ontario, Canada L6G 1C7*

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

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

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

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

COPYRIGHT LICENSE:

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

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

(C) (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. (C) Copyright IBM Corp. 2000, 2003. All rights reserved.

Programming interface information

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

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

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

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

Trademarks and service marks

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

- AIX
- CICS
- Cloudscape
- DB2
- DB2 Extenders
- DB2 Universal Database
- e-business
- IBM
- iSeries
- OS/390
- S/390
- VisualAge
- WebSphere
- z/OS

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

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

UNIX is a registered trademark of The Open Group.

Linux is a registered trademark of Linus Torvalds.

Rational and ClearCase are trademarks or registered trademarks of Rational Software Corp. in the U.S. and other countries.

Other company, product or service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.