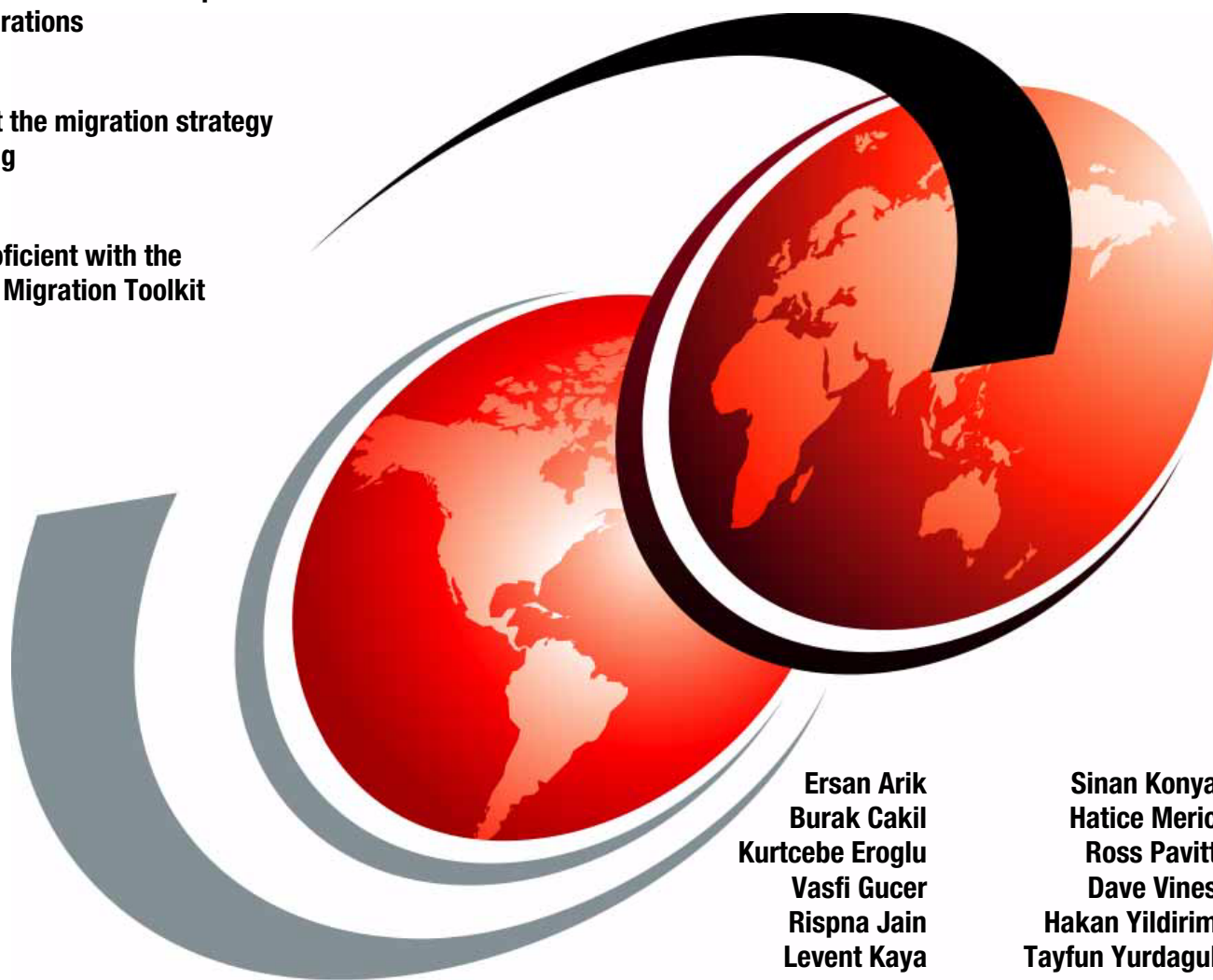


# WebSphere Application Server V8.5 Migration Guide

Review case studies for competitive and version migrations

Learn about the migration strategy and planning

Become proficient with the Application Migration Toolkit



Ersan Arik  
Burak Cakil  
Kurtcebe Eroglu  
Vasfi Gucer  
Rispna Jain  
Levent Kaya

Sinan Konya  
Hatice Meric  
Ross Pavitt  
Dave Vines  
Hakan Yildirim  
Tayfun Yurdagul





International Technical Support Organization

**WebSphere Application Server V8.5 Migration Guide**

November 2012

**Note:** Before using this information and the product it supports, read the information in “Notices” on page xxi.

### **First Edition (November 2012)**

This edition applies to the following IBM products:

- ▶ IBM WebSphere Application Server V8.5
- ▶ IBM Rational Application Developer V8.5
- ▶ IBM WebSphere Application Server Migration Toolkit V3.5
- ▶ IBM DB2 Express Edition V10.1.

© Copyright International Business Machines Corporation 2012. All rights reserved.

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

# Contents

<b>Figures</b> .....	xi
<b>Tables</b> .....	xvii
<b>Examples</b> .....	xix
<b>Notices</b> .....	xxi
Trademarks .....	xxii
<b>Preface</b> .....	xxiii
The team who wrote this book .....	xxiii
Now you can become a published author, too! .....	xxvi
Comments welcome. ....	xxvi
Stay connected to IBM Redbooks .....	xxvi
<b>Part 1. WebSphere Application Server V8.5: Concepts and architecture</b> .....	1
<b>Chapter 1. Overview of WebSphere Application Server V8.5</b> .....	3
1.1 Overview of WebSphere Application Server .....	4
1.2 WebSphere Application Server packaging .....	4
1.3 WebSphere Application Server concepts .....	6
1.3.1 Servers .....	6
1.3.2 Nodes and node groups .....	8
1.3.3 Cells, a deployment manager, and node agents. ....	8
1.3.4 Administrative agents .....	10
1.3.5 Job managers .....	11
1.4 Containers .....	12
1.5 Applications .....	13
1.5.1 Java EE applications .....	13
1.5.2 Portlet applications .....	14
1.5.3 SIP applications .....	14
1.5.4 Business-level applications .....	14
1.5.5 WebSphere Batch applications .....	15
1.5.6 OSGi applications .....	16
1.5.7 Communications enabled applications .....	16
1.5.8 Service Component Architecture .....	16
1.5.9 XML .....	17
1.5.10 WebSphere Application Server Web 2.0 and Mobile Toolkit .....	17
1.5.11 Development and testing of applications .....	17
1.5.12 Deploying, maintaining, and upgrading applications .....	18
1.6 Profiles .....	19
1.7 Workload management .....	20
1.7.1 HTTP servers .....	20
1.7.2 DMZ proxy servers .....	21
1.7.3 Clusters .....	21
1.8 High availability .....	22
1.9 Intelligent management .....	24
1.9.1 Intelligent routing and dynamic operations .....	25
1.9.2 Application edition management .....	26

1.10 Health management . . . . .	27
1.11 Administration and problem determination . . . . .	27
1.11.1 Administration . . . . .	27
1.11.2 Problem determination . . . . .	29
1.12 Messaging . . . . .	30
1.13 Service integration . . . . .	30
1.14 Security . . . . .	31
1.15 Liberty profile . . . . .	34
1.15.1 Liberty profile architecture . . . . .	35
1.15.2 Liberty profile configuration . . . . .	35
1.15.3 Liberty profile administration . . . . .	36
1.15.4 Liberty profile application development and deployment . . . . .	36

**Part 2. Migrating methodology and common issues for migration . . . . . 37**

<b>Chapter 2. Migration strategy and planning . . . . .</b>	<b>39</b>
2.1 Migration process overview . . . . .	40
2.1.1 Types of migrations . . . . .	41
2.1.2 Important aspects and considerations . . . . .	41
2.2 Migration assessment . . . . .	42
2.2.1 Migration questionnaires . . . . .	43
2.2.2 Migration questionnaire review meeting . . . . .	43
2.2.3 Migration assessment report . . . . .	43
2.3 Migration planning . . . . .	44
2.3.1 Roles and responsibilities . . . . .	44
2.3.2 Migration project plan . . . . .	44
2.4 Migration implementation . . . . .	45
2.4.1 Implementation considerations . . . . .	45
2.4.2 Implementation phases . . . . .	45
2.4.3 Application migration . . . . .	47
2.4.4 IDE migration . . . . .	48
2.4.5 Runtime migration . . . . .	52
2.5 Post deployment . . . . .	53
2.5.1 Performance tuning and optimization . . . . .	53
2.5.2 Getting help . . . . .	53
2.6 Summary . . . . .	55
<b>Chapter 3. Common migration issues . . . . .</b>	<b>57</b>
3.1 Java EE application server compatibility . . . . .	58
3.1.1 Differences in Java EE implementations . . . . .	58
3.1.2 Classloader related problems . . . . .	58
3.1.3 Using vendor-specific features . . . . .	65
3.1.4 Deployment descriptors . . . . .	65
3.2 Application portability . . . . .	68
3.2.1 Application packaging . . . . .	68
3.2.2 Java source code and JSP file issues . . . . .	71
3.2.3 Usage of native code . . . . .	71
3.2.4 Database-related issues . . . . .	72
3.2.5 Java EE application clients . . . . .	73
3.3 J2EE to Java EE migration considerations . . . . .	75
3.3.1 Java Message Service . . . . .	78
3.3.2 JavaServer Pages . . . . .	78
3.3.3 Servlets . . . . .	79
3.3.4 JavaServer Faces . . . . .	79

3.3.5	Web services	79
3.3.6	Java Persistence API	81
3.4	Runtime migration issues	82
3.4.1	Migrating other products at the same time	82
3.4.2	Resource definitions	83
3.4.3	Development environment issues	85
3.5	Interoperability and integrations	85
<b>Part 3.</b>	<b>Migrating your applications to WebSphere Application Server V8.5</b>	<b>87</b>
<b>Chapter 4.</b>	<b>Installation and configuration of the Application Migration Tools</b>	<b>89</b>
4.1	IBM WebSphere Application Server Migration Toolkit overview	90
4.1.1	Migration Toolkit basics	92
4.2	Installing the Application Migration Tools on Eclipse	96
4.3	Installing the Application Migration Tools on Rational Application Developer V8.5	98
4.4	Command-line installation	99
4.5	Configuring the Application Migration Tools	100
4.5.1	Running the analysis	101
4.5.2	Configuring and running the analysis	102
4.5.3	Generating reports	107
4.6	Troubleshooting	109
<b>Chapter 5.</b>	<b>Differences between Eclipse and Rational Application Developer</b>	<b>111</b>
5.1	Feature portfolio for Eclipse and Rational Application Developer	112
5.2	Eclipse	113
5.3	WebSphere Application Server Developer Tools for Eclipse	113
5.3.1	Installing WebSphere Developer Tools for Eclipse	114
5.3.2	IBM WebSphere Application Server Developer Tools for Eclipse capabilities	117
5.4	Rational Application Developer	118
5.4.1	Java EE Specifications Upgrade wizard	119
5.4.2	Visualization	119
5.4.3	Portal and portlet	119
5.4.4	iWidgets	120
5.4.5	Java EE Connector Architecture	120
5.4.6	Service Component Architecture	121
5.4.7	JavaServer Faces	121
5.4.8	Communications Enabled Applications	122
5.4.9	Session Initiation Protocol	122
5.4.10	Modern Batch Tools	122
5.4.11	Analysis	123
5.4.12	Profiling	123
5.4.13	Cloud	123
5.4.14	Team Debug	124
5.4.15	Code Review (static analysis)	124
5.4.16	Team Code Coverage	124
5.4.17	XML tools	126
<b>Chapter 6.</b>	<b>Migrating from Oracle WebLogic</b>	<b>127</b>
6.1	Introduction	128
6.1.1	Comparison of WebLogic Domain and WebSphere Cell	128
6.1.2	Comparison of the JMS Providers	129
6.1.3	Migrating sample applications from Oracle WebLogic Server 10.3.6	130
6.2	Prerequisites and assumptions	131
6.3	Oracle WebLogic Server 10.3.6 installation	132

6.4 Application Migration Tool - WebLogic to WebSphere . . . . .	133
6.5 Migrating Trade 3.1 for Oracle WebLogic Server 10.3.6. . . . .	133
6.5.1 Migration approach . . . . .	134
6.5.2 Configuring the initial environment . . . . .	134
6.5.3 Migrating the Trade application. . . . .	140
6.5.4 Summary. . . . .	194
6.6 MVC WebLogic Application migration. . . . .	194
6.6.1 Migration approach . . . . .	194
6.6.2 Migrating the MVC application . . . . .	195
6.6.3 Adding the CMP application into MVC project . . . . .	209
6.6.4 Summary. . . . .	219
<b>Chapter 7. Migrating from Oracle Application Server . . . . .</b>	<b>221</b>
7.1 Introduction . . . . .	222
7.2 Prerequisites and assumptions. . . . .	222
7.3 Application Migration Tool - Oracle AS to WebSphere . . . . .	222
7.4 Comparison of WebSphere Application Server V8.5 and Oracle Application Server 10.1 terminology. . . . .	223
7.5 Installing Oracle Application Server 10.1.2. . . . .	223
7.6 Migrating the sample application. . . . .	224
7.6.1 Migration approach . . . . .	224
7.6.2 Configuring the initial environment . . . . .	224
7.6.3 Importing the sample application into Rational Application Developer V8.5. . . . .	226
7.6.4 Running the Application Migration Tool . . . . .	229
7.6.5 Deploying the application in to WebSphere Application Server V8.5 . . . . .	232
7.6.6 Summary. . . . .	234
<b>Chapter 8. Migrating from JBoss. . . . .</b>	<b>235</b>
8.1 Migrating from older versions . . . . .	236
8.2 Application Migration Tool - JBoss AS to WebSphere . . . . .	236
8.2.1 Comparison of the JBoss V7.1.1 and WebSphere Application Server V8.5 terminology . . . . .	237
8.3 Preparing the environment . . . . .	237
8.3.1 Installing and configuring Apache Ant. . . . .	237
8.3.2 Installing and configuring Apache Maven . . . . .	238
8.3.3 Installing and configuring JBoss Application Server . . . . .	238
8.4 Migrating the Kitchensink application of the JBoss Quickstarts distribution . . . . .	239
8.4.1 Migration approach . . . . .	239
8.4.2 Verifying the Kitchensink sample application . . . . .	240
8.4.3 Importing the EAR file and source code to Rational Application Developer . . . . .	241
8.4.4 Analyzing and fixing migration problems. . . . .	244
8.4.5 Checking the project using Application Migration Tool . . . . .	246
8.4.6 Managing more runtime dependencies. . . . .	248
8.4.7 Migrating to the WebSphere built-in JPA provider and the default data source . . . . .	249
8.4.8 Building and running an application on an integrated test environment . . . . .	253
8.4.9 Summary. . . . .	254
8.5 Migrating the Online Brokerage application . . . . .	254
8.5.1 Migration approach . . . . .	254
8.5.2 Building the Online Brokerage application . . . . .	255
8.5.3 Creating and populating the database . . . . .	255
8.5.4 Importing the EAR file and source code to Rational Application Developer . . . . .	258
8.5.5 Analyzing and fixing the problems . . . . .	261
<b>Chapter 9. Migrating from Apache Tomcat . . . . .</b>	<b>273</b>



9.1	Introduction	274
9.1.1	Comparison of Apache Tomcat 7.0.27 and WebSphere Application Server V8.5 Liberty profile terminology.	275
9.2	Application Migration Tool - Apache Tomcat to WebSphere.	275
9.3	Prerequisites and assumptions.	275
9.3.1	Installation of Apache Tomcat 7.0.27	276
9.3.2	Installing the developer tools	277
9.3.3	Creating a Liberty profile server	278
9.3.4	Installing the Application Migration Tool	279
9.4	IBM Redbooks Publications Sample Application	280
9.4.1	Migration approach	280
9.4.2	Configuring the initial environment	280
9.4.3	Deploying the application in Tomcat	282
9.4.4	Testing the application in Tomcat	282
9.4.5	Migrating the application	282
9.4.6	Configuring the target environment.	287
9.4.7	Deploying the application in the Liberty profile	290
9.4.8	Testing the application in the Liberty profile	290
9.5	MvnForum migration	291
9.5.1	Migration approach	291
9.5.2	Configuring the source environment	292
9.5.3	Deploying the application in Tomcat	294
9.5.4	Testing the application in Tomcat	296
9.5.5	Migrating the MvnForum application	296
9.5.6	Configuring the target environment.	300
9.5.7	Deploying the application in the Liberty profile	301
9.5.8	Testing the application in the Liberty profile	302
9.6	Easy JSP Forum migration	302
9.6.1	Migration approach	303
9.6.2	Configuring the source environment	303
9.6.3	Deploying the application in Tomcat	306
9.6.4	Testing the application in Tomcat	306
9.6.5	Migrating the Easy JSP Forum application to the Liberty profile.	306
9.7	Summary	312
<b>Chapter 10. Application Framework migration.</b>		<b>313</b>
10.1	Migrating a Seam Framework application.	314
10.1.1	Migration approach	315
10.1.2	Installing the Seam Framework.	315
10.1.3	Preparing the database schema	315
10.1.4	Generating a Seam application using seam-gen	317
10.1.5	Importing the EAR file and source code to Rational Application Developer	319
10.1.6	Analyzing and fixing problems	323
10.1.7	Summary.	338
10.2	Migrating a Spring Framework application	338
10.2.1	Migration approach	339
10.2.2	Installing Spring Roo	339
10.2.3	Installing Apache Derby DB	339
10.2.4	Generating a sample application using Spring Roo	340
10.2.5	Importing the EAR file and source code to Rational Application Developer	342
10.2.6	Analyzing and fixing problems	343
10.2.7	Summary.	352

<b>Part 4. Migrating from earlier versions of WebSphere Application Server</b> . . . . .	<b>355</b>
<b>Chapter 11. Installation and configuration of the Application Migration Tool -     WebSphere Version to Version</b> . . . . .	<b>357</b>
11.1 Application Migration Tool - WebSphere Version to Version overview . . . . .	358
11.1.1 Application Migration Tool V3.5 . . . . .	358
11.1.2 WebSphere Application Server V8.5 Liberty profile . . . . .	359
11.2 Installing Application Migration Tool on Eclipse . . . . .	359
11.2.1 Installing Application Migration Tool using Eclipse Marketplace . . . . .	359
11.2.2 Installing Application Migration Tool using the CLI . . . . .	361
11.2.3 Installing Application Migration Tool using Rational Application Developer Version 8.5 . . . . .	362
<b>Chapter 12. Migrating from earlier versions of WebSphere Application Server</b> . . . .	<b>365</b>
12.1 Concepts . . . . .	366
12.1.1 The overall migration process . . . . .	366
12.1.2 Ease of migration . . . . .	369
12.1.3 Known issues . . . . .	369
12.2 Plants by WebSphere sample application migration . . . . .	380
12.2.1 Overview of the application . . . . .	381
12.2.2 Modifying and rebuilding the Plants by WebSphere sample . . . . .	384
12.2.3 Removing and reinstalling the PlantsByWebSphere.ear file . . . . .	385
12.2.4 Installing Plants By WebSphere from the administrative console . . . . .	386
12.2.5 Running the sample application . . . . .	388
12.3 Migration of Plants By WebSphere . . . . .	390
12.3.1 Exporting PlantsByWebSphere.ear and importing it into Eclipse for analysis . . . . .	390
12.3.2 Configuring and running the Application Migration Tool - WebSphere Version to Version . . . . .	394
12.3.3 Fixing errors that are reported by Application Migration Tool . . . . .	399
12.3.4 Other migration problems not detected by Application Migration Tool . . . . .	402
12.3.5 Exporting the fixed Plants By WebSphere application and running the application on WebSphere Application Server V8.5 . . . . .	403
12.4 Probability Distribution Sample . . . . .	404
12.4.1 Importing the application into Rational Application Developer V8.5 . . . . .	404
12.4.2 Installing the application on WebSphere Application Server V6.1 . . . . .	405
12.4.3 Testing the application on WebSphere Application Server V6.1 . . . . .	405
12.4.4 Running the Application Migration Tool - WebSphere Version to Version . . . .	405
12.4.5 Resolving the reported migration problems . . . . .	406
12.4.6 Preparing the application in Rational Application Developer for installation on WebSphere Application Server V8.5 . . . . .	408
12.4.7 Installing the application on WebSphere Application Server V8.5 . . . . .	408
12.4.8 Testing the application on WebSphere Application Server V8.5 . . . . .	409
12.4.9 Options for upgrading the application after migration . . . . .	409
12.5 Web services Axis 2 stock quote . . . . .	410
12.5.1 Overview . . . . .	410
12.5.2 Installation on WebSphere Application Server V6.1 . . . . .	410
12.5.3 Testing on WebSphere Application Server V6.1 . . . . .	412
12.5.4 Importing the source of the application into Eclipse . . . . .	413
12.5.5 Running the Application Migration Tool - WebSphere Version to Version . . . .	413
12.5.6 Preparing to install on WebSphere Application Server V8.5 . . . . .	414
12.5.7 Installing on WebSphere Application Server V8.5 . . . . .	416
12.5.8 Configuring WebSphere Application Server V8.5 to support the application . .	416
12.5.9 Testing the application on WebSphere Application Server V8.5 . . . . .	417
12.5.10 Summary of migrating the Web services Axis 2 stock quote application . . . .	418

<b>Part 5. Appendixes</b> .....	419
<b>Appendix A. Migration questionnaires</b> .....	421
Business requirements .....	422
General information .....	422
Application architecture .....	422
Dependencies .....	423
Persistence .....	423
National language .....	423
Code .....	424
Java .....	424
EJB usage .....	424
Servlets and JSPs .....	424
Web services .....	425
Database access .....	425
JMS .....	426
JNDI naming .....	426
Application trace and logging .....	426
Struts .....	427
Transactions .....	427
Threads .....	427
Sockets .....	427
XML .....	428
Development migration questionnaire .....	428
Workstation configuration .....	428
Integrated development environment .....	428
Development test configuration .....	428
Software development skills .....	428
Development methodology .....	429
Build and packaging .....	429
Ant .....	429
Runtime migration questionnaire .....	430
General .....	430
Current hardware .....	430
Software .....	431
HTTP Server .....	431
Network edge .....	431
Availability .....	432
Rollout issues .....	432
Administration .....	432
Security .....	433
Testing migration questionnaire .....	433
Hardware .....	433
Practices and tools .....	434
<b>Appendix B. Additional material</b> .....	435
Locating the web material .....	435
Using the web material .....	435
System requirements for downloading the web material .....	436
Downloading and extracting the web material .....	436
<b>Related publications</b> .....	437
IBM Redbooks .....	437
Other publications .....	437

Online resources .....	437
Help from IBM .....	439

# Figures

1-1	WebSphere Application Server V8.5 editions	5
1-2	Stand-alone application server environment	9
1-3	Network Deployment environment	9
1-4	Administrative agent in a stand-alone configuration	10
1-5	Job manager architecture	11
1-6	WebSphere Application Server V8.5 container services	12
1-7	Business-level applications	15
1-8	Files in WebSphere Application Server	19
1-9	Application server cluster	21
1-10	Dynamic cluster	22
1-11	Conceptual diagram of a core group	23
1-12	Dynamic cluster	26
1-13	Overview of WebSphere security service	32
1-14	Liberty profile architecture	35
2-1	Migration process overview	40
2-2	Migration key activities	46
2-3	Select an import source window	49
2-4	Select the EAR file	49
2-5	Selecting target run time	50
2-6	Select the JAR file	50
2-7	Existing project into workspace	51
2-8	Browse and select the root directory or archive file	52
3-1	Java classloaders hierarchy	59
3-2	A typical Java EE classloader	60
3-3	WebSphere Application Server classloader hierarchy	61
3-4	Application server classloader settings	62
3-5	WAR classloader policy settings	63
3-6	Preferred packaging structure of a Java EE application	69
3-7	Clients that are provided for WebSphere Application Server	74
3-8	Java EE Specifications Migration wizard - J2EE Version selection	77
3-9	Increases the version level of the project facet - before migration	77
3-10	Increases the version level of the project facet - after migration	78
3-11	Converts .xmi files to .xml files - before migration	78
3-12	Converts .xmi files to .xml files - after migration	78
3-13	Typical infrastructure for web services applications	80
4-1	Steps for planning and implementation for migrating an application	90
4-2	Migration aspects	91
4-3	Example configuration for migrating from WebLogic to WebSphere	92
4-4	Modules with red Xs indicating that there are one or more errors	93
4-5	A high-level flow of how the Migration Toolkit works	93
4-6	Software Analyzer Reporting pane for creating analysis reports	94
4-7	Sample HTML report	95
4-8	Help view	96
4-9	Add Repository window for adding the Application Migration Tool as a repository	97
4-10	Plug-in selection in the Install window	98
4-11	Application server settings	99
4-12	Analysis option	101
4-13	Software Analyzer	102

4-14	Software Analyzer . . . . .	102
4-15	Creating a Software Analyzer configuration . . . . .	103
4-16	Software Analyzer Configurations . . . . .	103
4-17	New configuration . . . . .	104
4-18	WebLogic Application Migration rules . . . . .	105
4-19	Rule set configuration . . . . .	106
4-20	Software Analysis Results view . . . . .	106
4-21	Generating reports . . . . .	107
4-22	Create a report . . . . .	108
4-23	Sample report . . . . .	109
5-1	Feature portfolio for Eclipse and Rational Application Developer . . . . .	112
5-2	Eclipse Marketplace results window . . . . .	114
5-3	Security warning when installing WebSphere Application Server Version 8.5 Developer Tools . . . . .	115
5-4	Adding a server . . . . .	115
5-5	Servers view . . . . .	116
5-6	Mobile Browser Simulator . . . . .	118
5-7	J2C wizards on Rational Application Developer . . . . .	120
5-8	SCA diagram . . . . .	121
5-9	Code Coverage properties . . . . .	125
5-10	Code Coverage . . . . .	125
6-1	Comparison of components between WebLogic and WebSphere . . . . .	128
6-2	Comparison of JMS Providers . . . . .	129
6-3	Trade3 J2EE components . . . . .	134
6-4	Existing Project Import . . . . .	142
6-5	Trade Module Selections . . . . .	143
6-6	Workspace Migration . . . . .	143
6-7	Project resource file in workspace . . . . .	144
6-8	Define Server Runtime . . . . .	145
6-9	Migration Validation . . . . .	145
6-10	Application Migration Tool configuration . . . . .	146
6-11	Migration tool workspace scope settings . . . . .	147
6-12	Migration Tool rule settings . . . . .	148
6-13	Rule set configuration . . . . .	149
6-14	Software Analyzer View . . . . .	149
6-15	Java Code Review tab . . . . .	150
6-16	Problems view . . . . .	151
6-17	Setting Java Build Path . . . . .	152
6-18	Problems view . . . . .	152
6-19	Log.java . . . . .	153
6-20	Project Clean dialog box . . . . .	154
6-21	Migration tool analysis . . . . .	155
6-22	Java Code Review tab . . . . .	156
6-23	Quick fix preview . . . . .	156
6-24	Java Code Review tab . . . . .	157
6-25	TradeBean.java . . . . .	157
6-26	Vendor-specific deployment descriptors . . . . .	158
6-27	XML File Review tab . . . . .	158
6-28	Quick Fix Preview . . . . .	159
6-29	EJB JNDI Quick Fix . . . . .	159
6-30	WebSphere Binding file . . . . .	160
6-31	XML File Review tab . . . . .	161
6-32	Convert the CMP Mappings file . . . . .	162

6-33	CMP mapping file . . . . .	163
6-34	Database schema files . . . . .	164
6-35	Create and map database elements to the database table . . . . .	164
6-36	Show unmapped objects. . . . .	165
6-37	Class-Path Review tab . . . . .	166
6-38	Quick Fix View of the MANIFEST.MF file . . . . .	166
6-39	WebLogic EJB deployment descriptors . . . . .	167
6-40	WebLogic web deployment descriptor . . . . .	167
6-41	J2EE Migration wizard . . . . .	168
6-42	J2EE version selection . . . . .	169
6-43	Migration Completion dialog box. . . . .	170
6-44	Creating a Service Integration Bus . . . . .	172
6-45	Service Integration Bus. . . . .	173
6-46	WebSphere scope selection . . . . .	175
6-47	MDB activation specification configuration . . . . .	179
6-48	Project J2EE specification . . . . .	179
6-49	Add and Remove Projects . . . . .	185
6-50	Enable application security . . . . .	186
6-51	Creating a user . . . . .	187
6-52	Manage Users. . . . .	187
6-53	Creating a group . . . . .	188
6-54	Group properties . . . . .	188
6-55	Adding member into the group . . . . .	189
6-56	Testuser properties . . . . .	189
6-57	Project facets of the trade3wls EAR file . . . . .	190
6-58	Checking the security role name. . . . .	190
6-59	Mapping group in Rational Application Developer . . . . .	191
6-60	Authentication window . . . . .	192
6-61	Trade Application welcome page . . . . .	192
6-62	Trade Application Configuration options. . . . .	193
6-63	Import the EAR file . . . . .	195
6-64	Enterprise Explorer View . . . . .	196
6-65	XML File Review . . . . .	198
6-66	XML file Quick Fix Preview . . . . .	198
6-67	The contents of ibm-ejb-jar-bnd.xmi . . . . .	199
6-68	Java Code Review . . . . .	199
6-69	The Problems Category . . . . .	200
6-70	The Java Build Path of MVCWeb Application. . . . .	200
6-71	Quick Fix Preview of MVCController.java. . . . .	201
6-72	The code changes that the Migration Tool finds. . . . .	201
6-73	JDBC Providers . . . . .	202
6-74	Create New JDBC Provider . . . . .	203
6-75	The database classpath information . . . . .	204
6-76	Selecting the existing JDBC Provider . . . . .	205
6-77	The database properties. . . . .	205
6-78	Specifying JAAS-J2C authentication data . . . . .	206
6-79	New authentication alias . . . . .	206
6-80	Security settings . . . . .	207
6-81	Test connection. . . . .	207
6-82	Test connection message. . . . .	207
6-83	Run on the server . . . . .	208
6-84	The test result . . . . .	208
6-85	Import EJB JAR file. . . . .	210

6-86	EJB Application . . . . .	210
6-87	XML File Review tab . . . . .	211
6-88	Generate Map . . . . .	212
6-89	Database Connection definition . . . . .	213
6-90	Edit Jar List . . . . .	214
6-91	The database table selection . . . . .	215
6-92	Match By Name . . . . .	216
6-93	Mapped EJB objects . . . . .	216
6-94	Backend ID in ejb-jar.xml . . . . .	217
6-95	Defining the JNDI Name in ejb-jar.xml . . . . .	217
6-96	Creating an EJB object . . . . .	218
6-97	Use the set and get methods . . . . .	219
7-1	The Markers view in Rational Application Developer after you import all the EAR files that make up the sample application . . . . .	226
7-2	The remaining problems after the target run time is set . . . . .	227
7-3	Setting the Java compliance level . . . . .	228
7-4	Rule set configuration for migrating from Oracle AS . . . . .	230
7-5	The Java Code Review results from the Application Migration Tool . . . . .	230
7-6	Applying a quick fix to resolve the Java serialization warning . . . . .	231
7-7	XML Code Review results . . . . .	231
7-8	Add and Remove Projects . . . . .	233
7-9	Adding missing JAR files . . . . .	234
8-1	Importing the WAR file for Kitchensink application using the import wizard . . . . .	241
8-2	Project structure after the initial import . . . . .	242
8-3	Removing the shirinkwrap libraries from the project . . . . .	243
8-4	Replacing the class file with the respective source files in the web module . . . . .	244
8-5	Selecting the Faces Servlet . . . . .	245
8-6	Adding a servlet mapping to the Faces Servlet . . . . .	246
8-7	Software Analyzer configuration with JBoss migration rule set . . . . .	247
8-8	Application Migration Tool XML file review . . . . .	248
8-9	Web module after you import more dependencies . . . . .	249
8-10	Enabling the JPA Project Facet . . . . .	250
8-11	Enterprise Explorer after you enable the JPA project facet . . . . .	251
8-12	Configuring JPA validation warnings . . . . .	251
8-13	Adding factory class path for annotation processing at compile time . . . . .	252
8-14	Enabling the metamodel generation . . . . .	253
8-15	Creating the brkrpdb database for the sample application . . . . .	257
8-16	brokerage_WEB module before and after the source code import . . . . .	258
8-17	Importing source code files to brokerage_WEB module . . . . .	259
8-18	Java Code Review section . . . . .	260
8-19	XML File Review section . . . . .	260
8-20	Adding a dependency to HAR resources . . . . .	262
8-21	JDBC provider settings . . . . .	265
8-22	Data Source details . . . . .	266
8-23	Data Sources window of the WebSphere Application Server deployment . . . . .	267
8-24	Resource references for brokerage_WEB . . . . .	268
8-25	Creating a utility module . . . . .	269
8-26	Importing the source files in to the utility project . . . . .	270
8-27	Utility module contents . . . . .	270
8-28	Module dependencies of web module . . . . .	271
9-1	The developer tools for operating a Liberty profile server from Eclipse . . . . .	278
9-2	The security warning when you install the developer tools . . . . .	278
9-3	Creating a Software Analyzer configuration . . . . .	283



9-4	Setting the rules for an Apache Tomcat migration . . . . .	284
9-5	Settings for the configuration of the target WebSphere version, and the source and target JVM levels . . . . .	284
9-6	Results of running the Software Analyzer . . . . .	287
9-7	The location of the View menu . . . . .	293
9-8	Software Analyzer new configuration . . . . .	298
9-9	Files selection - do not select the forum folder itself . . . . .	304
9-10	Creating a package in the src directory . . . . .	304
9-11	Files selection from beans directory - do not select the beans directory itself . . . . .	305
9-12	Software Analyzer new configuration . . . . .	309
10-1	EAR import wizard . . . . .	320
10-2	Importing myproject.ear . . . . .	321
10-3	Selecting EAR Modules during import . . . . .	322
10-4	EJB module structure before and after source code import . . . . .	322
10-5	Importing source files . . . . .	323
10-6	Before and after libraries in EAR moved to a shared library . . . . .	324
10-7	Shared library properties . . . . .	325
10-8	EJB module class path . . . . .	326
10-9	Framework migration rules . . . . .	327
10-10	XML File Review . . . . .	328
10-11	Java Code Review . . . . .	328
10-12	ClassPath review . . . . .	329
10-13	Previewing a quick fix . . . . .	329
10-14	JDBC provider settings . . . . .	332
10-15	Selecting provider for data source . . . . .	333
10-16	Data source details . . . . .	334
10-17	Resource properties for data source . . . . .	335
10-18	Resource properties for a data source . . . . .	336
10-19	Web module dependencies . . . . .	337
10-20	WAR import wizard . . . . .	342
10-21	Importing a sample application WAR file . . . . .	343
10-22	Shared library properties . . . . .	344
10-23	Framework migration rules . . . . .	345
10-24	XML File review . . . . .	346
10-25	JDBC provider settings . . . . .	348
10-26	Selecting a provider for a data source . . . . .	349
10-27	Data Source details . . . . .	350
10-28	Resource properties for data source . . . . .	351
11-1	Eclipse Marketplace . . . . .	360
11-2	Select the features . . . . .	361
11-3	Install the Migration Toolkit using Rational Application Developer V8.5 . . . . .	362
11-4	Available Software window . . . . .	363
12-1	Migration process . . . . .	366
12-2	Plants by WebSphere application high-level architecture . . . . .	382
12-3	Step 1: Select installation options . . . . .	387
12-4	Step 2: Map modules to servers . . . . .	387
12-5	Index page of Plants By WebSphere sample . . . . .	389
12-6	Login/Registration page of Plants By WebSphere sample . . . . .	389
12-7	Shopping cart of Plants By WebSphere . . . . .	390
12-8	Exporting the PlantsByWebSphere.ear file from the WebSphere Application Server administration console . . . . .	391
12-9	Importing the EAR file . . . . .	392
12-10	Select the EAR file to import . . . . .	393

12-11	EAR modules and Utility JAR projects during application import . . . . .	394
12-12	Software Analyzer configuration . . . . .	395
12-13	Setting up a new Software Analyzer configuration . . . . .	396
12-14	Setting rules for the Software Analyzer configuration . . . . .	397
12-15	Rule set configuration dialog box . . . . .	397
12-16	A ready Software Analyzer configuration . . . . .	398
12-17	Software Analyzer results . . . . .	398
12-18	Java Code Review . . . . .	399
12-19	XML File Review . . . . .	399
12-20	View Java Code Review results . . . . .	400
12-21	Clean Java Code Review results . . . . .	401
12-22	Fixing XML file review results . . . . .	402
12-23	Configuring the Application Migration Toolkit for the Probability Distribution Sample application . . . . .	406
12-24	Installing the Axis 2 application on WebSphere Application Server V6.1 . . . . .	411
12-25	Configuring the Application Migration Toolkit for the Stock Quote Axis2 application	414
12-26	Selecting the “parent last” option for class loader order . . . . .	417

# Tables

3-1	Java and vendor deployment descriptors and configuration files	67
3-2	Java EE version numbers	75
4-1	Application Migration Tools and installIU	100
6-1	Explanation of the cell and domain components	128
6-2	Explanation of the JMS components	129
6-3	Explanation of JMS extensions	130
6-4	Data source values	136
6-5	Database connection properties values	136
6-6	JMS Server values	137
6-7	JMS queue values	137
6-8	JMS topic values	138
6-9	JMS Topic values	138
6-10	Queue connection factory values	138
6-11	Topic connection factory values	138
6-12	Explanation of WebLogic specific file names	161
6-13	JMS queue connection factory configuration	175
6-14	JMS topic connection factory configuration	176
6-15	JMS queue configuration	176
6-16	JMS queue configuration	176
6-17	JMS topic configuration	177
6-18	JMS activation specification configuration	177
6-19	JMS activation specification configuration	178
6-20	OrderProcessor MDB's WebSphere binding configuration	178
6-21	Mailer MDBs WebSphere binding configuration	180
6-22	JDBC provider configuration	180
6-23	JDBC data source configuration	181
6-24	J2C authentication alias configuration	181
6-25	WebSphere bindings for the Account EJB	182
6-26	Trade EJB JNDI names	182
6-27	EJB reference names	183
6-28	Default data source configuration	183
6-29	Trade web application reference JNDI names	184
6-30	The contents of the MVC projects	196
6-31	The contents of the EJB projects	211
7-1	Terminology comparison	223
7-2	Creating the data source	225
8-1	Terminology comparison	237
8-2	Required libraries	262
9-1	Terminology comparison	275
9-2	Parameters for the Fileset Service	300
9-3	Parameters for the Properties for DB2 JCC	301
9-4	Parameters for the Fileset Service	310
9-5	Parameters for the Properties for DB2 JCC	311
10-1	Shared dependencies to be copied	325
12-1	EJB changes	371
12-2	JAX-WS changes	372
12-3	JCA enhancements	372
12-4	JPA changes	372

12-5 JSP changes . . . . .	373
12-6 OSGi changes . . . . .	373
12-7 SIP changes . . . . .	373
12-8 Support for security custom properties . . . . .	377
12-9 Miscellaneous classes that are removed . . . . .	378
12-10 MustGather table . . . . .	379

# Examples

3-1	Web.xml excerpt . . . . .	66
3-2	WebSphere Application Server V8.5 deployment descriptor excerpt . . . . .	66
3-3	Thin client looking up an EJB deployed on a single server. . . . .	73
4-1	Installing the Migration Toolkit through the command line . . . . .	99
4-2	Uninstalling the Application Migration Tools through the command line . . . . .	100
6-1	Basic authentication definition in web.xml . . . . .	139
6-2	Mapping definition in weblogic.xml . . . . .	139
6-3	org.eclipse.wst.common.project.facet.core file . . . . .	144
6-4	WebLogic Query Language construct. . . . .	161
6-5	The basic authentication settings in web.xml . . . . .	185
6-6	Table.ddl file . . . . .	209
7-1	The corrected persistence.xml file . . . . .	229
8-1	Warning message . . . . .	252
8-2	Command area code snippet . . . . .	256
8-3	Changes to the file . . . . .	263
9-1	Required configuration of the required global data source . . . . .	281
9-2	The configuration that is required to add a user and a Test role . . . . .	282
9-3	The commands that are required to create the necessary DB2 databases . . . . .	288
9-4	Database creation commands . . . . .	294
9-5	Changing the mvncore.xml file code snippet . . . . .	294
9-6	Define data source in mvncore.xml. . . . .	297
9-7	Define paths and trusted domains in mvncore.xml . . . . .	297
9-8	Updating mvnforum.xml with the location of the mvnforum_home directory. . . . .	298
9-9	Contents of a Liberty profile server configured to run the MvnForum application. . . . .	302
9-10	Creating the EasyJSP database . . . . .	306
9-11	Creating the data structure and sample data of the EasyJSP database . . . . .	307
9-12	Adding the required packages to access the data source . . . . .	308
9-13	Changing the connection parameters . . . . .	308
9-14	Remove the square brackets in the following SQL syntaxes . . . . .	308
9-15	The resource reference to add to the deployment descriptor . . . . .	310
10-1	Code snippet. . . . .	316
10-2	Seam setup command interactive output . . . . .	317
10-3	Code snippet. . . . .	330
10-4	Roo script file for generating a sample application . . . . .	340
10-5	Overriding hashCode and equals methods. . . . .	341
11-1	Installation . . . . .	361
11-2	Uninstall . . . . .	361
12-1	Command to build the sample application . . . . .	385
12-2	Command to uninstall the PlantsByWebSphere sample application on Windows . . . . .	385
12-3	Command to uninstall the PlantsByWebSphere sample application on Linux . . . . .	386
12-4	Command to install the PlantsByWebSphere sample application on Windows . . . . .	386
12-5	Command to install the PlantsByWebSphere sample application on Linux . . . . .	386
12-6	Java code to be analyzed . . . . .	400
12-7	Usage of the equals method of Duration class leading to a flag. . . . .	401
12-8	Usage of equals method of Duration class leading to a flag. . . . .	401
12-9	A snippet from the error.jsp file included in the PlantsByWebSphere. . . . .	403
12-10	The manifest.mf file for the jsf-ibm.jar in the Probability Distribution Sample . . . . .	407
12-11	The EnableChildFirstClassLoading property in the axis2.xml file . . . . .	415

12-12 The META-INF/MANIFEST.MF file in the WebContent of stock\_quote\_example\_axis2  
project. .... 415

# Notices

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

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

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

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

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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

## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate 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.

# Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

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

AIX 5L™	IBM SmartCloud™	Redbooks®
AIX®	IBM®	Redbooks (logo)  ®
CICS®	IMS™	System p®
DB2 Universal Database™	MQSeries®	System x®
DB2®	MVS™	System z®
developerWorks®	Passport Advantage®	WebSphere®
GDDM®	Rational Team Concert™	z/OS®
Global Technology Services®	Rational®	

The following terms are trademarks of other companies:

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

ITIL is a registered trademark, and a registered community trademark of The Minister for the Cabinet Office, and is registered in the U.S. Patent and Trademark Office.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

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



# Preface

This IBM® Redbooks® publication helps you plan and execute the migration of J2EE applications that are developed for Oracle WebLogic Server, Oracle Application Server, JBoss, and Apache Tomcat, so that they run on IBM WebSphere® Application Server V8.5. In addition, this book covers migration from earlier versions of WebSphere Application Server to WebSphere Application Server V8.5.

This book provides detailed information to plan migrations, suggested approaches for developing portable applications, and migrating working examples for each of the platforms from which we migrated in our examples. The primary tool that is used in the migration scenarios that are covered in this book is the IBM WebSphere Application Server Migration Toolkit V3.5.

It is not our intention to provide a feature-by-feature comparison of these application servers versus WebSphere Application Server, but to produce practical technical advice for developers who must migrate applications from these vendors to WebSphere Application Server V8.5.

This publication is an update of *WebSphere Application Server V7: Competitive Migration Guide*, SG24-7870.

The book is intended as a migration guide for IT specialists who are working on migrating applications that are written for other application servers or earlier versions of WebSphere Application Server to WebSphere Application Server V8.5.

## The team who wrote this book

This book was produced by a team of specialists from around the world working at the IBM Office in Istanbul, Turkey.

**Ersan Arik** is an IT Specialist working for Akbank in Turkey acting as Solution Architect Job Role with WebSphere Application Server products. He has nine years of professional experience in IT and banking industries, the last five years of which were dedicated to implementing integrated solutions using WebSphere Application Server and IBM WebSphere MQ. His areas of expertise include infrastructure architecture, capacity planning, clustering and high availability architecture for mission-critical environments, and performance tuning on cross platforms. He also specializes in Document Management Systems products, such as IBM Content Management and OnDemand Server. He holds a Bachelor's degree in Computer Engineering from Istanbul Technical University.

**Burak Cakil** is an IT Specialist and has six years of IT experience. He is the Team Leader of Web Middleware Enablement Service Line Component in Integrated Technology Delivery (ITD) SSO in Turkey, and has four years of experience in implementation, design, and administration of WebSphere Application Server, IBM Content Manager, and J2EE technologies. He is certified in WebSphere Application Server administration and IBM Content Manager solution design. He designs J2EE applications for the internal needs of ITD SSO Turkey. He gave lectures about WebSphere Application Server, IBM DB2®, and J2EE components in several universities in Turkey. He has a Bachelor of Science degree in Computer Engineering from Yildiz Technical University, and is studying for a Master of Science degree in the same program.

**Kurtcebe Eroglu** works for Sade Yazilim Ve Danismanlik as an IT specialist. He has over 10 years of experience in IT, focusing on application integration middleware and enterprise software development. He has a Bachelor of Science degree in Environmental Engineering, a Master of Science degree in Computer Engineering, a Master of Science degree in Engineering Management from Middle East Technical University, and is studying for a degree in Philosophy.

**Vasfi Gucer** is a Project Leader at the IBM International Technical Support Organization. He has been with the ITSO since January 1999. He has more than 12 years of experience in the areas of Systems Management, networking hardware, and software on mainframe and distributed platforms. He writes extensively and teaches IBM classes worldwide on IBM products. Vasfi is also an IBM Certified Senior IT Specialist, PMP, and ITIL Expert.

**Rispna Jain** is a Technical Software Deployment Manager for the WebSphere suite of products in IBM Global Technology Services® and works with clients in North America. She has seven years of experience with WebSphere Application Server product development at IBM Software Group in various roles, such as development, Level 3 support, and testing. Rispna has also been a technical speaker for WebSphere Application Server related topics at various WebSphere conferences. She is an IBM Certified SOA associate and holds a Master of Technology degree in Computer Science.

**Levent Kaya** is an IT Specialist working for IBM Software Services for WebSphere (ISSW) in Turkey for two years. He has five years of experience in the IT industry. He has two years of development experience with IBM WebSphere Process Server. He has a Bachelor of Science degree in Computer Engineering from Yildiz Technical University

**Sinan Konya** is the Chief Architect and founder of eteration a.s. He is an object mentor and an educator. He is a founding member of ObjectLearn and wrote ObjectWeb Lombox in 2000, which is open source software for JavaEE development. Before he worked at eteration, Sinan was with BEA Systems Inc. and The Object People as a Mentor. He has more than 17 years of experience in designing and implementing enterprise-wide and mission-critical systems. He has been involved in teaching object technology and enterprise software development using Java and JavaEE technologies for over 15 years in various industries. He has a Master of Science degree in Control and Computer Engineering from Istanbul Technical University. He was involved as a committer in the Eclipse WTP.

**Hatice Meric** is a Technical Consultant working for IBM Istanbul Innovation Center in Turkey. In her previous IBM role, she was an IT Specialist in Software Services at IBM Turkey. She has a Bachelor of Science degree in Computer Engineering from Yildiz Technical University.

**Ross Pavitt** is a Software Engineer at IBM Hursley in the United Kingdom. He joined the WebSphere Application Server Liberty profile team in 2012 as a test lead, focusing on customer scenario testing. Before that position, he was part of the IBM CICS® Transaction Gateway team. Ross joined IBM in 2008 and focused on functional verification and other automated testing for multiple platforms, including IBM System z®, IBM System x® and IBM System p®.

**Dave Vines** is an IBM Master Inventor and Senior Software Engineer at the IBM Hursley Laboratory in the United Kingdom working in WebSphere Application Server Development specializing in the Liberty profile and IBM Workload Deployer. He joined the laboratory in 1984 and has worked on various IBM licensed programs, including GDDM®, IBM MQSeries®, LANDP and, since 1998, on Component Broker and WebSphere Application Server. He received a Bachelor of Science degree from the University of Exeter, England, in 1984.

**Hakan Yildirim** is a Senior Specialist in Platforms and Application Management at Garanti Teknoloji, Turkey. Hakan has over 14 years of experience, mainly in the financial sector. He graduated from Yildiz Technical University and studied in their Electronics & Communication Engineering department. He started his professional carrier at Emlak Bank in 1998. He worked as engineer, specialist, and senior specialist in Istanbul, Turkey. He worked extensively with the WebSphere suite of products, such WebSphere Application Server, WebSphere Portal Server, WebSphere Process Server, and WebSphere ESB. His core areas of skill are WebSphere Application Server administration and the planning and design of WebSphere Application Server environments. He joined Garanti Teknoloji in 2009 as a senior platforms and application management specialist. He gained his experience by working with developers to debug Java applications. He participated in many large banking projects, primarily internet banking, and e-commerce solutions. Currently, Hakan is in charge of managing Java application servers, such as WebSphere and WebLogic.

**Tayfun Yurdagul** is a Solution Architect at VBT Vizyon Information Technologies. He has more than nine years of experience in IBM WebSphere, IBM Rational® products, DB2, and Java technologies. He worked for large and mission-critic software projects. Tayfun also has DB2, WebSphere, and Rational product certificates.

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

David Bennin, Emma Jacobs, Wade Wallace, Erica Wazewski  
**International Technical Support Organization**

Dana Duffield, Surya Duggirala, Cindy High, Roman Kharkovski, Tom Kristek, Dr. Alex Mulholland, Mike Morin, Rengan Sundararaman, Donald Vines, Nathan Ziemann  
**IBM USA**

Graham Hopkins and Ian Robinson  
**IBM UK**

Jale Akyel, Gulsun Emuler, Yesim Mutlu, Serkan Sahin, Engin Turpcular, Kivanc Uslu  
**IBM Turkey**

Andrea Bauer and Tim deBoer  
**IBM Canada**

Ping Shao  
**IBM China**

The team would like to express special thanks to the project's Executive Sponsor Richard Baird, IBM VP, WebSphere Foundation Development for his continuous support throughout the project.

Thanks to the authors of the previous editions of this book:

- ▶ Authors of the first edition, *WebSphere Application Server V8.5 Migration Guide*, SG24-7870 published in July 2010, were:
  - Joao Emilio Santos Bento da Silva, Kurtcebe Eroglu, Kiran Mantripragada, Hamdy Eed, Fabio Xavier Albertoni

## Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at: [ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

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

- ▶ Use the online **Contact us** review Redbooks form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- ▶ Send your comments in an email to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

## Stay connected to IBM Redbooks

- ▶ Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- ▶ Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- ▶ Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- ▶ Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>



# Part 1

# WebSphere Application Server V8.5: Concepts and architecture

This part provides an overview of WebSphere Application Server V8.5, including the lightweight Liberty profile. If you are new to WebSphere Application Server or to the Liberty profile that is introduced in V8.5, read this part first.





# Overview of WebSphere Application Server V8.5

IBM WebSphere Application Server helps drive business agility with an innovative and performance-based foundation to build, reuse, run, integrate, and manage service-oriented architecture (SOA) applications and services. From business critical enterprise-wide applications to the smallest departmental level applications, WebSphere Application Server offers reliability, availability, security, and scalability.

WebSphere Application Server V8.5 addresses the needs of today's agile enterprises and developers. For the enterprise, it provides a choice of application server solutions with fidelity for the available editions. It also provides increased scalability, resiliency, and security for critical applications and the flexibility to deploy new offerings quickly and efficiently. It includes a lightweight and powerful, yet simple, application server to satisfy multiple requirements around a simplified "low-end" application environment. For the developer, it provides an improved developer experience and a simplified server configuration that can be versioned and maintained in source control along with the applications.

This chapter provides an overview of WebSphere Application Server V8.5, including the lightweight Liberty profile. This chapter describes the following topics:

- ▶ Overview of WebSphere Application Server
- ▶ WebSphere Application Server packaging
- ▶ WebSphere Application Server concepts
- ▶ Containers
- ▶ Applications
- ▶ Profiles
- ▶ Workload management
- ▶ High availability
- ▶ Health management
- ▶ Administration and problem determination
- ▶ Messaging

- ▶ Service integration
- ▶ Security
- ▶ Liberty profile

## 1.1 Overview of WebSphere Application Server

WebSphere Application Server is the industry's leading standards-based, high performance, proven, and trusted application foundation that is optimized for developer and operational productivity. WebSphere Application Server V8.5 is a major step forward in several areas, especially resiliency and developer productivity.

WebSphere Application Server V8.5 builds on over a decade of leadership for Java application servers by expanding its programming model portfolio to allow more flexibility in application development and by improving administrator productivity, increasing security robustness, and delivering new functions for added resiliency.

WebSphere Application Server V8.5 provides the following enhancements:

- ▶ Rapid delivery of new applications and services
- ▶ Improved operational efficiency and reliability
- ▶ Increased security and control

For a full description of the new features of WebSphere Application Server V8.5, see *WebSphere Application Server V8.5 Concepts, Planning, and Design Guide*, SG24-8022.

## 1.2 WebSphere Application Server packaging

Because different application scenarios require different levels of application server capabilities, WebSphere Application Server is available in multiple packaging options and on a range of platforms. WebSphere Application Server also serves as the base for other WebSphere products and many other IBM software products. In addition to the application server component, each package contains an appropriate combination of complementary products, for example, IBM HTTP Server, IBM Assembly and Deploy Tools for WebSphere Administration, Edge components, and other products. As the business grows, the WebSphere Application Server family provides a migration path to more complex configurations.



Figure 1-1 illustrates the various WebSphere Application Server editions.

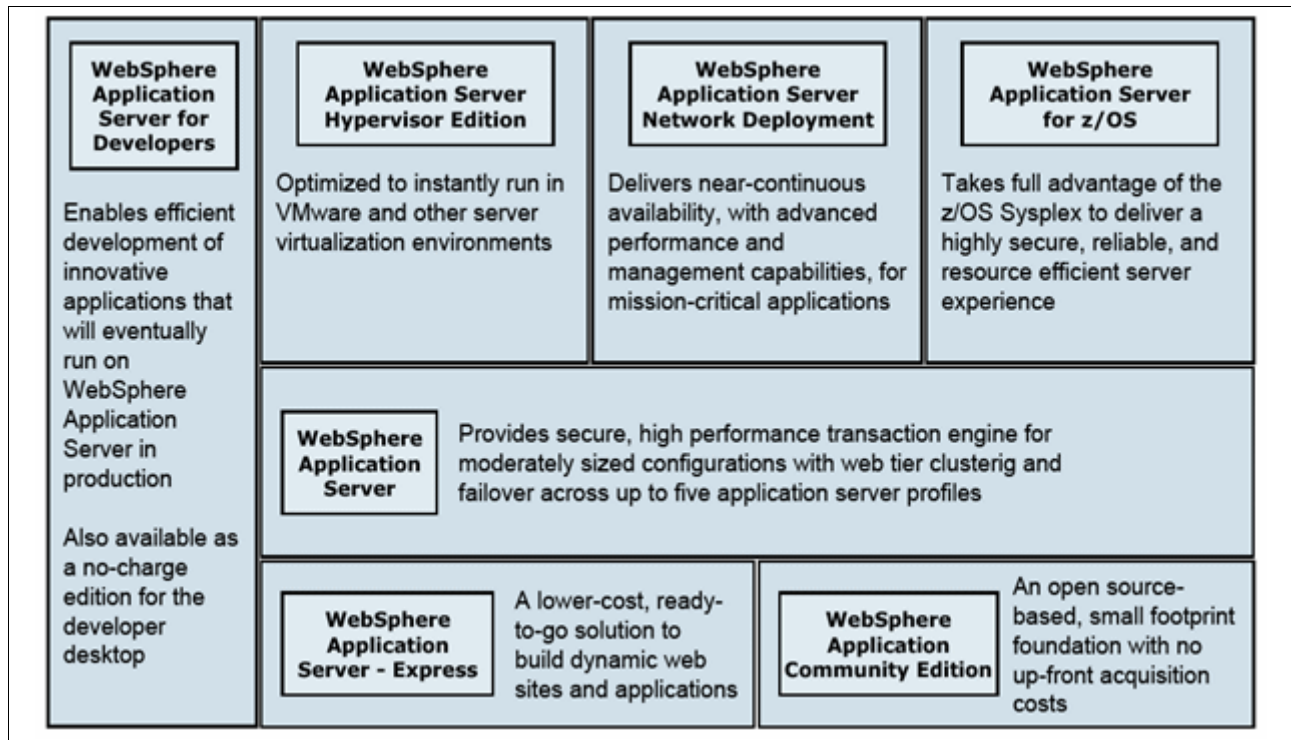


Figure 1-1 WebSphere Application Server V8.5 editions

WebSphere Application Server is available in as the following packages:

► WebSphere Application Server—Express

The WebSphere Application Server—Express V8.5 package provides a ready-to-go application foundation for single server and small scale deployments of dynamic web applications. When your business needs grow, this package can be easily integrated in to more advanced versions of the WebSphere Application Server family.

For more information about WebSphere Application Server—Express V8.5, see:

<http://www.ibm.com/software/webservers/appserv/express/>

► WebSphere Application Server—Base

The WebSphere Application Server V8.5—Base package is the next level of server infrastructure in the WebSphere Application Server family. Although the WebSphere Application Server package is functionally equivalent to WebSphere Application Server—Express (single-server environment), this package differs in packaging and licensing. This package is ideal for lightweight application solutions where cost and simplicity are key.

For more information about WebSphere Application Server- Base V8.5, see:

<http://www.ibm.com/software/webservers/appserv/was/>

- ▶ WebSphere Application Server Network Deployment

WebSphere Application Server Network Deployment V8.5 provides enterprises with advanced performance, management, and high-availability for mission-critical applications. It extends the base package of WebSphere Application Server to include key features that tend to be more important in larger enterprises, where applications tend to service a larger client base and where more elaborate performance and high availability requirements are in place. These features include various functions for high availability and high scalability, to ensure responsiveness to application requests.

For more information about WebSphere Application Server Network Deployment V8.5, see:

<http://www.ibm.com/software/webservers/appserv/was/network/>

- ▶ WebSphere Application Server for z/OS®

IBM WebSphere Application Server for z/OS V8.5 provides the same functions of WebSphere Application Server Network Deployment but adds key functions that use z/OS qualities of service. The z/OS qualities of service allow the user to optimize performance and provide continuous availability for mission critical applications.

For more information about WebSphere Application Server for z/OS V8.5, see:

[http://www.ibm.com/software/webservers/appserv/zos\\_os390/](http://www.ibm.com/software/webservers/appserv/zos_os390/)

- ▶ WebSphere Application Server for Developers

The WebSphere Application Server for Developers V8.5 package is functionally equivalent to the Base and Express packages, but it is licensed for development use only. It provides simplified and no-charge access to enable developers to build and test in the same environment that ultimately supports their applications.

For more information about WebSphere Application Server for Developers V8.5, see:

<http://www.ibm.com/software/webservers/appserv/developer/index.html>

- ▶ WebSphere Application Server Liberty profile

The Liberty profile is a small (less than 50 MB download) and fast (less than a five second start) profile of the WebSphere Application Server focused on web, OSGi, and mobile applications. The Liberty profile can be installed using Installation Manager from any of the WebSphere Application Server editions or can be downloaded separately.

For more information about WebSphere Application Server Liberty profile V8.5, see:

[https://www.ibm.com/developerworks/mydeveloperworks/blogs/wasdev/entry/download\\_wlp](https://www.ibm.com/developerworks/mydeveloperworks/blogs/wasdev/entry/download_wlp)

## 1.3 WebSphere Application Server concepts

WebSphere Application Server is organized based on the concept of cells, nodes, and servers. All of these elements are present in each configuration. However, cells and nodes are largely hidden from view until the features of Network Deployment are used. These cells, nodes, and servers can be managed by a combination of deployment managers, administrative agents, and job managers.

### 1.3.1 Servers

WebSphere Application Server V8.5 provides the infrastructure and capabilities that are required to host applications and proxy servers that distribute work to the application servers.

It also defines generic external servers to the administration process and to associate web servers for routing purposes.

When you use WebSphere Application Server V8.5, you can have the following types of servers:

- ▶ Application servers
- ▶ Proxy servers
- ▶ Generic servers

### **Application servers**

The *application server* is the core of the WebSphere Application Server product and is the runtime environment in which business applications run. It provides containers and services, such as database connectivity, threading, and workload management, that can be used by applications. Each application server runs in its own Java virtual machine (JVM) and can have one or many applications that are deployed to it.

Because each application server runs in a single JVM, you can deploy applications using one of the following options, which are based on machine resources and risk:

- ▶ A model that uses one application per application server can reduce the risk of JVM issues that are caused by one application that affects other applications. However, more machine resource impact for each JVM can occur. So, there is a trade-off.
- ▶ A model that uses multiple applications per application server optimizes system resources by minimizing the JVM machine resource impact. However, poorly written applications can cause negative impacts to other applications within the same JVM.

WebSphere Application Server V8.5 introduces the concept of the *Liberty profile*. The Liberty profile is a composable and dynamic application server runtime environment. It is a subset of the functionality of the Base and Express packages and is installed separately, using either Installation Manager or a downloadable compressed file format. It is a stand-alone server, similar to the stand-alone Base or Express application servers, that uses an easy-to-configure XML configuration file format.

The Liberty profile is a simplified and lightweight run time for web applications. The small footprint and low impact, along with a simplified configuration, makes the WebSphere Application Server V8.5 Liberty profile a good option for developers who are building web applications that do not require the full Java EE environment of traditional enterprise application server profiles.

The Liberty profile provides a lightweight development and application-serving environment for web and OSGi applications that is configured with the correct level of capabilities that are required for the individual applications. You can use the Liberty profile to specify only those features that are required for the applications that are deployed, reducing the memory footprint and increasing performance.

The Liberty profile is optimized for developer and operational productivity. You can use it in both development and production environments. Within the development environment, the Liberty profile supports the same platforms as the base application server and Mac OSX. In addition, enterprise qualities of service, such as security and transaction integrity, are enabled as required.

For details about the Liberty profile, see 1.15, “Liberty profile” on page 34.

## Proxy servers

A *proxy server* is a specific type of application server that routes requests to content servers that perform the work. The proxy server is the initial point of entry, after the protocol firewall, for requests that enter the environment.

You can use WebSphere Application Server to create the following types of proxy servers:

- ▶ The *WebSphere Application Server Proxy* classifies, prioritizes, and routes HTTP and Session Initiation Protocol (SIP) requests to servers in the enterprise and to cache content from servers.
- ▶ The *DMZ Secure Proxy Server* comes in a separate installation package and provides security enhancements to allow deployments inside of a DMZ. It improves security by minimizing the number of external ports that are opened and running as an unprivileged user when you bind to well-known ports.
- ▶ The *on-demand router* (ODR) is an intelligent Java based HTTP proxy server and SIP proxy server that is built on the WebSphere run time. The ODR manages the flow of requests to application servers based on a set of configurable routing rules. It is asynchronous, high performing, scalable, and can be clustered for high availability.

## Generic servers

A *generic server* is a server that can be managed as part of the administrative domain of WebSphere Application Server, but it is not supplied by WebSphere Application Server. With the generic servers function of WebSphere Application Server, a generic server can be defined as an application server instance within the WebSphere Application Server administration and associate it with a non- WebSphere Application Server or process.

### 1.3.2 Nodes and node groups

A *node* is an administrative grouping of application servers for configuration and operational management within one operating system instance. Virtualization allows multiple operating systems on one machine. You can create multiple nodes inside one operating system instance, but a node cannot leave the operating system boundaries. A stand-alone application server configuration has only one node. With Network Deployment, you can configure a distributed server environment that consists of multiple nodes, which are managed from one central administration server.

A *node group* is collection of nodes within a cell that have similar capabilities, in terms of installed software, available resources, and configuration. A node group is used to define a boundary for server cluster formation so that the servers on the same node group host the same applications. A node group validates that the node can perform certain functions before it allows them. A DefaultNodeGroup is created automatically. This DefaultNodeGroup contains the deployment manager and any new nodes with the same platform type. A node can be a member of more than one node group.

### 1.3.3 Cells, a deployment manager, and node agents

A *cell* is a group of nodes in a single administrative domain that encompasses the entire management domain. In the Base, Developer, and Express packages, a cell contains one node and that node contains one server, which is largely hidden from view. Each cell contains at least one node, at least one application server, and an administration console. The configuration and application files for all nodes in the cell are centralized into the *master repository*.

Multiple stand-alone application servers can exist on a system, either through independent installations of WebSphere Application Server binary files or by creating multiple application server profiles within one installation (Figure 1-2).

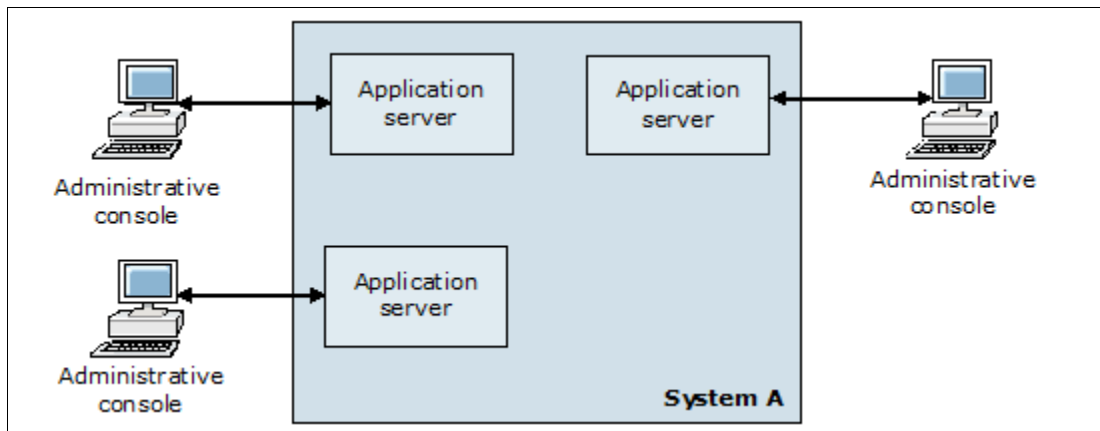


Figure 1-2 Stand-alone application server environment

In a Network Deployment and WebSphere Application Server for z/OS environments, a cell can consist of multiple nodes and node groups, which are all administered from a single point, the *deployment manager* (using GUI or scripting) (Figure 1-3).

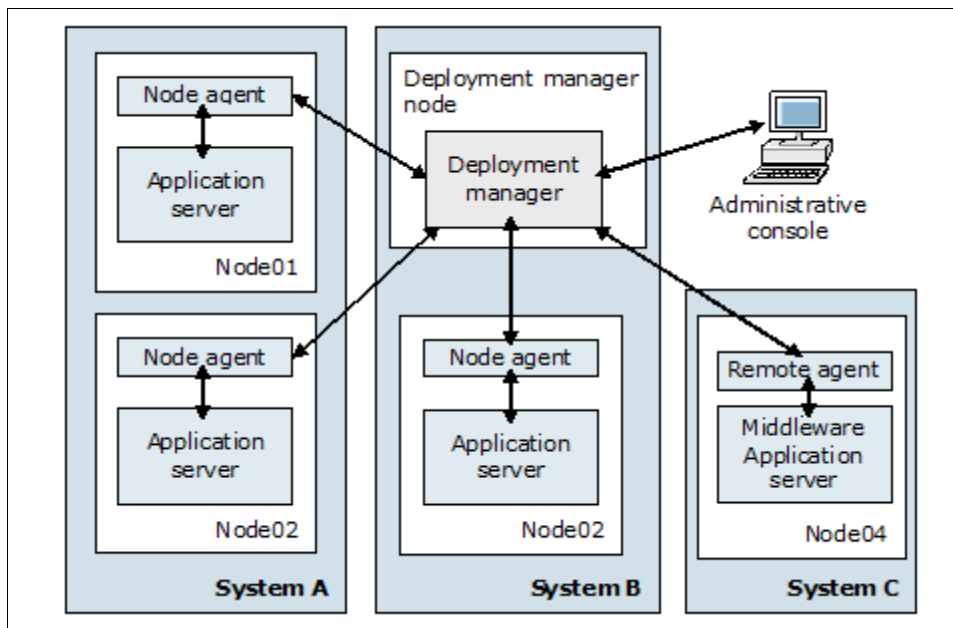


Figure 1-3 Network Deployment environment

Cells can contain nodes that are running any of the supported operating systems or platforms, including spanning z/OS Sysplex environments and other operating systems. For example, z/OS nodes, Linux nodes, UNIX nodes, and Windows system nodes can exist in the same WebSphere Application Server cell.

The deployment manager is used to administer the cell from a single point. It communicates with the *node agents* of the cell to manage the applications servers within each node. A node is federated into the cell and can be managed only by the deployment manager that is managing that cell. The node agent is an administrative server that is running in the node. It monitors the application servers on that node and routes administrative requests from the deployment manager to those application servers.

The master repository is managed by the deployment manager and regularly synchronized with local copies that are held on each of the nodes.

### 1.3.4 Administrative agents

An *administrative agent* (Figure 1-4) is a component that provides enhanced management capabilities for stand-alone (Express and Base) application servers. All configurations that are related to the application server are connected directly to the administrative agent that provides services to administrative tools.

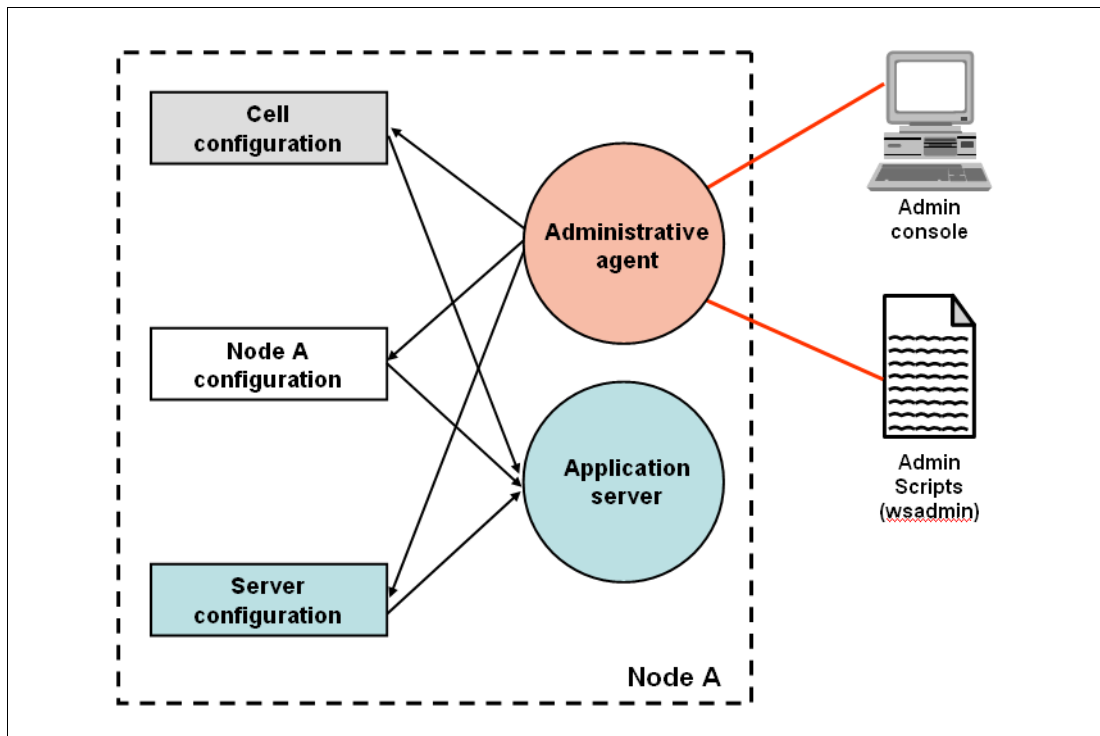


Figure 1-4 Administrative agent in a stand-alone configuration

An administrative agent can manage multiple stand-alone server instances on a single system or z/OS image. When you use an administrative agent, because the number of application server instances increases, the redundancy of the administration footprint (for each application server) is eliminated. The administrative agent acts as the main component for the expanded multiple node remote management environment that is provided with the job manager.

### 1.3.5 Job managers

The *job manager* (Figure 1-5) is available with WebSphere Application Server Network Deployment and WebSphere Application Server for z/OS. It provides centralized management capabilities for multiple stand-alone application servers, administrative agents, and deployment managers. From a daily task perspective, the job manager can run daily tasks in one step for multiple installations, including such tasks as starting and stopping servers, and distributing and deploying applications. These jobs can be submitted and run asynchronously to application server environments and topologies and to any number of servers over a geographically dispersed area.

From an installation perspective, you can use a job manager to provide enhanced multiple node installation options for your environment by submitting centralized installation manager jobs to multiple cells and server topologies. Job Manager can submit jobs to stand-alone application servers and WebSphere Application Server Liberty profiles.

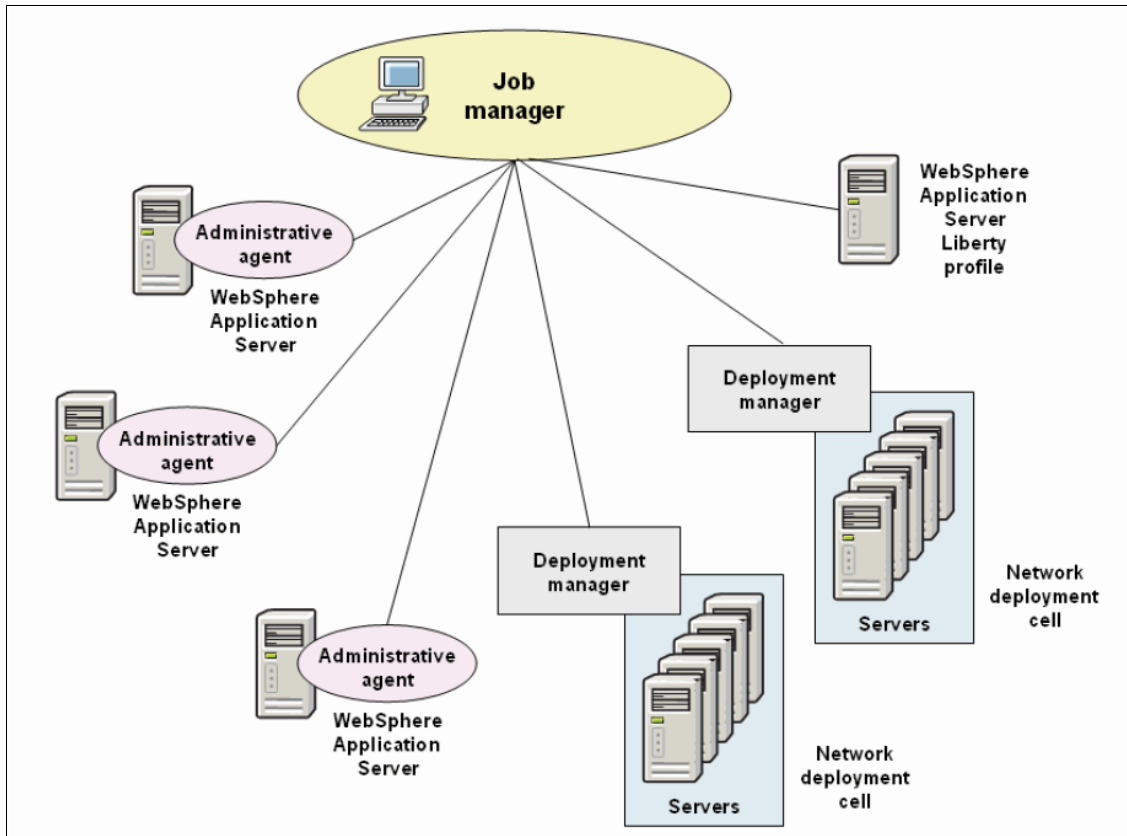


Figure 1-5 Job manager architecture

## 1.4 Containers

Figure 1-6 illustrates applications that run in different containers inside the JVM.

*Containers* provide runtime support for applications. They are specialized to run specific types of applications and can interact with other containers by sharing session management, security, and other attributes.

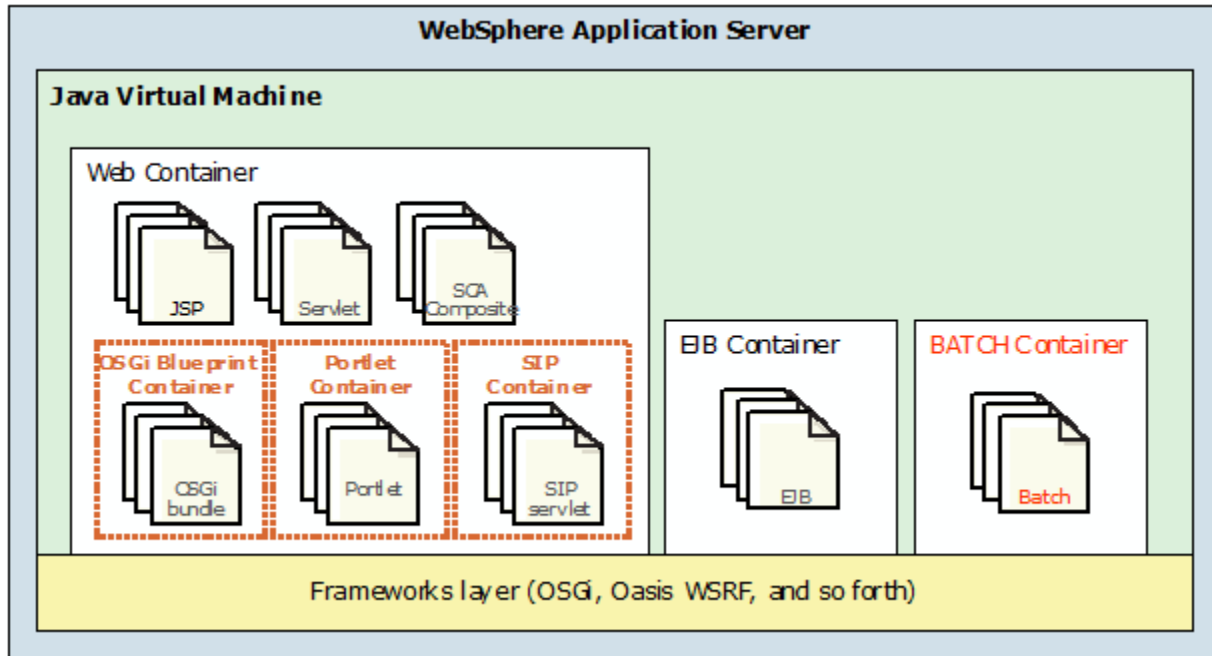


Figure 1-6 WebSphere Application Server V8.5 container services

WebSphere Application Server V8.5 includes the following logical containers:

- ▶ The *Web container* processes servlets, JSPs, and other types of server-side objects. Each application server run time has one logical web container. Requests are received by the web container through the web container *inbound transport chain*.
- ▶ The *Enterprise JavaBeans (EJB) container* provides all of the runtime services that are needed to deploy and manage enterprise beans. This container is a server process that handles requests for both session and entity beans. A single container can host more than one JAR file.
- ▶ The *Batch container*, new in WebSphere Application Server v8.5, is where the batch scheduler runs jobs that are written in XML job control language (xJCL). The batch container provides an execution environment for the execution of Java EE -based batch applications. Batch applications are deployed as EAR files and follow either the transactional batch or compute-intensive programming models.

The following containers can be considered as logical extensions of the web container main functionality:

- ▶ The *Portlet container* provides the runtime environment to process JSR 286-compliant portlets. The portlet container is an extension to the web container. A simple portal framework is built on top of the container to render a single portlet into a full browser page.



- ▶ The *SIP container* processes applications that use at least one SIP servlet that is written to the JSR 289 specification. It provides network services over which it receives requests and sends responses. It determines which applications to start and in what order. The container supports UDP, TCP, and TLS/TCP protocols.
- ▶ The *OSGi Blueprint container* processes OSGi applications that are based on the OSGi framework. Blueprint is separated from Java EE technology, but it can be mixed in to deploy modular applications that use both Java EE 6/7 and OSGi R4 V4.2 technologies.

## 1.5 Applications

At the heart of WebSphere Application Server is its ability to run *applications*. WebSphere Application Server V8.5 supports multiple technologies and programming models that can be used within enterprise applications. In addition to the programming models and technologies, applications can use either the Java 6 or Java 7 Java runtime environment (JRE). Java 6 is installed with the product and used by default. WebSphere Application Server V8.5 supports IBM WebSphere SDK Java Technology Edition Version 7.0 as a pluggable JDK that can be optionally installed and enabled by using the `managesdk` tool.

For more information, see the IBM SDK Java Technology Edition Version 7 Information Center at:

<http://publib.boulder.ibm.com/infocenter/java7sdk/v7r0/index.jsp>

The following programming models and technologies are supported in WebSphere Application Server V8.5:

- ▶ Java EE applications
- ▶ Portlet applications
- ▶ SIP applications
- ▶ Business-level applications
- ▶ WebSphere Batch applications
- ▶ OSGi applications
- ▶ Communications enabled applications
- ▶ Service Component Architecture
- ▶ XML
- ▶ WebSphere Application Server Web 2.0 and Mobile Toolkit

### 1.5.1 Java EE applications

The Java EE specification is the standard for developing, deploying, and running enterprise applications. WebSphere Application Server V8.5 provides full support for the Java EE 6 specification. The Java EE programming model has multiple types of application components:

- ▶ Enterprise beans
- ▶ Servlets and JavaServer Pages
- ▶ Application clients

WebSphere Application Server V8.5 includes IBM Assembly and Deploy Tools for WebSphere Administration. This tool replaces the previously available IBM Rational Application Developer Assembly and Deploy Tool.

For information about the Java EE specification, see:

<http://www.oracle.com/java>

## 1.5.2 Portlet applications

The portlet container in WebSphere Application Server provides the runtime environment for Java Specification Request (JSR) 286 compliant portlets. Portlet applications are intended to be combined with other portlets collectively to create a single page of output. Portlets are packaged in web archives (WAR files). The portlet run time does not provide the advanced capabilities of WebSphere Portal, such as portlet aggregation and page layout, personalization and member services, or collaboration features.

For more information about JSR 286, see:

<http://jcp.org/en/jsr/detail?id=286>

## 1.5.3 SIP applications

SIP applications are Java programs that use at least one SIP servlet that is written to the JSR 116 specification. WebSphere Application Server V8.5 also supports SIP Servlet Specification 1.1, also referred to as *JSR 289*. SIP is used to establish, modify, and terminate multimedia IP sessions. SIP negotiates the medium, the transport, and the encoding for the call. After the SIP call is established, the communication takes place over the specified transport mechanism, independent of SIP. Examples of application types that use SIP include voice over IP (VOIP), click-to-call, and instant messaging.

In the application server, the web container and SIP container are converged and can share session management, security, and other attributes. In this model, element of an application that includes SIP servlets, HTTP servlets, and portlets can interact, regardless of the protocol. High availability of these converged applications is made possible because of the integration of HTTP and SIP in the base application server.

For more information about SIP applications, see the following resources:

- ▶ JSR 289 SIP Servlet API 1.1 Specification:  
<http://www.jcp.org/aboutJava/communityprocess/final/jsr289/index.html>
- ▶ JSR 116:  
<http://jcp.org/en/jsr/detail?id=116>
- ▶ RFT 3261:  
<http://www.ietf.org/rfc/rfc3261.txt>

## 1.5.4 Business-level applications

A *business-level application* (BLA) expands the definition of an application beyond the Java EE definition. Typically, a business thinks of the pieces that make up an application, including both WebSphere and non- WebSphere artifacts, such as SCA packages, libraries, and proxy filters, as being under a single application definition that is stored in the product configuration repository. Business-level applications do not introduce new programming, runtime, or packaging models. Thus, application business logic or runtime settings do not need to change.

Figure 1-7 illustrates business-level applications.

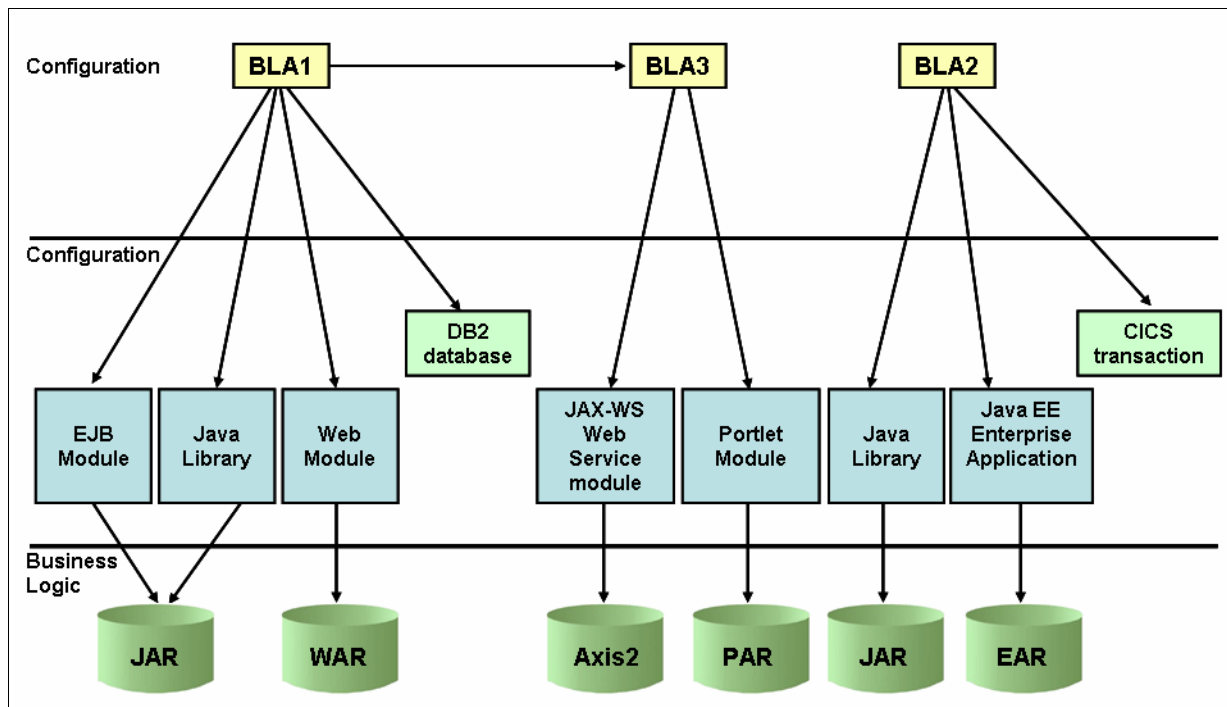


Figure 1-7 Business-level applications

Business-level applications have the following characteristics:

- ▶ Contain one or more composition units (CUs) that represent the application binary files.
- ▶ Can span several WebSphere Application Server deployment target run times, such as a proxy server, a web server, and WebSphere Application Server Community Edition.
- ▶ Provide installation, distribution, activation, monitor, update, and removal management features for applications.
- ▶ Support application service provider (ASP) scenarios by allowing single application binary files to be shared between multiple deployments.
- ▶ Align WebSphere applications closer with the business rather than IT configuration.

### 1.5.5 WebSphere Batch applications

Batch applications can perform long running bulk transaction processing and compute-intensive work. They run as background jobs, described by a job control language, and are supported by infrastructure components that are aimed at supporting batch workloads. WebSphere Batch provides both a transactional batch programming model and a compute-intensive programming model. Both models are implemented as Java objects, which are packaged in an enterprise archive (EAR file), and deployed into the application server environment. The individual programming models provide details about how the lifecycle of the application and jobs that are submitted to it are managed by WebSphere Application Server V8.5.

The XML job control language provides a method to describe the steps that are involved in the batch jobs. This application runs in batch containers that also run at the same time in designated WebSphere Application Server environments. A job scheduler determines the appropriate container to run the job and maintains job histories to support job visibility and control.

### 1.5.6 OSGi applications

OSGi applications are modular applications that use both Java EE and OSGi technologies. With this design, you can improve control and flexibility by designing and building applications and groups of applications from coherent, multiversion, and reusable OSGi bundles.

WebSphere Application Server support for OSGi applications includes the capability of deploying web applications that use the Java Servlet 3.0 Specification. Additionally, you can update a running application and impact only those bundles that are affected by the update. You can use this method to extend and scale running applications as your business demands change.

You can use OSGi applications to compose isolated enterprise applications using multiple, and multiversion bundles that have dynamic lifecycles. Application maintenance and upgrades can be simplified using standard OSGi mechanisms to simultaneously load multiple versions of classes in the same application. Existing web archives (WAR files) can be reused and deployed as OSGi web application bundles.

For more information about OSGi applications, see:

<http://www.osgi.org/About/WhatIsOSGi>

### 1.5.7 Communications enabled applications

Communications enabled applications (CEA) allow developers to add dynamic web communications to any application or business process. CEA provides Representational State Transfer (REST) and web service interfaces to enable existing applications to take advantage of communication features that involve phone calls and web collaboration.

CEA applications do not require developers to have extensive knowledge of telephony or SIP to implement this type of applications. CEA capabilities deliver call control, notifications, and interactivity, providing the platform with more complex communications.

### 1.5.8 Service Component Architecture

The Service Component Architecture (SCA) programming model supports the principles of service-oriented architecture (SOA) through application flexibility and agility. Service implementation and access method details are moved out of business application logic and into metadata that can be operated on by middleware. As service details or locations change, the application can remain unaffected. SCA helps application developers maximize productivity through focus on solving business problems with application code rather than protocols and locations.

Service compositions can be created by using the following protocols:

- ▶ Plain Object Java Objects (POJOs)
- ▶ EJB 2.1, 3.0, and 3.1
- ▶ OSGi applications
- ▶ Spring components

- ▶ Java servlets
- ▶ JavaScript for Asynchronous JavaScript and XML (Ajax)

## 1.5.9 XML

Since first being standardized in 1999, XML usage in application development environments grew to include the following types of scenarios:

- ▶ A simplified programming model for building presentation flexible web applications using end-to-end declarative XML that can be adapted to various platform types from mobile to desktop
- ▶ Access, search, retrieval, and presentation of data or documents that are stored in native XML
- ▶ Filter for XML data sets, such as web feeds in Java applications
- ▶ XML data transformations for application development scenarios

WebSphere Application Server V8.5 provides XML support to work with web applications that process data by using standard XML technologies such as Extensible Stylesheet Language Transformation (XSLT) 2.0, XML Path Language (XPath) 2.0, and XML Query Language (XQuery) 1.0. You can use XQuery 1.0 to query large amounts of data that is stored in XML outside of a database. All together, this technologies offer the benefit of simplifying application development and improving its performance and reliability

## 1.5.10 WebSphere Application Server Web 2.0 and Mobile Toolkit

The WebSphere Application Server Web 2.0 and Mobile Toolkit simplifies the addition of Asynchronous JavaScript and XML (Ajax) rich desktop and mobile user interfaces and REST web services to Java web applications. Web 2.0 capabilities, such as Ajax and REST, help application developers create more connected and interactive applications, that result in higher customer satisfaction, user productivity, and enhanced decision making. Mobile Ajax components enable developers to create mobile web applications that run on mobile devices, such as smartphones and tablets.

## 1.5.11 Development and testing of applications

WebSphere Application Server V8.5 provides several features for application development and testing.

The IBM WebSphere Application Server Developer Tools for Eclipse Version 8.5 provides a development environment for developing, assembling, and deploying Java EE, OSGi, Web 2.0, and mobile applications, and supports multiple versions of WebSphere Application Server. When combined with Eclipse SDK and Eclipse Web Tools Platform, WebSphere Application Server Developer Tools for Eclipse provides a lightweight environment for developing Java EE applications.

IBM Rational Application Developer for WebSphere Software Version 8.5 provides a development environment for building applications that run on WebSphere Application Server V8.5. This tool supports all Java EE artifacts that are supported by WebSphere Application Server V8.5, such as servlets, JavaServer Pages (JSP), JavaServer Faces (JSF), Enterprise JavaBeans (EJB), Extensible Markup Language (XML), SIP, Portlet, and web services. It also includes integration with the Open Services Gateway initiative (OSGi) programming model. The workbench contains wizards and editors that help in building standards-compliant, business-critical Java EE, Web 2.0, and service-oriented architecture applications.

Code quality tools help teams find and correct problems before they escalate into expensive problems.

For developers not using one of the development environments that are based in WebSphere Application Server, WebSphere Application Server for Developers provides a runtime environment for development and testing. This runtime environment is identical to the production runtime environment on which the applications eventually run. It provides for efficient development and aids in reducing testing effort.

The Liberty profile supports web applications, OSGi applications, and Java Persistence API. Associated services, such as transaction and security, are supported for these application types and for Java Persistence API. It is lightweight, easy to install, and allows for dynamic updates to the runtime environment, making the Liberty profile a good platform for developing and testing web and OSGi applications. This platform is beneficial when you are developing an application to run on the WebSphere Application Server Full profile, because any application that runs on the Liberty profile also runs on the Full profile.

### **1.5.12 Deploying, maintaining, and upgrading applications**

Companies commonly use a build and deployment process while an application moves from development to production. As changes are made to the application, multiple versions are created and deployed.

The WebSphere Application Server V8.5 Intelligent Management functionality makes it possible to store these application versions in the system management repository and then deploy them as needed.

WebSphere Application Server V8.5 includes the monitored (or “dropins”) directory feature, which increases developer productivity by installing, updating, and uninstalling applications automatically as they are moved in and out of the monitored directory. If you add an enterprise archive (EAR file), Java archive (JAR file), web archive (WAR file), or store archive (SAR file) to any monitored directory, the application is installed and started automatically.

The monitored directory works with both the Liberty profile and the Full profile runtime environments for applications that do not require more configuration, such as security role mapping.

In addition to the monitored directory support, you can also deploy or remove an application in the Liberty profile by modifying the server configuration file. When the configuration file is saved, WebSphere Application Server dynamically determines the applications that must be deployed or uninstalled. You can also use the developer tools that are supported by WebSphere Application Server V8.5 to add an application to the server and run it.

Using application edition management, you can validate a new edition of an application in the production environment without affecting users, and you can upgrade applications without incurring outages to users. The application edition management feature provides an application versioning model that supports multiple deployments of the same application in a cell. You can choose which edition to activate on a cluster, enabling you to either roll out an application update or revert to a previous level.

IBM Assembly and Deploy Tools for WebSphere Administration is the application assembly and deployment tool that is shipped with WebSphere Application Server V8.5. With IBM Assembly and Deploy Tools for WebSphere Administration, developers can accomplish key assembly and deployment needs, including editing of deployment artifacts, script development and testing, and application deployment and debugging. This tool is not intended for general application development.

## 1.6 Profiles

WebSphere Application Server runtime environments are built by creating a set of configuration files, named *profiles*, that represent a WebSphere Application Server configuration. The following categories of WebSphere Application Server files are available (Figure 1-8):

- ▶ *Product files* are a set of read-only static files or product binary files that are shared by any instances of the WebSphere Application Server product.
- ▶ *Configuration files (profiles)* are a set of user-customizable data files. This file set includes WebSphere configuration, installed applications, resource adapters, properties, log files, and so on.

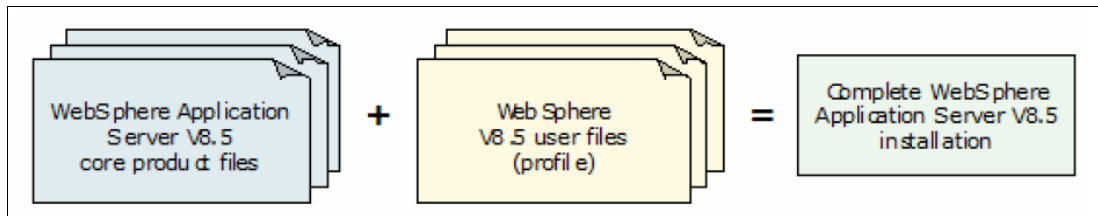


Figure 1-8 Files in WebSphere Application Server

Administration is greatly enhanced when you use profiles instead of multiple product installations. Disk space is saved and updating the product is simplified when you maintain a single set of product core files.

You can use the Customization Toolbox to create separate environments, such as for development or testing, without having to perform a separate product installation for each environment. Different profile templates are available in WebSphere Application Server V8.5 through the Customization Toolbox - Profile Management Tool (PMT), including:

- ▶ **Cell**  
A Cell template contains a federated application server node and a deployment manager.
- ▶ **Deployment manager**  
The Network Deployment profile provides the necessary configuration for starting and managing the deployment manager server.
- ▶ **Default profile (for stand-alone servers)**  
This server default profile provides the necessary configuration file for starting and managing an application server, and all the resources that are required to run enterprise applications.
- ▶ **Administrative Agent**  
This profile is used to create the administrative agent to administer multiple stand-alone application servers.
- ▶ **Default secure proxy**  
This profile is available when you install the DMZ secure proxy server feature.

- ▶ Job manager

This profile is useful for coordinating administrative actions among multiple Deployment Managers, administering multiple stand-alone application servers, asynchronously submitting jobs to start servers, and completing various other tasks.

- ▶ Custom

This profile, also known as *Empty Node* because it has no application server inside, can be *federated* to a deployment manager cell later. It is used to host application servers, clusters, an ODR, or other Java processes.

## 1.7 Workload management

Workload management is the concept of sharing requests for multiple instances of a resource. Workload management techniques are implemented expressly to provide scalability and availability within a system. These techniques allow the system to serve more concurrent requests.

Workload management encompasses the following main concepts:

- ▶ *Load balancing* is the ability to send the requests to alternative instances of a resource. Workload management allows for better use of resources by distributing loads evenly. Components that are overloaded, and therefore a potential bottleneck, can be routed around with workload management algorithms. Workload management techniques also provide higher resiliency by routing requests around failed components to duplicate copies of that resource.
- ▶ *Affinity* is the ability to route concurrent requests to the same component that served the first request. A web client can establish session affinity for requests to be routed to a particular application server.

In WebSphere Application Server, workload management is achieved by sharing requests for one or more application servers, where each server is running a copy of the application. In more complex topologies, workload management is embedded in load balancing technologies that can be used in front of web servers.

The Workload Manager (WLM) for WebSphere Application Server for z/OS works differently from the Workload Manager for distributed platforms. With z/OS, the workload management structure is integrated with the workload management feature of the z/OS MVS™.

### 1.7.1 HTTP servers

An IP sprayer, similar to IBM WebSphere Edge Component Load Balancer or a network appliance, can perform load balancing and workload management functionality for incoming web traffic. There are several routing techniques, including static and dynamic weighting. In static weighting, as the name implies, each HTTP server that is being handled by the IP sprayer has a weight and the IP sprayer routes requests based on this weight. It does not take into account the workload of the HTTP server itself. Dynamic weighting, alternatively, calculates the load of each HTTP server and dynamically routes requests to a server that is not as busy.

In addition, the WebSphere plug-in provides workload management capabilities for applications that are running in an application server.



## 1.7.2 DMZ proxy servers

As with HTTP servers, you can use an IP sprayer component, such as Edge Component Load Balancer, to perform load balancing and workload management functionality for incoming web traffic. In addition, the DMZ proxy server provides workload management capabilities for applications that are running in an application server.

## 1.7.3 Clusters

A *cluster* is a collection of servers that is managed together. With clusters, enterprise applications can scale beyond the amount of throughput that can be achieved with a single application server. Also, enterprise applications can be highly available because requests are automatically routed to the running servers if there is a failure. The servers that are members of a cluster can be on different host machines. A cell can include no clusters, one cluster, or multiple clusters.

An *application server cluster* (Figure 1-9) is a logical collection of application server processes that provides workload balancing and high availability. It is a grouping of application servers that run an identical set of applications that are managed so that they behave as a single application server (parallel processing). WebSphere Application Server Network Deployment or WebSphere Application Server for z/OS is required for clustering.

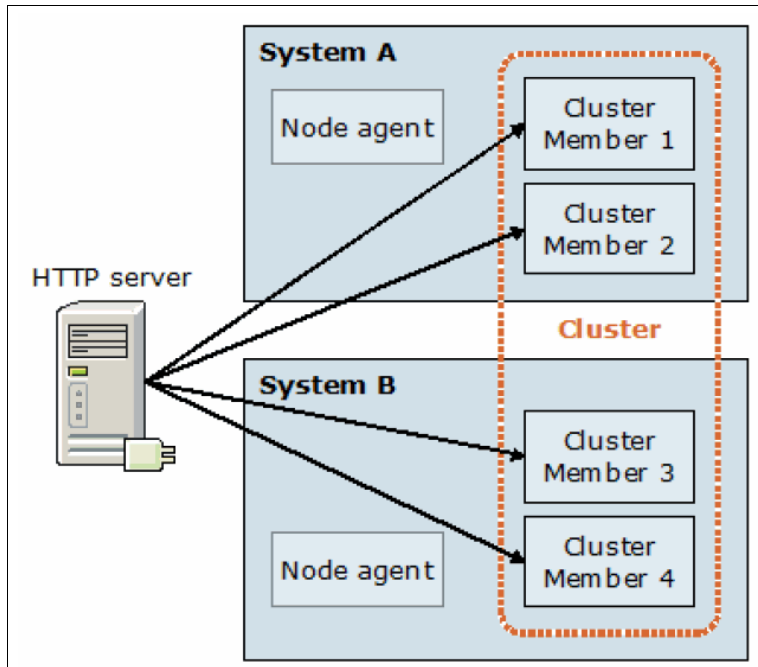


Figure 1-9 Application server cluster

WebSphere Application Server V8.5 provides integrated support for dynamic clusters. A *dynamic cluster* (Figure 1-10) is an application deployment target that can expand and contract depending on the workload in the environment. Dynamic clusters work with autonomic managers, including the application placement controller and the dynamic workload manager to maximize the usage of computing resources. A dynamic cluster uses weights and workload management. This management is used to balance the workloads of its cluster members dynamically, and is based on performance information that is collected from the cluster members.

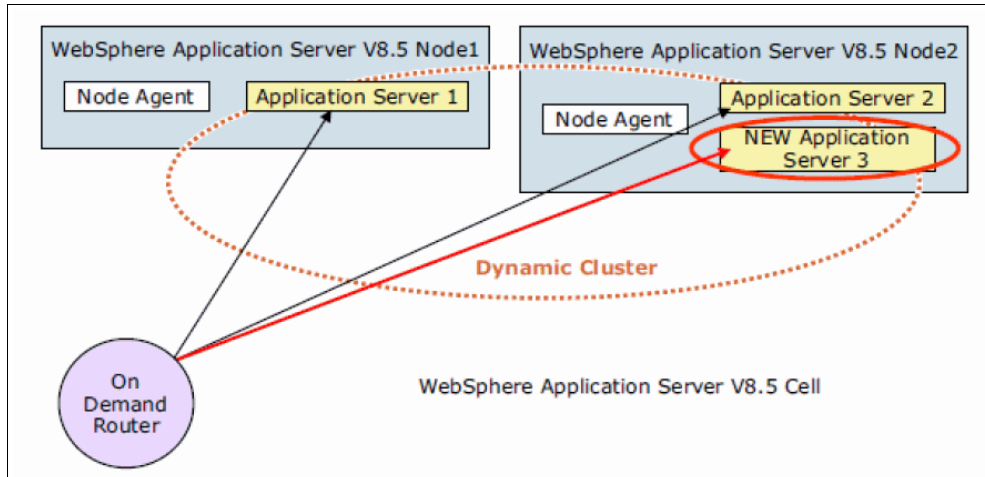


Figure 1-10 Dynamic cluster

Dynamic workload management is a feature of the ODR that applies the same principles as workload management, such as routing based on a weight system, which establishes a prioritized routing system. With dynamic workload management, the system can dynamically modify the weights to stay current with the business goals, compared to manually setting static weights in workload management. It also balances requests across the available nodes to regulate response times.

## 1.8 High availability

High availability is also known as *resiliency*. High availability is the ability of a system to tolerate an number of failures and remain operational. It is achieved by adding redundancy in the infrastructure to support failures. It is critical that your infrastructure continues to respond to client requests regardless of the circumstances and that you remove any single points of failure. Planning for a highly available system takes planning across all components of your infrastructure, because the overall infrastructure is only available when all of the components are available. As part of the planning, you must define the level of high availability that is needed in the infrastructure.

WebSphere Application Server Network Deployment uses a high availability manager (HA manager) to address these single points of failure within the WebSphere Application Server environment. An HA manager instance runs on every application server, proxy server, node agent, and deployment manager in a cell. A cell can be divided into multiple high availability domains that are known as *core groups* (Figure 1-11).

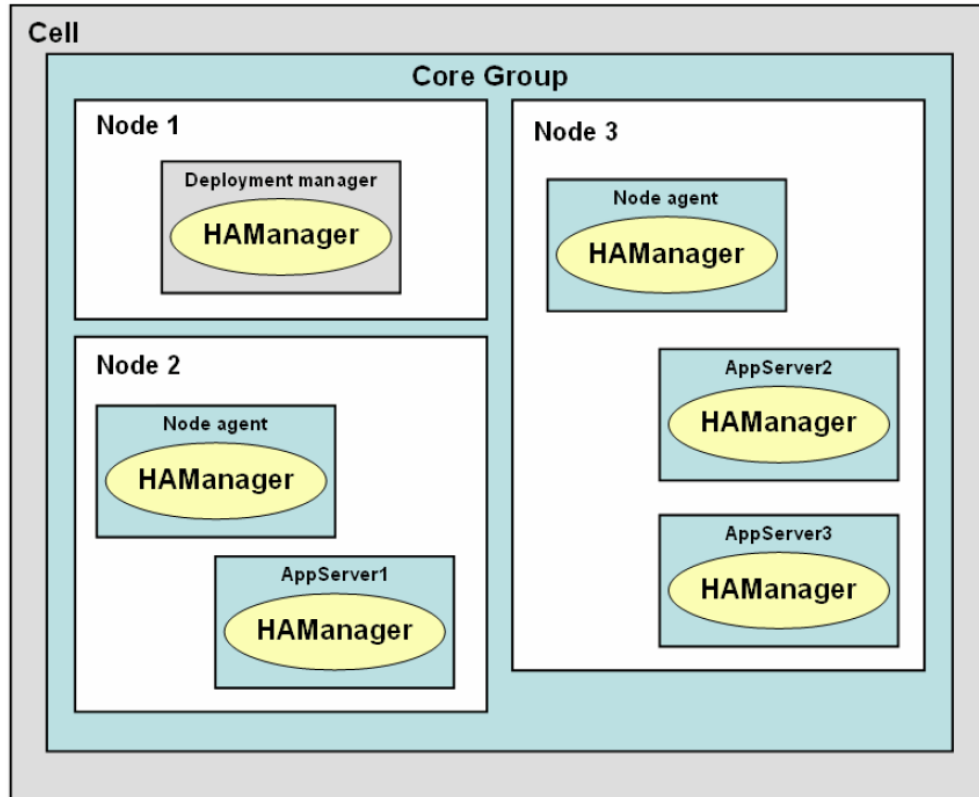


Figure 1-11 Conceptual diagram of a core group

Each HA manager instance establishes network connectivity with all other high availability manager instances in the same core group, using a specialized, dedicated, and configurable transport channel. The transport channel provides mechanisms that allow the HA manager instance to detect when other members of the core group start, stop, or fail.

Within a core group, HA manager instances are elected to coordinate high availability activities. An instance that is elected is known as a *core group coordinator*. The coordinator is highly available, so that if a process that is serving as a coordinator stops or fails, another instance is elected to assume the coordinator role, without loss of continuity.

The HA manager provides a specialized messaging mechanism that enables processes to exchange information about their current state. Each process sends or posts information that is related to its current state, and can register to be notified when the state of the other processes changes.

The Data Replication Service (DRS) that is provided with WebSphere Application Server V8.5 is used to replicate stateful EJB sessions, HTTP session data, and dynamic cache information among cluster members. When DRS is configured for memory-to-memory replication, the transport channels that are defined for the HA managers are used to pass this data among the cluster members.

The ODR is an intelligent Java based HTTP proxy server and SIP proxy server that is built on the WebSphere run time that can manage the flow of requests into both WebSphere and non WebSphere environments. The ODR is asynchronous, high performing, scalable, and can be clustered for high availability.

You can use the ODR to set up a highly available deployment manager using a hot-standby model for availability. For example, you can have two or more deployment managers, where one deployment manager is active (known as the *primary* deployment manager) and one or more deployment managers are backup managers in standby mode. The primary deployment manager hosts the cell's administrative function, and if the active manager is stopped or fails, a standby manager takes over and is designated the new active deployment manager.

Although the deployment manager process is not considered a single point of failure (SPOF) for applications and application server functionality, using a highly available deployment manager eliminates the deployment manager as a SPOF for cell administration. This type of high availability is especially important in environments that rely on automated operations, including application deployment and server monitoring.

The WebSphere Application Server V8.5 default messaging engine, the service integration bus (bus), provides high availability to the messaging system process. By creating the messaging engine within the cluster, it can fail over to any other server in the cluster. To achieve a seamless failover, the queue information and message data must be stored in a shared location that is accessible by all members of the cluster. This shared location can be an external database or shared disk environment.

High availability also means ensuring that you have adequate resources to meet the changing demands. By using Intelligent Management capabilities in WebSphere Application Server V8.5, applications are mapped to dynamic clusters that are spread throughout hardware pools. Each node in the dynamic cluster can run one or more instances of an application server that cluster's applications. By deploying applications to dynamic clusters, requests to the application can be managed autonomically. The dynamic cluster can expand and contract with application server virtualization. By deploying applications to dynamic clusters, you enable application virtualization.

## 1.9 Intelligent management

Intelligent management is the integration of WebSphere Virtual Enterprise into WebSphere Application Server Network Deployment V8.5. Intelligent Management features extend the quality of service that is provided by the middleware environment. Total cost of ownership (TCO) is decreased through server consolidation and less administrative effort, and you experience lower response times and increased availability.

With Intelligent Management, virtualization, autonomic computing, and cloud computing are integrated into the single architectural model of WebSphere Application Server V8.5:

- ▶ Virtualization

By configuring application infrastructure virtualization in WebSphere Application Server with the Intelligent Management functionality, resources are pooled. This pool accommodates the fluctuations of workload in the environment, increasing the quality of service. Workloads are then dynamically placed and spread across a pool of application server resources, which allows the infrastructure to adapt and respond to business needs.

The static relationships of an application with the server to which it is deployed is replaced with a dynamic relationship. Instead of statically binding applications to servers or clusters, you deploy applications to dynamic clusters.

You can combine the advantages that infrastructure virtualization provides with the advantages of hardware virtualization by using both in the same infrastructure.

- ▶ **Autonomic computing**

Autonomic computing refers to the self-managing characteristics of distributed computing resources that adapt to unpredictable changes while hiding intrinsic complexity from operators and users. Using Intelligent Management in WebSphere Application Server V8.5, you can create a self-managing environment that serves applications. It can help you overcome the complexity of systems management and reduce the barrier that complexity poses to further growth. An autonomic system makes decisions on its own, using high-level policies. It constantly checks and optimizes the status of the system, and automatically adapts it to changing conditions.

- ▶ **Cloud computing**

Virtualization and autonomic computing are steps toward cloud computing. By building a virtualization foundation, you put the secure, scalable, and efficient system in place in which to build a cloud.

Intelligent management includes the primary features of intelligent routing, dynamic operations, performance management, health management, and application edition management, which are described in the following sections.

## **1.9.1 Intelligent routing and dynamic operations**

Intelligent routing improves business results by ensuring priority is given to business critical applications. The ODR in WebSphere Application Server V8.5 is used to intelligently route requests. The ODR can momentarily queue requests for less important applications so that requests from more important applications are handled more quickly. Dynamic operations allow an application environment to scale as needed by virtualizing WebSphere resources and by using a goals-directed infrastructure. To take advantage of dynamic operations, a dynamic cluster is used.

*Dynamic cluster* is a server cluster that uses weights and workload management to balance the workloads of its cluster members dynamically. It balances based on performance information that is collected from the cluster members. Dynamic clusters consist of a number of servers that can stop or start in response to changing workload.

These clusters are application deployment targets that can expand and contract, depending on the workload in the environment. After you deploy applications to dynamic clusters, the placement of the applications is determined by the operational policies that you define. Autonomic managers control the placement of the server instances and how workload is routed to each application. If workload increases for a specific application, the number of server instances for the dynamic cluster that is hosting the application can increase. The application can also use available resources from other applications that are not experiencing increased workload.

Figure 1-12 illustrates how the workload increases for a specific application. The number of server instances for the dynamic cluster that is hosting the application can increase by using available resources from other applications. In this example, *New Application Server 3* contributes bandwidth to satisfy higher workload requests. The ODR manages this dynamic cluster growth and sends requests to the new server.

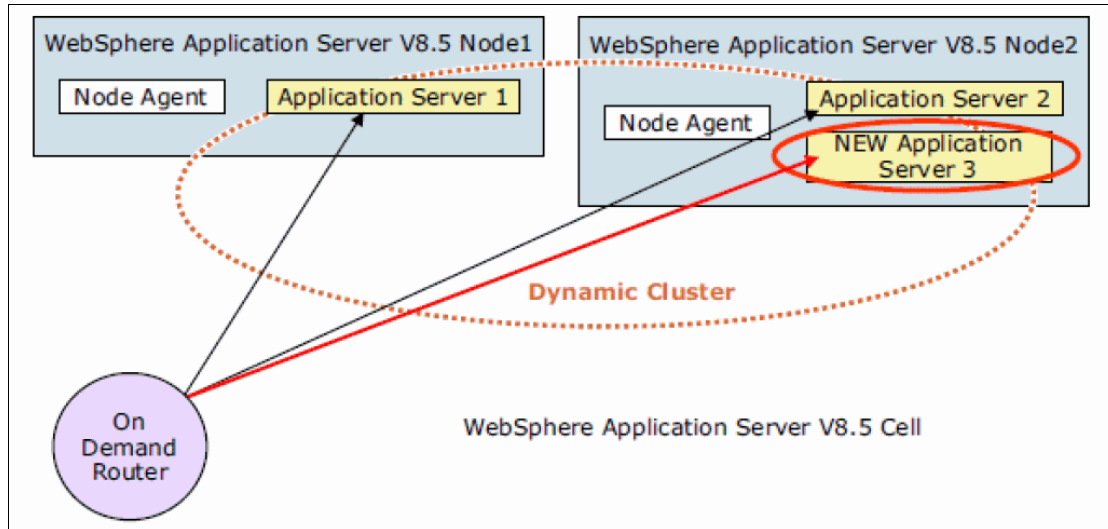


Figure 1-12 Dynamic cluster

The *ODR* is an intelligent Java based HTTP proxy server and Session Initiation Protocol (SIP) proxy server that is built on the WebSphere run time. It is responsible for managing the flow of requests into the WebSphere environment and non WebSphere environment. The ODR, similar to the web server plug-in for WebSphere Application Server, uses session affinity to route work requests. The ODR is asynchronous, high performance, and scalable. It can be clustered for high availability.

The ODR can be used to set up a *highly available deployment manager*. It can then route to an active deployment manager and possibly a hot-standby deployment manager.

*Dynamic workload management* is a feature of the ODR. It applies the same principles as Workload Manager (WLM), such as routing based on a weight system, which establishes a prioritized routing system. The dynamic workload controller autonomously sets the routing weights in WLM. With workload management, you manually set static weights in the administrative console. The system can dynamically modify these weights to stay current with the business goals.

## 1.9.2 Application edition management

With the Intelligent Management functionality, you can store multiple application versions in the *system management repository* and deploy them as needed. While source control systems typically store the source code, the system management repository stores the compiled code.

Using application edition management, you can validate a new edition of an application in your production environment. This process does not affect users, so you can upgrade your applications without causing outages for your users. You can also run multiple editions of a single application concurrently, directing different users to different editions.

The application edition management feature also provides an application versioning model that supports multiple deployments of the same application in a cell. You can choose which edition to activate on a cluster, so you can roll out an application update or revert to a previous level.

## 1.10 Health management

Another important key to business success is the ability to autonomically manage the health of your servers and quickly respond to potential issues, preferably before they impact your environment. The Intelligent Management capabilities in WebSphere Application Server V8.5 provides several health management features to monitor the status of application servers to sense and respond to problem areas before an outage occurs, similar to preventive medicine. You can manage the health of an application serving environment with a policy-driven approach that enables specific actions to occur when monitored criteria is met. For example, when memory usage exceeds a percentage of the heap size for a specified time, health actions can run to correct the situation.

Health monitoring can help with unexpected issues and with unanticipated problems in your environment. It can help you bypass problems that would otherwise disrupt operations and affect the performance and availability of your applications. With WebSphere Application Server V8.5, you can automatically detect and handle application health problems without requiring administrator intervention, or in a supervisor mode that requires administrator approval before an autonomic action is performed. Health monitoring intelligently handles health issues in a way that maintains continuous availability and allows you to configure how the system should handle individual applications and requests.

You can have the system monitor for various conditions, such as high response timeout rates, excessive response times, or high memory usage, and proactively run certain actions in an attempt to correct these issues.

## 1.11 Administration and problem determination

An enterprise must be able to efficiently and effectively manage the WebSphere Application Server environment. The combination of administrative tools that you employ depend on the size and complexity of the runtime environment. Automation and scripting can also be valuable in situations where the same tasks must be performed multiple times in a consistent and repeatable fashion. In addition, after your system is operational, you must be able to determine problems when something goes wrong.

### 1.11.1 Administration

In the WebSphere Application Server Express and Base packages, you can administer stand-alone server instances individually and administer multiple stand-alone server instances on a single system using an administrative agent. In WebSphere Application Server Network Deployment, you can administer an entire cell of application servers using a deployment manager.

WebSphere Application Server V8.5 provides the following administrative tools that you can use to configure and manage a runtime environment:

▶ WebSphere scripting client (**wsadmin**)

The **wsadmin** client provides a non-graphical scripting and interactive interface that you can use to administer WebSphere Application Server from a command-line prompt. By connecting the client to an application server instance, you can run commands using either the Jython or Jacl scripting languages.

▶ Administrative console

The administrative console is a browser-based client that allows administrators to monitor, update, and configure the WebSphere Application Server environment, except for the Liberty profile. Unfederated application servers, administrative agents, deployment managers, and job managers can have their own administrative consoles.

For actions run in the administrative console, you can use command assistance to view the **wsadmin** scripting commands in Jython for the last action that was run. This assistance provides a way to develop automation using **wsadmin** scripting by first stepping through the tasks that you need in the administrative console.

▶ Command-line utilities

Administrative utilities can help to manage the environment from the command line. These utilities include the ability to perform common administrative tasks, such as starting and stopping application server instances and backing up the configuration. The utilities are run on and act against the local server or node, including the deployment manager.

▶ WebSphere Customization Toolbox

The WebSphere Customization Toolbox includes the following tools for customizing various parts of the WebSphere Application Server environment:

- You can use the Web Server Plug-ins Configuration Tool to configure web server plug-ins for any operating system on which the WebSphere Customization Toolbox can be installed.
- You can use the WebSphere Customization Toolbox command-line utility to start the command-line version of the Plug-ins Configuration Tool (PCT).
- The Profile Management Tool (z/OS only) can be started on an Intel based Windows or Linux operating system to generate jobs and instructions for creating profiles for WebSphere Application Server on z/OS systems.
- The z/OS Migration Management Tool generates definitions for migrating WebSphere Application Server for z/OS profiles from an Intel based Windows or Linux operating system.

▶ Job manager

You can use the job manager to manage multiple WebSphere Application Server domains (multiple deployment managers and administrative agents) through a single administration interface. It also provides a centralized interface for asynchronous job submissions.

In WebSphere Application Server V8.5, you can use the job manager to package the Liberty profile runtime environments, configurations, and applications to distribute and deploy a Liberty profile server and applications and to start embedded profile packages.

▶ Centralized installation manager

You can use the centralized installation manager to consolidate and simplify the required steps in installing and applying maintenance. With the centralized installation manager, you can install, update, uninstall, and apply maintenance to WebSphere Application Server remotely.



You access the centralized installation manager functions through the job manager or deployment manager. Because the functions are implemented as jobs, the process also supports job scheduling. Using the centralized installation manager with the job manager, you can manage multiple product offerings in an agentless manner for multiple cells.

- ▶ Administrative applications

With WebSphere Application Server Version V8.5, you can develop custom Java applications using Java Management Extensions (JMX) to perform any of the administrative features of the WebSphere Application Server administrative tools. For example, you can prepare, install, uninstall, edit, and update applications through programming.

## 1.11.2 Problem determination

WebSphere Application Server V8.5 includes the following modes of logging for problem determination:

- ▶ Basic logging and tracing is the default mode.

WebSphere Application Server writes system messages into several general-purpose logs. These logs are plain-text files, and you can view them from the administrative console (including logs for remote systems) or from a text editor on the system where the log files are stored.

The `SystemOut.log` and `SystemErr.log` files contain system output and error messages. For processes that are running native code, `native_stdout.log` and `native_stderr.log` are used. The IBM service log (`activity.log`) is a special log file that is written in a binary format. You cannot view the log file directly with a text editor, but you can view the log file using one of the following methods:

- The Log Analyzer tool provides interactive viewing and analysis capabilities that help in identifying problems.
- You can use the Showlog tool to convert the contents of the service log to a text format.

- ▶ High Performance Extensible Logging (HPEL) stores log and trace data in a proprietary binary format. Log and trace performance is enhanced with HPEL because of the following factors:

- Log and trace events are stored only in one place and not redundantly in several locations. Log events, `System.out`, and `System.err` information is stored in the log data repository. Trace events are stored in the trace data repository.
- Repositories are not shared across processes. Each server process has its own repositories and text log file. The server environment, therefore, does not need to synchronize with other processes when it writes data.
- Data is not formatted until it is viewed. Log and trace data is stored in a proprietary binary format in the repositories rather than being formatted at run time. The log and trace data is not formatted until it is viewed by using the LogViewer tool.
- Log and trace data is buffered before it is written to disk.

The HPEL LogViewer is a simple command-line tool for HPEL users to work with the log and trace data repositories. The LogViewer provides filtering and formatting options that make finding important content in the log and trace data repositories easy. For example, a user can filter any errors or warnings and then filter all log and trace entries that occurred within 10 seconds of a key error message. This filtering can be done on the same thread.

WebSphere Application Server V8.5 provides the **server dump** command for problem diagnosis for a Liberty profile server. The result file that is obtained from this command contains server configuration, log information, and details of the deployed applications in the work area directory.

Usually, a dump of a running Liberty profile server includes the following information:

- ▶ State of each OSGi bundle in the server
- ▶ Wiring information for each OSGi bundle in the server
- ▶ A component list that is managed by the Service Component Runtime (SCR)
- ▶ Detailed information about each component from SCR

## 1.12 Messaging

Generically, the term *messaging* describes communication or the exchange of information between two or more interested parties. Messaging is a facility for exchanging data between applications and clients of different types. It is also an excellent tool for communication between heterogeneous platforms.

WebSphere Application Server V8.5 implements a powerful and flexible messaging platform within the WebSphere Application Server environment that is called the *service integration bus* (SIBus). WebSphere Application Server applications run asynchronous messaging services, using the Java Messaging Service (JMS) application programming interface (API), to interface with a messaging provider. The messaging provider can be the WebSphere MQ messaging provider, the default messaging provider (bus), or a third-party messaging provider.

Applications can use one of the following styles of asynchronous messaging:

- ▶ *Point-to-point* applications typically use queues to pass messages to each other. An application sends a message to another application by identifying a destination queue. The underlying messaging and queuing system receives the message from the sending application and routes the message to its destination queue, where the receiving application retrieves it.
- ▶ In *publish/subscribe* messaging, the subscriber is the consumer of the information and specifies the topics of interest by submitting a subscription request. The publisher supplies information in the form of messages, which contain both a topic and the actual information. A message broker sits between the subscriber and the publisher, and as it receives published messages, it forwards those messages to the subscribers who requested messages in the topic that is given in each individual message.

## 1.13 Service integration

*Service integration* is a set of technologies that provides asynchronous messaging services. In asynchronous messaging, producing applications do not send messages directly to consuming applications. Instead, producing applications send messages to *destinations*. Consuming applications then receive messages from these destinations. A producing application can send a message and continue processing without waiting until a consuming application receives the message.

Service integration bus capabilities are fully integrated into WebSphere Application Server V8.5, enabling it to take advantage of WebSphere security, administration, performance monitoring, trace capabilities, and problem determination tools. A service integration bus (SIBus) is a group of one or more *bus members* in a WebSphere Application Server cell that cooperate to provide asynchronous messaging services. A cell requires only one bus, but a cell can contain any number of buses. A messaging engine is a component that is responsible for processing messages, sending and receiving requests, and hosting destinations.

A destination is defined within a bus and represents a logical address to which applications can attach as message producers, consumers, or both. Destinations are associated with a messaging engine by using a *message point*.

Service integration has the following types of bus destinations, each with a different purpose:

- ▶ A *queue* destination is used for point-to-point messaging.
- ▶ A *topic space* destination is used for publish/subscribe messaging.
- ▶ A *foreign destination* is a destination that is defined on another bus (called a *foreign bus*) and that is also used in point-to-point messaging.
- ▶ An *alias* destination is an alternative name for either a queue destination or a topic space destination.

## 1.14 Security

WebSphere Application Server provides a security infrastructure and mechanisms to protect sensitive resources and to address enterprise end-to-end security requirements. From a broad perspective, a WebSphere security service runs locally in each process (deployment manager, node agent, and application server).

This architecture (Figure 1-13) distributes the security workload so that one process does not act as a bottleneck for the entire environment. If a security service failure occurs, then only a single process is affected. The authentication mechanism and user registry are separated.

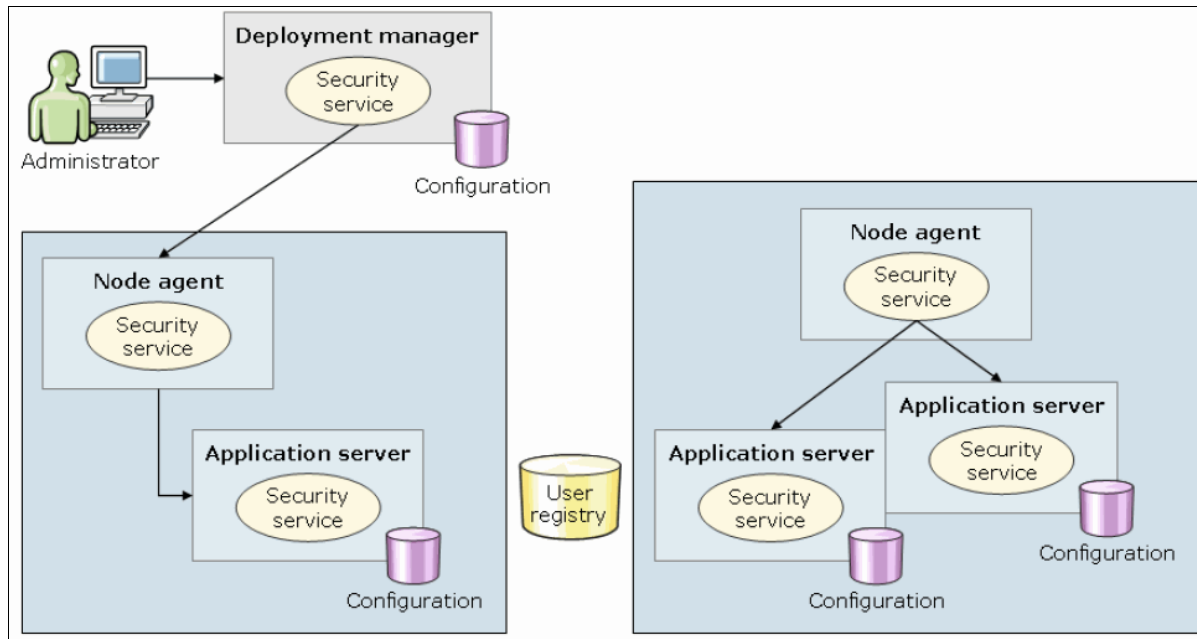


Figure 1-13 Overview of WebSphere security service

You can secure the environment at various levels, from the physical security of the location where the server hardware is, to the security within the application itself.

WebSphere Application Server V8.5 security includes the following levels:

- ▶ Java virtual machine (JVM) security

The JVM provides a set of standards-based security services for Java applications and an installation layer between Java applications and the various services within the operating system. It provides an isolated environment to the Java application that is running in it, which in this case is WebSphere Application Server. In addition, the JVM protects memory from unrestricted access, creates exceptions when errors occur within a thread, and defines array types.

- ▶ Java 2 security

The Java 2 security model offers access control to system resources, including file system, system property, socket connection, threading, class loading, and so on. Application code must explicitly grant the required permission to access a protected resource. Java 2 uses security policy files, which can control the access to the resources by applications. A WebSphere Application Server application has its own policy files, so that it can use files and directories on the host operating system.

- ▶ Java EE security API

The Java EE standard API describes methods with which the application can obtain the logged in user's name and role membership. WebSphere Application Server never returns the password of any user using the API methods.

The Java EE security policy describes how application resources are accessed. The developer, when creating the application, has no information about real users of the application. Instead, the developer defines *user roles*, and during development the user roles are mapped to access rights. The rule set is stored in the descriptor files of the application. Then, when the application is deployed, the deployer is responsible for mapping users and groups to the security roles.

- ▶ CSiv2 security

CSiv2 is an IIOP-based, three-tiered, security protocol that is developed by the Object Management Group (OMG). This protocol provides message protection, interoperable authentication, and delegation. Any calls that are made among secure Object Request Brokers (ORBs) are started over the CSiv2 security protocol, which sets up the security context and the necessary quality of protection. After the session is established, the call is passed up to the enterprise bean layer.

- ▶ WebSphere security

WebSphere security enforces security policies and services with regard to access to its resources. It covers a wide range of features. It begins with the administrator user management in the administrative console, controlling which administrative user is allowed to do what on the administrative console. Administrative security is enabled by default.

The WebSphere Application Server security infrastructure provides a set of services. These services are integrated with the underlying operating system, the runtime environment, and other servers and server components. The WebSphere Application Server security infrastructure includes the following key security areas:

- ▶ Authentication
- ▶ Authorization
- ▶ Key and certificate management
- ▶ Auditing
- ▶ Flexible configuration option

WebSphere Application Server V8.5 provides the capability to use security domains. Each security domain can have its own, separately configured Virtual Member Manager (VMM) instance. WebSphere Application Server V8.5 provides the following options and combinations to select the best user registry setting for an application environment:

- ▶ Security domains

WebSphere *security domains* provide the flexibility to use several security configurations within a single WebSphere Application Server cell. With security domains, you can configure several security attributes, such as the user registry, for various applications in the same cell. The global security configuration applies to all administrative functions, naming resources, and Mbeans, and is the default security configuration for user applications. You must define a global security configuration before you can create the security domains. If you do not configure security domains, the applications use information from the global security configuration.

When you create a security domain and associate it with a scope, only the user applications in that scope use the security attributes that are defined in the security domain. The administrative applications and the naming operations in that scope use the global security configuration. Each security domain must be associated with a scope (cell, or specific clusters, servers, and service integration buses) where it is applied.

► User registries

Information about users and groups are in a *user registry*. In WebSphere Application Server, a user registry authenticates a user. It contains information about users and groups so that security-related functions, including authentication and authorization, can be performed. Although WebSphere Application Server supports several types of user registries, only one user registry can be active in a certain scope.

WebSphere Application Server supports the following types of user registries:

- Local operating system
- Stand-alone Lightweight Directory Access Protocol (LDAP)
- Federated repository (a combination of a file-based registry and one or more LDAP servers in a single realm)
- Custom registry

## 1.15 Liberty profile

The Liberty profile is a dynamic and composable profile of WebSphere Application Server V8.5 that enables WebSphere Application Server to provision only the features that are required by the application (or set of applications) that are deployed to the server. For example, if an application requires just a servlet engine, a Liberty profile can be configured to start only the WebSphere Application Server kernel, the HTTP transport, and the web container. Thus, the Liberty profile is quick to start and has a small footprint.

The Liberty profile provides a simplified and lightweight development and application-serving environment that is optimized for developer and operational productivity. This profile is intended for use as a development environment and as a production environment for running web applications that do not require a full Java Platform, Enterprise Edition (Java EE) stack. The Liberty profile provides enterprise qualities of service, such as security and transaction integrity.

The Liberty profile includes the following key features:

- A dynamic and flexible run time to load only what the application needs
- A quick startup time (under five seconds with simple web applications)
- A simplified configuration that uses a single configuration file or modular configuration
- Support for deploying applications that are developed in the Liberty profile to run in the full profile
- Support of web applications, OSGi applications, and Java Persistence API
- Support for LDAP registry
- Ability to deploy an application and configured server as a package
- Managed and centralized deployment to many nodes of a packaged application and server
- Availability of WebSphere Application Server Developer Tools as Eclipse plug-ins for broad tooling support
- Support for z/OS platform native features like System Authorization Facility (SAF), Resource Recovery Service (RRS), and z/OS workload management (WLM)

The Liberty profile provides a development and a test environment and a production environment on all WebSphere Application Server V8.5 supported platforms. Additionally, the Liberty profile provides a development environment on the Macintosh operating system.

### 1.15.1 Liberty profile architecture

As shown in Figure 1-14, the Liberty profile is built on OSGi technologies. The server process runs as OSGi bundles and consists of a single Java virtual machine (JVM), the Liberty profile kernel, and any number of optional features.

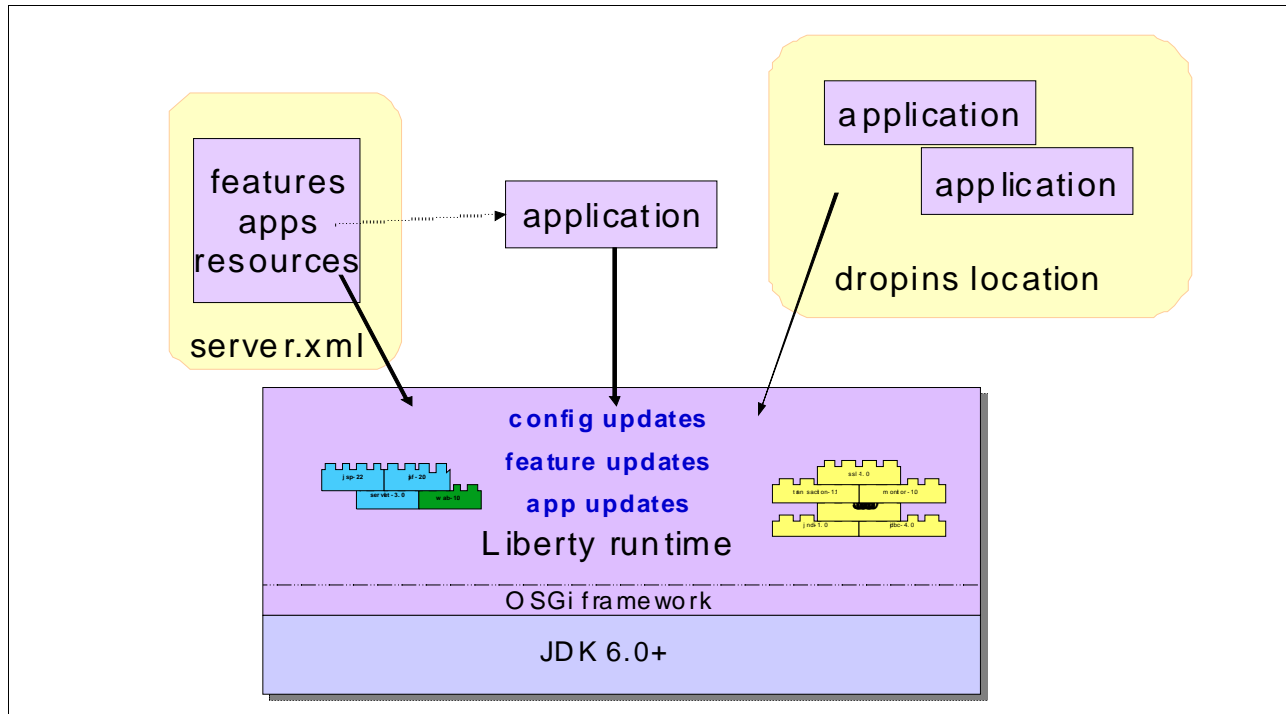


Figure 1-14 Liberty profile architecture

A functional server is produced by starting the runtime environment with a configuration that includes a list of features that are to be used. *Features* are the units of capability, by which the runtime environment is defined and controlled.

By default, a server runs no features. You can use the feature manager to add the features that are needed. The feature manager maps each feature name to a list of bundles that provide the feature. When a feature configuration is changed, the feature manager recalculates the list of required bundles, stops and uninstalls those bundles that are no longer needed, and then installs and starts any additions. All features are designed to cope with other features that are added or removed dynamically.

### 1.15.2 Liberty profile configuration

A Liberty profile server configuration consists of a `server.xml` and an optional `bootstrap.properties` file and any (optional) files that are included by these files. The `server.xml` file is the primary configuration file for the server, and the file that users work with the most. The `bootstrap.properties` file specifies properties that need to be available before the main configuration is processed and these properties are kept to a minimum.

In a Liberty profile configuration, the features of the profile provide the configuration default values. Thus, user-specified configuration is kept to a minimum. Any property can be overridden in a user-specific server configuration, and any changes that are made to the configuration are dynamically injected into the contributing feature immediately. There is no need to restart the server.

### 1.15.3 Liberty profile administration

The Liberty profile runtime environment is designed for easy administration and operates from a set of built-in configuration defaults. There is no administrative console for the Liberty profile. However, administrators and developers can use the Liberty profile developer tools or a text editor to edit the configuration files.

The include functionality for the `server.xml` file can be useful in managing multiple Liberty profile environments. The `server.xml` file can point to (include) one or more local or remote XML files. When you change values in the included XML file, that change takes effect in all of the Liberty profile runtime environments that include that remote XML file. For example, if an application and its data sources are defined in thousands of Liberty profile runtime environments, using the remote include functionality, an administrator can change one value for a data source, and have that change reflected in all of the runtime environments without changing each `server.xml` file individually. Included files that are on the local file system are monitored for changes while the server is running, so those changes can be applied dynamically to the running server. Changes to remote included files require a server restart to take effect.

A `server.env` file can be used for specifying environment variables, and a `jvm.options` file can be used for customizing JVM options.

### 1.15.4 Liberty profile application development and deployment

A Liberty profile supports web applications, OSGi applications, and Java Persistence API. Associated services, such as transaction and security, are supported for these application types. With its lightweight and easy installation, and quick to use features, the Liberty profile provides a convenient and capable platform for developing and testing web and OSGi applications, and the Java Persistence API. This platform is beneficial when you are developing an application to run on the WebSphere Application Server full profile because any application that runs on the Liberty profile also runs on the full profile.

The Liberty profile provides the following options to deploy an application:

- ▶ Dropping the application into a previously defined “dropins” directory  
You can use the “dropins” directory for applications that do not require additional configuration, such as security role mapping.
- ▶ Adding an application entry to the server configuration file  
For applications that are not in a “dropins” directory, you can specify the location of the application as an entry in the server configuration file. The location of the server configuration file can be either a file system path or a URL.

You can use the developer tools that are supported by WebSphere Application Server V8.5 to add an application to the server and run it. For more information, see 9.3.2, “Installing the developer tools” on page 277.





## Part 2

# Migrating methodology and common issues for migration

In this part, we describe the migrating methodology and common issues for migration. These topics are applicable to migrations from any Java Enterprise Edition (EE) platform to WebSphere Application Server V8.5, and are not tied into a specific product.





## Migration strategy and planning

This chapter provides relevant information that you can use to build a migration process plan. The information in this chapter is not intended to be a final methodology, but to help you organize the migration process, address people about certain tasks, know where you need help or not, and decide how to go through the project.

It provides guidance for you to start the migration and build a project plan and a few artifacts that can be used as input to effect a successful migration.

This chapter describes the following topics:

- ▶ Migration process overview
- ▶ Migration assessment
- ▶ Migration planning
- ▶ Migration implementation
- ▶ Post deployment
- ▶ Summary

## 2.1 Migration process overview

There are several factors that influence the decision to perform a migration, including business decisions and technical assessments that lead companies to adapt in a constantly changing and challenging IT environment.

Migration of Java EE applications is not just a change from one application server to another or just changing source code. Migration is a process that involves many phases and people from various areas and with various skills. When you migrate to a new application server, you are likely doing it because you want to take advantage of new capabilities, improve the way that you manage your applications, refactor your code, improve performance, or decrease downtime.

Although every migration is complex, the best approach is to follow a formal process that is aimed at evaluating an application development environment, source code, design consideration, and so on, and make a recommendation about how an application is migrated to a WebSphere environment. You should make these changes in an ordered environment, avoiding issues that can be prevented by appropriately planning and implementing strategies to execute the plan.

Figure 2-1 shows the migration process overview.

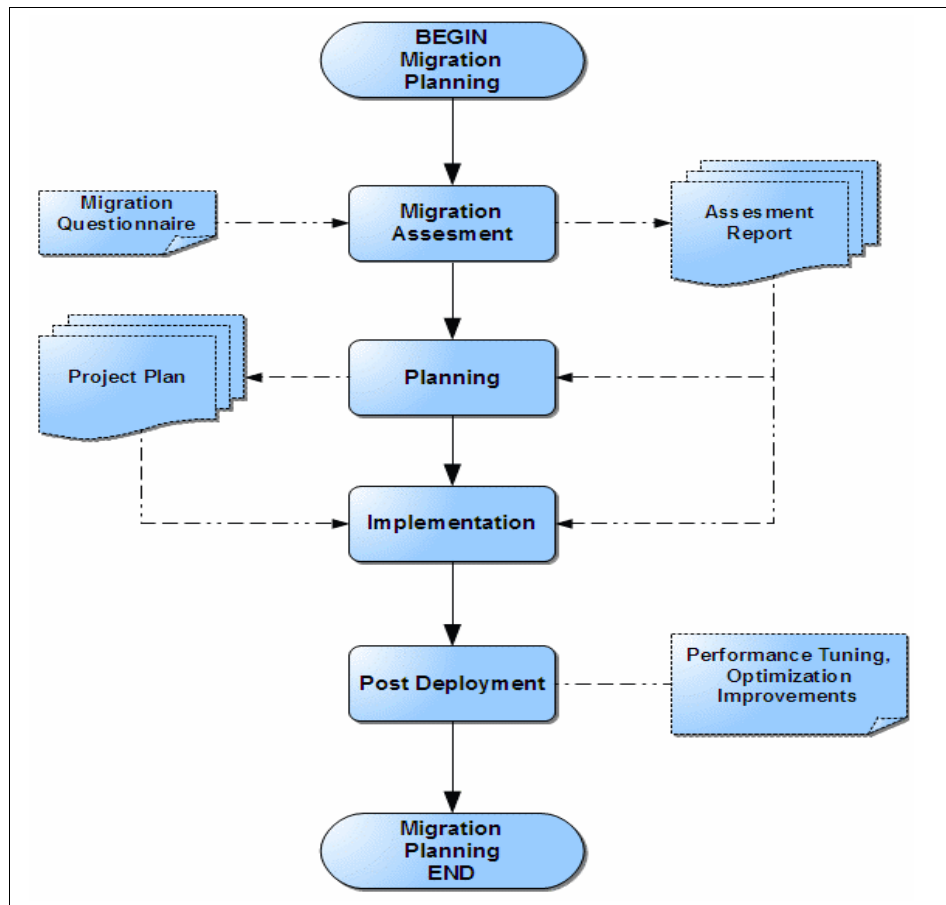


Figure 2-1 Migration process overview

## 2.1.1 Types of migrations

There are many approaches to take when you decide how the migration occurs. This chapter identifies three approaches that cover most types of application migrations alternatives. It is a case-by-case decision on which approach to adopt. Runtime migration is done regardless of type. In each of these approaches, a migration assessment might be done. (Migration assessment is described in 2.2, “Migration assessment” on page 42).

Depending on your scenario, it might be possible that you decide to create a program to manage projects.

### Sample migration

A sample migration is defined by a migration of a single application that is representative of the other applications. A representative application is one that has architectural and implementation similarities with most of the applications that are migrated within the enterprise.

This process can be repeated for more applications after the first one is complete. This migration is the most common type of application migration because it lays a foundation for future migrations that are based on challenges that are encountered in migrating the sample application. You can also determine how quickly a migration can be done and highlight all the problems that are faced so that you can build a concise plan.

### Partial migration

Partial migration consists of migrating multiple or a few applications at a time in a phased approach where the schedule can cover them until the project is completed. It is a good choice when you have teams that own applications in a shared environment or when one team owns multiple applications, which makes it impossible to migrate every application at the same time because of time conflicts.

Applications that are written based on that same architecture and modularized can also be used with this type of migration (for example, moving user interface modules before business components or vice versa).

### Full migration

Full migration is obvious, but has its own set of characteristics. When you decide to go with this option, either you have a dedicated environment for an application or you must migrate the entire environment because of certain reasons, such as end of support, impossible interoperability of certain applications, and other possible issues.

## 2.1.2 Important aspects and considerations

This section describes the considerations that are involved with a Java EE applications migration. It assumes that migrations are done to get benefits from newer versions of code, compilers, new feature availability, runtime administration, and performance improvements.

It is important to understand the challenges that you are facing during migration. You can avoid many issues by making informed decisions. Every migration is different. It can vary in terms of complexity, sizing, and other factors. To help you in this process, here is a list of considerations we discovered with our migrations experiences and when writing this book:

- ▶ It is impossible to create a migration roadmap that works for all environments.
- ▶ Keep the migration as simple as you can, avoiding multiple changes at once.
- ▶ Do not forget environments other than production, such as development, QA, and testing.

- ▶ Get the applications migrated with the smallest number of changes and as quickly as possible.
- ▶ Do not skip the migration assessment before you begin, whether it is considered a simple or complex migration.
- ▶ In the assessment, review the code and evaluate how compatible your code is with a new platform's Java SE.
- ▶ In the assessment, review the code and evaluate how applications are following or deviating from Java EE specifications.
- ▶ Do not forget Integrated Development Environment (IDE) migration.
- ▶ Although both the application code and application servers (old and new) comply with Java EE specifications, each vendor can implement their code in their own way and can also provide extensions to the specification that must be migrated.
- ▶ Do not change too much at a time. Use an iterative approach. Migrate a small portion of code and test it, then go back and repeat the process. It is easier to find problems when you have a small controlled environment.
- ▶ Keep change variables small. For example, freeze code changes other than the changes that are migration-related. Evaluate the changes that are needed for build scripts and required runtime configurations.
- ▶ Determine how to deal with Java SE and Java EE specifications. Should you compile using compatibility with an earlier version or upgrade?
- ▶ Do not replicate your performance tuning from a previous application server. Make specific performance tuning decisions by using WebSphere.
- ▶ Get interested parties to discuss the migration and make them interact with each other to learn from previous implementations.
- ▶ Never forget that there are tools to help in the migration process.
- ▶ WebSphere Application Server V8 provides compatibility with an earlier version as far back as J2EE 1.2. However, most of the older application codes include third-party libraries, use vendor-specific extensions, and use non-public APIs. These codes must be redesigned, replaced, and reimplemented.
- ▶ There are cases where we face external factors that contribute to migration complexity and requires attention from all interested parties. These external factors are integrations with existing systems and interoperability between migrated applications that are dependent of other applications functionality or services.
- ▶ Always include a rollback plan. A rollback plan is critical to the business to ensure that the migration does not include any unplanned downtime. It is also important to practice the rollback plan before you attempt migration of the production system. In addition, make a backup copy of your code before you start a migration.

## 2.2 Migration assessment

Sometimes you are tempted to jump into an implementation and planning without enough information to do so, which sooner or later causes undesirable, but predictable, problems. These problems are anticipated and mitigated by performing a migration assessment.

The migration assessment is the starting point of a migration. There is no other way to know the scope, expectations, current environment, teams that are involved, and all other relevant information to the migration team except by research. The information that is gathered during the assessment phase provides an understanding of the application environment. This information can be used during the planning stage.

This section describes how the assessment can be conducted and the artifacts that can be produced during this stage.

## 2.2.1 Migration questionnaires

Because the assessment is one of the first steps in a migration, and you sometimes have no previous experience from the actual customer. You must know which information is required to start the project and determine the roles and responsibilities of interested parties. Stakeholders must be committed to the project to effect a successful migration.

To build this knowledge base, use migration questionnaires, which are a set of questions to gather necessary information about the migration, business requirements and decisions, hardware and software, topology, architecture, application code, and so on. During this step, remember to ensure that all questions are answered appropriately without generic answers such as “yes” and “no”.

The IBM WebSphere Application Server Migration Toolkit can be used when you gather information for the questionnaires. For sample migration questionnaires, see Appendix A, “Migration questionnaires” on page 421.

The Migration Toolkit has a reporting feature that discovers known Java issues that can be exported to a PDF report file. XML, JSP, and classpath results can be exported to an XML file.

## 2.2.2 Migration questionnaire review meeting

After you collect the required information, typically from separate teams, it is important to discuss it with interested parties. Make this review regularly, and have the stakeholders come together to discuss the responses that are received, resolve issues, and initiate related activities, such as code review and proof of concepts. These meetings can be done through calls, web conferences, and in-person consultations.

## 2.2.3 Migration assessment report

At the end of the cycle of questions and answers, put together all the information received in a report where everyone can see the key indicators that drive decision making. The contents of this report should include sections such as Introduction, Education, Sizing, Topology, Application Migration, Runtime (Server) migration, Testing, Deployment, Tuning and performance, Risks, and anything else that you feel is relevant to the project plan.

This report can minimize risks and positively affect the success of the project. It answers several questions that you might not have considered:

- ▶ What is the goal of the project?
- ▶ Which benefits does the project bring to the business?
- ▶ Are developers and administrators familiar with IBM WebSphere Application Server V8.5?
- ▶ Do testers know important areas of testing?
- ▶ How many resources and what skills are necessary to execute migration activities?
- ▶ How long it can take?

- ▶ What is the topology that we are planning to use?
- ▶ What other environments are involved?
- ▶ What are the recommendations for application code migration?
- ▶ What configurations must be done in the runtime migration process?
- ▶ Will you create the deployment and administration scripts?

These examples are ones of how the report assessment helps the migration, as the better the report, the better the project plan and the implementation.

## 2.3 Migration planning

The next step is to plan the migration itself. The assessment report is ready, and you know who the stakeholders are and the size of the work to be accomplished. Use the migration assessment as the basis to create a formal project plan.

### 2.3.1 Roles and responsibilities

In a migration process, a key decision is to define who does what. One of the first roles to be assigned is the project manager. The project manager plans and executes the project. Other roles are application owners, developers, testers, administrators, and decision makers. Assign the appropriate resources to be responsible for certain activities and make them accountable for the tasks that are assigned to that role.

Usually, the definition of who does the migration is decided in advance, but after the assessment report is created, the plans can change based on what is found. The assigned team influences the project plan. For example, depending on who does the migration, more or less time and resources are needed because of education issues or previous experiences and the complexity of the project.

### 2.3.2 Migration project plan

A project plan for the migration is based on the assessment report. From there, you can decide which activities are dependent upon each other, which activities can run in parallel, and the complexity of each piece that is migrated. Decisions that are taken and decisions that still must be taken are all part of a list of activities.

A project manager that is assigned to any project should know exactly how to deal with activities, but as the audience in this book is flexible, here are the sections that a good migration project plan should contain:

- ▶ Dates (rollout and milestones)
- ▶ Tasks
- ▶ Number of resources assigned
- ▶ Test plans
- ▶ Dependencies
- ▶ Risks
- ▶ Production rollout
- ▶ Meetings and check points
- ▶ Education activities
- ▶ Critical path

Some tasks can be done in parallel (education, setup, testing, and so on).



This list is not final, so you can add or remove sections that fit your needs. Documenting findings from migration implementation is helpful for future migrations.

Production rollout resources may be different from migration resources.

## 2.4 Migration implementation

An implementation and plan are closely related. In an iterative project, the final phases of an iteration are input for coming iterations. This approach helps adjust any issues that are encountered in the project plan, make it more accurate, and set expectations for future rollouts.

There are decisions to be made at the planning phase (for example, which application gets migrated first and which type of migration is used (sample, partial, or full)). You also must make implementation decisions, which are more likely technical, that can be addressed now.

### 2.4.1 Implementation considerations

Consider migrating the run time (server) and applications in parallel. This way, your environment is ready to deploy after you finish code changes and adjustments. Use the assessment report to accomplish the following tasks:

- ▶ Identify possible changes in the environment.
- ▶ Engage interested parties and communicate with them.

Study the run time and integration questionnaires to see how those answers can help you make the implementation a success. For example:

- ▶ Check for topologies and network answers.
- ▶ Identify firewall rules that might prevent your application from running appropriately.
- ▶ Check if interoperability of servers is required or if integration with existing systems is being addressed.
- ▶ Ensure that test plans cover functional and non-functional requirements.
- ▶ Try specific scenarios and separate sources of review to avoid issues when you move your application to production environments.

### 2.4.2 Implementation phases

Earlier in this chapter, we pointed out that the iterative process is a good option for migration projects. It helps reduce risk by successfully navigating critical paths between phases and providing feedback to future iterations. More on common issues are described in Chapter 3, “Common migration issues” on page 57.

Consider the five phases that are shown in Figure 2-2.

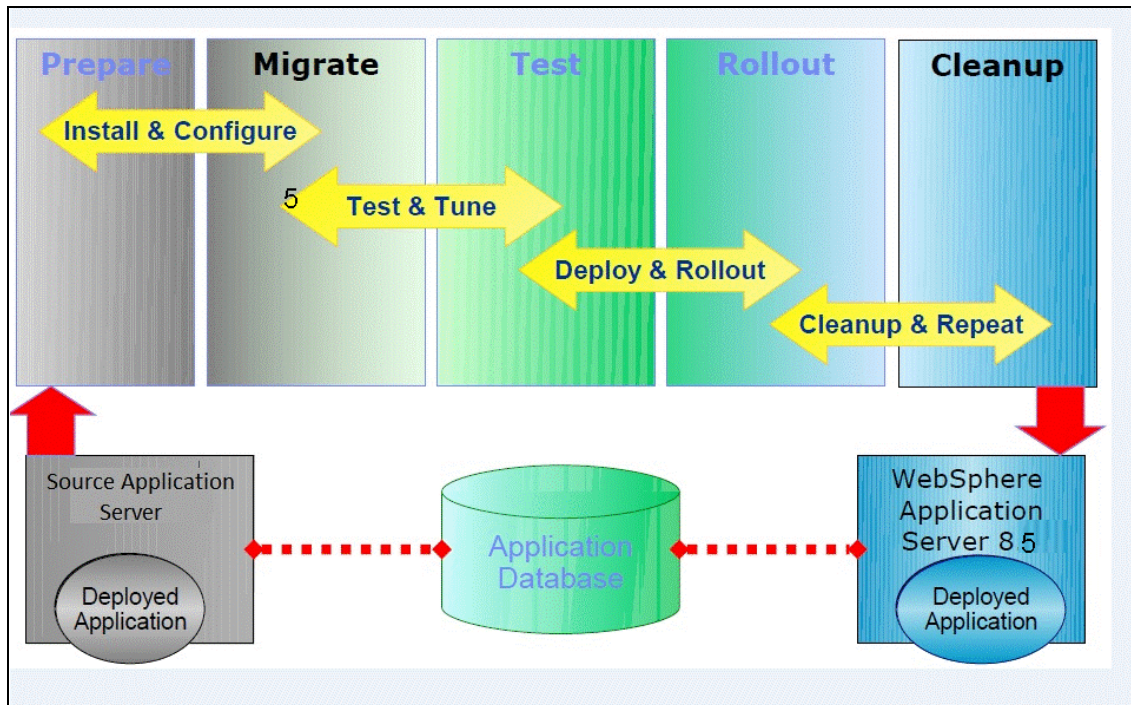


Figure 2-2 Migration key activities

- ▶ Prepare phase

During this phase, check for product and application prerequisites. You should have hardware, software, and people available. Set up development environments, schedule meetings, provide education, and get help if needed.

- ▶ Migrate phase

Start migrating both application code, runtime environments, and the Integrated Development Environment (IDE). Always remember that the assessment report is your reference; you can find important information regarding the decisions that are taken and sometimes even code review feedback that states the challenging areas of the migration.

You must decide about the migration type, so document everything that is happening and you benefit from your feedback immediately. If you are a project manager or a delegate, your role is important in addressing issues that are raised when everyone is together or interfacing with other parties, such as software and hardware vendors or third-party service providers.

The migration and test phases are closely integrated. It might be useful to test small pieces of functionality until you have everything migrated. This approach helps in two ways:

- Challenges of first migrated functionality serve as input to other migrations and speeds up the process.
- Tests run in parallel with code migration and are prepared for QA and load tests after migration is complete.

- ▶ Test phase

The test phase is dynamic and has intersections with the migrate and rollout phases. There are three main types of tests that are welcome in a migration process:

- Hands on testing, which is done together with application migration

- Functional/regression testing to make sure that all functions are working as expected
- Load tests to assess whether the application/server is performing equal or better than the previous environment
- ▶ Rollout phase
 

The rollout phase starts after all previous tests are completed and you ensured that the application is ready to provide the same or better responses than the previous environment. We know that there are issues that are only raised during production, which is why you have plans and the other previous phases to prevent or nullify chances of rollout back outs.

Deploy the application into the production environment, run the tests again, and if everything is okay, then decide whether to proceed. Usually, these rollouts require a detailed plan with milestones, owners, and check points to put the team together and address any problems or initiate the backout plan.

Rollouts with interoperability and interfaces are much harder to implement because of the potential compatibility problems with the existing applications. Be mindful about these challenges and prepare plans to prevent these issues. A good approach is to have this migration occur during a maintenance window so that you can stop external inputs to the systems affected and test integration and interoperability, to minimize the risks of having tests that are mixed with actual valid requests. A cleanup of databases or message queues might also be needed.
- ▶ Cleanup phase
 

This phase starts when you successfully deploy the migrated application into production. There are tools that help monitor performance and system stability that are useful in this phase. Assign specialists to make the appropriate tuning of settings that might be required. Make the changes in the non-production environments before you make them in the production environment and run through the test plans again.

### 2.4.3 Application migration

There are three distinct migrations:

- ▶ The server run time from the application server to IBM WebSphere Application Server V8.x.
- ▶ The development environment migration from vendor-specific IDE to WebSphere Application Server V8.x Developer Tools for Eclipse or Rational Application Developer for WebSphere Software.
- ▶ The application that is running on top of the application server.

Migrations of Java EE applications vary in complexity, depending on a series of factors that might be identified during the migration assessment:

- ▶ Source and target Java SE specification (JRE 1.4 to JRE 1.6)
- ▶ Source and target Java EE specifications (J2EE 1.4 to Java EE 6)
- ▶ Usage of specific Java components (EJB, JSPs, and Taglibs)
- ▶ Usage of proprietary code that is specific for an application server
- ▶ Vendor-specific deployment descriptors
- ▶ Extended capabilities
- ▶ Vendor-specific server configurations (JMS providers and JDBC connection pooling)
- ▶ Class loading issues
- ▶ Integration with complex external systems such as a CRM or SAP.

These changes can be done manually or by using tools to speed migration. IBM Rational Application Developer V8.5 has extended functions to help you in that matter. Additionally, a new Migration Toolkit is released that helps you migrate applications from Oracle WebLogic, Oracle Application Server, JBoss, Tomcat, and previous versions of WebSphere application servers to IBM WebSphere Application Server V8.x. For applications that use annotations, Rational Application Developer can provide them through WRD tags to build and maintain applications that are using EJB 3.x, web services, and other components. Other tools are also available to provide you with guidance in code reviews.

Application migration has common steps to be followed. Details about the technical activities are covered in the following chapters:

- ▶ Chapter 3, “Common migration issues” on page 57
- ▶ Chapter 6, “Migrating from Oracle WebLogic” on page 127
- ▶ Chapter 8, “Migrating from JBoss” on page 235
- ▶ Chapter 9, “Migrating from Apache Tomcat” on page 273

## 2.4.4 IDE migration

If you are using vendor-specific IDE, which provides tools and wizards for its application server and includes its own project structure, you must migrate it to WebSphere Application Server Tools for Eclipse or Rational Application Developer. Most of the Eclipse based IDEs (such as Oracle Workshop for Weblogic and JBoss Tools) are based on Eclipse WTP that makes it easy to migrate.

WebSphere Application Server Tools for Eclipse contains tools for managing the server, publishing to a local or remote server, and for controlling incremental publishing. The tools include a Rich Page Editor, and a WYSIWYG editor that makes it easy to edit HTML files, add Dojo widgets to HTML pages, and create and edit web pages for mobile devices.

Rational Application Developer V8.5 provides a complete suite for enterprise application developers from design to deployment. Rational Application Developer V8.5 helps Java developers rapidly design, develop, assemble, test, profile, and deploy high quality Java/Java EE, Portal, Web/Web2.0, OSGi, Web Services, and SOA applications.

The tool that is chosen in this book that helps the most with the migration samples is Rational Application Developer V8.5. Look at how Rational Application Developer V8.5 can help you improve and migrate from previous Java EE specifications and also take advantage of WebSphere Application Server V8.x new features. For more information, see:

- ▶ *Experience JEE Using Rational Application Developer V7.5*, SG24-7827
- ▶ *Getting Started with WebSphere Application Server Feature Pack for Service Component Architecture*, REDP-4633

### How to import code into Eclipse

You can import an individual EAR file to a project or import a project (with multiple EAR files or WAR files) into the Eclipse platform. They are all based on the same logic.

### Importing an EAR file to a project

If you want to import an EAR file to a project, complete the following steps:

1. From the Java EE Eclipse Enterprise Explorer tab, right-click and select **Import** → **EAR file** (Figure 2-3).

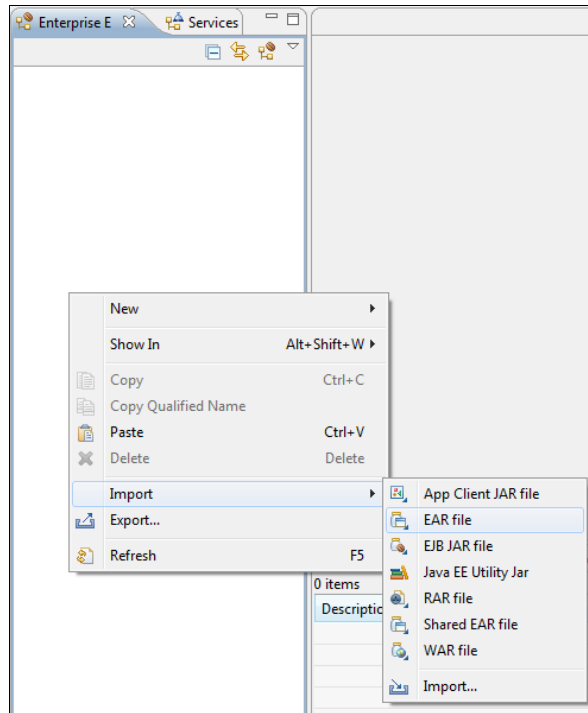


Figure 2-3 Select an import source window

2. Select the EAR file, for example, sampleEJB.ear (Figure 2-4).

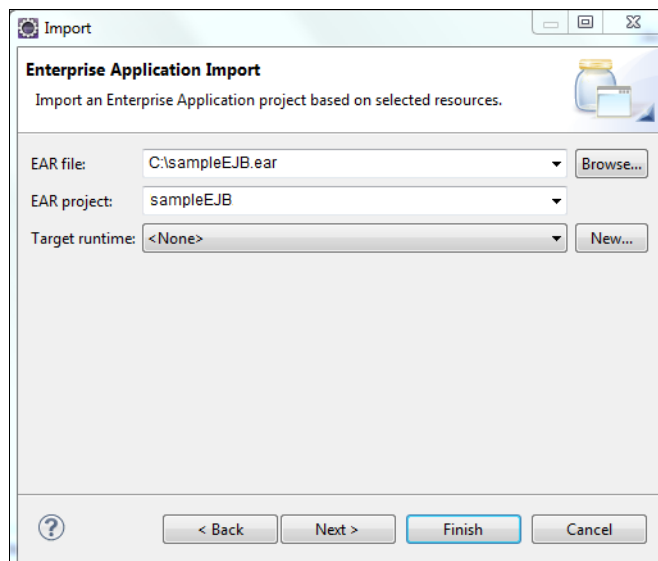


Figure 2-4 Select the EAR file

3. While you are importing your project, if you have not added WebSphere Application Server into the workspace, you must click **New** in the Enterprise Application Import window to select the target run time (Figure 2-5).

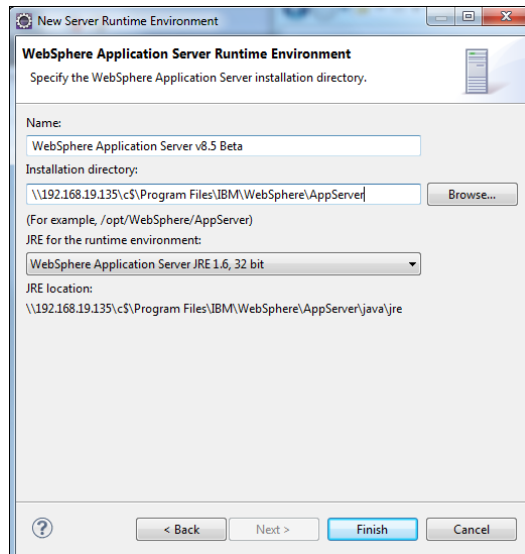


Figure 2-5 Selecting target run time

4. Click **Finish**. In the next window, select the JAR file to be imported (Figure 2-6).

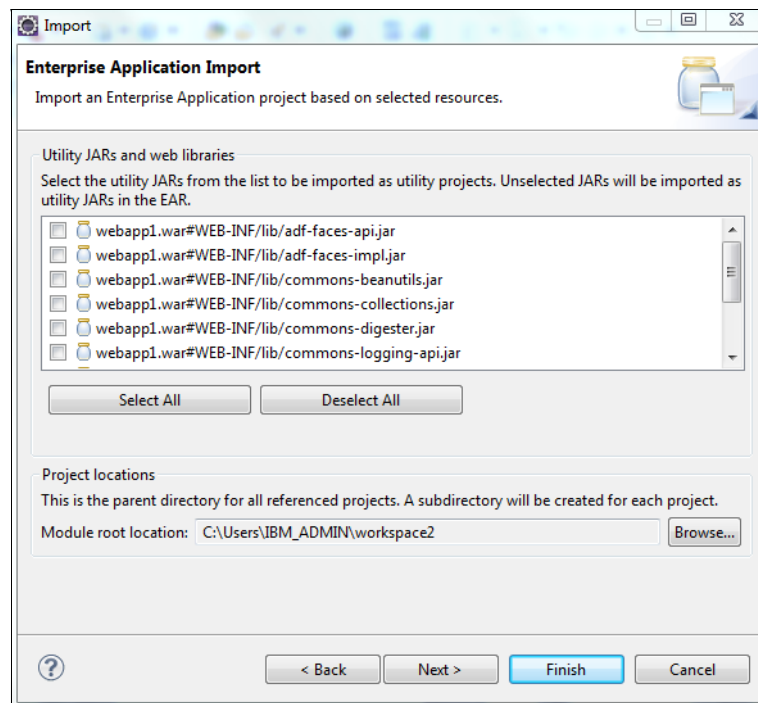


Figure 2-6 Select the JAR file

**Importing the Java source code:** Importing the EAR file builds the project structure for the application, web modules, and EJB modules. Usually, the EAR file does not include the Java source code, and it must be imported as an additional step.

### ***Importing an existing project into Eclipse***

You can also import an existing project into Eclipse by completing the following steps:

1. From the Eclipse main menu, click **File** → **Import**.
2. In the Select an import source window, enter “Existing project into workspace” and click **Next**. Alternatively, you can click **General** → **Existing project into workspace** and click **Next** (Figure 2-7).

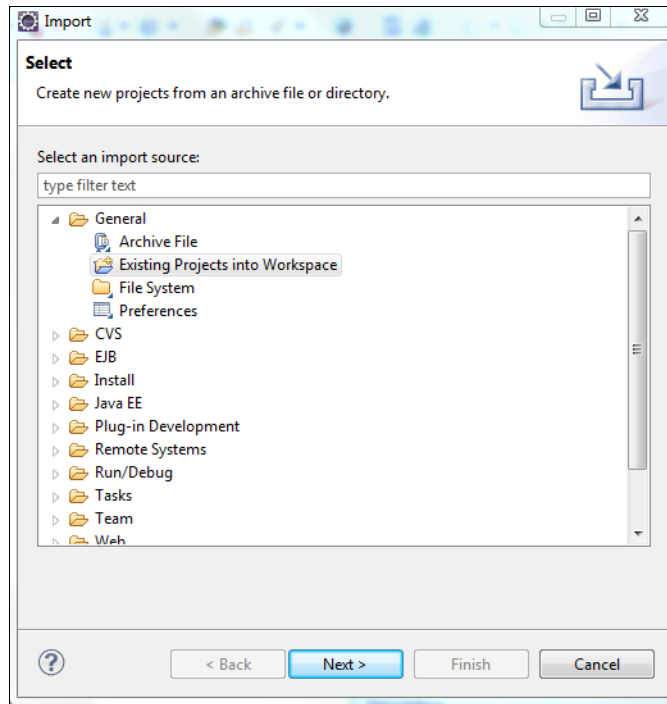


Figure 2-7 Existing project into workspace

3. Browse and select **Select root directory** or **Select archive file**, then select the projects that you want to add (Figure 2-8).

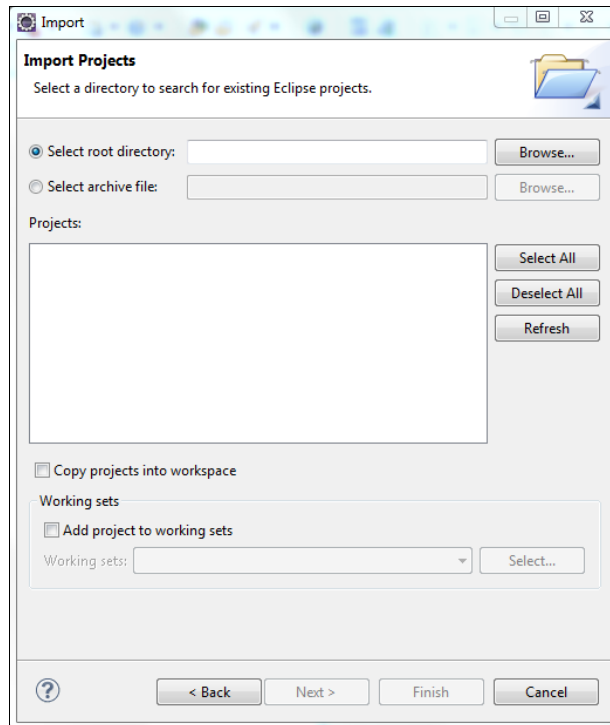


Figure 2-8 Browse and select the root directory or archive file

4. Click **Finish**.

## 2.4.5 Runtime migration

Sometimes we focus and concentrate only on code changes, and when we realize that there are also migration challenges in other areas, it might be too late. In a complex or simple environment, there is always something to change, implement, or migrate. Specifically, with the migration of Java EE application servers, though compliant with specifications, each vendor has its own way to implement and provide services and improve capabilities.

Considering that the application server is a product that runs under an operating system that can also be part of the migration, be mindful about the changes and configurations that are made at both the OS and application server levels. Migrations of run time can include the following elements:

- ▶ Administrative scripts changes or replacement
- ▶ Resources configuration to support application server functionality
- ▶ Operating system prerequisites
- ▶ Firewall rules
- ▶ Security configurations and authentication databases
- ▶ Deployment bindings
- ▶ Start/Stop scripts
- ▶ JVM special or additional configurations
- ▶ Classloader configurations
- ▶ Libraries
- ▶ Pooling and thread configurations



Migration also includes specific tasks from product-to-product. We try to cover those task in our product-specific chapters.

## 2.5 Post deployment

The new environment at least performs as well as the previous one, but you always expect more. To achieve this goal, you have work to do. This work is done on two fronts: application and run time.

Regarding code changes, avoidance of unnecessary modifications is prudent. Get your applications migrated with the smallest number of changes. After the application is migrated, revisit the application to improve it and make the application and the new environment clean, right, and fast.

### 2.5.1 Performance tuning and optimization

Performance tuning and optimization are activities that require expertise and information that support your decisions. This information has two sources: application code review and load test results. With these results in hand, the experts have enough information to make the necessary adjustments. Again, an iterative process is ideal. Take small pieces of code or specific configurations and focus on those elements until you get the expected results. This way, you pass through critical paths that lead to better performance for the entire application or environment.

This process of having better performance must be conducted carefully by following the project plan. Try to stay at least with what you had before, then after the migration is deployed in production, you can see how much you can improve performance. There are tools that help you to get there. IBM provides help through separate channels. Check the WebSphere Application Server V8 Information Center web page to get started:

[http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=welc\\_howdoi\\_tprf](http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=welc_howdoi_tprf)

### 2.5.2 Getting help

There are teams ready to help you in the migration process:

- ▶ IBM Business Partners
- ▶ IBM Business Partner Services Practice
- ▶ IBM Software Services for WebSphere Migration Practice (ISSW)
- ▶ WebSphere Competitive Migration Team
- ▶ Third-party sources

IBM migration specialists have been working in these environments for a long time. Expertise in migrations and in the target implementation environment is important and, in a few cases, crucial to minimizing disruption during implementation.

## **Get support from IBM**

Use the channels that are described in the following sections to get support and assistance from IBM.

### ***IBM WebSphere Developer Services***

IBM WebSphere Developer Services provides developer-to-developer technical assistance to IBM Business Partners when you are developing applications or building solutions that involve IBM WebSphere products. If you require any assistance, you can raise a ticket that is called a *problem management record (PMR)*. A PMR can be raised at the following web page:

<http://www.ibm.com/isv/tech/remoteEmail/entryForm.jsp>

### ***IBM Passport Advantage***

IBM Passport Advantage® provides support for production issues and product bugs. To open PMRs (or a service request) with Passport Advantage, go to:

<http://www-01.ibm.com/software/support/probsub.html>

### ***IBM Software Services for WebSphere (ISSW)***

IBM Software Services for WebSphere provides hands-on support and skill-transfer activities to help you successfully deploy your new WebSphere software solution. For more information about ISSW, go to:

<http://www3.software.ibm.com/ibmdl/pub/software/dw/wes/pdf/services/DevelopDeployFinal.pdf>

### ***WebSphere education***

WebSphere education helps you build and enhance our WebSphere skills. It has more than 250 courses across both the WebSphere software portfolio and SOA. It provides flexible classroom, online, and private courses. The courses are designed by award winning instructors with first-hand product knowledge.

The role-based training path in WebSphere education assists you by defining a path to acquiring skills for specific WebSphere product offerings. For a list of WebSphere education offerings, go to:

<http://www-01.ibm.com/software/websphere/education/>

## **WebSphere Competitive Migration Team**

WebSphere Competitive Migration Team is composed of migration specialists around the world who can provide the services that are described in the following sections.

### ***Migration Discoveries***

Helps customers understand the value proposition that is provided by WebSphere and the advantages to be gained over the competitive application servers.

### ***Migration Assessments***

Provides thorough application analysis with specific recommendations about the steps and methodology to be used to migrate competitive application server-based applications. In addition, provides implementation assistance to clients and ISVs to accelerate time-to-value and mitigate risk.

### ***Mentoring and Skill Transfer***

Creates more value by helping customers navigate and fully use the wealth of capabilities that are offered with WebSphere.

### ***Who to contact to get started***

To get started, contact your local IBM marketing representative, or send an email to [wascmt@us.ibm.com](mailto:wascmt@us.ibm.com).

You might qualify for some or all of the following services at no additional charge:

- ▶ An IBM migration expert to give you a high-level estimate of to what it takes to migrate (requires 2 - 3 hours.)
- ▶ An IBM migration expert to come on-site for a deep evaluation of your environment and applications and provide a full migration assessment at no additional charge.
- ▶ A Migration Workshop, in which several of your company's technical personnel or management attend a 1 - 3 day training course to get an introduction to WebSphere Application Server V8.5 and do hands-on labs to learn about the business value and migration process.
- ▶ A Proof of Concept (PoC) where an IBM migration expert comes on-site to migrate 1 or 2 of your applications to WebSphere Application Server on your test environment using the IBM proven methodology and automation tools.

## **2.6 Summary**

This process and activities are not supposed to be a final methodology or even the only way to go through a migration project. Instead, it helps you to be prepared and anticipate challenges to come. Following this process as a base for your own projects definitely helps. As you can see, there are several factors that influence your planning, but basically two things are essential:

- ▶ A migration assessment with the assessment report used as the input to the project plan
- ▶ Using an iterative process where first waves or iterations serve as a knowledge base for future ones

Another thing that is important to note is that the approach of migrate first and improve later is a good way to separate pitfalls of both activities, although they are related.





## Common migration issues

This chapter describes some of the common migration issues that the reader might come across when migrating from other Java Enterprise Edition (EE) platforms to WebSphere Application Server V8.5.

Java EE defines a framework for applications that can be deployed independently by different vendors. This configuration provides enterprise users with the assurance that they can write applications in a way that does not depend on a single vendor's solution. Although the main idea behind Java and Java EE is portability and *Write Once, Run Anywhere (WORA)*, or sometimes *Write Once, Run Everywhere (WORE)* functionality, the unique way in which each vendor implements the Java EE specification often leads to problems when you migrate a Java EE application.

Vendors also implement features that are not included in the Java EE specification, which is why Oracle provides a list of vendors with Java EE-compatible products. Testing is used by Oracle to verify that a Java EE application server complies with the specification. This testing ensures that applications deployed on one Java EE application server are also run on another vendor's application server.

Java EE applications can also conflict with the Java EE specification. The Java EE specification is so complex that it is difficult for developers to know all the details in the lifecycle. Tools, such as Rational Application Developer V8.5, can be used to verify that an application is written according to the specifications.

This chapter describes the following topics:

- ▶ Java EE application server compatibility
- ▶ Application portability
- ▶ J2EE to Java EE migration considerations
- ▶ Runtime migration issues
- ▶ Interoperability and integrations

## 3.1 Java EE application server compatibility

Although the Java EE specification covers an increasingly wider spectrum of enterprise applications development, there is still the need for vendor-specific features.

If the vendor-specific features are used without proper planning, they cause migration problems. Small differences or bugs in the vendor's implementation of the Java EE specification might also cause compatibility problems. These issues are summarized in the following sections:

- ▶ Differences in Java EE implementations
- ▶ Classloader related problems
- ▶ Using vendor-specific features
- ▶ Deployment descriptors

### 3.1.1 Differences in Java EE implementations

This section describes the differences that the authors encountered during the migration process and which are differences about how the Java EE specification is implemented by vendors. The Java EE specification leaves room for interpretation, which leads to issues when you migrate the application to a new application server. These interpretations are also subject to discussion and affected by updates to the specification.

A few of the problems that are faced during migration are not predictable by the developers. Other problems are more likely a bug or resiliency of the application server or web container implementation. For example, a regular request for a JSP or Servlet using a duplicate slash "/", as with the URL `http://localhost:8080//testapp/index.jsp`, works in Apache Tomcat 7, but does not work in WebSphere Application Server V8.5.

These problems require much testing to catch, but after you find the first problem, the search for other similar codes is easier. The IBM WebSphere Application Server Migration Toolkit detects many of these problems. Document your findings. Use a common repository, so they can be brought to light and shared across the entire team. Most likely, these implementation issues are discussed in a regular meeting.

### 3.1.2 Classloader related problems

Generally, the main migration issue is classloader implementations when you migrate to a new application server because the Java EE specification does not completely cover classloaders. Each vendor has its own way to implement the classloader. WebSphere Application Server V8.5 is compliant with Java EE specifications and has sophisticated techniques for structuring and loading classes for the applications using various versions of the same library.

Another problem is that vendors use open source libraries (Apache Commons, log4j, axis, and so on). JBoss, for example, has an implementation of classloader that does not differentiate classes from applications and the server itself. So, for example, an application that uses log4j does not need to deploy a log4j library. The best solution in that case is to use the exact same version of log4j that it is shipped with JBoss to avoid problems. You might face problems, such as a `ClassNotFoundException` error, when deploying JBoss-migrated applications. In these cases, you have a couple of options. WebLogic Server and Apache Tomcat Server use a similar parent-child classloader hierarchy as WebSphere Application Server, so we did not face this problem when you migrate sample applications from these servers.

If you want to avoid such problems, you must understand the fundamental mechanism of class loading and classloaders within WebSphere Application Server. The following section includes an explanation of configuration options for the classloader and how they can be used with the WebSphere Application Server V8.5.

The classloader hierarchy and delegation model are common between WebSphere Application Server V6.1, V7.0, and V8.x. There is one additional feature, which is called the *isolated shared library*, which is introduced with Version 7.

### Brief introduction to Java classloaders

Classloaders are part of the Java virtual machine (JVM) code and are responsible for finding and loading class files. In the Java platform, there are three classloaders:

- ▶ The *bootstrap* classloader is responsible for loading only the core Java libraries in the `Java_home/jre/lib` directory. This classloader, which is part of the core JVM, is written in native code.
- ▶ The *extensions* classloader is responsible for loading the code in the extensions directories (`Java_home/jre/lib/ext` or any other directory that is specified by the `java.ext.dirs` system property).
- ▶ The *application* classloader is responsible for loading code that is found on `java.class.path`, which ultimately maps to the system CLASSPATH variable.

The *classloader delegation model* passes loading requests to each other. Each classloader is a child of the parent classloader (Figure 3-1). The bootstrap classloader is at the top of this hierarchy. By default, when a resource must be loaded, a classloader delegates class loading to its parent before it tries to load the class itself. The parent classloader can be either another custom classloader or the bootstrap classloader. If it cannot be found, the resource is loaded by the child classloader. If none of the classloaders can find the resource, you receive a `ClassNotFoundException` error. But what is important is that a classloader can delegate requests only to its parent classloader, never to its child classloaders (it can go up the hierarchy but never down).

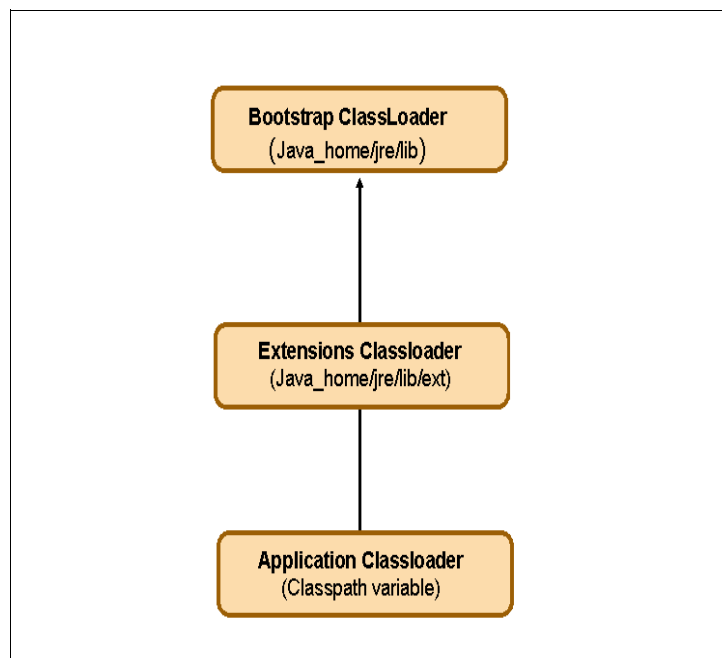


Figure 3-1 Java classloaders hierarchy

## WebSphere Application Server V8.5 classloader

WebSphere Application Server 8.5 is compliant with Java EE specifications and also has more features to suit your particular application using various versions of the same library.

When you work with Java EE applications, two additional types of classloaders are involved with WebSphere Application Server:

- ▶ The *application server* classloaders, which load all the classes that are needed for the application server in which the enterprise applications are running.
- ▶ The *application* classloaders, which load the application classes that are defined in the `web.xml` file.

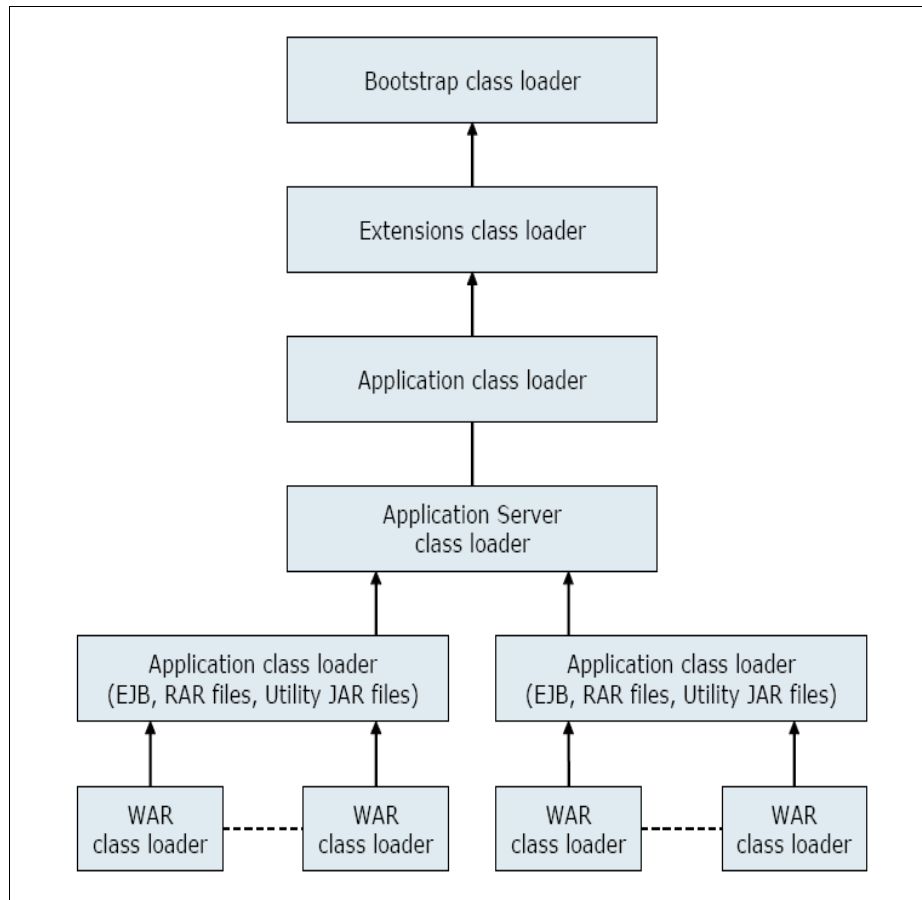


Figure 3-2 A typical Java EE classloader



WebSphere provides several custom delegated classloaders (Figure 3-3).

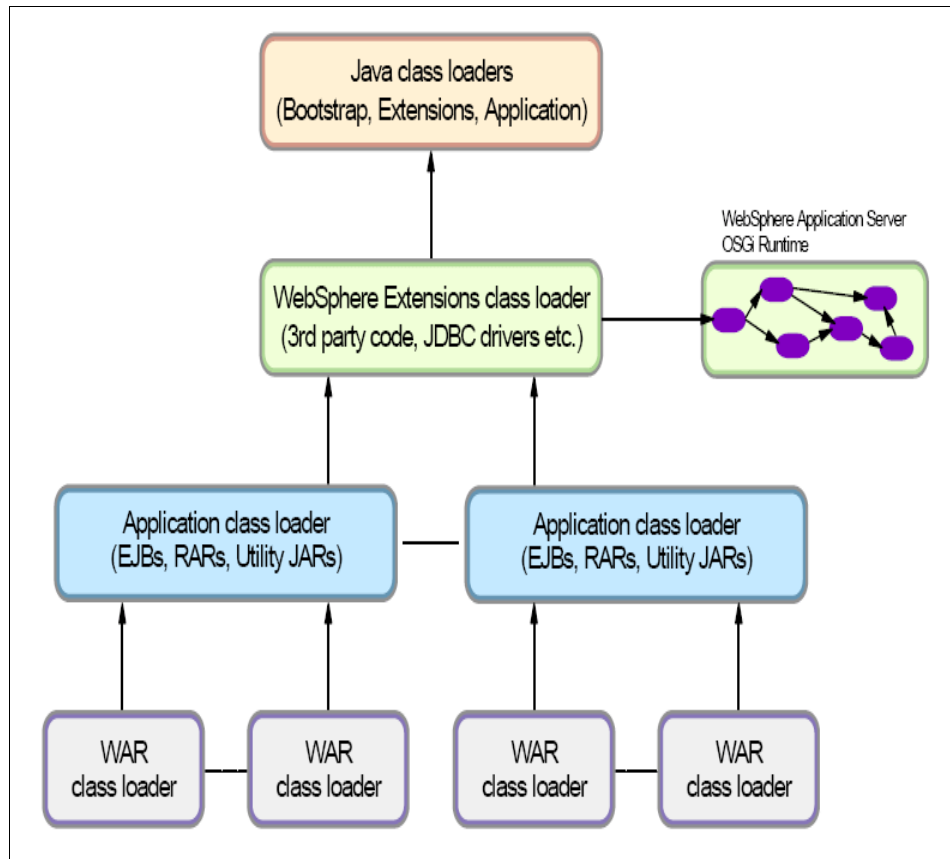


Figure 3-3 WebSphere Application Server classloader hierarchy

The top box in Figure 3-3 represents the Java classloaders (bootstrap, extensions, and application). WebSphere loads just the components that it needs to bootstrap itself and to initialize the WebSphere extensions classloader.

Information about the other components in this hierarchy is covered in the following sections.

### **WebSphere extensions classloader**

This component loads the WebSphere Application Server V8.5 runtime classes and all of classes in the `ws.ext.dirs` class path are added to this classloader.

WebSphere is packaged as a set of OSGi bundles with each OSGi bundle loaded separately by its own classloader. This network of OSGi classloaders is then connected to the extensions classloader and the remainder of the classloader hierarchy through an OSGi gateway classloader. Beginning with WebSphere V6.1, extensions began using OSGi packaging and the runtime classes are stored in the `install_root/plugins` directory.

### **Application and web module classloaders**

Java EE 6 applications consist of the following primary elements:

- ▶ Web modules
- ▶ EJB modules
- ▶ Resource adapter archives (RAR files)
- ▶ Utility JAR files
- ▶ Application client modules

The *application classloader* is responsible for loading resources that are part of an EAR module, for example, utility JAR, EJB modules, resource adapter files, and shared libraries. Depending on the classloader policy, this classloader can be shared by multiple applications (EAR files) or can be unique for each application, which is the default.

The class-loading behavior of an installed enterprise application can be configured at the application server level. For each application server in the system, you can set the classloader policy to either Single or Multiple. From the administrative console, click **Servers** → **Server Types** → **WebSphere application servers** and click the correct server. Then, on the Configuration tab under the “Server-specific Application Settings” section, select the appropriate classloader policy (Figure 3-4).

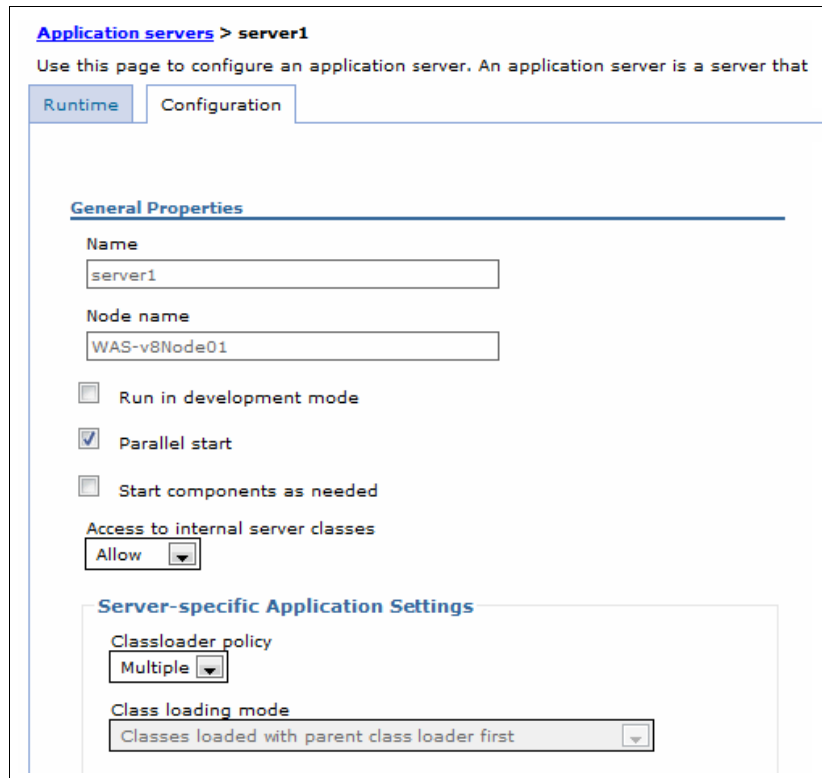


Figure 3-4 Application server classloader settings

When the application server classloader policy is set to Single, a single application class loader is used to load all EJB, utility JAR files, and shared libraries within the application server (JVM). If the WAR classloader policy is set to use the “Single classloader for application” option, the web module contents for this particular application are also loaded by this single classloader.

There are limitations when you use this mode, such as for JSF applications. For more information, see:

<http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=cwebjvaserverfaces>

When the application server classloader policy is set to Multiple, which is the default, each application receives its own classloader for loading EJB, utility JAR files, and shared libraries. Depending on whether the WAR classloader policy is set to use the “Classloader for each WAR file in application” option or the “Single classloader for application” option, the web module might or might not receive its own classloader.

The *web module classloader* loads the contents of the WEB-INF/classes and WEB-INF/lib directories. By default, web modules receive their own classloader, a WAR classloader, to have a classloader for each WAR file in the application. It is possible to modify the default behavior by changing the application's WAR classloader policy. You can find this policy setting in the administrative console by clicking **Applications** → **WebSphere Enterprise Applications**, select the application, and click **Class loading and update detection** → **WAR class loader policy**, which opens the window that is shown in Figure 3-5.

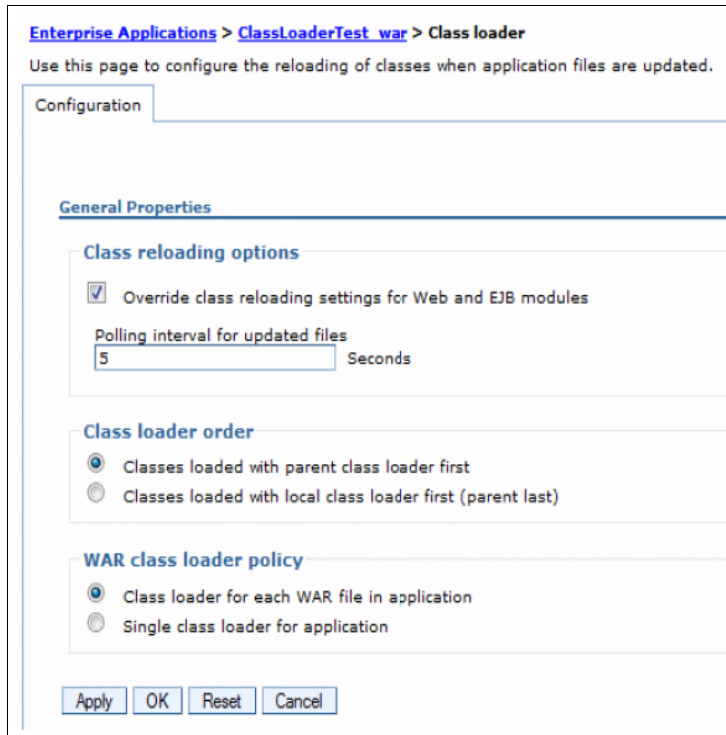


Figure 3-5 WAR classloader policy settings

If the WAR classloader policy is set to the “Single classloader for application” option, the web module contents are loaded by the application classloader in addition to the EJB, RAR files, utility JAR files, and shared libraries. The application classloader is the parent of the WAR classloader.

**Tip:** During the authors’ migration experiences, we encountered clients that require enterprise applications to communicate across various WebSphere cells. In this case, a CORBA style and an explicit JNDI name are specified. For more information about this topic, see:

<http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=cejbbindingsejbf>

### **WebSphere class loading and delegation mode**

The WebSphere application classloader and WAR classloader both have a setting that is called the *classloader order*. This setting determines whether the classloaders should follow the normal Java classloader delegation mechanism or override it.

There are two possible options for the class loading mode:

- ▶ Classes that are loaded with the parent classloader first

This mode is the default mode for all classloaders. It is the Java EE and Java standard. The loading of the resources is first delegated to the parent classloader. If the parent classloader is unable to find the resource, the child classloader tries to handle it.

- ▶ Classes that are loaded with the application classloader first

Using the parent last class loading policy makes the classloader work opposite of the parent first mode. The resource is loaded from the application or web module classloader first. If the resource is not found, the loading is delegated to the parent classloader.

In previous WebSphere releases, these settings were called PARENT\_FIRST and PARENT\_LAST.

### **Shared libraries**

*Shared libraries* are files that are used by multiple applications. Examples of shared libraries are commonly used frameworks, such as Apache Struts or log4j. You use shared libraries typically to point to a set of JAR files and associate those JAR files to an application, a web module, or the classloader of an application server. Shared libraries are useful when you have different versions of the same framework that you want to associate to different applications.

Shared libraries are configured using the administration console and can be defined at the cell, node, server, or cluster level. Before you use shared libraries, they must be associated with an application, a web module, or the classloader of an application server level.

### **Bundled libraries**

WebSphere Application Server V8.5 contains open source libraries that are in the server class path. If you are using the same libraries and there is a version mismatch, you might receive one of the following errors:

- ▶ `java.lang.NoSuchMethodError`
- ▶ `java.lang.ClassCastException`
- ▶ `java.lang.NoClassDefFoundError`

To fix these errors, you can try to experiment with various class loading policies, such as parent last instead of parent first.

The following list details the common libraries that might create problems:

- ▶ WebSphere Application Server V8.5 includes the Jakarta Commons Logging (JCL), which is known to cause problems when you migrate.

The following topics describe how to solve JCL problems on WebSphere:

- [http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=ctrb\\_classload\\_jcl](http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=ctrb_classload_jcl)
- <http://www.ibm.com/support/docview.wss?uid=swg27004610>

- ▶ JDom is also bundled with WebSphere Application Server V8.5.

### **Problem determination for classloader exceptions**

Classloader problems normally appear as an exception in the JVM SystemOut log file. When faced with problems, you can use a Classloader viewer service to collect classloader specific traces and other diagnostic data. You can use this information to identify possible root causes of the problem.

For more information, see *WebSphere Application Server V6.1: Classloader Problem Determination*, REDP-4307.

### 3.1.3 Using vendor-specific features

When faced with the choice of using a vendor-specific feature, you have the option of not using it and accepting the limitations of the Java EE specification. This is a good idea from a portability and migration point of view.

You can also use the vendor-specific feature and accept the risk and potential future migration costs. If you decide to proceed with this option, ensure that you make the implementation details abstract, for example, by designing interfaces, not concrete classes.

Vendors have various additional features:

- ▶ Startup and shutdown tools
- ▶ Timer services
- ▶ Automatic primary key generation
- ▶ JDBC connection pooling

### 3.1.4 Deployment descriptors

The Java EE specification does not cover all the aspects of application deployment and configuration, which is why there are vendor-specific deployment descriptors. Table 3-1 on page 67 lists both the Java EE and vendor-specific deployment descriptors. Each application server uses a separate set of files because they are implemented differently and have separate feature sets.

The migration of a Java EE application using previous specifications such as J2EE 1.4 come with one of the biggest problems that are associated with deployment descriptors when you migrate to WebSphere Application Server V8.5. The problem and solutions to the problem are summarized as follows:

- ▶ Container-Managed Persistence (CMP) to database schema mapping

WebSphere Application Server V8.5 uses another way of mapping CMPs to the database schema, compared to the other application servers covered in this publication. The better way to create these mappings is to use Rational Application Developer V8.5. You can use the EJB to Relational Database Synchronization (RDB) mapping in Rational Application Developer V8.5 to create your back end and map your entity beans to the corresponding database table. This wizard also creates the necessary JDBC code that is used for accessing the database at run time.

**ELB 3.0:** The EJB 3.0 or later specification does not support the usage of container-managed persistence (CMP) or bean-managed persistence (BMP). JPA is used for data persistence instead. The EJB 2 or later specification and earlier modules continue to support CMP and BMP as per the J2EE specifications. WebSphere Application Server V8.5 also supports CMP and BMP.

- ▶ The Java EE standard deployment descriptors are mapped to the WebSphere Application Server V8.5 specific deployment descriptors with IDs, as illustrated in Example 3-1 and Example 3-2. The ID in the WebSphere Application Server V8.5 specific deployment descriptor must match the one in the Java EE standard deployment descriptor.

*Example 3-1 Web.xml excerpt*

---

```
...
    <resource-ref id="ResourceRef_2">
        <res-ref-name>jms/QueueConnectionFactory</res-ref-name>
        <res-type>javax.jms.QueueConnectionFactory</res-type>
        <res-auth>Application</res-auth>
        <res-sharing-scope>Shareable</res-sharing-scope>
    </resource-ref>
...
```

---

*Example 3-2 WebSphere Application Server V8.5 deployment descriptor excerpt*

---

```
...
    <resRefBindings xmi:id="ResourceRefBinding_1117409057536"
jndiName="jms/QueueConnectionFactory">
        <bindingResourceRef href="WEB-INF/web.xml#ResourceRef_2"/>
    </resRefBindings>
...
```

---

- ▶ WebSphere Application Server V8.5 validates the deployment descriptors more strictly than, for example, JBoss and Oracle Weblogic, which leads to the following problems when migrating:
  - Deployment descriptors fail validation because the elements are not in the correct order or the DTD is incorrect.
  - JSPs do not compile.
- ▶ Use `web.xml` to map JNDI references that are placed on vendor-specific deployment descriptors. If you are migrating to Java EE 6, you are likely using annotations that simplify references using embedded dependency injection that is implemented by WebSphere Application Server V8.5, such as `@EJB`.
- ▶ The switch JNDI reference mapping since WebSphere Application Server V8.5 JNDI standards is different than in other vendors. It is usually different in all other vendors, so it is actually a migration issue for almost every application server to another.

Next, we show an example of how the standard Java EE `web.xml` deployment descriptor and WebSphere Application Server V8.5 are mapped with IDs. In the example, a resource reference, with the `ResourceRef_2` ID, is mapped to a JNDI name that points to a resource that is managed by WebSphere Application Server V8.5. See Example 3-1 and Example 3-2.

Table 3-1 lists the Java EE standard and vendor-specific deployment descriptors.

Table 3-1 Java and vendor deployment descriptors and configuration files

Description	Java EE specification	WebSphere Application Server V8.5	Apache Tomcat	JBoss	Oracle Application Server	Oracle WebLogic
EAR application	application.xml	ibm-application-bnd.xmi ibm-application-ext.xmi	N/A	jboss-app.xml	orion-application.xml	weblogic-application.xml
WAR application	web.xml	ibm-web-bnd.xmi ibm-web-ext.xmi	context.xml <sup>1</sup>	jboss-web.xml	orion-web.xml	weblogic.xml
EJB application	ejb-jar.xml	ibm-ejb-jar-bnd.xmi ibm-ejb-jar-ext.xmi ibm-ejb-access-bean.xml	N/A	jboss.xml jaws.xml jbosscmp-jdbc.xml	orion-ejb-jar.xml	weblogic-ejb-jar.xml weblogic-cmp-rdbms-jar.xml persistence-configuration.xml
J2EE client application	application-client.xml	ibm-application-client-bnd.xmi ibm-application-client-ext.xmi	N/A	jboss-client.xml	orion-application.xml oracle-application-client.xml	client-application.runtime.xml
Server configuration	Not covered by the Java EE specification	Mainly in <was_home>\profiles\ <profile_name&gt;\config\cells< td=""> <td>Mainly in the &lt;tomcat_home&gt;\conf directory and &lt;web_app&gt;\META-INF/context.xml</td> <td>Mainly in the &lt;jboss_home&gt;\server directory</td> <td>Mainly in ORACLE_HOME\j2ee\<server_name&gt;\config directory<="" td=""> <td>Mainly in &lt;domain_home&gt;\config.xml</td> </server_name&gt;\config></td></profile_name&gt;\config\cells<>	Mainly in the <tomcat_home>\conf directory and <web_app>\META-INF/context.xml	Mainly in the <jboss_home>\server directory	Mainly in ORACLE_HOME\j2ee\ <server_name&gt;\config directory<="" td=""> <td>Mainly in &lt;domain_home&gt;\config.xml</td> </server_name&gt;\config>	Mainly in <domain_home>\config.xml

1. The context.xml file can have JNDI names and a configuration that would normally be in web.xml.

WebSphere Application Server binding and extension files that are defined in Table 3-1 can be in XMI or XML format depending on the Java EE version of the application. XMI formatted files are used with a J2EE version of 1.4 and earlier. Starting with Java EE 5, binding and extension definitions support files with the suffix of XML for EJB 3.x and Web 2.5 and later modules. During migrating an application, it is important to include the correct file format with the application. If an application is developed with Java EE 5 or later, bindings and extensions are expected in the XML file format.

**New in V8.5:** WebSphere Application Server V8.5 supports the Java EE 6 and the EJB 3.1 specifications, which allow developers to use annotations in their source code. These annotations contain information about how the application should be deployed, making much of the packaging and deployment tasks. Annotations can also reduce the number of classes and interfaces that the developer needs to manage within the project.

In previous Java EE versions, deployment descriptors are necessary to tell the application server how to deploy the modules and how clients should locate EJB interfaces. In Java EE 6 applications, the use of deployment descriptors is optional. If the code is correctly annotated, the EAR file or modules do not need to contain any deployment descriptors when you migrate to WebSphere Application Server V8.5.

## 3.2 Application portability

This section describes the migration issues that affect application portability. Application portability is perhaps the most important issue from a migration point of view.

### 3.2.1 Application packaging

Enterprise applications (Java EE applications) can consist of EJB modules, web modules, resource adapters, SIP modules, utility JARs, and application client modules. There may be differences because of the need for vendor-specific features when you package an application for various platforms. This situation, combined with the classloader implementations in each application server, can potentially create problems when you migrate. The application might run in the source environment but not in WebSphere Application Server V8.5.

It is important to follow the Java EE specifications from a portability and migration point of view.

Java EE applications are usually packaged in an enterprise archive (EAR) file, which can consist of the following modules:

- ▶ EJB Module

The module contains the EJB class files and deployment descriptors. The EJB module depends on utility classes and JARs. Each dependency is declared in the META-INF/MANIFEST.MF file.

**EJB interfaces:** The way you package the application might be different, depending on whether you are using Local or Remote EJB interfaces. If you use Local EJB interfaces, the client must have access to all EJB classes. If you use Remote interfaces, you can optionally create a JAR containing only the client view of the EJBs. Rational Application Developer V8.5 usually creates a separate project with EJB interfaces.

- ▶ WAR (web application) Module

A web module can contain deployment descriptors, servlet code, JSPs, static HTML pages, images, JavaScript, style sheets, and so on.

A common challenge when you work with web modules is to make sure that the correct version of a required Java library is loaded. Often, web application developers need to include specific third-party libraries, such as log4j or Xalan/Xerces, and must ensure that the correct version of a library is loaded for an application.

The web application module depends on the utility library and the EJBmodule. Each dependency is declared in the META-INF/MANIFEST.MF file.

**application.xml descriptor:** If an application.xml deployment descriptor is not included in the EAR file, the context root for a web module defaults to the web module's name without the .war extension.



► Utility JARs

Common utility JAR (Java ARchive) or class files are common application code and are also used by other modules and deployment descriptors for a Java EE application client. Every application is unique and its requirements are different; it is not simple to determine where best to place a utility file.

To use the utility JARs that come with your packages, it is important to determine where these files should be included in the WebSphere Application Server environment and which classloader policy settings should be used. If the utility JARs are shared by more than one enterprise application, you can configure a shared library at the server, node, or cell scope, depending on where the applications using these shared libraries are, and associate it with all the applications that use it.

If multiple modules within the application need this utility file, you can place the utility file in the root directory of the EAR, and modify the `manifest.mf` file to reference the utility JAR. After you place the utility file in the correct location, you can alter the classloader settings to use a utility file that is already included with WebSphere Application Server.

For example, Apache Log4j is a popular logging implementation to perform logging from both the EJB module and the web modules. It is tempting to configure log4j as a utility JAR so you have only a single copy of it in your EAR or keep the JAR file in each EJB and web module. For more information about this topic, see *WebSphere Application Server V6.1: Classloader Problem Determination*, REDP-4307.

► RAR (Resource Adapter)

A resource adapter is a system-level software driver, also called a *connector module*, that a Java application uses to connect to a back-end enterprise information system (EIS), such as CICS, SAP, and PeopleSoft. RAR files are packaged as a standard Java archive with a `.rar` file extension. It also contains a mandatory deployment descriptor file named `ra.xml` that is in the module's META-INF directory.

Figure 3-6 illustrates the preferred way of packaging an application.

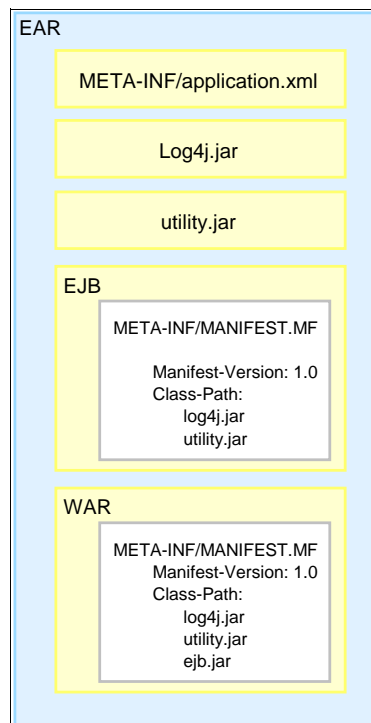


Figure 3-6 Preferred packaging structure of a Java EE application

The migration and creation of the package can be better accomplished by using Rational Application Developer V8.5. When you use Rational Application Developer V8.5, you can look for any potential migration problems easily, and it assists you with the creation of the configuration of the server.

**Tip:** If you are moving a project into Rational Application Developer, export the project from your source Eclipse as an EAR file that includes the source, and then import this file into Rational Application Developer. If you choose to import the file as an existing project, you must manually enable the Project Facets for WebSphere (co-existence) and WebSphere (extended).

## Migration problems

In Java EE 6 applications, EJB 3.1 content can be packaged and deployed as part of a WAR file. A bean that is packaged inside a WAR module has the same behavior as a bean that is packaged inside an EJB JAR module.

**Tip:** Always include the version number of libraries, frameworks, and applications in either the MANIFEST.MF or the name of the file. During our migration process, we were unable to identify the version of many frameworks. Knowing the version number is critical when you need support.

During the migration of the sample applications that contain a previous Java EE specification, we discovered the following packaging-related problems:

- ▶ EJB module references classes in the WAR module

The application worked on JBoss because of the JBoss Unified classloader flat hierarchy. It stops working when deployed to WebSphere Application Server V8.5, because the EJB cannot view what the WAR module contains.

**Tip:** The EJB module should not depend on classes in the web module. This situation creates a dependency that does not work on platforms other than JBoss. Extract these classes from the web application module and put them in a utility JAR. Use the META-INF/MANIFEST.MF file to declare the dependency.

- ▶ EJB module references libraries that are in the WAR module

In one application, EJBs imported classes from both Apache Axis and Apache Struts. The required JAR files are in the web modules WEB-INF/lib folder and are only visible to that module.

**Tip:** Dependencies on libraries that are shared between Java EE modules in an application are in the root of the EAR file. The modules that have the dependencies should declare the dependency by using the META-INF/MANIFEST.MF file.

You can place common JARs in a shared library in WebSphere and set the scope to **Application**. All modules in the EAR file are able to use such libraries. Shared libraries can also be set at the server scope.

### 3.2.2 Java source code and JSP file issues

Many of the issues that are related to the Java source code are related to specific vendor's implementations of specifications. Here is a list of changes that you might need to perform:

- ▶ JNDI-related:
  - Context Factory that is provided by WebSphere:  
`com.ibm.websphere.naming.WsnInitialContextFactory`
  - Provider URL, for example, `corbaloc:iiop:<hostname>:2809` (or other if changed)
  - Discard other JNDI properties specific to other application servers, such as in JBoss:  
`java.naming.factory.url.pkgs`
- ▶ Avoid using a `.jsp` extension for JSP fragments. Use `.jspx` instead. Every `.jsp` must compile.
- ▶ Use the correct case for tag/attribute names. Tag/attribute names are case-sensitive.

### 3.2.3 Usage of native code

Java allows the usage of native code. If your application code uses the Java Native Interface (JNI) code, be aware that the JNI allows Java code that runs in a virtual machine to operate with applications and libraries that are written in other languages, such as C, C++, and assembly. Using native code can cause problems when you migrate to a new operating system.

During migration assessment, discover whether an application is using native code. For example:

- ▶ Libraries in windows to manage services through WMI
- ▶ Usage of shell scripts capabilities in an UNIX environment
- ▶ Integration with a device
- ▶ Any other library that makes usage of native code

Sometimes migration happens in many areas, including operating systems, so make the code portable after you move to the new server. Migrating applications that make usage of native code might also require the native code to be recompiled, for example, in a migration from a 32-bit environment to a 64-bit environment. In addition, changes in the Java code that call the JNI API might be needed as the JNI specifications can change from version to version.

Additionally, when you define shared libraries, if a native path for loading JNI libraries is set, check it as well.

## 3.2.4 Database-related issues

In our migration examples, we used applications that were developed for and running on various databases, including HSQL and MySQL. We migrated these databases to IBM DB2 Universal Database™ V10.1 Express Edition. This section describes the issues that we encountered and the solutions to them:

### ► Migrating database schemas

Most issues with migrating database schemas are easy to solve either by manually editing the DDL or SQL scripts or by running them through a tool.

For more information about MySQL migration to DB2, see *MySQL to DB2 Conversion Guide*, SG24-7093.

In our example, the following problems were the most difficult:

#### – Data type overflow

When you move one database to DB2 from MySQL, we changed the data type of one column from DATE to TIMESTAMP, because the application was inserting TIMESTAMP.

#### – Automatic primary key generation

MySQL supports AUTO\_INCREMENT columns. The DB2 equivalent for this column is the *identity* column. We changed the schema to use identity columns.

### ► Migrating J2EE 1.4 CMPs

In our migration examples, we mapped CMPs to the existing database schemas. We used the Rational Application Developer V8.5 EJB to RDB mapping tool, which supports the following options:

#### – Top down

This option creates a database schema from existing CMPs.

#### – Bottom up

By using the bottom up approach, you can generate CMPs from an existing database schema.

#### – Meet-in-the-middle

You can use this option to map the CMPs to database tables and columns automatically. The tool tries to match column and table names to the CMP. If the column or table names do not match, you can create the mapping manually with the Mapping Editor.

### ► Primary key generation

Both JBoss and Oracle WebLogic contain vendor-specific EJB extensions that provide support for automatic primary key generation, as described in 3.1.3, “Using vendor-specific features” on page 65.

**Migrating from Oracle to DB2:** If you are migrating your application from Oracle to DB2, see *Oracle to DB2 Conversion Guide: Compatibility Made Easy*, SG24-7736.

### 3.2.5 Java EE application clients

A Java EE application client is a stand-alone client that runs in a separate process from the application server. The client is given access to resources in the WebSphere Application Server V8.5 Java EE Client Container. The Java EE application client can be used to communicate with EJBs or other resources that are deployed on the application server. For example, you can use a graphical Swing application that calls EJB beans on an application server. There are several use cases where Java EE application clients are appropriate:

- ▶ As an alternative user interface for the application
- ▶ Running unit tests
- ▶ Administrative tasks

Migrating Java EE application clients is not covered in this book. This book provides only an overview of the options that are provided by WebSphere Application Server V8.5.

WebSphere Application Server V8.5 Java EE application clients are run by running the following command:

```
<was_home>/bin/launchClient.bat client.ear
```

The script sets up all the necessary class path and configuration variables and runs the Java EE application client, which must be packaged as an EAR file.

WebSphere Application Server V8.5 supports several different application client environments. The following application client environments are available:

- ▶ Java EE client

This client is a Java application program that accesses EJB beans, JDBC databases, and JMS queues using the Java EE client container.

The Java EE application client provides the following components:

- XML deployment descriptors
- Java EE naming (`java:comp/env`), including EJB references and resource references

- ▶ Thin clients

Thin application clients provide a more lightweight alternative to Java EE application clients. Thin clients do not support a Client Container, and to communicate with the WebSphere Application Server, the RMI-IIOP protocol is used. They do not have access to all the services provided by WebSphere.

For example, JNDI lookups do not have access to `java:comp/env`. Instead, the thin client must provide the fully qualified path, including the physical location (Example 3-3).

*Example 3-3 Thin client looking up an EJB deployed on a single server*

---

```
...
context = new InitialContext(env);
remote =
(CustomerServiceRemote)context.lookup("com.ibm.itso.CustomerServiceRemote");
...
```

---

To run thin clients, you need to install the Java EE and Java thin application client, which is part of the WebSphere Application Server V8.5 installation package and can be installed separately.

For more information about how to run and develop thin clients, see the WebSphere Application Server V8.5 Information Center:

[http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=welc6tech\\_cli\\_dev](http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=welc6tech_cli_dev)

► Applet application client

The Applet application client provides a browser-based Java run time capable of interacting with enterprise beans directly, instead of indirectly through a servlet. A Java applet that is embedded in an HTML document runs in a web browser.

► ActiveX to EJB Bridge application client

The ActiveX application client allows ActiveX programs to access enterprise beans through a set of ActiveX automation objects. The ActiveX application client uses the Java Native Interface (JNI) architecture to programmatically access the Java virtual machine (JVM) API. The ActiveX to EJB Bridge is supported on Windows systems only.

For detailed capabilities about each client container, search on “Client Application” at the Information Center, or go to the following website:

[http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=tcli\\_developactivex](http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=tcli_developactivex)

Several types of clients are installed either with WebSphere Application Server or, optionally, with the Application Client for WebSphere Application Server (Figure 3-7).

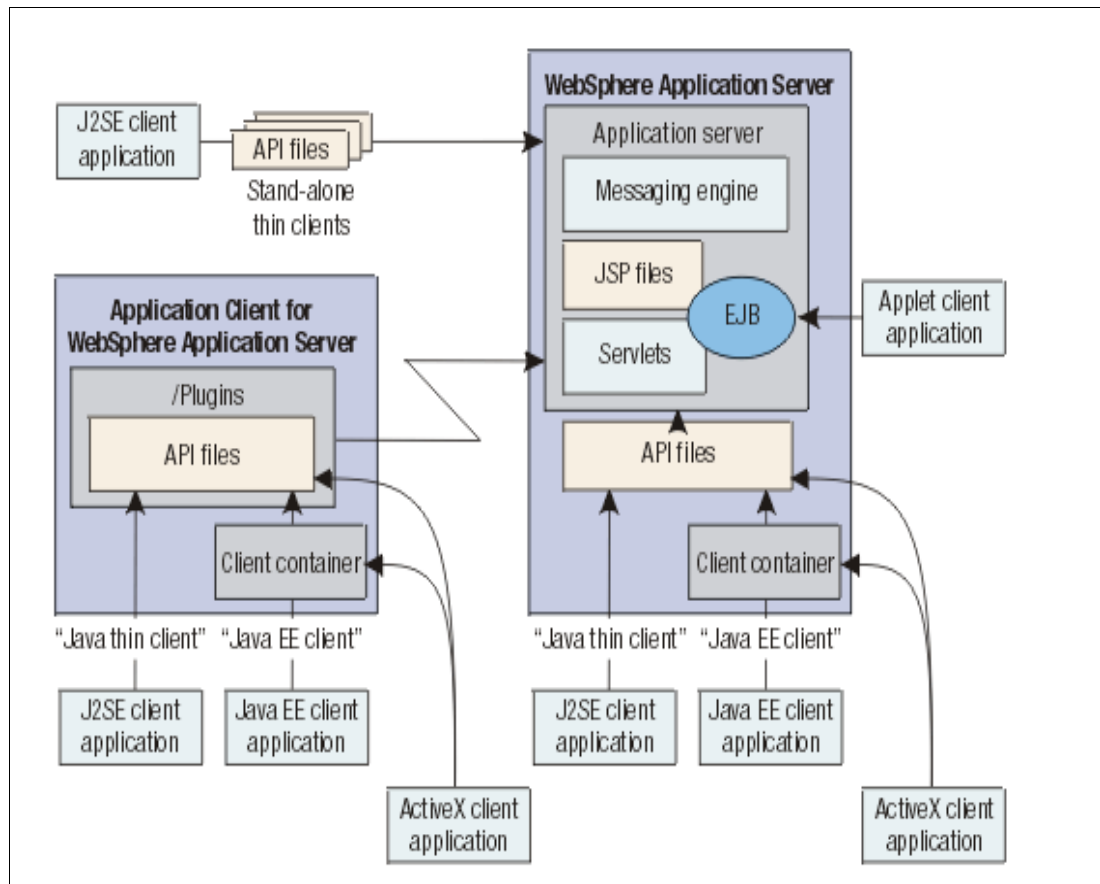


Figure 3-7 Clients that are provided for WebSphere Application Server

### 3.3 J2EE to Java EE migration considerations

Java EE is growing and evolving for the first release of specifications and new features are defined periodically with a new Java EE specification.

J2EE 1.4 solved many of the problems of the previous versions. The main change to the J2EE 1.4 platform was the strong integration of web services. J2EE components could interact with web services and implement them in a standard by using Java API for XML Remote Procedure Calls (JAX-RPC) and SOAP with Attachments API for Java (SAAJ 1.1). Other than web services, the major changes in EJB 2.1 include important enhancements to EJB-QL, a new timer service, and the extension of Message-Driven Beans (MDBs) to support any type of messaging system (not just JMS).

It was a significant transformation from J2EE to Java EE. EJB 3.0 was introduced as a new business component development model, the old EJB 2.x Entity Bean model was replaced by JPA as the persistence module, JAX-RPC was replaced by JAX-WS 2.0 as the SOAP web services API, and JSF was introduced as the standard presentation tier framework.

Java EE 5 clearly focused on reducing complexity by using annotations, POJO programming, zero-configuration systems, and freedom from XML complexity. Java language annotations simplify development of Java EE applications. By using annotations, many applications can avoid deployment descriptors. Java EE 5 succeeded because of these changes.

Java EE 6 is another major step and includes a rich set of innovations, such as new APIs, Contexts and Dependency Injection (CDI) 1.0, WebBeans 1.0, JAX-RS1.1, and Java Servlet 3.0. The main themes of Java EE 6 are pruning and profiles. Java EE 6 begins the elimination of APIs that are outdated, not well supported, or not widely deployed to make the platform more lightweight. Small to medium range Java web applications do not need the full Java EE functionality. For example, the SOA applications that would use features like transactions, persistence, messaging, and web services, but have no need for presentation layers like JSF or JSP. Profiles are designed to address this issue. A subset of Java EE APIs is collected as a profile that is used to better organize the increasingly complex world of web services standards. The Liberty profile that is part of WebSphere Application Server 8.5 is built based on this concept.

Table 3-2 shows more module version details.

Table 3-2 Java EE version numbers

Specification/API	J2EE	Java EE 5	Java EE 6
Web	2.2	2.5	3.0
EJB	1.1 You can migrate EJB 1.1 to EJB 3 or EJB 3.1, providing that there are no CMP or BMP beans (EJB 1. x and 2. x projects with CMP or BMP beans can be left as is in a Java EE EAR).	3.0	3.1
JDBC	3.0	4.0	4.1

Specification/API	J2EE	Java EE 5	Java EE 6
JCA	1.0	1.5	1.6
JPA	n/a	2.0	2.0
JMS	1.1	1.1	1.1
Java Servlet	2.4	2.5	3.0
JSP	2.0	2.1	2.1
JSF	1.0	1.2	2.0
CDI	n/a	n/a	1.0
JAX-RPC	1.1	1.1	1.1
JAX-WS	n/a	2.1	2.2
JAX-RS	n/a	n/a	1.1
Bean Validation	n/a	n/a	1.0
SIP	n/a	1.1	1.1
EAR	1.2	5.0	6.0
Application Client	1.2	5.0	6.0

WebSphere Application Server V8.5 supports applications that are written to Java EE 6 and supports portable applications that are written for the previous Java EE versions, specifically Java EE 5, Java 2 Platform, Enterprise Edition (J2EE) 1.4, and J2EE 1.3.

If you plan to maintain your applications, consider migrating them to Java EE 6. This way, you are up to date, and the changes in the next Java EE specifications are easier to carry out.

For more information about Java EE 6 new opportunities, see *Experience JEE Using Rational Application Developer V7.5*, SG24-7827.

## Specifications level migration

**Important:** Upgrading your version of Java SE either manually or by using the Migration Toolkit does not change the Java EE specification requirements of your application. If you are migrating your application from other Java EE application servers to WebSphere Application Server, the suggested approach is to first perform the platform migration, ensure that the application runs on WebSphere Application Server without any problems, and then proceed with the Java EE specification migration. Specification migration should be done *after* the platform migration, because WebSphere Application Server is compatible with earlier versions of Java EE.

If you decide to upgrade the Java EE level of your application, you can use the Java EE specifications upgrade wizard within the Rational Application Developer. We are going to explain how this upgrade is done in this section. You can use the Migration Toolkit for Java SE level migration, not for Java EE level migration.

The Java EE upgrade wizard supports migration from J2EE 1.2, 1.3, and 1.4 specifications to the Java EE 5.0 specification level for all Java EE module types. The wizard also supports migration from Java EE 5.0 to the Java EE 6.0 specification level for all Java EE module types.



Using the Java EE specifications upgrade wizard within the Rational Application Developer V8.5, complete the following steps:

1. In the Enterprise Explorer view, right-click the project that you want to upgrade, and select **Java EE → Specifications Upgrade wizard**.
2. Follow the instructions on the Java EE Specifications Upgrade wizard Welcome Page and click **Next**.
3. In the J2EE version field, select the Java EE version level that you want to upgrade to and click **Next**.
4. Select the modules that you want to upgrade, and click **Finish** (Figure 3-8).

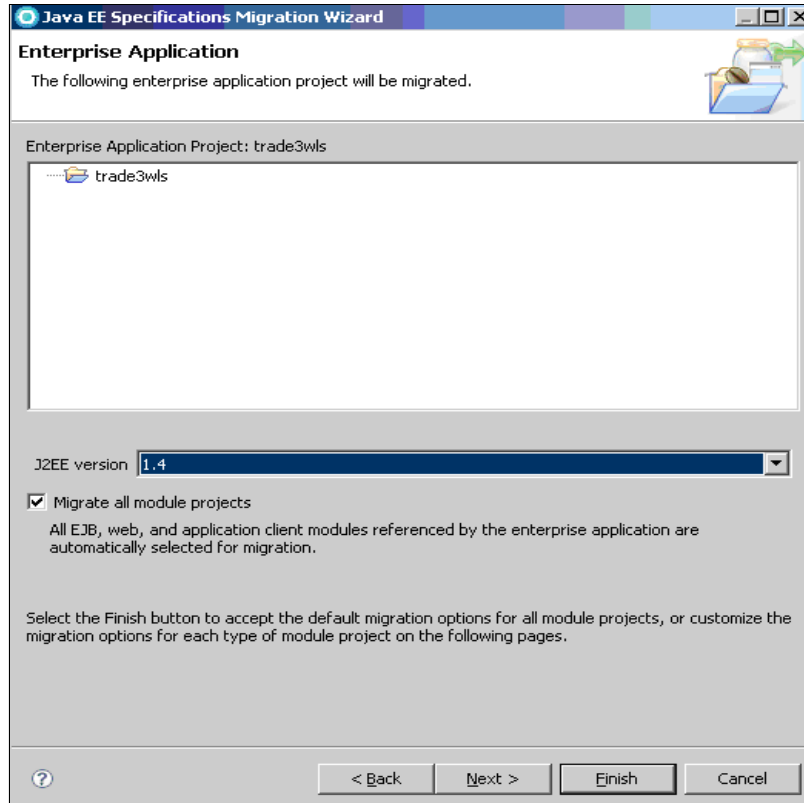


Figure 3-8 Java EE Specifications Migration wizard - J2EE Version selection

Figure 3-9 through Figure 3-12 on page 78 show the modifications done by the Rational Application Developer Jmigration process automatically. The specification level of modules before and after the migration (Figure 3-9 and Figure 3-10 on page 78) and descriptors file in the application package before and after the migration (Figure 3-11 on page 78 and Figure 3-12 on page 78) are seen within the Rational Application Developer V8.5 as follows.

Project Facet	Version
<input type="checkbox"/> Add WebSphere XDocet support	6.1
<input checked="" type="checkbox"/> EJB Module	2.1 ▼
<input type="checkbox"/> EJBDodet (XDocet)	1.2.3 ▼
<input checked="" type="checkbox"/> Java	1.4 ▼

Figure 3-9 Increases the version level of the project facet - before migration

Project Facet	Version
<input type="checkbox"/> Add WebSphere XDoclet support	6.1
<input checked="" type="checkbox"/> EJB Module	3.1 ▾
<input type="checkbox"/> EJBDoclet (XDoclet)	1.2.3 ▾
<input type="checkbox"/> iWidgets	1.1
<input checked="" type="checkbox"/> Java	1.6 ▾

Figure 3-10 Increases the version level of the project facet - after migration

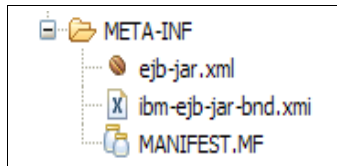


Figure 3-11 Converts .xmi files to .xml files - before migration

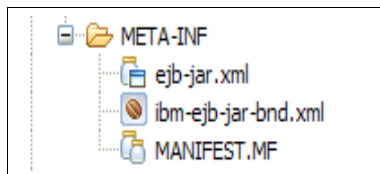


Figure 3-12 Converts .xmi files to .xml files - after migration

### 3.3.1 Java Message Service

Java Message Service messaging is a common issue when you migrate from other vendors to WebSphere Application Server V8.5. Each vendor has its own way to implement specifications and there are many approaches on how to use messaging in applications.

Changes in the JMS 1.1 should preserve compatibility with JMS 1.0.2b and simplify and unify the programming API. Check your old application to see if you can use these new JMS 1.1 functions.

Most of the issues that are related to JMS are ensuring that all resource configurations available in the old environment are set up in WebSphere Application Server V8.5. The systems administrator should take on that assignment, as he is likely receiving education about maintaining and supporting this new environment. 3.4, “Runtime migration issues” on page 82 describes more about runtime migration and resource configurations.

### 3.3.2 JavaServer Pages

JavaServer Pages (JSPs) might vary from vendor to vendor. WebSphere Application Server V8.5 is more restricted about validating DTDs and schemas and does not allow deviations (such as case-sensitive names).

Another common issue with migrating JSP pages is that a few containers have implicit importation of classes in packages, such as `java.util`, and requires updating JSPs with import statements. Otherwise, the migration is straightforward.

The Java EE 6 platform requires JSP 2.2.

### 3.3.3 Servlets

Servlets are as old as Java EE implementations, which is why no issues were raised during our migration. Because you are going to a new Java EE implementation with great improvements in code maintainability, you might not want to lose the annotations.

The usage of new annotations (such as `@servlet`) helps avoid problems for teams that face issues in updating deployment descriptors. If you do not want to maintain XDoclet, this situation is a good opportunity to get rid of it.

Here are the new Java Servlet technology features that are included with the Java EE 6 platform:

- ▶ Annotation support
- ▶ Asynchronous support
- ▶ Ease of configuration
- ▶ Enhancements to existing APIs
- ▶ Pluggability

The Java EE 6 platform requires Servlet 3.0.

### 3.3.4 JavaServer Faces

JavaServer Faces application migrations that were done during our labs did not present any issues other than the ones related to JSP pages for Java EE 6 applications. This situation does not mean that there are no issues with migrating JSF applications. Each vendor has its own implementation of JSF for Java EE 6.

In previous versions of WebSphere Application Server and with other vendors, JSF applications had to import JARs into web modules or place them in the class path, such as in the `JBoss lib` directory. For more information about how to handle those issues, see the appropriate vendor-specific chapters in this book.

Here are the new features of JavaServer Faces that are included with the Java EE 6 platform:

- ▶ The ability to use annotations instead of a configuration file to specify managed beans and other components
- ▶ Facelets, a display technology that replaces JavaServer Pages (JSP) technology using XHTML files
- ▶ Ajax support
- ▶ Composite components
- ▶ Implicit navigation

The Java EE 6 platform requires JavaServer Faces 2.0 and Expression Language 2.2.

### 3.3.5 Web services

Migrating web services involve several areas of expertise and resource skills. During our migrations, we found that there are multiple approaches for web services migrations. There are three main areas of changes to be performed:

- ▶ Development environment and migration of IDEs
- ▶ Web services clients
- ▶ Web services deployment

One of the most important considerations when you migrate a web services application is the URL you use because you might need to update references to these services (in the WSDL file).

You must consider the various approaches, depending upon your environment:

- ▶ The first approach is to update and redeploy the application clients (the consumers for your web services, represented by number 1 in Figure 3-13). For this approach, you must rewrite the application client and the references that point to the new WSDL provider. Depending on how many consumers use the service, it can take too much software development and deployment effort. If the web service is published on the Internet, this approach is not viable.

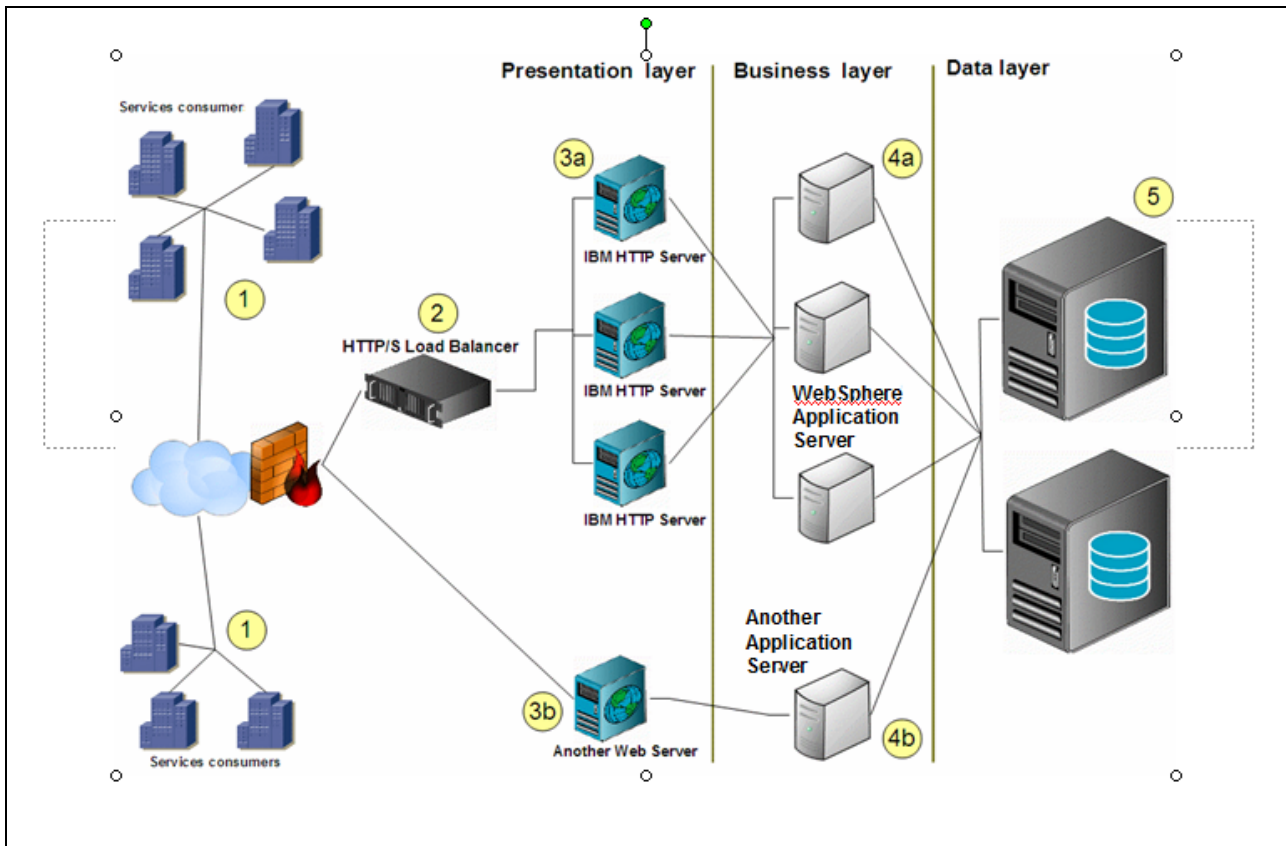


Figure 3-13 Typical infrastructure for web services applications

- ▶ If your web service is developed and installed over a three tier infrastructure, you can also make changes only on the upper tier, such as a Load Balancer, NAT, or Reverse Proxy devices (see number 2 in Figure 3-13). In this case, you must reconfigure these devices to point to the new URL for the web service.
- ▶ Looking to the first tier in your architecture (see 3a and 3b in Figure 3-13), you probably have web servers as front-end devices for static pages, and another web application and proxy for the application server itself. For example, if you use IBM HTTP Server, you can use the HTTP modules to replace the old URLs with the new ones. For more information, visit the Apache HTTP Modules (mod\_redirect, mod\_proxy, mod\_rewrite, and mod\_alias) web page at the Apache HTTP website:

<http://httpd.apache.org/docs/2.2/mod/>

- ▶ Because you are likely moving from one application server to another (layer 4 in Figure 3-13 on page 80), you should consider development issues when you build it on Rational Application Developer. If possible, try to avoid changes on the URLs (server name and server port) and path names (context root and servlet mappings). If you do not change these items, it is easier to migrate and keep connectivity with other client applications. It is not possible to keep the same URLs because you might need to move to other servers or even to another infrastructure. In these cases, you should consider the other options to remap the URLs.

### 3.3.6 Java Persistence API

Persistence is vital to enterprise applications because of the required access to relational databases. The Java Persistence API (JPA) provides a mechanism for managing persistence and object-relational mapping and functions for the EJB 3.0 and EJB 3.1 specifications. JPA standardizes the important task of object-relational mapping by using annotations or XML to map objects into one or more tables of a database. To further simplify the persistence programming model:

- ▶ The EntityManager API can persist, update, retrieve, or remove objects from a database.
- ▶ The EntityManager API and object-relational mapping metadata handle most of the database operations without requiring you to write JDBC or SQL code to maintain persistence.
- ▶ JPA provides a query language, extending the independent EJB querying language (also known as JPQL), that you can use to retrieve objects without writing SQL queries specific to the database you are working with.

There are two built-in JPA persistence providers that are included in WebSphere Application Server V8.5:

- ▶ JPA for WebSphere Application Server
- ▶ Apache OpenJPA

Apache OpenJPA is a compliant implementation of the JPA specification. Using OpenJPA as a base implementation, WebSphere Application Server employs extensions to provide more features and utilities for WebSphere Application Server customers. Because JPA for WebSphere Application Server is built from OpenJPA, all OpenJPA function, extensions, and configurations are unaffected by the WebSphere Application Server extensions. You do not need to change OpenJPA applications to use these applications in WebSphere Application Server. If an explicit provider element is not specified in the persistence unit definitions (`persistence.xml`), the application server uses the default persistence. In addition to two providers, you can also use the third-party persistence providers.

After you decide which persistence implementation best fits your needs, you should specify the provider in the persistence unit definition to avoid any incompatibilities. Depending on which provider is used, you might need to set the classloader order for your application to load the classes with the application classloader first, especially if a third-party JPA provider is defined.

To learn how to configure the JPA persistence provider, see the following topic:

[http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=tejb\\_configjpa](http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=tejb_configjpa)

## 3.4 Runtime migration issues

Runtime migration is a full-time job for migration specialists. It involves several areas, roles, and responsibilities. The following list details groups that might be interested in runtime migration:

- ▶ Support teams:
  - Operating system
  - Web middleware
  - Database
- ▶ Test teams
- ▶ Deployment specialists
- ▶ Infrastructure architects
- ▶ DNS and network teams
- ▶ Security groups
- ▶ Project managers
- ▶ Legacy systems
- ▶ Application focal points
- ▶ Others

There is a whole set of activities that must run in parallel with application migration:

- ▶ Installation
- ▶ Configuration of all hardware and software, operating systems, database products, and so on

Infrastructure architects must design solutions for the new environment. They need to decide about which topology to implement, the interfaces with other systems, compatibility checks, such as operating system level versus WebSphere Application Server V8.5 or JDBC drivers for databases, firewall rules, SOA infrastructure, and other related infrastructure components.

### 3.4.1 Migrating other products at the same time

Building a new environment is not in the scope of this book, but we do present a few considerations about this matter.

Usually, migrations of operating system levels are straightforward, such as migrating from IBM AIX® 5L™ V5.3 or AIX V6.1 to AIX V7 to meet the prerequisites of WebSphere Application Server V8.5. Other cases (such as migrating to UNIX implementations) are more challenging because of the features that might be used by the previous environment and a few commands and parameters that might be different in the new version. The more problematic migration is from Windows to UNIX and vice versa, which is a totally separate environment and requires changes in the application, resolution of education issues, and changes in the scripts that are used to automate support activities.

You have the same situation when you migrate other components (such as databases or messaging products). New drivers must be used, new features must be discovered, deprecated functions must change in both application code and resource configurations, and other features might fill gaps that were not covered on the previous environment.

You need to take into account all these factors when you migrate from one application server to another application server.

## 3.4.2 Resource definitions

Java EE containers provide implementation of specifications that developers can use to interact with databases, the systems integration bus, and web services. They also provide additional configurations to set up security for application users and groups, authentication mechanisms and providers, SSL certificates, and administrative roles and functions. The following sections present preferred practices for resource definitions during an application server migration project.

### JMS resources

A common issue in migrating JMS resources is that the way each application server configures JMS resources varies from vendor to vendor.

Before migrating JMS definition to WebSphere, you should decide which messaging engine to use to exchange messages asynchronously. WebSphere Application Server supports the following messaging providers:

- ▶ The WebSphere Application Server default messaging provider (which uses the service integration bus as the provider).
- ▶ The WebSphere MQ messaging provider (which uses your WebSphere MQ system as the provider).
- ▶ Third-party messaging providers that implement either a JCA Version 1.5 - 1.6 resource adapter or the ASF component of the JMS Version 1.1 specification.

Your applications can use the messaging resources from any of these JMS providers. In WebSphere Application Server V8.5, before you define or create the JMS resources (such as queues, topics, and connection factories), you can first create a service integration bus. A service integration bus in WebSphere Application Server V8.5 is the back end that is used by the default messaging provider.

**Listener ports:** The default messaging provider (SIB) does not use listener ports. The only applications that connect through listener ports to SIB would be using older WebSphere MQ type configurations, which are mainly for supporting WebSphere Application Server 5 type applications (the Version 5 default messaging provider) connecting through an MQClientLink. MDBs using SIB connect using activation specs, and MDBs connecting to WebSphere MQ messaging provider can connect through listener ports or activation specs. The Version 5 default messaging provider is removed from WebSphere Application Server V8.5.

For more information about messaging, including tutorials and samples, can be found at the Information Center at:

[http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=welc6tech\\_msg](http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=welc6tech_msg)

### JDBC resources

Reconfiguring the JDBC resources is probably the most common migration step between application servers. Generally, migrating from application servers might also require the migration of database products or versions, which requires JDBC drivers updates, so be sure to use the appropriate JDBC driver to avoid problems with new implementations for new databases. When you keep the database, ensure that you are using the correct driver as well.

JDBC drivers can be supplied by database vendors or by application server vendors. Use the drivers that are provided by the database vendor. Changing drivers is normally transparent to the applications, unless nonstandard extensions or driver-specific properties is not used. If applications are written to the standard JDBC (4.0), the specifications work correctly and require minimal migration effort.

In WebSphere Application Server V8.5, there are three configurations to be performed to set up a data source:

- ▶ The JDBC driver, which is usually ready to go for most databases
- ▶ J2C Authentication data, where you configure the user and password to be used to connect in the database
- ▶ The data source itself

For more information about configuring JDBC resources, see:

[http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=welc6tech\\_dat](http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=welc6tech_dat)

## Security resources

There are several areas where you configure security resources for your migrated applications and the administrative functions and roles, such as authentication mechanisms, authorization mechanisms, and security for web services.

Because this area is a sensitive one of migration that probably involves security standards that must be applied (such as ITCS 104 (Information Technology Security Standards, which are internal IBM security standards), you should see the appropriate reference materials that are available. A few basic configurations are shown in the product-specific migration chapters of this book.

For more information about security configurations and resources on WebSphere Application Server V8, see:

<http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=welc6topsecuring>

Also, see *WebSphere Application Server V7.0 Security Guide*, SG24-7660.

## JVM arguments, class path, and system properties

The runtime migration varies case-by-case because system administrators have their own way of managing configurations. A few use start script commands, others set up environmental variables in the process user ID profile, and others use specific configurations or behavior for the application servers.

Here are a few tips for you to try to avoid migration issues:

- ▶ Verify your startup scripts and check for JVM arguments.
- ▶ On UNIX machines, check the profile of the process ID for any related environmental configuration.
- ▶ On Windows machines, the user ID that you use is a non-administrator that you use to install WebSphere Application Server V8 on a Windows Vista, Windows 7, or Windows Server 2008 operating systems into the Program Files or Program Files (x86) directory with User Account Control (UAC) enabled. Otherwise, WebSphere Application Server does not function correctly.
- ▶ Open vendor-specific configuration files (such as `server.xml` in JBoss and `config.xml` in Weblogic) to check for more configurations.



- ▶ Understand how the old application server implements class path.
- ▶ Ensure that you know what JARs are extending your old application server libraries.

For more information about configuring process definitions, see *WebSphere Application Server V8: Administration and Configuration Guide*, SG24-7971.

### 3.4.3 Development environment issues

Development environments and techniques that are used to compile, assemble, and deploy applications have several combinations that vary case-by-case. We see a few patterns of these techniques, such as using vendor-specific IDEs (Rational Application Developer V8.5), using Eclipse with a different sort of plug-ins, and a combination of Maven, Apache Ant, XDoclet, and other IDEs.

WebSphere Application Server V8.5 does not depend on any Integrated Development Environment (IDE), as it is a 100% compliant application server according to Java EE specifications. During our migrations, we used two approaches:

- ▶ Keep the existing development environment and manually adjust the configuration files and Apache Ant build files, and manually replace XDoclet tags, where applicable. Depending on which Java EE components, extensions that are based on the previous application server, and specific configurations that are used by the application code and deployment descriptors, the complexity of these changes might vary from a simple to a time-consuming exercise.
- ▶ Import applications into Rational Application Developer V8.5 and use the Migration Toolkit for Oracle WebLogic, Oracle Application Server, JBoss, or Apache Tomcat. Rational Application Developer V8.5 facilitates the migration of application servers, Java EE specifications (for example, J2EE 1.4 to Java EE 6), and known migration issues from WebLogic and JBoss to WebSphere Application Server V8.5.

## 3.5 Interoperability and integrations

There are cases where we face external factors that contribute to migration complexity and require attention from all interested parties. These external factors are integrations with existing systems and interoperability between migrated applications that depend on other applications' functionality or services.

Interoperability is the ability of an application, system, or product to interact with another application, system, or product in a way that they can coexist without any special changes. For example, the application you are migrating might interface with an external CRM or ERP system through XML or web services. In this case, you must understand how the application interfaces with those external applications and determine if any change is required for the external application interfaces.

Another example is if you have IBM WebSphere Application Server V8.5 and Oracle WebLogic Vx in the same network without a firewall between them and you decide to migrate an Oracle WebLogic application to WebSphere Application Server V8.5. That application has a dependency on an EJB deployed in a second application that is hosted by the Oracle WebLogic Server. It is probably true that when you migrate the first application to WebSphere Application Server V8.5, no or minimum changes are required to have them working together. However, be aware that cross-server dependencies during migration can be challenging, as the J2EE specification leaves the implementation of the specifications up to individual vendors.

There are cases where you face problems with this approach (for example, choosing a sample application migration that is serving components or services to other applications, such as EJBs).

Migration complexity discussions require a good amount of time to decide how to move forward when there are integrations with existing systems. Integration is a sensitive topic in the middleware world, despite standards, tools, and a proven architecture that minimizes its complexity. When you manage or participate in a migration project, remember that integrations are not as easy as changing configurations or a few lines of code. Integrations require the revision of standards and compatibility. In many cases, the team should consider it a separate project.

# Migrating your applications to WebSphere Application Server V8.5

This part describes how to migrate your Oracle WebLogic Server, Oracle Application Server, Red Hat JBoss, and Apache Tomcat J2EE applications to WebSphere Application Server V8.5.

This part provides detailed information to plan migrations, suggested approaches for developing portable applications, and migration working examples for each of the platforms from which we migrated. The primary tool that is used in the migration scenarios that are covered in this book is the IBM WebSphere Application Server Migration Toolkit V3.5.





# Installation and configuration of the Application Migration Tools

This chapter provides information about the steps that are needed to install and configure the tools from the IBM WebSphere Application Server Migration Toolkit (Migration Toolkit), which guides you while you migrate your applications to IBM WebSphere Application Server V8.5. We explain the steps to be complete with both Eclipse and Rational Application Developer V8.5. To be able to choose between the two environments, we also explain the differences.

**Important:** This chapter does not explain how to install WebSphere Application Server V8.5. For information about how to install WebSphere Application Server V8.5, go to the WebSphere Application Server V8.5 Information Center, and search for the phrase “Installing and configuring your application server environment”:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/index.jsp>

This chapter describes the following topics:

- ▶ IBM WebSphere Application Server Migration Toolkit overview
- ▶ Installing the Application Migration Tools on Eclipse
- ▶ Installing the Application Migration Tools on Rational Application Developer V8.5
- ▶ Command-line installation
- ▶ Configuring the Application Migration Tools
- ▶ Troubleshooting

## 4.1 IBM WebSphere Application Server Migration Toolkit overview

Migration of an application from one platform to another normally requires much time and effort, because different platforms are developed using different ideas and concepts. Even Java itself is modified and used differently among the application servers that it runs on.

There are a series of steps that must be followed while you convert a Java EE application to make it run on an application server that is different from it was originally designed for (Figure 4-1). These steps include preliminary assessment and planning of migration, and modification of Java classes, XML files, deployment descriptors, JavaServer Pages (JSP) files, and so on. Thus, it is obvious that the migration process is a risky and costly operation.

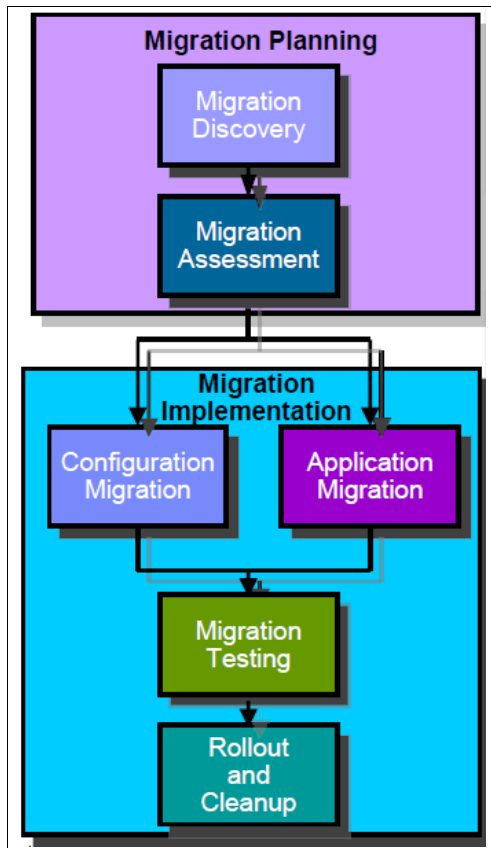


Figure 4-1 Steps for planning and implementation for migrating an application

As you can see in Figure 4-2, application changes and testing is 42% of the whole migration process.

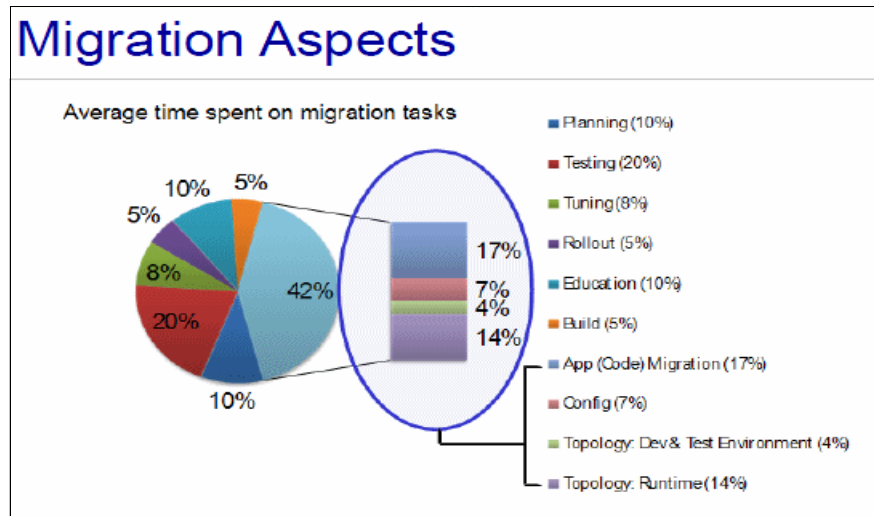


Figure 4-2 Migration aspects

The Migration Toolkit is a set of plug-ins for Eclipse and Rational Application Developer products, which reduces the cost and duration of application migration processes. It includes several migration tools, each of which can be used for migrating applications from different application servers. The Migration Toolkit includes the following migration tools:

- ▶ Application Migration Tool - WebLogic to WebSphere (see “Application Migration Tool - WebLogic to WebSphere” on page 133)
- ▶ Application Migration Tool - Oracle AS to WebSphere (see “Application Migration Tool - Oracle AS to WebSphere” on page 222)
- ▶ Application Migration Tool - JBoss AS to WebSphere (see “Application Migration Tool - JBoss AS to WebSphere” on page 236)
- ▶ Application Migration Tool - Apache Tomcat to WebSphere (see “Application Migration Tool - Apache Tomcat to WebSphere” on page 275)
- ▶ Application Migration Tool - WebSphere Version to Version (see “Application Migration Tool - WebSphere Version to Version overview” on page 358)

These tools are based on IBM Rational Software Analyzer, which has code scanning capabilities that help identify code constructs that must be reviewed and possibly changed while you migrate an application.

The process of converting Java applications can involve modifying Java source code, JSP files, deployment descriptors and XML files, and class path information, which can be a costly and time-consuming process. The tools also have rules for Java SE version migration that identify migration issues that can occur when you change the version of the Java run time. There are additional rules to improve code regarding Java coding preferred practices.

The tool flags a number of known differences between applications that are hosted on Oracle WebLogic, JBoss, or Apache Tomcat and WebSphere Application Server. In many cases, the tool can automatically convert these key differences. If the tool is unable to perform the fix, it flags the file in question to provide insight as to where changes are needed and how to perform a manual fix.

In summary, the Migration Toolkit is useful in migrating applications to WebSphere Application Server because it simplifies the migration process and speeds time to value for developers.

**Tip:** The IBM Education Assistant includes a multimedia educational module at no additional cost for the WebSphere Application Server Migration Toolkit. For more information, see:

<http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/topic/com.ibm.iea.wasmt/wasmt/3.5/Overview.html>

### 4.1.1 Migration Toolkit basics

The application migration tools in the Migration Toolkit have predefined rules and rule sets, each specific to the tool's target application server (Figure 4-3). After you create an analysis configuration for an application, run an analysis to start scanning the application based on these rules, to discover parts of the application that match these rules. The tool then lists the changes that are required to migrate the application. Many of the rules have a self-correcting capability, which means that the migration tool can fix the issue automatically. For the rules that the tool cannot fix, it provides detailed help that describes the required changes that need to be implemented.

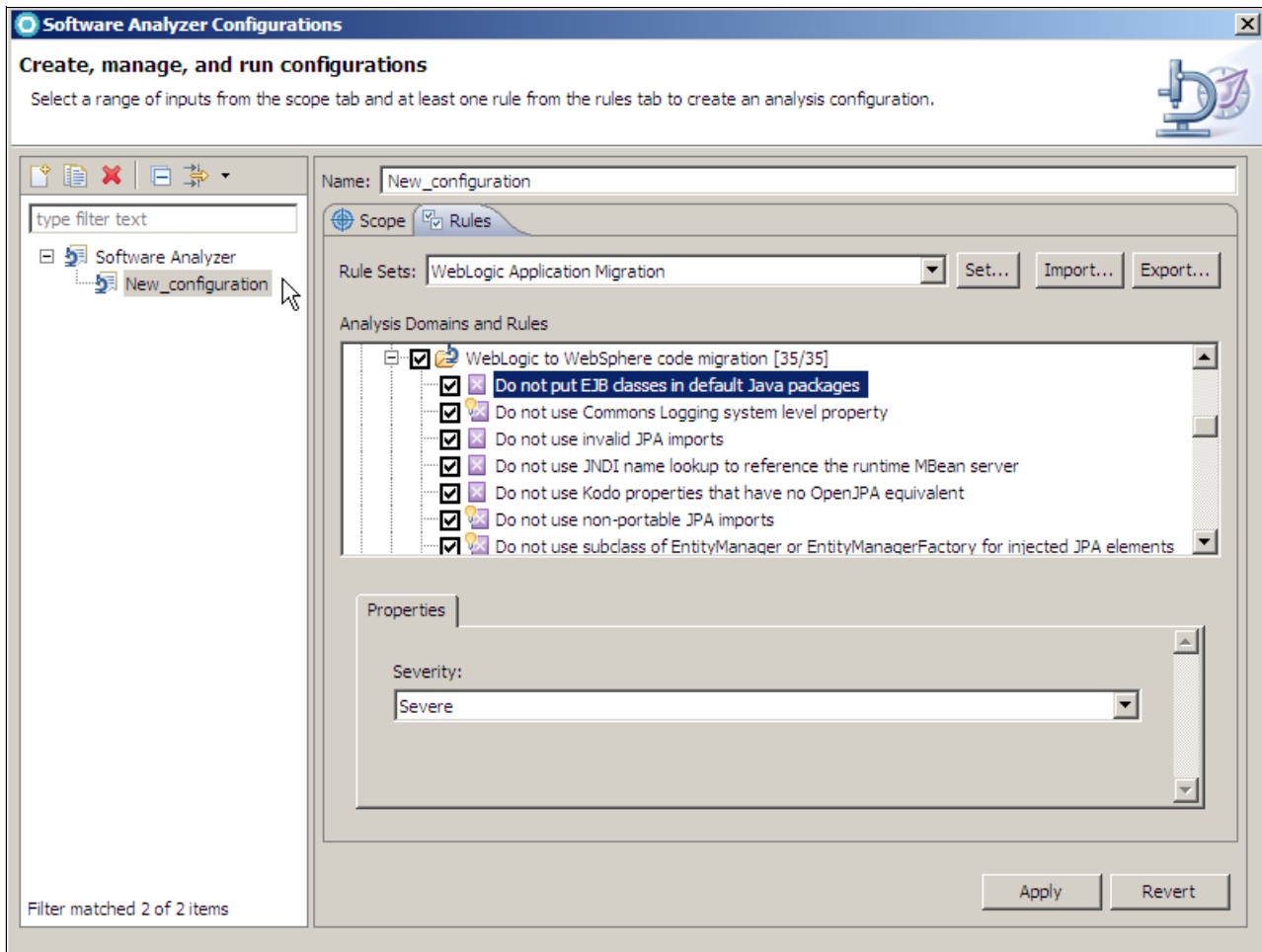


Figure 4-3 Example configuration for migrating from WebLogic to WebSphere



After the changes are done, you should see no red Xs (Figure 4-4) next to the name of the application in Enterprise Explorer, which is a sign that there should be no errors that prevent the application from running on the target application server.

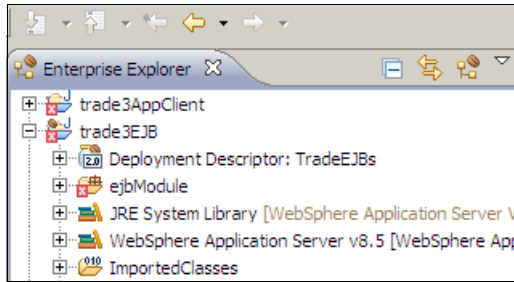


Figure 4-4 Modules with red Xs indicating that there are one or more errors

In Figure 4-5, you can see the workflow of migrating a competitive application server application (or a previous version of WebSphere Application Server) to a WebSphere Application Server V8.5 application.

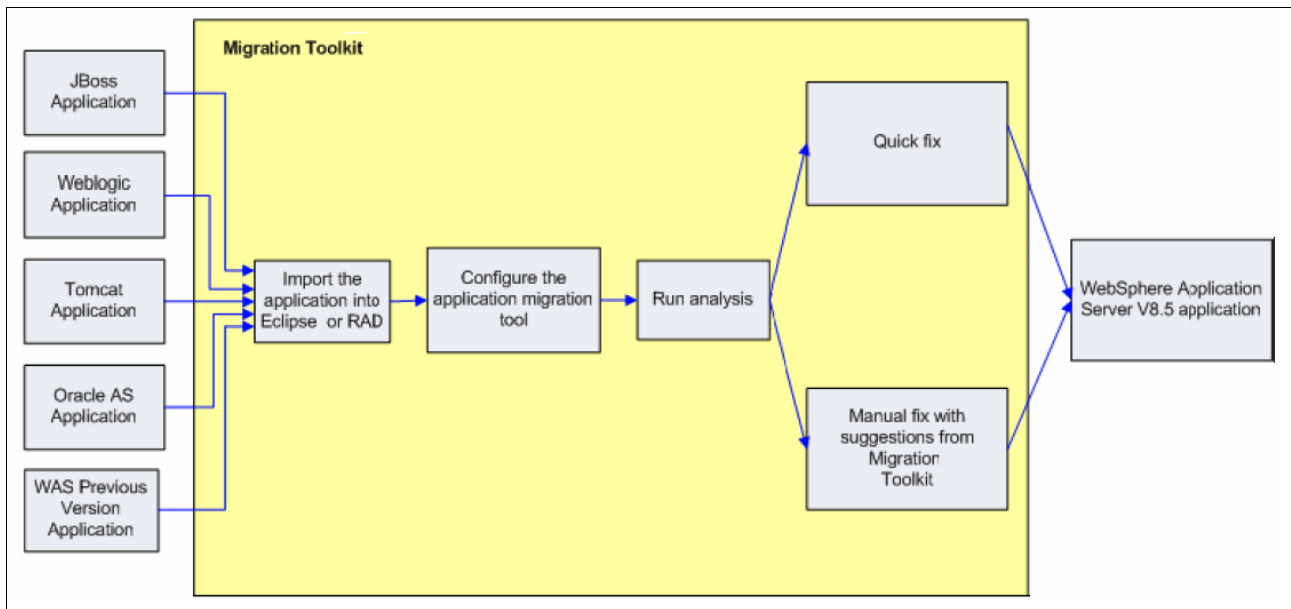


Figure 4-5 A high-level flow of how the Migration Toolkit works

The Migration Toolkit is based on the following concepts:

- ▶ *Rules*, which are known issues that might prevent the application from running on the target application server. The Migration Toolkit scans the applications using these rules.
- ▶ *Rule sets*, which are sets of rules that are grouped to accomplish a specific task. For example, rule sets are provided for each type of competitive application server migration and to do more general tasks, such as identify violations in Java preferred practices. These rule sets can be imported and exported between different development environments.
- ▶ *Code Reviews*, which organize rules by the type of documents that are scanned, such as Java, JSPs, XMLs, and Manifest (.MF) files.
- ▶ *Quick Fixes*, which are actions that are taken to fix the migration issue in the application. You can choose to quick fix all the issues, a single issue, or all issues that are placed in a single category.

You can use the Migration Toolkit to create multiple configurations, which eliminate the requirement of deleting and re-creating configurations each time you plan a migration analysis for different application servers or migration strategies.

The Migration Toolkit can also generate reports for you to better estimate the scope of the migration of your applications. The reports summarize your past analysis records, and provide a high-level output of the application code's quality, complexity, and structure. You can generate an analysis report after you run an analysis by clicking the icon in the right pane in the Software Analyzer Results window (Figure 4-6). You can also generate reports for your historical scans.

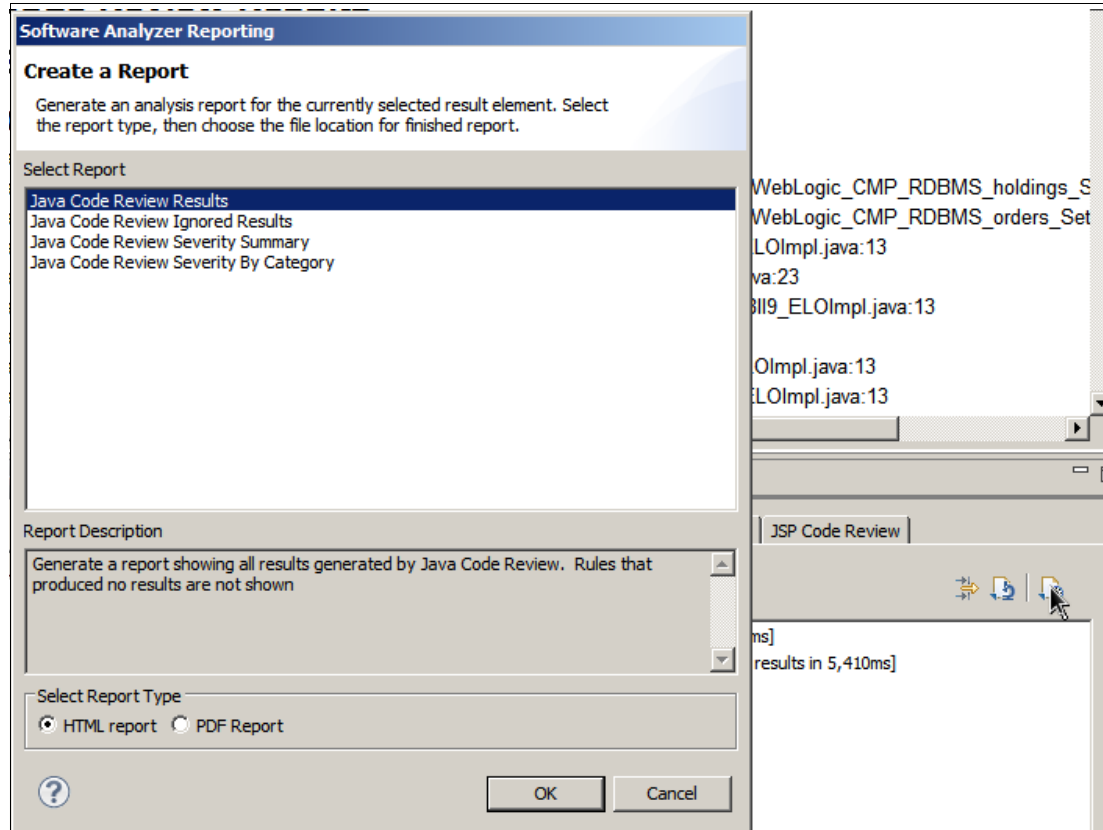


Figure 4-6 Software Analyzer Reporting pane for creating analysis reports

For Java code reviews, you can create the analysis report as PDF or HTML formats. You see an HTML report example in Figure 4-7. For Java, XML, JSP, and class path code reviews, you can export the analysis records in XML format.

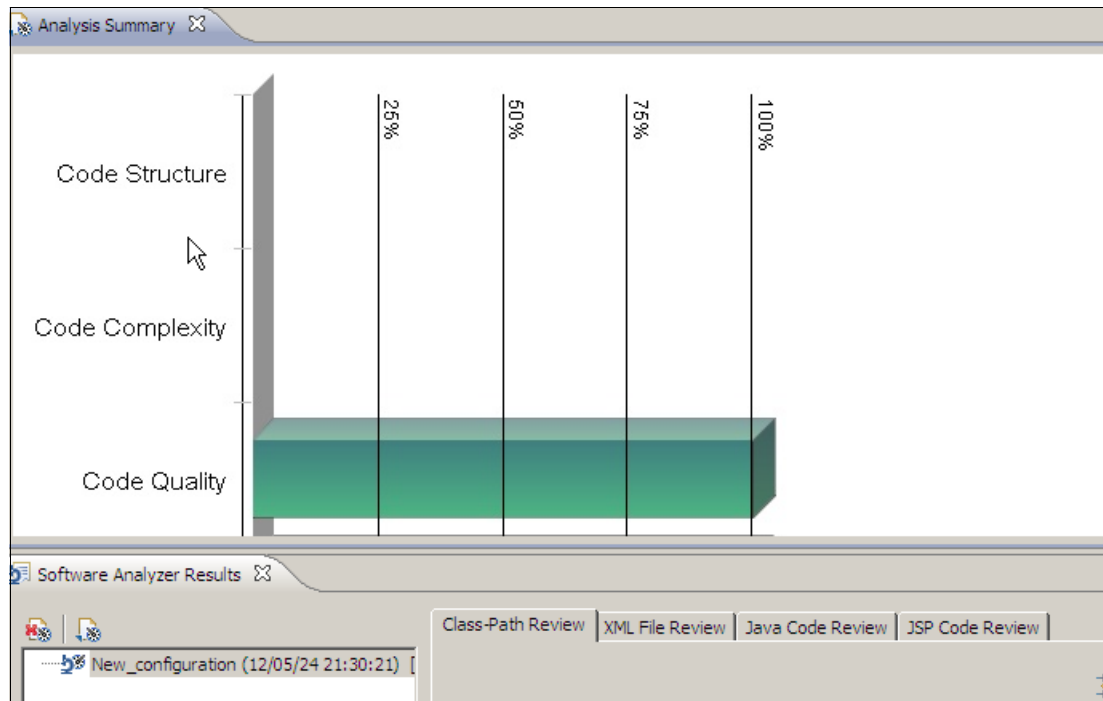


Figure 4-7 Sample HTML report

At the time of the writing of this book, the latest version of the Migration Toolkit is V3.5, and it currently supports the following migration operations:

- ▶ Migrating applications to WebSphere Application Server Version 7, 8, or 8.5
- ▶ Migrating WebLogic Java, JSP, and class path artifacts (Java EE 5 and earlier versions)
- ▶ Migrating WebLogic deployment descriptors (Java EE 5 and earlier versions)
- ▶ Migrating JBoss Java and class path artifacts (Java EE 5 and earlier versions)
- ▶ Migrating JBoss deployment descriptors (Java EE 5 and earlier versions)
- ▶ Migrating OracleAS Java and JSP artifacts (Java EE 5 and earlier versions)
- ▶ Migrating OracleAS deployment descriptors (Java EE 5 and earlier versions)
- ▶ Migrating Apache Tomcat Java and JSP artifacts (Java EE 5 and earlier versions)
- ▶ Migrating Apache Tomcat Context XML information that is contained in the application
- ▶ Providing assistance for migrating applications using third-party frameworks (Spring, Hibernate, and Seam)
- ▶ Migrating applications to later Java SE run times, from Java SE 1.4, 5, or 6 to either Java SE 6 or 7

The *Migration Toolkit* is publicly available at no additional cost, with detailed documentation to ease the migration of Java EE applications from various application servers to WebSphere Application Server. Support for the Migration Toolkit is also included for WebSphere Application Server customers.

In summary, knowledge-based migration assistance that is provided by the Migration Toolkit improves the reliability of the migration. No more wading through documentation and source code or performing trial-and-error tests to determine what breaks when you migrate your application.

## Getting help

You can use a context-sensitive function while you work with the Migration Toolkit. When you work on either Eclipse or Rational Application Developer, press F1 to view help topics about the current issue (Figure 4-8). You can also see the Help contents by clicking **Help** → **Dynamic Help**.

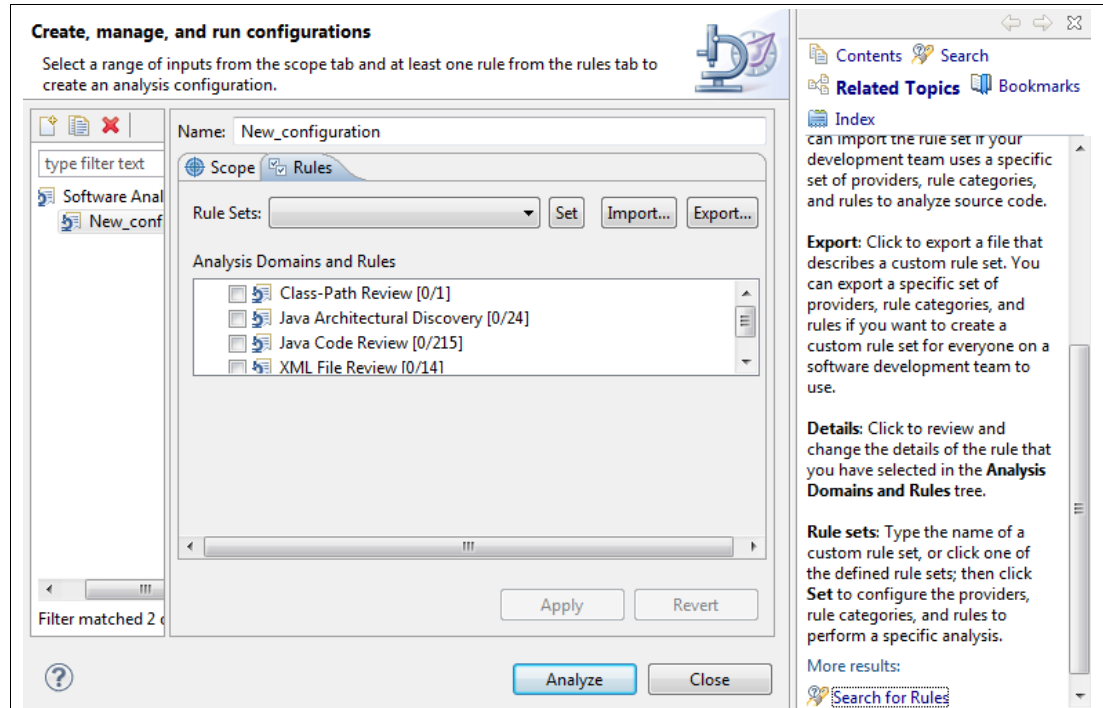


Figure 4-8 Help view

To get online information about the Migration Toolkit, visit:

<http://www.ibm.com/developerworks/websphere/downloads/migtoolkit/index.html>

The Migration Toolkit forum is available to provide input and to answer questions:

<http://www.ibm.com/developerworks/forums/forum.jspa?forumID=2106>

## 4.2 Installing the Application Migration Tools on Eclipse

Each of the Application Migration Tools are installed as Eclipse features. Here we describe the steps for installing the Application Migration Tools into the Eclipse environment.

**Tip:** Uninstall Versions 1.x, 2.0, or 3.5 Beta of the Migration Tools before you install Version 3.5. Versions 2.1, 3.0, and 3.1 can be upgraded to Version 3.5.

First, you should download the Application Migration Tool that is suitable for the application to be migrated. We selected WebLogic to WebSphere Application Migration Tool for our example. For more information about the analysis phase, see the corresponding chapters in this book. The migration steps for each application server are explained in detail in its own chapter.

You can download and install the Application Migration Tools by completing the following steps:

1. Download the Application Migration Tool suitable for the application server to be migrated from. We used the Application Migration Tool - WebLogic to WebSphere for our example. The Application Migration Tools and documentation can be downloaded at:

<http://www.ibm.com/developerworks/websphere/downloads/migtoolkit/compmig.html>

2. From the menu bar, click **Help** → **Install New Software**.
3. In the Install window, click **Add**.
4. In the Add Site window, enter the following information (Figure 9-6):
  - Name: Enter an appropriate name for the tool, for example, Application Migration Tool – WebLogic to WebSphere
  - Location: Enter the location of the compressed file that you downloaded, for example, C:\MigrationToolkit\Application\_Migration\_Tool\_WebLogic\_to\_WebSphere\_v3.5.0.zip.

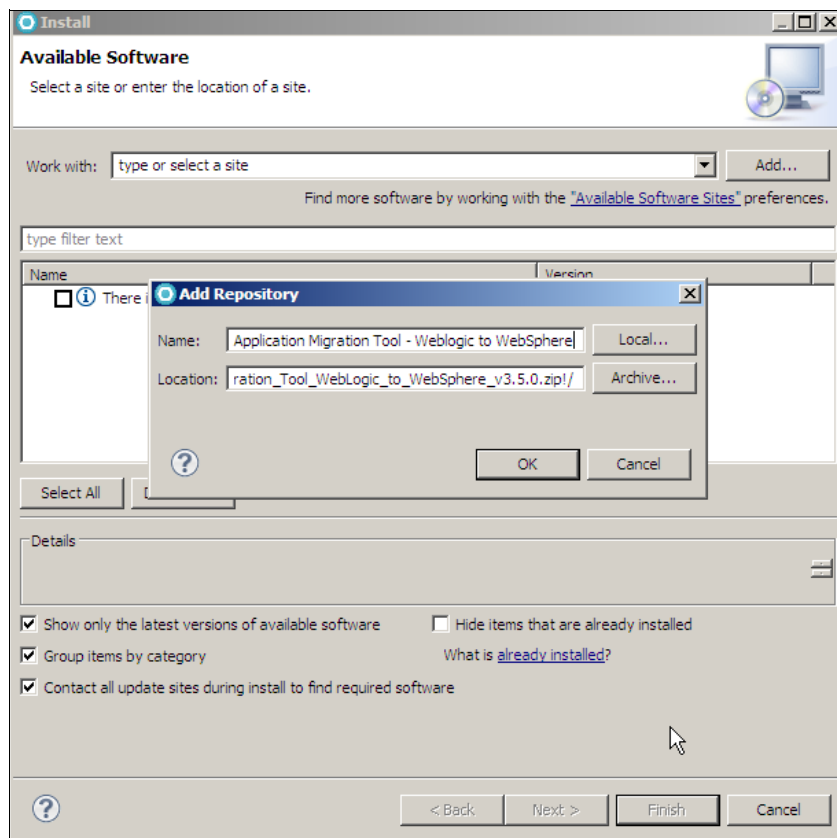


Figure 4-9 Add Repository window for adding the Application Migration Tool as a repository

5. In the Install window, select the plug-in that you want to install (Figure 9-7). Click **Next**.

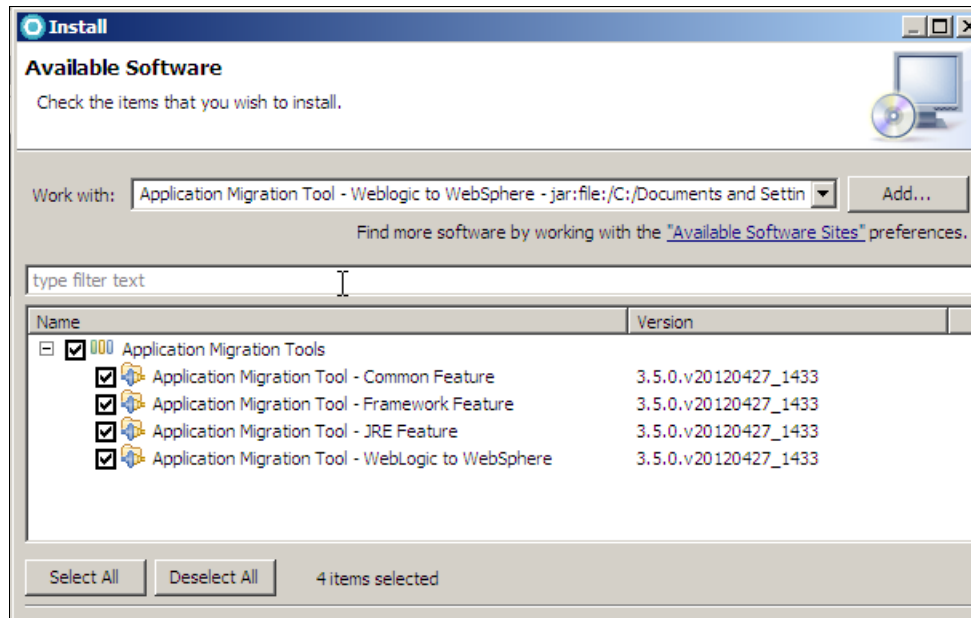


Figure 4-10 Plug-in selection in the Install window

6. Accept the license agreement and click **Finish** to begin the installation.
7. After the installation is completed, click **Restart Now** to restart Rational Application Developer.

The Application Migration Tool is now installed and ready to be used. You can analyze your applications by clicking **Run** → **Analysis** and creating the appropriate run configuration. You can find details about the analysis phase in the chapters that are related to the application server you want to migrate from.

## 4.3 Installing the Application Migration Tools on Rational Application Developer V8.5

This section describes the steps to install the Application Migration Tools on Rational Application Developer V8.5. Complete the following steps:

1. Download the Application Migration Tool suitable for the application server to be migrated from. We used the Application Migration Tool - WebLogic to WebSphere for our example. The Application Migration Tools and documentation can be downloaded at:

<http://www.ibm.com/developerworks/websphere/downloads/migtoolkit/compmig.html>

2. Rational Application Developer should automatically discover the local WebSphere Application Server V8.5 at startup after it is installed (Figure 4-11). Check if the information is correct and click **Finish**. You can find your created server in the **Servers** tab.

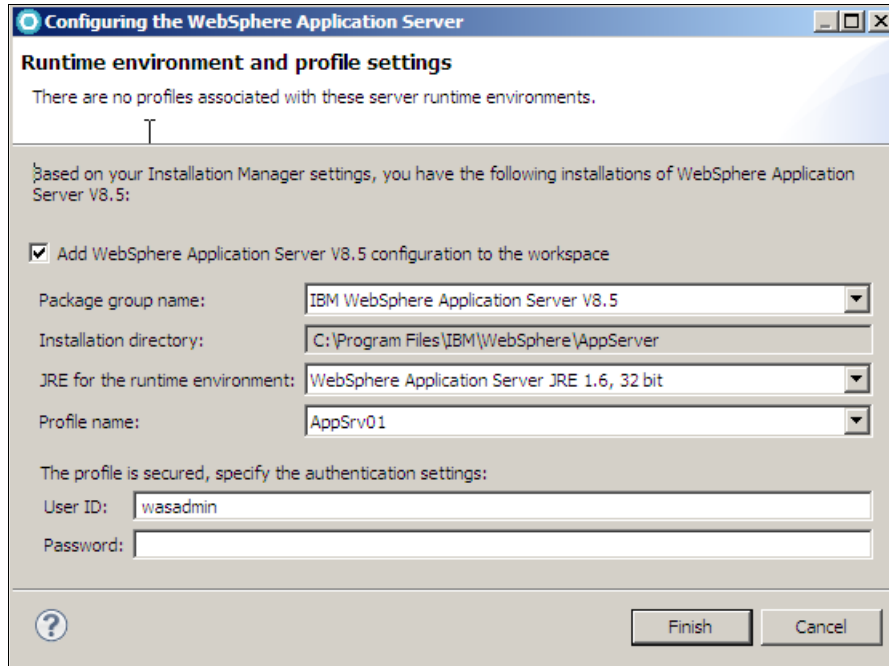


Figure 4-11 Application server settings

3. If Rational Application Developer does not discover your server, you can create it manually, as described in "Installation verification" on page 115.
4. For the rest of the installation, you can follow the steps that are described in 4.2, "Installing the Application Migration Tools on Eclipse" on page 96, starting with step 2 on page 97.

**Installing the Application Migration Tools:** The steps for installing the Application Migration Tools on Eclipse and Rational Application Developer are identical, except for the discovery step (step 2 in 4.3, "Installing the Application Migration Tools on Rational Application Developer V8.5" on page 98).

## 4.4 Command-line installation

The Application Migration Tools can also be installed by using the command line. You should download the appropriate application migration tool first, then you should run the commands in Example 4-1.

*Example 4-1 Installing the Migration Toolkit through the command line*

```
<installLocation>\eclipse.exe
-application org.eclipse.equinox.p2.director -metadataRepository
jar:file:C:/<amtDownloadDir>/Application_Migration_Tool_Weblogic_to_WebSphere_vn.n
.n.zip!/ -artifactRepository
jar:file:C:/<amtDownloadDir>/Application_Migration_Tool_Weblogic_to_WebSphere_vn.n
```

```
.n.zip!/ -installIU com.ibm.ws.appconversion_feature.weblogic.feature.group  
-nosplash
```

---

Replace n.n.n in the repository arguments with the version number, such as 3.5.0. The arguments for the metadataRepository and artifactRepository start with jar:file: and end with !/. The argument for installIU ends with an appended .feature.group.

In our example, one of application migration tools, WebLogic to WebSphere migration, is chosen. You can select the appropriate installIU type from Table 4-1 for another Application Migration Tool.

Table 4-1 Application Migration Tools and installIU

Application Migration Tools	installIU
JBossAS	com.ibm.ws.appconversion_feature.jboss.feature.group
Oracle AS	com.ibm.ws.appconversion_feature.oracle.feature.group
WebLogic	com.ibm.ws.appconversion_feature.weblogic.feature.group
WebSphere Version to Version	com.ibm.ws.appconversion_feature.was2was.feature.group
Apache Tomcat	com.ibm.ws.appconversion_feature.tomcat.feature.group

The Application Migration Tools can be removed by using the appropriate installIU parameter the same way you installed it (Example 4-2).

Example 4-2 Uninstalling the Application Migration Tools through the command line

```
<installLocation>\eclipse.exe  
-application org.eclipse.equinox.p2.director  
-uninstallIU com.ibm.ws.appconversion_feature.weblogic.feature.group  
-nosplash
```

---

For more information about the Eclipse p2 director application, see:

[http://help.eclipse.org/indigo/index.jsp?topic=/org.eclipse.platform.doc.isv/guide/p2\\_director.html](http://help.eclipse.org/indigo/index.jsp?topic=/org.eclipse.platform.doc.isv/guide/p2_director.html)

## 4.5 Configuring the Application Migration Tools

After any of the Application Migration Tools is installed on Eclipse or Rational Application Developer, there are new analysis options to configure and run an analysis. The scope to analyze can be configured to a project, a working set, or the entire workspace. The tool can be configured to define a set of rules to run that are known issues that prevent the application from running on the target application server.



## 4.5.1 Running the analysis

The following section describes different ways to run the analysis using the Application Migration Tool.

**Tool example:** For the examples in this section, we choose the Application Migration Tool - WebLogic to WebSphere, but these examples apply to other Application Migration Tools as well.

To start the analysis from the Eclipse menu, click **Run** → **Analysis** from the Eclipse menu (Figure 4-12).

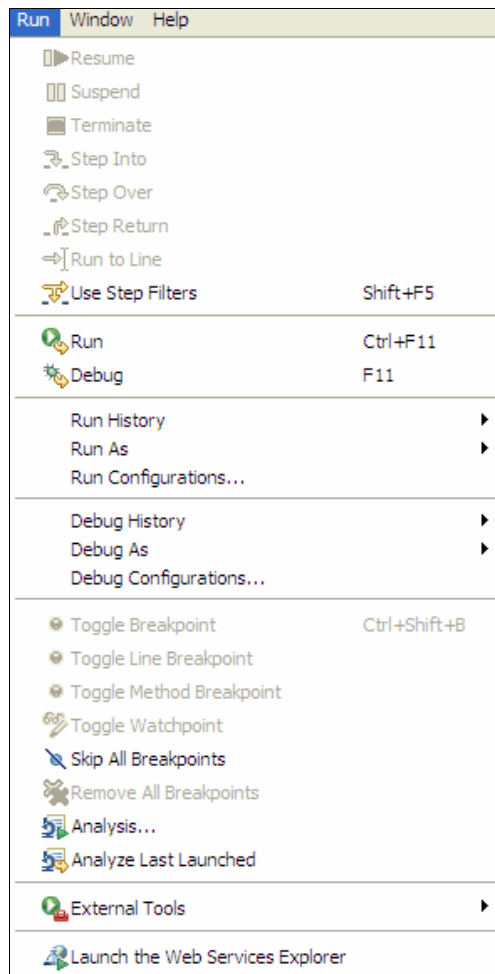


Figure 4-12 Analysis option

To start the analysis from the Launch toolbar, look for the icon that is shown in Figure 4-13 and click it.

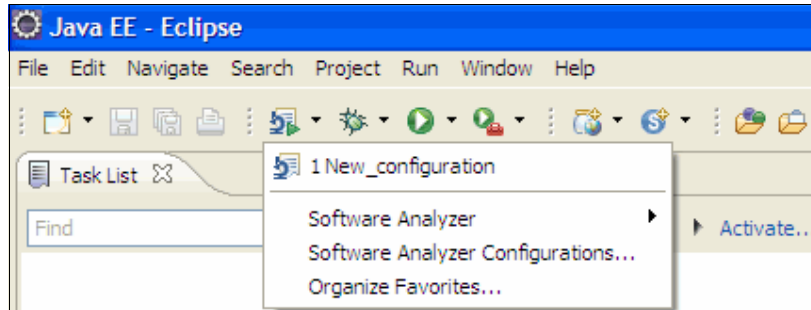


Figure 4-13 Software Analyzer

To start the analysis from the Enterprise Explorer in Eclipse, right-click the project that should be analyzed (Figure 4-14).

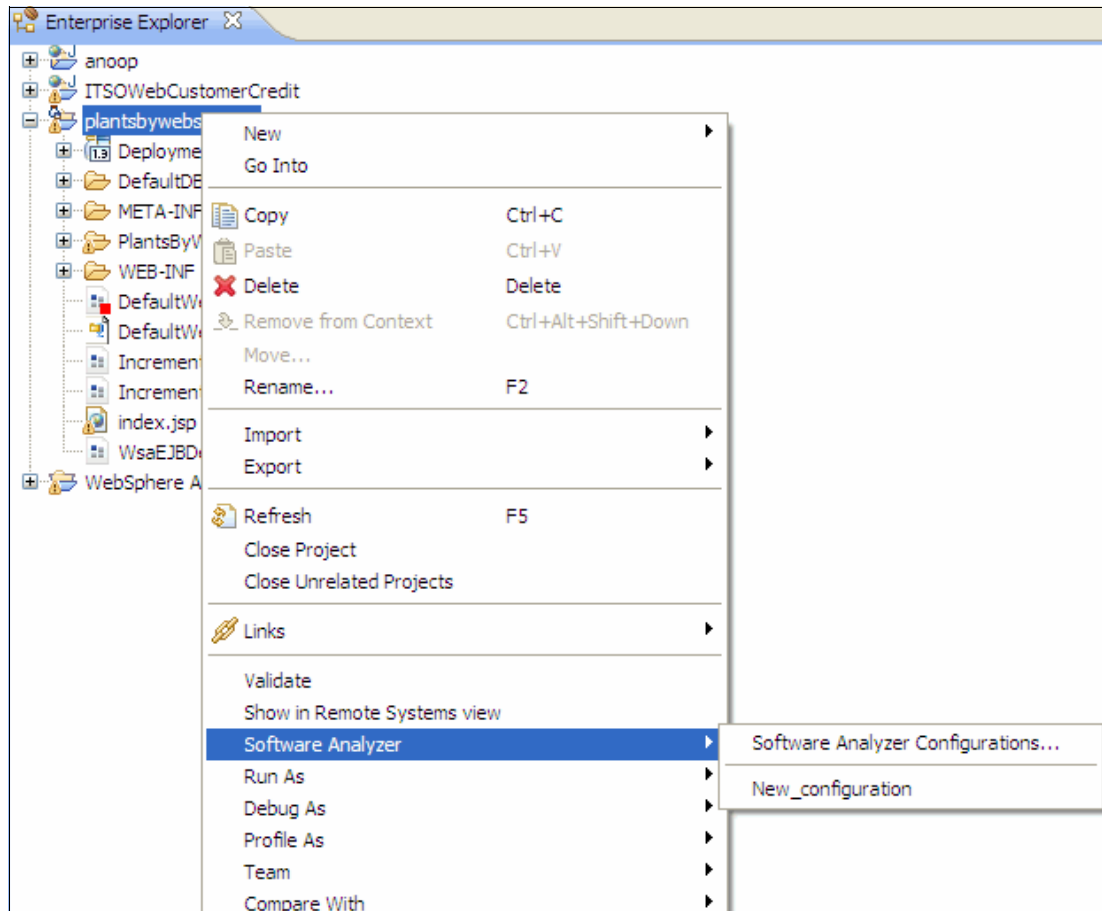


Figure 4-14 Software Analyzer

## 4.5.2 Configuring and running the analysis

After you start the analysis using one of the methods that are described in 4.5.1, “Running the analysis” on page 101, complete the following steps to configure a new analysis:

1. Right-click the project and select **Software Analyzer**.

2. Select **Software Analyzer Configurations...**
3. Right-click the **Software Analyzer** on the left and select **New** (Figure 4-15).

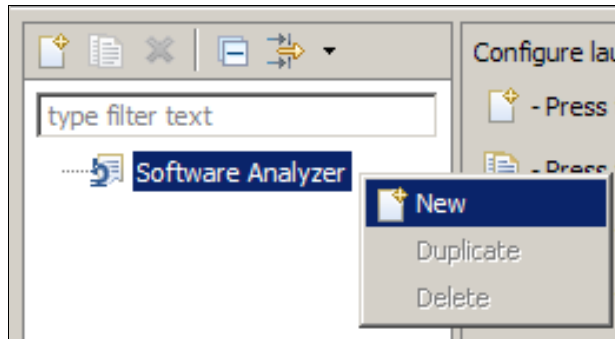


Figure 4-15 Creating a Software Analyzer configuration

You see the Software Analyzer Configurations window (Figure 4-16).

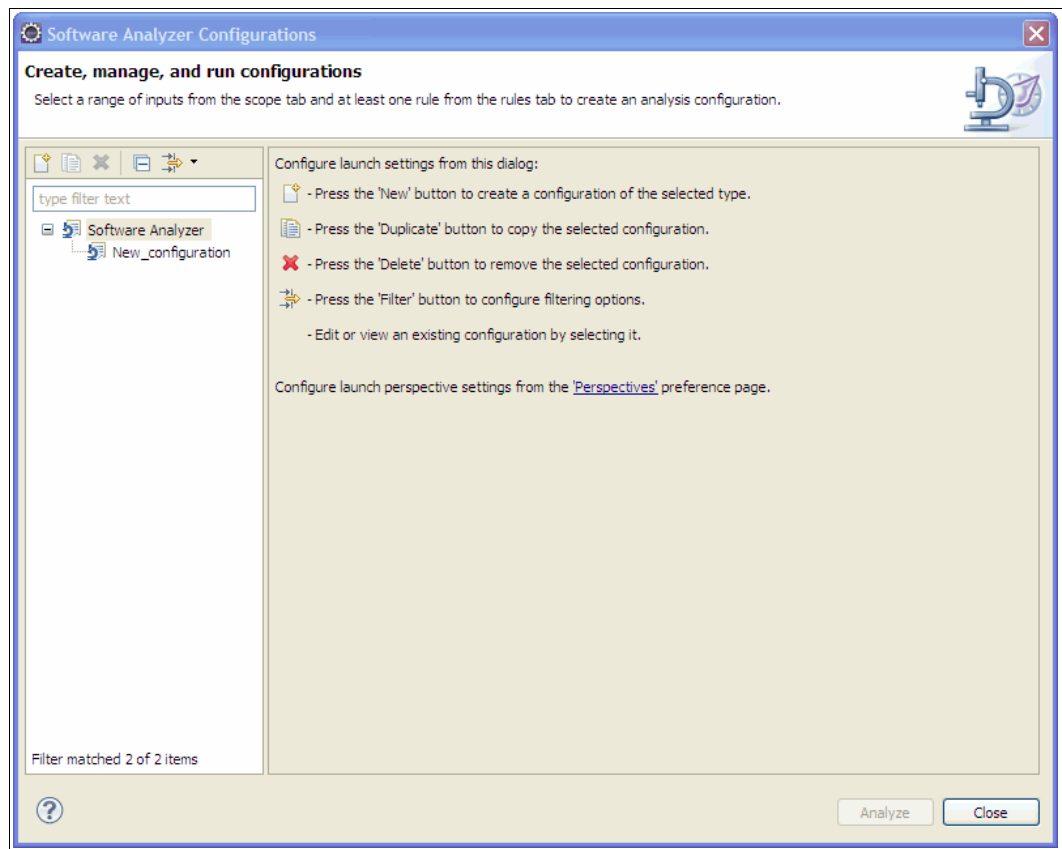


Figure 4-16 Software Analyzer Configurations

4. Click **New Configuration** to see the dialog changes to the basic configuration interface (Figure 4-17).

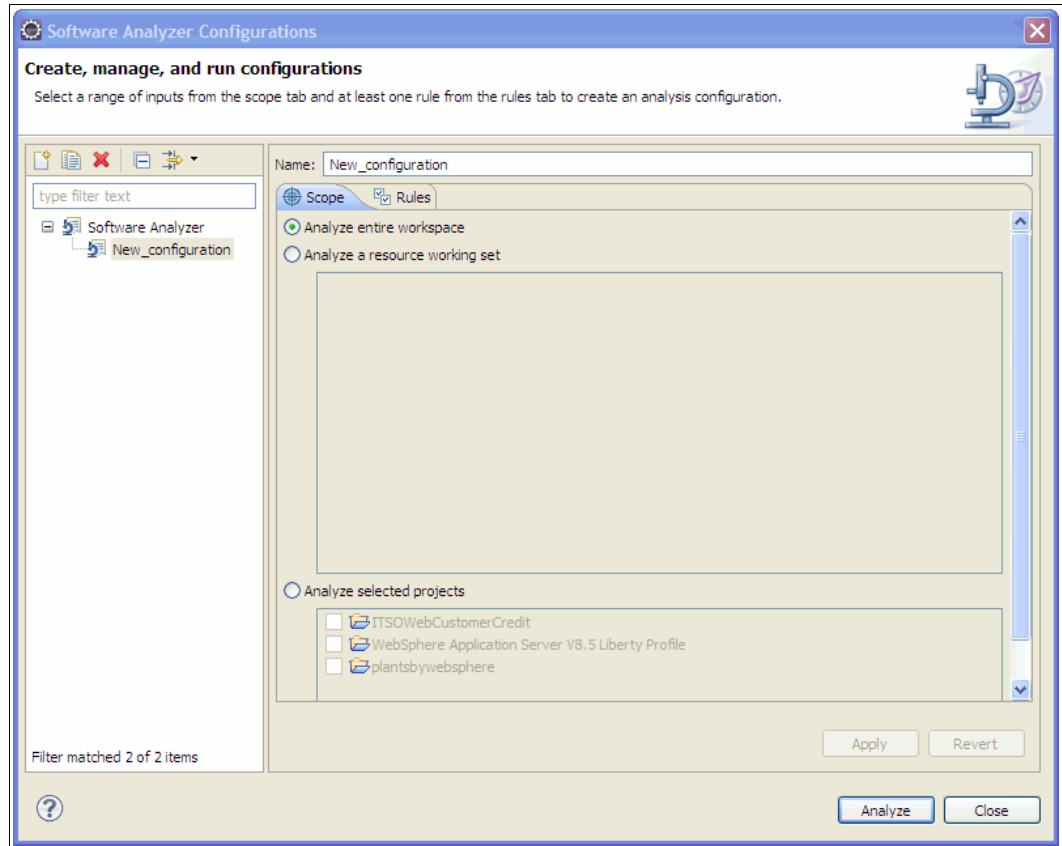


Figure 4-17 New configuration

In the Software Analyzer Configuration dialog, enter a name for the configuration, such as “SampleAppMigration”. On the Scope tab, click **Analyze entire workspace** to scan all the projects in the workspace. The scope of an analysis can be limited by using the other options on this dialog to analyze a working set or a selection of projects.

**Tip:** When you use the Enterprise Explorer menu to run your analysis, the scope of the analysis is limited to the node in the project where the menu item was selected. You can perform a quick analysis on a limited set of code.

5. On the Rules tab, select the type of analysis to perform by using the Rule Sets list. From the Rule Set menu, select **WebLogic Application Migration**, for WebLogic to WebSphere Application Server migration.

The rule set contains rules in the following analysis domains:

- Java Code Review: Examines the Java code to determine overall quality. The rules in this domain can range from simple code style enforcement to more complex rules used to detect null pointer issues.

**Tip:** Not all rules apply to all environments because some rules can generate many results. In these situations, avoid running these rules or ignore the results that they produce.

- JSP Code Review: This category contains rules that are related to JSP code.
- XML File Review: This category contains rules that are related to XML and associated files.
- Class-Path Review: This category contains rules that are related to class path issues with included libraries.

**Tip:** To obtain more information about a rule, highlight the rule and press F1. Help for the rule is shown in the configuration dialog. The initial help dialog includes a short description and a link to more information.

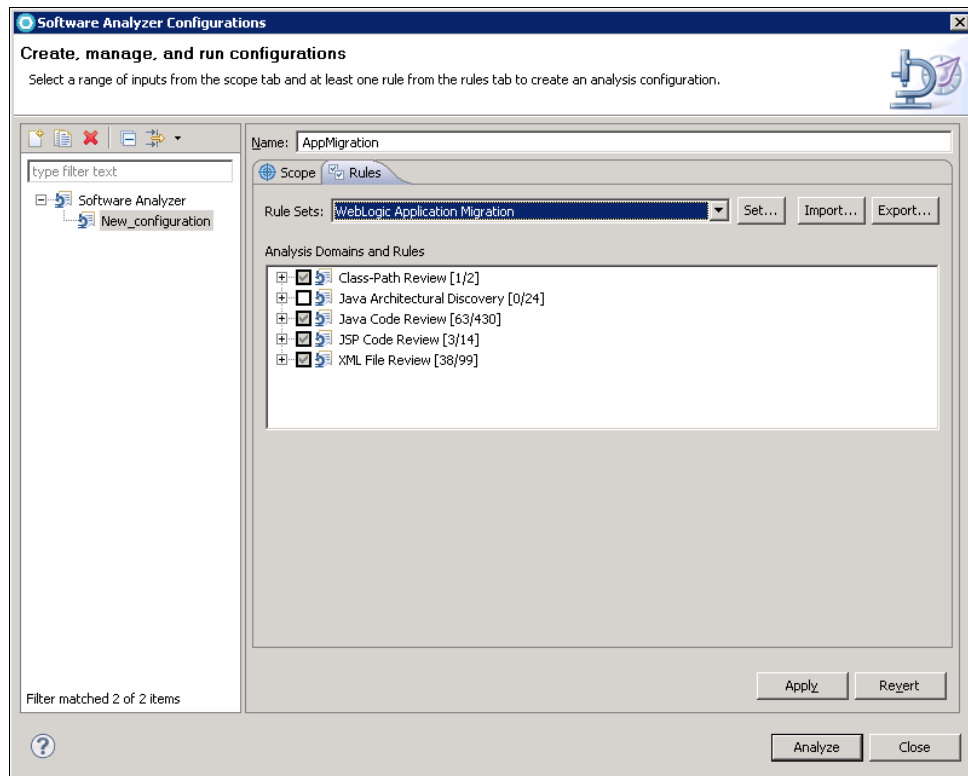


Figure 4-18 WebLogic Application Migration rules

- After the WebLogic Application Migration rule set is selected, click **Set** to open the dialog that is used to configure the version of WebSphere you are migrating to and the version of Java you were using in WebLogic (Figure 4-19).

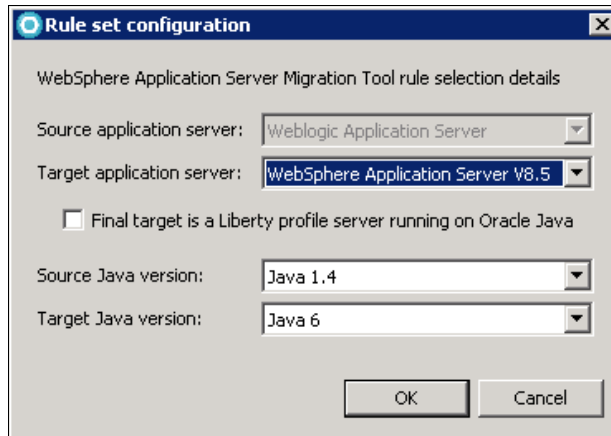


Figure 4-19 Rule set configuration

The target WebSphere Application Server version can be either Version 7, Version 8, or Version 8.5. Also, select the source Java version that was being used. If the target WebSphere platform is Version 8.5, both Java V6 or Java V7 can be selected.

**Number of rules:** The number of rules in the analysis configuration dialog varies depending on the platform on which the tool is installed.

- Click **Apply** to save the rule configuration and click **Analyze**. The results are shown in the Software Analysis Results view (Figure 4-20).

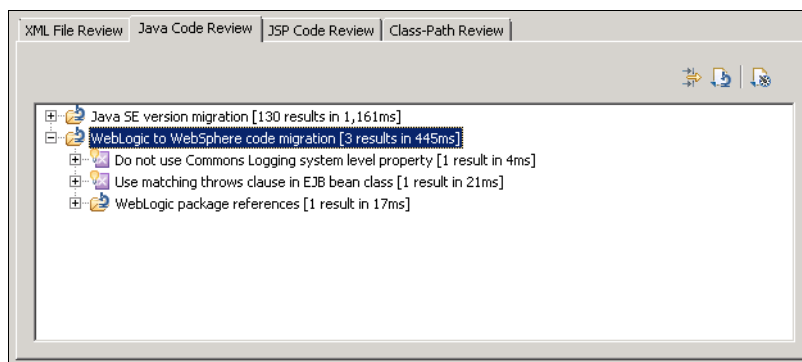


Figure 4-20 Software Analysis Results view

Right-clicking individual results gives access to options, such as viewing the source code where the problem occurred or correcting the problem with a provided fix.

**No results:** If no results are shown in the window, no issues were identified while scanning.

Possible actions on the results are:

- ▶ **View Result:** Opens an editor that shows the source file that triggered the rule. The cause of the problem is highlighted, and a rule violation icon is shown in the left margin of the editor.
- ▶ **Quick Fix:** Selecting this option runs the conversion that modifies the affected Java code, XML file, or JSP or manifest file, allowing it to run in WebSphere Application Server. The quick fix might change the code directly or it might present the steps that are needed to complete the fix.
- ▶ **Quick Fix Preview:** This option is available for the rules that support showing a side-by-side comparison of the original code and the code after the quick fix is applied. You can use this option to see the changes before they are made.
- ▶ **Ignore Result:** This option removes the rule from the list without making a code change. For Java files, a comment annotation is added to the file so that the rule is not triggered on future analysis runs.
- ▶ **Quick Fix All:** This option resolves all issues that are identified for a rule.
- ▶ **Quick Fix All Category:** This option runs all quick fixes that are identified for the category to which the rule belongs. A rule must have Quick Fix All enabled for the Quick Fix All Category option to run its quick fix. For example, if you choose this option on a Java rule, the quick fixes for all Java rules that have the Quick Fix All option are run.

### 4.5.3 Generating reports

The Application Migration Tools can also generate reports for you to make better estimations about the scope of the migration of your applications. The reports summarize your past analysis records, and give a high-level output of the application code's quality, complexity, and structure. You can generate an analysis report after you run an analysis by clicking the icon (highlighted with a red circle in Figure 4-21) on the right in the Software Analyzer Results window.

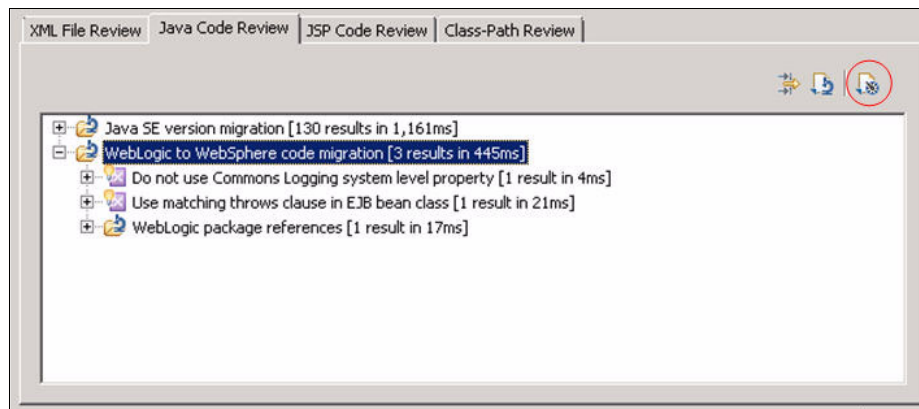


Figure 4-21 Generating reports

After you click the icon, an analysis report for the currently selected result element can be generated. For example, for Java Code Review, four options, shown in Figure 4-22, are available:

- ▶ Java Code Review Results
- ▶ Java Code Review Ignore Results
- ▶ Java Code Review Severity Summary
- ▶ Java Code Review By Category

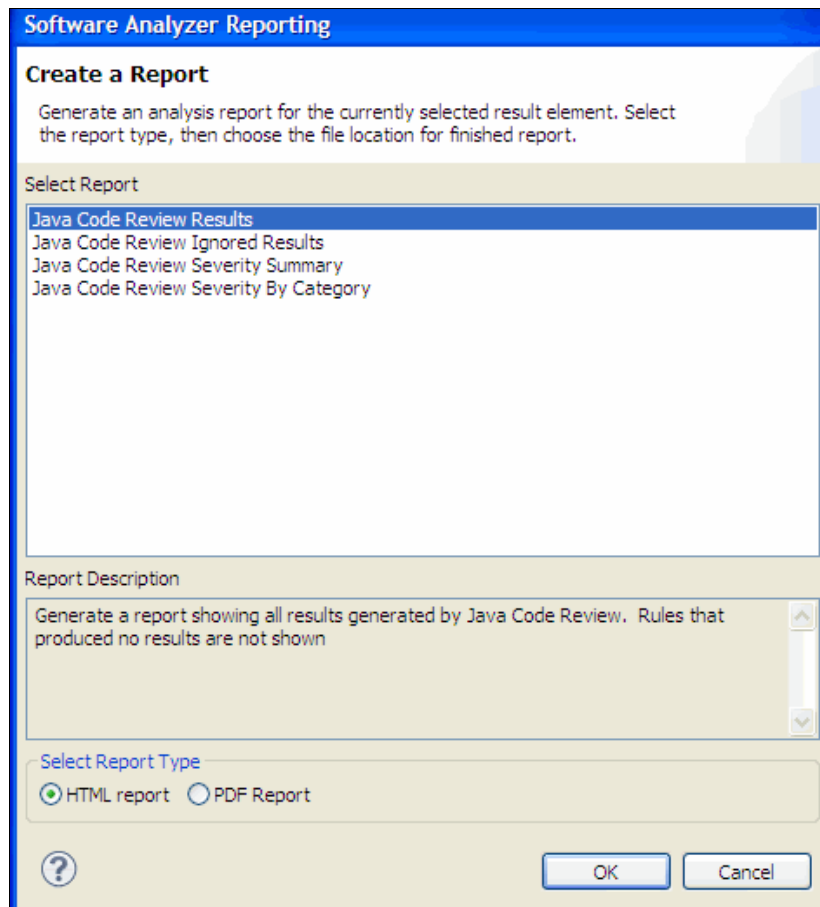


Figure 4-22 Create a report

The report can be generated in both HTML and PDF format. The generated files are put in the workspace/.metadata/.plugins/com.ibm.xtools.analysis.reporting/reports directory.



To select a summary of the report, click **Generate reports for historical scans**. Figure 4-23 shows the sample report that is generated for Analysis Summary.

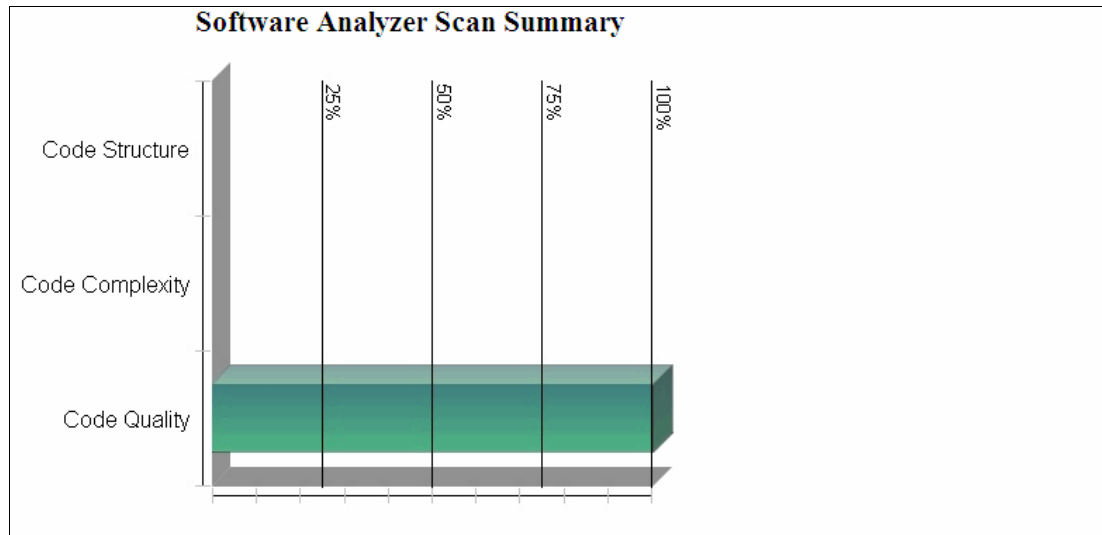


Figure 4-23 Sample report

## 4.6 Troubleshooting

The Application Migration Tools write error information to two log files:

- ▶ The workspace log (workspace/.metadata/.log) contains messages that are logged by the Application Migration Tools and messages that are logged by Rational Software Analyzer.
- ▶ The service logs for the application migration tools are in the workspace/.metadata/.plugins/com.ibm.ws.appconversion.base directory.

This log contains all error information and detailed trace information if trace is enabled.

Enable trace in the migration tools by setting the `appconversion.trace` system variable on the command line to start the Integrated Development Environment (IDE) or in the `eclipse.ini` property file:

- ▶ Command-line option

Add the system variable to the command line that starts Eclipse:

```
eclipse.exe -vmargs -Dappconversion.trace=true
```

- ▶ `eclipse.ini` option

Add the following option to the `eclipse.ini` file found in the same directory as the `eclipse.exe` file:

```
-Dappconversion.trace=true
```

**Tip:** Running trace slows the tool. Turn it on only if requested to do so for service.





# Differences between Eclipse and Rational Application Developer

This chapter describes the differences between Eclipse and Rational Application Developer, in order for you to be able to choose between the two environments and choose the one that suits your development needs the most. We also briefly explain the components that Rational Application Developer offers that are different from Eclipse. You can use the Migration Toolkit with both environments with no differences.

This chapter describes the following topics:

- ▶ Feature portfolio for Eclipse and Rational Application Developer
- ▶ Eclipse
- ▶ WebSphere Application Server Developer Tools for Eclipse
- ▶ Rational Application Developer

## 5.1 Feature portfolio for Eclipse and Rational Application Developer

WebSphere Developer Tools for Eclipse provides you with a lightweight desktop development environment for developing Java EE and Web 2.0 applications. Rational Application Developer provides you with a complete solution to develop, deliver, and manage high-quality enterprise applications.

If we compare their capabilities, Rational Application Developer comes with many components (Figure 5-1). For enterprise-level development, consider using Rational Application Developer as your integrated development environment.

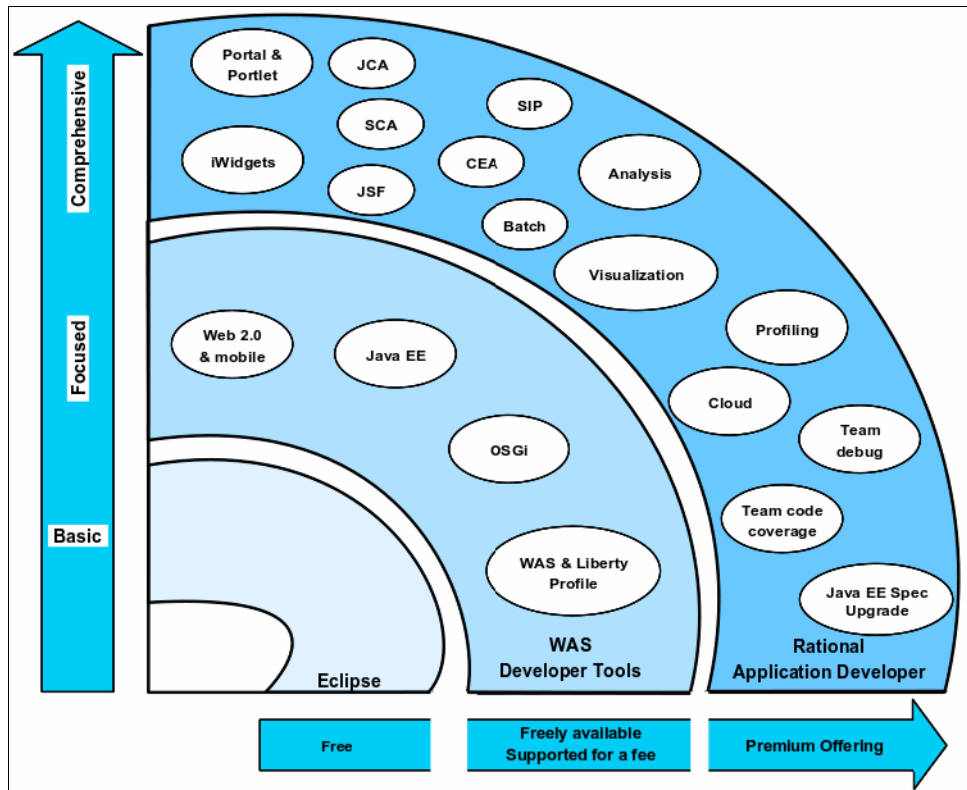


Figure 5-1 Feature portfolio for Eclipse and Rational Application Developer

### Going from Eclipse to WebSphere Developer Tools to Rational Application Developer:

Figure 5-1 on page 112 describes the developer tool offerings, going from Eclipse to WebSphere Developer Tools to Rational Application Developer. The Migration Toolkit complements all offerings that are shown in Figure 5-1 on page 112 and is fully compatible with them. Rational Application Developer includes additional value during the migration, especially with the Java EE Specification Migration wizard, automated code analysis, and debugging tools for improving code quality and performance.

Together with the other features shown in the figure and explained in 5.4, “Rational Application Developer” on page 118, Rational Application Developer provides a complete environment for enterprise development for Java, Java EE, web, web services, SOA, OSGi, and WebSphere Portal applications. Designers and developers can use this environment to develop, assemble, and deploy your applications to WebSphere Application Server V8.5.

## 5.2 Eclipse

Eclipse is an open source and platform-independent software technology that is a rich client platform upon which you can create integrated software applications. Written in Java, Eclipse is a modular integration platform with a sophisticated plug-in architecture that can be used to combine many types of developer tools for many different programming languages.

Eclipse offers multiple packages for different development needs. You can see the descriptions of all these packages and download them at:

<http://www.eclipse.org/downloads/>

Because your goal is to migrate Java EE applications from one application server platform to another, the package that you should use is the Eclipse Integrated Development Environment (IDE) for Java EE Developers package. This Eclipse package provides support for most popular application servers and offers many features for developing Java and Java Enterprise Edition applications, such as:

- ▶ Java editing capability with incremental compilation
- ▶ A graphical HTML/JSP/JSF editor
- ▶ Database management tools

## 5.3 WebSphere Application Server Developer Tools for Eclipse

WebSphere Application Server Developer Tools for Eclipse is a lightweight set of tools for developing, assembling, and deploying Java EE applications to WebSphere Application Server. It contains a subset of the tools from Rational Application Developer that are aimed at Java EE, Web 2.0, mobile web, and OSGi application development for WebSphere Application Server V7.0, V8.0, V8.5, and the Liberty profile.

Rational Application Developer supports all these versions and Version 6.1.

### 5.3.1 Installing WebSphere Developer Tools for Eclipse

When you use the Eclipse environment for development, WebSphere Developer Tools for Eclipse help you work with the WebSphere Application Server V8.5 server run times within the Eclipse environment. It provides productivity enhancements for developers, including ease of use for deploying applications. The installation steps for the WebSphere Developer Tools are as follows:

**Important:** In order for the installation to be completed successfully, install the latest version of Eclipse IDE for Java EE Developers from the Eclipse website. Otherwise, you might encounter problems that are related to missing module dependencies.

1. Click **Help** → **Eclipse Marketplace**.
2. In the Search tab, Enter WebSphere in the search box, and click **Go**.
3. In the search results window (Figure 5-2), click **Install** next to the section that includes IBM WebSphere Application Server V8.5 Developer Tools.



Figure 5-2 Eclipse Marketplace results window

4. After the dependency check, select the feature you want to install in the Features window. The WebSphere Application Server V8.5 Developer Tools feature must be selected.
5. Accept the license agreement and click **Finish**.

6. Click **OK** if a Security Warning opens (Figure 5-3).

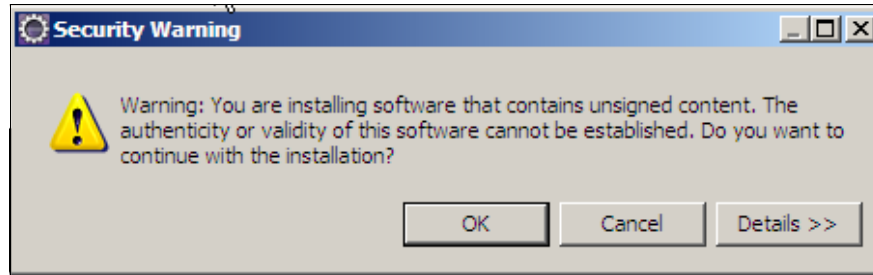


Figure 5-3 Security warning when installing WebSphere Application Server Version 8.5 Developer Tools

7. After the installation ends, click **Restart Now** to restart Eclipse.

You should now be able to add the WebSphere Application Server V8.5 run time as a runnable server on Eclipse. You can also manage this server from the Eclipse environment locally or remotely, access the administrative console, publish your enterprise applications, and adjust security and publishing settings that are based on your preferences.

## Installation verification

The installation of the WebSphere Developer Tools for Eclipse can be verified by creating a WebSphere Application Server V8.5 server by completing the following steps:

1. Right-click anywhere in the Servers view and select **New** → **Server**.
2. In the New Server definition window, click **IBM** → **WebSphere Application Server V8.5** (Figure 5-4).

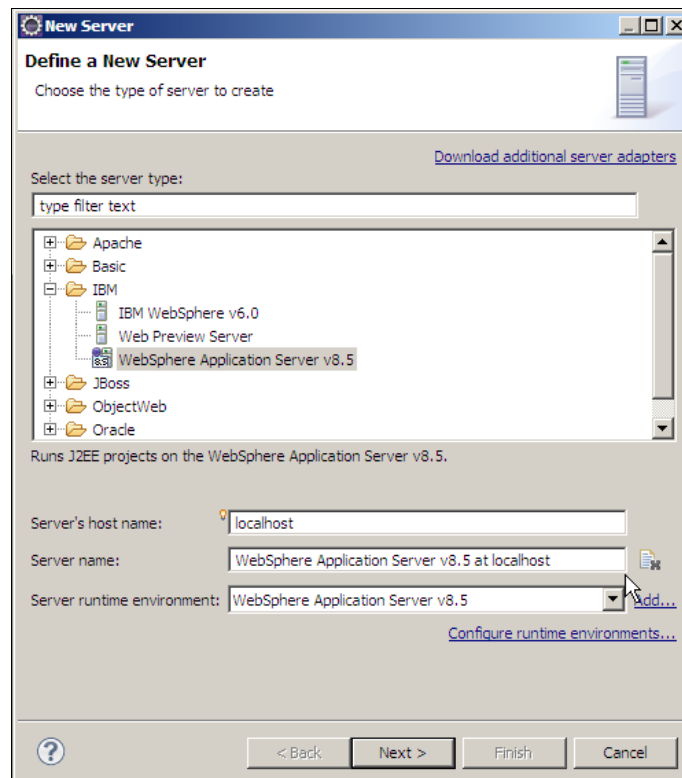


Figure 5-4 Adding a server

3. Enter the host name and a name for your application server and click **Next**.
4. Specify the installation directory of your application server. If the directory is correct, the JRE selection is entered automatically. Click **Next**.

**Tip:** If your WebSphere Application Server instance is on another machine within the same network, you can enter the host name of the remote server instead of localhost. You should also enter the network path of the remote WebSphere Application Server JRE directory to the JRE field. Additionally, the necessary ports and WebSphere Application Server Java runtime directory must be accessible from the Eclipse machine.

5. Select the appropriate profile and port information, enter your user name and password that is based on your security settings, and click **Next**.
6. Choose the applications that must be configured on the server by moving them from the Available section to the Configured section. Click **Finish**.
7. Your new server is now configured and ready to be started (Figure 5-5). If it is not already started, right-click the new server in the Servers window and click **Start**.

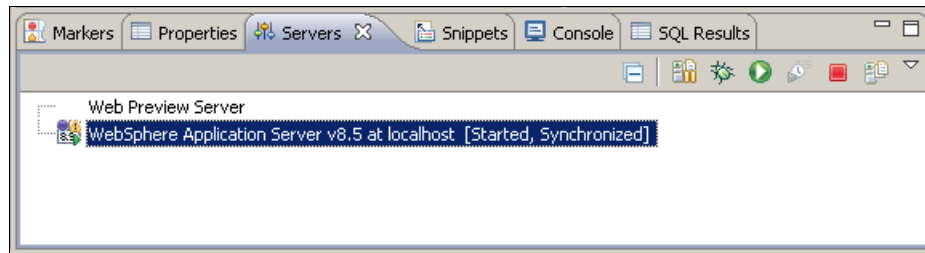


Figure 5-5 Servers view

**Important:** WebSphere Developer Tools For Eclipse supports security with created servers only with Java equal to or greater than Version 6.0. If you are using WebSphere V8.5 with global security enabled, then you must start Eclipse with the Java that comes with WebSphere Application Server V8.5. To accomplish this task add the following lines to the `eclipse.ini` file, which is in the Eclipse directory. The lines must be added just before the `-vmargs` parameter, with `$WAS_HOME` being the WebSphere Application Server installation directory.

- ▶ For Windows:  
-vm  
\$WAS\_HOME\java\jre\javaw.exe
- ▶ For Linux\AIX:  
-vm  
\$WAS\_HOME/java/jre/javaw

We cover the WebSphere Developer Tools for the Liberty profile in 9.3.2, “Installing the developer tools” on page 277.



## 5.3.2 IBM WebSphere Application Server Developer Tools for Eclipse capabilities

IBM WebSphere Application Server Developer Tools for Eclipse has tools for:

- ▶ Managing the server, such as starting and stopping
- ▶ Starting a remote server
- ▶ Publishing local and server-side code
- ▶ Controlling incremental publishing

When combined with Eclipse IDE for Java EE Developers, WebSphere Application Server Developer Tools for Eclipse provides a lightweight environment for developing Java EE applications.

The following sections describe the tools that come with WebSphere Application Server Developer Tools for Eclipse.

### OSGi

The Open Service Gateway initiative (OSGi) Applications framework provides a programming model for developing, assembling, and deploying, as bundles, modular applications that use both Java EE and OSGi technologies.

WebSphere Application Server Developer Tools contains all OSGi development tools, including:

- ▶ Deployment to WebSphere Application Server V8 and V8.5 OSGi bundle repository support
- ▶ Blueprint form editor and Blueprint validation Bundle explorer

Rational Application Developer V8.5 has new support for EJB applications. You can create an OSGi bundle from an EJB facet. You can import or export an EJB as a bundle, and promote an EJB as an OSGi service.

### Java EE

You can use WebSphere Application Server Developer Tools for Eclipse to write applications that works with the Java EE specification. Most of the programming models under the Java EE umbrella are supported by both Rational Application Developer and WebSphere Application Server Developer Tools for Eclipse, such as Servlets, JSP, EJB, JPA, JAX-RS, JAX-WS, and JAXB.

Rational Application Developer supports all these tools. In addition, other tools for the Java EE family, such as JSF and JAX-RPC, are found only in Rational Application Developer.

### Web 2.0 and mobile

WebSphere Application Server Developer Tools include a Rich Page Editor, which is a multi-tabbed editor that makes it easy to edit HTML files, add Dojo widgets to HTML pages, and create and edit web pages for mobile devices.

You can install Dojo Toolkit V1.7 in WebSphere Application Server Developer Tools. Dojo helps you design and develop rich interfaces. It is also usable in any mobile platforms.

Using Rational Application Developer, you can build applications on the latest Web 2.0 technologies, such as HTML 5, CSS3, JavaScript, Dojo, and JSO. Using the Mobile Browser Simulator, which is shown in Figure 5-6, you can easily develop and test mobile web applications that are based on the dojox.mobile framework. Rational Application Developer V8.5 includes IBM Worklight Studio V5.0 from the IBM Mobile Foundation family so that you can also develop native or hybrid applications.

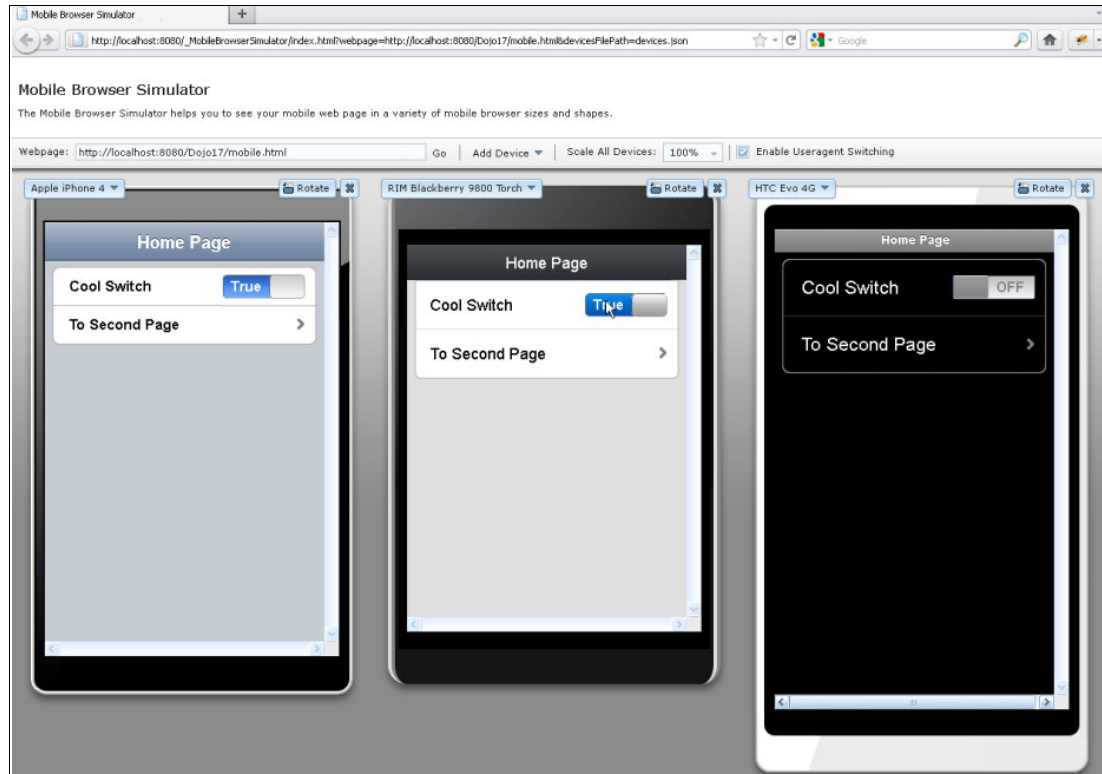


Figure 5-6 Mobile Browser Simulator

### WebSphere Application Server and Liberty profile

WebSphere Application Server Developer Tools comes with WebSphere Application Server and WebSphere Application Server Liberty profile server run times. You can create server connections, manage your profiles and application servers, and deploy and run your applications on your application servers.

## 5.4 Rational Application Developer

Rational Application Developer provides optimized development for IBM middleware. It helps Java developers rapidly design, develop, assemble, test, profile, and deploy high quality Java/Java EE, Portal, Web/Web 2.0, OSGi, web services, and SOA applications. It also has a Java EE Specifications Upgrade feature that is useful while you migrate your applications.

Here are some of the features of the Rational Application Developer.

**Want to learn more about Rational Application Developer and what is new with Version 8.5?** Check out the “Rational Application Developer V8.5 What's New” wiki at:

<http://www.ibm.com/developerworks/wikis/display/rad/Rational+Application+Developer+V8.5+What%27s+New>

In this wiki, you find many videos about most of the features that we explain in the following sections and the new features that are available with Rational Application Developer V8.5. These videos help you understand the added value that is provided by Rational Application Developer.

Also, *Rational Application Developer for WebSphere Software V8 Programming Guide*, SG24-7835 is an excellent reference for the features that are described in this section.

### 5.4.1 Java EE Specifications Upgrade wizard

You can use the Java EE Specifications Upgrade wizard to upgrade your applications' Java EE Specification from J2EE 1.2, 1.3, or 1.4 to Java EE 5.0 and from Java EE 5.0 to Java EE 6.0. This feature eliminates the time spent for manually making code changes to comply with a newer Java EE Specification. We describe the wizard in more detail in 3.3, “J2EE to Java EE migration considerations” on page 75.

You can get online information about the Java EE Specifications Upgrade wizard in the Rational Application Developer Information Center at:

<http://publib.boulder.ibm.com/infocenter/radhelp/v8/index.jsp?topic=/com.ibm.javaee.doc/topics/tmgv6j2eewiz.html>

### 5.4.2 Visualization

Several visualization tools are provided in Rational Application Developer, including support for the creation and editing of UML diagrams (including using the UML diagram to directly edit Java interfaces and classes), support for visualization of XML documents, and schemas.

Visualization of Java classes, EJBs, JPA entity beans, and web services is another feature that comes with Rational Application Developer. You can use it to configure all the Enterprise beans in the same window, and perform such activities as EJB validation, refactoring, quick fixes, and EJB Annotation view. Additionally, the editor includes sections and pages that are related specifically to WebSphere Application Server bindings and extensions to the EJB specification.

### 5.4.3 Portal and portlet

Rational Application Developer provides development tools for portal and portlet applications for WebSphere Portal. Several portal tools are bundled with Rational Application Developer that you can use to create, test, debug, and deploy portal and portlet applications:

- ▶ Visual portal application development and editing of the themes and skins that control their appearance
- ▶ Compatible WebSphere Portal Server v8.0 unit test environments
- ▶ Integrated support for JSF and the Struts framework
- ▶ Business portlet development using IBM WebSphere Portal Application Integration (for SAP and Siebel)

- ▶ Portlet templates
- ▶ Support for IBM Portlet API and the JSR 168 and JSR 286 standard portlet API

### 5.4.4 iWidgets

An iWidget is a small application or piece of dynamic content that can be easily placed into a web page. It is a reusable application that is designed to work within the framework that is defined by the iWidget specification.

An iWidget acts as a wrapper for any web content that you create, such as servlets, JavaServer Pages (JSP) files, HTML, PHP, or CGI.

**Portlet project features:** Two new portlet project features are added in to Rational Application Developer. These new portlet project features include the Dojo-enabled portlets and the creation of iWidgets inside a portlet project. The Dojo-enabled portlets run the portal runtime environment, with the option to create custom widgets that can be reused and enhanced further for WebSphere Portal V6.1 and later. You can also create and publish iWidget projects or create widgets inside a portlet project.

### 5.4.5 Java EE Connector Architecture

Java EE Connector Architecture (JCA) is a standard architecture for connecting enterprise information systems (EISs), such as CICS, IBM IMS™, SAP, Siebel, PeopleSoft, JD Edwards, and Oracle. JCA standardizes the way that Java EE application components, Java EE-compliant application servers, and EIS resources interact with each other. Resource adapters and application servers implement the contract that is defined in the JCA specification. Resource adapters run in the context of the application server and enable Java Enterprise Edition (Java EE) application components to interact with the EIS using a common client interface. JCA-compliant application servers can support any JCA-compliant resource.

Java EE Connector Architecture (JCA) tools and adapters for Rational Application Developer include simplified wizards to generate Java connector code and J2C standards-based connectors for building WebSphere applications that integrate CICS and IMS transactions (Figure 5-7).

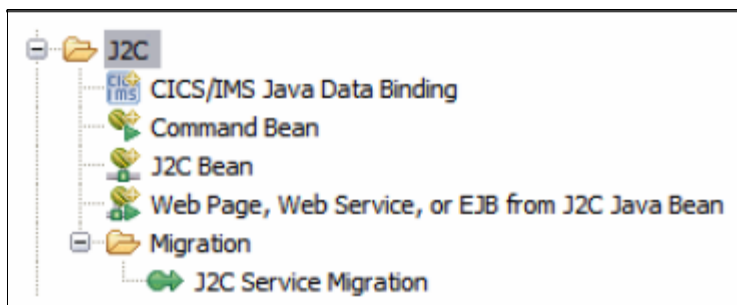


Figure 5-7 J2C wizards on Rational Application Developer

## 5.4.6 Service Component Architecture

You can use the Service Component Architecture (SCA) Project wizard (Figure 5-8) to create a Service Component Architecture (SCA) project. SCA applies service-oriented architecture (SOA) concepts to the development of software components. You can use the wizard to choose the target run time and facet configuration, and the types of implementation that can be used for the SCA components, such as Java and EJB.

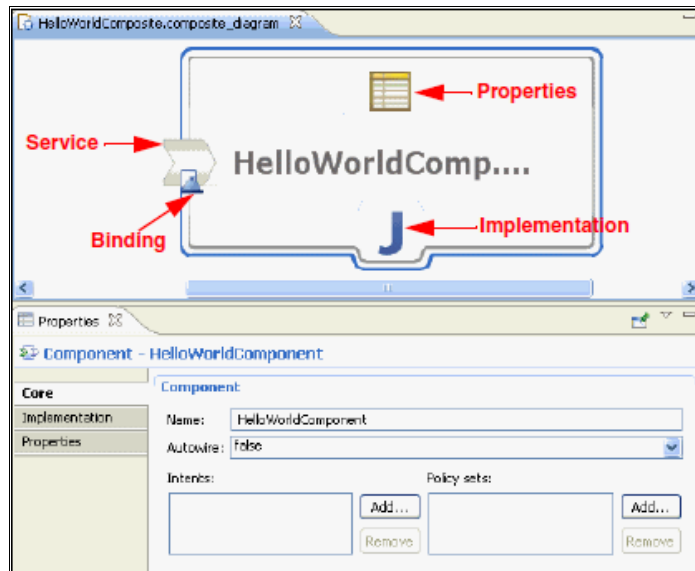


Figure 5-8 SCA diagram

Rational Application Developer V8.5 provides a preview of SCA Tools for the development of OASIS SCA V1.1 applications, including deployment to WebSphere Application Developer V8.5. For more information about OASIS SCA 1.1, see <http://oasis-openca.org/sca>.

You can use the following SCA extension types:

- ▶ Composite, and Java Implementation types
- ▶ Default (SCA) and web service bindings
- ▶ JAXB 2.0 data bindings

The SCA tools work closely with the Java (JDT) and web services tools, so you can locate the various interface binding or implementation artifacts (Java interfaces, classes, and WSDL port types), or use the new OASIS SCA annotations.

## 5.4.7 JavaServer Faces

You can use the JavaServer Faces (JSF) framework to use pre-built components and easily create web applications. For example, JSF provides Input Text, Output Text, and Data Table components. You can add these components easily to a JSF web page and connect them to your project's data. JSF technology and the JSF tooling that is provided by Rational Application Developer enable even inexperienced developers to quickly develop web applications.

Rational Application Developer supports JSF 1.x and 2.0 specifications and tag libraries, and provides a number of rich features to enhance your usage of the JSF framework:

- ▶ JSF Trace
- ▶ Integration of third-party JSF tag libraries
- ▶ Customizable data templates

## 5.4.8 Communications Enabled Applications

You can use Communications Enabled Applications (CEA) to add dynamic web communications to any application or business process. The CEA feature, which was originally delivered in the WebSphere Application Server V7 Feature Pack for Communications Enabled Applications, and is shipped in the base WebSphere Application Server V8 product, provides a suite of integrated telephony and collaborative services that extends the interactivity of enterprise and web commerce applications.

With the CEA capability, enterprise solution architects and developers can use a single core application to enable multiple modes of communication. You do not need to have extensive knowledge of telephony or Session Initiation Protocol (SIP) to implement CEA; the CEA capability delivers call control, device notifications, and real-time data communications, and provides a platform for easily integrating these capabilities into web and mobile applications.

With this feature available in Rational Application Developer V7.5, developers can build, test, and deploy communications enabled applications for WebSphere Application Server V7 and above with Rational Application Developer V7.5 and above.

## 5.4.9 Session Initiation Protocol

SIP is a peer-to-peer protocol that is used to establish, modify, and terminate multimedia IP sessions between two end points, including telephony and instant messaging. It is an extension to the Java EE servlet API that is intended for telecommunications applications using technologies. The following examples are common usage examples of SIP in telecommunications-based applications:

- ▶ Voice-over-IP (VoIP)
- ▶ Instant messaging
- ▶ Click to call
- ▶ Call notification, forwarding, and blocking

The SIP wizard in Rational Application Developer creates a web project with the appropriate facets selected so that you can construct SIP applications.

## 5.4.10 Modern Batch Tools

Modern Batch Tools adds support for the Batch programming model. The Batch programming model allows applications processing to be scheduled by the application server (for example, overnight processing when web traffic is expected to be low).

The Modern Batch Tools, in addition to supporting the WebSphere Application Server Modern Batch Feature pack, now support targeting for both WebSphere Application Server V8 augmented with WebSphere Compute Grid V8, and WebSphere Application Server V8.5 (with the full WebSphere Compute Grid built in).

The Modern Batch Tools include more enhancements, such as:

- ▶ Improved support for more xJCL commands and elements
- ▶ Improved validation and codegen options
- ▶ Support for mixed type jobs for more flexibility
- ▶ Editor changes for new model elements
- ▶ Support for Parallel jobs (run element)
- ▶ Listener types

### 5.4.11 Analysis

You can use the Analysis tools to analyze existing code to discover its architecture and to discover potential software problems. Some (but not all) of this functionality is also available as part of the Migration Toolkit.

### 5.4.12 Profiling

Profiling is a technique that developers use to collect runtime data and detect application problems, such as memory leaks, performance bottlenecks, excessive object creation, and exceeding system resource limits during the development phase.

You can use the profiling tools in the following cases to gather data on applications that are running in these situations:

- ▶ Inside an application server, such as WebSphere Application Server
- ▶ As a stand-alone Java application
- ▶ On the same system as Rational Application Developer
- ▶ In multiple Java virtual machines (JVMs)
- ▶ On a remote WebSphere Application Server with IBM Rational Agent Controller installed.

IBM Rational Agent Controller V8.3 can be used for remote profiling on any WebSphere Application Server server that runs the Java Development Kit (JDK) V1.5 or later.

### 5.4.13 Cloud

You can use Rational Application Developer for provisioning and using virtual instances of WebSphere Application Server or WebSphere Portal on IBM SmartCloud™ Enterprise, and elastic Virtual Application Pattern workloads for IBM Workload Deployer and IBM expert integrated systems. Rational Application Developer supports cloud-based solutions to help you better manage IT costs.

**IBM SmartCloud Enterprise:** IBM SmartCloud Enterprise is an agile cloud-computing infrastructure that provides rapid access to security-rich, enterprise-class virtual server environments, which are suited for development and test activities and other dynamic workloads.

The IBM SmartCloud Enterprise can be accessed at:

<https://www-147.ibm.com/cloud/enterprise/dashboard>

#### **5.4.14 Team Debug**

You can use the debug extension for IBM Rational Team Concert™ Client (Team Debug) to run Java-based debug collaborative sessions between Rational Team Concert team members. While you are debugging complex applications, any team member might require the expertise of a specialist to understand and correct a specific part of the code. However, re-creating a debug session to reproduce a particular scenario can be time-consuming. Rational Application Developer offers the capability to share the complete state of an existing debug session with another user, including any breakpoints that are already set.

#### **5.4.15 Code Review (static analysis)**

The Code Review function includes over 200 predefined rules for coding guidelines and preferred practices. It provides “Quick Fix” suggestions and allows customization of rules through rules templates.

#### **5.4.16 Team Code Coverage**

Code Coverage is an important aspect of software testing and can be considered fundamental to the overall system testing of a component. The motivation behind coverage tooling is to give developers and testers more insight into the areas of code that are being exercised by a set of test cases. This information is useful to developers and testers who can then use it to devise test cases so that adequate coverage can be achieved.



To get this information for your project, right-click the project and select **Preferences**. In the left pane, you can see the Code Coverage selection. In the Code Coverage properties window, enable Code Coverage by selecting the **line** check box in the Package, Source file, Type and Method coverage options. Then you can create new filters for project. (Figure 5-9).

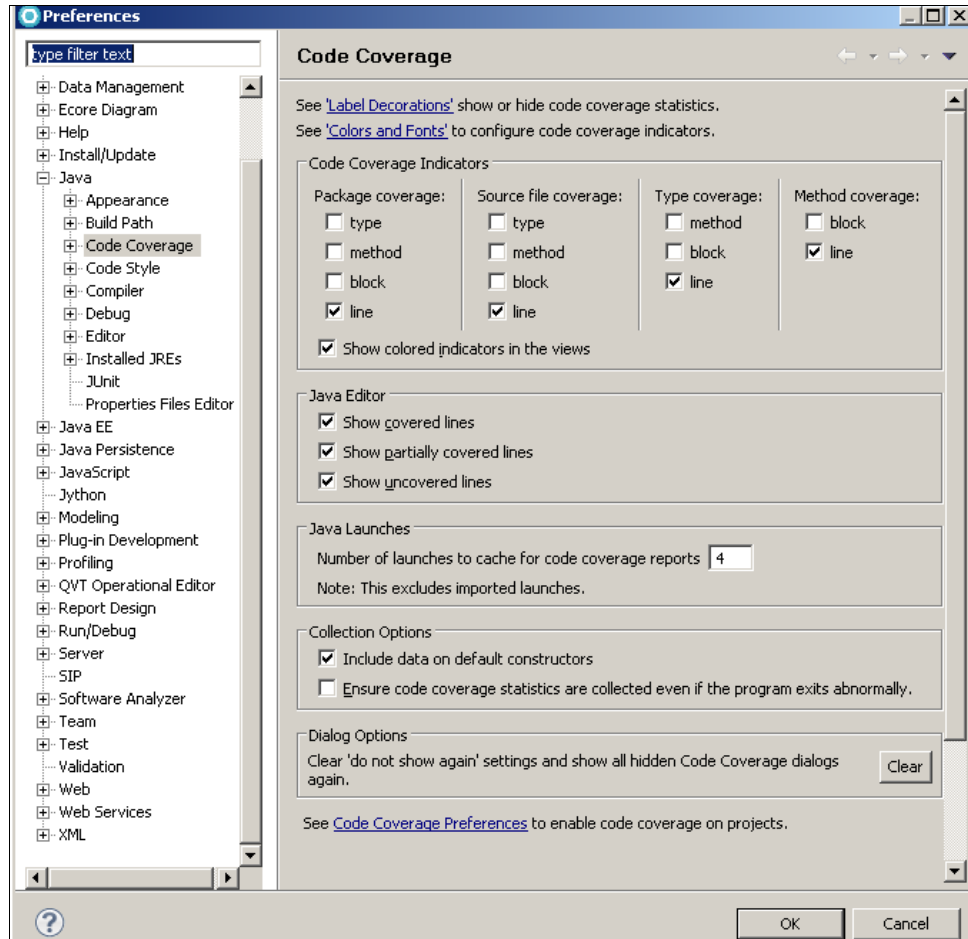


Figure 5-9 Code Coverage properties

You can get the results of your code coverage by clicking **Run** → **Code Coverage** → **Generate Reports**.

As shown in Figure 5-10, you can use Code Coverage to find the results line by line.

Element	Coverage <sup>▲</sup>	Covered Lines	Total Lines
com.ibm.storeapp.models	85%	75	88
Customers.java	50%	7	14
Customers	50%	7	14
Customers()	100%	3	3
incrementCapacity(): void	0%	0	6
addCustomer(Customer): void	75%	3	4
getCustomers(): Customer[]	100%	1	1
Part.java	63%	5	8
Store.java	94%	51	54
Customer.java	100%	6	6
Product.java	100%	6	6

Figure 5-10 Code Coverage

When you run a Code Coverage enabled application on the server, you can choose to reset the Code Coverage indicators so that Code Coverage data is collected fresh based on your interaction with the application.

### 5.4.17 XML tools

Rational Application Developer provides a comprehensive visual XML development environment. The tool set includes components for building DTDs, XML schemas, XML documents, and XSL files.

The features of the Graphical Data Mapper include:

- ▶ Powerful graphical mapping capability
- ▶ Support for built-in XPath 1.0 and 2.0 constructs
- ▶ Support for XSLT 1, XSLT 2, and XQuery standards
- ▶ Ability to map to and from database mapping sources
- ▶ Support for auto-mapping of similar fields
- ▶ Support for quick search of data structures and mappings
- ▶ Support for reusable submaps and local maps
- ▶ Smart data-type conversions and direct casting
- ▶ Support for “if/else” conditions
- ▶ Support for Nillable and Empty elements
- ▶ Interoperability with user-defined Java transformation code

The other functions that are not on Eclipse can be listed to highlight what the differences between Eclipse and Rational Application Developer 8.5 are:

- ▶ Java EE Connector Architecture (JCA) tools and adapters for building WebSphere applications that integrate CICS and IMS transactions.
- ▶ Create web services from beans, EJBs, or WSDL/WSIL files using the IBM WebSphere JAX-RPC web service run time.
- ▶ Web-tier facade layer for easy consumption with JSF/servlets/JPSs.
- ▶ Drag and drop support for Dojo widgets.
- ▶ Profiling: Object names and live data content are now available during memory-usage analysis.
- ▶ Web diagram editor for Model-View-Controller design.



## Migrating from Oracle WebLogic

This chapter describes the migration process and individual migration issues and solutions we found in our examples when we migrated both J2EE 1.3 and J2EE 1.4 applications from Oracle WebLogic Server 10.3.6 to WebSphere Application Server V8.5. We provide step-by-step instructions for migrating these applications.

We also describe the capabilities of the Migration Toolkit V3.5 and the configuration options available to take advantage of the tool's analysis capabilities.

We did not have major problems with migrating our sample applications. In both cases, we were able to deploy WebLogic compiled EAR files into WebSphere with minimal changes to the Java source code. Most of the changes were limited to deployment descriptors and configuring security. This situation is as expected because the J2EE specification allows for vendor-specific deployment descriptors.

This chapter describes the following topics:

- ▶ Introduction
- ▶ Prerequisites and assumptions
- ▶ Oracle WebLogic Server 10.3.6 installation
- ▶ Application Migration Tool - WebLogic to WebSphere
- ▶ Migrating Trade 3.1 for Oracle WebLogic Server 10.3.6
- ▶ MVC WebLogic Application migration

## 6.1 Introduction

Before you start migrating your applications, we compare the WebSphere and WebLogic architectural concepts by reviewing the terminology that is used for those concepts by WebLogic and WebSphere administrators and developers.

### 6.1.1 Comparison of WebLogic Domain and WebSphere Cell

Figure 6-1 compares WebSphere Domain and Cell components to WebLogic components.

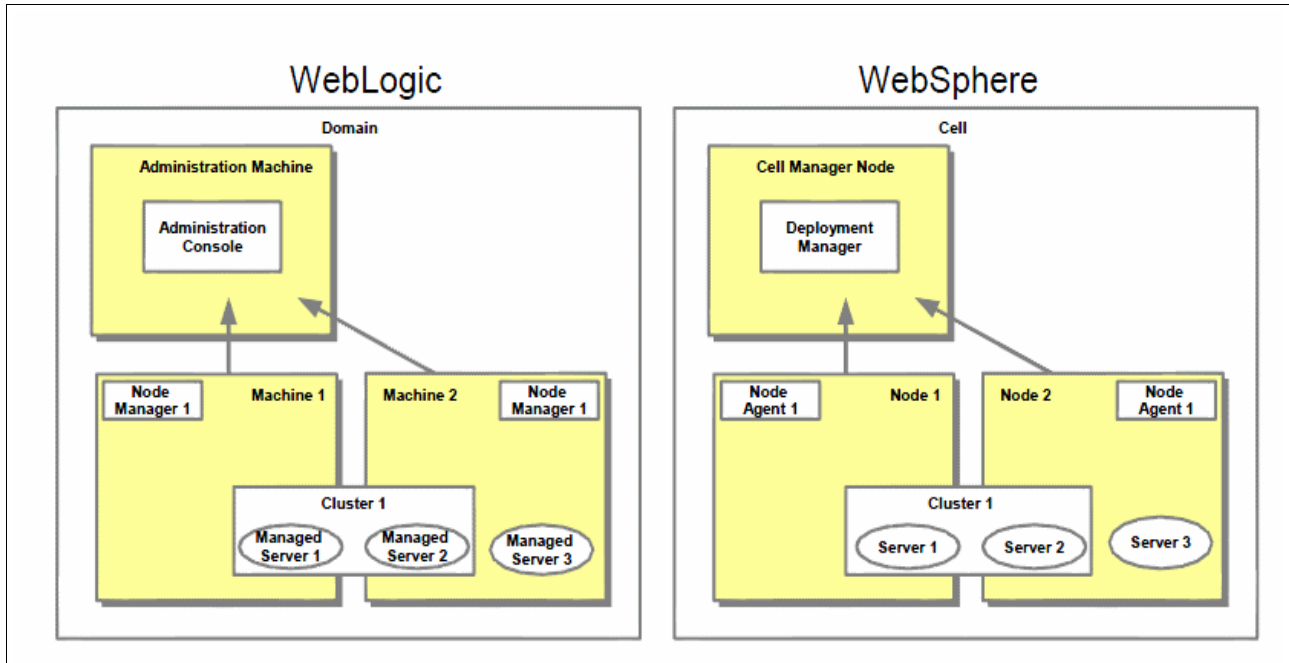


Figure 6-1 Comparison of components between WebLogic and WebSphere

Table 6-1 explains the cell and domain components of the WebLogic and WebSphere Application Server.

Table 6-1 Explanation of the cell and domain components

WebLogic	WebSphere
Domain: One or many logical groups that are defined on a single machine or spanning multiple machines	Cell: One single logical group that is managed by the deployment manager
Machine: Physical machine	Node: Contains one or many profiles or server instances
Node Manager: A daemon process that is used by the Administration Server	Node Agent: A daemon process that is used by the Deployment Manager
Managed Server: A WebLogic Server instance	Server instance
Administration Server	Deployment Manager

## 6.1.2 Comparison of the JMS Providers

We compare the JMS Providers of WebLogic and WebSphere before we define their components in the migration phase to understand their configurations. See Figure 6-2.

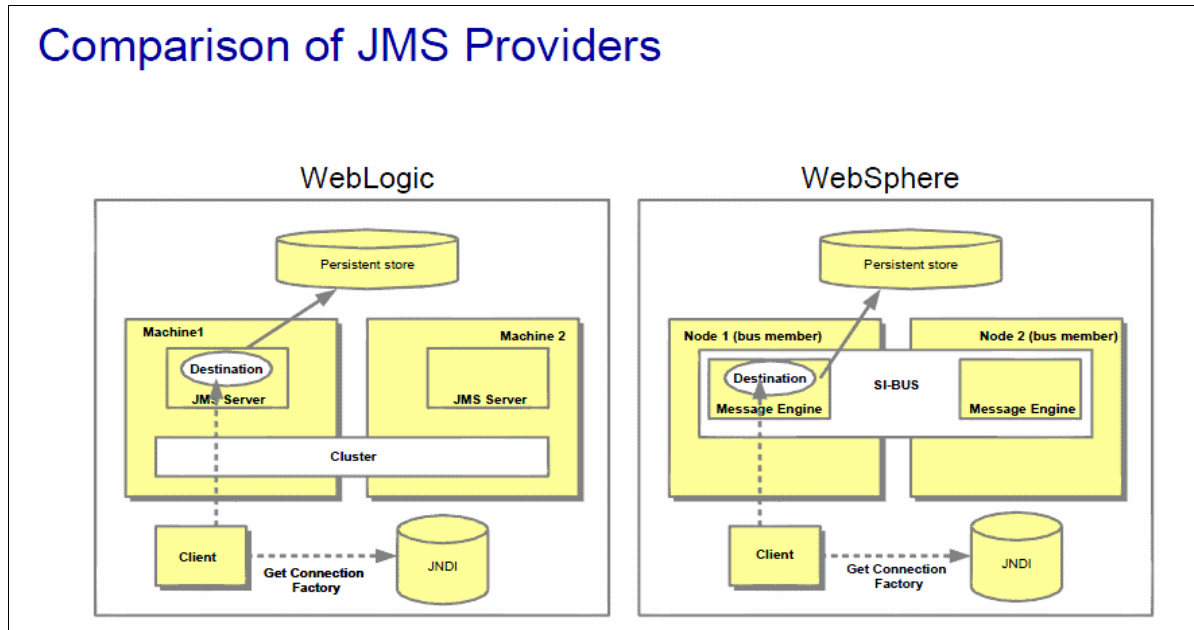


Figure 6-2 Comparison of JMS Providers

Table 6-2 and Table 6-3 on page 130 explains the JMS components and extensions of the WebLogic and WebSphere Application Servers.

Table 6-2 Explanation of the JMS components

WebLogic	WebSphere
JMS Server: A server that is running inside a WLS instance, whether stand-alone or clustered. Multiple servers can be defined.	Service Integration Bus: A shared communication channel that provides a JMS Server, that is optionally bridged to the WebSphere MQ Queue Manager, running inside a WebSphere Application Server instance, whether stand-alone or clustered. Multiple buses can be defined.
Included in JMS Server above.	Bus Member: Each bus can have servers or clusters or both as Bus Members.
Included in JMS Server above.	Messaging Engine: Created on a Bus Member. It manages message resources and its own data store. Multiple Messaging engines can be defined.

Table 6-3 Explanation of JMS extensions

WebLogic	WebSphere
<p>Redirection of failed or expired messages (“dead letter queue”): For each queue defined in WLS, you can specify the name of a specific error queue to receive the message.</p>	<p>Redirection of failed or expired messages (“dead letter queue”): Can be achieved in three different ways (the second one is preferable):</p> <ol style="list-style-type: none"> <li>1. By using the default exception destination that belongs to each messaging engine.</li> <li>2. By specifying a non-default exception destination.</li> <li>3. By setting it to none, that is, disable it.</li> </ol>
<p>WebLogic supports “distributed queues”. A distributed queue is a set of physical JMS queue members. When a message is sent to a distributed queue, it is sent to exactly one of the physical queues in the set of members for the distributed queue. After the message arrives at the queue member, it is available for receipt by consumers of that queue member only.</p>	<p>WebSphere supports “partitioned queues”. When you deploy a queue destination to a cluster, the queue is automatically partitioned across the set of messaging engines that is associated with the cluster.</p> <ul style="list-style-type: none"> <li>▶ If there is only one messaging engine in the cluster, the destination is localized by that messaging engine. The destination is not partitioned.</li> <li>▶ If there is more than one messaging engine in the cluster, the destination is partitioned across all messaging engines in the cluster. Each messaging engine deals with a subset of the messages that the destination handles.</li> </ul>

### 6.1.3 Migrating sample applications from Oracle WebLogic Server 10.3.6

Oracle WebLogic Server 10.3.6 was released in May of 2011 and is the J2EE runtime version from Oracle that is used throughout our migration examples in this publication.

During our migration exercise for this platform, we used two J2EE 1.3 and J2EE 1.4 sample applications:

- ▶ Trade 3.1 for Oracle WebLogic Server 10.3.6
- ▶ MVC WebLogic Application

The Trade 3.1 application that we select for our migration example was developed by IBM for performance testing purposes and does not use any non-J2EE features of Oracle WebLogic Server 10.3.6. We selected Trade for the following reasons:

- ▶ It is a well built application.
- ▶ It uses many of the J2EE 1.3 APIs.
- ▶ It implements many good programming techniques for performance.
- ▶ It implements many patterns to reuse in your application development efforts.

**WebLogic APIs:** Although the samples in the book do not use WebLogic proprietary APIs, it is possible that your application might depend on APIs that are not available in WebSphere Application Server. The Migration Toolkit helps with some WebLogic APIs, such as startup and shutdown classes, but others might require redesign and manual rewriting.

The second application that we migrate is MVC, which is a J2EE application that includes a Container Managed Persistence (CMP) - EJB application that runs on WebLogic. The reason that we select this application is to demonstrate the CMP mappings of EJB objects manually.

We use IBM WebSphere Application Server Migration Tool V3.5 to migrate both the Trade and MVC applications. We explain the capabilities of the new Migration Toolkit and how to configure it in 6.4, “Application Migration Tool - WebLogic to WebSphere” on page 133.

Another useful tool that you can use is the Oracle Java Application Verification Kit (AVK). This tool is intended to help developers test their applications for the correct usage of J2EE APIs and portability across J2EE. For more information, go to:

[http://java.sun.com/j2ee/verified/avk\\_enterprise.html](http://java.sun.com/j2ee/verified/avk_enterprise.html)

## 6.2 Prerequisites and assumptions

There are prerequisites and assumptions you must consider before you continue with the migration examples. One basic requirement is that you should have the destination environment installed and configured as described in Chapter 4, “Installation and configuration of the Application Migration Tools” on page 89. That chapter provides initial background knowledge if you are not yet familiar with IBM products.

You should be familiar with the J2EE specification and architecture, and be able to understand and run basic SQL commands. You should be familiar with Rational Application Developer V8.5 and Apache Ant. You should also have basic familiarity with Oracle WebLogic Server 10.3.6.

Install the following software before you start the migration:

- ▶ Java SE 6

Use Java SE 6 because it is supported by WebSphere Application Server V8.5 and you must ensure that your applications work in this JVM. Before Java SE 6, the specification name was Java 2 Platform, Standard Edition (J2SE). The WebSphere Application Server V8.5 installation includes Java SE 6. You do not need to install additional JDKs on your machine.

- ▶ Apache Ant 1.6.5

Apache Ant is a Java based build tool. It is similar to the Make tool, but has many enhanced features, so it can be extended using Java classes instead of writing shell commands, which makes it a good choice for Java applications.

- ▶ IBM WebSphere Application Server V8.5

Before you migrate to WebSphere Application Server V8.5, explore the sample applications that are shipped with the product and read the publications that are listed in “Related publications” on page 437.

► IBM DB2 UDB 10.1

WebSphere Application Server V8.5 supports many databases, but we decided to use DB2 for all our examples.

► IBM Rational Application Developer V8.5

We used the new WebSphere Application Migration Tool V3.5 plug-in for Rational Application Developer V8.5 for our examples. Details about the tool's capabilities can be found in 6.4 Chapter 5, "Differences between Eclipse and Rational Application Developer" on page 111.

## 6.3 Oracle WebLogic Server 10.3.6 installation

This section provides high-level steps and tasks for installing and configuring Oracle WebLogic Server 10.3.6 on the Microsoft Windows 7 platform. Detailed instructions for installing, configuring, and managing Oracle WebLogic Server 10.3.6 are provided in the product download page available at:

[http://www.oracle.com/technology/software/products/ias/htdocs/wls\\_main.html](http://www.oracle.com/technology/software/products/ias/htdocs/wls_main.html)

The following list highlights the high-level tasks that are performed to install and configure the initial WebLogic domain environment. This setup provides a clean setup as a starting point for deploying the two sample applications.

1. Download Oracle WebLogic Server 10.3.6 from the Oracle web page.
2. Install the software in the C:\Oracle\Middleware directory. This directory is later referred to as <weblogic\_home>.
3. Start the Oracle WebLogic Configuration wizard by clicking **Start** → **Programs** → **BEA Products** → **Tools** → **Configuration wizard**.
4. On the Welcome window, select **Create a new WebLogic domain**.
5. On the Select Domain Source window, select **Generate a domain configured automatically to support the following BEA products**.
6. On the Configure Administrator Username and Password window, enter the user name and password of your choice. We use weblogic/Passw0rd in our installation.
7. On the Configure Server Start Mode and JDK window, make the following selections:
  - WebLogic domain startup mode: Development Mode
  - JDK: Sun SDK 1.6.0\_29
8. On the Customize Environment and Services Settings window, keep the default selections and click **Next**.
9. On the Create WebLogic Domain window, enter the following information:
  - Domain name: base\_domain
  - Domain location: C:\Oracle\Middleware\user\_projects\domains\base\_domain
10. Click **Create**.
11. As we included the implementations of classes that are loaded from wlcommons-logging.jar, put the JAR file into C:\Oracle\Middleware\wlserver\_10.3\server\lib\consoleapp\APP-INF\lib.



## 6.4 Application Migration Tool - WebLogic to WebSphere

Application Migration Tool - WebLogic to WebSphere (Application Migration Tool in this chapter) is part of the Migration Toolkit, which is described in Chapter 4, “Installation and configuration of the Application Migration Tools” on page 89. Application Migration Tool can help you in the following areas when you migrate J2EE 1.4 and Java EE 5 applications, and earlier from Oracle WebLogic Server V8.x and higher to WebSphere Application Server V7, V8, or V8.5:

- ▶ Class path setup: Converts Enterprise Archive (EAR) class paths to J2EE specifications
- ▶ WebLogic Server Startup Classes: Detects and converts T3StartupDef and T3ShutdownDef classes
- ▶ Non-Portable JNDI lookups: Detects and refactors JNDI lookups to use the parameter-less InitialContext
- ▶ JSP files: Enforces case sensitivity of JSP tag/attribute names against the TLD file
- ▶ Deployment Descriptors: Detects and converts WebLogic specific deployment descriptor elements to IBM deployment descriptor elements
- ▶ WebLogic proprietary Java package references: Detects and converts proprietary package references to standards-based package references
- ▶ Web services: Generates an Ant script, Service Endpoint Interface, WSDL, deployment descriptors, mapping files, and other JAXRPC web service artifacts
- ▶ Java EE5 items: Detects JPA, JDBC, XML JTA, and logging items and converts many of them

## 6.5 Migrating Trade 3.1 for Oracle WebLogic Server 10.3.6

The Trade 3.1 application was developed by IBM to evaluate J2EE performance issues. Although built for performance evaluation, the application provides various features that we are interested in covering during the migration exercise. The latest version of Trade is V6. It is built as a complete J2EE 1.4 application that uses most components of the specification. Except for the J2EE level, both versions of the Trade application are functionally equivalent.

The Trade 3.1 version we used for Oracle WebLogic Server 10.3.6 is a slightly modified version. For details about how to download the Trade 3.1 version that is used for Oracle WebLogic Server 10.3.6, see Appendix B, “Additional material” on page 435.

The modifications that are made to the application were to configuration files and deployment descriptors. The core of the application remains unchanged, so we do not expect any issues with the Java code. We mainly focus on migrating deployment descriptors and creating resources in WebSphere Application Server V8.5 to support the application.

The main features that are covered and provided by this application make extensive use of the following J2EE 1.3 APIs:

- ▶ JMS 1.1
- ▶ JDBC 2.1
- ▶ Entity and Stateless Session EJB 2.0
- ▶ Primary Key generation pattern for CMP EJB 2.0
- ▶ Message Driven beans 2.0
- ▶ Two-phase commit transactions

- ▶ JSP 1.2
- ▶ Servlet 2.3

Figure 6-3 shows the Trade3 J2EE components.

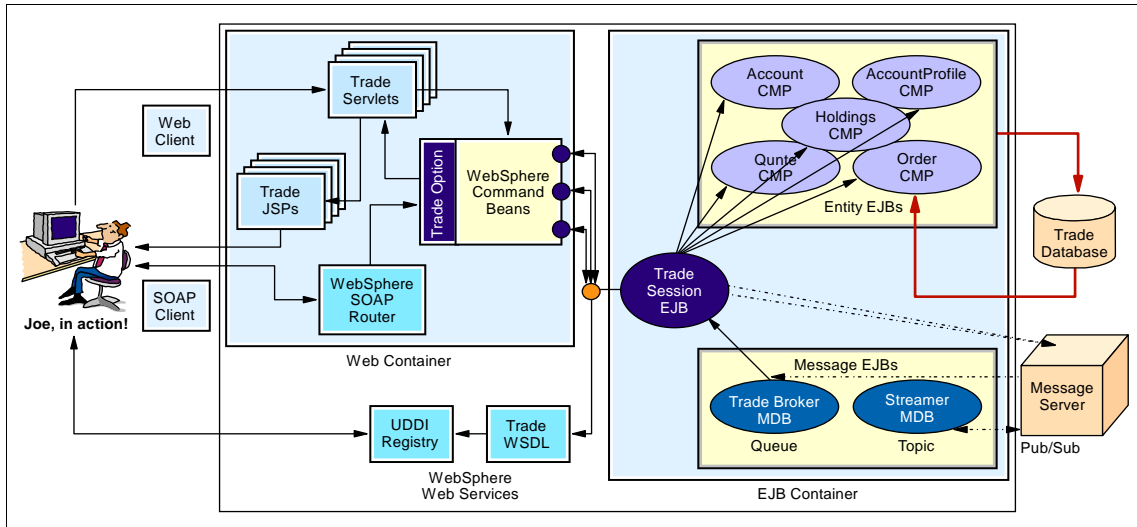


Figure 6-3 Trade3 J2EE components

## 6.5.1 Migration approach

We decided to perform the Trade migration using Rational Application Developer V8.5. We decided to use the Application Migration Tool because it supports the migration of EJB deployment descriptors, Java source code, and JSP code from applications previously running on Oracle WebLogic Server 10.3.6 to WebSphere Application Server V8.5.

The migration is performed by completing the following steps:

1. Verify that the application works in its original environment. In this case, we deploy and run the Trade 3.1 application in Oracle WebLogic Server 10.3.6.
2. Import the EAR file with the source code into Rational Application Developer V8.5.
3. Configure and run the Application Migration Tool.
4. Fix errors that are reported by the Application Migration Tool.
5. Migrate the EJB deployment descriptors using the Application Migration Tool.
6. Build Trade for WebSphere Application Server V8.5 by using Rational Application Developer V8.5.
7. Deploy and test the application on WebSphere Application Server V8.5 and identify and solve any problems that are reported.

## 6.5.2 Configuring the initial environment

After you install Oracle WebLogic Server 10.3.6 and create a server configuration, deploy the sample application to the initial environment. This section describes the procedures that we followed for creating a sample database, configuring Oracle WebLogic Server 10.3.6 to hold the Trade application, deploying the application, and testing.

Because of the size of this section, the following list details the subsections it contains:

- ▶ “Acquiring the Trade application” on page 135
- ▶ “Creating the sample application database” on page 135
- ▶ “Creating a data source” on page 136
- ▶ “Creating the JMS resources” on page 137
- ▶ “Deploying the sample application and populating the database” on page 138
- ▶ “Creating a user and group on WebLogic” on page 139

## Acquiring the Trade application

For information about how to download the Trade 3.1 application for Oracle WebLogic Server 10.3.6, see Appendix B, “Additional material” on page 435.

After you download the application, create a Trade3 directory and extract the downloaded file with the name `trade3-WebLogic.zip`. The compressed file contains the following files and directories:

- ▶ `db2_schema.ddl`  
Script for creating the database and tables in DB2 Universal Database V9.5
- ▶ `trade3wls.ear`  
Trade application binary files
- ▶ `trade3EJB`  
Directory containing source code for the EJB module
- ▶ `trade3Web`  
Directory containing the source code for the web module and J2EE client application

In our scenario, we used the `D:\sampleApp\Trade3` directory to extract the application. It is also referred to as `<source_home>`.

## Creating the sample application database

In this section, we describe the steps for creating the Trade database. The database schema is included in the `trade3-WebLogic.zip` file.

Complete the following steps:

1. From a command line, start the DB2 command line processor (CLP) by running the following commands:
  - `cd <source_home>` (for our example, `cd D:\sampleApp\Trade3`)
  - `db2cmd`
2. Run the following commands at the DB2 command prompt to create the Trade database and tables:

**User name:** The user that you use to connect to the database defines the name of the schema where the DDL creates the tables.

- `db2 create db trade3db`
- `db2 connect to trade3db user db2admin using db2admin`
- `db2 -tvf db2_schema.ddl`
- `db2 disconnect all`
- `db2 update db config for trade3db using logfilsiz 1000`
- `db2 update db cfg for trade3db using maxappls 100`
- `db2stop`

- db2start
- db2 connect to trade3db user db2admin using db2admin
- cd <db2\_home>\bnd
- db2 bind @db2cli.1st blocking all grant public

The database is empty and ready to for the data to be loaded. The creation of the initial data in the database is done by the application when it is deployed.

3. Start the Oracle WebLogic Server 10.3.6 instance (Admin Server) from the Windows Start menu or from the command line by running the following commands:

- cd C:\Oracle\Middleware\user\_projects\domains\base\_domain
- startWebLogic.cmd

4. Access the Console Login when the server is up and running. When we created the WebLogic configuration, we accepted the default values for port and location. In our scenario, the Console Login can be accessed at the following URL:

http://localhost:7001/console

The user name and password that are required for login are specified during the creation of the new WebLogic configuration. In our scenario, we used weblogic/Passw0rd for the user name and password.

## Creating a data source

To create a data source, complete the following steps:

1. On the WebLogic Server Console, navigate to base\_domain and click **Services** → **JDBC** → **Data Sources** and create a data source with the characteristics listed in Table 6-4.

Table 6-4 Data source values

Parameter	Value
Database type	DB2
Database Driver	Oracle DB2 Driver (Type4) Versions: 7.X and later
Name	TradeDataSource
JNDI Name	jdbc/TradeDataSource

Click **Next**. The Transaction Options window opens and an informational message is displayed stating that, based on the selected driver type, only XA 2-phase commit transactions are supported. Click **Next**.

2. On the Connection Properties window, enter the information that is listed in Table 6-5.

Table 6-5 Database connection properties values

Parameter	Value
Database Name	trade3db
Host Name	localhost
Port	50000
Database User Name	db2admin
Password	db2admin

**User name and password:** Use the database user name and password for your DB2 installation. In our example, we used db2admin/db2admin.

3. On the Test Database Connection page, click **Test Configuration**. You should see the message “Connection test succeeded”.
4. On the Select Targets window, select **AdminServer** as the target server and click **Finish**.

### Creating the JMS resources

To create the JMS resources, complete the following steps:

1. From the WebLogic Server Console home page, create a JMS Server by navigating to Base\_domain and clicking **Services** → **Messaging** → **JMS Servers**. Use the parameters in Table 6-6 to create a JMS Server.

Table 6-6 JMS Server values

Parameter	Value
Name	TradeJMSServer
Target	AdminServer

2. Create a JMS module to act as the container to define JMS queues, topics, and connection factories.
3. Create a JMS Module by navigating to base\_domain and clicking **Services** → **Messaging** → **JMS Modules**.
4. Enter JMS Resource for the JMS Module name and click **Next**.
5. Select **AdminServer** as the target and click **Finish**.
6. Create the JMS queues, topics, and connection factories that are needed by the Trade application at run time.
7. Click **JMS Resource**, then click **New**. Configure the destinations by creating a JMS Queue and a JMS Topic.
8. Create a JMS Queue with the values shown in Table 6-7. Click **Next**

Table 6-7 JMS queue values

Parameter	Value
Name	TradeBrokerQueue
JNDI Name	jms/TradeBrokerQueue

9. Select **JMSServerSubDep** in the Subdeployment drop-down menu. In the Targets field, ensure that **TradeJMSServer** is selected. Click **Finish**.

10. From the JMS Resource window, create a JMS Topic with the values listed in Table 6-8. Click **Next**.

Table 6-8 JMS topic values

Parameter	Value
Name	TradeStreamerTopic
JNDI Name	jms/TradeStreamerTopic

11. Select **JMSServerSubDep** in the Subdeployment drop-down menu. In the Targets field, ensure that **TradeJMSServer** is selected. Click **Finish**.
12. Create a second JMS Topic with the values listed in Table 6-9. Click **Next**.

Table 6-9 JMS Topic values

Parameter	Value
Name	TradeStatsTopic
JNDI Name	jms/TradeStatsTopic

13. Select **JMSServerSubDep** in the Subdeployment drop-down menu. In the Targets field, ensure that **TradeJMSServer** is selected. Click **Finish**.
14. Create two connection factories:

- Queue connection factory
- Topic connection factory

Create the connection factories with the values shown in Table 6-10.

Table 6-10 Queue connection factory values

Parameter	Value
Name	QueueConnectionFactory
JNDI Name	jms/QueueConnectionFactory

15. In the Targets field, click **Advanced Targeting**.
16. In the Subdeployment drop-down menu, select **JMSServerSubDep**, and in the Targets field, ensure **TradeJMSServer** is selected. Click **Finish**.
17. Create a Topic Connection Factory by repeating steps 14 through 16 using the values that are listed in Table 6-11.

Table 6-11 Topic connection factory values

Parameter	Value
Name	TopicConnectionFactory
JNDI Name	jms/TopicConnectionFactory

## Deploying the sample application and populating the database

The next step is to deploy the Trade application. To accomplish this task, you must upload the trade3w1s.ear through the WebLogic administration console by completing the following steps:

1. Log in to the WebLogic administration console. From the home page, click **Configure Applications** and then click **Install**.
2. Select **Upload your file(s)**.

3. Click **Browse** and select the `trade3wls.ear` file. Under Location, ensure that you select the **trade3wls.ear** radio button. Click **Next**.
4. Select the **Install this deployment as an application** radio button and click **Next**.
5. In the Optional Settings window, accept all defaults and click **Finish**.

This process is quick and normally completes in less than a minute. You should see two successful messages, one for the deployment status for the EJB Modules and another for the deployment status for the web application modules.

## Configuring basic authentication on WebLogic

The application has basic authentication in `web.xml` (Example 6-1) and role name definition in `weblogic.xml` (Example 6-2) in the web application.

*Example 6-1 Basic authentication definition in web.xml*

---

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>secure resources</web-resource-name>
    <url-pattern>*/</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>threadUser</role-name>
  </auth-constraint>
</security-constraint>
<login-config>
  <auth-method>BASIC</auth-method>
</login-config>
<security-role>
  <role-name>threadUser</role-name>
</security-role>
```

---

*Example 6-2 Mapping definition in weblogic.xml*

---

```
<security-role-assignment>
  <role-name>threadUser</role-name>
  <principal-name>ThreadUserGroup</principal-name>
</security-role-assignment>
<session-descriptor>
```

---

The security mapping definition is already defined in `weblogic.xml`, so we must create a group named `ThreadUserGroup` and a user as a member of the group.

**Security mapping definition:** The security mapping definition for WebSphere is different from WebLogic. You must go through the steps that are described in “Configuring basic authentication on WebSphere” on page 185.

## Creating a user and group on WebLogic

To create a user and group on WebLogic, complete the following steps:

1. Open the WebLogic console from the following URL:  
`http://localhost:7001/console`
2. Click **Security Realms** and **myrealm**.
3. Click **Users and Groups** and click **Users**. Click **New** to create a user.

4. Enter `testuser` as the user name and `password1` as the password.
5. Click **Group** and click **New** to create a group named `ThreadUserGroup`, as defined in `weblogic.xml`.
6. Go back to the Users and click `testuser`. Click **Groups** and choose **ThreadUserGroup** to add the user as a member of that group.
7. Click **Save** and restart the application server.

### Testing the Trade Application on Oracle WebLogic

Test the newly deployed application by accessing the following web page from your web browser:

`http://localhost:7001/trade3Web`

You are prompted to enter a user name and password. Enter `testuser` as the user name and `password1` as the password.

To start loading the database with sample data, complete the following steps:

1. Select **Configuration** from the Trade home page.
2. Select **(Re)-populate Trade Database** to start loading the database with sample data.

This process takes a few minutes depending on the hardware configuration used. When finished, you should see 1000 quotes that are created and 500 users registered. On the left pane, click **Go Trade!** to browse the application.

If you click the **Configuration** link on the left side and click **Trade Scenario**, you can see how the application automatically navigates itself by clicking **Reload** in your browser.

This feature randomly selects the next action to be performed and populates all the data instead of the user doing it by hand. This feature greatly simplifies performance scripts. If you want to test this application's performance, you do not need to navigate to all the windows and enter all the data. You can point your stress tool to the URL

`http://localhost:7001/trade3Web/scenario` and configure your stress tool to see how many hits and how many concurrent users access this URL, and then start your test.

## 6.5.3 Migrating the Trade application

This section describes the steps that are performed to migrate the Trade application. You configure the Migration Toolkit in Rational Application Developer V8.5 to scan for WebLogic specific files and flag potential problems with Java source code. Then, you build the Trade application for WebSphere. We explain the issues and the solutions to the problems we experienced when we migrated the Trade application in our example.

Because of the size of this section, the following list details the subsections it contains:

- ▶ "Installing the Application Migration Tool - WebLogic to WebSphere" on page 141
- ▶ "Importing the Trade EAR file" on page 141
- ▶ "Configuring and running the Application Migration Tool" on page 146
- ▶ "Fixing errors reported by the Application Migration Tool" on page 150
- ▶ "Rerunning the Application Migration Tool" on page 155
- ▶ "Migrating the Trade Application J2EE level" on page 168
- ▶ "Configuring WebSphere Application Server V8.5" on page 170
- ▶ "Configuring JMS" on page 171
- ▶ "Creating a data source" on page 180
- ▶ "Migrating EJBs" on page 181



- ▶ “Specifying the web application references” on page 183
- ▶ “Configuring basic authentication on WebSphere” on page 185
- ▶ “Testing the application” on page 191

## Installing the Application Migration Tool - WebLogic to WebSphere

The Migration Toolkit is a set of tools that supports the migration of Java EE applications that are developed for competitive application servers to WebSphere Application Server V7, V8, or V8.5. Each tool targets a competitive application server and is an installable feature that supports Rational Application Developer and Eclipse. For WebLogic, you must install Application Migration Tool - WebLogic to WebSphere. For more information, see Chapter 4, “Installation and configuration of the Application Migration Tools” on page 89.

## Importing the Trade EAR file

We start the migration of the Trade application by importing its EAR file with the source code in to Rational Application Developer V8.5.

**Source code required:** The Application Migration Tool requires the presence of the source code to perform Java code analysis. The source code must be imported as a project in the Rational Application Developer.

Complete the following steps:

1. In the Rational Application Developer Enterprise Explorer perspective, right-click and select **Import** → **Import** → **Existing Projects into Workspace**.

2. Select the archive file, click **Browse**, and select the Trade EAR file. In our example, the file is in C:\ProjectShare\Trade3.zip (Figure 6-4).

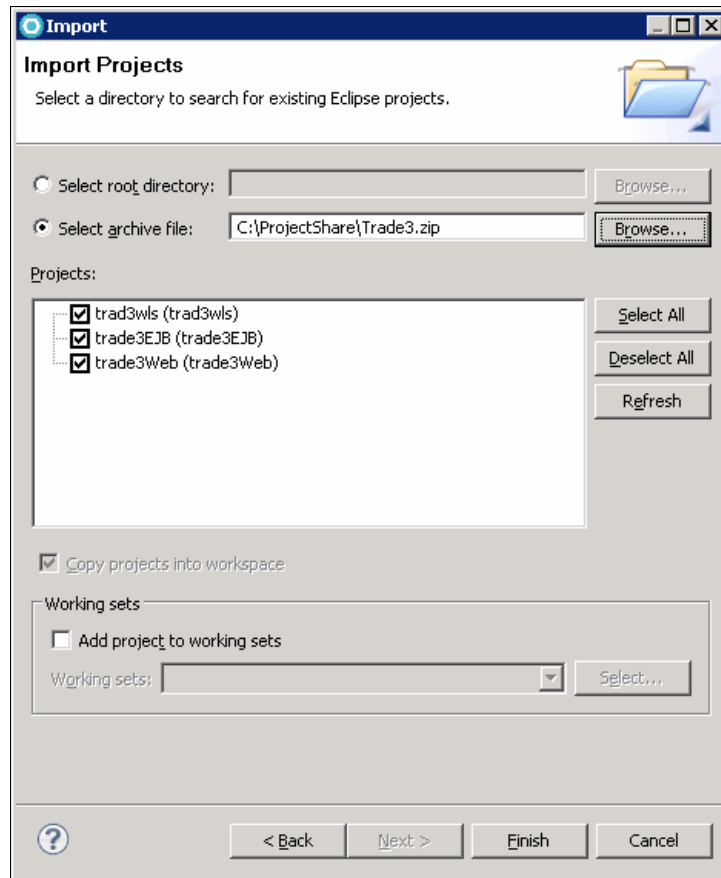


Figure 6-4 Existing Project Import

3. In the Ear Module and Utility JAR Projects window, ensure that all three modules are selected (Figure 6-5) and click **Finish**.

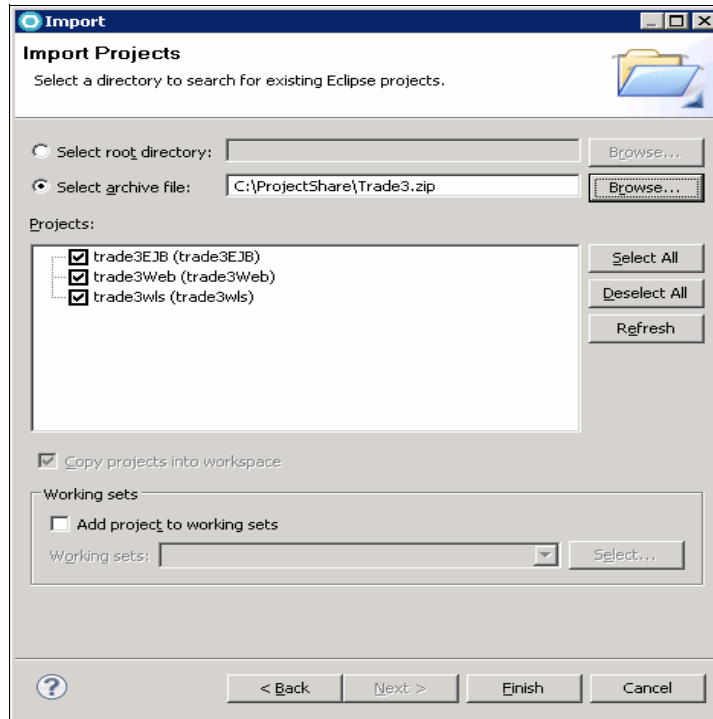


Figure 6-5 Trade Module Selections

4. The Workspace Migration wizard opens (Figure 6-6). Click **Next**.

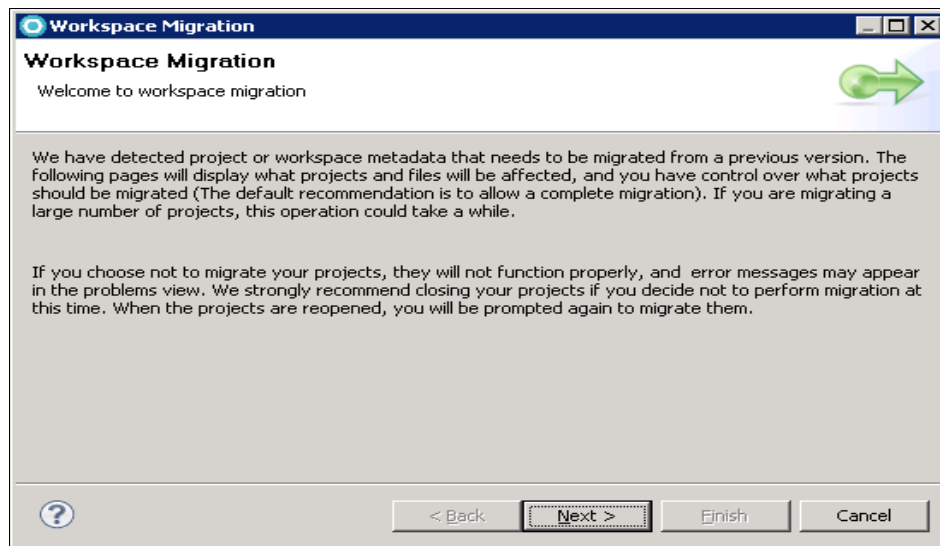


Figure 6-6 Workspace Migration

5. Ensure that the tradew1s EAR file is selected and click **Next**. You see that resources in the org.eclipse.wst.common.project.facet.core file are modified by the Workspace Migration wizard (Figure 6-7).

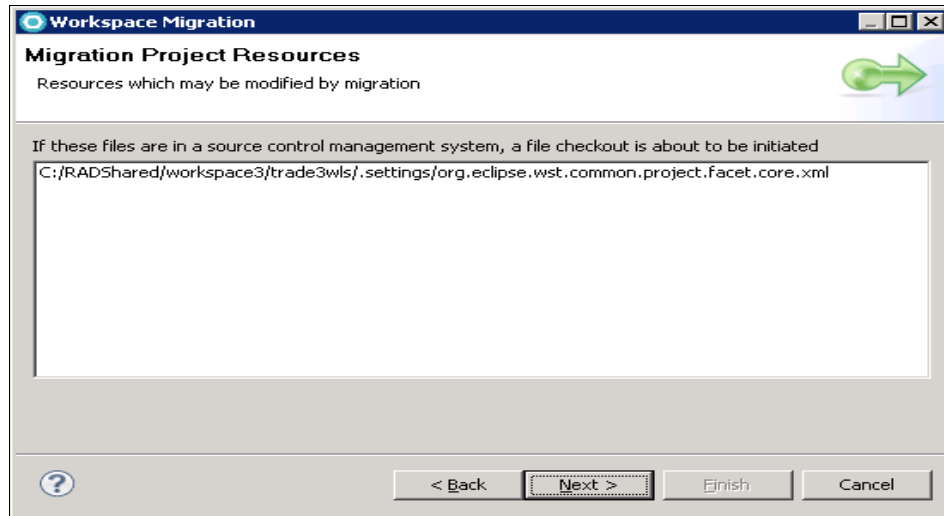


Figure 6-7 Project resource file in workspace

6. Look at the file (Example 6-3) before you continue. As you can see, the run time is Oracle WebLogic Server 10.3.6 and the J2EE version is 1.3.

*Example 6-3 org.eclipse.wst.common.project.facet.core file*

---

```
<?xml version="1.0" encoding="UTF-8"?>
<faceted-project>
  <runtime name="Oracle WebLogic Server 11gR1 (10.3.6)"/>
  <fixed facet="jst.ear"/>
  <installed facet="jst.ear" version="1.3"/>
</faceted-project>
```

---

- Click **Next**. To change Server Runtime from WebLogic to WebSphere 8.5, click the drop-down menu and choose **WebSphere Application Server 8.5** as New Server Runtime (Figure 6-8). Click **Next** and click **Finish**.

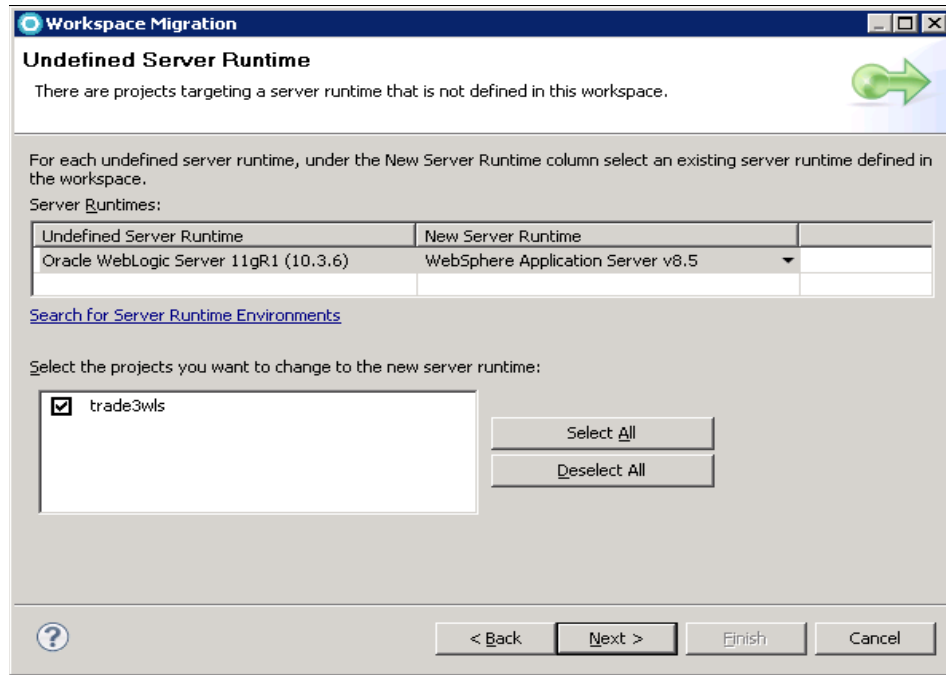


Figure 6-8 Define Server Runtime

- Click **OK** (Figure 6-9).

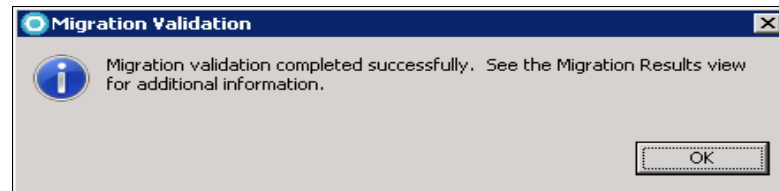


Figure 6-9 Migration Validation

**Tip:** By default, Rational Application Developer V8.5 automatically rebuilds the project after each change to the source code. To save time, we decided in our example to turn this feature off by clicking **Project** → **Build automatically**, because with this option turned on, Rational Application Developer V8.5 builds the project automatically with every change. If you turn it off, you need to build the project manually, as shown in step 6 on page 152.

You should have the following projects in the Enterprise Explorer view:

- ▶ trade3EJB  
This project contains the Trade EJBs.
- ▶ trade3Web  
The Trade web application contains servlets and JSPs.
- ▶ trade3wls  
This project contains the EAR application.

## Configuring and running the Application Migration Tool

In this section, you configure the Application Migration Tool analysis capabilities to scan the WebLogic Trade application files and provide input as to what needs to be fixed in the application before it can run on WebSphere.

The Application Migration Tool needs to be set up to look for WebLogic code-related issues. However, it can also be run to do a complete Java code review. The tool provides quick fixes, where available, for issues it flags as violations of the migration rules.

**Important:** The tool does not provide a quick fix for every possible scenario. Whenever a rule violation occurs and there is a quick fix available, the rule icon includes a light bulb icon and the Quick fix option is available in the context menu.

To configure and run the Application Migration Tool, complete the following steps:

1. Select **Analysis** from the Run menu to display the Application Migration Tool main configuration dialog box (Figure 6-10).

**Tip:** If you start a new workspace, the Software Analyzer does not have the New\_Configuration suboption. In this case, select **Software Analyzer** and click the **New icon** on the toolbar (the first icon with the "+" sign).

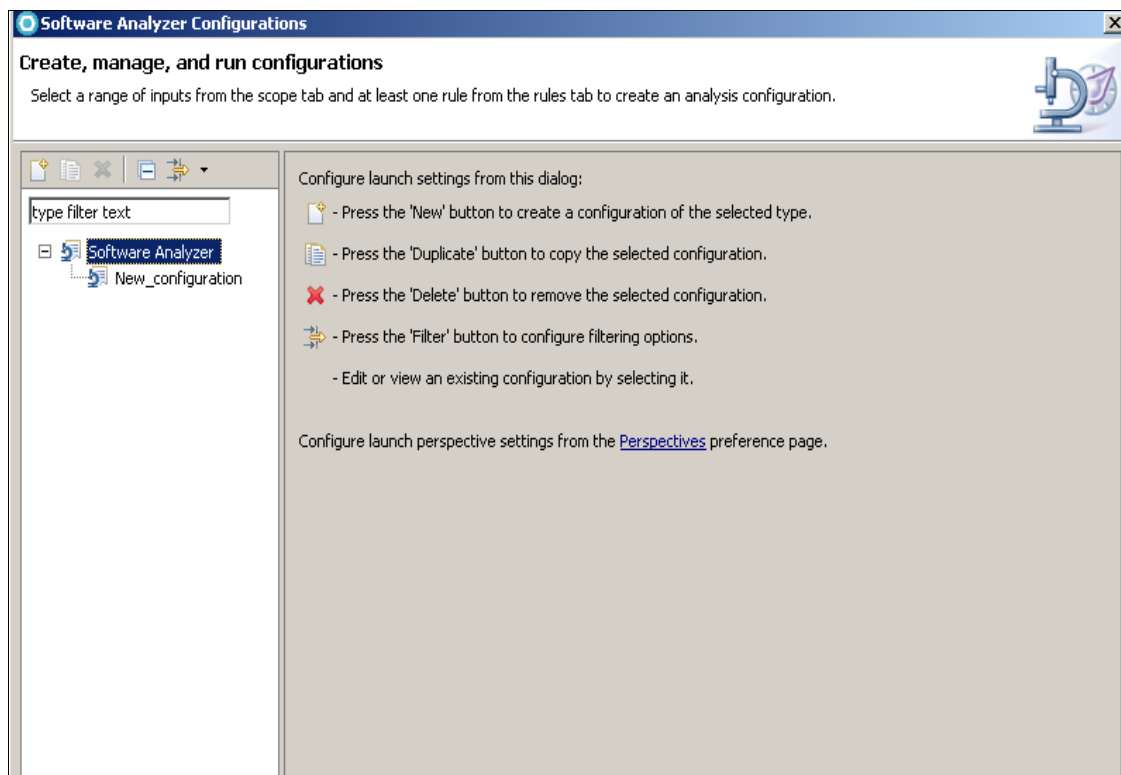


Figure 6-10 Application Migration Tool configuration

2. Expand the **Software Analyzer** and click **New\_configuration**. The right side changes to show an explanation of the basic configuration options available.
3. In the Name field, enter AppMigration as the name of the configuration.

4. In the Scope tab, leave the **Analyze entire workspace** radio button selected (Figure 6-11).

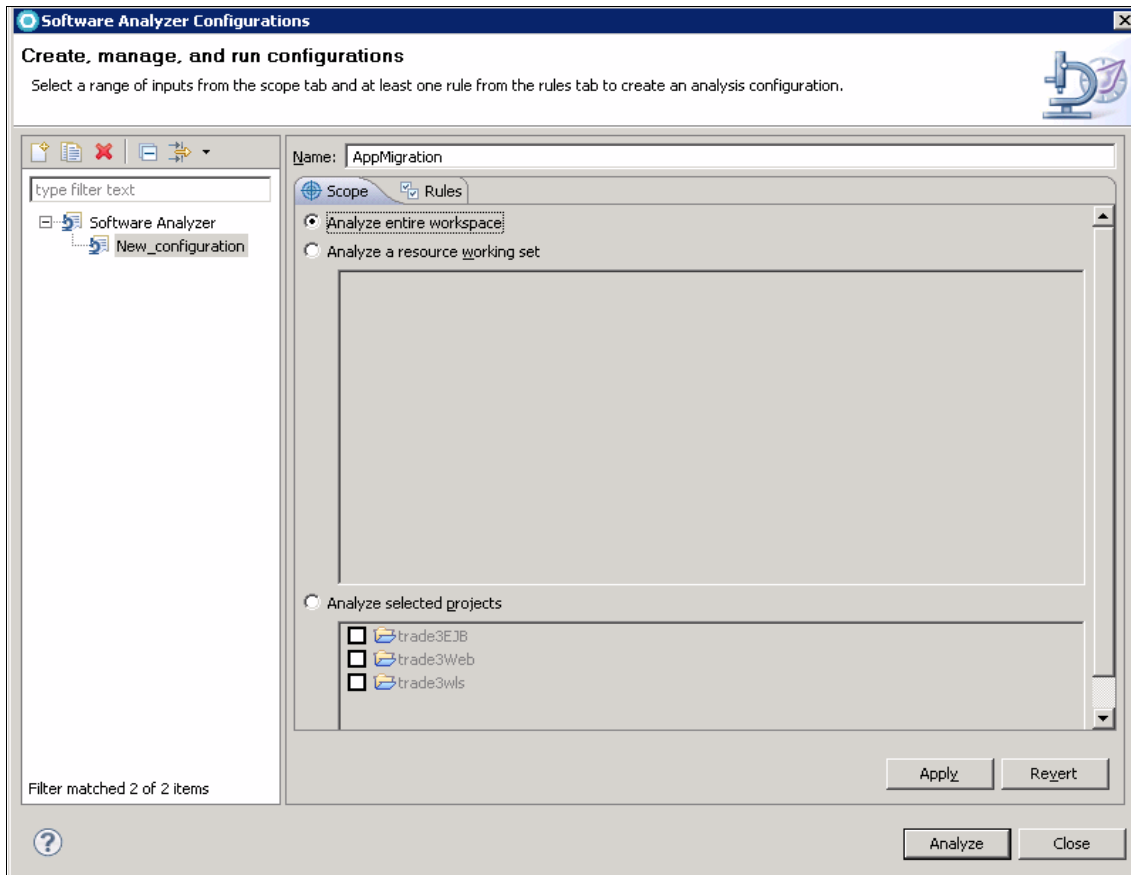


Figure 6-11 Migration tool workspace scope settings

5. Click the **Rules** tab.

6. Select **WebLogic Application Migration** from the Rules Sets drop-down menu, and click **Set**. (Figure 6-12).

**Tip:** The default is that no rules are selected until the user selects the ones that are of interest. Rules can be chosen from rule sets or individually. After the configuration is selected and saved by clicking **Apply**, the configuration can be reused.

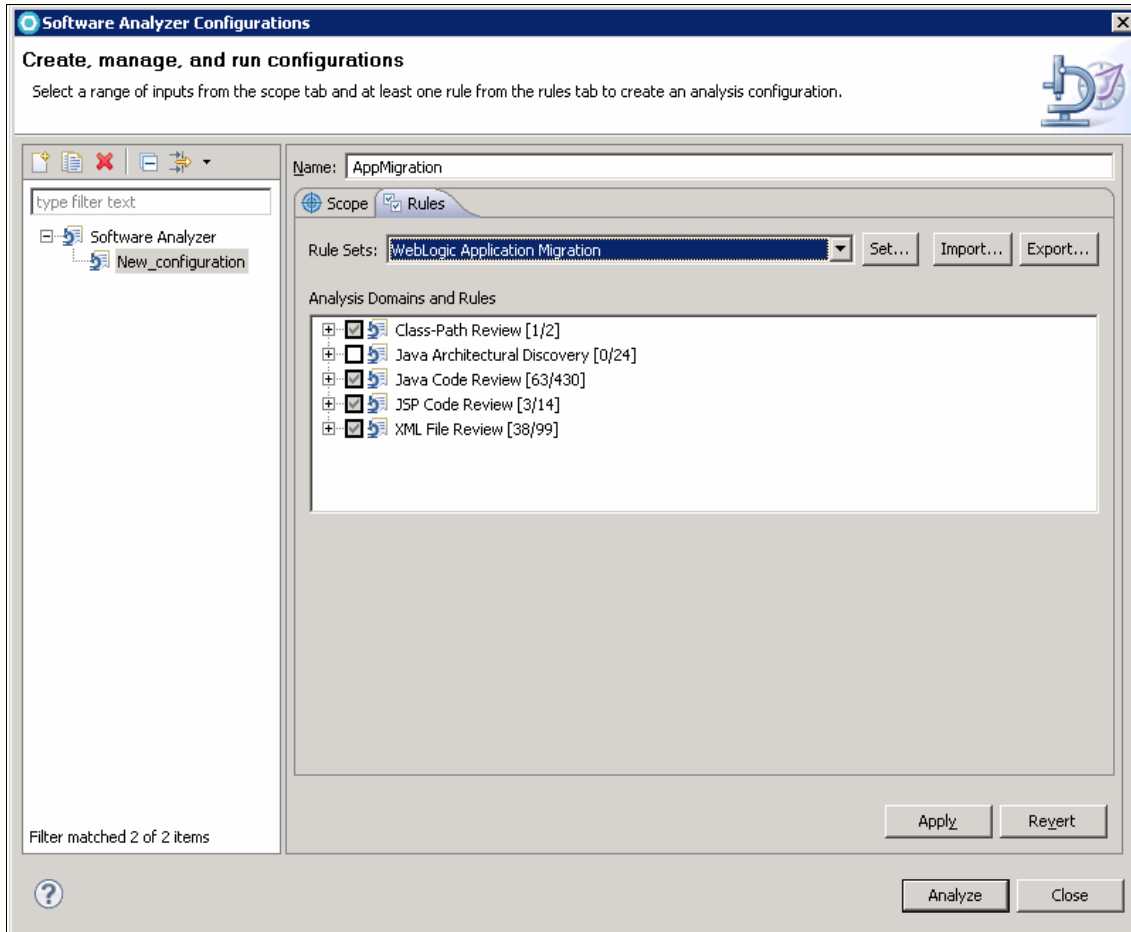


Figure 6-12 Migration Tool rule settings



7. You see the Rule set configuration dialog box. Leave **Source Java Version** and **Target Java Version** selected(Figure 6-13) and click **OK**.

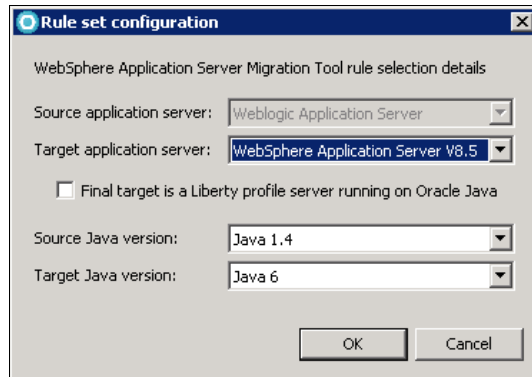


Figure 6-13 Rule set configuration

The Application Migration Tool and the rules set is now configured. We are ready to run the analysis against the Trade application. You can expand any of the analysis domains to see which rules are selected.

8. Click **Analyze** to analyze all the projects in the workspace.

The Software Analyzer View is opened at the bottom of the window. It contains four tabs. Each tab corresponds to one of the analysis domains that are selected based on the Rule Set selection made earlier.

- The Class-Path Review tab lists any class path related issues.
- The Java Code Review tab lists Java code-related issues.
- The XML File Review tab lists issues with deployment descriptors and the presence of any WebLogic specific files.
- The JSP Code Review tab lists issues that are found in JSP files, commonly associated with web projects.

9. Click the **XML File Review** tab (Figure 6-14).

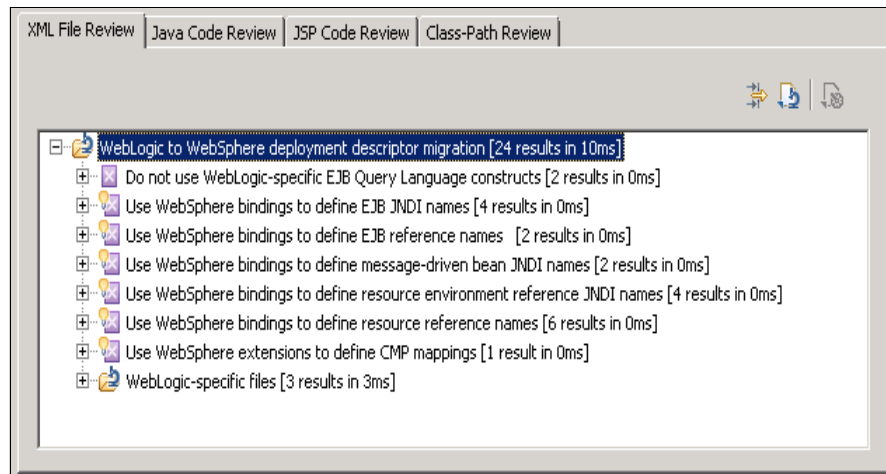


Figure 6-14 Software Analyzer View

As you can see, the tool scanned the XML files in the project and flagged the potential problems with WebLogic deployment descriptors by category/rule violation. For example, the tool points out that EJB and message-driven bean (MDB) Java Naming and Directory Interface (JNDI) references are defined in a WebSphere binding file. It also detected WebLogic specific files. You can expand each of the rules to see the files in question and which project to which they belong.

10. Click the **Java Code Review** tab (Figure 6-15).

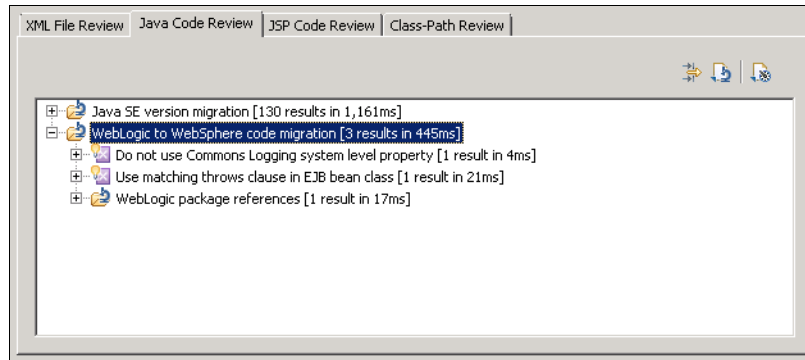


Figure 6-15 Java Code Review tab

The tool found three rule violations. The first one is related to the usage of the Commons Logging system level property in `Log.java`. The second one indicates that you should use the `throws Exception` method instead of `throws javax.jms.JMSEException, Exception` in `TreadBean.java`. The third one is related to the existing WebLogic-specific package in `Log.java`; the tool flags only this issue, but does not provide a quick fix option.

**Tip:** You can right-click any of the flagged files under a specific rule to see a list of available options in the menu. For example, under the Java Code Review tab, expand the WebLogic package references rule, right-click any of the files in the list, and select **View Results**. The Java file in question is opened and the cursor is at the code problem line.

## Fixing errors reported by the Application Migration Tool

In this section, you fix the issues reported by the Application Migration tool and the problems reported by Rational Application Developer when you imported the Trade application.

Complete the following steps:

1. Open the **Problems view** and analyze all the errors and warnings reported by Rational Application Developer V8.5.
2. Fix the errors reported under the Java Code Review tab.

Rational Application Developer V8.5 reports errors (Figure 6-16) for the web application because the project depends on classes that are in the EJB project.

Description	Resource	Path	Location	Type
Errors (100 of 708 items)				
accountData cannot be resolved	account.jsp	/trade3Web/Web...	line 100	JSP Problem
accountData cannot be resolved	account.jsp	/trade3Web/Web...	line 103	JSP Problem
accountData cannot be resolved	account.jsp	/trade3Web/Web...	line 109	JSP Problem
accountData cannot be resolved	account.jsp	/trade3Web/Web...	line 113	JSP Problem
accountData cannot be resolved	account.jsp	/trade3Web/Web...	line 117	JSP Problem
accountData cannot be resolved	account.jsp	/trade3Web/Web...	line 122	JSP Problem
accountData cannot be resolved	account.jsp	/trade3Web/Web...	line 126	JSP Problem
accountData cannot be resolved	account.jsp	/trade3Web/Web...	line 130	JSP Problem
accountData cannot be resolved	accountImg...	/trade3Web/Web...	line 99	JSP Problem
accountData cannot be resolved	accountImg...	/trade3Web/Web...	line 102	JSP Problem

Figure 6-16 Problems view

To fix the problem, complete the following steps:

1. Right-click the trade3Web application project and select **Properties**.
2. From the left pane in the trade3Web properties window, click **Java Build Path**.
3. Select the Projects tab and select **Add**.
4. In the Required Project Selection dialog box, select **trade3EJB** and click **OK**.

5. Click **OK** in the Properties dialog box to close it (Figure 6-17).

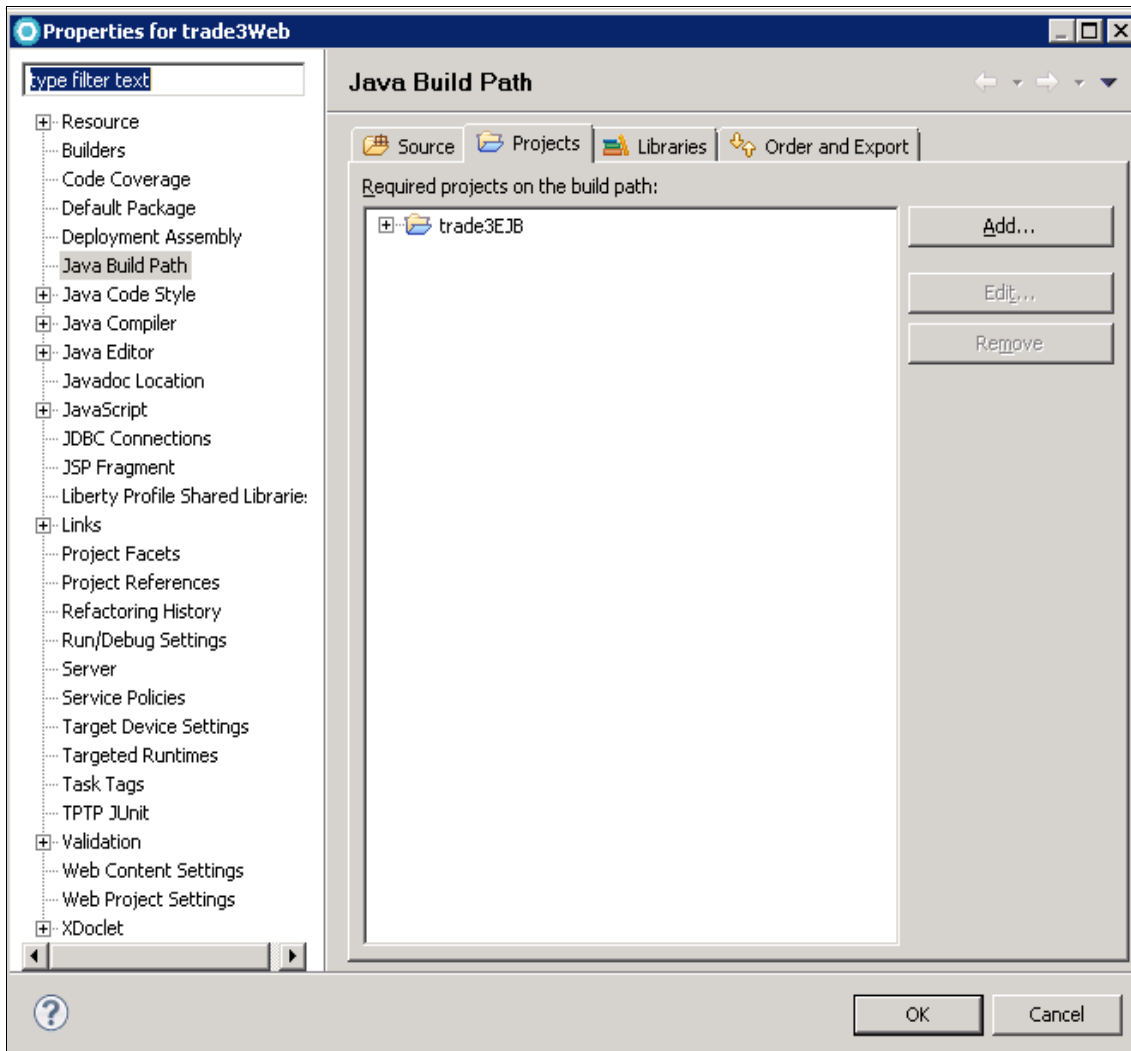


Figure 6-17 Setting Java Build Path

6. Switch to the Problems view (Figure 6-18). After you build the project automatically or manually, you do not see any dependency errors.

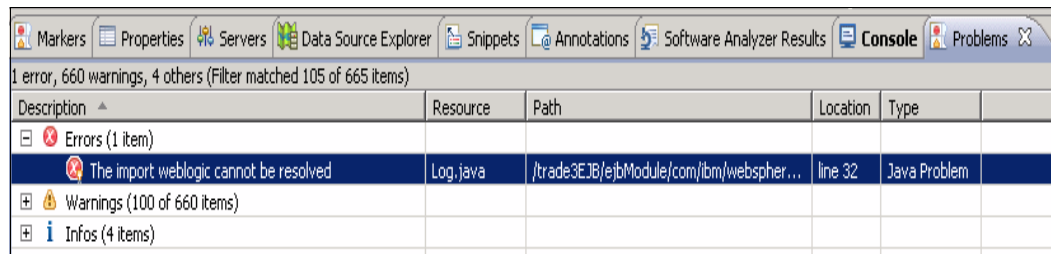
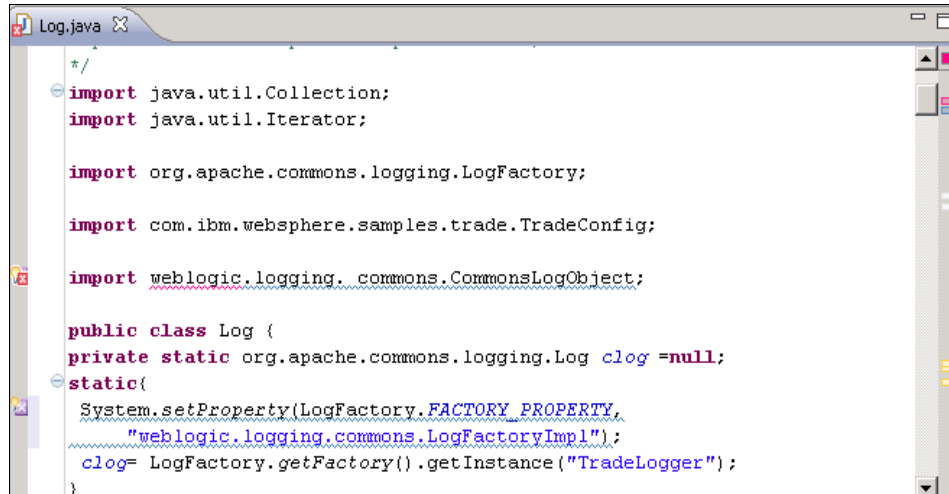


Figure 6-18 Problems view

**Tip:** To see the Problem view on frame, click **Window** → **Show View** → **Other type Problems**, choose the view, and click **OK**. When you start a new workspace, the Software Analyzer does not have the **New\_Configuration** suboption. In this case, highlight **Software Analyzer** and click the new icon on the toolbar (the first icon with the "+" sign).

7. The error reported here is reported in Software Analyzer Results. Go back to Software Analyzer Results. In the Java Code Review tab, expand the **WebLogic package references** category and right-click the `Log.java` entry. Click **View Result** (Figure 6-19).



```
Log.java
*/
import java.util.Collection;
import java.util.Iterator;

import org.apache.commons.logging.LogFactory;

import com.ibm.websphere.samples.trade.TradeConfig;

import weblogic.logging.commons.CommonsLogObject;

public class Log {
    private static org.apache.commons.logging.Log clog = null;
    static{
        System.setProperty(LogFactory.FACTORY_PROPERTY,
            "weblogic.logging.commons.LogFactoryImpl");
        clog= LogFactory.getFactory().getInstance("TradeLogger");
    }
}
```

Figure 6-19 `Log.java`

8. Delete the **import weblogic.logging.commons.CommonsLogObject;** line, as the package is WebLogic-specific and not used in the code. Click **Save** and go back to Problems tab. As you can see, there is no error reported anymore.

**Tool functionality:** We added the package in the code to demonstrate that the tool can find the vendor-specific package in the project if you import the project with the source codes.

9. From the Project menu, clear the **Build Automatically** option.
10. From the Project menu, select **Clean**.

11. Select the **Clean all projects** radio button and click **OK** (Figure 6-20).

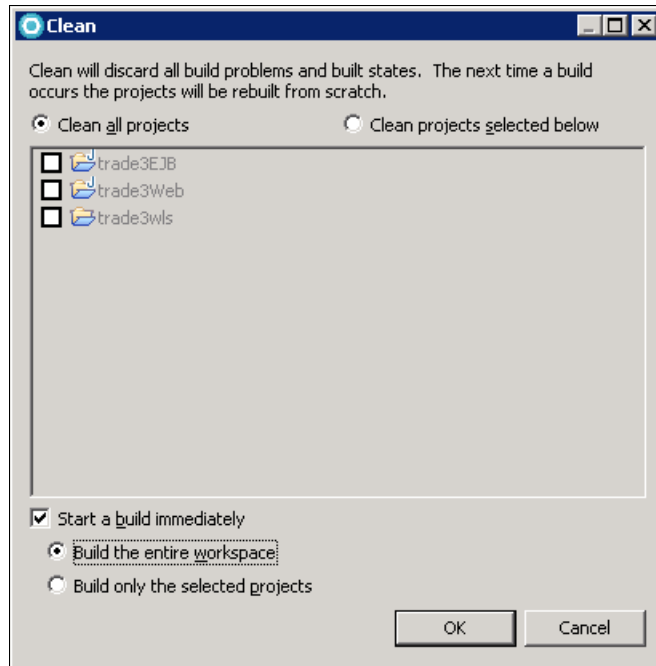


Figure 6-20 Project Clean dialog box

12. From the Project menu, select the **Build Automatically** option. This option ensures that projects are automatically built as you make changes.

All errors reported by Rational Application Developer are now fixed.

## Rerunning the Application Migration Tool

You should rerun the Application Migration Tool after you make changes to the source code. To do so, complete the following steps:

1. From the Eclipse Run menu, click **Analyze Last Launched** (Figure 6-21).

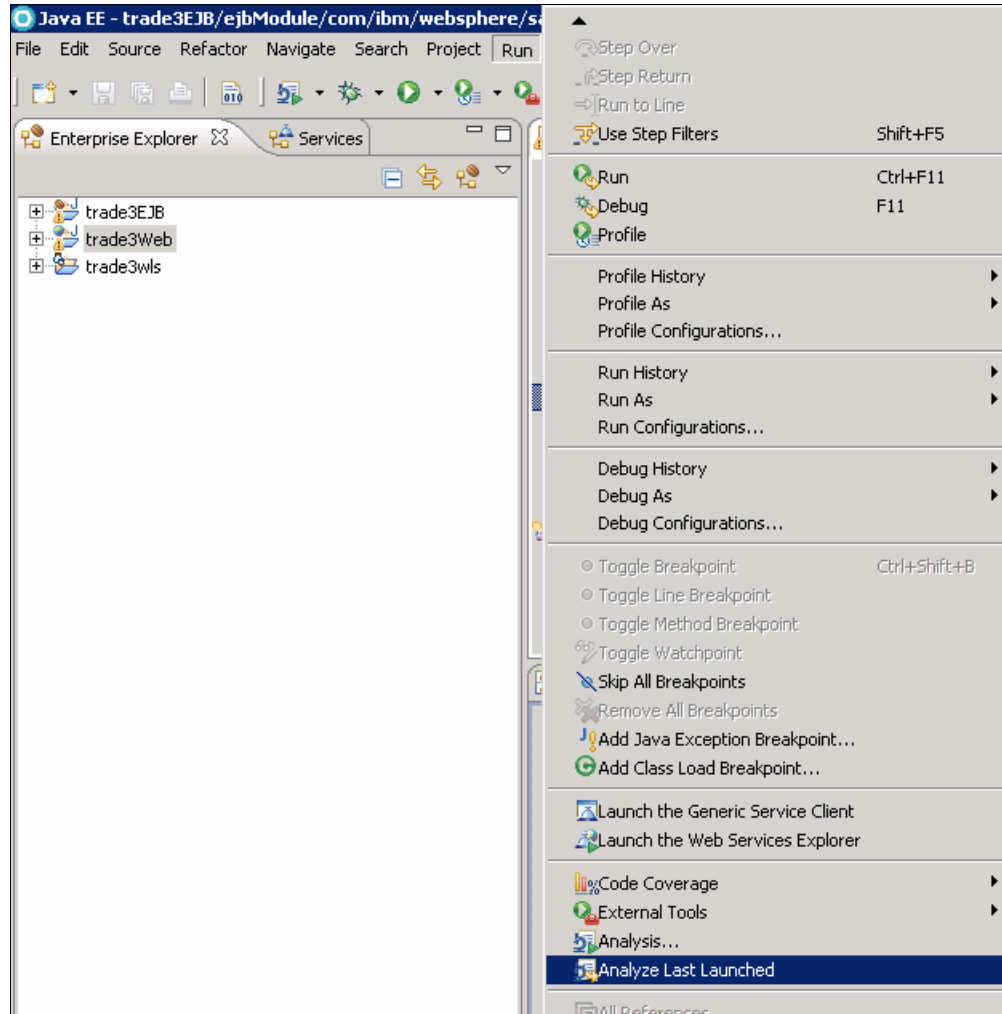


Figure 6-21 Migration tool analysis

This action reruns the analysis tool with the saved configuration. The focus is returned to the Software Analyzer Results view. The tool creates a report in the left frame of the view with the date and time stamp.

- Click the **Java Code Review** tab. Expand the **Do not use Commons Logging system level property** under the **WebLogic to WebSphere code migration** category (Figure 6-22).

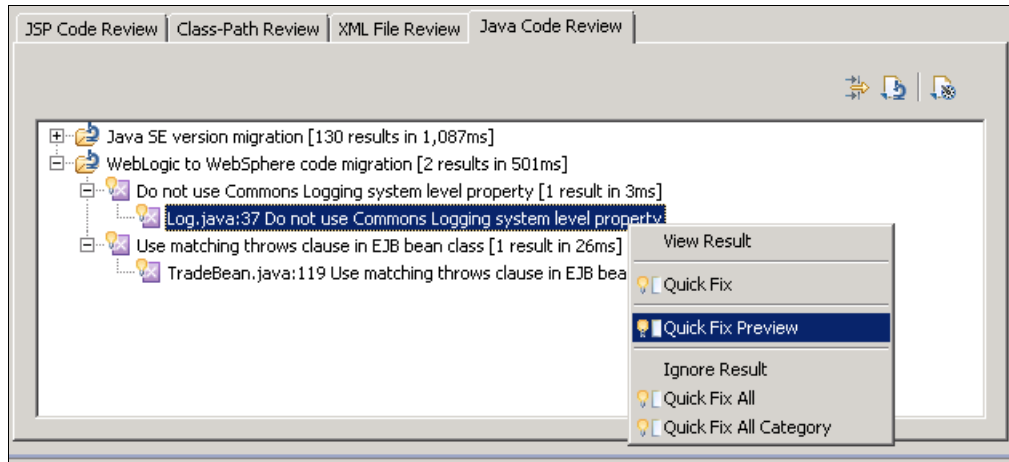


Figure 6-22 Java Code Review tab

- Right-click the **Log.java** entry and select **Quick Fix Preview**. See Figure 6-22.
- The Quick Preview editor tools window opens (Figure 6-22). Examine the code changes before they are committed. In this case, there is no “weblogic.logging.commons.LogFactory” system level property in the left pane, so the tool removes this line from Log.java after the quick fix (Figure 6-23).

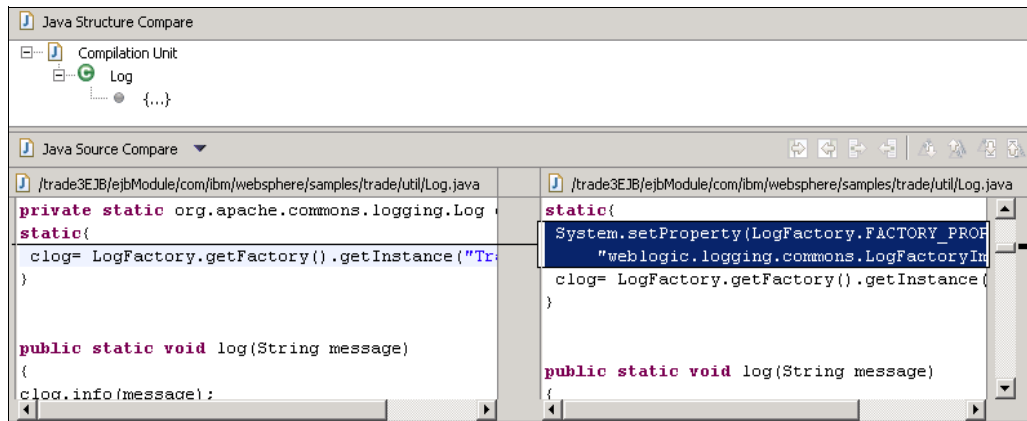


Figure 6-23 Quick fix preview



- Highlight the **Log.java** entry under the Do not use Commons Logging system level property category. Right-click and select **Quick Fix** (Figure 6-24).

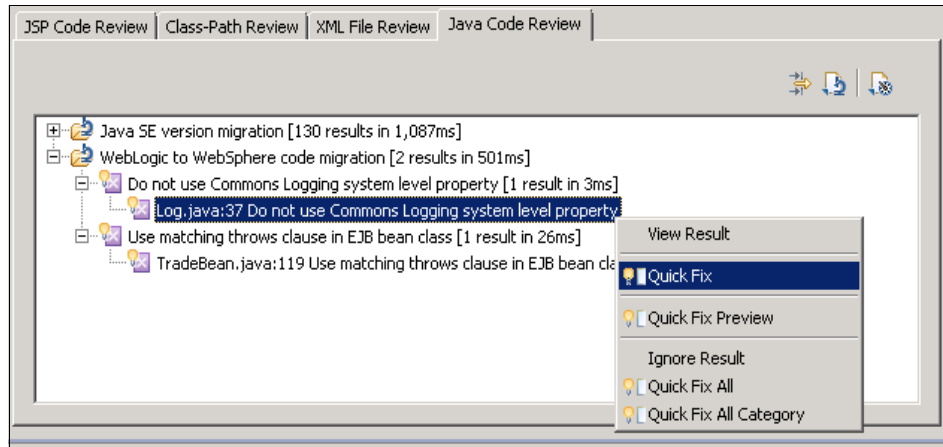


Figure 6-24 Java Code Review tab

- Click the **Java Code Review** tab. Expand the **TradeBean.java** entry under the Use matching throws clause in EJB bean class category. Right-click the **TradeBean.java** entry and select **Quick Fix Preview** (Figure 6-25).

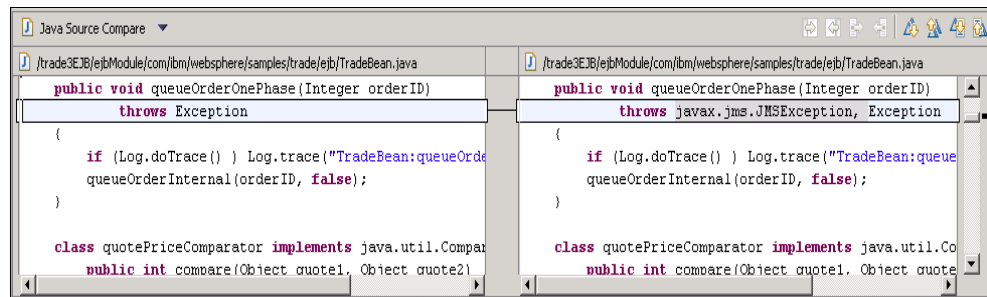


Figure 6-25 TradeBean.java

- In this case, the tool shows that it replaces the throws `javax.jms.JMSEException, Exception` with the throws `javax.jms.JMSEException, Exception` in Tradebean.java.
- Highlight the **TradeBean.java** entry under Use matching throws clause in EJB bean class category. Right-click the **TradeBean.java** and select **Quick Fix**.

The remaining XML violations errors are related to vendor-specific deployment descriptors, so review Figure 6-26, which shows the WebSphere deployment descriptors that correspond to WebLogic deployment descriptors.

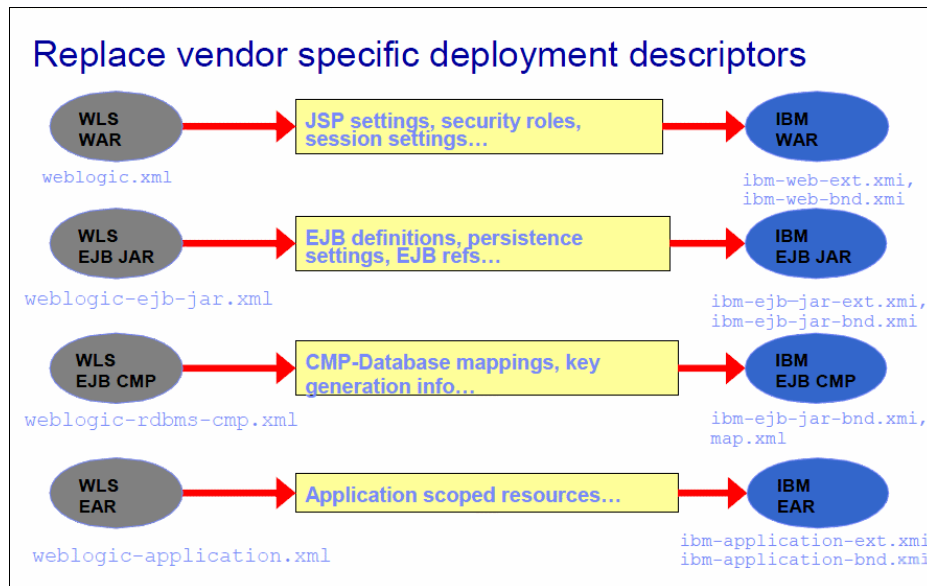


Figure 6-26 Vendor-specific deployment descriptors

**J2EE:** For J2EE 1.4 and prior versions, bindings and extensions are created in XMI files. In Java EE 5 and later, the bindings and extensions are in XML format files.

- In the XML File Review tab, expand the **Use WebSphere bindings to define EJB JNDI names** category (Figure 6-27).

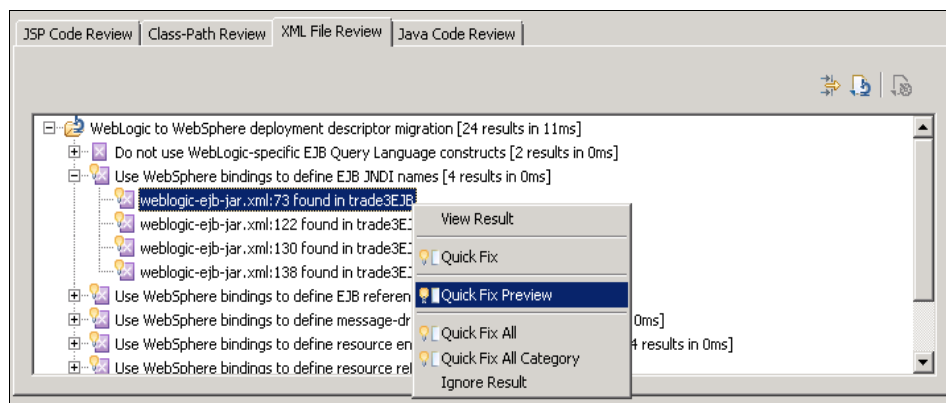


Figure 6-27 XML File Review tab

- Right-click the **weblogic-ejb-jar.xml** entry and select **Quick Fix Preview** (Figure 6-30 on page 160).

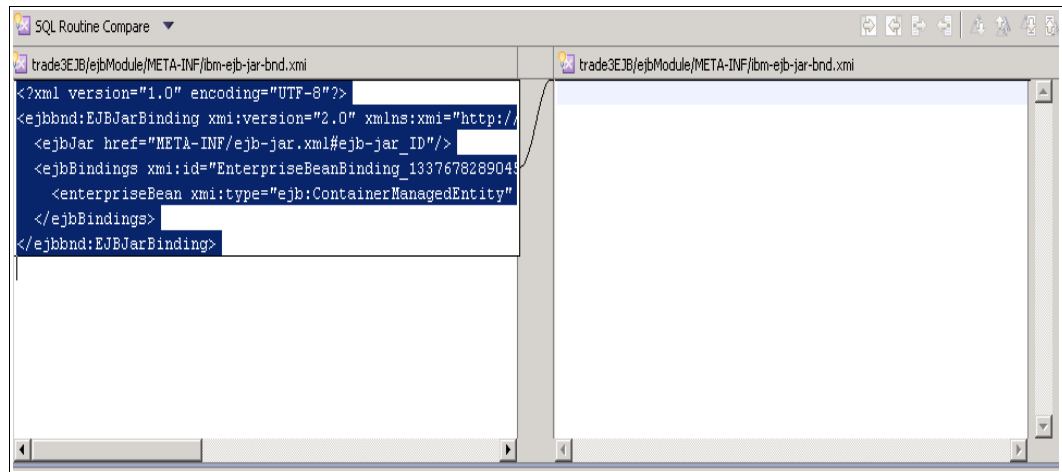


Figure 6-28 Quick Fix Preview

The Quick Preview Editor tool is displayed. Examine the code changes before they are committed. In this case, the tool creates the `ibm-ejb-jar-bnd.xml` binding file and places it in to the EJB module. This file did not exist because this application is a WebLogic application.

- Highlight the **Use WebSphere bindings to define EJB JNDI names** category, right-click, and select **Quick Fix All** (Figure 6-29).

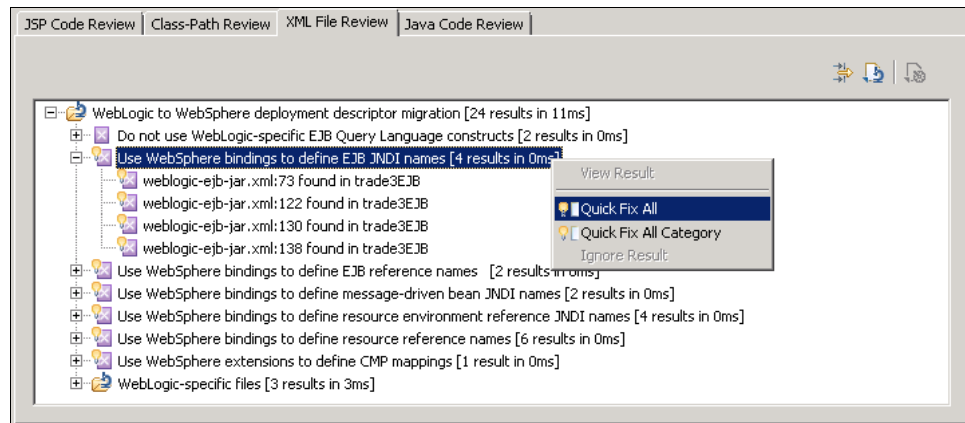


Figure 6-29 EJB JNDI Quick Fix

**Important:** If you select **Quick Fix All**, it applies every quick fix applicable in the analysis. This feature is good if you already know what changes the tool does to the application. This action is not suggested for a first time application migration.

The WebSphere binding file is created and the EJB JNDI names are copied from the corresponding WebLogic deployment descriptor. Double-click the file and explore its contents (Figure 6-30).

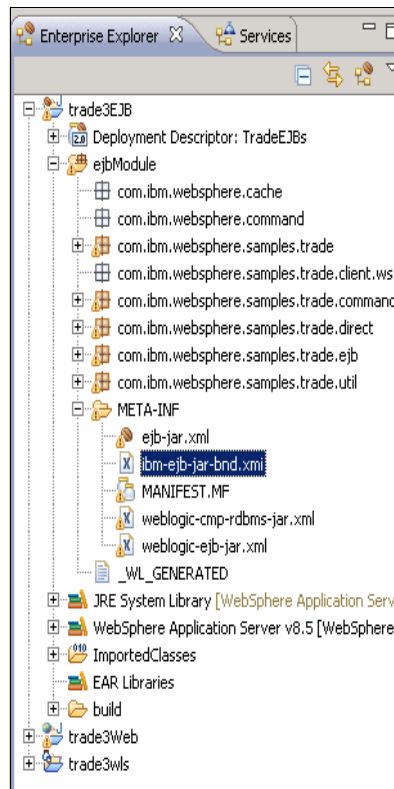


Figure 6-30 WebSphere Binding file

12. In the XML File Review tab, select the **Use WebSphere binding to define EJB reference names** category, right-click, and select **Quick Fix All**.

This rule flags EJB references found in `weblogic.xml` and `weblogic-ejb-jar.xml` if the corresponding quick fix is not already applied. After a quick fix is applied, the WebLogic XML is marked with a comment that indicates that it is migrated. This quick fix is used to determine whether to flag the issue during analysis. It can be used at the end of migration to identify how much of the XML file is migrated.

The quick fix that is provided for this rule takes the EJB reference information that is defined in the WebLogic-specific deployment descriptors and migrates it to the standard deployment descriptors and the WebSphere bindings files `ibm-ejb-jar-bnd.xmi` or `ibm-web-bnd.xmi`.

13. In the XML File Review tab, select the **Use WebSphere bindings to define Message Driven beans JNDI names** category, right-click, and select **Quick Fix All**.

The provided quick fix copies the destination JNDI name from the `weblogic-ejb-jar.xml` file to the `ibm-ejb-jar-bnd.xmi` file. The quick fix, however, does not set the `ActivationSpec` JNDI name.

For more information about manual intervention, highlight the rule and press the F1 function key. This action opens the Help window. Click **See details** to show the preferred practices for manually addressing issues such as the `ActivationSpec` JNDI name.

14. Repeat step 13 on page 160 for the **Use WebSphere bindings to define resource environment reference JNDI names** and **Use WebSphere bindings to define resource environment reference names** categories. From the context menu, select **Quick Fix All**.
15. Expand the remaining rules (Figure 6-31).

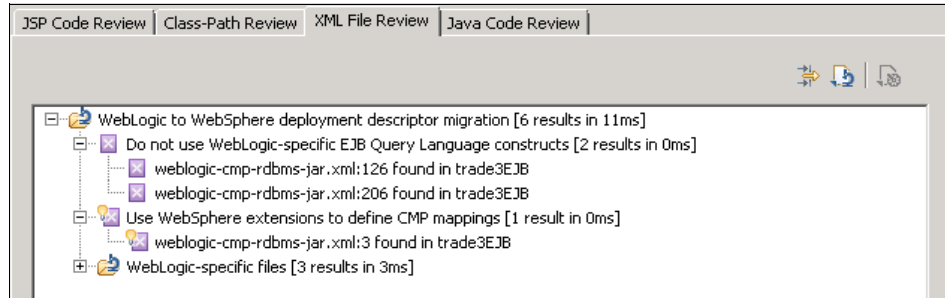


Figure 6-31 XML File Review tab

16. Review the help for the WebLogic Query language constructs by highlighting one of the rules and pressing the F1 function key. This action opens the Help window. After you open the window, click **See details**.
17. Double-click the **weblogic-cmp-rdbms-jar.xml:126 rule** in trade3EJB. This action shows the weblogic-ql construct (Example 6-4). You should see that ORDERBY is used and is a WebLogic construct and not an ejb-ql construct. The ejb-ql equivalent is ORDER BY, as shown in the help, but the weblogic-ql does not accept this construct. Edit the ejb-jar.xml file using step 23 on page 165.

Example 6-4 WebLogic Query Language construct

```

<weblogic-query>
  <query-method>
    <method-name>findByUserID</method-name>
    <method-params>
      <method-param>java.lang.String</method-param>
    </method-params>
  </query-method>
  <weblogic-ql>SELECT OBJECT(o) FROM Orders o WHERE
o.account.profile.userID = ?1 ORDERBY o.orderID DESC</weblogic-ql>
</weblogic-query>
</weblogic-rdbms-bean>

```

Table 6-12 explains the WebLogic specific files that are in the JNDI name and CMP fields for EJB.

Table 6-12 Explanation of WebLogic specific file names

File	Description
weblogic-ejb-jar.xml	This file is specific to WebLogic. It defines the weblogicenterprise bean. The JNDI name for an EJB is specified in the weblogic-ejbjar.xml deployment descriptor with the jndi-name element.

File	Description
weblogic-cmp-rdbms-jar.xml	Defines the database persistence information for a CMP entity EJB. It includes the table name for an entity EJB, the data source to connect to the database, and the columns corresponding to the entity EJB CMP fields.

18. To convert the CMP Mappings, right-click the rule and select **Quick Fix**.

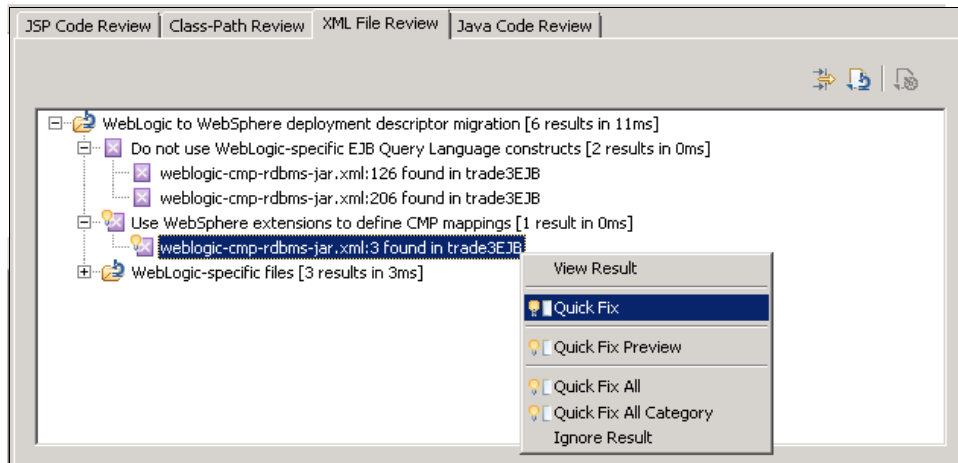


Figure 6-32 Convert the CMP Mappings file

19. The quick fix creates the Map.mapxmi file, which is used by WebSphere to map CMP fields to EJB to database table columns.

20. Open the mapping editor to verify that everything is mapped. The file is in the following folder:

/trade3EJB/ejbModule/META-INF/backends/DB2UDBNT\_V95\_1

See Figure 6-33.

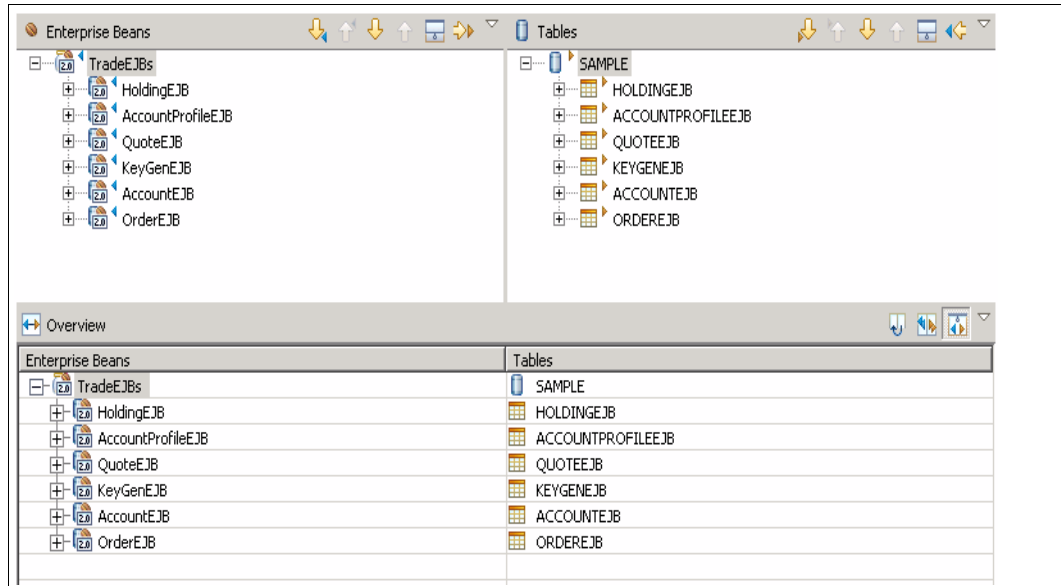


Figure 6-33 CMP mapping file

The database schema is imported and in the following folder:

/trade3EJB/Data Models/SAMPLE.dbm





Open the Map.mapxml file and enable the **Show only the unmapped objects** option (Figure 6-36).

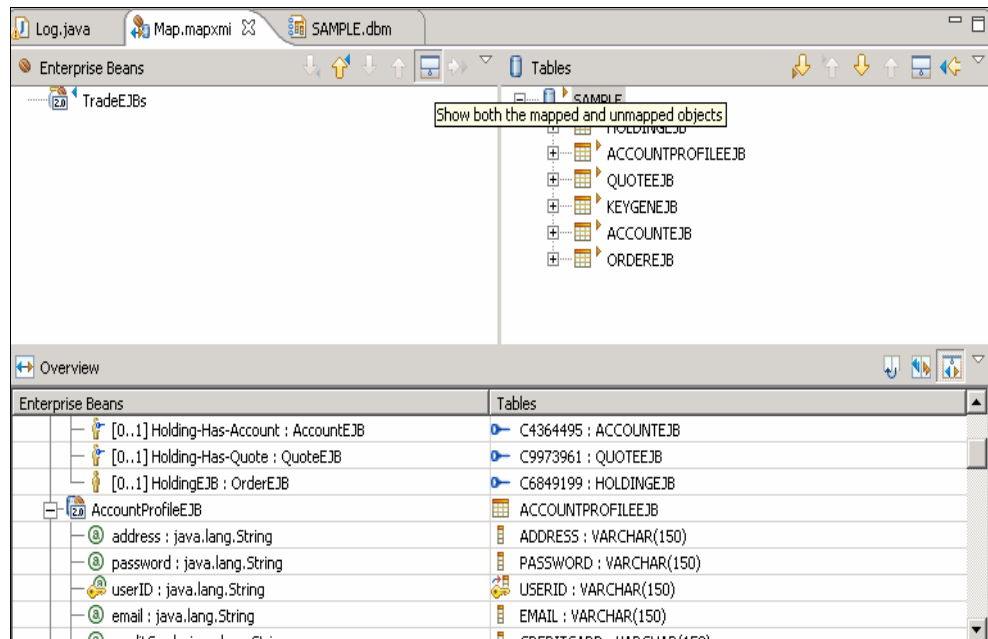


Figure 6-36 Show unmapped objects

23. If you want to create CMP mappings manually, complete the steps in “Creating the CMP mappings manually” on page 212 that were performed to migrate the MVC application.

**CMP mappings:** The Application Migration Tool performs CMP mappings by examining the definitions in the `weblogic-cmp-rdbms-jar.xml` file and creates the `Map.mapxml` mapping editor and the database model named `SAMPLE.dbm`. If you want to map the CMP fields of the EJB object to database table and columns you explicitly specify, complete the steps in “Creating the CMP mappings manually” on page 212.

24. Switch to the **Source** tab of the `ejb-jar.xml` and change `ORDERBY` to `ORDER BY`. Save and close the EJB deployment descriptor editor.
25. Return to the Software Analyzer view, right-click the `weblogic-ql` rule, and select **Ignore Result**. This action removes the rule from the view. It also enters a comment in the `weblogic-cmp-rdbms-jar.xml` file to tell the analyzer to ignore the rule in the next iteration.
26. Run a project clean and review the Problems view. If you missed changing `ORDERBY` to `ORDER BY`, Rational Application Developer flag this situation as an error with a red X. Open the **Source** tab of the deployment descriptor and search for “`ORDERBY`” and make the appropriate changes.

27. In the Class-Path Review tab, use the MANIFEST.MF file for the application class path category (Figure 6-37) and click **Quick Fix Preview** to see what the Application Migration tool changes in MANIFEST.MF of trade3EJB.

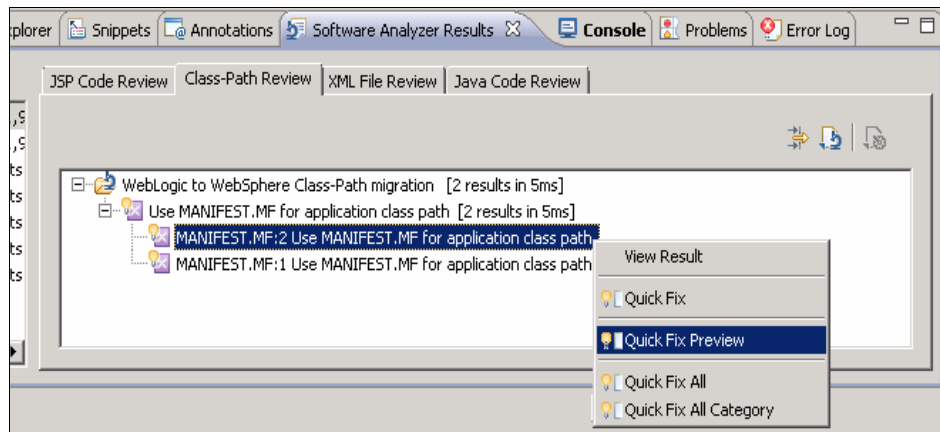


Figure 6-37 Class-Path Review tab

As you can see in Figure 6-38, the tool places the `wlcommons-logging.jar` definition into the MANIFEST.MF file of the trade3EJB because the trade3wls EAR file contains the `wlcommons-logging.jar` file under `/trade3wls/APP-INF/lib`. After examine the changes, click **Quick Fix**. Repeat the same step for the other result.

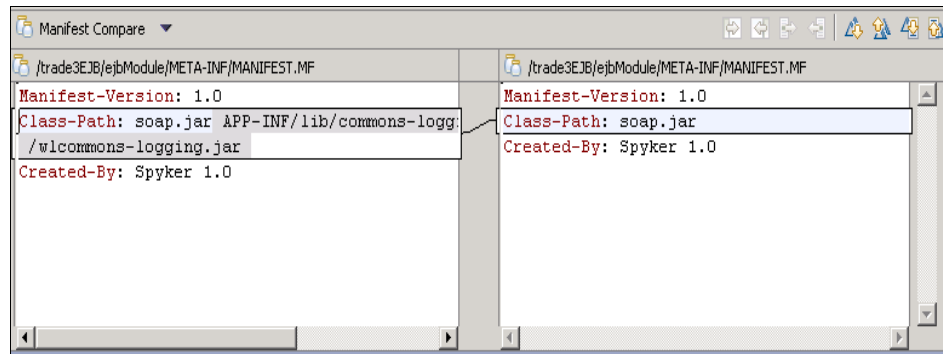


Figure 6-38 Quick Fix View of the MANIFEST.MF file

28. In the XML Review tab, expand the WebLogic specific files category and expand all suboptions. Those options are the WebLogic specific deployment descriptors files. Delete them because you migrated all of their entries to the corresponding WebSphere binding files.

29. In the Enterprise Explorer view, expand the EJB project and browse to the /trade3EJB/ejbModule/META-INF folder and delete the following files:

- weblogic-ejb-jar.xml
- weblogic-cmp-rdbms-jar.xml

See Figure 6-39.

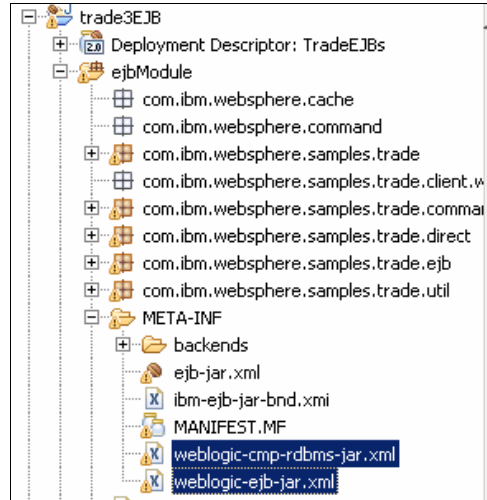


Figure 6-39 WebLogic EJB deployment descriptors

30. In the Enterprise Explorer view, expand the **trade3web** project and browse to the /trade3web/webContents/WEB-INF folder. Delete the weblogic.xml file. See Figure 6-40.

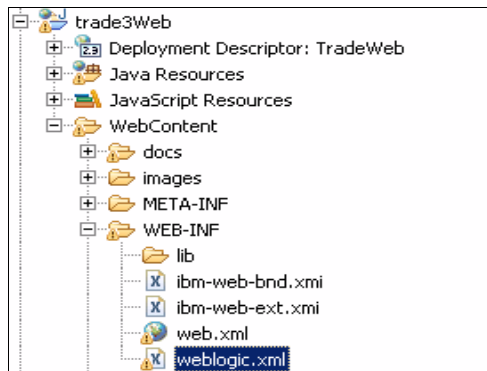


Figure 6-40 WebLogic web deployment descriptor

All errors reported by the Migration Tool and Rational Application Developer v8.5 should now be fixed.

## Migrating the Trade Application J2EE level

As part of this migration example, we migrate this application's structure from J2EE 1.3 to J2EE 1.4. Review each window before you go to the next step.

### Important:

- ▶ Doing a J2EE migration is not always required because of the WebSphere product's and specifications compatibility with earlier versions.
- ▶ The Java EE Specification Migration wizard used is only available in Rational Application Developer.

Complete the following steps:

1. In the Java EE perspective, right-click **trade3wls** and select **Java EE** → **Specification Migration wizard** (Figure 6-41).

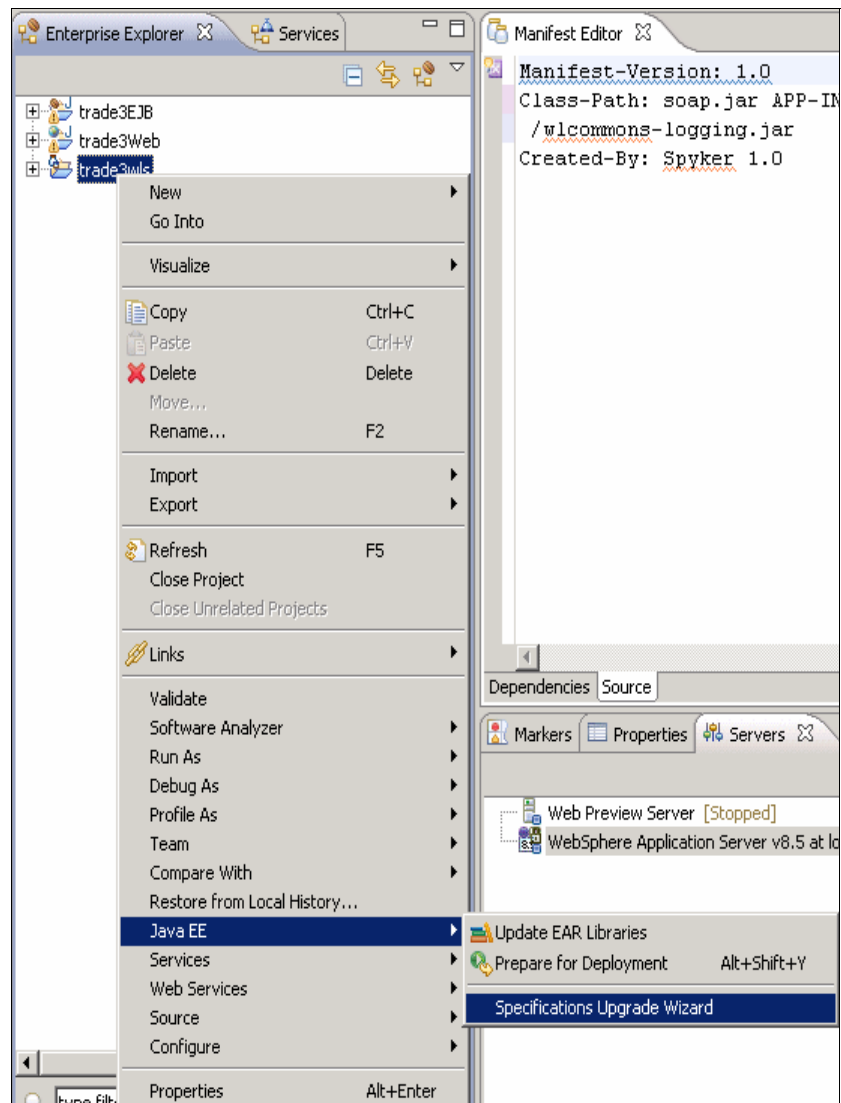


Figure 6-41 J2EE Migration wizard

The next window provides informational messages before you run the J2EE specification migration wizard. Click **Next**.

2. In the J2EE version drop-down menu, select **1.4** and click **Next** (Figure 6-42).

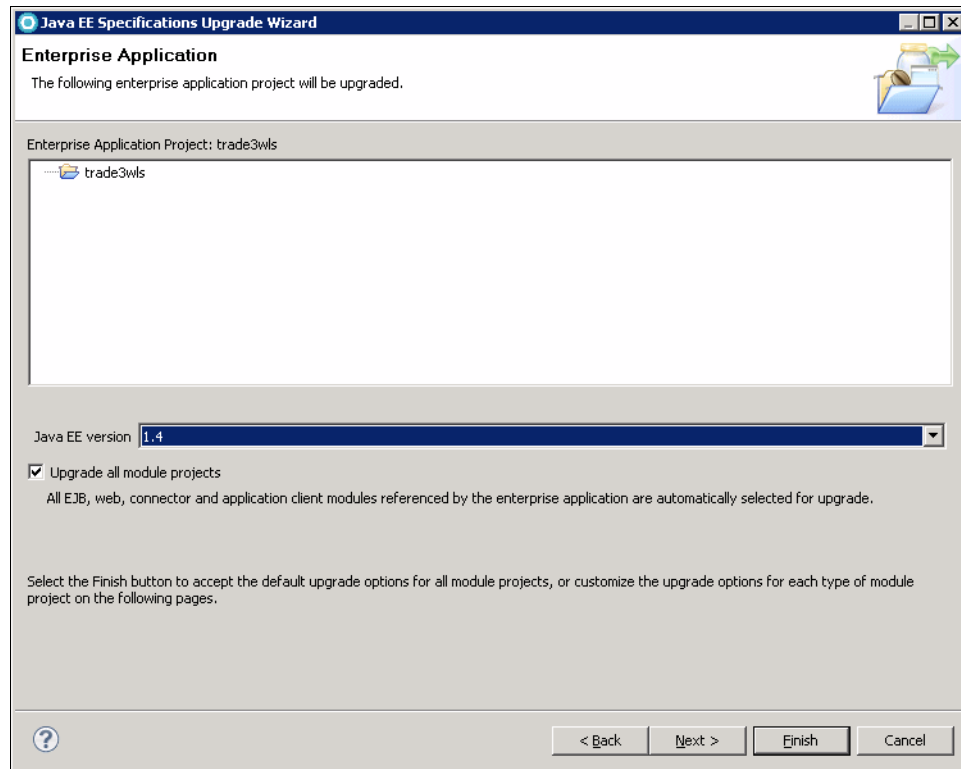


Figure 6-42 J2EE version selection

3. In the EJB Module Migration window, ensure that **trade3EJB** is selected and click **Next**.
4. In the Web Projects window, ensure that **trade3web** is selected and click **Finish**.

The Migration Complete dialog box (Figure 6-43) is displayed.

**Migration wizard changes:** You can click **Details** on the Migration Complete dialog box to see the exact changes that the migration wizard performed against the project.

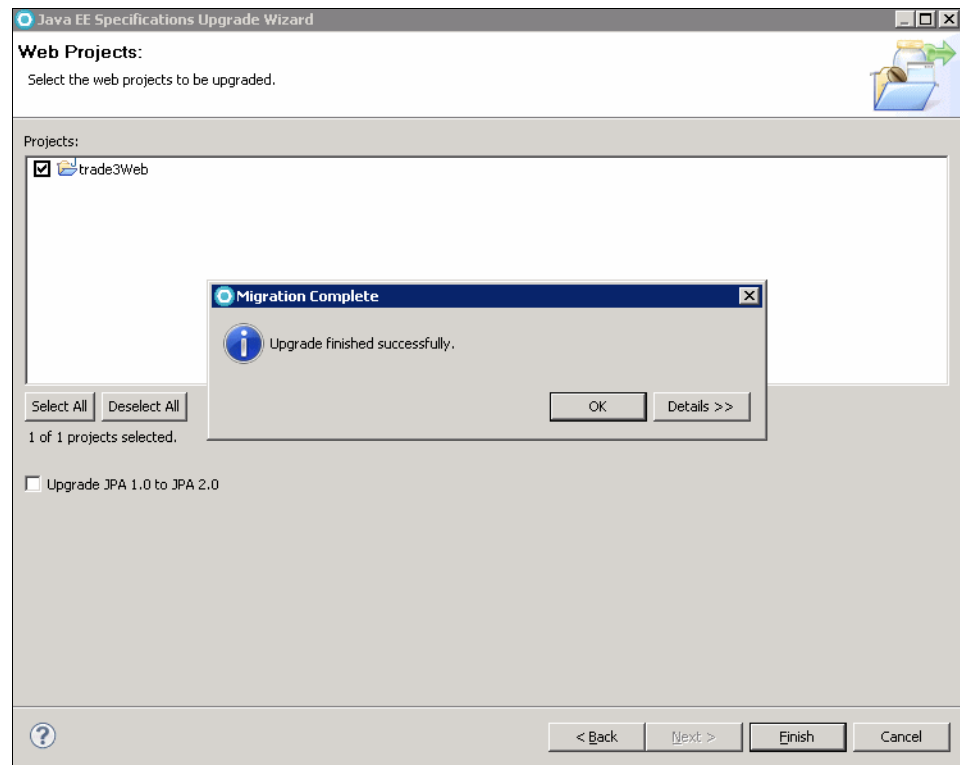


Figure 6-43 Migration Completion dialog box

5. Click **OK**.

## Configuring WebSphere Application Server V8.5

In this section, we describe how to configure WebSphere Application Server V8.5 to run the Trade application. You might want to create a WebSphere profile and a WebSphere test server that is based on that profile.

### Verifying that WebSphere Application Server is running

Verify that WebSphere Application Server is running on Rational Application Developer by completing the following steps:

1. To start the WebSphere Application Server, navigate to the **Servers** tab on Rational Application Developer, right-click your server, and click **Start**.
2. After a successful start of the server, you see the message `Server server1 open for e-business` on the Console, which indicates that the server is ready.

**Tip:** If you want to see the application server log files aside from the Console category in Rational Application Developer, go to the `C:\WebSphere_Install_Root\profiles\AppSrv01\logs\server1` path.

After the server start, you configure WebSphere Application Server V8.5. To do so, open a web browser and access the administrative console at the following URL:

`http://localhost:9060/ibm/console`

The major difference that is compared to Oracle WebLogic Server 10.3.6 is in how we configure JMS. This difference exists because WebSphere Application Server V8.5 uses Service Integration Bus, which is an ESB-like approach, as the back end for the default JMS messaging engine that is based on JMS 1.1 specifications.

**Service integration bus:** A service integration bus in WebSphere supports applications using message-based and service-oriented architectures (SOA). A bus is a group of one or more interconnected servers or server clusters that are added as members of the bus. Applications connect to a bus at one of the messaging engines that are associated with its bus members. For more information about SIB, see Chapter 13, “Messaging and service integration”, in *WebSphere Application Server V8.5 Concepts, Planning, and Design Guide*, SG24-8022.

## Configuring JMS

This section describes how to configure the components necessary for JMS messaging. In our example, we use the browser-based administrative GUI, but this task can also be scripted through the `wsadmin` command-line tool.

This task is broken into several parts. The following sections describe the overall process:

- ▶ “Creating a bus” on page 172
- ▶ “Creating a bus member” on page 173
- ▶ “Creating bus destinations for JMS queues and topics” on page 174
- ▶ “Creating a JMS connection factory” on page 174
- ▶ “Creating the JMS queue” on page 176
- ▶ “Creating a second JMS queue” on page 176
- ▶ “Creating a JMS topic” on page 177
- ▶ “Creating a JMS activation specification for the JMS queues” on page 177
- ▶ “Configuring Message Driven beans” on page 178

## Creating a bus

To create a bus, complete the following steps:

1. Open the administrative console and click **Service Integration** → **Buses** in the left navigation pane.
2. On the following window, create an ESB. Specify bus as the name. Click **OK**.
3. Click **Finish** on the Summary window (Figure 6-44).

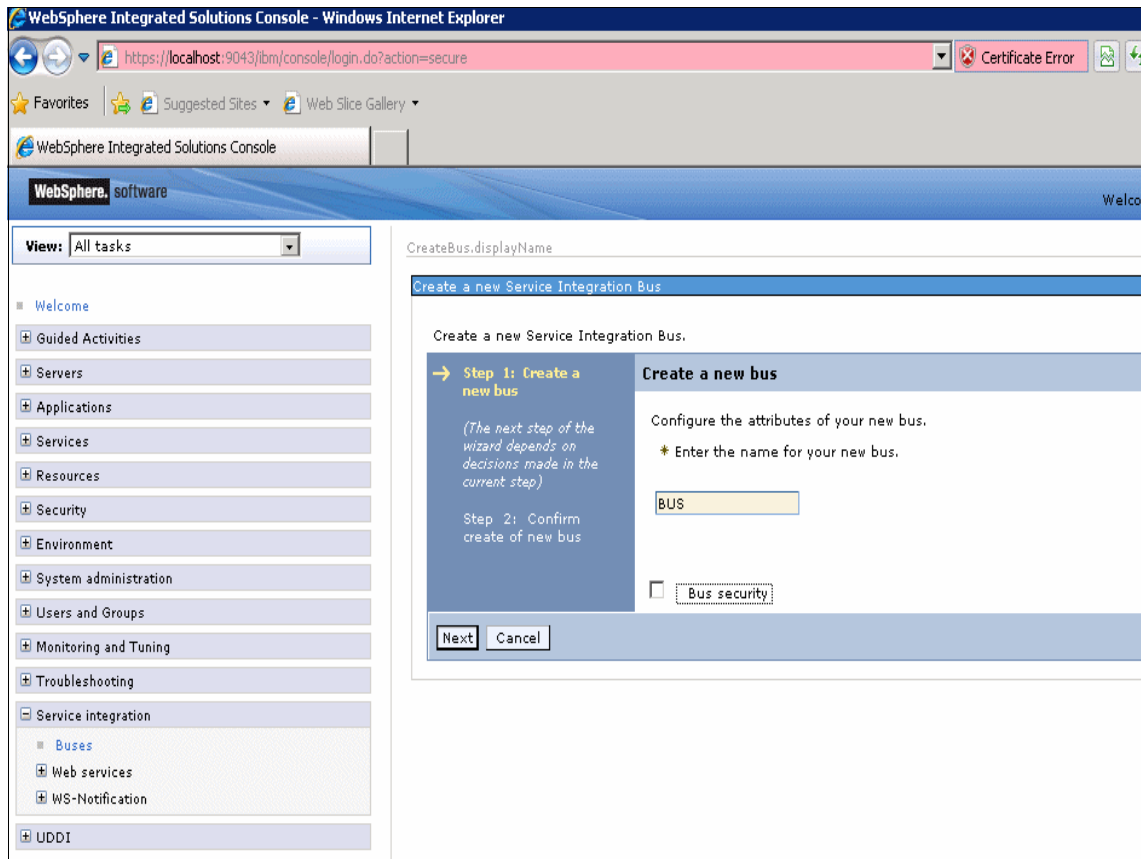


Figure 6-44 Creating a Service Integration Bus



4. Save the changes that are made to the configuration by clicking **Save** at the top of the window. You should see the newly created bus that is listed under buses (Figure 6-45).

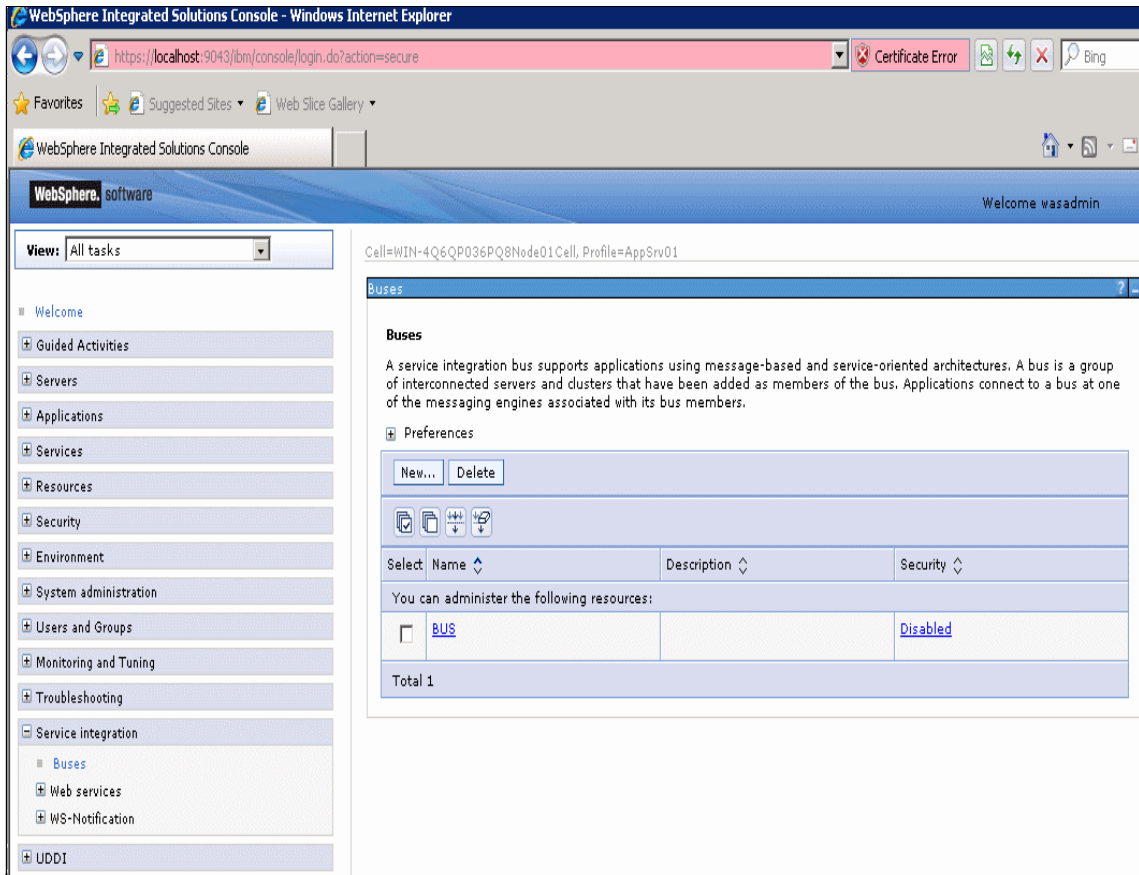


Figure 6-45 Service Integration Bus

### Creating a bus member

The bus must be deployed to a bus member. The members of a service integration bus are the application servers or clusters in which messaging engines for that bus can run.

To create a bus member, complete the following steps:

1. From the administrative console, click **Service Integration** → **Buses** in the navigation pane on the left. Select the bus that is created in “Creating a bus” on page 172.
2. Click **Bus members** in the Topology section.
3. Click **Add** and select the server to which the bus is deployed. Accept all defaults on the remaining windows and click **Finish**. Save the changes.

**Tip:** Manual configuration is error-prone and difficult. Automating these tasks saves time. Download the WebSphere version of the Trade application together with Jacl scripts for the `wsadmin` scripting facility. Those scripts create all the necessary resources and perform the application deployment in a clustered environment. Download it from the following web page:

<http://www.ibm.com/software/webservers/appserv/was/performance.html>

For more information about the administrative scripting, see the WebSphere Information Center at:

<http://publib.boulder.ibm.com/infocenter/iisinfsv/v8r5/index.jsp>

### ***Creating bus destinations for JMS queues and topics***

The destination is the target for all messages sent to the enterprise service bus. To create the bus destinations for the JMS queue and topic, complete the following steps:

1. From the administration console, navigate to **Service Integration** → **Buses**.
2. Select the bus that is created in “Creating a bus” on page 172.
3. Click **Destinations** in the Destination resources section.
4. Click **New**, and then select **Queue** as the destination type. Click **Next**.
5. Specify **TradeBrokerJSD** as the identifier. Click **Next**.
6. Assign the new queue to a bus member. Because there is only one bus member, click **Next** and then **Finish**.
7. Create another queue destination by repeating steps 1 through 6. Use **TradeBrokerDirect** as the identifier.
8. Create a topic space by repeating steps 1 through 6. Specify **Trade.Topic.Space** as the identifier.
9. Save the changes.

### ***Creating a JMS connection factory***

Before you configure the JMS queues, you must configure a JMS connection factory, which is used by the Trade MDBs to create a connection to the JMS provider. Complete the following steps:

1. From the administration console, navigate to **Resources** → **JMS** → **JMS providers** from the left navigation pane.

- In the right pane, select **Node=<node\_name>,Server=server1** in the scope drop-down menu (Figure 6-46).



Figure 6-46 WebSphere scope selection

- Click **Default messaging provider**.
- Click **Queue connection factories** in the Additional Properties section.
- Click **New**, and create a queue connection factory using the information that is shown in Table 6-13.

Table 6-13 JMS queue connection factory configuration

Parameter	Values
Name	TradeBrokerQCF
JNDI name	jms/TradeBrokerQCF
Bus name	BUS

- Click **OK**, and then save the changes that you made.
- Navigate to **Resources** → **JMS** → **JMS Providers**.
- Click **Default messaging provider**.
- Click **Topic connection factories** in the Additional Properties section.

10. Click **New**, and create a JMS topic connection factory using the information in Table 6-14.

Table 6-14 JMS topic connection factory configuration

Parameter	Values
Name	TradeStreamerTCF
JNDI name	jms/TradeStreamerTCF
Bus name	BUS

11. Save the changes.

### Creating the JMS queue

Trade contains two message driven beans (MDBs). One MDB requires that you configure a new JMS queue. To do so, complete the following steps:

1. In the right pane, click **Default messaging provider**.
2. Click **Queues** in the Additional Properties section.
3. Click **New**, and create a JMS queue using the information in Table 6-15.

Table 6-15 JMS queue configuration

Parameter	Value
Name	TradeBrokerQueue
JNDI name	jms/TradeBrokerQueue
Bus name	BUS
Queue name	TradeBrokerJSD

4. Click **OK** and save your changes.

### Creating a second JMS queue

The second MDB in Trade requires that you configure a new JMS topic. To do so, complete the following steps:

1. Repeat steps 1 - 3 of the procedure in "Creating the JMS queue" on page 176 to create a second JMS queue using the information that is shown in Table 6-16.

Table 6-16 JMS queue configuration

Parameter	Value
Name	TradeBrokerDirectQueue
JNDI name	jms/TradeBrokerDirectQueue
Bus name	BUS
Queue name	TradeBrokerDirect

2. Click **OK** and save the changes.

### Creating a JMS topic

Trade contains two MDBs. The second MDB is for the JMS topic. To configure the JMS topic, complete the following steps:

1. From the administration console, navigate to **Resources** → **JMS** → **JMS providers** from the navigation pane.
2. In the right pane, select **Node=<node\_name>**, **Server=server1** in the Scope drop-down menu.
3. Click **Default messaging provider**.
4. Click **Topics** in the Additional Properties section.
5. Click **New**, and create a JMS topic using the information that is shown in Table 6-17.

Table 6-17 JMS topic configuration

Parameter	Value
Name	TradeStreamerTopic
JNDI name	jms/TradeStreamerTopic
Bus name	BUS
Topic space	Trade.Topic.Space

6. Click **OK** and save the changes.

### Creating a JMS activation specification for the JMS queues

The last thing to configure in WebSphere Application Server V8.5 related to JMS is the JMS activation specification, which binds the MDBs to the default JMS messaging provider. The JNDI name of the JMS activation specification that you create is used in the MDB's deployment descriptor to bind the MDB to the messaging queue.

Complete the following steps:

1. In the right pane, click **Default messaging provider**.
2. Click **Activation specification** in the Additional Properties section.
3. Click **New**, and create a JMS activation specification using the information that is shown in Table 6-18.

Table 6-18 JMS activation specification configuration

Parameter	Values
Name	TradeBrokerMDB
JNDI name	eis/TradeBrokerMDB
Destination type	Queue
Destination JNDI name	jms/TradeBrokerQueue
Bus name	BUS

4. Create a second JMS activation specification with the information shown in Table 6-19.

Table 6-19 JMS activation specification configuration

Parameter	Value
Name	TradeStreamerMDB
JNDI name	eis/TradeStreamerMDB
Destination type	Topic
Destination JNDI name	jms/TradeStreamerTopic
Bus	BUS

5. Click **OK** and save your changes.

You have all the necessary components for JMS messaging configured. The next step is to configure the MDBs to use the JMS activation specification and the queues.

### **Configuring Message Driven beans**

Use Rational Application Developer V8.5 and its Deployment Descriptor Editor to edit the standard `ejb-jar.xml` deployment descriptor and create the WebSphere Application Server V8.5 specific deployment descriptors. To do so, complete the following steps:

1. In Rational Application Developer, open the Deployment Descriptor Editor by navigating to EJB Projects / trade3EJB/ejbModule/META-INF and double-clicking the `ejb-jar.xml` file.
2. Click the **Bean** tab.
3. Select **TradeBrokerMDB** from the list of EJBs. In the WebSphere Bindings section, click **JCA Adapter** and specify the values that are shown in Table 6-20.

Table 6-20 OrderProcessor MDB's WebSphere binding configuration

Parameter	Value
ActivationSpec JNDI name	eis/tradeBrokerMDB
Destination JNDI name	jms/TradeBrokerQueue

You have now configured the MDB to use the JCA Activation Specification and Destination that is configured in “Creating a JMS activation specification for the JMS queues” on page 177. See Figure 6-47.

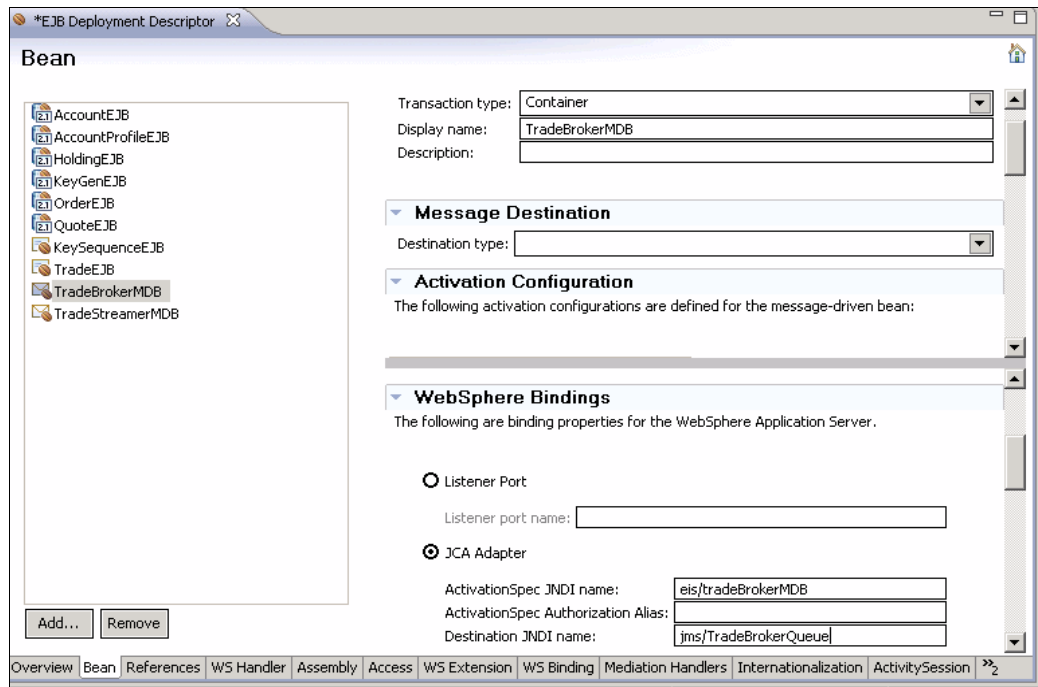


Figure 6-47 MDB activation specification configuration

**JCA Adapter option:** If you cannot see the JCA Adapter option under WebSphere Bindings, verify that you changed the J2EE specification to 1.4 with the Specifications Upgrade wizard that is described in “Migrating the Trade Application J2EE level” on page 168. To check that the application J2EE level is 1.4, right-click the EAR file in the Enterprise Explorer category, click **Properties**, and view the EAR version under Project Facets, as shown in Figure 6-48.

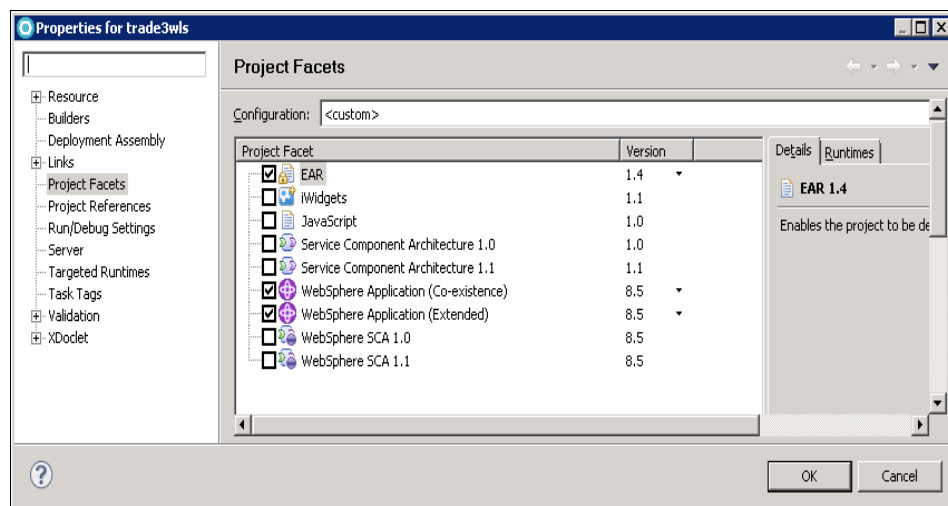


Figure 6-48 Project J2EE specification

4. Select **TradeSteamerMDB** from the list of EJBs. In the WebSphere Bindings section, select **JCA Adapter** and specify the values that are shown in Table 6-21.

Table 6-21 Mailer MDBs WebSphere binding configuration

Parameter	Value
ActivationSpec JNDI name	eis/tradeStreamerMDB
Destination JNDI name	jms/TradeStreamerTopic

You have now completed the configuration of the MDBs and JMS messaging.

## Creating a data source

Trade uses a JDBC data source to connect to the database. To set up the data source, you must configure a JDBC provider, a J2C alias for storing the database authentication information, and the data source itself.

**JCA data entries:** J2EE Connector Architecture (JCA) data entries are used by resource adapters and JDBC data sources. A JCA data entry contains authentication data, which includes an alias (an identifier), user ID (a user identity), password, and description.

The configuration is done from the administrative console by completing the following steps:

1. From the administration console, click **Environment** → **WebSphere Variables**.
2. In the right frame, select **Node=<node\_name>** in the Scope drop-down menu.
3. Click the **DB2UNIVERSAL\_JDBC\_DRIVER\_PATH** environment variable.
4. Specify the correct path to the DB2 JDBC driver and click **OK**. In our migration example, the path to the JDBC driver is C:\DB2\_10\IBM\SQLLIB\java.
5. Save the changes.
6. Click the **UNIVERSAL\_JDBC\_DRIVER\_PATH** environment variable and specify the value `${WAS_INSTALL_ROOT}/universalDriver/lib`. This value points to the `db2jcc_license_cu.jar`, which is the DB2 license file.
7. Click **OK** and save the changes.
8. Create the JDBC provider, data source, and J2C authentication alias. From the administration console, navigate to **Resources** → **JDBC** → **DBC providers**.
9. In the Scope drop-down menu, select **Node=<Node-name>**.
10. Click **New** to create a JDBC provider using the information that is shown in Table 6-22.

Table 6-22 JDBC provider configuration

Parameter	Value
Database type	DB2
Provider type	DB2 Universal JDBC Driver Provider
Implementation type	XA data source

11. We already defined the class path values for the database using WebSphere Variables, so the values are already populated. Click **Next**.
12. The Summary window is displayed. Click **Finish** and save the changes.



With the JDBC provider defined and the DB2 JDBC driver path that is specified, the next step is to create the actual data source and authentication alias by completing the following steps:

1. From the administration console, navigate to **Resources** → **JDBC** → **JDBC providers**, and select the newly created JDBC provider by clicking its name, which is **DB2 Universal JDBC Driver Provider (XA)**.

**JDBC Provider:** We select JDBC Provider implementation type as our XA data source instead of the Connection pool data source because the Trade application includes the operations that are wrapped in a global 2-phase transaction and commit.

2. Click **Data sources** in the Additional properties section.
3. Click **New**, and create a data source using the information that is shown in Table 6-23.

*Table 6-23 JDBC data source configuration*

Parameter	Value
Driver type	4
Server name	localhost
Database name	trade3db

4. Click **Next**.
5. On the Setup security alias window, leave all the defaults and click **Next**.
6. The Summary window is displayed. Click **Finish** and save the changes.
7. Select the newly created data source by clicking its name, which is **TradeDataSource**.
8. Click **JAAS – J2C authentication data** in the Related Items section.
9. Click **New**, and create an alias using the information that is shown in Table 6-24.

*Table 6-24 J2C authentication alias configuration*

Parameter	Value
Alias	TradeDataSourceAuthData
UserId	db2admin
Password	db2admin

10. Click **Apply**.
11. Click **OK** and go back to the data source. From the Component-managed authentication alias drop-down menu, select the J2C alias.
12. Click **OK** and save your configuration.
13. Select the data source you created from the list and click **Test connection**.

## Migrating EJBs

This section describes how to configure the Trade Session beans and Container Managed beans to run on WebSphere Application Server V8.5. In our migration example, we use the Deployment Descriptor Editor in Rational Application Developer V8.5 and the migration tool to generate the WebSphere Application Server V8.5 specific deployment descriptors.

### Specifying EJB JNDI names

The first step is to assign JNDI names to the Trade EJBs, because the migration tool did not do it for all of the EJBs.

**Important:** The migration tool migrates the JNDI names that are in the WebLogic EJB deployment descriptor to the corresponding WebSphere files. However, most of the EJB's JNDI names for Trade application are not specified in the EJB deployment descriptor, so you must perform those steps manually.

1. From the Rational workspace, open the `ejb-jar.xml` file with the Deployment Descriptor Editor and click the **Bean** tab.
2. Click **AccountEJB**. In the WebSphere Bindings section, enter the information that is shown in Table 6-25.

Table 6-25 WebSphere bindings for the Account EJB

Parameter	Value
AccountEJB	ejb/AccountEJB

3. Repeat step 2 for each EJB in the list, except for the MDBs. Specify a unique JNDI name for each EJB in the format `ejb/<EJBName>` using the values that are shown in Table 6-26.

Table 6-26 Trade EJB JNDI names

EJB	JNDI name values
AccountProfileEJB	ejb/AccountProfileEJB
HoldingEJB	ejb/HoldingEJB
KeyGenEJB	ejb/KeyGenEJB
OrderEJB	ejb/OrderEJB
QuoteEJB	ejb/QuoteEJB
KeySequenceEJB	ejb/KeySequenceEJB
TradeEJB	ejb/TradeEJB

4. Save the changes.

### Specifying the EJB references

This process specifies the EJB reference names under the WebSphere Bindings for the EJBs. These names are the JNDI names from the `weblogic-ejb-jar.xml` file that were not migrated from the tool.

Complete the following steps:

1. In Rational Application Developer, open the EJB deployment descriptor editor by navigating to the `/trade3EJB/ejbModule/META-INF` folder. Double-click the `ejb-jar.xml` file.
2. Click the **References** tab.
3. Expand AccountEJB by clicking the + sign next to it.
4. Select the **ejb/AccountProfile**.
5. Under the WebSphere Bindings section, enter `ejb/AccountProfileEJB` as the JNDI reference name.

- Repeat steps 1 on page 182 through 5 on page 182 using the information in Table 6-27.

Table 6-27 EJB reference names

EJB to expand	Reference	JNDI name
AccountEJB	ejb/AccountProfile	ejb/AccountProfileEJB
KeySequenceEJB	ejb/KeyGen	ejb/KeyGenEJB
TradeEJB	ejb/Quote	ejb/QuoteEJB
TradeEJB	ejb/Account	ejb/AccountEJB
TradeEJB	ejb/Holding	ejb/HoldingEJB
TradeEJB	ejb/Order	ejb/OrderEJB
TradeEJB	ejb/KeySequence	ejb/KeySequenceEJB
TradeEJB	ejb/AccountProfile	ejb/AccountProfileEJB
TradeEJB	jdbc/TradeDataSource	jdbc/TradeDataSource
TradeEJB	jms/QueueConnectionFactory	jms/TradeBrokerQCF
TradeEJB	jms/TopicConnectionFactory	jms/TradeStreamerTCF
TradeEJB	jms/TradeBrokerQueue	jms/TradeBrokerQueue
TradeEJB	jms/TradeStreamerTopic	jms/TradeStreamerTopic
TradeBrokerMDB	ejb/Trade	ejb/TradeEJB

- Save changes to the deployment descriptor by pressing Ctrl+s. Or, from the Rational Application Developer Window menu, click **File** → **Save**. Do not close the EJB deployment descriptor editor.

### Configuring the default data source for CMPs

In this procedure, specify a default data source that is used by all CMP beans by completing the following steps:

- Select the **Overview** tab in the Deployment Descriptor Editor.
- In the WebSphere Bindings section, under the JNDI - CMP Connection Factory Binding heading, enter the values that are shown in Table 6-28.

Table 6-28 Default data source configuration

Parameter	Value
JNDI name	jdbc/TradeDataSource

### Specifying the web application references

Complete the following steps to specify web application references. The tool adds some of the JNDI names, such as TradeEJB, QuoteEJB, and jdbc. In our example, we add the other names manually. Also, the tool creates the connection factory names with its default names, and we change them to the ones shown in Table 6-29 on page 184.

- In Rational Application Developer, open the Web Deployment Descriptor Editor by navigating to the /trade3Web/webContent/WEB-INF folder. Double-click the **web.xml** file.
- Click the **References** tab.
- Select the **jdbc/TradeDataSource** reference.
- Under the WebSphere Bindings section, verify that JNDI name is jdbc/TradeDataSource.

- Repeat steps 1 on page 183 through 4 on page 183 using the information in Table 6-29 for the reference and JNDI names.

Table 6-29 Trade web application reference JNDI names

Reference	JNDI name
jdbc/TradeDataSource	jdbc/TradeDataSource
jms/QueueConnectionFactory	jms/TradeBrokerQCF
jms/TopicConnectionFactory	jms/TradeStreamerTCF
ejb/Trade	ejb/TradeEJB
ejb/Quote	ejb/QuoteEJB
jms/TradeBrokerQueue	jms/TradeBrokerQueue
jms/TradeStreamerTopic	jms/TradeStreamerTopic
ejb/LocalQuote	ejb/QuoteEJB
ejb/LocalAccountHome	ejb/AccountEJB
jms/TradeBrokerDirectQueue	jms/TradeBrokerDirectQueue

### Restarting WebSphere Application Server

In this procedure, you restart WebSphere Application Server by completing the following steps:

- In a command prompt, navigate to C:\<WAS\_Install\_Root>\profiles\AppSrv1\bin.
- Run **stopServer server1** to stop the server.
- Run **startServer server1** to start the server.

After a successful start of the server, you should see the message `Server server1 open for e-business`, which indicates that the server is ready.

## Deploying the application

After the resources for the application are configured, the next step is to deploy the application to the test environment on Rational Application Developer. To do so, complete the following steps:

1. In the Rational Application Developer Servers view, right-click **WebSphere Application Server v8.5 at localhost** and select **Add and Remove Projects**. See Figure 6-49. The Add and Remove Projects dialog box is displayed.

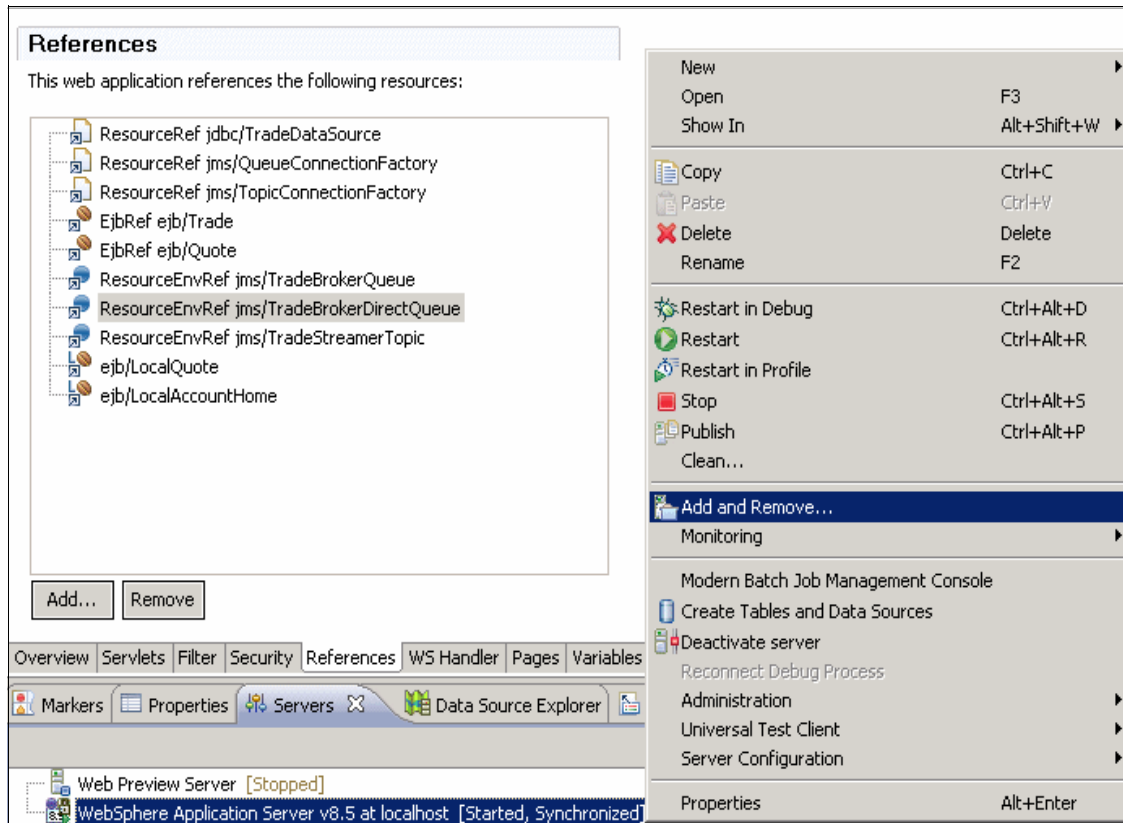


Figure 6-49 Add and Remove Projects

2. Under Available projects, select **trade3wls** and click **Add**. The trade3wls project is found under the Configured projects.
3. Click **Finish**.

It takes a couple of minutes for the application to deploy to the server. You can expand the server and verify that the application status is Started.

## Configuring basic authentication on WebSphere

Here are the settings in web.xml of the trade web application that we added to demonstrate what you should do about configuring application security on WebSphere. There are differences with WebLogic and WebSphere in terms of mapping users/groups to role name. See Example 6-5.

*Example 6-5 The basic authentication settings in web.xml*

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>secure resources</web-resource-name>
    <url-pattern>*/</url-pattern>
  </web-resource-collection>
```

```

    <auth-constraint>
      <role-name>threadUser</role-name>
    </auth-constraint>
  </security-constraint>
</login-config>
  <auth-method>BASIC</auth-method>
</login-config>
<security-role>
  <role-name>threadUser</role-name>
</security-role>

```

---

To map users/groups to security role, which is threadUser in our example, complete the steps in the following sections.

### ***Configuring security and creating user and group on WebSphere***

Complete the following steps:

1. Open WebSphere Admin Console, click **Global Security**, and check **Enable application security** to enable application security on WebSphere (Figure 6-50).

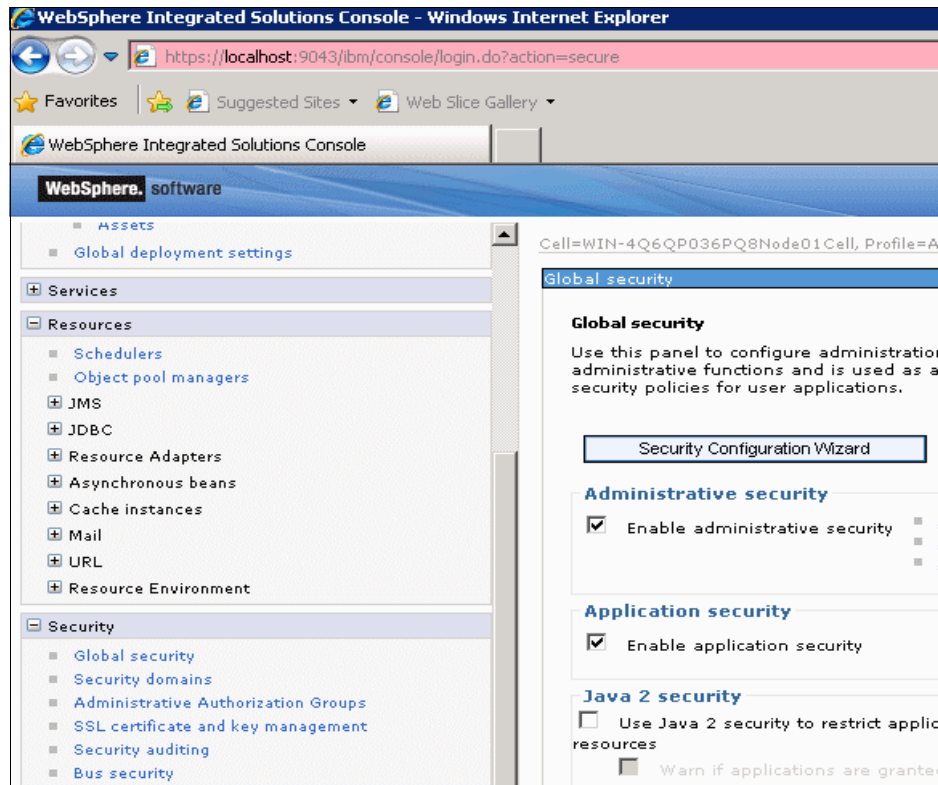


Figure 6-50 Enable application security

2. Click **OK**.

3. Click **Manage Users** under **Users and Groups** and click **Create** to add a user (Figure 6-51).

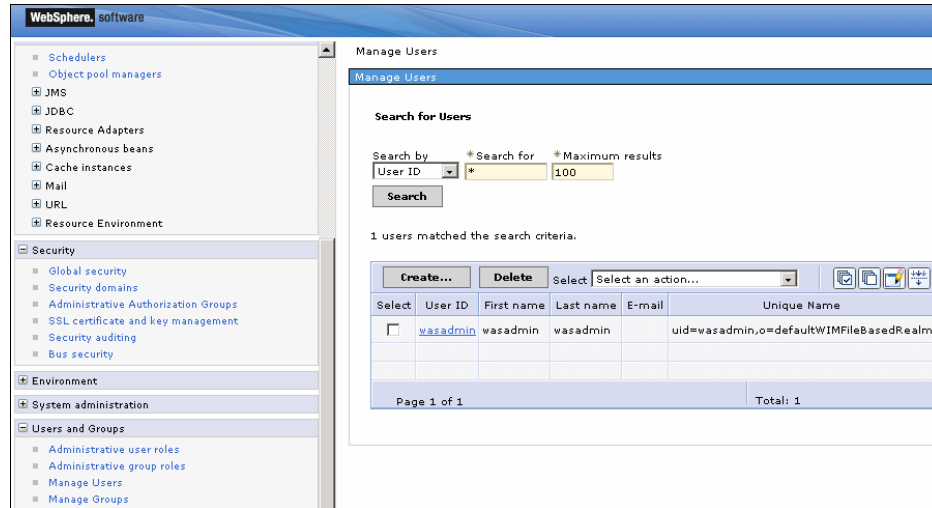


Figure 6-51 Creating a user

4. Complete the fields as shown in Figure 6-52. We used testuser/testuser as the user name and password in our example.

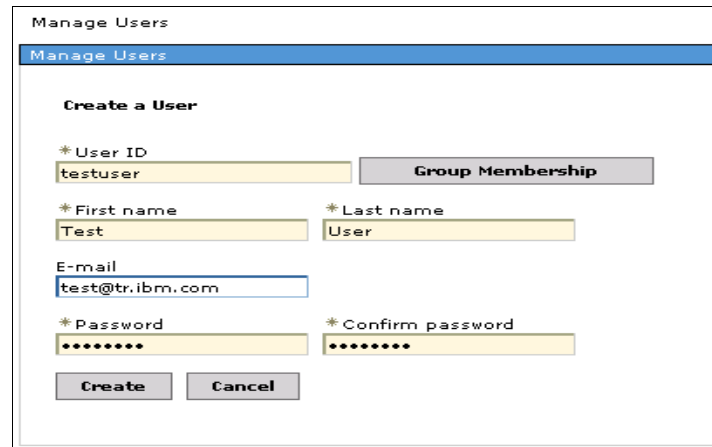


Figure 6-52 Manage Users

5. Click **Create** and **Close**.

- Click **Manage Groups** under Users and Groups and click **Create** to add a group (Figure 6-53).

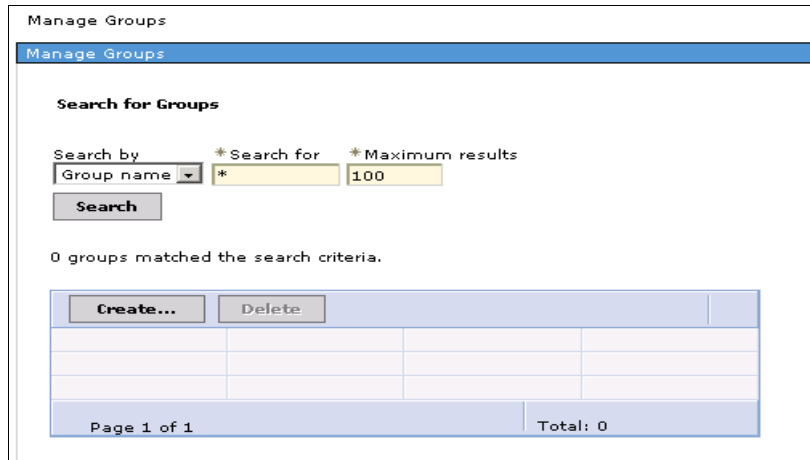


Figure 6-53 Creating a group

- Enter threadUserGroup as the Group name.
- Click **Create** and **Close**.
- Click **threadUserGroup** and click **Members** to add the user we created in the Group (Figure 6-54).

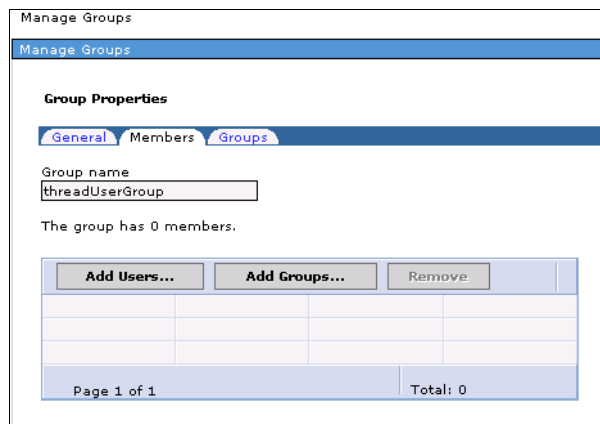


Figure 6-54 Group properties



10. Click **Add Users** and click **Search**. Select **testuser** and click **Add** (Figure 6-55).

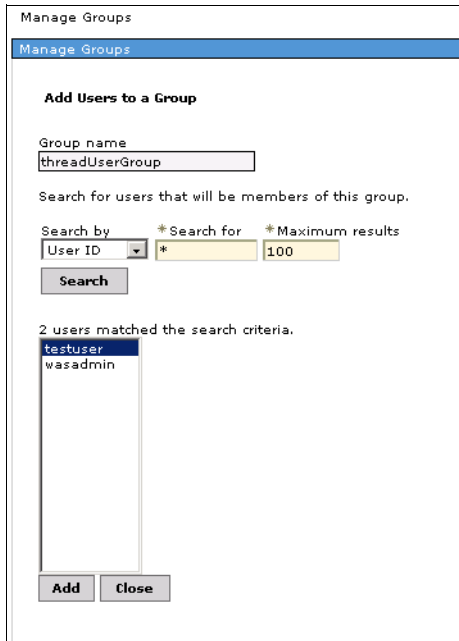


Figure 6-55 Adding member into the group

11. You see the “The users were added to the group successfully” message. Click **Close**.

12. You see the window that is shown in Figure 6-56.

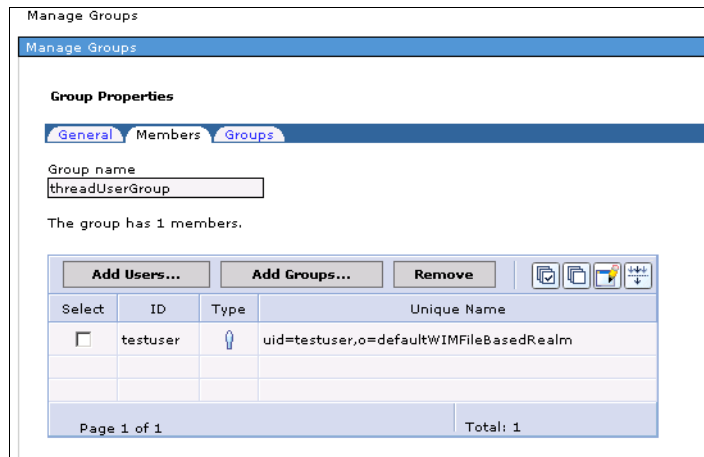


Figure 6-56 Testuser properties

### **Mapping a security user and group to role name in Rational Application Developer**

To map a security user and group to role name in Rational Application Developer, complete the following steps:

1. Right-click the Trade EAR project and click **Project Facets**.

2. Ensure that **WebSphere Application (Co-existence)** and **WebSphere Application (Extended)** are selected (Figure 6-57).

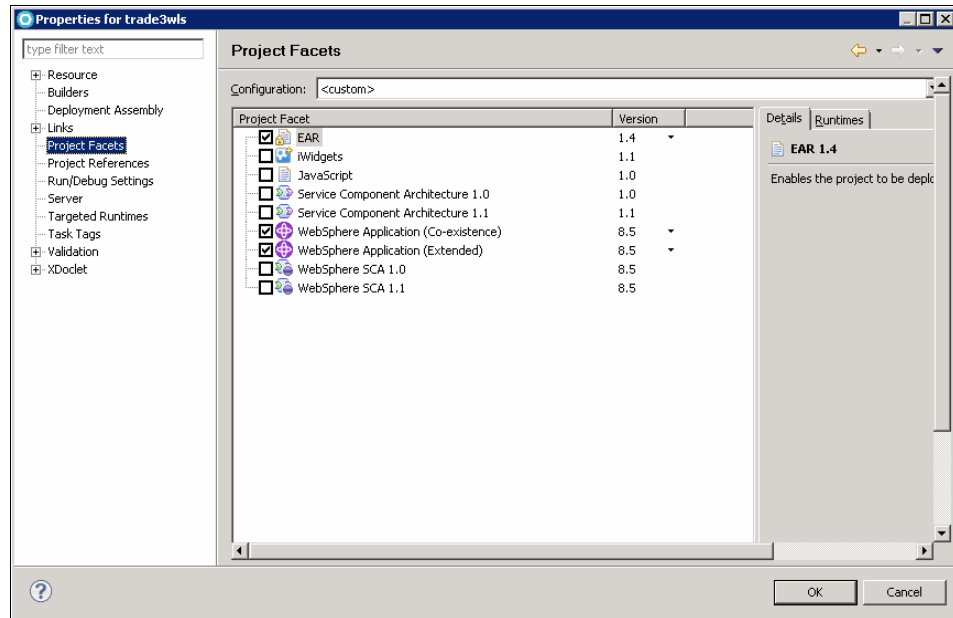


Figure 6-57 Project facets of the trade3wls EAR file

3. Click **Deployment Descriptor** of the EAR file and click the **Security** tab. Ensure that you can see the threadUser role name. If you cannot see the role name, click **Gather** to let Rational Application Developer get the role name from the trade web application web.xml file (Figure 6-58).

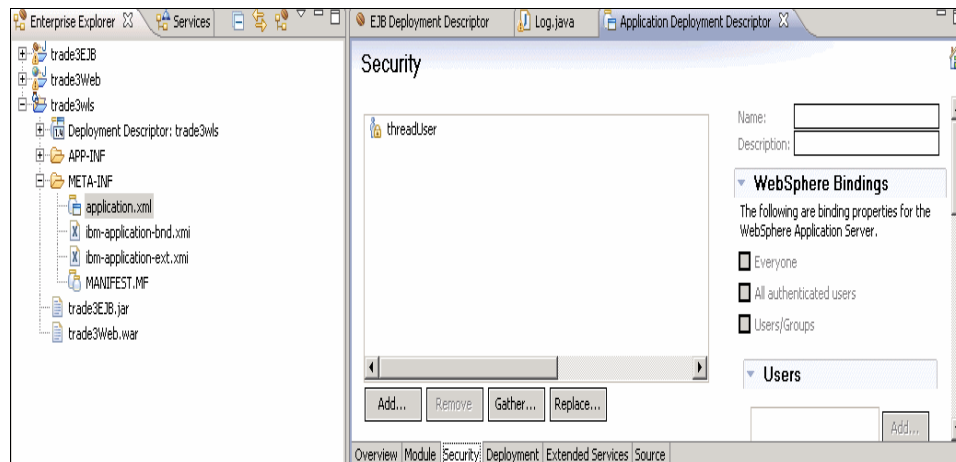


Figure 6-58 Checking the security role name

4. Click the **threadUser** role name, click **Users/Groups** under WebSphere Bindings and Add threadUserGroup, which you defined in “Configuring security and creating user and group on WebSphere” on page 186. See Figure 6-59.

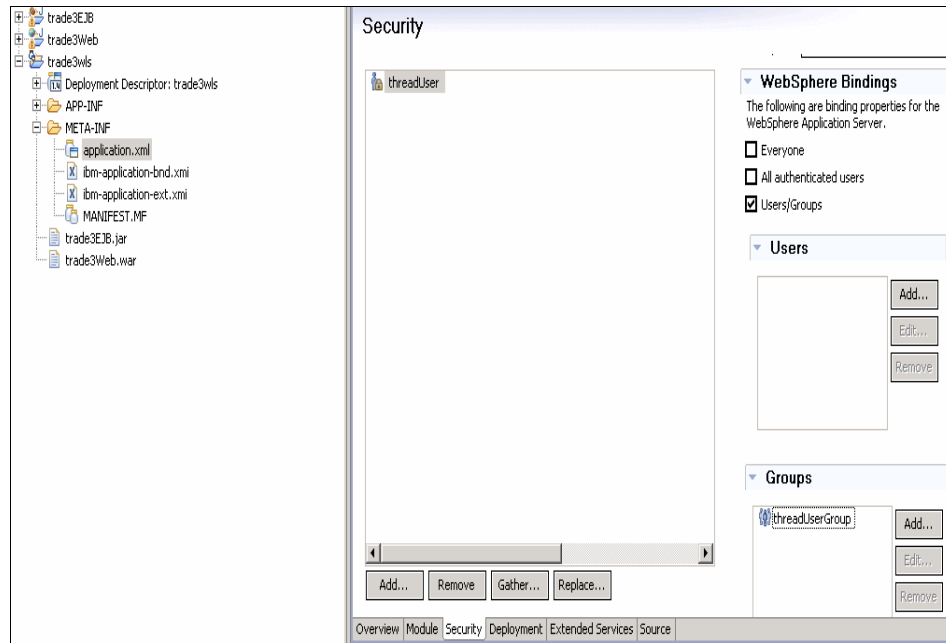


Figure 6-59 Mapping group in Rational Application Developer

5. **Save** the file.

**Importing the security user role:** We create the security user role mapping by using Rational Application Developer instead of WebSphere admin console, as we import the project into Rational Application Developer from the archive file. If you prefer importing the project into Rational Application Developer as an EAR file, you can map the security roles in WebSphere admin console by clicking **Application** → **Security role to user/group mapping** → **Map Groups** and specifying the threadUsergroup as the security group.

## Testing the application

The application can be accessed at the following URL:

<http://localhost:9080/trade3Web>

As we enabled basic authentication in our sample, we are prompted for the user name and password (Figure 6-60).

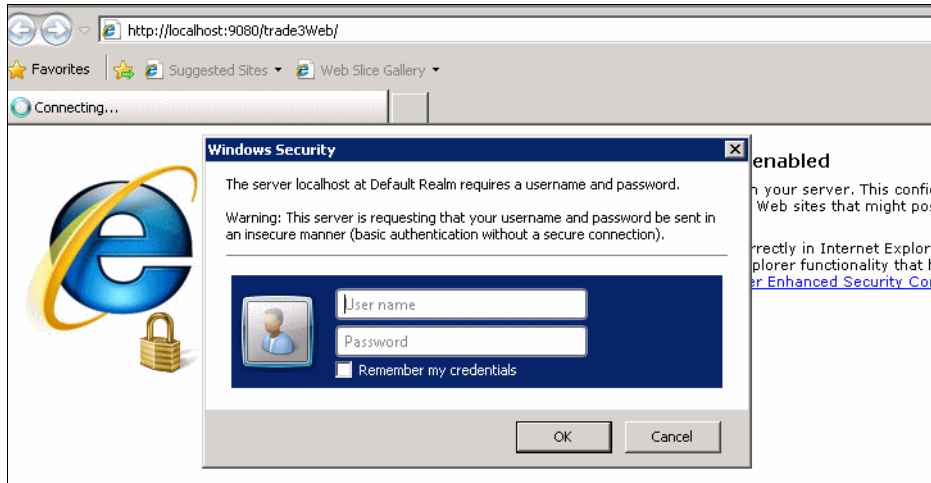


Figure 6-60 Authentication window

Enter the user name and password (testuser in both cases) to log in to the application (Figure 6-61).

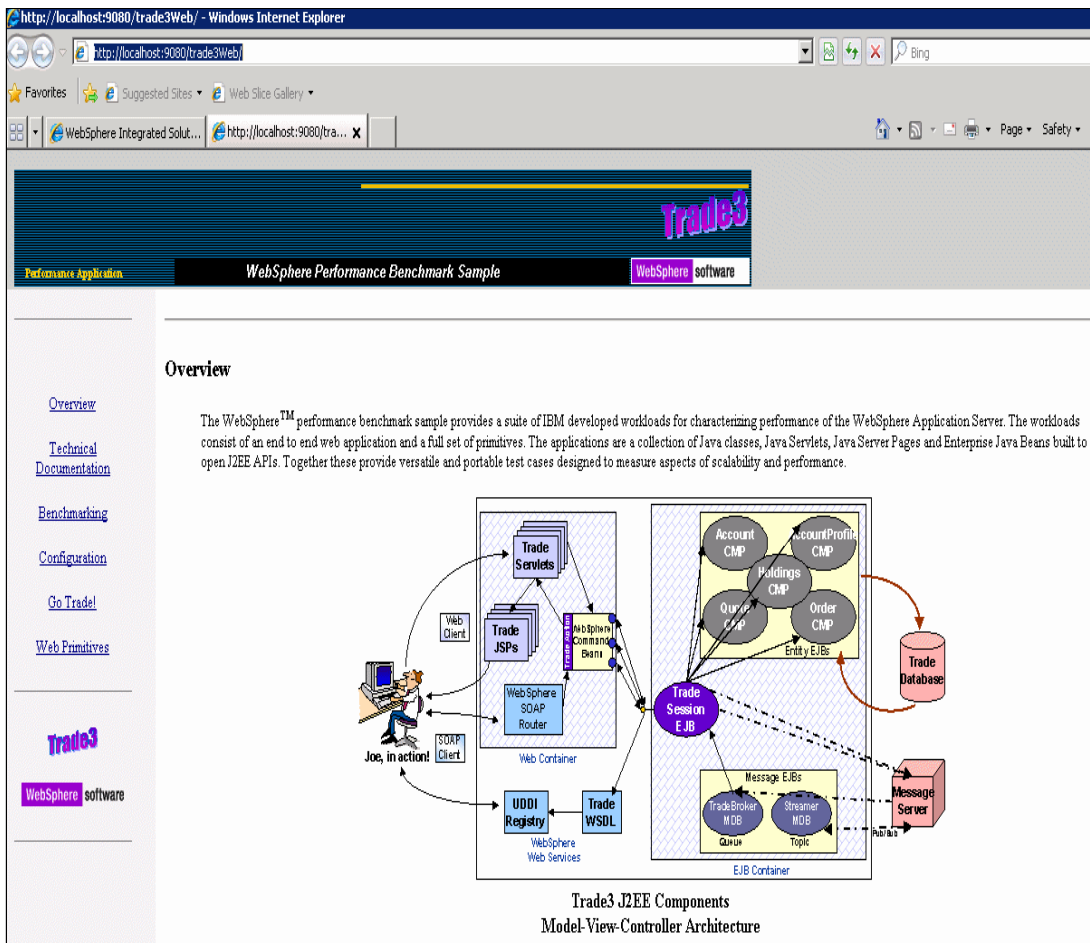


Figure 6-61 Trade Application welcome page

Browse through the site to check that the application is running correctly.

To test the application, complete the following steps:

1. Click the **Configuration** option in the left navigation pane (Figure 6-62).

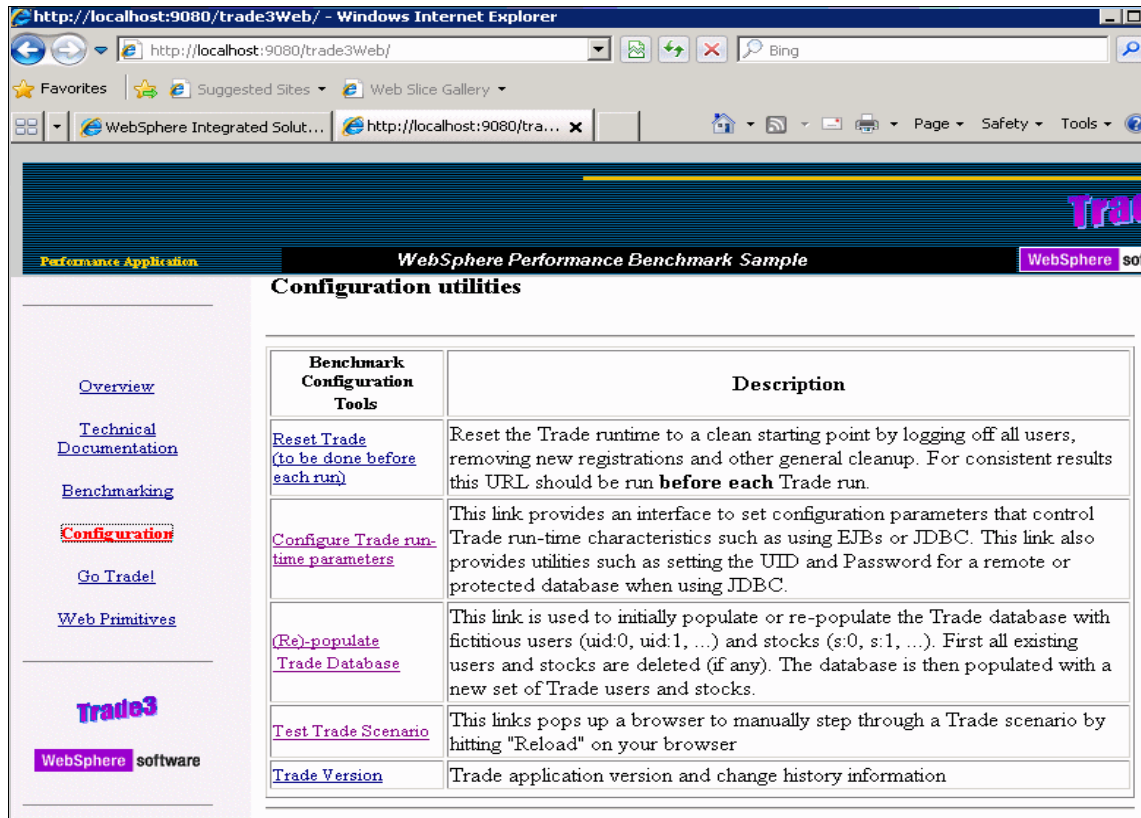


Figure 6-62 Trade Application Configuration options

2. Click the **(Re)-Populate Trade Database** option under the Benchmark Configuration Tools. This action populates the Trade database with fictitious users and stocks. Any existing users are deleted, and a new set of users and stocks are created. (This action might take some time to complete.)
3. Scroll down to the Trade Configuration section **Order-Processing Mode**. The three choices are as follows:
  - Synchronous (the default selection)
  - Asynchronous\_1-Phase
  - Asynchronous\_2-Phase

The mode can be changed for subsequent runs of the Trade application by selecting another mode and clicking **Update Config**.

4. Close this window and go back to the original Trade Application window.
5. While still in the Configuration utilities, select **Reset Trade**. Note the Order Processing mode in the table.

For the next set of tests, access the following URL:

`http://localhost:9080/trade3Web/scenario`

This URL is handled by TradeScenarioServlet. The servlet emulates a population of users by executing an action for a randomly chosen user upon each access to the URL.

## 6.5.4 Summary

We completed the migration of the Trade application with minimum changes to the Java code. This application was originally developed for WebSphere, then ported to WebLogic, and we ported it back to WebSphere. Trade is a fairly complex application and uses many vendor-specific deployment descriptors with the same code base.

We redeployed the compiled EAR file without having to change much of the Java code. We were able to demonstrate the following concepts:

- ▶ Usage of the WebSphere Application Server Migration Tool V3.5 for Rational Application Developer V8.5 for automatic migration of CMP EJB mappings and vendor-specific deployment descriptors and package.
- ▶ Configuration of JDBC provider and data source, Service Integration Bus, JMS queues, and topics and application deployment tasks with WebSphere Application Server V8.5.
- ▶ Basic development activities using Rational Application Developer V8.5.
- ▶ Enabling and configuring basic authentication and performing mapping security users/groups to the role name defined in application on WebSphere.

## 6.6 MVC WebLogic Application migration

This section describes how to migrate an MVC J2EE application that runs on WebLogic to WebSphere Application Server V8.5. It describes the steps to import the application into Rational Application Developer and how to run a Migration Toolkit. You find the common issues and the solutions to problems you may experience in this section as well.

### 6.6.1 Migration approach

Migrate the MVC J2EE application that is built for WebLogic to WebSphere Application Server V8.5. First, configure the WebSphere Migration Toolkit and create a connection to a DB2 database. Then, start Rational Application Developer V8.5 and import the MVC, including the CMP EJB 2.0 application, analyze and fix the errors from WebLogic generated code, compile JSPs using the batch JSP compiler, and migrate the deployment descriptors using the WebSphere Migration Toolkit. Finally, configure the required JDBC resources and deploy the MVC and CMP applications to WebSphere Application Server V8.5.

The migration of the MVC application is done by completing the following steps:

1. Import the J2EE compliant EAR file into Rational Application Developer V8.5 to take advantage of its advanced deployment descriptor editing functionality.
2. Run the Application Migration Tool.
3. Fix the errors and build path dependencies between projects.
4. Configure WebSphere Application Server V8.5 to create a JDBC Connection pool for the database.
5. Build an MVC application for WebSphere Application Server V8.5 using Rational Application Developer.
6. Deploy the application into WebSphere Application Server V8.5 and start the page to make sure that it is loaded successfully.

## 6.6.2 Migrating the MVC application

This section describes the steps to perform to migrate the MVC application. This migration effort is straightforward because the MVC application was developed using preferred practices and J2EE standards.

**Downloading the MVC application:** For information about how to download the MVC application that is used in this scenario, see Appendix B, “Additional material” on page 435.

### Importing the EAR file

Start the migration by importing the EAR file that was built for and deployed to WebLogic into Rational Application Developer V8.5.

**Rational Application Developer:** We assume that Rational Application Developer is installed. We also assume that you have a database installed (preferably DB2). No content is needed in the database, as we simply test that a connection can be made. We also assume that WebSphere Application Server V8.5 is installed and created an Application Server profile. We used a new Rational Application Developer V8.5 workspace for our migration example. The workspace folder is referred to as <rad\_workspace>.

Start the migration of MVC application by importing the EAR file with the source code in to Rational Application Developer V8.5 by completing the following steps:

1. In Rational Application Developer Java EE Enterprise Explorer perspective, right-click and select **Import** → **EAR file** (Figure 6-63).

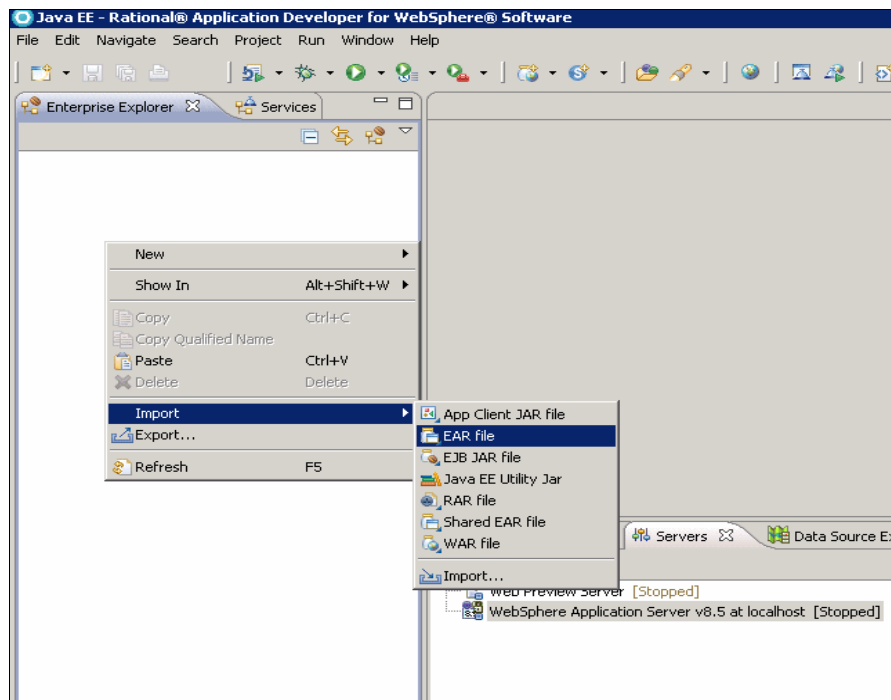


Figure 6-63 Import the EAR file

2. Specify the location of the EAR file and the target run time as **WebSphere Application Server V8.5**. Click **Finish**.

3. You should have three projects in the Enterprise Explorer View (Figure 6-64).
  - Enterprise Application Project: MVC
  - J2EE EJB Project for MVC: MVCEJB
  - J2EE Dynamic Web Project containing servlets and JSPs: MVCWeb

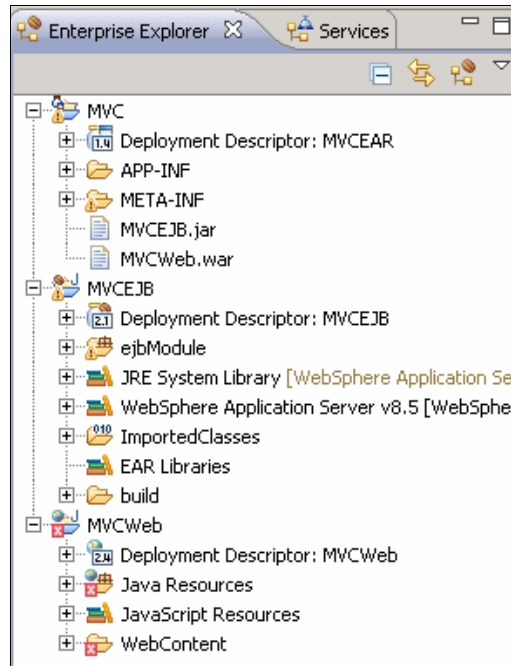


Figure 6-64 Enterprise Explorer View

Here we give a brief description of the EAR file and the contents of the application. An EAR file is a standard JAR (and therefore a compressed file) with an .ear extension, with one or more entries that represent the modules of the application, and a metadata directory that is called META-INF, which contains one or more deployment descriptors.

Table 6-30 shows the contents of our example MVC projects.

Table 6-30 The contents of the MVC projects

Directory structure	Description
MVC.ear/META-INF/	Metadata folder for deployment descriptors.
MVC.ear/META-INF/application.xml	This file contains a mapping to the modules that this application uses. In this case, web module to MVCWeb.war, and EJB Module to MVCEJB.jar.
MVC.ear/META-INF/weblogic-application.xml	WebLogic specific deployment descriptors. In this specific example, there are no WebLogic specific configurations.
MVC.ear/APP-INF/	This folder contains any libraries and classes that are used by the application. In this case, there are none.
MVC.ear /MVCEJB.jar	This is the EJB module. This module contains the EJB beans that are used in this application.



Directory structure	Description
MVC.ear /MVCWeb.war	This is the web module. It contains the servlets and JSPs used in this application.
MVCEJB.jar/META-INF/ejb-jar.xml	This XML file defines the enterprise beans that are used in the application. It is not server-specific and should not require migration. Here we define the EJB MVCModel and the logical reference to the data source.
MVCEJB.jar/META-INF/weblogic-ejb-jar.xml	This XML file is WebLogic specific. This file defines the weblogic-enterprise-bean, and defines the resource-description to map the database JNDI name with the resource reference. This file also exposes the JNDI name for the bean with the jndi-name. This file should be migrated automatically with the WebSphere Migration Toolkit.
MVCEJB.jar/ejbs/*	We have the MVCModel enterprise bean. This bean has its corresponding classes: <ul style="list-style-type: none"> <li>▶ MVCModel.class: The enterprise bean remote interface</li> <li>▶ MVCModelBean.class: The enterprise bean</li> <li>▶ MVCModelHome.class: The enterprise bean remote home</li> </ul> We also have MVCModelDBObject, which is just an object to hold the database information.
MVCWeb.war/WEB-INF/web.xml	This file is not server-specific. It defines the servlet and the servlet mapping.
MVCWeb.war/WEB-INF/weblogic.xml	This file describes the resource references used inside this module. In our case, we reference an EJB bean. Inside, we have a resource-env-description, which maps a resource-env-refname to the jndi-name defined in the EJBs weblogic-ejb.jar.xml file.
MVCWeb.war/WEB-INF/classes/com/ibm/mvc/MVCController.class	Here we have the servlet. This servlet, when called, calls the MVCModel bean. The servlet receives a MVCModelDBObject and forwards it to the JSP.
MVCDemoweb.war/MVCView.jsp	This JSP is called from the MVCController servlet. It takes in an attribute (MVCModelDBObject) and displays it on the browser.

Run the Application Migration Tool by completing the following steps:

1. Run the Application Migration Tool. From the Run menu, click **Analyze** to analyze all the projects in the workspace.

The Software Analyzer View open at the bottom of the window. It contains four tabs. Each tab corresponds to one of the analysis domains that are selected based on the Rule Set selection that you made earlier.

- The Class-Path Review tab list any class path related issues.
- The Java Code Review tab lists Java code-related issues.
- The XML File Review tab lists issues with deployment descriptors and the presence of any WebLogic specific files.
- The JSP Code Review tab lists the issues that are found in JSP files, commonly associated with web projects.

2. Click the **XML File Review** tab (Figure 6-65).

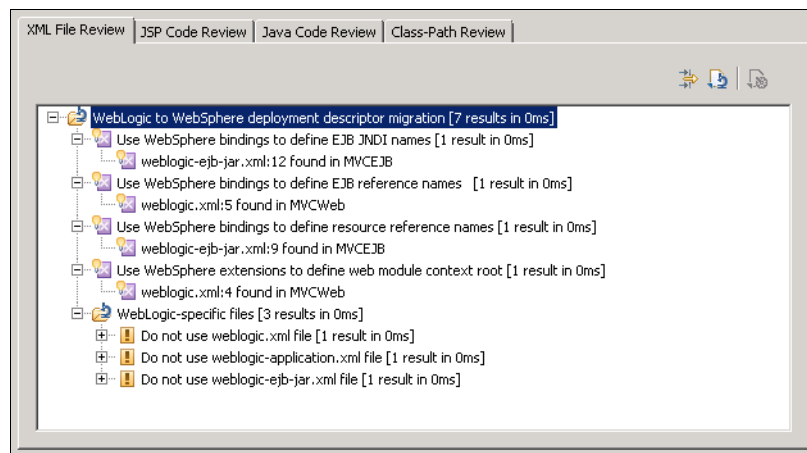


Figure 6-65 XML File Review

The tool scans the XML files in the project and flags the potential problems with WebLogic deployment descriptors by category/rule violation. You can expand each of the rules to see the files in question and which project they belong to.

3. We convert the WebLogic specific JNDI mappings to WebSphere bindings. Right-click the first result, **Use WebSphere bindings to define EJB JNDI names** → **weblogic-ejb-jar.xml**, and select **Quick Fix Preview** (Figure 6-66).

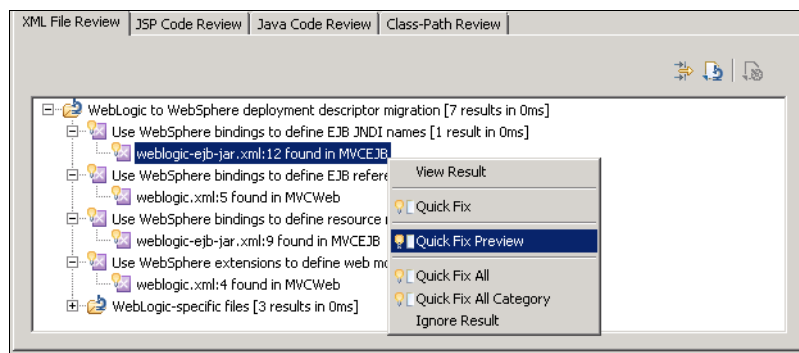


Figure 6-66 XML file Quick Fix Preview

This action opens a comparison and creates a file named MVCEJB/ejbModule/MATAINF/ibm-ejb-jar-bnd.xmi. You can review the changes. See Figure 6-67.

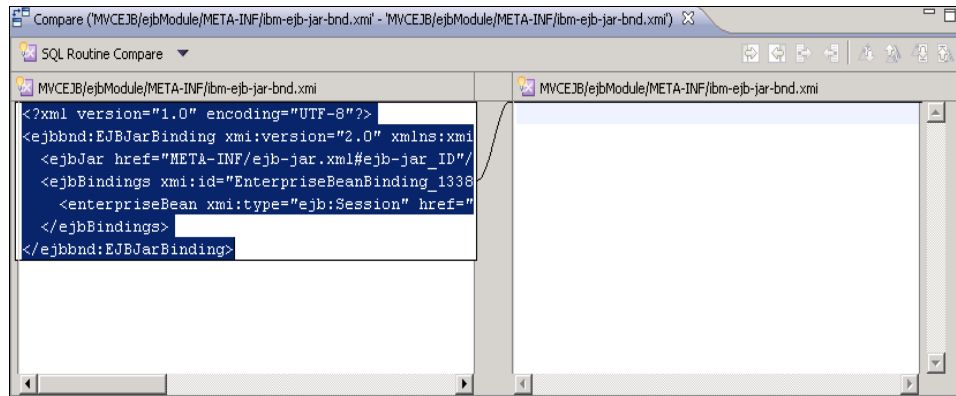


Figure 6-67 The contents of *ibm-ejb-jar-bnd.xmi*

After you are satisfied with the change, you can select **Quick Fix**. This action makes the change in your workspace.

4. Perform step 3 on page 198 for the other three results as well. Ignore the warnings about the WebLogic files that exist, as they are only warnings.
5. Click the **Java Code Review** tab (Figure 6-68).

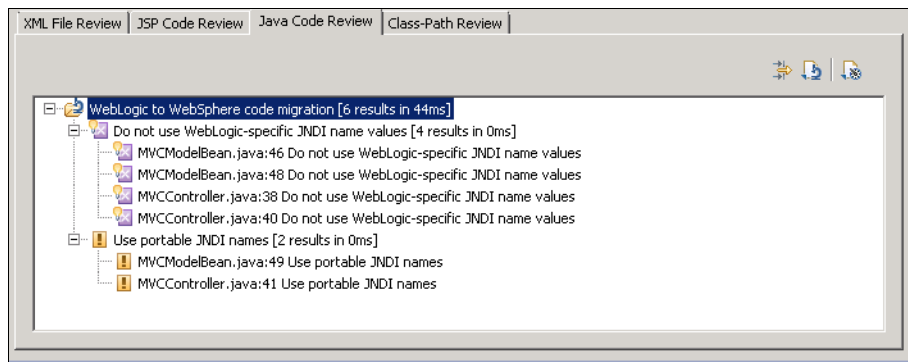


Figure 6-68 Java Code Review

The tool found two rule violations.

**Tip:** You can right-click any of the flagged files under a specific rule to see a list of available options in the menu. For example, under the Java Code Review tab, expand the first rule that is labeled **Do not use WebLogic-specific JNDI name values** and right-click any of the files in the list. From the context menu, select **View Results**. The Java file in question is opened and the cursor is at the line where the code problem is. This tool is handy because it points out the file/line of code that must be fixed if the tool does not provide a quick fix.

- Fix the issues in Java reported by the Migration Toolkit and the problems reported by Rational Application Developer when you first imported the MVC application. Open the **Problems** view to view all errors and warnings reported by Rational Application Developer V8.5 by clicking **Window** → **Show View** → **Other...** **Select General** → **Problems** (Figure 6-69).

Description	Resource	Path	Location	Type
<b>Errors (12 items)</b>				
MVCModel cannot be resolved to a type	MVCController.java	/MVCWeb/src/com/ibm/mvc	line 56	Java Problem
MVCModelDBObject cannot be resolved to a type	MVCController.java	/MVCWeb/src/com/ibm/mvc	line 58	Java Problem
MVCModelHome cannot be resolved to a type	MVCController.java	/MVCWeb/src/com/ibm/mvc	line 32	Java Problem
MVCModelHome cannot be resolved to a type	MVCController.java	/MVCWeb/src/com/ibm/mvc	line 44	Java Problem
MVCModelHome cannot be resolved to a type	MVCController.java	/MVCWeb/src/com/ibm/mvc	line 44	Java Problem
MVCModelHome cannot be resolved to a type	MVCController.java	/MVCWeb/src/com/ibm/mvc	line 44	Java Problem
MVCModelHome cannot be resolved to a type	MVCController.java	/MVCWeb/src/com/ibm/mvc	line 56	Java Problem
MVCModelHome cannot be resolved to a variable	MVCController.java	/MVCWeb/src/com/ibm/mvc	line 43	Java Problem
The import ejbs cannot be resolved	MVCController.java	/MVCWeb/src/com/ibm/mvc	line 19	Java Problem
The import ejbs cannot be resolved	MVCController.java	/MVCWeb/src/com/ibm/mvc	line 20	Java Problem
The import ejbs cannot be resolved	MVCController.java	/MVCWeb/src/com/ibm/mvc	line 21	Java Problem
Undefined type: ejbs.MVCModelDBObject	MVCView.jsp	/MVCWeb/WebContent	line 9	JSP Problem
<b>Warnings (7 items)</b>				

Figure 6-69 The Problems Category

## Fixing the errors and build path dependencies between the projects

These problems are because of dependencies between the MVCWeb project and the MVCEJB project. We must configure the Build path to correct these errors. Complete the following steps:

- To fix the problem, right-click the **MVCWeb** application project and select **Properties**. From the left pane in the Properties window, select **Java Build Path**. Click the **Projects** tab and click **Add**. In the Required Project Selection dialog box, select **MVCEJB** and click **OK** (Figure 6-70).

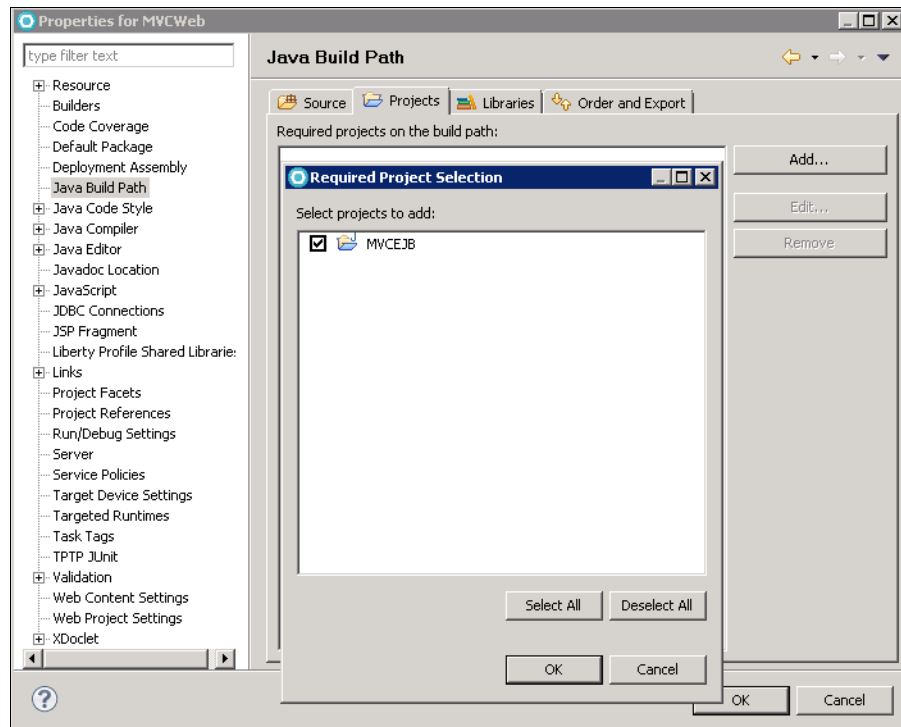


Figure 6-70 The Java Build Path of MVCWeb Application

- Click **OK** on the Properties dialog box to close it.
- From the Eclipse Run menu, select **Analyze Last Launched**. This action reruns the analysis tool with the saved configuration. The focus is returned to the Software Analyzer Results view. The tool creates a report in the left frame of the view with the date and time stamp.
- Click the **Java Code Review** tab. View the category that is labeled **Do not use WebLogic-specific JNDI name values**.
- Right-click the first entry **MVCController.java: Do not use WebLogic-specific JNDI name values** and select **Quick Fix Preview** (Figure 6-71).

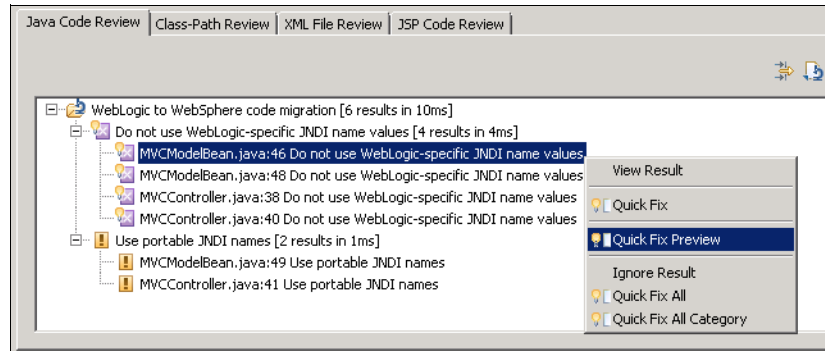


Figure 6-71 Quick Fix Preview of MVCContreller.java

The Quick Fix Preview editor opens. You can examine the code changes before they are committed. In this case, the editor wants to replace the call reference to “weblogic.jndi.WLInitialContextFactory” with the WebSphere equivalent “com.ibm.websphere.naming.WsnInitialContextFactory” (Figure 6-72).

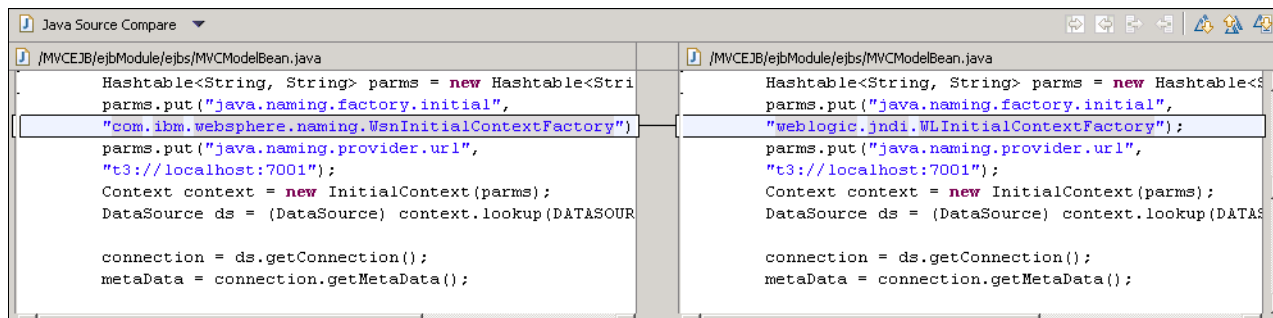


Figure 6-72 The code changes that the Migration Tool finds

- If you are satisfied with the quick fix change, you can choose the **Quick Fix** option to make the change in your workspace.
- Repeat this action for each error, or to fix all the results for the **Do not Use WebLogic-specific JNDI name values** rule, you can select any result within the rule or the rule name itself to start **Quick Fix All**.
- In the Enterprise Explorer view, expand the **MVCEJB** project, browse to the **MVCEJB/ejbModule/META-INF** folder, and delete the **weblogic-ejb-jar.xml** file.
- Rerun the Application Migration Tool. All errors reported by the migration tool and Rational Application Developer v8.5 should be fixed now.

**Tip:** You can look in the files to ensure that everything is migrated.

10. In the Enterprise Explorer view, expand the **MVCWeb** project, browse to the MVCWeb /webContents/WEB-INF folder, and delete the weblogic.xml and weblogic-ejb-jar.xml files.
11. In the Enterprise Explorer view, expand the MVC project, browse to the MVC/META-INF folder, and delete the weblogic-application.xml file.
12. Fix the issues in Java reported by the Migration Toolkit and the problems reported by Rational Application Developer when you first import the MVC application. Open the **Problems view** to view all errors and warnings reported by Rational Application Developer V8.5 by clicking **Window** → **Show View** → **Other...** **Select General** → **Problems**. See Figure 6-69 on page 200.

## Setting up the database connection

Here you set up your connection to the DB2 database through the WebSphere admin console. It is assumed that you already have a database setup. Complete the following steps:

1. Start the server if it is not already running. In the Servers tab, right-click **WebSphere Application Server v8.5 at local host** and select **Start**.
2. Start the administrative console. You can use Rational Application Developer to accomplish this task. In the Servers tab, right-click **WebSphere Application Serverv8.5 at local host** and select **Administration** → **Run Administrative Console**.

**Note:** You can also connect through your browser: By default, the URL is <http://localhost:9060/ibm/console>.

3. Log in using the user name wasadmin and password wasadmin.
4. In the left pane, click **Resources** → **JDBC** → **JDBC providers**. In the Scope section, click **Node=<nodename, Server=<servername>** (Figure 6-73).

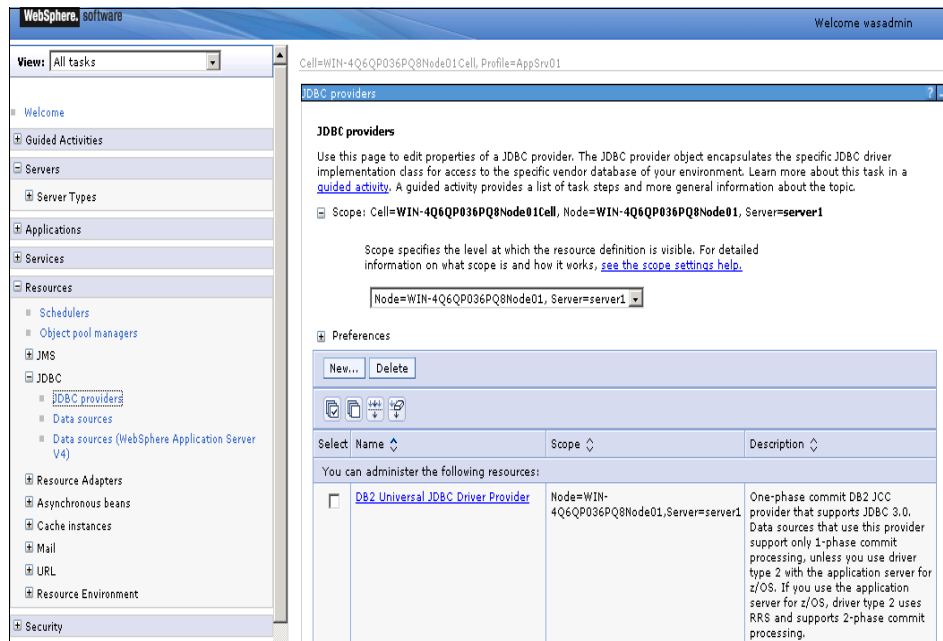


Figure 6-73 JDBC Providers

5. In the Preferences section, select **New**.

6. On the Create a new JDBC Provider page, specify the following information (Figure 6-74):

- Database type: DB2.
- Provider Type: DB2 Universal JDBC Driver Provider.
- Implementation Type: Connection pool data source.
- Name: DB2 Universal JDBC Driver Provider.

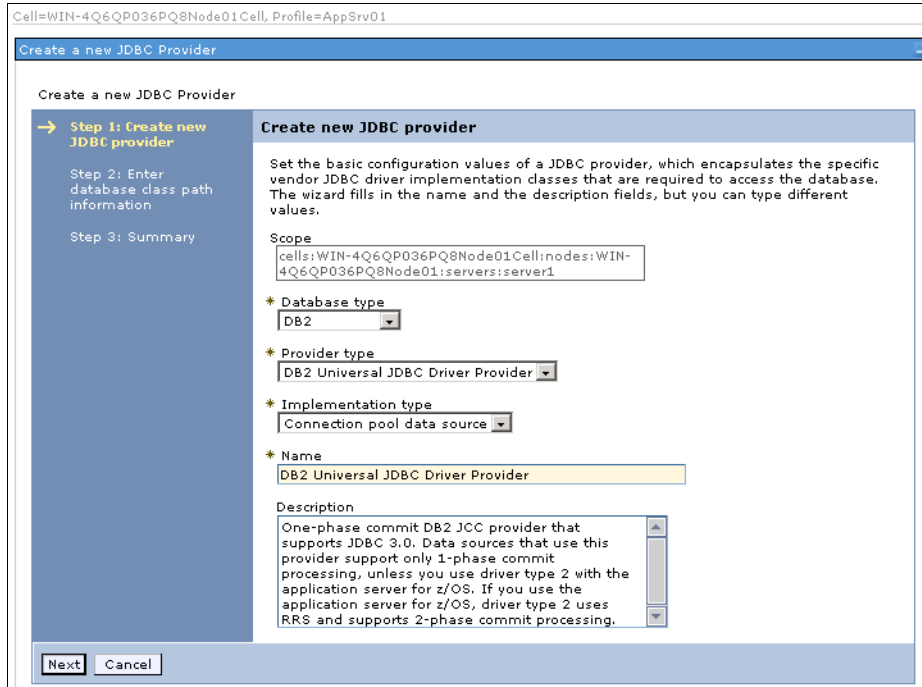


Figure 6-74 Create New JDBC Provider

7. Click **Next**.

8. Specify the location of the db2jcc.jar, db2jcc\_license\_cu.jar, and db2jcc\_license\_cisuz.jar files. These files are in C:\DB2\_INST\_ROOT\IBM\SQLLIB\java. See Figure 6-75.

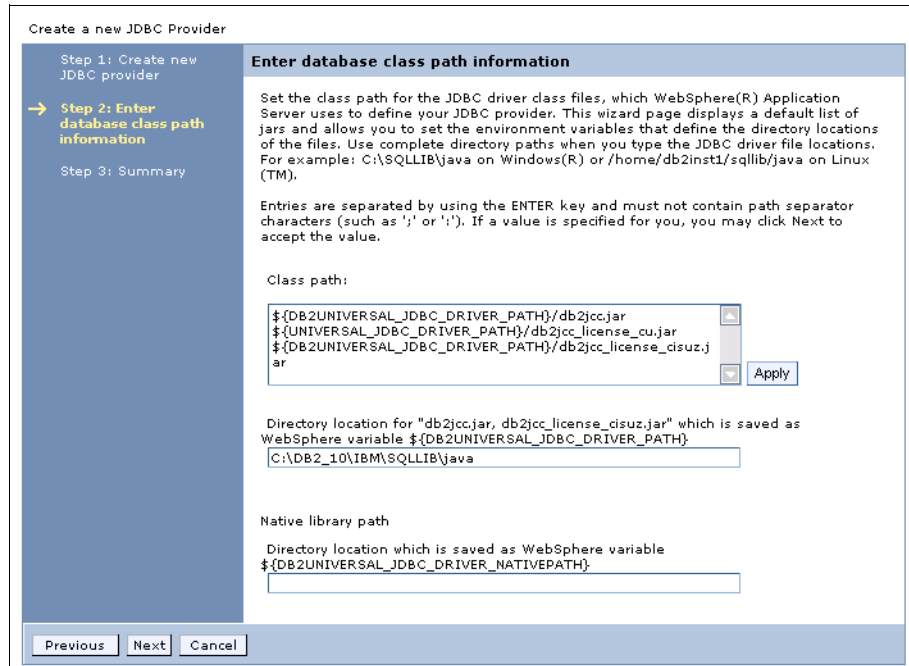


Figure 6-75 The database classpath information

9. Click **Next**.
10. Review the Summary and click **Finish**.
11. Click **Save** at the top of the window.
12. Click **Data sources** from the **JDBC** menu on the left side of the page.
13. In the Scope section, click **Node=<nodename>, Server=<servername>**.
14. Click **New**.
15. On the Create a data source page, specify the following values:
  - Data source name: DB2 Universal JDBC Driver DataSource.
  - JNDI name: jdbc/trade3db.
16. Click **Next**.



17. Select the **Select an existing JDBC provider** radio button, and choose the JDBC provider that you created, that is, **DB2 Universal JDBC Driver Provider** (Figure 6-76).

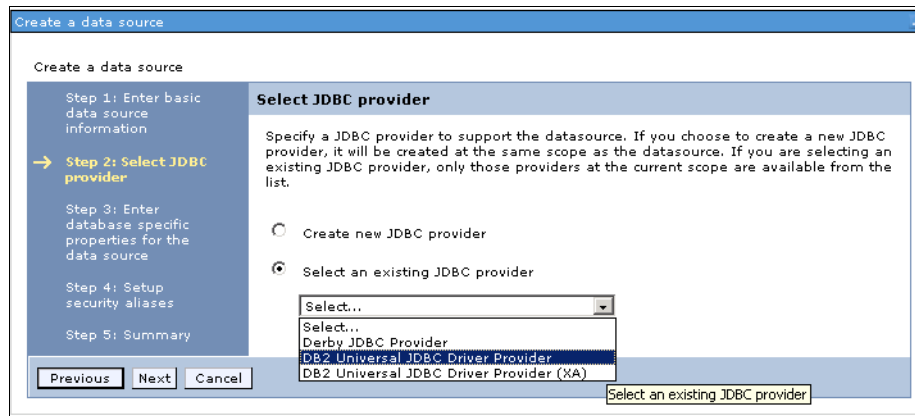


Figure 6-76 Selecting the existing JDBC Provider

18. Click **Next**.

19. Specify the following values (Figure 6-77):

- Database name: trade3db.
- Server name: localhost.
- Port number: 50000.

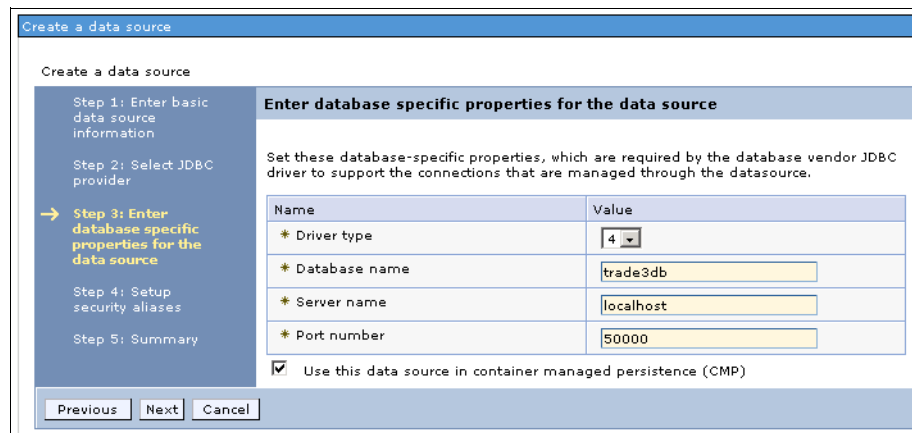


Figure 6-77 The database properties

20. Click **Next**.

21. Leave Step 4 blank for now, as you return to the security configuration later. If you already created a security configuration for this database, you can select it under the Component-managed authentication alias. Click **Next**.

22. Review the Summary and click **Finish**.

23. Click **Save** at the top of the page.

24. Click the newly create Data Source DB2 Universal JDBC Driver DataSource.

25. On the right side of the Configuration tab, under the Related Items header, click the link **JAAS – J2C authentication data** (Figure 6-78). Skip this step if you created the authentication.

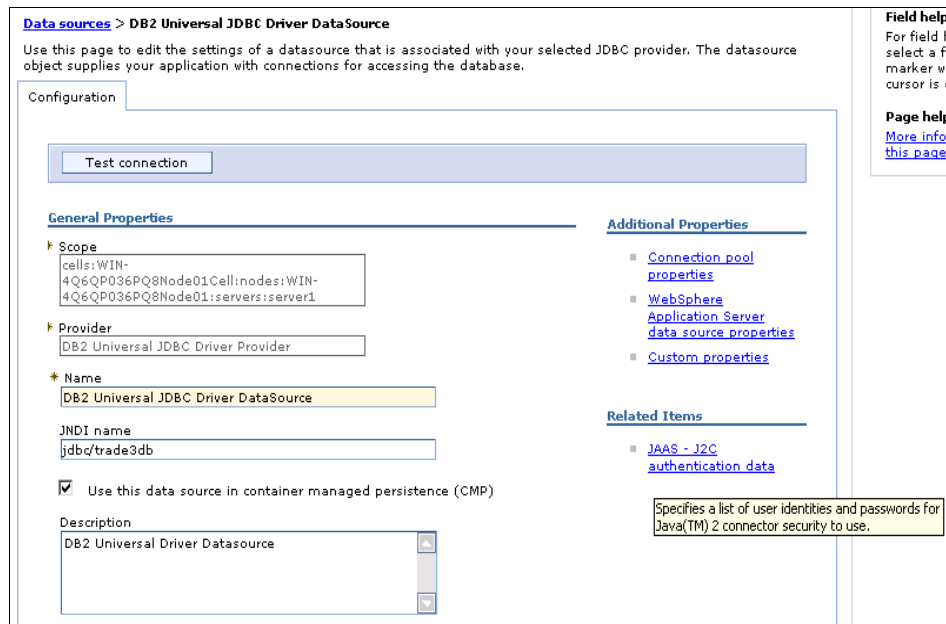


Figure 6-78 Specifying JAAS-J2C authentication data

Here you specify the user name and password for the database. Click **New**.

Specify the following values (Figure 6-79):

- Alias: db2admin.
- User ID: <Database username>. In this example, it is db2admin.
- Password: <Database password>. In this example, it is db2admin.

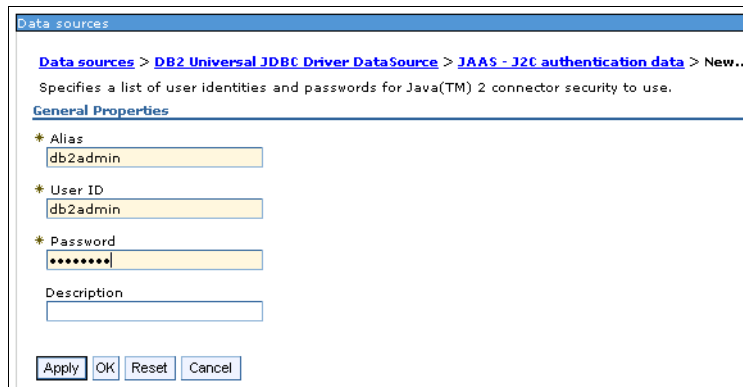


Figure 6-79 New authentication alias

26. Click **OK**.

27. Click **Save**.

28. Return to the **DB2 Universal JDBC Driver DataSource** window. In the Security settings section, specify the following values (Figure 6-80):

- Component-manages authentication alias: <nodeName>/db2admin.
- Mapping-configuration alias: (none).
- Container-managed authentication alias: (none).

Name	Value
* Driver type	4
* Database name	trade3db
* Server name	localhost
* Port number	50000

Figure 6-80 Security settings

29. Click **Apply**.

30. Click **Save**.

31. From the Data sources main page, click **DB2 Universal JDBC Driver DataSource** and click the **Test connection** (Figure 6-81).

Figure 6-81 Test connection

32. You should see message that is shown in Figure 6-82.

Figure 6-82 Test connection message

## Starting the application

To start the application, complete the following steps:

1. Right-click **MVCWeb/Java Resources/src/com.ibm.mvc/MVCController.java** and select **Run As** → **Run on Server** (Figure 6-83).

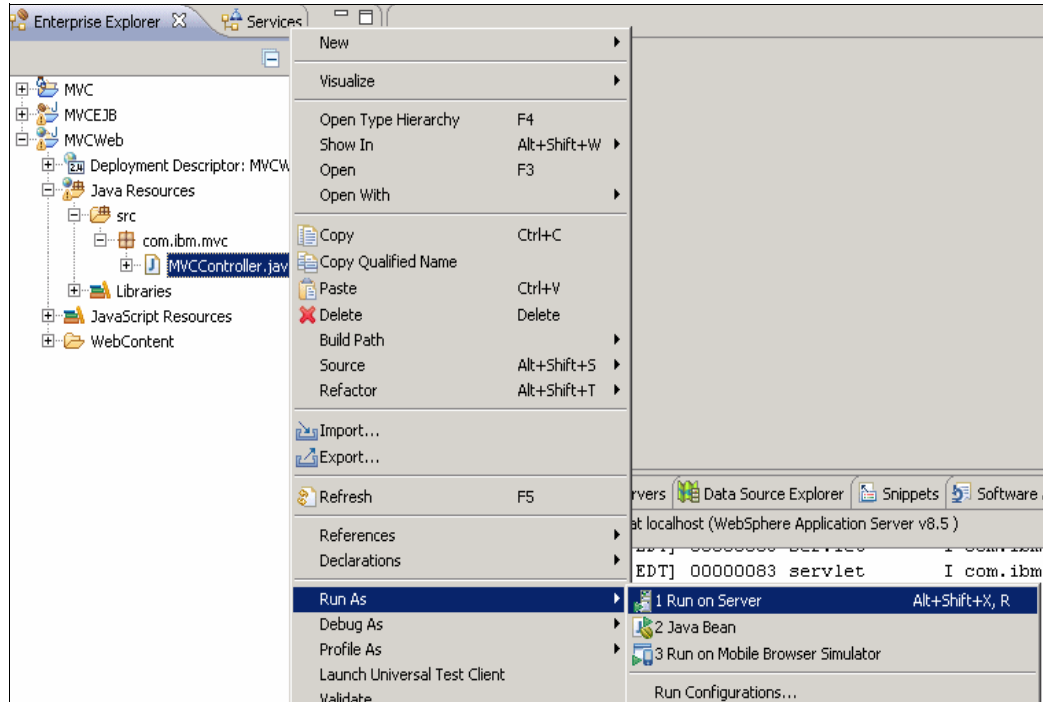


Figure 6-83 Run on the server

2. Select **WebSphere Application Server v8.5 on localhost**.
3. Click **Finish**. If the server is not started, it is started for you.
4. The application should start, and you should see information about the database that you are connected to (Figure 6-84).

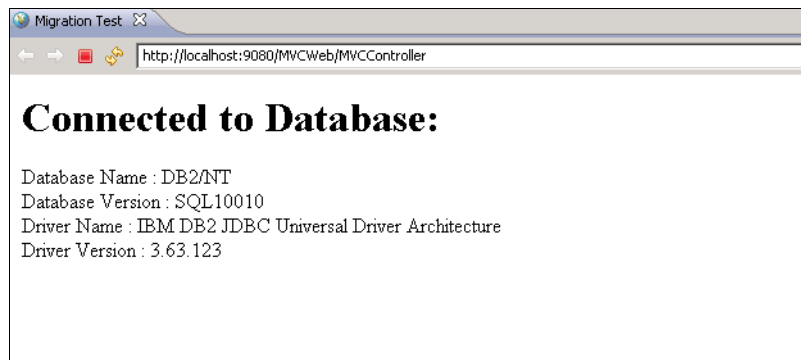


Figure 6-84 The test result

### 6.6.3 Adding the CMP application into MVC project

This section describes how to migrate a Container Managed Persistence (CMP) -EJB application that runs on WebLogic V10.3.6 to WebSphere Application Server V8.5.

#### Creating a table on the Trade database

You have the trade database, so you can create the table on this database. Run the following commands to create a DB2 table. You must start the DB2 CLP.

- ▶ **db2 connect to trade3db user db2admin using db2admin**
- ▶ **db2 -tvf Table.ddl**

You can see the contents of the Table.ddl file in Example 6-6.

*Example 6-6 Table.ddl file*

---

```
CREATE TABLE DB2ADMIN.CATALOG (  
    JOURNAL CHAR(150) DEFAULT 'NULL',  
    CATALOGID CHAR(150) NOT NULL DEFAULT 'NULL',  
    PUBLISHER CHAR(150) DEFAULT 'NULL'  
)  
DATA CAPTURE NONE ;  
ALTER TABLE DB2ADMIN.CATALOG ADD CONSTRAINT CC1325841019875 PRIMARY KEY  
(CATALOGID);  
GRANT ALTER ON TABLE DB2ADMIN.CATALOG TO USER DB2ADMIN WITH GRANT OPTION;  
GRANT CONTROL ON TABLE DB2ADMIN.CATALOG TO USER DB2ADMIN;  
GRANT DELETE ON TABLE DB2ADMIN.CATALOG TO USER DB2ADMIN WITH GRANT OPTION;  
GRANT INDEX ON TABLE DB2ADMIN.CATALOG TO USER DB2ADMIN WITH GRANT OPTION;  
GRANT INSERT ON TABLE DB2ADMIN.CATALOG TO USER DB2ADMIN WITH GRANT OPTION;  
GRANT REFERENCES ON TABLE DB2ADMIN.CATALOG TO USER DB2ADMIN WITH GRANT OPTION;  
GRANT SELECT ON TABLE DB2ADMIN.CATALOG TO USER DB2ADMIN WITH GRANT OPTION;  
GRANT UPDATE ON TABLE DB2ADMIN.CATALOG TO USER DB2ADMIN WITH GRANT OPTION;
```

---

**The Table.ddl file:** The Table.ddl file creates the required tables and grants the necessary permissions that are required for this application.

#### Importing the CMP-EJB JAR file

Start the migration by importing the application that is built for and deployed to WebLogic in to Rational Application Developer V8.5.

Start the migration of the EJB module by importing the JAR file into Rational Application Developer V8.5 by completing the following steps:

1. In Rational Application Developer Java EE Enterprise Explorer perspective, right-click and select **Import** → **EJB JAR file** (Figure 6-85).

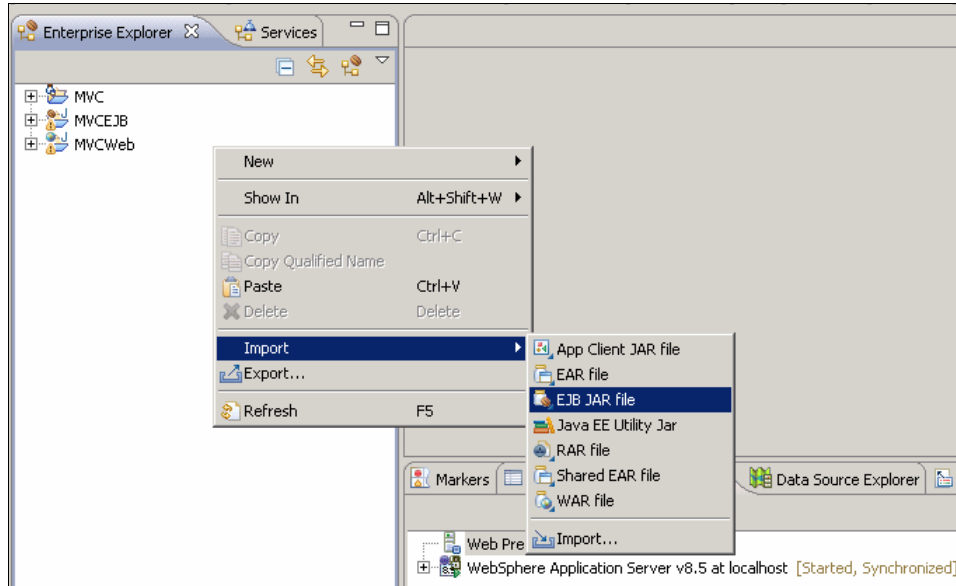


Figure 6-85 Import EJB JAR file

2. Specify the location of the EJB JAR file and the target run time as WebSphere Application Server V8.5. Click **Finish**.
3. You should have entityejbEAR Enterprise Application Project in addition to three MVC Projects in the Enterprise Explorer View (Figure 6-86).

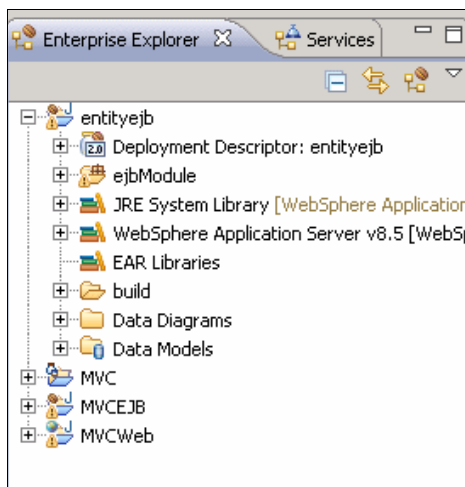


Figure 6-86 EJB Application

Here is a brief description of the JAR file, and the contents of the application. An Enterprise JavaBeans (EJB) module is used to assemble one or more enterprise beans into a single deployable unit. An EJB module is stored in a standard Java archive (JAR) file (Table 6-30 on page 196).

Table 6-31 The contents of the EJB projects

Directory structure	Description
entityejb.jar/META-INF/ejb-jar.xml	The <code>ejb-jar.xml</code> file defines an entity EJB with the EJB name "Catalog." The example EJB has the CMP fields <code>catalogId</code> , <code>journal</code> , and <code>publisher</code> . The primary key field is <code>catalogId</code> . This structure is not WebLogic specific and should not require migration.
entityejb.jar/META-INF/weblogic-ejbjar.xml	This file defines the <code>weblogicenterprise-bean</code> . The JNDI name for an EJB is specified in the <code>weblogic-ejbjar.xml</code> deployment descriptor with the <code>jndi-name</code> element.
entityejb.jar/META-INF/weblogic-cmp-rdbms-jar.xml	This file is specific to WebLogic and defines the database persistence information for a CMP entity EJB. The file includes the table name for an entity EJB, the data source to connect to the database, and the columns corresponding to the entity EJB CMP fields.

## Running the Application Migration Tool

To run the Application Migration Tool, complete the following steps:

1. Run the Application Migration Tool. From the Run menu, click **Analyze** to analyze the `entityejb` project in the workspace.
2. Click the **XML File Review** tab (Figure 6-87).

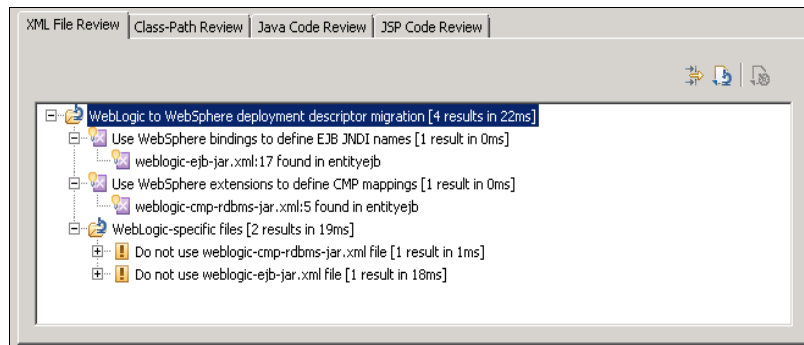


Figure 6-87 XML File Review tab

As you can see, the tool scanned the XML files in the project and flagged the potential problems with WebLogic deployment descriptors by category/rule violation. You can expand each of the rules to see the files in question and which project they belong to.

3. Convert the WebLogic specific JNDI mappings to WebSphere bindings. Right-click the first result (**Use WebSphere bindings to define EJB JNDI names**). Select `weblogic-ejb-jar.xml` and then select **Quick Fix Preview**.

After you are satisfied with the change, select **Quick Fix**, which makes the change in your workspace.

4. Perform step 3 on page 211 for the other three results as well. Ignore the warnings about the WebLogic files that exist, as they are only warnings.

When you click **Quick Fix** for Use WebSphere extensions to define the CMP mappings rule, the migration tool examines the `weblogic-cmpdbms-jar.xml` WebLogic specific file and creates a Data Model named `SAMPLE.dbm` and map file named `Map.mapxmi`. You can map the EJB beans and fields corresponding to tables and columns, as you did when you migrated the Trade application, as shown in Figure 6-35 on page 164.

## Creating the CMP mappings manually

Use the Rational Application Developer to complete EJB to RDB mapping. Use the EJB to RDB mapping wizard, which creates the necessary mapping files that are needed by the CMPs to use the existing database schema.

Complete the following steps:

1. In the Enterprise Explorer view, right-click the EJB project `entityejb` and select **JavaEE** → **EJB to RDB mapping(1.x-2.x)** → **GenerateMap** (Figure 6-88).

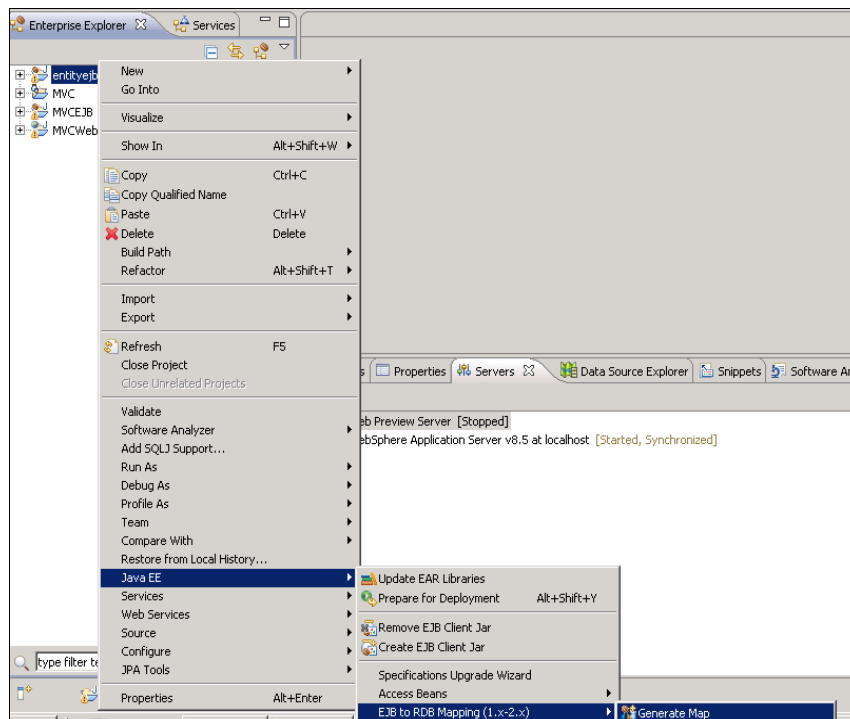


Figure 6-88 Generate Map

2. Click **Create a new backend folder** and click **Next**.
3. Click **Meet-In-The-Middle** on the next window and click **Next**.
4. Click **New** to create a Database Connection.



5. Clear **Use default naming convention** and enter TRADE3 as the Connection Name. Select **DB2 for Linux, UNIX, and Windows** under the **Select a database manager** category. Change the JDBC driver type, and enter a database name, user name, and password (Figure 6-89). Click **Save Password**.

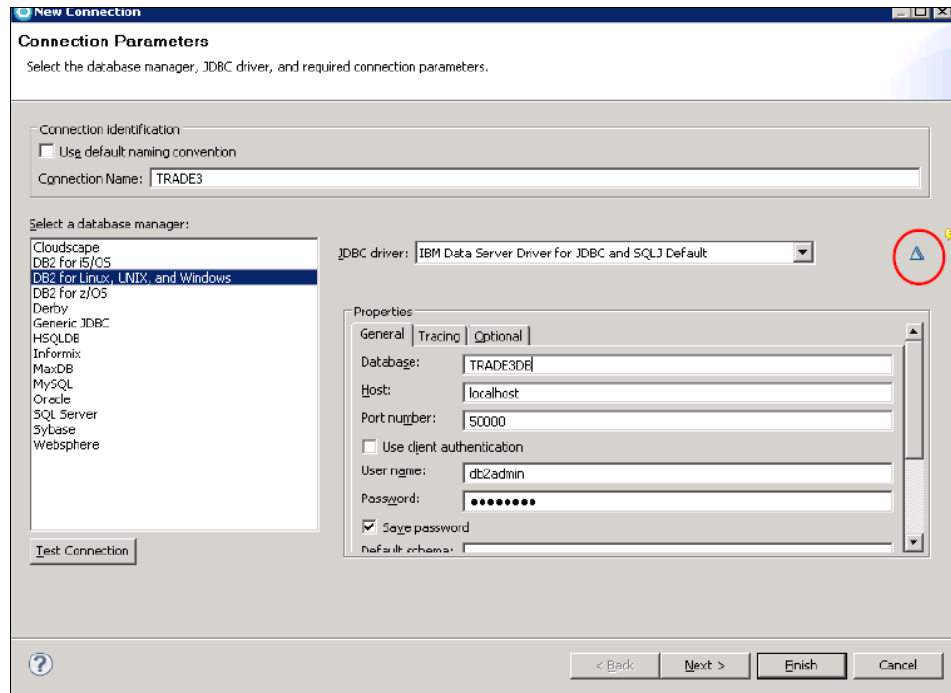


Figure 6-89 Database Connection definition

6. In Figure 6-89, click the **blue triangle icon** (to the right of the Drivers drop-down menu).
7. The Edit Driver Definition page is displayed. Click the **Jar List** tab.
8. Highlight the existing driver JAR files, one at a time, and click **Remove JAR/Zip**.

9. Click **Add JAR/Zip** and browse to the C:\DB2\_97\IBM\SQLLIB\java directory. Select the **db2jcc.jar** file and click **OK**. Repeat this step to add the db2jcc\_license\_cu.jar file (Figure 6-90).

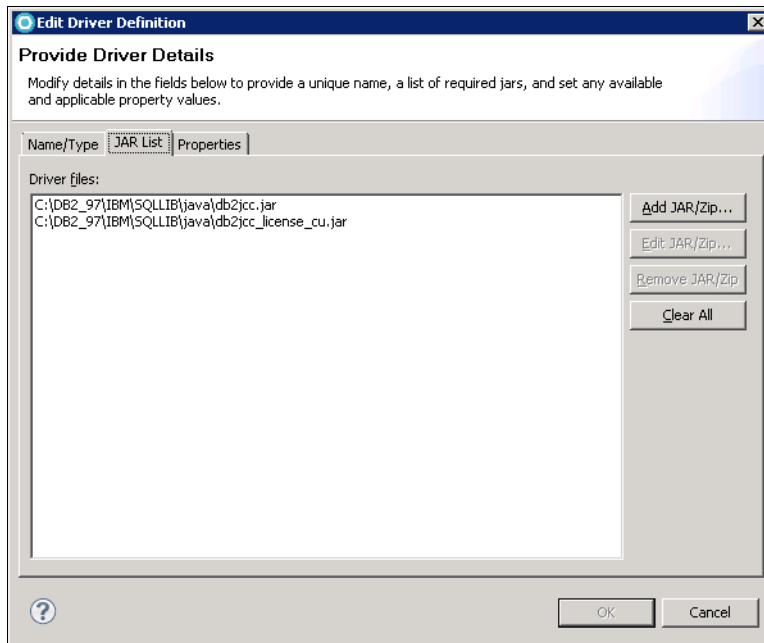


Figure 6-90 Edit Jar List

10. Click **OK** in the Edit Driver Definition dialog box. You should return to the Driver Properties dialog box.
11. In the Driver Properties dialog box, click **Test connection**. You should see a success message.
12. Click **Next** and click **Finish**. You have a database connection named TRADE3.
13. Click the TRADE3 database connection and click **Next**.

14. Select only CATALOG tables, as we want to map beans and CMP fields to the columns under that table (Figure 6-91).

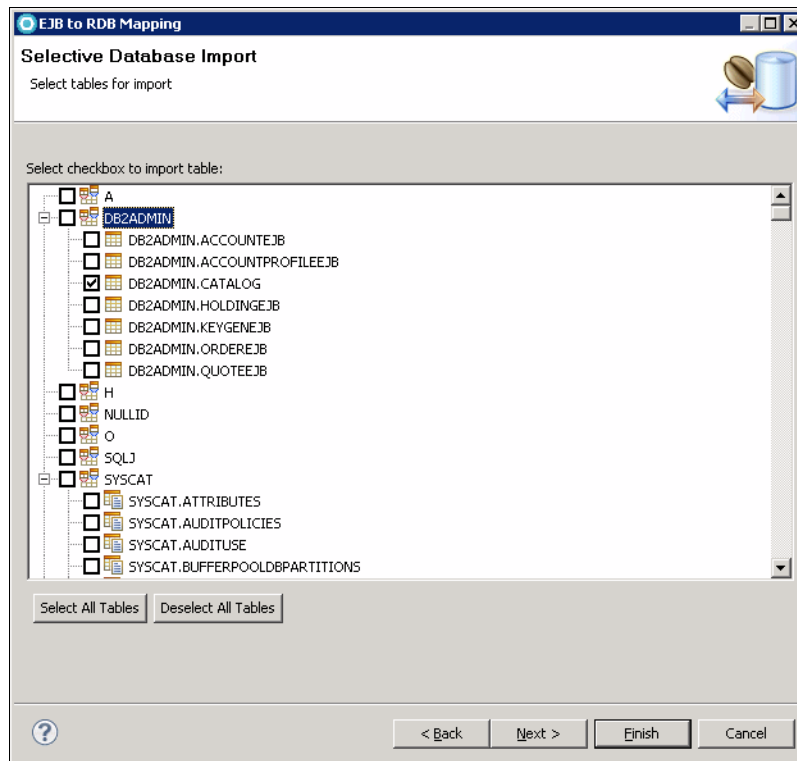


Figure 6-91 The database table selection

15. Click **Next**, and for **Select Meet-in-the-Middle Mapping Options**, select **None**.
16. Click **Finish**.
17. The wizard creates the Map.mapxmi file, which is used to map CMP fields to table columns. The mapping editor is open. The file is in the following folder:  
`/entityejb/ejbModule/META-INF/backends/DB2UDBNT_V97_1`
18. The data model is imported and in the following folder:  
`/entityejb/Data Models/TRADE3DB.dbm`

**Tip:** If you look at the Problems view, you should see one or more errors that are reported because of CMP mappings. This situation occurs because you have not created the actual mapping of the EJBs attributes to table columns.

19. With the mapping editor still open, right-click the EJB **project:entityejb** and select **Match by Name** (Figure 6-92). It should take a few seconds for the mapping to complete. Click **File** → **Save** to save the mapping file. All errors should disappear from the Problems view.

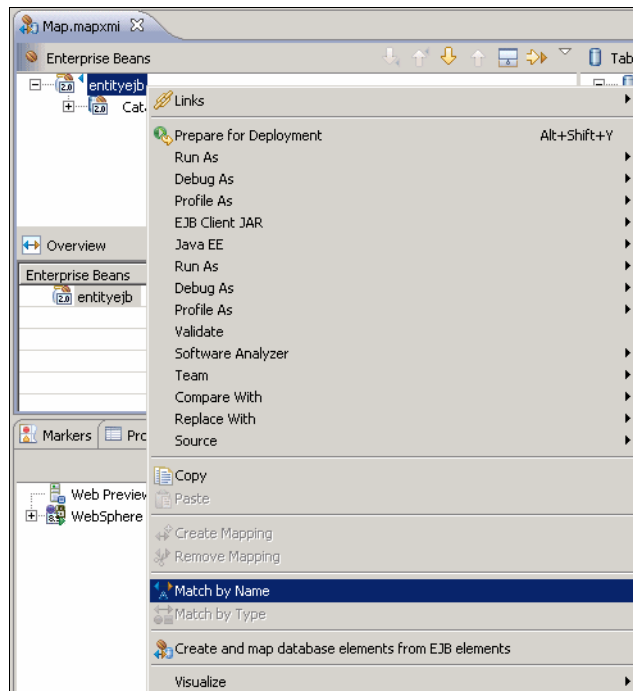


Figure 6-92 Match By Name

20. To verify that all EJB elements are mapped to their corresponding table columns, navigate to the `/entityejb/ejbModule/META-INF/backends/DB2UDBNT_V97_1` folder, open the `Map.mapxmi` file, and enable the **Show only the unmapped objects** (Figure 6-93).

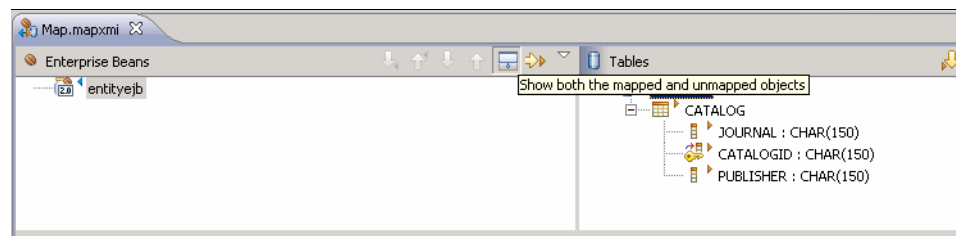


Figure 6-93 Mapped EJB objects

21. Close the mapping file.

**Important:** Verify that the back end that you created is selected for the application. Open the `/entirety/ejb/ejbModule/META-INF/ejb-jar.xml` file with the Deployment Descriptor Editor. On the Overview tab, scroll all the way down and verify that the Backend ID is `DB2UDBNT_V97_1`, as shown in Figure 6-94.

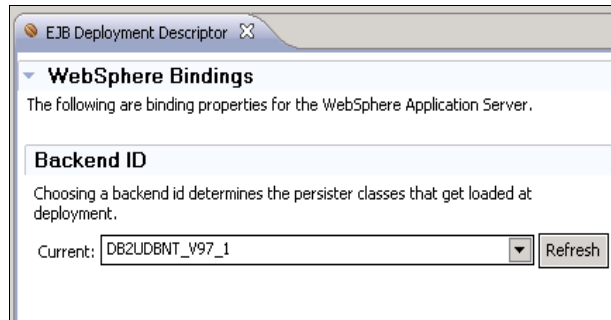


Figure 6-94 Backend ID in `ejb-jar.xml`

**Tip:** You can use the Data Source Explorer option of Rational Application Developer to see database-specific properties, such as schema, tables, roles, and groups of the database profile that you defined so far.

22. Click the **Deployment Descriptor**, select the **Bean** tab, and click **Catalog**.
23. In the WebSphere binding section at the lower right, update the CMP connection factory JNDI Name: `jdbc/trade3db` and Container authorization type to `Per_Connection_Factory` (Figure 6-95).

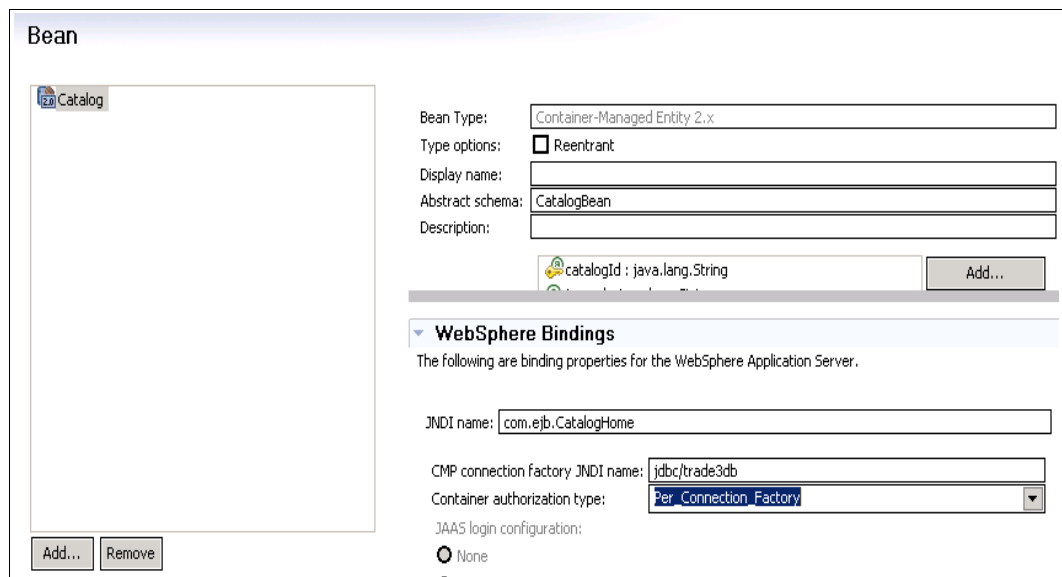


Figure 6-95 Defining the JNDI Name in `ejb-jar.xml`

24. Save the Deployment Descriptor.

## Testing the CMP EJB Module using the IBM Universal Test Client

To test the CMP EJB Module using the IBM Universal Test Client, complete the following steps:

1. Right-click **ejbEAR Project in Rational Application Developer** and select **Run As** → **Run On Server**. Select the **WebSphere Application Server v8.5 server** and click **Finish**.
2. A Test Client opens. When prompted, enter the Administrator user name (wasadmin) and password (wasadmin). You are now logged in to the Universal Test Client.
3. Using the JNDI Explorer, select **com.ejb.CatalogHome** to add it to the EJB Beans section of the navigation pane. Select the home's **create()** method. Enter a value, such as 1111, and click **Invoke** (Figure 6-97 on page 219).

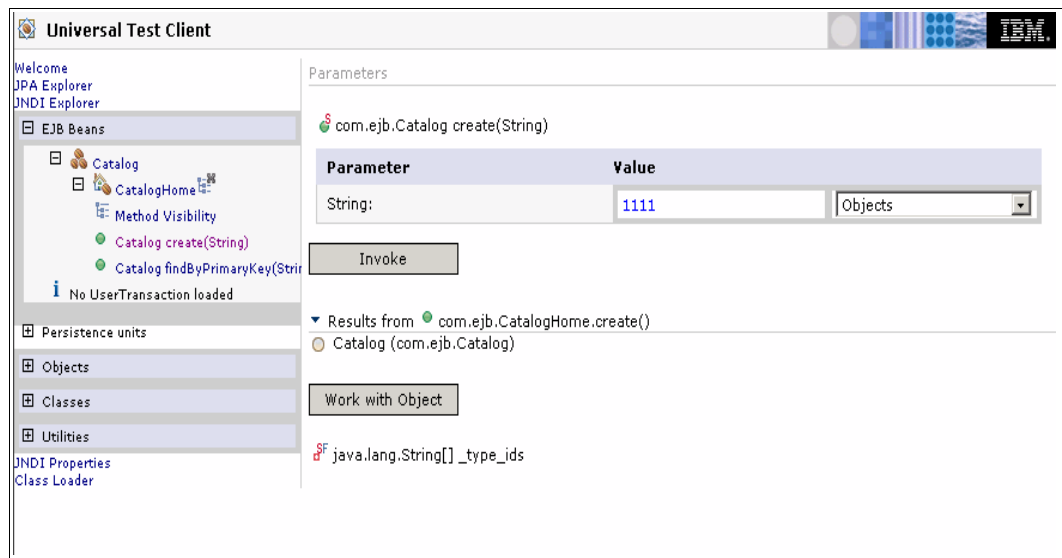


Figure 6-96 Creating an EJB object

4. In the Results window, there is an instance named **com.ejb.Catalog**. Click **Work with Object** to add the instance to the EJB Beans list.

5. Use the available setter and getter methods to write and read the data (Figure 6-97).

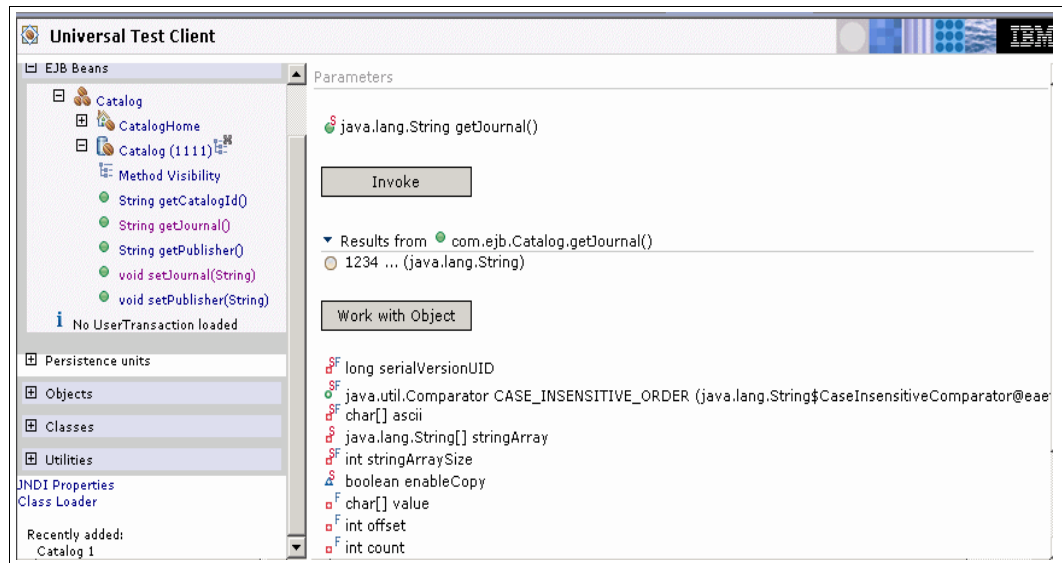


Figure 6-97 Use the set and get methods

## 6.6.4 Summary

You completed the migration of the MVC application, including the CMP-EJB beans by using the Migration Toolkit and Rational Application Developer EJB to RDP mapping wizard.

You deployed the compiled EAR file without having to change much of the Java code. We demonstrated manual mapping of CMP EJB mappings for developers who want to map an EJB object and CMP fields to explicitly specified database table and columns.







# Migrating from Oracle Application Server

This chapter explains the process for migrating J2EE applications from Oracle Application Server 10.1.2 to WebSphere Application Server V8.5. We describe the details of the problems we encountered during the migration process, and the solutions to these problems.

This chapter describes the following topics:

- ▶ Introduction
- ▶ Prerequisites and assumptions
- ▶ Application Migration Tool - Oracle AS to WebSphere
- ▶ Comparison of WebSphere Application Server V8.5 and Oracle Application Server 10.1 terminology
- ▶ Installing Oracle Application Server 10.1.2
- ▶ Migrating the sample application

## 7.1 Introduction

Oracle Application Server is a J2EE 1.4 compliant application-serving platform, which is one of the products in the Oracle Fusion Middleware product family. The example application that is used for the migration process was developed by the authors of this book, and it includes the dependent features of Oracle Application Server to make the migration process realistic enough to be taken as a reference for actual migration operations. We used the Application Migration Tool - Oracle AS to WebSphere included within the Application Migration Toolkit V3.5 for the migration process. We describe the details of the IBM WebSphere Application Server Migration Toolkit in Chapter 4, “Installation and configuration of the Application Migration Tools” on page 89.

## 7.2 Prerequisites and assumptions

There are prerequisites and assumptions you must consider before you continue with the migration examples. One basic requirement is that you should have the destination environment installed and configured as described in Chapter 4, “Installation and configuration of the Application Migration Tools” on page 89. That chapter provides initial background knowledge if the reader is not yet familiar with IBM products.

You should be familiar with the J2EE specification and architecture, and be able to understand and run basic SQL commands. You should also be familiar with Eclipse or Rational Application Developer V8.5, and have basic familiarity with Oracle Application Server 10.1.2.

The following list of products should be installed to complete the migration example:

- ▶ IBM WebSphere Application Server V8.5

You can see the terminology differences between Oracle Application Server and WebSphere Application Server on Table 7-1 on page 223.

- ▶ IBM Rational Application Developer V8.5

We use Rational Application Developer V8.5. Rational Application Developer V8.5 provides a complete suite for an enterprise application developer, from design to deployment.

- ▶ Oracle Database 11g Release 2

In the sample application that is used in this chapter, we used Oracle 11g Release 2.

- ▶ Java SE 6

We use Java SE 6 because it is supported by WebSphere Application Server V8.5 and we must make sure that our applications work well in this Java virtual machine (JVM). The WebSphere Application Server V8.5 installation includes Java SE 6, so you do not need to install this additional JDKs on your machine.

## 7.3 Application Migration Tool - Oracle AS to WebSphere

Application Migration Tool - Oracle AS to WebSphere (Application Migration Tool) is part of the Migration Toolkit, which is described in detail in Chapter 4, “Installation and configuration of the Application Migration Tools” on page 89.

Application Migration Tool - Oracle AS to WebSphere can help you in the following areas when you migrate J2EE 1.4 applications from Oracle Application Server (starting with Oracle Containers for J2EE V9.x) to WebSphere Application Server V7, V8, or V8.5:

- ▶ Oracle Server Startup Classes: Detect the usage of Oracle startup and shutdown interfaces
- ▶ Deployment Descriptors: Detect and convert Oracle specific deployment descriptor elements to IBM deployment descriptor elements
- ▶ Oracle-specific APIs: Detect Oracle proprietary packages in Java code
- ▶ Oracle-specific files: Detect Oracle proprietary XML files in the application
- ▶ Oracle-specific properties: Detect Oracle proprietary properties that are used in Java code
- ▶ JSPs: Detect Oracle proprietary tag libraries
- ▶ Web services: Migrate Oracle web services

## 7.4 Comparison of WebSphere Application Server V8.5 and Oracle Application Server 10.1 terminology

Some of the similarities and differences in terminology between WebSphere Application Server and Oracle Application Server can be found in Table 7-1.

Table 7-1 Terminology comparison

WebSphere Application Server	Oracle Application Server
Cell	Oracle Application Server cluster topology
Node	Physical Machine
Node Agent	Daemon Process Used by OEM Application Server Control
Server Instance	OC4J Instance
Deployment Manager	Oracle Enterprise Manager Application Server Control
Cluster	OC4J groups
Base Server	OC4J Server

## 7.5 Installing Oracle Application Server 10.1.2

This section briefly describes how to install Oracle Application Server 10.1.2, which is necessary to complete the migration scenario. You can install Oracle Application Server 10.1.2 by completing the following steps:

1. In the setup directory of Oracle Application Server, double-click the setup.exe file to start the installation process.
2. The Welcome window opens. Click **Next**.
3. At the Specify Inventory Directory window, choose the default Oracle Inventory directory. The default is C:\Program Files\Oracle\Inventory. Click **Next**.

4. When the Specify File Locations window opens, provide the requested products list file as the source, provide a name to the application server (the default is `oracleas1`), and specify the installation directory (the default is `C:\OraHome_1`). Click **Next**.
5. Select **Oracle Application Server 10g** from the list of products, and click **Next**.
6. Select **J2EE and Web Cache** as the Installation Type (and other components as needed) and click **Next**.
7. At the Pre-Installation Requirements window, make sure that you meet the requirements listed first. Then, check the check box next to the requirements to confirm. Click **Next**.
8. Select the Configuration Options for the modules you selected to install. Click **Next**.
9. Specify the Port Configuration Options. The default is Automatic, which means Oracle Application Server uses default ports automatically. You can also choose the **Manual** option if you specify a `staticports.ini` file that includes the port information. Click **Next**.
10. Specify an instance name and the `ias_admin` password. Click **Next**.
11. When the Summary window opens, check the summary and click **Install**.
12. If the installation is successful, the end of installation window opens and indicates that the installation finished successfully. Click **Exit** to end the installation process.

You should now have a working copy of Oracle Application Server 10.1.2.i.

## 7.6 Migrating the sample application

In this section, we cover the steps for migrating a sample application that is written for Oracle Application Server to WebSphere Application Server V8.5. This sample application was written by the residency team for this project, and can be used to query a book catalog by the name of the author, publisher, and so on.

### 7.6.1 Migration approach

To migrate the sample applications, you must import them into the development environment that the Migration Toolkit is installed on. We chose Rational Application Developer V8.5 for our example. We set up an initial source environment to ensure that the application runs correctly in Oracle Application Server. Then, we used the Application Migration Tool - Oracle AS to WebSphere r to identify and fix the migration-related issues within the applications.

### 7.6.2 Configuring the initial environment

After you install the Oracle Application Server, configure the environment to be able to run the applications on the Oracle platform. This section includes information about acquiring the application files, deploying the sample applications into Oracle Application Server, and creating the database configuration.

#### Acquiring the sample application

For details about how to download the sample application that is used in this scenario, see Appendix B, “Additional material” on page 435. After you download the file, you see three EAR files, each of which corresponds to one sample application.

- ▶ `SampleEJB.ear`
- ▶ `SampleEJBClient.ear`
- ▶ `SampleWebService-GetDates-WS.ear`

## Creating the application database

The sample applications require only one table named CATALOG. You should run the following DDL to create the CATALOG table:

```
CREATE TABLE Catalog (id INTEGER PRIMARY KEY NOT NULL,  
journal VARCHAR(100), publisher VARCHAR(100), edition VARCHAR(100),  
title VARCHAR(100), author VARCHAR(100));
```

## Creating the data source

In order for the application to work, you must create a data source in Oracle Application Server named `jdbc/OracleDBConnectionDS`. This value is specified in the JPA `persistence.xml` file in the `ejb2` module. To create the data source, complete the following steps:

1. Log in to the Oracle Enterprise Manager.
2. Click the appropriate OC4J instance.
3. Click **Administration** in the instance menu.
4. Click **Data Sources** in the Application Defaults section.
5. Click **Create** at the upper right of the Data Sources window.
6. Enter the information in Table 7-2 in to the corresponding fields.

Table 7-2 Creating the data source

Field name	Value
Name	OracleDataSource
DataSource Class	com.evermind.sql.DriverManagerDataSource
JDBC URL	jdbc:oracle:thin:@//localhost:1521/SID_VALUE
JDBC Driver	oracle.jdbc.driver.OracleDriver
Schema	SCHEMA_NAME

7. Enter the appropriate user name and password in the DataSource user name and password section.
8. Enter `jdbc/OracleDBConnectionDS` in the Location field in JNDI Locations section.
9. Click **Apply**.

**Important:** The `SID_VALUE` and `SCHEMA_NAME` are specific to your Oracle Database installation. By default, Oracle creates an SID named XE, and a schema named OE or HR.

## Deploying the application into Oracle Application Server

To deploy the application into Oracle Application Server, complete the following steps:

1. Log in to the Oracle Enterprise Manager.
2. Click the appropriate OC4J instance to deploy in to.
3. Click **Applications** on the instance menu.
4. In the Applications window, click **Deploy EAR File**.
5. Click **Browse** to choose the sample application EAR file.
6. Enter the name of the application in to the Application Name field and click **Continue**.

7. Specify the URL Mapping to your application and click **Next**.
8. Leave the JAZN XML User Manager selection chosen and click **Next**.
9. Review the settings and click **Deploy**.

## 7.6.3 Importing the sample application into Rational Application Developer V8.5

In our sample migration, we use Rational Application Developer V8.5 to identify and perform the changes that are needed in the application to migrate it to WebSphere Application Server V8.5. The first stage is to import the application, which is made up of three EAR files, into Rational Application Developer by completing the following steps:

1. Click **File** → **Import**, then **Java EE** → **EAR File**, and then click **Next**.
2. Click **Browse** and select the **SampleEJB.ear** file.
3. Click **Finish** to import the `SampleEJB.ear` file. This action creates two projects: the `SampleEJB` project and the `ejb2` project, which is the `ejb` module for the `SampleEJB` project.

**XML error:** You see in the Markers view that there is an XML problem reported in the `ejb2` project. You can correct that problem by completing the steps that are shown in “Fixing the reported problems” on page 226.

4. Repeat this process for the `SampleEJBClient.ear` and the `SampleWebService-GetDates-WS.ear` files. Importing the `SampleEJBClient.ear` file creates two projects, `SampleClientEJB` and `webapp1`, while importing `SampleWebService-GetDates-WS.ear` creates `SampleWebService-GetDates-WS` and `WebServices`.

**Application source:** In this sample, the EAR files include the source of the application, so a separate step is not needed to import the source.

### Fixing the reported problems

After you import the EAR files into Rational Application Developer, 35 errors and three warnings are reported (Figure 7-1).

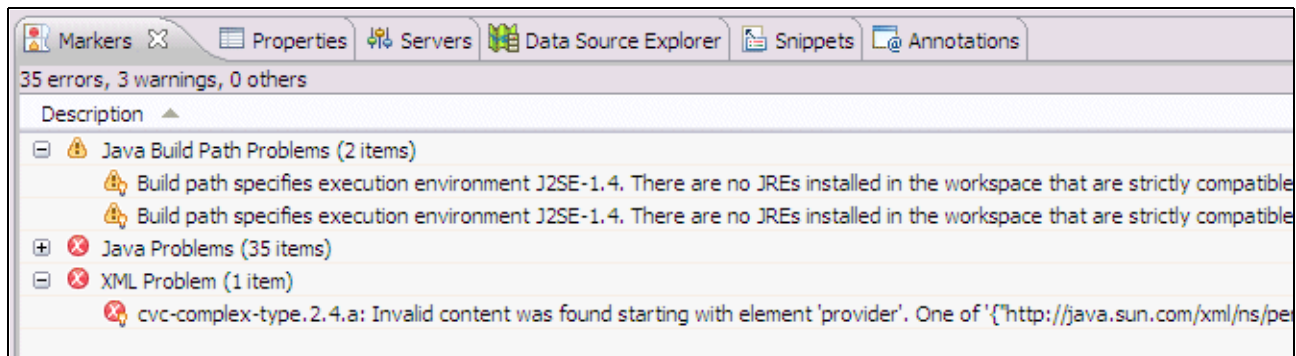


Figure 7-1 The Markers view in Rational Application Developer after you import all the EAR files that make up the sample application

We resolve these problems to have a clean workspace in which to run the Application Migration Tool and to do the deployment to WebSphere Application Server V8.5.

### Java Build Path problems

The first group of issues are all related to the availability of the various JDKs originally used to develop the application to Rational Application Developer. You can resolve these issues by setting the target run time for these projects to WebSphere Application Server V8.5. To do so, complete the following steps:

1. Right-click the **ejb2** project and select **Preferences**.
2. Select the **Targeted Runtimes** view in the list of properties
3. If “WebSphere Application Server v8.5” is not shown in the list of run times, click **New...** and complete the following steps (we assume that you already have an application server profile that is defined on your WebSphere Application Server v8.5 installation):
  - a. Select **WebSphere Application Server v8.5** in the list on run times on the New Server Runtime Environment wizard and click **Next**.
  - b. Click **Browse...** and select the root directory of your WebSphere Application Server V8.5 installation and then click **Finish**.
4. Select the **WebSphere Application Server V8.5** check box in the list of Targeted Runtimes and click **OK**.

These steps must be repeated for the other five projects that make up the application.

After these updates are made, both the Java Build Path problems and most (but not all) of the Java problems are resolved.

### The remaining Java problems

There are four remaining Java problems (Figure 7-2).

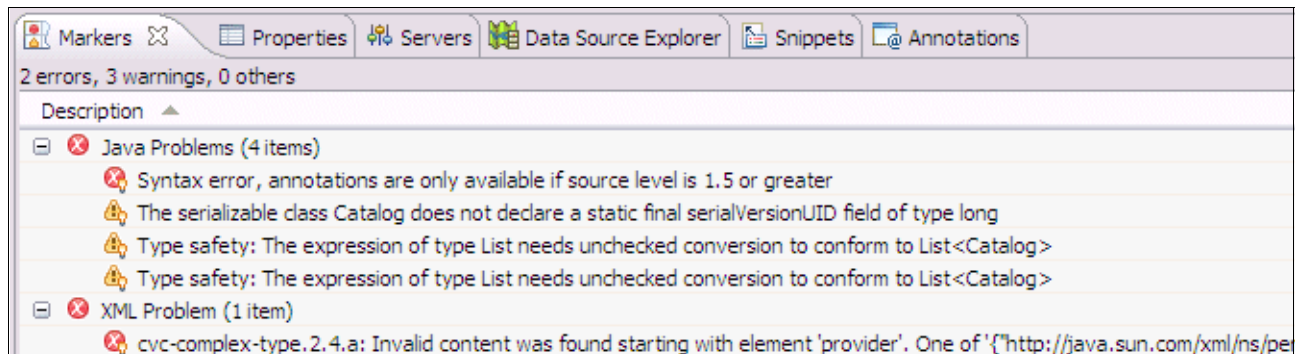


Figure 7-2 The remaining problems after the target run time is set

The first problem, the syntax error, indicates that the Java compiler compliance level for the project is still set to 1.4 and must be updated to 1.6. Check (and correct if necessary), the Java compiler compliance level on all three of the Java projects (ejb2, webapp1, and WebServices). To do so, complete the following steps:

1. Right-click **ejb2** and select **Properties**.

2. Select the Java Compiler properties in the left pane and change the Java compiler compliance level to 1.6 (Figure 7-3).

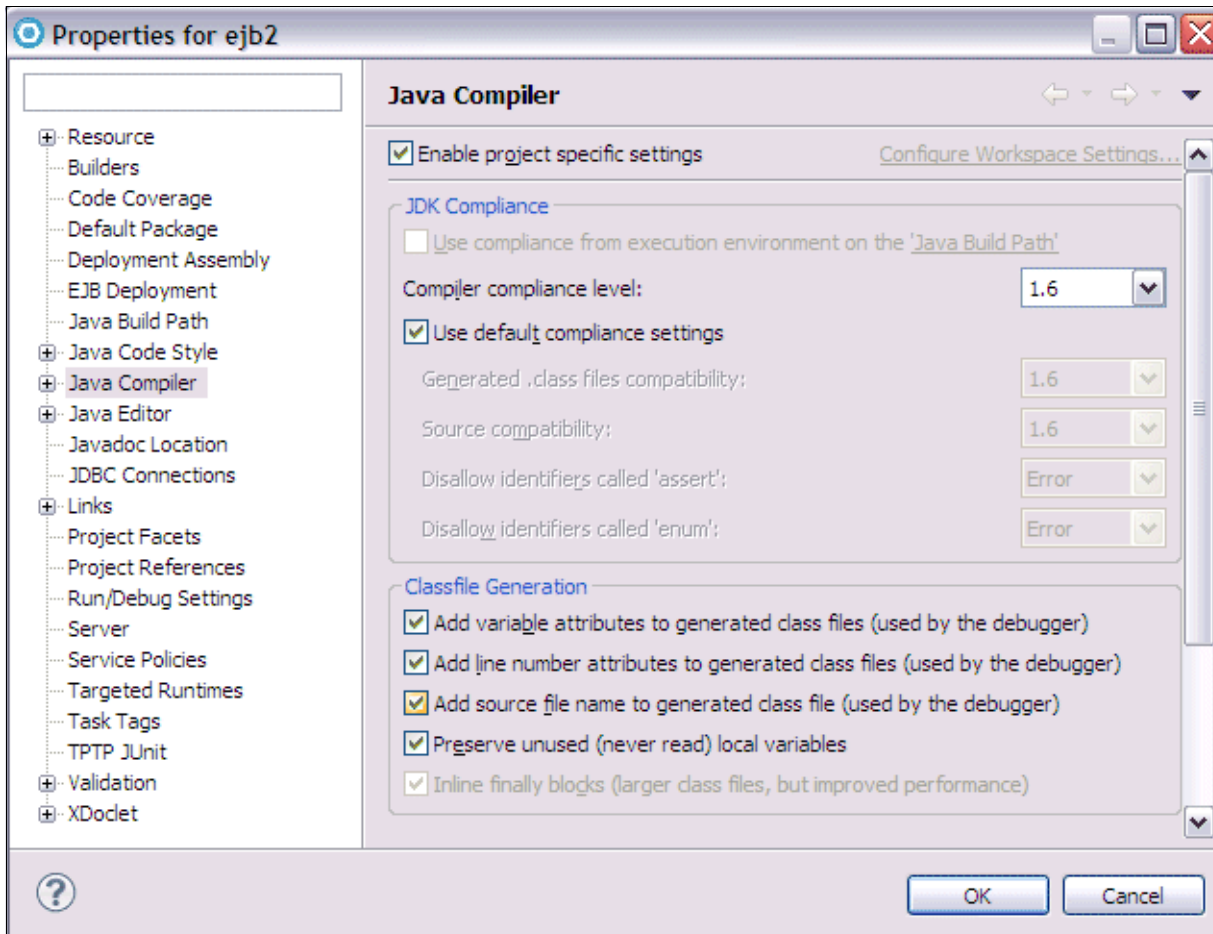


Figure 7-3 Setting the Java compliance level

3. The “Project Facets” properties also must be updated to indicate that you are using a Java 6 run time and that you are deploying to WebSphere Application Server V8.5. Click **Project Facets** in the left pane and change the version of the Java facet to 1.6. In addition, check the **WebSphere Web (co-existence)** and **WebSphere Web (Extended)** facets.
4. Repeat steps 1 on page 227 to 3 for the other two projects.
5. Click **OK** to accept the new properties and then **Yes** when Rational Application Developer prompts you to do a full project rebuild now.

**WebService project:** When we performed this task on the WebService project in our example, Rational Application Developer added an issue to the Markers view to indicate that, by default, WebSphere Application Server V7.0 and above does not scan for JAX-WS annotations without a change to the manifest.mf file. This problem is addressed in “Web Service problems” on page 229.

The remaining three Java Problems are all warnings, which you do not have to resolve now (they do not affect the running of the application).



### **Web Service problems**

When you update the WebService project to indicate that this project will be deployed on to WebSphere Application Server V8.5, Rational Application Developer added an issue that indicates that, by default, WebSphere Application Server V7.0 and above does not scan for JAX-WS annotations without a change to the manifest.mf file. You can use the following quick fix to update the manifest.mf file. To do so, complete the following steps:

- ▶ Right-click the issue in the Markers view and select **Quick Fix**.
- ▶ Click **Finish** to implement the quick fix.

### **XML problems**

The final problem is an XML problem in the application's persistence.xml file. This problem is one that the Oracle Application Server tolerates, while WebSphere Application Server V8.5 requires that the persistence.xml file conform to the schema defined for the persistence.xml file. You must correct persistence.xml to match the schema. The problem is that the persistence-unit stanza ends too soon and should include the provider stanza that immediately follows it. You must correct this XML so that the persistence-unit stanza includes the provider stanza (Example 7-1).

*Example 7-1 The corrected persistence.xml file*

---

```
<?xml version="1.0" encoding="windows-1252" ?>
<persistence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
             version="1.0" xmlns="http://java.sun.com/xml/ns/persistence">
  <persistence-unit name="Model">
    <provider>
      org.eclipse.persistence.jpa.PersistenceProvider
    </provider>
    <jta-data-source>
      java:/app/jdbc/jdbc/OracleDBConnectionDS
    </jta-data-source>
    <class>
      model.Catalog
    </class>
    <properties>
      <property name="eclipselink.target-server" value="Oracle_IAS"
/>
      <property name="javax.persistence.jtaDataSource"
value="java:/app/jdbc/jdbc/OracleDBConnectionDS" />
    </properties>
  </persistence-unit>
</persistence>
```

---

## **7.6.4 Running the Application Migration Tool**

You can use the Application Migration Tool to identify potential migration problems. To do so for our probability distribution sample, complete the following steps:

1. Right-click any of the projects and select **Software Analyzer** → **Software Analyzer Configurations....**
2. Click **New launch configuration**.

3. Click the **Rules** tab and then use the Rule Sets: drop-down menu to select the Oracle Application Migration rule set. Click **Set...**
4. In the Rule set configuration window, select the Target application server as **WebSphere Application Server V8.5**, as shown in Figure 7-4.

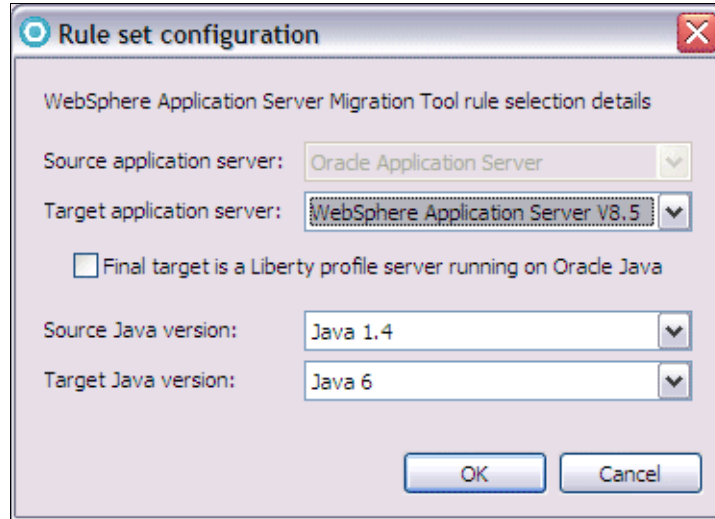


Figure 7-4 Rule set configuration for migrating from Oracle AS

5. Click **OK**.
6. Back in the Software Analyzer Configurations wizard, click **Analyze** to get the Application Migration Tool to analyze all the projects.

A total of four problems are reported: three XML problems and one Java problem.

### Fixing the Java Code Review result

As shown in Figure 7-5, the Java Code Review result is similar to one of the remaining Java problems that is still in the Markers view. You can use the quick fix for the Java problem to also resolve the Java Code Review result.

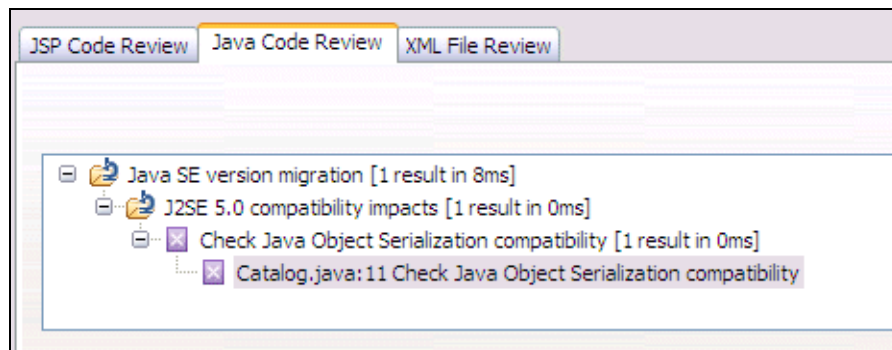


Figure 7-5 The Java Code Review results from the Application Migration Tool

Complete the following steps:

1. Double-click the Java Code Review result to open Catalog.java in the Java Editor View.

2. Click the light bulb/warning triangle to open a menu of possible fixes and click **Add generated serial version ID** to apply the quick fix (Figure 7-6).

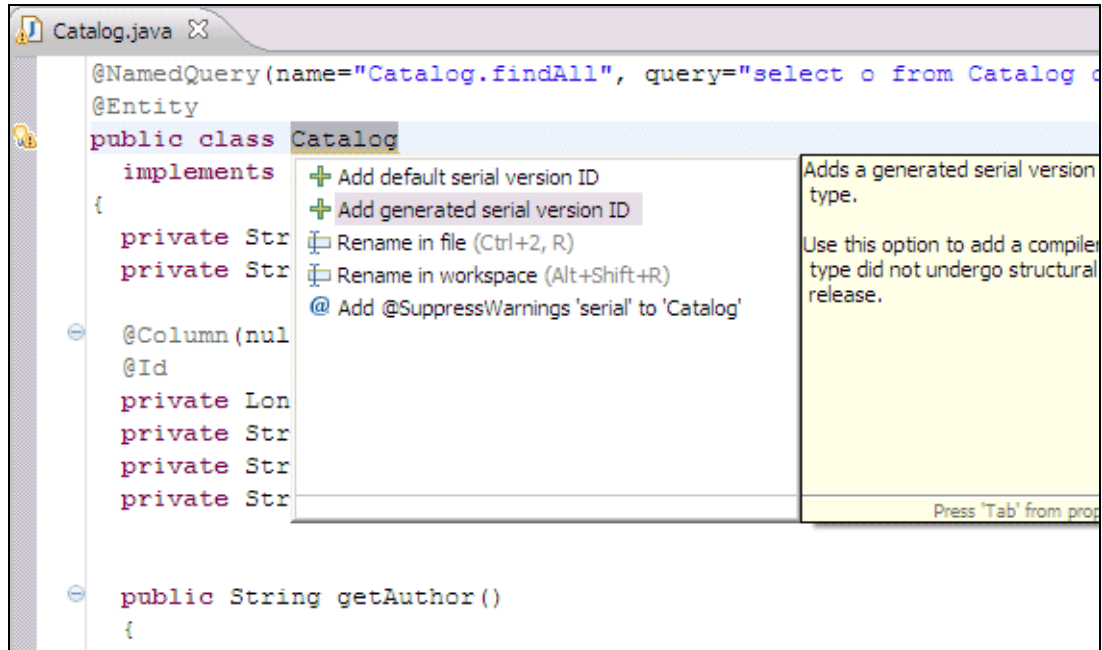


Figure 7-6 Applying a quick fix to resolve the Java serialization warning

## Fixing the XML File Review results

There are three results that are shown in the XML File Review results page (Figure 7-7).

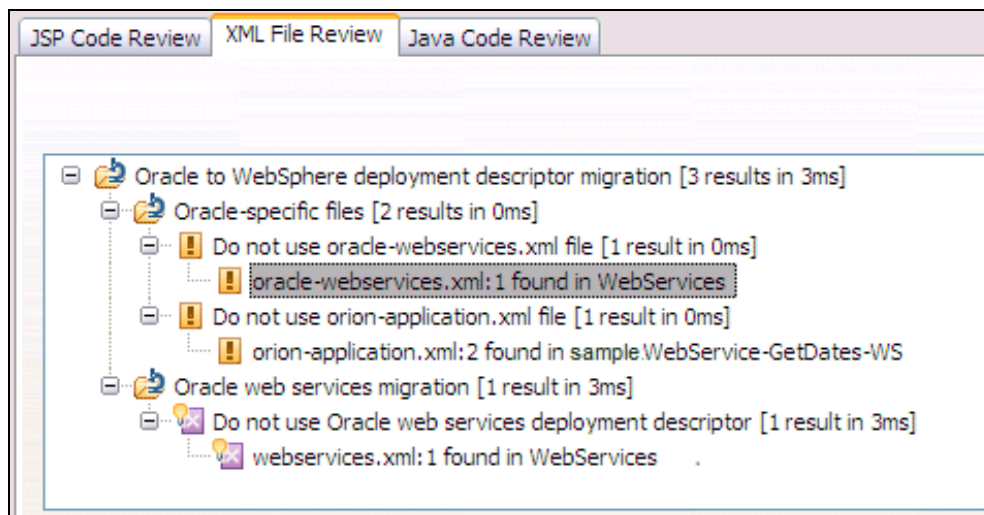


Figure 7-7 XML Code Review results

Fixing the first result is straightforward: Delete the file. WebSphere Application Server V8.5 does not need this file to determine the web services operations that must be made available through SOAP. Instead, WebSphere Application Server V8.5 makes all the operations available. The `oracle-webservices.xml` file can be in the `WEB-INF` directory of the `WebService` project.

The next code review result is for the usage of the `orion-application.xml`. When you double-click that result to open that file, you discover that it is empty apart from an empty `orion-application` stanza. This file can be deleted as well and can be found in the `META-INF` directory of the `SampleWebService-GetDates-WS` project.

To resolve the last remaining problem is more complicated, as you must convert the Oracle web service deployment descriptor into a WebSphere Application Server deployment descriptor. The Application Migration Tool can help you do this conversion. To do the conversion, complete the following steps:

1. Right-click `webservices.xml` and select the **Generate Ant script**. This action creates an Ant script (called `build-ibm-ws.xml`) that you must run to generate a new WSDL file to use with WebSphere Application Server V8.5 and to update `webservices.xml` for use on WebSphere Application Server V8.5.
2. To run the Ant script, open a command-line interface (CLI) and then run the command from within the Rational Application Developer workspace:
  - a. Open a CLI and navigate to the root directory of your WebSphere Application Server V8.5 installation.
  - b. Change to the `bin` directory.
  - c. Run the following command:

```
ws_ant.bat -f "$WORKSPACE/WebServices/build-ibm-ws.xml"  
genService_GetDatesWSSoapHttpPort
```

**Command syntax:** You must replace `$WORKSPACE` in the command with the location of your Rational Application Developer workspace. "WebServices" is the name of the project where `build-ibm-ws.xml` was created by the Application Migration Toolkit. `genService_GetDateWSSoapHttpPort` is the name of the target in that `build.xml`, which performs the relevant conversion for the `GetDateWS` web service. One of these targets is generated for each port that is listed in the original `webservices.xml` file, and you must repeat this process for each target generated.

3. After the Ant script completes successfully, return to Rational Application Developer, right-click the **WebService** project, and select **Refresh**.
4. You see that a new folder, `ibm-ws-gen`, is created by the Ant script. Find the `webservices.xml` file inside `ibm-ws-gen/src/WEB-INF`. As you have only the single web service, you can copy that file to the real `WEB-INF` directory.

**Multiple services:** If you have more than one service, you must copy the contents of the generated `webservices.xml` in to the real `webservices.xml`, overwriting each original web services as you copy over the changes.

## 7.6.5 Deploying the application in to WebSphere Application Server V8.5

Deploy the application into WebSphere Application Server by completing the following steps:

1. From the Server tab in Rational Application Developer, right-click **WebSphere Application Server v8.5 at localhost**.

2. Select **Add and Remove Projects** from the menu (Figure 7-8).

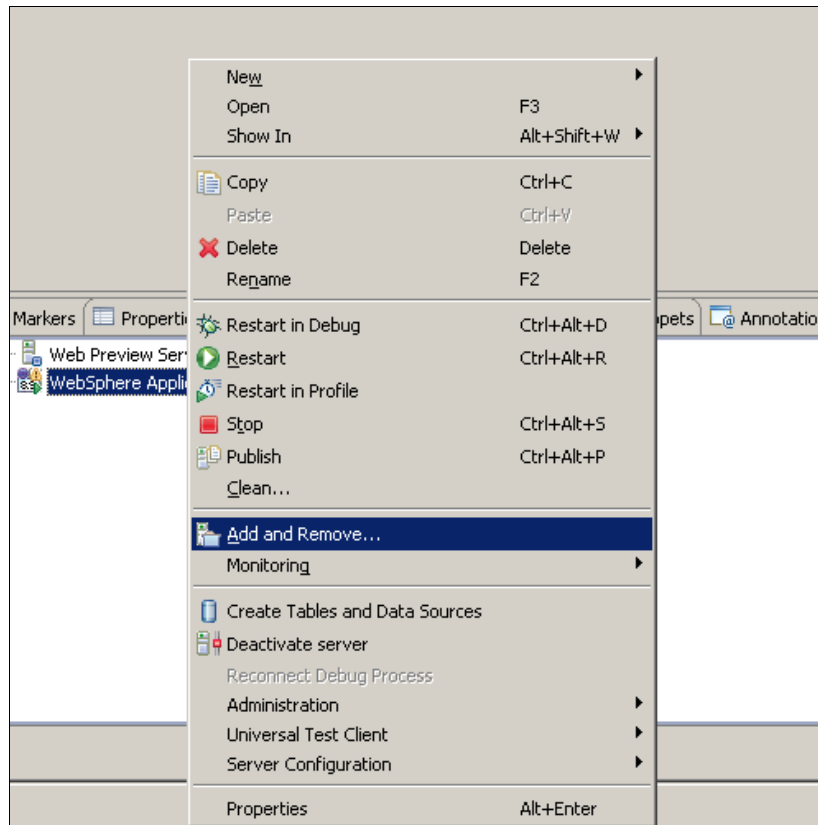


Figure 7-8 Add and Remove Projects

3. Move all resources to the right to configure them on the server and click **Finish**.
4. Right-click **WebSphere Application Server v8.5 at localhost** and select **Publish**.

In our sample application, you see that the “SampleEJBClient” failed to start. When you look at the logs, you find the error `java.lang.ClassNotFoundException: oracle.adf.share.logging.ADFLogger`. This class is included in the `adfshare.jar` file. The absence of this JAR file is expected, as `adfshare.jar` is deployed as a part of the ADF runtime libraries. You should add this JAR file in to the build path of the application.

5. After you add `adfshare.jar` in to the Web App Libraries of `webapp1` and publishing `SampleEJBClient` again, the `SampleEJBClient` starts successfully (Figure 7-9).

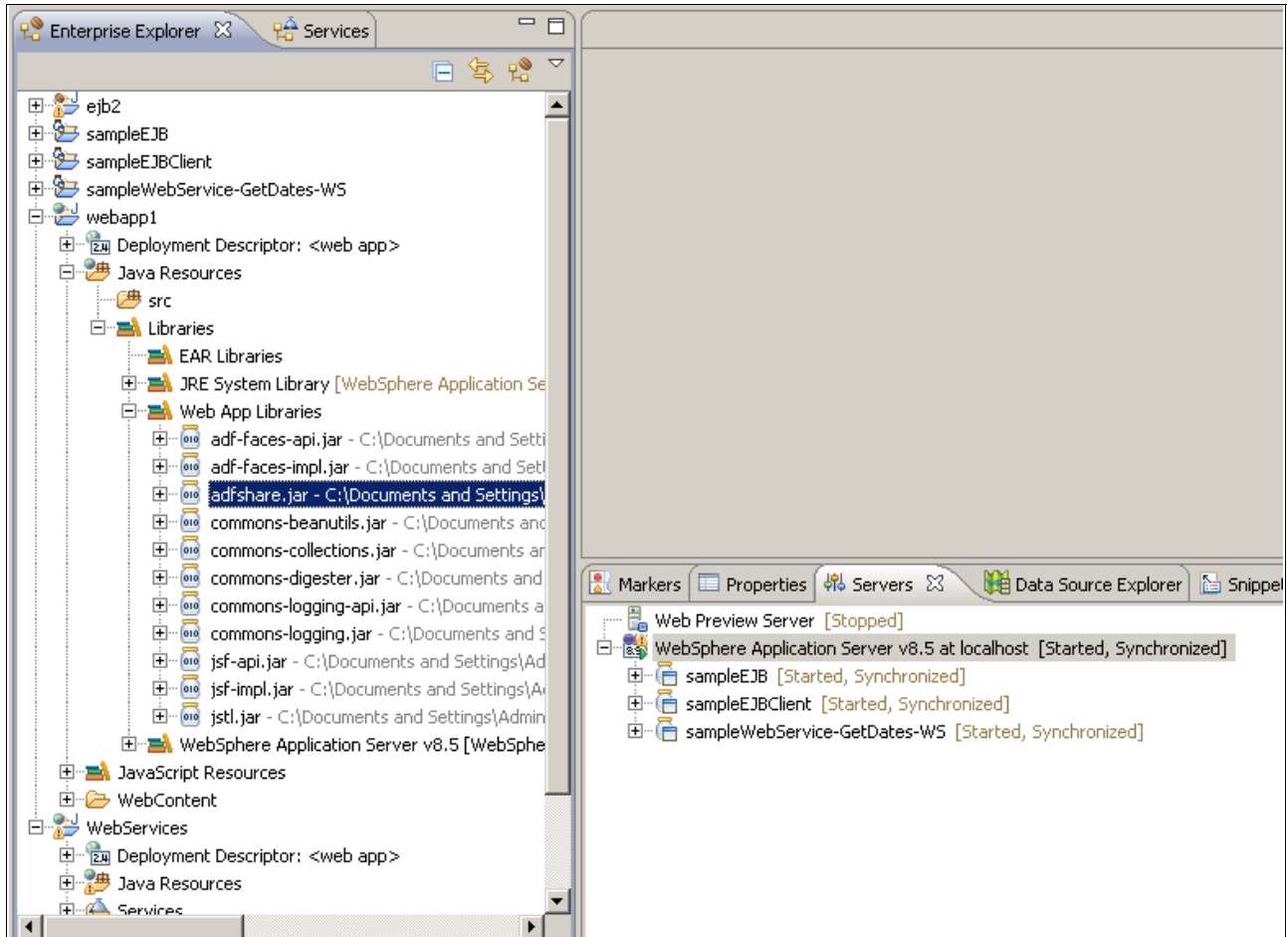


Figure 7-9 Adding missing JAR files

You can start and stop the server and application by using the WebSphere Application Server console. To reach the console, enter the URL `http://<hostname>:9080/ibm/console` or click **Start** → **Programs** → **IBM WebSphere** → **IBM WebSphere Application Server V8.5** → **Profiles** → **AppSrv01** → **Administrative console**.

The application that is imported from Rational Application Developer can be controlled in WebSphere Application Server. You can also import EAR files through this console.

## 7.6.6 Summary

In this chapter, we explained the process for migrating J2 EE applications from Oracle Application Server 10.1.2 to WebSphere Application Server V8.5 using a sample application that was created by the authors of this book. We used Application Migration Tool - Oracle AS to WebSphere to automate most of the migration steps. The tool was able to catch and fix the incompatibilities between Oracle Application Server and WebSphere Application Server in the code.



## Migrating from JBoss

This chapter describes the migration of Java Enterprise Edition applications from JBoss to WebSphere Application Server V8.5 using the following examples:

- ▶ The Kitchensink application from JBoss Quickstarts distribution.
- ▶ The online brokerage sample application from Geronimo project V1.0, which has JBoss specific features.

At the time of the writing of this book, we used the latest version of JBoss Application Server, 7.1.1 Final, which is certified for Java EE 6 full profile, for our example.

This chapter describes the following topics:

- ▶ Migrating from older versions
- ▶ Application Migration Tool - JBoss AS to WebSphere
- ▶ Preparing the environment
- ▶ Migrating the Kitchensink application of the JBoss Quickstarts distribution
- ▶ Migrating the Online Brokerage application

**Legal disclaimer:** IBM does not warrant or represent that the code provided is complete or up-to-date. IBM does not warrant, represent or imply reliability, serviceability, or function of the code. IBM is under no obligation to update content nor provide further support.

All code is provided "as is," with no warranties or guarantees whatsoever. IBM expressly disclaims to the fullest extent permitted by law all express, implied, statutory and other warranties, guarantees, or representations, including, without limitation, the warranties of merchantability, fitness for a particular purpose, and non-infringement of proprietary and intellectual property rights. You understand and agree that you use these materials, information, products, software, programs, and services, at your own discretion and risk and that you will be solely responsible for any damages that may result, including loss of data or damage to your computer system.

In no event will IBM be liable to any party for any direct, indirect, incidental, special, exemplary or consequential damages of any type whatsoever related to or arising from use of the code found herein, without limitation, any lost profits, business interruption, lost savings, loss of programs or other data, even if IBM is expressly advised of the possibility of such damages. This exclusion and waiver of liability applies to all causes of action, whether based on contract, warranty, tort or any other legal theories.

*Third-party* software is licensed and distributed to you by the *third-party* distributors and/or respective copyright and other right holders under their terms and conditions. IBM makes no express or implied warranties or representations with respect to such *software* and provides no indemnity for such software. IBM grants no express or implied patent or other license with respect to *and* is not liable for any damages arising out of the use of *such software*.

## 8.1 Migrating from older versions

*Migrating Applications from WebLogic, JBoss and Tomcat to WebSphere V6*, SG24-6690 describes how to migrate J2EE 1.3 applications from JBoss Application Server Version 3.2.7 to WebSphere Application Server V6. If your application is at specification levels J2EE 1.3 and J2EE 1.4, and you are using XDoclet for code generation, look at this version.

## 8.2 Application Migration Tool - JBoss AS to WebSphere

Application Migration Tool - JBoss AS to WebSphere (Application Migration Tool) is part of the Migration Toolkit, which is described in Chapter 4, "Installation and configuration of the Application Migration Tools" on page 89. Application Migration Tool can help you in the following areas when migrating applications from J2EE 1.4 and Java EE 5 applications from JBoss Server (starting with Version 4.x) to WebSphere Application Server V7, V8, or V8.5:

- ▶ Class path setup: Converts enterprise archive (EAR) class paths to J2EE specifications
- ▶ JBoss Server Startup Classes: Detect and convert JBoss start and shutdown MBeans
- ▶ Non-Portable JNDI lookups: Detect and refactor JNDI lookups to use the parameter-less InitialContext
- ▶ Deployment Descriptors: Detect and convert JBoss specific deployment descriptor elements to IBM deployment descriptor elements



- ▶ JBoss proprietary Java packages and files: Detect JBoss proprietary packages and files
- ▶ Web services: Migrate JBoss JAXRPC web services

## 8.2.1 Comparison of the JBoss V7.1.1 and WebSphere Application Server V8.5 terminology

Here are some of the similarities and differences in terminology between JBoss V7.1.1 and WebSphere Application Server V8.5. WebSphere Application Server and JBoss use different structures for managing distributed environments, which host many application server instances that work together as clusters to provide enterprise level quality of services, such as high availability and load distribution. A comparison in terminology can be found in Table 8-1.

Table 8-1 Terminology comparison

WebSphere Application Server V8.5	JBoss V7.1.1
Cell	Domain
Server Instance	Server or Node
Node	N/A
Node agent	N/A
Deployment Manager	Master (One of the server instances assumes this role.)
Cluster	Server Group

## 8.3 Preparing the environment

This section provides high-level steps and tasks for installing and configuring Apache Ant, Apache Maven, and JBoss Application Server.

### 8.3.1 Installing and configuring Apache Ant

Download the Apache Ant binary distribution from the Apache Ant website at:

<http://ant.apache.org/>

We used Version 1.8.3 on our environment.

You can find detailed installation instructions at:

<http://ant.apache.org/manual/install.html>

To install and configure Ant, complete the following steps:

1. Extract the downloaded file to a suitable location. Use a short path name, such as C:\Ant, as long file names might cause problems because of OS restrictions. We used C:\Ant as the installation directory in our example.

2. Set the environment variables by clicking **Start** → **Control Panel** → **System**. Click **Advanced** → **Environment Variables**. We want to use the same JRE as our integrated testing environment, so in our scenario, in the user variables section, we set the JAVA\_HOME variable as follows:
 

```
JAVA_HOME=C:\Program Files (x86)\IBM\SDP\jdk
```
3. Set the ANT\_HOME user variable to the folder that you specified in step 1 on page 237. In our case, we set it as follows:
 

```
ANT_HOME=C:\Ant
```
4. Add Java and Ant scripts to your path by appending the following code to your PATH user variable or creating one if it does not exist:
 

```
PATH=%PATH%;%JAVA_HOME%\bin;%ANT_HOME%\bin
```
5. Verify the environment variables and Ant installation by opening a new command prompt and checking the outputs of the following commands:
  - **java -version**
  - **ant -version**

The output shows the current versions of the tools.

### 8.3.2 Installing and configuring Apache Maven

Download the Apache Maven binary distribution from the Apache Maven website at:

<http://maven.apache.org/>

We used Version 3.0.4 in our environment.

You can find detailed installation instructions at:

<http://maven.apache.org/download.html>

To install and configure Apache Maven, complete the following steps:

1. Extract the downloaded file to a suitable location. In our scenario, we use C:\Maven as the installation directory.
2. Set the environment variables by clicking **Start** → **Control Panel** → **System**. Choose **Advanced** → **Environment Variables**. Set the M2\_HOME and M2 user variables to the folder that you specified in step 1. In our case, we set it as follows:
 

```
M2_HOME=C:\Maven
M2=%M2_HOME%\bin
```
3. Add a Maven script to your path by appending the following code to your PATH user variable or create one if it does not exist. At this stage, our PATH variable looks as follows:
 

```
PATH=%PATH%;%JAVA_HOME%\bin;%ANT_HOME%\bin;%M2%
```
4. Verify the Maven installation by opening a new command prompt and checking the output of the following command:
 

```
mvn -version
```

The output shows the current version of Maven.

### 8.3.3 Installing and configuring JBoss Application Server

This section provides the JBoss Application Server installation instructions.

## Installing JBoss Application Server Version 7.1.1

Download the JBoss Application Server distribution from the JBoss website. We use Version 5.1.0 in our environment.

<http://www.jboss.org/jbossas/downloads/>

You can find detailed installation instructions for JBoss Application Server at:

<https://docs.jboss.org/author/display/AS7/Getting+Started+Guide>

To install or configure the JBoss Application Server, complete the following steps:

1. Extract the downloaded file to a suitable location. We used `C:\jboss-as-7.1.1.Final` as the installation directory for our example.
2. Set the environment variables by clicking **Start** → **Control Panel** → **System**. Click **Advanced** → **Environment Variables**. Select **JBOSS\_HOME**. This location is referred to as `<jboss_home>` throughout the document. In our case, we set it as follows:

```
JBOSS_HOME=C:\jboss-as-7.1.1.Final
```

3. Add the JBoss script to your path by appending the following line to your PATH user variable. Our PATH variable looks like:

```
PATH=%PATH%;%JAVA_HOME%\bin;%ANT_HOME%\bin;%M2%;%JBOSS_HOME%\bin
```

4. Verify the installation by starting the server by running the `standalone.bat` command.

## 8.4 Migrating the Kitchensink application of the JBoss Quickstarts distribution

The Kitchensink application is a part of the JBoss Quickstarts distribution, which is a collection of sample applications that demonstrate how to use new features of the product. This collection can be downloaded separately from the JBoss website. More information about this application can be found at:

<https://docs.jboss.org/author/display/AS7/Kitchensink+quickstart>

Kitchensink is a simple application that demonstrates the new features of Java EE 6. We choose to migrate this application for the following reasons:

- ▶ It uses JPA 2.0 (Hibernate as the JPA provider) and EJB 3.1.
- ▶ It uses JSF 2.0, CDI 1.0, and Bean Validation 1.0 (Hibernate Validator as the validation provider).
- ▶ It uses JAX-RS (a new Java API for RESTful Web Services).
- ▶ It is simple, which is ideal for demonstrating the basic features of a migration. We can demonstrate project scope changes without getting lost in the repetitive work that large projects might require for each module or component.

### 8.4.1 Migration approach

The Kitchensink application was built with non IBM development tools: Ant and Maven. In this particular migration exercise, we also migrate the build environment to IBM Rational Application Developer for WebSphere Software, and use the build cycle that ships with this product.

To complete the migration, complete the following steps:

1. Verify that the application works in its original destination environment. In our case, we build, deploy, and run the application on JBoss Application Server 7.1.1 Final.
2. Import the EAR file that is built for JBoss to IBM Rational Application Developer for WebSphere Software to create the project structure. Import the source files in to projects created from the source distribution.
3. Check the source code and the configuration files with the Application Migration Tool.
4. Analyze and fix problems that are reported by the build environment and Application Migration Tool.
5. Build the Kitchensink sample application using IBM Rational Application Developer for WebSphere Software.
6. Migrate the Hibernate JPA provider to OpenJPA.
7. Deploy and test the application on an integrated WebSphere Application Server V8.5 test environment.

## 8.4.2 Verifying the Kitchensink sample application

The Kitchensink application does not require any additional configuration on the target JBoss application server. The application uses the default data source that is preconfigured on the application server, and uses default JPA and Bean validation providers.

To verify the Kitchensink sample application, complete the following steps:

1. Download the Quickstarts distribution from the following web page:  
<http://download.jboss.org/jbossas/7.1/jboss-as-7.1.1.CR2/jboss-as-quickstarts-7.1.1.CR2-dist.zip>
2. Extract the downloaded archive file to a suitable location. In our case, we use C:\quickstarts. Hereafter, this location is referred to as <quickstarts\_home>.
3. Open a command prompt and change to the directory that contains the sample application code:  
`<quickstarts_home>\kitchensink`
4. Build and deploy the sample application by running `mvn clean install` in the <quickstarts\_home>\kitchensink folder.  
The build script terminates successfully with the message BUILD SUCCESSFUL.
5. Open another command prompt and start your JBoss Application Server Version 4.2.3 by running `<jboss_home>/bin/standalone.bat`. Maven builds the project and output, including the WAR archive, which is generated in the <quickstarts\_home>\kitchensink\target folder.
6. Copy the generated `jboss-as-kitchensink.war` file in the <quickstarts\_home>\kitchensink\target folder to the <jboss\_home>\standalone\deployments folder.
7. Point your browser to the following address and test the sample application by generating a couple of contact items, using REST endpoint links to see JSON data you enter. Notice how form validation is handled by using a bad input string (that is, a malformed email address of phone number).

`https://localhost:8080/jboss-as-kitchensink/`

### 8.4.3 Importing the EAR file and source code to Rational Application Developer

To import the EAR file and source code to Rational Application Developer, complete the following steps:

1. Start Rational Application Developer by clicking **Start** → **IBM Software Delivery Platform** → **IBM Rational Application Developer 8.5** → **Rational Application Developer**.
2. Start the import wizard by clicking **File** → **Import**. Expand the web node in the import source tree and select the WAR file.
3. In the WAR window, browse to the location of the WAR file that is generated by the Ant build, which might be found at the following location:

<quickstarts\_home>\kitchensink\target\jboss-as-kitchensink.war

See Figure 8-1. Click **Next** to continue.

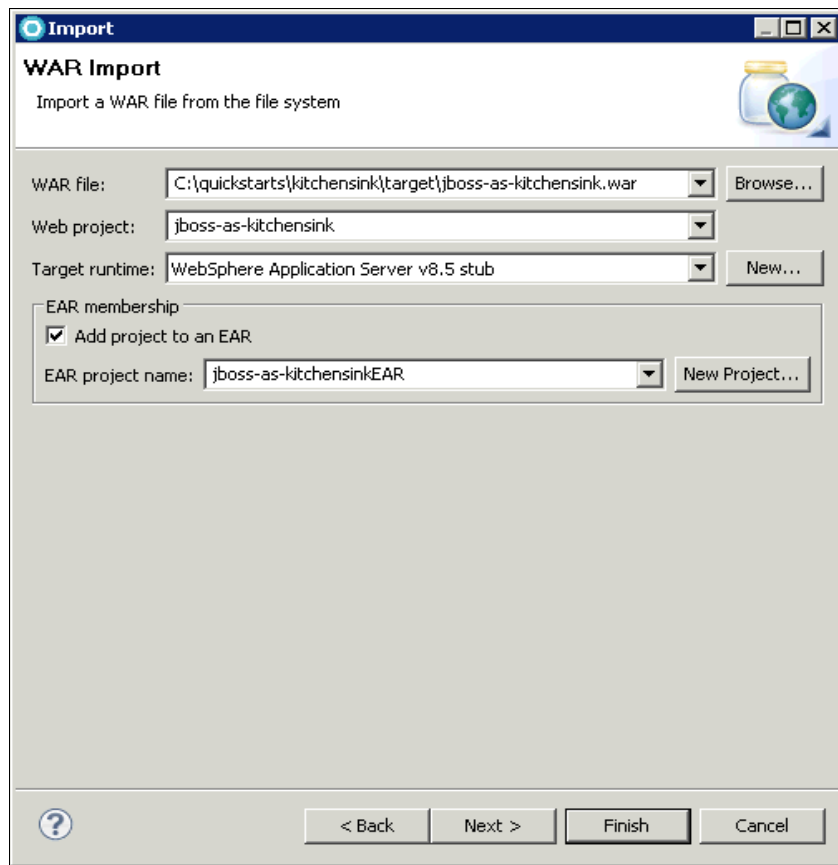


Figure 8-1 Importing the WAR file for Kitchensink application using the import wizard

4. On the WAR Import: Web libraries window, click **Finish** to continue without selecting any of the JAR files to be imported as a utility project.

5. Click **Finish** on the WAR Import: Web libraries window. The Import wizard runs and creates the necessary project and module structure (Figure 8-2).

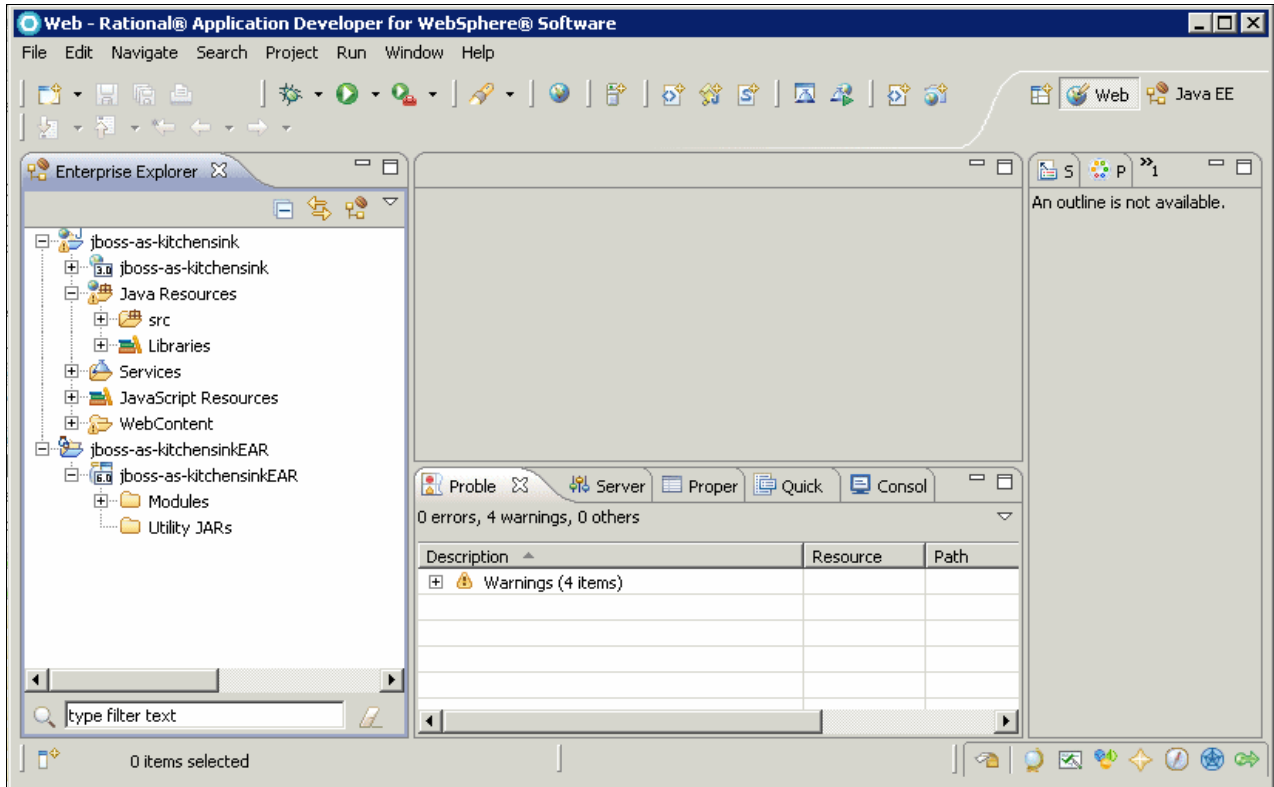


Figure 8-2 Project structure after the initial import

6. If you are not already in the Web perspective, a prompt asks if you want to switch to the Web perspective. Click **Yes**.

- Expand **jboss-as-kitchensink** → **WebContent** → **WEB-INF** → **lib** and delete all the JAR files in this node (Figure 8-3).

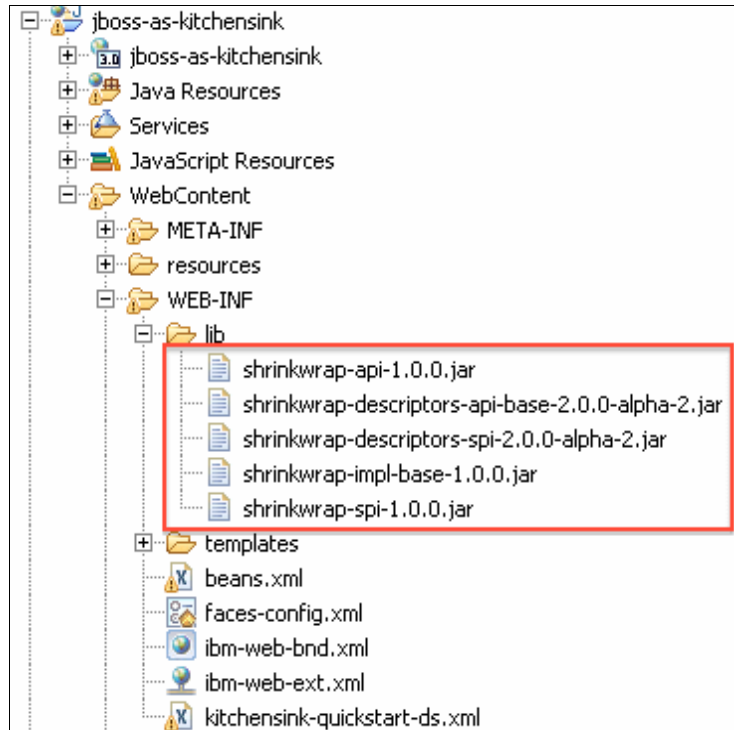


Figure 8-3 Removing the shirinkwrap libraries from the project

**Shrinkwrap:** Shrinkwrap is a packaging library for Java. We do not depend on Shrinkwrap libraries in the run time, so we can remove them from the project. More information about this library can be found at:

<http://www.jboss.org/shrinkwrap>

- Import the source code for the web module. As shown in Figure 8-4, you must replace classes that are imported in compiled form with their respective source codes. Click **Java Resources/src** under the web module, and then click **File** → **Import** from the main menu.

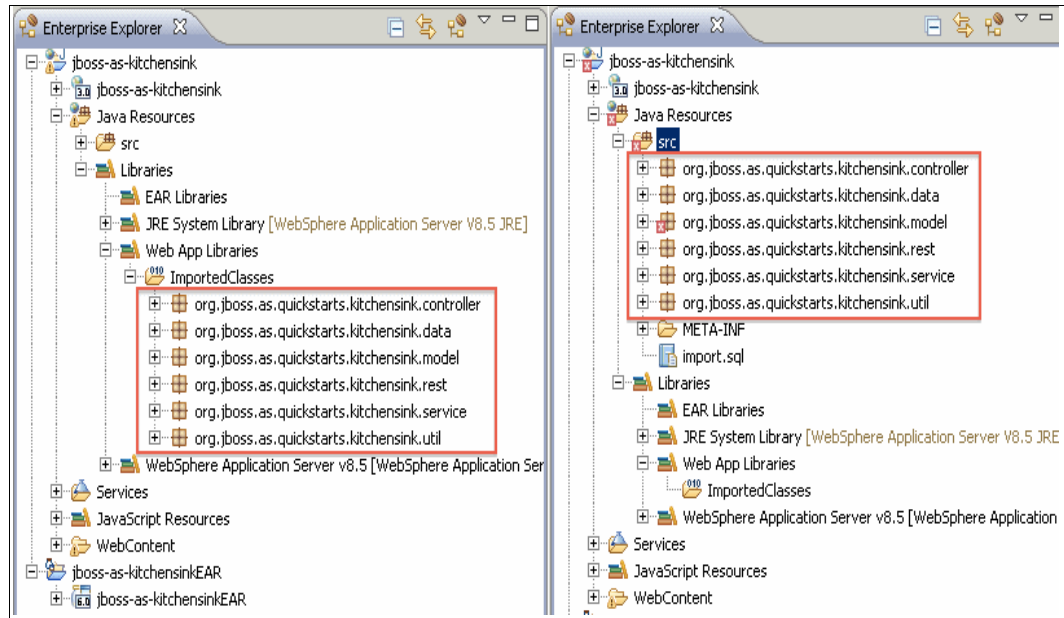


Figure 8-4 Replacing the class file with the respective source files in the web module

- On the Import window, clicking **General** → **File System** displays the File System Import window. Click **Browse** and select the folder that contains the source code of our project:  
`<quickstarts_home>\kitchensink\src\main\java`
- Select the Java node. Click **Finish**.

**Importing source files:** When you import source files to replace classes imported by the import wizard, the following rules apply:

- ▶ In an EJB module, source files must be added under the `ejbModule` node.
- ▶ In a web module, source files must be added under the `Java Resources/src` node.

- Because we imported the source files, we can now remove the compiled class files that are imported by the import wizard for the same classes. Expand **Web App Libraries** → **Imported Classes**. Select all packages in this node. Click **File** → **Delete** from the main menu. Click **OK** on the deletion confirmation dialog box that opens. The web module should now look like the right pane Figure 8-4.

## 8.4.4 Analyzing and fixing migration problems

The following subsections describe the steps that we performed for analyzing and fixing the problems.



## Fixing the web module dependencies

As seen in Figure 8-4 on page 244, there is a compile-time problem in the imported `org.jboss.as.quickstarts.kitchensink.model` package in the `Member.java` file. This problem exists because we have a dependency on the Hibernate Validator library, because the source code uses Hibernate specific bean validation annotations. To fix this issue, complete the following steps:

1. Download the Hibernate Validator distribution from the following URL and extract it to a suitable location. At the time of the writing of this book, we used the latest version of Hibernate Validator, which was 4.2.0.Final.  
<http://www.hibernate.org/subprojects/validator/download>
2. Click the web module and expand the node **Web Content** → **WEB-INF** → **lib**. Click **File** → **Import** to start the Import wizard.
3. Click **File** → **General** and select **hibernate-validator-4.2.0.Final.jar**. Click **OK** to import the library.
4. Click **OK** to continue. As the workspace is rebuilt, the compile error disappears.

## Defining Faces Servlet mapping

Because of the notation that is used in the Kitchensink application, we must define a servlet mapping for Faces Servlet by mapping to the `*.jsf` resources. To do so, complete the following steps:

1. In the Enterprise Explorer view, expand the web module node. Right-click the **Servlets** node and select **New** → **Servlet**.
2. In the new Servlet wizard, select the **Use and existing servlet class or JSP** check box. Click **Browse**, and select **javax.faces.webapp.FacesServlet** class. See Figure 8-7 on page 247. Click **Next**.

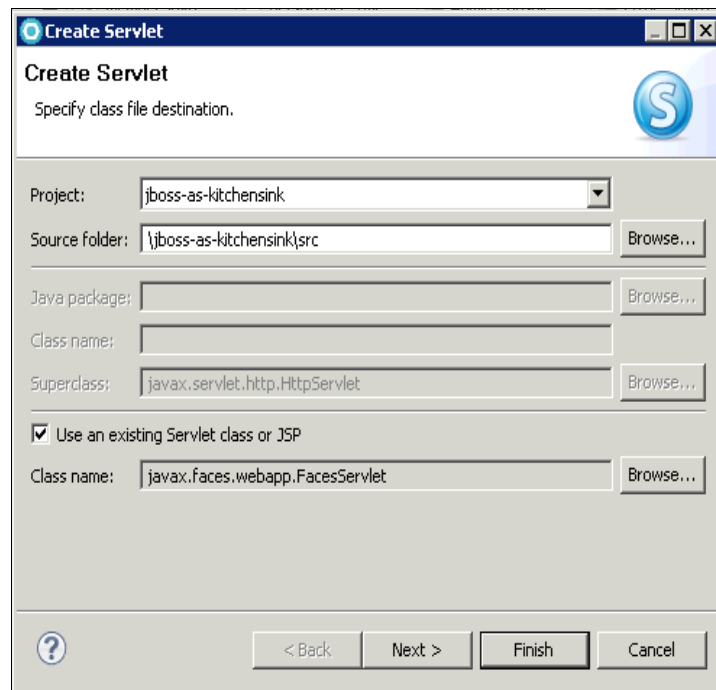


Figure 8-5 Selecting the Faces Servlet

3. On the next page, click **Add...** in the URL Mapping section, and add the pattern \*.jsf. See Figure 8-6. Click **Finish**.

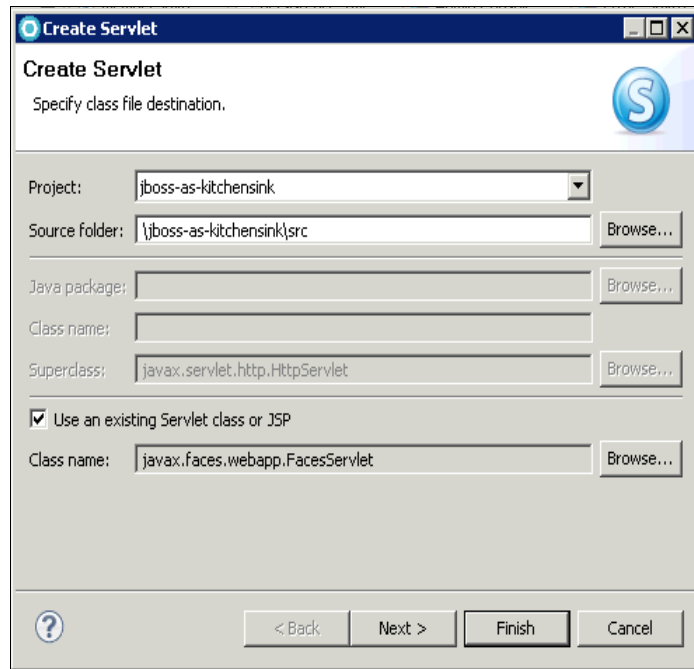


Figure 8-6 Adding a servlet mapping to the Faces Servlet

4. You should now see the new mapping in the Enterprise Explorer window.

### 8.4.5 Checking the project using Application Migration Tool

Even if you are working on a simple application, checking the source code and configuration files is a good idea. To run the Application Migration Tool on your project, complete the following steps:

1. Click the **jboss-as-kitchensinkEAR** application module. Right-click and select **Software Analyzer** → **Software Analyzer Configurations**.
2. In the left pane of the Software Analyzer Configurations window, right-click **Software Analyzer** and select **New**.
3. Click the **Rules** tab. Select **JBoss Application Migration** from the drop-down menu.

- Click **Set ...** to select the JBoss rules in the Analysis Rules and Domains area. Set the Target Application Server to **WebSphere Application Server V8.5** and the Source Java version to **Java 6**. Click **OK**. See Figure 8-7.

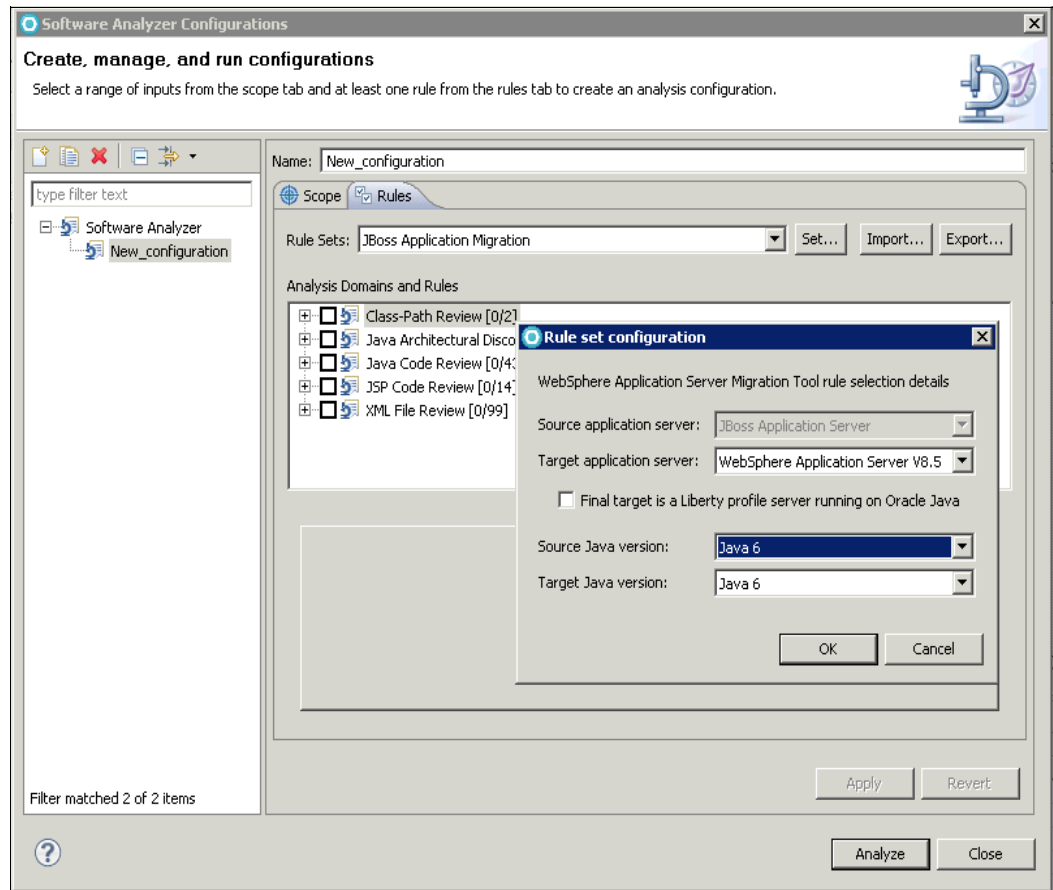


Figure 8-7 Software Analyzer configuration with JBoss migration rule set

- Click **Analyze**. After the analysis is complete, the Software Analyzer Results view is displayed (attached to the bottom pane).

In the Software Analyzer Results view, you can see a tab for each domain of analysis. In our case, we see a tab for ClassPath Review, Java Code Review, XML File Review, and JSP Code Review.

In Figure 8-8, Application Migration Tool warns us about the package naming convention and usage of the Hibernate framework. In following sections, we migrate Hibernate JPA provider to OpenJPA, which is provided with the WebSphere Application Server v8.5.

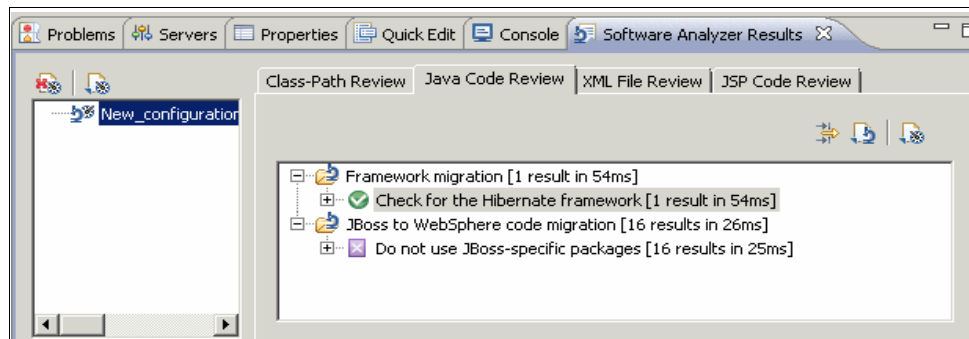


Figure 8-8 Application Migration Tool XML file review

## 8.4.6 Managing more runtime dependencies

In this section, we describe the steps that are taken to manage runtime dependencies.

### Fixing runtime dependencies

WebSphere Application Server V8.5 does not contain certain libraries that are required to run the sample application. These libraries are basically logging and hibernate libraries for persistence, and their dependencies. These libraries are included in the JBoss Application Server distribution, so we can import these libraries into our application from the JBoss Application Server installation.

There are several approaches about how to include these libraries, including packaging within the application archive file or using shared libraries. For our sample application, because we do not use Hibernate as the JPA provider, but migrate it to OpenJPA, we need only logging libraries. Because we need only a couple of libraries, we can add them to Web Module class path by importing them in to the `WEB-INF/lib` folder of web module, as we did for Hibernate-Validator.

Complete the following steps:

1. Download the SLF4J distribution from the following URL and extract it to a suitable location. At the time of the writing of this book, we use the latest version of SLF4J, which is 1.6.4.  
<http://www.slf4j.org/download.html>
2. Click the web module and expand the node **Web Content** → **WEB-INF** → **lib**. Click **File** → **Import** to start the Import wizard.
3. Click **File** → **General**, select **slf4j-api-1.6.4.jar** and **slf4j-simple-1.6.4.jar**, and click **OK** to import the libraries.
4. Click **OK** to continue.

The web module now looks like Figure 8-9.

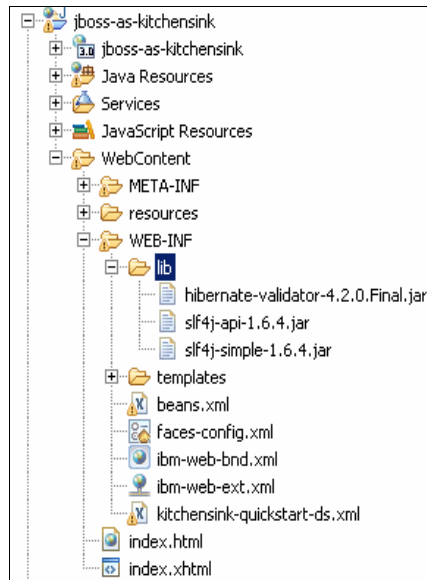


Figure 8-9 Web module after you import more dependencies

**Including libraries:** Because injected code like Hibernate Validator depends on SLF4J, not including these libraries might cause problems with the injected resources, and it might be difficult to determine that the root cause of the problem is the missing libraries.

## 8.4.7 Migrating to the WebSphere built-in JPA provider and the default data source

Two persistence providers are included in WebSphere Application Server:

- ▶ JPA for WebSphere Application Server persistence provider
- ▶ Apache OpenJPA persistence provider

The JPA for WebSphere Application Server persistence provider is the default provider for WebSphere Application Server. You can use one of the two providers, or a third-party persistence provider, as the default provider. In this exercise, we migrate our application to use OpenJPA JPA provider, which is included in WebSphere Application Server V8, instead of Hibernate JPA provider, which is included in JBoss Application Server.

Complete the following steps:

1. Click the **jboss-as-kitchensink** web module. Open the `persistence.xml` file in an editor by double-clicking it at `Java Resources/src/META_INF/persistence.xml`.
2. To use the WebSphere Application Server V8.5 default data source that is based on Derby, change the `jta-data-source` element value from `<jta-data-source>java:jboss/datasources/KitchensinkQuickstartDS</jta-data-source>` to `<jta-data-source>DefaultDataSource</jta-data-source>`.

- Remove all property elements and add the following property elements to the properties element. These elements set the schema to use and create and synchronize the schema as defined in the persistence units and database:

```
<property name="openjpa.jdbc.SynchronizeMappings"
value="buildSchema(ForeignKeys=true)" />
```

```
<property name="openjpa.jdbc.Schema" value="APP"/>
```

- Save the persistence.xml file by clicking **File** → **Save**.

To use JPA-related views, complete the following steps to activate the JPA project facet:

- Right-click the web module node and select **Properties**.
- In the Properties dialog box, select **Project Facets**, and select the **JPA** check box (Figure 8-10).

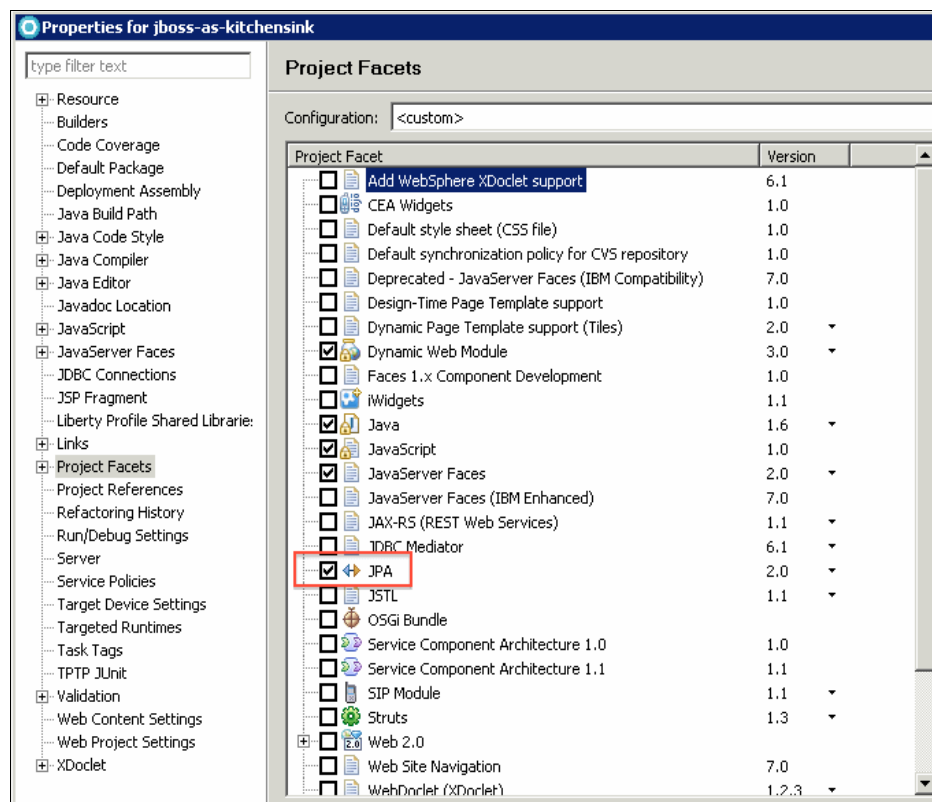


Figure 8-10 Enabling the JPA Project Facet

- Click **Apply** and then **OK**.

You can see the JPA node, which also lists the persistence units and entities in Enterprise Explorer. See Figure 8-11.

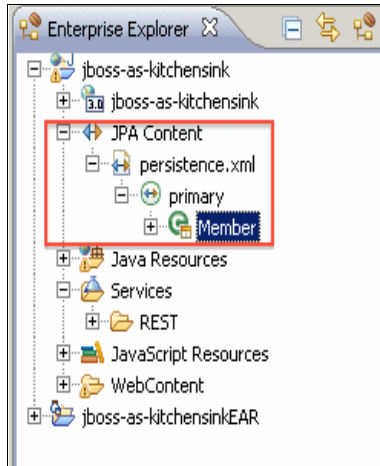


Figure 8-11 Enterprise Explorer after you enable the JPA project facet

The validation error Member.java, stating “class Member is mapped, but is not included in any persistence units” occurs. Because you know that the Member entity is defined in JPA context, you can ignore this exception. To do so, complete the following steps:

1. Click **Window** → **Preferences**.
2. Click **Java Persistence** → **Errors/Warnings** from left pane, and change the type from error to warning for your specific problem (Figure 8-12).

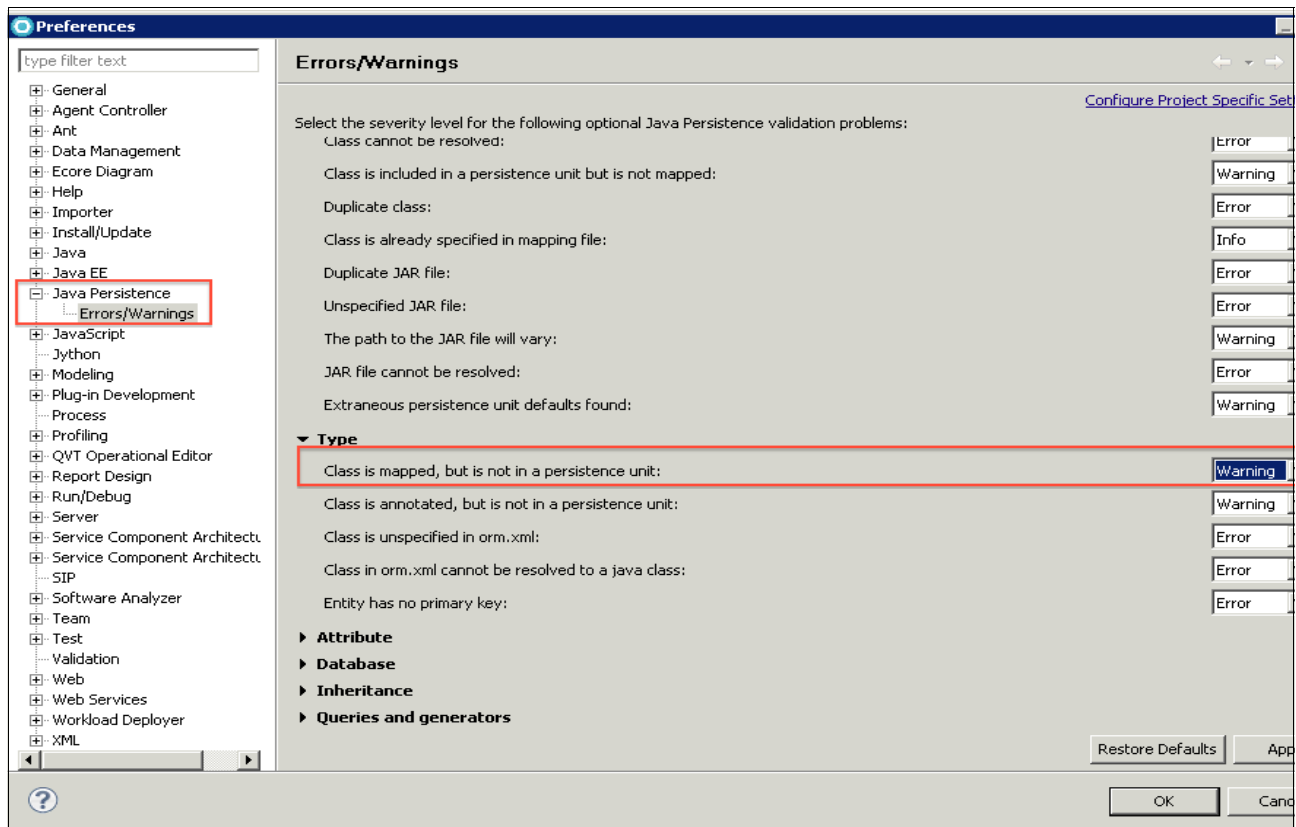


Figure 8-12 Configuring JPA validation warnings

3. Click **Apply** and then **OK**.

## Enabling JPA 2.0 typesafe queries

At this point, the persistence infrastructure works as expected, but you may notice the warning that is shown in Example 8-1 in the application server logs.

### Example 8-1 Warning message

```
CWWJP9991W: openjpa.Metadata: Warn: Meta class
"org.jboss.as.quickstarts.kitchensink.model.Member_" for entity class
org.jboss.as.quickstarts.kitchensink.model.Member can not be registered with
following exception "java.security.PrivilegedActionException:
java.lang.ClassNotFoundException:
org.jboss.as.quickstarts.kitchensink.model.Member_"
```

This error occurs because the compiler did not generate typesafe query classes by processing annotations.

To enable this feature, complete the following steps:

1. Right-click the web module project and select **Properties**. Click **Java Compiler** → **Annotation Processing** → **Factory Path**.
2. On the Factory Path window, click **Add External Jar....** Browse to C:\Program Files (x86)\IBM\WebSphere\AppServer\runtimes\com.ibm.ws.jpa.thinclient\_8.5.0.jar. Add the file. See Figure 8-13.

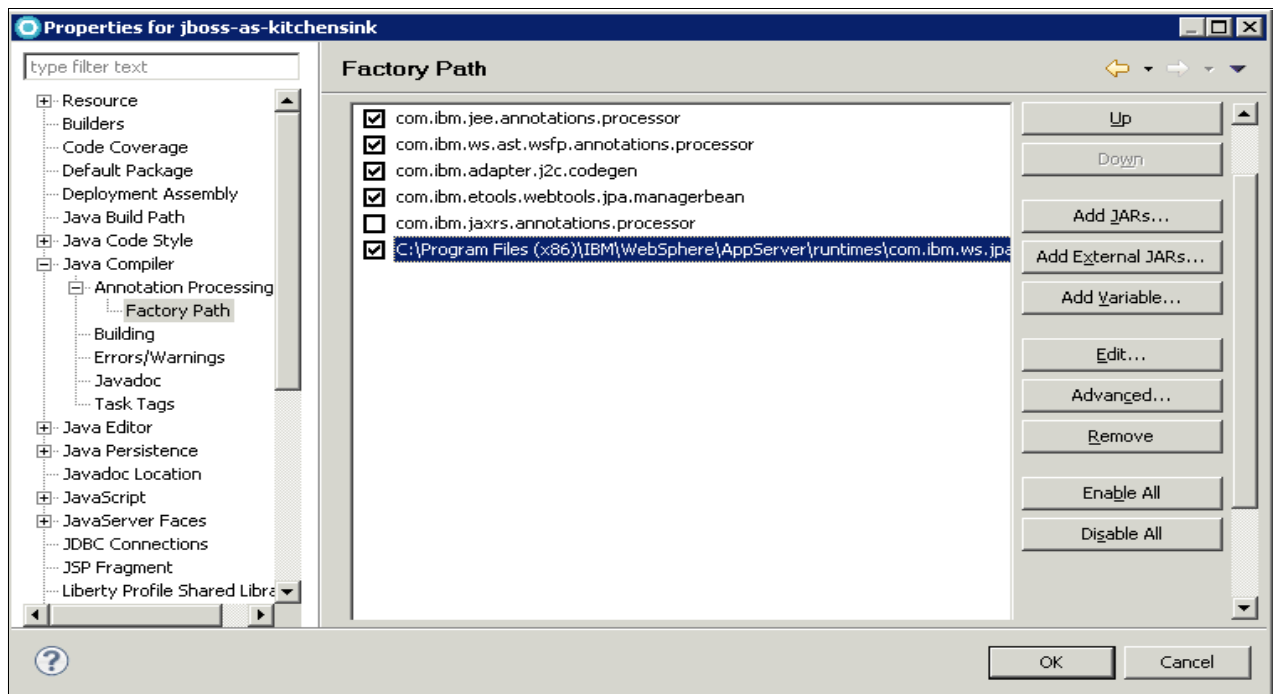


Figure 8-13 Adding factory class path for annotation processing at compile time

3. Click **Java Compiler** → **Annotation Processing**. Click **New...** and add the following property:  
`openjpa.metamodel=true`



See Figure 8-14.

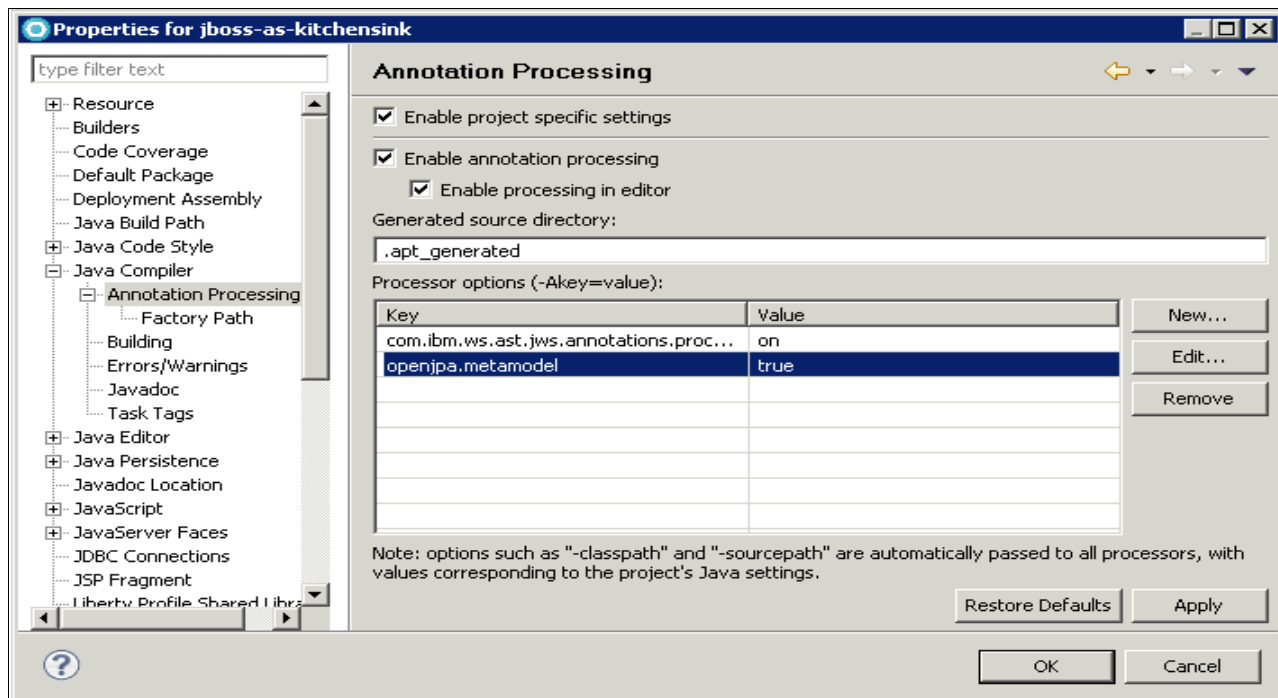


Figure 8-14 Enabling the metamodel generation

A metamodel class describes the meta information of a persistent class. A canonical metamodel class is static in the sense that all its member variables are declared static (and public). The `org.jboss.as.quickstarts.kitchensink.model.Member_name` is one such static member variable. The `openjpa.metamodel=true` parameter instructs the annotation processing engine to generate these metamodel classes (`Member_`) given entities (`Member`).

**Note:** For more information about using typesafe queries on JPA 2.0, see: <http://www.ibm.com/developerworks/java/library/j-typesafejpa/>

4. Click **Apply** and then **OK** to save your changes. Upon rebuild, the warning disappears.

## 8.4.8 Building and running an application on an integrated test environment

Migration is complete and the application can be run on WebSphere Application Server V8.5. You can check how the migrated application works by using the integrated WebSphere Application Server V8.5 test environment.

To build, deploy, and run the application, complete the following steps:

1. Click the `jboss-as-kitchensink\WebContent\index.html` file. Right-click and select **Run As** → **Run** on the server. The Run on server dialog box is displayed. Keep the default settings, and click **Finish**.

2. After the application is deployed to the test environment, an embedded web browser is displayed in the main docking area, presenting the entry page of the Kitchensink application. Follow the links to test the application functionality. Things to check are form validations and JAX-RS endpoints, which can be used for both query and creation purposes.

## 8.4.9 Summary

While we migrate the Kitchensink Quickstart sample application, we demonstrated the following concepts:

- ▶ Source migration to Rational Application Developer for WebSphere Software as a build and test environment
- ▶ Use of Application Migration Tool
- ▶ Migrating from third-party JPA providers to WebSphere Open JPA built-in provider

The main points to highlight during this migration are as follows:

- ▶ Libraries that ship with JBoss Application Server distributions and not with WebSphere Application Server create more runtime dependencies that must be taken care of.
- ▶ Different built-in data sources that are used in this example made it necessary to modify the persistence settings.
- ▶ JAX-RS did not need configuration changes and worked as expected.

## 8.5 Migrating the Online Brokerage application

Online Brokerage is the sample application from the Apache Geronimo project. This application is used to demonstrate migration from JBoss Application Server to Geronimo Application Server. Online Brokerage includes JBoss specific features. The version that we choose for our example is the one that is used in the Geronimo V1.0 documentation. This version contains the setup that is common among many JBoss users, that is, J2EE 1.4, Hibernate (using JBoss specific HAR files), and JBoss Application Server V4.

You can download and discover more information about this application at:

<https://cwiki.apache.org/GMOxDOC10/jboss-to-geronimo-hibernate-migration.html#JBoss toGeronimo-HibernateMigration-sampleApp>

We chose to migrate this application for the following reasons:

- ▶ It uses the J2EE 1.4 application, which uses JSP and Hibernate.
- ▶ It uses the JBoss-specific HAR archives for Hibernate.
- ▶ This configuration still has a significant deployment base among JBoss users.

### 8.5.1 Migration approach

The Online Brokerage application is built using Apache Ant. In this particular migration exercise, we migrate the build environment to IBM Rational Application Developer for WebSphere Software, and use the build cycle that ships with this product. The migration is completed by completing the following steps:

1. Build an EAR file for JBoss Application Server. Eliminate Geronimo dependencies in the build process to build the application without installing Geronimo Application Server.

2. Create a database for the application using DB2 and populate the database with data.
3. Import the EAR file that is built for JBoss to IBM Rational Application Developer for WebSphere Software, to create the project structure. Import source files into projects that are created from the source distribution.
4. Check the source code and configuration files with the Application Migration Tool.
5. Analyze and fix problems that are reported by the build environment.
6. Build the Online Brokerage sample application using IBM Rational Application Developer for WebSphere Software. Deploy and test the application on the integrated WebSphere Application Server V8.5 test environment.

## 8.5.2 Building the Online Brokerage application

To build the Online Brokerage application archive file, complete the following steps:

1. Download the source distribution of the application from the following website:  
<https://cwiki.apache.org/GMOxDOC10/jboss-to-geronimo-hibernate-migration.data/brokerage.zip>
1. Extract the downloaded archive file to a suitable location. We choose C:\brokerage. We refer to this location as <brokerage\_home>.
2. Open the <brokerage\_home>/build.properties file in an editor. Replace the line `jboss.home=E:/jboss-4.0.3SP1/server/hibernate` with the line `jboss.home=<jbossV7.1.1_home>/server/default`.

**Important:** We are reusing the JBoss version 7.1.1 Final installation that we use in 8.4, “Migrating the Kitchensink application of the JBoss Quickstarts distribution” on page 239. If you skipped that exercise, refer to that section. Also, ensure that you use forward slashes as path separators when you edit build.properties.

3. In the build.properties file, replace the line `dependency.dir=E:/hibernate-3.1/lib` with the line `dependency.dir=<jbossV7.1.1_home>/server/default/lib`. Save your changes.
4. Open the <brokerage\_home>/build.xml file in an editor. Add the following line to the javac element of the jboss-compile Ant target, among other class path elements:  
`<classpath path="{jboss.home}/lib/servlet-api.jar"/>`  
 Save your changes.
5. Open a command prompt, change the directory to <brokerage\_home>, and run **ant**.
6. After the Ant build completes, the Online Brokerage application archive files are generated at <brokerage\_home>/jboss-artifact/brokerage.ear.

## 8.5.3 Creating and populating the database

Because we are using an IBM DB2 database in our example, we use a slightly modified version of database scripts that are included in sample distribution. To create and populate the database using the dbuser database user, complete the following steps:

1. Click **Start** → **Programs** → **IBM DB2** → **DB2Copy1 (default)** → **Command Line Tools** → **Command Editor**.

2. Enter the text that is shown in Example 8-2 in the command area, which is based on the <brokerage\_home>/db/db.sql file.

*Example 8-2 Command area code snippet*

---

```
CREATE DATABASE BRKRGDB;

CONNECT TO BRKRGDB USER DBUSER USING passwd;

CREATE TABLE "DBUSER"."USERS" (
    "USERID" VARCHAR(255) NOT NULL ,
    "NAME" VARCHAR(255) ,
    "PASSWORD" VARCHAR(255) ,
    "ADDRESS" VARCHAR(255) ,
    "CASH" DOUBLE ) ;

ALTER TABLE "DBUSER"."USERS"
    ADD PRIMARY KEY
    ("USERID");

CREATE TABLE "DBUSER"."STOCKS" (
    "ID" VARCHAR(255) NOT NULL ,
    "NAME" VARCHAR(255) ,
    "PRICE" DOUBLE ) ;

ALTER TABLE "DBUSER"."STOCKS"
    ADD PRIMARY KEY
    ("ID");

CREATE TABLE "DBUSER"."USERSTOCKS" (
    "ID" VARCHAR(255) NOT NULL ,
    "USERID" VARCHAR(255) NOT NULL ,
    "NAME" VARCHAR(255) ,
    "PRICE" DOUBLE ,
    "QUANTITY" INTEGER ) ;

ALTER TABLE "DBUSER"."USERSTOCKS"
    ADD PRIMARY KEY
    ("ID",
    "USERID");

INSERT INTO Users VALUES('srini', '6 Inch',
    'cheemu','Madivala Bangalore',10000);

INSERT INTO Users VALUES('j2ee', 'Lee Whittaker',
    'password','Tampa Florida',100000);

INSERT INTO userstocks VALUES('00000001', 'j2ee','BTDA',1,10);

INSERT INTO stocks VALUES('00000001','BTDA',1.00);
```

```

INSERT INTO stocks VALUES('00000002','Hitech Software',2.00);
INSERT INTO stocks VALUES('00000003','Bill s Cold Storage',3.00);
INSERT INTO stocks VALUES('00000004','Colonel s Fried Chicken',300.00);
INSERT INTO stocks VALUES('00000005','Toyo Motors',30.00);
INSERT INTO stocks VALUES('00000006','Timbuctoo Airlines',53.00);
INSERT INTO stocks VALUES('00000007','Happy Undertakers',73.00);
INSERT INTO stocks VALUES('00000008','Wacko Brothers',54.00);
INSERT INTO stocks VALUES('00000009','Mental Studios',456.00);
INSERT INTO stocks VALUES('00000013','ASFG',54.89);
INSERT INTO stocks VALUES('00000023','BFG',564.50);
INSERT INTO stocks VALUES('00000033','Mack',3444.50);
INSERT INTO stocks VALUES('00000043','Ronan & Sons',7.50);
INSERT INTO stocks VALUES('00000053','Bulls & Bears',553.40);
INSERT INTO stocks VALUES('00000343','HGHU',456.00);
INSERT INTO stocks VALUES('00000603','TTTM',77.00);
INSERT INTO stocks VALUES('00000463','GRASF',88.00);

COMMIT WORK;

CONNECT RESET;

TERMINATE;

```

3. Click **Execute**. A progress indicator is displayed. After the operations are completed, the results view displays the output (Figure 8-15).

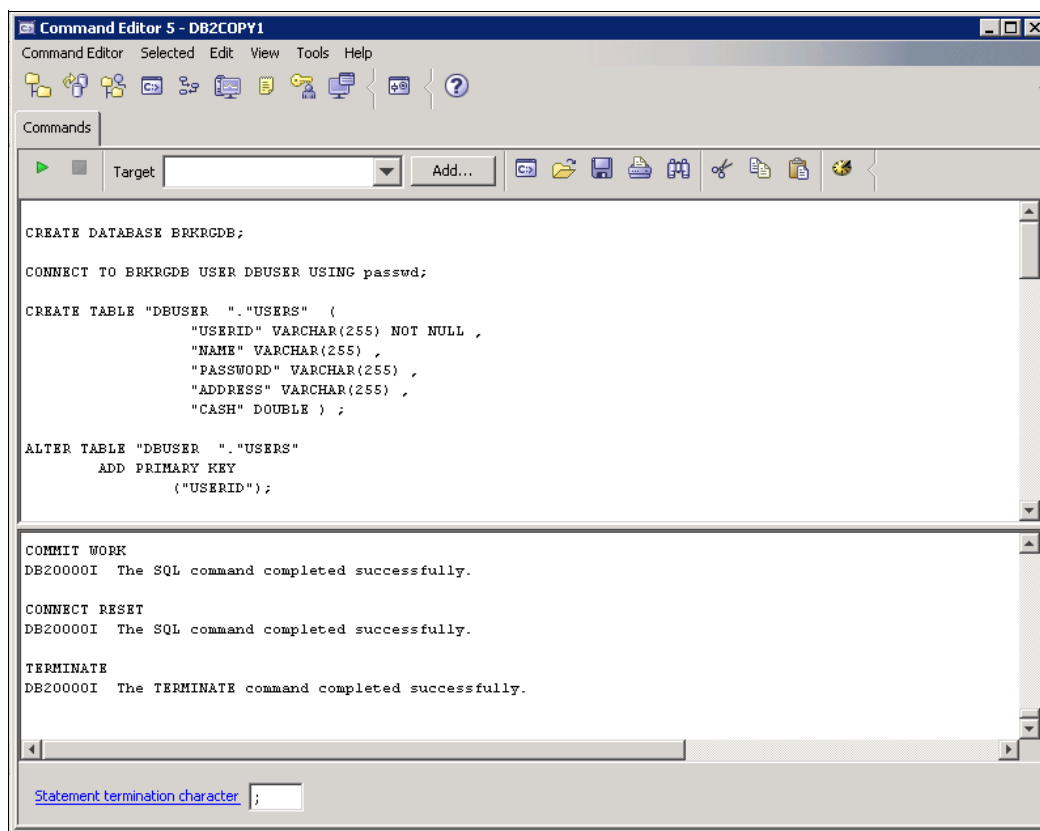


Figure 8-15 Creating the brkr gdb database for the sample application

## 8.5.4 Importing the EAR file and source code to Rational Application Developer

There are several alternatives for importing components in the HAR files in to our project:

- ▶ Reuse the generated HAR file in its binary form. Choose this alternative if you prefer to use other means of generating HAR files rather than Rational Application Developer for WebSphere Software.
- ▶ Create a utility module that contains contents of the HAR file, and on the web module add a dependency to this project. This alternative migrates all source code to Rational Application Developer for WebSphere Software as a second alternative, while keeping persistence-related classes at a separate place. This action is the main reason for using a HAR file.
- ▶ Import the contents of the HAR file (in our case, the `com.dev.trade.bo` package and `hbm.xml` files) in to a web module with other source code. This action results in a simpler project structure, as the web module contains all the source code.

In the following procedures, we demonstrate the second alternative. To do so, complete the following steps:

1. Start Rational Application Developer for WebSphere Software by clicking **Start** → **IBM Software Delivery Platform** → **IBM Rational Application Developer 8.5** → **IBM Rational Application Developer**.
2. Start the Import wizard by clicking **File** → **Import**. Expand the Java EE node in the import source tree and select the EAR file.
3. In the Enterprise Application Import window, browse to the location of the EAR file at `<brokerage_home>/jboss-artifact/brokerage.ear`. Click **Finish**.
4. Upon inspection, you might see that the `brokerage_WEB` module imported classes under Java Resources/Libraries/Web App Libraries/ ImportedClasses. We import the source files of these classes to the Java Resources/src folder in the same module and remove the class files from the project. Figure 8-16 shows the module structure before and after source code importation.

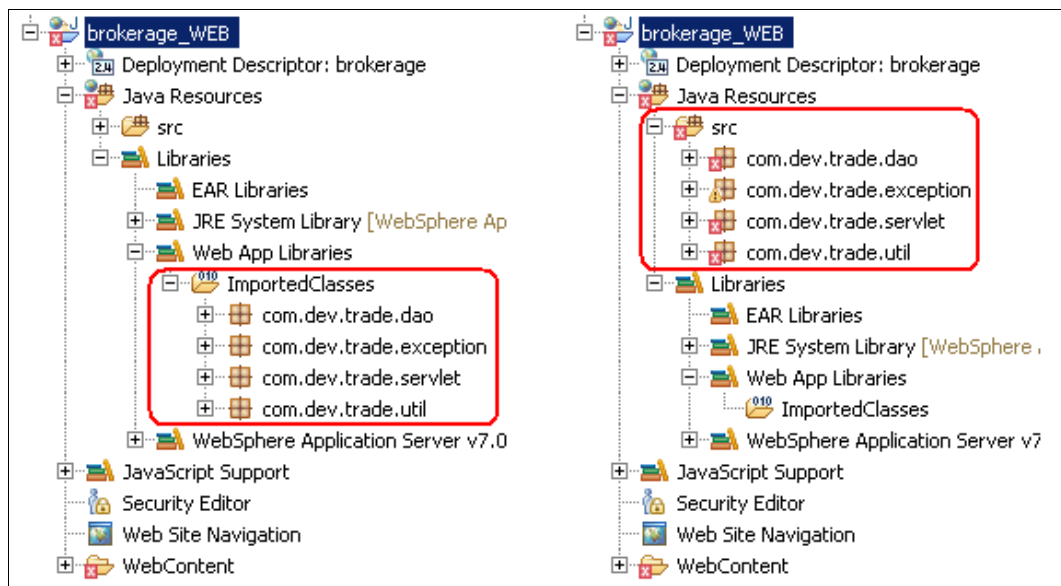


Figure 8-16 `brokerage_WEB` module before and after the source code import

5. Click the **Java Resources/src** node in the brokerage\_WEB module, and click **File** → **Import** from the main menu.
6. In the Import window, click **General** → **File System**. The File system import window is displayed. Browse to the C:\brokerage\src folder and import the source code (Figure 8-17).

**Importing source code:** We are not importing the com.dev.trade.bo package source code, as this package is packaged into the brokerage.har file, and is not a part of the web module.

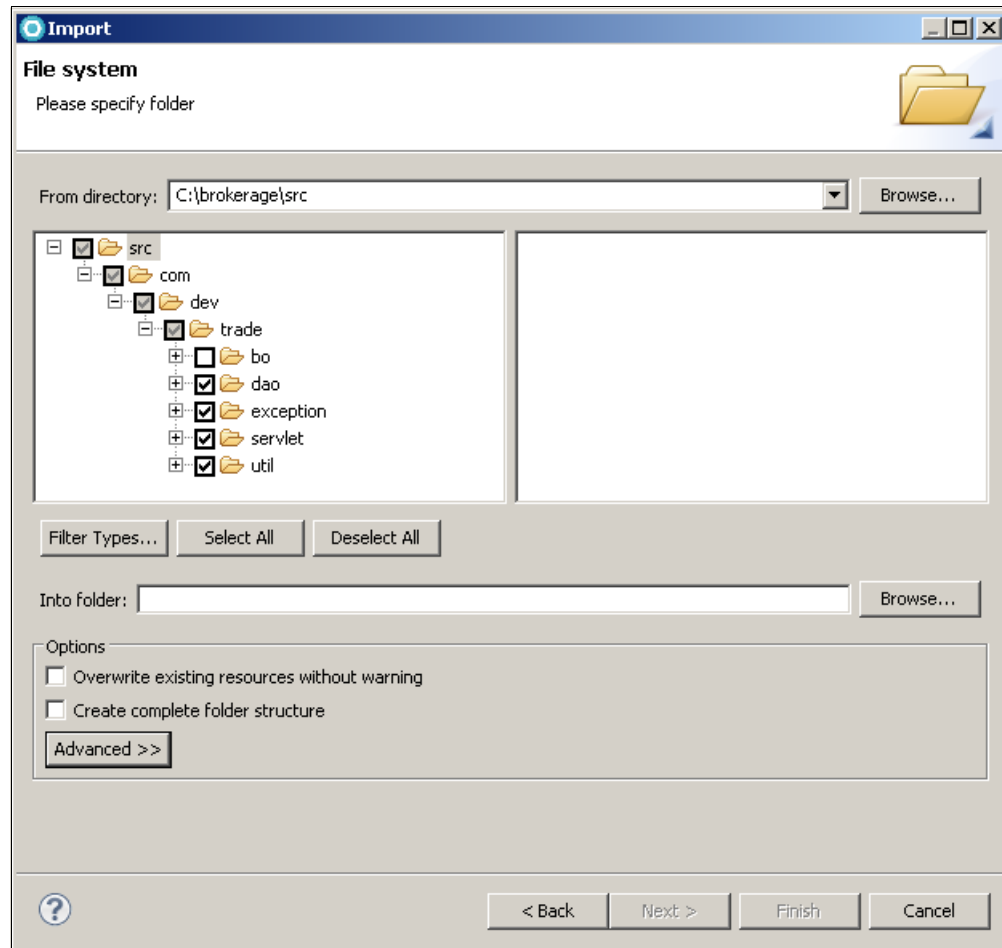


Figure 8-17 Importing source code files to brokerage\_WEB module

7. Delete the packages under the Java Resources/Libraries/Web App Libraries/ImportedClasses node by selecting them and then selecting **Delete**. Confirm the deletion. The project structure should now look like the right pane of Figure 8-16 on page 258.

## Checking the project using the Application Migration Tool

To run the Application Migration Tool on our example project, complete the following steps:

1. Click the brokerage application module. Right-click and select **Software Analyzer** → **Software Analyzer Configurations**. The Software Analyzer Configurations window opens. In the left pane, right-click **Software Analyzer** and select **New**.
2. Click the **Rules** tab. Select **JBoss Application Migration** in the rule set drop-down menu.

3. Click **Set....** The respective rules become selected in the Analysis Rules and Domains area. You see the dialog to choose WebSphere Application Server V8.5 after you click **Set....**
4. Click **Analyze.** After the analysis is complete, the Software Analyzer Results view opens, and is attached to the bottom pane. Inside the Software Analyzer Results view, you can see a tab for each domain of analysis. In our example, there are tabs named Java Code Review, XML File Review, and Class-path Review.
5. Expand the **Java Code Review** section and the results in this section. Note the “Check for Hibernate framework” warning. You are going to perform some changes in the next part to make the Hibernate framework run on the WebSphere Application Server. See Figure 8-18.

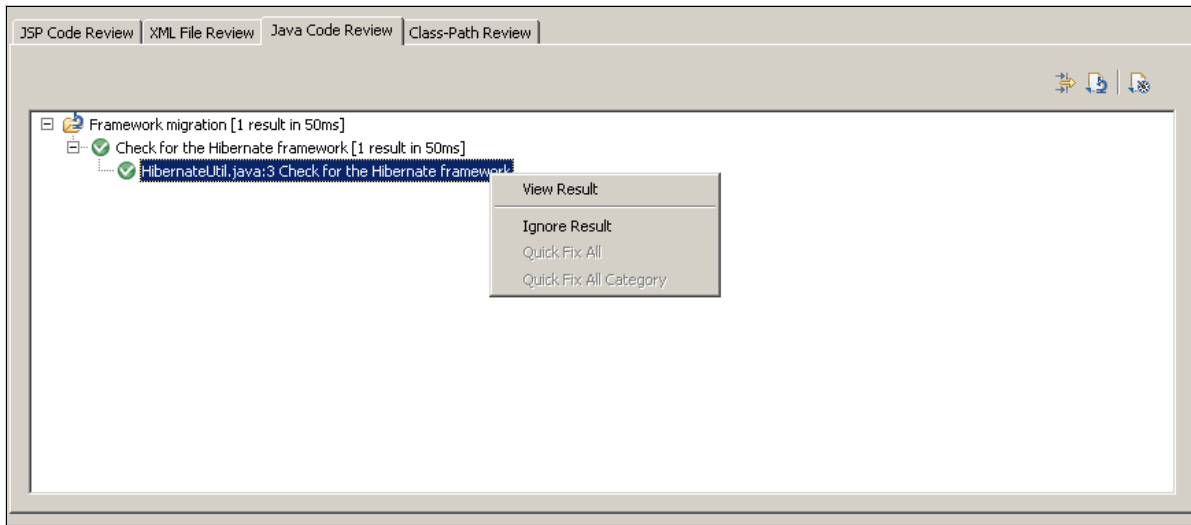


Figure 8-18 Java Code Review section

6. Expand the **XML file review** section and the results in this section. Under Use WebSphere bindings to define resource reference names, right-click **jboss-web.xml:5 found in brokerage\_WEB** and select **Quick Fix**. See Figure 8-19.

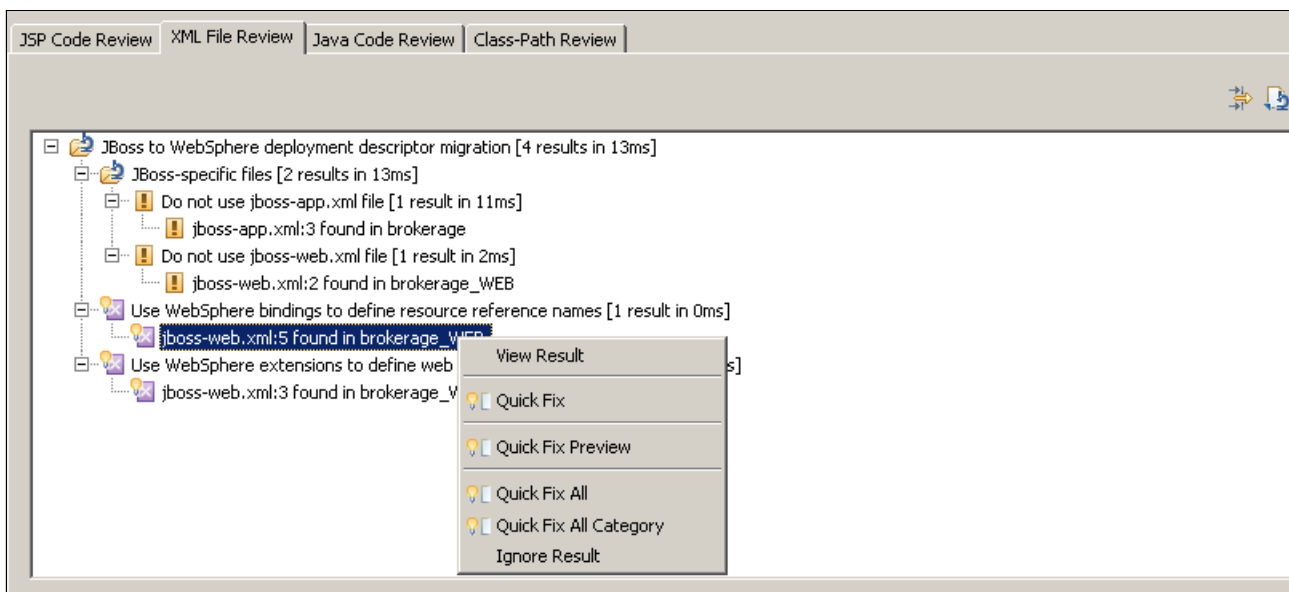


Figure 8-19 XML File Review section



7. Right-click **jboss-web.xml:5 found in brokerage\_WEB** and select **Quick Fix All**. See Figure 8-19 on page 260.
8. Under Use WebSphere extensions to define web module context root, right-click **jboss-web.xml:3 found in brokerage\_WEB** and select **Quick Fix**.

**Processing:** Running Quick Fix for these two rules migrates a JNDI name in the WebSphere bindings file and the web module context root in the WebSphere web module extension file.

After you complete the quick fixes and verifying that you do not need anything else from these JBoss extensions, you can delete `jboss-app.xml` and `jboss-web.xml`.

9. Remove the `jboss-app.xml` file by selecting **myproject\META-INF\jboss-app.xml** and clicking **Delete**.
10. Remove the `jboss-web.xml` file by selecting **myproject\META-INF\jboss-web.xml** and clicking **Delete**.
11. Run the Software Analyzer again to ensure that no more warnings are generated.

## 8.5.5 Analyzing and fixing the problems

Here the procedures to analyze and fix migration problems.

### Fixing compile time and runtime dependencies

As shown in the right pane of Figure 8-16 on page 258, there are compile time errors (flagged with red cross icons) on several classes. These errors are caused by missing third-party libraries, such as Hibernate. Also, in our example, we are missing the dependency to the resources that are packaged in the HAR archive.

To fix the compile time and runtime dependencies, complete the following steps:

1. Download the Hibernate distribution that is found at:  
<http://sourceforge.net/projects/hibernate/files/>  
At the time of the writing of this book, the latest version was 4.1.3.Final. Extract the file to a suitable location. We refer to this location as `<hibernate_home>` in our example.
2. Download the SLF4J distribution found at:  
<http://www.slf4j.org/download.html>  
At the time of the writing of this book, the latest version was 1.6.4. Extract the file to a suitable location. We refer to this location as `<slf4j_home>` in our example.
3. Select the **brokerage\_WEB/ WebContent/ WEB\_INF/ lib** folder. Right-click the selection and select **Import**. The Import wizard is displayed.
4. Click **General** → **File System** as the source and click **Next**. The File System window opens.

- Click **Browse** and navigate to the folder that contains the library and select the library to import. Do this step for each library that is shown in Table 8-2.

Table 8-2 Required libraries

Location	Library
<hibernate_home>	hibernate-core-4.1.3.Final.jar
<hibernate_home/lib/required	antlr-2.7.7.jar
<hibernate_home/lib/required	dom4j-1.6.1.jar
<hibernate_home/lib/required	javassist-3.15.0-GA.jar
<slf4j_home>	slf4j-api-1.6.4.jar
<slf4j_home>	sl4j-simple-1.6.4.jar

- To establish dependencies to resources in the HAR archive, rename the brokerage.har file in the brokerage application module root directory to brokerage.jar, by right-clicking this node and selecting **Rename**.
- Open the brokerage\_WEB module properties by pressing Alt+Enter with this node selected.
- Click **Java EE Module Dependencies**, and switch to the **Web Libraries** tab on this view.
- Click **Add Jars**. The Jar Selection window opens.
- Select **brokerage/brokerage.jar** and click **OK**. See Figure 8-20.

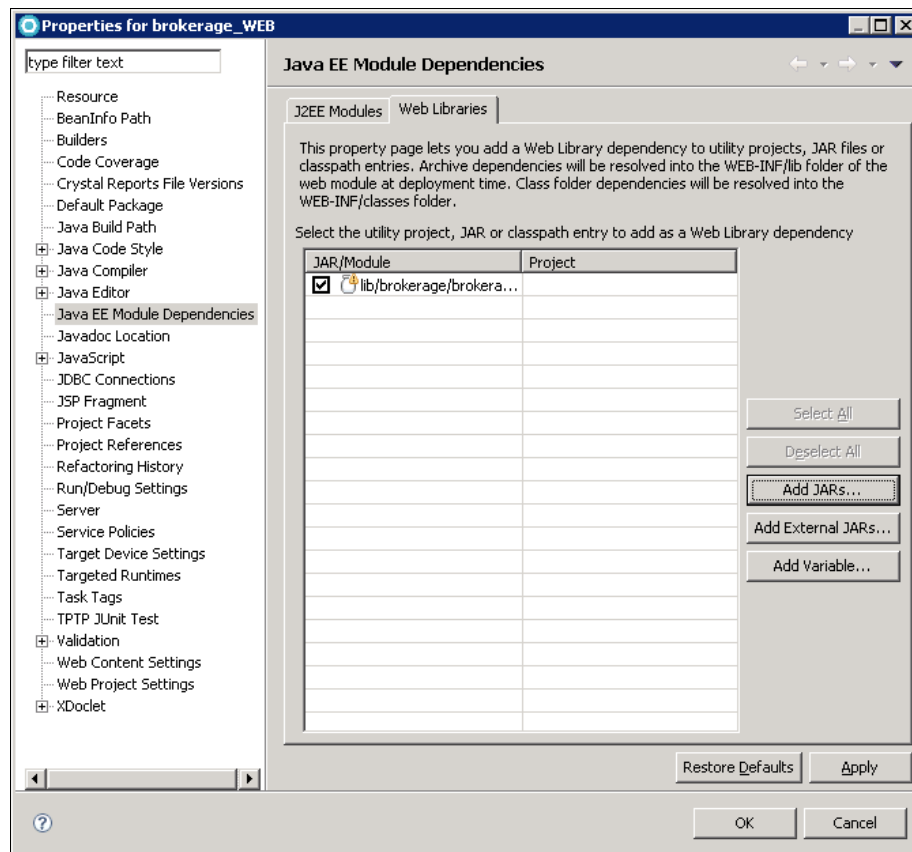


Figure 8-20 Adding a dependency to HAR resources

11. Click **OK**. There are no more compile time problems.

## Fixing the Hibernate configuration

This section describes how to fix the Hibernate configuration.

**Tip:** JBoss Application Server automatically processes HAR archives and configures the session factories, effectively binding them to the JNDI tree for further usage by other parts of the application. Because this action does not happen automatically on WebSphere Application Server, we must configure the session factories ourselves.

The first step in this process is to come up with a hibernate configuration file (hibernate.cfg.xml). A file with a similar purpose is already present in the META-INF directory of the HAR file (hibernate-service.xml). By using information in this file and the list of hbm.xml files in the HAR file, we can come up with a hibernate.cfg.xml file.

To fix the Hibernate configuration, complete the following steps:

1. Create a file in the `brokerage_WEB/Java Resources/src` folder by right-clicking this node and selecting **New** → **Other**.
2. Click **XML** → **XML** and click **Next**.
3. On the filename field, enter `brokerage.cfg.xml` and click **Finish**.
4. Add the code snippet in Example 8-3 as the contents of this file and save your changes.

### *Example 8-3 Changes to the file*

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<!-- name attribute is used for the JNDI name to be used for lookups -->
<hibernate-configuration>
    <session-factory name="java:hibernate/BrokerageSessionFactory">

        <!-- Datasource properties, tailored for DB2 -->
        <property
name="connection.datasource">java:comp/env/jdbc/HibernateDB</property>
        <property
name="hibernate.dialect">org.hibernate.dialect.DB2Dialect</property>
        <property name="hibernate.default_schema">dbuser</property>

        <!-- WebSphere specific properties -->
        <property
name="hibernate.transaction.manager_lookup_class">org.hibernate.transaction.Web
SphereExtendedJTATransactionLookup</property>
        <property
name="hibernate.transaction.factory_class">org.hibernate.transaction.JTATransac
tionFactory</property>
        <property name="jta.UserTransaction">
java:comp/UserTransaction</property>

        <!-- Mapping files (for each hbm.xml file in HAR file) -->
        <mapping resource="Stock.hbm.xml" />
    </session-factory>
</hibernate-configuration>
```

```
        <mapping resource="UserStock.hbm.xml" />
        <mapping resource="User.hbm.xml" />
    </session-factory>
</hibernate-configuration>
```

---

The following points are related to Example 8-3 on page 263 require your attention:

- The name attribute of the session-factory element is used later for looking up the session factory.
  - Because we are using IBM DB2 as our test database, we add the respective dialect.
  - We add WebSphere specific properties that are related to transaction strategy configuration according to topic found at the following web page:  
[http://www.ibm.com/developerworks/websphere/techjournal/0609\\_alcott/0609\\_alcott.html](http://www.ibm.com/developerworks/websphere/techjournal/0609_alcott/0609_alcott.html)
  - We add a mapping element for each hbm.xml file in our HAR archive.
5. Ensure that the Hibernate session factory that is defined in the `brokerage.cfg.xml` file is initialized properly before any part of the code looks up the Hibernate session factory using JNDI. The Online Brokerage application contains a single servlet that acts as a front controller, so a good place to initialize the session factory is the `init` method of this servlet.

**Tip:** For larger applications, using servlets to initialize session factories might not be a feasible solution. In this case, if you are migrating to WebSphere Application Server V7 or V8, use the startup beans. Startup beans allow business logic to run when an application starts or stops and gives you the opportunity to initialize session factories. Details about how to use startup beans can be found at following web page:

[http://www14.software.ibm.com/webapp/wsbroker/redirect?version=compass&product=was-nd-mp&topic=tasb\\_confstb](http://www14.software.ibm.com/webapp/wsbroker/redirect?version=compass&product=was-nd-mp&topic=tasb_confstb)

If you are migrating a large application to WebSphere Application Server V8.5, you should consider using the startup singleton session beans (EJB 3.1 specification). The capabilities that are provided with startup singleton session beans (EJB 3.1 specification) are preferred to the WebSphere Application Server proprietary startup beans function. Startup beans are supported in Version 8.5, but were deprecated in that release. For more information, see:

[http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=tasb\\_confstb](http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=tasb_confstb)

6. Open `TradeDispatcherServlet.java` in an editor by double-clicking it at **brokerage\_WEB/Java Resources/ src** in the `com.dev.trade.servlet` package.
7. At the `init` method, after the `super.init();` line, add the following line:  
`new Configuration().configure("brokerage.cfg.xml").buildSessionFactory();`  
The editor marks the `Configuration` class as not found.
8. Import the `org.hibernate.cfg.Configuration` class by pressing `Ctrl+I` with the configuration selected and select this class in the drop-down menu.
9. Save the `TradeDispatcherServlet` by pressing `Ctrl+S`.

## Creating the data source

As seen in `brokerage.cfg.xml`, our Hibernate configuration depends upon the existence of a data source with the JNDI name `jdbc/HibernateDB`. To create this data source, complete the following steps.

1. Select the brokerage application module, right-click, and select **Java EE** → **Open WebSphere Application Server Deployment**.
2. Scroll down to the Authentication section. Click **Add** near the JAAS Authentication list subsection. The Add JAAS Authentication Entry dialog box is displayed.
3. Enter `db2user` for the alias, user ID, and description fields, and `passwd` for the password field. Click **OK**. The new authentication alias is listed in the JAAS Authentication list.
4. Scroll up to the Data Sources section and click **Add** near the JDBC provider list subsection. The Create JDBC Provider window opens.
5. Select IBM DB2 as the database type and DB2 Universal JDBC Driver Provider (XA) as the JDBC provider type. Click **Next**. The JDBC Provider details window opens.
6. Enter `DB2 JDBC Provider (XA)` in the name field.
7. Select the entries in the classpath section and click **Remove**.
8. Click **Add External JARs**. A file selection dialog box is displayed. Navigate to the `C:\Program Files\IBM\SQLLIB\java` folder and select the following JAR files:
  - `db2jcc.jar`
  - `db2jcc_license_cu.jar`

Click **OK** to add. The JDBC provider settings window looks like Figure 8-21. Click **Finish**.

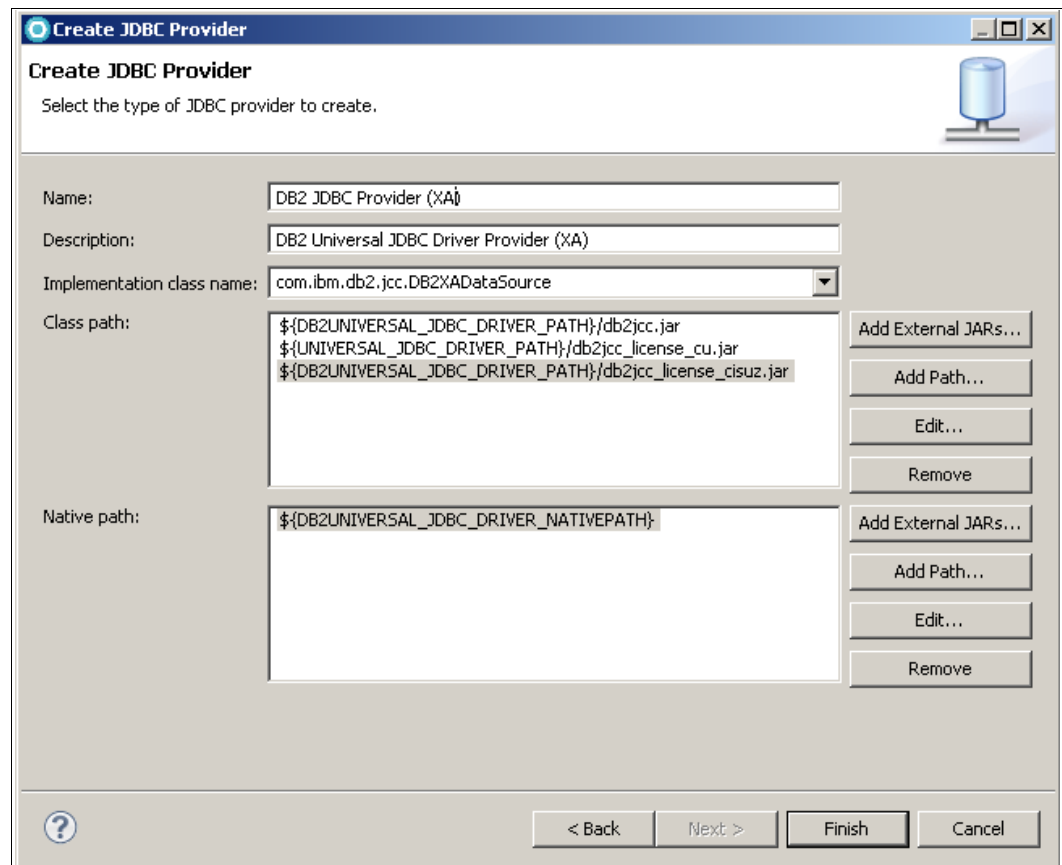


Figure 8-21 JDBC provider settings

The new JDBC provider is listed in the JDBC providers list.

9. Click the new JDBC provider listed in the JDBC providers list to select **DB2 JDBC Provider (XA)**, and then click **Add** in the data sources subsection. The Create Data Source window opens.
10. Select **DB2 Universal JDBC Driver Provider (XA)**. Click **Next**. The Data Source details window opens.
11. In the JNDI Name field, type jdbc/HibernateDB.
12. In the Container-managed authentication alias and Component-managed authentication alias fields, select the dbuser authentication alias that we created. See Figure 8-22. Click **Next**.

**Create Data Source**  
Select the type of data source to create.

Name:	Data source 1
JNDI name:	jdbc/HibernateDB
Description:	minVer null - maxVer null - DB2 Universal Driver Datasource
Category:	
Statement cache size:	10
Data source helper class name:	com.ibm.websphere.rsadapter.DB2UniversalDataStoreHelper
Connection timeout:	180
Maximum connections:	10
Minimum connections:	1
Reap time:	180
Unused timeout:	1800
Aged timeout:	0
Purge policy:	EntirePool
Component-managed authentication alias:	dbuser
Container-managed authentication alias:	dbuser

Use this data source in container managed persistence (CMP)

Figure 8-22 Data Source details

13. The resource properties window opens. Define the `databaseName` and `serverName` values. Specify `brkrpdb` for `databaseName`. Similarly, define `localhost` for `serverName`. Click **Finish**.

The Data Sources section of WebSphere Application Server Deployment window now looks like Figure 8-23. Press `Ctrl+s` to save your changes.

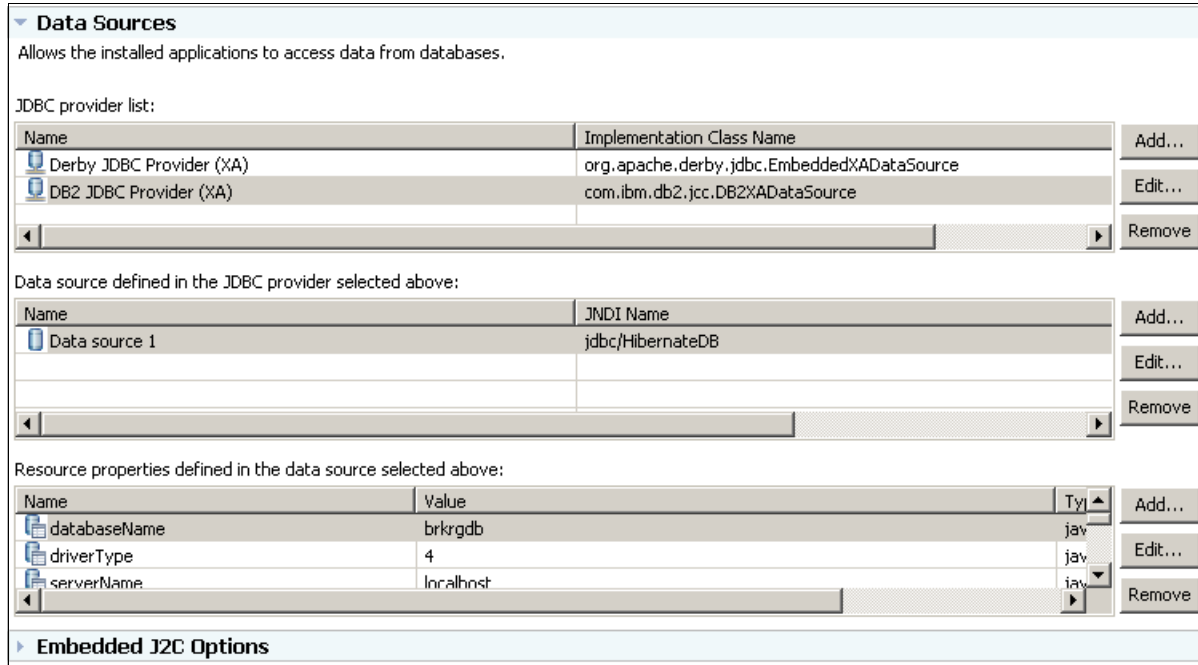


Figure 8-23 Data Sources window of the WebSphere Application Server deployment

14. Attach the data source to the defined resource reference in the web module. Open the `brokerage_WEB` module deployment descriptor in the editor by double-clicking **brokerage\_WEB/Deployment Descriptor: brokerage**.

15. Switch to the **References** tab and click **Add**. The Reference window opens.

16. Select **Resource Reference** and click **Next**.

17. On the References view, click **ResourceRef jdbc/HibernateDB**.

18. In the WebSphere Bindings section, enter jdbc/HibernateDB in the JNDI name field. See Figure 8-24.

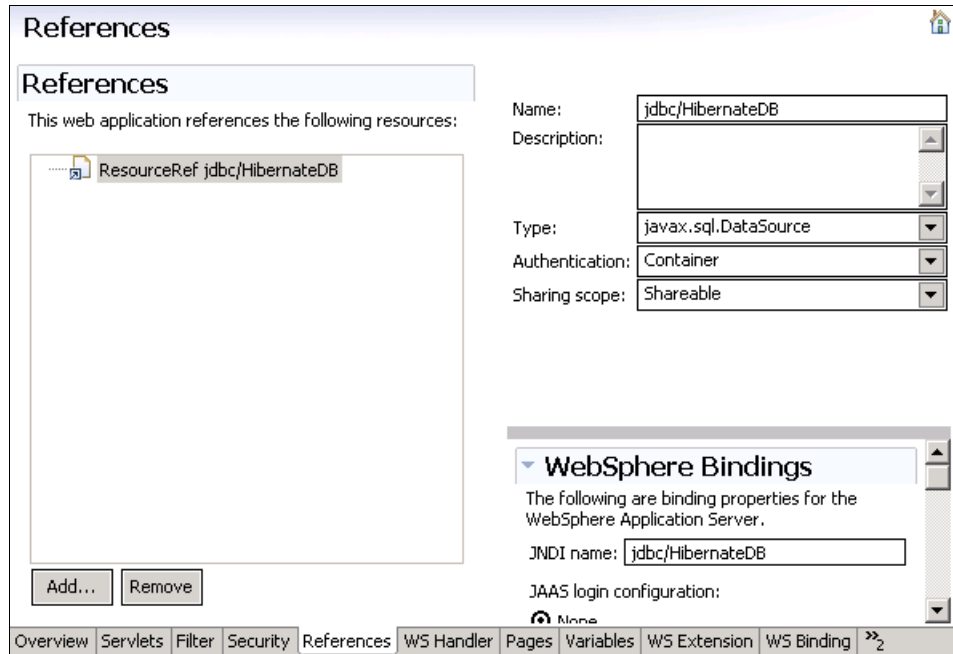


Figure 8-24 Resource references for brokerage\_WEB

19. Save your changes by pressing Ctrl+s.

## Building and running the application on an integrated test environment

The migration is complete and the application can be run on WebSphere Application Server V8.5. You can check how the migrated application works by using the integrated WebSphere Application Server V8.5 test environment.

To build, deploy, and run the application, complete the following steps:

1. Right-click the **brokerage\_WEB\WebContent\login.html** file and select **Run As** → **Run on server**. The Run on server dialog box is displayed. Keep the default settings. Click **Finish**.

After the application is deployed to the test environment, an embedded web browser is displayed in the editor pane, showing the entry page of the application.

2. Test the application by registering a new user for yourself and buying and selling stock.

## Importing HAR contents in to the source form

As mentioned in 8.5.4, “Importing the EAR file and source code to Rational Application Developer” on page 258, the configuration reuses the HAR archive in its binary form. You might also import HAR contents in source form in to Rational Application Developer for WebSphere Software. To do so, add a project with the contents of the HAR file, and configure a dependency to this new project at the web module. This way, developers can take full advantage of Rational Application Developer for WebSphere Software, and still keep persistence logic as a coherent unit, which is probably the intention for packaging this unit into a HAR archive in the first place.



Complete the following steps:

1. Create a Utility project by right-clicking the brokerage application module and selecting **New** → **Other**. In the wizard selection window, click **Java EE** → **Utility Project**.
2. Enter `brokerage_persistence` as the module name. See Figure 8-25. Click **Finish**.

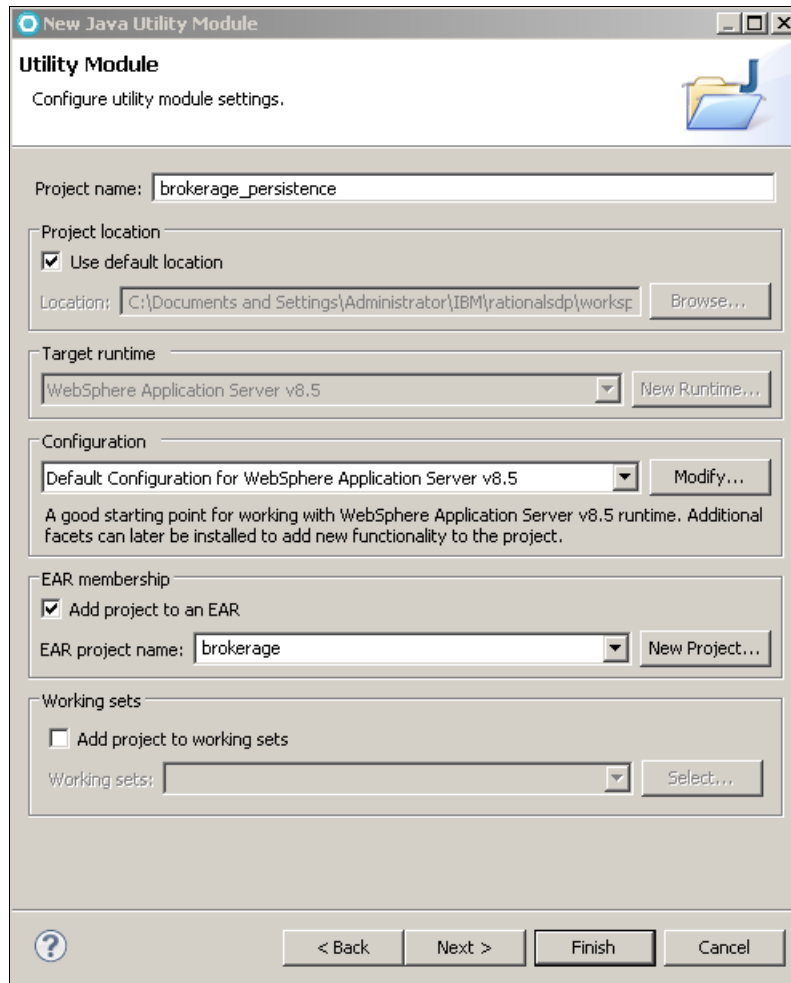


Figure 8-25 Creating a utility module

3. Import the `com.dev.trade.bo` package from `<brokerage_home>/src` in to the `brokerage_persistence/src` folder using the import wizard. See Figure 8-26.

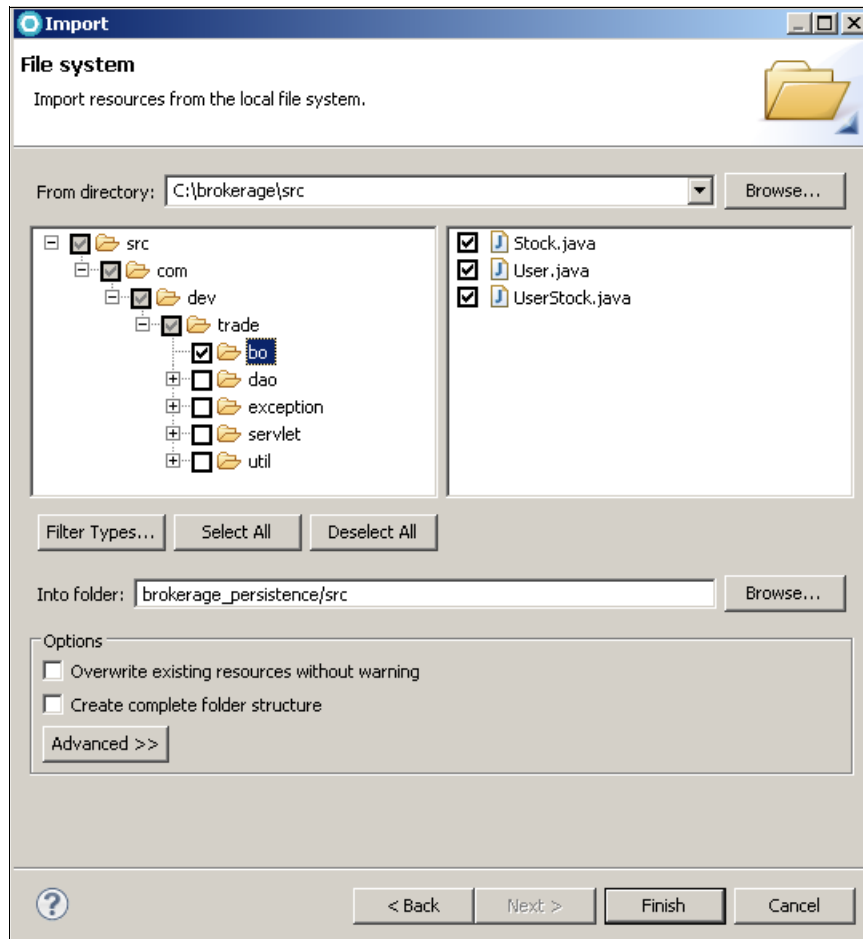


Figure 8-26 Importing the source files in to the utility project

4. Import the `hbm.xml` files from the `<brokerage_home>/hibernate` folder in to the `brokerage_persistence/src` folder using the import wizard. The utility module should look like Figure 8-27.

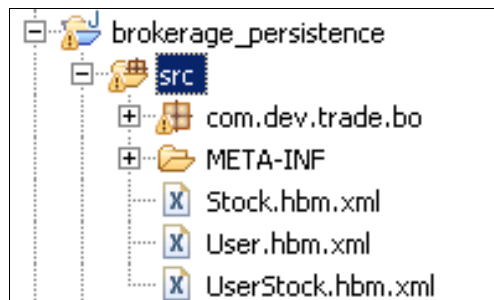


Figure 8-27 Utility module contents

5. Open the `brokerage_WEB` module properties by right-clicking the module and selecting **Properties**.

- In the Properties window, click **Java EE Module Dependencies**. Switch to the **Web Libraries** tab. Clear the check box next to the `brokerage.jar` and select the check box next to `brokerage_persistence` module. See Figure 8-28. Click **OK**.

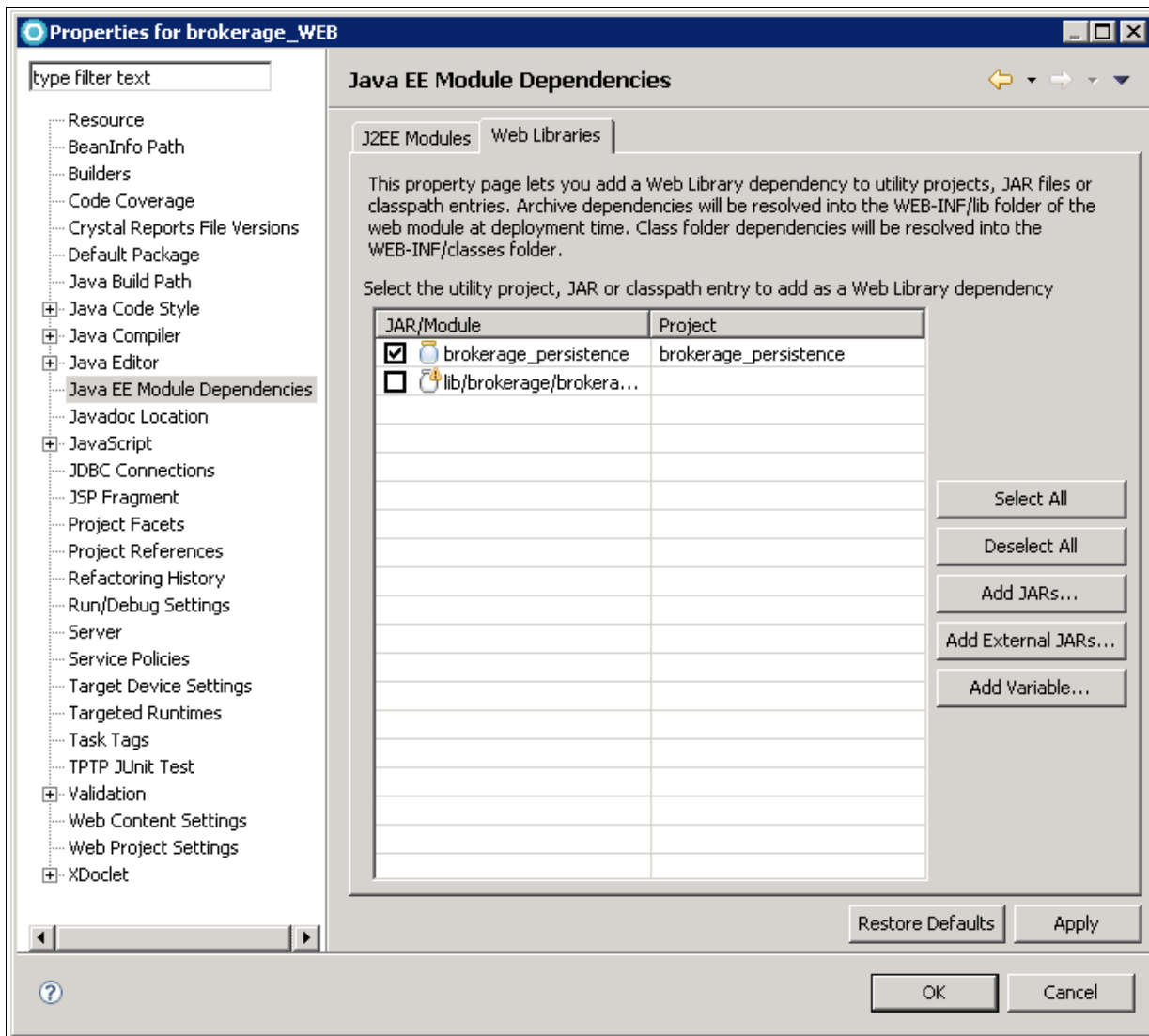


Figure 8-28 Module dependencies of web module

- Delete `persistence.jar` from the persistence application module.

## Summary

While migrating our sample generated application, we demonstrated the following concepts:

- ▶ Source a migration to Rational Application Developer for WebSphere Software as a build and test environment.
- ▶ Several alternatives for organizing the project structure for Hibernate applications using HAR archives.
- ▶ Configuration of Hibernate for WebSphere.
- ▶ Using Hibernate for persistence requires transaction a strategy configuration specific to WebSphere Application Server.
- ▶ Initializing session factories is not automatic on WebSphere Application Server. You must explicitly initialize session factories by using startup beans or other means.





## Migrating from Apache Tomcat

This chapter describes how to migrate three Java EE web applications from Apache Tomcat 7.0.27 to WebSphere Application Server V8.5 Liberty profile. There are several challenges that are associated with such migrations. One of them is the fact that Tomcat applications are often built on top of open source frameworks and libraries. The other challenge is that the development and build environment is often command line-based and uses Ant or Maven tools with no dependency on any particular Integrated Development Environment (IDE). We selected two typical Tomcat applications from <http://www.sourceforge.net> to reflect real world experience during migration, together with a third application that was written during the residency to show how to handle the migration of a more challenging application.

This chapter describes the following topics:

- ▶ Introduction
- ▶ Application Migration Tool - Apache Tomcat to WebSphere
- ▶ IBM Redbooks Publications Sample Application
- ▶ MvnForum migration
- ▶ Easy JSP Forum migration
- ▶ Summary

**Legal disclaimer:** IBM does not warrant or represent that the code provided is complete or up-to-date. IBM does not warrant, represent or imply reliability, serviceability, or function of the code. IBM is under no obligation to update content nor provide further support.

All code is provided "as is," with no warranties or guarantees whatsoever. IBM expressly disclaims to the fullest extent permitted by law all express, implied, statutory and other warranties, guarantees, or representations, including, without limitation, the warranties of merchantability, fitness for a particular purpose, and non-infringement of proprietary and intellectual property rights. You understand and agree that you use these materials, information, products, software, programs, and services, at your own discretion and risk and that you will be solely responsible for any damages that may result, including loss of data or damage to your computer system.

In no event will IBM be liable to any party for any direct, indirect, incidental, special, exemplary or consequential damages of any type whatsoever related to or arising from use of the code found herein, without limitation, any lost profits, business interruption, lost savings, loss of programs or other data, even if IBM is expressly advised of the possibility of such damages. This exclusion and waiver of liability applies to all causes of action, whether based on contract, warranty, tort or any other legal theories.

*Third-party* software is licensed and distributed to you by the *third-party* distributors and/or respective copyright and other right holders under their terms and conditions. IBM makes no express or implied warranties or representations with respect to such *software* and provides no indemnity for such software. IBM grants no express or implied patent or other license with respect to *and* is not liable for any damages arising out of the use of *such software*.

## 9.1 Introduction

Apache Tomcat 7.0.27 is an open source implementation of the Oracle Java EE web container. It implements the Java Servlet and JavaServer Pages specifications. Tomcat is developed by a community of developers under the umbrella of the Apache Software Foundation (ASF). Tomcat is widely used by many websites. It is popular for its small footprint, simplicity, and the many toolkits and frameworks that run on it.

Applications tend to grow over time and need a higher quality of service (QoS) and more sophistication than Tomcat can offer (such as centralized administration for clustered topologies, failover, advanced role-based security for users and administrators, script-based administration, advanced monitoring tools, portal infrastructure, page fragment and distributed Java caching, and transactions). Then, applications must be redeployed or migrated into enterprise-grade application servers, such as WebSphere Application Server, using either the full profile or the Liberty profile.

The transition to running applications on WebSphere Application Server can be made easier by using the Application Migration Tool - Apache Tomcat to WebSphere (Application Migration Tool). The toolkit provides the capability to identify common issues with migrating applications, and provides a number of automatic fixes (called *quick fixes*) to reduce the manual intervention that is involved in migration. Both Application Migration Tool and the WebSphere Application Server V8.5 Liberty Profile Developer Tools (referred to as "the developer tools") can be installed independently into the Eclipse IDE for Java EE Developers Version 3.7, which is freely available from the Eclipse website. For this reason, we used Eclipse, with Application Migration Tool and the developer tools installed, as our development and deployment environment throughout this chapter.

The three applications we selected for migration are:

- ▶ IBM Redbooks Publications Sample Application
- ▶ MvnForum
- ▶ Easy JSP Forum

### 9.1.1 Comparison of Apache Tomcat 7.0.27 and WebSphere Application Server V8.5 Liberty profile terminology

Here we present some of the similarities and differences in terminology between Apache Tomcat 7.0.27 and WebSphere Application Server V8.5 Liberty profile. The full profile and Liberty profile share common terminology.

Table 9-1 Terminology comparison

Description	WebSphere Application Server V8.5 Liberty profile	Apache Tomcat 7.0.27
Configure a file of users and roles for security.	Creates users in a <code>server.xml</code> file using a <code>BasicUserRegistry</code> or <code>LdapUserRegistry</code> element.	Creates users in a <code>tomcat-users.xml</code> file using role and user elements.
Mapping of JNDI naming to database drivers.	Creates JNDI mappings in a <code>server.xml</code> file using <code>jdbcDriver</code> and <code>DataSource</code> elements.	Creates JNDI mappings either in a <code>META-INF/context.xml</code> file in the application's WAR file, or in the <code>server.xml</code> file.
File Monitoring.	Configures file monitoring in a <code>server.xml</code> file using an <code>applicationMonitor</code> element.	Configures file monitoring in a <code>context.xml</code> file using a <code>WatchedResource</code> element.

## 9.2 Application Migration Tool - Apache Tomcat to WebSphere

Application Migration Tool - Apache Tomcat to WebSphere (Application Migration Tool) is part of the Migration Toolkit, which is described in detail in Chapter 4, "Installation and configuration of the Application Migration Tools" on page 89. Application Migration Tool can help you in the following areas when migrating Java web applications from Apache Tomcat (starting with Version 6.x) to WebSphere Application Server V7, V8, or V8.5 or a WebSphere Application Server V8.5 Liberty profile.

- ▶ Apache Tomcat proprietary Java package references: Detect proprietary package references.
- ▶ Apache Tomcat JSP issues: Detect nonstandard JSP.
- ▶ Apache Tomcat `context.xml` file: Detect and migrate `context.xml` file entries.
- ▶ Preferred practices for Spring: Detect and migrate Spring preferred practice issues.

## 9.3 Prerequisites and assumptions

There are a number of prerequisites and assumptions you must consider before you begin the migration examples. You should be familiar with the Application Migration Tool as described in Chapter 4, "Installation and configuration of the Application Migration Tools" on page 89.

You should also be familiar with the Java EE specification and architecture, and be able to understand and run basic SQL commands. You must also be familiar with Apache Tomcat 7.0.27.

The following software should be installed before you start the migration:

- ▶ Java SE 6

For our example source environments, we use Oracle Java Development Kit (JDK) 6. It is supported by both Apache Tomcat 7.0.27 and WebSphere Application Server V8.5 Liberty profile. However, the WebSphere Application Server V8.5 full profile supports IBM JDK 6 and IBM JDK 7. To demonstrate the migration of applications that are compatible with both the full profile and Liberty profile, we use the IBM JDK 6 that is provided as part of the IBM Installation Manager installation process of WebSphere Application Server V8.5. If it is not possible to use this JDK, or if you installed the Liberty profile from the archive installation process, you must install a supported JDK.

- ▶ Apache Tomcat 7.0.27

This version is the latest version available at the time of the writing of this book.

- ▶ IBM WebSphere Application Server V8.5 Liberty profile

Before you migrate to WebSphere Application Server V8.5 Liberty profile, familiarize yourself with the sample applications shipped with the product. Also, examine the publications that are listed in “Related publications” on page 437.

- ▶ IBM DB2 Universal Database 10.1

WebSphere Application Server V8.5 Liberty profile supports many databases, but for our example, we use DB2 for MvnForum in its source environment and all applications when using WebSphere Application Server V8.5 Liberty profile.

- ▶ WebSphere Application Server V8.5 Liberty Profile Developer Tools

WebSphere Application Server V8.5 Liberty profile Developer Tools (referred to as the developer tools) and the Application Migration Tool are used for all three applications in our example. For Easy JSP Forum, the official distribution available at <http://www.sourceforge.net> does not use an enterprise package application model, so you must import the application into the Eclipse to create the EAR file.

- ▶ Application Migration Tool

The Application Migration Tool is used throughout this chapter to analyze the applications that we migrate. The Application Migration Tool works best with the source code of an application, so, where possible, we scanned the source code and provided details of the results. For binary only packages, such as MvnForum, the Application Migration Tool provides information about the non-compiled elements, such as JSP and XML files. Although this information is useful, analyze the source code when possible, as it improves the ease of migration.

### 9.3.1 Installation of Apache Tomcat 7.0.27

This section provides high-level steps and tasks for installing and configuring Tomcat to run our applications in their original environment. We chose to create our Tomcat server in Eclipse, as it best demonstrates moving an application from Tomcat to the Liberty profile from within the same IDE.



Detailed instructions for installing, configuring, and managing Apache Tomcat 7.0.27 are provided in the product documentation guides available at the following web page:

<http://tomcat.apache.org/tomcat-7.0-doc/index.html>

The following steps illustrate the process that we use to install and configure Apache Tomcat 7.0.27, and then to create a Tomcat server in Eclipse:

1. Download the Apache Tomcat 7.0.27 binary distribution from the following web page:

<http://tomcat.apache.org/download-70.cgi>

The file that is downloaded is `apache-tomcat-7.0.27.zip` (7.72 MB).

2. Extract all archives into the same directory (referred to as `<tomcat_home>`):

`C:\apache-tomcat-7.0.27`

3. Open Eclipse.
4. Right-click in the **Servers** view and select **New** → **Server**.
5. When prompted to select the server type, click **Apache** → **Tomcat v7.0 Server**.
6. Click **Next**.
7. In the Tomcat installation directory, click **Browse** to navigate to `<tomcat_home>`.
8. Click **Installed JREs...**
9. In the **Installed JREs** Preferences page, click **Add...**
10. Select **Standard VM** and click **Next**.
11. Click **Directory**, and navigate to the root of your Oracle JVM.
12. Click **OK**.
13. Provide a JRE name, for example, `Oracle_JVM`.

**JRE name:** If this JRE name is used by another installed JRE, you might receive an error message that says that the JRE is in use. If so, you must provide a different name and be aware of this name when you perform the later steps.

14. Click **Finish**.
15. Click **OK** to close to preferences page.
16. In the **JRE:** drop-down menu, select **Oracle\_JVM** and click **Finish**.
17. Install the DB2 JDBC driver. The DB2 driver is used by our applications for connecting to IBM DB2 Universal Database 10.1. Copy `db2jcc4.jar` and `db2jcc_license_cu.jar` from the `<db2_home>\java` directory to the `<tomcat_home>\lib` directory.

### 9.3.2 Installing the developer tools

To work with the WebSphere Application Server V8.5 Liberty profile from Eclipse, install the IBM WebSphere Application Server V8.5 Liberty profile Developer Tools. These tools contain adapters for use with the Liberty profile and provide useful developer productivity benefits. They are freely available from Eclipse Marketplace.

To install the tools into Eclipse IDE for Java EE Developers Version 3.7, complete the following steps:

1. Open Eclipse.
2. Click **Help** → **Eclipse Marketplace...**

3. In the Search tab, enter Liberty or Liberty profile.
4. Click **Install** for **IBM WebSphere Application Server V8.5 Liberty Profile Developer Tools**, as shown in Figure 9-1.

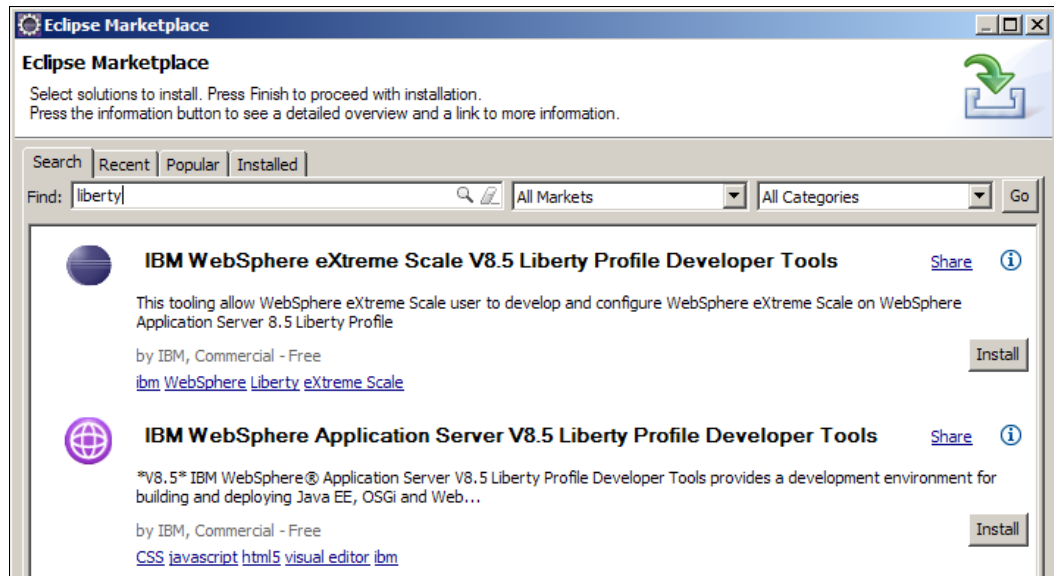


Figure 9-1 The developer tools for operating a Liberty profile server from Eclipse

5. After the dependency check is complete, click **Next**.
6. Review and accept the license agreement, and then click **Finish**.
7. If a security warning is displayed like the one in Figure 9-2, click **OK**.

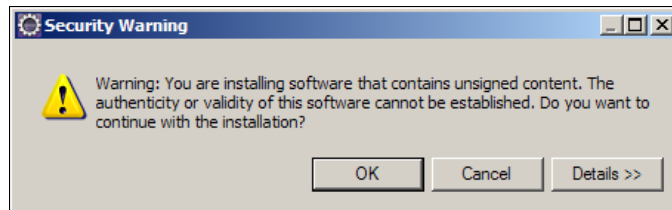


Figure 9-2 The security warning when you install the developer tools

8. After the installation is complete, click **Restart Now**.

### 9.3.3 Creating a Liberty profile server

To use WebSphere Application Server V8.5 Liberty profile, a Liberty profile server must be created. Each server contains a configuration and the required applications. Create the server in the developer tools by completing the following steps:

1. Open the developer tools.
2. Right-click in the **Servers** view and select **New** → **Server**.
3. Click **IBM** → **WebSphere Application Server V8.5 Liberty Profile**. The other options can be left as their defaults.

4. Create the server. You have three choices:
  - If the Liberty profile is already installed (for example, as part of an installation of WebSphere Application Server V8.5), then point to the root of the Liberty profile runtime environment (this directory is the one that contains the `bin` and `lib` directories, for example, `<was_home>/wlp`).
  - To use the archive installer from a local file system, click the **Download or install** link, then click **Install a new runtime environment from an archive** and click **Browse** to select the archive.
  - To use the archive installer from the IBM website, click the **Download or install** link, then click **Download and install a new runtime environment from:** and select **WebSphere Application Server V8.5 download site**. This website requires an IBM.com ID.

**Download or Install link:** If you install using the **Download or Install** link, this action requires further steps, including accepting license agreements and determining a location on which to install. We used the Liberty profile that we selected during the IBM Installation Manager installation process.

5. On the Liberty Profile Runtime Environment page, click the **Configure JREs** link.
6. In the Installed JREs Preferences page, click **Add...**
7. Click **Standard VM** and click **Next**.
8. Click **Directory**, and navigate to the root of your WebSphere Application Server V8.5 JVM (this root is in `<was_home>\java`).
9. Click **OK**.
10. Provide a JRE name, for example, `WebSphere_JVM`.

**JRE name:** If this JRE name is used by another installed JRE, you might receive an error message that says that the JRE is in use. If so, you must provide a different name and be aware of this name when you perform the later steps.

11. Click **OK**.
12. Click **Use a specific JRE:** and select **WebSphere\_JVM** from the drop-down menu.
13. Click **Next**.
14. To create a Liberty profile server, click **New** on the New Server window and provide the name of your server. We used `server1`, but any name using alphanumeric characters is valid. Complete this dialog box to create the server with a default configuration file, and click **Next**.
15. Click **Finish** on the Add and Remove page to complete the wizard.

Throughout the chapter, we refer to the root of the Liberty profile as `<liberty_home>`.

### 9.3.4 Installing the Application Migration Tool

Next, you must install the Application Migration Tool. For step by step instructions to accomplish this task, see 4.1, “IBM WebSphere Application Server Migration Toolkit overview” on page 90.

## 9.4 IBM Redbooks Publications Sample Application

The IBM Redbooks Publications Sample Application demonstrates the usage of the Application Migration Tool to detect and resolve problems in a more complex and difficult application. The application represents a process that might be required in a library, providing users with the ability to log in, view their borrowed books, search for and borrow a new book, or return a book. In addition, the foreground and background colors that are used on the page can be set on a per member basis. To enable easy testing of the application, there is a hidden test mode that the user can use to construct the required database tables and populate them with sample test data.

**Downloading the sample application:** For details about how to download the IBM Redbooks Publications Sample Application used in this scenario, see Appendix B, “Additional material” on page 435.

### 9.4.1 Migration approach

To maximize the effectiveness of the Application Migration Tool, the application, including source code, is imported into Eclipse. As we set up an initial environment using Tomcat, Eclipse IDE for Java EE Developers Version 3.7.2 is used as a development environment. To migrate the application to the Liberty profile, we install the WebSphere Application Server V8.5 Liberty Profile Developer Tools into the same IDE.

### 9.4.2 Configuring the initial environment

There are some steps that are required to ensure that the application runs successfully. These steps include configuring access to import the application into Eclipse, configuring access to log JAR files, configuring Derby database support, configuring data sources, and configuring the security roles.

#### Getting the application

The application can be accessed from the IBM Redbooks publications FTP site. The application is made available as a compressed file that contains a WAR archive file. For downloading instructions, see Appendix B, “Additional material” on page 435.

#### Importing the application into Eclipse

To import the application, complete the following steps:

1. Extract the WAR file from the downloaded sample application.
2. From Eclipse, click **File** → **Import...**
3. From the Import menu, click **Web** → **WAR file**.
4. Click **Next**.
5. Click **Browse** to navigate to the extracted WAR file.
6. Ensure that the Target run time is set to **Apache Tomcat v7.0**.
7. Click **Next**.
8. Click **Finish**.

## Configuring access to logging JAR files

When we import the project, we set the target run time to Tomcat. However, this action does not add access to the `tomcat-juli.jar` file that Tomcat uses for logging, and so this access must be configured manually. To add `tomcat-juli.jar` to the build path, right-click the application's project in the Project Explorer view and select **Build Path** → **Configure Build Path...** From here, switch to the **Libraries** tab, click **Add External JARs**, and navigate to the `bin` directory of the Apache Tomcat 7.0.27 installation. Select `tomcat-juli.jar`, click **Open**, and then click **OK**. This action flags an issue in the Markers view for classpath entry issues. This situation is expected and can safely be ignored.

**XML issue:** It is expected that the application has an XML issue in Eclipse (as show in the Markers view). This issue occurs because of the missing grammar constraints in `context.xml`. This issue is expected and does not impact the scenario.

## Configuring Tomcat to support embedded Derby databases

Apache Derby is an open source relational database that is written in Java. We select it here because it provides a JDBC driver that is easily deployed in a test and development environment. To get the required Apache Derby files, download the correct version from the following link. The specific file is `db-derby-10.8.2.2-bin.zip`.

<http://db.apache.org/derby/releases/release-10.8.2.2.cgi>

After the file downloaded, extract the `derby.jar` file from the `lib` directory of the compressed file and copy it into the `lib` directory of the Apache Tomcat 7.0.27 installation.

## Configuring data sources that are required by the application

Two data sources are configured within the application. One is for storing member data, the other is for storing book data. These data sources are pre-configured in the application's `META-INF/context.xml` file. These data sources need modifying only if the default database location (`/temp/TomcatDatabases`) must be modified. If you must modify the database location, find the relevant Resource element in the `META-INF/context.xml` file and edit the `url` attribute.

**Note:** The `url` attribute must still start with `jdbc:derby:` and end with `;create=true`.

## Configuring the global data source

In addition to the application-specific data sources, a global data source that records the background and foreground colors of the member's home page must also be specified. As data source is a global resource, it is configured in `server.xml` in the Tomcat configuration. To configure the data source, add the line that is shown in Example 9-1 to the `GlobalNamingResources` element inside the `server.xml` file of the Tomcat server. As we are using Eclipse, this file is under the Servers project in Eclipse. If the database location must be changed, it can be changed in the same way as the previous two resources.

*Example 9-1 Required configuration of the required global data source*

---

```
<Resource auth="Container" description="Derby database IBM Redbooks style Demo
app" driverClassName="org.apache.derby.jdbc.EmbeddedDriver" maxActive="100"
maxIdle="30" maxWait="10000" name="jdbc/globalStyleSettings" password=""
type="javax.sql.DataSource"
url="jdbc:derby:/temp/TomcatDatabases/styles;create=true" username=""/>
```

---

## Configuring the required security roles

To use the test mode, a Test role must be configured. To configure this role, add the configuration in Example 9-2 to the `tomcat-users.xml` file in the Servers project in Eclipse. If you are not using Eclipse, this file is found in the `conf` directory of the Apache Tomcat 7.0.27 installation.

*Example 9-2 The configuration that is required to add a user and a Test role*

---

```
<role rolename="Test" />
<user username="test" password="your_password" roles="Test" />
```

---

### 9.4.3 Deploying the application in Tomcat

The application is deployed directly from Eclipse to a running Apache Tomcat 7.0.27 server. This situation requires the Tomcat server to be associated with the same Eclipse environment. To run the application, right-click the project that you imported, and select **Run As** → **Run On Server**, select your Tomcat server, and click **Finish**. This action starts the Tomcat server and starts a browser window that displays the main page of the application.

### 9.4.4 Testing the application in Tomcat

To test the application, you must build the databases and create the test data. To accomplish this task, click the **Build The Databases** link in the main page of the browser window that is opened by your application when you deploy it. When prompted for login details, use the details of the user you created in Example 9-2.

**Performing test functions:** To perform any of the test functions, ensure that the `inTesting` environment variable is set to `true`. This variable can be found in the `META-INF/context.xml` file of the application. If this variable is set to anything other than `true`, creating the test database and test data is not possible.

After this action completes successfully, go back to the main page and click **Populate the Database with test data**. Again, this action requires a user in the test role, although if this action is done from the same browser window, it is likely your credentials are remembered from the previous action.

After this action completes successfully, go back to the main page, attempt to log in using a user ID of `user1` and a password of `password1`, set foreground and background colors, and return a book. In addition, you can search for a book and borrow it. This test is satisfactory one in a development environment, but in a production environment, a more rigorous test process is expected.

### 9.4.5 Migrating the application

Now that the application runs successfully in Tomcat, you can migrate to the WebSphere Application Server V8.5 Liberty profile. We continue with the use of Eclipse as a development environment, as you can install plug-ins that you can use to perform tasks such as installation and configuration of WebSphere Application Server V8.5 Liberty profile, and running the Application Migration Tool. Although we focus on the use of Eclipse with the WebSphere Application Server V8.5 Liberty Profile Developer Tools installed in our example, the same steps can be performed by using IBM Rational Application Developer for WebSphere V8.5.

**Workspace:** In our example, we use the same workspace to migrate the application from Tomcat to the Liberty profile, highlighting tasks such as changing the targeted run time for a project. However, you should have a back-up of your workspace and application before you start the migration. Remove the application from the Tomcat server and stop Tomcat before you start the migration.

## Changing the targeted run time of the application

In the previous sections, you added the application to Eclipse and set the targeted run time to Tomcat. To run the application on the Liberty profile, you must change the target run time to use the Liberty profile server that you configured. To accomplish this task, complete the following steps:

1. Right-click the project in Eclipse and select **Properties**.
2. Click **Targeted Runtimes**, select the **WebSphere Application Server V8.5 Liberty profile** check box, and clear the **Apache Tomcat V7.0** check box.
3. Click **Apply** and then **OK**.

These steps set the location where standard libraries are referenced from Apache Tomcat 7.0.27 to WebSphere Application Server V8.5 Liberty profile. In addition, the JRE used to compile the application is changed from Oracle\_JVM to WebSphere\_JVM.

Lastly, the `tomcat-juli.jar` file should be removed, as it is part of Apache Tomcat 7.0.27. To do this task, open the **Java Resources** in the project, click **Libraries**, right-click `tomcat-juli.jar`, and select **Build Path** → **Remove from Build Path**. These actions causes build errors, which are corrected by Application Migration Tool.

## Running the Application Migration Tool

The task of migrating the application to work on WebSphere Application Server V8.5 can be simplified by using the Application Migration Tool. To configure the Application Migration Tool, complete the following steps:

1. Right-click the project and select **Software Analyzer**.
2. Click **Software Analyzer Configurations....**
3. Right-click **Software Analyzer** and select **New**, as shown in Figure 9-3.

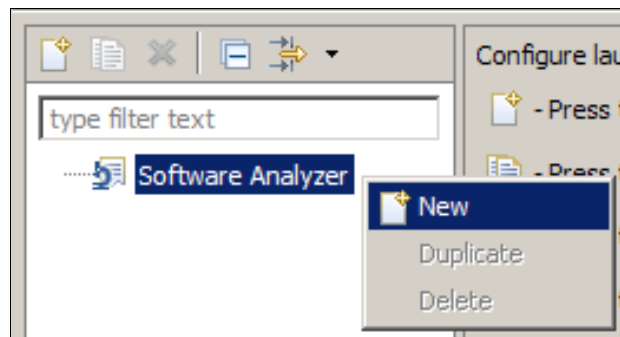


Figure 9-3 Creating a Software Analyzer configuration

4. Provide a name (for example, IBM Redbooks Library Migration).
5. Set the scope to **Analyze selected projects**, and then select the individual project.
6. Click the **Rules** tab.

- Click the **Apache Tomcat Application Migration** rule set and click **Set** (Figure 9-4). This action opens a prompt to select more detailed migration options.

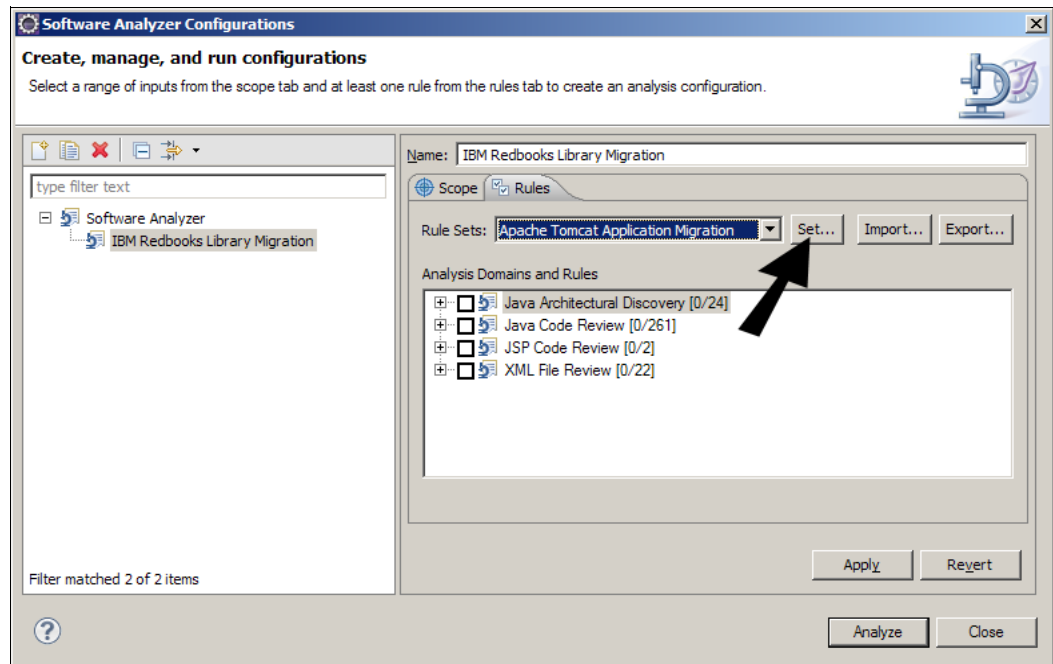


Figure 9-4 Setting the rules for an Apache Tomcat migration

- Set the Target application server to WebSphere Application Server V8.5, the Source Java version to Java 6, and the Target Java version to Java 6 (Figure 9-5).

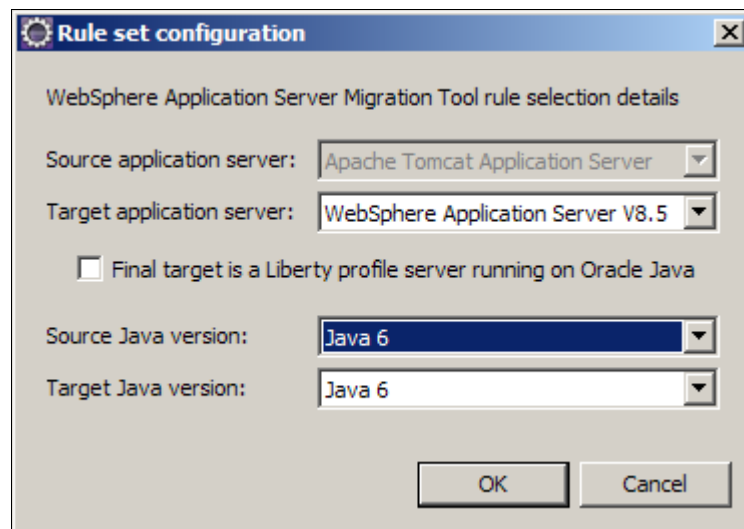


Figure 9-5 Settings for the configuration of the target WebSphere version, and the source and target JVM levels

**IBM JVM:** In our example, we ran this application on an IBM JVM, so we did not select the **Final target is a Liberty profile server running on Oracle JVM** check box. You should select this check box only if you are moving the application to the Liberty profile and using only the Oracle JVM.



9. Click **OK**, **Apply**, and then **Analyze**.

After Application Migration Tool finishes examining the application, the Software Analyzer Results view opens in Eclipse. This view highlights nine XML issues and 12 Java code issues.

***XML file review issues with quick fixes***

There are seven unique XML issues raised that can be automatically fixed:

- ▶ Avoid using a / in a web module welcome file name.

Welcome files are the pages that are served by the application server when a request is made that does not reference a specific resource. These files may not contain leading or trailing slashes, and so Application Migration Tool raises the issue because of the leading slash. The quick fix for this issue is to remove the leading slash.

- ▶ Set the sharing scope on resource references.

The jdbc/Members resource reference is defined in web.xml, and does not contain a scope, defining whether the resource is shareable or not. The quick fix for this issue adds the Shareable value to a res-sharing-scope element in the resource reference definition.

- ▶ Use Java EE deployment descriptors and WebSphere bindings to define resource link references.

The ResourceLink element is used in Tomcat to access a global style settings data source. Resource links allow an application to use a different JNDI name to locate another JNDI entry. To achieve this setup in WebSphere Application Server V8.5, the quick fix creates a resource mapping in an ibm-web-bnd file.

**Java EE Servlet:** For applications using Java EE Servlet 2.5 or later, this file is an XML file; applications using older specifications generate XMI files. As the IBM Redbooks Publications Sample Application is a Servlet 2.5 specification application, it uses XML files.

- ▶ Use Java EE deployment descriptors and WebSphere bindings to define resource references.

All resource references must be defined in the deployment descriptor web.xml file. For applications that run on WebSphere Application Server V8.5, resource references should also be specified in the ibm-web-bnd file. The quick fix for both cases is to create entries in the ibm-web-bnd file, and to create or update the resource references in web.xml as appropriate.

- ▶ Use Java EE deployment descriptors to define context parameters.

The application uses a parameter that is called notifyOnGet, that controls whether a log entry is created if the main servlet is accessed through an HTTP GET request. A more portable way of accomplishing this task is to specify the parameter in the web.xml file. Therefore, the quick fix for this issue is to re-create the parameter in the web.xml file with the same value of true.

- ▶ Use Java EE deployment descriptors to define environment references.

The application uses environment entries to control whether it is in a testing mode. If it is, certain operations that build and populate test databases are visible; otherwise, they are not displayed. The quick fix for this issue re-creates the entry in the web.xml file.

- ▶ Use Java EE deployment descriptors to define missing security roles.

As a secondary protection for the test mode, the database cannot be populated unless a user in the user role of Test is used to log in to build the database. However, this security role should be listed in the `web.xml` file. The quick fix for this issue adds the security role, with a role name of Test defined, to the `web.xml` file.

### **Java code review issues with quick fixes**

There are two unique Java code issues that have quick fixes available:

- ▶ Avoid using the invalid context `java:/comp`.

When you use JNDI lookups, the correct way of accessing the JNDI namespace is to use the prefix `java:comp`. The quick fix for this issue updates the lookup to the correct value.

- ▶ Do not use Apache Tomcat `org.apache.juli.logging`.

The application uses a JAR file in Apache Tomcat 7.0.27 to log messages. This JAR file is not exposed by default, and it is incorrect to use it instead of the more standard `java.util.logging.Logger` logging capability. Therefore, the quick fix updates all instances of the `org.apache.juli.logging` class with the relevant `java.util.logging` class, and updates the code to make the correct method calls.

### **Resolving all quick fixes**

To resolve all issues that have a quick fix available, complete the following steps:

1. In the Software Analyzer Results view, select the **Java Code Review** tab.
2. Right-click the **Apache Tomcat to WebSphere code migration** rule and select **Quick Fix All Category**.
3. In the Software Analyzer Results view, select the **XML File Review** tab.
4. Right-click the **Apache Tomcat to WebSphere deployment descriptor migration** rule and select **Quick Fix All Category**.

After these steps are complete, there is one XML result and four Java code results left. These results require manual investigation.

### **Manual adjustments**

The remaining XML issue relates to using the context valve component. This component is an Tomcat specific feature, which is used in the application to generate a log of all HTTP requests that are made to the application, and the server's response. Currently, there is no equivalent feature available in the Liberty profile, although you can use the full profile to configure HTTP access logs. Therefore, although the logs are not generated, the warning can be ignored without affecting the functionality of the booking system. To flag the issue as ignored, complete the following steps:

1. Expand the **Apache Tomcat to WebSphere deployment descriptor migration** category in the XML File Review tab of the Software Analyzer Results view.
2. Expand the **Do not use the context valve component** rule.
3. Right-click the `context.xml` result, and select **Ignore Result**. This action tags the result as ignored and removes it for the reported results.

The next step is to update the Java code. Application Migration Tool highlights the usage of `org.apache.tomcat.dbcp.dbcp.BasicDataSource` in the `DataSourceManager` class. This usage is not compatible with WebSphere Application Server. To correct this issue, change the reference to `javax.sql.DataSource`. Using `javax.sql.DataSource` is more portable than the `BasicDataSource`, which is Tomcat specific.

To fix the issue of using Tomcat specific APIs, click **Java Code Review** → **Apache Tomcat to WebSphere code Migration** and expand the rule labeled **Do not use Apache Tomcat packages and APIs**. This action opens the menu as shown in Figure 9-6. Double-click the result for **BuildDatabases**. This action opens the **BuildDatabases** class, and moves the cursor to the reported line. In this case, the application uses a Tomcat specific class when it outputs server version information, but this class is not critical to the application. To fix this issue, comment out this line. Currently, the Liberty profile does not provide an equivalent mechanism.

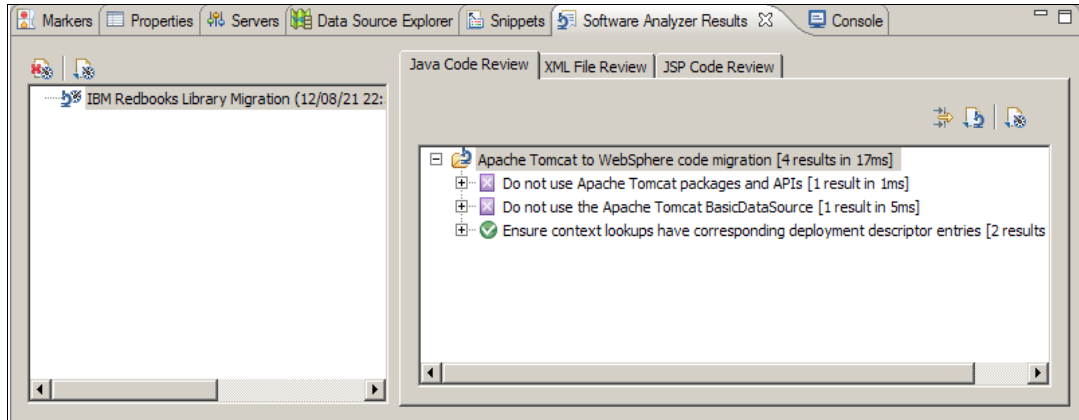


Figure 9-6 Results of running the Software Analyzer

The last two issues that are reported are checks to ensure that the correct context lookups are defined in the deployment descriptors. For lookups in the `DataSourceManager` class, the process of performing the quick fixes adds all the resource references to the `web.xml` file. For the `BuildDatabases` class, the quick fix adds this class to the `web.xml` file of the application. If the quick fixes were applied, you can now flag these results as ignored. To clear these results, expand the rule, right-click each result, and select **Ignore Result**. This action flags the result as ignored and removes it from the reported results.

**Deployment descriptor bindings:** You can see from the migration steps that the necessary deployment descriptor bindings are created, so you can ignore these tasks. When you analyze an application for the first time, these settings should be checked to ensure that the required deployment descriptor bindings are created where they are required.

You can rerun Application Migration Tool by right-clicking the project in Eclipse and selecting **Software Analyzer** → **IBM Redbooks Library Migration**. This action should return no results.

## 9.4.6 Configuring the target environment

There are two configuration tasks that must be performed to set up the scenario:

1. Create the relevant data sources.
2. Create a test role in a user registry.

For our migration, we use DB2 as a back-end database. Using the application in test mode provides the necessary capabilities to create the tables required. The only setup that DB2 requires is to create the databases in the DB2 instance. You can create the database by running the commands in Example 9-3 on page 288.

```
db2 create db MEMBER
db2 create db BOOKS
db2 create db STYLES
```

---

## Creating the data sources

After the databases are defined in DB2, create the data sources and JDBC driver by completing the following steps:

1. Open the Server Configuration window of the Liberty profile server from the Servers view in Eclipse. If the Source view is shown, switch to the Design view.
2. Create the Fileset Service:
  - a. Right-click the **Server Configuration** in the Configuration Structure window, and select **Add** → **Fileset Service**.
  - b. Provide an ID for the Fileset Service, such as DB2DriverFileSet.
  - c. Set the Base directory to the location of the db2jcc4.jar and db2jcc\_license\_cu.jar files.
  - d. Set the Includes pattern to db2jcc4.jar, db2jcc\_license\_cu.jar.
3. Create the Shared Library:
  - a. Right-click the **Server Configuration** in the Configuration Structure window and select **Add** → **Shared Library**.
  - b. Provide an ID for the Shared Library, for example, DB2DriverSharedLibrary.
  - c. Provide a Fileset reference that is the Fileset Service you created (DB2DriverFileSet in our example).
4. Create the JDBC driver:
  - a. Right-click the **Server Configuration** in the Configuration Structure window and select **Add** → **JDBC Driver**.
  - b. Set an ID for the JDBC driver, for example, DB2JDBCdriver.
  - c. Set the Shared libraries field to your Shared Library. In our example, it is DB2DriverSharedLibrary.
5. Create the Members data source:
  - a. Right-click **Server Configuration** in the Configuration Structure window and select **Add** → **Data Source**.
  - b. Set the ID to MembersDS. This value is useful for debugging purposes, as it means that the Liberty server identifies this resource uniquely in output messages.
  - c. Set the JNDI name to jdbc/Members.
  - d. Set the JDBC driver to the JDBC Driver you previously created (in our example, this value is DB2JDBCdriver).
  - e. Right-click the **MembersDS Data Source** in the Configuration Structure window and select **Add** → **Properties for DB2 JCC**.
  - f. Click the **Properties for DB2 JCC** section to open it.
  - g. Set the Database Name to MEMBER.
  - h. Set the Server name to the host name of your server.
  - i. Set the User and Password fields as required.

**Password field editing:** The Password field cannot be directly edited. You must click **Edit....**

6. Create the Books data source:
  - a. Right-click **Server Configuration** in the Configuration Structure window and select **Add → Data Source**.
  - b. Set the ID to BooksDS. This value is useful for debugging purposes, as it means that the Liberty server identifies this resource uniquely in output messages.
  - c. Set the JNDI name to DefaultDatasource.

**JNDI name:** The JNDI name for the BooksDS data source does not start with jdbc/. It is important that the JNDI name is entered exactly as shown in the instructions, including case, or the application throws an exception at run time.

- d. Set the JDBC driver to the JDBC Driver you created (in our example, this drive is DB2JDBCdriver).
  - e. Right-click the data source in the Configuration Structure window and select **Add → Properties for DB2 JCC**.
  - f. Click the **Properties for DB2 JCC** section to open it.
  - g. Set the Database Name to BOOKS.
  - h. Set the Server name to the host name of your server.
  - i. Set the User and password fields as required.

**Password field:** The Password field cannot be directly edited. You must click **Edit....**

7. Create the Style data source:
  - a. Right-click **Server Configuration** in the Configuration Structure window and select **Add → Data Source**.
  - b. Set the ID to StyleDS. This value is useful for debugging purposes, as it means that the Liberty profile server identifies this resource as unique in output messages.
  - c. Set the JNDI name to jdbc/globalStyleSettings.
  - d. Set the JDBC driver to the JDBC Driver you created (in our example, it is DB2JDBCdriver).
  - e. Right-click the data source in the Configuration Structure window and select **Add → Properties for DB2 JCC**.
  - f. Click the **Properties for DB2 JCC** section to open it.
  - g. Set the Database Name to STYLES.
  - h. Set the Server name to the host name of your Liberty profile server.
  - i. Set the User and Password fields as required.

## Creating the security registry

Mapping the security role must be done after the application is added to the server configuration. However, the registry that contains the test role must exist. To create the registry, use a Basic User Registry in the Liberty profile server. A Basic User Registry is part of the server configuration, and consists of a set of users and roles that are stored in an XML structure. To create the registry, complete the following steps:

1. Right-click the **Server Configuration** in the Configuration Structure window and select **Add** → **Basic User Registry**. Provide an ID of TestRegistry and a Realm name of BasicRegistry.

**Registry type:** Only one type of registry can be configured in the Liberty profile server at any one time. If you decide to set this application to use an LDAP server, the Basic User Registry must be removed.

2. Right-click the **Basic User Registry** and select **Add** → **user**. Set the User name and Password to any convenient value. We use a User name of test and a Password of testpasswd.
3. Save the configuration.

## 9.4.7 Deploying the application in the Liberty profile

To deploy the application, right-click the server in the Servers view and select **Add and Remove**. From here, select the application and add it to the server. Click **Finish** to complete this wizard.

The security role is not required to run the application, but without it, the test mode is inaccessible. To configure the test role to have access to the application, complete the following steps:

1. Open the server configuration by double-clicking the **Server Configuration** section of the server in the Servers view.
2. Right-click the application in the Server Configuration window and select **Add** → **Application Bindings**.
3. Right-click the **Application Bindings** and select **Add** → **security-role**.
4. Set the Security role name to Test.
5. Right-click the security-role and select **Add** → **user**. For the User name, enter the name of the user in the Basic User Registry. In our example, it is test.
6. Click the **Feature Manager** in the Configuration Structure window.
7. In the Feature Manager Details window, click **Add** and select **appSecurity-1.0**.
8. Save the configuration.
9. To run the application, right-click the application in the Project Explorer view and select **Run As** → **Run on Server** and then click **Finish**. This action should open a browser on your system and point to the following URL:

`http://localhost:9080/RedbookLibrary/`

## 9.4.8 Testing the application in the Liberty profile

To test the application, you must build the databases and create the test data. To do this task, click the **Build The Databases** link in the main page of the browser started by the application.

When prompted for login details, use the details of the user you created in the Basic User Registry.

**Testing:** To perform any of the Test functions, ensure that the `inTesting` environment variable is set to `true`. This variable can be found in the `WEB-INF/web.xml` file of your application. If this variable is set to anything other than `true`, you cannot create the test database and test data.

After this action runs successfully, go back to the main page and click **Populate the Database with test data**. Again, this action requires a user in the test role, although if this action is done from the same browser window, it is likely that your credentials are remembered from the previous action.

After this action runs successfully, go back to the main page, attempt to log in with a user ID of `user1` and a password of `password1`, set the foreground and background colors, and return a book. In addition, you can search for a book and borrow it. Although this test is a satisfactory one in a development environment, in a production environment, a more rigorous test process is expected.

## 9.5 MvnForum migration

The MvnForum open source edition is used in this book. This application implements the bulletin board, also known as the forum. The product runs using most Java EE servers and most databases. In this book, we use DB2 as the back-end database. The MvnForum application works on Linux, UNIX, and Windows, and is implemented as a J2EE 1.4 web application. This application is distributed with the full source code under the terms of the GNU General Public License (GPL). It is an open source, powerful, easy to use, and easy to set up bulletin board (forum) built on Java EE technology (JSP and Servlet) that you can use at no additional charge to build your own discussion communities. MvnForum is compatible with any servlet containers that support JSP 1.2 and Servlet 2.3.

This application is an interesting one for migration for the following reasons:

- ▶ It is based on open source frameworks.
- ▶ It is designed to run on Tomcat, JBoss, WebSphere Application Server, and other Java EE application servers.
- ▶ It has never been tested on WebSphere Application Server V8.5 Liberty profile.
- ▶ The application has a range of functions and is relatively large. It uses Java EE and Java SE functionality (such as JSP 1.2) with a number of custom tags, Servlet 2.3, and JSTL 1.0.2.

Overall, this application is representative of a typical ISV build if the ISV is using Tomcat or JBoss as its deployment platform.

### 9.5.1 Migration approach

The MvnForum application is built with non-IBM development tools using Apache Ant as the build tool. The application must be enhanced in the future and it must support Tomcat and WebSphere Application Server as deployment targets. You must decide which tool to use to migrate the application and which tools to use to continue development of this application going forward.

For this particular project, we decide to import the application in to Eclipse after it is compiled, rather than importing the source code first and compiling within Eclipse. Here we take advantage of the application's pre-existing build process, which is developed by a team already familiar with the application and preferred practices in building it. When you test the application with WebSphere Application Server, we found that all the tested functionality worked as expected, so we are satisfied in this case. However, we did perform a scan of the source code, which we describe later. In addition, in the case of a migration for a production system, running the Application Migration Tool on the source of the application has a significant effect on reducing the complexity of a migration.

## 9.5.2 Configuring the source environment

This section describes how we configure Apache Tomcat 7.0.27 and MvnForum to run in the source environment in our example. Instructions about how we installed Apache Tomcat 7.0.27, and details about the other steps that are required to run the applications that listed in this chapter, are described in 9.2, "Application Migration Tool - Apache Tomcat to WebSphere" on page 275.

### Getting the MvnForum application

The MvnForum application is available from <http://www.sourceforge.net>. To create a web application that can easily be imported into Eclipse and deployed to an application server, complete the following steps.

1. Download the `mvnforum-1.2.2-mvnad-1.0.1-bin-20100817.zip` file, which contains the files for the web application, from the following web page:

<http://sourceforge.net/projects/mvnforum/files/>

2. After the file is downloaded, extract its contents to a folder. We refer to this folder as `<mvnforum_app>`.
3. Create a compressed file with a WAR extension from the contents of `<mvnforum_app>\webapp_mvnforum_only`. To do this task, open a command prompt, navigate to `<mvnforum_app>\webapp_mvnforum_only`, and run the following command:

```
jar -cf MvnForum.war *
```

**The jar command:** We used the `jar` command to create the archive, as it is included in the JDK from Oracle and IBM, and can be found in either product's `bin` directory. However, any compression tool can be used if the resulting archive contains all the files and folders that are found in `<mvnforum_app>\webapp_mvnforum_only` in the root of the archive (that is, they are not inside another directory).

The resulting package is ready to import into Eclipse.

### Importing the application in to Eclipse

To import the application in to Eclipse, complete the following steps:

1. From Eclipse, click **File** → **Import...**
2. From the Import menu, click **Web** → **WAR file**.
3. Click **Next**.
4. Click **Browse** to locate `MvnForum.war` in `<mvnforum_app>\webapp_mvnforum_only`.
5. Ensure that the Target run time is set to Apache Tomcat V7.0.
6. Click **Next**.
7. Click **Finish**.



After the import is complete and the workspace is rebuilt, Eclipse reports a number of potential problems with the application:

- ▶ A number of broken links. These broken links are in the original MvnForum application.
- ▶ A number of HTML warnings about the usage of obsolete HTML tags. This usage is from the original MvnForum application.
- ▶ A number of reported JSP problems.
- ▶ Few Javascript Tasks. These issues are caused by “FIXME :” tags in the original MvnForum application.
- ▶ Few XML problems, including an error reported in a file that is downloaded from the following URL:

[http://www.mvnforum.com/dtd/mvnforum\\_1\\_0\\_rc2.dtd](http://www.mvnforum.com/dtd/mvnforum_1_0_rc2.dtd)

These warnings and errors do not prevent the application from running in Tomcat or from using the Liberty profile, so they do not need to be addressed as part of the migration. The errors and warnings can be suppressed by configuring the **Markers** view. To do this task, complete the following steps:

1. Select **Configure Contents...** from the View menu, as shown in Figure 9-7.

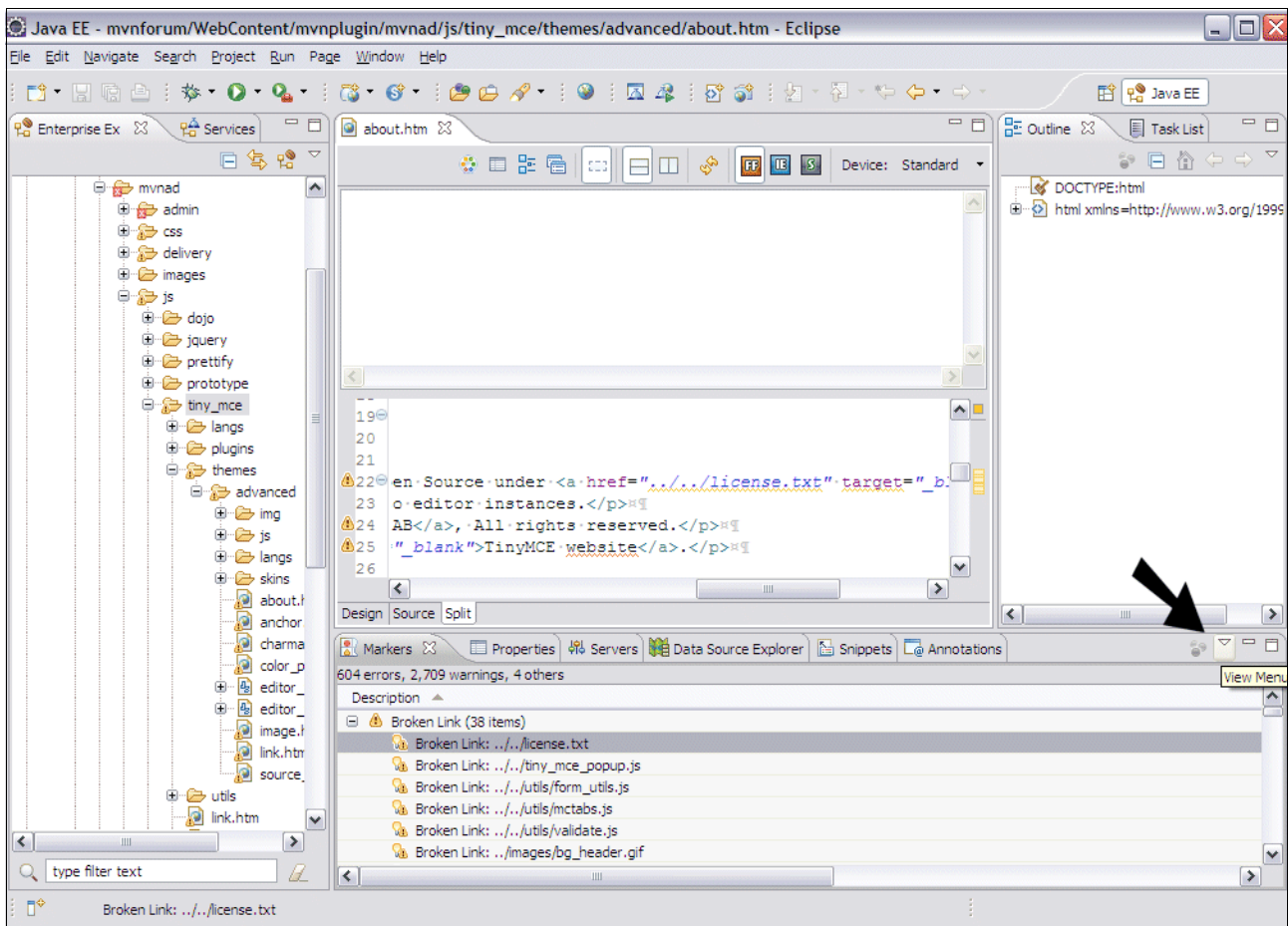


Figure 9-7 The location of the View menu

2. Click **New** to create a configuration for the Markers view.

3. In the Types listing, find and clear the following items to suppress all the existing warnings:
  - **Broken Link**
  - **HTML Problem**
  - **JSP Problem**
  - **JavaScript Task**
  - **XML Problem**

### Creating the DB2 database

The MvnForum application is compatible with nine database managers, including DB2. The table creation scripts are provided by MvnForum. To create the DB2 database, complete the following steps:

1. Create a database instance with the default options.
2. Get the mvnForum\_DB2.sql file from the <mvnforum\_app>\sql\_mvnforum directory.
3. Open a DB2 command window and run the commands that are listed in Example 9-4 to create the database and tables.

*Example 9-4 Database creation commands*

---

```
db2 create db MVNFORUM
db2 connect to MVNFORUM user db2admin using your_password
db2 -tvf mvnForum_DB2.sql
```

---

## 9.5.3 Deploying the application in Tomcat

To deploy the application in Tomcat, complete the steps that are listed in the following sections.

### Configuring the data source

To deploy the application to Tomcat, you must configure the data source that the application requires. To do this task, complete the following steps:

1. In the MvnForum project in Eclipse, open the **Java Resources** section and then open the src folder.
2. Double-click the mvncore.xml file. If it opens in Design view, switch to the Source view.
3. Update the properties that are listed in Example 9-5. The example shows the values that we used in our example; you might need to make adjustments to these values, such as for host name, user, and password, for your environment.

*Example 9-5 Changing the mvncore.xml file code snippet*

---

```
<driver_class_name>com.ibm.db2.jcc.DB2Driver</driver_class_name>
<database_url>jdbc:db2://localhost:50000/MVNFORUM</database_url>
<database_user>db2admin</database_user>
<database_password>your_password</database_password>
```

---

4. Save mvncore.xml.

## Configuring Tomcat to run MvnForum in Eclipse

The MvnForum application has certain functions, such as sending activation emails to users, that rely on some specific properties of the deployment. They rely on the application being deployed to a folder called `webapps` and on the application having a current working directory that is in the same directory as the `webapps` directory. Although these properties do not cause any issues when they are deployed from the command line, they cause issues when the application is deployed from Eclipse.

**Migrating to the Liberty profile:** As is documented later in this chapter, you encounter similar issues when you migrate the application to the Liberty profile. Using relative paths can lead to inconsistent behavior across different application servers, especially when these paths involve paths outside of the application. Avoid this situation if possible.

### *Configuring the deployment path in Eclipse*

By default, applications are deployed to a directory that is called `webapps` in Tomcat. However, when deploying applications through Eclipse, the deployment path is set to `wtwebapps`. This setting causes issues with the MvnForum application.

To modify the deployment path, complete the following steps:

1. Ensure that the Tomcat server is running.
2. Remove all applications from the Tomcat server by right-clicking the server in the Servers view, selecting **Add and Remove...** → **Remove All**, and then clicking **Finish**.
3. Double-click the Tomcat server in the Servers view.
4. In the Overview, change the deployment path under the Server Locations window from `wtwebapps` to `webapps`.
5. Save your changes and restart the server.

The application is deployed to a folder called `webapps` instead, which is what is expected by the application.

### *Configuring the current working directory for the Tomcat server*

Apache Tomcat 7.0.27 sets the current working directory of a server process that is based on the current working directory of the process that started it. When you start the server from the command line, it is likely that this server process is in the `bin` directory of the Apache Tomcat 7.0.27 installation. However, when the server process is started within Eclipse, the current working directory is configured for each run configuration that is defined for the server, with the default being the location of the Eclipse application. To change this setup, the server configuration and the application run configuration must both be altered.

Complete the following steps:

1. Ensure that the Tomcat server is started and has no applications added.
2. Double-click the server in the Servers view to open the **Overview**.
3. From the Server Locations window, click **Use custom location (does not modify Tomcat installation)**.
4. Set the Server path to be a location on the file system, for example, `C:\tomcatServerLocation`.
5. Save the changes.
6. From the menu, click **Run** → **Run Configurations**.
7. Click **Tomcat v7.0 Server at localhost** and click the **Arguments** tab.

8. Change the **Working directory** radio button to Other, and click **FileSystem** to navigate to the webapps folder that is created inside your server location (C:\tomcatServerLocation\webapps in our example).
9. Click **Apply** → **Close** and then restart the Tomcat server.

To deploy the application, right-click the project in Eclipse, select **Run As** → **Run On Server**, and restart the server if prompted. After Tomcat and the application start, a browser opens and shows the MvnForum application.

## 9.5.4 Testing the application in Tomcat

Log in to the application using the user name admin and password admin. In our example, we test the application manually, to verify all the functionality of the application. We log in to the application, and change the profile avatar picture and email address.

After that simple test, we check the log files and do not find any errors. This simple test was implemented to ensure that the application was up and running without configuration problems. For a production environment, perform more thorough testing.

You have successfully built, deployed, and tested MvnForum running on Apache Tomcat 7.0.27, which is its original environment. Having the application working properly on the source environment is the starting point for migrating it to WebSphere Application Server V8.5 Liberty profile.

## 9.5.5 Migrating the MvnForum application

This section describes the steps necessary for migrating the application from the source environment to the WebSphere Application Server V8.5 Liberty profile environment. We use the same database in both environments.

The migration steps for this application do not include source code changes; instead, modifications are made to the `mvncore.xml` file to adjust the WebSphere Application Server data source, server address, and web container port number. In addition, you modify the `mvnforum.xml` file to support the altered current working directory. You must configure the Liberty profile server to install the application and to add support for the required data sources. Then, test the application.

### Changing the default connection pool to the WebSphere Application Server data source

When we performed the migration for our example, we moved in small steps. We changed the minimum number of properties before we moved forward to the next step. We tried to make sure that the baseline worked well (the application runs in Tomcat) and then we tried to deploy the same configuration to the Liberty profile server.

This technique is a useful one to isolate changes and helps control the migration. However, in this book, we are not taking you through all the steps in the exact order we used (we simplified the process and omitted the trial and error). Instead, we show you how to make all changes first and then build and deploy the application in one step. Usually, this process is an iterative one, but to simplify the description, we are not showing all intermediate deployments and builds that we performed.

Considering that the application works with DB2 using the default connection pool, you must change the connection pool mechanism. The default connection pool that is provided in MvnForum is not designed for production use, so you must use the WebSphere provided connection pool.

There are many benefits to using the WebSphere connection pool implementation. The connections are obtained through JNDI from the WebSphere data source and automatically participate in the container-managed transactions of the server.

To implement these changes, you must update the `mvncore.xml` configuration file. To use WebSphere Application Server data sources, you must change two XML parameters. The file is in the `src` directory of the Java Resources window in the MvnForum project in Eclipse. The required updates are described in Example 9-6.

*Example 9-6 Define data source in mvncore.xml*

---

```
<?xml version="1.0" encoding="UTF-8" ?>
  <baseElement>
    <dboptions>
      ...
      <use_datasource>true</use_datasource>
      ...
      <datasource_name>jdbc/dsmvnforum</datasource_name>
      ...
    </dboptions>
  </baseElement>
```

---

**Defined properties:** We have only two properties that are defined here. These properties are the only ones that are needed by MvnForum to get connected to the database. The other properties can be left as the defaults.

When you develop an application, you generally do not know the name of the data source on the target application server. In your code, you do not look up the data source directly. Instead, you look up the resource reference in the `java:comp/env` namespace file. The JNDI resource `jdbc/dsmvnforum` is configured in the next section.

You also must modify the context path and server path to fit the WebSphere Application Server webcontainer context and port number. In our example, the webcontainer uses port 9080 and the default context path is `/mvnforum/mvnforum`. The trusted domains entry should also be changed to fit your environment settings, as shown in Example 9-7.

*Example 9-7 Define paths and trusted domains in mvncore.xml*

---

```
<paramoptions>
  <context_path>/mvnforum/mvnforum</context_path>
  <server_path>http://localhost:9080</server_path>
</paramoptions>
...
<mvncoreconfig>
  ...
  <allow_http_referer_prefix_list>
    http://localhost:9080;http://127.0.0.1:9080
  </allow_http_referer_prefix_list>
  ...
</mvncoreconfig>
```

---

## Configuring the application for running in the Liberty profile

One difference for the MvnForum application between Apache Tomcat 7.0.27 and the Liberty profile is the current directory that is used when you start the server. Although Tomcat uses the current directory of the process that started it, the Liberty profile changes the current directory to the root directory of the server that is started. The MvnForum application uses an entry in the `mvnforum.xml` file to determine where some of the other configuration files are on the file system. These other files include the template files that are used by the application to generate emails.

You must update the `mvnforum.xml` file to provide the location of these files when the application is installed using the Liberty profile, as shown in Example 9-8. In our example, we provide the location where the files are when the application is installed to the Liberty profile. This file is found in the same location in Eclipse as the `mvncore.xml` file.

*Example 9-8 Updating `mvnforum.xml` with the location of the `mvnforum_home` directory*

```
<mvnforum_home>apps/MvnForum.war/WEB-INF/mvnForumHome</mvnforum_home>
```

**File locations:** For this scenario, we use the file location to which the files are placed by Eclipse. This scenario assumes that you use the default project name when you import the WAR file into Eclipse. If you choose to change the project name, you must update the `mvnforum.xml` file and use the new project name. By using this technique, you can avoid using an absolute path to the configuration file, which prevents the application from working if the location of the Liberty profile server is moved.

## Checking for migration problems with Application Migration Tool

Although we have not imported the Java source code into Eclipse in our example, you can still use the Application Migration Tool to check for potential migration problems with the JSP and XML files that are part of the MvnForum application. To do this task, complete the following steps:

1. Right-click the `mvnforum` project in the Enterprise Explorer view and select **Software Analyzer** → **Software Analyzer Configurations....**
2. Click **Software Analyzer** and then click **New launch Configuration** to create a Software Analyzer configuration (Figure 9-8).

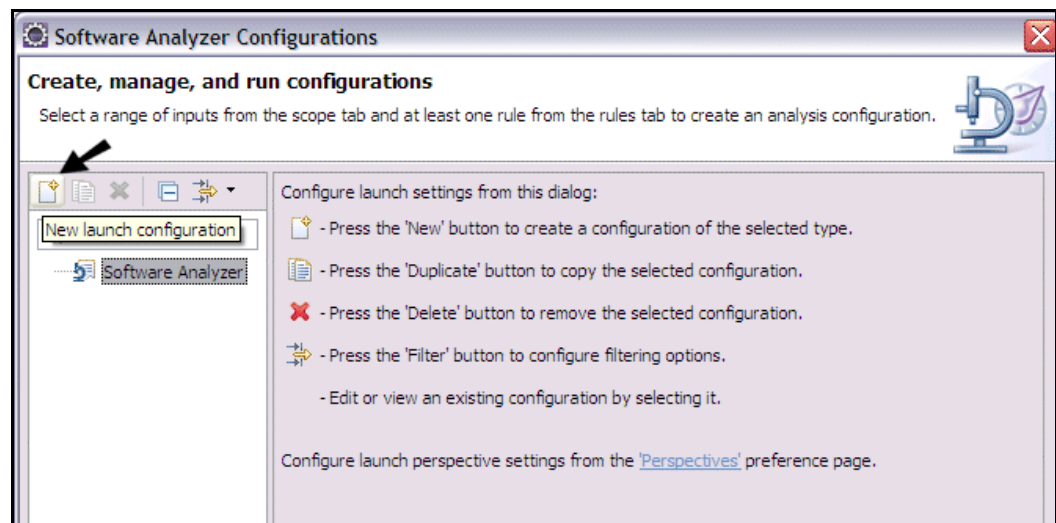


Figure 9-8 Software Analyzer new configuration

3. Provide a name, such as MvnForum Migration.
4. In the new Software Analyzer configuration, change the scope to cover MvnForum by selecting the **Analyze selected projects** radio button and then selecting the MvnForum project.
5. Now change to the **Rules** tab in the Software Analyzer Configurations wizard and select the **Apache Tomcat Application Migration** rule set. Click **Set...**
6. In the Rule set configuration dialog box, set the Target application server to WebSphere Application Server V8.5, set the Source Java version to Java 1.4, and set the Target Java version to Java 6.
7. Click **OK** to close the dialog box.
8. Now click **Apply** and **Analyze** to run the analysis.

The analysis takes a few seconds to run, but reports no problems. This situation indicates that there are no detected migration problems in the JSP and XML files of the MvnForum application.

### Running Application Migration Tool against the MvnForum source code

In our example, we decided not to import the source code of the application to avoid altering the established build process. However, as a separate exercise, it is useful to run the Application Migration Tool on the source code of the MvnForum application. The main reason to do this task is because it is considered the preferred practice for using Application Migration Tool. Scanning the source code allows a better examination of the application, significantly increasing the chance of discovering issues that might otherwise be difficult to trace.

We ran Application Migration Tool against both the `mvnforum` and `myvietnam` source directories with the same settings that we used on the binary application. This action highlighted a number of issues that we needed to investigate to ensure that they did not cause issues in our migration.

When you investigate an issue raised by Application Migration Tool, it is important to examine the issue so that it can be understood and the potential impact that it might have on the migration can be determined. We found a number of issues that required researching. These include changes to the Java EE specification, use of specific vendor packages, and missing metadata.

We discovered that the `myvietnam` source code uses a number of JNDI lookups that are not listed in a deployment descriptor. Defining resource references in the deployment descriptor is a preferred practice, as it enables the separation of JNDI names from the application and the server configuration. In this example, we were satisfied that the required data sources were already available and this situation would not cause any issues in our local deployment. In a production environment, it might be preferable to include the resource references so that the application deployment is easier in the future.

We also noticed issues with serializable classes. Serialization can be used in database persistence or network communication applications. Changes in serialization between Java 1.4.2 and Java 5 mean that there is a risk of problems when you migrate an application. In our case, we are not using a persistence model such as JPA (instead, Direct Access Objects are used to retrieve data from a database). We are also aware that our application does not require transmission of Java objects outside the JVM. As such, we determine that these issues do not affect our migration. In a production environment, where historic data might exist and communications between different systems might be more prevalent, this situation must be considered carefully, and the appropriate steps must be taken to ensure a successful migration.

By investigating the issues that are raised, we could determine the areas likely to cause issues to our migration. After we analyzed the issues, we decided we were happy to use the application without any changes. We see this situation as a beneficial exercise, as it enabled us to act to address issues before we attempted a deployment.

When you migrate any application, any potential issues that are highlighted by the Application Migration Tool should be investigated and researched thoroughly. These actions provide a good understanding of potential problems and how they should be addressed.

## 9.5.6 Configuring the target environment

There are two main tasks that are required to configure the application:

- ▶ Configure the developer tools to allow direct deployment of applications.
- ▶ Configure the data sources that are required to communicate with back-end databases.

### Configuring direct deployment of applications

As MvnForum reads files from the Liberty profile server's apps directory, you must configure the Liberty profile server definition so that all of the application's files are copied to the server. To do this task, select the Liberty profile server in the Servers view and click **Open**. The overview page for the server opens. Clear the **Run applications directly from the workspace** option and save the overview.

**Using options:** The **Run applications directly from the workspace** option allows file changes to be immediately reflected by a running server without the need for the file changes to be copied by the developer tools to the server. With this option not selected, any file changes take a short time to be copied by the developer tools to the server.

### Creating a data source

Because we configured the MvnForum application to use a data source to access the database in our example, we use the developer tools to create the data source.

Complete the following steps:

1. From the Servers view, expand server1 and double-click the **Server Configuration**. This action opens the server.xml file. If it opens in Source view, switch to the Design view using the tab in the lower left of the view.
2. You must enable JDBC support in the server by adding it as a feature in the server's configuration. Click **Feature Manager** and then click **Add...** in the Feature Manager details window. Select jdbc-4.0 from the list of possible features and click **OK**.
3. We are using the DB2 JDBC driver for our application, which requires two driver files, db2jcc4.jar and db2\_license\_cu.jar, to be referenced. Complete the following steps:
  - a. Create a Fileset Service by right-clicking **Server Configuration** in the Configuration Structure column. Enter the values that are shown in Table 9-2.

Table 9-2 Parameters for the Fileset Service

Parameter	Value
ID	DB2DriverFileSet
Base directory	Location of db2jcc4.jar and db2jcc_license_cu.jar
Includes pattern	db2jcc4.jar and db2jcc_license_cu.jar



- b. Create a Shared Library by right-clicking **Server Configuration** in the Configuration Structure column. Set the ID to DB2DriverSharedLibrary, and add a reference to DB2DriverFileSet.
  - c. Create a JDBC Driver by right-clicking **Server Configuration** in the Configuration Structure column. Set the ID field to DB2JDBCdriver, and from the Shared Libraries field, select DB2DriverSharedLibrary from the drop-down menu.
4. Create a Data Source by right-clicking **Server Configuration** in the Configuration Structure column. Provide an ID of MvnForumDS, a JNDI name of jdbc/dsmvforum, and set the JDBC Driver to DB2JDBCdriver using the drop-down menu.
  5. Right-click **Data Source: jdbc/dsmvforum** and add a Properties for DB2 JCC element. The values that we used are shown in Table 9-3.

Table 9-3 Parameters for the Properties for DB2 JCC

Parameter	Value
Database name	MVNFORUM
Server name	localhost
Port number	50000
User	db2admin
Password	your_password

## 9.5.7 Deploying the application in the Liberty profile

There are two methods to deploy an application to a Liberty profile server. You can use the `dropins` folder or the `apps` folder. Both these mechanisms are acceptable in production. The `dropins` folder provides a deployment that does not require configuration in `server.xml` (application files are placed into the `dropins` folder in the root of the Liberty profile server), while the `apps` folder requires configuration in `server.xml`, but offers application bindings and classloader services. Both these mechanisms are available in development as well, with the developer tools deploying an application to the `apps` directory.

*WebSphere Application Server V7: Competitive Migration Guide*, SG24-7870 provides instructions on deployment to a full profile server. These steps are still accurate and should be used when you deploy to the WebSphere Application Server V8.5 full profile. For more information, see that publication.

In this case, we have only one application, which we stored as a project in Eclipse and all the necessary resources are already configured in the server. For this reason, the application is deployed through the developer tools.

To deploy the application, right-click `server1` in the Servers view and select **Add and Remove**. From here, select the `MvnForum` web project and click **Add**. Click **Finish** and the application is deployed to the Liberty profile server. You can now start the server by right-clicking `server1` in the Servers view and selecting **Start**.

You can then access the application by using the following URL:

`http://localhost:9080/MvnForum`

## The Liberty profile server configuration and deployment

When the developer tools are used to make configuration changes, they alter a single XML file that is called `server.xml`, which is in the root of the Liberty profile server (by default, this location is `<liberty_home>\usr\servers\<server_name>`). The contents of this file should be similar to the contents shown in Example 9-9, although the file does not include other applications that might be installed.

*Example 9-9 Contents of a Liberty profile server configured to run the MvnForum application*

---

```
<featureManager>
  <feature>jsp-2.2</feature>
  <feature>jdbc-4.0</feature>
</featureManager>
<fileset id="DB2DriverFileset" dir="<DB2_driver_location>" includes="db2jcc4.jar,
db2jcc_license_cu.jar"/>
<library filesetRef="DB2DriverFileset" id="DB2DriverSharedLibrary"/>
<jdbcDriver id="DB2Driver" libraryRef="DB2DriverSharedLibrary"/>
<dataSource jdbcDriverRef="db2Driver" jndiName="jdbc/dsmvnforum">
  <properties.db2.jcc databaseName="MVNFORUM" password="your_password"
user="db2admin"/>
</dataSource>
<application id="MvnForum" location="MvnForum.war" name="MvnForum" type="war"/>
```

---

To run the MvnForum application without using the developer tools, configure the `server.xml` file of your server (in our example, `server1`) to contain the contents listed in Example 9-9. Extract the `MvnForum.war` file into `<liberty_home>/usr/servers/server1/apps/MvnForum.war` (ensuring that the directory name has “.war” at the end). Start the server by running the following command:

```
<liberty_home>/bin/server start server1
```

### 9.5.8 Testing the application in the Liberty profile

You are ready to test the application as you tested it when it was deployed to Tomcat.

Log in to the application using the admin user and change the avatar picture and email address. After that simple test, check the log files; you should not find any errors.

Once again, this test is not a complete test, and it is not used for a production environment, but it is good enough to know that the application successfully migrated.

## 9.6 Easy JSP Forum migration

Easy JSP Forum is an open source project that is hosted by SourceForge and licensed under the terms of the Sun Public License. This license has the approval of the Free Software Foundation (FSF) as a no charge software license, and by the Open Source Initiative (OSI) as an open source license. You can modify the application and distribute the source code by using software under the GPL license. Easy JSP Forum is a J2EE 1.4 application that demonstrates the capabilities of this platform.

Easy JSP Forum implements the JSP 2.0 and Servlet 2.4 specifications. In the distribution code, the application connects to a Microsoft Access database under an ODBC data source. In this example, we migrate the Easy JSP Forum to a DB2 database under a WebSphere Application Server data source using a type 4 driver. Easy JSP Forum is included in this book for the following reasons:

- ▶ It is based on open source frameworks.
- ▶ It was designed to run on multiple application servers.
- ▶ It is not specifically designed to run on WebSphere Application Server V8.5 Liberty profile.
- ▶ It is not set up to use DB2 back-end systems.

Easy JSP Forum is an interesting migration example because it is implemented with several technologies and requires both database adjustments and source code updates.

## 9.6.1 Migration approach

To migrate the application to WebSphere Application Server V8.5 Liberty profile, we used WebSphere Application Server V8.5 Liberty profile Developer Tools in our example.

## 9.6.2 Configuring the source environment

This section describes all the required steps for setting up the source environment. Follow the instructions in 9.2, “Application Migration Tool - Apache Tomcat to WebSphere” on page 275 to install the software that is needed in the source environment.

Easy JSP Forum is designed to run on Tomcat 5.x. To run it in Apache Tomcat 7.0.27, no extra steps are required. The distribution instructions show examples in Tomcat 5.x, but you should follow the steps exactly. In our example, we do not migrate to IBM DB2 Universal Database 10.1 in the source environment, but used Microsoft Access, as it is supported by Easy JSP Forum by default.

### Obtaining the Easy JSP Forum application

Before you start the migration, obtain the source files for the Easy JSP Forum application. You can download the `easyjspforum.zip` file with all source files from the following web page:

<http://sourceforge.net/projects/easyjspforum/files/>

After the application is downloaded, it is extracted and imported into Eclipse.

### Importing the Easy JSP Forum application into Eclipse

The Easy JSP Forum is not distributed as a WAR or an EAR file and is not compatible with the folder standards of the Java EE specification. The application cannot be imported directly into Eclipse. Instead, create a Dynamic Web Project in Eclipse and add the required files from the Easy JSP Forum application. To do so, complete the following steps:

1. Open Eclipse.
2. Click **File** → **New** → **Dynamic Web Project**.
3. Enter a project name. We entered `forum` as the project name for our example. Ensure that the Target run time is set to Apache Tomcat v7.0. The other values can be left as the defaults.
4. Click **Finish**.
5. In the Java EE perspective, right-click the `WebContent` folder in the `forum` project and select **Import...** Expand the **General** list, select **File System**, and click **Next**.

6. Using the **Browse** button, navigate to the directory where you extracted the easyjspforum.zip file.
7. Select all the folders under the forum folder except for the beans and WEB-INF folders. Also, select the index.jsp and popup.jsp files, as shown in Figure 9-9, and click **Finish**.

**The WEB-INF folder:** In most cases, the WEB-INF folder is included in the list of files to import, as it often contains the web.xml deployment descriptor file. However, in this case, the file is not included in the application. During migration, you must create and edit a web.xml file, so you generate one when you must.

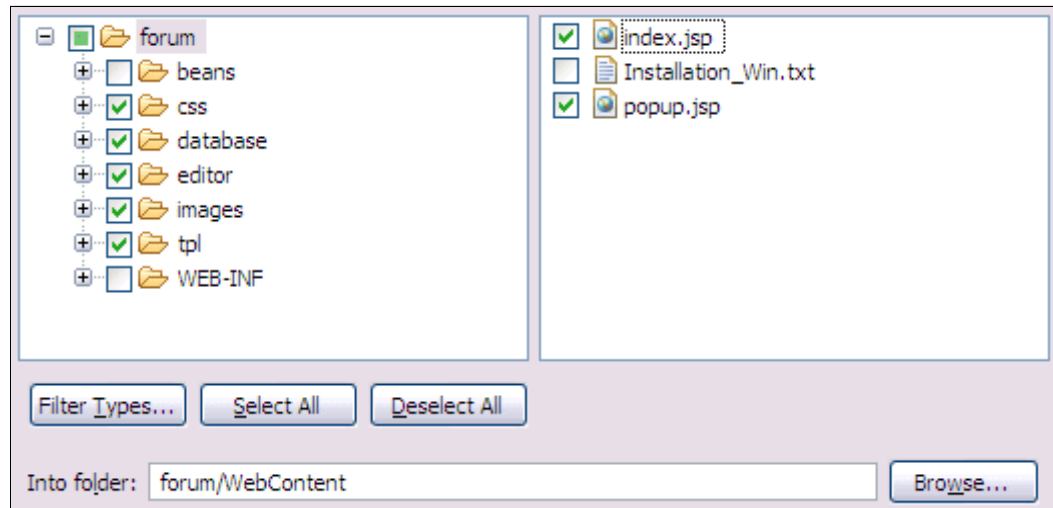


Figure 9-9 Files selection - do not select the forum folder itself

8. Now open the forum project and right-click the src folder in Java Resources (Figure 9-10) and select **New** → **Package**. Set the package name to forum and click **Finish**.

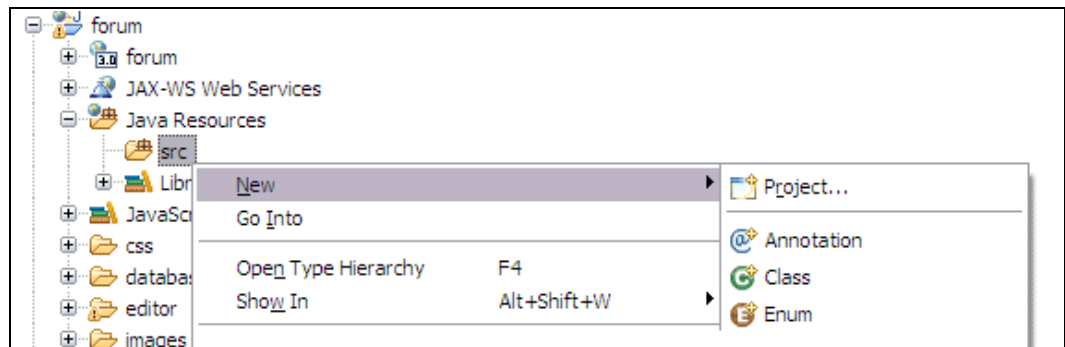


Figure 9-10 Creating a package in the src directory

9. Right-click the new package and select **Import...**. Expand the **General** list, select **File System**, and click **Next**.
10. Using the **Browse** button, navigate to the directory where you extracted the easyjspforum.zip file.

11. Select all the Java files in the beans folder (Figure 9-11) and click **Finish**.

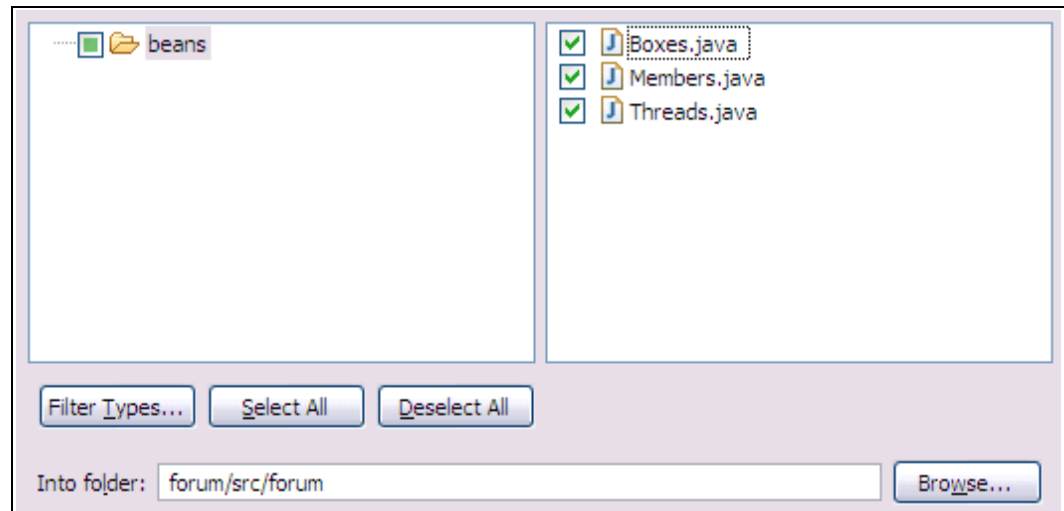


Figure 9-11 Files selection from beans directory - do not select the beans directory itself

12. This action updates the package to forum.beans. To switch this setup back, right-click the forum.beans package and select **Refactor** → **Rename...** Set the new name to **forum** and click **OK**.

The application is now successfully created in Eclipse.

## Configuring the database

This section describes how to configure the Easy JSP Forum application to access the Microsoft Access database that is supported by default. No changes are required in the source code of the application.

To configure Easy JSP Forum to use the Microsoft Access database, complete the following steps:

1. Open the Data Sources configuration menu. Depending on whether you have a 32-bit or a 64-bit system, this process is different:
  - For 32-bit Windows systems, click **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Data Sources**.
  - For 64-bit Windows system, you must run **odbcad32.exe** from `C:\Windows\SysWOW64\odbcad32.exe`.
2. Click the **System DSN** tab of the ODBC dialog box.
3. Click **Add**. Select **Microsoft Access (\*.mdb)** and click **Finish**.
4. In the Database area, click **Select**. Go to `<source_home>\database\Forum.mdb`.
5. In the Data Source Name field, enter **Forum**.
6. Click **OK** twice to leave the ODBC Data Source Administrator window.

### 9.6.3 Deploying the application in Tomcat

As the application is now in Eclipse, it can be deployed from here directly to Tomcat. To do this task, complete the following steps:

1. Right-click the forum project in Eclipse and select **Run As** → **Run On Server...**
2. Click **Finish**.

### 9.6.4 Testing the application in Tomcat

The Easy JSP Forum has no automated tests that are implemented with JUnit, JUnitEE, HttpUnit, or any other automated test tool. Therefore, you must test the application manually.

In our sample test, we browsed the application, signed in using admin as both the user and password, and edited the members (tasks such as changing the last name and email fields). We created a forum and posted threads on it and replies to the same thread. We change the current admin profile. Those simple procedures took us through the application with no errors. We checked the log files and did not find any errors or warnings.

You successfully built, deployed, and tested Easy JSP Forum in Apache Tomcat 7.0.27, which is its original environment. Having the application working correctly on the source environment is the starting point for migrating into WebSphere Application Server V8.5 Liberty profile.

### 9.6.5 Migrating the Easy JSP Forum application to the Liberty profile

This section describes the necessary steps for migrating the application from Tomcat to the Liberty profile. The steps of this migration involve IBM DB2 Universal Database 10.1 database creation, WebSphere Application Server data source configuration, source code changes to connect to WebSphere Application Server data sources, and SQL syntax fixes.

The last part of this section describes how to configure the resources that are required by the Easy JSP Forum to run in WebSphere Application Server V8.5 Liberty profile, and how to deploy and test the migrated application.

#### Creating the DB2 database

The Easy JSP Forum comes with Microsoft Access database. The IBM DB2 Universal Database 10.1 is chosen as the database back end for the migration. There is no database creation script that is provided with the application. In our scenario, we create the database and tables manually. You can also use tools to export the Access database to DB2. There is an IBM developerWorks® topic, *Migrating a Microsoft Access 2000 Database to IBM DB2 Universal Database 7.2*, that provides more information. The topic can be found at the following web page:

<http://www.ibm.com/developerworks/data/library/techarticle/alazzawe/0112a1azzawe2.html>

To create the database and connect to it, run the commands that are shown in Example 9-10.

*Example 9-10 Creating the EasyJSP database*

---

```
db2 create db EASY
db2 connect to EASY user db2admin using your_password
```

---

Next, create a file that is called `easyJSP_DB2.sql` and copy the data in Example 9-11 into the file. This action creates the data structure and the initial sample data. After this file is populated, run it from the DB2 prompt by running the following command:

```
db2 -tvf easyJSP_DB2.sql
```

*Example 9-11 Creating the data structure and sample data of the EasyJSP database*

---

```
-----
-- Creation commands for table "BOXES"
-----
CREATE TABLE "BOXES" (
    "BOX_NAME" VARCHAR(100) ,
    "SORT_DESC" VARCHAR(100) ,
    "BOX_ID" INTEGER ,
    "MEMBER_ID" INTEGER )
    IN "USERSPACE1" ;

-----
-- Creation commands for table "LEVELS"
-----
CREATE TABLE "LEVELS" (
    "GRADE" VARCHAR(100) ,
    "MIN_POST" INTEGER ,
    "MAX_POST" INTEGER ,
    "DESCRIPTION" VARCHAR(100) )
    IN "USERSPACE1" ;

-----
-- Creation commands for table "MEMBERS"
-----
CREATE TABLE "MEMBERS" (
    "MEMBER_ID" BIGINT NOT NULL ,
    "USERNAME" VARCHAR(100) NOT NULL ,
    "PASSWORD" VARCHAR(100) NOT NULL ,
    "FIRSTNAME" VARCHAR(100) NOT NULL ,
    "LASTNAME" VARCHAR(100) NOT NULL ,
    "EMAIL" VARCHAR(100) NOT NULL ,
    "REGDATE" VARCHAR(100) NOT NULL ,
    "TYPE" VARCHAR(100) NOT NULL )
    IN "USERSPACE1" ;

-----
-- Creation commands for table "THREADS"
-----
CREATE TABLE "THREADS" (
    "THREAD_ID" BIGINT NOT NULL ,
    "BOX_ID" BIGINT NOT NULL ,
    "PARENT_ID" BIGINT NOT NULL ,
    "MEMBER_ID" BIGINT NOT NULL ,
    "SUBJECT" VARCHAR(100) NOT NULL ,
    "POST_TEXT" VARCHAR(1000) NOT NULL ,
    "POST_DATE" BIGINT NOT NULL )
    IN "USERSPACE1" ;
INSERT INTO MEMBERS VALUES
(1,'admin','admin','Forum','Admin','fxavier@br.ibm.com','Fri May 07 10:44:38 ICT
2010','Administrator');
INSERT INTO MEMBERS VALUES (2,'mod','mod','Forum','Moderator','dt@test.com','Fri
May 07 10:44:38 ICT 2010','Moderator');
```

```
INSERT INTO MEMBERS VALUES
(3, 'member', 'member', 'Forum', 'Member', 'dt@test.com', 'Fri May 07 10:44:38 ICT
2010', 'Member');
```

---

The DB2 database setup is now complete.

## Altering Easy JSP Forum to use data sources

The Easy JSP Forum code is not designed to use data sources. A source code modification is required to access the DB2 database. The application contains three classes that perform all operations:

- ▶ Members.java
- ▶ Threads.java
- ▶ Boxes.java

Add the snippet that is shown in Example 9-12 before these class definitions.

*Example 9-12 Adding the required packages to access the data source*

---

```
import javax.naming.*;
import javax.sql.*;
```

---

To connect to the data source, you must change the connection parameters. Example 9-13 shows how you should perform this operation.

*Example 9-13 Changing the connection parameters*

---

**Comment out all entries of the following lines:**

```
//      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
//      Connection con = DriverManager.getConnection("jdbc:odbc:Forum");
```

**Add the following lines after the commented out lines:**

```
Context ctx=new InitialContext();
DataSource ds=(DataSource)ctx.lookup("jdbc/easy");
Connection con=ds.getConnection();
```

---

During our tests, SQL syntaxes did not work when they accessed the DB2 database. You must fix the queries so that the application does not show an exception. The problem in our case was the square brackets symbol between the fields. In Example 9-14, we fixed this problem by opening the files and removing the square brackets in the SQL syntaxes.

*Example 9-14 Remove the square brackets in the following SQL syntaxes*

---

File Boxes.java

```
Line 121 : String query = "INSERT INTO Boxes (box_id, member_id, [box_name],
[sort_desc]) " +
```

File Members.java

```
Line 75 : String query = "SELECT member_id FROM Members WHERE [username]=' " +
username + "'";
Line 133 : String query = "INSERT INTO Members (member_id, [username], [password],
[firstname], [lastname], [email], [regdate], [type]) " +
Line 166 :      "email=' " + email + "' WHERE [username]=' " + username + "'";
Line 211 : String query = "SELECT * FROM Members WHERE [username]=' " + username +
"'";
```



```
File Threads.java
Line 220 : String query = "INSERT INTO Threads (thread_id, parent_id, box_id,
member_id, [subject], [post_text], post_date) " +
Line 244 : String query = "INSERT INTO Threads (thread_id, parent_id, box_id,
member_id, [subject], [post_text], post_date) " +
```

---

## Checking the application with Application Migration Tool

To check if there are any other potential migration problems, scan the application with the Application Migration Tool. To do this task, complete the following steps:

1. Right-click the forum project in the Enterprise Explorer and select **Software Analyzer** → **Software Analyzer Configurations...**
2. Click **Software Analyzer** in the left pane and then click **New launch Configuration** to create a Software Analyzer configuration (see Figure 9-12).

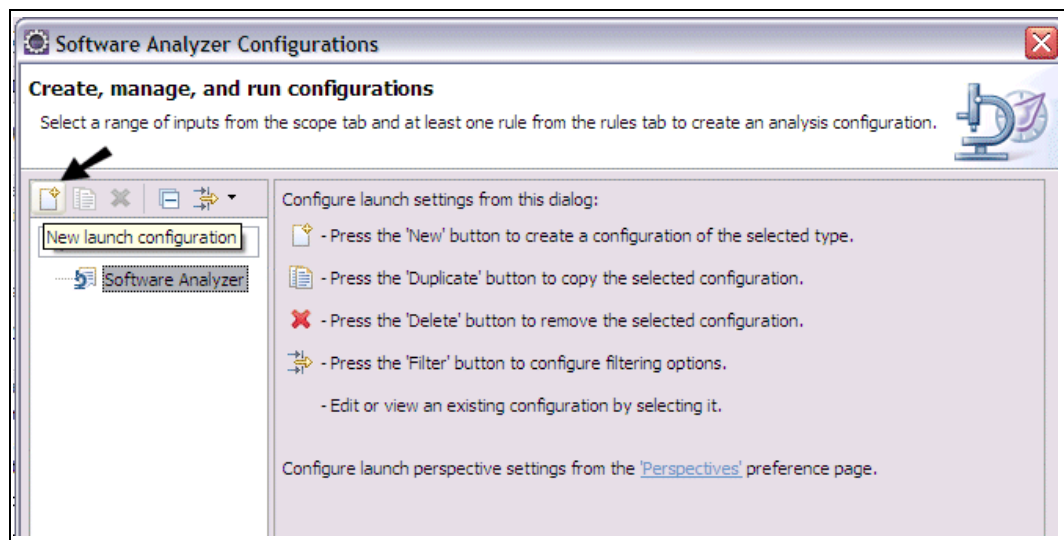


Figure 9-12 Software Analyzer new configuration

3. Enter a name for the configuration, such as Easy JSP Forum Migration.
4. In the new Software Analyzer configuration, change the scope to cover the forum project by selecting the **Analyze selected projects** radio button and then selecting the forum project.
5. Now change to the Rules tab in the Software Analyzer Configurations wizard.
6. Select the Apache Tomcat Application Migration Rule set and click **Set...**
7. In the Rule Set configuration dialog box, set the Target application server to WebSphere Application Server V8.5, the Source Java version to Java 1.4, and the Target Java version to Java 6, and click **OK**.
8. Click **Apply** and then click **Analyze**.

**Java rules:** For this scenario, we are interested only in rules specific to our application migration. However, you can select all the Java rules if you want. We found that for Easy JSP Forum, the remaining rules generated no errors and many suggestions to use preferred practices. These items are the ones that we would choose to implement if possible, as they are designed to improve performance and stability where possible.

The results of the analysis report no problems in either XML or JSP files and 31 results for the **Ensure that JNDI lookups have corresponding deployment descriptor entries** option. Investigating these results shows that they are all due to no deployment descriptor entry for the JNDI lookup on jdbc/easy. As the web.xml file that contains the resource references did not exist in the original application, you must generate this file and populate it with the required resource reference. To do this task, complete the following steps:

1. Right-click the forum project and select **Java EE Tools Generate** → **Deployment Descriptor Stub**.
2. Open the web.xml file that is created in the WebContent/WEB-INF folder of the forum project.
3. In the Source view, add the information from Example 9-15.

*Example 9-15 The resource reference to add to the deployment descriptor*

---

```
<resource-ref>
  <res-ref-name>jdbc/easy</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
  <res-sharing-scope>Shareable</res-sharing-scope>
</resource-ref>
```

---

These actions do not remove the results that are shown, but you now have the definition that you needed to check for, so no further action is required.

## Creating a data source

In our example, we alter the application to use a data source to access the database, so we use the developer tools to create the data source.

To create the data source, complete the following steps:

1. From the Servers View, expand the Liberty profile server and double-click the **Server Configuration** box. This action opens the server.xml file. If it opens in the Source view, switch it to the Design view by clicking the **Design** tab in the lower left of the view.
2. Add JDBC support to the server's configuration by clicking the **Feature Manager** and then clicking **Add...** in the Feature Manager details window. Select jdbc-4.0 from the list of possible features and click **OK**.
3. We are using the DB2 JDBC driver for our application. This application requires two driver files, db2jcc4.jar and db2\_license\_cu.jar, to be referenced. To refer to these files, complete the following steps:
  - a. Create a Fileset Service by right-clicking **Server Configuration** in the Configuration Structure column. When you create the Fileset Service, use the values that are shown in Table 9-4.

*Table 9-4 Parameters for the Fileset Service*

Parameter	Value
ID	DB2DriverFileSet
Base directory	Location of db2jcc4.jar and db2jcc_license.jar
Includes pattern	db2jcc4.jar and db2jcc_license_cu.jar

- b. Create a Shared Library by right-clicking **Server Configuration** in the Configuration Structure column. Set the ID to DB2DriverSharedLibrary, and add a reference to DB2DriverFileSet.
- c. Create a JDBC Driver by right-clicking **Server Configuration** in the Configuration Structure column. Set the ID field to DB2JDBCdriver, and from the Shared Libraries field, select **DB2DriverSharedLibrary** from the drop-down menu.
4. Create a data source by right-clicking **Server Configuration** in the Configuration Structure column. Provide an ID of ForumDS, a JNDI name of jdbc/easy, and set the JDBC Driver to DB2JDBCdriver using the drop-down menu.
5. Right-click the data source jdbc/easy and add a Properties for DB2 JCC element. Provide the details for this element. See Table 9-5 for the values in our example.

Table 9-5 Parameters for the Properties for DB2 JCC

Parameter	Value
Database name	EASY
Server name	localhost
Port number	50000
User	db2admin
Password	your_password

## Deploying the application

There are two methods to deploy an application to a Liberty profile server. You can use the `dropins` folder or the `apps` folder. Both these mechanisms are acceptable in production. The `dropins` folder provides a deployment that does not require configuration in `server.xml` (application files are placed into the `dropins` folder in the root of the Liberty profile server), while the `apps` folder requires configuration in `server.xml`, but offers application bindings and classloader services. Both these mechanisms are available in development as well. When you use the developer tools, you deploy the application using the `apps` directory.

**Deploying Easy JPSP Forum:** The previous version of this book, *WebSphere Application Server V7: Competitive Migration Guide*, SG24-7870, provides instructions to deploy the Easy JSP Forum application to WebSphere Application Server V7.0. These steps are still accurate and should be used if you deploy the WebSphere Application Server V8.5 full profile.

In this case, the application is created as a project in Eclipse and all the necessary resources are already configured in the server. For this reason, the application is deployed through the developer tools.

To deploy the application, right-click **server1** in the Servers view, and select **Add and Remove**. From here, select the forum project and click **Add**. Click **Finish** and the application is deployed to the Liberty profile server. You can now start the server by right-clicking **server1** in the Servers view and selecting **Start**.

You can then access the application by going to the following URL:

`http://localhost:9080/forum`

## Testing the application

Because we do not have an automated test for our example, we must test the application manually, following the same steps that we performed to test the application that is running in Tomcat. Again, we signed in using admin as the user name and password, and edited the members (such as changing the last name and email fields).

We created a forum and posted threads on it and replied to the same thread. The current admin profile was changed also. This simple procedure took us through the application with no errors. We checked the log files and did not find any errors or warnings. The results were the same, so we can say that the application was successfully migrated.

## 9.7 Summary

This chapter provided steps to migrate three applications from Apache Tomcat 7.0.27 to WebSphere Application Server V8.5 Liberty profile. To assist with this task, we used the Application Migration Tool. The toolkit provides useful instructions for simple applications, and it shows an even greater benefit when migrating applications that use many features and configurations specific to Apache Tomcat 7.0.27. In these cases, using quick fix processes and issuing highlighting enables a faster migration and a reduction in the complexity of the task. In addition to using Application Migration Tool, we also provided manual steps to address specific issues where required.

We addressed a number of issues that users might see with their applications, such as a difference between an application's current working directory under different configuration and different servers. We found that some interesting issues were uncovered by our testing process, a situation that is likely to apply to all migrations. In a production environment, you would expect a migrated application to undergo a thorough testing process to ensure that the migration is complete.



# Application Framework migration

This chapter describes the migration of two sample applications, one written for Seam and the other for Spring, to WebSphere Application Server V8.5.

This chapter describes the following topics:

- ▶ Migrating a Seam Framework application
- ▶ Migrating a Spring Framework application

**Legal disclaimer:** IBM does not warrant or represent that the code provided is complete or up-to-date. IBM does not warrant, represent or imply reliability, serviceability, or function of the code. IBM is under no obligation to update content nor provide further support.

All code is provided "as is," with no warranties or guarantees whatsoever. IBM expressly disclaims to the fullest extent permitted by law all express, implied, statutory and other warranties, guarantees, or representations, including, without limitation, the warranties of merchantability, fitness for a particular purpose, and non-infringement of proprietary and intellectual property rights. You understand and agree that you use these materials, information, products, software, programs, and services, at your own discretion and risk and that you will be solely responsible for any damages that may result, including loss of data or damage to your computer system.

In no event will IBM be liable to any party for any direct, indirect, incidental, special, exemplary or consequential damages of any type whatsoever related to or arising from use of the code found herein, without limitation, any lost profits, business interruption, lost savings, loss of programs or other data, even if IBM is expressly advised of the possibility of such damages. This exclusion and waiver of liability applies to all causes of action, whether based on contract, warranty, tort or any other legal theories.

*Third-party* software is licensed and distributed to you by the *third-party* distributors and/or respective copyright and other right holders under their terms and conditions. IBM makes no express or implied warranties or representations with respect to such *software* and provides no indemnity for such software. IBM grants no express or implied patent or other license with respect to *and* is not liable for any damages arising out of the use of *such software*.

## 10.1 Migrating a Seam Framework application

Seam is an application framework for Enterprise Java. In this section, we migrate a Seam 2 application as an example. More information about Seam can be found at the following website:

<http://seamframework.org/>

Seam distribution contains a utility called *seam-gen*, which helps you set up a Seam project from scratch. The script generates the project structure, including the skeleton code. It also can generate a fully functional create, read, update, and delete type application with a schema as a data source. We use this ability to generate our own sample application using a database schema with a single table.

We chose to migrate this application for the following reasons:

- ▶ A project structure that is created with *seam-gen* is representative of a well-designed Seam application structure. *seam-gen* is often used to start many similar real-world applications.
- ▶ It is a simple and complete application, and is good for demonstrating and focusing on framework-specific constructs without getting in to too much repetitive work for various modules and components.
- ▶ It uses older JSF 1.2 while WebSphere Application Server 8.5 support JSF 2.0, so we can demonstrate its usage using third-party JSF implementations.
- ▶ It uses Hibernate as a JPA 1.0 provider, so we can demonstrate the configuration of third-party JPA providers.

## 10.1.1 Migration approach

In this migration exercise, we use IBM Rational Application Developer for WebSphere Software as the target development and build environment. The seam-gen utility also generates an Eclipse project that uses an Ant based build, which can be imported into Eclipse and IBM Rational Application Developer for WebSphere Software. Because this alternative is already provided, we demonstrate how we can use the build cycle of IBM Rational Application Developer for WebSphere Software instead.

The migration is performed by completing the following steps:

1. Assume that DB2 is already installed. Create a simple database schema with a single table on DB2.
2. Use seam-gen to generate a Seam web application that does create, retrieve, update, and delete operations on this table.
3. Produce EAR files for JBoss and Glassfish. Import the EAR file that is built for JBoss to IBM Rational Application Developer for WebSphere Software to create the project structure. Import source files into projects that are created from source distribution.
4. Analyze and fix problems that are reported by the build environment and the Application Migration Tool (We used Application Migration Tool - JBoss AS to WebSphere for the Seam Framework application scenario and the Application Migration Tool - Oracle WebLogic to WebSphere for the Spring Framework application scenario).
5. Deploy and test the application in an integrated WebSphere Application Server V8.5 test environment.

## 10.1.2 Installing the Seam Framework

Download the JBoss Seam Framework distribution from the following website:

<http://seamframework.org/Download>

We used Version 2.2.2.Final in our environment. Extract the downloaded file to a suitable location. We used C:\jboss-seam-2.2.2.Final as the directory.

Seam depends on Apache Ant for operation. For installation instructions for Apache Ant, see 8.3.1, “Installing and configuring Apache Ant” on page 237.

## 10.1.3 Preparing the database schema

During the lab, we used a fresh installation of DB2 Express-C 10.1, which can be downloaded from the following website:

<http://www-01.ibm.com/software/data/db2/express/>

Because the seam-gen utility reverse engineers the database schema, ensure that the table you create is the only one under the current schema. If you are using a pre-existing installation, switch to a new schema before you complete the following steps:

1. Click **Start** and select **Programs** → **IBM DB2** → **DB2Copy1 (default)** → **Command Line Processor**.

2. Enter the information that is highlighted in bold in Example 10-1 into the interactive script. Replace the database's user name and password with the ones for your environment.

*Example 10-1 Code snippet*

---

(c) Copyright IBM Corporation 1993,2007  
Command line processor for DB2 Client 10.1.0

You can issue database manager commands and SQL statements from the command prompt. For example:

```
db2 => connect to sample
db2 => bind sample.bnd
```

For general help, type: ?.

For command help, type: ? command, where command can be the first few keywords of a database manager command. For example:

```
? CATALOG DATABASE for help on the CATALOG DATABASE command
? CATALOG           for help on all of the CATALOG commands.
```

To exit db2 interactive mode, type QUIT at the command prompt. Outside interactive mode, all commands must be prefixed with 'db2'.

To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

```
db2 => CREATE DATABASE TESTDB
DB20000I The CREATE DATABASE command completed successfully.
db2 => CONNECT TO TESTDB USER DB2ADMIN USING passw0rd
```

Database Connection Information

```
Database server      = DB2/NT 10.1.0
SQL authorization ID = DB2ADMIN
Local database alias = TESTDB
```

```
db2 => CREATE TABLE DB2ADMIN.BOOK ( \
db2 (cont.) => ISBN VARCHAR(13) NOT NULL, \
db2 (cont.) => TITLE VARCHAR(100) NOT NULL, \
db2 (cont.) => AUTHOR VARCHAR(100) NOT NULL )
DB20000I The SQL command completed successfully.
db2 => ALTER TABLE DB2ADMIN.BOOK \
db2 (cont.) => ADD CONSTRAINT PRIM_KEY PRIMARY KEY \
db2 (cont.) => (ISBN)
DB20000I The SQL command completed successfully.
db2 => quit
DB20000I The QUIT command completed successfully.
C:\IBM\SQLLIB\BIN>
```

---

3. Close the command line processor.



## 10.1.4 Generating a Seam application using seam-gen

Next, create a Seam web application that performs create, retrieve, update, and delete operations on the table you created. Configure seam-gen for your environment by completing the following steps:

1. Open a command prompt and change the directory to the Seam installation folder. In our example, the folder is C:\jboss-seam-2.2.1.CR1.
2. Enter **seam setup** and press Enter. This command begins an interactive script. The responses for our example are in bold in Example 10-2. Where you do not see a response, press Enter to accept the default answer for the question. Replace the database driver path, user name, and password with the ones for your environment.

### *Example 10-2 Seam setup command interactive output*

---

```
c:\jboss-seam-2.2.2.Final>seam setup
SEAM_HOME: c:\jboss-seam-2.2.2.Final
Using seam-gen sources from: c:\jboss-seam-2.2.2.Final\seam-gen
Buildfile: c:\jboss-seam-2.2.2.Final\seam-gen\build.xml

init:

setup:
  [echo] Welcome to seam-gen 2.2.2.Final :-)
  [echo] Answer each question or hit ENTER to accept the default (in brackets
)
  [echo]
  [input] Enter the directory where you want the project to be created (should
not contain spaces) [C:/Projects] [C:/Projects]

  [input] Enter your JBoss AS home directory [C:/Program Files/jboss-5.1.0.GA]
[C:/Program Files/jboss-5.1.0.GA]

  [input] Enter your JBoss AS domain [default] [default]

  [input] Enter your GlassFish V2 or V3 home directory (Ignore if you aren't d
eploying to GlassFish) [C:/Program Files/glassfish-v2.1] [C:/Program Files/glass
fish-v2.1]

  [input] Enter your GlassFish domain (Ignore if you aren't deploying to Glass
Fish) [domain1] [domain1]

  [input] Enter the project name [myproject] [myproject]

  [echo] Accepted project name as: myproject
  [input] Do you want to use ICEfaces instead of RichFaces? [n] (y, [n])

  [input] skipping input as property icefaces.home.new has already been set.
  [input] Select a RichFaces skin [glassX] (blueSky, classic, darkX, deepMarin
e, DEFAULT, emeraldTown, [glassX], japanCherry, laguna, ruby, wine)

  [input] Is this project deployed as an EAR (with EJB components) or a WAR (w
ith no EJB support)? [war] (ear, [war])
ear
  [input] Enter the base package name for your Java classes [com.mydomain.mypr
oject] [com.mydomain.myproject]

  [input] Enter the Java package name for your session beans [com.mydomain.myp
roject.action] [com.mydomain.myproject.action]
```

[input] Enter the Java package name for your entity beans [com.mydomain.myproject.model] [com.mydomain.myproject.model]

[input] Enter the Java package name for your test cases [com.mydomain.myproject.test] [com.mydomain.myproject.test]

[input] What kind of database are you using? [hsq1] ([hsq1], mysql, derby, oracle, postgres, mssql, db2, sybase, enterprisedb, h2)  
db2

[input] Enter the filesystem path to the JDBC driver jar [] []  
C:/IBM/SQLLIB/java/db2jcc.jar

[input] Enter the filesystem path to the license jar [] []  
C:/IBM/SQLLIB/java/db2jcc\_license\_cu.jar

[input] Enter the Hibernate dialect for your database [org.hibernate.dialect.DB2Dialect] [org.hibernate.dialect.DB2Dialect]

[input] Enter the JDBC driver class for your database [com.ibm.db2.jcc.DB2Driver] [com.ibm.db2.jcc.DB2Driver]

[input] Enter the JDBC DataSource class for your database [com.ibm.db2.jcc.DB2SimpleDataSource] [com.ibm.db2.jcc.DB2SimpleDataSource]

[input] Enter the JDBC URL for your database [jdbc:db2:test] [jdbc:db2:test]

jdbc:db2://localhost:50000/TESTDB

[input] Enter the database username [sa] [sa]  
db2admin

[input] Enter the database password [] []  
passwd

[input] Enter the database schema name (Enter '-' to clear previous value) [] []  
db2admin

[input] Enter the database catalog name (Enter '-' to clear previous value) [] []

[input] Are you working with tables that already exist in the database? [n] (y, [n])

[input] Do you want to recreate the database tables and execute import.sql each time you deploy? [n] (y, [n])

[propertyfile] Creating new property file: c:\jboss-seam-2.2.2.Final\seam-gen\build.properties

[echo] Installing JDBC driver jar to JBoss AS

[copy] Copying 2 files to C:\Program Files\jboss-5.1.0.GA\server\default\lib

init:

init-properties:

[echo] C:/Program Files/jboss-5.1.0.GA

validate-workspace:

validate-project:

settings:

[echo] JBoss AS home: C:/Program Files/jboss-5.1.0.GA

[echo] GlassFish home: C:/Program Files/glassfish-v2.1

[echo] Project name: myproject

```
[echo] Project location: C:/Projects/myproject
[echo] Project type: ear
[echo] IceFaces: n
[echo] Action package: com.mydomain.myproject.action
[echo] Model package: com.mydomain.myproject.model
[echo] Test package: com.mydomain.myproject.test
[echo] JDBC driver class: com.ibm.db2.jcc.DB2Driver
[echo] JDBC DataSource class: com.ibm.db2.jcc.DB2SimpleDataSource
[echo] Hibernate dialect: org.hibernate.dialect.DB2Dialect
[echo] JDBC URL: jdbc:db2://localhost:50000/TESTDB
[echo] Database username: db2admin
[echo] Database password: passw0rd
[echo]
[echo] Type 'seam create-project' to create the new project
```

BUILD SUCCESSFUL

---

3. Run **seam new-project** to generate a new project. You should see a BUILD SUCCESSFUL message.
4. Run **seam generate-entities** to generate the components for our database table. You should see a BUILD SUCCESSFUL message.
5. Run **seam deploy** to deploy the application to JBoss Application Server. You should see a BUILD SUCCESSFUL message.

Our example application is now packaged as an EAR file in C:\Projects\myproject\dist, which we defined as our project directory during script execution.

### 10.1.5 Importing the EAR file and source code to Rational Application Developer

To import the EAR file and the source code to Rational Application Developer, complete the following steps:

1. Start the Rational Application Developer for WebSphere Software by clicking **Start** → **IBM Software Delivery Platform** → **IBM Rational Application Developer 8.5** → **IBM Rational Application Developer**.

2. Start the Import wizard by clicking **File** → **Import**. Expand the **JavaEE** node in the import source tree and select **EAR file**. See Figure 10-1.

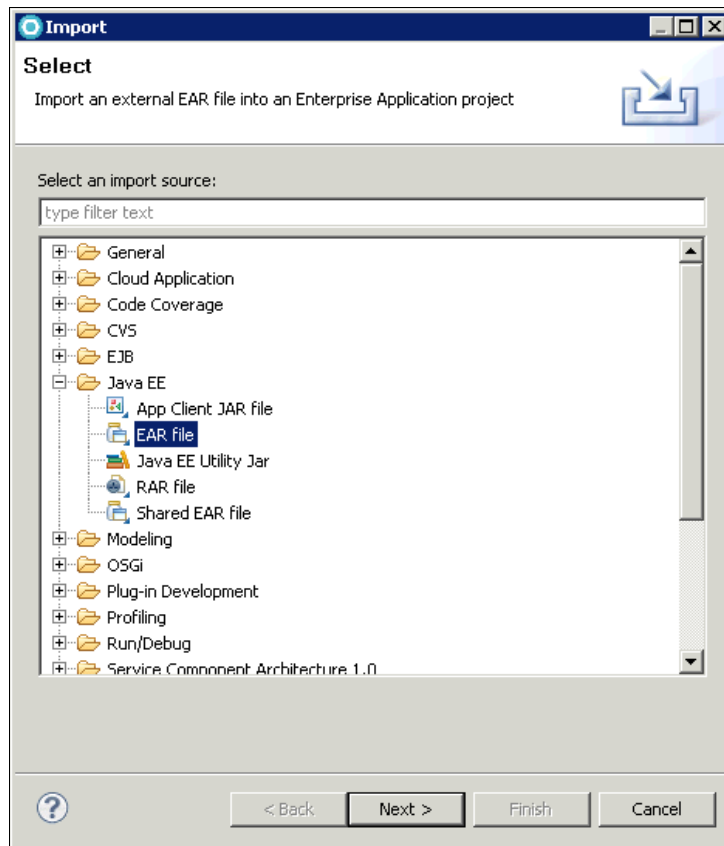


Figure 10-1 EAR import wizard

3. In the Enterprise Application Import window, browse to the location of the EAR file that is generated by seam-gen, which can be found at `C:\Projects\myproject\dist\myproject.ear`. See Figure 10-2. The location of this file is determined by the choices we made during the seam setup script execution. Click **Next**.

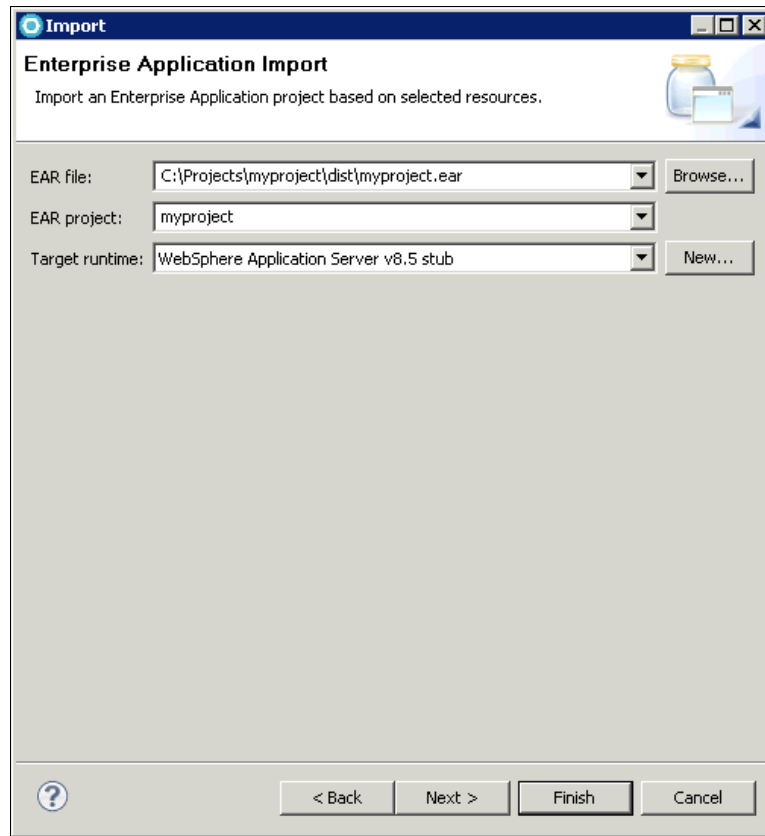


Figure 10-2 Importing *myproject.ear*

4. Click **Next** on the Utility JARs and Web Libraries window without selecting any libraries.

- On the EAR Module and Utility JAR Projects window, clear the check box next to the `jboss-seam.jar`. See Figure 10-3. Click **Finish**.

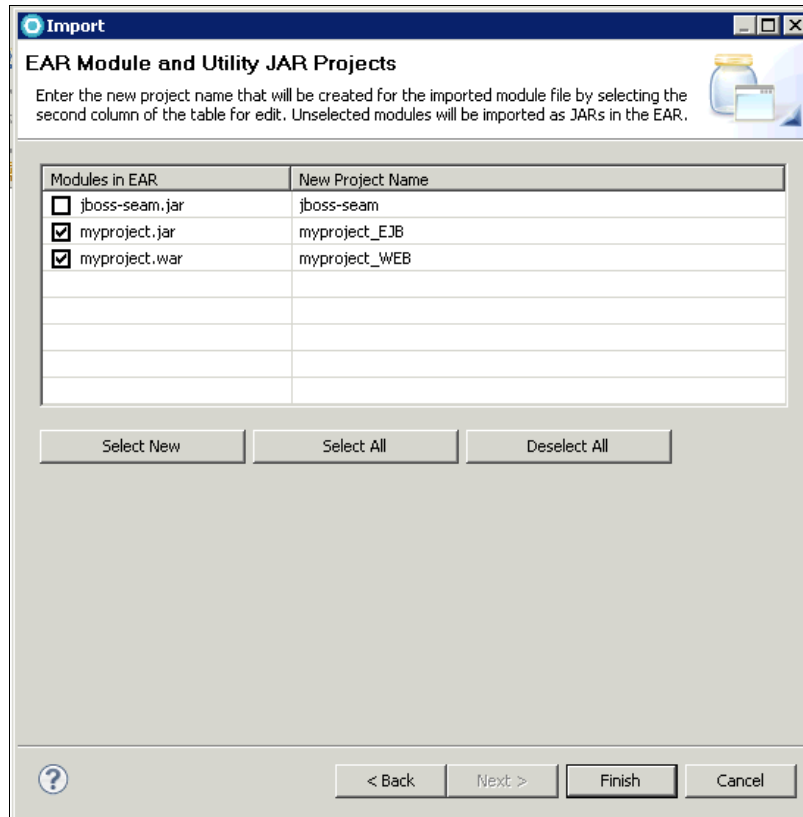


Figure 10-3 Selecting EAR Modules during import

- Import the source files to the project structure. Upon inspection, you see that the `myProject_EJB` module imported the classes. Import the source files of these classes to the `ejbModule` folder and remove the class files from the project.

Figure 10-4 shows the module structure before and after the source code import.

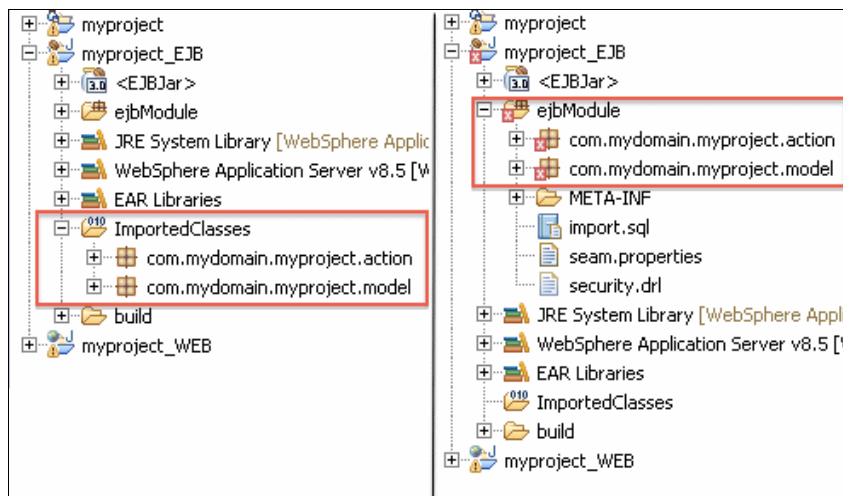


Figure 10-4 EJB module structure before and after source code import

7. Click the **ejbModule** node in the myproject\_EJB module and select **File** → **Import**.
8. Click **General** → **File System** in the Import window. The File System Import window opens. Import the source code from two locations:
  - C:\Projects\myproject\src\hot
  - C:\Projects\myproject\src\main
9. Click **Browse** and select the C:\Projects\myproject\src\hot folder that contains the source code files for the com.mydomain.myproject.action package. Select all source files, as shown in Figure 10-5. Click **Finish**.

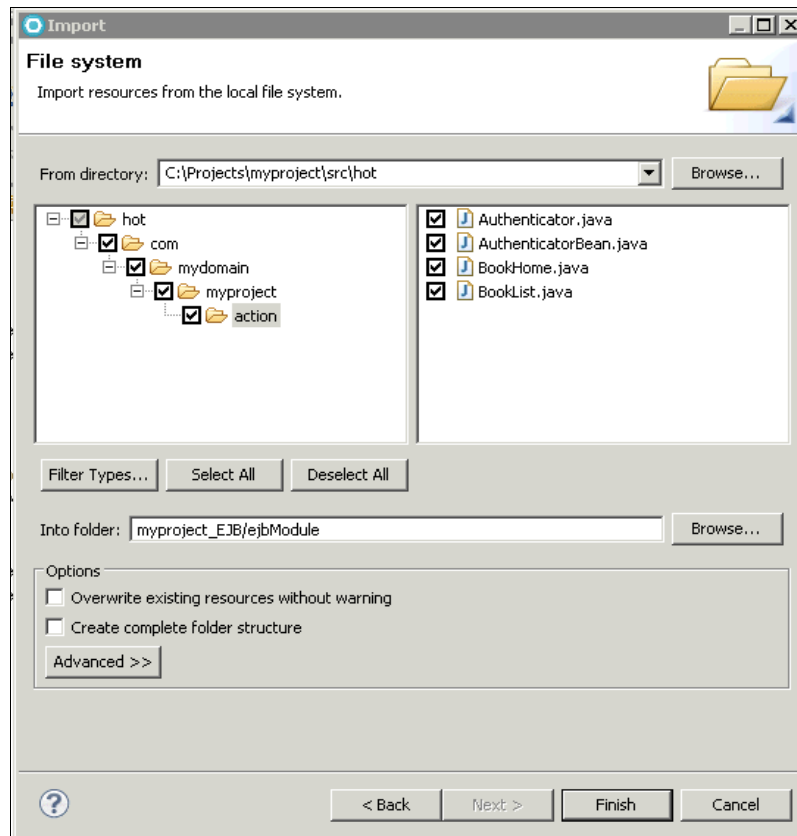


Figure 10-5 Importing source files

10. Repeat step 9 for the C:\Projects\myproject\src\main folder, which contains the source code for the com.mydomain.myproject.model package.
11. Delete the packages under the ImportedClasses node by selecting them, clicking **Delete**, and confirming the deletion.

The project structure looks like the right pane of Figure 10-4 on page 322.

Analyzing the web module, we can see that there are no imported classes that we must replace with source files. Resources such as html and xhtml files, style sheets, and images are imported automatically during the EAR import.

## 10.1.6 Analyzing and fixing problems

The following sections describe the steps that are taken to analyze and fix migration problems.

## Fixing compile time and runtime dependencies

In the right pane of Figure 10-4 on page 322, there are compile-time errors for several classes (flagged with red cross icons). When there are many libraries that can be reused by applications, you may use shared libraries rather than libraries in packaged applications. This situation reduces the packaged size of the application, speeds up deployment, eases maintenance of libraries, and helps conserve server resources. We use shared libraries on this migration exercise.

To fix the compile time and runtime dependencies, complete the following steps:

1. Create a folder to contain the libraries. We chose C:\sharedlibseam. We refer to this location as <shared\_lib\_dir>.
2. As shown in the left pane of Figure 10-6, our EAR file contains 16 libraries that can be moved to a shared library. Select all of the libraries.

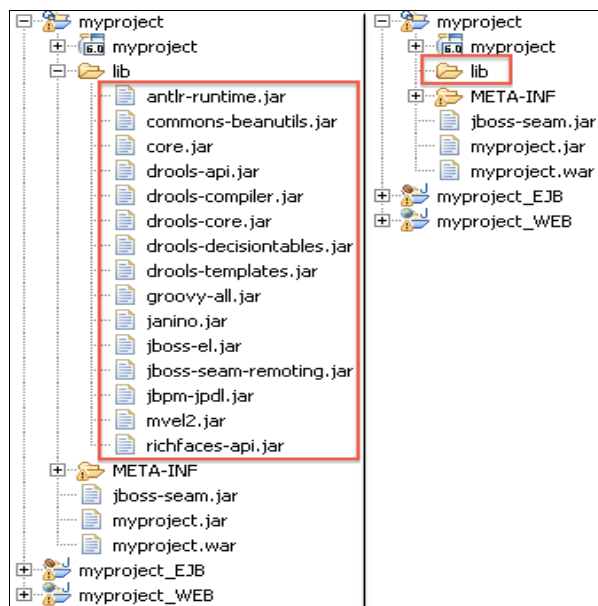


Figure 10-6 Before and after libraries in EAR moved to a shared library

3. Copy these libraries by right-clicking and selecting **Copy**.
4. Paste the copied libraries in to <shared\_lib\_dir>.
5. Delete the selected libraries from EAR module by right-clicking and selecting **Delete**. The result is the project structure that is shown in the right pane of Figure 10-6.

We do not move myproject.war and myproject.ear. These files are refreshed as the corresponding module projects are built.

There are more dependencies. These dependencies are the libraries that are shipped with JBoss Application Server, which is a natural target for most Seam applications, so they are not included in the application. Because these libraries are not part of the WebSphere Application Server distribution, we must include them. These libraries are also included in our generated project, so we can copy them from our project folder.



For our project, the additional dependencies in Table 10-1 must be copied to <shared\_lib\_dir> from C:\Projects\myproject\lib.

Table 10-1 Shared dependencies to be copied

antlr.jar	antlr-runtime.jar	bsh.jar
cglib-nodep.jar	commons-collections.jar	concurrent.jar
dom4j.jar	gwt-servlet.jar	hibernate-annotations.jar
hibernate-commons-annotations.jar	hibernate-core.jar	hibernate-entitymanager.jar
hibernate-validator.jar	javassist.jar	jboss-common-core.jar
jboss-seam-flex.jar	jsf-api.jar	jsf-impl.jar
log4j.jar	slf4j-api.jar	slf4j-log4j12.jar

- Click the **myproject** application module. Right-click and select **Java EE** → **Open WebSphere Application Server Deployment** to open the WebSphere Application Server Deployment window.
- Scroll down to the Shared Library section. Expand the section and click **Add**.
- Enter the Name and Description fields. We used SeamSharedLibs as the shared library name.
- Click **Browse** on the Class Path section of the window. A file selection dialog box is displayed (Figure 10-7). Select <shared\_lib\_folder> and click **OK**.

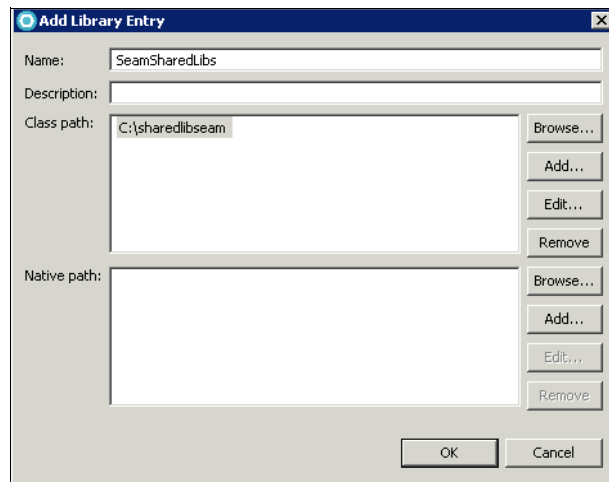


Figure 10-7 Shared library properties

We must modify the application class loading mechanism because we are using a third-party persistence provider. The hibernate-entitymanager.jar file and its dependencies serve this purpose.

We also use a third-party JSF implementation. WebSphere Application Server supports JSF 2.0, but our application is built with JSF 1.2. Therefore, we put the JSF implementation libraries in to the shared library, as we already added `jsf-api.jar` and `jsf-impl.jar`. More information about JSF implementation configuration can be found at the following web page:

<http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=twebjsf>

10. Scroll down to the Application section and change the Classloader mode to `PARENT_LAST`.
11. Press `Ctrl+s` to save your changes.
12. While the shared library resolves the runtime dependencies, the `myproject_EJB` module also has compile-time dependencies on the libraries we moved to the shared library. To resolve the compile-time dependencies, select the **myproject\_EJB** module, right-click, and select **Build Path** → **Configure Build Path**.
13. Select the **Libraries** tab in the Java Build Path window. Click **Add External JARs**. A file selection dialog box is displayed.
14. Navigate to `<shared_lib_folder>` and select the following libraries to add to build path:
  - `jboss-seam.jar`
  - `hibernate-validator.jar`
15. Click **OK** to continue. The class path looks like Figure 10-8.

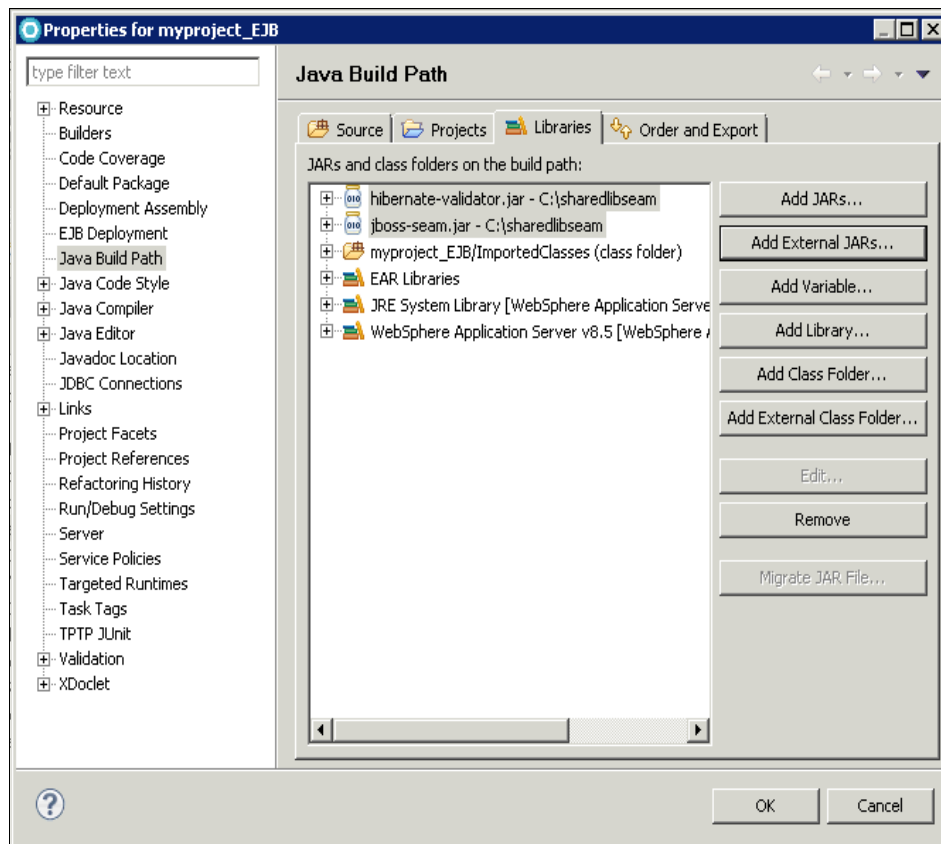


Figure 10-8 EJB module class path

16. Click **Project** → **Clean** to rebuild the project. No compile-time errors remain.

## Checking the project using the Application Migration Tool - JBoss AS to WebSphere

To run the Application Migration Tool for our project, complete the following steps:

1. Click the **myproject** application module. Right-click and select **Software Analyzer** → **Software Analyzer Configurations**. The Software Analyzer Configurations window opens. In the left pane, right-click **Software Analyzer** → **New**.
2. Our focus here is on Framework migration, the rules for which are included in the server-specific rule set, so we can select any server. Select the rule set for JBoss migration, as it is one of the platforms that the EAR file is built for. Click the **Rules** tab. Select **JBoss Application Migration** in the drop-down menu. As you can see at Figure 10-9, all the Framework migration rules are automatically selected.

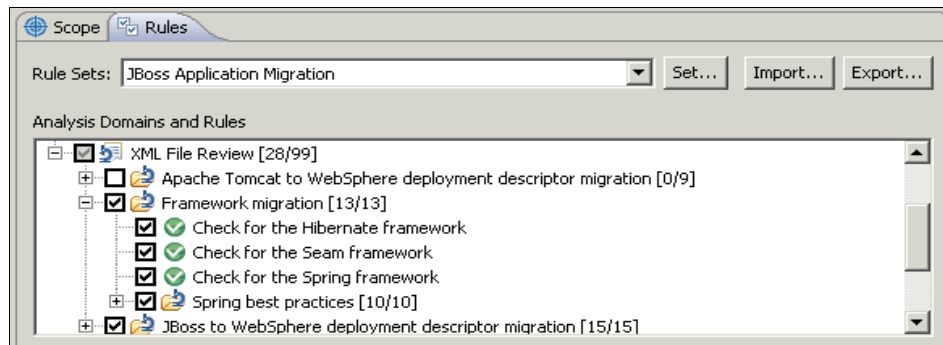


Figure 10-9 Framework migration rules

3. Click **Set**. The rule set configuration dialog box opens.
4. Select **WebSphere Application Server V8.5** as the target application server, and **Java 6** for the Source and Target Java version. Click **OK**. The respective rules become selected in the Analysis Rules and Domains area.
5. Click **Analyze**. After the analysis is complete, the Software Analyzer Results view is displayed. Inside the Software Analyzer Results view, you can see a tab for each domain of analysis. In our case, the tabs are Class-path Review, XML File Review, JSP Review, and Java Code Review.

- Expand the **XML File Review** section and the results in this section. The Application Migration Tool detects Hibernate as a persistence provider, and the Seam framework (Figure 10-10).

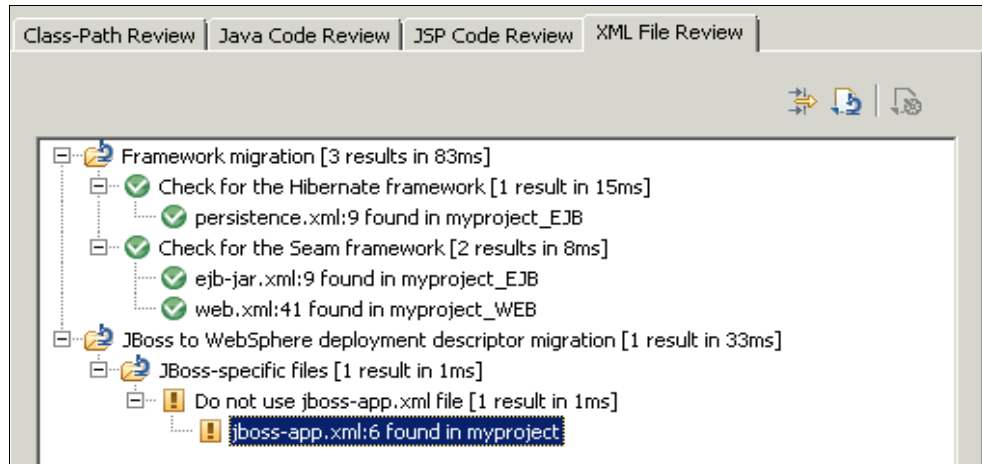


Figure 10-10 XML File Review

- A JBoss specific file is detected. Remove the `jboss-app.xml` file by right-clicking **myprojecy\META-INF\jboss-app.xml** and then selecting **Delete**.
- Switching to the **Java Code Review** tab (Figure 10-11), we see a list of files to review. `Book.java`, our persistent object, and `AuthenticationBean`, our `Seam Named` object, are correctly marked for review.

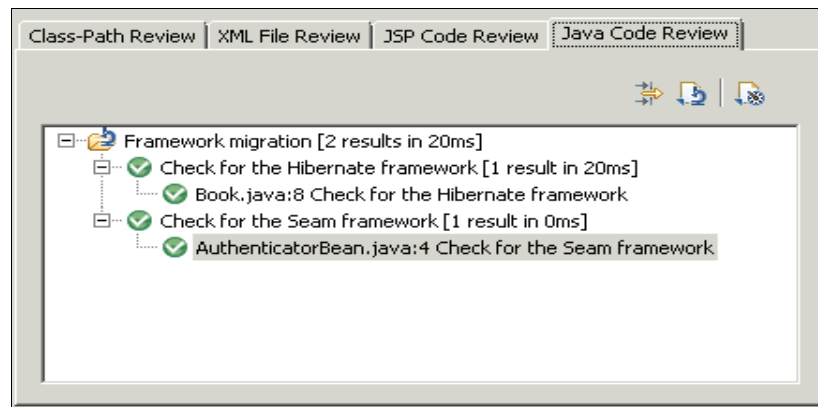


Figure 10-11 Java Code Review

- The Class-Path Review highlights several classpath issues that can be quick fixed (Figure 10-12).

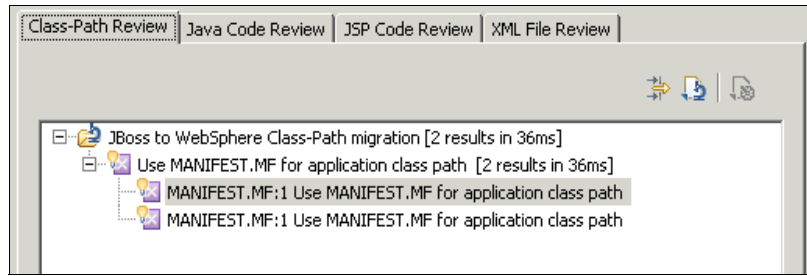


Figure 10-12 ClassPath review

- Expand the Class-Path Review section and the results in this section. Right-click one of the result lines and click **Quick Fix All**. You may also preview changes that are performed through quick fixes (Figure 10-13).

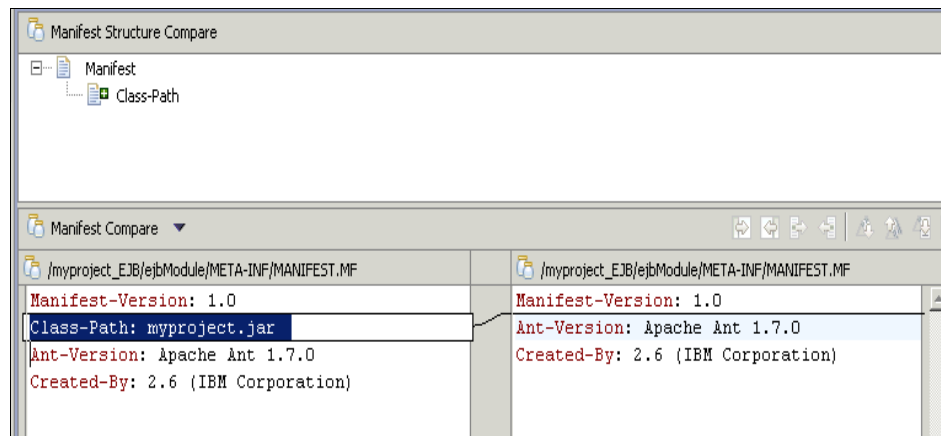


Figure 10-13 Previewing a quick fix

## Fixing the Seam configuration

Configuring Seam for a WebSphere JNDI name space is documented with several alternatives in the Seam documentation at:

<http://docs.jboss.org/seam/snapshot/en-US/html/websphere.html>

For our project, we use strategy 1. We add an `@JndiName` annotation to session beans, create a seam-specific properties file (the content of which is specified in the Seam documentation), change the JNDI matching pattern in `components.xml`, and add an EJB reference for `LocalEJBsynchronizations`.

To accomplish these tasks, complete the following steps:

- Annotate the session beans with the `@JndiName` annotation. Our single session bean is `AuthenticatorBean` at `myproject_EJB/ejbModule/com.mydomain.myproject.action.AuthenticatorBean.java`. Open this file in the editor.
- Add the following line between the `@Name` annotation and the class declaration line:

```
@JndiName("ejblocal:com.mydomain.myproject.action.Authenticator")
```

**Class name:** `JndiName` uses the name of the localinterface class instead of bean class name.

- This annotation requires the `org.jboss.seam.annotations.JndiName` class to be added to imports. Use a quick fix on the annotation by pressing `Ctrl+1` on the `@JndiName` annotation and clicking the **Import JndiName** option.
- Open the `myproject_WEB/WebContent/WEB-INF/components.xml` file in the editor.  
Replace `<core:init debug="@debug@" jndi-pattern="@jndiPattern@" />` with `<core:init debug="false" jndi-pattern="java:comp/env/#{ejbName}" />`.
- Create a file named `seam-jndi.properties` at `myproject_WEB/Java Resources/src`. The contents of this file are described in the Seam documentation. Copy the contents that are specified by the Seam documentation to the generated file and save your changes.

**Properties file:** The Seam documentation states that the `seam-jndi.properties` file is created at `WEB-INF/classes/seam-jndi.properties` in the web module. Because this folder is not visible in Rational Application Developer for WebSphere Software, use the `src` folder, because non-Java resources in this folder are copied to the classes folder during the build.

- Open the `myproject_WEB/WebContent/WEB-INF/web.xml` file in the editor. Switch to the Source tab for easier editing.
- Add the snippet that is shown in Example 10-3 inside the `web-app` element as the last element and save your changes.

*Example 10-3 Code snippet*

---

```
<ejb-local-ref>
  <description></description>
  <ejb-ref-name>EjbSynchronizations</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <local-home></local-home>
  <local>org.jboss.seam.transaction.LocalEjbSynchronizations</local>
  <ejb-link>EjbSynchronizations</ejb-link>
</ejb-local-ref>
```

---

## Fixing the persistence configuration

Seam can create a persistence unit by using only its name instead of using its JNDI name to look it up. This method is simpler than fixing the persistence unit JNDI configuration. To use this feature, complete the following steps:

- Replace the `<!-- persistence:managed-persistence-context name="entityManager" auto-create="true" persistence-unit-jndi-name="@puJndiName@" /-->` snippet in this file with `<persistence:entity-manager-factory name="entityManagerFactory" persistence-unit-name="myproject" /><persistence:managed-persistence-context name="entityManager" auto-create="true" entity-manager-factory="#{entityManagerFactory}" />`.

Our sample application uses Hibernate as JPA 1.0 persistence provider. We included the Hibernate libraries in our shared library, which is a necessary step in configuring third-party JPA providers for WebSphere. More information about configuring third-party JPA providers can be found at:

[http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=tejb\\_jpa3rdparty](http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=tejb_jpa3rdparty)

2. To enable the WebSphere transaction manager lookup, add the following property to the persistence.xml file:

```
<property name="hibernate.transaction.manager_lookup_class"
value="org.hibernate.transaction.WebSphereExtendedJTATransactionLookup" />
```

### Fixing the RichFaces configuration

The version of RichFaces bundled with the seam-gen utility depends on JSF 1.2, while WebSphere Application Server V8.5 is configured with JSF 2.0. To use the older implementation, we include JSF implementation libraries in our shared library configuration. More information about configuring JSF implementations can be found at:

<http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=twebjsf>

To initialize the Faces context, you must add a corresponding listener configuration to the web.xml file:

```
<listener>
<listener-class>com.sun.faces.config.ConfigureListener</listener-class>
</listener>
```

### Creating the data source

As seen in persistence.xml, the persistence configuration depends upon the existence of a data source with the myprojectDatasource JNDI name. Creating this data source for the test environment can be done in Rational Application Developer for WebSphere Software. Making configuration changes on a normal WebSphere Application Server installation through the administrative console is similar.

To create the data source, complete the following steps:

1. Select the **myproject** application module and right-click it. Select **Java EE** → **Open WebSphere Application Server Deployment**.
2. Scroll down to the Authentication section. Click **Add** near the JAAS Authentication list subsection. The Add JAAS Authentication Entry dialog box is displayed.
3. Enter db2admin for the alias, user ID, and description fields, and passw0rd for the password field. Click **OK**. A new authentication alias is listed in the JAAS Authentication list.
4. Scroll up to the Data Sources section and click **Add** near the JDBC provider list subsection.
5. Select **IBM DB2** as the database type and **DB2 Universal JDBC Driver Provider** as the JDBC provider type in the Create JDBC Provider window. Click **Next**.
6. Enter DB2 JDBC Provider in the name field of the JDBC Provider Details window.
7. Select the entries in the classpath section and click **Remove**.

8. Click **Add External JARs**. A file selection dialog box is displayed. Navigate to the C:\IBM\SQLLIB\java folder and select the following JARs and click **OK** to add them:
  - db2jcc.jar
  - db2jcc\_license\_cu.jar

The JDBC provider settings window (Figure 10-14) opens. Click **Finish**.

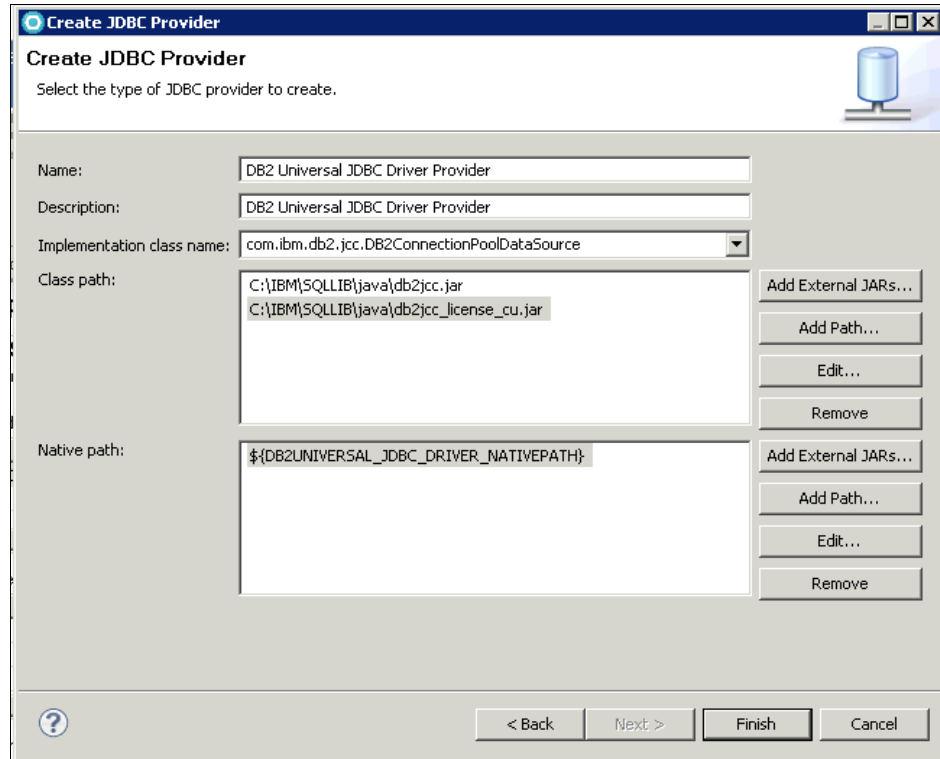


Figure 10-14 JDBC provider settings

9. The new JDBC provider is listed in the JDBC providers list. Click this line to select **DB2 JDBC Provider**, and then click **Add** in the data sources subsection.



10. Select **DB2 Universal JDBC Driver Provider** in the Create Data Source window (Figure 10-15). Click **Next**. The Data Source details window opens.

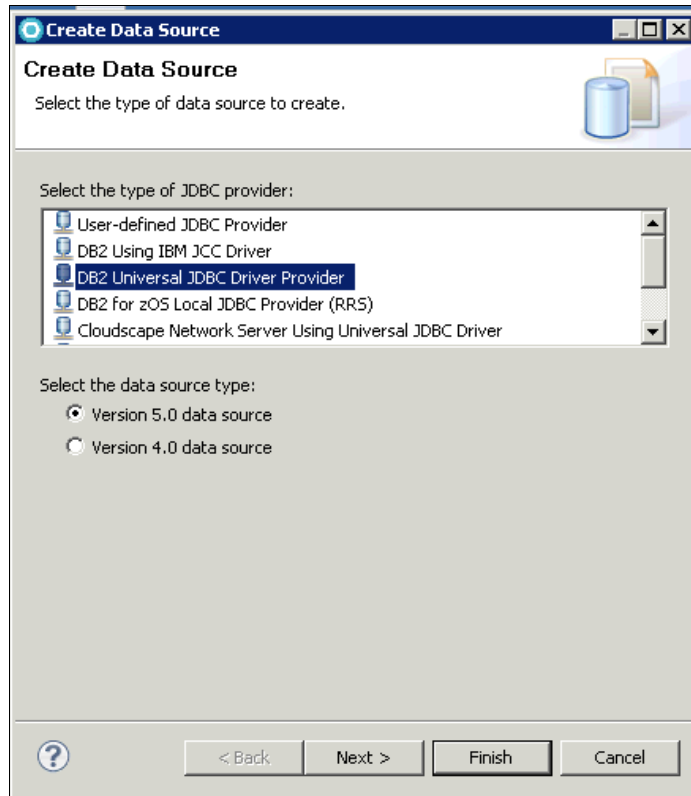


Figure 10-15 Selecting provider for data source

11. In the JNDI Name field, enter myprojectDatasource.

12. In the Container-managed authentication alias and Component-managed authentication alias fields, select the db2admin authentication alias that we created (Figure 10-16). Click **Next**. The Create Resource Properties window opens.

**Create Data Source**  
Select the type of data source to create.

Name: myprojectDataSource  
JNDI name: myprojectDataSource  
Description: minVer null - maxVer null - DB2 Universal Driver DataSource  
Category:  
Statement cache size: 10  
Data source helper class name: com.ibm.websphere.rsadapter.DB2UniversalDataStoreHelper  
Connection timeout: 180  
Maximum connections: 10  
Minimum connections: 1  
Reap time: 180  
Unused timeout: 1800  
Acquire timeout:  
Purge policy: EntirePool  
Component-managed authentication alias: db2admin  
Container-managed authentication alias: db2admin  
 Use this data source in container managed persistence (CMP)  
\* Required field.

? < Back Next > Finish Cancel

Figure 10-16 Data source details

13. In the Create Resource Properties window (Figure 10-17), select the **databaseName** Resource Property. In the value field for this property, enter TESTDB. Select the **serverName** property. In the value field for this property, enter localhost. Click **Finish**.

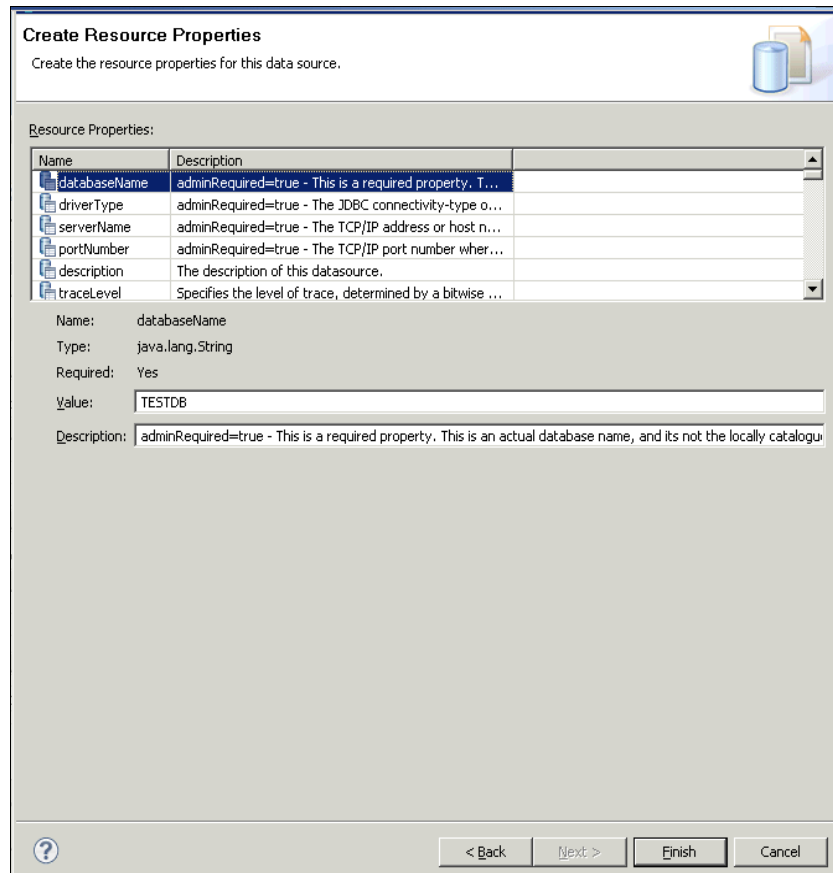


Figure 10-17 Resource properties for data source

14. The Data Sources section of the WebSphere Application Server Deployment window now looks like Figure 10-18. Press Ctrl+s to save your changes.

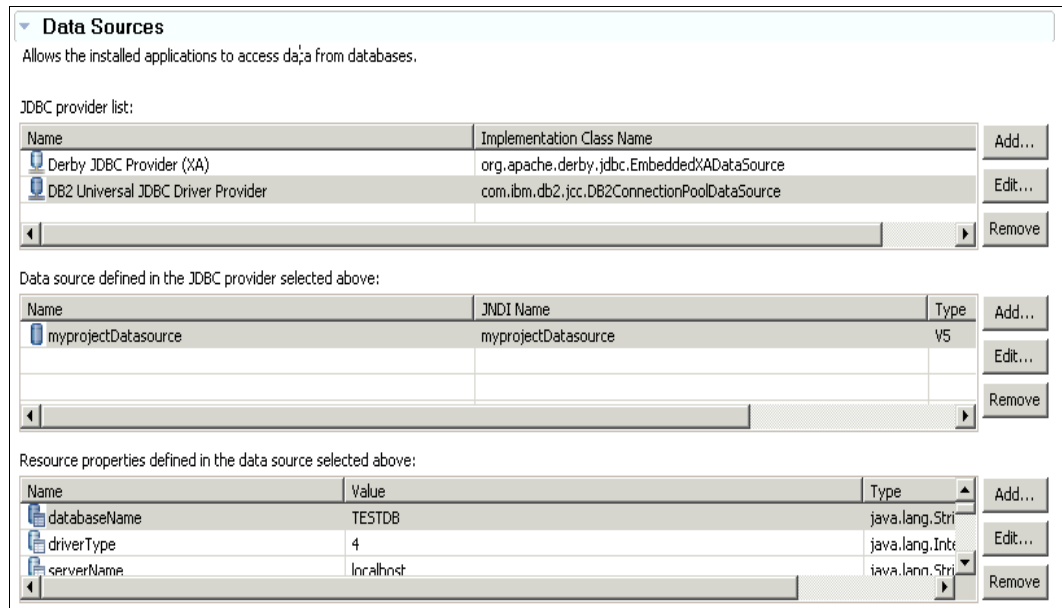


Figure 10-18 Resource properties for a data source

### Fixing the persistence configuration

The web module deployment descriptor, by generation, contains a persistence-unit-ref to the persistence unit definition at the EJB module. The indirect naming scheme that is used in the persistence-unit-name element of this reference does not work with WebSphere. Because of this situation, move persistence.xml to the web module and use the jar-file element inside persistence.xml to reference beans at the EJB module. Also, configure the Hibernate properties for WebSphere by completing the following steps:

1. Select the persistence.xml and orm.xml files in the myprojectEJB/ejbModule/META-INF directory.
2. Right-click and select **Move**.
3. In the destination area, select the myProject\_WEB/WebContent/META-inf folder and click **OK**.
4. Select **myproject\_Web**, right-click, and select **Properties** → **Java EE module dependencies**.

5. Select the **myproject\_EJB** check box (Figure 10-19) and click **OK** to continue.

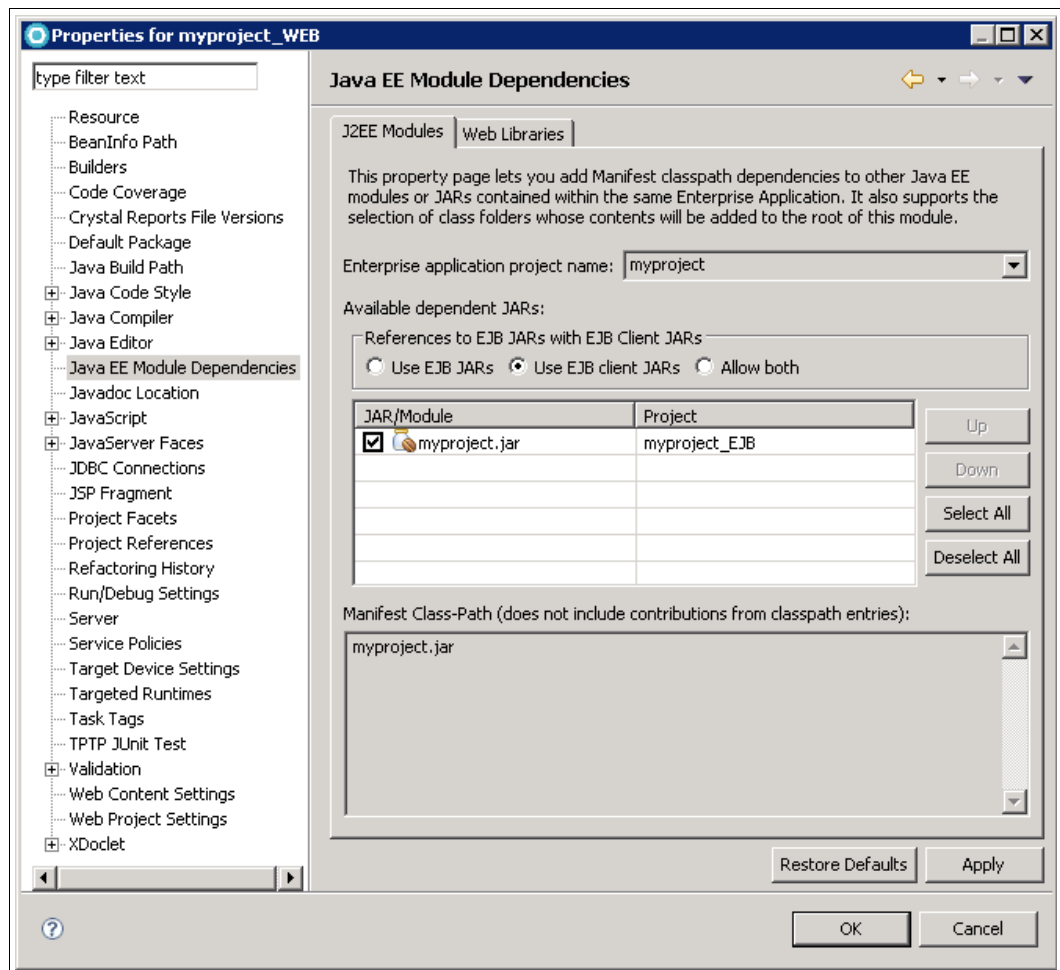


Figure 10-19 Web module dependencies

6. Open the copied `persistence.xml` file in the editor. Switch to the **Source** tab for faster editing.
7. Add the following snippet to `persistence.xml`, beneath the `<jta-data-source>` element:
 

```
<jar-file>myproject.jar</jar-file>
```
8. Remove the following line from `persistence.xml`:
 

```
<property name="jboss.entity.manager.factory.jndi.name" value="java:/myprojectEntityManagerFactory"/>
```
9. Add the following line inside the `<properties>` element in `persistence.xml`:
 

```
<property name="hibernate.transaction.manager_lookup_class" value="org.hibernate.transaction.WebSphereExtendedJTATransactionLookup" />
```
10. Press **Ctrl+s** to save your changes.
11. Fix the persistence unit reference in the web module. Open `myproject_WEB/WebContent/WEB-INF/web.xml` in the editor. Switch to the **Source** tab.
12. Replace the line
 

```
<persistence-unit-name>../myproject.jar#myproject</persistence-unit-name>
```

 with
 

```
<persistence-unit-name>myproject</persistence-unit-name>
```

## Building and running the application on an integrated test environment

The migration is complete and the application can be run on WebSphere Application Server V8.5. You can check how the migrated application works by using an integrated WebSphere Application Server V8.5 test environment.

To build, deploy, and run the application, complete the following steps:

1. Click the `myproject_WEB\WebContent\index.html` file. Right-click and select **Run As** → **Run on server**. The Run on server dialog box is displayed.
2. Keep the default settings, and click **Finish**.

After the application is deployed to the test environment, an embedded web browser is displayed in the editor pane, presenting the entry page of your seam-gen application.

3. Test the application by creating, updating, deleting, and listing entities. Use admin for the user name with an empty password to log in.

### 10.1.7 Summary

While we migrate our sample seam-gen generated application, we demonstrated the following concepts:

- ▶ A source migration to Rational Application Developer for WebSphere Software as a build and test environment
- ▶ Using the Application Migration Tool
- ▶ Using shared libraries for managing dependencies

The main points to highlight during this migration are as follows:

- ▶ Differences in default JNDI bindings require rework on JNDI lookups. Seam frameworks require a fair amount of configuration to work with WebSphere default JNDI bindings.
- ▶ Libraries that ship with JBoss Application Server distributions and not with WebSphere Application Server create more runtime dependencies that must be corrected.
- ▶ Using Hibernate for persistence requires a transaction strategy configuration specific to WebSphere.
- ▶ Using a third-party JPA provider requires changes to the classloading configuration.
- ▶ Using an old JSF implementation requires changes in the classloading configuration and the addition of the corresponding listener to web module configuration.

## 10.2 Migrating a Spring Framework application

To demonstrate the Spring framework migration, we use an application that is generated by Spring Roo, which is a rapid application development tool that uses many open source Java components with the Spring modules. More information about Spring framework and Spring Roo may be found at:

- ▶ <http://www.springsource.org/>
- ▶ <http://www.springsource.org/spring-roo>

We generate a simple application using a simple data model. This data model contains four different entities with relationships of different cardinalities. For the purposes of this sample, an “IBMRedBook” consists of “Chapters”, each of which has a primary “Owner” and a number of “Participants”.

We chose to migrate this application for the following reasons:

- ▶ A project structure that is created with Spring Roo is representative of a well-designed Spring application structure.
- ▶ It is a simple and complete application that is good for demonstrating and focusing on framework-specific constructs without getting into too much repetitive work for various modules and components.
- ▶ It uses PrimeFaces on JSF 2.1 support, while WebSphere Application Server V8.5 supports JSF 2.0, so we can demonstrate third-party JSF implementations.
- ▶ It uses Hibernate as a JPA2 provider, so we can demonstrate third-party JPA providers.
- ▶ It uses Spring 3.1.1.

## 10.2.1 Migration approach

In this migration exercise, we use IBM Rational Application Developer for WebSphere Software as the build environment.

The migration is performed by completing the following steps:

1. Install Spring Roo.
2. Install Apache Derby.
3. Generate a sample application and import it in to Rational Application Developer.
4. Check the configuration files with the Application Migration Tool.
5. Analyze and fix problems that are reported by the build environment and Application Migration Tool.
6. Deploy and test the application in an integrated WebSphere Application Server V8.5 test environment.

## 10.2.2 Installing Spring Roo

Download the Spring Framework distribution from the following website:

<http://www.springsource.org/download/community?project=Spring%2520Roo>

We use Version 1.2.2.RELEASE in our environment, which is the latest version at the time of the writing of this book. Extract the downloaded file to a suitable location. We used `C:\spring-roo-1.2.2.RELEASE` as our example directory.

Define an environment variable `R00_HOME` that points to the installation directory, and add `%R00_HOME%/bin` to your class path.

Spring depends on Apache Ant and Apache Maven for operation. For installation instructions for Apache Ant and Apache Maven, see 8.3.1, “Installing and configuring Apache Ant” on page 237.

## 10.2.3 Installing Apache Derby DB

Download Apache Derby DB distribution from the following website:

[http://db.apache.org/derby/derby\\_downloads.html](http://db.apache.org/derby/derby_downloads.html)

We use Version 10.8.2.2 in our environment, which is the latest version at the time of the writing of this book. Extract the downloaded file to a suitable location. We use C:\db-derby-10.8.2.2-bin as the directory.

## 10.2.4 Generating a sample application using Spring Roo

To generate the sample application, complete the following steps:

1. Create a directory for your project. We use C:\sample.
2. Open a command prompt and change the directory to the folder you created.
3. Create a text file with the contents shown in Example 10-4. We use the name sample.roo for this file.

In Example 10-4, you might notice several choices that we made, such as using Hibernate as the JPA provider, which is backed up by a Derby database, and JSF for the user interface. The `--fetch EAGER` directives in the set types field definitions are introduced as an issue (described at <https://jira.springsource.org/browse/R00-2870>), and the proposed solution works in our example.

*Example 10-4 Roo script file for generating a sample application*

---

```
project --topLevelPackage test.ibmredbookproject
jpa setup --provider HIBERNATE --database DERBY_CLIENT
entity jpa --class ~.domain.Participant
field string --fieldName name --notNull --sizeMin 2
entity jpa --class ~.domain.Owner
field string --fieldName name --notNull --sizeMin 2
entity jpa --class ~.domain.Chapter
field string --fieldName name --notNull --sizeMin 2
field set --fieldName participants --type ~.domain.Participant --fetch EAGER
field reference --fieldName owner --type ~.domain.Owner
entity jpa --class ~.domain.IBMRedBook
field string --fieldName name --notNull --sizeMin 2
field number --fieldName pages --type java.lang.Integer
field date --fieldName publishDate --type java.util.Date
field set --fieldName chapters --type ~.domain.Chapter --fetch EAGER
web jsf setup
web jsf all --package test.ibmredbookproject.ui
```

---

4. Enter **roo** and press Enter. You see the welcome messages as an interactive session begins.
5. At the roo prompt, enter **script --file sample.roo** and press Enter. You see lines that report how components are being created and updated.
6. After the script execution ends, enter **exit** to terminate the interactive session.
7. The source code is generated and ready to be built, but while we validated the application, we noticed the issue described at <http://forum.springsource.org/showthread.php?124150-JSF-OneToMany-validation-issues> and noticed that the proposed solution works for our sample.



To implement this fix, for all classes that can be retrieved as a set with a result of a one-to-many relationship (in our case, `IBMRedBook`, `Chapter`, and `Participant`), open the source file at `C:\sample\src\main\java\test\ibmredbookproject\domain` folder in your project folder, and override the `hashCode` and `equals` methods, as shown in Example 10-5. The example code is for the `IBMRedBook` class. You may append the same block of code to all three classes just by changing the class name in the `equals` method. Be careful to establish only the equality and `hashCode` methods on the name fields of these classes.

*Example 10-5 Overriding hashCode and equals methods*

---

```

@Override
public int hashCode() {
    return (name == null) ? 0 : name.hashCode();
}

@Override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    IBMRedBook other = (IBMRedBook) obj;
    if (name == null) {
        if (other.name != null)
            return false;
    } else if (!name.equals(other.name))
        return false;
    return true;
}

```

---

8. Edit `src\main\resources\META-INF\spring\database.properties` and add a user name and password for the Derby database by inserting the following lines:
 

```

database.username=derbyuser
database.password=derbyuser

```
9. Before you build the project, make sure that the database is running, because the Maven build process involves running automated tests, some of which require a connection to the database. To start the database, complete the following steps:
  - a. Open a new command prompt and change the directory to `C:\db-derby-10.8.2.2-bin\lib`.
  - b. Enter `java -jar derbyrun.jar server start` and press Enter.
  - c. Leave the comment prompt open for the rest of the exercise.
10. To build the generated source code, enter `mvn install` at the command prompt in the project directory and press Enter.
11. Verify the application at the Jetty server by entering `mvn jetty:run-exploded` and pressing Enter. After the necessary libraries are automatically downloaded, you may check the application at `http://localhost:8080/ibmredbookproject/` and try creating, listing, or modifying Chapters and `IBMRedBooks`. At the time of the writing of this book, dialog components were not being properly handled and there were some outstanding issues with using PrimeFaces modal dialogs in Internet Explorer Version 8.

## 10.2.5 Importing the EAR file and source code to Rational Application Developer

To import the WAR file and source code to Rational Application Developer, complete the following steps:

1. Start Rational Application Developer for WebSphere Software by clicking **Start** → **IBM Software Delivery Platform** → **IBM Rational Application Developer 8.5** → **IBM Rational Application Developer**.
2. Start the Import wizard by clicking **File** → **Import**. Expand the **Web** node in the import source tree and select **WAR file**. See Figure 10-20.

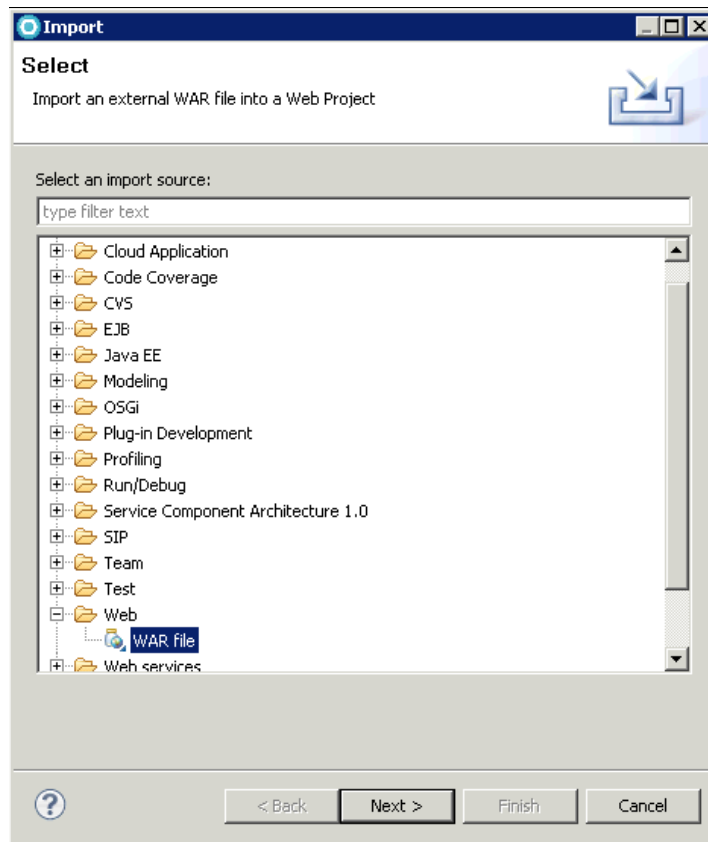


Figure 10-20 WAR import wizard

3. In the Enterprise Application Import window, browse to the location of the WAR file that is generated by Spring Roo, which can be found at `C:\sample\target\ibmredbookproject-0.1.0.BUILD-SNAPSHOT.war`. See Figure 10-21. Click **Next**.

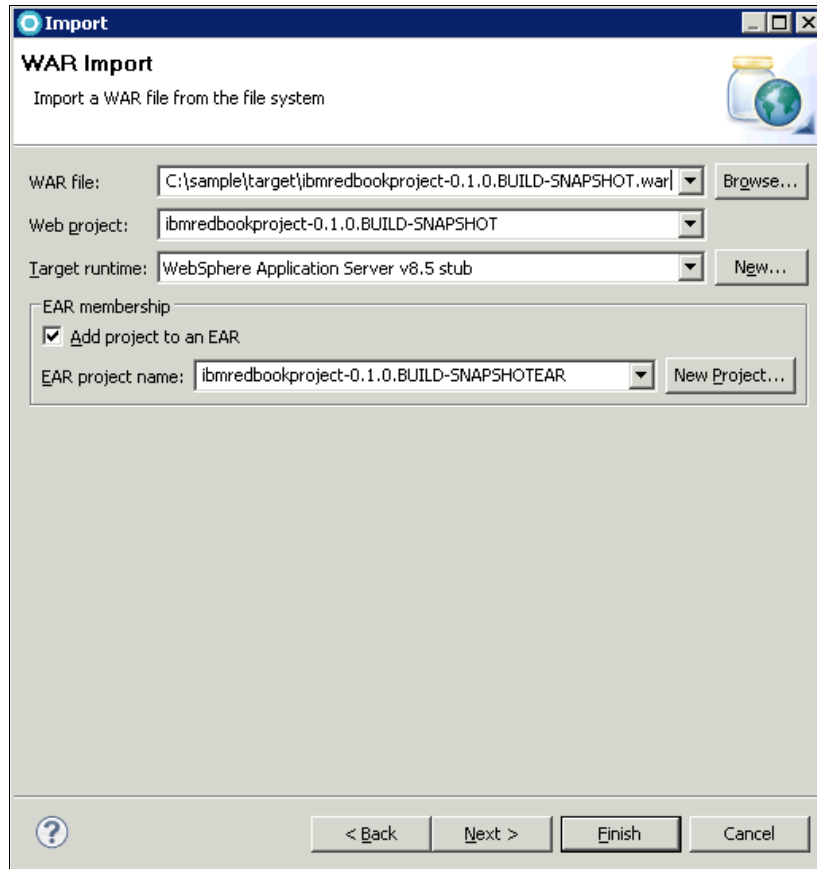


Figure 10-21 Importing a sample application WAR file

4. Click **Finish** on the Utility JARs and Web Libraries window without selecting any libraries.

## 10.2.6 Analyzing and fixing problems

The following sections describe the steps that are taken to analyze and fix migration problems.

### Fixing compile time and runtime dependencies

Because we are using third-party JSF and JPA implementations, we load libraries for these implementations from an isolated shared library classloader. Also, when there are many libraries that can be reused by applications, we may use shared libraries rather than libraries in packaged applications. This setup helps reduce the packaged size of the application, speeds up deployment, eases maintenance of libraries, and helps conserve server resources. We use shared libraries in this migration exercise.

To fix compile time and runtime dependencies, complete the following steps:

1. Create a folder to contain the libraries. We choose `C:\sharedLibsRoo`. We refer to this location as `<shared_lib_dir>`.

2. The WAR file contains numerous libraries, including libraries for Hibernate and Spring, under the WebContent/WEB-INF/lib node. Move these libraries to a shared library. Select all of the libraries by pressing and holding the Ctrl key.
3. Copy these libraries to the clipboard by right-clicking and selecting **Copy**. Paste the copied libraries in to <shared\_lib\_dir>.
4. Delete the selected libraries from the EAR module by selecting **Delete**.
5. Delete the following JAR files from <shared\_lib\_dir>, because classes contained in these libraries are provided by WebSphere Application Server:
  - hibernate-jpa-2.0-api-1.0.1.Final.jar
  - jta-1.1.jar
  - validation-api-1.0.0.GA.jar
6. Click the **myproject** application module. Right-click and select **Java EE** → **Open WebSphere Application Server Deployment** to open the WebSphere Application Server Deployment window.
7. Scroll down to Shared Library section. Expand the section and click **Add**.
8. Complete the Name and Description fields. We used SharedLibsRoo as the shared library name.
9. Click **Browse** on the Class path section of the window. A file selection dialog box is displayed (Figure 10-22). Using the dialog box, select <shared\_lib\_folder> and click **OK**.

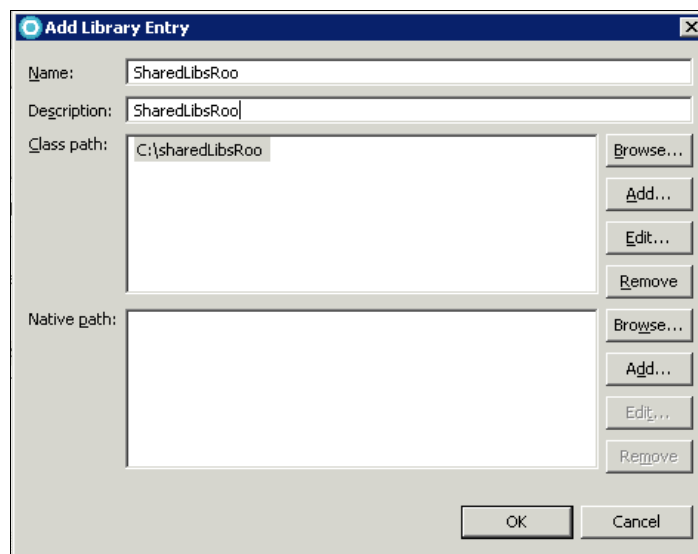


Figure 10-22 Shared library properties

You must modify the application class loading mechanism because you use a third-party persistence provider. The `hibernate-entitymanager-3.6.9.Final.jar` file and its dependencies that we added serve this purpose.

We are also using a third-party JSF implementation. WebSphere Application Server supports JSF 2.0, but our application is built with JSF 2.1. Therefore, we add the JSF implementation libraries to the shared library, as we added `jsf-api-2.1.2.jar` and `jsf-impl-2.1.2.jar` in the previous section. More information about the JSF implementation configuration can be found at the following website:

<http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=twebjsf>

10. Scroll down to the Application section and change the Classloader mode to PARENT\_LAST.
11. Press Ctrl+s to save your changes.

## Checking the project using the Application Migration Tool - WebLogic to WebSphere

To run the Application Migration Tool - WebLogic to WebSphere on your project, complete the following steps:

1. Click the **myproject** application module. Right-click and select **Software Analyzer** → **Software Analyzer Configurations**. The Software Analyzer Configurations window opens. In the left pane, right-click and select **Software Analyzer** → **New**.
2. Our focus is on Framework migration, the rules for which are included in a server-specific rule set, so we can select any server. Let us select the rule set for WebLogic Application Migration, as it is one of the platforms that the EAR file was built for. Click the **Rules** tab. Select **WebLogic Application Migration** in the drop-down menu. As you can see in Figure 10-23, all the Framework migration rules are automatically selected.

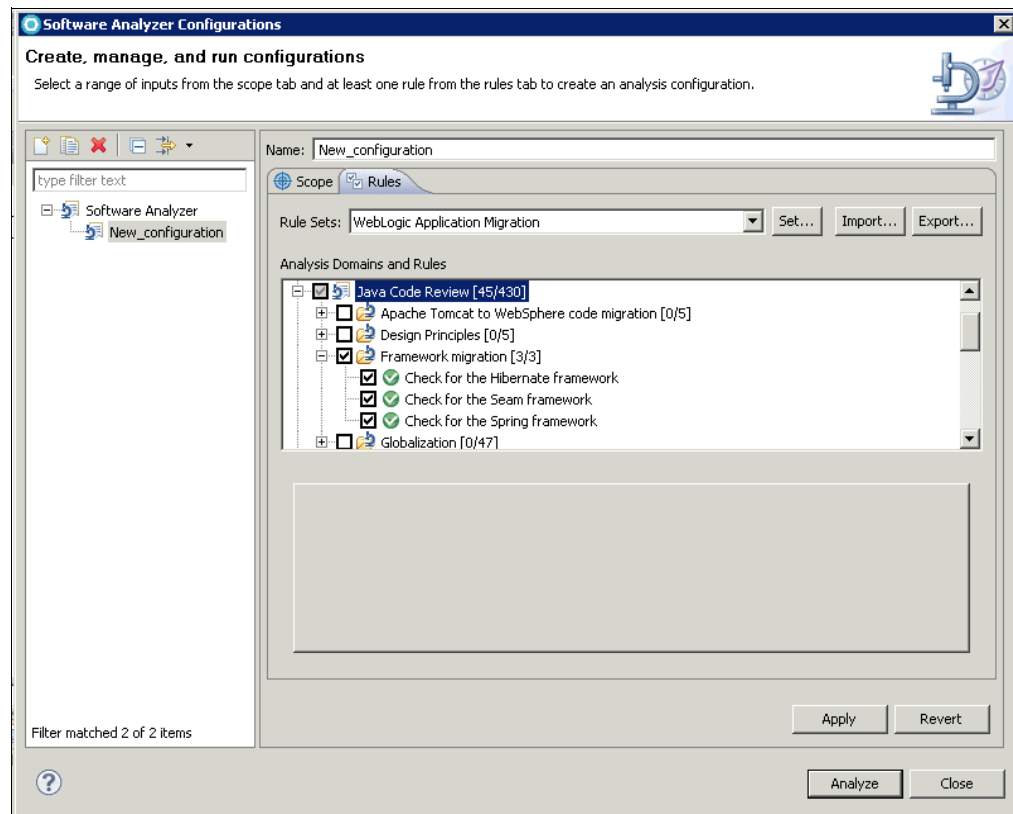


Figure 10-23 Framework migration rules

3. Click **Set**. The rule set configuration dialog box opens.
4. Select **WebSphere Application Server V8.5** as the target application server and **Java 6** for the Source and Target Java version. Click **OK**. The respective rules become selected in the Analysis Rules and Domains area.

5. Click **Analyze**. After the analysis is complete, the Software Analyzer Results view is displayed. Inside the Software Analyzer Results view, you can see a tab for each domain of analysis. In our case, the tabs are named Class-path Review, XML File Review, JSP Review, and Java Code Review.
6. Expand the XML file review section and the results in this section. In Figure 10-24, Application Migration Tool detected the usage of Hibernate as a persistence provider, and Spring framework usage, and shows their respective configuration files. Also, as a part of Spring preferred practices rules, you are reminded to check the configuration of LocalContainerEntityManagerFactoryBean Spring Framework configuration element.

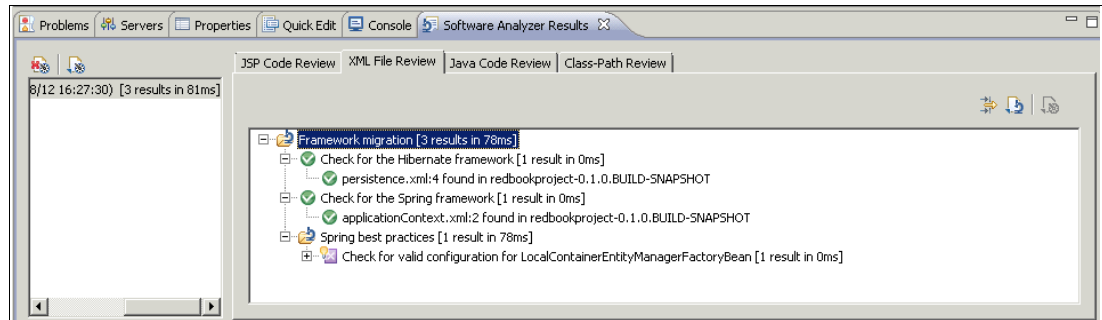


Figure 10-24 XML File review

## Building and running the application on an integrated test environment

The migration is complete and the application can be run on WebSphere Application Server V8.5. You can check how the migrated application works by using an integrated WebSphere Application Server V8.5 test environment.

To build, deploy, and run the application, complete the following steps:

1. Click the `WebContent\index.html` file in the web module. Right-click and select **Run As** → **Run on server**. The Run on server dialog box is displayed.
2. Keep the default settings, and press **Finish**.

After the application is deployed to test environment, an embedded web browser is displayed in the editor pane, presenting the entry page of our seam-gen application.

3. Test the application by creating, updating, deleting, and listing entities. Use admin for the user name with an empty password to log in.

You may notice the following exception during the start:

```
ERROR org.hibernate.tool.hbm2ddl.SchemaExport - schema export unsuccessful
java.lang.UnsupportedOperationException: The user must supply a JDBC connection
```

This exception occurs because while the application is being started, persistence units are initialized before the application's modules are initialized. Because the Spring framework is initialized within the web module of the application, the database definition and connection pooling properties that are defined in the Spring configuration are not yet available. As the persistence unit is initialized, Hibernate tries to make sure that the tables for the managed entities exist, but it cannot find a connection to do so.

You may safely ignore this error because the database and schema were created during the automated testing part of the build process, so you do not need to create it again during the application start. Another solution is to define a data source in `persistence.xml` instead of a Spring configured connection pool, as we demonstrate when migrating Hibernate to OpenJPA.

## Using a data source instead of Spring managed connection pooling

Spring Roo, by default, generates applications so that database connections are managed by a Spring managed connection pool that uses an Apache DBCP project.

Even though it is possible to configure Spring Roo according to our choices of data source and persistence provider, we demonstrate the steps manually for clarity.

If you are using a similar application and prefer to use a data source that is managed by WebSphere Application Server itself instead of DBCP, complete the following steps:

1. Define a data source that points to your database. Select the **myproject** application, module, right-click it, and select **Java EE → Open WebSphere Application Server Deployment**.
2. Scroll down to the Authentication section. Click **Add** near the JAAS Authentication list subsection. The Add JAAS Authentication Entry dialog box is displayed.
3. Enter derbyuser for the alias, user ID, and description fields, and derbyuser for the password field. Click **OK**. A new authentication alias is listed in the JAAS Authentication list.
4. Scroll up to the Data Sources section and click **Add** near the JDBC provider list subsection.
5. Select **Derby** as the database type and **Derby Network Server Using Derby Client** as the JDBC provider type in the Create JDBC Provider window. Click **Next**.
6. Enter Derby Network Server Using Derby Client in the name field of the JDBC Provider Details window.
7. Select the entries in the classpath section and click **Remove**.
8. Click **Add External JARs**. A file selection dialog box is displayed. Navigate to the C:\db-derby-10.8.2.2-bin\lib folder, select the following JAR files, and click **OK** to add it:

derbyclient.jar

The JDBC provider settings window (Figure 10-25) opens. Click **Finish**.

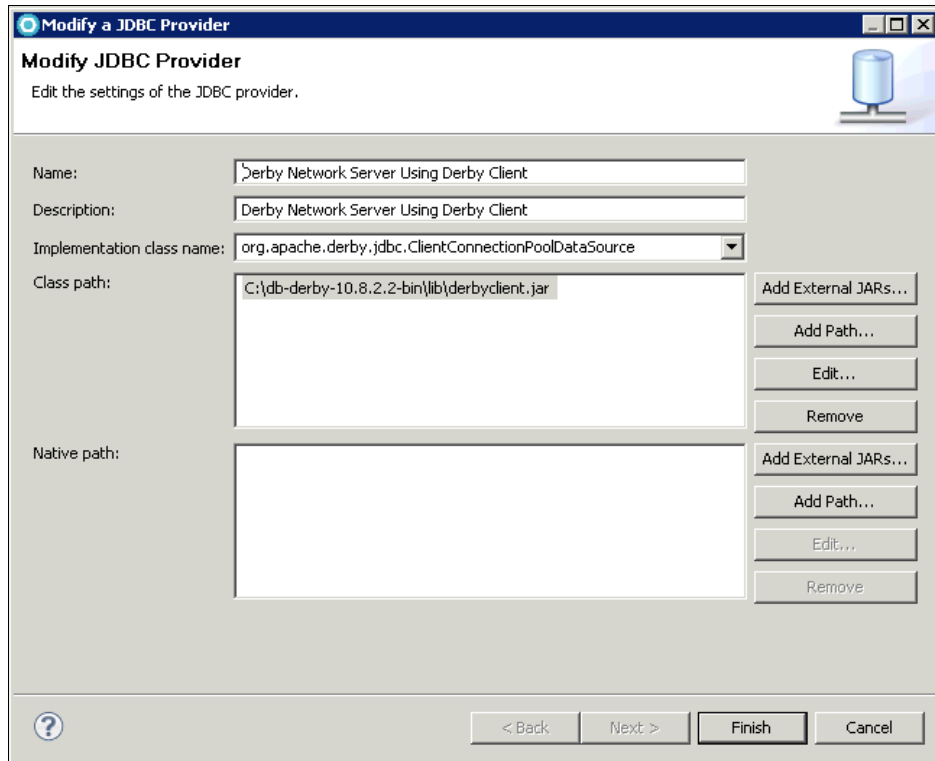


Figure 10-25 JDBC provider settings

9. The new JDBC provider is listed in the JDBC providers list. Click this line to select **DB2 JDBC Provider**, and then click **Add** in the data sources subsection.



10. Select **Derby Network Server Using Derby Client** in the Create Data Source window (Figure 10-26). Click **Next**. The Data Source details window opens.

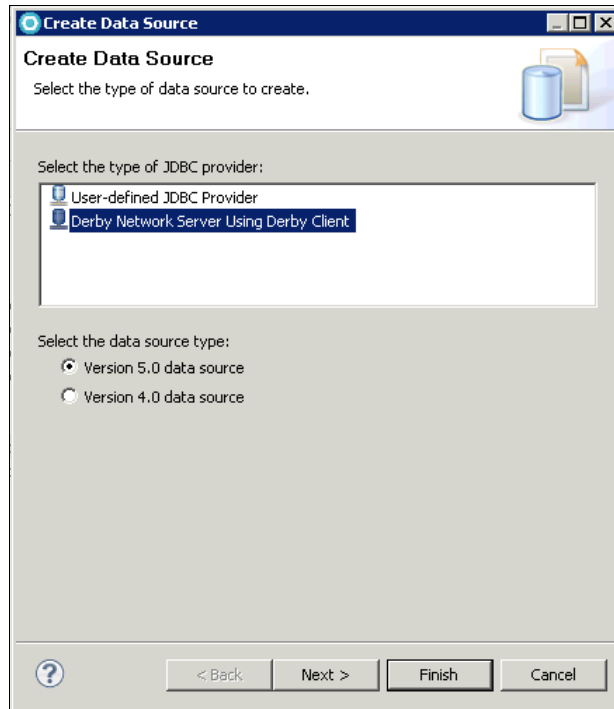


Figure 10-26 Selecting a provider for a data source

11. In the JNDI Name field, enter jdbc/testds.

12. In the Container-managed authentication alias and Component-managed authentication alias fields, select the **derbyuser** authentication alias that you created before (Figure 10-27). Click **Next**. The Create Resource Properties window opens.

**Create Data Source**  
Select the type of data source to create.

Name: Data source 1

JNDI name: jdbc/testds

Description: minVer 6.1 - maxVer null - New Network Server JDBC Datasource that uses Derby Network Client. This Datasource type is only configurable in version 6.1 and later nodes

Category:

Statement cache size: 10

Data source helper class name: com.ibm.websphere.rsadapter.DerbyNetworkServerDataStoreHelper

Connection timeout: 180

Maximum connections: 10

Minimum connections: 1

Reap time: 180

Unused timeout: 1800

Anead timeout:

Purge policy: EntirePool

Component-managed authentication alias: derbyuser

Container-managed authentication alias: derbyuser

Use this data source in container managed persistence (CMP)

\* Required field.

< Back Next > Finish Cancel

Figure 10-27 Data Source details

- In the Create Resource Properties window (Figure 10-17 on page 335), select the **databaseName** Resource Property. In the value field for this property, enter `ibmredbookproject`. Select the **serverName** property. In the value field for this property, enter `localhost`. Click **Finish**.

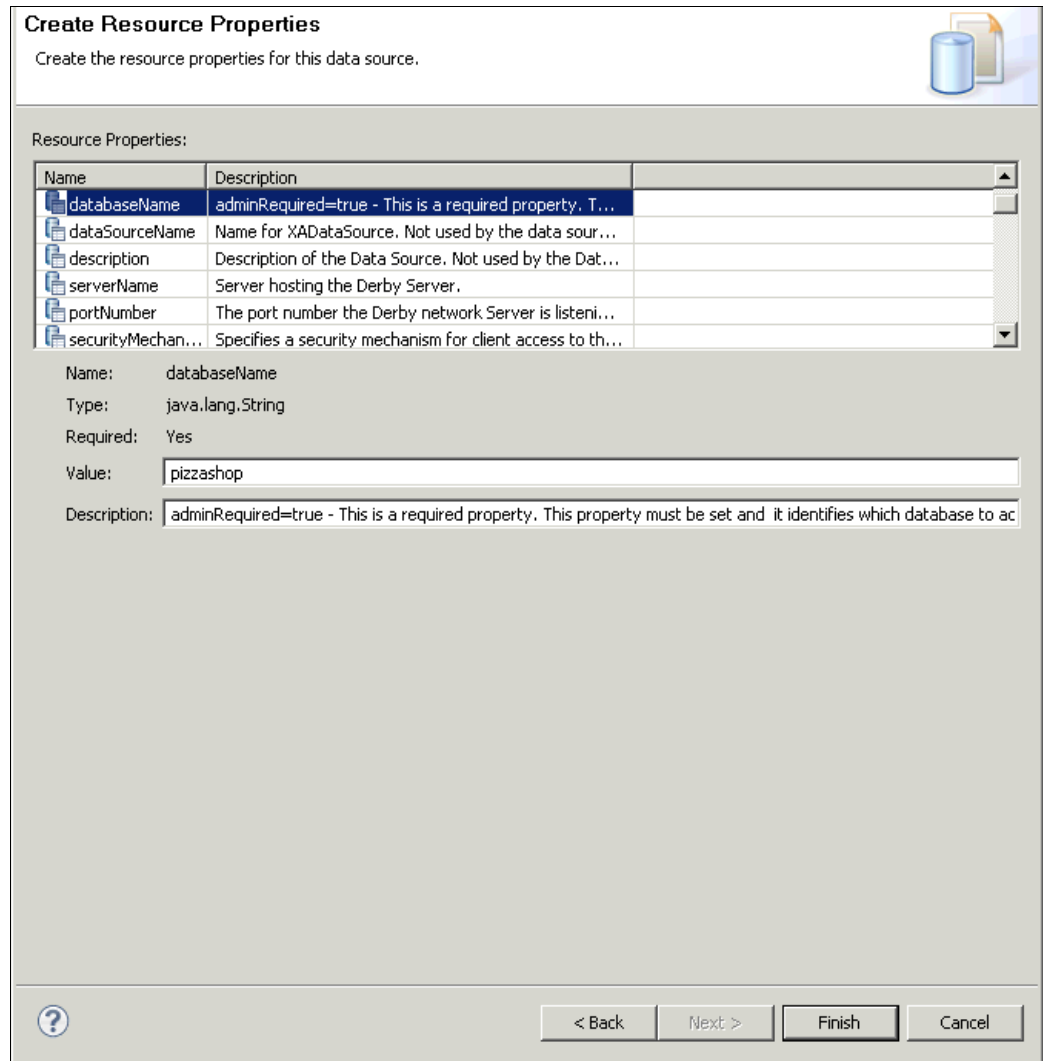


Figure 10-28 Resource properties for data source

- Press `Ctrl+s` to save your changes.
- Open `applicationContext.xml` in the Java Resources/`src/META_INF/spring` folder of the web module. Switch to the **Source** tab of the XML editor.
- Comment out the existing bean definition `C:\db-derby-10.8.2.2-bin\lib` for the 'dataSource' ID.
- Add the following bean definition:

```
<jee:jndi-lookup
id="testDatasource"
jndi-name="jdbc/testds"
cache="true"
expected-type="javax.sql.DataSource" />
```

18. Change the 'dataSource' property of the entityManagerFactory bean from `<property name="dataSource" ref="dataSource" />` to `<property name="dataSource" ref="testDatasource" />`.
19. Press Ctrl+s to save your changes.
20. Publish your changes to the test server. Select the test server from the Servers view, right-click, and select **Publish**.
21. Test the application.

The sample application is configured to use a data source that is defined at the application server instead of the Spring managed Apache DBCP connection pool.

### Using OpenJPA instead of Hibernate as the persistence provider

WebSphere Application Server V8.5 uses OpenJPA as the default JPA provider. To use the default JPA provider with our sample application, complete the following steps:

1. Expand the web module and open Java Resources/src/META-INF/persistence.xml. Switch to the **Source** tab.
2. Modify the provider element to point to OpenJPA by replacing `<provider>org.hibernate.ejb.HibernatePersistence</provider>` with `<provider>org.apache.openjpa.persistence.PersistenceProviderImpl</provider>`.
3. Add a non-jta-data source by adding the following line under the provider element: `<non-jta-data-source>jdbc/testds</non-jta-data-source>`
4. Remove all Hibernate specific properties and add their OpenJPA equivalents as follows:
 

```
<property name="openjpa.jdbc.DBDictionary" value="derby" />
<property name="openjpa.RuntimeUnenhancedClasses" value="supported" />
<property name="openjpa.jdbc.SynchronizeMappings" value="buildSchema" />
```
5. Open the applicationContext.xml at Java Resources/src/META-INF/spring folder of the web module. Switch to the **Source** tab of the XML editor.
6. Remove the dataSource property of the entityManagerFactory bean.
7. Press Ctrl+s to save your changes.
8. Publish your changes to the test server. Select the test server from servers view, right-click, and select **Publish**.
9. Test the application.

The sample application is configured to use the built-in OpenJPA provider instead of Hibernate.

## 10.2.7 Summary

While we migrated our sample Spring Roo generated application, we demonstrated the following concepts:

- ▶ Using Rational Application Developer for WebSphere Software as a build and test environment
- ▶ Using WebSphere Application Server Migration Tool
- ▶ Using shared libraries for managing dependencies

The main points to highlight during this migration are:

- ▶ Applications may contain libraries that provide classes that are defined by Java EE specifications, so they are already provided by WebSphere Application Server V8.5. These libraries must be removed from the application before deployment.
- ▶ Using a third-party JPA provider requires changes to the classloading configuration.
- ▶ Using a third-party JSF implementation requires changes in the classloading configuration.
- ▶ Database connection pooling and persistence providers may easily be migrated to their WebSphere provided counterparts.

For detailed preferred practices when you develop Spring and Hibernate applications for a WebSphere Application Server environment, go to:

- ▶ [http://www.ibm.com/developerworks/websphere/techjournal/0609\\_alcott/0609\\_alcott.html](http://www.ibm.com/developerworks/websphere/techjournal/0609_alcott/0609_alcott.html)
- ▶ [http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-n-d-mp&topic=cspr\\_intro](http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-n-d-mp&topic=cspr_intro)





## Part 4

# Migrating from earlier versions of WebSphere Application Server

This part describes how to migrate applications that are written for earlier versions of WebSphere Application Server (starting with Version 5.1) to later versions of WebSphere Application Server. We show you how you can use the Application Migration Tool - WebSphere Version to Version in your migrations.







# Installation and configuration of the Application Migration Tool - WebSphere Version to Version

This chapter provides information for installing the Application Migration Tool - WebSphere Version to Version, which guides you while you migrate your applications to IBM WebSphere Application Server V8.5. We explain the steps to be completed whether you are using Eclipse Marketplace, command-line installation, and Rational Application Developer V8.5.

This chapter describes the following topics:

- ▶ Application Migration Tool - WebSphere Version to Version overview
- ▶ Installing Application Migration Tool on Eclipse

## 11.1 Application Migration Tool - WebSphere Version to Version overview

The Application Migration Tool - WebSphere Version to Version (Application Migration Tool) is part of the IBM WebSphere Application Server Migration Toolkit. This toolkit is part of an overall migration strategy. The Application Migration Tool uses its scanning capabilities to look for specific constructs that changed between versions of WebSphere Application Server that can affect compilation and application server runtime behaviors. The tool provides a way to review problematic code, detailed help on the issue, and in some cases, makes automatic fixes to the code so that the application can run on IBM WebSphere Application Server Versions 7, 8, or 8.5.

There are a number of issues that affect the code migration and development steps when you migrate to newer WebSphere Application Server releases. The migration process can involve modifying Java source code, JavaServer Pages (JSP), and deployment descriptors. These changes include:

- ▶ Changes to the Java Runtime Environment (JRE) encountered in Java SE 5, 6, and 7
- ▶ Removal of previously deprecated features
- ▶ Behavioral changes in the product APIs
- ▶ Changes resulting from Java Enterprise Edition (Java EE) specification clarifications
- ▶ Deprecated features

The migration process can involve modifying Java source code, JavaServer Pages (JSP), and deployment descriptors. The Application Migration Tool helps perform these types of code changes. The Application Migration Tool scans for a number of known issues in applications that are being migrated from WebSphere Application Server Versions 5.1, 6.0, 6.1, or 7.0 to WebSphere Application Server Versions 7.0, 8.0, or 8.5. Where possible, a quick fix is provided to change the application code to address the changes between versions.

For WebSphere Application Server version migration, the tool supports migration from:

- ▶ WebSphere Application Server Version 5.1
- ▶ WebSphere Application Server Version 6.0
- ▶ WebSphere Application Server Version 6.1
- ▶ WebSphere Application Server Version 7.0
- ▶ WebSphere Application Server Version 8.0

to:

- ▶ WebSphere Application Server Version 7.0
- ▶ WebSphere Application Server Version 8.0
- ▶ WebSphere Application Server Version 8.5

### 11.1.1 Application Migration Tool V3.5

Application Migration Tool adds new rules to help migrate applications, including:

- ▶ New Java and XML rules to migrate to WebSphere Application Server Version 8.5
- ▶ New Java SE 7 rules to flag Java 7 compatibility changes if you choose to use Java 7
- ▶ A new dynamic rule set dialog box, which simplifies rule selection and ease of use
- ▶ New depreciation rules

## 11.1.2 WebSphere Application Server V8.5 Liberty profile

WebSphere Application Server V8.5 includes a new Liberty profile server that provides a highly composable, rapid starting, and dynamic application server runtime environment. The Application Migration Tool can be used concurrently with the WebSphere Application Server V8.5 Liberty profile Developer Tools to run applications in Liberty profile environments.

The Application Migration Tool is compatible with the IBM WebSphere Application Server Versions 8.5, 8.0, and 7.0 Developer Tools, so that you can migrate, develop, deploy, and test within a single lightweight Eclipse Integrated Development Environment (IDE).

For an overview of the migration process and Application Migration Tool, see Chapter 4, “Installation and configuration of the Application Migration Tools” on page 89.

## 11.2 Installing Application Migration Tool on Eclipse

The Application Migration Tool - WebSphere Version to Version can be installed as described in Chapter 4, “Installation and configuration of the Application Migration Tools” on page 89. Here we show you a step by step installation using Eclipse Marketplace and the command-line interface (CLI).

**Tip:** Uninstall Versions 1.x, 2.0, or 3.5 Beta of Application Migration Tool before you install Version 3.5. Versions 2.1, 3.0, and 3.1 can be upgraded to Version 3.5.

### 11.2.1 Installing Application Migration Tool using Eclipse Marketplace

To install the Application Migration Tool using Eclipse Marketplace, complete the following steps:

1. Click **Help** → **Eclipse Marketplace**.
2. In the Search tab, Enter WebSphere in the search box and click **Go**.

3. From the search results window (Figure 11-1), click the **Install** button next to the section that includes IBM WebSphere Application Server Migration Toolkit.

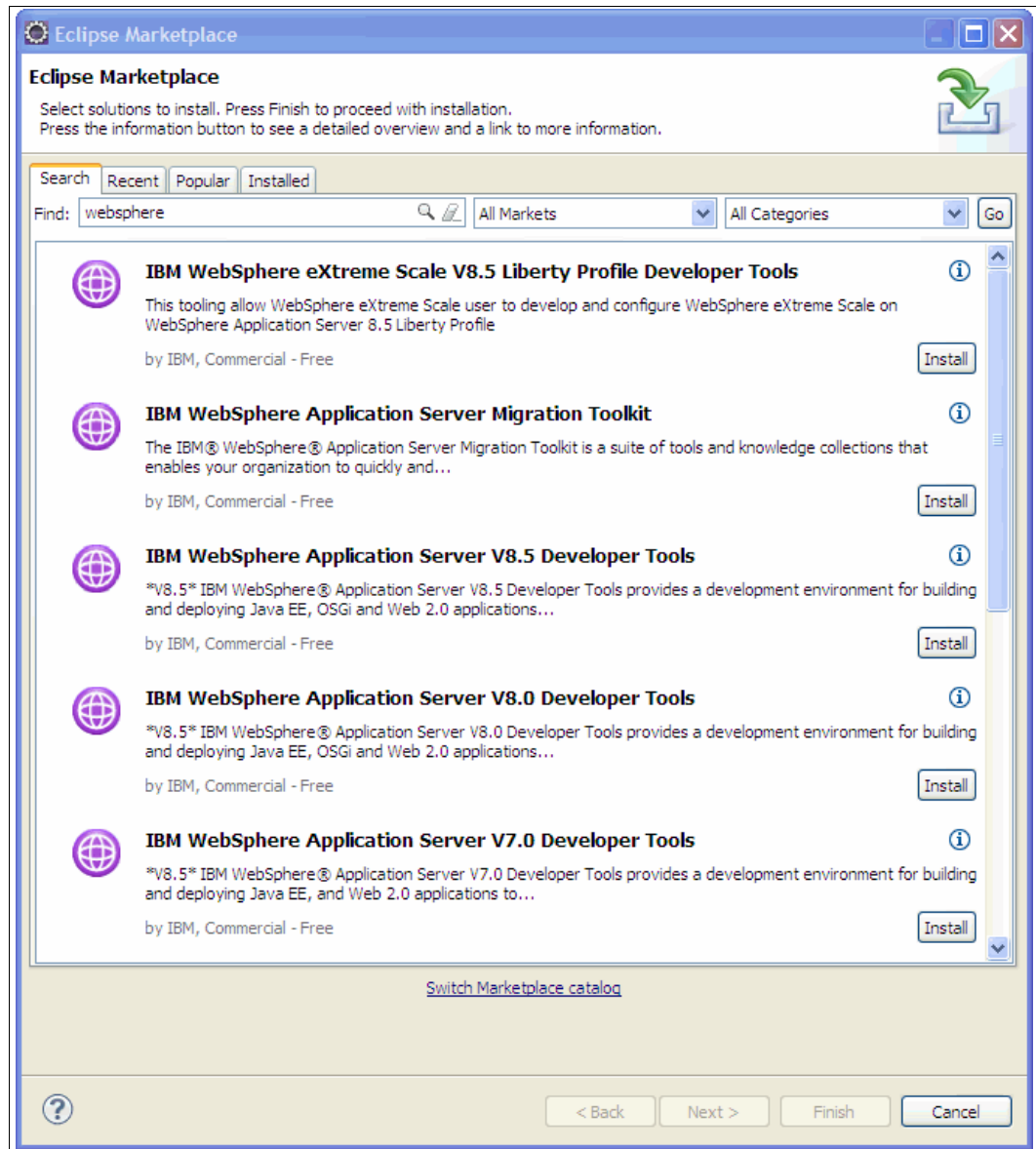


Figure 11-1 Eclipse Marketplace

4. Select the **Application Migration Tool** check box, which selects both the main feature and the other included features, as shown in Figure 11-2.

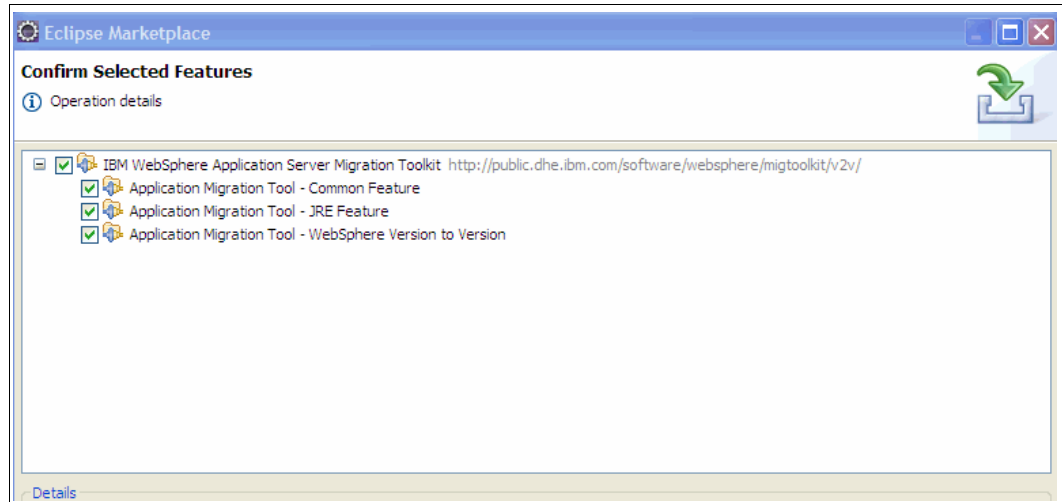


Figure 11-2 Select the features

5. Accept the installation license and start the installation.
6. When the installation completes, restart Eclipse.

## 11.2.2 Installing Application Migration Tool using the CLI

The Migration Toolkit features can be installed using the Eclipse command-line installation facility. Example 11-1 shows the installation of Application Migration Tool - WebSphere Version to Version using the command line.

### Example 11-1 Installation

---

```
C:\Program Files\IBM\SDP\ eclipse.exe
-application org.eclipse.equinox.p2.director
-metadataRepository
jar:file:C:/<amtDownloadDir>/Application_Migration_Tool_WebSphere_Version_to_Versi
on_vn.n.n.zip!/
-artifactRepository
jar:file:C:/<amtDownloadDir>/Application_Migration_Tool_WebSphere_Version_to_Versi
on_vn.n.n.zip!/
-installIU com.ibm.ws.appconversion_feature.was2was.feature.group
-nosplash
```

---

The corresponding uninstall command is shown in Example 11-2.

### Example 11-2 Uninstall

---

```
C:\Program Files\IBM\SDP\ eclipse.exe
-application org.eclipse.equinox.p2.director
-uninstallIU com.ibm.ws.appconversion_feature.was2was.feature.group
-nosplash
```

---

## 11.2.3 Installing Application Migration Tool using Rational Application Developer Version 8.5

This section describes the steps to install and implement the Application Migration Tool using Rational Application Developer V8.5. Complete the following steps:

1. Download the latest version of the Application Migration Tool - WebSphere Version to Version feature from the developerWorks site at [http://www.ibm.com/developerworks/websphere/downloads/migration\\_toolkit.html](http://www.ibm.com/developerworks/websphere/downloads/migration_toolkit.html) and save it locally.
2. From the Rational Application Developer V8.5 menu bar, click **Help** → **Install New Software**.
3. In the Install window, click **Add**.
4. In the Add Site window, enter the following information (Figure 11-3):
  - Name: Enter an appropriate name for the tool, for example, Application Migration Tool
  - Version to Version.
  - Location: Enter the location of the compressed file that you downloaded, for example, C:\Application\_Migration\_Tool\_WebSphere\_Version\_to\_Version\_v3.5.0.zip.

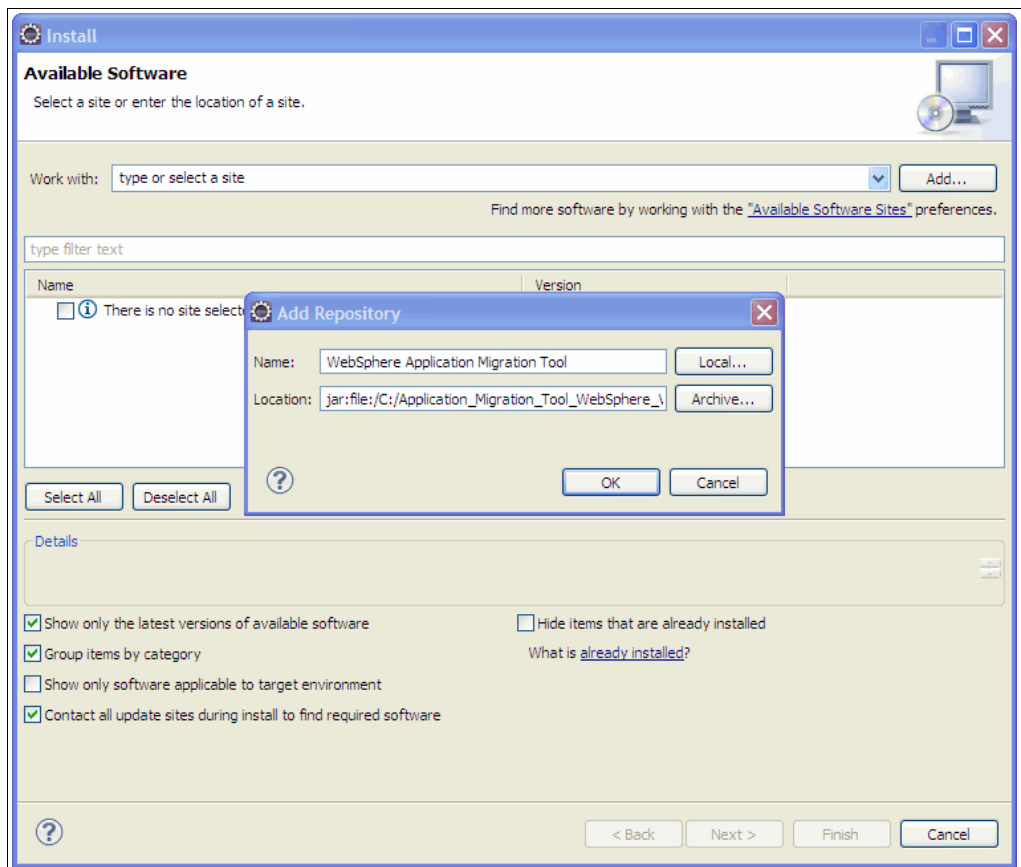


Figure 11-3 Install the Migration Toolkit using Rational Application Developer V8.5

5. In the Install window, select the plug-in that you want to install (Figure 11-4). Click **Next**.

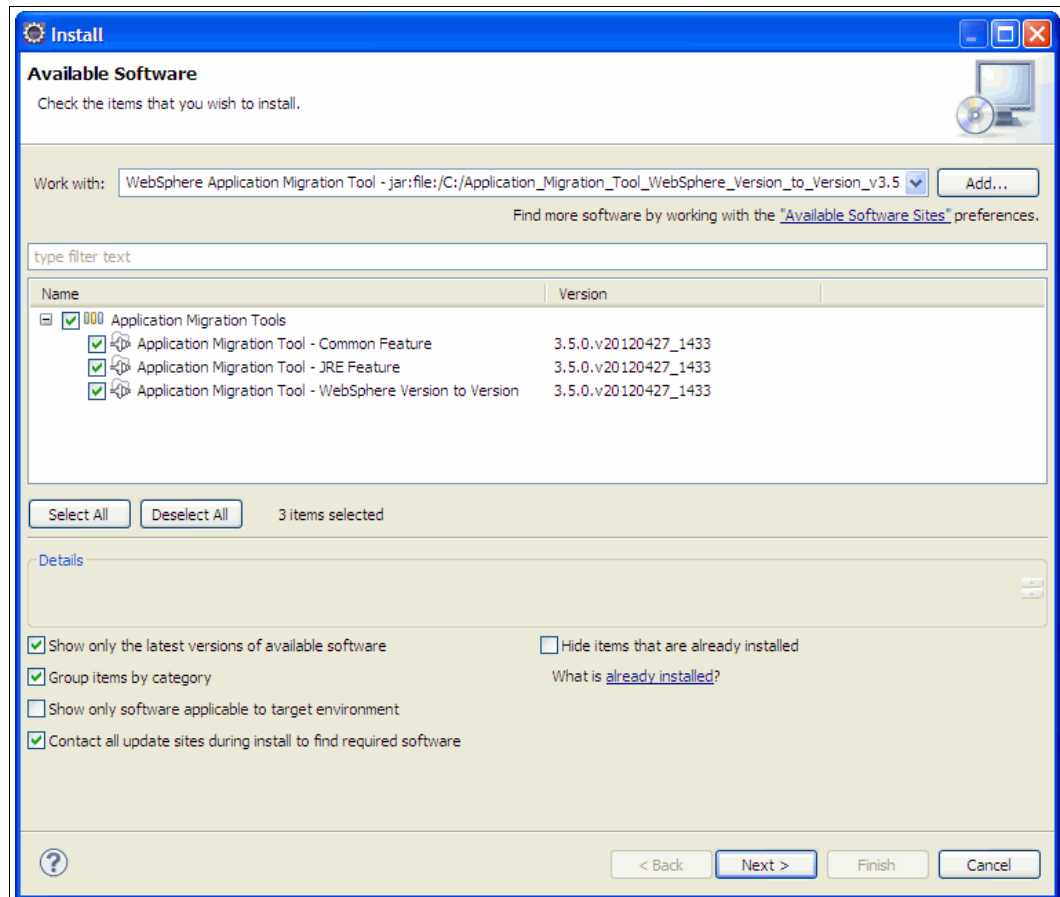


Figure 11-4 Available Software window

6. Accept the license agreement and click **Finish** to begin the installation.
7. After the installation is completed, click **Restart Now** to restart Rational Application Developer.







# Migrating from earlier versions of WebSphere Application Server

This chapter describes migrating applications from earlier versions of WebSphere Application Server, particularly from Version 6.1, Version 7.0, or Version 8.0 to Version 8.5.

This chapter describes the following topics:

- ▶ Concepts
- ▶ Plants by WebSphere sample application migration
- ▶ Migration of Plants By WebSphere
- ▶ Probability Distribution Sample
- ▶ Web services Axis 2 stock quote

## 12.1 Concepts

For most applications, migrating from one version of the WebSphere Application Server to a later version is straightforward and occurs when the version of the application server that is used to host the application is upgraded. However, that upgrade of the installed application server is only a small part of the overall version to version migration effort.

### 12.1.1 The overall migration process

Any migration effort, whether it is a competitive migration or a version to version migration, should have a formal project plan. Figure 12-1 depicts the overall migration process.

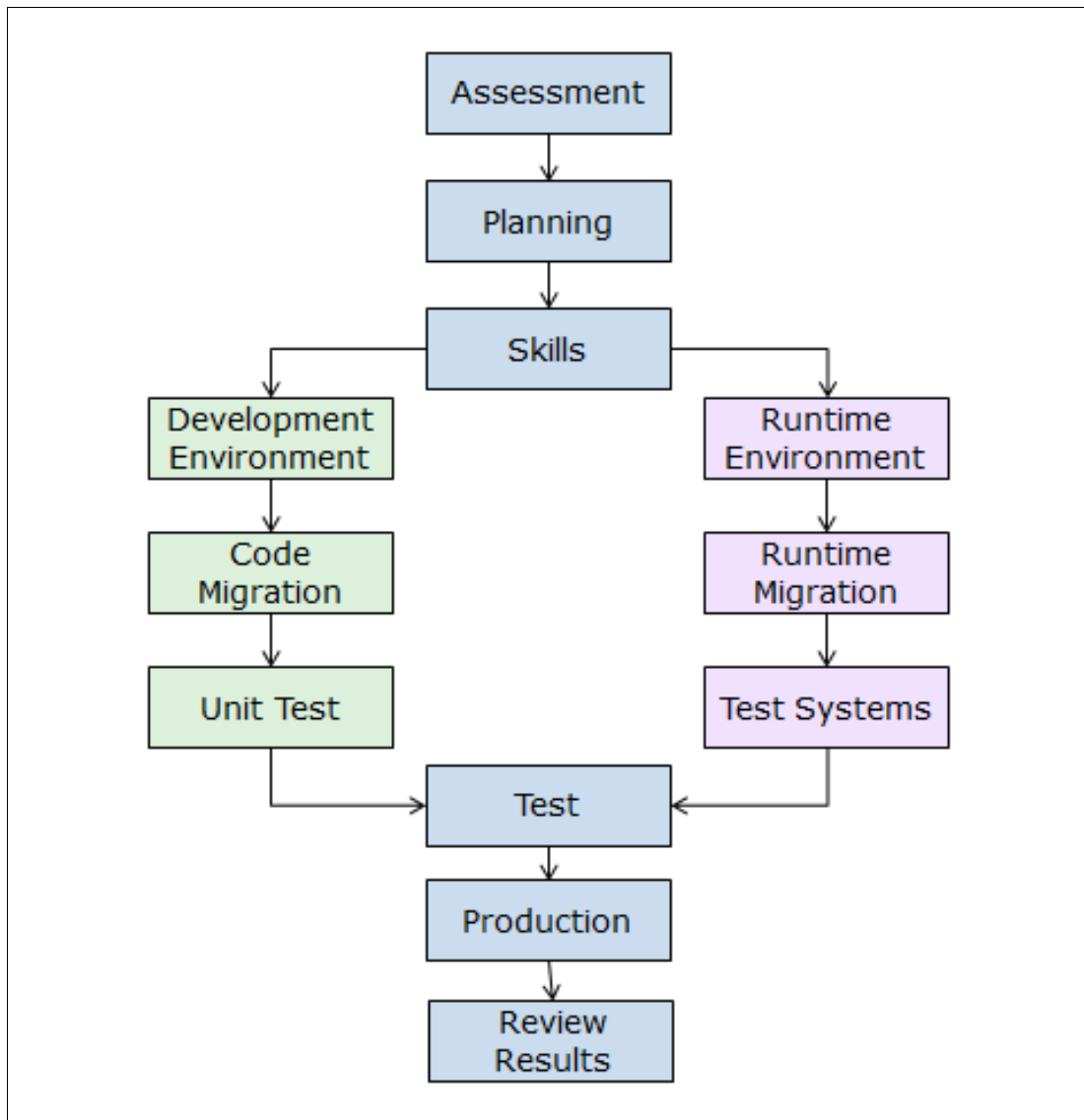


Figure 12-1 Migration process

The overall migration process has several components:

- ▶ Assessment
- ▶ Planning
- ▶ Skills
- ▶ Runtime environment
- ▶ Development environment
- ▶ Test, production, and review

## Assessment

Assessment is the first step of migration planning and is important, as it identifies all factors that have an impact on the migration plan. It is requirements gathering based on each individual company's situation. It has the following steps:

- ▶ Identify the stakeholders and identify a core migration team (especially in the case of large migration projects).
- ▶ Identify the education requirements for both developers and administrators. Hardware requirements should be assessed during this phase and any possible upgrade should be considered. A hardware upgrade should be planned for production systems and any systems that are used for development. A possible hardware upgrade could be 32-bit systems to 64-bit systems.
- ▶ Identify the level of WebSphere Application Server to migrate to. The factors that affect this decision are requirements for any specific Java EE and Java SE version, any requirements of third-party applications or other WebSphere products and any architectural or compatibility changes in WebSphere Application Server. During this step, the vendor applications should be assessed for readiness to execute the version of WebSphere Application Server to which migration is being planned. The application should be assessed for the topology for downtime tolerance and failover support. Then, the application should be assessed for Java EE specification changes and JRE changes.
- ▶ Review the testing practices to determine if there is adequate coverage. The amount of automated testing is also important to ensure repeatable success.

## Planning

A migration plan should be built that takes each of the assessment results into account. It has the following steps:

- ▶ Acquire and plan for upgrading hardware requirements.
- ▶ Upgrade any prerequisite software, such as the operating system and web server levels.
- ▶ Address your education needs.
- ▶ Identify early adopters of the migrated application, so that the lessons learned may be applied to further iterations with other teams. While you work with the early adopters, pilot projects that can be used to drive the migration process should be identified.
- ▶ Create an application rollout strategy. From the information is collected so far, build an execution timeline, taking the availability requirements, the maintenance window, and lockdown periods into consideration.
- ▶ Include and practice a rollback plan before you attempt the migration of production systems. The WebSphere Application Server configuration migration runtime tools and techniques provide a good rollback plan by not destroying the old WebSphere Application Server environment before you install the new WebSphere Application Server environment. This situation enhances the ease of supporting a rollback plan.

## Skills

Various techniques can be used for education, including formal classes and online education. The aspects of education that should be considered are new development tooling, changes in WebSphere Application Server administration model, changes in the new version of WebSphere Application Server, and new standards, such as Java EE.

## Runtime environment

The migration of the runtime environment covers the migration of the application server's configuration. We touch on aspects of this part of the migration process. In addition, the runtime environment migration is covered in more detail in *WebSphere Application Server V7 Migration Guide*, REDP-4635.

Most organizations have more than one runtime environment, including development test, system test, performance, and pre-production environments. For the migration period, you need parallel test environments in most if not all of the test environments so that you can maintain applications that are not migrated, but also to support testing for the new version. It is also important to use some of these test systems to test your migration plan, especially those systems that are the closest to representing the production environment.

## Development environment

A development environment or application migration is one of the most time-consuming parts of the version to version process.

Moving to the next version of WebSphere Application Server may require a change in integrated development environment (IDE). Iteration can be useful in this transition as well.

There is usually no need to upgrade applications to the latest J2EE specification level supported by the version of WebSphere Application Server that you use. Support is also provided to run applications that are written to these earlier level J2EE specifications.

There might be no development that is required to change existing applications, but this situation is not always the case. If changes are required, it is important to try and keep the changes minimal, and make only those changes that are required to get the application migrated and running on the new version. After a successful and stable migration to the WebSphere Application Server, it is safe to use the new features and functions of the latest version. Also, deprecation should be addressed at some point in time.

## Test, production, and review

After the application and runtime environments are initially migrated, the proposed migration must move through the standard test cycle and be put into production. This test cycle should ideally be performed using a set of standard regression tests that are used whenever an application is modified for maintenance or for business upgrade. Automating these tests is a preferred practice.

Another important aspect is to monitor the application's performance. Because of the differences in WebSphere Application Server and JDK versions, ensure that the application performance is not degraded. Have a rollback plan for the production environment.

After this task is accomplished, you are ready for the final step of migration of the production environment.

After the migration is complete, review the migration experience to examine what went correct, what went wrong, and what lessons can be learned. You can then iteratively improve your migration experience with this knowledge and adjustments.

## 12.1.2 Ease of migration

Most applications migrate between versions of the WebSphere Application Server without encountering any problems. Any changes between different versions of the application are now minimized to reduce the set of applications that encounter problems. The more versions that you migrate between can increase the possible number of incompatibilities.

## 12.1.3 Known issues

There is a set of known issues that might cause problems when migrating applications between different versions of the application server. They are included in the Application Migration Tool to provide a programmatic way to identify the required changes in your applications. IBM also maintains a full list of the issues as part of the IBM Migration Knowledge Collection, available at:

<http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27008724>

The following section describes the major development and WebSphere API changes in WebSphere Application Server V8.5, WebSphere Application Server V8, WebSphere Application Server V7, and WebSphere Application Server V6.1.

### Major changes in Version 8.5

This section describes major known issues in Version 8.5 that should be considered during application migration.

#### *Moving to the Java 7 run time*

The following section lists the changes as a result of moving to the Java 7 run time:

- ▶ AWT:
  - The `MouseEvent.getButton()` method may return values outside of the 0 - 3 range in the plan. Previously, the `MouseEvent.getButton()` method returned a value 0 - 3 when the user clicked a button or used the scroll wheel. To accommodate newer models of mice with two scroll wheels, or four and five buttons, the method now returns a value from 0 to the number of buttons.
  - Invoking `Windows.setBackground` may result in an `UnsupportedOperationException` exception. Applications that apply a non-opaque background color to their frames might fail when the application is run on a system that does not support translucency effects.
  - `Toolkit.getPrintJob(Frame, String, Properties)` now throws `NullPointerException`. Before this release, when you invoke `Toolkit.getPrintJob(Frame, String, Properties)` in a headless environment, a `HeadlessException` is thrown instead of the specified `NullPointerException`.
  - Various `Toolkit` methods now throw `HeadlessException`. For example, `Toolkit.isFrameStateSupported(int)`, and `Toolkit.loadSystemColors(int[])`, now throw a `HeadlessException` when used in a headless environment
  - The `sun.awt.exception.handler` system property is replaced with a `Thread.UncaughtExceptionHandler` class.
- ▶ Globalization
  - There is a separation of User Locale and User Interface Locale. The default locale can be independently set for two types of uses: the format setting is used for formatting resources, and the display setting is used in menus and dialogs. The new `Locale.getDefault(Locale.Category)` method takes a `Locale.Category` parameter.

- The UTF-8 implementation is updated so that Corrigendum adheres to Unicode 3.0.1. Previously, there were 5 byte and 6 byte forms of UTF-8 sequences that were allowed. These forms are now rejected.
- ▶ IO
 

The `java.io.File.setReadOnly` and `setWritable` methods have a new behavior. There is no longer a need to set the DOS read-only attribute on directories. This situation means that these methods fail and return false, if the file is a directory. To preserve the relationship with `canWrite`, the `canWrite` method returns true if the file is a directory.
- ▶ JDBC
 

New JDBC methods are introduced, including new methods in Interfaces. There are new methods to support JDBC 4.1. These methods include methods added to the `java.sql.Connection`, `java.sql.Driver`, `javax.sql.CommonDataSource`, and `java.sql.Statement` interfaces.
- ▶ JAXP
  - The `XSLTPProcessorApplet` class was removed. The `XSLTPProcessorApplet` class is an application-level convenience class that had various problems and was removed.
  - JAX-WS Server throws a SOAP Fault when it encounters a DTD. SOAP Message Construct, which is the XML infoset of a SOAP message, must not contain a document type declaration (DTD) information item.
- ▶ Language
  - Previously, `ThreadGroup.setMaxPriority` did not behave as specified if the passed-in value was less than `Thread.MIN_PRIORITY`; it resets the input value to `Thread.MIN_PRIORITY`. The specification states that a value less than `Thread.MIN_PRIORITY` is ignored. The `ThreadGroup.setMaxPriority` method now behaves as specified.
  - `java.lang.Character.isLowerCase/isUpperCase` methods are updated to comply with the specified Unicode definition.
  - The `TypeVisitor` interface is updated. To model the language changes in this release, several updates were made to `javax.lang.model.*`, including adding a method to the `javax.lang.model.type.TypeVisitor` interface.
  - Do not define methods as final on `java.lang.Throwable`. This setting affects classes that extend `Throwable`. Methods `addSuppressed` and `getSuppressed` are added.
- ▶ Networking
 

A server connection shuts down when you attempt to read data when the HTTP Response Code is -1. The HTTP protocol handler closes the connection to a server that sends a response without a valid HTTP status line. When this action occurs, any attempt to read data on that connection results in an `IOException`.
- ▶ Text
 

The `java.text.BreakIterator.isBoundary(int)` method now behaves as specified. The method now returns false, as specified, when the offset is out of bounds, rather than throwing an `IllegalArgumentException`.
- ▶ Utilities
  - The new sort behavior for Arrays and Collections might throw an `IllegalArgumentException`, if it detects a `Comparable` that violates the `Comparable` contract.

- Inserting an invalid element through constructor or puts methods in to a TreeMap or TreeSet throws a NullPointerException. Previously, it was possible to insert invalid null elements and elements not implementing the Comparable interface in to an empty TreeMap or TreeSet. Additional elements cause the expected NullPointerException or ClassCastException. Most other operations upon collection would also fail.
- `Formatter.format()` now throws `FormatFlagsConversionMismatchException` when the "#" flag is specified for conversion "s" and the following argument is not a `Formattable` instance (including the special case "null").

### **JPA changes**

Here are the JPA changes:

- ▶ Change in the JPA cascade strategy: There is a behavior change for entity relationships that use cascade types PERSIST, MERGE, and ALL. The previous release checks the database for the existence of the related Entity before it persists the relationship to that Entity. This action resulted in an extra Select being sent to the database. In the current release, code is added so that when you cascade a persist to a related Entity without a persistence state, the persist (insert) happens without first checking the database. This action might result in an `EntityExistsException` if the related Entity exists in the database. To revert this behavior to the previous release, set the value of the `openjpa.Compatibility` property `CheckDatabaseForCascadePersistToDetachedEntity` to true.
- ▶ JPA MetaModel code generation that concerns `ListAttribute` changed. In previous releases, the MetaModel implementation generated a `ListAttribute` for every array. This behavior is correct if the array is annotated as a `PersistentCollection`, but not correct for unannotated arrays (for example, `byte[]`, `char[]`). This behavior is corrected so that arrays that are not stored as `PersistentCollections` use a `SingularAttribute` instead of a `ListAttribute`. The behavior can be reverted by setting the `Compatibility` property `<UseListAttributeForArrays` to true in `persistence.xml` to the property `name="openjpa.Compatibility" value="UseListAttributeForArrays=true"`.

### **Major changes in Version 8.0**

This section describes major known issues in Version 8.0 that should be considered during application migration.

#### ***Moving to Java EE 6***

WebSphere Application Server V8.0 conforms to Java EE 6, and the migration of applications should consider changes in Java EE 6 when you migrate to V8.0.

#### ***EJB changes***

Table 12-1 describes the EJB changes that must be considered.

*Table 12-1 EJB changes*

<b>Interface</b>	<b>Detail</b>	<b>Old behavior</b>	<b>New behavior</b>
EJB	EJBs in web modules.	Ignored	Now processed. This action might result in latent errors.
@ApplicationException	In EJB 3.1, inherit is a new keyword and defaults to true.	inherit=false	inherit=true. Can impact the subclassing exceptions behavior.

### JAX-WS changes

Table 12-2 describes the JAX-WS changes that must be considered.

Table 12-2 JAX-WS changes

Interface	Detail	Old behavior	New behavior
JAX-WS client	Some calls that result in local exceptions are caused by an invalid host or port.	WebServiceException is thrown on the invoked method with an empty message.	The Handlers handleFault message is called with a SOAPFaultException.
JAX-WS client	Handling of Policy:Addressing in WSDL.	Ignored.	Processed.
SOAPMessage	There is getSOAPHeader and getSOAPBody behavior if there are no headers present.	No error.	Throws a SOAPException.

**JAX\_RPC:** JAX\_RPC is deprecated in Java EE 6 and stabilized by WebSphere Application Server.

### JCA enhancements

Table 12-3 describes the JCA enhancements that must be considered.

Table 12-3 JCA enhancements

Interface	Detail	Old behavior	New behavior
SQL exception error code	Paused data sources have no unique return code.	Always returned 0	2147117569

### JPA changes

Table 12-4 describes the JPA changes that must be considered.

Table 12-4 JPA changes

Interface	Detail	Old behavior	New behavior
EntityManager	The refresh(...) method passes a null.	No error	An IllegalArgumentException exception is thrown.
detach(Object entity)	A new method on numerous APIs, JPA 1.x has a similar method: <T> T detach(<T> pc).	<T> T detach(<T> pc)	Change from <T> T detach(<T> pc) to the new <T> T detachCopy(<T> pc).



### **JSF changes**

Here are the JSF changes:

- ▶ The default JSF implementation is changed from SUN RI to MyFaces. MyFaces is at JSF Version 2.0. SUN RI is left at JSF Version 1.2 and is used only for compatibility with earlier versions. If you want to take advantage of JSF 2.0 features, you must rewrite your JSPs to use Facelets.
- ▶ More exceptions are passed on in JSF 2.0 that would not have surfaced in JSF 1.2.

### **JSP changes**

Table 12-5 describes the JSP changes that must be considered.

*Table 12-5 JSP changes*

<b>Interface</b>	<b>Detail</b>	<b>Old behavior</b>	<b>New behavior</b>
JSP-Property-Groups	The is-xml and page-encoding configuration options should apply only to those JSPs that match the URL pattern.	Matching JSPs AND their included JSPs	Matching JSPs only

### **OSGi changes**

Table 12-6 describes the OSGi changes that must be considered.

*Table 12-6 OSGi changes*

<b>Interface</b>	<b>Detail</b>	<b>Old behavior</b>	<b>New behavior</b>
Composite bundles (.cba)	Scope of bundles that are packaged within composites	Public visibility outside of the composite	Private visibility within the composite

### **SIP changes**

Table 12-7 describes the SIP changes that must be considered.

*Table 12-7 SIP changes*

<b>Interface</b>	<b>Detail</b>	<b>Old behavior</b>	<b>New behavior</b>
SIPFactory	The createRequest and createAddress(String sipAddress) with URI as input or return parameters should be enclosed within brackets (“<>”).	Brackets ignored	Brackets are handled correctly

### **WebSphere removals**

Here are the features that are removed from WebSphere Application Server V8.5:

- ▶ Apache SOAP implementation is removed and standard WebServices support can be used instead.
- ▶ `com.ibm.websphere.ant.tasks.StopServer.setHost (java.lang.String host)`.

- ▶ `com.ibm.websphere.rsadapter.WSConnectJDBCDataStoreHelper`.
- ▶ `com.ibm.websphere.servlet.error.ServletErrorReport.getCause( )`.

## Major changes in Version 7.0

This section describes major known issues in Version 7.0 that should be considered during application migration.

### ***Moving to JRE 6***

The following section describes the impact of moving to JRE 6:

- ▶ Applications using the new language features and JRE 6 can be deployed only to WebSphere Application Server V7.0 nodes. When you compile applications, you can specify '-source' and '-target' modes for earlier JRE targets. For example, '-source 1.4', and 'target 1.4'.
- ▶ Java Serialization is not compatible across JRE 1.4 and earlier releases. Force UUIDs as a preferred practice.
- ▶ Any new features in Java that result in new classes can cause ambiguous references if these new classes match ones that are defined in your program.

For more information, see the Java SE 6 website at:

<http://java.sun.com/javase/6/>

### ***JRE 6 source compatibility***

JRE 6 is generally upwards source-compatible with JRE 5 except for some minimal factors:

- ▶ Some APIs in the `sun.*` packages have changed. These APIs are not intended for use by developers. Developers importing from `sun.*` packages do so *at their own risk*.
- ▶ In some cases, `javac` can now reject previously accepted yet incorrect programs. For example:
  - `javac` correctly rejects invalid casts.
  - EJB business methods that are not declared as public.
  - `FilterMapping` that is mapped to a non-existing servlet.
- ▶ Debug and Profiler interfaces have changed. The Java virtual machine Debug Interface (JVMDI) is removed, and the Java virtual machine Profiler Interface (JVMPDI) is disabled.
- ▶ Non-class files are moved from `rt.jar` in Java SE 6. Java applications that specify `-Xbootclasspath:<path to rt.jar>` and request any resource files fail, because these resources are now in a different JAR file called `resources.jar`.
- ▶ Miscellaneous API changes:
  - The `Duration` and `XMLGregorianCalendar.equals()` methods now return false for a null parameter.
  - `java.beans.EventHandler` enforces valid arguments.

For a complete list of factors, see:

<http://java.sun.com/javase/6/webnotes/compatibility.html>

### ***Moving to Java Platform, Enterprise Edition 5***

The following section describes some specification changes that must be considered when you migrate to Java Platform, Enterprise Edition 5. This section focuses on those interfaces in Version 7.0 that are upgraded to JEE5 with no alternative to using the older support in J2E 1.4.

For example:

- ▶ JSP is upgraded to 2.1 from 2.0.
- ▶ EJB is upgraded to 3.0, but EJB 2.1 applications still run at 2.1.

For complete details about specification level options, see:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.webSphere.base.doc/info/aes/ae/rovr\\_specs.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.webSphere.base.doc/info/aes/ae/rovr_specs.html)

### ***Some JSP 2.1 changes***

Some of the JSP 2.1 changes are listed here:

- ▶ New reserved character: The character sequence '#{ ' is now reserved by JSP. So if '#{ ' is used in the template text or as a literal in an attribute value for a 1.2-based taglib, the sequence must be escaped.
- ▶ Resource injection can cause some JSPs to no longer compile. Large JSPs may now be over the 64 K limit because of new Resource injection support. This support can be turned off by setting the JSP attribute "disableResourceInjection" to true in `ibm-ext-web.xmi` in Version 7.0.0.11 and later.
- ▶ Redefining taglibs in a current scope is no longer supported. To allow the taglib definition, set the JSP attribute '`<jsp-attribute name="allowTaglibPrefixRedefinition" value="true"/>`' or set a webcontainer custom property for the server to '`com.ibm.wsspi.jsp.allowtaglibprefixredefinition=true`'. This custom property affects all applications on the server.
- ▶ A tag library directive that defines a prefix must occur before that prefix is used in a custom tag. To use a prefix before, set a webcontainer custom property for the server to '`com.ibm.wsspi.jsp.allowtaglibprefixusebeforedefinition=true`'. This custom property affects all applications on the server.
- ▶ Multiple occurrences of properties in the `jsp:output` element is no longer supported. To support this configuration, set a webcontainer custom property for the server to '`com.ibm.wsspi.jsp.allowjspoutputelementmismatch=true`'. This custom property affects all applications on the server.

### ***Differences from the V 6.1 feature pack***

This section explains major differences between V6.1 and V7.0 feature packs.

#### ***EJB 3.0 Feature Pack for v6.1***

Here are some of the differences:

- ▶ EJB 3.0 binding file errors: Some applications might not start on WebSphere Application Server V7.0 because uniqueness checks are now performed on names that are used in the EJB 3.0 bindings file.
- ▶ Using runtime JAR files implicitly: When Enterprise beans (EJB) applications are deployed, a runtime `ClassNotFoundException` exception might be thrown because the class path entry Java archive (JAR) file is not exported or published and uses one or more runtime JAR files. This message is only a warning.
- ▶ WebSphere Application Server V7.0 *does not* support the use of bean managed persistence (BMP) and container managed persistence (CMP) entity beans in EJB 3.0-level modules. BMP entity beans are supported in the Feature Pack for EJB 3.0.

### **WebServices Feature Pack for V6.1**

Here are some of the differences:

- ▶ In WebSphere Application Server V7.0, JAX-WS annotations are supported only in modules whose version is Java™ EE 5 or later.
- ▶ In WebSphere Application Server V7.0, JAXB is provided as part of JRE 6. The factory implementation is different in Version 7.0 than in Version 6.1.

### **Both WebServices and EJB Feature packs**

To preserve compatibility with Version 6.1 Feature Packs, you must enable one of the following properties to request scanning during application installation and server startup:

- ▶ Use `UseWSFEP61ScanPolicy` property for Feature Pack for Web Services
- ▶ Use `UseEJB61FEPScanPolicy` property for Feature Pack for EJB 3.0

### **JSF 1.2 Impacts**

This section describes the JSF 1.2 differences:

- ▶ There are some fixes to resolve some previous problems with content-interweaving between JSF and non-JSF tags.
- ▶ ViewHandler extenders are affected. Some custom scripting and some third-party packages must support JSF 1.2, for example, "Tiles". Other examples may be Tomahawk, Trinidad, IceFaces, Facelets, and so on.
- ▶ JSF Portlets report higher memory consumption, resulting in larger sessions.
- ▶ For JSF 1.0 applications, remove the JSF 1.0 JAR files (`jsf-impl.jar` and `jsf-api.jar`) that Rational Application Developer puts in to the application. JSF is now part of the Java EE stack and is a first class technology in the run time.

### **Scanning for annotations**

JEE5 introduces support for annotations. This support requires a new step to scan Java annotations during application installation, which can take a significant amount of time. A web module includes both the classes that are packaged directly within the WAR file (under the `WEB-INF/classes` directory) and classes that are packaged in JAR files within the WAR file.

Applications can be optimized directly by marking modules as JEE5 level when they contain Java Platform, Enterprise Edition 5 content. If the module is known to have no annotations, the "metadata-complete" flag can be used to optimize the scan. Also, the application can be restructured to place utility JAR files, which are known to contain no annotations information, in to shared libraries or the root directory of the EAR file. These files are not scanned for annotations.

For more information, see:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.webSphere.nd.doc/info/ae/ae/urun\\_rapp\\_metadata\\_lockdd.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.webSphere.nd.doc/info/ae/ae/urun_rapp_metadata_lockdd.html)

Configurable filtering can be used in WebSphere Application Server V7.0.0.5 and later versions to identify which modules or Java packages to ignore for annotations processing.

For more information, see:

<http://www.ibm.com/support/docview.wss?uid=swg1PK87053>

With annotations, more file types are processed as part of the scanning, for example, encrypted compressed files that used to be ignored before WebSphere Application Server V7.0. Some problems can be avoided in WebSphere Application Server v7.0.0.7 by completing the following steps:

1. Log in to the administrative console of the Deployment Server and click **Application servers** → **<deployment server name>** → **Java and Process Management** → **Process Definition** → **Control** → **Java Virtual Machine** → **Custom Properties**. Add `com.ibm.websphere.application.migration.disabled` to the list of custom properties, with its value set to true, and save the configuration.
2. Navigate to the `<profile_home>/properties/wsadmin.properties` file, edit it, and add `com.ibm.websphere.application.migration.disabled=true`.

For more information, see:

<http://www-01.ibm.com/support/docview.wss?uid=swg1PK92880>

### **WebSphere API migration details**

The following section provides details about removed features:

- ▶ Derby Network Server Provider, using the Universal JDBC driver, is removed. Derby Network Server using the Derby Client can be used instead.
- ▶ Support for the DB2 CLI-based Type 2 JDBC Driver and the DB2 CLI-based Type 2 JDBC Driver (XA) is removed, and DB2 Universal JDBC Driver should be used instead.
- ▶ WebSphere Connect JDBC driver, Microsoft® SQL Server 2000 Driver for JDBC, and WebSphere SequeLink JDBC driver for Microsoft SQL Server are removed. You can use DataDirect Connect JDBC driver or Microsoft SQL Server 2005 JDBC driver. You can use the **WebSphereConnectJDBCdriverConversion** command to convert data sources from the WebSphere Connect JDBC driver.
- ▶ The Integrated Cryptographic Services Facility (ICSF) authentication mechanism is removed. You can use the Lightweight Third-Party Authentication (LTPA) mechanism.

Table 12-8 describes the support for security custom properties.

*Table 12-8 Support for security custom properties*

<b>Old property</b>	<b>New property</b>
<code>com.ibm.security.SAF.unauthenticatedId</code>	<code>com.ibm.security.SAF.unauthenticated</code>
<code>com.ibm.security.SAF.useEJBROLEAuthz</code>	<code>com.ibm.security.SAF.authorization</code>
<code>com.ibm.security.SAF.useEJBROLEDelegation</code>	<code>com.ibm.security.SAF.delegation</code>

Here are some interfaces that are removed:

- ▶ All classes in the `com.ibm.websphere.servlet.filter` package, including `ChainedRequest`, `ChainedResponse`, `ChainerServlet`, and `ServletChain`, are removed. You can use `javax.servlet.filter` classes.
- ▶ The Web Services Gateway customization API is removed. You can replace the existing filters with a combination of JAX-RPC handlers and service integration bus mediations.

Table 12-9 depicts the miscellaneous classes that are removed.

Table 12-9 Miscellaneous classes that are removed

Old class	New class
com.ibm.websphere.servlet.session.UserTransactionWrapper	Store a UserTransaction directly into the HTTP session.
com.ibm.websphere.rsadapter.DataDirectDataStoreHelper	com.ibm.websphere.rsadapter.ConnectJDBCDataStoreHelper.
com.ibm.websphere.rsadapter.MSSQLDataStoreHelper	com.ibm.websphere.rsadapter.MicrosoftSQLDataStoreHelper.

## Major issues in Version 6.1

This section describes the major known issues in V6.1 that should be considered during application migration.

### Moving to JRE 5

Here are the major considerations for moving to JRE 5:

- ▶ Applications using the new language features and JRE 5 can be deployed only to WebSphere Application Server V6.1 nodes. When you compile applications, you can specify '-source' and '-target' modes for earlier JDK targets. For example, '-source 1.4', and 'target 1.4'.
- ▶ Serialization is not compatible across JRE 1.4 and earlier releases.
- ▶ Any new features in Java that result in new classes can cause ambiguous references, if these new classes match ones that are defined in your program.

### JRE 5 source compatibility

The following section lists major considerations for JRE 5 source compatibility:

- ▶ JRE 5 is generally upwards source-compatible with JRE 1.4.2, except for some APIs in the sun.\* packages that changed. These APIs are not intended for use by developers. Developers importing from sun.\* packages do so *at their own risk*.
- ▶ A variable that is named 'enum' is now a language keyword and some IBM WebServices generated code may include "enum" as package names. If the application is using enum, it no longer compiles under JDK 5 unless "-source 1.4" is specified on the **javac** command.
- ▶ Most source code that uses genericized classes, constructors, methods, and fields continue to compile in JDK 5, though some do not.
- ▶ There might be errors when you import java.net.Proxy.\*, java.lang.reflect.\*, or java.util.Queue classes, because new classes that exist in Java 2 Platform, Standard Edition (J2SE) Version 5 conflict with other package names, such as javax.jms.Queue. When a class name is referenced, provide all of the necessary information, including the base name with the corresponding extension names to resolve the problem.
- ▶ For changes in JAXP, see:  
[http://java.sun.com/j2se/1.5.0/docs/guide/xml/jaxp/JAXPCompatibility\\_150.html](http://java.sun.com/j2se/1.5.0/docs/guide/xml/jaxp/JAXPCompatibility_150.html)
- ▶ In JDBC 5.0, comparing a java.sql.Timestamp to a java.util.Date by invoking compareTo on the time stamp results in a ClassCastException.

- ▶ The `BigDecimal toString()` method behaves differently than in earlier versions. J2SE 5.0 added `toStringPlainString()` to `BigDecimal`, which behaves exactly like the `toString()` method in earlier versions.
- ▶ Direct use of private implementations of XML and XSL parsers is discouraged. Existing classloader support to use an application class path can be used instead of a Java virtual machine bootstrap class path.

### **JRE 5 VerboseGC output**

The GC output format varies depending on the garbage collection policy that is used. Table 12-10 provides the links MustGather information.

Table 12-10 *MustGather table*

<b>Title</b>	<b>Description</b>	<b>URL</b>
MustGather: Analyzing Verbose GC Output for -Xgcpolicy:gencon	Format of verbose GC output from the gencon GC policy	<a href="http://www-01.ibm.com/support/docview.wss?uid=swg21222486">http://www-01.ibm.com/support/docview.wss?uid=swg21222486</a>
MustGather: Analyzing Verbose GC Output for -Xgcpolicy:optavgpaue	Format of verbose GC output from optavgpause GC policy	<a href="http://www-01.ibm.com/support/docview.wss?rs=727&amp;uid=swg21222459">http://www-01.ibm.com/support/docview.wss?rs=727&amp;uid=swg21222459</a>
MustGather: Analyzing Verbose GC Output for -Xgcpolicy:optthruput	Format of verbose GC output from optthruput GC policy	<a href="http://www-01.ibm.com/support/docview.wss?rs=727&amp;uid=swg21222443">http://www-01.ibm.com/support/docview.wss?rs=727&amp;uid=swg21222443</a>

### **JRE 5 and JSSE 2**

The `com.ibm.net.ssl` package is deprecated since JDK V1.4 and is removed in WebSphere Application Server V6.1. It is replaced with the `javax.net.ssl` package. Classes that are related to creating and configuration secure socket factories, such as `KeyManager`, `TrustManager`, `X509KeyManager`, and `X509TrustManager`, are affected. For further details, go to <http://www.ibm.com/developerworks/java/jdk/security/50/> and search for “IBMJSSE2 Guide” and samples.

### **JDK 5 feature usage and JSPs**

As the default runtime compiler setting for JSPs is JDK 1.3, JSPs using any JDK 1.5 specific code receive an error during compilation. There are two solutions:

- ▶ Specify the JDK source level when you install the application.
- ▶ From Rational Application Developer, go to the **Web Extension** tab. From the JSP Attributes table, add “`jdkSourceLevel jdkSourceLevel`” as the name and 1.5 as the value. These actions add an entry to `Web Project/WEB-INF/ibm-web-ext.xml`, for example `<jspAttributes xmi:id="JSPAttribute_1" name="jdkSourceLevel" value="15"/>`.

### **WebSphere removed APIs**

Common Connector Framework (CCF) is deprecated since Version 5.1. Use `Java2Connectors(J2C)` instead. WebSphere Application Server V6.1 dropped Cloudscape V5.1 and replaced it with Cloudscape V10.1. This change could impact the JDBC driver configuration, which must be modified, and because the Cloudscape DBs are not compatible, they must also be migrated.

Mozilla Rhino JavaScript and JDOM are removed from WebSphere Application Server V6.1. Also, the security property `com.ibm.websphere.security.CustomRegistry` is removed. Use `com.ibm.websphere.security.UserRegistry` instead.

For more information, see:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/rmig\\_deprecationlist.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/rmig_deprecationlist.html)

### ***Changes in EJB setRollbackOnly()***

Consider two EJBs, EJB1 and EJB2, that are hosted in the same WebSphere Application Server. EJB1:method1 and EJB2:method2 have Container Managed Transactions and are defined with a transaction attribute of 'TX\_REQUIRED'. An HTTP request invokes EJB1:method1 outside of any transaction (The Enterprise JavaBeans Container for EJB1 creates a transaction) and EJB1:method1 invokes EJB2:method2. (EJB2:method2 runs under the same transaction). When EJB2:method2 invokes setRollbackOnly(), EJB2:method2 returns normally and the HTTP request receives a RemoteException org.omg.CORBA.TRANSACTION\_ROLLEDBACK. However, in WebSphere Application Server V6.0 and earlier, the HTTP request receives the business result of EJB1:method1, not a RemoteException. In each case, the transaction is rolled back, but in WebSphere Application Server V6.1, the HTTP request does not receive the business results returned by EJB1:method1.

### ***JNI on Solaris 10 x86\_64***

WebSphere Application Server V6.0 on Solaris/x86\_64 is built as a 32-bit application (32-bit Java and 32-bit JNI). Customers might have their own applications, which rely on JNI that is not part of WebSphere Application Server, for example, a special database driver. To run in this environment, the JNI must be 32-bit. WebSphere Application Server V6.1 on Solaris/x86\_64 is built as a 64-bit application (64-bit Java and 64-bit JNI). If there are applications on Version 6.0 that use 32-bit JNI that are not part of WebSphere Application Server, those applications cannot be used in WebSphere Application Server V6.1. 32-bit JNI cannot be loaded by a 64-bit JVM. Either JNI should be rebuilt to be 64-bit, or you should contact the vendor of that JNI to obtain a 64-bit version.

For more information, see:

<http://www-1.ibm.com/support/docview.wss?uid=swg2700752>

## **12.2 Plants by WebSphere sample application migration**

The Plants by WebSphere application is an application that is available with the WebSphere Application Server samples gallery. As the name suggests, the application demonstrates several Java Platform, Enterprise Edition functions using an online store that specializes in selling plants, trees, and accessories.

Using the Plants by WebSphere storefront, customers can open accounts, browse for items to purchase, view product details, and place orders. The Plants by WebSphere application uses container-managed persistence (CMP), container-managed relationships (CMR), stateless session beans, a stateful session bean, JSP pages, and servlets.

This application is only intended to showcase the capabilities of WebSphere Application Server itself. Nevertheless, it is useful in many testing and development scenarios. The source code of Plants by WebSphere is distributed with WebSphere Application Server, and, as such, you can modify it as you want.



Plant by WebSphere is included with WebSphere Application Server in all versions greater than 5.0. Although this chapter's instructions can be applied to several versions of WebSphere Application Server, this chapter assumes an installation of WebSphere Application Server V6.1 and, therefore, the provided directory paths and application commands may be incorrect if used with earlier versions.

The directory location of the source is <WAS\_DIR>\samples\bin\PlantsByWebSphere. The code can be modified in any development environment. However, it may be more convenient to take advantage of the functionality that is provided by an IDE that supports J2EE applications.

For example, Eclipse V3.2 and later with the WebSphere Application Server Developer Tools work well because you can import the source code easily while you offer seamless integration with WebSphere Application Server. Further information about this environment can be found at the Eclipse Foundation Webtools website or WebSphere developer tools website. To download the tool, go to:

[https://www.ibm.com/developerworks/mydeveloperworks/blogs/wasdev/entry/download\\_wdt?lang=en](https://www.ibm.com/developerworks/mydeveloperworks/blogs/wasdev/entry/download_wdt?lang=en)

## 12.2.1 Overview of the application

The Plants by WebSphere Sample incorporates the following technologies:

- ▶ Container-managed persistence (CMP) entity beans
- ▶ Bean-managed persistence (BMP) entity beans
- ▶ Stateless session beans
- ▶ Stateful session beans
- ▶ Servlets
- ▶ JavaServer Pages (JSP) files and HTML
- ▶ Container-managed relationships (CMR)
- ▶ Java 2, Enterprise Edition (J2EE) security
- ▶ Java API for XML-based remote procedure call (JAX-RPC)

The Plants by WebSphere application is supported through a series of JSP pages and HTML pages. These pages communicate with the following servlets: AccountServlet, ShoppingServlet, ImageServlet, and AdminServlet. The servlets use the various enterprise bean business methods, which in turn access data from the database as needed. In general, stateless session beans are used to interface with the entity beans, to reduce the number of transactions.

Figure 12-2 illustrates the high-level architecture of the application.

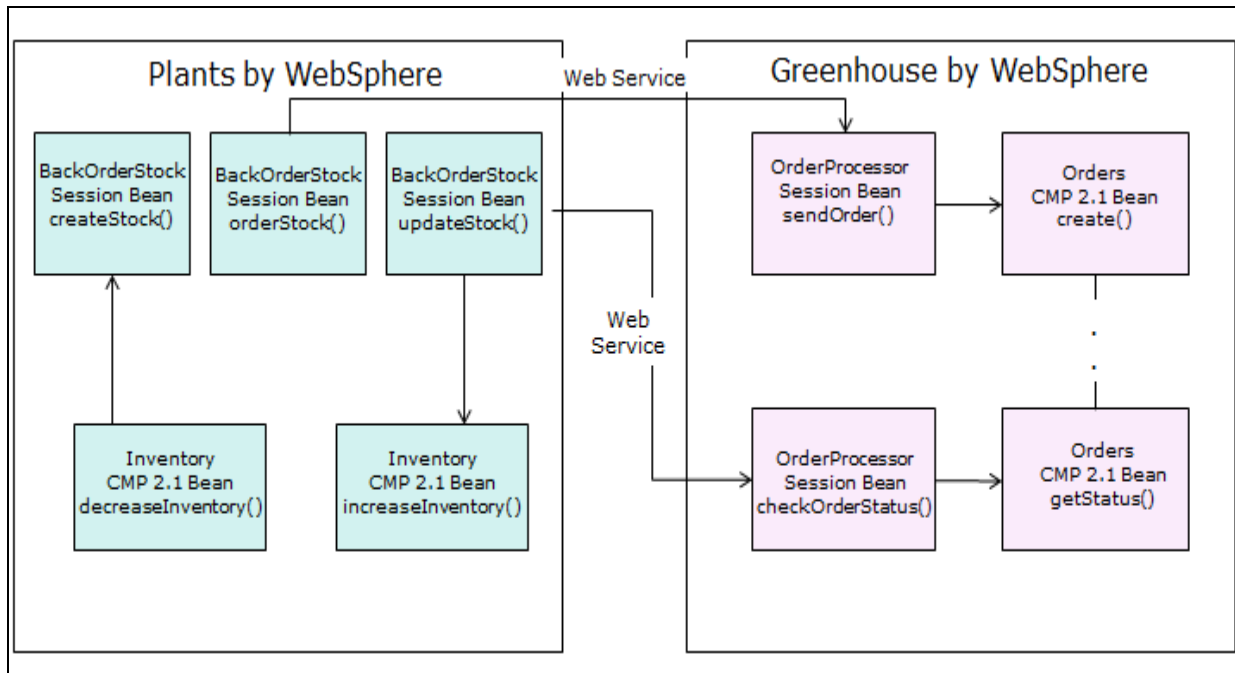


Figure 12-2 Plants by WebSphere application high-level architecture

## Servlets

This section provides information about the servlets that are used in the Plants by WebSphere application.

### ► ShoppingServlet

The ShoppingServlet servlet processes most of the interactions between the browser and the enterprise beans. This interaction includes inventory, shopping cart, and order functions. Selecting a shopping category to browse uses a Catalog session bean, which is used to find and display all of the relevant Inventory objects. Displaying the details of an item can also be done by using a Catalog session bean to obtain information from an Inventory CMP entity bean.

Adding an item to the shopping cart creates a ShoppingCart stateful session bean. A Catalog session bean obtains the Inventory data, and places the item in the ShoppingCart bean. Viewing and updating the shopping cart is done using the ShoppingCart stateful session bean. After you enter billing and shipping information, the ShoppingCart bean creates an Order CMP entity bean. Upon completing the checkout procedure, a Mailer stateless session bean is created to send a confirmation email, using the JavaMail API.

### ► ImageServlet

The ImageServlet servlet obtains and places product images into the database. The servlet obtains images from the JSP pages and the HTML pages from the database and serves back to the browser through the HTTP response.

### ► AdminServlet

The AdminServlet servlet processes the requests from a user (browser) for inventory back orders, database repopulation, and supplier configuration. During a customer checkout procedure, the inventory quantity is checked. When the inventory quantity reaches a minimum threshold, the Inventory bean creates a BackOrder CMP entity bean. An administrator can process this back order in the Plants by WebSphere administrative pages.

When processing requests from the `supplierconfig.jsp` page, the `AdminServlet` servlet creates a `Suppliers` stateless session bean, which is used to pass requests to a `Supplier CMP 2.0` bean. The `Supplier` configuration information includes name, address, phone, and a web service web address. The backorder requests are sent to the `Supplier` through a web service that is at this web address.

When it processes repopulation requests from the `help.jsp` page, the `AdminServlet` servlet creates a bean-managed persistence (BMP) bean called `ResetDB`. This bean deletes all database tables and repopulates the tables with initial data values from the `pbw.properties` properties file.

## Enterprise beans

This section provides information about the enterprise beans that are used in `Plants by WebSphere` application.

### ► Catalog

`Catalog` is a stateless session bean. It is the primary access to the `Inventory` container-managed persistence entity bean (CMP). Stateless session beans generally access entity bean data, while limiting the number of transactions used. The `Catalog` session bean has business methods to obtain information from one or more `Inventory` beans. Methods exist to add and remove an `Inventory` item. The `Catalog` session bean also has methods to modify existing `Inventory` beans.

### ► Customer

`Customer` is a 2.0 CMP entity bean. It contains and manages the account data that is needed for a customer. The `Customer` entity bean has methods for creating, finding, and updating customer information, and verifying a password and getting fields in the `Customer` entity bean.

### ► Inventory

`Inventory` is a 2.0 CMP entity bean. This entity bean contains and manages `inventory` item data. Methods are available for finding, creating, getting, and setting data fields.

### ► Login

`Login` is a stateless session bean. This session bean interfaces with the `Customer` CMP entity bean to create and update customer accounts.

### ► Mailer

`Mailer` is a stateless session bean. This session bean creates and sends an order confirmation email using the `JavaMail` API.

### ► Order

`Order` is a CMP entity bean. This session bean contains and manages order data. Methods are available for finding, creating, and getting data fields.

### ► OrderItem

`OrderItem` is a CMP entity bean. This entity bean contains and manages a single order item. Methods are available for finding, creating, and getting data fields.

### ► ReportGenerator

`ReportGenerator` is a stateless session bean. This session bean generates reports that are based on information that is found in orders. The `ReportGenerator` session bean interfaces with the `Order` CMP entity bean. The reports consist of top-selling items for a set date range, and top-selling postal codes for a set date range.

- ▶ **ShoppingCart**  
ShoppingCart is a stateful session bean. This session bean maintains a list of inventory items to purchase throughout the HTTP session. The ShoppingCart session bean has business methods to add, remove, and update inventory items. The ShoppingCart session bean also has a method to create an Order CMP entity bean when the customer is ready to complete a purchase.
- ▶ **BackOrder**  
BackOrder is a CMP entity bean. This entity bean contains and manages inventory backorder data. Methods are available for finding, creating, and getting data fields.
- ▶ **BackOrderStock**  
BackOrderStock is a stateless session bean. This session interfaces with the BackOrder CMP bean to create and process inventory backorder items.
- ▶ **Supplier**  
Supplier is a CMP entity bean. This entity bean contains and manages supplier configuration data. Methods are available for finding, creating, and getting data fields.
- ▶ **Suppliers**  
Suppliers is a stateless session bean. This session interfaces with the Supplier CMP bean to create and process Supplier configuration information.
- ▶ **ResetDB**  
ResetDB is a BMP entity bean. This entity bean deletes all rows in the database tables.

## 12.2.2 Modifying and rebuilding the Plants by WebSphere sample

The sample applications that are shipped with WebSphere Application Server V 6.1 are available in the following directories. Make sure that the samples are installed as part of the WebSphere Application Server V6.1 installation.

- ▶ `<profile_root>samples/src/PlantsByWebSphere`: Contains the Sample source code for the Plants by WebSphere sample.
- ▶ `<profile_root>/samples/lib/PlantsByWebSphere`: Contains the Web archive (WAR) files, Java archive (JAR) files, and the final `PlantsByWebSphere.ear` file for the Plants by WebSphere sample.
- ▶ `<profile_root>/samples/javadoc/PlantsByWebSphere`: Contains the Javadoc documentation that is generated when the Plants by WebSphere sample builds.
- ▶ `<profile_root>/samples/bin/PlantsByWebSphere`: Contains a `wsadmin jac1` script to configure and install the PlantsByWebSphere sample application.

**z/OS customers:** The source code tree for samples is not provided on the z/OS platform because sample applications are not built on the z/OS platform.

The Plants by WebSphere sample is installed with WebSphere Application Server if samples are chosen to be installed during product installation. You do not have to build the Plants by WebSphere sample before you use it. The following sections describe the steps to rebuild the sample, along with more information for modifications.

**Modified sample:** The Plants By WebSphere sample from WebSphere Application Server Version 6.1 is modified to suit the purpose of demonstrating the migration process. You can download the application with this minor modification by following the instructions in Appendix B, “Additional material” on page 435.

### Running the sample build script

To run the Sample build script, which rebuilds the sample, complete the following steps:

1. Open a command prompt.
2. At the command line, change the directory to the `<profile_root>/samples/src/PlantsByWebSphere` directory.
3. Run the Sample build script by running the command that is shown in Example 12-1.

*Example 12-1 Command to build the sample application*

---

On Windows development platforms, type the following command:  
`<profile_root>\samples\bin\PlantsByWebSphere\buildplantsby.bat`

On UNIX and Linux development platforms, type the following command:  
`<profile_root>/samples/bin/PlantsByWebSphere/buildplantsby.sh`

---

The sample is now built.

**Build process:** During the build process, only class files that have changes are rebuilt.

### 12.2.3 Removing and reinstalling the PlantsByWebSphere.ear file

By default, the Plants by WebSphere sample installs in to the `profile_root/installedApps/<host name>` directory, where `<host name>` is the name of the host where your application is installed. Reinstall the sample by completing the following steps:

1. Stop the application server.
2. Remove the Plants by WebSphere Sample EAR file by running the commands that are illustrated in the following examples.

Example 12-2 illustrates the commands run on a Windows operating system.

*Example 12-2 Command to uninstall the PlantsByWebSphere sample application on Windows*

---

```
cd <profile_root>\bin
    setupCmdLine
where <profile_root> represents the profile installation directory.

wsadmin -conntype none -c "$AdminApp uninstall PlantsByWebSphere"
```

Note: `install_root/bin` must be in set in your PATH

---

Example 12-3 illustrates the commands run on a Linux operating system.

*Example 12-3 Command to uninstall the PlantsByWebSphere sample application on Linux*

---

```
cd <profile_root>/bin
. ./setupCmdLine.sh
where <profile_root> represents the profile installation directory.

wsadmin.sh -conntype none -c "\$AdminApp uninstall PlantsByWebSphere"
```

---

3. Reinstall the Plants by WebSphere sample by completing the following steps:
  - a. Change the directory to the profile\_root/samples/lib/PlantsByWebSphere directory.
  - b. Locate the PlantsByWebSphere.ear file.
  - c. Run the commands that are illustrated in following examples.

Example 12-4 illustrates the command to install the Plants By WebSphere sample application on a Windows operating system.

*Example 12-4 Command to install the PlantsByWebSphere sample application on Windows*

---

```
install_root/bin/wsadmin -conntype none -c "$AdminApp install
PlantsByWebSphere.ear
{-appname PlantsByWebSphere -installdir
$(APP_INSTALL_ROOT)/$(CELL) -usedefaultbindings -node <node> -deployejb
-deployejb.dbtype DERBY_V10}"
```

where <node> is the name of node on which to install the Sample.

---

Example 12-5 illustrates the command to install the Plants By WebSphere sample application on Linux operating system.

*Example 12-5 Command to install the PlantsByWebSphere sample application on Linux*

---

```
install_root/bin/wsadmin.sh -conntype none -c "\$AdminApp install
PlantsByWebSphere.ear
{-appname PlantsByWebSphere -installdir
\$(APP_INSTALL_ROOT)/\$(CELL) -usedefaultbindings -node <node> -deployejb
-deployejb.dbtype DERBY_V10}"
```

---

4. Start the application server.
5. Run the application by accessing the following URL:

`http://localhost:9081/PlantsByWebSphere`

Where:

- localhost should be replaced by the host name or IP address of the system on which WebSphere Application Server is up and running.
- 9081 should be replaced by the port number on which the application is running.

The rebuilt PlantsByWebSphere.ear file is now deployed to your WebSphere Application Server.

## 12.2.4 Installing Plants By WebSphere from the administrative console

This section describes the installation procedure to install the Plants By WebSphere sample application using the administrative console.

Complete the following steps:

1. Log in to AdminConsole and browse to **Applications** → **Install New Application**.
2. Browse to the location where the rebuilt `PlantsByWebSphere.ear` file is.
3. Provide the application name, location, and other details (Figure 12-3).

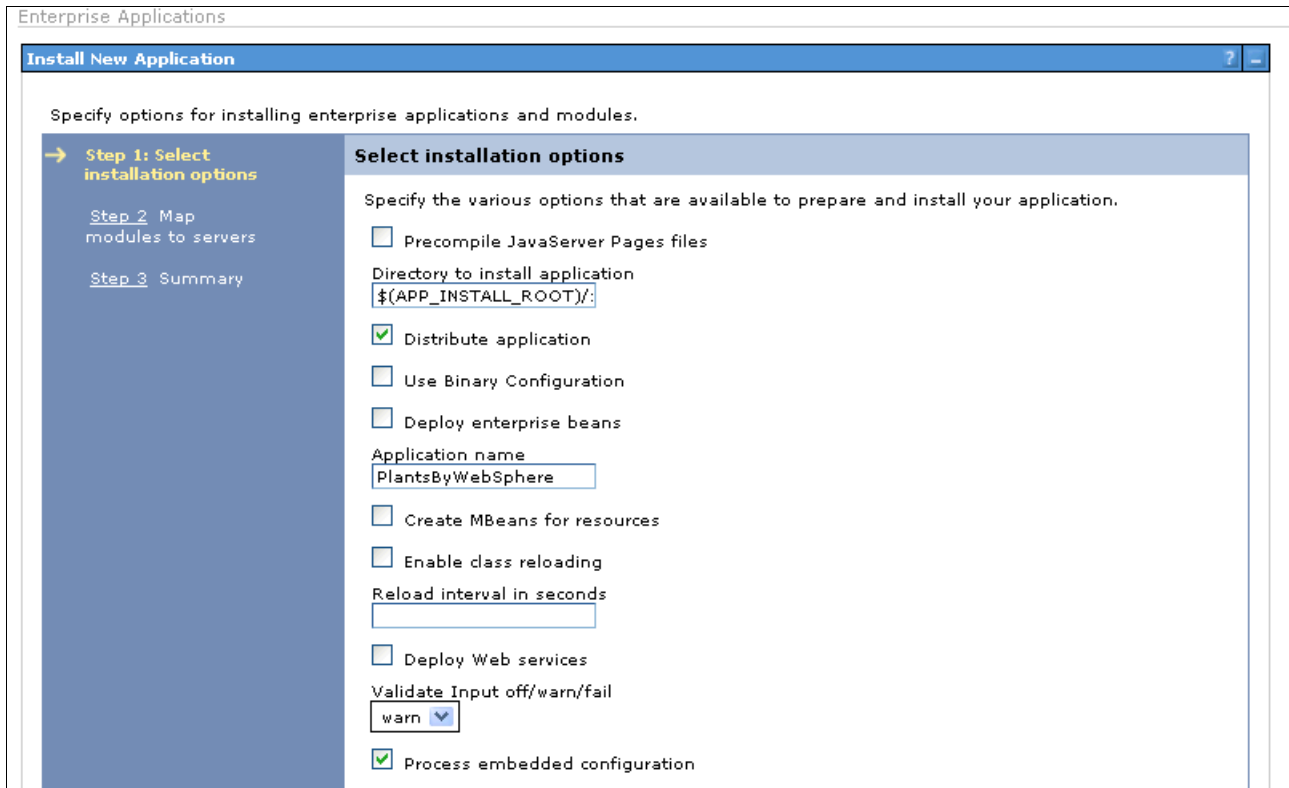


Figure 12-3 Step 1: Select installation options

4. Map the Application Modules to the target Application Server (Figure 12-4).

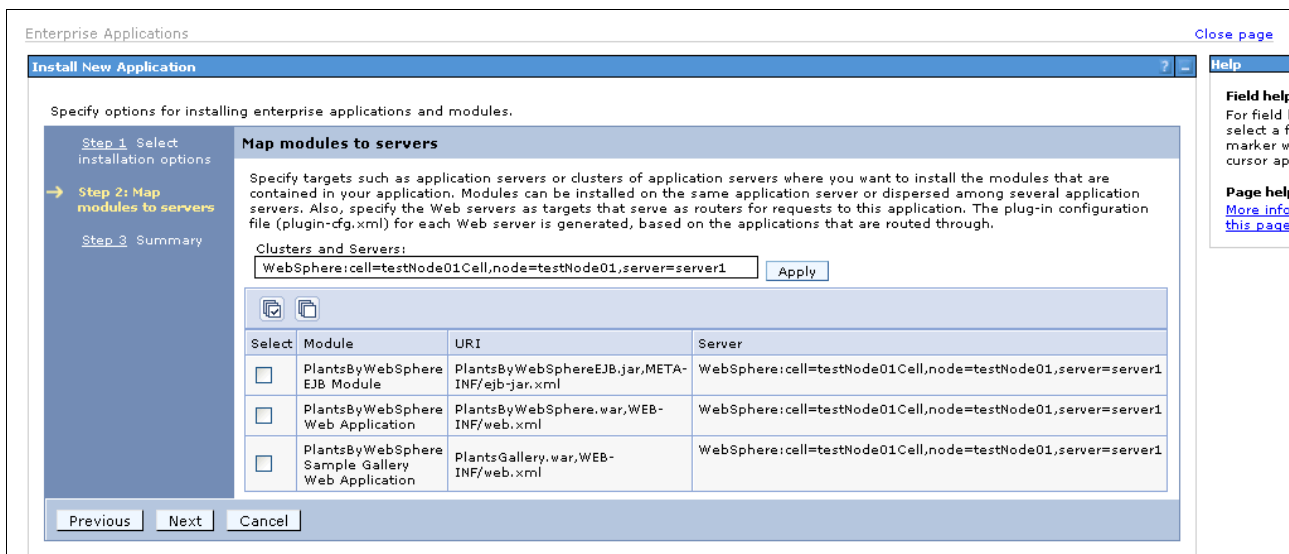


Figure 12-4 Step 2: Map modules to servers

5. Review the summary and finish the installation.

### Optional configuration of email resources

To send an order confirmation email, you must configure a mail session for the Plants by WebSphere sample. If you do not want to send an order confirmation email, skip this section.

To configure mail resources, complete the following steps:

1. In the administrative console, click **Resource** → **Mail Providers** → (set scope to **Node <node>**) → **Built-in Mail Provider** → **Mail Sessions**.
2. Click **New** to create a mail session.
3. Enter PlantsByWebSphere mail session in the Name field.
4. Enter mail/PlantsByWebSphere in the JNDI Name field.
5. Enter a value, such as yourcompany.com, in the Mail Transport Host field.
6. Enter a value, such as userid@yourcompany.com in the Mail From field.
7. Click **OK**.
8. Click **Save** at the top of the main pane in the administrative console.
9. Click **Save** in the Save to Master Configuration window.
10. Restart WebSphere Application Server so that all configuration changes are applied.

Mail resources for the Plants by WebSphere sample are now configured.

### Locating the database tables

The database tables for this sample are CUSTOMER, INVENTORY, ORDERINFO, ORDERITEM, IDGENERATOR, BACKORDER, and SUPPLIER. These tables are found in the PLANTSDB database. The PLANTSDB database is in the PlantsByWebSphere.ear file, and is in the

profile\_root/installedApps/cellname/PlantsByWebSphere.ear/Database/PLANTSDB directory, where cellname is the name of the cell where your application is installed.

## 12.2.5 Running the sample application

Start the application and access Plans By WebSphere by going to the following URL:

`http://localhost:9081/PlantsByWebSphere`

Where:

- ▶ localhost should be replaced by the host name or IP address of the system on which WebSphere Application Server is up and running
- ▶ 9081 should be replaced by the port number on which the application is running.



The main index page opens (Figure 12-5).



Figure 12-5 Index page of Plants By WebSphere sample

A unique email address and password for the Plants by WebSphere sample can be created by clicking **Register** on the Login window (Figure 12-6).

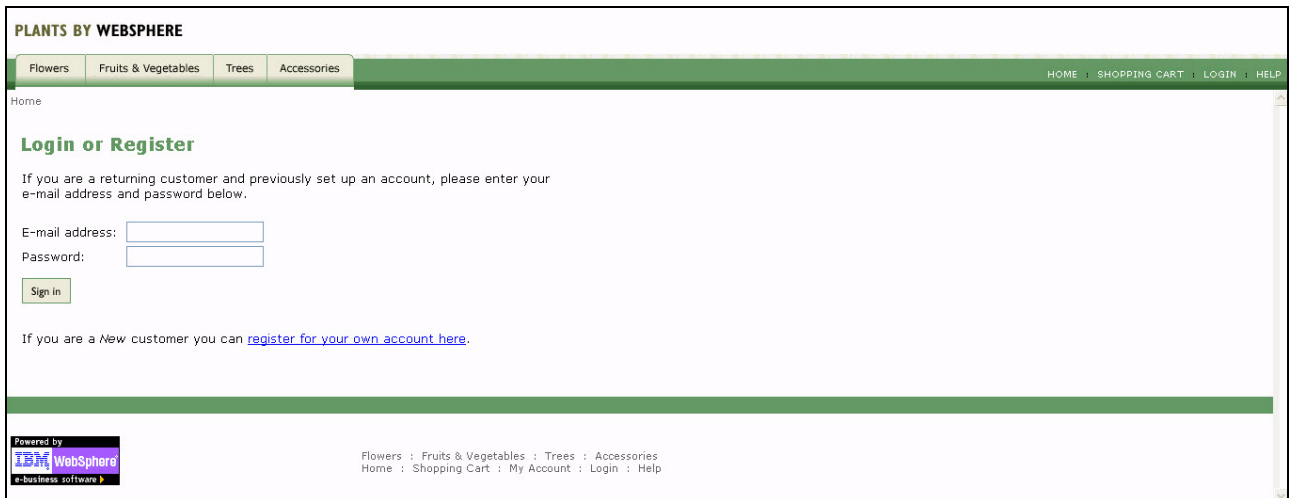


Figure 12-6 Login/Registration page of Plants By WebSphere sample

Once logged in to the account, the customer can start shopping and can add items into the shopping cart (Figure 12-7).

ITEM #	ITEM DESCRIPTION	PACKAGING	QUANTITY	PRICE	SUBTOTAL
T0003	<b>Bonsai</b>	0.5 gallon mature tree	<input type="text" value="1"/>	\$30.00	\$30.00

**Order Subtotal:\$30.00**

Figure 12-7 Shopping cart of Plants By WebSphere

After the required items are added to the shopping cart, the customer can proceed to check out by providing the billing and shipping information. A message is shown with the order number and expected date of arrival for the items purchased.

## 12.3 Migration of Plants By WebSphere

This section provides detailed information about the process of migrating the Plants By WebSphere sample application using the WebSphere Application Migration Toolkit - WebSphere Version to Version migration.

### 12.3.1 Exporting PlantsByWebSphere.ear and importing it into Eclipse for analysis

This section describes details about exporting the PlantsByWebSphere.ear file and importing it in to Eclipse.

#### Exporting the PlantsByWebSphere.ear file

Complete the following steps:

1. Log in to the Administration Console and browse to **Enterprise Application**.
2. Select the **PlantsByWebSphere** application.

3. Click **Export** and download the `PlantsByWebSphere.ear` file (Figure 12-8).

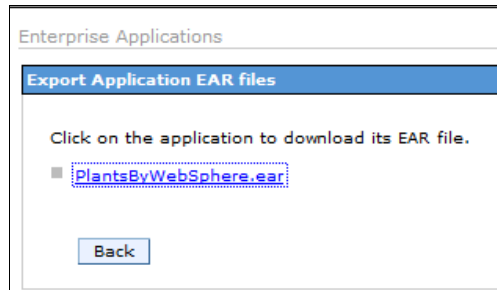


Figure 12-8 Exporting the `PlantsByWebSphere.ear` file from the WebSphere Application Server administration console

4. Optionally, you can also get the EAR file from `<profile_root>/samples/lib/PlantsByWebSphere</code>.`

### Importing the `PlantsByWebSphere.ear` file

Start the migration of the Plants By WebSphere application by importing the EAR file with the source code into Eclipse Java EE IDE.

**Note:** The Application Migration Tool requires the presence of the source code to perform Java code analysis.

Complete the following steps:

1. From the Eclipse Java EE Enterprise Explorer perspective, click **File** → **Import** → **EAR File** (within Java EE) and click **Next**. You import an Enterprise Application project that is based on the selected resources, as illustrated in Figure 12-9.

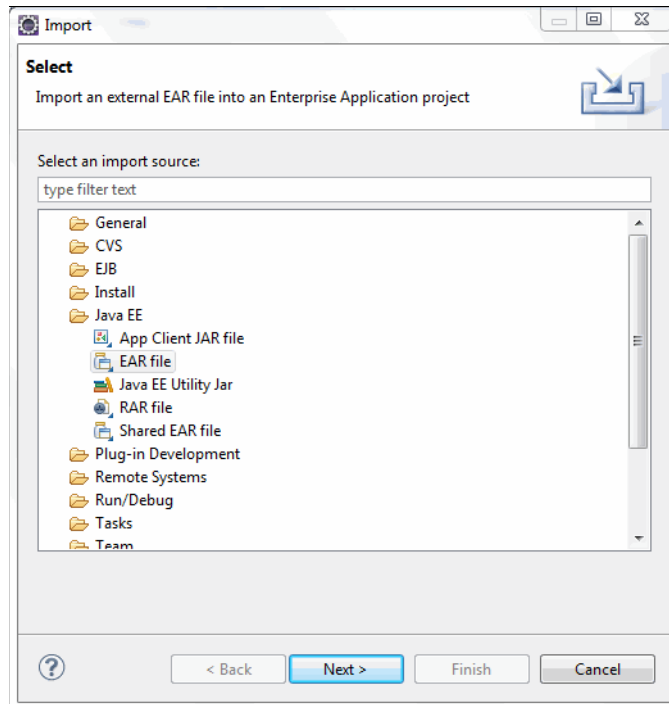


Figure 12-9 Importing the EAR file

2. Click **Browse** and select the `PlantsByWebSphere.ear` file. In our example, it is in `C:\test\PlantsByWebSphere.ear` (Figure 12-10).

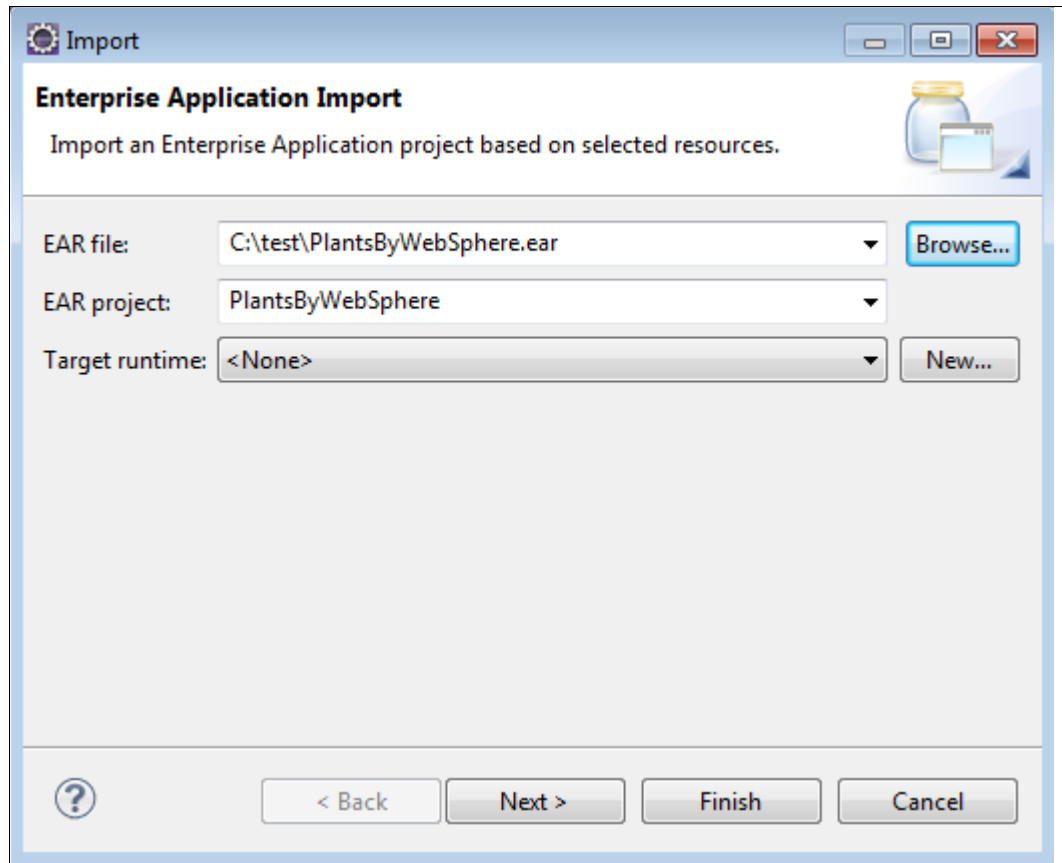


Figure 12-10 Select the EAR file to import

3. In the Ear Module and Utility JAR Projects window, make sure that all three modules are selected (Figure 12-11) and click **Finish**.

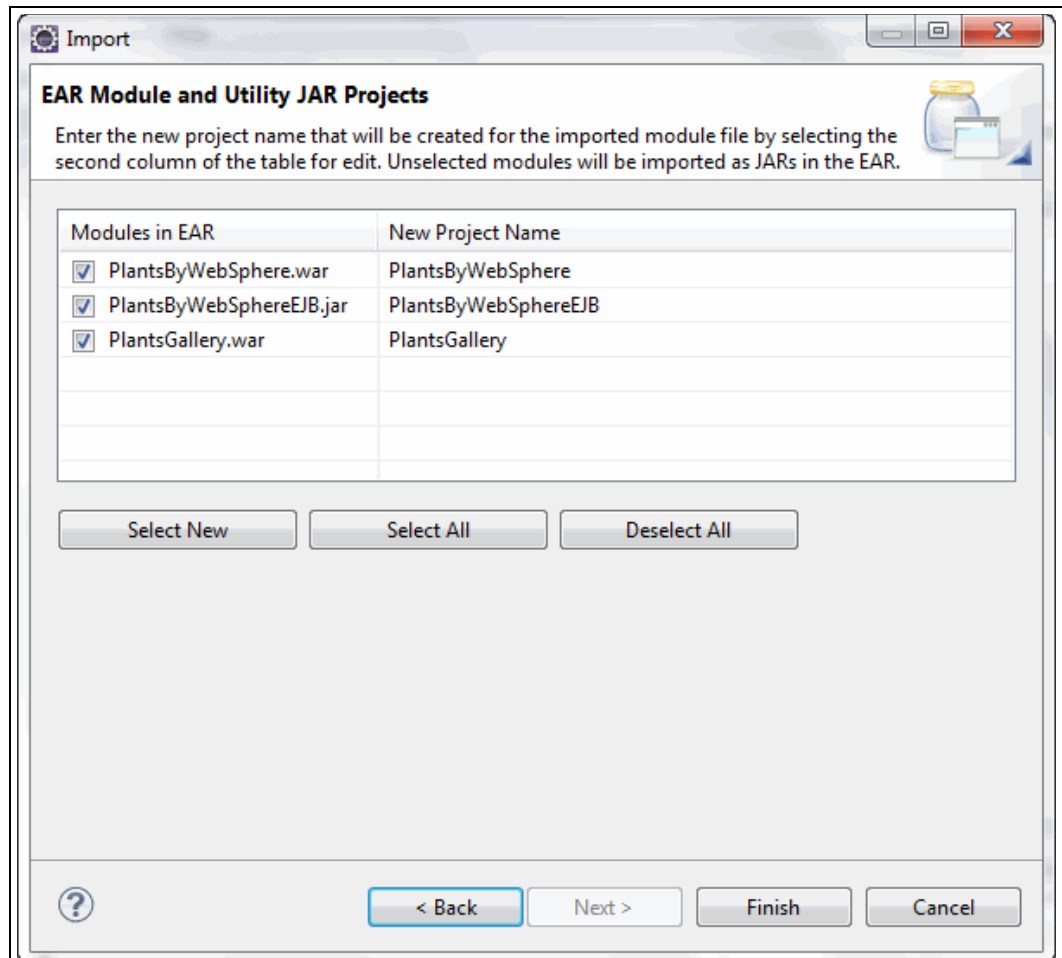


Figure 12-11 EAR modules and Utility JAR projects during application import

### 12.3.2 Configuring and running the Application Migration Tool - WebSphere Version to Version

In this section, we configure the Application Migration Tool analysis capabilities to scan the WebSphere Application Server V6.1 Plants By WebSphere application and provide input about what must be fixed in the application before it can run on WebSphere Application Server V8.5.

The Application Migration Tool must be set up to look for Java 6 or Java 7 code-related issues. However, it can also be run to do a complete Java code review. The tool provides quick fixes, where available, for issues it flags as violations of the migration rules.

**Quick fixes:** The tool does not provide quick fix for every possible scenario. Whenever a rule violation occurs and there is no quick fix available, the rule icon includes a light bulb icon and the Quick Fix option is not available in the context menu.

Complete the following steps:

1. Select **Analysis** from the **Run** menu to display the Application Migration Tool main configuration dialog box (Figure 12-12).

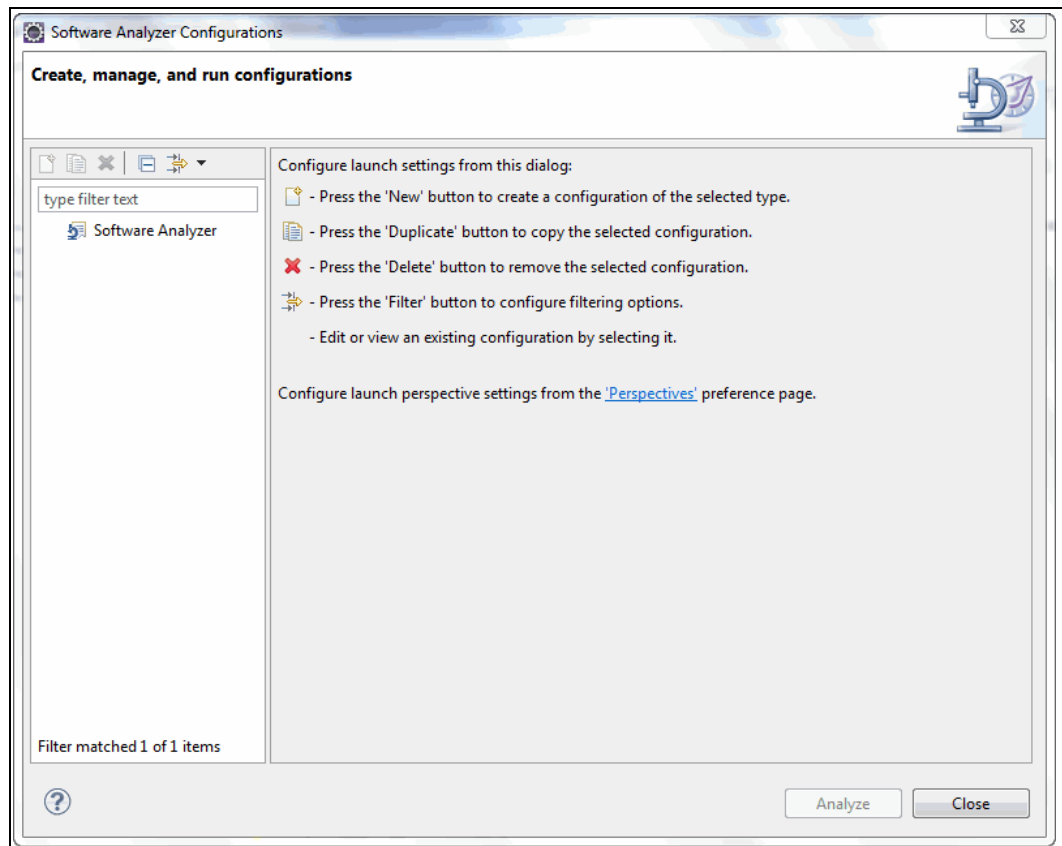


Figure 12-12 Software Analyzer configuration

2. Right-click the **Software Analyzer** and select **New** option to create a New\_configuration under Software Analyzer. You can also highlight Software Analyzer and click the New icon on the toolbar (the first icon with the "+" sign). The right side changes to show the explanation of the basic configuration options available.
3. In the Name field, enter PlantsVersiontoVersion as the name of the configuration.

4. In the Scope tab, leave the **Analyze entire workspace** radio button selected (Figure 12-13).

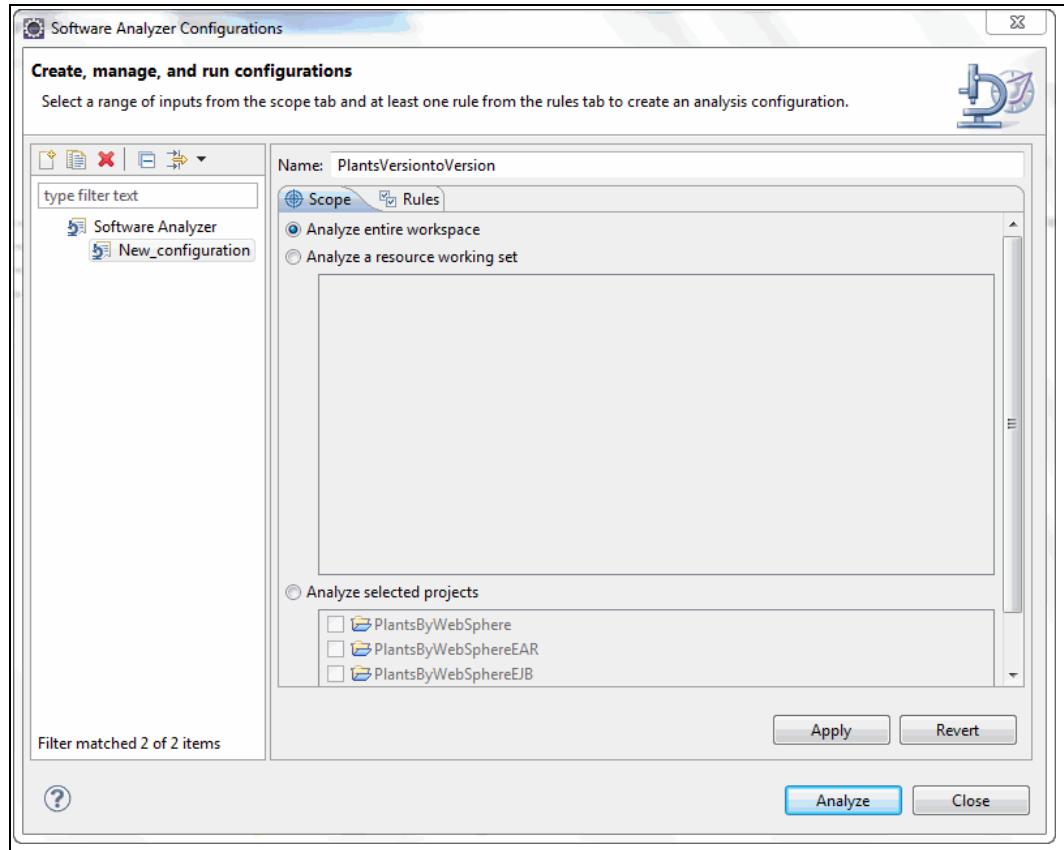


Figure 12-13 Setting up a new Software Analyzer configuration

5. Click the **Rules** tab.



6. Select **WebSphere Application Server Version Migration** from the Rules Sets drop-down menu, and click **Set** (Figure 12-14).

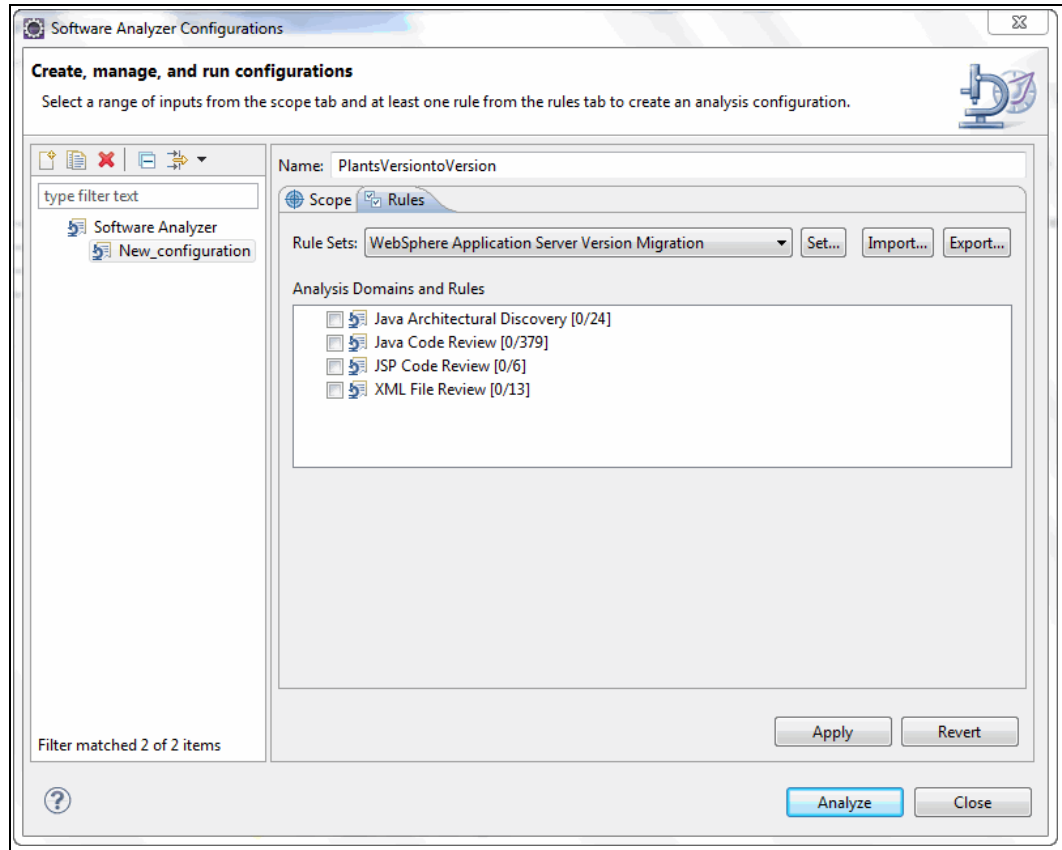


Figure 12-14 Setting rules for the Software Analyzer configuration

7. You see the Rule set configuration dialog box (Figure 12-15). Set the Source application server to **WebSphere Application Server V6.1** and the Target application Server to **WebSphere Application Server V8.5**. **Source Java Version**. The **Target Java Version** category may be selected, as depicted in Figure 12-15. Click **OK**.

**Rules:** The default is that no rules are selected until the user selects the ones that are of interest and clicks **Set**. After the configuration is set, it can be reused.

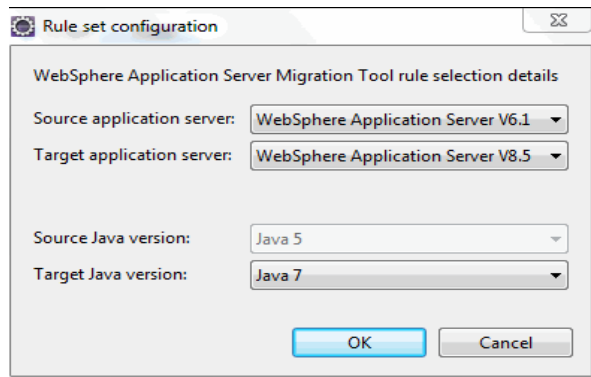


Figure 12-15 Rule set configuration dialog box

8. The Application Migration Tool is now configured and the rules are set. You can run the analysis against the Plants By WebSphere sample. You can expand any of the analysis domains to see which rules are selected (Figure 12-16).

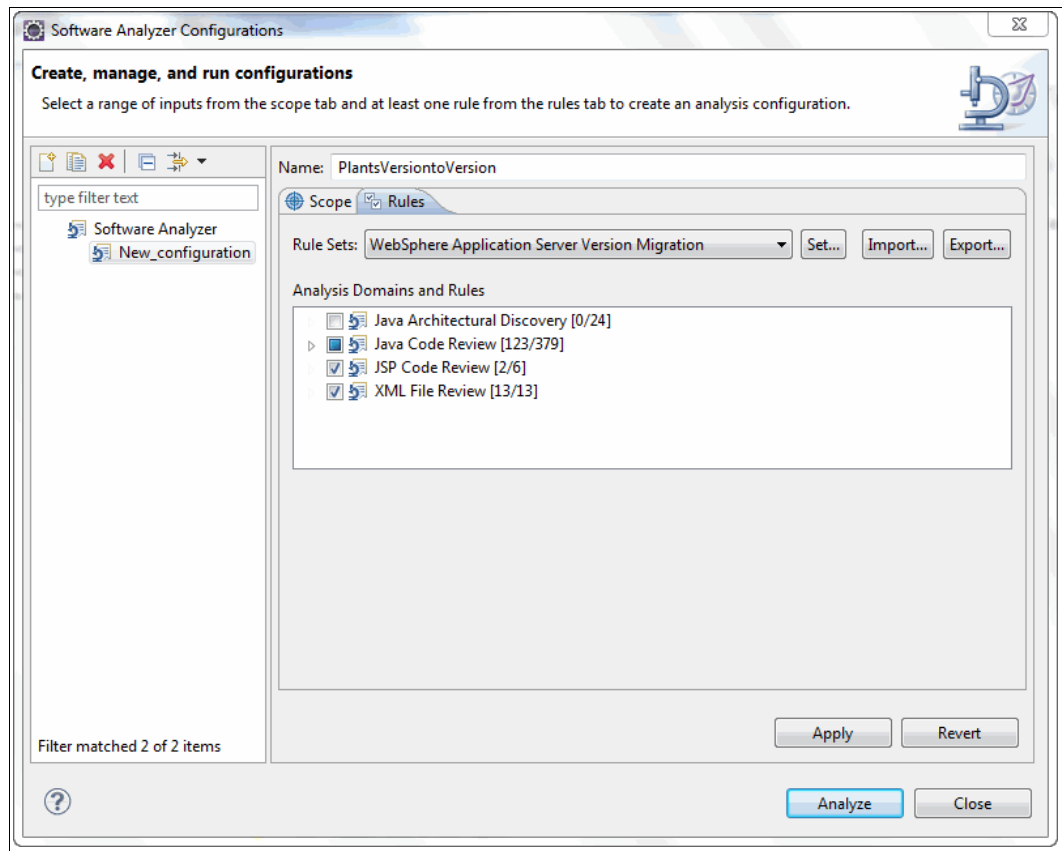


Figure 12-16 A ready Software Analyzer configuration

9. Click **Analyze** to analyze all the projects in the workspace. The Software Analyzer view is opened at the bottom of the window. It contains three tabs. Each tab corresponds to one of the analysis domains selected, based on the Rule Set selection that was made earlier.
- The Java Code Review tab lists Java code-related issues.
  - The XML File Review tab lists issues with deployment descriptors.
  - The JSP Code Review tab lists issues that are found in JSP files, commonly associated with web projects.

The result of the analysis is shown in Figure 12-17.

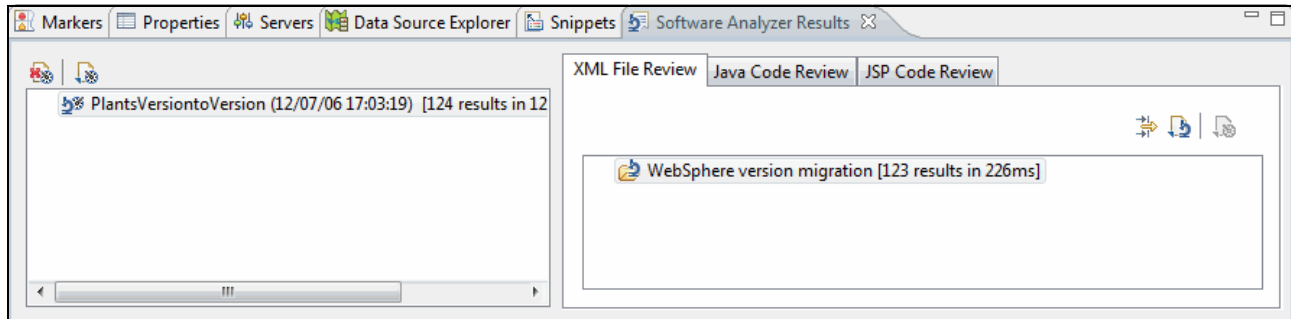


Figure 12-17 Software Analyzer results

10. Click the **Java Code Review** tab.

As you can see, the Application Migration Tool detects a compatibility issue between Java versions (Figure 12-18).

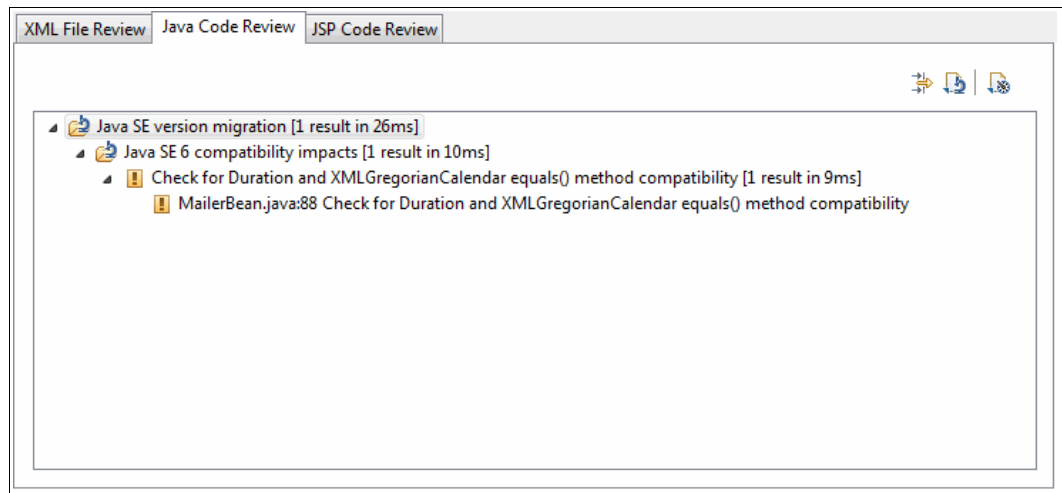


Figure 12-18 Java Code Review

11. Click the **XML File Review** tab.

The tool scanned the XML files in the project and flagged the potential problems with deployment descriptors by category/rule violation. You can expand each of the rules to see the files in question and which project to which they belong (Figure 12-19).

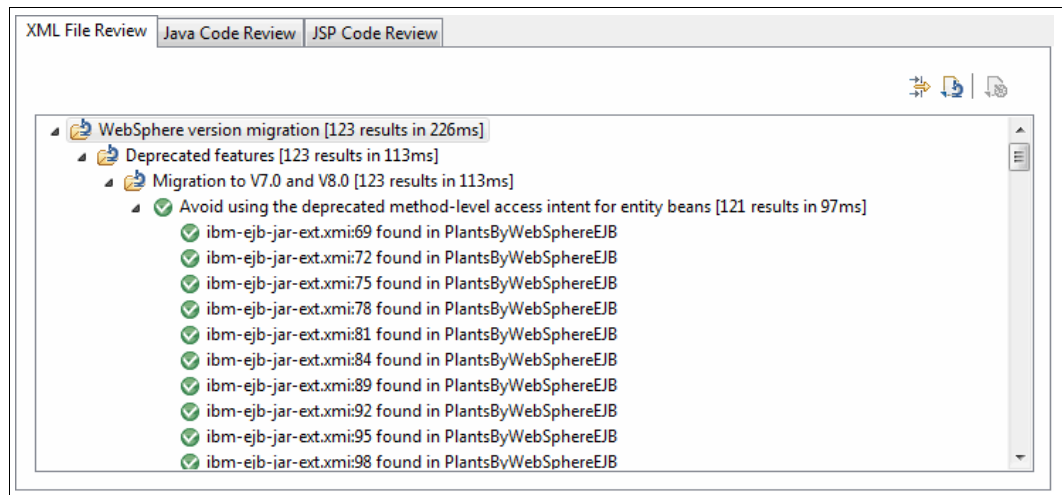


Figure 12-19 XML File Review

You can right-click any of the flagged files under a specific rule to see a list of available options in the menu. For example, under the Java Code Review tab, expand the Java SE Version Migration and right-click any of the files in the list. Then, select **View Results**. The Java file in question is opened and the cursor is at the code problem line.

### 12.3.3 Fixing errors that are reported by Application Migration Tool.

This section describes the procedure to fix the issues reported by the Application Migration Tool.

## Fixing the errors reported under the Java Code Review tab

Expanding the Java Code Review section shows the details of the Java SE 6 compatibility impact results. Right-clicking individual results gives access to options, such as viewing the source code where the problem occurred or correcting the problem with a provided fix.

To fix the errors reported under the Java Code Review tab, complete the following steps:

1. Right-click a result and select **View result**.
2. An editor opens that shows the source file that triggered the rule. The cause of the problem is highlighted, and a rule violation icon is shown in the left margin of the editor. (Figure 12-20).

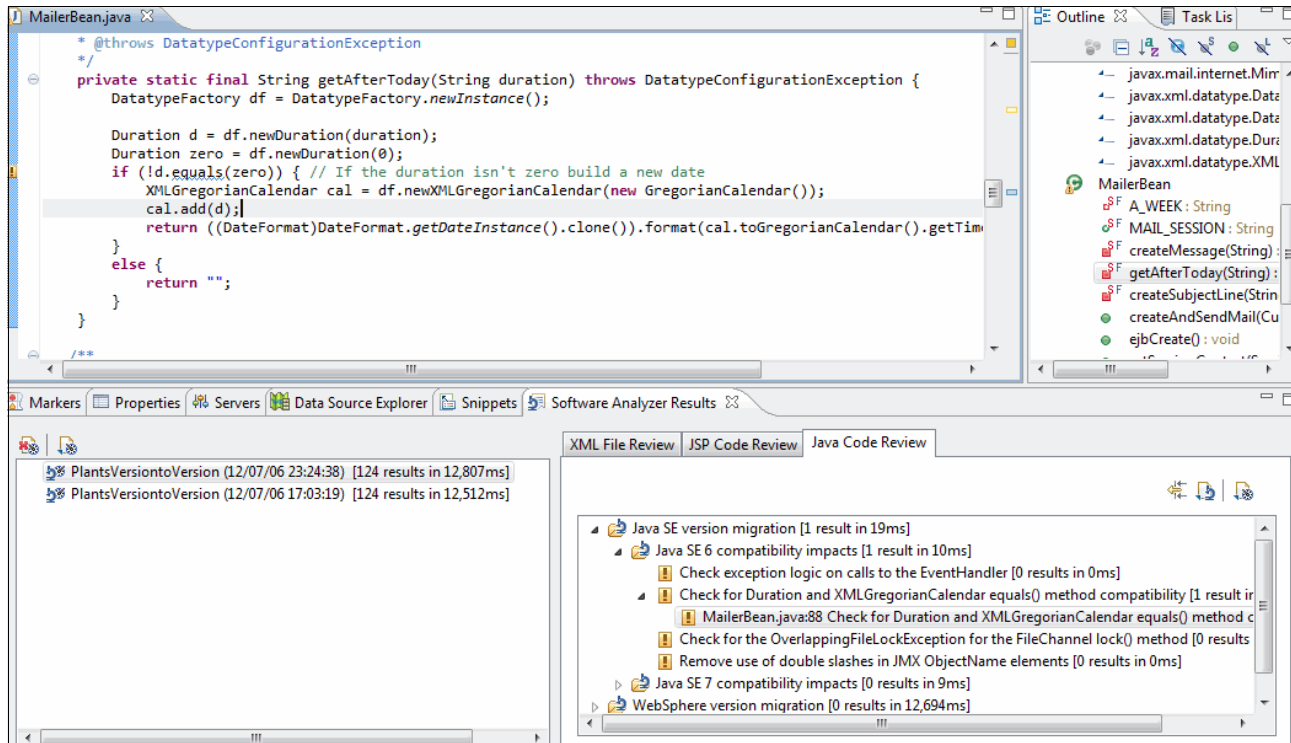


Figure 12-20 View Java Code Review results

The code in question here is shown in Example 12-6.

### Example 12-6 Java code to be analyzed

```
Duration d = df.newDuration(duration);
Duration zero = df.newDuration(0);
if (!d.equals(zero))
```

This rule flags the use of the `equals(Object param)` method on `javax.xml.datatype.Duration` or `javax.xml.datatype.XMLGregorianCalendar`. Java 6 now returns `false` if the parameter passed is `null`. It used to throw a `NullPointerException` in earlier versions of Java. Check the application logic to see if the code must be tested for `false` instead of `NullPointerException`.

In Example 12-7, the `super.equals(someObject)` call is flagged.

*Example 12-7 Usage of the equals method of Duration class leading to a flag*

```
public MyClass extends Duration{
private void doX(){
boolean b = super.equals(someObject);
}
}
```

In Example 12-8, the `c.equals(someObject)` call is flagged.

*Example 12-8 Usage of equals method of Duration class leading to a flag*

```
public MyClass {
private void doX(){
XMLGregorianCalendar c = getCalendar();
boolean b = c.equals(someObject);
}
}
```

Because the code does not require a fix, ignoring the result and rerunning the analysis results in clearing the Java Code Review results (Figure 12-21).

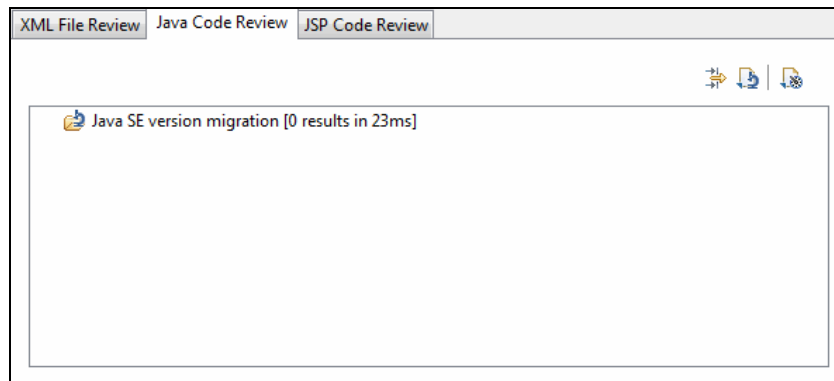


Figure 12-21 Clean Java Code Review results

## Fixing the errors reported under the XML File Review tab

Based on the reported results, you should avoid using the deprecated method level access intent for entity beans. This rule flags the configuration of method level access intent for entity beans. Method level access intent was deprecated because it might run into data access problems, such as a deadlock. To fix reconfigure entity beans to, use bean level access intent, as shown in Figure 12-22.

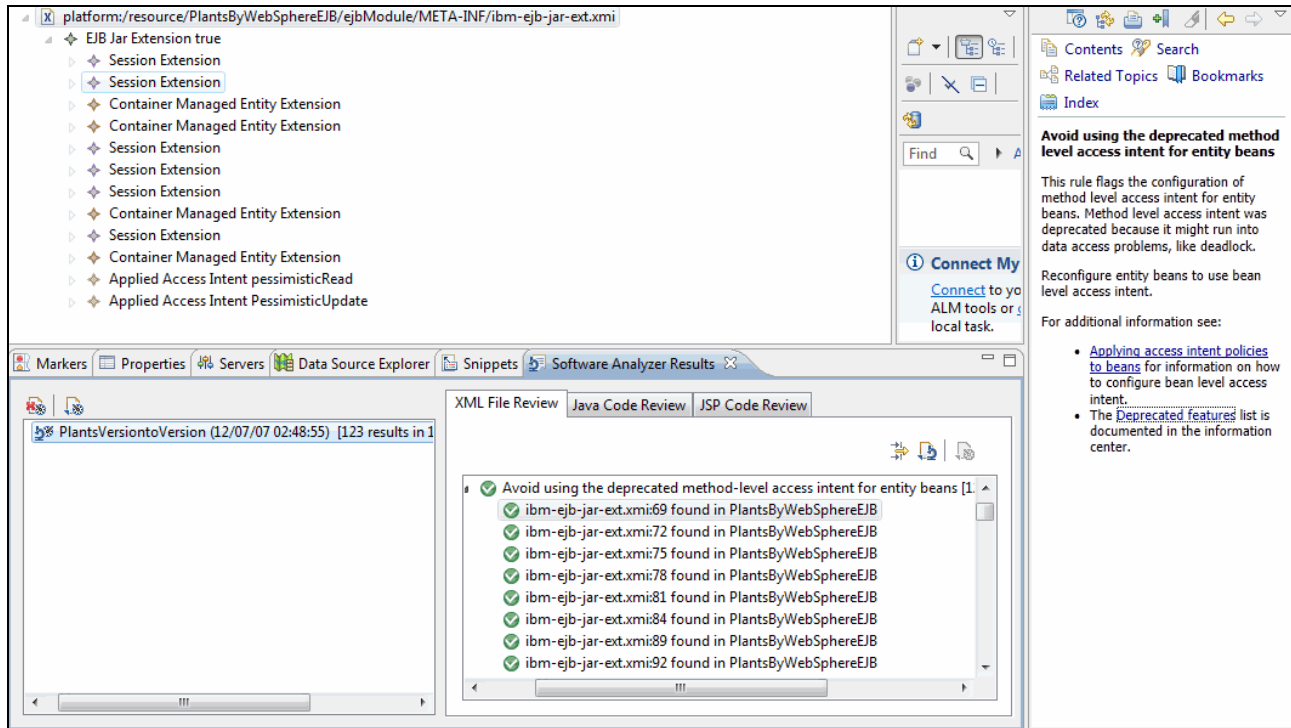


Figure 12-22 Fixing XML file review results

### 12.3.4 Other migration problems not detected by Application Migration Tool

While Application Migration Tool detects many migration problems, there are a few classes of migration problems that are currently not detected by the tools:

- ▶ Migration problems in applications that are originally written for versions of WebSphere Application Server before Version 5.1
- ▶ Migration problems in Java code that is included as snippets inside of JSPs
- ▶ Migration problems in Java code where the migration problem is hidden by the use of Java reflection

If you suspect that an application being migrated has code that falls into any of these categories, you must put in place a plan to manage the risks of undetected migration problems, for example, by putting in place a manual code review plan for the affected application or application parts.

The Plants By WebSphere application actually has such a problem in one of its JSP files. The `error.jsp` file has a section of Java code, as shown in Example 12-9, that uses Java reflection to invoke the `getStackTrace` method on the `ServletErrorReport` class that is provided by the WebSphere Application Server. This method was modified in WebSphere Application Server V5.1 to avoid a conflict that is caused by a change in Java 1.4 to the `Throwable` class. This particular change falls into all three of the categories of potential application migration problems that are not detected by the application migration tools.

*Example 12-9 A snippet from the `error.jsp` file included in the `PlantsByWebSphere`*

```
//Using reflection here so that if the class com.ibm.websphere.servlet.error.ServletErrorReport
//does not exist at compile time there will not be a problem
//if this class does not exist we will just use the attributes specified by Servlet
2.2

Class myClass = Class.forName("com.ibm.websphere.servlet.error.ServletErrorReport");
Method myMethod = myClass.getMethod("getErrorCode", null);
Object o = myMethod.invoke(myReport, null);
status_code = ((Integer) o).intValue();

myMethod = myClass.getMethod("getMessage", null);
o = myMethod.invoke(myReport, null);
message = (java.lang.String) o;

myMethod = myClass.getMethod("getStackTrace", null);
o = myMethod.invoke(myReport, null);
exception_info = (java.lang.String) o;
needInfo = 0;
method = "Using attribute of type com.ibm.websphere.servlet.error.ServletErrorReport
to get information.";
```

The fix for the problem is straightforward: Change the reference to `getStackTrace` to the replacement method `getStackTraceAsString`.

**Other undetectable problems:** Other problems that are not detected by the toolkit are documented at:

<http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg27008724>

### 12.3.5 Exporting the fixed Plants By WebSphere application and running the application on WebSphere Application Server V8.5

This section describes the steps to export the migrated `PlantsByWebSphere`. Complete the following steps:

1. From Eclipse IDE, right-click the **PlantsByWebSphereEAR** project from the Enterprise Explorer perspective.
2. Select **Export**
3. Choose the directory you want to export the `PlantsByWebSphere.ear` file to and save the application.
4. Start the WebSphere Application Server V 8.5 server.
5. Log in to the Administration Console and browse to **Applications** → **Install New Application**.

6. Browse to the location where the rebuilt `PlantsByWebSphere.ear` file is.
7. Provide the Application name, location, and other details.
8. Map the Application Modules to the target application server.
9. Review the summary and save the configuration.
10. Start the application and access `PlansByWebSphere` at:

`http://localhost:9081/PlantsByWebSphere/`

For a detailed step by step procedure for installing the `PlantsByWebSphere.ear` file on WebSphere Application Server and running it, see 12.2.4, “Installing Plants By WebSphere from the administrative console” on page 386 and 12.2.5, “Running the sample application” on page 388.

## 12.4 Probability Distribution Sample

The Probability Distribution Sample is a small JSF application that you can use to do some simple experiments with various random number distributions. The application uses the JSF Widget Library (JWL) that is provided as part of Rational Application Developer. This library is only supported for JSF pages that are built using the JavaServer Pages (JSP) technology. The most recent versions of this library (Version 3.1.6 or higher) are compatible with the JSF 2.0 specification and run on WebSphere Application Server V8.0 and V8. However, when we enhance the application to take advantage of the new JSF 2.0 functionality, we must address the deprecation of this library.

**Downloading sample applications:** For details about how to download the sample applications that are used in this chapter, see Appendix B, “Additional material” on page 435.

### 12.4.1 Importing the application into Rational Application Developer V8.5

Use Rational Application Developer to install the application into both versions (Version 6.1 and Version 8.5) of the WebSphere Application Server. To start the migration process, import the application into the Rational Application Developer workspace by completing the following steps:

1. Click **File** → **Import...**, click **Java EE** → **EAR File**, and click **Next**.
2. Click **Browse** and select the `blackswan2.ear` file.
3. As the file name of the EAR file is the same as the name of the WAR file that is included in the EAR file, Rational Application Developer warns us that there is a name clash in the project names. To resolve this clash, change the name of the EAR project to `blackswan2ear`.
4. Because you are installing the imported EAR file in to WebSphere Application Server Version 6.1, set up that run time now by clicking **New....** On the “New Server Runtime Environment” wizard, select **WebSphere Application Server 6.1** and click **Next**. Click **Browse**, select the installation directory for WebSphere Application Server Version 6.1, and click **Finish**.
5. Back in the Enterprise Application Import wizard, click **Finish**.



## 12.4.2 Installing the application on WebSphere Application Server V6.1

You can now install the application directly on to your WebSphere Application Server Version 6.1 server from Rational Application Developer. You must add the server to Rational Application Developer and then run the application on that server. To do this task (assuming that you already created an application server profile on WebSphere Application Server V6.1), complete the following steps:

1. Right-click in the Servers view in Rational Application Developer and select **New** → **Server**.
2. Select **WebSphere Application Server V6.1** and click **Next**.
3. Select the name of the profile you want to use (we used the default of AppSrv01), provide the admin user ID and password if security is enabled on your server, and then click **Next**.
4. As we want to run the blackswan2 application on this server, select the **blackswan2ear** project, click **Add**, and then click **Finish**.

## 12.4.3 Testing the application on WebSphere Application Server V6.1

After the server is defined in Rational Application Developer and the application is installed, you can test the application. Complete the following steps:

1. Expand the **BlackSwan2** project and right-click `index.html` (found in the WebContent folder). Select **Run As** → **Run On Server**.
2. Check that **WebSphere Application Server V6.1** is selected and then click **Finish**.
3. If the application server is not started, Rational Application Developer starts the server, then the application, and then the `index.html` page of the application. Now you can test the functionality of the application by selecting the distribution and entering a sample size. When you click **Submit**, the relevant distribution is samples and a graph of the samples that are collected is displayed.

## 12.4.4 Running the Application Migration Tool - WebSphere Version to Version

The Application Migration Tool - WebSphere Version to Version can be used to identify potential migration problems. To do that task for the probability distribution sample, complete the following steps:

1. Right-click the **stock\_quote\_example\_axis2** project and select **Software Analyzer** → **Software Analyzer Configurations...**
2. Click **New launch configuration**.
3. Click the **Rules** tab and then use the Rule Sets: drop-down menu to select the **WebSphere Application Server Version Migration** rule set and click **Set....**

4. In the Rule set configuration window, set the Source application server to “**WebSphere Application Server V6.1**” and the Target application server to “**WebSphere Application Server V8.5**” (Figure 12-23) and click **OK**.

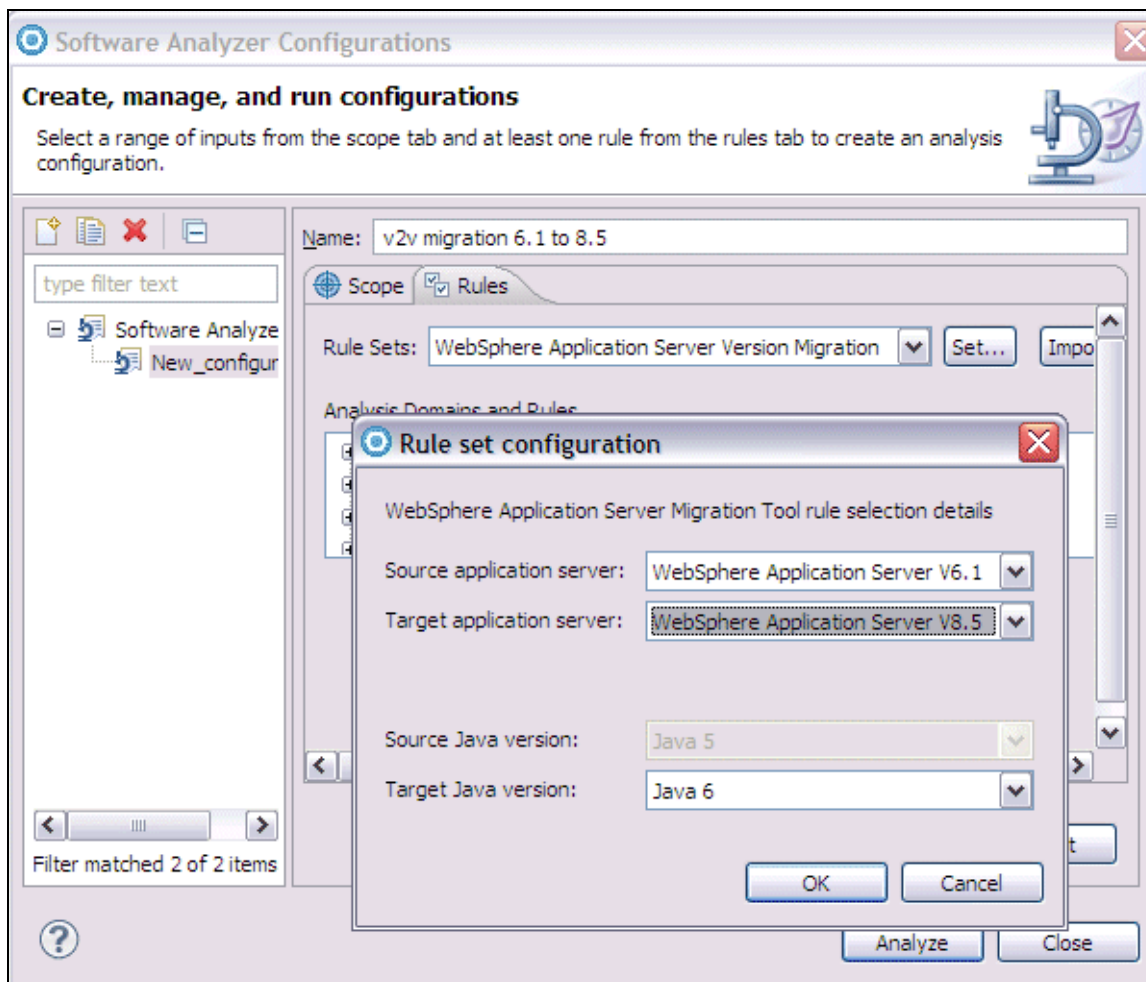


Figure 12-23 Configuring the Application Migration Toolkit for the Probability Distribution Sample application

5. Click **Analyze** to run the analysis.

For this application, three potential problems are reported:

- ▶ Two instances of “Avoid using the deprecated reload attribute of the IBM deployment descriptor extensions.” occur.
- ▶ One instance of “Check for a behavior change in JavaServer Faces (JSF) Applications.” occurs.

### 12.4.5 Resolving the reported migration problems

Resolving the reported migration problems is an important step in the application migration process. Although fixing the problems does not ensure that the application works after migration, any problems that are identified by the Application Migration Tool are problems that caused applications to stop working in later versions of the application server.

## Resolving the deprecated reload attributes

Fixing the two instances of the “Avoid using the deprecated reload attribute of the IBM deployment descriptor extensions” error is straightforward. Double-clicking each entry takes you to the XML attribute that is detected as a potential problem. After you examine the attributes “reloadInterval” and “reloadingEnabled” in our example, we determine that these attributes are not needed by the application and can be removed (the attributes were initially added by Rational Application Developer when the application was first created).

## Checking for a behavior change in JSF applications

The remaining problem is the “Check for a behavior change in JavaServer Faces (JSF) Applications” result. This problem is telling you to review the application’s use of JSF to confirm that it can run on a JSF 2.0 run time. This error requires that you review the JSF usage, which includes checking any libraries that are being used by the application to confirm whether they can run on JSF 2.0 or not.

In the case of our application, it is using the JWL provided by Rational Application Developer. After we review the material at the JavaServer Faces Migration in the WebSphere Application Server v8.5 Information Center at

[http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=cweb\\_jsfmigrate](http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=cweb_jsfmigrate), we must check whether the version of the JWL that the application is using is supported. The JWL is provided as a set of JAR files that are placed in the WEB-INF/lib directory of the web content. The version number of JWL is available inside the manifest of the jsf-ibm.jar file. To view this version number, complete the following steps:

1. Right-click the JAR file and click **Export...**
2. Expand **General**, select **File System**, and then click **Next**.
3. Enter a temporary directory in the “To directory:” entry field and click **Finish**.
4. On a command line, with that temporary directory as the current directory, enter the following command:

```
jar -xvf jsf-ibm.jar META-INF/MANIFEST.MF
```

We assume that you have a JDK on the path. If that is not the case, then you can use the jar.exe from the JDK included in your WebSphere Application Server installation).

5. After the file is extracted from the JAR file, you can examine the manifest.mf file using an editor. The version number is specified on the line that starts with Implementation-Version:. The contents of the manifest.mf file from the Probability Distribution Sample is shown in Example 12-10.

*Example 12-10 The manifest.mf file for the jsf-ibm.jar in the Probability Distribution Sample*

---

```
Manifest-Version: 1.0
Ant-Version: Apache Ant 1.8.2
Created-By: IBM Corporation
Specification-Title: JavaServer Faces
Implementation-Title: JSF Widget Library (JWL) - Base
Implementation-Version: JWL v3_1_13
Implementation-Vendor: IBM
Build-Version: 20120425.1128
Build-Date: April 25 2012
Copyright-Info: Copyright (c) 2003,2008, International Business Machines
Corporation. All Rights Reserved.
```

---

The version of the JWL used in the Probability Distribution Sample is Version 3.1.13, which supports running on a JSF 2.0 run time. If the version is earlier than Version 3.1.6, you must replace the version of the JWL used in the application with the latest version of JWL provided with Rational Application Developer V8.5.

As you verified that the application work with JSF 2.0, you can return to the Software Analyzer results page and right-click the **Check for a behavior change in JavaServer Faces (JSF) Applications** problem and click **Ignore Result** to remove this entry from the Software Analyzer results.

## 12.4.6 Preparing the application in Rational Application Developer for installation on WebSphere Application Server V8.5

Rational Application Developer supports deployment of the same application to multiple run times. However, to support this deployment, the correct set of project facets are required on the projects that make up the application. Our application is targeted for WebSphere Application Server Version 6.1 only, and we must modify the set of project facets and targeted run times to support deployment on Version 8.5. To do this task, complete the following steps:

1. Right-click the **BlackSwan2** project and select **Preferences**.
2. Select the **Project Facets** view in the list of properties.
3. Clear the **WebSphere Web (Extended)** facet and click **Apply**. This facet adds support in the deployment descriptor editor for the WebSphere extensions to the deployment descriptor.
4. Select the **Targeted Runtimes** view in the list of properties.
5. If “WebSphere Application Server v8.5” is not shown on the list of run times, click **New...** and complete the following steps (we assume that you already have an application server profile that is defined on your WebSphere Application Server V8.5 installation):
  - a. Select **WebSphere Application Server v8.5** in the list on run times on the New Server Runtime Environment wizard and click **Next**.
  - b. Click **Browse...** and select the root directory of your WebSphere Application Server V8.5 installation and then click **Finish**.
6. Select the **WebSphere Application Server V8.5** check box in the list of Targeted Runtimes and then click **OK**.

These steps must be repeated for the blackswan2ear project.

## 12.4.7 Installing the application on WebSphere Application Server V8.5

As with the installation of WebSphere Application Server V6.1, you define a Version 8.5 server and add the application to it. To do so, complete the following steps:

1. Right-click in the Servers view in Rational Application Developer and select **New** → **Server**.
2. Select **WebSphere Application Server V8.5** and click **Next**.
3. Select the name of the profile you want to use (we used the default of AppSrv01) and provide the admin user ID and password if security is enabled on your server and then click **Next**.
4. As we want to run the blackswan2 application on this server, select the **blackswan2ear** project, click **Add**, and then **Finish**.

## 12.4.8 Testing the application on WebSphere Application Server V8.5

After the server is defined in Rational Application Developer and the application is installed, you can test the application by completing the following steps:

1. Expand the **BlackSwan2** project, right-click the `index.html` file (found in the `WebContent` folder), and select **Run As** → **Run On Server**.
2. Check that WebSphere Application Server V8.5 is selected and click **Finish**.
3. If the application server is not started, Rational Application Developer starts the server, then the application, and then the `index.html` page of the application. From here, the functionality of the application can be tested by selecting the distribution and entering a sample size. When Submit is clicked, the relevant distribution is `samples` and a graph of the samples that are collected is displayed.

## 12.4.9 Options for upgrading the application after migration

After the Probability Distribution Sample is migrated, you have many options to modify it.

### Leaving the application unchanged

One option is to either leave the application unchanged or, at least, to limit the changes to the application to those changes that do not require the removal of the JWL library.

This option is a viable one for applications that do not need major enhancement or modification for the foreseeable future.

### Restricting application changes to the changes supported by JWL

The JSF Widget Library is supported on JSF 2.0 if the restrictions for its use are respected. The main restriction is that the application cannot use the Facelets web template system. In general, this restriction is not too onerous for this application. You typically use Facelets in a new JSF 2.0 application, but you would not typically mix Facelets with non-Facelets JSF code in the same application. So, you can maintain and enhance the application.

### Removing the dependency on the JSF Widget Library

JWL is deprecated and does not support all of the functionality of JSF 2.0 (in particular, any JSF 2.0 application that uses Facelets cannot also use the JSF Widget Library). If you intend to update the application to use features that are added in JSF 2.0, you must remove the JSF Widget Library and replace it with a different widget library that provides similar functionality. In our case, we use the JSF Widget Library to draw a bar chart of the number of occurrences of each value that was found in the set of samples that are drawn from the probability distribution that is sampled.

At the time of the writing of this book, there is no single alternative to the JSF Widget Library, but several third-party JSF widget libraries include charting components. We would need to evaluate these alternatives and replace our bar chart with a chart from one of those alternatives or write our own JSF component to perform the charting capability.

## 12.5 Web services Axis 2 stock quote

The third application that we migrate is a web services application that is based on the Axis 2 web services engine. Web services is a technology that allows applications to be invoked using internet standards and protocols. The applications are divided into multiple tiers, typically at least two: client and server. These two pieces communicate using XML messages that are defined by the SOAP 1.1 or 1.2 protocols.

WebSphere Application Server V6.1 provides support for the JAX-RPC and JAX-WS protocols and provides more features, such as a UDDI registry and extensions to the SOAP protocol known as WSIF.

The WebSphere Application Server also supports several more protocols that build on top of the basic web services protocol, such as WS-Notification and WS-ReliableMessaging.

All of these functions are provided by the web services engine that is provided by the application server. However, in some circumstances you must use a third-party web services engine, such as Axis 2. For example, if you want to deploy applications that use a single run time across various application servers, such as WebSphere Application Server, JBoss, and WebLogic, you would use a third-party engine.

IBM supports the enablement of third-party JAX-WS run times to run on WebSphere Application Server and ensures the successful deployment of applications that use such run times. However, IBM does not provide support for resolving JAR file conflict problems or any problem that a stack trace indicates is in the third-party code. In addition, using a third-party web services engine prevents the application server from recognizing that web services provided by the third-party engine are, in fact, web services, and therefore prevents the use of application level policy sets, such as WS-Security, WS-RM, and WS-Transactions policy sets, and also prevents the use of WSDM or the use of JNDI lookups to retrieve JAX-WS Services or Port Instances

### 12.5.1 Overview

This section shows an example of an application that uses a third-party web services engine, specifically the Axis 2 web services engine. The application is simple (in fact, it provides just two web services), but demonstrates the potential problems of migrating such an application.

The application provides a stock quote service that a client can use to discover stock prices by providing the symbol of the required stock price as input. You use the Axis Object Model (AXIOM) to allow arbitrary pieces of XML to be passed and returned in the body of the web services messages. In this example, a simple Plain Old Java Object (POJO) could be used as the implementation of the web service together with a suitable definition of the web service. The developer of this particular application chose to use the AXIOM style for the web service.

### 12.5.2 Installation on WebSphere Application Server V6.1

The StockQuoteAxis2Example application is provided as a WAR file that includes the Axis 2 run time and the StockQuote web service. The web service is provided as a WAR file within the services directory of the WAR file. In our example, we install the application through the application console of WebSphere Application Server V6.1 by completing the following steps:

1. Click **Install New Application** in the list of tasks in the left pane.
2. Click **Browse...** and select the WAR file.

3. Enter “stockquote” in the context root field (Figure 12-24).

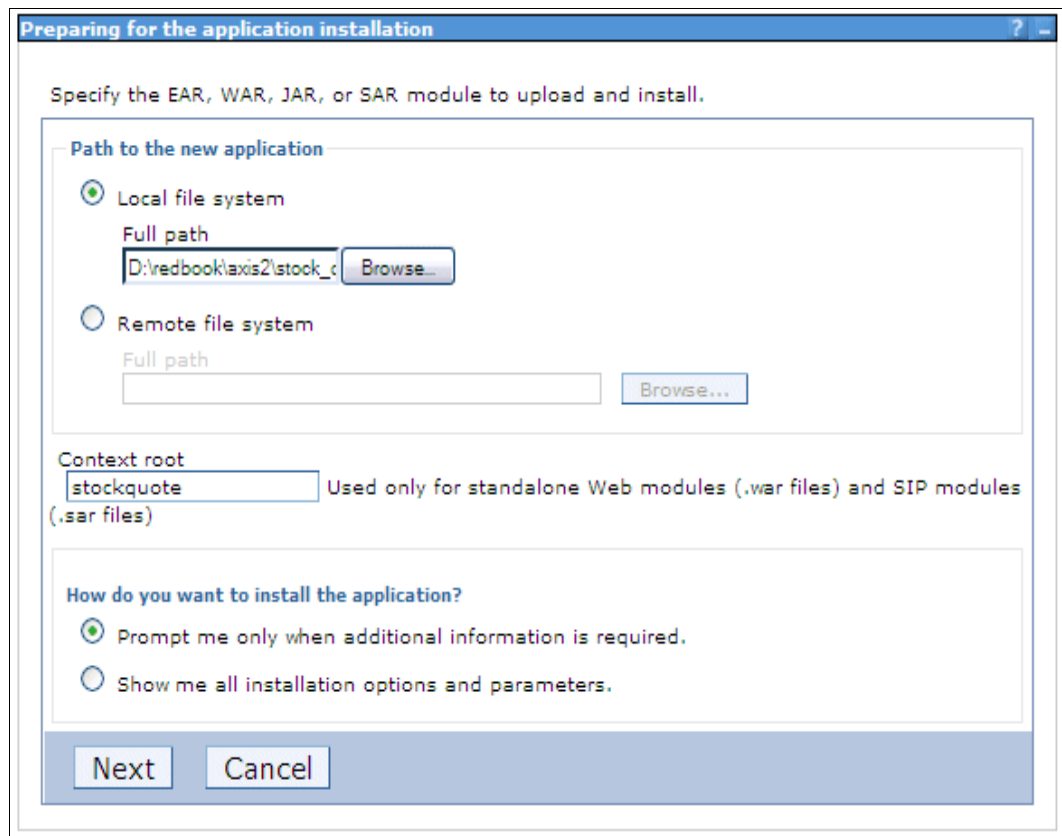


Figure 12-24 Installing the Axis 2 application on WebSphere Application Server V6.1

4. Click **Next**.
5. On “Step 1. Select installation options”, click **Next**.
6. On “Step 2. Map modules to servers”, click **Next**.
7. On “Step 3. Map virtual hosts for Web modules”, click **Next**.
8. On “Step 4. Map context roots for Web Modules”, click **Next**.
9. On “Step 5. Summary”, click **Finish**.
10. After the application installation process is complete, click **Save** to save the server configuration changes to the master configuration.

Now that the application is installed, it must be started so that it can be tested. To do so, complete the following steps:

11. Click **Enterprise Applications** in the list of tasks in the left pane.
12. Select the entry for the “stock\_quote\_example\_axis2\_war” and click **Start**.

### 12.5.3 Testing on WebSphere Application Server V6.1

To test that the web services are operational, invoke the test client program that is provided by the application. As the client program needs the axis2 JAR files on the classpath, download the axis2 binary distribution from <http://axis.apache.org/axis2/java/core/download.cgi> and extract the downloaded compressed file. We then imported the client test program into Eclipse and updated the projects classpath with the JAR files in the `lib` directory of the downloaded axis2 distribution.

For applications with well-defined web services, the generic service client or the web services explorer that is provided in the WebSphere Developer Tools (and in Rational Application Developer) could be used to directly interact with the web service. The stock quote application uses AXIOM and relies on the ability of the axis2 web services engine to generate a definition of the web service that is based on the signature of the implementation. However, this action generates a definition that is not suitable for use by either of these test clients, so you cannot use these test clients. Fortunately, the application provides a set of tests using JUnit that you can use to test the function of the web service.

The first stage of the testing is to validate that the axis2 web services engine used by the application is installed correctly. The engine is provided as a web application and the developers of the engine provide a set of web pages to validate and examine the status of the engine. We can use these pages to check that the basic installation of the engine is successful.

To do this validation, use a web browser to navigate to the home page of the web services engine. On our test system, that page is at `http://localhost:9082/stockquote` (the port number depends on `WC_defaultHost` port that is defined on the profile of the application). This action displays the home page of the axis2 web service engine. We clicked the **Validate** link to run the validation checks provided by the axis2 engine. Finally, we navigate to the “Services” page to verify that our StockQuote service is available.

The next stage is to test the actual web service. This action requires actual web services requests to be generated and submitted to the web service. In our case, we use an existing set of JUnit tests to verify the functionality of the application. To do so, complete the following steps:

1. Start Eclipse and click **File** → **Import...**
2. Select **Existing Projects into Workspace** and then **Next**.
3. Select **Archive** and click **Browse...**
4. Locate the compressed file that contains the StockQuoteClient UnitTest and click **Finish**.
5. After the StockQuoteClientSideUnitTest project is created, right-click the project and select **Build Path** → **Configure Build Path**.
6. Click the **Libraries** tab and then click **Add External Jars**. Select all of the JAR files in the `lib` directory of the axis2 binary distribution. Click **OK**.
7. The StockQuoteClientSideUnitTest might need to be updated to give the correct endpoint for the web service. To do this task, complete the following steps:
  - a. Expand the **StockQuoteClient** project and then the **src** folder. After you expand the stock project, double-click the `StockQuoteClient.java` file.
  - b. Update the `STOCK_SERVICE_EPR` constant to have the URL of the endpoint. Most likely, you must update the port number to match the port number of your WebSphere Application Server V6.1 server.



In our example, we use

`http://localhost:9082/stockquote/services/StockQuoteService` based on the port number of our server and the context root that is provided when the application was installed on the application server.

8. After the `StoickQuoteClient` is compiled with no errors by Eclipse, you can run the test program. To do so, complete the following steps:
  - a. Expand the **StockQuoteClient** project and then the **src** folder. After you expand the stock project, right-click the `StockQuoteClient` class.
  - b. Click **Run As** → **JUnit Test**.

The JUnit tests all complete with a green bar showing that the web service successfully passed the set of tests.

### 12.5.4 Importing the source of the application into Eclipse

The Stock Quote is provided as a WAR file that includes the axis2 run time. To import this file into Eclipse, complete the following steps:

1. Click **File** → **Import...**, click **Web** → **WAR file**, and then click **Next**.
2. Click **Browse** and locate the WAR file.
3. Change the target run time to “WebSphere Application Server v8.5”. If you have not created a target run time for a server that is running on WebSphere Application Server V8.5, you can create one. To do so, complete the following steps:
  - a. Click **New....**
  - a. Open the IBM folder, select **WebSphere Application Server V8.5**, and then click **Next**.
  - b. Click **Browse...** and locate the root directory of your WebSphere Application Server V8.5 installation. Click **Finish** to return to the WAR Import wizard.
4. Clear the **Add project to an EAR** check box if that option is selected.
5. Click **Finish**.

### 12.5.5 Running the Application Migration Tool - WebSphere Version to Version

As with all application migration in our example, we run the Application Migration Tool to identify potential migration problems with the code of the application. To do this task, we create and run a new Software Analyzer configuration. (We assume that the tool is installed.)

Complete the following steps:

1. Right-click the `stock_quote_example_axis2` project and select **Software Analyzer** → **Software Analyzer Configurations....**
2. Click **New launch**.
3. Click the **Rules** tab and then use the Rule Sets: drop-down menu to select the WebSphere Application Server Version Migration rule set. Click **Set....**

4. In the Rule set configuration window, set the Source application server to “WebSphere Application Server V6.1” and the Target application server to “WebSphere Application Server V8.5”, as shown in Figure 12-25. Click **OK**.

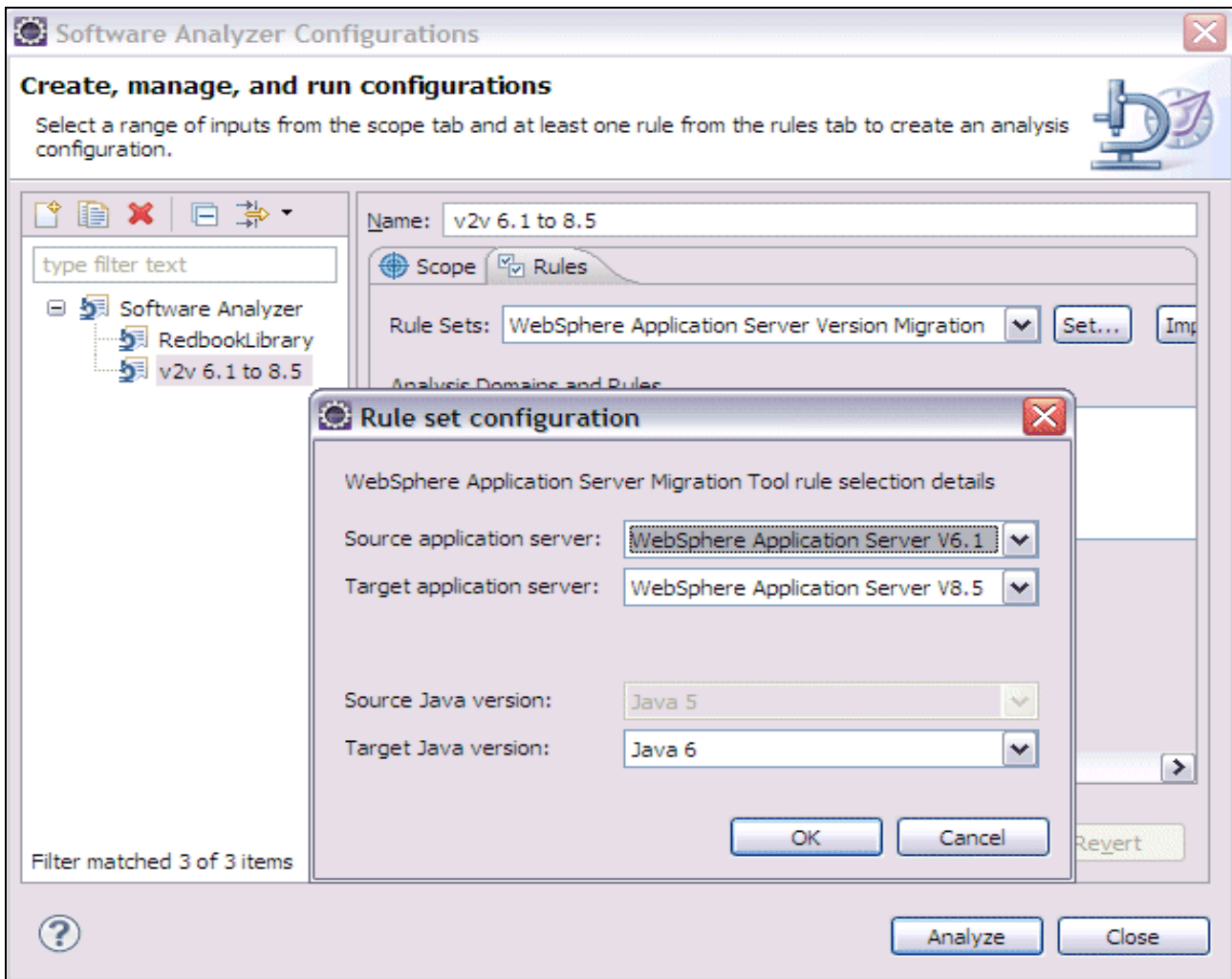


Figure 12-25 Configuring the Application Migration Toolkit for the Stock Quote Axis2 application

5. Click **Analyze** to analyze the workspace for potential migration problems.

The analysis reports that no problems are detected. However, Version 3.5 of the Application Migration Toolkit does not check for third-party web services engines, such as axis2, and does not report that some small changes are needed to the configuration of that engine to successfully run the application in WebSphere Application Server V8.5.

## 12.5.6 Preparing to install on WebSphere Application Server V8.5

To install an application that uses the axis2 web services engine on WebSphere Application V8.5, some changes are needed to that application.

## Updating the axis2.xml file to specify EnableChildFirstClassLoading

Starting with Version 1.5.5 of the axis2 web services engine, a new parameter is added to the axis2 configuration. This parameter controls how class loaders created by the axis2 web service engine behave. In order for the axis2 web service engine to operate correctly in WebSphere Application Server V.5, this parameter must be changed from the default of false to true. To do this task, expand the stock\_quote\_example\_axis2 project and locate the axis2.xml file in the WEB-INF/conf folder of WebContent. Update the EnableChildFirstClassLoading property to true, as shown in Example 12-11.

*Example 12-11 The EnableChildFirstClassLoading property in the axis2.xml file*

---

```
<parametername="EnableChildFirstClassLoading">false</parameter>
```

---

## Disabling web services annotation scanning for the application

By default, WebSphere Application Server performs annotation scanning for all installed applications to determine what web services are provided by those applications. When you use a third-party web services engine, this scanning is unnecessary and can cause problems. To disable the annotation scanning for this application, update the MANIFEST.MF file for the application's WAR file to add the DisableIBMJAXWSEngine property, as shown in Example 12-12.

*Example 12-12 The META-INF/MANIFEST.MF file in the WebContent of stock\_quote\_example\_axis2 project*

---

```
Manifest-Version: 1.0
DisableIBMJAXWSEngine: true
Archiver-Version: Plexus Archiver
Created-By: Apache Maven
Built-By: sagara
Build-Jdk: 1.6.0_22
```

---

**Disabling annotation scanning:** As an alternative to modifying the META-INF/MANIFEST.MF file, you can turn off annotation scanning for the entire WebSphere Application Server instance by setting a JVM property through the administration console. To do this task, click **Application Server** → **Server** → **Process Definition** → **Java Virtual Machine** and then add the following line to the Generic JVM arguments field:

```
-Dcom.ibm.websphere.webservices
```

## Exporting the WAR file that is ready for installation on Version 8.5

As the application is packaged as a simple WAR file, it cannot be directly installed on the application server from Eclipse. It must be exported as a WAR file. To do this task, complete the following steps:

1. Right-click the **stock\_quote\_example\_axis2** project and click **Export...**
2. Provide a file name for the exported WAR file. Do not specify the same name and location as the original WAR file that is installed onto WebSphere Application Server Version 6.1 so that you still have the original WAR file if you need to reimport the application into Eclipse. In our example, we call the file stock\_quote\_example\_axis2\_85.war.
3. Click **Finish**.

## 12.5.7 Installing on WebSphere Application Server V8.5

In our example, we install the application onto WebSphere Application Server V8.5 using its administration console:

1. Click **New Application** in the list of tasks in the left pane.
2. Click the **New Enterprise Application** link.
3. Click **Browse...** and select the WAR file.
4. Click **Next**.
5. Click **Next** on the “How do you want to install the application?” window.
6. On “Step 1. Select installation options”, click **Next**.
7. On “Step 2. Map modules to servers”, click **Next**.
8. On “Step 3. Map virtual hosts for Web modules”, click **Next**.
9. On “Step 4. Map context roots for Web Modules”, enter /stockquote for the context root (this entry matches the context root that we specified when installing on V6.1) and then click **Next**.
10. On “Step 5. Summary”, click **Finish**.
11. After the application installation process is complete, click **Save** to save the server configuration changes to the master configuration.

## 12.5.8 Configuring WebSphere Application Server V8.5 to support the application

To resolve conflicts between the axis2 web services engine and the code that is used internally by the WebSphere Application Server V8.5 run time, we must configure the application using the axis2 web service engine to parent last class loading strategy. For more information, see 3.1.2, “Classloader related problems” on page 58.

To update the configuration of the application, complete the following steps:

1. Click **WebSphere Enterprise Applications** in the list of tasks in the left pane (you might need to expand **Application Types** to find this task).
2. Click the **stock\_quote\_example\_axis2\_85\_war**, then **Manage Modules**, and **Apache-Axis2**.

- Use the Class loader order drop-down menu to select **Classes loaded with local class loader first (parent last)**, as shown in Figure 12-26, and then click **OK**.

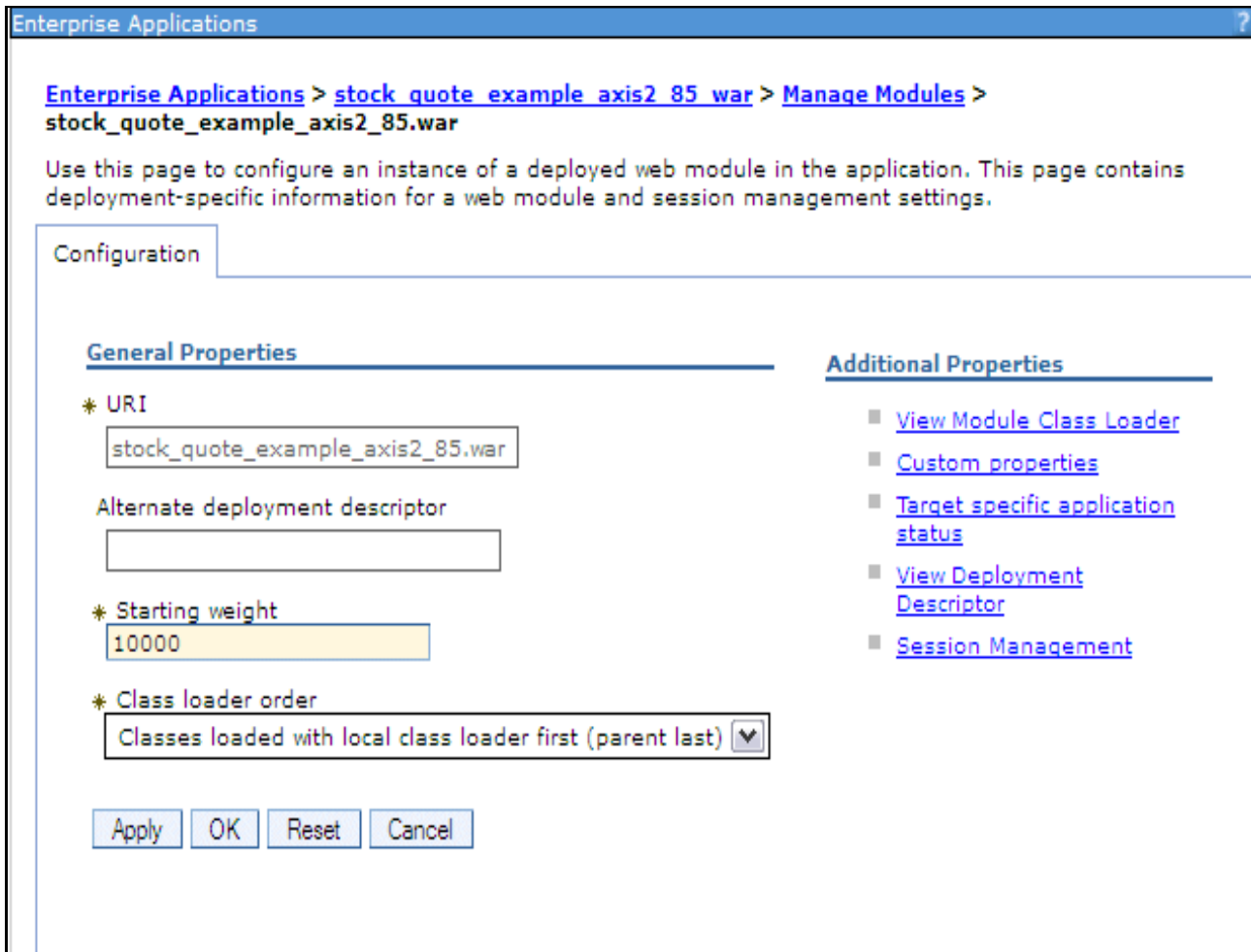


Figure 12-26 Selecting the “parent last” option for class loader order

## 12.5.9 Testing the application on WebSphere Application Server V8.5

Now that the application is installed and configured, it must be started so that it can be tested. To do this task, complete the following steps:

- Click **WebSphere Enterprise Applications** in the list of tasks in the left pane (you might need to expand **Application Types** to find this task).
- Select the entry for `stock_quote_example_axis2_85_war` and click **Start**.

After the application is started, you can reuse the `StockQuoteClientSideUnitTest` to test the application that is running on WebSphere Application Server V8.5. As with testing the application on Version 6.1, the test case must be updated to provide the URL of the HTTP endpoint of the web service. To do this task, complete the following steps:

- Expand the **StockQuoteClient** project and then the **src** folder. After you expand the project, double-click the `StockQuoteClient.java` file.
- The `STOCK_SERVICE_EPR` constant must be updated to give the URL of the endpoint. Most likely, you must update the port number to match the one of your WebSphere Application Server Version 8.5 server.

In our example, we use `http://localhost:9083/stockquote/services/StockQuoteService`, based on the port number of our Version 8.5 server and the context root that is provided when the application was installed on that application server.

The JUnit tests all complete with a green bar showing that the web service successfully passed the set of tests included.

### **12.5.10 Summary of migrating the Web services Axis 2 stock quote application**

Normally, migrating applications between versions of the Application Server is straightforward, but some applications, such as the ones using a third-party web services engine, can be more complicated. This section has shown that, with some care, migration of such applications is still easy to achieve.



# Part 5

# Appendixes







# A

## Migration questionnaires

This appendix contains sets of questions that you can use to build your own questionnaire that is based on the unique challenges you foresee. It is not intended to be a final questionnaire, but a base from which you can learn how migration questionnaires are built. Other specific questionnaires might be required (such as SOA and Web 2.0 questionnaires). For more information, see 2.5.2, “Getting help” on page 53.

This appendix describes the following topics:

- ▶ Business requirements
- ▶ General information
- ▶ Application architecture
- ▶ Code
- ▶ Development migration questionnaire
- ▶ Runtime migration questionnaire
- ▶ Testing migration questionnaire

## Business requirements

Business requirements questions are as follows:

1. What is motivating the business to make this change?
2. Is the application business critical?
3. What is the target version of WebSphere Application Server?
4. What other WebSphere products are involved in this migration?  
WebSphere Portal Server and WebSphere Edge Server possible are examples.
5. By what date must this application migration be complete?
6. What is the risk to the business if the change cannot be implemented by that date?
7. What is the risk to the business when the application is not running?
8. Is this migration part of a new application release/version?
9. Describe any important milestones for the migration. Include dates for development, QA, and so on.
10. How do you describe a successful migration?

## General information

General information questions are as follows:

1. What is the name of the application?
2. What URL is used to access the application?
3. What is the current version of the application?
4. What is the audience for the application: internal, external, or both?
5. What is the primary function of the application?
6. What services does this application provide?
7. Who are the users of the application?
8. Describe any upgrades that are considered part of this migration?  
These upgrades might include such things as platform infrastructure (OS or related stack components) or exploitation of features that are provided by Java EE.
9. Can the migration be partitioned?
10. Can the application be divided into smaller pieces that can individually be migrated?  
Staged migration of the database, JSP, EJB, and so on, enables incremental migration, providing greater opportunity for quality control and isolation of unexpected problems.
11. How frequently do you schedule an independent review of your application code?

## Application architecture

General application architecture questions are as follows:

1. Does the application architecture conform to the Model 2 architecture that is prescribed by Java EE?
2. Describe the application's use of established design patterns.

3. Does the application, for example, employ well-known patterns such as Facade, Command, Strategy, or Proxy?

## Dependencies

Dependency questions are as follows:

1. Are there dependency cycles between packages?

A dependency cycle results when two or more types are involved in a cycle of references that binds them tightly to each other. Dependency cycles are not always bad. Cycles are often wanted between two or more types. Dependency cycles become a problem, however, when they exist between packages or modules. The danger is that they introduce a tight coupling between those packages or modules, which makes the code generally more unreliable and difficult to change. The point of this question is to understand if any such cycles exist between packages.

2. Are there dependency cycles between Java EE modules?

A common dependency cycle between Java EE modules involves an EJB module that makes reference to helper classes in a Java utility module that, in turn, makes references back to the EJB module. Such helper classes are typically used as convenience methods for finding EJB homes, for example.

3. Describe any classloading dependencies in the application.

Class loading issues and dependencies to libraries are common in older Java EE-style applications. The earlier versions of the specifications did not state clearly how the class loading must happen and how the library classes must be accessed.

## Persistence

There is one question that is related to persistence:

1. How is persistence of application data implemented?

This question is concerned with understanding how application data is made persistent. Typically, application data is stored in a database and a layer in the application is responsible for moving that data from the database into Java objects and back again. Persistence might be implemented using EJB CMP entity beans, a third-party product such as TopLink for Java, or a custom implementation of a Data Access Object (DAO) pattern. It might also be implemented in an *ad hoc* fashion using JDBC statements throughout the application code.

## National language

National language questions are as follows:

1. What national languages are supported by the application?
2. How is language determined by the application?

The application might use the language attribute from the HTTP Request Header, or perhaps profile information that is provided by a user using a combination of cookies and HTTP session data.

3. How is language data stored by the application?

## Code

This section has questions that are related to code. Approximate answers suffice for these questions. However, accurate answers yield more accurate results

## Java

Java questions are as follows:

1. How many custom classes are defined in the application?

A custom class is one that was created by your development team. Do not include classes that are part of the standard Java libraries, middleware, or third-party supplied libraries. Also, do not include classes that are generated by the container to support your application in run time.

2. How many custom classes in the application are declared as abstract?

3. How many custom interfaces are in the application?

A custom interface is one that was created by your development team. Do not include interfaces that are part of the standard Java libraries, middleware, or third-party supplied libraries.

4. How many lines of code are in the application?

Provide an indication of useful lines of code (exclude comment lines from this value).

## EJB usage

EJB usage questions are as follows:

1. What version of the EJB specification is used?
2. Do all the enterprise beans in the application conform strictly to this version of the specification?
3. How many Stateless Session beans are in the application?
4. How many Stateful Session beans are in the application?
5. How many CMP Entity beans are in the application?
6. How many BMP Entity beans are in the application?
7. How many Message Driven beans are in the application?

## Servlets and JSPs

Servlets and JSP questions are as follows:

1. What version of the JSP specification is used?
2. Do all the JSP files in the application conform strictly to this version of the specification?
3. How many JSPs files are in the application?
4. Do your JSP files contain Java code?  
List the third-party tag libraries (taglibs) used by the application. Include version numbers if possible.
5. Describe any custom tag libraries (taglibs) you constructed for the application.
6. What version of the servlet specification is used?

7. Do all the servlets in the application conform strictly to this version of the specification?
8. How many servlets are in the application?
9. Approximately how much data is maintained in the HttpSession?

This information can be difficult to estimate, especially if the size of the session data varies widely. One possible way to get an accurate measure of a session at a particular moment in time is to build a servlet that uses an `ObjectOutputStream` to dump the contents of the `HttpSession` to a file. The size of the file is a good approximation (though not a precise measure of the session size).

10. Are all objects that are stored in the HttpSession serializable?

The application server uses serialization to pass data between servers (JVMs). An object is serializable if it implements the `java.io.Serializable` interface.

Answer Yes to this question only if all objects are serializable, which includes objects that are indirectly included in the `HttpSession` through references from other objects.

11. Describe any issues that might prevent sharing data your application stores in the HttpSession across servers in a cluster.

## Web services

Web services questions are as follows:

1. Enumerate the external web services your application accesses.

By *external*, we mean external to the application, which includes any web services that are hosted by other applications regardless of the language or platform they are running on. If possible, provide a description of each web service that is used and note any potential issues that you are aware of that might affect the migration.

2. Enumerate the internal web services your application accesses.

An *internal* web service is one that is part of your application. Your application might, for example, use a web service to implement the Facade pattern between your servlet and EJB layers.

3. Enumerate the web services your application implements.

For this answer, list the web services that your application provides for internal or external clients. If possible, provide a description of the function that each web service provides.

4. Are the web services implemented by your application WS-I compliant? If so, indicate the WS-I version.

## Database access

Database access questions are as follows:

1. What version of the JDBC specification is used?
2. Does the usage of JDBC conform strictly to this version of the specification?
3. How many distinct JDBC queries does the application make?
4. What database managers are used by the application?
5. Do the JDBC queries use vendor-specific SQL extensions?
6. Does the application make direct use of `DriverManager`?
7. Does the application use J2EE connection pooling through data sources?
8. Describe any third-party connection pooling mechanisms in use.
9. Does the application make use, for example, of Oracle Connection Pooling?
10. Are all JDBC connections explicitly closed?

11. Are all JDBC statements explicitly closed?
12. Are all JDBC resources closed inside `finally` clauses?

Code that is contained in a `finally` block of an exception handler runs (whether an exception occurs or not). By closing these shared resources in a `finally` clause, you ensure that these resources can be kept available.

## JMS

JMS questions are as follows:

1. What version of the JMS specification is used?
2. Does the usage of JMS conform strictly to this version of the specification?
3. How many queues does the application use?
4. How many topics does the application use?
5. What data is passed through JMS?  
For example, "plain text", "XML", or "serialized objects"
6. Does the application write to JMS queues?
7. Does the application read from JMS?
8. Does the application do blocking reads from JMS?
9. Does the application do blocking reads from JMS in Session or Entity beans?

## JNDI naming

JNDI naming questions are as follows:

1. Does the application use JNDI to access resources other than EJB Homes?
2. Do the application's deployment descriptors contain local JNDI resource references?  
JNDI resource references sections are in `web.xml` and `ejb-jar.xml`.
3. Describe a typical JNDI access from the application. Provide a code snippet if possible.
4. Are all JNDI resources (`InitialContexts`) closed inside `finally` clauses?

Code that is contained in a `finally` block of an exception handler runs (whether an exception occurs or not). By closing these shared resources in a `finally` clause, you ensure that these resources can be kept available.

## Application trace and logging

Application trace and logging questions are as follows:

1. What technology is used by the application to provide logging?  
The application might use, for example, a third-party product, such as Log4J. Alternatively, the application might write messages to standard out (`System.out`).
2. Is logging used consistently throughout the application?

## Struts

If the application does not use Struts, skip this section. Struts questions are as follows:

1. What version of Struts is used?
2. How many custom ActionForms are in the application?
3. How many custom ActionHandlers are in the application?
4. Describe any modifications that you made to Struts.
5. Struts is an open source framework and is customized by organizations to provide extended functionality. If you extended Struts, describe that extension.

## Transactions

Transactions questions are as follows:

1. Are transactions managed implicitly or explicitly?
2. Does the application require two-phase commits (2PC)?
3. What transactional systems does the application interact with?

## Threads

This section is concerned with the application's direct manipulation of threads. This section is not concerned with threads managed by the application server.

Thread questions are as follows:

1. Does the application create threads?
2. Does the application create daemon threads?
3. Does the application explicitly stop threads using the **stop()** method?
4. List any third-party thread pool libraries that are used by the application. Include version numbers if possible.
5. Describe any custom thread pool libraries that are used by the application.
6. Describe how the application uses threads.

This question is concerned with how your application makes explicit use of threads. For example, does your application create threads to perform work in parallel, or to provide timeouts for various services? If it is the latter, indicate what services the threads are providing timeouts for.

## Sockets

Skip this section if your application does not make direct use of sockets. This section is concerned with sockets created and used directly by your application; it is not concerned with sockets managed by the application server.

Socket questions are as follows:

1. Describe how the application makes direct use of sockets.
2. Does your application create any daemon sockets?
3. Does your application use secure sockets (SSL)?

## XML

XML questions are as follows:

1. What XML libraries does the application use? Include version information.
2. What XSLT libraries does the application use? Include version information.

## Development migration questionnaire

The following questions are related to the development environment for the migration.

### Workstation configuration

Workstation configuration questions are as follows:

1. What hardware is used?
2. What operating system and version is used?
3. How much RAM is installed?

### Integrated development environment

Integrated development environment questions are as follows:

1. What Java integrated development environments (IDE) are used?
2. What source code management (SCM) tool is in use?
3. How are source directories organized?
4. What tools are used for unit testing?
5. What other tools do you use as part of your development process?
6. Will WebSphere development tools be adopted as part of this migration effort?
7. Which development tools are adopted as part of this migration?
  - WebSphere Studio Application Developer
  - WebSphere Studio Application Developer Integration Edition
  - Rational Application Developer
  - Other

### Development test configuration

Development test configuration questions are as follows:

1. Is dedicated development test hardware available for the development teams?
2. What hardware is used?
3. What operating system and version is used?
4. How much RAM is installed?
5. What application server and version is used for development testing?
6. What other software is used for development testing?

### Software development skills

Software development skills questions are as follows:

1. How many software developers form the team that built the application?
2. How many of your developers have architect-level skills?
3. How many developers are available to do the migration?



4. How familiar are your developers with Java EE?
  - They are experts with a comprehensive knowledge of Java EE.
  - They are familiar with the various technologies that make up Java EE (such as servlets and EJBs).
  - They are not at all familiar with Java EE.
5. Do your developers have any experience using Eclipse for software development?
6. Do your developers have experience using WebSphere Studio for software development?
7. Are there specific education requirements for the development team?

## Development methodology

Development methodology questions are as follows:

1. What development methodology is used?
2. How long is a typical development cycle?
3. How frequently are internal releases of the application delivered?  
These deliveries include releases for internal testing or user acceptance testing.
4. How frequently are external releases of the application delivered?
5. How often are new versions of application delivered into the production environment?

## Build and packaging

Build and packaging questions are as follows:

1. What tools are used to build the application?  
Examples: Apache Ant (Ant), make, or none. Provide version information if possible.
2. In what form are packaged applications delivered to the runtime environment? Are applications packaged as EAR, WAR, or JAR files? Or are they delivered in another form?
3. How are packaged applications delivered to the runtime environment?
4. What mechanisms are in place to deliver the application to the runtime environment? Is the packaged application delivered through FTP, or by another transport mechanism?
5. What process is followed to deliver the application?
6. How is static content packaged and delivered to the runtime environment?
7. Are installation and configuration scripts included with the packaged application?

## Ant

Skip this section if you do not use Ant for builds.

1. What version of Ant is used?
2. Does your build process use one or many Ant scripts?
  - One
  - Many
3. What optional Ant tasks do you use as part of your build?

# Runtime migration questionnaire

The following questions are runtime migration-related.

## General

General questions are as follows:

1. Will the existing runtime infrastructure be migrated?
2. Will you support the existing runtime infrastructure after the migration?  
Answer Yes if you expect to migrate only your current applications and leave them running on the current infrastructure.
3. Does your runtime infrastructure include a dedicated pre-production staging environment?  
Your pre-production environment might be called the Quality Assurance environment. Regardless of the given name, this environment is used to test the overall quality of your environment and applications before you move those applications into the production environment.
4. Does your runtime infrastructure include a dedicated performance testing environment?  
Describe the separate runtime environments (production, pre-production, system test, development test, performance test, and so on) that are included in your runtime infrastructure.

## Current hardware

Questions about the current hardware are as follows:

1. Describe the hardware that is used to run your current application server.  
If your environment is heterogeneous, describe each hardware configuration. Indicate the quantity of each configuration. Also, indicate the operating system and version in use.
2. How many production machines currently host the production application?  
Describe any additional hardware that is used in the production environment. Include all the details from firewalls, IP sprayers, web servers, clustering manager, database servers, firewall, intrusion detection systems, and so on. Indicate software in use (if any) on these systems, including operating system and version.
3. Describe how the configuration of the pre-production environment differs from the configuration of the production environment.
4. Is the pre-production environment configured with a similar complexity, but at a smaller scale? Are there key elements that are missing from the pre-production environment?
5. Describe the hardware that you plan to use to run the migrated applications.  
If your environment is heterogeneous, describe each hardware configuration. Indicate the quantity of each configuration. Also, indicate the operating system and version in use.
6. How many production machines are used to host the production application?  
Describe any additional hardware that is used in the production environment. Include all the details from firewalls, IP sprayers, web servers, clustering manager, databases, firewall, intrusion detection systems, and so on. Indicate software in use (if any) on these systems, including operating system and version.

## Software

Questions about software are as follows:

1. Which version of Java Runtime Environment (JRE) do your production hosts use?
2. Provide the name of the provider of the JRE, which might be Sun, IBM, or another JRE vendor.
3. Describe other software that is installed and used on the same production hardware as the application servers.
4. Describe the infrastructure of the host bridge network if the application employs the connection services to host systems (for example, CICS).

## HTTP Server

HTTP Server questions are as follows:

1. What HTTP Server is used?
2. Is the HTTP Server configured to provide affinity to a particular application server instance?
3. How many servers host the web servers?  
  
IBM WebSphere Application Server can work with almost all web servers in the market today. It is important to understand how the current web server is being used and what is the impact if WebSphere Application Server continues to use the same web server during the production switch.
4. Describe any other services that are provided by the HTTP server, which includes services beyond serving static content and fronting for the application server. Does your HTTP server, for example, provide additional functionality using CGI/Perl?
5. Describe the infrastructure for secure network propagation.
6. Enabling HTTPS on the web server is one way of doing it from the browser to the web server end. Enterprises also use hardware accelerators to improve the speed of the SSL encryption and the decryption process. If so, how does it integrate with the web server?
7. Is the communication from the HTTP server plug-in to the application server secure in the current environment?

## Network edge

Network edge questions are as follows:

1. What load balancing technologies are used?
2. Is the load balancer configured to provide affinity to a particular HTTP server instance?
3. Describe how failover is provided for your load balancer.

## Availability

Availability questions are as follows:

1. Describe any existing service level agreements that are concerned with the availability of the applications, which determines the availability and the amount of flexibility to switch to a WebSphere Application Server run time.
2. Describe how clustering is provided in the current environment. Explain your clustering environment.
3. Describe the process for the production release rollout of an application?  
If the process differs for various applications, describe each of the processes.
4. What is the current backup strategy when a node or server fails or breaks down?  
Explain in detail your failover strategy.

## Rollout issues

Questions about rollout issues are as follows:

1. Will old applications need to coexist with migrated applications?
2. Does the migration include all the applications on the production server? If not, is there a plan to migrate those applications as well?  
Co-existence is important, especially when migrating common frameworks and application development models into a new platform. There can be application dependencies that must be maintained when moved into the new platform.
3. Are there any other software updates scheduled to be done to the production machines during the migration process?  
Migration of the production environment is an important task and mitigation of risk is key during the migration. Hardware and software updates are only preferred when they are required as part of the migration. They should be decoupled to keep the variables to a minimum.

## Administration

Administration questions are as follows:

1. How many administrators do you have to support the production environment?
2. Do you have dedicated administrators for your production infrastructure?
3. Describe the skill level and experience of the administrators of the production environment.
4. How do you deploy the applications?
5. Do the production administrators use deployment scripts to accomplish this task? Does it involve custom scripting? Explain the process.
6. Describe any custom scripts that are used for administering the application servers.
7. Do you use any custom scripts for administering the current setup? If so, what are they used for?
8. Is the production environment set up to take care of log rotation and file system maintenance?

9. Log files can become large and consume the available space within the production host. Is there a procedure or a process in place to handle file system maintenance?

## Security

Security questions are as follows:

1. How important is security in your production runtime infrastructure?
  - Critical
  - Very important
  - Important
  - Not important
2. Describe how internal security of the application server is managed.

There are several roles applicable within the production environment, including root administrator, machine maintenance operator, application deployer, and developer. Explain how access to the application server hardware and software is managed regarding these roles.
3. How are internal security policies enforced?
4. How are users authenticated?
5. How are user authorizations managed?
6. Describe your use of application-specific services for user authentication and authorization.
7. Describe your use of custom services for user authentication and authorization.
8. Describe any third-party security mechanisms in use, which might include products such as Netegrity SiteMinder.
9. Describe how your production run time provides (or participates) in a single sign-on solution. Describe any network security measures in place.
10. Are firewalls positioned between tiers in your configuration? If so, how and where are they configured?

## Testing migration questionnaire

The following questions are for the testing environment.

### Hardware

Questions about the hardware that is used are as follows:

1. Is dedicated hardware available specifically for user acceptance testing?
2. Is dedicated hardware available specifically for quality assurance testing?
3. Is dedicated hardware available specifically for performance testing?
4. Describe how the testing hardware is different in configuration from the production hardware.

## Practices and tools

Questions about the practices and tools that are used are as follows:

1. What testing is done and when?
2. What is the typical length of the regression test cycle?
3. How frequently do applications undergo regression testing?
4. What is the typical length of the user acceptance test cycle?
5. How frequently do applications undergo user acceptance testing?
6. Is user acceptance testing required as part of this migration effort?
7. What is the typical length of the quality assurance test cycle?
8. How frequently do applications undergo quality assurance testing?
9. Describe the testing stages that an application is taken through when you move it from the development environment into production.
10. How do you test applications for compliance with the J2EE specification?
11. Do you have an established set of application development preferred practices? These practices include coding standards, code maintenance, and performance preferred practices.
12. How are application development preferred practices enforced?
13. What tools do you use to unit test your application code?
14. What tools do you use to function test your application code?
15. What tools do you use to regression test your application code?
16. What tools do you use to performance test your application code?
17. Does the testing procedure contain any harness whose code is dependent on the source platform?



## Additional material

This book refers to additional material that can be downloaded from the Internet as described in the following sections.

### Locating the web material

The web material that is associated with this book is available in softcopy on the Internet from the IBM Redbooks web server. Point your web browser at:

<ftp://www.redbooks.ibm.com/redbooks/SG248048>

Alternatively, you can go to the IBM Redbooks website at:

[ibm.com/redbooks](http://ibm.com/redbooks)

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG24-8048.

### Using the web material

The additional web material that accompanies this book includes the following files:

<i>File name</i>	<i>Description</i>
<b>WebLogic_migration.zip</b>	Configuration and application files that are used in Chapter 6, “Migrating from Oracle WebLogic” on page 127
<b>OracleAS_migration.zip</b>	Configuration and application files that are used in Chapter 7, “Migrating from Oracle Application Server” on page 221

<b>Tomcat_migration.zip</b>	Configuration and application files that are used in Chapter 9, “Migrating from Apache Tomcat” on page 273
<b>WASV2V_migration.zip</b>	Configuration and application files that are used in Chapter 11, “Installation and configuration of the Application Migration Tool - WebSphere Version to Version” on page 357

## **System requirements for downloading the web material**

The web material requires the following system configuration:

**Hard disk space:** 20 MB minimum  
**Operating system:** Linux or Windows

## **Downloading and extracting the web material**

Create a subdirectory (folder) on your workstation, and extract the contents of the web material compressed file into this folder.



# Related publications

The publications that are listed in this section are considered suitable for a more detailed discussion of the topics that are covered in this book.

## IBM Redbooks

The following IBM Redbooks publications provide more information about the topic in this document. Some publications referenced in this list might be available in softcopy only.

- ▶ *Experience JEE Using Rational Application Developer V7.5*, SG24-7827
- ▶ *Getting Started with WebSphere Application Server Feature Pack for Service Component Architecture*, REDP-4633
- ▶ *Oracle to DB2 Conversion Guide: Compatibility Made Easy*, SG24-7736
- ▶ *Rational Application Developer for WebSphere Software V8 Programming Guide*, SG24-7835
- ▶ *WebSphere Application Server V6.1: Classloader Problem Determination*, REDP-4307
- ▶ *WebSphere Application Server V7: Competitive Migration Guide*, SG24-7870
- ▶ *WebSphere Application Server V8.5 Concepts, Planning, and Design Guide*, SG24-8022

You can search for, view, download, or order these documents and other Redbooks, Redpapers, Web Docs, draft and more materials, at the following website:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Other publications

This publication is also relevant as a further information source:

- ▶ *Migrating a Microsoft Access 2000 Database to IBM DB2 Universal Database 7.2*

## Online resources

These websites are also relevant as further information sources:

- ▶ Apache Tomcat 7.0.27 download website  
<http://tomcat.apache.org/download-70.cgi>
- ▶ Apache Tomcat 7.0.27: Instructions for installing, configuring, and managing  
<http://tomcat.apache.org/tomcat-7.0-doc/index.html>
- ▶ Hibernate Validator distribution download page  
<http://www.hibernate.org/subprojects/validator/download>
- ▶ IBM developerWorks article on typesafe queries on JPA 2.0  
<http://www.ibm.com/developerworks/java/library/j-typesafejpa>

- ▶ IBM Education Assistant - IBM WebSphere Application Server Migration Toolkit V3.5  
<http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/topic/com.ibm.iea.wasmt/wasmt/3.5/Overview.html>
- ▶ IBM Passport Advantage  
<http://www-01.ibm.com/software/support/probsub.html>
- ▶ IBM WebSphere Developer Services - How to raise a PMR  
<http://www.ibm.com/isv/tech/remoteEmail/entryForm.jsp>
- ▶ Java EE specification  
<http://www.oracle.com/java>
- ▶ Java Specification Request (JSR) 286 compliant portlets  
<http://jcp.org/en/jsr/detail?id=286>
- ▶ JBoss download website  
<http://www.jboss.org/jbossas/downloads/>
- ▶ Online information about the Migration Toolkit  
<http://www.ibm.com/developerworks/websphere/downloads/migtoolkit/index.html>
- ▶ OSGi applications  
<http://www.osgi.org/About/WhatIsOSGi>
- ▶ Oracle Java Application Verification Kit (AVK)  
[http://java.sun.com/j2ee/verified/avk\\_enterprise.html](http://java.sun.com/j2ee/verified/avk_enterprise.html)
- ▶ Oracle WebLogic Server 10.3.6 installation instructions  
[http://www.oracle.com/technology/software/products/ias/htdocs/wls\\_main.html](http://www.oracle.com/technology/software/products/ias/htdocs/wls_main.html)
- ▶ Quickstarts distribution download page  
<http://download.jboss.org/jbossas/7.1/jboss-as-7.1.1.CR2/jboss-as-quickstarts-7.1.1.CR2-dist.zip>
- ▶ Shrinkwrap library  
<http://www.jboss.org/shrinkwrap>
- ▶ SLF4J distribution download page  
<http://www.slf4j.org/download.html>
- ▶ Spring Framework introduction  
[http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=cspr\\_intro](http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-mp&topic=cspr_intro)
- ▶ Using Spring and Hibernate with WebSphere Application Server article  
[http://www.ibm.com/developerworks/websphere/techjournal/0609\\_alcott/0609\\_alcott.html](http://www.ibm.com/developerworks/websphere/techjournal/0609_alcott/0609_alcott.html)
- ▶ WebSphere Application Server—Base V8.5  
<http://www.ibm.com/software/webservers/appserv/was/>
- ▶ WebSphere Application Server—Express V8.5  
<http://www.ibm.com/software/webservers/appserv/express/>
- ▶ WebSphere Application Server for Developers V8.5  
<http://www.ibm.com/software/webservers/appserv/developer/index.html>

- ▶ WebSphere Application Server Information Center  
<http://publib.boulder.ibm.com/infocenter/iisinfsv/v8r5/index.jsp>
- ▶ WebSphere Application Server Network Deployment V8.5  
<http://www.ibm.com/software/webservers/appserv/was/network/>
- ▶ WebSphere Application Server for z/OS V8.5  
[http://www.ibm.com/software/webservers/appserv/zos\\_os390/](http://www.ibm.com/software/webservers/appserv/zos_os390/)

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)





Redbooks

## WebSphere Application Server V8.5 Migration Guide

(0.5" spine)  
0.475" x 0.873"  
250 <-> 459 pages







# WebSphere Application Server V8.5 Migration Guide



**Redbooks®**

**Review case studies for competitive and version migrations**

**Learn about the migration strategy and planning**

**Become proficient with the Application Migration Toolkit**

This IBM Redbooks publication helps you plan and execute the migration of J2EE applications that are developed for Oracle WebLogic Server, Oracle Application Server, JBoss, and Apache Tomcat, so that they run on IBM WebSphere Application Server V8.5. In addition, this book covers migration from earlier versions of WebSphere Application Server to WebSphere Application Server V8.5.

This book provides detailed information to plan migrations, suggested approaches for developing portable applications, and migrating working examples for each of the platforms from which we migrated in our examples. The primary tool that is used in the migration scenarios that are covered in this book is the IBM WebSphere Application Server Migration Toolkit V3.5.

It is not our intention to provide a feature-by-feature comparison of these application servers versus WebSphere Application Server, but to produce practical technical advice for developers who must migrate applications from these vendors to WebSphere Application Server V8.5.

This publication is an update of *WebSphere Application Server V7: Competitive Migration Guide*, SG24-7870.

The book is intended as a migration guide for IT specialists who are working on migrating applications that are written for other application servers or earlier versions of WebSphere Application Server to WebSphere Application Server V8.5.

## **INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION**

### **BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)

SG24-8048-00

ISBN 0738437247