

Getting Started With

Hands-on
exercises
Included

WebSphere Application Server **Community Edition**

Ideal for application developers and administrators



by:

Jiang Lin Quan

Dai Xuan

Wang Lei

Juliano Marcos Martins

Chi Run Hua

Xia Ming

Tang Ying

Raul F. Chong

GETTING STARTED WITH

WebSphere Application Server

Community Edition

A book for the community by the community

Jiang Lin Quan, Dai Xuan, Wang Lei,
Juliano Marcos Martins, Chi Run Hua, Xia Ming, Tang Ying, Raul F. Chong

FIRST EDITION

First Edition (June 2010)

© Copyright IBM Corporation 2010. All rights reserved.

IBM Canada
8200 Warden Avenue
Markham, ON
L6G 1C7
Canada

This edition applies to WebSphere Application Server Community Edition (Community Edition) version 2.1.1.3 and later.

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.*

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

*Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711*

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

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

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

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

6 Getting Started with WebSphere Application Server Community Edition

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

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.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates.

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

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "[Copyright and trademark information](http://www.ibm.com/legal/copytrade.shtml)" at www.ibm.com/legal/copytrade.shtml.

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

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

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

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

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

Table of Contents

Preface	13
Who should read this book?	13
How is this book structured?.....	13
A book for the community	13
Conventions	14
What's next?	14
About the Authors.....	17
Contributors	18
Acknowledgements	18
PART I – OVERVIEW AND SETUP	19
Chapter 1 – Introduction to WebSphere Application Server Community Edition....	21
1.1 A brief history of Community Edition.....	21
1.2 WebSphere application server family	22
1.3 Target users of Community Edition	24
1.4 User assistance and technical support.....	24
1.5 Components of Community Edition	25
1.6 Java EE compliance matrix	26
1.7 Related free products: DB2 Express-C and IBM Data Studio	28
1.8 Summary.....	29
1.9 Review questions.....	29
Chapter 2 – Installing Community Edition	33
2.1 Installing Community Edition: The big picture	33
2.2 System requirements	34
2.3 Obtaining Community Edition	34
2.4 Installing Community Edition	36
2.5 Validating the installation	40
2.6 Community Edition and DB2 Express-C on the Cloud	42
2.7 Exercises	42
2.8 Summary.....	43
2.9 Review questions.....	43
PART II - JAVA EE DEVELOPMENT WITH COMMUNITY EDITION.....	45
Chapter 3 – Development with Community Edition	47
3.1 Development with Community Edition: The big picture	47
3.2 Downloading and installing IBM Data Studio or Eclipse.....	48
3.3 Eclipse Integration with WEP.....	49
3.4 Creating and deploying a "Hello World" Web application.....	54
3.5 Servlets	58
3.6 JSPs.....	59
3.7 Manually deploying an application.....	60

3.8 Exercises	63
3.9 Summary.....	63
3.10 Review questions.....	63
Chapter 4 – Working with databases	65
4.1 Community Edition database connectivity: The big picture	65
4.2 Steps to work with a database in Community Edition.....	66
4.2.1 Creating a database	66
4.2.2 Configuring and deploying a database pool	67
4.2.3 Executing SQL statements to load some data	73
4.3 Developing a Web application to access a database	75
4.3.1 Create a Dynamic Web Project	75
4.3.2 Create a JSP and configure the data source reference	77
4.3.3 Deploy and run the project on the Community Edition server	83
4.4 Support for other databases in Community Edition	86
4.5 Summary.....	86
4.6 Exercises	86
4.7 Review questions.....	87
Chapter 5 – Enterprise Java Beans.....	89
5.1 Community Edition Enterprise Java Beans: The big picture	89
5.2 Developing EJBs in Community Edition	90
5.2.1 Creating an EJB.....	90
5.2.2 Deploying an EJB	92
5.3 Working with EJBs in Community Edition.....	93
5.3.1 Using EJBs in a servlet.....	93
5.3.2 Using an EJB in an application client	95
5.4 Java Persistence API Entities Development in Community Edition	97
5.4.1 Creating JPA entities manually.....	97
5.4.2 Generating JPA entities from tables of a database	98
5.5 Summary.....	99
5.6 Exercises	99
5.7 Review questions.....	107
Chapter 6 – Messaging.....	109
6.1 Community Edition Messaging: The big picture	109
6.2 Java Message Service.....	110
6.2.1 JMS application roles.....	110
6.2.2 JMS application models.....	111
6.2.3 JMS API.....	112
6.2.4 JMS application development steps in Community Edition.....	112
6.3 Configuring a JMS resource group in Community Edition	113
6.3.1 Creating a JMS resource group.....	113
6.3.2 Creating a JMS connection factory.....	114
6.3.3 Creating a JMS queue and topic destinations	116
6.3.4 Stop, restart, or uninstall an installed JMS resource group.....	118

6.4 Using Community Edition JMS resource	118
6.4.1 Accessing queues or topics from a Web application	119
6.4.2 Message-driven beans	120
6.4.3 Stand-alone Java application	121
6.5 Summary.....	121
6.6 Exercises	122
6.7 Review questions.....	123
Chapter 7 – Web Services	127
7.1 Community Edition Web Services: The big picture	127
7.2 Developing Web Services in Community Edition	128
7.2.1 Creating a Web Service project in Eclipse	129
7.2.2 Creating a service endpoint interface	129
7.2.3 Providing the service implementation class.....	129
7.2.4 Deploying the Web Service in Community Edition	130
7.3 Consuming a Web Service deployed in Community Edition	131
7.3.1 Creating necessary stubs to consume the Web Service.....	131
7.3.2 Creating a POJO client.....	133
7.3.3 Creating a Web client	136
7.3.4 Using service reference tag in a Java EE application	138
7.4 Creating Data Web services with IBM Data Studio	139
7.5 Summary.....	141
7.6 Exercises	141
7.7 Review questions.....	142
Chapter 8 – Security	145
8.1 Community Edition Security: The big picture.....	145
8.2 Configuring the security realm	146
8.3 Implementing security in a Java EE application	149
8.3.1 Defining role mapping.....	149
8.3.2 Implementing EJB Security.....	149
8.3.3 Implementing Web security	152
8.4 Managing users and groups for Community Edition administrative interface	154
8.5 Securing your data with trusted contexts.....	155
8.6 Summary.....	156
8.7 Exercises	156
8.8 Review questions.....	158
PART III – ADMINISTERING COMMUNITY EDITION	161
Chapter 9 – Administering Community Edition	163
9.1 Administering Community Edition: The big picture.....	163
9.2 Starting and stopping the server.....	164
9.2.1 Starting the server	164
9.2.2 Stopping the server.....	165
9.3 Configuring the Community Edition server manually.....	165
9.3.1 Setting the IP address and hostname	166

9.3.2 Changing port numbers	166
9.3.3 Changing the username and password.....	167
9.4 Introducing the administrative console	167
9.4.1 Welcome	168
9.4.2 Server	169
9.4.3 Services	169
9.4.4 Applications	169
9.4.5 Security	170
9.4.6 Debug Views.....	170
9.4.7 Embedded DB	171
9.5 Adding JARs to the Community Edition repository.....	171
9.6 Administering applications	172
9.6.1 Deploying and undeploying applications	172
9.6.2 Starting and stopping applications.....	174
9.7 Tools and commands.....	175
9.7.1 The deploy command	175
9.8 Configuring multiple server instances.....	176
9.9 Configuring multiple repositories	176
9.10 Exercises	177
9.11 Summary.....	178
9.12 Review questions.....	178
Chapter 10 – Tuning a Community Edition server	181
10.1 Tuning a Community Edition server: The big picture.....	181
10.2 Monitoring Community Edition.....	182
10.3 Community Edition Server tuning	185
10.3.1 Thread pool size	185
10.3.2 Monitoring thread pools	186
10.3.3 Configuring the thread pool size.....	187
10.4 JVM and operating system tuning	188
10.4.1 JVM tuning.....	188
10.4.2 Operating system tuning.....	191
10.5 Summary.....	191
10.6 Exercises	191
10.7 Review questions.....	192
Chapter 11 - Troubleshooting.....	195
11.1 Troubleshooting: The big picture	195
11.2 Problems during installation/un-installation	196
11.2.1 JVM not found.....	196
11.2.2 Platform-specific Problem.....	196
11.2.3 Uninstalling Community Edition doesn't remove all the files	197
11.3 Problems starting or stopping the Community Edition server.....	197
11.3.1 JAVA_HOME or JRE_HOME environment variable is not specified.....	197
11.3.2 Port already in use.....	198

11.3.3 Could not communicate with the server	198
11.4 Classpath and dependency	199
11.5 Using Community Edition log files as diagnostic tools	199
11.5.1 Installation and un-installation logs.....	199
11.5.2 Server log	200
11.5.3 Client log	201
11.5.4 Deployer log.....	201
11.5.5 Web log.....	201
11.5.6 DB2 database log	201
11.5.7 System.out and System.err	202
11.6 Summary.....	202
11.7 Review questions.....	203
Chapter 12 – Advanced features	205
12.1 GShell	205
12.2 Customizing a new server assembly	206
12.3 Plug-in management.....	206
12.4 WADI Clustering	207
12.5 Farming deployment.....	207
12.6 Review questions.....	208
Appendix A – Solutions to review questions.....	211
Appendix B – Up and running with DB2.....	219
B.1 DB2: The big picture.....	219
B.2 DB2 Packaging.....	220
B.2.1 DB2 servers.....	220
B.2.2 DB2 Clients and Drivers	221
B.3 Installing DB2	222
B.3.1 Installation on Windows.....	222
B.3.2 Installation on Linux.....	223
B.4 DB2 tools	224
B.4.1 Control Center	224
B.4.2 Command Line Tools	225
B.5 The DB2 environment	229
B.6 DB2 configuration.....	230
B.7 Connecting to a database	231
B.8 Basic sample programs	233
B.9 DB2 documentation	234
Appendix C – Using the sample code.....	235
Resources.....	237
Web sites	237
Books	239
Contact.....	240

Preface

Keeping your skills current in today's world is becoming increasingly challenging. There are too many new technologies being developed, and little time to learn them all. The DB2® on Campus Book Series has been developed to minimize the time and effort required to learn many of these new technologies.

Who should read this book?

This book is intended for anyone who works with or intends to work with a Java™ Platform Enterprise Edition (Java EE) application server, such as Java EE application developers, deployers, administrators, consultants, software architects, instructors, and students. The book assumes you have a basic knowledge of Java and Java EE; therefore concepts related to these topics such as a **servlet** will not be fully explained or not explained at all. If you don't have these skills, see the "What's Next?" section below for other free eBooks where you can gain these skills.

How is this book structured?

This book has three parts:

- **Part I, Overview and Setup**, explains what WebSphere® Application Server Community Edition is all about, introduces the WebSphere Application Server family of products and features, and discusses installation.
- **Part II, Java EE Development with Community Edition** introduces WebSphere Application Server Community Edition application development, including an introduction to the developer environment, Servlet/JSP, EJB/JPA, JMS, and Web service development.
- **Part III – Administering Community Edition** is designed to familiarize you with the Community Edition admin console, application deployment, tuning, and troubleshooting.

Exercises are provided for most chapters; and any input files required are provided in the compressed file `gettingStartedWithWasceEdition1st_src.zip` that accompanies this book.

A book for the community

This book was created by the community; a community consisting of university professors, students, and professionals (including IBM employees). The online version of this book is released to the community at no-charge. Numerous members of the community from around the world have participated in developing this book, which will also be translated to

several languages by the community. If you would like to provide feedback, contribute new material, improve existing material, or help with translating this book to another language, please send an email of your planned contribution to db2univ@ca.ibm.com with the subject "Getting started with Community Edition book feedback."

Conventions

Many examples of commands, SQL statements, and code are included throughout the book. Specific keywords are written in uppercase bold. For example: A **NULL** value represents an unknown state. Commands are shown in lowercase bold. For example: The **dir** command lists all files and subdirectories on Windows®. SQL statements are shown in upper case bold. For example: Use the **SELECT** statement to retrieve information from a table.

Object names used in our examples are shown in bold italics. For example: The ***flights*** table has five columns.

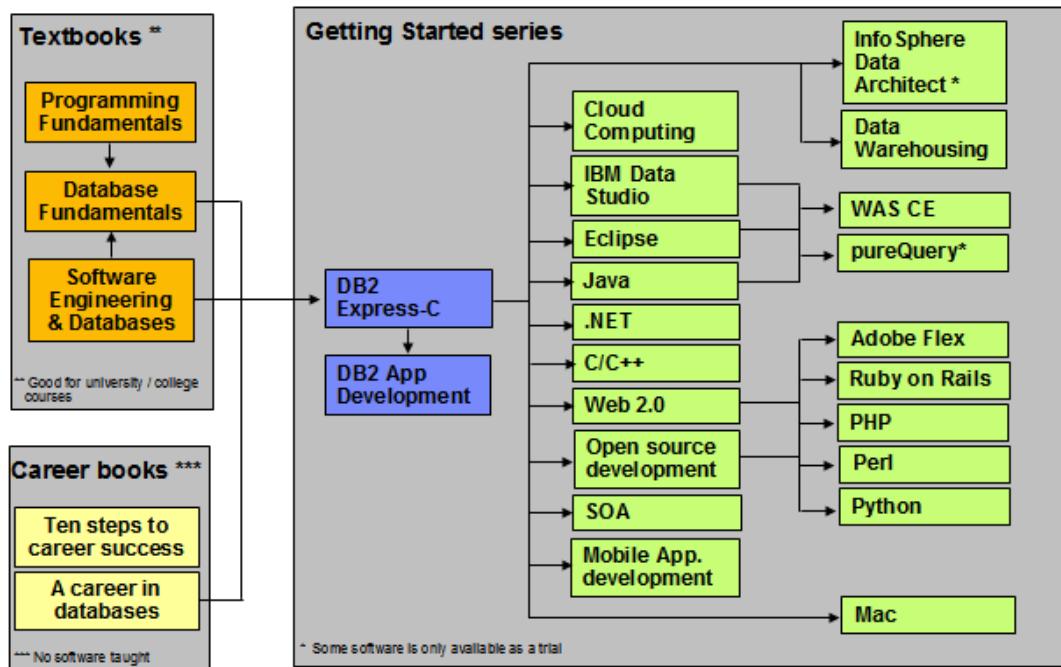
Italics are also used for variable names in the syntax of a command or statement. If the variable name has more than one word, it is joined with an underscore. For example: **CREATE TABLE *table_name***

What's next?

We recommend you to review the following books in this book series for more details about related topics:

- *Getting started with DB2 Express-C*
- *Getting started with IBM Data Studio for DB2*
- *Getting started with Java*
- *Getting started with Open source development*
- *Getting started with Eclipse*

The following figure shows all the different eBooks in the DB2 on Campus book series available for free at <http://www.db2university.com>



The DB2 on Campus book series

About the Authors

Jiang Lin Quan – Lead Author

Jiang Lin Quan is a Community Edition developer at the IBM China Development Lab.

Dai Xuan – Co-author and Editor

Dai Xuan is a Community Edition developer at the IBM China Development Lab.

Wang Lei – Co-author and Editor

Wang Lei is a Community Edition developer at the IBM China Development Lab.

Juliano Marcos Martins – Co-author and Editor

Juliano is a DB2 QA Software Engineer at the IBM Brazil Lab

Chi Run Hua – Co-author and Editor

Chi Run Hua is a Community Edition Information developer at the IBM China Development Lab.

Xia Ming – Co-author and Editor

Xia Ming is the Community Edition SVT lead at the IBM China Development Lab.

Tang Ying – Co-author and Editor

Tang Ying is a Community Edition information developer at the IBM China Development Lab.

Raul F. Chong – Co-author and Editor

Raul is the DB2 on Campus Program Manager at the IBM Toronto Lab.

Contributors

The following people edited, reviewed, and contributed significantly to this book.

Contributor	Company/University	Position/Occupation	Contribution
Ge Kang	IBM China Development Lab	Community Edition release manager	Technical review
Cai Jun Jie	IBM China Development Lab	Community Edition technical lead	Technical review
Xu Hai Hong	IBM China Development Lab	Community Edition developer	Technical review
Brian Holroyd	Database Consulting Services	IBM Gold Consultant	Full technical review
Cristian Molaro	MConsulting Bvba Belgium	DB2 Consultant and IBM Information Champion	Partial technical review
David Beulke	Pragmatic Solutions, Inc.	President	Partial technical review
Leon Katsnelson	IBM Toronto Lab	Program Director, IBM Data Servers	Technical review

Acknowledgements

We greatly thank the following individuals for their assistance in developing materials referenced in this book:

- Natasha Tolub who designed the cover of this book.
- Susan Visser who assisted with publishing this book.
- Carla Sadtler, Mohamed Ahmed, Rafael Thomas Goz Coutinho, Gianluca Finocchiaro, Anish Pathadan, Susil Piyanandana; authors of the redbook [WebSphere Application Server Community Edition 2.0 User Guide](#)
- Ueli Wahli, Charles P Brown, Steven Calello, Rafael Coutinho, Patrick Gan, Cedric Hurst, Maan Mehta; authors of the redbook [Experience Java EE! Using WebSphere Application Server Community Edition 2.1](#)

Both Redbooks® served as the framework for this book.

PART I – OVERVIEW AND SETUP

1

Chapter 1 – Introduction to WebSphere Application Server Community Edition

IBM® WebSphere® Application Server Community Edition (Community Edition) is a member of the IBM WebSphere Application Server family. Community Edition is a lightweight Java Platform Enterprise Edition (Java EE) application server built on Apache Geronimo, the open source application server project of the Apache Software Foundation. The community using Community Edition consists of all sorts of people and companies, who design, develop, deploy, or utilize Java EE solutions.

In this chapter you will learn about:

- The history of Community Edition
- The WebSphere application server family
- Target users and service support of Community Edition
- Components of Community Edition
- Java EE compliance matrix of Community Edition
- Related free products

1.1 A brief history of Community Edition

After the acquisition of Gluecode Software in May 2005, IBM devoted many technical resources to contribute to Apache Geronimo, the main Gluecode application server software, so it could attain Java EE certification.

By October 2005, this goal was reached with Geronimo Milestone 5 (M5), which was used as the foundation of Community Edition. Community Edition was one of the first IBM products to follow a new business model: Develop products based on open source software, offer them for free to use for development, testing and production; and optionally provide customers with fee-based IBM technical support. This provided users comfort when working with open source software especially in a production environment.

At the time this book was written, the latest Community Edition release is version 2.1.1.3 which is based on Geronimo version 2.1.4. *Figure 1.1* provides a roadmap that maps Geronimo with Community Edition release dates since 2008.

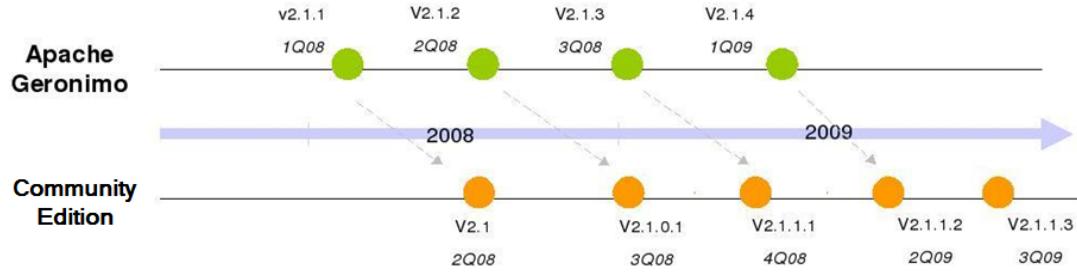


Figure 1.1 – Geronimo and Community Edition release roadmaps since 2008

As shown in the figure, nearly every 3 months after a new Geronimo release is announced, IBM delivers an updated release of Community Edition. Approximately 90% of Apache Geronimo code is used on Community Edition.

1.2 WebSphere application server family

Depending on your needs, the WebSphere Application Server family includes different editions that vary in footprint, capability and scalability. As shown in *Figure 1.2*, first in the family at the bottom left corner is Community Edition. As mentioned earlier, Community Edition is a lightweight application server you can use in development, test, or production to run your Java EE applications. If you have a small company, Community Edition is your best choice, both technically and economically. Community Edition is free to download and use. You can install it and get it running in a short period of time.

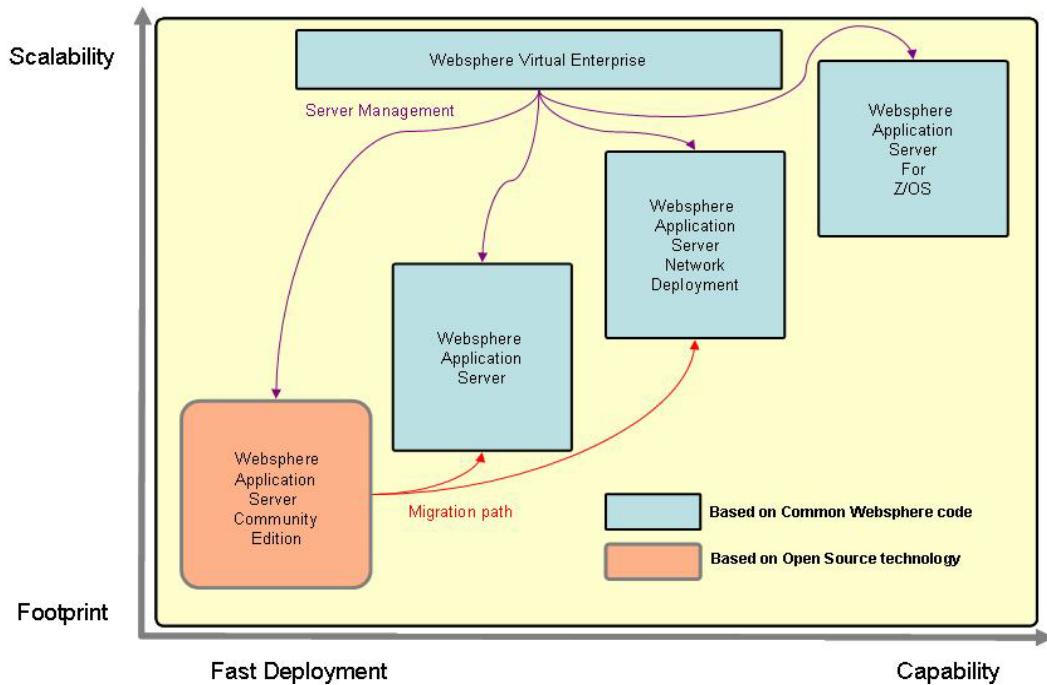


Figure 1.2 – WebSphere application server family

If you intend to run critical applications that require high availability, and want sophisticated management console and tooling capabilities to help you deploy and debug your Java EE applications, then you should explore other IBM WebSphere Application Server products. For example, if you require scalability, enterprise-wide deployment, functional depth and robustness, or if you have high transaction volumes, want autonomic application deployment, or business goal-based application prioritization, look into WebSphere Application Server or WebSphere Application Server Network Deployment. If your company has a mainframe, you may also consider WebSphere Application Server for z/OS®.

Figure 1.2 also shows a suggested upgrade path in the case that your company grows and requires an application server with more capability and scalability. You can take advantage of the free **Application Advancement Assistant** migration tools offered by IBM to upgrade to another WAS edition. For more information about this tool, visit <http://www.alphaworks.ibm.com/tech/wasma>.

You can also take advantage of virtualization to consolidate application servers and maximize utilization using WebSphere Virtual Enterprise.

For more information about the WAS family, visit.

<http://www.ibm.com/software/dre/hmc/compare.wss?HMC02=L666761W91427E61>

1.3 Target users of Community Edition

Community Edition is ideal for:

- **Application developers** who require an open standards Java EE application server for building Java EE applications, and who want to focus on the core business applications based on pre-integrated services
- **System administrators** who require a standard and flexible environment for application administration and scalability
- **Independent software vendors(ISVs) and other types of solution providers** who want to bundle or embed a full-featured Java EE application server as part of their solutions
- **Small and medium-sized companies** who need an entry-level Java EE application server with world-class support for their applications and operations
- **Departments of companies** who need a production Java EE environment without budget approval
- **Java EE hobbyists and cutting-edge technology enthusiasts** who want to leverage and experience open source technologies
- **Students, teachers, and other academic users** who want a highly versatile Java EE application server for teaching, courseware, projects and research

1.4 User assistance and technical support

Community Edition enables you to quickly develop applications and grow them incrementally, and provides basic security, easy administration, and different deployment alternatives. As you will see in *Chapter 2*, Community Edition can run on either 32-bit or 64-bit architecture with Linux® or Windows operating systems; it is also available on Solaris and AIX® systems. Community Edition can run on both Sun Java run time and IBM Java run time, where the latter one has significant advantages in terms of performance. Along with a Community Edition installation package, IBM offers dozens of out-of-the-box sample applications for practice and evaluation.

If you have technical questions about Community Edition, you can post your questions in the Community Edition forum at

<http://www.ibm.com/developerworks/forums/forum.jspa?forumID=541>

This free forum is monitored by Community Edition experts from IBM, though it is the community who provides most of the answers on a voluntary basis.

IBM also gives users the choice to purchase three tiered (entry, enhanced and elite support tiers), fee-based support as an annual subscription, priced per server. For more details, visit

<http://www-306.ibm.com/software/webservers/appserv/community/detail/table.html>

1.5 Components of Community Edition

Apache Geronimo brings together technologies from the broader open source community to support the Java EE stack. For example: Apache Tomcat, ActiveMQ, Tranql, OpenEJB and so on. Apache Geronimo uses the Apache license, is highly customizable, and is community-driven. *Figure 1.3* shows the components of Community Edition.

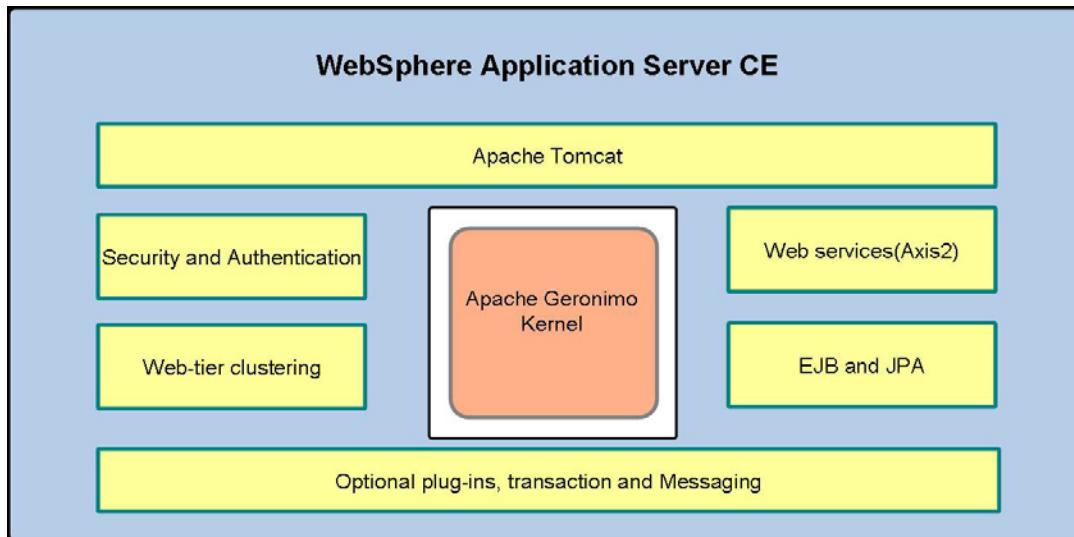


Figure 1.3 – Components of Community Edition

In the figure you can see that Community Edition is composed of the following:

- **Apache Geronimo kernel**, which provides a framework for adding other services like the custom business services;
- **Leading components** that developers need, like Web services, security, and messaging. They are integrated and tested with Community Edition, so developers don't have to worry about any custom integration work or version compatibility issues.
- **Administrator console**, powerful tool that allows developers to define Tomcat connections (HTTP, HTTPS, AJP), database pools, and security realms without any coding or restarting the server.

Table 1.1 provides a short description of the major integrated components in Community Edition:

Components or Features	Description
Apache Geronimo kernel	An open source Java EE 5 application server project which provides the foundation for Community Edition
Apache Derby	An embedded small-footprint database server suitable for very small and simple projects. For projects where

	future growth is a consideration, use DB2 Express-C database server which is free.
Apache Tomcat	A Web-tier container that is used in the Reference Implementation for Java Servlet and JavaServer Pages technologies
Apache OpenEJB	An embeddable and lightweight EJB 3.0 implementation
Apache OpenJPA	A Java persistence API implementation
Apache ActiveMQ	A Java messaging services implementation
WebSphere Application Server Community Edition Server Adapter (Eclipse plug-in)	A plug-in used to develop, deploy, and debug Java EE applications within the Eclipse IDE.
Built-in JDBC support for popular third party RDBMSs	Support Apache Derby, IBM DB2 , Oracle®, Microsoft® SQL Server®, MySQL® Community Edition
JVM support	IBM JVM support and Sun Java VM compatible

Table 1.1 – Components of Community Edition

A complete list of components can be found at

<http://publib.boulder.ibm.com/wasce/V2.1.1/en/open-source.html>

1.6 Java EE compliance matrix

Java EE is an industry standard for developing portable, robust, scalable and secure server-side Java applications. Java EE provides Web services, component model, management, and communications APIs that make it the industry standard for implementing enterprise class service-oriented architecture (SOA) and next-generation Web applications. A certified Java EE server provides standard interface for easy migration and upgrading.

The Sun Java EE site at <http://java.sun.com/javaee/> provides more information about Java EE.

The latest Java EE specification is version 5 which was created under the Java Community Process as JSR-244.

The current Community Edition release passed the Java EE 5 Compatibility Test Suite (CTS) which is a suite of compatibility tests to verify if a Java EE platform product complies with the Java EE 5 platform standard.

Table 1.2 shows the Community Edition feature compliance matrix with the Java EE 5 standard.

Components or packages	Specification
Web applications	<ul style="list-style-type: none"> ▪ Servlet 2.5 ▪ JSP 2.1 ▪ JSTL 1.2 ▪ JSF 1.2
Enterprise Applications	<ul style="list-style-type: none"> ▪ Common Annotations 1.0 ▪ Connector 1.5 ▪ EJB 3.0 ▪ JPA 1.0 ▪ JAF 1.1 ▪ JTA 1.1 ▪ JavaMail 1.4 ▪ JMS 1.1
Web Services	<ul style="list-style-type: none"> ▪ Web Services Metadata 2.0 ▪ Web Services 1.2 ▪ JAXB 2.0 ▪ JAX-WS 2.0 ▪ JAX-RPC 1.1 ▪ SAAJ 1.3 ▪ SOAP 1.1/1.2 ▪ WSDL 1.6 ▪ StAX 1.0
Management and Security	<ul style="list-style-type: none"> ▪ JACC 1.1 ▪ Java EE Management 1.1 ▪ Java EE Application Deployment 2.7

Table 1.2 – Compliance matrix with Java EE 5

Note:

Java Platform, Enterprise Edition (Java EE) was formerly known as Java 2 Platform,

Enterprise Edition (J2EE).

1.7 Related free products: DB2 Express-C and IBM Data Studio

Most applications require some sort of persistent storage for the application data. IBM provides a database product that is a perfect complement to the Community Edition application server: **DB2 Express-C**. DB2 Express-C is the free version of DB2. The "C" in its name stands for "Community". DB2 Express-C delivers performance, reliability, scalability and security to Community Edition applications. DB2 Express-C is well suited for the new breed of Web-based Java applications that need to work with XML data. It provides a unique hybrid relational-XML database server that delivers excellent performance for both traditional relational, and XML data.

Like Community Edition, DB2 Express-C is available at no charge for development and production deployment. It can be redistributed as part of an overall solution without any royalty charges. Optional support and subscription is available for a low per server yearly charge. DB2 Express-C is built on the same product code as the rest of the DB2 server portfolio; therefore, an application written to run on DB2 Express-C is guaranteed to run on any other DB2® for Linux®, UNIX®, and Windows® server. In most cases, the application will also run unchanged with DB2 for z/OS® mainframe and DB2 for i5/OS® midrange database servers. In other words, DB2 Express-C delivers unprecedented range of scalability for your application.

DB2 Express-C comes with a free **IBM Data Studio**, an Eclipse-based development and administration tool that will be very familiar to most Community Edition users and will make you instantly productive. IBM Data Studio IDE in conjunction with DB2 Express-C can be used to publish industry standard SOAP/XML and RESTfull Web services as a means of accessing data in DB2.

DB2 Express-C is available for 32 and 64-bit Linux, 32 and 64-bit Windows, Solaris x64, and 64-bit Linux for POWER® servers. A beta version is also available for Apple Mac OS

X. Figure 1.4 shows the progression of the different DB2 Editions available.

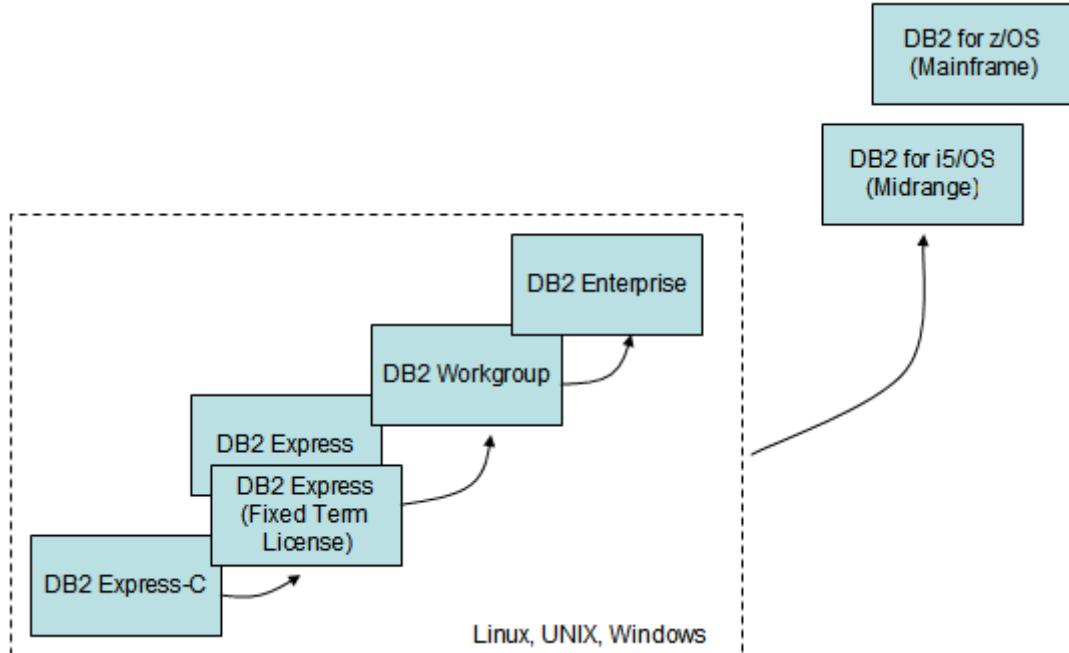


Figure 1.4 - DB2 editions and their progression

Note:

For more information about DB2 Express-C and IBM Data Studio, visit ibm.com/db2/express. You can also learn more from the books *Getting Started with DB2 Express-C* and *Getting Started with IBM Data Studio for DB2*. Both books are part of the DB2 on Campus free book series.

1.8 Summary

IBM WebSphere Application Server Community Edition offers a best-of-breed product at no cost. It delivers the freedom to develop, deploy and distribute Java EE platform applications without any limitations. It is based on Apache Geronimo open source application server, and you can optionally purchase IBM technical support. If you want to upgrade to other editions of WAS, the free Application Advancement Assistant for WebSphere migration tool can make it easy for you. DB2 Express-C database server and IBM Data Studio IDE are free products that can complement Community Edition to store persistent data and develop Web services.

1.9 Review questions

1. What is the relationship between Geronimo and Community Edition?

2. What is the URL of the Community Edition forum where users can post questions, and collaborate?
3. What operating system platforms is DB2 Express-C available?
4. What is the Java persistence API implementation in Community Edition?
5. What is the distinguishing characteristic of the DB2 database server?
6. Which one of the following components in Community Edition is used for Java Message Services implementation?
 - A. Apache Derby
 - B. Apache Geronimo kernel
 - C. Apache OpenEJB
 - D. Apache ActiveMQ
 - E. None of the above
7. What is the tool used for configuration and administration of DB2 Express-C servers?
 - A. Community Edition
 - B. DB2 for z/OS
 - C. IBM Data Studio
 - D. All of the above
 - E. None of the above
8. Which one of the following specifications is not about Web application in Java EE 5 standard?
 - A. Servlet 2.5
 - B. JSP 2.1
 - C. JSTL 1.2
 - D. JSF 1.2
 - E. Servlet 3.0
9. Which one(s) of the following groups are not the target users of Community Edition?
 - A. Application developers
 - B. ISVs
 - C. Departments of companies
 - D. Teachers and students
 - E. None of Above

10. Which of the following products is the best choice if you need a robust environment for mission-critical applications?
 - A. Geronimo
 - B. WebSphere Application Server Community Edition
 - C. WebSphere Application Server
 - D. All of the above
 - E. None of the above

2

Chapter 2 – Installing Community Edition

Community Edition can run on Linux, UNIX or Windows and is very easy to install. It has no licensing limitations in terms of the hardware resources you can use. If you are working on the cloud and don't want to bother installing products, there are Community Edition and DB2 Express-C images available for development, test, and production use on the Amazon EC2 cloud.

In this chapter you will learn about:

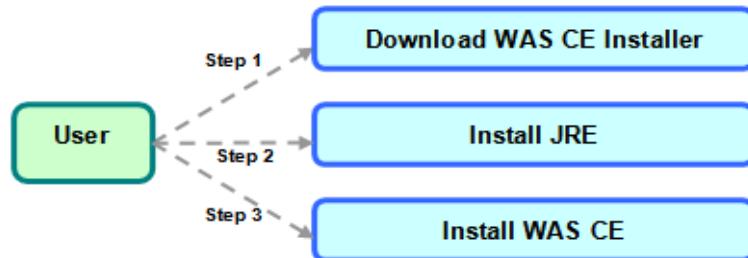
- How to obtain the Community Edition code
- How to install Community Edition on Windows
- How to install Community Edition on Linux
- Community Edition and DB2 Express-C on the cloud

2.1 Installing Community Edition: The big picture

Installing Community Edition consists of 3 basic steps:

1. Download Community Edition,
2. Obtain and install a Java Runtime Environment (JRE), and
3. Install Community Edition.

We focus our discussion on step 1 and 3 in this book. For step 2, refer to the *Getting started with Java* free ebook. *Figure 2.1* illustrates the steps to follow to get started with Community Edition.

**Figure 2.1 - Steps to get started with Community Edition**

2.2 System requirements

In terms of operating systems, Community Edition can run on Windows (XP, Vista, 2003), Linux (Red Hat, Suse, Asianux), AIX, and Solaris. For Linux, Community Edition may run on other distributions, but the product may not have been officially tested with those distributions. In this book we use Windows XP SP3 and Ubuntu Linux.

In terms of hardware requirements, Community Edition needs 120 MB of disk space and a minimal of 140MB of memory, but this will depend on how you plan to use your Community Edition server. For more details about the requirements visit

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

2.3 Obtaining Community Edition

You can get the latest version of Community Edition by going to

<http://www.ibm.com/developerworks/downloads/ws/wasce/> as shown in *Figure 2.2*

developerWorks > WebSphere > Downloads >

Download: WebSphere Application Server Community Edition

[Learn](#) [Try](#) [Buy](#) [Support](#)

 [Download WebSphere App Server - Community](#)

Download WebSphere Application Server Community Edition, a lightweight Java EE application server built on Apache Geronimo technology.

WebSphere Application Server Community Edition is a fully licensed product available for a no-charge download.

Download

Requirements

System requirements

To download using Download Director or HTTP, click the **Download now** link below. The estimated download time over a 1.5Mbps connection is 9 minutes for the minimum file size and 21 minutes for the maximum file size. We recommend that you use Download Director, which requires a Java-enabled browser, provides pause-and-resume capability for large downloads, high-speed transfer, and file access through firewalls.

Operating system	Version	Size *	Download method
Download now AIX, Linux, Solaris, UNIX, Windows	2.1.1.3	83MB to 194MB	HTTP Download Director

* The minimum file size represents the core code, whereas the maximum file size includes all optional components.

Figure 2.2 - Download site for Community Edition

In this book we use version 2.1.1.3. After answering some questions and agreeing to the license, you will be presented with a window as shown in *Figure 2.3*. Be sure to download the version applicable to your operating system.

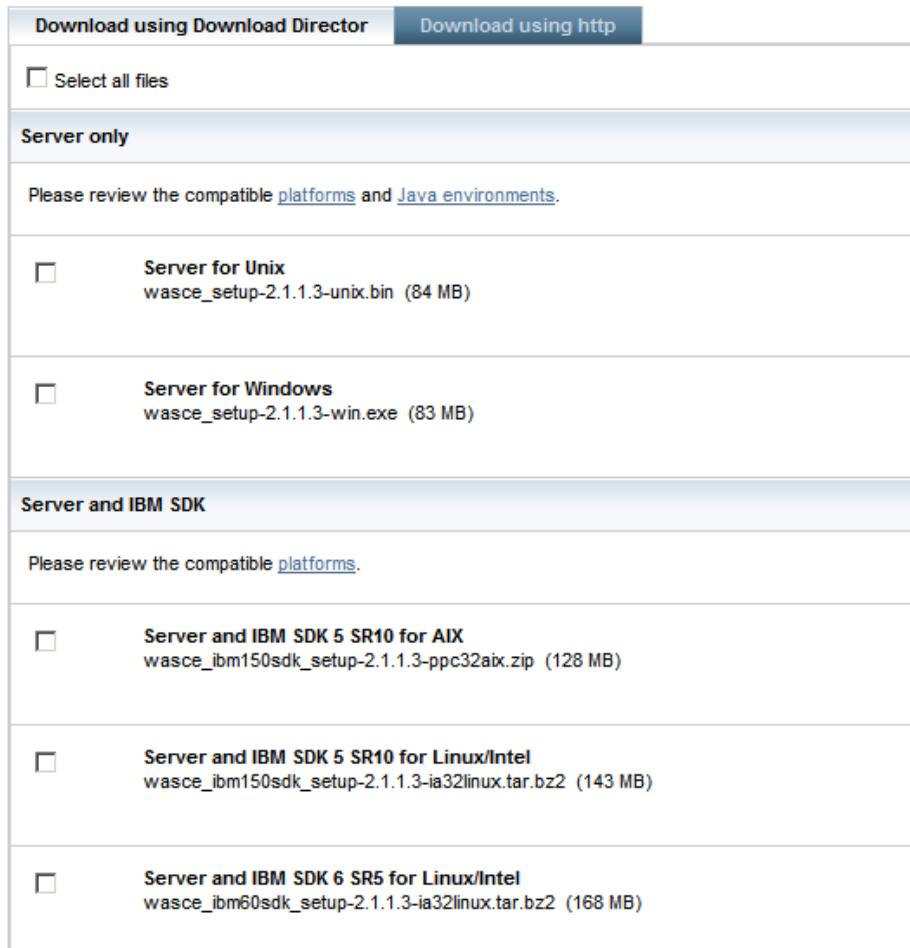


Figure 2.3 - Selecting what to download

As shown in the figure, you can choose to download the Community Edition server only if you already have a JRE installed. If not, choose to download the server and an IBM® SDK for Java™.

In this book, since we will install on Windows and Linux on a 32-bit system, and we don't have a JRE already installed, we choose these options:

- Server and IBM SDK 6 SR5 for Linux/Intel - wasce_ibm60sdk_setup-2.1.1.3-ia32linux.tar.bz2 (168 MB)
- Server and IBM SDK 6 SR5 for Windows - wasce_ibm60sdk_setup-2.1.1.3-ia32win.zip (194 MB) (not shown in *Figure 2.3*)

You can also download the samples application package (not shown in the figure) from the above URL. This is optional but recommended since the samples package includes templates for developing and deploying your own Java EE assets.

After extracting the samples from the sample package, review the `Readme.txt` file included in each sample's directory to learn how to customize and use the sample.

2.4 Installing Community Edition

In order to install Community Edition, you have to follow the instructions below:

1. (Linux only) Log on as the root user, and create a directory under which the Community Edition v2.1.1.3 package can be installed by the non-root user.
 - Run the following command to create the `/opt/IBM` directory:
`mkdir /opt/IBM`
 - Run the following command to transfer ownership of this directory to the `users` group (or to whichever group your non-root user belongs to):
`chgrp users /opt/IBM`
 - Run the following command to allow members of the users group to create and make changes inside of `/opt/IBM`:
`chmod g+w /opt/IBM`
 - Log out as the root user, and log in as a non-root user.
2. Go to the folder where you downloaded the installation image (there should be only one file) and run it. For Windows it should be the `wasce_setup-2.1.1.3-win.exe` file. Double-click to execute it. For Linux it should be the `wasce_setup-2.1.1.3-unix.bin` file. Execute it as follows:

```
$ chmod +x wasce_setup-2.1.1.3-unix.bin  
$ ./wasce_setup-2.1.1.3-unix.bin
```
3. You will see a splash screen as shown in *Figure 2.4*.

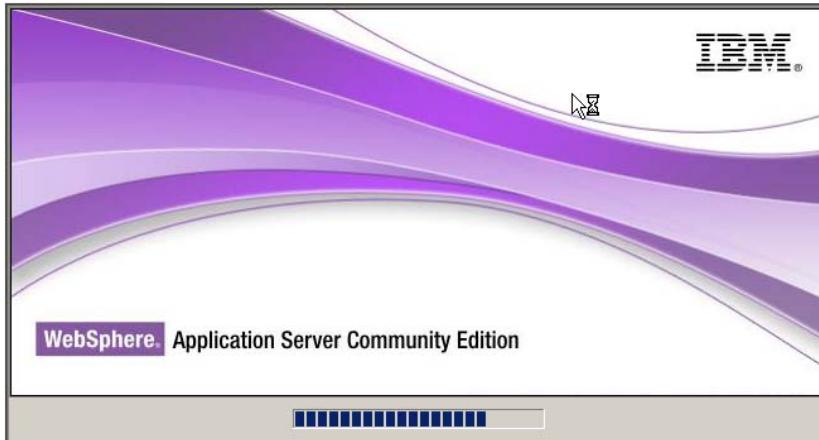


Figure 2.4 - Splash screen when installing Community Edition

4. Click *Next* in the welcome screen shown in *Figure 2.5*.

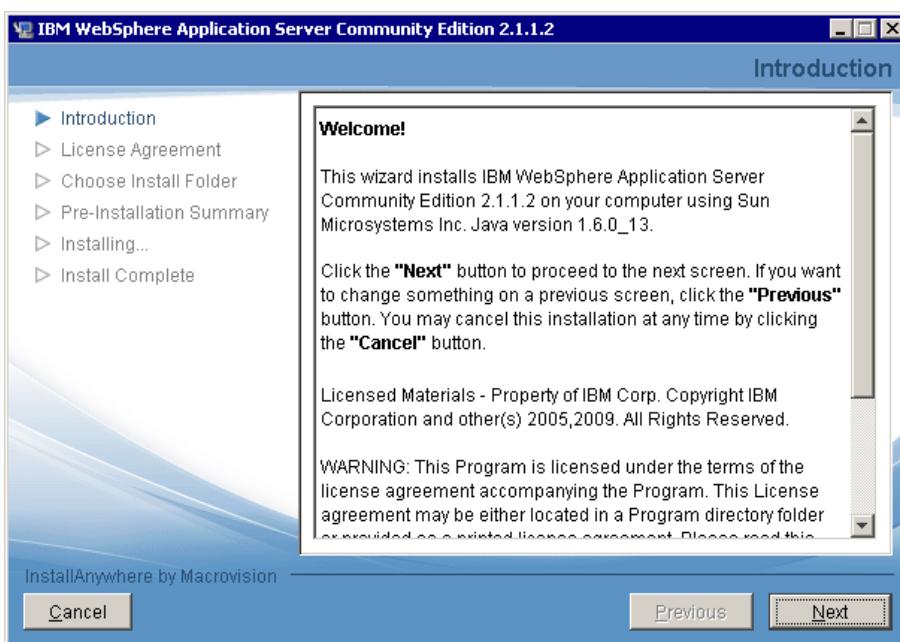


Figure 2.5 - Welcome screen

5. Read and accept the license shown in *Figure 2.6*. Then click *Next*.

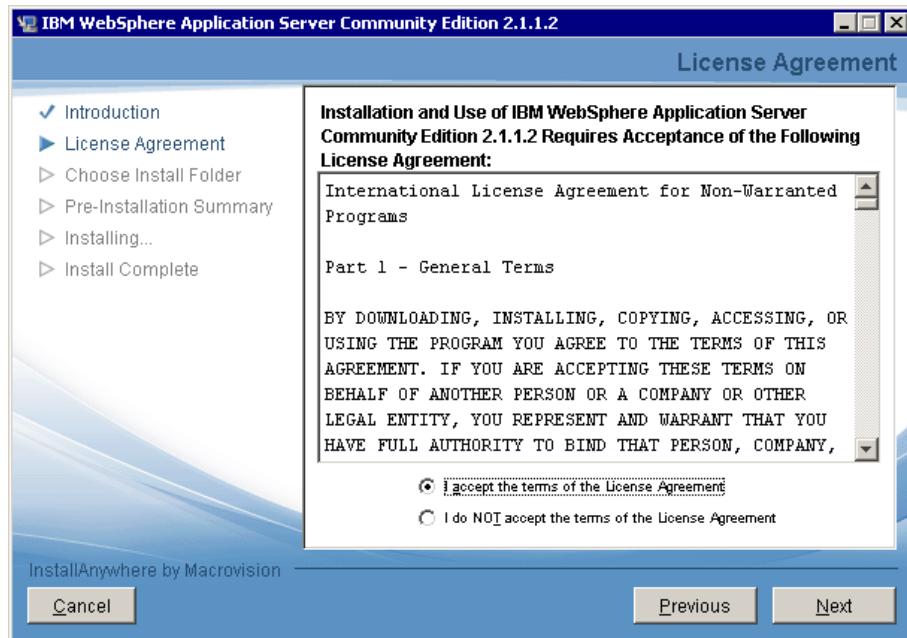


Figure 2.6 - License Agreement page

6. In the window shown in *Figure 2.7*, enter the installation directory. For Windows we used C:\IBM\WASCE21, for Linux we used /opt/IBM/WASCE21

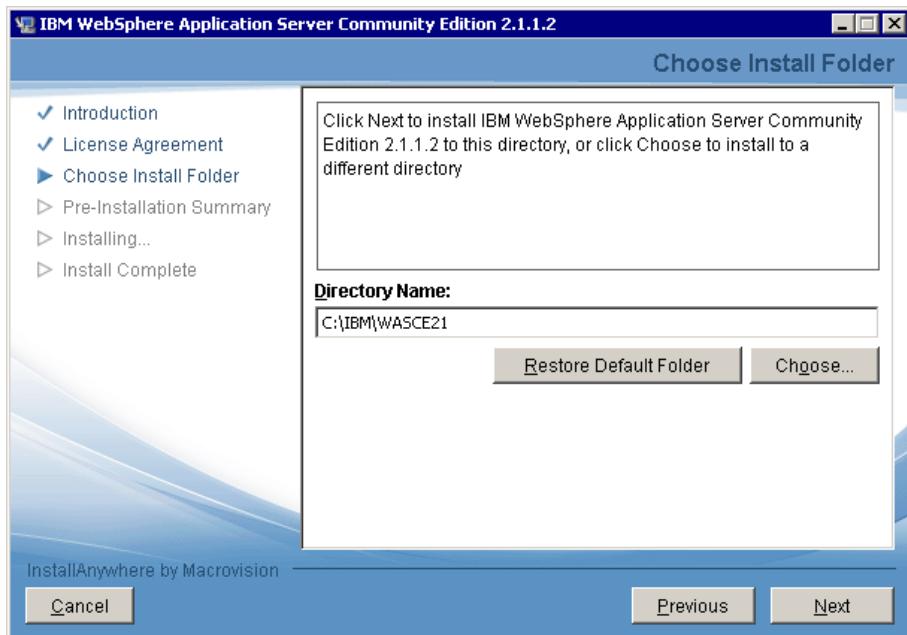


Figure 2.7 - Choosing the install folder

7. Click *Install* in the pre-installation summary page shown in *Figure 2.8*.

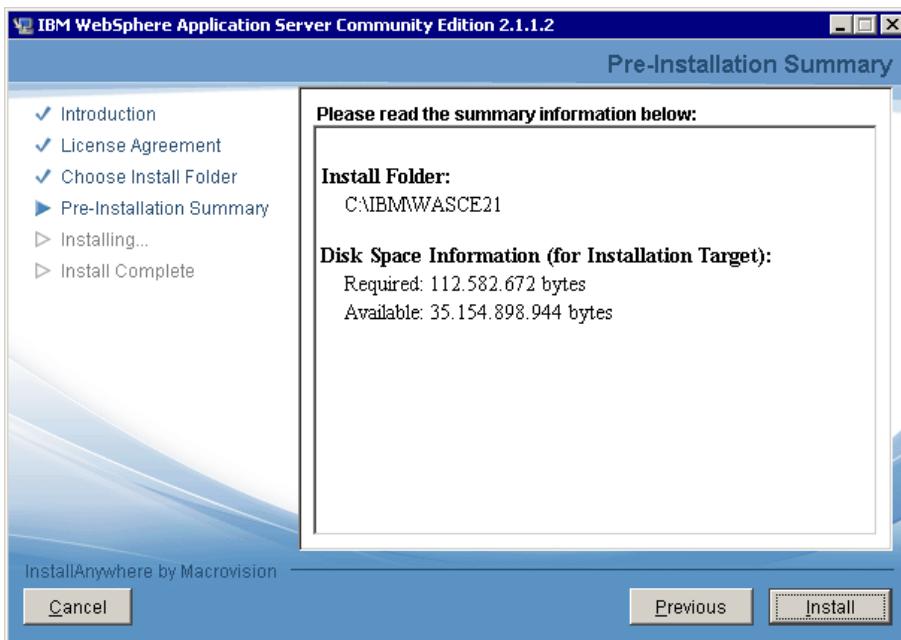


Figure 2.8 - Pre-installation summary page

8. The installation will start; it can take some minutes depending on your machine configuration. This is shown in *Figure 2.9*.

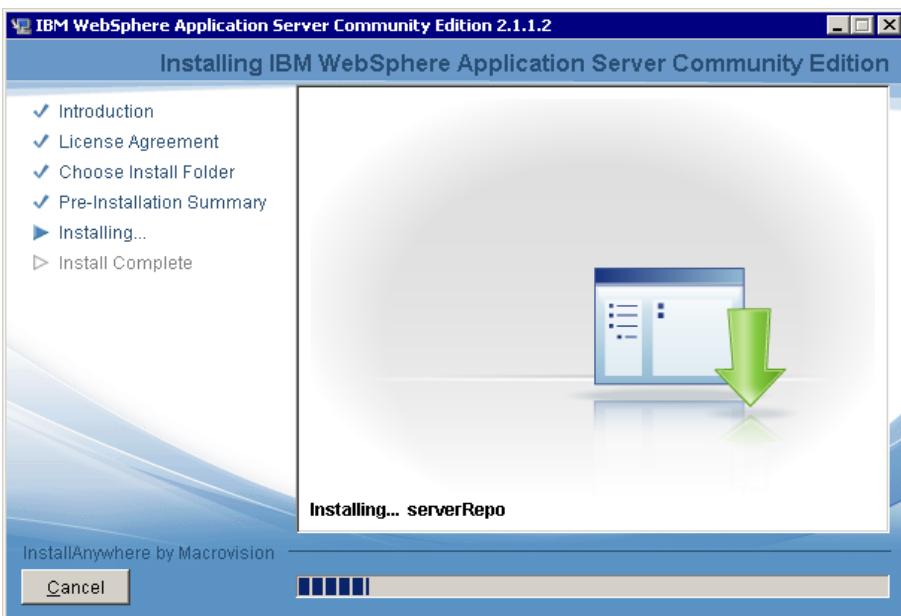


Figure 2.9 - Installation progress bar

9. When the install is completed, you can see the success message as shown in *Figure 2.10*. Click *Done*.

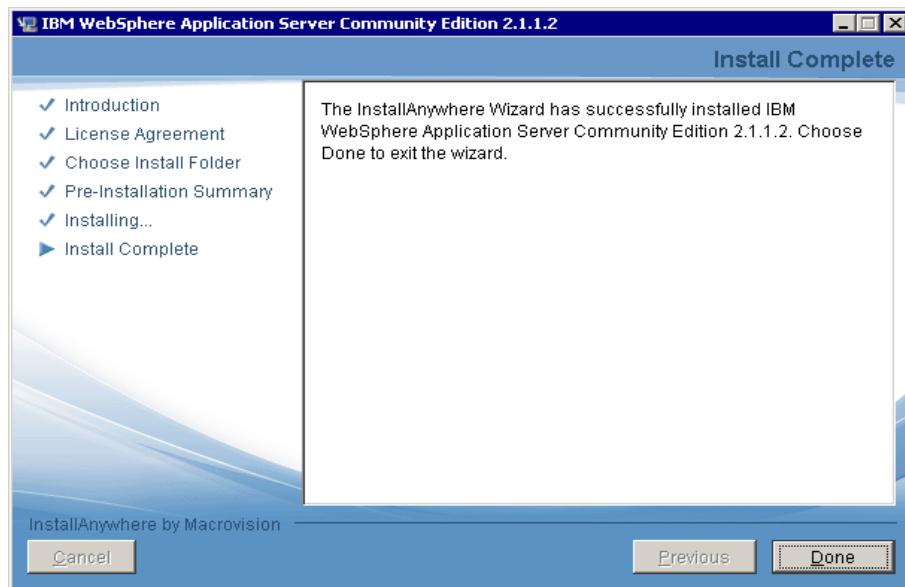
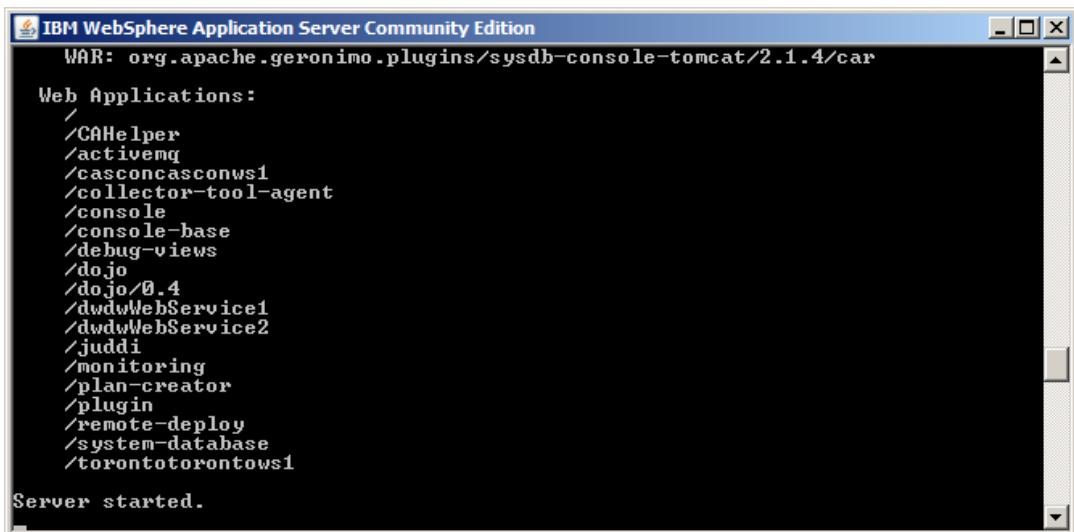


Figure 2.10 - Installation complete

2.5 Validating the installation

In order to ensure Community Edition was correctly installed, start the Community Edition server and launch the Administration Console.

On Windows you can start the Community Edition server by going to *Start -> Programs -> IBM WebSphere -> Application Server Community Edition -> Start the server*. A black window as shown in *Figure 2.11* will appear displaying a list of messages. This may take a few minutes.



```
IBM WebSphere Application Server Community Edition
WAR: org.apache.geronimo.plugins/sysdb-console-tomcat/2.1.4/car

Web Applications:
/
/CAHelper
/activemq
/casconcasconws1
/collector-tool-agent
/console
/console-base
/debug-views
/dojo
/dojo/0.4
/dwdwWebService1
/dwdwWebService2
/juddi
/monitoring
/plan-creator
/plugin
/remote-deploy
/system-database
/torontotorontows1

Server started.
```

Figure 2.11 - Starting Community Edition: Server started message

Once you receive the message "Server started", go back to the same Windows menu and choose *Administrative Console*. This will start the Administrative Console in your browser located at <https://localhost:8443/console> as shown in *Figure 2.12*

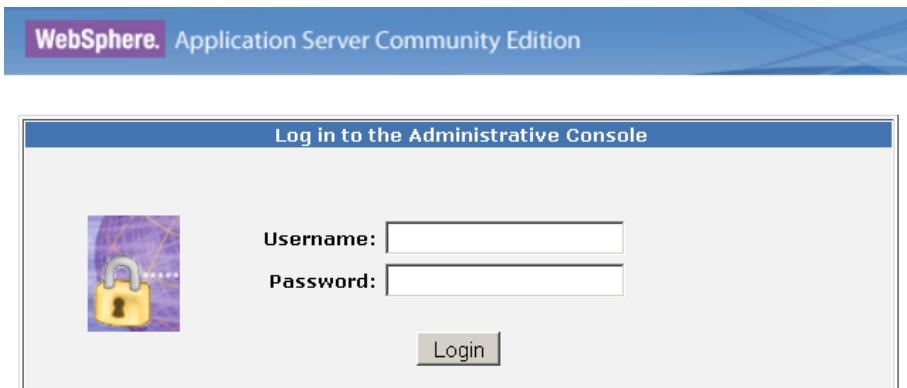


Figure 2.12 - Starting the Community Edition Administrative Console to validate your installation

To start the Community Edition server you can also go to the folder where you installed Community Edition (See *Figure 2.7* above) and run **start-server.bat**.

On Linux start the Community Edition server using **start-server.sh**. To start the admin console, simply open a browser and point it to <https://localhost:8443/console>

To log in to the Administrative Console, the default user is **system** and the password is **manager**. When you click login, you should see the Administrative Console main window. We will discuss more about the Administrative Console later in this book.

Later in the book we will use the Eclipse plug-in to start the server, so at this time, if this test was successful, we suggest you stop the server.

To stop the server, on Windows choose *Start -> Programs -> IBM WebSphere -> Application Server Community Edition -> Stop the server*. Alternatively use **stop-server.bat** from the same directory where you executed the commands to start it. On Linux use **stop-server.sh**. To stop the server, you will be prompted for a user ID and a password. Use the **system/manager** combination.

2.6 Community Edition and DB2 Express-C on the Cloud

If you would like to experience Community Edition and DB2 Express-C without having to install these products, they are both available on the Cloud. The Cloud is ideal for development and testing, and can get you up and running in no time without having you to get a server on your own to install these products. IBM has partnered with RightScale™ and Amazon Web Services to make these products available on the Amazon EC2 cloud.

You will find a Community Edition and DB2 Express-C all-in-one template in the template library at <https://my.RightScale.com>. The direct link is https://my.rightscale.com/server_templates/51010

First, you need to get a free developer account at http://www.rightscale.com/products/free_edition.php. There are no charges for using Community Edition or DB2 Express-C on the cloud but you will have to pay Amazon for the use of the hardware resources but only for the time that your Community Edition and DB2 servers are up and running.

Note:

For more information about Cloud Computing refer to the book *Getting started with Cloud Computing* which is part of this DB2 on Campus free book series.

2.7 Exercises

In this exercise, you will download Community Edition, install it, and then try to start/stop the Community Edition server.

Procedure

1. Based on your platform, Download the latest Community Edition here:
<http://www.ibm.com/developerworks/downloads/ws/wasce/>
2. Install Community Edition on your machine.
3. Use **start-server.bat** (on Windows) or **start-server.sh** (on Linux) to start Community Edition.
4. Login in the administrative console to verify the startup of Community Edition.

5. Use `stop-server.bat` (on Windows) or `stop-server.sh` (on Linux) to shutdown Community Edition.

2.8 Summary

In this chapter you have learned how to install Community Edition on Linux and Windows platforms. You also learned how to start and stop the Community Edition server, and how to launch and log on to the Community Edition Administrative Console. If you do not have a server available to install Community Edition, a good alternative is to use the Cloud. RightScale is an IBM partner and it has an all-in-one template that includes Community Edition and DB2 Express-C ready to use on the Amazon Cloud.

2.9 Review questions

1. Where can I get the latest Community Edition image?
2. What platforms does Community Edition support?
3. I would like to work with Community Edition, but my company doesn't have a server available for me to install it. What are my choices?
4. Do I have to pay for using Community Edition and DB2 on the Cloud?
5. How can I verify if the Community Edition server is started?
6. Which of following platforms are supported by Community Edition?
 - A. Windows (XP, Vista, 2003)
 - B. Linux (Red Hat, Suse, Asianux)
 - C. AIX
 - D. Solaris
 - E. All of the above
7. How much disk space is the minimum needed to install Community Edition?
 - A. 120 MB
 - B. 30 MB
 - C. 50 MB
 - D. 500 MB
 - E. 1 GB
8. How much memory is the minimum needed to install Community Edition?
 - A. 30 MB
 - B. 140 MB

- C. 500 MB
 - D. 1 GB
 - E. 120MB
9. Which script should you use to start Community Edition on Linux or UNIX?
- A. start-server.sh
 - B. stop-server.sh
 - C. start-server.bat
 - D. stop-server.bat
 - E. deploy.bat
10. Which URLs should you use to test if the Community Edition server is started?
- A. <https://localhost:8443/console>
 - B. <http://localhost:8080/admin>
 - C. <http://localhost:8080/console>
 - D. <http://localhost:8888/console>
 - E. <https://localhost:8443/admin>

PART II - JAVA EE DEVELOPMENT WITH COMMUNITY EDITION

3

Chapter 3 – Development with Community Edition

This chapter discusses how to develop a Java EE application in Eclipse and how to deploy it into Community Edition. In order to work with Community Edition within Eclipse, the **WebSphere Application Server Community Edition Eclipse Plugin (WEP)** must be installed. WEP is the adapter that facilitates Web application development in Eclipse with Community Edition. As described earlier in this book, the free IBM Data Studio is an Eclipse-based IDE for working with DB2 database servers. WEP functionality is included in IBM Data Studio allowing you to work with Community Edition and DB2 immediately. Whenever you see Eclipse mentioned in this chapter, you can substitute it with IBM Data Studio.

In this chapter you will learn about:

- How to install WEP into Eclipse
- How to develop a basic Java Web Application
- How to deploy an application into Community Edition
- Working with IBM Data Studio as an alternative to "plain" Eclipse

3.1 Development with Community Edition: The big picture

Figure 3.1 provides an overview about development with Community Edition. On the right side of the figure, a developer creates a Java EE application using Eclipse and deploys it using WEP to the Community Edition server shown in the middle of the figure. JSPs or Servlets created and stored in the Web container in Community Edition can then be accessed through HTTP by a Web client shown on the right side of the figure.

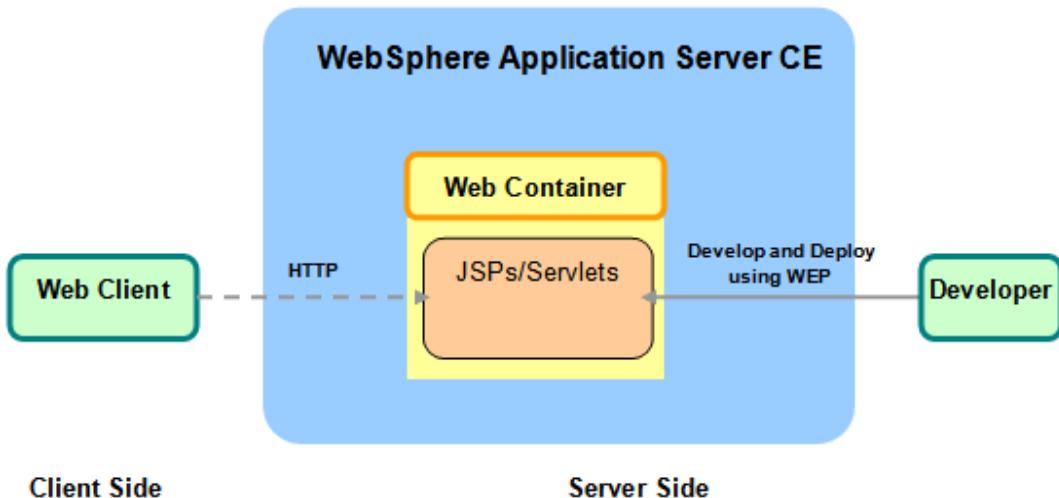


Figure 3.1 - Development with Community Edition: The big picture

3.2 Downloading and installing IBM Data Studio or Eclipse

In this book, you can work with IBM Data Studio or a plain vanilla Eclipse IDE to develop Java EE applications.

Download IBM Data Studio from ibm.com/db2/express and choose the *Data Studio IDE* package. This package will allow you to work immediately with Community Edition, and develop Java EE applications. If you are not familiar with IBM Data Studio, review the eBook *Getting started with IBM Data Studio for DB2* that is part of this book series.

If you prefer to work with a plain vanilla Eclipse, download Eclipse from <http://www.eclipse.org/downloads/> and look for the latest version. If you are not familiar with Eclipse review the ebook *Getting started with Eclipse* that is part of this book series, or visit eclipse.org for more information.

Eclipse is packaged in a variety of ways to support Java developers, PHP developers, and so on. You should download the package titled “Eclipse IDE for Java EE Developers”, which includes all the plug-ins and features you need to work in a Web development project. Be sure to select the version applicable to your operating system (For example, Windows or Linux).

After the download is completed, the installation procedure is fairly straight forward. Simply uncompress the file to whichever directory you want to install Eclipse. When the uncompress process is finished, execute Eclipse by double-clicking on the `eclipse.exe` file on Windows, or the `eclipse.bin` file on Linux. On Linux you can run the `.bin` file as follows:

```
$ chmod +x eclipse.bin  
$ ./eclipse.bin
```

When you start Eclipse, it asks for a workspace; this is where you will work and store your projects. You can take the default workspace.

The first time that you open a new workspace, Eclipse will display a welcome screen. You can close it by clicking on the X in the tab. Then you should see the workbench where you can start your work.

3.3 Eclipse Integration with WEP

In order to work with Community Edition within Eclipse, you need to download and install the WebSphere Application Server Community Edition Eclipse plug-in (WEP). If you are working with IBM Data Studio, you can skip steps 1 through 7 since WEP functionality is already included.

Follow these steps:

1. Start Eclipse
2. Go to *Help -> Software Updates*.
3. Select the *Available Software* tab and click on the *Add Site* button. Then enter this URL:
<http://download.boulder.ibm.com/ibmdl/pub/software/websphere/wasce/updates/>
4. You will see this URL added to the list. Click on the + symbol to have Eclipse search the site for updates and wait for the new features to appear. You will see the "Pending..." message as shown in *Figure 3.2* until all features are found. This can take some minutes depending on your Internet speed.

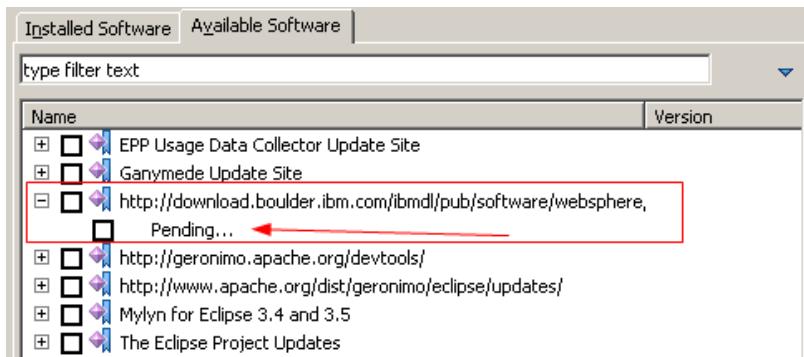


Figure 3.2 - Eclipse software updates to obtain the WEP

5. Select *WASCE v2.1 Server Adapter* as shown in *Figure 3.3*. Then click *Install*.

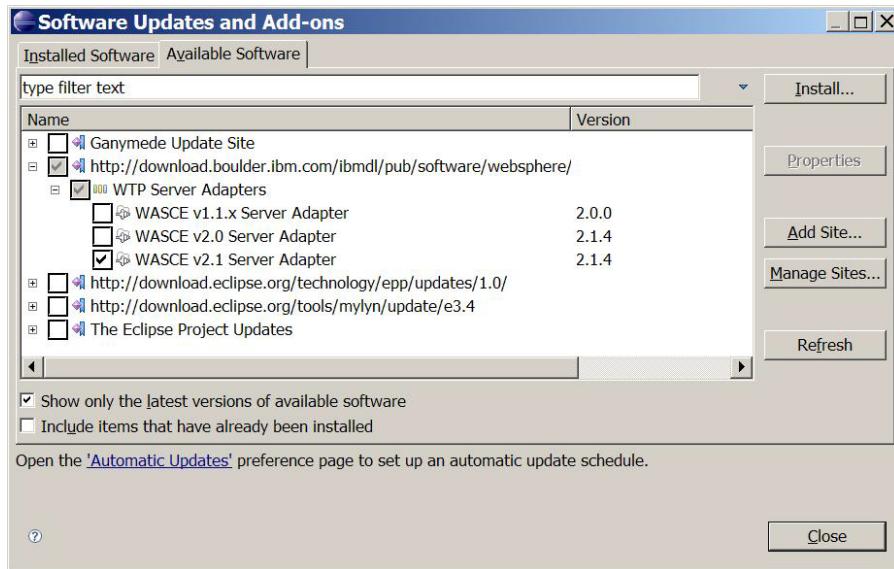


Figure 3.3 - Selecting the plugins to install into Eclipse

6. Read and accept the licenses and click *Finish*. Again, this can take a few minutes. A progress bar will appear as shown in *Figure 3.4*.

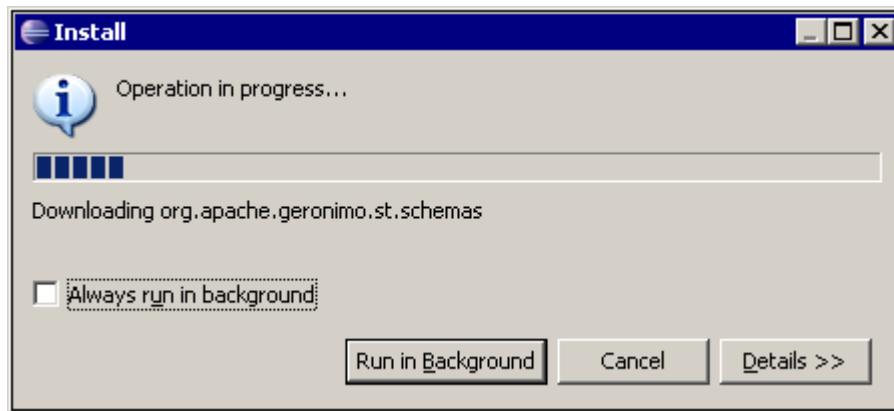


Figure 3.4 - WEP install in progress

7. After the download and install is completed, restart Eclipse.
8. Go to *Window -> Open Perspective -> Other*, select *Java EE* as shown in *Figure 3.5* and click *OK*.

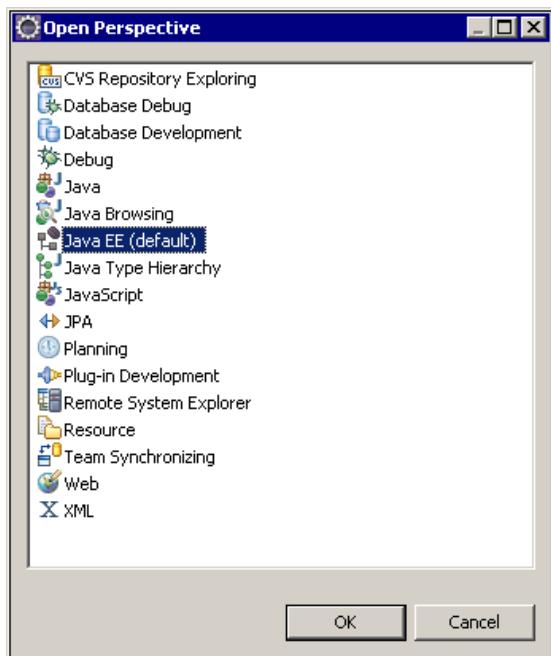


Figure 3.5 - Opening the Java EE perspective

9. Now you can see the Servers tab at the bottom of the screen. In this tab, right-click on a blank space and choose *New -> Server* as shown in *Figure 3.6*. This will add a new server to the tab. If for any reason you do not see the Servers tab, go to *Window -> Show View*, and then select the Servers view.

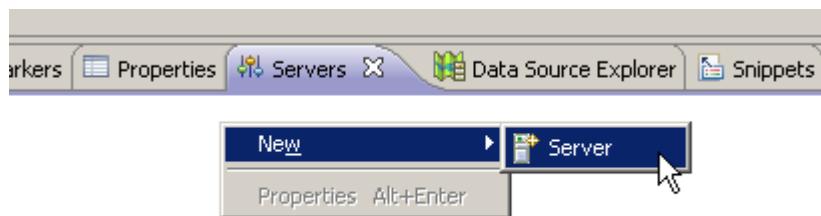


Figure 3.6 - Adding a new server

10. Select *IBM WASCE v2.1 Server* as shown in *Figure 3.7* and click *Next*.

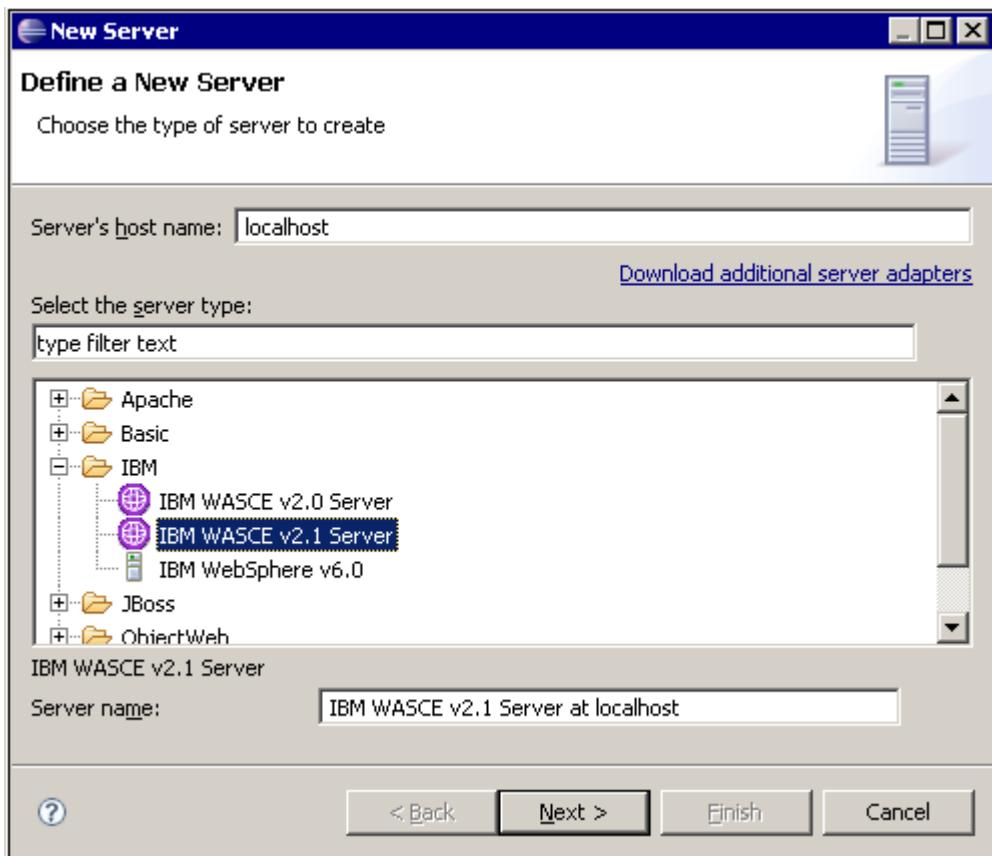


Figure 3.7 - Defining a new Community Edition server

11. Click *Browse* and specify the JRE and the location where you have installed Community Edition as shown in *Figure 3.8*. Then click *Next*.

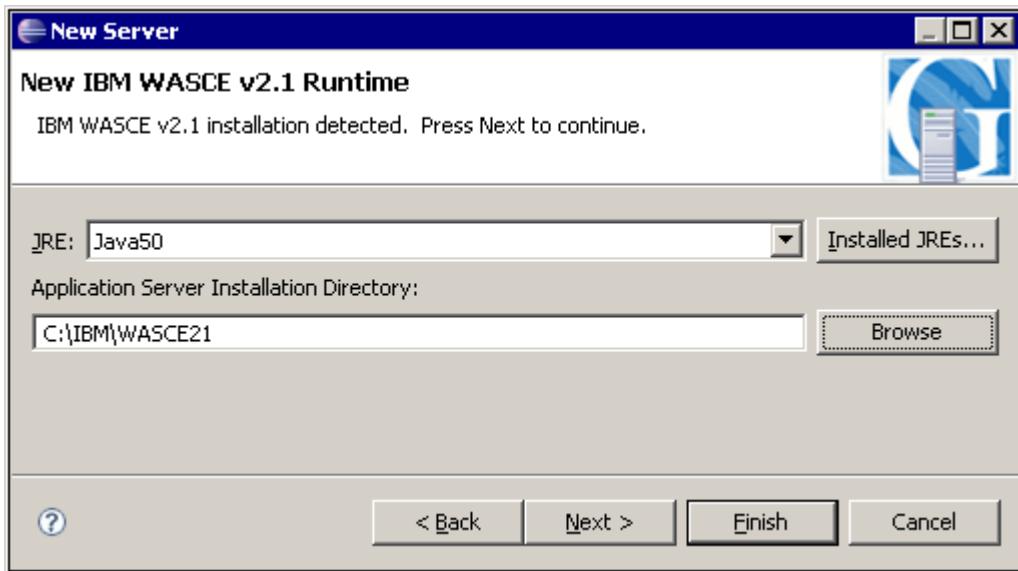


Figure 3.8 - Specifying the JRE and Community Edition installation directory

12. Set the administrator's user ID (defaults to `system`) and password (defaults to `manager`) for your Community Edition server. This is shown in *Figure 3.9* and click *Next*.

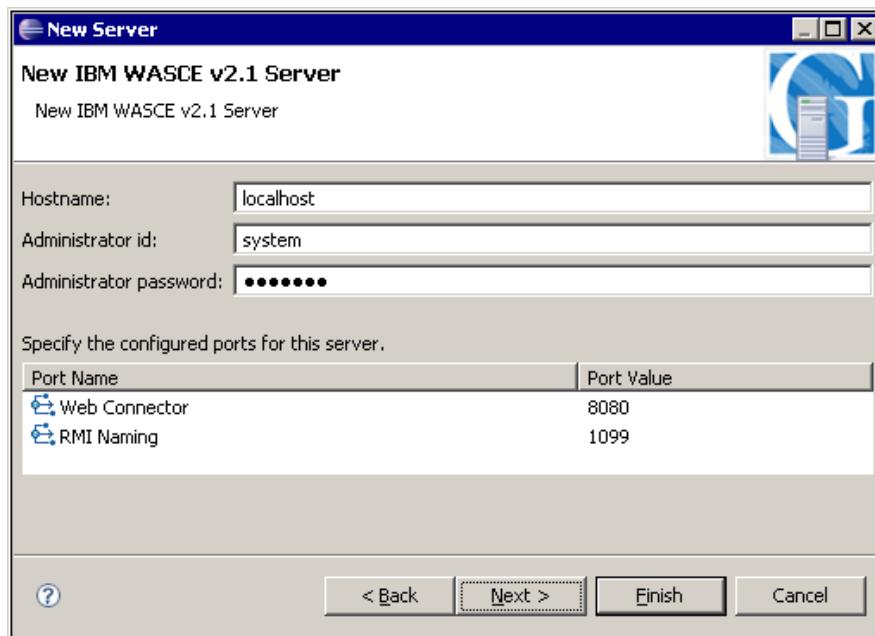


Figure 3.9 - Specifying the administrator's user ID and password

13. In the *Add and Remove Projects* window simply click *Finish*. Community Edition is now configured to work in Eclipse! You should be able to see your server in the Servers tab as shown in *Figure 3.10*.

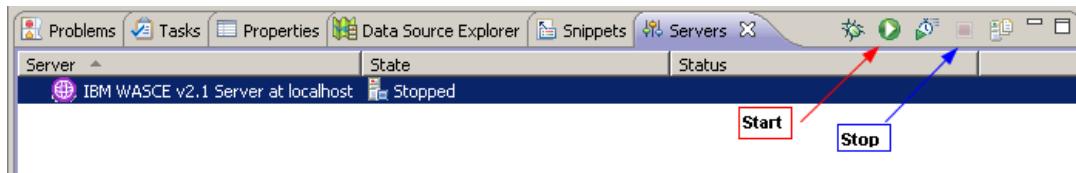


Figure 3.10 - The Community Edition server appears in the servers tab

To start and stop your server, use the appropriate buttons as shown in the above figure. After you start the server, you should see the state column change to a value of **started**. You can also start the Administration Console by opening a browser and pointing to <https://localhost:8443/console/portal/Welcome> to verify the server is up. When you stop the server the status column should change to **stopped**.

This is only one method to install WEP to Eclipse. Other methods are documented at <http://download.boulder.ibm.com/ibmdl/pub/software/websphere/wasce/updates/>

3.4 Creating and deploying a "Hello World" Web application

Now it's time to create your first Web application. You will create a Web project using HTML stored in the `index.html` file. You will not need to write any Java code for now, just HTML! Follow this procedure:

1. In Eclipse, go to *File -> New -> Dynamic Web Project*
2. As shown in *Figure 3.11*, enter the name **FirstProject**. Be sure that the *Target Runtime* field is set to your Community Edition server and click *Finish*.

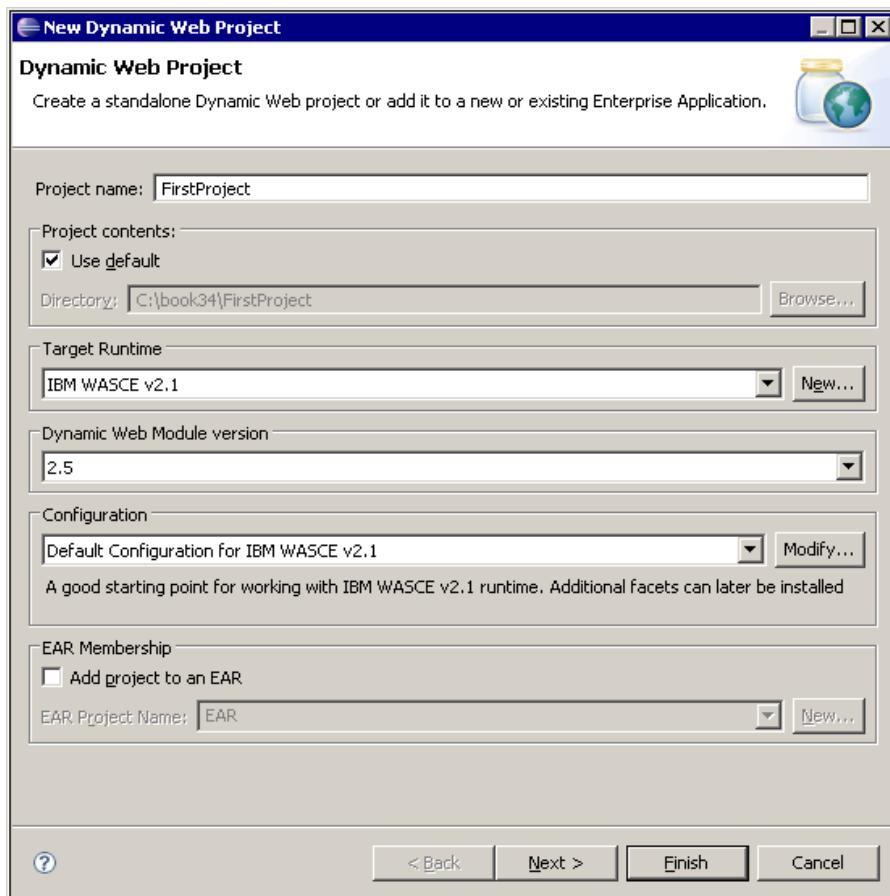


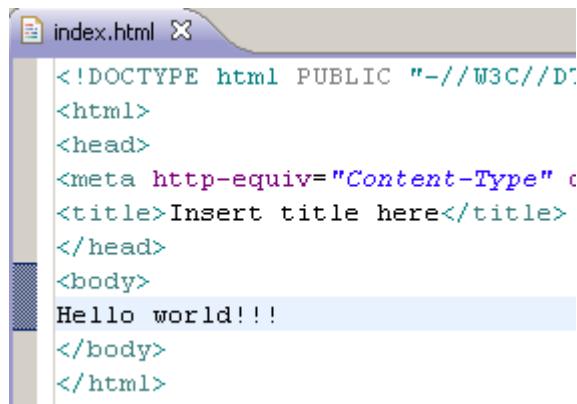
Figure 3.11 - Creating your first dynamic Web project

3. In the *Project Explorer* in Eclipse, you should see your project *FirstProject*. When you drill down, you should see the project structure as shown in *Figure 3.12*. The *src* folder will include all the Java code, and the *WebContent* folder will store all the Web pages.



Figure 3.12 - The structure of a dynamic Web project

4. To create the index.html file, right-click on **FirstProject** and choose New -> HTML. In the New HTML Page window, type index in the *Filename* field and click *Finish*.
5. An editor will appear with the file index.html opened. You can edit this file by adding the text “**Hello world!!!**” inside the body section as shown in *Figure 3.13*. Then save your page by right-clicking on the editor and choosing Save.



```
<!DOCTYPE html PUBLIC "-//W3C//D...
<html>
<head>
<meta http-equiv="Content-Type" ...
<title>Insert title here</title>
</head>
<body>
Hello world!!!
</body>
</html>
```

Figure 3.13 - Adding the text "Hello world!!!" in the index.html file

6. To deploy the application into Community Edition right-click in the application **FirstProject** in the Project Explorer, and choose *Run As -> Run on Server*.
7. Select your server as shown in *Figure 3.14* and click *Finish*.

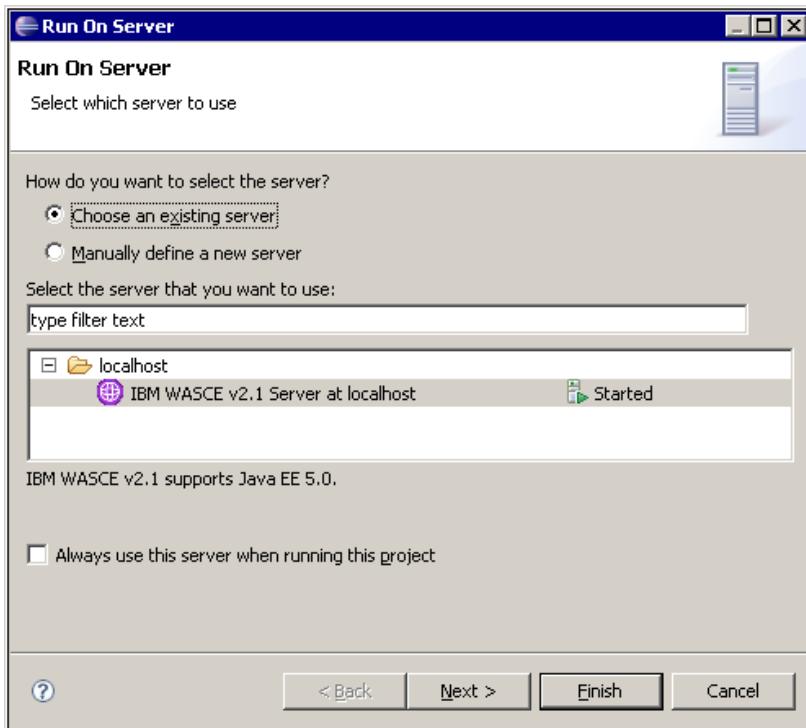


Figure 3.14 - Choosing the server where to run the application

8. You should now see your first application running as shown in *Figure 3.15*. Congratulations!

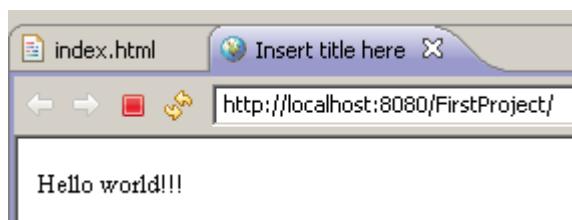


Figure 3.15 - Your first application running!

9. In the Servers tab, if you drill down on your server you should now see the deployed application *FirstProject* as illustrated in *Figure 3.16*.

Server	State	Status
IBM WASCE v2.1 Server at localhost	Started	Synchronized
FirstProject		Synchronized

Figure 3.16 - Deployed application *FirstProject*

3.5 Servlets

A servlet is a mechanism in Java to provide dynamic Web content by processing HTTP requests and returning HTTP responses in the form of HTML and XML. To create a servlet follow this procedure:

1. With your project selected, go to *File -> New -> Servlet*
2. Create a package called ***servlets*** and use the name ***HelloWorld*** as shown in *Figure 3.17*; then click *Finish*.

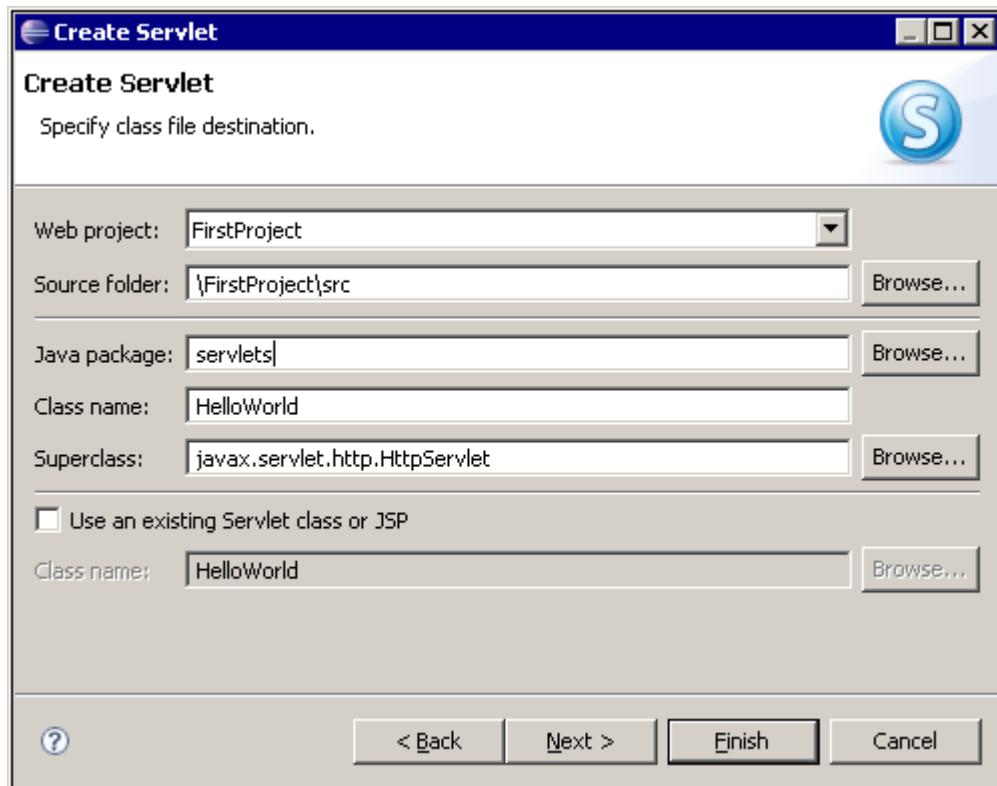


Figure 3.17 - Creating a servlet

3. An editor will be started and will open the file **HelloWorld.java** which will include a template that makes it easy to write Servlet code. Add the following lines to the **doGet** method and save it:

```

try{
    PrintWriter out = response.getWriter();
    out.println("Hello World");
}
catch(Throwable theException){
    theException.printStackTrace();
}

```

4. To run the servlet, right-click on the window and choose *Run As -> Run on Server*. Your servlet should now run as shown in *Figure 3.18* below.



Figure 3.18 - Running the servlet

5. If you go to `WebContent/WEB-INF`, and open the file `web.xml`, you should see your Servlet definition as shown below.

```
<servlet>
    <description></description>
    <display-name>HelloWorld</display-name>
    <servlet-name>HelloWorld</servlet-name>
    <servlet-class>servlets.HelloWorld</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>HelloWorld</servlet-name>
    <url-pattern>/HelloWorld</url-pattern>
</servlet-mapping>
```

3.6 JSPs

JavaServer Pages (JSPs) provide a high-level abstraction to servlets. To create a JSP file, follow this procedure:

1. Right-click your project in the Project Explorer and choose *New -> JSP*
2. For the file name use `jsppage.jsp` and click *Finish*. An editor will open with a template where you can start adding your JSP code.
3. Inside the body section enter:


```
<%>
        out.println("Hello world JSP!!!");
      %>
```
4. Right-click on the window and choose *Save*. Right-click again and choose *Run as -> Run on server*
5. Select your server and click *Finish*. You should now see your JSP running as shown in *Figure 3.19*.

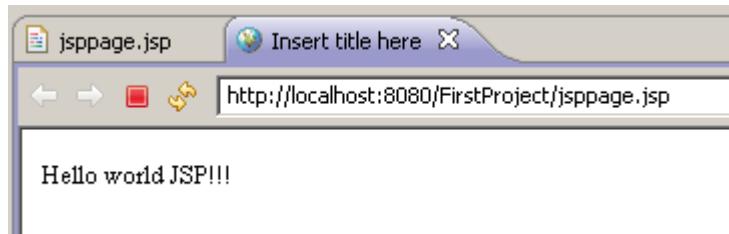


Figure 3.19 - Running the JSP

6. Thus far you have created an HTML file (`index.html`), a servlet (`HelloWorld.java`), and a JSP (`jsppage.jsp`). *Figure 3.20* shows the location of these items in the project.

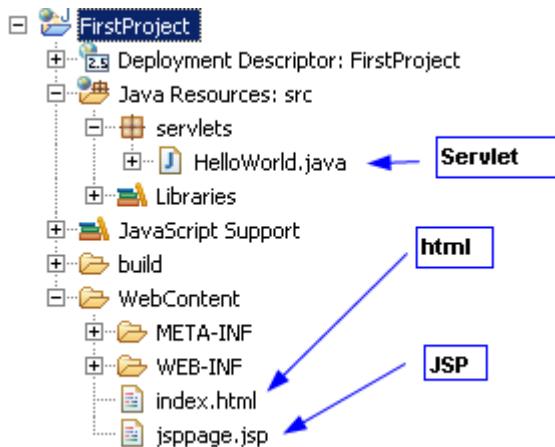


Figure 3.20 - Location of the items created for the project *FirstProject*

3.7 Manually deploying an application

Applications can be deployed directly to your Community Edition server, or they can be packaged into a Web application archive (WAR) file for later deployment at the same or other server. To create a WAR file (if your application is a simple Web application), export your project by right-clicking on the project and choosing *Export -> WAR file*. On the *WAR Export* panel, specify the destination where you will export the project. For this example you can store it in `C:\FirstProject.war` as shown in *Figure 3.21*.

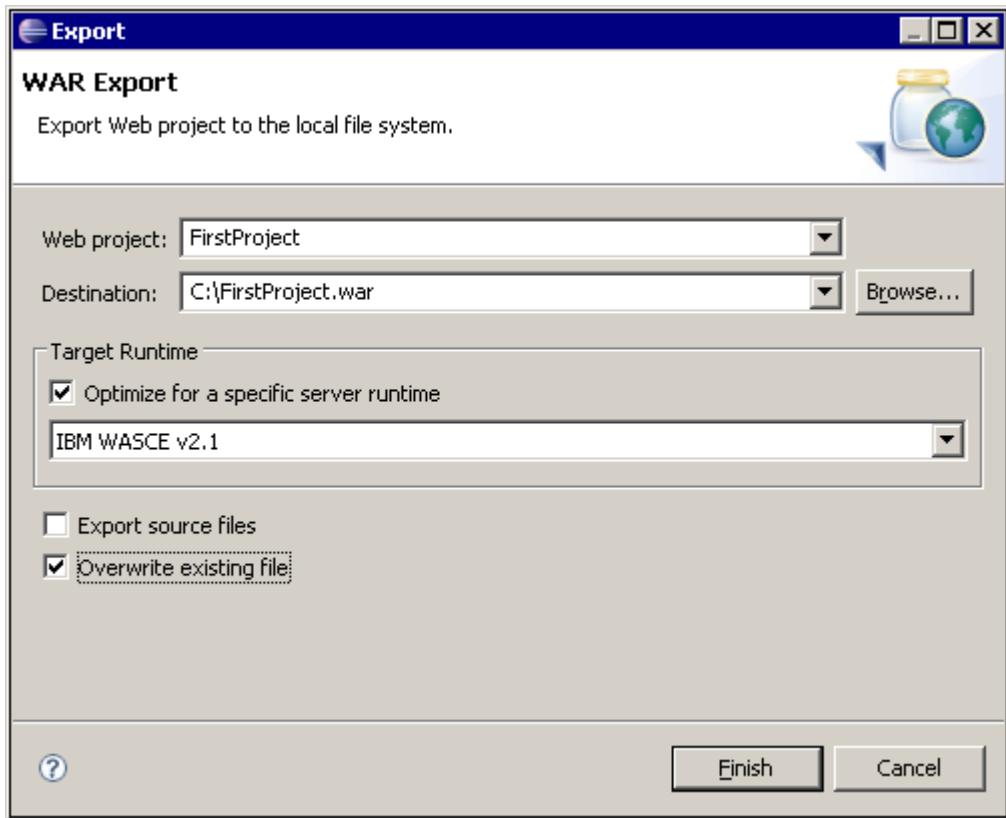


Figure 3.21 - Exporting your project to a WAR file

Now that you have your project exported to a WAR file, you can deploy it into Community Edition. Follow this procedure:

1. Open the WAS Admin Console by choosing *Start -> Programs -> IBM WebSphere -> Application Server Community Edition -> Administrative Console* (On Windows), or simply by opening a browser and pointing to <https://localhost:8443/console/portal/Welcome>.
Log in with the default user ID of **system** and the default password of **manager**.
2. In the *Console Navigation* section on the left frame, under the *Applications* folder, choose *Deploy new* as shown in *Figure 3.22*.

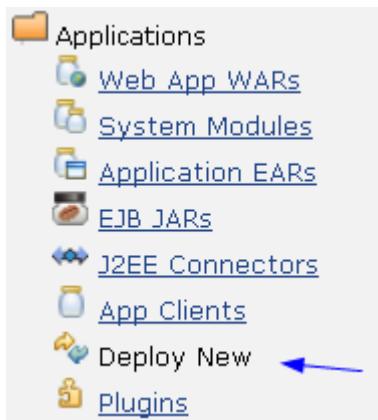


Figure 3.22 - Choosing the Deploy New option under the Applications Folder

3. In the *Install New Applications* panel, select the WAR file you created earlier and click *Install* as shown in *Figure 3.23*. You should see a message indicating the success of the operation.

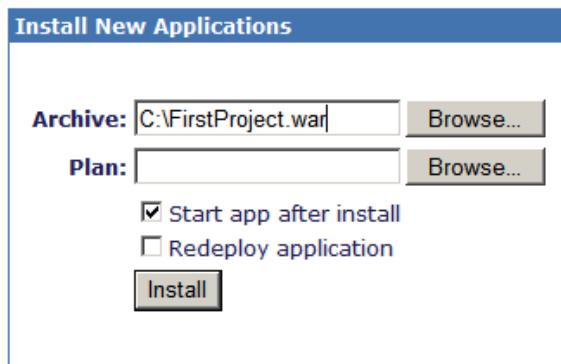


Figure 3.23 - Install New Applications panel

Note:

To get to the *Install new Applications* panel you can also choose in the *Welcome* page under the *Common Console Action* section the item *Deploy New Applications*. This is illustrated in *Figure 3.24*.



Figure 3.24 - Another way to reach the Install New Applications panel

3.8 Exercises

1. Install WEP to your Eclipse IDE.
2. Create a servlet like the "Hello world" example provided in this chapter.

3.9 Summary

In this chapter you have learned how to integrate Community Edition into Eclipse by installing the WebSphere Application Server Community Edition Eclipse plug-in (WEP) first. The chapter also explained that IBM Data Studio is an Eclipse-based IDE that includes WEP functionality and can work with Community Edition right away, in addition to providing an environment to work with DB2 databases; so it's a good alternative to plain vanilla Eclipse. Later in the chapter you learned how to develop a Java Web application with Servlets and JSPs. The chapter also discussed how to create a WAR file, and how to manually deploy it using the Community Edition Admin Console.

The next chapters will discuss how to develop more advanced applications.

3.10 Review questions

1. What is the WebSphere Application Server Community Edition Plug-in?
2. What is the benefit of using IBM Data Studio as opposed to just plain Eclipse?
3. How can you deploy a servlet into Community Edition in Eclipse?
4. After you create an application, how can you test it in Community Edition using Eclipse?
5. Is it possible to test an application developed with Eclipse and Community Edition without using Eclipse to deploy? If so, how?
6. Which version of WEP should you use for Community Edition server 2.1.1.3?
 - A. WASCE 2.1 Server adapter
 - B. WASCE 2.0 Server adapter
 - C. WASCE 1.1 Server adapter
 - D. WASCE 1.0 Server adapter
 - E. Any version
7. Which version of Eclipse should you download first before you install WEP?
 - A. Eclipse IDE for Java EE Developers
 - B. Eclipse IDE
 - C. Eclipse RCP
 - D. Eclipse SDK

E. Eclipse Runtime

8. Which are the possible installation modes of WEP?
 - A. Using the online update site
 - B. Using a local update site
 - C. Through deployable mode
 - D. With a Windows installer
 - E. Extract it to any directory you wish
9. Which operations can be performed on the Community Edition server through Eclipse?
 - A. Starting a server
 - B. Stopping a server
 - C. Deploying an artifact into the server
 - D. Installing/uninstalling a server
 - E. All of the above
10. How can you deploy a WAR into Community Edition with Eclipse?
 - A. Deploy it by WEP in Eclipse
 - B. Export a WAR and deploy it using the *Deploy New* option in the admin console
 - C. Just run the project in Eclipse
 - D. Just export the project in Eclipse
 - E. Export a WAR and double click on it

4

Chapter 4 – Working with databases

Database connectivity is a key feature of application servers as most of applications need to work with data. Community Edition provides developers with convenient functions to connect to various databases and execute SQL statements using Java Database Connectivity (JDBC). This chapter uses DB2 Express-C, the free version of DB2, as the database provider.

In this chapter you will learn about:

- How to configure a database pool
- How to query data in a Web application

4.1 Community Edition database connectivity: The big picture

Figure 4.1 provides an overview of the process followed to access a database.

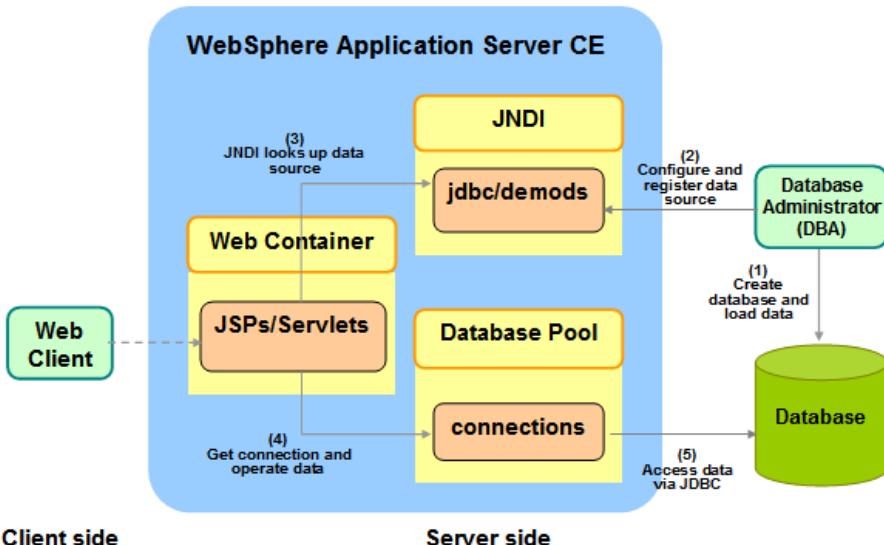


Figure 4.1 - Database connectivity: The big picture

On the right side in (1), a Database Administrator (DBA) creates a database and loads it with data. Next, in (2) the DBA configures and registers this data source in the Community Edition Java Naming and Directory Interface (JNDI) namespace. A developer develops the JSPs/Servlets that will be accessing the database through JDBC. On the left side, when the Web client requests some page that requires a database access, the corresponding JSP/servlet in (3) will first perform a JNDI look up to find the registered data source, and will then create a connection object to get database access as depicted in (4). The connection objects are pooled by the Community Edition server for application reuse, and the data can be accessed through JDBC in (5).

4.2 Steps to work with a database in Community Edition

To develop a Web application with database access in Community Edition, the typical steps are:

1. Create a database.
2. Configure and deploy a database pool via Community Edition administrative console.
3. Execute SQL statements to load some data through the database pool.

Though step 3 could be done directly with the tools provided by the database, we chose to perform it through the database pool configured in Community Edition.

4.2.1 Creating a database

In this book, we use DB2 Express-C 9.7. To download the latest version of DB2 Express-C, visit ibm.com/db2/express and choose the appropriate file to download for the operating system you are using. If you are not familiar with DB2, refer to *Appendix B* or review the eBook *Getting started with DB2 Express-C* which is part of this book series.

After downloading DB2 Express-C, you need to install it. If you don't have a server available, you can always work with DB2 and Community Edition on the Cloud. As mentioned in *Chapter 2*, there is an all-in-one template that contains DB2 and Community Edition available on RightScale for deployment on the Amazon EC2 cloud at <https://my.RightScale.com>.

If you do have a server where you have installed Community Edition and DB2, create a database in two ways:

- Using a graphical tool like **IBM Data Studio**
- Using a command line interface like the **DB2 Command Line Processor (CLP)** included with DB2.

We will use the CLP to create a database called TESTDB as follows:

1. On Windows, open a command prompt and type `db2cmd`. This will open the CLP.
On Linux, log on to the DB2 server as the DB2 instance owner (normally `db2inst1`), and open a terminal window.
2. Input this statement to create the DB2 database TESTDB as shown in *Figure 4.2*. Creating the database takes a few minutes to complete.

```
db2 create db testdb using codeset utf-8 territory US
```

```
DB2 CLP - DB2COPY1
C:\Program Files\IBM\SQLLIB\BIN>db2 create db testdb using codeset UTF-8 territory US
DB20000I The CREATE DATABASE command completed successfully.
C:\Program Files\IBM\SQLLIB\BIN>
```

Figure 4.2 - Creating the DB2 database TESTDB from the CLP

You now have a DB2 database ready for use.

Note:

DB2 provides a sample database called SAMPLE. This database is normally created by default after DB2 installation; therefore, you could use this database for the exercises in this book too.

4.2.2 Configuring and deploying a database pool

Community Edition provides a user interface to configure a database pool step by step. A typical database pool configuration includes these steps:

1. Install the database vendor JDBC driver
2. Configure and deploy the database pool

4.2.2.1 Installing the database vendor JDBC driver

JDBC is a set of APIs for Java that define how a client accesses relational databases. Most database vendors provide JDBC drivers for their databases. There are different drivers supported in DB2 as shown in *Table 4.1* below.

Driver Type	Driver Name	Packaged as	JDBC specification supported	Minimum level of SDK for Java required
Type 2	DB2 JDBC Type 2 Driver for Linux, UNIX and Windows (Deprecated*)	db2java.zip	JDBC 1.2 and JDBC 2.0	1.4.2

Type 2	IBM Data Server Driver and for JDBC and SQLJ	db2jcc.jar and sqlj.zip	JDBC 3.0 compliant	1.4.2
Type 4		db2jcc4.jar and sqlj4.zip	JDBC 4.0 and earlier	6

Table 4.1 - DB2 JDBC and SQLJ drivers

* Deprecated means it is still supported, but no longer enhanced

As you can see from *Table 4.1*, Type 2 is provided with two different drivers; however the DB2 JDBC Type 2 Driver for Linux, UNIX and Windows, packaged as `db2java.zip`, is deprecated.

The IBM Data Server Driver for JDBC and SQLJ packaged as `db2jcc.jar` (`com.ibm.db2.jcc`) includes support for both, the type 2 and type 4 drivers. The choice of driver is determined based on the syntax used to connect to the database in your Java program: If a hostname or IP address, and a port are included in the connection string, then type 4 is used, otherwise, type 2 is used. The IBM Data Server Driver for JDBC and SQLJ has been optimized to access all DB2 servers in all platforms including the mainframe.

When you install a DB2 server, a DB2 client or the IBM Data Server Driver for JDBC and SQLJ, the `db2jcc.jar` and `sqlj.zip` files compliant with JDBC 3.0 are automatically added to your `CLASSPATH`. If you would like to use the JDBC 4.0 specification, make sure to replace `db2jcc.jar` and `sqlj.zip` with `db2jcc4.jar` and `sqlj4.zip` respectively in the `CLASSPATH`.

Note:

For more information about JDBC drivers, refer to the eBook *Getting started with DB2 Express-C*. If you are new to the Java programming language and the Java EE framework, review the eBook *Getting Started with Java*. Both books are part of this book series.

In this book we use the JDBC 4 compliant driver packaged in file `db2jcc4.jar` that is included with DB2 Express-C under `C:\Program Files\IBM\SQLLIB\Java` on Windows or `/opt/ibm/db2/V9.7/java` on Linux. Note that `db2jcc4.jar` requires Java 6.

To install the JDBC driver in Community Edition follow these steps:

1. Start the Community Edition server if it is not started yet. Open a browser and log in to the Administrative Console using the user ID of `system` and the password of `manager`: <https://localhost:8443/console>
2. In the *Console Navigation* section on the left panel, under the *Services* folder, click on *Repository* to open the repository viewer portlet as shown in *Figure 4.3*.



Figure 4.3 - The repository viewer

3. At the top of the repository viewer, you can add new archives, and in the lower section you can view the current repository entries. Community Edition bundles several JDBC drivers for different DB2 versions. You can choose one according to your DB2 version. If there is no repository entry for DB2 9.7 (an entry for com.ibm.db2/db2jcc4/9.7/jar), you can add it as explained in the next steps.
4. Click the *Browse* button to locate the JDBC driver db2jcc4.jar in your DB2 installation directory (for example, C:\Program Files\IBM\SQLLIB\java on Windows). Then input a Group ID, Artifact ID, Version, and Type information as shown in *Figure 4.4*.

Add Archive to Repository

File: C:\Program Files\IBMSQI

Group: com.ibm.db2

Artifact: db2jcc4

Version: 9.7

Type: jar

Figure 4.4 - Adding an archive to the repository

5. Click the *Install* button and check if the JDBC driver is now listed in the repository as illustrated in *Figure 4.5* below.

Current Repository Entries

Click on an entry to view usage.

- [annogen/annogen/0.1.0/jar](#)
- [asm/asm-commons/3.1/jar](#)
- [asm/asm/3.1/jar](#)
- [aspectj/aspectjrt/1.5.3/jar](#)
- [axis/axis/1.4/jar](#)
- [backport-util-concurrent/backport-util-concurrent/2.2/jar](#)
- [cglib/cglib-nodep/2.1_3/jar](#)
- [com.ibm.db2/db2jcc/8.2/jar](#)
- [com.ibm.db2/db2jcc/9.1/jar](#)
- [com.ibm.db2/db2jcc/9.5/jar](#)
- [com.ibm.db2/db2jcc4/9.7/jar](#)
- [com.ibm.db2/db2jcc_license_cu/8.2/jar](#)
- [com.ibm.db2/db2jcc_license_cu/9.1/jar](#)
- [com.ibm.db2/db2jcc_license_cu/9.5/jar](#)

Figure 4.5 - Verifying an entry for db2jcc4.jar has been added

6. Now that you have installed the JDBC driver, you need to configure a database pool to use it as described in the next section.

4.2.2.2 Configuring and deploying a database pool

A database connection pool is a cache of database connections handled by the application server so that the connections can be reused when the Web application needs a database connection in future.

To configure and deploy a database pool from the Admin console, follow these steps:

1. On the left panel under the *Services* folder, click on *Database Pools*. Then click *Using the Geronimo database pool wizard* to start to create a DB2 database pool. This is illustrated in *Figure 4.6* below.

Name	Deployed As	State	Action
MonitoringClientDS	Server-wide	running	Edit Usage
NoTxDatasource	Server-wide	running	Edit Usage
SystemDatasource	Server-wide	running	Edit Usage
jdbc/ActiveDS	Server-wide	running	Edit Usage
jdbc/ArchiveDS	Server-wide	running	Edit Usage
jdbc/juddiDB	org.apache.geronimo.configs/uddi-tomcat/2.1.4/car	running	Edit Usage

Figure 4.6 - Creating a new database pool

2. Let's input *jdbc/demods* as the database pool name (where *demods* is short for demo dataset); and for the database type dropdown list, choose *DB2 XA* as shown in *Figure 4.7*. Then click *Next*.

Create Database Pool -- Step 1: Select Name and Database

Name of Database Pool:

A name that is different than the name for any other database pool (no spaces in the name please).

Database Type:

The type of database the pool will connect to.

Figure 4.7 - Creating a new database pool - Step 1

3. On the next page, specify the driver JAR, database name, user name, password, and server name as illustrated in *Figure 4.8* and *Figure 4.9*. For the rest of the fields, accept the defaults.

This page edits a new or existing database pool.

Pool Name: A name that is different than the name for any other database pools in the server (no spaces in the name please).

Pool Type: *TransQ XA Resource Adapter for DB2*
A resource adaptor that provides access to a DB2 database with XA transaction support. The following properties were taken from the DB2 JCC Driver Documentation located at <http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.udb.doc/ad/cjvjcsp.htm>

Basic Connection Properties	
Driver JAR:	<input type="text" value="backport-util-concurrent/backport-util-concurrent/2.2.jar"/> <input type="text" value="cglib/cglib-nodep/2.1_3.jar"/> <input type="text" value="com.ibm.db2/db2jcc/8.2.jar"/> <input type="text" value="com.ibm.db2/db2jcc/9.1.jar"/> <input type="text" value="com.ibm.db2/db2jcc/9.5.jar"/> com.ibm.db2/db2jcc4/9.7.jar <input type="text" value="com.ibm.db2/db2jcc_license_cu/8.2.jar"/> <input type="text" value="com.ibm.db2/db2jcc_license_cu/9.1.jar"/>

Figure 4.8 - Creating a new database pool - Step 2a

Database Name: Specifies the name for the database.

User Name: The name of the userID used to connect.

Password:

Confirm Password:

Server Name: ServerName is the name / IP address of the remote server.

Port Number: Specifies the port number the remote server uses.

Driver Type:

Figure 4.9 - Creating a new database pool - Step 2b

Community Edition supports many database vendors. For different database vendors, you may need to input different information.

- Click *Deploy* to deploy this database pool. You can now see it deployed and running as shown in *Figure 4.10*.

Name	Deployed As	State	Actions
MonitoringClientDS	Server-wide	running	Edit Usage Delete
NoTxDatasource	Server-wide	running	Edit Usage Delete
SystemDatasource	Server-wide	running	Edit Usage Delete
jdbc/ActiveDS	Server-wide	running	Edit Usage Delete
jdbc/ArchiveDS	Server-wide	running	Edit Usage Delete
jdbc/demods	Server-wide	running	Edit Usage Delete
jdbc/juddiDB	org.apache.geronimo.configs/uddi-tomcat/2.1.4/car	running	Edit Usage Delete

Create a new database pool:

- [Using the Geronimo database pool wizard](#)

Figure 4.10 - Deployed database pool

Note:

You can configure a database pool programmatically using a script by invoking within this script `deploy.bat` (on Windows) or `deploy.sh` (on Linux).

4.2.3 Executing SQL statements to load some data

Using the database pool created in the previous step (which will connect you to the TESTDB database), create some tables and load them with data using SQL statements. This is also done to test that the database pool is working correctly. The SQL statements to use are shown in *Listing 4.1*. Note that the last column of the table is defined as an XML column.

```
create table employees(empID INT, empName VARCHAR(20), empDOC XML);

insert into employees(empID, empName, empDOC) values (1, 'Tom',
    '<employee id="1">
        <firstname>Tom</firstname>
        <lastname>Clinton</lastname>
        <age>26</age>
    </employee>');

insert into employees(empID, empName, empDOC) values (2, 'John',
    '<employee id="2">
        <firstname>John</firstname>
        <lastname>Park</lastname>
        <age>28</age>
    </employee>');

insert into employees(empID, empName, empDOC) values (3, 'Jerry',
    '<employee id="3">
        <firstname>Jerry</firstname>
        <lastname>Lee</lastname>
        <age>25</age>
    </employee>');
```

```
</employee>' );
```

Listing 4.1 - SQL statements to run in the new database pool deployed

To execute the above statements, follow these steps:

1. From the *Console Navigation* under the *Services* folder, click *Database Pools*. In the *Run SQL* portlet at the bottom of your screen (you may have to scroll down), input the created data source *jdbc/demods* in the *Use DataSource* drop down list. Then input the SQL statements in *Listing 4.1* above and click the *Run SQL* button as shown in *Figure 4.11*.

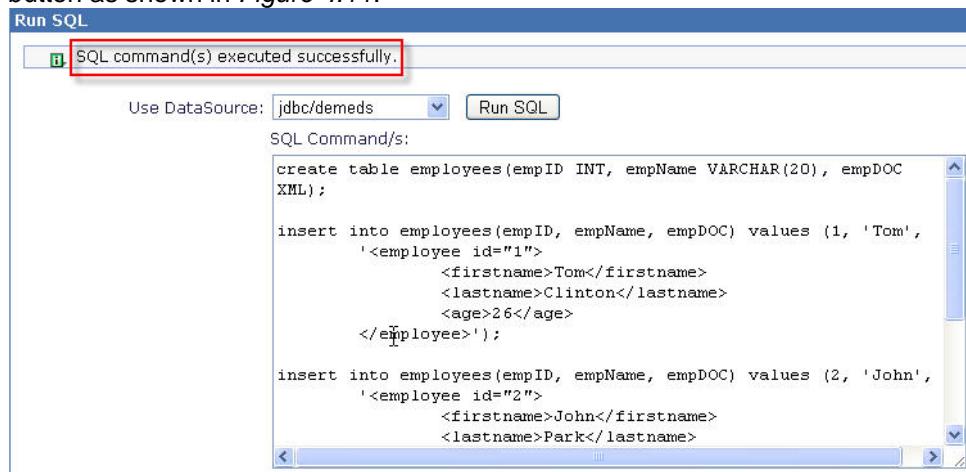


Figure 4.11 - Running SQL using the Run SQL portlet

You should see the “SQL command(s) executed successfully” message. If you are using Internet Explorer there is a known issue where there is no response after clicking the *Run SQL* button. If that's the case, please try another browser.

2. Verify the data was inserted by issuing a `select * from employees` statement. You should receive an output as shown in *Figure 4.12*.

The screenshot shows a SQL command-line interface. At the top, a message says "SQL command(s) executed successfully." Below it, there's a dropdown menu labeled "Use DataSource: jdbc/demods" and a "Run SQL" button. A text area labeled "SQL Command/s:" contains the query "select * from employees". Below this, a note provides instructions: "1) Use ';' to separate multiple statements", "2) Query results will be displayed for single 'Select' statement", and "3) Use single quotes to encapsulate literal strings". The results are presented in a table titled "Query Result" with columns "EMPID", "EMPNAME", and "EMPDOC". The data shows three rows:

EMPID	EMPNAME	EMPDOC
1	Tom	com.ibm.db2.jcc.am.ke@aa4a14
2	John	com.ibm.db2.jcc.am.ke@1e57ae0
3	Jerry	com.ibm.db2.jcc.am.ke@e64498

Figure 4.12 - Output of the SELECT statement

4.3 Developing a Web application to access a database

To access a database using a Web application, follow these steps:

1. Create a *Dynamic Web Project* in Eclipse
2. Create a JSP with code to access data, and configure the data source reference in the Web project configuration files
3. Deploy and run the project on the Community Edition server

4.3.1 Create a Dynamic Web Project

As discussed in Chapter 3, to create a dynamic Web project follow these steps:

1. On Eclipse go to *File->New->Dynamic Web Project* as shown in *Figure 4.13*.

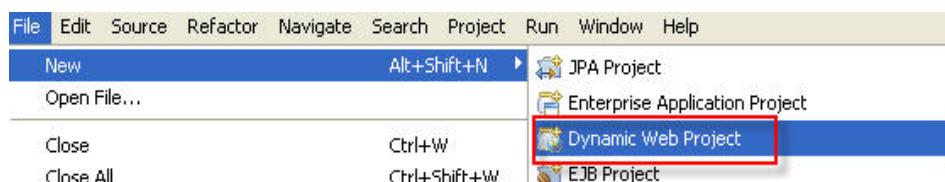


Figure 4.13 – New a Web Dynamic Project

2. Input a project name such as *chapter4-databases-dbdemo* and choose a project location as shown in *Figure 4.14*. Ensure *IBM WASCE v2.1* is the target runtime. Then click *Finish*.



Figure 4.14 – Input project name and location

3. The project that is created is shown in *Figure 4.15*.

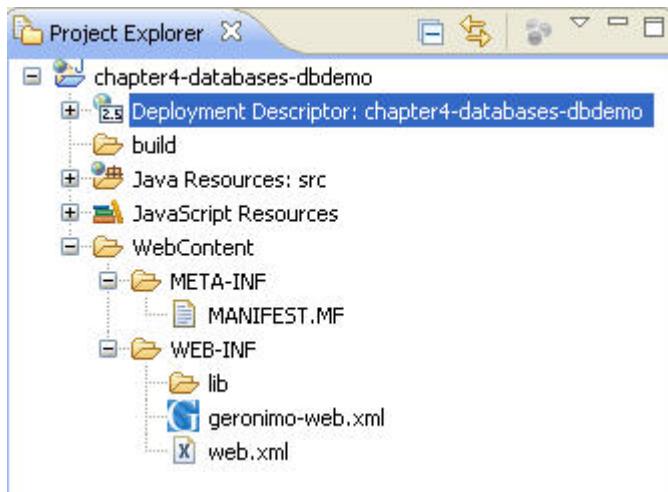


Figure 4.15 – Generated project layout

Because you chose *IBM WASCE v2.1* as the target runtime, a Community Edition configuration file `geronimo-web.xml` will be created in the `WEB-INF` folder. This folder will be used for the Web project to define the Web context as well as some resource references in the later sections.

4.3.2 Create a JSP and configure the data source reference

In a Java EE programming environment, to access data in a database you normally need to write code that performs the following:

1. Initialize a JNDI naming context and look up the JDBC data source
2. Get the database connection, execute SQL statements, and present the data
3. Close the database connection

Let's create a JSP file to implement the database access logic:

1. In Eclipse, right click on the `WebContent` folder, then *New -> JSP* to create a JSP file as shown in *Figure 4.16*.

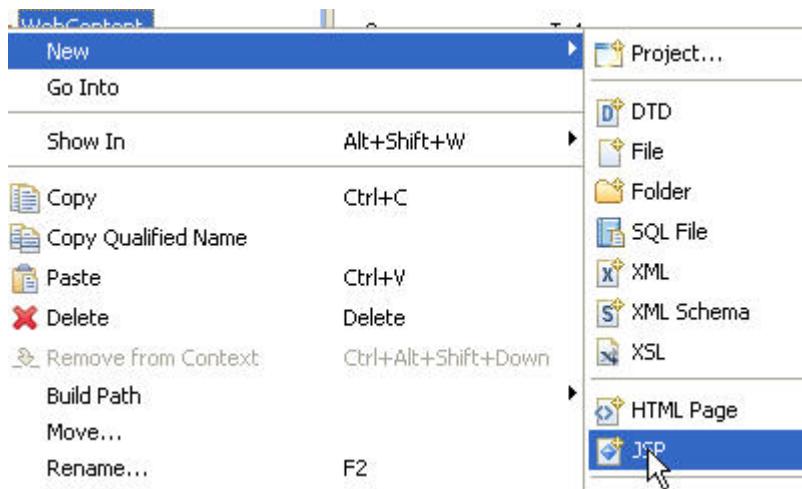


Figure 4.16 - New a JSP file

In the pop up window, input the JSP file name (dbdemo.jsp) as shown in *Figure 4.17* and click *Finish*.

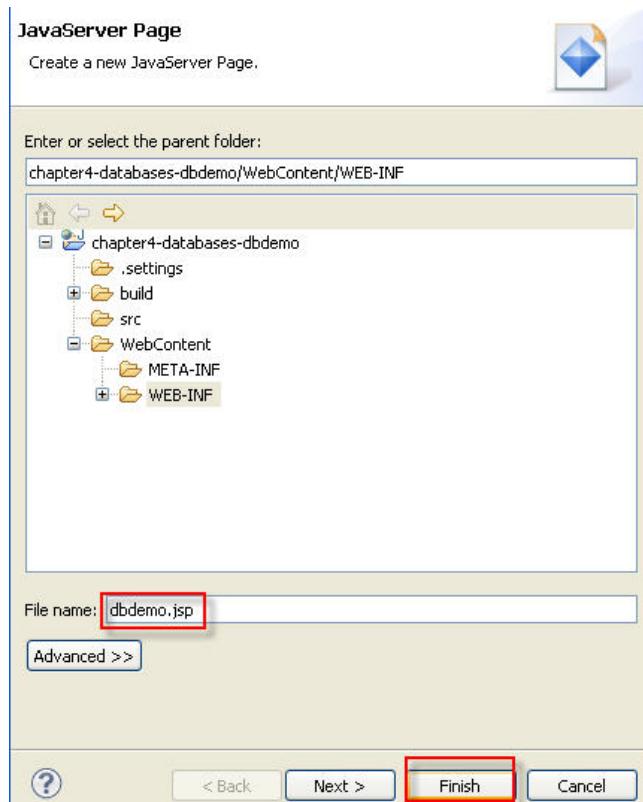


Figure 4.17 – Input the JSP file name

2. An editor showing the new JSP file appears as shown in *Figure 4.18*. Change the title text to **DBDemo**.



```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
</body>
</html>
```

Figure 4.18 – The generated JSP file

3. In the JSP file, follow these steps to access the data in the database:
- Import some necessary database access java classes as shown in *Listing 4.2*.

```
<%@ page import="javax.sql.*">
<%@ page import="java.sql.*">
<%@ page import="javax.naming.*">
```

Listing 4.2 – Import the necessary java classes for database access

- Initialize a naming context to look up a data source as shown in *Listing 4.3*. In the lookup string, `java:comp/env/` is the common prefix of the naming space, `jdbc/DataSource` is the logical name of the defined data source.

```
Context initContext = new InitialContext();
DataSource ds =
    (DataSource)initContext.lookup("java:comp/env/jdbc/DataSource");
Listing 4.3 - Initializing a naming context and looking up a datasource
```

- Open the Web project descriptor file WEB-INF/web.xml, define the logical name as shown in *Listing 4.4*.

```
<resource-ref>
    <res-ref-name>jdbc/DataSource</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
    <res-sharing-scope>Shareable</res-sharing-scope>
</resource-ref>
```

Listing 4.4 - The logical name defined in web.xml

Then open WEB-INF/geronimo-web.xml, add some configurations to map the logical name to the physical name `jdbc/demods` as shown in *Listing 4.5*. The

physical name is the one used in *Section 4.2.2.2*. The mapping makes application deployment flexible.

```
<name:resource-ref>
    <name:ref-name>jdbc/DataSource</name:ref-name>
    <name:resource-link>jdbc/demods</name:resource-link>
</name:resource-ref>
```

Listing 4.5 - Mapping the logical name to the physical name in geronimo-web.xml

- d) Back to the JSP file, add some code to get a database connection and then retrieve data from the database. This is shown in *Listing 4.6*. There are inline comments to explain the code.

```
Connection con = null;
Statement stmt = null;

// Get a database connection
con = ds.getConnection();

// Prepare a statement object used to execute query
stmt = con.createStatement();

String sql = "SELECT empID, empName FROM employees " +
            "WHERE XMLEXISTS('$i/employee[age > \"25\"]' " +
            "PASSING empDOC AS \"i\")";

// Fill returned data into ResultSet object.
ResultSet rs = stmt.executeQuery(sql);
```

Listing 4.6 - Code snippet to get database connection and query data

- e) After you retrieve data and store it in a *ResultSet* object, you need to present them properly using HTML as shown in *Listing 4.7*.

Employees who are older than 25 years old are:

```
<BR>
<table>
    <tr>
        <th>Employee ID</th>
        <th>Employee Name</th>
    </tr>
    <%
        while (rs.next()) {
            int empid = rs.getInt("empID");
            String empname = rs.getString("empName");

    %>
```

```

<tr>
    <td><%=empid%></td>
    <td><%=empname%></td>
</tr>
<%
}
%
</table>

```

Listing 4.7 HTML code snippet to present the retrieved data

- f) Finally, don't forget to close the database access objects in your code, so that the database connection object is returned to the connection pool, and is available for other clients to call.

```

<%
    rs.close();
    stmt.close();
    con.close();
}
catch(java.lang.Exception e) {
    e.printStackTrace();
    System.err.print(e.getClass().getName());
    System.err.println(e.getMessage());
}
%

```

Listing 4.8 – Code snippet to close database objects.

The complete JSP code is shown in *Listing 4.9*.

```

<!-- 2009 IBM Copyright Reserved. -->
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>DBDemo</title>
</head>
<body>
<%@ page import="javax.sql.*"%>
<%@ page import="java.sql.*"%>
<%@ page import="javax.naming.*"%>
<%

```

```
Connection con = null;
Statement stmt = null;

try {
    Context initContext = new InitialContext();
    DataSource ds =
(DataSource)initContext.lookup("java:comp/env/jdbc/DataSource");
    // Get a database connection
    con = ds.getConnection();
} catch(java.lang.Exception e) {
    e.printStackTrace();
    System.err.print(e.getClass().getName());
    System.err.println(e.getMessage());
}

try {
    // Prepare a statement object used to execute query
    stmt = con.createStatement();
    String sql = "SELECT empID, empName FROM employees " +
"WHERE XMLEXISTS('$i/employee[age > \"25\"]' PASSING empDOC AS \"i\")";

    // Fill returned data into ResultSet object.
    ResultSet rs = stmt.executeQuery(sql);
%>
The employees who are older than 25 years old are:
<BR>
<table>
    <tr>
        <th>Employee ID</th>
        <th>Employee Name</th>
    </tr>
    <%
        while (rs.next()) {
            int empid = rs.getInt("empID");
            String empname = rs.getString("empName");
%
        <tr>
            <td><%=empid%></td>
            <td><%=empname%></td>
        </tr>
        <%
    }
%
%>
```

```

</table>
<%
    rs.close();
    stmt.close();
    con.close();
}
catch(java.lang.Exception e) {
    e.printStackTrace();
    System.err.print(e.getClass().getName());
    System.err.println(e.getMessage());
}
%>
</body>
</html>

```

Listing 4.9 – Complete JSP code

Next, let's see how to run this project in the Community Edition server.

4.3.3 Deploy and run the project on the Community Edition server

To run this project in Community Edition, follow these steps:

1. Complete the Web project configurations.

Add a welcome page definition in WEB-INF/web.xml. The complete file content is shown in *Listing 4.10*.

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID"
  version="2.5">
  <display-name>chapter4-databases-dbdemo</display-name>
  <resource-ref>
    <res-ref-name>jdbc/DataSource</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
    <res-sharing-scope>Shareable</res-sharing-scope>
  </resource-ref>
  <welcome-file-list>
    <welcome-file>dbdemo.jsp</welcome-file>
  </welcome-file-list>

```

```
</web-app>
```

Listing 4.10 – Full content of WEB-INF/web.xml

Add an environment dependency to the data source in WEB-INF/geronimo-web.xml. The dependency definition tells Community Edition where to find the data source physical definition to map the logical name for the application. The full file content is as shown in *Listing 4.11*.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<web:web-app
  xmlns:app="http://geronimo.apache.org/xml/ns/j2ee/application-2.0"
  xmlns:client="http://geronimo.apache.org/xml/ns/j2ee/application-client-2.0"
  xmlns:conn="http://geronimo.apache.org/xml/ns/j2ee/connector-1.2"
  xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2"
  xmlns:ejb="http://openejb.apache.org/xml/ns/openejb-jar-2.2"
  xmlns:name="http://geronimo.apache.org/xml/ns/naming-1.2"
  xmlns:pers="http://java.sun.com/xml/ns/persistence"
  xmlns:pkgen="http://openejb.apache.org/xml/ns/pkgen-2.1"
  xmlns:sec="http://geronimo.apache.org/xml/ns/security-2.0"
  xmlns:web="http://geronimo.apache.org/xml/ns/j2ee/web-2.0.1">
  <dep:environment>
    <dep:moduleId>
      <dep:groupId>default</dep:groupId>
      <dep:artifactId>chapter4-databases-dbdemo</dep:artifactId>
      <dep:version>1.0</dep:version>
      <dep:type>car</dep:type>
    </dep:moduleId>
    <dep:dependencies>
      <dep:dependency>
        <dep:groupId>console.dbpool</dep:groupId>
        <dep:artifactId>jdbc_demo</dep:artifactId>
      </dep:dependency>
    </dep:dependencies>
    <dep:hidden-classes/>
    <dep:non-overridable-classes/>
  </dep:environment>
  <web:context-root>/chapter4-databases-dbdemo</web:context-root>
  <name:resource-ref>
    <name:ref-name>jdbc/DataSource</name:ref-name>
    <name:resource-link>jdbc/demods</name:resource-link>
  </name:resource-ref>
</web:web-app>
```

Listing 4.11 – Full content of WEB-INF/geronimo-web.xml

2. Define a server runtime in Eclipse as described in *Chapter 3*.

3. Right click on the project name, click *Run As* -> *Run on Server* as shown in *Figure 4.19*.

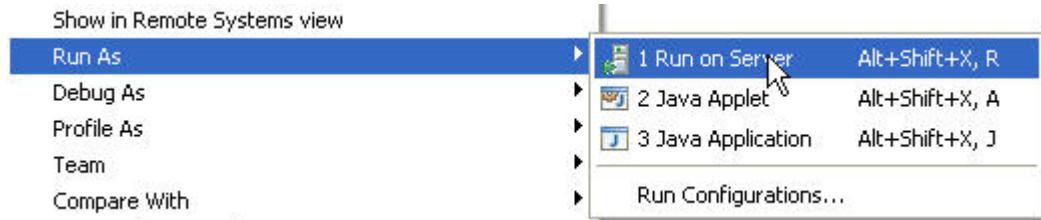


Figure 4.19 – Run on Server menu

4. On the *Run on Server* window, choose the existing server **IBM WASCE v2.1 Server at localhost**, then click *Finish* to start the deployment as shown in *Figure 4.20*.

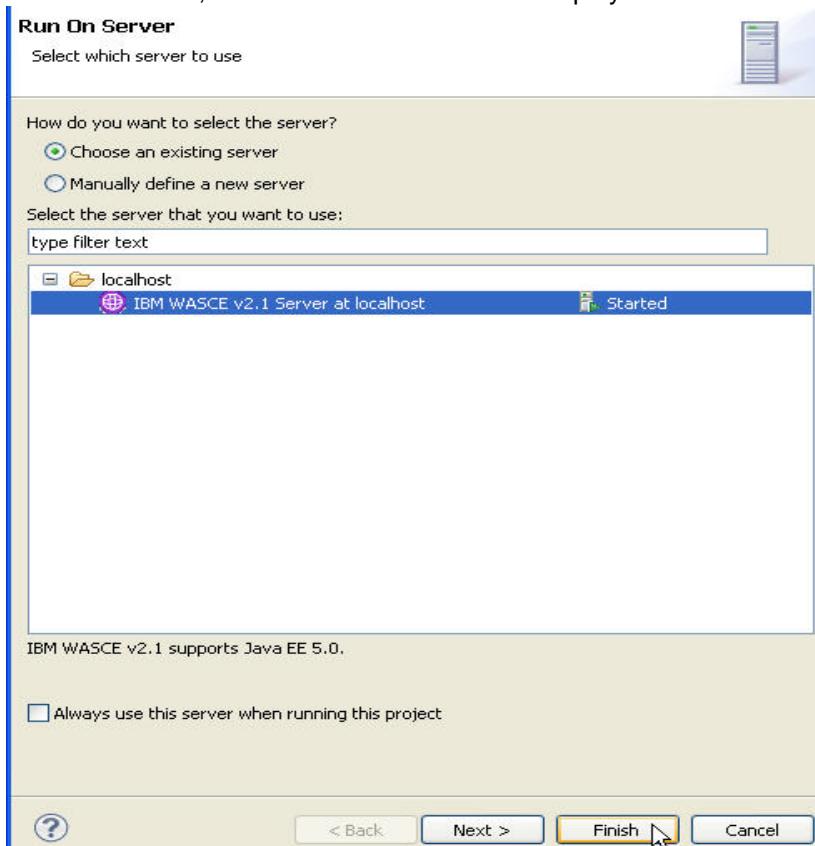
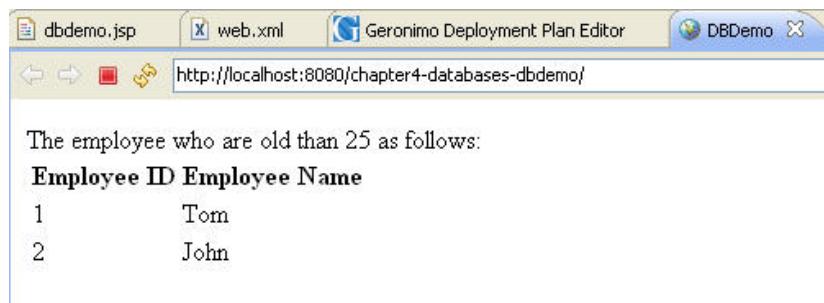


Figure 4.20 – Choose a server to deploy

5. Wait a few seconds, then you will see a browser window opened in Eclipse and displaying the retrieved data as shown in *Figure 4.21*.

**Figure 4.21 – Successful retrieved data from the database**

Congratulations! You have successfully developed a simple Web project to retrieve data from a database.

4.4 Support for other databases in Community Edition

Though we are using DB2 in this book, Community Edition supports many other database servers including Oracle, SQL Server, MySQL, Informix®, HSQLDB, PostgreSQL, Sybase, and Apache Derby. In the case of Apache Derby, it is actually embedded with Community Edition and can be accessed from the administrative console.

For a complete list of supported database servers, refer to the database type dropdown list in the Community Edition administrative console.

4.5 Summary

In this chapter you learned how to work with databases in Community Edition. Using DB2 Express-C, you first created a database using DB2 tools. Then you created and deployed a database pool within Community Edition using the Admin Console. With this database pool, you created some tables, inserted some data, and then ran a query. Finally, you learned how to develop a simple Web application that accessed the database. Community Edition supports many database vendors in addition to the embedded Apache Derby database.

4.6 Exercises

Part 1 - Configure a data source for DB2

Procedure

1. Create a DB2 database using IBM Data Studio or the CLP.
2. Create a database pool by referring to section 4.2.2

Part 2 - Validate the DB2 data source

Procedure

1. Execute the SQL statements below on the created data source using the Community Edition Administrative Console.

```
create table employees(empID INTEGER, empName VARCHAR(20), empAge  
INTEGER);  
insert into employees(empID, empName, empAge) values (1, 'Tom', 26);  
insert into employees(empID, empName, empAge) values (1, 'John', 28);  
insert into employees(empID, empName, empAge) values (1, 'Jerry', 25);
```

2. Use the SQL statement below to retrieve data using the Administrative Console.

```
select empID, empName from employees where empAge > 25
```

4.7 Review questions

1. What is the name of the Java API that is used to access relational databases?
2. What are the steps to use this Java API to access the database?
3. What type of JDBC driver for DB2 is used in this book?
4. What is the JNDI prefix for a JDBC data source?
5. List at least three database vendors that Community Edition supports.
6. What tools could be used to create DB2 databases?
 - A. DB2 Replication Center
 - B. DB2 Health Center
 - C. IBM Data Studio
 - D. DB2 Command Line Processor
 - E. All of the above
7. Which of the following files supplies only a type 2 JDBC driver for DB2?
 - A. db2jcc.jar
 - B. db2java.zip
 - C. db2jcc4.jar
 - D. sqlj.jar
 - E. None of the above
8. Which of the following Java versions does the type 4 JDBC 4.0 compliant driver of DB2 work with?
 - A. 1.3
 - B. 1.4.2

- C. 1.5
 - D. 6
 - E. None of the above
9. Which of the following java classes is used to create a database connection?
- A. java.sql.Statement
 - B. java.sql.ResultSet
 - C. javax.sql.DataSource
 - D. java.sql.Connection
 - E. javax.naming.Context
10. Which of the following database products is the embedded database in Community Edition?
- A. DB2
 - B. MySQL
 - C. Derby
 - D. Informix
 - E. None of the above

5

Chapter 5 – Enterprise Java Beans

An *Enterprise Java Bean* (EJB) is a mechanism to encapsulate business logic of an application. Community Edition supports the latest specification, EJB 3.0. This chapter discusses the development of EJBs in Community Edition.

In this chapter you will learn about:

- How to develop session beans in Community Edition
- How to develop an application client in Community Edition
- How to develop JPA-based applications in Community Edition

5.1 Community Edition Enterprise Java Beans: The big picture

Figure 5.1 provides an overview of EJB development in Community Edition. In the figure, there is an Admin Client on the right side which represents the Community Edition Admin console or the Eclipse IDE from where a user is managing the EJB application. As shown in (1), first, developers create EJBs and deploy them into the server. Then, developers make use of the EJBs by invoking them in a servlet (2) or by JNDI naming in an application client (3) or a Plain Old Java Object (POJO) client (4). As part of EJB specification, you can also create JPA-based applications (5) and deploy them with Community Edition.

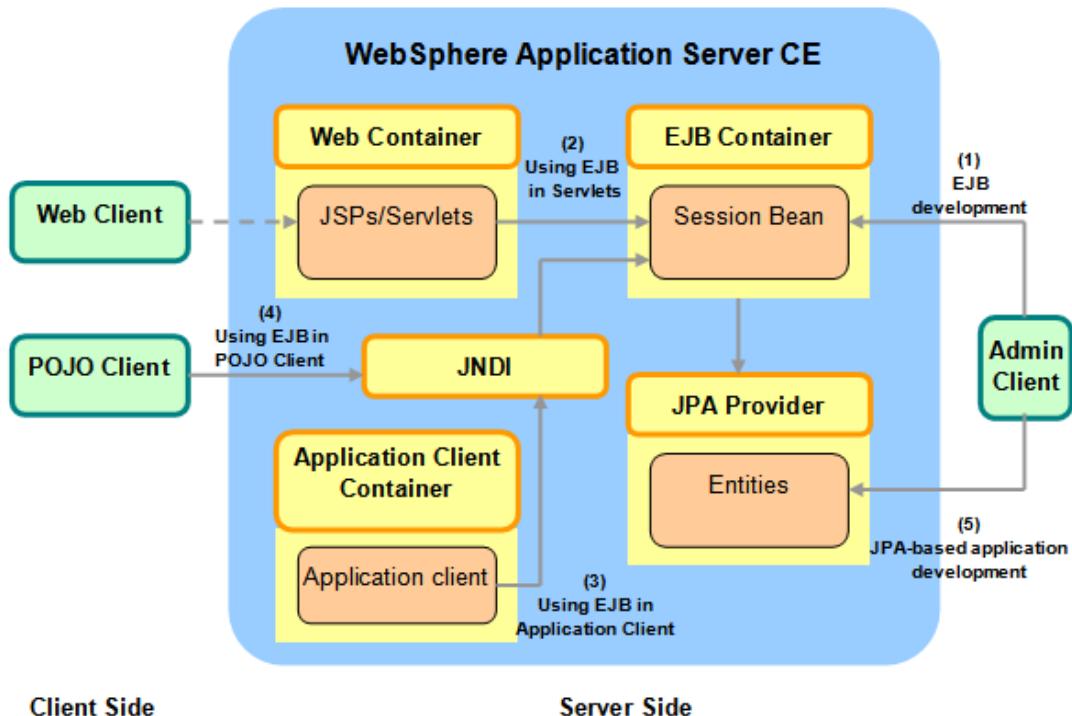


Figure 5.1 - Community Edition EJBs: The big picture

5.2 Developing EJBs in Community Edition

Community Edition includes *OpenEJB*, an open source EJB container. Since Community Edition is Java EE 5 compatible, you can create EJBs with annotations defined using the EJB 3.0 specification. Annotations are like meta-tags that you can add to your code and apply them to package declarations, type declarations, constructors, methods, fields, parameters, and variables. Annotations do not affect your code directly, but tools can inspect these annotations at compile time or runtime to generate additional constructs.

Before using an EJB, define the interface and implementation class first; then, deploy it into Community Edition.

Use the following steps to develop an EJB application in Community Edition:

1. Create an EJB in Community Edition
2. Deploy the EJB in Community Edition

These steps will be described in more detail in the next sections.

5.2.1 Creating an EJB

To create an EJB, you need to:

-
1. Define local and remote interfaces for the EJB
 2. Implement the EJB class

5.2.1.1 Defining local and remote interfaces

To avoid code duplication, define a common interface which can then be inherited by both, local and remote interfaces. In an EJB, a local interface is used by an invoker using the same *Java Virtual Machine (JVM)* while a remote interface is used by an invoker in a different JVM. Following the EJB 3.0 specification, the local interface is annotated with `@Local` while the remote interface is annotated with `@Remote`. *Listing 5.1* below provides an example of how you can define the business interface of an EJB. In the example, you define a business interface named `HelloInterface`.

```
package hello;
public interface HelloInterface {
    public void sayHello();
}
```

Listing 5.1 - Defining the business interface of the EJB

Listing 5.2 provides an example of how you can define the local interface of the EJB. It inherits the business interface.

```
package hello;
import javax.ejb.Local;
@Local
public interface HelloInterfaceLocal extends HelloInterface {}
```

Listing 5.2 - Defining local interface of the EJB

Listing 5.3 provides an example of how you can define the remote interface of the EJB.

```
package hello;
import javax.ejb.Remote;
@Remote
public interface HelloInterfaceRemote extends HelloInterface{}
```

Listing 5.3 - Defining the remote interface of the EJB

5.2.1.2 Implementing the EJB class

Now that the interfaces have been defined, you can implement them. In this step, all the methods of the EJB class will be implemented. As defined in the EJB 3.0 specification, `@Stateless` is used for a stateless EJB, an EJB that is normally short lived and does not maintain the state on behalf of the client. `@Stateful` is used for a stateful EJB. *Listing 5.4* provides an example.

```
package hello;
import javax.ejb.Stateless;
@Stateless
public class HelloEJB implements HelloInterfaceLocal,
HelloInterfaceRemote {
```

```
    public void sayHello() {
        System.out.println("Hello!");
    }
}
```

Listing 5.4 - Implementing the EJB class

In the listing, we implement the only method `sayHello()` in `HelloEJB`. This EJB has been completed.

5.2.2 Deploying an EJB

To deploy an EJB you need to:

1. Create a deployment plan for the EJB
2. Package the EJB
3. Deploy the EJB

5.2.2.1 Creating a deployment plan for the EJB

Just in the same way a servlet included in a Web project is accompanied with a deployment plan named `geronimo-web.xml`, for an EJB project in Community Edition the deployment plan is called `openejb-jar.xml` which is placed in the `META-INF` directory. All the dependencies should be declared in the deployment plan.

In Community Edition, every deployable artifact must have a unique configuration id. A **Configuration id** is a combination of a group id, artifact id, and artifact version and artifact type. A Configuration id is defined in the deployment plan. As a deployable artifact, the EJB also needs to define a configuration id in its deployment plan stored in `openejb-jar.xml`. Other resources or dependent artifacts can also be stated in the deployment plan. More details about an EJB deployment plan can be found in the deployment section of the Community Edition user guide at
<http://publib.boulder.ibm.com/wasce/V2.1.1/en/developing-deployment-plans.html>

To deploy EJBs, developers need to add an EJB namespace in the deployment plan. Listing 5.5 provides a simple example of an EJB deployment plan; it only defines the configuration id of the EJB.

```
<?xml version="1.0" encoding="UTF-8"?>
<ejb:openejb-jar
  xmlns:ejb="http://openejb.apache.org/xml/ns/openejb-jar-2.2"
  xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2">
  <dep:environment>
    <dep:moduleId>
      <dep:groupId>com.ibm.wasce.start</dep:groupId>
      <dep:artifactId>testejb</dep:artifactId>
      <dep:version>1.0</dep:version>
```

```

<dep:type>car</dep:type>
</dep:moduleId>
</dep:environment>
</ejb:openejb-jar>
```

Listing 5.5 - Defining the configuration id of EJB testejb

As shown in the above listing, this EJB is in group `com.ibm.wasce.start` with artifact as `testejb`. Its version number is `1.0` while its type is `car`. After deployed in Community Edition, the EJB will be placed in the following directory.

```
<WASCE_HOME>/repository/com.ibm.wasce.start/testejb/1.0/testejb-1.0.car/
```

5.2.2.2 Packaging the EJB

Packaging the EJB means that you can bundle the EJB java class and deployment plan into a **JAR** package. You can do this by exporting the EJB project in Eclipse or just packaging the EJB resources with the `jar` script.

5.2.2.3 Deploying the EJB

To deploy the EJB jar package in Community Edition, use this command:

```
<WASCE_HOME>/bin/deploy.[bat/sh] deploy -username system -password manager
[jarfile]
```

Note:

In Community Edition, you don't have to deploy EJBs with a deployment descriptor since your EJBs include annotations defined in the EJB 3.0 specification.

5.3 Working with EJBs in Community Edition

To make use of EJBs, a client must invoke the EJB interface methods. The client may be a servlet or an application client.

5.3.1 Using EJBs in a servlet

To make use of EJBs in a servlet you have to:

1. Refer to the EJB in the Servlet
2. Add the EJB to the dependency list of the servlet.

5.3.1.1 Refer to the EJB in the servlet

You can use the annotation `@EJB` to reference the EJB in a Java class. You can declare a class field referring to an EJB with this annotation in front of it. *Listing 5.6* provides an example.

```
public class TestServlet extends HttpServlet {  
    ...  
    @EJB(name="HelloEJB")  
    private HelloInterfaceLocal helloejb;  
  
    protected void doGet(HttpServletRequest request,  
                         HttpServletResponse response) throws  
    ServletException, IOException {  
        response.getWriter().println(ejb.sayHello());  
    }  
    ...  
}
```

Listing 5.6 - Using the @EJB annotation

With `@EJB`, `helloejb` will reference to an EJB named `HelloEJB`. If there is only one implementation for `HelloInterfaceLocal`, the name value can be omitted in the `@EJB` annotation.

5.3.1.2 Add the EJB into the dependency list of the servlet

As mentioned earlier, the deployment plan of an EJB project is called `openejb-jar.xml` and all dependencies should be declared there. If you want to reference an EJB, add it as a dependency in the deployment plan. *Listing 5.7* provides an example.

```
...  
<dep:environment>  
    <dep:moduleId>  
        <dep:groupId>default</dep:groupId>  
        <dep:artifactId>TestWeb</dep:artifactId>  
        <dep:version>1.0</dep:version>  
        <dep:type>car</dep:type>  
    </dep:moduleId>  
    <dep:dependencies>  
        <dep:dependency>  
            <dep:groupId>default</dep:groupId>  
            <dep:artifactId>TestEjb</dep:artifactId>  
            <dep:version>1.0</dep:version>  
            <dep:type>car</dep:type>  
        </dep:dependency>  
    </dep:dependencies>  
</dep:environment>  
...
```

Listing 5.7 - Adding an EJB as a dependency in the deployment plan

As you can see, the dependency declaration here is similar to the EJB definition in `openejb-jar.xml` of *Listing 5.5*.

5.3.2 Using an EJB in an application client

An application client is a stand-alone Java application invoking methods in an EJB. Like other Java EE compliant application servers, Community Edition provides an application client container where you can deploy the application client. You can create an application client following these steps:

1. Import or copy the interface file of an EJB to the application client
2. Find the EJB with JNDI
3. Invoke the methods in the EJB interface

5.3.2.1 Import or copy the interface file of an EJB to the application client

A deployed server-side application client has to reference the local interface of an EJB. A stand-alone or deployed client-side application client has to reference remote interface. To reference the right interface, copy the interface file of the EJB to the application client.

5.3.2.2 Find the EJB with JNDI

In Community Edition the JNDI name for the local interface is by default `[EJB name] + Local`. For the remote interface the JNDI name is by default `[EJB name] + Remote`. For example, for an EJB named `HelloEJB`, `HelloEJBLocal` is the local interface JNDI name and `HelloEJBRemote` is the remote interface JNDI name.

Before using the JNDI name to find the EJB, you need to initialize the context with some Community Edition specific properties. As mentioned earlier, Community Edition uses OpenEJB as the EJB container; therefore, the value of the properties context factory and naming provider URL has to be set. Normally, they are set to `org.apache.openejb.client.RemoteInitialContextFactory` and `ejbd://localhost:4201`. Listing 5.8 provides an example.

```
public class HelloEJBAppClient {
    public static void main(String[] args) throws NamingException{
        Properties prop=new Properties();
        prop.put(Context.INITIAL_CONTEXT_FACTORY,
        "org.apache.openejb.client.RemoteInitialContextFactory");
        prop.put("java.naming.provider.url", "ejbd://localhost:4201");
        Context context = new InitialContext(prop);
        HelloInterfaceRemote hellEjb
        =(HelloInterfaceRemote)context.lookup("HelloEJBRemote");
        hellEjb.sayHello();
    }
}
```

Listing 5.8 - An application client invoking HelloEJB

5.3.2.3 Invoke the methods in the EJB interface

There are two ways to run the application client:

a) Run as an standalone application client (POJO Client)

Just as a regular Java application, run the application client on a different JVM from that of the Community Edition server. This client is also known as a POJO Client. To launch the application client, use this command:

```
java [ApplicationClientQualified Name]
```

For example,

```
java appclient.HelloEJBAppclient
```

b) Deployed and run in the application client container

To deploy the application client in the application client container, a deployment plan is required. The deployment plan is called *geronimo-application-client.xml*. Similar to EJB's deployment plan, a *configuration id* has to be defined. However, because the application client can be used in the server environment and client environment, this deployment plan has to include two *configuration ids*. Listing 5.9 provides an example.

```
<?xml version="1.0" encoding="UTF-8"?>
<client:application-client
  xmlns:client="http://geronimo.apache.org/xml/ns/j2ee/application-client-
  2.0"
  xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2">
  <dep:client-environment>
    <dep:moduleId>
      <dep:groupId>default</dep:groupId>

      <dep:artifactId>HelloAppClientClientSide</dep:artifactId>
        <dep:version>1.0</dep:version>
        <dep:type>car</dep:type>
    </dep:moduleId>
  </dep:client-environment>
  <dep:server-environment>
    <dep:moduleId>
      <dep:groupId>default</dep:groupId>
      <dep:artifactId>HelloAppClientServerSide</dep:artifactId>
      <dep:version>1.0</dep:version>
        <dep:type>car</dep:type>
    </dep:moduleId>
  </dep:server-environment>
</client:application-client>
```

Listing 5.9 - Deployment plan of an application client

Note:

The *configuration id* for server and client environments should be different.

The application client with its deployment plan can be packaged as a jar file. *Figure 5.2* provides an example of a jar file structure. All the class files are built by Eclipse automatically from the source code created above; *geronimo-application-client* is the deployment plan for this application client.



Figure 5.2 - Jar file structure of an application client with its deployment plan

Deploy and run the application client with this command.

```
java -jar <WASCE_HOME>/bin/client.jar [PATH to application client]
```

For example:

```
java -jar <WASCE_HOME>/bin/client.jar
      HelloAppClient/HelloEJBAppClient
```

5.4 Java Persistence API Entities Development in Community Edition

Java Persistence API (JPA) is a new API as part of the Java EE 5 platform for data persistency. Although JPA is part of EJB 3.0, its development process is quite different from an EJB. This section describes how to develop JPA entities in Community Edition. There are two methods to create JPA entities:

- Create JPA entities manually with JPA annotations
- Generate JPA entities from tables of a database

5.4.1 Creating JPA entities manually

To create JPA entities manually, you need to use the annotation defined in the JPA specification. All the annotations defined there, such as *@Entity*, *@EntityManager*, *@PersistenceUnit* and so on, can be used in your JPA classes.

You can create JPA entities within an EJB project, so that JPA entities can be included in an EJB jar package. To successfully deploy JPA entities, you need to provide another configuration file named `persistence.xml`. This file is located in `META-INF`, together with `openejb-jar.xml`.

In Community Edition, you have to specify some parameters in this file so that Community Edition can find your entities. *Listing 5.10* provides an example of the `persistence.xml` template.

```
<persistence>
    <persistence-unit name="persistence unit name">
        <provider>JPA provider class name</provider>
        <description>Description information here</description>
        <jta-data-source>JNDI name of Data source </jta-data-source>
        <jar-file>Jar file name of persistence unit</jar-file>
        <class>persistence class</class>
        <properties>
            <!--following are property names and values-->
            <property name="propertyName" value="PropertyValue" />
        </properties>
    </persistence-unit>
</persistence>
```

Listing 5.10 - The `persistence.xml` template

Note:

Community Edition uses OpenJPA as the JPA provider, so by default the JPA provider class name in `persistence.xml` is:

```
org.apache.openjpa.persistence.PersistenceProviderImpl.
```

5.4.2 Generating JPA entities from tables of a database

For convenience, it is recommended to develop a JPA project with Eclipse and the WebSphere Application Server Community Edition eclipse plug-in (WEP) installed. Follow these steps:

1. Create a database pool in Community Edition

In this step, a database pool to be used by JPA needs to be added. This was explained in *Chapter 4, Working with databases*.

2. Create the JPA project in Eclipse

In this step, a JPA project is created with eclipse. The JPA project often contains several entities generated from the database. To make use of the JPA entities, other projects such as Web projects, and EJB projects often depend on JPA projects. The referenced JPA project will be deployed with the Web project or EJB project together in Community Edition.

3. Create a database connection in Eclipse

Since the database pool has been created in the Community Edition server, a connection is needed in Eclipse. With this database connection, the data management plug-in can get database information to prepare for the generation of JPA entities.

4. Generate JPA entities

JPA can map Java classes to database entities. Once the tables in the database have been defined, several JPA entities can be generated directly from these tables. In that way, both EJB project and Web project can make use of the generated entities.

An example using JPAs is provided in the *Exercises* section of this chapter.

5.5 Summary

This chapter discussed EJB 3.0 related topics with Community Edition, including EJB 3.0 development, application client development and JPA entities application in Community Edition.

5.6 Exercises

In this exercise you will develop an EJB, JPA and a servlet to calculate university credits of students. You will use Eclipse with the WEP installed and DB2 Express-C 9.7. Credits of all the students will be shown using a servlet.

Note:

The exercise complete code and solutions are provided in the zip file `gettingStartedWithWasceEdition1st_src.zip` accompanying this book.

Follow these steps:

1. Create a database and a table in DB2 Express-C 9.7

Follow the steps introduced in *Chapter 4.2.1 – Creating a database*, to create a database called **Studentdb** as shown in *Figure 5.3*.

```
C:\>db2 create db studntdb using codeset utf-8 territory US
DB20000I The CREATE DATABASE command completed successfully.
```

Figure 5.3 – Create database *Studntdb* DB2 Express-C 9.7

Then, connect to the database, and create the table *Student* using the SQL statements in *Listing 5.11*.

```
CONNECT TO STUDNTDB;

CREATE TABLE Student(
    ID INT NOT NULL, SCORE1 INT NOT NULL, SCORE2 INT NOT NULL,
    SCORE3 INT NOT NULL, CREDIT INT ,
    CONSTRAINT STUDENT_SOLE_NUM PRIMARY KEY (ID));
```

Listing 5.11 – SQL statement to create table Student

2. Create a database pool

Follow the steps in section 4.2.2 to create a database pool for database *Studntdb* and deploy it into Community Edition. The database pool name is *jdbc/Student* as shown in *Figure 5.3*.

Name	Deployed As	State	Actions
MonitoringClientDS	Server-wide	running	Edit Usage Delete
NoTxDatasource	Server-wide	running	Edit Usage Delete
SystemDatasource	Server-wide	running	Edit Usage Delete
jdbc/ActiveDS	Server-wide	running	Edit Usage Delete
jdbc/ArchiveDS	Server-wide	running	Edit Usage Delete
jdbc/Student	Server-wide	running	Edit Usage Delete
jdbc/juddiDB	org.apache.geronimo.configs/uddi-tomcat/2.1.4/car	running	Edit Usage Delete

Figure 5.4 – Deployed database pool in Community Edition

3. Create a JPA project in Eclipse

Choose *File->New->JPA project*, to create a JPA project named *ejb-studentjpa*. If you can't see *JPA project* as a choice, just click *Other...* then choose *JPA project* from the popped up dialog as shown in *Figure 5.5*.

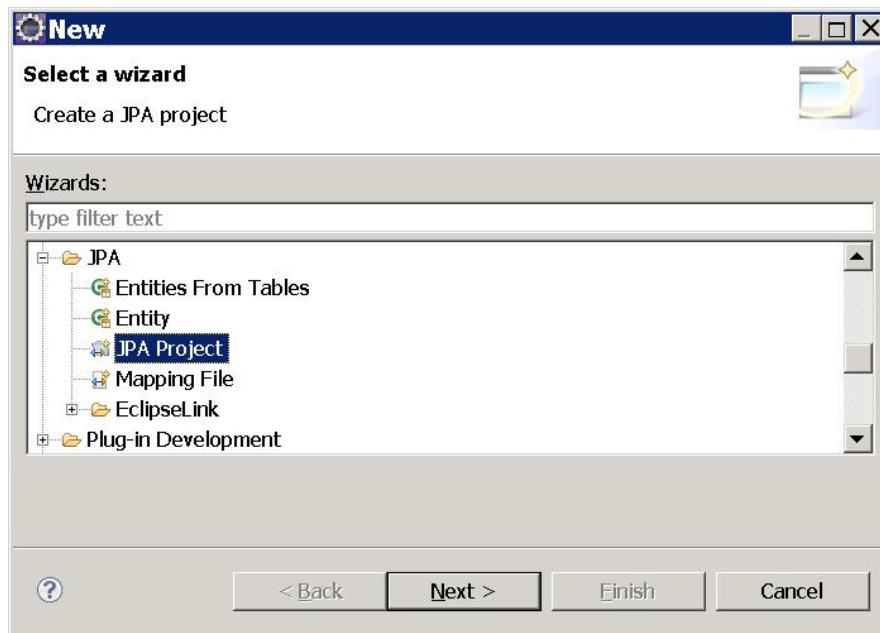


Figure 5.5 – Dialog for creating a JPA project

4. Create an entity in the JPA project

Right click on `ejb-studentjpa`, choose *New->Class*, create a class named `Student` as shown in *Figure 5.6*. The code snippet is shown in *Listing 5.12*. The complete code is in the accompanying zip file with exercises for this book.

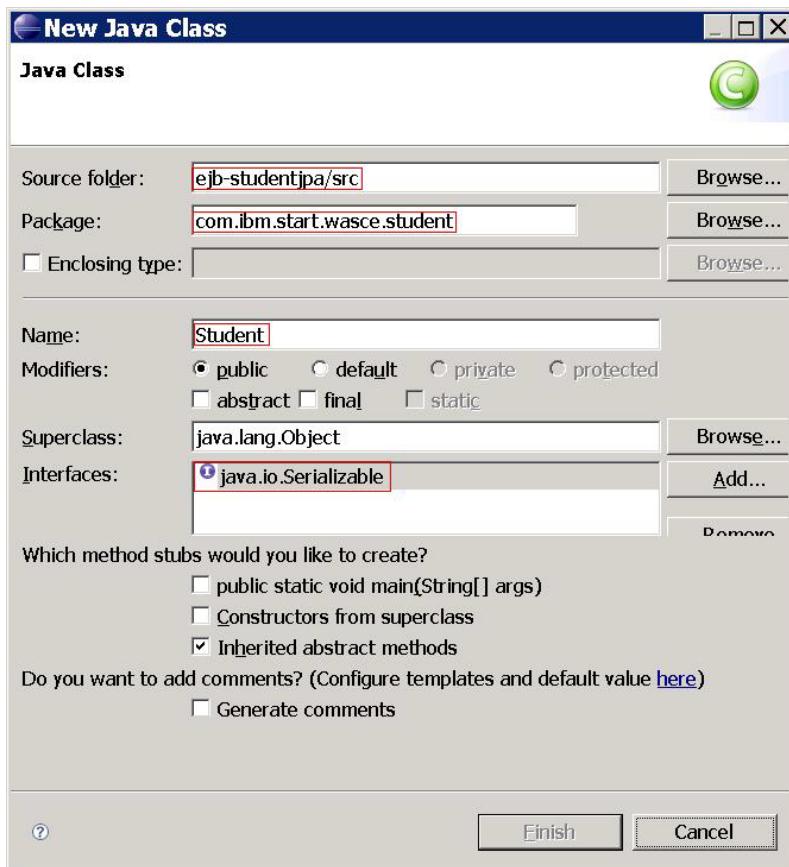


Figure 5.6 – Create an entity in JPA project

```
package com.ibm.start.wasce.student;
.....
@Entity
@Table(name="Student")
public class Student implements Serializable {
    @Id
    private int id;
    private int score1;
    private int score2;
    private int score3;
    private int credit;
    private static final long serialVersionUID = 1L;
    public Student(){
        super();
    }
}
```

```
.....
```

```
}
```

Listing 5.12 – definition of the entity *Student*

5. Update *persistence.xml*

Add configuration information to *persistence.xml* which should look as shown in *Listing 5.13*

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0"
    xmlns="http://java.sun.com/xml/ns/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">
    <persistence-unit name="Student" transaction-type="JTA">
        <provider>
            org.apache.openjpa.persistence.PersistenceProviderImpl
        </provider>
        <jta-data-source>jdbc/Student</jta-data-source>
        <class>com.ibm.start.wasce.student.Student</class>
        <properties>
            <property name="openjpa.TransactionMode"
                value="managed"/>
            <property name="openjpa.ConnectionFactoryMode"
                value="managed"/>
            <property name="openjpa.jdbc.DBDictionary"
                value="db2"/>
        </properties>
    </persistence-unit>
</persistence>
```

Listing 5.13 – *persistence.xml* for JPA project *ejb-studentjpa*

6. Create an EJB

Step 1:

Choose *File -> New->Other...-> EJB project, ejb-studentejb* then create the interface ***StudentEJBInterface*** and its implementation class ***StudentEJB*** as shown in *Listings 5.14 and 5.15*

```
package com.ibm.start.wasce.studentejb;
import java.util.List;
import javax.ejb.Local;
@Local
public interface StudentEJBInterface {
    public List<Integer> calculateCredit();
```

```
}
```

Listing 5.14 – Definition of *StudentEJBInterface*

```
package com.ibm.start.wasce.studentejb;
.....
@Stateless
public class StudentEJB implements StudentEJBInterface {
    @PersistenceUnit(unitName = "Student")
    EntityManagerFactory emf;

    public List<Integer> calculateCredit() {
        EntityManager em = emf.createEntityManager();
        List<Integer> creditList = new ArrayList<Integer>();
        Query query =
            em.createQuery("select s from Student AS s order by s.id");
        List<Student> studentList = query.getResultList();
        for (Student student : studentList) {
            int credit = (int) ((float) (student.getScore1()
                + student.getScore2() + student.getScore3()) / 300 * 5);
            student.setCredit(credit);
            creditList.add(credit);
        }
        em.flush();
        em.close();
        return creditList;
    }
}
```

Listing 5.15 – Implementation of *StudentEJB*

Step 2:

Right click *ejb-studentejb* -> *Build Path* -> *Configure Build Path* -> *Projects*, add *ejb-studentjpa* project on build path

Step 3:

Right click *ejb-studentejb* -> *Properties* ->*Project Reference*, click *ejb-studentjpa* as a reference

7. Create a Dynamic Web Project

Step 1:

In Eclipse click on *File* -> *New* -> *Other...-> Dynamic Web Project* to create a dynamic Web project **ejb-studentweb**.

Step 2:

Right click on ejb-studentweb -> New -> Servlet to create a Servlet named **StudentCreditCalculator**. Paste the code from the accompanying zip file with exercises for this book. The code snippet is shown in *Listing 5.16*

```
package com.ibm.start.wasce.studentweb;
public class StudentCreditCalculator extends HttpServlet {
.....
    @EJB
    private StudentEJBInterface studentejb;
    protected void doGet(HttpServletRequest request,
    HttpServletResponse response)
        throws ServletException, IOException{
        List<Integer> creditList = studentejb.calculateCredit();
        for (int i=0;i<creditList.size();i++) {
            response.getWriter().println("student id:"+ (i+1)
+ " \t credit: " + creditList.get(i)+"\n");
        }
    }
.....
}
```

Listing 5.16- The servlet referencing ejb-studentejb**Step 3:**

Right click *ejb-studentweb* -> Build Path -> Configure Build Path -> Projects, add *ejb-studentejb* project on build path

Step 4:

Right click *ejb-studentweb* -> Properties -> Project Reference, click *ejb-studentejb* as a reference.

8. Create an Enterprise Application Project**Step 1:**

In Eclipse click on File -> New -> Other...-> Enterprise Application Project to create a enterprise application project **ejb-studentear**. In the creating wizard, fill in the second dialog as shown in *Figure 5.7*, then click *Finish*.

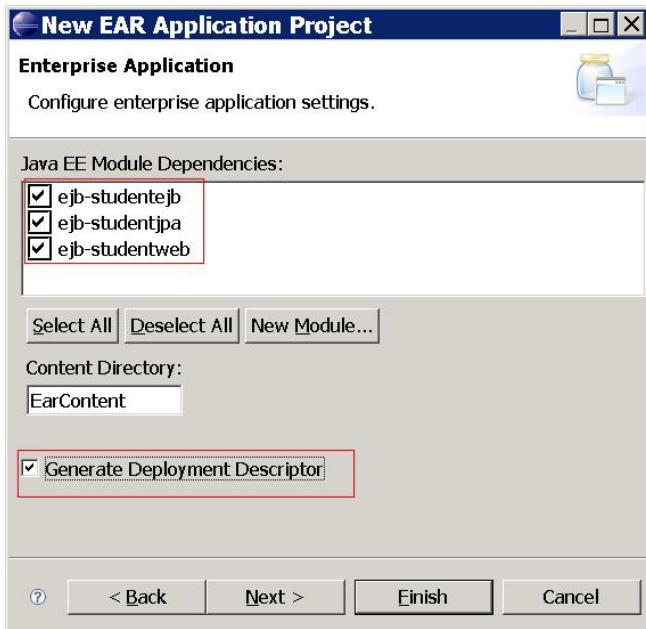


Figure 5.7 - Create an enterprise application project

Step 2:

Right click on *ejb-studentear* -> *Properties* -> *Project Reference*, select all the projects there.

Step 3:

Right click *ejb- studentear* -> *Properties* ->*Java EE Module Dependency*, select all the projects there.

Step 4:

In project *ejb-studentear*, expand *META-INF* under *EarContent*, open the file *geronimo-application.xml* and add the snippet in *Listing 5.17* into the file. It will add dependency for database pool deployed before.

```
<dep:environment>
.....
<dep:dependencies>
    <dep:dependency>
        <dep:groupId>console.dbpool</dep:groupId>
        <dep:artifactId>jdbc_Student</dep:artifactId>
    </dep:dependency>
</dep:dependencies>
</dep:environment>
```

Listing 5.17- add dependency for the database pool in geronimo-application.xml

9. Deploy and Run

Finally, deploy the `ejb-studentear` project into the Community Edition server. Insert some sample data into the `Student` table, and then invoke the servlet `StudentCreditCalculator` with the URL below.

<http://localhost:8080/ejb-studentweb/StudentCreditCalculator>.

The result will look as shown in *Figure 5.8*.

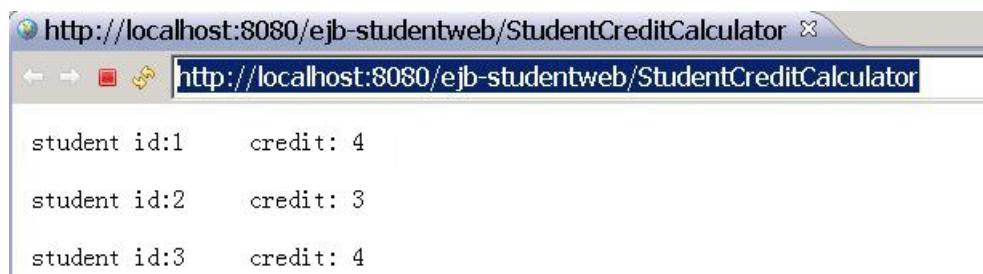


Figure 5.8 - Invoking the `StudentCreditCalculator` servlet

5.7 Review questions

1. What is the URL to use when you query an EJB using JNDI?
2. What are the default names for an EJB local interface and remote interface?
3. What is the typical procedure to use an EJB in a servlet?
4. What is the typical procedure to use an EJB in an application client?
5. What is the typical procedure to create JPA entities from tables?
6. Which are the annotations used for EJB 3.0?
 - A. `@EJB`
 - B. `@Remote`
 - C. `@Local`
 - D. `@Servlet`
 - E. `@JSP`
 - F. `@Null`
7. Which are the annotations used for JPA?
 - A. `@Entities`

- B. @Entity
 - C. @Persistence
 - D. @Persistences
 - E. @PersistenceUnit
8. Which type of application clients are there in Community Edition?
- A. Standalone application client
 - B. Container managed application client
 - C. WebSphere Application Server Community Edition application client
 - D. Geronimo application client
 - E. Normal application client
9. Which are the methods of creating JPA entities?
- A. Create JPA entities manually with annotations
 - B. Generating JPA entities from tables
 - C. Community Edition can generate JPA entities automatically
 - D. A servlet can generate JPA entities automatically
 - E. An EJB can generate JPA entities automatically
10. What is the name of the EJB deployment plan in Community Edition?
- A. ejb-jar.xml
 - B. openejb-jar.xml
 - C. geronimo-web.xml
 - D. web.xml
 - E. None of the above

6

Chapter 6 – Messaging

Messaging is a way to communicate asynchronously, similar to an e-mail system. The sender creates a message, and sends it to a messaging destination through a messaging subsystem. The messaging subsystem accepts it for delivery and holds it. The receiver checks the messaging subsystem to receive the message.

Messaging allows for a loosely coupled solution. The sender application only has to ensure that the message was accepted for delivery and then can proceed with other processing. An outage on either the sender or receiver parts, do not immediately affect the entire system.

Java Message Service (**JMS**) is used to support messaging in the Java EE world.

In this chapter you will learn about:

- How to configure a JMS resource group
- How to use JMS resources in different applications of Community Edition.

6.1 Community Edition Messaging: The big picture

Figure 6.1 provides an overview on how messaging works in Community Edition. First in (1), an administrator creates a JMS resource group in the server. Then there are three ways to use the created JMS resource group:

- In (2a) and (2a.1), the Servlet/jsp uses JNDI lookup or resource injection to get a reference to the JMS resources, and then uses this reference to send or receive messages of the JMS resource in the server.
- In (2b) and (2b.1), you can use annotation to specify the JMS queue or topic that the *Message Driven Bean (MDB)* listens to. When the message arrives, the MDB will react according to the content of received message.
- In (2c), a standalone java client (POJO client) can use `ActiveMQConnectionFactory` API to get the JMS resource and use the JMS resource directly.

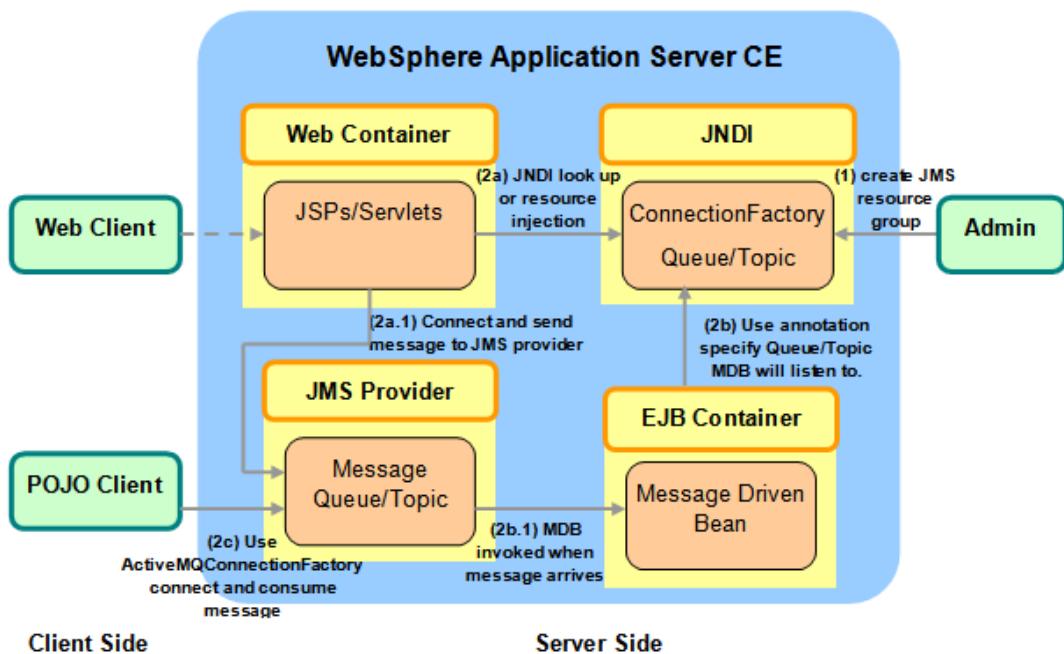


Figure 6.1 - Messaging: The big picture

6.2 Java Message Service

Java EE includes the Java Message Service (**JMS**) specification, which defines a set of interfaces and semantics that Java EE applications can utilize to send or receive messages to a messaging provider. A messaging provider provides the infrastructure that supports messaging, such as Active MQ or IBM WebSphere MQ products. JMS is the API specification only. It is analogous to what JDBC does for accessing databases.

6.2.1 JMS application roles

A JMS application is composed of the following parts:

1. **JMS provider** - This is a messaging system that implements the JMS specification. Community Edition uses *Active MQ* as its default JMS provider.
2. **JMS client** - This is a Java application that sends and receives messages. There are two kinds:
 - **JMS Producer** - A JMS client that sends messages to a destination of a JMS provider.
 - **JMS Consumer** - A JMS client that receives messages from a destination of a JMS provider.

3. **Messages** - These are objects that are used to communicate information between JMS clients.
4. **Administered objects** - These are preconfigured JMS objects that are created by an administrator of the JMS provider for the use of JMS clients. In Community Edition, these are called the **JMS resource group**.
 - **Destination** - A destination encapsulates addressing information for a specific JMS provider. A JMS client uses a destination object to address a message to a specific destination on the underlying JMS provider.
 - **ConnectionFactory** - A connection factory encapsulates the configuration information that is required to connect to a specific JMS provider. A JMS client uses a connection factory to create a connection to that JMS provider.

6.2.2 JMS application models

As messaging technologies have evolved, two types of asynchronous messaging models have emerged, **Point-to-Point** and **Publish/Subscribe**. These models describe how JMS clients communicate with each other through JMS destination provided by the JMS provider.

6.2.2.1 Point-to-Point

In this model, there is a one-to-one mapping between the sender and receiver of a message. The destination is usually referred to as a **queue**. *Figure 6.2* shows the Point-to-Point messaging model.

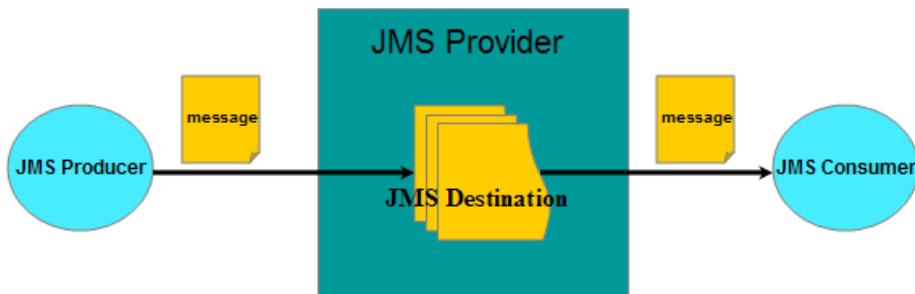


Figure 6.2 - Point-to-Point messaging model

6.2.2.2 Publish/Subscribe

In this model, there is potentially a one-to-many relationship between the sender and receiver of a message. The destination is usually referred to as a **topic**. *Figure 6.3* shows the Publish/Subscribe messaging model.

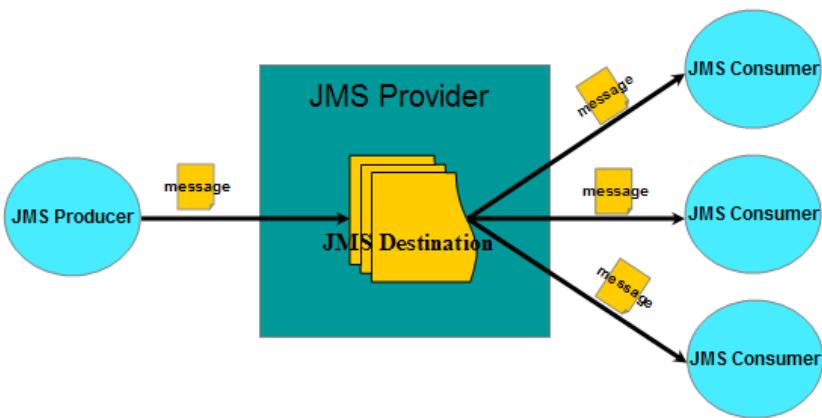


Figure 6.3 - Publish/Subscribe messaging model

6.2.3 JMS API

Table 6.1 below lists the essential APIs in JMS.

JMS unified API	Point-to-Point (Queue specific)	Publish/Subscribe (Topic specific)
ConnectionFactory	QueueConnectionFactory	TopicConnectionFactory
Connection	QueueConnection	TopicConnection
Destination	Queue	Topic
Session	QueueSession	TopicSession
MessageProducer	QueueSender	TopicPublisher
MessageConsumer	QueueReceiver	TopicSubscriber

Table 6.1 - JMS API

The JMS unified API shown in the first column of *Table 6.1* is the recommended API to use. This unified API started with JMS 1.1; it is merged from both, the Queue JMS API and the Topic JMS API by abstracting the same logic out of the two models.

6.2.4 JMS application development steps in Community Edition

To develop a JMS application with Community Edition, follow these steps:

1. Configure a JMS resource group in Community Edition. This will be described in section 6.3.

2. Get JMS resources including `ConnectionFactory` and `Queue/Topic` in the JMS resource group. There are three ways to get JMS resources.
 - Use JNDI looking up in a servlet/EJB.
 - Use resource injection in a servlet/EJB.
 - Use `ActiveMQConnectionFactory` API to connect to a JMS provider directly from a stand-alone Java application.
3. After getting `ConnectionFactory` and `Destinations`:
 - Use `ConnectionFactory` to create a `Connection`.
 - Use `Connection` to create a `Session`.
4. Use the `Session` to create a `MessageProducer` or `MessageConsumer` to the `Destination`.
5. Use the `MessageProducer` to send messages, use the `MessageConsumer` to receive messages.

6.3 Configuring a JMS resource group in Community Edition

A JMS provider acts as an intermediary between message producers and consumers. You need a JMS provider for JMS applications to send and receive messages. Before developing JMS applications, you have to create a JMS resource group including `ConnectionFactory` and `Destinations` in JMS provider.

6.3.1 Creating a JMS resource group

To create a JMS resource group in Community Edition, you need to:

1. Start the Community Edition server and launch the Community Edition administrative console (login as `system` with password `manager`).
2. In the left hand navigation pane, select *Services -> JMS Resources* to open the JMS Resources editor as shown in *Figure 6.4*.

**Figure 6.4 – Create a new JMS Resource Group**

3. Under the *Create a new JMS Resource Group* section in *Figure 6.4* above, click *For ActiveMQ*.
4. In the *JMS Resource Group -- Configure Server Connection* page input a name in the *Resource Group Name* field such as *myJmsResourceGroup*. This is shown in *Figure 6.5*. Accept the rest of the defaults and click *Next*.

JMS Resources □ + -

JMS Resource Group -- Configure Server Connection

The settings on this screen are different for each JMS provider, but they generally configure connectivity to the JMS server. Connection factories or destinations you create in the next step typically use these settings to communicate with the server.

Resource Group Name: A unique name for the resource adapter; used to generate the configuration name for this resource group as well as to connect Message-Driven Beans to the JMS server using the settings on this page.

Basic Configuration Settings.

ServerUrl:

Figure 6.5 – Configure Server Connection

Now you can start creating the JMS connection factory following the instructions in the next section.

6.3.2 Creating a JMS connection factory

To create a JMS connection factory, follow these steps:

1. In the *JMS Resource Group -- Current Progress* page, click *Add Connection Factory*. This is illustrated in *Figure 6.6*.

JMS Resource Group -- Current Progress

So far, you've entered the basic configuration information required for a JMS resource group. Now you can create connection factories and destinations. When you're finished adding connection factories and destinations, you can review the Geronimo deployment plan for this JMS resource group, or go ahead and deploy it.

[Add Connection Factory](#) [Add Destination](#)

[Cancel](#)

Figure 6.6 – Add Connection Factory to the JMS Resource Group

2. In the *JMS Resource Group -- Select Connection Factory Type* page, select `javax.jms.ConnectionFactory` from the *JMS Factory Type* drop down list as shown in *Figure 6.7*. This is the unified interface that is independent of the Destination Type. We always encourage you to use the unified interface. Click *Next*.

JMS Resource Group -- Select Connection Factory Type

JMS Factory Type: `javax.jms.ConnectionFactory`

possible connection factory interfaces. Select the one you want to use.

[Next](#)

Current Status for JMS Resource Group myJmsResourceGroup

- 1 Connection Factory
 - In Process
- 0 Destinations

[Cancel](#)

Figure 6.7 – Select Connection Factory Type

3. In the *JMS Resource Group -- Configure Connection Factory* page, set the Connection Factory Name to `myConnectionFactory` as shown in *Figure 6.8*. Then click *Next*.

JMS Resource Group -- Configure Connection Factory

Connection Factory Name: `myConnectionFactory`

A unique name for the connection factory; used to refer to this connection factory when mapping resource references from application components.

Transaction Support: `XA`

Which JMS interface this connection factory should support.

Connection Pool Parameters

Pool Min Size:

Figure 6.8 – Configure Connection Factory

Now you can start to create the JMS Destinations following the instructions in next section.

6.3.3 Creating a JMS queue and topic destinations

To create a JMS queue and topic destinations, follow these steps:

1. In the *JMS Resource Group -- Current Progress* page, click *Add Destination* as shown in *Figure 6.9*.

The screenshot shows the 'JMS Resources' interface with the 'JMS Resource Group -- Current Progress' tab selected. A table lists a single connection factory named 'myConnectionFactory'. Below the table are four buttons: 'Add Connection Factory', 'Add Destination' (which is highlighted with a red box), 'Show Plan', and 'Deploy Now'.

Resource Group myJmsResourceGroup		
Type	Name	Interface
Connection Factory	myConnectionFactory	javax.jms.ConnectionFactory

Add Connection Factory Add Destination Show Plan Deploy Now

Figure 6.9 – Add Destination

2. In the *JMS Resource Group -- Select Destination Type* page, select *javax.jms.Queue* as the *JMS Destination Type* and click *Next*. This is illustrated in *Figure 6.10*.

The screenshot shows the 'JMS Resources' interface with the 'JMS Resource Group -- Select Destination Type' tab selected. A dropdown menu under 'Type' is set to 'javax.jms.Queue'. A tooltip explains that this interface declares several possible destination interfaces. Below the dropdown is a 'Next' button.

JMS Destination Type *javax.jms.Queue*

javax.jms.Queue declares several possible destination interfaces. Select the desired interface type for this destination.

Next

Current Status for JMS Resource Group myJmsResourceGroup

- 1 Connection Factory
 - myConnectionFactory
- 1 Destination
 - *In Process*

Figure 6.10 – Select Destination Type

3. In the *JMS Resource Group -- Configure Destination* page, set the *Message Destination Name* to *myQueue*, set the *PhysicalName* to *myQueue*, and click *Next*. This is shown in *Figure 6.11*.

JMS Resources

JMS Resource Group -- Configure Destination

Message Destination Name: myQueue
A unique name for the connection factory; used to refer to this connection factory when mapping resource references from application components.

PhysicalName: myQueue

Destination Configuration Settings

Current Status for JMS Resource Group myJmsResourceGroup

- 1 Connection Factory

Figure 6.11 – Configure Destination

- In the *JMS Resource Group -- Current Progress* page, click *Deploy Now* to deploy the JMS resource group to Community Edition. This is illustrated in *Figure 6.12*.

JMS Resources

JMS Resource Group -- Current Progress

These are the connection factories and destinations you've added to the JMS resource group so far. When you're finished adding connection factories and destinations, you can review the Geronimo deployment plan for this resource group, or go ahead and deploy it.

Type	Name	Interface
Connection Factory	myConnectionFactory	javax.jms.ConnectionFactory
Destination	myQueue	javax.jms.Queue

Resource Group myJmsResourceGroup

Add Connection Factory **Add Destination** **Show Plan** **Deploy Now** **Cancel**

Figure 6.12 – Deploy the created JMS Resource Group

- In the resulting *JMS Resources* page, verify that the **myJMSResourceGroup** group exists with the connection factory and destination as illustrated in *Figure 6.13*.

JMS Resources

Completed with id console.jms/myJmsResourceGroup/1.0/rar

This page lists all the available JMS Resource Groups.

ActiveMQ RA (org.apache.geronimo.configs/activemq-ra/2.1.4/car)				
Type	Name	Deployed As	State	Actions
Connection Factory	DefaultActiveMQConnectionFactory	Server-wide	running	
Queue	SendReceiveQueue	Server-wide	running	
Queue	MDBTransferBeanOutQueue	Server-wide	running	

myJmsResourceGroup (console.jms/myJmsResourceGroup/1.0/rar)				
Type	Name	Deployed As	State	Actions
Connection Factory	myConnectionFactory	Server-wide	running	
Queue	myQueue	Server-wide	running	

Figure 6.13 – JMS Resource Group deployed successfully

6. You can also see the group module full name ***console.jms/myJmsResourceGroup/1.0/rar*** from the resulting page. You will use the JMS Resource group in applications later.

Note:

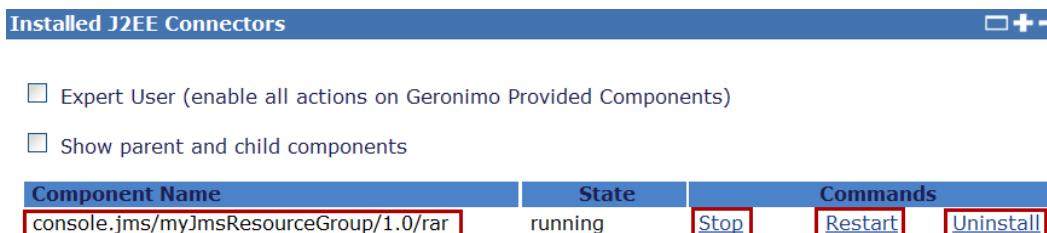
You can also save the JMS resource deployment plan by clicking the *Show Plan* button in the step 4 above. With the saved deployment plan. You can deploy the JMS resource group within enterprise applications easily. For details, see

<http://publib.boulder.ibm.com/wasce/V2.1.1/en/configuring-resources-in-the-asset-scope.html#Configuringresourcesintheassetscope-ConfiguringaJMSresourcegroup>

6.3.4 Stop, restart, or uninstall an installed JMS resource group.

To stop, restart or uninstall an installed JMS resource group, follow these steps:

1. In the left hand navigation pane, select *Applications -> J2EE Connectors* to open the *Installed J2EE Connectors* page as shown in *Figure 6.14*.



The screenshot shows a table titled 'Installed J2EE Connectors'. The table has three columns: 'Component Name', 'State', and 'Commands'. The first row contains the following data:

Component Name	State	Commands
console.jms/myJmsResourceGroup/1.0/rar	running	Stop Restart Uninstall

There are also two checkboxes at the top left of the table: 'Expert User (enable all actions on Geronimo Provided Components)' and 'Show parent and child components'.

Figure 6.14 – Installed J2EE Connectors

2. You can see the component ***console.jms/myJmsResourceGroup/1.0/rar*** you just created.
 - Click *Stop* to stop the JMS resource group.
 - Click *Restart* to restart the JMS resource group.
 - Click *Uninstall* to uninstall the JMS resource group.

6.4 Using Community Edition JMS resource

After creating the JMS resource group, you get a running JMS resource group as well as its module name in Community Edition. To use a JMS resource group in JMS applications, you must add a dependency to the resource group module in your deployment plan.

- If your Web application accesses a queue to send messages, you need to add a dependency in `geronimo-web.xml` to the JMS resource group which the queue belongs to.
- If you are writing an MDB, you need to add a dependency in `openejb-jar.xml` to the JMS resource group which the queue belongs to.

For example, to use the JMS resource group `console.jms/myJmsResourceGroup/1.0/rar` you just created in Community Edition, you could add the dependency in the application deployment plan as shown in *Listing 6.1*.

```
<sys:dependencies>
    <sys:dependency>
        <sys:groupId>console.jms</sys:groupId>
        <sys:artifactId>myJmsResourceGroup</sys:artifactId>
        <sys:version>1.0</sys:version>
        <sys:type>rar</sys:type>
    </sys:dependency>
</sys:dependencies>
```

Listing 6.1 - Defining dependencies to the JMS resource group in geronimo-web.xml or openejb-jar.xml

6.4.1 Accessing queues or topics from a Web application

Developers specify resource references and resource environment references to the connection factories, queues, and topics used in their JSPs and Servlets.

For a Web application that sends messages to `myQueue`, you still need to define the resource referenced in `web.xml` of the Web application. *Listing 6.2* shows `web.xml` with resource references.

```
<resource-ref>
    <res-ref-name>myConnectionFactory</res-ref-name>
    <res-type>javax.jms.ConnectionFactory</res-type>
    <res-auth>Container</res-auth>
    <res-sharing-scope>Shareable</res-sharing-scope>
</resource-ref>
<message-destination-ref>
    <message-destination-ref-name>jms/myQueue</message-destination-ref-name>
    <message-destination-type>javax.jms.Queue</message-destination-type>
    <message-destination-usage>Produces</message-destination-usage>
    <message-destination-link>myQueue</message-destination-link>
</message-destination-ref>
```

Listing 6.2 - web.xml with resource references

In the following code examples, `myConnectionFactory` and `jms/myQueue` are logical names used in the application. The following code snippet shows how to access the queue with JNDI look up.

```
Context jndiContext = new InitialContext();
ConnectionFactory connectionFactory = (ConnectionFactory)jndiContext
    .lookup("java:comp/env/myConnectionFactory");
```

```
Destination destination =
(Queue) jndiContext.lookup( "java:comp/env/jms/myQueue" );
```

The following code snippet shows how to access the queue with resource injection.

```
@Resource(name = "myConnectionFactory")
private ConnectionFactory connectionFactory;

@Resource(name = "jms/myQueue")
private Destination myQueue;
```

6.4.2 Message-driven beans

The `onMessage()` method of an MDB is triggered when a message comes to a specific queue or topic. For this to happen, the deployer has to perform the following:

1. Define a dependency to the JMS resource group in `openejb-jar.xml` just like what you did in *Listing 6.1*
2. Specify the JMS resource link in the `openejb-jar.xml` as shown in *Listing 6.3*.

```
<enterprise-beans>
    <message-driven>
        <ejb-name>GreetingMDB</ejb-name>
        <resource-adapter>
            <resource-link>myJmsResourceGroup</resource-link>
        </resource-adapter>
    </message-driven>
</enterprise-beans>
```

Listing 6.3 – Define JMS resource link for MDB.

3. Register `myQueue` as the destination which MDB is listening to as shown in *Listing 6.4*.

```
@MessageDriven(activationConfig = {
    @ActivationConfigProperty
    (propertyName = "destinationType", propertyValue =
    "javax.jms.Queue"),
    @ActivationConfigProperty
    (propertyName = "destination", propertyValue = "myQueue")
})

public class GreetingMDB implements MessageListener {
    public void onMessage(Message message) {
        TextMessage textMessage = (TextMessage) message;
        try {
            System.out.println("Received message: "+
```

```
textMessage.getText());  
    } catch (JMSException e) {  
        e.printStackTrace();  
    }  
}  
}
```

Listing 6.4 – Sample MDB that uses JMS Destination

6.4.3 Stand-alone Java application

With message provider specific APIs, you can access JMS resources in stand-alone Java applications. For example, you can use [ActiveMQ API](#) [ActiveMQConnectionFactory](#) to access JMS resources in Community Edition.

To access JMS resources in Community Edition using **ActiveMQConnectionFactory**, you need to add following packages under <WASCE_HOME>/repository/ in the classpath:

```
org/apache/activemq/activeio-core/3.0.1/activeio-core-3.0.1.jar  
org/apache/activemq/activemq-core/4.1.2/activemq-core-4.1.2.jar  
org/apache/geronimo/specs/geronimo-jms_1.1_spec/1.1.1/geronimo-jms_1.1_spec-1.1.1.jar  
backport-util-concurrent/backport-util-concurrent/2.2/backport-util-concurrent-2.2.jar  
commons-logging/commons-logging/1.0.4/commons-logging-1.0.4.jar  
org/apache/geronimo/specs/geronimo-j2ee-management_1.1_spec/1.0.1/geronimo-j2ee-management_1.1_spec-1.0.1.jar
```

After that, you can use `ActiveMQConnectionFactory` in your stand-alone Java application. The code snippet in *Listing 6.5* shows how to use the JMS resources in a stand-alone Java application.

```
ActiveMQConnectionFactory connectionFactory = new  
ActiveMQConnectionFactory("tcp://localhost:61616");  
Connection connection = connectionFactory.createConnection();  
Session session =  
connection.createSession(false,Session.AUTO_ACKNOWLEDGE);  
Destination topic = session.createQueue("myTopic");
```

Listing 6.5 – Use JMS Resources in stand-alone Java application

6.5 Summary

In this chapter we discussed the typical steps to develop JMS applications. We also described how to configure a JMS resource group and how to develop JMS applications with the created JMS resource in Community Edition.

6.6 Exercises

So far, you have learned how to create JMS resources and how to access them in different applications. In this exercise, you will create your own JMS resources group deployment plan in Community Edition, and then deploy it to Community Edition.

Part 1 - Create and save a JMS resource group deployment plan.

Procedure

- Follow the steps as described in *Section 6.3* to create a JMS resource group including your **ConnectionFactory**, **Queue**, and **Topic** until step 4 in *Section 6.3.3*. You should see a page like the one shown in *Figure 6.15*.

JMS Resources

JMS Resource Group -- Current Progress

These are the connection factories and destinations you've added to the JMS resource group so far. When you're finished adding connection factories and destinations, you can review the Geronimo deployment plan for this resource group, or go ahead and deploy it.

Type	Name	Interface
Connection Factory	myConnectionFactory	javax.jms.ConnectionFactory
Destination	myQueue	javax.jms.Queue
Destination	myTopic	javax.jms.Topic

Add Connection Factory Add Destination Show Plan Deploy Now

[Cancel](#)

Figure 6.15 – Show JMS Resource deployment plan.

- Click *Show Plan* instead of *Deploy Now* button to show the created deployment plan for the JMS resource group. This is illustrated in *Figure 6.16*.

JMS Resources

JMS Resource Group -- Show Deployment Plan

Deployment Plan:

```
<?xml version="1.0" encoding="UTF-8"?>
<connector xmlns="http://geronimo.apache.org/xml/ns/j2ee/connector-1.2">
    <dep:environment xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2">
        <dep:moduleId>
            <dep:groupId>console.jms</dep:groupId>
        <dep:artifactId>myJmsResourceGroup</dep:artifactId>
        <dep:version>1.0</dep:version>
        <dep:type>rar</dep:type>
        </dep:moduleId>
        <dep:dependencies>
            <dep:dependency>
                <dep:groupId>org.apache.geronimo.configs</dep:groupId>
                    <dep:artifactId>activemq-broker</dep:artifactId>
```

Copy created deployment plan content in this edit box to a xml file.

Figure 6.16 – JMS Resource deployment plan edit box.

3. Copy the deployment plan content in the edit box into an xml file. With the saved deployment plan, you can deploy a JMS resource group into Community Edition directly whenever you want, without repeating the wizard.

Part 2 - Deploy the saved JMS resource group deployment plan into Community Edition.

Procedure

1. Start the Community Edition server and launch the Community Edition administrative console. Login as *system* with password *manager*.
2. Follow the steps in *Section 6.3.4* to uninstall the existing JMS resource group to make sure there is no deployed JMS resource group which has the same ID with your deployment plan.
3. In the left hand navigation pane, select *Applications* -> *Deploy New* to open the *Install New Applications* page as shown in *Figure 6.17*.

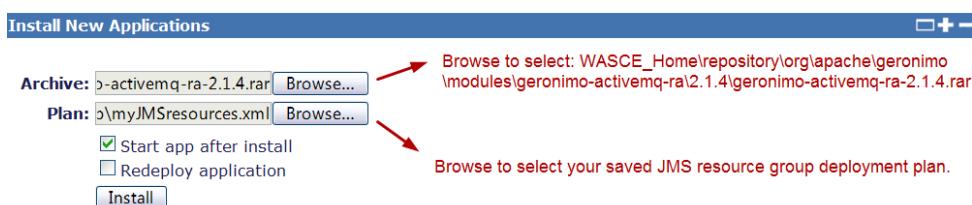


Figure 6.17 – Deploy JMS Resource from existing deployment plan.

4. Click *Browse* at the right side of *Archive*; select
`<WASCE_HOME>\repository\org\apache\geronimo\modules\geronimo-activemq-ra\2.1.4\geronimo-activemq-ra-2.1.4.rar`
5. Click *Browse* at the right side of *Plan*; select your saved JMS resource group deployment plan xml file.
6. Click *Install* to deploy.

6.7 Review questions

1. What's the major difference between HTTP and message applications?
2. What are the two popular JMS application models?
3. What are the steps to develop a JMS client application that produces a message?
4. What do you have to do before developing JMS applications with Community Edition?
5. How do you use a created JMS resource group in a Web application?

6. Which of the following is the typical destination type in a subscription/publish JMS application model?
 - A. Queue
 - B. Destination
 - C. Topic
 - D. Receiver
 - E. None of the above
7. Which of the following message products is the default JMS provider in Community Edition?
 - A. ActiveMQ
 - B. Websphere MQ
 - C. OPENJMS
 - D. JORAM
 - E. None of the above
8. Which of the following APIs represents a JMS Client that sends message to a JMS destination?
 - A. MessageConsumer
 - B. MessageProducer
 - C. Topic
 - D. Connection
 - E. Session
9. Which of the following APIs represents a JMS Client that receives a message in a JMS destination?
 - A. MessageConsumer
 - B. MessageProducer
 - C. Queue
 - D. Connection
 - E. Session
10. Which of the following interface must be implemented by a MDB?
 - A. Destination
 - B. Topic
 - C. Message

D. MessageListener

E. Queue

7

Chapter 7 – Web Services

Web Services are software services that allow for the interaction between different businesses over the internet. Web services are designed to allow for communication between machines in a loosely coupled fashion. This can be accomplished by use of a **Web Services Description Language (WSDL)** XML document that provides the description required by the invoker to call the service and to understand the XML messages returned by the service.

This chapter describes Community Edition's implementation of Web Services, and showcases IBM Data Studio with DB2 Express-C to easily create Data Web services that can be deployed in both SOAP and REST style to Community Edition using a "drag and drop" approach.

In this chapter you will learn about:

- How to develop Web Services in Community Edition
- How to generate stub classes using the tools of Community Edition
- How to consume the Web Services provided by Community Edition
- How to develop Data Web Services using IBM Data Studio and DB2 using drag and drop

7.1 Community Edition Web Services: The big picture

Figure 7.1 provides an overview of Web Services support in Community Edition.

On the server side where Community Edition is installed, developers can create and deploy a Web service to Community Edition as shown in (1). Then users can get the Web Service Description Language (WSDL) of the Web service to generate stubs as shown in (2), which will be used by the client.

On the left side you can see there are two types of clients invoking the Web Services deployed in Community Edition:

- A POJO client, as shown in (3), that consumes Web Services from a stand alone Java application

- A Web client, as shown in (4), that consumes Web Services in a Web application deployed in Community Edition

As a part of the Java EE specification, users can also use the service reference tag to import a Web service to the Web client. This is shown in (5).

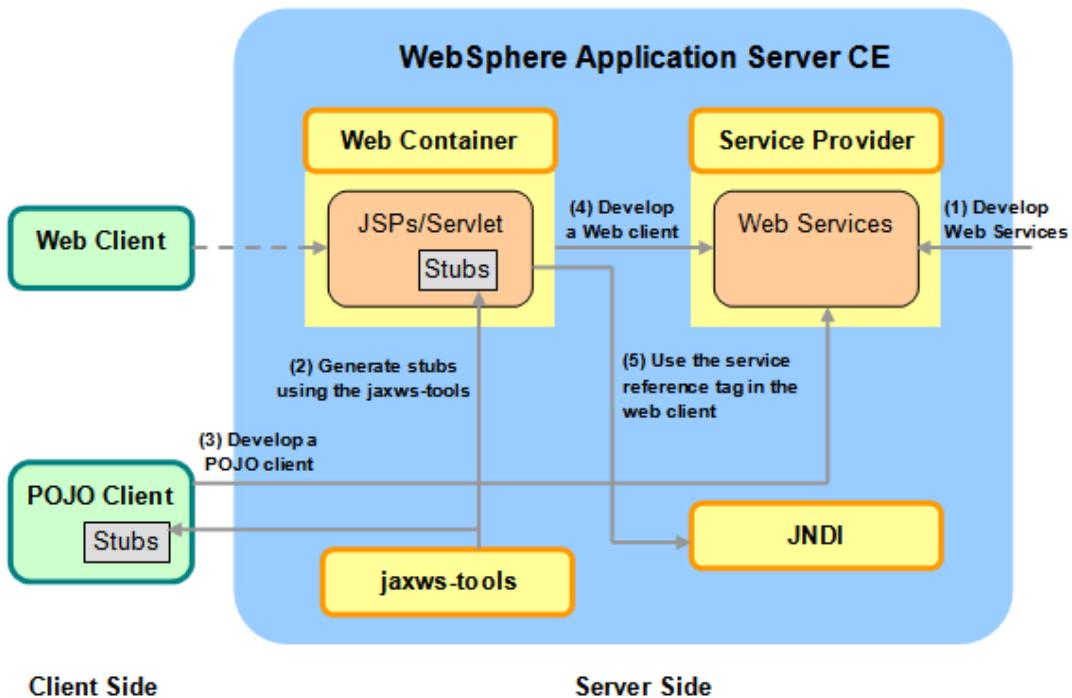


Figure 7.1 - Community Edition Web services: The big picture

Not shown in the figure is how to work with IBM Data Studio and DB2 to create Data Web services in minutes. This is discussed in section 7.4.

7.2 Developing Web Services in Community Edition

Follow these steps to develop Web Services in Community Edition:

1. Create a Dynamic Web project to host the Web Services classes in Eclipse.
2. Create a **Service Endpoint Interface (SEI)**
3. Provide the SEI service implementation class
4. Declare the SEI implementation class in the `web.xml` file and deploy the Web project into Community Edition.

Each of these steps will be discussed in more detail in the following sections.

7.2.1 Creating a Web Service project in Eclipse

As discussed in earlier sections, you can create a Dynamic Web Project to host Web Services classes as follows:

1. From Eclipse main menu, select *File -> New -> Dynamic Web Project*
2. In the *New Dynamic Web Project* dialog, input the Project Name, such as **HelloServiceProject**, and click *Next*
3. Keep the default values on the following pages and click *Finish*.

Note:

We do not recommend you to use the Axis2's Web Services wizard provided by the Eclipse Web Tools Platform (WTP) to create a Web Services project.

7.2.2 Creating a service endpoint interface

Service endpoint interface (SEI) is a normal Java interface, which is used to describe what a Web service can do. To develop a SEI, you need to define an interface and annotate it with the `@WebService` annotation. In this section, you will use the **Java First Development**, this means that the WSDL document is not available and therefore you should not provide the `wsdlLocation` property; otherwise you will receive a WSDL *file not found* exception. This is illustrated in *Listing 7.1* below.

```
package com.ibm.wasce.samples;
import javax.jws.WebService;
@WebService (name="HelloPeople",
            targetNamespace="http://samples.wasce.ibm.com")
public interface HelloPeople {
    public String sayHello(String targetName);
}
```

Listing 7.1 – SEI: HelloPeople.java

Though in the above listing the `@WebService` annotation properties `name` and `targetNamespace` have some values, it is fine to provide the `@WebService` annotation without any properties to use defaults.

7.2.3 Providing the service implementation class

After creating a SEI, you need to provide a Java bean to implement it. You have to designate the `endpointInterface` property when the service class is implementing a SEI as shown in *Listing 7.2*.

```
package com.ibm.wasce.samples;
import javax.jws.WebService;
@WebService(serviceName = "HelloBillService",
```

```
        portName = "HelloBillPort",
        endpointInterface = "com.ibm.wasce.samples.HelloPeople",
        targetNamespace = "http://samples.wasce.ibm.com")
public class HelloBill implements HelloPeople {
    public String sayHello(String targetName) {
        return "Hello " + targetName;
    }
}
```

Listing 7.2 – SEI's implementation: HelloBill.java

The `@WebService` annotation properties `serviceName`, `portName`, and `targetNamespace` shown in the above listing are sample values. It is fine to provide the `@WebService` annotation only with values for the `endpointInterface` property and use defaults for the rest.

7.2.4 Deploying the Web Service in Community Edition

To publish a Web Service in Community Edition, you need to add some information under the `<servlet>` and `<servlet-mapping>` elements in the Web project's WEB-INF/web.xml file. However, this does not mean that the service class must be a Servlet, which must extend the HttpServlet. The Web service is never a real servlet; the deployer of Community Edition can distinguish whether or not it is a Web service implementation class. Listing 7.3 provides an example of the contents of web.xml.

```
<servlet>
    <servlet-name>Hello</servlet-name>
    <servlet-class>com.ibm.wasce.samples.HelloBill</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>Hello</servlet-name>
    <url-pattern>/Hello</url-pattern>
</servlet-mapping>
```

Listing 7.3 - The web.xml file

The `<url-pattern>` element defines how to invoke the Web service from a browser and also the path to retrieve its WSDL file.

After deploying the Web project to Community Edition, launch a browser and go to the following URL: <http://localhost:8080>HelloServiceProject>Hello>

You should see a message from the Axis2 engine as shown in Figure 7.2 below.



Figure 7.2 – Message from the Axis2 engine

And you can get the WSDL file generated by Geronimo from the following location:
<http://localhost:8080/HelloServiceProject>Hello?wsdl>

7.3 Consuming a Web Service deployed in Community Edition

To test and invoke (consume) a Web service from a client, follow these steps:

1. Review the WSDL from a browser, and use jaxws-tools to generate stub classes.
2. Import stubs into your client application.
3. Construct a Service object through the stubs, and get the target port object from the service; then, invoke the methods on the port object.

This section will show you how to create a POJO client and a Web-based client for a Web service. You only need the WSDL file of the Web service deployed for creating the client.

7.3.1 Creating necessary stubs to consume the Web Service

You need to create a Service Endpoint Interface based on the WSDL document by using the JAX-WS tool `wsimport`. The syntax to use this tool is:

```
wsimport [options] <WSDL_URI>
```

Common options of `wsimport` are shown in *Table 7.1*:

Options	Description
-b <path>	specify external jaxws or jaxb binding files
-d <directory>	specify where to place generated class files
-s <directory>	specify where to place generated source files
-p <pkg>	specifies the target package
-verbose	output messages about what the compiler is doing
-keep	keep generated files
-help	display help
-version	print version information

Table 7.1 - Common options of wsimport

Community Edition includes a built-in plug-in named **jaxws-tools** which provides tools for generating WSDL and other necessary files used in JAX-WS web services deployment. The plug-in relies on Sun's **wsgen** and **wsimport** tools to generate the Web services artifacts. Follow the steps below to use this tool:

1. Open a command prompt and change the directory to the **bin** folder under the Community Edition installation directory
2. Use the following command to generate the necessary stubs that are needed to consume the Web service:

```
jaxws-tools.bat wsimport -s <OUTPUT_DIR> <WSDL_FILE>
```

For example:

```
jaxws-tools.bat wsimport -verbose -s c:\stubs
```

```
http://localhost:8080/HelloServiceProject>Hello?wsdl
```

Then the following messages will appear as shown in *Listing 7.4*

```
Using GERONIMO_HOME: /opt/IBM/wasce-server
Using GERONIMO_TMPDIR: var/temp
Using JRE_HOME: /usr/java/jdk1.5.0_15/jre
com/ibm/wasce/samples/HelloBillService.java
com/ibm/wasce/samples/HelloPeople.java
com/ibm/wasce/samples/ObjectFactory.java
com/ibm/wasce/samples/SayHello.java
com/ibm/wasce/samples/SayHelloResponse.java
```

```
com/ibm/wasce/samples/package-info.java
```

Listing 7.4 – Output stubs generated by wsimport

3. All the stubs should have been created and placed in c:\stubs directory. You now need to create the class **HelloBillService** and interface **HelloPeople** for accessing the Web Service:
 - HelloBillService.java - This file extends the **javax.xml.ws.Service** and provides a method to create a service and retrieve the port.
 - HelloPeople.java - This is the Service Endpoint Interface that is required to access the Web Service.

7.3.2 Creating a POJO client

To create a POJO client, follow these steps:

1. Create a general Java Project in Eclipse
In the *File* menu, click *New -> Java Project*.
2. Import the stubs to the project
In Eclipse, right click the *src* folder of the client project in the project explorer and select *Import -> General: File System*. A window will open to let you select the stubs folder and help you import the stubs to the client project automatically. This is shown in *Figure 7.3* below.

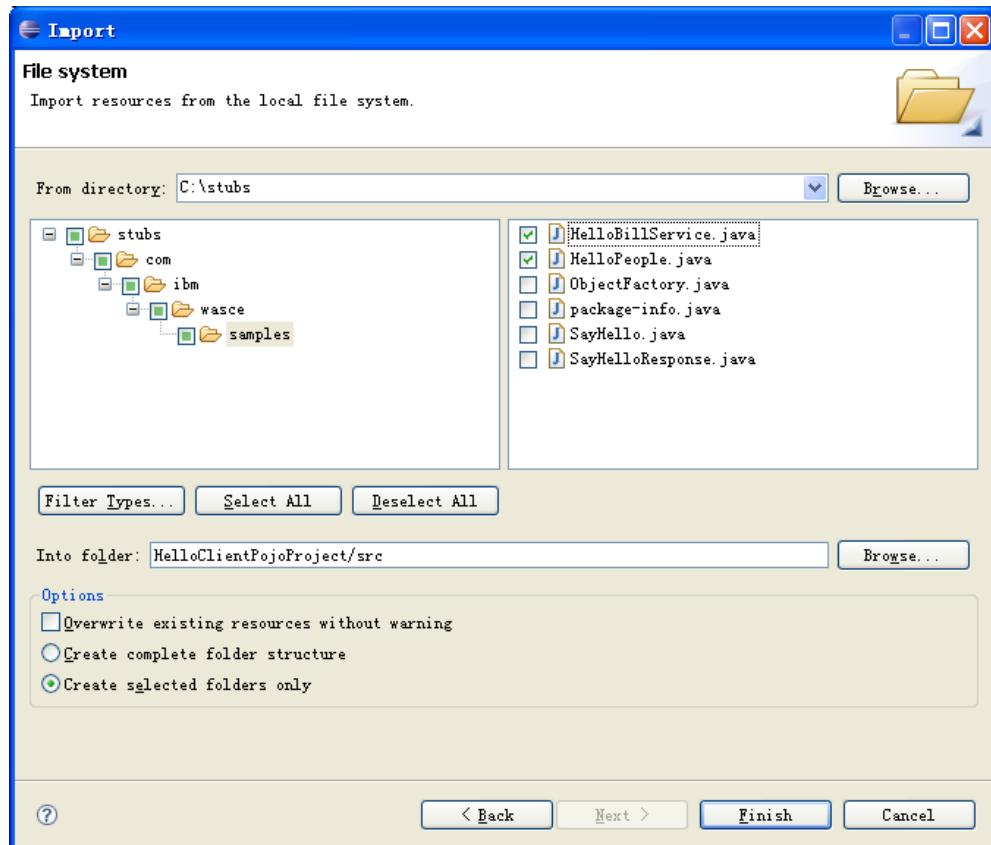


Figure 7.3 - Import the stubs to the project

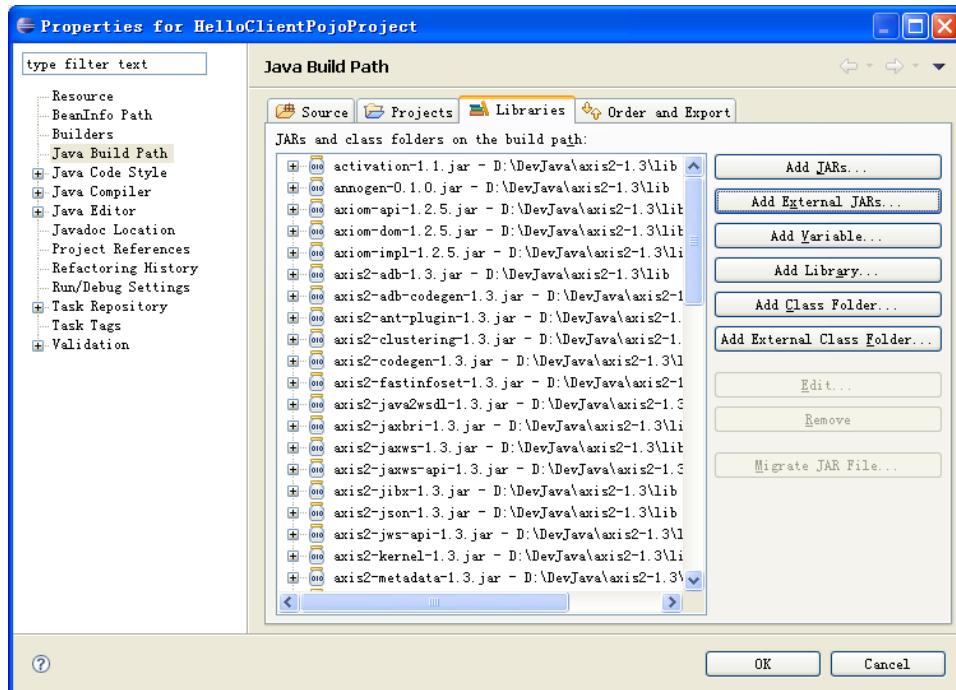
3. Resolve class path errors

At this point the client project may report errors. This is due to missing jar files required to access and understand SOAP messages sent by the Web service.

To resolve this problem, since you used the jaxws-tools of Community Edition V2.1.1.3 to generate the stubs, then the Community Edition version should also adopt the axis2 v1.3 as the JAX-WS v2.0 Engine. Download axis2 v1.3 from:
http://archive.apache.org/dist/ws/axis2/1_3/axis2-1.3-bin.zip

Then, in Eclipse:

- Right click the client project and then select *Properties*
- Select *Java Build Path* and then go to *Libraries* tab
- Add all the external jars underneath the `axis2-1.3/lib` folder to the build path of your project. This is illustrated in *Figure 7.4* below.

**Figure 7.4 - Resolve class path errors**

4. Developing the project

Right click the package **com.ibm.wasce.clients** and select *New -> Class* to create a new java class named **PojoClient**. Then paste the code shown in *Listing 7.5*.

```
package com.ibm.wasce.clients;
import com.ibm.wasce.samples.*;
public class PojoClient {
    public static void main(String args[]) {
        try {
            HelloBillService service = new HelloBillService();
            HelloPeople bill = service.getHelloBillPort();
            System.out.println(bill.sayHello("Rex"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Listing 7.5 – A POJO Client

5. Run the project

Right click on the project and select *Run As -> Java Application*. You will get the message in the console as shown in *Figure 7.6*

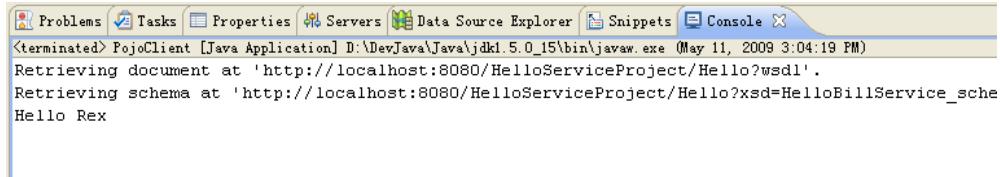


Figure 7.6 – Output when you run the POJO client

7.3.3 Creating a Web client

To create a Web client, follow these steps:

1. Create a Dynamic Web Project.

In *File* menu, click *New -> Other*. From the popup dialog, select *Web: Dynamic Web Project*.

2. Add stubs to the project

Manually copy the stubs files and place them in the appropriate folder according to their package declaration to the `src` directory underneath the client project, then refresh the project in Eclipse. This is the same as using the import menu introduced in the previous section.

3. Develop the project

- Create a jsp file named `index.jsp` and add the content provided in *Listing 7.6*:

```
<h3>This form invokes a Web Service.</h3><br />
Please type a name you want to say hello.
<form method="post" action="DoSayHello">
    Name: <input type="text" name="target">
    <input type="submit" value="Submit">
</form>
```

Listing 7.6 – Web client's index.jsp

- Create a servlet named `DoSayHello` underneath the `com.ibm.wasce.clients` package.

Add the following code illustrated in *Listing 7.7* to its `doPost` method:

```
protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
    PrintWriter out = response.getWriter();
    String target = request.getParameter("target");
```

```

HelloBillService service = new HelloBillService();
HelloPeople bill = service.getHelloBillPort();
out.println(bill.sayHello(target));
}

```

Listing 7.7 – The Servlet to consume Web Services

- Resolve the type errors by adding:

```

import com.ibm.wasce.samples.*;
import java.io.*;

```

4. Deploy and run the project in Community Edition

- Right click the project and select *Run As -> Run on Server*
- In the popup, select Community Edition as the server runtime and check the check box *Always use this server when running the project* and then click *Finish*

If the server is stopped, Community Edition will start automatically and deploy the project to run. Wait for some time until the server status changes to *Synchronized*, then a welcome page will appear as shown in *Figure 7.7*



Figure 7.7 – Web client's index page

- Input *Rex* in the text box and press *Submit*. You should see the result as shown in *Figure 7.8*:



Figure 7.8 – Response after calling the Web service from a servlet

7.3.4 Using service reference tag in a Java EE application

In the previous servlet, you accessed the Web Service by instantiating the Service class manually. This method is not optimal because each time you process the servlet, you have to create a new service instance. A better alternative is to use **JNDI** to directly access your Web service without instantiating any Service class. This method requires you to add a **service-ref** element to web.xml to provide the necessary information about the WSDL file location, Service name, etc. *Listing 7.8* provides and example of what you need to add to web.xml.

```
<service-ref>
    <service-ref-name>services/Hello</service-ref-name>
    <service-interface>
        com.ibm.wasce.samples.HelloBillService
    </service-interface>
    <wsdl-file>
        http://localhost:8080/HelloServiceProject>Hello?wsdl
    </wsdl-file>
</service-ref>
```

Listing 7.8 – The service reference tag

Let's go through the elements that were added to web.xml:

1. **service-ref-name** - This is the name by which JNDI identifies your deployed service. These services are located at `java:comp/env/<service-ref-name>`
2. **service-interface** - This is the interface which a client uses to access the Web service and create necessary stubs under the covers. This interface must extend either to `javax.xml.ws.Service` or `javax.xml.rpc.Service`. Note that this is not the Service-Endpoint Interface that we used to deploy the service.
3. **wsdl-file** - The location of the WSDL file of the deployed web service.

The changes required to access the service are shown in *Listing 7.9*

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    String target = request.getParameter("target");
    try {
        Context ctx = new InitialContext();
        HelloBillService billService =
            (HelloBillService)ctx.lookup("java:comp/env/services/Hello");
        HelloPeople bill = billService.getHelloBillPort();
        out.println(bill.sayHello(target));
    } catch (NamingException e) {
```

```

        e.printStackTrace();
    }
}

```

Listing 7.9 – Get Web service from Web application context

To resolve the type errors add this line:

```
import javax.naming.*;
```

You can also use annotation to get the Web service resource into the servlet. Add the lines below in the Servlet class definition:

```
@Resource(name="services/Hello")
Service service;
```

Then change the *doPost* method as shown in *Listing 7.10*:

```
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    String target = request.getParameter("target");
    HelloBillService billService = (HelloBillService)service;
    HelloPeople bill = billService.getHelloBillPort();
    out.println(bill.sayHello(target));
}
```

Listing 7.10 – Changes in Servlet after using @Resource annotation

To resolve the type errors add these lines:

```
import javax.annotation.Resource;
import javax.xml.ws.Service;
```

7.4 Creating Data Web services with IBM Data Studio

Data Web Services refer to the ability to wrap Web services around the logic provided by the database. For example, you might already have a SQL script or stored procedure that provides business logic for returning the current price of a particular item in inventory from the database. Using Data Web Services, you are simply making it much easier for a Web application (or other client) to invoke that capability, perhaps even as simple as putting the HTTP request in a Web browser.

This approach to creating a Web service based on existing database operations/business logic is called “bottoms up” as opposed to a “top down” approach in which the Web services description is defined first and then logic is provided to map to that particular description.

IBM Data Studio supports the development and deployment of Data Web Services without you having to write a single line of code. *Figure 7.9* provides an overview of data Web services using Data Studio.

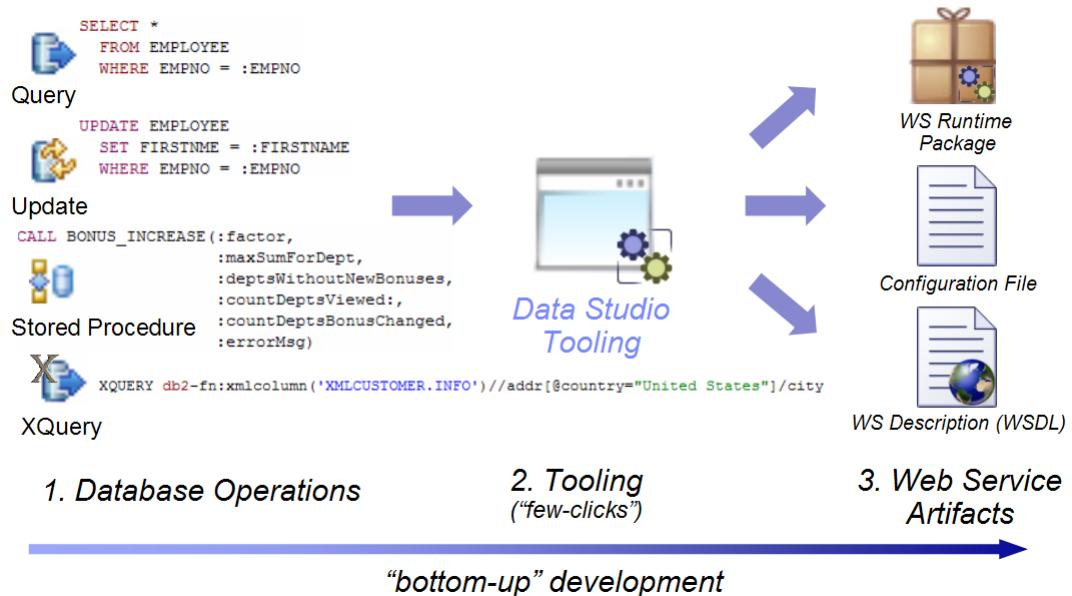


Figure 7.9 - Developing data Web services with IBM Data Studio

On the left side of the figure, you can see different database operations. For example, there is a query to return all information about an employee when an employee number is provided. There is an update statement to update the first name of an employee based on an employee number; there is a stored procedure that does some bonus calculations, and there is an XQuery that is retrieving information from an XML document. Using Data Studio, these operations can be converted to data Web services without any coding on your part. Dragging and dropping and a few clicks are all you need to have the Data Web Service created for you. On the right side of the figure, you can see that Data Studio automatically creates the artifacts needed to deploy this Web service, including the WSDL document, and the Java EE runtime artifacts such as a configuration file and the runtime package.

Note:

To see Data Web Services in action, take a look at this video:
<http://www.channeldb2.com/video/807741:Video:1482>

For more information about Data Web Services, refer to the free eBook *Getting started with IBM Data Studio for DB2*, which is part of the DB2 on Campus book series.

7.5 Summary

In this chapter, you learned how to develop a Web service in Community Edition using the Web Eclipse plug-in. This included the development of two types of client applications, a Web client, and a POJO client to consume the Web service. Later in the chapter, you learned how to create Data Web services using IBM Data Studio and DB2. Data Web services are Web services based on logic around your database such as database scripts or stored procedures. Using "drag and drop" you can create data Web services in minutes.

7.6 Exercises

In this exercise, you will create your own Web Service and test it using the Eclipse Web Services Explorer.

Part 1 - Creating a Web Service

Procedure

1. Follow the steps in *Section 7.2* to create a Web Service and deploy it to Community Edition.

2. Test the Web Service.

- From Eclipse's main menu, click *Run->Launch the Web Services Explorer*. On the top right corner of this explorer, click on the icon for *WSDL Page*.
- Under the *Navigation* frame, you should see *WSDL Main* (instead of *UDDI Main*). Click that *WSDL Main* link and the *Actions* frame should show a form for the WSDL URL.
- Enter the WSDL URL <http://localhost:8080/HelloServiceProject>Hello?wsdl> in the form and click *Go*. You can now see the methods that are exposed by your Web Service.
- Clicking on the *sayHello* method will take you to a form asking for input parameters. If you don't see any field to enter input arguments click *Add*.
- Enter the input arguments and click *Go*. You can see the output of the Web service in the *Status* frame at the bottom as shown in *Figure*

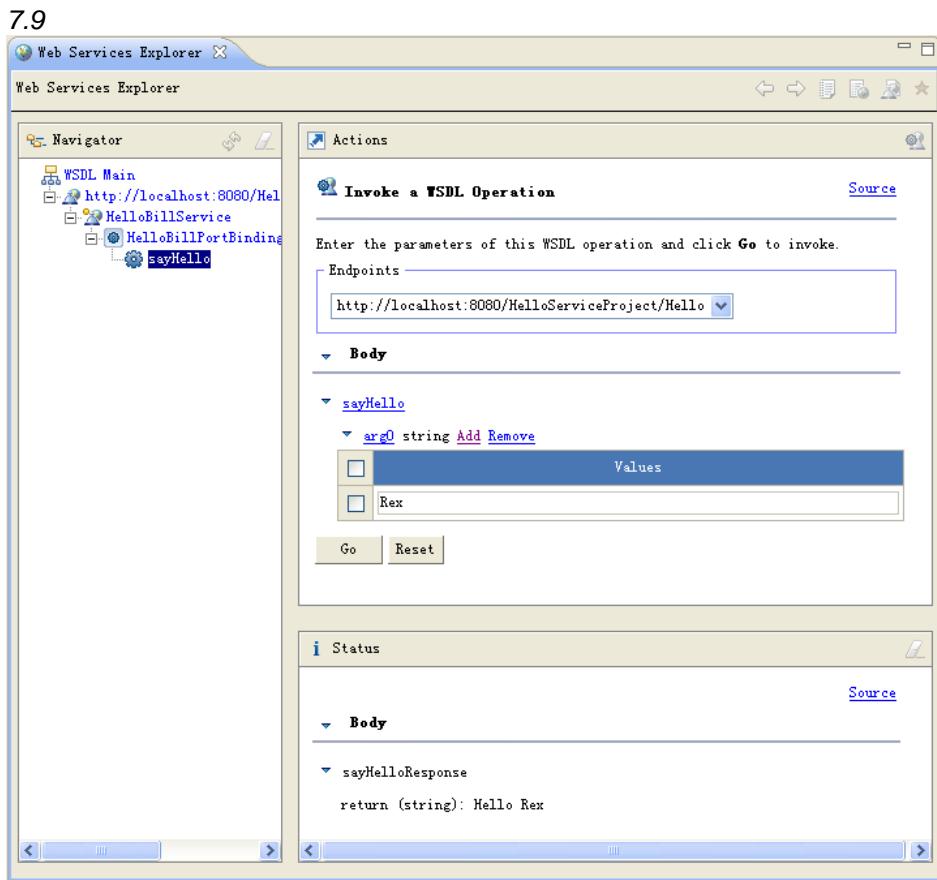


Figure 7.9 – Test results in Web Services Explorer of Eclipse

Part 2 - Creating a Data Web Service

Procedure

After reviewing the video mentioned in section 7.4:

Create a project in IBM Data Studio connecting to the SAMPLE database in DB2

1. Create a script using the following statement: `select * from staff`
2. Create a SOAP and REST Web service using the above script, deploy it to Community Edition and test it

7.7 Review questions

1. Is there any difference between developing a service endpoint on Community Edition and on other application servers?
2. How do you generate the stubs that are used in a Web client to consume Web services?

3. How do you resolve class path errors after importing the stubs to the POJO Client project?
4. How do you declare a Web Service in its project's web.xml before you deploy it into Community Edition?
5. What is a *Data* Web service?
6. Which type of Eclipse projects can be used to host a Web Service?
 - A. Java Project
 - B. Plug-in Project
 - C. Dynamic Web Project
 - D. Static Web Project
 - E. None of the above
7. Which annotation can be used to declare a Web Service definition in SEI?
 - A. @WebService
 - B. @Remote
 - C. @Service
 - D. @Resource
 - E. All of the above
8. Which annotation is used to get the Web Service when using *service-ref* element in web.xml?
 - A. @WebService
 - B. @Remote
 - C. @Service
 - D. @Resource
 - E. All of the above
9. Business logic for a Data Web Service can be provided by:
 - A. SQL procedures
 - B. XQuery statements
 - C. SQL statements
 - D. All of the above
 - E. None of the above
10. You create a new Data Web Service in:
 - A. A Data Design project

- B. A Data Development project
- C. The Data Source Explorer
- D. SQL and XQuery Editor
- E. None of the above

8

Chapter 8 – Security

Any Java EE application server that utilizes role-based security requires a mechanism for authenticating users and groups. The Java EE mechanism for this type of security is a user registry. **Security Realm**, a user registry definition, containing users and groups which can map to Java EE roles used to protect resources.

In this chapter, you will learn about:

- How to configure Security Realm in Community Edition
- How to implement security in your Java EE application.
- How to take advantage of trusted contexts to secure your data and improve performance

8.1 Community Edition Security: The big picture

Figure 8.1 provides an overview of Community Edition security.

On the server side where Community Edition is installed, developers configure the security realms for the Java EE applications as shown in (1) and define the role mapping in their deployment plans as shown in (2).

On the left side a POJO client can access a secured EJB application as shown in (3) and a Web client can access a secured Web application as shown in (4).

From the admin console, you can also manage Community Edition administrative security as shown in (5).

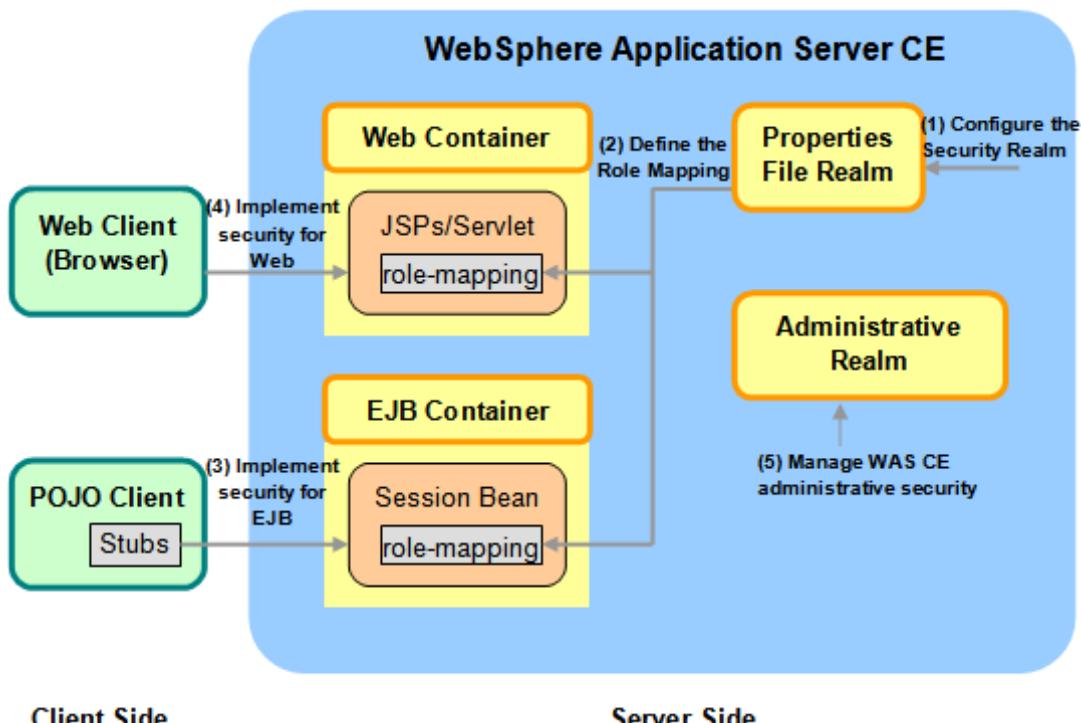


Figure 8.1 - Community Edition security: The big picture

Use the following steps to secure a Java EE application in Community Edition:

1. Configure the security realm in the Administrative Console.
2. Define the role mapping in Geronimo deployment plan.
3. Restrict roles to access Java EE resources either by annotations or by Java EE deployment descriptors

Each of these steps is discussed in more detail in the next sections. Not shown in the figure is the use of trusted contexts to protect your data and improve performance on three-tier applications. This is discussed in section 8.5

8.2 Configuring the security realm

Security realms can be defined as standalone or as part of an individual application. In Community Edition, you can configure the Security Realm from the Administrative Console following steps below:

1. Start Community Edition, open the Administrative Console and go to the Security Realms portlet. Then click *Add new security realm*.
2. Type the realm name you want, such as `helloRealm` and for the Realm Type select *Properties File Realm* as illustrated in *Figure 8.2*. Then click *Next*.

Security Realms

Create Security Realm -- Step 1: Select Name and Type

Name of Security Realm: helloRealm

Realm Type: Properties File Realm

A name that is different than the name for any other security realms in the service (no spaces in the name please). Other components will use this name to refer to this realm.

The type of login module used as the master for this security realm. Select "Other" for manual configuration options including custom login modules and realms that use login modules to populate user principals.

[Next](#)

[Cancel](#)

Figure 8.2 – Create a security realm

The Community Edition admin console provides four predefined realm types and their wizards such as: *Properties File Realm*, *LDAP Realm*, *Database (SQL) Realm* and *Certificate Properties File Realm*. Each can map to one physical method to store user IDs, passwords and group information. For more information, review the Community Edition manuals at <http://publib.boulder.ibm.com/wasce/V2.1.1/en/security-realms.html>

Note:

Using the LDAP realm, groups and users are defined in an LDAP server; this is the most common choice in a production environment. For simplicity, we use the Properties File Realm.

3. In the *Create Security Realm - Step 2* panel input the location of *Users File* and *Groups File*. First, let's create two files as illustrated in *Listing 8.1* and *8.2* below.

```
andy=aaa
bill=bbb
cindy=ccc
dan=ddd
```

Listing 8.1 - hello_users.properties

```
employee=andy,cindy,dan
manager=bill
```

Listing 8.2 - hello_groups.properties

In *Listing 8.1*, you define the user's id and password pairs. In *Listing 8.2* you define the group and user pairs. If there are many users that belong to one group, separate each user id with a comma.

Secondly, copy the above files into the <WASCE_HOME>/var/security folder. Then input the file names including the relative path as shown in *Figure 8.3*

Security Realms

Create Security Realm -- Step 2: Configure Login Module

Users File URI: The location of a properties file (relative to the Geronimo home dir) holding user information. The format of each line should be `username=password`.

Groups File URI: The location of a properties file (relative to the Geronimo home dir) holding group information. The format of each line should be `group=user, user, ...`

Digest Algorithm: Message Digest algorithm (e.g. MD5, SHA1, etc.) used on the passwords. Leave empty if no digest algorithm is used.

Digest Encoding: Encoding to use for digests (e.g. hex, base64). This is used only if a Message Digest algorithm is specified. If no encoding is specified, hex will be used.

Next **Cancel**

Figure 8.3 – Provide the property files for users and groups

The *Digest Algorithm* field indicates the type of encryption that will be used to create the binary representation of the password, such as MD5 or SHA. The *Digest Encoding* field indicates the text format that will be used to store the binary representation. The two supported formats are HEX and BASE64. Leave both of them empty for this book.

4. Then you can choose *Skip Test and Deploy*. The new security realm is now visible in the Security Realms portlet in the Community Edition console as illustrated in *Figure 8.4*.

Name	Deployed As	
geronimo-admin	Server-wide	Edit usa
helloRealm	Server-wide	Edit usa

Figure 8.4 – New security realm: helloRealm

8.3 Implementing security in a Java EE application

In general, access to a secured Java EE resource requires the validation of credentials (user ID/password) against the user registry. The user registry by itself does not enforce any security policies or constraints (authorization); it simply validates the supplied user, group, and password information by the Java EE authentication mechanisms.

Role-based security allows a developer to secure access to Java EE resources based on roles. Often a developer has no knowledge of the users and groups that will exist at runtime; by using roles, he can grant or revoke access to a specific artifact. Later, an administrator can map these roles to actual users and groups.

Java EE has two forms of role-based security:

- **Declarative security**, where the security rules are defined in the deployment descriptors and the Java EE runtime manages access to the Java EE artifact. A user will see all or none of a resource.
- **Programmatic security**, where the program determines the role of a user and grants access to different resources depending on this role. The user may only see a subset of a resource depending on his role.

8.3.1 Defining role mapping

The mapping of specific users and groups to roles is defined in the Community Edition deployment plan, which can either be Web, EJB, or EAR. For example, the code snippet shown in *Listing 8.3* defines that user *bill* will take the role as a *super user*, and the group *employee* will take the role of *general users*.

```
<security>
  <role-mappings>
    <role role-name="super_user">
      <principal name="bill" designated-run-as="true"
class="org.apache.geronimo.security.realm.providers.GeronimoUserPrincipal"/>
    </role>
    <role role-name="general_user">
      <principal name="employee" designated-run-as="true"
class="org.apache.geronimo.security.realm.providers.GeronimoGroupPrincipal"/>
    </role>
  </role-mappings>
</security>
```

Listing 8.3 - Role mappings

8.3.2 Implementing EJB Security

This section describes how to implement declarative and programmatic EJB security.

8.3.2.1 Declarative security

Declarative security can restrict access to EJBs through annotations in the EJB class or through deployment descriptors based on roles in the EJB application. *Listing 8.4* provides an example where annotation is added before the EJB implementation class definition to declare the roles that can access the class.

```
@DeclareRoles( { "general_user" , "super_user" } )  
public class HelloBean implements HelloWorld {  
...  
}
```

Listing 8.4 - Declare the roles before EJB class definition

Listing 8.5 below shows the lines you need to insert before the methods you want to restrict.

```
@RolesAllowed( { "general_user" , "super_user" } )  
public string sayHello(string targetName){  
...  
@RolesAllowed( { "super_user" } )  
public string sayByebye(string targetName) {  
...  
}
```

Listing 8.5 - Restrict roles before methods

This means that both, the general user and the super user can say “Hello”, but only the super user has the rights to say “Bye bye”.

The above annotation statements could be replaced by adding the following lines in EJB’s deployment descriptor (ejb-jar.xml), as shown in *Listing 8.6*.

```
<assembly-descriptor>  
  <security-role>  
    <role-name>super_user</role-name>  
  </security-role>  
  <security-role>  
    <role-name>general_user</role-name>  
  </security-role>  
  <method-permission>  
    <role-name>super_user</role-name>  
    <role-name>general_user</role-name>  
    <method>  
      <ejb-name>HelloBean</ejb-name>  
      <method-name>sayHello</method-name>  
    </method>  
  </method-permission>  
  <method-permission>  
    <role-name>super_user</role-name>  
    <method>  
      <ejb-name>HelloBean</ejb-name>  
      <method-name>sayByebye</method-name>  
    </method>  
  </method-permission>
```

```
</method-permission>
</assembly-descriptor>
```

Listing 8.6 - The equivalent deployment descriptor

The benefit of using the deployment descriptors instead of the annotations is that the security definitions can be changed at deployment time, without having to modify the source code. Deployment descriptors always supersede annotations.

8.3.2.2 Programmatic security

Programmatic security is implemented in the EJB's program logic. It references the role definitions in the Geronimo deployment plan. *Listing 8.7* shows an example where you define the roles before the EJB implementation class definition.

```
@DeclareRoles( { "super_user", "general_user" })
public class HelloBean implements HelloWorld {
```

...

Listing 8.7 - Declare the roles before EJB class definition

Listing 8.8 provides an example of the class definition to insert the session context object that will be used to determine if the current user has the declared role.

```
public class HelloBean implements HelloWorld {
    @Resource
    private SessionContext ctx;
}
```

...

Listing 8.8 - Inserting a session context

Listing 8.9 provides an example of the logic required to perform access control in your program.

```
System.out.println(ctx.getCallerPrincipal().getName());
if (ctx.isCallerInRole("super_user")){
    //add any logic that is super user specific.
}
```

Listing 8.9 - Programmatic access control

8.3.2.3 Testing EJB security in a standalone client

To test EJB security in a standalone client, you first need to add the `geronimo-security-2.1.4.jar` located under

```
<WASCE_HOME>\repository\org\apache\geronimo\framework\geronimo-
security\2.1.4\
```

into your client project's build path. This JAR file contains classes that are returned to the client when initializing to the server with a user ID identity. Without this JAR file, you will receive a `ClassNotFoundException` at runtime.

For the initial context, use the properties as shown in *Listing 8.10*.

```
Hashtable props = new Hashtable();
```

```
props.put( "java.naming.factory.initial",
           "org.openejb.client.RemoteInitialContextFactory" );
props.put(Context.SECURITY_PRINCIPAL, "andy");
props.put(Context.SECURITY_CREDENTIALS, "aaa");
props.put("openejb.authentication.realmName", "helloRealm");
InitialContext ctx = new InitialContext(props);
```

Listing 8.10 - Initial properties

8.3.3 Implementing Web security

This section describes how to implement declarative and programmatic Web security.

8.3.3.1 Declarative security

Declarative security restricts access to URLs, such as servlets, JSPs, or even HTTP files or images served by the Web container through deployment descriptors (`web.xml`) of the Web application. *Listing 8.11* provides an example where you insert in `web.xml` the roles that will access the Web project.

```
<security-role>
  <role-name>super_user</role-name>
</security-role>
```

Listing 8.11 - Declarative security referencing roles in web.xml

Listing 8.12 provides an example where you define role constraints. In the example, the super user can access the resources where the URI matches the pattern `/jsp/*` with the GET and POST methods.

Listing 8.13 shows the code needed to configure the authentication method.

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Protected Resource</web-resource-name>
    <url-pattern>/jsp/*</url-pattern>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
  </web-resource-collection>
  <auth-constraint>
    <role-name>super_user</role-name>
  </auth-constraint>
</security-constraint>
```

Listing 8.12 - Security constraint

```
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>helloRealm</realm-name>
</login-config>
```

Listing 8.13 - Configuring the authentication method

In the above example, the realm `helloRealm` is used. Ensure you add the following line to the deployment plan (`geronimo-web.xml`):

```
<security-realm-name>helloRealm</security-realm-name>
```

The **BASIC** authentication mechanism is the default option for requesting authentication information from a browser-based client. When a user tries to access the protected pages using a browser, the browser will pop-up a dialog box to request you input the id and password. The authentication information is encrypted using base64 encoding.

Alternatively, the **FORM** authentication mechanism as illustrated in *Listing 8.14* is also commonly used. Using this mechanism, an application developer provides a customized logon page, and you request the authentication information this way. This is illustrated in *Listing 8.15* showing the contents of the file `login.jsp`.

```
<login-config>
  <auth-method>FORM</auth-method>
  <form-login-config>
    <form-login-page>/login.jsp</form-login-page>
    <form-error-page>/loginerror.jsp</form-error-page>
  </form-login-config>
</login-config>
```

Listing 8.14 - “FORM” authentication method

```
<form method="POST" action="j_security_check">
  Username:<input size="12" name="j_username" type="text">
  Password:<input size="12" name="j_password" type="password">
  <input name="submit" type="submit" value="Login">
</form>
```

Listing 8.15 - The login.jsp

There are also **Client_CERT** and **Digest** methods provided, where **Client_CERT** is using digital certificates passed over an SSL connection, and **Digest** is similar to **BASIC**, but the password is transmitted using a custom encryption mechanism.

8.3.3.2 Programmatic security

Using programmatic security in JSPs and servlets is very similar to EJB programmatic security described in an earlier section. *Listing 8.16* provides an example.

```
System.out.println(request.getUserPrincipal().getName());
if (request.isUserInRole("super_user")){
  //add any logic that is super user specific.
}
```

Listing 8.16 - Programmatic access control

8.3.3.3 Testing Web security from a browser

If you chose **BASIC** as your Web project's authentication method, a dialog window will be pop-up as shown in *Figure 8.5*



Figure 8.5 – Popup dialog for ID and password

The dialog will pop-up when you try to access a Web project's context path the first time, and then the browser can record your login information so that the dialog doesn't show again unless you reopen the address in a new browser.

For example, if you type **andy** as the id and password, the Web project will deny you access because **andy** was defined earlier as a general user. *Figure 8.6* shows the error message you would receive.

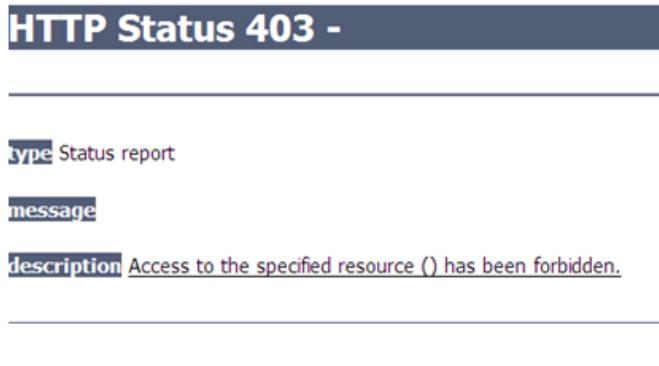


Figure 8.6 – Access denial

8.4 Managing users and groups for Community Edition administrative interface

All Java EE application servers should have secure access to their administrative interfaces (such as the Community Edition Administrative Console and the command line). Community Edition provides a built-in security realm called **geronimo-admin**, which is a properties file realm containing only one user called **system** with a password of **manager**, and only one group called **admin**.

Community Edition also provides a tool in the Administrative Console to configure this in a very easy way. Clicking on *Security -> Users and Groups* will open the page illustrated in *Figure 8.7* where you can create or edit a user and a group.

Console Realm Users

Username	Actions
system	Edit Delete

Console Realm Groups

Group Name	Actions
admin	Edit Delete

Figure 8.7 – Portlets for Community Edition administrative configuration

8.5 Securing your data with trusted contexts

Trusted contexts provide a way to build fast and more secure three-tier applications. A three-tiered application consists of a database server such as DB2, a middleware server such as Community Edition, and end users. With this model, Community Edition is responsible for accessing the DB2 database server on behalf of end users. Trusted context support ensures that an end user's DB2 database identity and DB2 database privileges are used when Community Edition performs any database requests on behalf of that end user.

The user's identity is always preserved for auditing and security purposes. When you require secure connections, trusted contexts improve performance because you do not have to get new connections.

A trusted context is an object that the DB2 database administrator defines that contains a system authorization ID and a set of trust attributes, like the IP address, that identify a connection as trusted. The relationship between a database connection and a trusted context is established when the connection to the database server is first created, and that relationship remains for the life of the database connection. Community Edition can use that database connection under a different user without re-authenticating the new user at the database server.

To avoid vulnerability to security breaches, a Community Edition server using these trusted methods should not use untrusted connection methods.

Note:

For more information about trusted contexts, refer to the article

<http://www.ibm.com/developerworks/data/library/techarticle/dm-0609mohan/index.html>

and this tutorial that uses Data Web Services and trusted context with IBM Data Studio:

<http://www.ibm.com/developerworks/edu/dm-dw-dm-0805misra2-i.html>

8.6 Summary

In this chapter, you learned how to configure the security realm in Community Edition's Administrative Console, and how to implement security in an EJB and Web project. You also learned about specific utilities to manage Community Edition administrative security. Later in the chapter you learned about trusted contexts which are ideal for performance and security in three-tiered applications.

8.7 Exercises

In this exercise, you will create your own security realm, and then deploy the deployment plan to Community Edition.

Procedure

1. Start the Community Edition Server and launch the Community Edition admin console. Log on as **system** (with password **manager**).
2. Follow the steps in section 8.2 to configure a properties file Security Realm, using `var/security/demo_users.properties` and `var/security/demo_groups.properties`.
3. On the next page, click *Skip Test and Show Plan* as shown in *Figure 8.8* below.

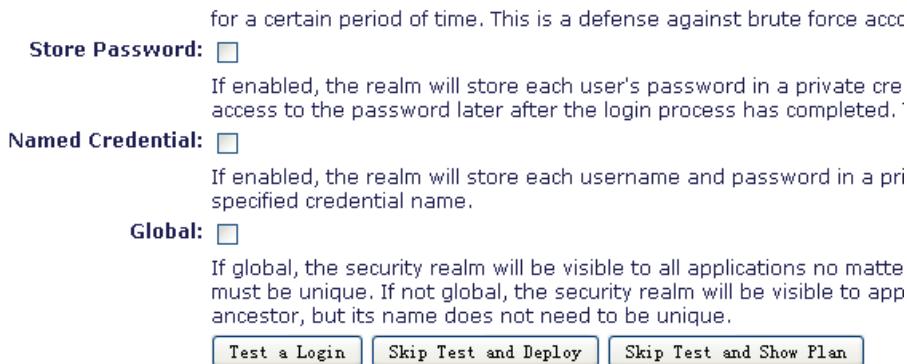


Figure 8.8 – Deploy a security realm

4. Copy the plan on the next page, and save it as an XML file as shown in *Figure 8.9*

Create Security Realm -- Show Deployment Plan

Deployment Plan:

```

<module xmlns="http://geronimo.apache.org/xml/ns
/deployment-1.2">
  <environment>
    <moduleId>
      <groupId>console.realm</groupId>
      <artifactId>MyRealm</artifactId>
      <version>1.0</version>
      <type>car</type>
    </moduleId>
    <dependencies>
      <dependency>
        <groupId>org.apache.geronimo.framework</groupId>
        <artifactId>j2ee-security</artifactId>
        <type>car</type>
      </dependency>
    </dependencies>
  </environment>
  <gbean name="MyRealm"
class="org.apache.geronimo.security.realm.GenericSecurityRealm"
xsi:type="dep:gbeanType" xmlns:dep="http://geronimo.apache.org
/xml/ns/deployment-1.2" xmlns:xsi="http://www.w3.org
/2001/XMLSchema-instance">
    <attribute name="realmName">MyRealm</attribute>
    <attribute name="global">false</attribute>
    <reference name="ServerInfo">
      <name>ServerInfo</name>
    </reference>
    <xml-reference name="LoginModuleConfiguration">
      <log.login-config xmlns:log="http://geronimo.apache.org
/xml/ns/loginconfig-2.0">
    
```

[Deploy] [Edit Settings]

Deploy Command: To deploy a security realm from the command line using this
xml file, and save it. Then run a command like:

Figure 8.9 – Show deployment plan

5. Customize the XML file you just saved.

6. Navigate to the *Applications -> Deploy New* portlet, and deploy your plan as shown in *Figure 8.10*

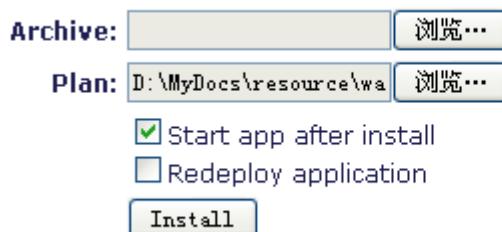


Figure 8.10 – Deploy the plan

8.8 Review questions

1. What is a security realm?
2. Why do you need to define role mapping?
3. What is the difference between the two types of role-based security?
4. How do you manage Community Edition's administrative security?
5. Why would using trusted contexts improve security and performance in a Java EE application using Community Edition and DB2?
6. Which file should you provide when defining a properties file realm?
 - A. Users' properties file
 - B. Group's properties file
 - C. A and B
 - D. Neither A nor B
 - E. Only one of A or B
7. Which is the pre-defined security realm type provided by Community Edition?
 - A. Properties File Realm
 - B. LDAP Realm
 - C. Database (SQL) Realm
 - D. Certificate Properties File Realm
 - E. All of the above
8. Which are the annotations used in declarative security? (Multi-choice)
 - A. @DeclareRoles

- B. @Roles
 - C. @Resource
 - D. @RolesAllowed
 - E. @Allowed
9. Which files are used to define the role mapping for a Java EE application?
- A. MANIFEST.MF
 - B. Deployment plan
 - C. Java Class
 - D. Deployment descriptor
 - E. None of the above
10. Which is the authentication mechanism that can be used in a Web application?
- A. BASIC
 - B. FORM
 - C. Client_CERT
 - D. Digest
 - E. All of the above

PART III – ADMINISTERING COMMUNITY EDITION

9

Chapter 9 – Administering Community Edition

This chapter discusses how to administer the Community Edition server using the administrative console and the command line.

In this chapter you will learn about:

- Working with the Community Edition administrative console (admin console)
- Configuring the server
- Working with the Community Edition repository
- Administering applications
- Configuring multiple server instances

9.1 Administering Community Edition: The big picture

Figure 9.1 provides an overview of how you can administer a Community Edition server. In the figure, the Community Edition admin console can be used to administer server, services, security, debug views and the embedded DB depicted on the left side of the figure. It can also be used to add JARs into Community Edition and manage applications as shown on the right side. The command line provides a series of sub-commands for managing server, applications and JARs.

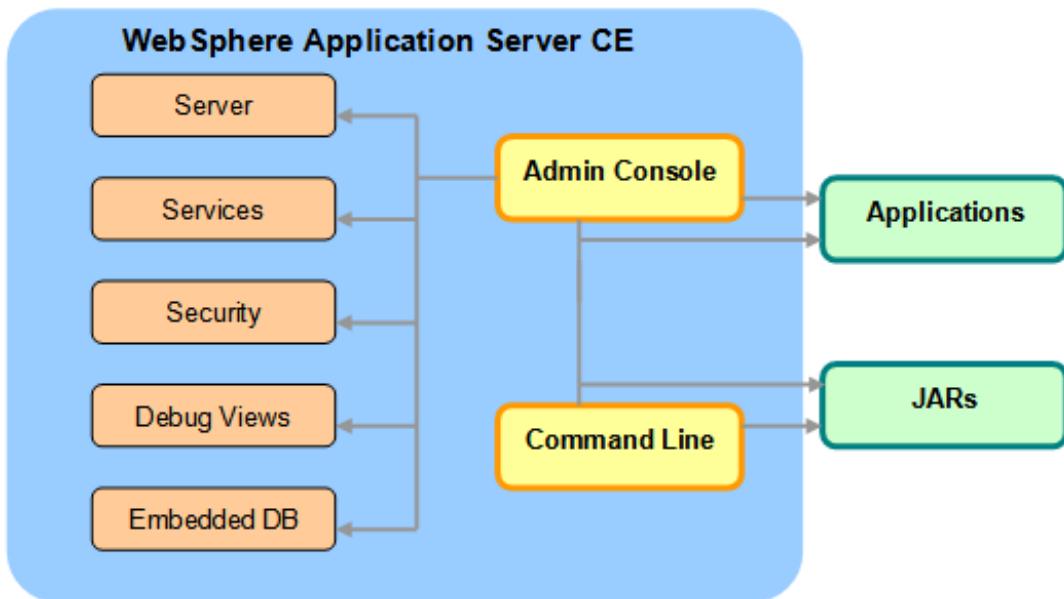


Figure 9.1 - Administering Community Edition: The big picture

9.2 Starting and stopping the server

This section describes the menus and commands you can use to start and stop the server. You need to start the Community Edition server before you can work with Community Edition and the administrative console. To make server configuration changes, you sometimes need to stop the server first. Starting and stopping the server is performed through platform-dependent scripts.

9.2.1 Starting the server

On Windows, to start the Community Edition server, follow these steps:

1. Click *Start->All Programs*.
2. Select *IBM WebSphere-> Application Server Community Edition*.
3. Click *Start the server*.

On Linux, follow these steps:

1. Change the current directory into `<WASCE_HOME>/bin`, where `<WASCE_HOME>` is the installation directory of the Community Edition server.
2. Run `startup.sh`.
3. When the server is started, a message similar to “*Geronimo started in background. PID: 1000*” shows up. The process number `PID` will be different each time.

9.2.2 Stopping the server

On Windows, to stop a running Community Edition server, follow these steps:

1. Click *Start->All Programs*.
2. Select *IBM WebSphere-> Application Server Community Edition*.
3. Click *Stop the server*.

On Linux, follow these steps:

1. Change the current directory into <WASCE_HOME>/bin, where <WASCE_HOME> is the installation directory of the Community Edition server.
2. Run `shutdown.sh`. If you have changed the default naming port number (by default it is 1099), you have to specify the port number in the command line. Use the following syntax, where `portnumber` is your naming port number:

```
shutdown.sh --port portnumber
```

3. You will be prompted to specify a username and password. By default the username is `system` and the password `manager`.

9.3 Configuring the Community Edition server manually

Although you can make most of the server configurations using the Community Edition administrative console, it is sometimes more convenient to make configuration changes manually by editing the `config.xml` and `config-substitutions.properties` files in the <WASCE_HOME>/var/config directory.

Note:

You should modify the `config.xml` and `config-substitutions.properties` files only when the server is stopped; otherwise the changes will be lost.

Listing 9.1 shows sample contents of the `config-substitutions.properties` file. The following sections discuss in more detail some of the parameters listed in this file.

```
ClusterNodeName=NODE
DefaultWadiSweepInterval=36000
ResourceBindingsFormat={groupId}/{artifactId}/{j2eeType}/{name}
COSNamingPort=1050
ORBPort=6882
HTTPPort=8080
WebConnectorConTimeout=20000
NamingPort=1099
webcontainer=TomcatWebContainer
ORBSSLPort=2001
```

```
JMXPort=9999
WADIClusterName=DEFAULT_WADI_CLUSTER
ResourceBindingsNamePattern=
RemoteDeployHostname=localhost
TmId=71,84,77,73,68
DerbyPort=1527
DefaultWadiNumPartitions=24
MaxThreadPoolSize=500
ResourceBindingsNameInNamespace=jca\:
OpenEJBPort=4201
ResourceBindingsQuery=?\#org.apache.geronimo.naming.ResourceSource
webcontainerName=tomcat6
SMTPPort=25
FarmName=DEFAULT_FRAM
HTTPSPort=8443
ActiveMQStompPort=61613
ActiveMQPort=61616
AJPPort=8009
PortOffset=0
SMTPHost=localhost
ServerHostname=0.0.0.0
EndPointURI=http\://localhost\:8080
ReplicaCount=2
JMXSecurePort=9998
MinThreadPoolSize=200
```

Listing 9.1 - config-substitutions.properties configuration file

9.3.1 Setting the IP address and hostname

The default IP address for the Community Edition server is 0.0.0.0. If you want to override this IP address, edit the file `config-substitutions.properties` as follows:

- Change the `ServerHostname` attribute with the IP address of your server; for example, 9.128.237.35, or
- Change the `ServerHostname` attribute with the host name of your server. First test your server can be accessed by using the `ping` command following this syntax:
`ping hostname`.

9.3.2 Changing port numbers

The easiest way to override all default port numbers is to change the `PortOffset` parameter into a non-zero number. In this case, the actual port number will be `portnumber + PortOffset`, where `portnumber` is the number specified in

`config-substitutions.properties`. For example, if you review the parameter values in *Listing 9.1* and change `PortOffset` to a value of 10, then `HTTPPort` will be set to 8090, and `NamingPort` will be set to 1109.

Note:

If you are running other software in your machine that uses TCP/IP ports, it is common to encounter port conflict problems when you start the Community Edition server. By changing port numbers or the `PortOffset` parameter, you can get past this problem. For more information, see *Chapter 11, Troubleshooting*.

9.3.3 Changing the username and password

The default username and password for the Community Edition administrative console is `system` and `manager` respectively. You can change these defaults by editing the `groups.properties` and `users.properties` files in the `<WASCE_HOME>/var/security` directory.

To add a new user and specify a password, follow these steps:

1. Open the file `<WASCE_HOME>/var/security/groups.properties`, and add a new username. For example, in *Listing 9.2*, another user named `user1` will be created in the admin group.

```
admin=system, user1
```

Listing 9.2 - groups.properties file

2. Open the file `<WASCE_HOME>/var/security/users.properties`, and add a line to specify the password for the user `user1`. For example, in *Listing 9.3* the password has been set to `password1`. The username has to be the same as the one added to the `groups.properties` file.

```
System=manager
user1=password1
```

Listing 9.3 - users.properties file

The Community Edition server does not save passwords as plain text in the file `users.properties`. Instead, it uses Advanced Encryption Standard (AES) encoded passwords after you start the server for the first time.

9.4 Introducing the administrative console

The Community Edition admin console is a Web-based application user interface for managing the Community Edition server. After you start the server, you can access the admin console by pointing your browser to the URL <http://localhost:8080/console>. The admin console is shown in *Figure 9.2* below.



Figure 9.2 - The Community Edition admin console

Log on to the admin console by specifying the username of **system** and the password of **manager**. Once you log on, you can navigate through the following portlet categories on the left side of the console to perform server management tasks:

- Welcome
- Server
- Services
- Applications
- Security
- Debug Views
- Embedded DB

9.4.1 Welcome

Figure 9.3 shows the welcome page you see when you log on to the admin console. It provides some quick links to common console actions and online resources on the right side.

Figure 9.3 - The Welcome page

9.4.2 Server

The **Server** category includes server information and operations:

- **Information** displays information about the server, such as the operating system and Java environment options when the server was started.
- **Java System Info** shows values of system properties in the Java environment.
- **Server logs** provides a log manager for changing the log configurations, and server log viewer, Web access viewer and Derby log viewer to display and filter log files.
- **Shutdown** shuts down the server.
- **Web Server** displays Web server statistics and enables you to configure network listeners.
- **Thread Pools** lists the status of the common thread pools.
- **Apache HTTP** guides you through a wizard to configure a remote Apache HTTP server.
- **JMS Server** enables you to configure JMS brokers and JMS network connectors.
- **Monitoring** provides server statistics and monitoring features to review the server performance.

9.4.3 Services

The **Services** category includes the management of server wide services:

- **Repository** lists the current repository entries and enables you to add archives to the server's repository.
- **Database Pools** lists all database pools available on the server and walks you through creating, editing and importing database connectors.
- **JMS Resources** lists all Java Messaging Service (JMS) resources available on the server and guides through creating new JMS brokers.

9.4.4 Applications

The **Applications** category provides a collection of portlets for administering applications in the Community Edition server:

- **Web App WARs** lists all Web application WARs available on the server and enables you to start, stop, restart or uninstall Web applications.
- **System Modules** lists all system modules available on the server and enables you to start, stop, restart or uninstall some of these system modules.

- **Application EARs** lists all enterprise application EARs available on the server and enables you to start, stop, restart or uninstall some of these applications.
- **EJB JARs** lists all Enterprise Java Bean (EJB) JARs available on the server and enables you to start, stop, restart or uninstall some of these applications.
- **J2EE Connectors** lists all J2EE connectors available on the server and enables you to start, stop, restart or uninstall some of these connectors.
- **App Clients** lists all application clients available on the server and enables you to start, stop, restart or uninstall these application clients.
- **Deploy New** enables you to install new applications on the server.
- **Plugins** enables you to install or create Geronimo plugins from applications. In this portlet you can also assemble a server from plugins.
- **Plan Creator** provides a wizard for creating deployment plans from an existing Web application.

9.4.5 Security

The **Security** category contains security related server operations:

- **Users and Groups** allows you to add or change users and user groups of the server.
- **Keystores** walks you through the process of creating new keystores to use with SSL connectors.
- **Certificate Authority** allows you to create Certification Authority (CA) and issue certificates in response to certificate signing requests (CSRs).
- **Security Realms** lists all security realms available on the server and allows you to create and edit realms.

9.4.6 Debug Views

The **Debug Views** category contains 5 types of debug views for monitoring the server:

- **JMX Viewer** lists different kinds of MBeans in a tree structure and their attributes, operations, and statistics on the right side of the panel.
- **LDAP Viewer** enables you to connect to a LDAP server. By default this portlet displays the information about the embedded LDAP server in Community Edition.
- **ClassLoader Viewer** lists all classloaders available on the server and the classes they load.
- **JNDI Viewer** lists all JNDI contexts on the server.
- **Dependency Viewer** lists all modules and their dependencies.

9.4.7 Embedded DB

The **Embedded DB** category contains database-related operations:

- **DB Info** displays information about the embedded Apache Derby database, including the database name and version, driver name and version, supported JDBC versions and supported SQL commands.
- **DB Manager** enables you to create or delete databases and execute SQL commands in a selected database.

9.5 Adding JARs to the Community Edition repository

The Community Edition server keeps a repository in <WASCE_HOME>/repository, where common Java libraries are stored. Its structure and naming conventions follow the Maven style (<http://maven.apache.org>). If your application depends on some libraries that are not included in the repository, you have to add them to the repository before you deploy the application to the server.

To add a Java library to the repository, open your Web browser and log on to the administrative console (by default the URL is <http://localhost:8080/console>), and then follow these steps:

1. Click *Repositories* under *Server*. The resulting *Repository Viewer* page illustrated in *Figure 9.4* displays all the Java libraries available in the repository.

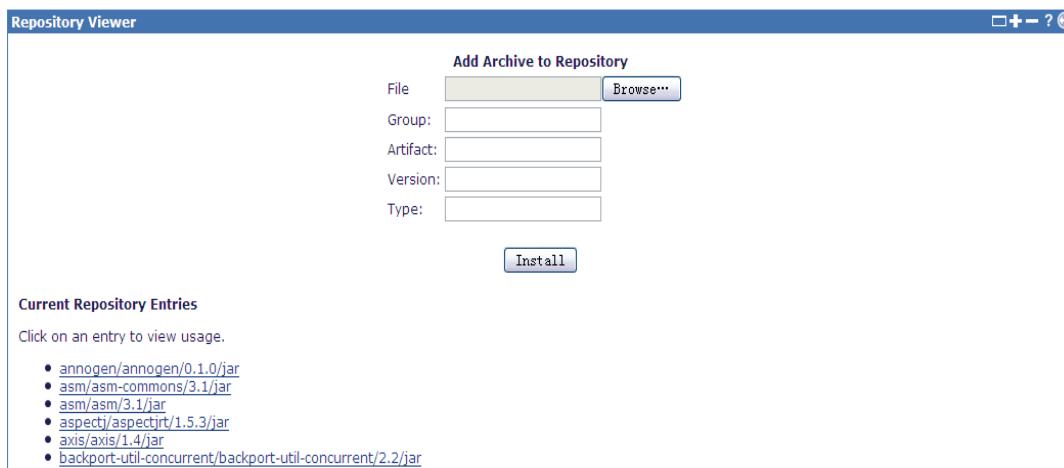


Figure 9.4 - Repository Viewer

2. In the *File* field, click *Browse* and select the directory where your JAR is placed.
3. Specify other fields for the library:
 - **Group** is the group identifier. Typically, it is the name of an open source project (for example log4j), or a directory tree matching the Java package

prefix (for example `org/apache/geronimo`) of the organization that supplies the library.

- **Artifact** is the file name prefix of the library.
- **Version** is the version identifier of the library.
- **Type** is the type of file, typically `jar`.

When you package your application, you have to prepare a deployment plan (for example, for a Web application, a `geronimo-web.xml` under `WEB-INF` directory). The Community Edition server can look into the plan and the deployment descriptor (for example, for a Web application, a `web.xml` under `WEB-INF` directory) to find information about the application. If your application uses the library that you have just installed, you have to add a `<dependency>` element under `<dependencies>` in the `<environment>` element in the application's deployment plan. *Listing 9.4* provides an example of an application deployment plan showing dependencies.

```
<sys:environment>
...
<sys:dependencies>
    <!-- Repeat the dependency element for each dependency -->
    <sys:dependency>
        <sys:groupId>Group<sys:groupId>
        <artifactId>Artifact<sys:artifactId>
        <sys:version>Version<sys:version>
        <sys:type>Type<sys:type>
        <!-- Not shown: import options -->
    </sys:dependency>
    <!-- End of dependency element -->
</sys:dependencies>
...
</sys:environment>
```

Listing 9.4 - Dependency snippet

9.6 Administering applications

This section describes how to administer applications, such as managing their deployment, and stopping and starting them.

9.6.1 Deploying and undeploying applications

When you finish packaging an application, you can deploy the application to the Community Edition server using the administrative console. For example, if you have a Web application `cviewer-2.1.1.2.war` under the directory `usr/local/wasce_samples/cviewer/target`, and the deployment plan under the

directory `usr/local/wasce_samples/cviewer/src/main/webapp/WEB-INF`.

Follow these steps to deploy your WAR:

1. Click *Deploy New* under *applications*. The resulting *Install New Applications* page is illustrated in *Figure 9.5*:



Figure 9.5 - Installing new applications into Community Edition

2. In the *Archive* field, browse into the directory that includes the Web application, and select the WAR file. In this example we use `usr/local/wasce_samples/cviewer/target/cviewer-2.1.1.2.war`.
3. In the *Plan* field, browse into the directory that includes a deployment plan. For this example, use `usr/local/wasce_samples/cviewer/src/main/webapp/WEB-INF/geronimo-web.xml`.
4. Click *Install*. *Figure 9.6* displays the resulting page of a successful installation.

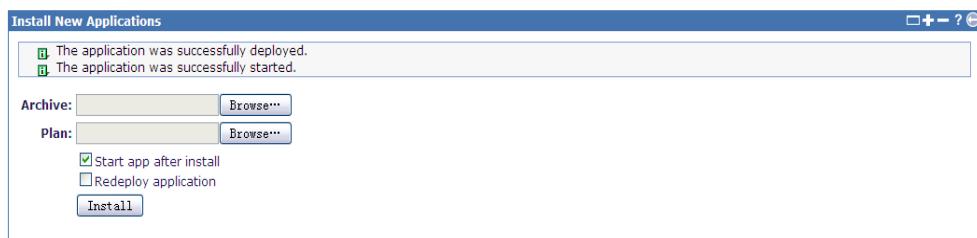


Figure 9.6 - Messages after a successful new application install into Community Edition

5. You can also verify the installation in the *Installed Web Applications* panel. Click *Web App WARs* under *Applications*. You can see the application *cviewer* in the running state as illustrated in *Figure 9.7*.

The screenshot shows the 'Installed Web Applications' panel. At the top, there are two checkboxes: 'Expert User (enable all actions on Geronimo Provided Components)' and 'Show parent and child components'. Below is a table with the following data:

Component Name	URL	State	Commands
com.ibm.wasce.configs/collector-tool-agent-config/2.1.1.3/car	/collector-tool-agent	running	Stop Restart Uninstall
com.ibm.wasce.configs/welcome-tomcat/2.1.1.3/car	/	running	Stop Restart Uninstall
com.ibm.wasce.samples/cviewer/2.1.1.2/car	/cviewer	running	Stop Restart Uninstall

Figure 9.7 - Verifying the installation of an application

- To remove the application from Community Edition, click *Uninstall* for *cviewer* on the *Installed Web Applications* panel. This will stop the application and then remove it from the server. *Figure 9.8* below shows the message you receive.

**Figure 9.8 - Uninstalling an application**

- Similarly, you can deploy or undeploy an EAR in the *Installed Application EARs* panel by clicking *Application EARs* from the administrative console.

Note:

You can download the sample applications (Version 2.1.1.2) from the *product download site* https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?lang=en_US&source=wsced and install them following the above steps.

9.6.2 Starting and stopping applications

To change the status of an application, you can log on to the administrative console, and click *Web App WARs* or *Application EARs* under *Applications*. The panel displays a list of all available WAR or EAR applications on the server and their current status.

Take the application *cviewer* described in the last section as an example.

- To stop the running application *cviewer*, click *Stop*. *Figure 9.9* shows the message you should receive.

**Figure 9.9 - Message to receive after stopping the cviewer application**

- To start the stopped application *cviewer*, click *Start*. *Figure 9.10* shows the message you should receive.

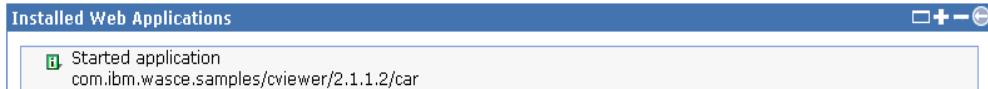


Figure 9.10 - Message to receive after starting the cviewer application

9.7 Tools and commands

Besides the administrative console, Community Edition also provides a series of tools and commands for administering the server from the command line.

9.7.1 The deploy command

The **deploy** command is used to manage modules and Java EE assets on a Community Edition server. It includes a set of options or sub-commands as described below:

1. The **deploy** sub-command within the **deploy** command, deploys a module (a Web application archive WAR, an Enterprise application EAR, a JavaBean archive JAR, or a Java Resource archive RAR) to the server. For example, to deploy a WAR to the server, use the syntax shown in *Listing 9.5* from the `<WASCE_HOME>/bin` directory:

```
./deploy.sh --user system --password manager deploy
<APP_HOME>/cviewer-2.1.1.2.war <PLAN_HOME>/geronimo-web.xml
```

Listing 9.5 - Example of deploy - deploy

where `<APP_HOME>` is the directory where the **cviewer** application WAR is placed, and `<PLAN_HOME>` is the location of the deployment plan that this WAR is using.

2. The **list-modules** sub-command within the **deploy** command lists all modules available on an active server. For example, use this syntax from the `<WASCE_HOME>/bin` directory:

```
./deploy.sh --user system --password manager list-modules
```

Listing 9.6 - Example of deploy list-modules

3. You will be prompted with a report as shown in *Listing 9.7*. The plus ("+") indicates that the module is running.

```
Found 91 modules
+ com.ibm.wasce.configs/collector-tool-agent-config/2.1.1.3/car
+ com.ibm.wasce.configs/collector-tool-config/2.1.1.3/car
+ com.ibm.wasce.configs/db2v82/2.1.1.3/car
+ com.ibm.wasce.configs/db2v91/2.1.1.3/car
+ com.ibm.wasce.configs/db2v95/2.1.1.3/car
...
...
```

Listing 9.7 - Results of the list-modules sub command

Note:

For more information about the `deploy` command, refer to the *product documentation* at <http://publib.boulder.ibm.com/wasce/V2.1.1/en/deploy.html>.

9.8 Configuring multiple server instances

The Community Edition server provides the capability of running multiple server instances on one server installation. The advantage of having multiple server instances is that you can better use the computing capability of your system. These instances share the following directories under `<WASCE_HOME>`:

- bin
- lib
- schema
- repository

Each Community Edition instance has its own `var` and `deploy` directories, so you have to create `var` and `deploy` directories for each non-default instance. To avoid port conflict, you can override the `portOffset` attribute in `config-substitutions.properties`.

Note:

For more information about the steps to create and run multiple server instances, see the *product documentation*:

<http://publib.boulder.ibm.com/wasce/V2.1.1/en/configuring-multiple-server-instances.html>

9.9 Configuring multiple repositories

You can deploy a single repository on a single server instance, or deploy multiple repositories on multiple instances. You can also share a repository among multiple server instances, or deploy multiple repositories on a single server instance.

The advantage of having multiple repositories is that you can deploy your applications or plugins to different repositories. Therefore, you can replace, migrate or remove applications or plugins without affecting unrelated repositories, especially the Community Edition default repository.

Note:

For more information about creating multiple repositories on a single server instance, and creating multiple repositories on multiple server instances, see the *product documentation*:

<http://publib.boulder.ibm.com/wasce/V2.1.1/en/configuring-multiple-repositories.html>

9.10 Exercises

In this exercise, you will start a non-default server instance with a port offset of 10, deploy the application *jsp-examples-war* obtained from the Community Edition samples for version 2.1.1.2, (https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?lang=en_US&source=wsc_ed), and then stop the server.

Part 1 - Start a non-default server

Procedure

1. While the server is stopped, follow the procedure described in section 9.3.2 to change the default naming port. In the file `<WASCE_HOME>/var/config/config-substitutions.properties`, the *PortOffset* entry should look like the line below:
`PortOffset=10`
2. Start the server using this syntax: `./startup.sh`

Part 2 - Deploy the application to Community Edition

Procedure

1. Log on to the Community Edition Console with the username **system** and password **manager**.
2. On the left navigation panel, select *Applications* → *Deploy New* to open the *Install New Applications* portlet.



3. Click *Browse* (besides the *Archive* field), and select `<SAMPLE_HOME>/applications/jsp-examples-war/target/jsp-examples-war-2.1.1.2.war`.
4. Select `<SAMPLE_HOME>/applications/jsp-examples-war/src/main/webapp/WEB-INF/geronimo-web.xml`.
5. Click *Install*.

Part 3 - Stop the server.

Procedure

1. Open a command window and go to the directory <WASCE_HOME>/bin.
2. Use this syntax to shutdown the server:
`./shutdown --port 1109 --user system --password manager`

9.11 Summary

In this chapter you have learned the basics to administer a Community Edition server. Most administering tasks can be performed from the administrative console, including configuration changes for parameters such as port numbers, user names and passwords. Community Edition also enables you to create multiple server instances and repositories so that applications can be deployed to different instances or targets.

9.12 Review questions

1. What are the steps to start the Community Edition server on Linux?
2. How can you override all the default port numbers in a Community Edition server?
3. What are the steps to add Java libraries into the Community Edition server repository?
4. Which file has all the user names for a Community Edition server?
5. On a Community Edition installation, which directories should be created for a new server instance?
6. Which command is used to list all available modules on the server?
 - A. deploy deploy
 - B. startup
 - C. deploy list-modules
 - D. deploy list-targets
 - E. None of the above
7. What's the name of the deployment plan for a Web application in Community Edition?
 - A. geronimo-application.xml
 - B. web.xml
 - C. application.xml
 - D. geronimo-web.xml
 - E. None of the above

8. Which portlet will you look into in the admin console if you want to find information about all JMS resources?
 - A. Server
 - B. Services
 - C. Applications
 - D. Security
 - E. None of the above
9. If you add a JAR to a Community Edition server, where will it be stored?
 - A. <WASCE_HOME>
 - B. <WASCE_HOME>/var
 - C. <WASCE_HOME>/repository
 - D. <WASCE_HOME>/deploy
 - E. None of the above
10. What are the default user name and password for the Community Edition admin console?
 - A. system and manager
 - B. admin and manager
 - C. admin and password
 - D. system and password
 - E. None of the above

10

Chapter 10 – Tuning a Community Edition server

Tuning a Community Edition server requires you to apply a series of techniques that will improve the system performance. This chapter provides an overview of how you can tune a Community Edition server.

In this chapter you will learn about:

- How to monitor a Community Edition server
- How to analyze Community Edition server statistics
- How to make changes to the Community Edition server, JVM, and OS to improve overall system performance

10.1 Tuning a Community Edition server: The big picture

Figure 10.1 provides an overview of the steps required to tune a Community Edition server.

As shown in the figure, performance tuning usually involves the following iterative multi-step cycle:

1. Measure the performance before you apply any changes to the system by using the appropriate monitoring tools.
2. Identify possible performance bottlenecks in the whole system. For Community Edition, the bottleneck might be in the Community Edition server itself, JVM, or the operating system.
3. Make changes to remove the bottlenecks.
4. Return to step 1 to measure the performance again to see if there are any improvements.

The cycle ends when the system reaches an acceptable performance.

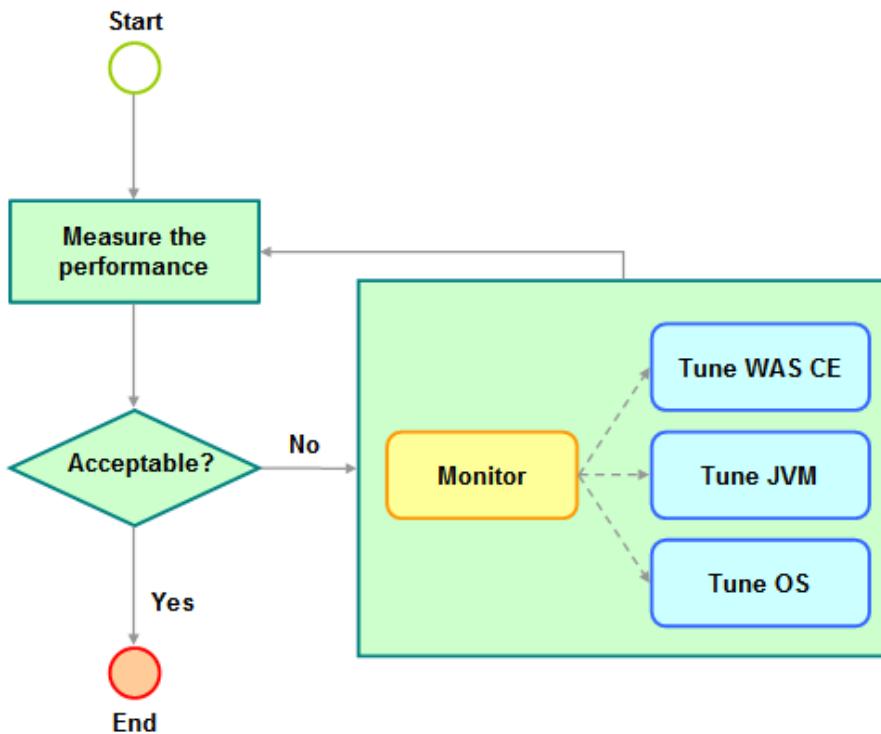


Figure 10.1 - Tuning a Community Edition server: The big picture

10.2 Monitoring Community Edition

Community Edition server admin console is integrated with a monitoring plug-in that allows you to monitor and obtain statistics for various Community Edition server components such as:

- Transaction Manager
- JVM
- AJP/Web/WebSSL connector
- ThreadPool
- Web application

To quickly monitor a server:

1. Start the Community Edition server and launch the admin console. Login as **system** (with password **manager**).
2. On the left hand navigation pane, select *Server -> Monitoring* to open the Monitoring portlet and click on *Add a new server* under the *Servers* section. This is illustrated in *Figure 10.2* below.

The screenshot shows the 'Monitoring' section of the Admin Console. On the left, there's a 'Console Navigation' sidebar with links like Welcome, Server, Information, Java System Info, Server Logs, Shutdown, Web Server, Thread Pools, Apache HTTP, JMS Server, Monitoring, and Services. The main area is titled 'Monitoring' and contains a 'Views' section with a table for 'Elements' and a 'Servers' section. The 'Servers' table has columns for Name, IP/Hostname, Status, and Actions. A red box highlights the '+ Create a new view' link. Below it is a 'Graphs' section with a similar table structure, and a red box highlights the '+ Add a new graph' link.

Figure 10.2 - Adding a new server to monitor from the admin console

- When the *Add a new server* window appears as shown in *Figure 10.3*, complete the fields as follows, and click on *Add*.

- **name** = localhost
- **IP/Hostname** = localhost (if you want to monitor a Community Edition instance remotely, provide the server IP address or hostname of that instance)
- **Port** = 4201
- **username** = system
- **password** = manager

The screenshot shows the 'Add a new server' dialog box. It has fields for Name (localhost), IP/Hostname (localhost), Protocol (EJB selected), Port (4201), Username (system), Password (redacted), and Password (verify) (redacted). There are 'Cancel' and 'Add' buttons, with 'Add' being highlighted with a red box. To the right is a 'Navigation' sidebar with links to Home, Views, Servers, and Graphs. Below that is an 'Actions' sidebar with a link to 'Test these settings'.

Figure 10.3 - Inputting details about the new server to monitor

- After adding a new server to monitor, the window shown in *Figure 10.4* appears. Under the Servers section, the new server is displayed. Click on the monitoring target **localhost** under the IP/Hostname column to view the live statistics of this server.

The screenshot shows the 'Monitoring' interface. At the top, there's a message: 'Server localhost at localhost has been added.' Below this is a 'Views' section with a table header: 'Name | Elements | Created | Modified | Actions'. A link '+ Create a new view' is also present. Under 'Servers', there's another table with columns: 'Name | IP/Hostname | Status | Stat. Query | Actions'. It lists a single server: 'localhost | localhost | Online | (stopped) | + Enable Query, / Edit, X Disable'. A link '+ Add a new server' is located below this table. At the bottom left is a 'Graphs' section.

Figure 10.4 - A new server is added and ready to be monitored

5. Figure 10.5 shows live statistics information. By default, there are no collected statistics.

The screenshot shows the 'Monitoring' interface with the 'localhost' server selected. On the left, under 'localhost', there's a 'Live Statistics' section with the following data:

Status:	Online
Snapshot Thread:	stopped
Added:	2009-08-10 18:09
Last Modified:	2009-08-10 18:09
Last Seen:	2009-08-10 18:09
IP/Hostname:	localhost
Snapshot Duration:	5minutes
Snapshot Retention:	30days

On the right, under 'Statistics Collected', there's a 'Statistics Available' section with a list of items and '+' icons:

- JTAResource
- TransactionManager
- JVM
- JVM
- Other
- ConnectorThreadPool
- DefaultThreadPool
- TomcatAJPConnector
- TomcatWebConnector
- TomcatWebSSLConnector
- WebModule
- activemq-console-tomcat

Figure 10.5 - Statistics information

6. To add monitoring items, click the + besides the monitoring items you want to enable for statistics collection under *Statistics Available* list. For example, click the + at the right side of *JVM* and *ConnectorThreadPool* as highlighted in the previous figure. You will see their live statistics in Figure 10.6 under the *Live Statistics* section.

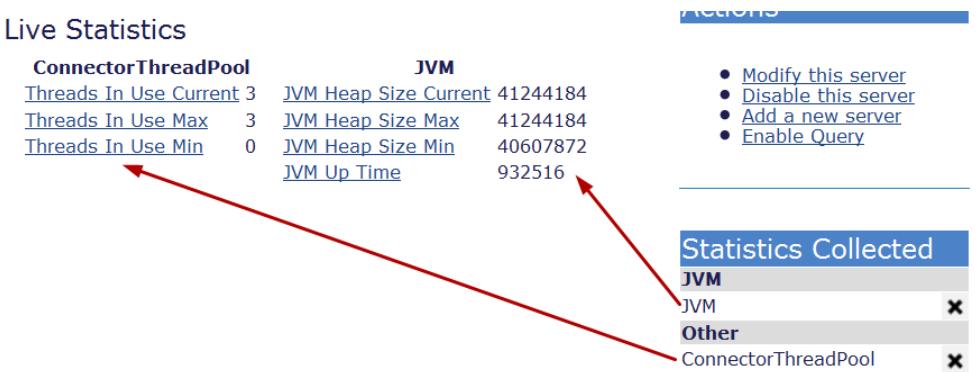


Figure 10.6 - Live statistics

Similarly, you can easily monitor TransactionManager, Tomcat connector, and Web apps in Community Edition.

This section provided a quick introduction to monitoring a Community Edition server with the monitoring plug-in. The monitoring plug-in has other powerful functions such as visualization of the history data of server statistics. For a complete reference about the monitoring plug-in, visit

<http://publib.boulder.ibm.com/wasce/V2.1.1/en/monitoring-the-health-of-a-server.html>

10.3 Community Edition Server tuning

In this section we describe parameters of a Community Edition server that you can monitor and modify to improve the performance.

10.3.1 Thread pool size

The thread pool size determines the processing capacity of a Community Edition server. Tuning this parameter is important to achieve the best performance. The Community Edition server allows the customization of three types of thread pools:

- **DefaultThreadPool** is the major thread pool in Community Edition; it specifies the maximum number of threads possible in the server. Its default size is 500.
- **ConnectorThreadPool** is used by the connector including JMS, Data source, and other JCA connectors in the server. Its default size is 30.
- **Web container connector thread pool.** Each connector of Web container has its own thread pool. Its default value varies for different Web connector types.

Note:

DefaultThreadPool is the major thread pool in Community Edition; however, some components including the Community Edition Web container are not using

DefaultThreadPool in the current release. Community Edition might migrate these components to use **DefaultThreadPool** in the future.

10.3.2 Monitoring thread pools

To choose the optimum value for pool sizes, you need to monitor the current utilization of thread pools. You can monitor thread pools using the administrative console:

1. Select the Thread Pools portlet in the console to open the configuration page. A page similar to *Figure 10.7* opens.

This page lists the thread pools defined in the Geronimo server. Note: Currently not all threads used by Geronimo come from one of these thread pools. We're working on migrating the different components of Geronimo toward these thread pools.

Name	Size	Actions
ConnectorThreadPool	0	Monitor
DefaultThreadPool	0	Monitor

Figure 10.7 - The thread pool configuration page

2. To monitor statistics related to a particular thread pool, click the *monitor* link corresponding to that pool in the *Actions* column. A page similar to *Figure 10.8* opens. You can monitor the followings statistics:

- **Pool Max** - The maximum number of threads that Community Edition server can create and use in the thread pool.
- **Lowest Recorded** - The lowest number of threads used up to now.
- **Highest Recorded** - The highest number of threads used up to now.
- **Threads in Use** - The number of threads currently in use.

Thread pools statistics for ConnectorThreadPool::

Pool Max: 30
Lowest Recorded: 0
Highest Recorded: 0
Threads in Use: 0

Figure 10.8 - Thread pool statistics pane

3. To monitor thread usage of Web connectors, you can still use the monitoring plug-in described in section 10.2. Click + besides **TomcatAJPConnector**, **TomcatWebConnector**, and **TomcatWebSSLConnector** to enable monitoring. You can observe the following attributes to monitor the Web container connector's thread pool as shown in *Figure 10.9*.

- **Busy Threads Current** - The number of threads currently in use.

- **Busy Threads Max** - The highest number of threads used up to now.
- **Busy Threads Min** - The lowest number of threads used up to now.

Live Statistics

TomcatAJPConnector		TomcatWebConnector		TomcatWebSSLConnector	
Active Request Count	0	Active Request Count	0 <th>Active Request Count</th> <td>0</td>	Active Request Count	0
Busy Threads Current	1	Busy Threads Current	2	Busy Threads Current	0
Busy Threads Max	25	Busy Threads Max	2	Busy Threads Max	0
Busy Threads Min	0	Busy Threads Min	0	Busy Threads Min	0
Bytes Received	0	Bytes Received	1579661	Bytes Received	0
Bytes Sent	0	Bytes Sent	4762583	Bytes Sent	0

Figure 10.9 - Live statistic information for Web connectors

10.3.3 Configuring the thread pool size

Choosing the optimum value for the thread pool size is important to successfully configure the Community Edition server for the best performance. Setting this value too low reduces the throughput because of an increased wait time for a request to be served. Setting it too high reduces the CPU efficiency because of frequent context switching among the threads. You must find the proper balance.

You can change thread pool sizes by updating the configuration files. This cannot be done using the administrative console. To apply the required changes:

1. Stop the Community Edition server.
2. Open the `<WASCE_HOME>/var/config/config.xml` with a text editor.
3. To specify the pool size for DefaultThreadPool, insert the following `<gbean>` element inside the `<module name="org.apache.geronimo.configs/rmi-naming/2.1.4/car">` element. *Listing 10.1* provides an example. In the listing, replace `xyz` with the pool size value you desire.

```
<gbean name="DefaultThreadPool">
    <attribute name="keepAliveTime">5000</attribute>
    <attribute name="poolSize">xyz</attribute>
</gbean>
```

Listing 10.1 - Specifying a DefaultThreadPool size

4. To specify the pool size for ConnectorThreadPool, insert the following `<gbean>` element inside the `<module name="org.apache.geronimo.configs/transaction/2.1.4/car">` element. *Listing 10.2* provides an example. In the listing, replace `xyz` with the pool size value you desire.

```
<gbean name="ConnectorThreadPool">
    <attribute name="keepAliveTime">5000</attribute>
    <attribute name="poolSize">xyz</attribute>
</gbean>
```

Listing 10.2 - Specifying a DefaultThreadPool size

5. To specify the pool size for connectors of Community Edition Web container, update the following `<gbean>` element inside the `<module name="org.apache.geronimo.configs/tomcat6/2.1.4/car">` element. Listing 10.3 provides an example. In the listing, replace `xyz` with the pool size value you desire. `XXX` stands for Web, AJP, or WebSSL based on the different type of Web container connector.

```
<gbean name="TomcatXXXConnector">
    ...
    <attribute name="maxThreads">PoolSize</attribute>
    ...
</gbean>
```

Listing 10.3 - Specifying a Web connector pool size

6. Save the changes and start the Community Edition server.
7. Verify your changes by:
 - Opening the `monitoring` portlet to monitor the three type thread pool as described in section 10.2.
 - Opening the `ConnectorThreadPool` page and the `DefaultThreadPool` page as described in 10.3.2 *Monitoring thread pools*

10.4 JVM and operating system tuning

As a Java EE application server, Community Edition runs on a JVM; which runs on the operating system. In this section, we discuss how to improve Community Edition performance by tuning JVM and the operating system.

10.4.1 JVM tuning

You may need to customize your Java virtual machine (JVM) to improve the performance of your server. Below, you will find tips for adjusting the JVM. Keep in mind that the Java environment may change, making these suggestions obsolete, and that your results can vary.

10.4.1.1 Monitoring memory usage

It is useful to monitor the current server memory usage before you apply any changes to the JVM parameters. To analyze the memory, log in to the administrative console and

select **Information** in the console navigation area which opens the server information page. From this page you can check the following relevant statistics:

- Current memory used
- Most memory used
- Total memory allocated

You can also view a real time graph of the current server memory usage as shown in *Figure 10.10*.

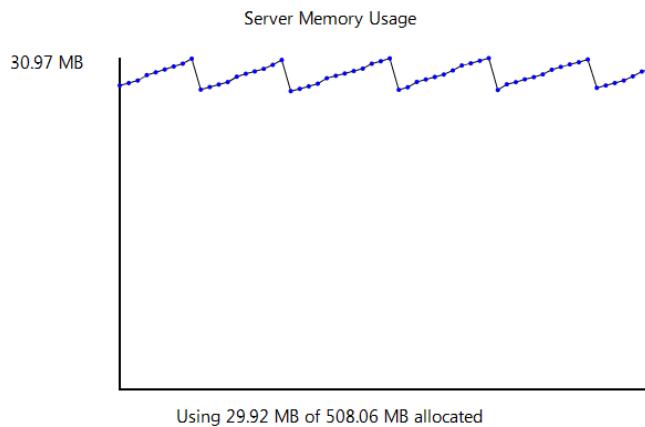


Figure 10.10 - Server memory usage dynamic graph

Note:

This feature requires you to install Adobe SVG Viewer when using Internet Explorer in a Windows environment. If you use Mozilla Firefox or Google Chrome this requirement is not needed.

10.4.1.2 Using the JConsole graphical tool to monitor

In addition to the monitoring plug-in described in section 10.2, you can use **JConsole** to monitor Community Edition. With **JConsole**, you can monitor JVM memory usage; threads stack trace, loaded classes, and VM information as well as Community Edition MBeans.

Refer to <http://publib.boulder.ibm.com/wasce/V2.1.1/en/working-with-jconsole.html> for more information.

10.4.1.3 Tuning the Java heap size

The Java heap is the runtime space from which Java objects are allocated. The heap size determines the time and the frequency of the garbage collection (GC) activities.

A value too high results in high memory consumption and a low number of major slow garbage collection cycles. A value too low results in a high number of minor fast garbage collection cycles.

The goal of properly tuning the heap size is to minimize the overhead of GC to improve the server response and the throughput. The IBM JDK which comes with Community Edition allows you to change several parameters that are related to the heap size:

- **Initial heap size**

The initial heap size specifies the initial amount of memory that is allocated in the heap when the JVM starts.

To set the initial heap size include the following option in the **JAVA_OPTS** environment variable where **mem** is the heap size in megabytes **-Xms<mem>m**

- **Maximum heap size**

The maximum heap size specifies the maximum amount of memory that the JVM can use.

To set maximum heap size, include the following option in the **JAVA_OPTS** environment variable where **mem** is maximum heap size in megabytes **-Xmx<mem>m**

Table 10.1 shows the IBM JVM default heap size values on Linux and Windows.

JVM setting	Linux	Windows
Initial Heap Size (-Xms)	4 MB	4 MB
Maximum Heap Size (-Xmx)	Half the real memory with a minimum of 16 MB and a maximum of 512 MB	Half the real memory with a minimum of 16 MB and a maximum of 2 GB

Table 10.1 - IBM JVM default heap size values

- **Heap with large pages**

It is also possible to configure the IBM JVM to allocate the heap using large pages. Use the following option **-xlp**

Note:

For information about the **JAVA_OPTS** environment variable see
<http://publib.boulder.ibm.com/wasce/V2.1.1/en/javaopts.html>

You can add the **JAVA_OPTS** variable to the following files on Windows:

```
<WASCE_HOME>/bin/geronimo.bat
<WASCE_HOME>/bin/setenv.bat
```

On Linux the filenames are the same but have a .sh extension.

10.4.2 Operating system tuning

Operating system tuning consists on using available sets of parameters and strategies to optimize operating systems. For a complete reference about system tuning, visit the following Web sites:

- For Linux and UNIX systems:

<http://publib.boulder.ibm.com/wasce/V2.1.1/en/tuning-unix.html>

- For Windows systems:

<http://publib.boulder.ibm.com/wasce/V2.1.1/en/tuning-windows.html>

10.5 Summary

This chapter discussed about tuning concepts including how to tune a Community Edition server, how to tune the JVM, and how to tune the operating system where Community Edition is running on. The chapter described the monitoring plug-in that can be used as part of the admin console and emphasized the importance of fine tuning thread pool sizes to obtain the best performance for your Community Edition server.

10.6 Exercises

In this exercise, you will set the Java heap size for Community Edition by adding JAVA_OPTS into a startup script.

Procedure

1. Shutdown the Community Edition server if it is running.
2. On Windows open <WASCE_HOME>/bin/geronimo.bat with an editor. On Linux the filename is geronimo.sh. Add the following line at the very beginning of the file:

- On Windows:

```
set JAVA_OPTS=-Xms512m -Xmx1024m
```

- On Linux:

```
JAVA_OPTS=-Xms512m -Xmx1024m
```

3. Start the Community Edition server again.
4. The **Initial heap size** and **Maximum heap size** of the running Community Edition server becomes 512m and 1024m.

10.7 Review questions

1. What are the typical steps to tune Community Edition?
2. What's the default size of `DefaultThreadPool size` and `ConnectorThreadPool` in Community Edition?
3. What are the steps to modify the size of `DefaultThreadPool size` and `ConnectorThreadPool` of Community Edition?
4. How do you monitor the memory usage of Community Edition?
5. What are the steps to set `Initial heap size` and `Maximum heap size` for Community Edition?
6. What's the Initial Heap Size for Windows and Linux?
 - A. 4 Mb
 - B. 8 Mb
 - C. 16 Mb
 - D. 32 Mb
 - E. 64 Mb
7. What's the Maximum Heap Size for Windows?
 - A. 128 Mb
 - B. 256 Mb
 - C. 512 Mb
 - D. Half the real memory with a minimum of 16 Mb and a maximum of 2 Gb
 - E. Half the real memory with a minimum of 16 Mb and a maximum of 512 Mb
8. What's the Maximum Heap Size for Linux?
 - A. 128 Mb
 - B. 256 Mb
 - C. 512 Mb
 - D. Half the real memory with a minimum of 16 Mb and a maximum of 2 Gb
 - E. Half the real memory with a minimum of 16 Mb and a maximum of 512 Mb
9. What's the size of the major thread pool of Community Edition?
 - A. 20
 - B. 40
 - C. 80

- D. 500
 - E. 1000
10. Which module do you need to modify in
`<WASCE_HOME>/var/config/config.xml` to configure the major thread pool size?
- A. `<module name="org.apache.geronimo.configs/rmi-naming/2.1.4/car">`
 - B. `<module name="org.apache.geronimo.configs/transaction/2.1.4/car">`
 - C. `<module name="org.apache.geronimo.configs/tomcat6/2.1.4/car">`
 - D. `<module name="org.apache.geronimo.configs/clustering/2.1.4/car">`
 - E. `<module name="org.apache.geronimo.configs/j2ee-server/2.1.4/car">`

11

Chapter 11 - Troubleshooting

This chapter describes troubleshooting techniques to some common problems when working with Community Edition.

In this chapter you will learn about:

- How to resolve problems during Community Edition installation and uninstallation
- How to resolve problems during start and stop of the Community Edition server
- How to correctly set class path and dependencies in an application
- How to use the Community Edition log files as diagnostic tool

11.1 Troubleshooting: The big picture

Figure 11.1 provides an overview of the different areas where you may encounter problems that need troubleshooting. Each of these will be discussed in more detail in the next sections.

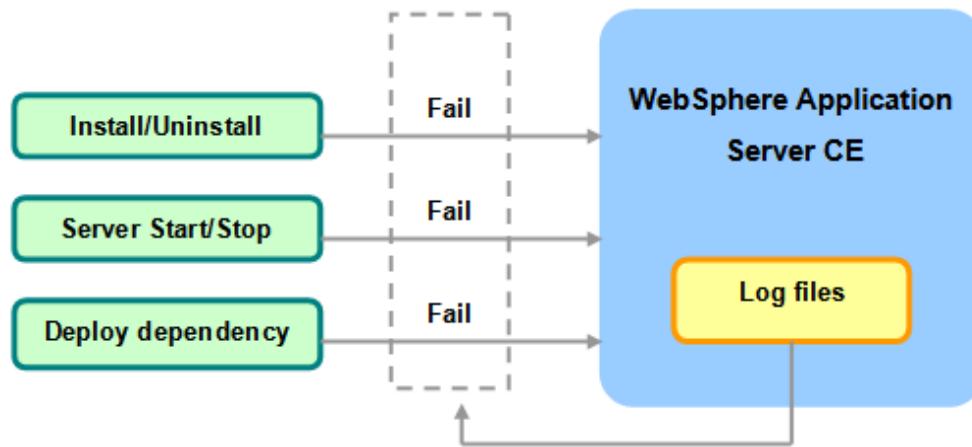


Figure 11.1 - Troubleshooting: The big picture

11.2 Problems during installation/un-installation

The process of installing or uninstalling Community Edition is normally straight-forward; however there may be situations as described in this section where you may encounter problems.

11.2.1 JVM not found

The install or uninstall wizard needs a Java run time to be launched. It is strongly recommended that the `JAVA_HOME` environment variable is set. Otherwise, you can specify the Java run time (JVM) location by invoking the launcher from a command prompt or terminal window with the `LAX_VM <javaPath>` wizard option where `javaPath` is the fully qualified location of your Java executable file. For example,

```
wasce_setup-2.1.1.3-win.exe LAX_VM c:\java
```

tells the installer the JVM location is `c:\java`. For more information about the `LAX_VM` option, review the online manuals at

<http://publib.boulder.ibm.com/wasce/V2.1.1/en/setup-reference.html#Setupreference-cmdwiz>

11.2.2 Platform-specific Problem

On different platforms, users may have different problems. Listed below are some common problems in the Linux platform:

11.2.2.1 Linux dependent packages are missing

The GUI installation wizard has dependencies on legacy compatibility libraries (`libstdc++.so.5` and `libXp.so.6`) that must be installed on the target system. Some Linux distributions have these packages installed by default, but not in others. *Table 11.1* lists the pre-requisite libraries required depending on the Linux distribution.

Platform	Prerequisite
32-bit openSUSE 11	<code>libXp-1.0.0-8</code> and <code>compat-libstdc++-33-3.2.3</code>
64-bit openSuSE 11	<code>32-bit or 64-bit libXp-1.0.0-8</code> <code>32-bit or 64-bit compat-libstdc++-33-3.2.3</code>
32-bit RHEL 5	<code>libXp-1.0.0-8</code>
32-bit Fedora 7 & 8	<code>compat-libstdc++-33-3.2.3</code>
32-bit Asianux Server 3	
64-bit RHEL 5,	<code>32-bit or 64-bit libXp-1.0.0-8</code>
64-bit Fedora and Fedora 8,	<code>32-bit or 64-bit compat-libstdc++-33-3.2.3</code>

64-bit Asianux Server 3	
32-bit RHEL 4,	xorg-x11-deprecated-libs-6.8.1
32-bit Asianux Server 2	compat-libstdc++-33-3.2.3
64-bit RHEL 4,	32-bit or 64-bit xorg-x11-deprecated-libs-6.8.1
64-bit Asianux Server 2	32-bit or 64-bit compat-libstdc++-33-3.2.3

Table 11.1 - Pre-requisites for the Linux platform depending on the distribution

11.2.2.2 Installing on the Ubuntu Linux distribution

To install Community Edition on Ubuntu, if the user runs the server installer in GUI (swing) mode, two prerequisites should be met:

- The legacy compatibility libraries (libstdc++.so.5 or higher) must be installed.
- Close the visual effect before launching ISMP installer as follows:
 - Click *System-> Preferences-> Appearance* and the *Visual Effects* tab
 - Select *None*, and click *OK*.

11.2.3 Uninstalling Community Edition doesn't remove all the files

Sometimes, there are some files left after you uninstall Community Edition. These files might be generated by applications deployed on Community Edition. To remove all the files in the installation location, make sure you have the checkbox *Remove All* checked in the first dialog of the un-installation wizard.

11.3 Problems starting or stopping the Community Edition server

If you have problems starting or stopping Community Edition, probably a variable was not correctly set, or there are port conflicts. This section describes the most common issues you can encounter.

11.3.1 JAVA_HOME or JRE_HOME environment variable is not specified

During installation, the install wizard searches for a supported Java environment by looking in "well known" locations. When found, this location is written to the <WASCE_HOME>/bin/setenv script. This type of error appears if the expected Java environment is no longer at that location. Most likely the Java environment was uninstalled, damaged, moved, or deleted.

To solve this problem, examine the <WASCE_HOME>/bin/setenv script and find the location given by the variable <WASCE_JAVA_HOME>. You can either install a supported Java environment at that location, or change <WASCE_JAVA_HOME> to another Java environment.

Community Edition supported environments are documented at:

<http://www.ibm.com/support/docview.wss?rs=2359&uid=swg27006834>

11.3.2 Port already in use

This problem is often accompanied with a `java.net.BindException`, when a user starts the Community Edition server. It means the port number Community Edition wants to use is already used by another application.

First, make sure that the server is not already running. If the server is already running, the second attempt to start the server will fail because the server's ports are already being used.

Next, examine the message text associated with the exception to determine the port number that was already in use. To find details about these messages you can review the server log located at <WASCE_HOME>/var/log/server.log

Then determine which application has caused the conflict and determine how to resolve the conflict. Consider simply stopping the application that caused the conflict or reconfiguring the Community Edition server to use a different port.

To change the port number for the Community Edition server instance, open the `config-substitutions.properties` file under <WASCE_HOME>/var/config/ and change the value of `portOffset`. This was discussed in more detail in *Chapter 9*.

11.3.3 Could not communicate with the server

If you get an error message indicating your application cannot communicate with the Community Edition server, most likely, the Community Edition server is not running. If this is the case, start the server and try again.

Another cause for this problem may be that the Community Edition server is not listening on the port number used by the application. Normally **remote method invocation** (RMI) is used, but maybe the RMI port configured at the application does not match the one configured at the Community Edition server. By default, the server is configured to listen on port number **1099** on all network interfaces for RMI requests.

To resolve this problem set up RMI to use the local loopback interface (127.0.0.1) for development or test machines by adding the following to your `JAVA_OPTS` environment variable as shown below:

```
-Djava.rmi.server.hostname=127.0.0.1
```

Also, make sure the server and the utility are using matching port numbers. To find the port number used by the server, examine <WASCE_HOME>/var/config/config.xml and find the value of the port attribute in the gbean tag where `name="RMIRRegistry"`. Make sure the same value is passed to the application.

Yet another cause of this problem may be that your server is using DHCP and there was a DHCP renewal. If that's the case, you can use a static IP address.

11.4 Classpath and dependency

Sometimes the deployment of a module such as WAR, JAR or EAR, fails with a dependency error. The reason may be an incorrect configuration of the dependencies in the deployment plan.

In the deployment plan, the `<dependencies>` element is used to specify that your module depends on a package kept in the server's repository. The specified package is added to your module's classpath. Begin with the template shown in *Listing 11.1* and update it appropriately.

```
<dependencies>
    <dependency>
        <groupId>group</groupId>
        <artifactId>artifact</artifactId>
        <version>version</version>
        <type>type</type>
    </dependency>
</dependencies>
```

Listing 11.1 - Dependency template

In Listing 11.1:

- **group** is the directory or directory tree in the repository where the unique artifact directory is located.
- **artifact** is the prefix (no version or file extension) of the library's file name
- **version** is the version number to be appended to the file name.
- **type** is the file extension for the file in the repository.

The server will construct a path of the form

```
<WASCE_HOME>/repository/group/artifact/version/file-version.type
```

11.5 Using Community Edition log files as diagnostic tools

The following log files are useful in troubleshooting situations.

11.5.1 Installation and un-installation logs

Both installation and un-installation logs record all the events and actions during the install and un-install process of Community Edition and are useful for analyzing problems during these operations. You can read the logs directly with a text editor.

Both the install and un-install log files are located at <USER_HOME> directory where <USER_HOME> is the user primary directory in the operating system. In Windows, <USER_HOME> is in C:\Document and Settings\<user_id> directory; while in Linux and UNIX it is in /home/<user_id> directory. *Table 11.2* below provides the file names for the install and uninstall log files and their description.

Log file name	Description
wasce_install_information.txt	Records every variable being revoked during installation.
wasce_install_YYYY-MM-DD_HH-MM-SS.log	Records system variable settings and JVM information during installation, where YYYY-MM-DD_HH-MM-SS is a timestamp.
wasce_uninstall_information.txt	Records every variable being revoked during uninstallation.
wasce_uninstall_YYYY-MM-DD_HH-MM-SS.log	Records system variable settings and JVM information during uninstallation, where YYYY-MM-DD_HH-MM-SS is a timestamp.

Table 11.2 - File name for install and uninstall logs

11.5.2 Server log

Messages from the server, applications and deployment are logged to the <WASCE_HOME>/var/log/server.log file. The file location, name, and the log levels can be changed by editing the <WASCE_HOME>/var/log/server-log4j.properties file. For example, if you would like to change the root log level to DEBUG, you can change the **log4j.appenders.FILE.Threshold** property in the server-log4j.properties to this value as shown in *Listing 11.2*.

```
log4j.rootLogger=DEBUG, CONSOLE, FILE
log4j.appenders.CONSOLE=org.apache.log4j.ConsoleAppender
log4j.appenders.CONSOLE.Threshold=${org.apache.geronimo.log.ConsoleLogLevel}
log4j.appenders.CONSOLE.Target=System.out
log4j.appenders.CONSOLE.lausert=org.apache.log4j.PatternLayout
log4j.appenders.CONSOLE.lausert.ConversionPattern=%d{ABSOLUTE} %-5p [%c{1}] %m%n
log4j.appenders.FILE=org.apache.log4j.RollingFileAppender
log4j.appenders.FILE.Threshold=DEBUG
log4j.appenders.FILE.lausert=org.apache.log4j.PatternLayout
log4j.appenders.FILE.lausert.ConversionPattern=%d{ABSOLUTE} %-5p [%c{1}] %m%n
```

Listing 11.2 - Sample of server-log4j.properties file

The server-log4j.properties file can also be edited from the admin console by selecting the *Server Logs* portlet. With the admin console, you can also filter the `server.log` entries.

In Linux, AIX, or Solaris users can use the `tail -f server.log` command to display current entries to the `server.log` file while the server is running.

11.5.3 Client log

Messages from Java EE client applications are written to the file `<WASCE_HOME>/var/log/client.log`. Similar to `server.log`, you can change the file location, name and the log levels by editing the `<WASCE_HOME>/var/log/client-log4j.properties`.

11.5.4 Deployer log

Messages are generated when an application is deployed to a server. These messages are logged to the `<WASCE_HOME>/var/log/deployer.log` file. You can change the file location, name and the log levels in `<WASCE_HOME>/var/log/deployer-log4j.properties`.

11.5.5 Web log

HTTP requests to the Web container are logged. The HTTP request information includes the type of request (GET or POST), the time of access, and other related data. You can see the access log files in the `<WASCE_HOME>/var/catalina/logs` folder. The file name has the format:

```
<client IP address>_access_log.<YYYY-MM-DD>.txt
```

For example, `0.0.0.0_access_log.2010-06-29.txt`, records all the access information on 2010.06.29 from every client.

From the admin console you can review and edit the Web server log file and edit its properties by selecting *Server Logs -> Web Access Log Viewer*. With the admin console, you can use a filter to narrow down log entries.

11.5.6 DB2 database log

DB2 database log files include information about the DB2 database server, such as database startup and shutdown information. There are two main logs:

- DB2 Administration Notification log
- db2diag.log

11.5.6.1 DB2 Administration Notification Log

The DB2 administration notification log provides diagnostic information about errors at the point of failure. On Linux and UNIX platforms, the administration notification log is a text file called <instance name>.nfy (e.g. "db2inst.nfy"). On Windows, all administration notification messages are written to the Windows Event Log.

11.5.6.2 db2diag.log

The db2diag.log provides more detailed information than the DB2 Administration notification log. It is normally used only by IBM DB2 technical support or experienced DBAs. Information in the db2diag.log includes:

- The DB2 code location reporting an error.
- Application identifiers that allow you to match up entries pertaining to an application on the db2diag.logs of servers and clients.
- A diagnostic message (beginning with "DIA") explaining the reason for the error.
- Any available supporting data, such as SQLCA data structures and pointers to the location of any extra dump or trap files.

On Windows (other than Vista), the db2diag.log is located by default under the directory:

```
C:\Documents and Settings\All Users\Application  
Data\IBM\DB2\DB2COPY1\<instance name>
```

On Windows Vista, the db2diag.log is located by default under the directory:

```
C:\ProgramData\IBM\DB2\DB2COPY1\<instance name>
```

On Linux/UNIX, the db2diag.log is located by default under the directory:

```
/home/<instance_owner>/sql1ib/db2dump
```

11.5.7 System.out and System.err

If you start the Community Edition server with the script **startup.bat** (on Windows) or **startup.sh** (On Linux and UNIX), **System.out** and **System.err** messages will be generated. In Windows, a new console will be opened to show all the messages. In Linux, AIX, or Solaris, all the output will be recorded in a log file under <WASCE_HOME>/var/log , the file is named **server.out**.

11.6 Summary

This chapter discussed some common problems users may encounter when working with a Community Edition server, and how to troubleshoot them. The chapter focused on install/uninstall, start/shutdown, and deployment problems. It also discussed how to analyze log files for problem diagnosis. More details about troubleshooting can be found in

the **Troubleshooting** and **FAQ** sections of the online *Community Edition User's Guide* at <http://publib.boulder.ibm.com/wasce/V2.1.1>

You can also post questions in the Community Edition forum at:

<http://www.ibm.com/developerworks/forums/forum.jspa?forumID=541&start=0>

11.7 Review questions

1. How do you uninstall all the files in the Community Edition installation directory?
2. How do you specify the JVM path when you start/stop a Community Edition server?
3. How do you set a port offset for Community Edition?
4. How do you specify an RMI host?
5. Where are the log files of Community Edition?
6. Which option can you use to specify the JVM location when you install a Community Edition server?
 - A. LAX
 - B. VM
 - C. LAX_VM
 - D. JAVA
 - E. JVM
7. Which of the following is not a log file generated by Community Edition installation?
 - A. wasce_install_information.txt
 - B. wasce_install_2010-01-01_01-01-01.log
 - C. wasce_uninstall_information.txt
 - D. wasce_uninstall_2010-01-02_01-01-01.log
 - E. wasce.log
8. Which of the following are solutions to resolve port number conflicts?
 - A. Change the value of **portOffset** in **substitutions.properties**
 - B. Shutdown other applications which were using the port number
 - C. Restart Community Edition server
 - D. Re-install Community Edition
 - E. Uninstall Community Edition

9. How can you start a Community Edition server, if you want to find `System.out` and `System.err` messages?
 - A. Run `startup.sh` in Linux platform
 - B. Run `startup.bat` in Windows platform
 - C. Run “`geronimo run`”
 - D. Run “`start-server.sh`” in Linux platform
 - E. Run “`start-server.bat`” in Windows platform
10. Where can you find `System.out` and `System.err` messages?
 - A. `server.out` in Linux platform
 - B. In a new console in Windows platform
 - C. In `<WASCE_HOME>/var`
 - D. In `<WASCE_HOME>/var/server`
 - E. You cannot find them anywhere

12

Chapter 12 – Advanced features

In addition to implementing the Java EE specifications, Community Edition also provides some extensive features to support flexibility and high-availability of an application server.

In this chapter, you will learn about:

- GShell command line processing environment
- Server assembly customization
- Plug-in management
- Web Application Distribution Infrastructure(WADI) clustering
- Farming deployment

12.1 GShell

GShell is a command-line processing environment that can be used for the execution of Community Edition commands. It is an extensible environment and includes support for editing, command history, and tab completion.

A number of Community Edition administrative commands have been implemented using GShell such as server start/stop, application deploy/undeploy and some other features.

To start GShell, go to <WASCE_HOME>\bin and type the command `gsh.bat` on Windows or `gsh.sh` on Linux. You will see the initialization of GShell as shown in *Listing 12.1* below.

```
gsh
IBM WebSphere Application Server Community Edition (2.1.1.3)

Type 'help' for more information.
-----
Administrator@Server:/> help
```

For information about IBM WebSphere Application Server Community Edition, visit:

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

Available commands:

?	Alias to: help
exit	Exit the shell
clear	Clear the terminal screen
echo	Echo or print arguments to STDOUT
unset	Unset a variable
set	Set a variable
source	Load a file/url into the current shell
print	Alias to: echo
.	Alias to: source
help	Show command help
quit	Alias to: exit

Listing 12.1 – Starting the GShell

Visit the following URL for more details about GShell commands:

<http://publib.boulder.ibm.com/wasce/V2.1.1/en/gshell.html>

12.2 Customizing a new server assembly

Community Edition is a flexible, component-based Java EE server. Most of the time, in our application or production environment, you will not need all the components of a Java EE application server. Users will only require some components critical to their applications available in the server. Community Edition provides two options for users to assemble a new server and customize which components to include:

- **Function-centric:** you can choose the desired set of server plug-ins, such as Web Container, JMS and Java EE application deployment capabilities;
- **Application-centric:** you can choose one or more application plug-ins, and Community Edition server will automatically encapsulate all other necessary plug-ins into the new server assembly.

You can use the administrative console or GShell command `deploy/assemble` to assemble a customized server as needed.

Visit the following URL for more details about customizable server assemblies:

<http://publib.boulder.ibm.com/wasce/V2.1.1/en/creating-custom-server-assemblies.html>

12.3 Plug-in management

The Community Edition server is assembled entirely out of plug-ins. A Community Edition plug-in is usually referred to as a **Configuration archive (.car)** file. You may download a plug-in from a remote running server or a plug-in repository, simply by specifying the

location or URL of the remote repository from the administrative console. In Community Edition, you can also export your applications or existing modules as plug-ins.

A Community Edition plug-in is a jar-structured file that contains a `META-INF/geronimo-plugin.xml` descriptor. A plug-in repository is basically a plug-in catalog together with a maven-structured repository containing plug-ins, which contains a `geronimo-plugins.xml` file to list all available plug-ins in the repository.

Visit the following URL for more details about plug-ins in Community Edition:
<http://publib.boulder.ibm.com/wasce/V2.1.1/en/managing-server-plugins.html>

12.4 WADI Clustering

Web application Distribution Infrastructure (WADI) is a project hosted on codehaus at <http://wadi.codehaus.org/>. WADI solution deals with session state in a cluster with a persistence mechanism in a distributed environment. WADI is integrated in Community Edition to provide an alternative clustering solution.

In a traditional Tomcat clustering environment, the state of each session is replicated among nodes and maintained in memory, which impacts the performance significantly with more nodes added into the cluster. WADI can be used to solve this problem.

To enable WADI clustering in Community Edition, follow the steps below:

1. Stop the server if it is running;
2. Enable relevant WADI modules by editing `<WASCE_HOME>/var/config/config.xml` with keywords `load="true"`;
3. Update the deployment descriptor file, `web.xml`, of the application with keyword `</distributable>`;
4. Update embedded deployment plan `geronimo-web.xml` file of the application with keyword `</Tomcat-clustering-wadi>`.

Visit the following URL for more details about WADI clustering in Community Edition:
<http://publib.boulder.ibm.com/wasce/V2.1.1/en/configuring-wadi-clusters.html>

12.5 Farming deployment

Administering an application in a clustering environment is always painful. To make sure the status of a clustered application is well-maintained, each node in the cluster should be identical, which means users need to deploy the application to be clustered on each node.

In a Community Edition server farm, you can reduce the workload greatly by leveraging **Farming** plug-ins. With this feature, once you deployed an application on one node, the application can be deployed on any other nodes in this cluster.

To enable farming in Community Edition, follow the steps below:

1. Stop the server if it is running;
2. Update the node configuration to make sure that application can be deployed on remotely;
3. Enable relevant Farming modules by editing
`<WASCE_HOME>/var/config/config.xml` with keywords `load="true"`;
4. Update information of nodes in `<WASCE_HOME>/var/config/config.xml`

Visit the following URL for more details about Farming deployment in Community Edition:
<http://publib.boulder.ibm.com/wasce/V2.1.1/en/configuring-wadi-clusters.html>

12.6 Review questions

1. What does WADI stands for?
2. What are the two options that Community Edition provides to assemble a customized server?
3. What is the command to start GShell up?
4. To enable a module in Community Edition, which keyword should be used when updating `config.xml`?
5. With Farming plug-ins enabled, do you have to install the application on each node?
6. Which command-line processing environment is used in Community Edition?
 - A. MS-DOS
 - B. C Shell
 - C. Bash Shell
 - D. G Shell
 - E. None of the above
7. Which of the following is not one of the sub commands used in the GShell environment?
 - A. Geronimo/start-server
 - B. Deploy/listmodules
 - C. Geronimo/start
 - D. Geronimo/stop-server
 - E. None of the above
8. Which is the exact suffix of a plug-in package name in Community Edition?
 - A. .jar

- B. .car
 - C. .ear
 - D. .war
 - E. .zip
9. Which file is used to identify a Geronimo plug-in repository?
- A. Plugin.xml
 - B. Plugins.xml
 - C. Geronimo-plugins.xml
 - D. Geronimo-plugin.xml
 - E. None of the above
10. Which element is used in the web.xml file to indicate that the application is to be clustered?
- A. </clustering>
 - B. </distributable>
 - C. </Tomcat-Clustering-WADI>
 - D. </WADI>
 - E. None of the above

A

Appendix A – Solutions to review questions

Chapter 1 – Introduction

1. Community Edition is based on Geronimo release with customized fixes and updates
2. <http://www.ibm.com/developerworks/forums/forum.jspa?forumID=541>
3. Linux, Windows and the Mac OS X (Beta).
4. OpenJPA, one of top projects in the Apache Software Foundation.
5. Native support for XML, and the ability to access data via SOAP/XML and RESTfull Web services.
6. D Apache Active MQ
7. C. IBM Data Studio
8. E. Servlet 3.0
9. E. None of the above
10. C. WebSphere Application Server

Chapter 2 - Installing

1. You can get the latest Community Edition build from
<http://www.ibm.com/developerworks/downloads/ws/wasce/>
2. Windows, Linux, AIX, Solaris
3. You can use Community Edition and DB2 on the Amazon EC2 Cloud
4. No. You only pay to Amazon for the hardware resources needed, and only when you use them. The cost per hour is fairly low. Visit the Amazon Web Services web site for details.
5. You can try to login at the Administrative Console at <http://localhost:8080/console> or <https://localhost:8443/console>
6. E

7. A
8. B
9. A
10. A, C

Chapter 3 – Development

1. WEP is the server adapter for Community Edition. It facilitates Web application development in Eclipse with Community Edition.
2. IBM Data Studio includes WEP functionality, so you don't need to install the WEP separately. It also includes all the functionality required to work with DB2 and data Web services. You can work immediately with Community Edition and DB2 after installing IBM Data Studio.
3. To deploy the application into Community Edition right-click on the servlet in the Project Explorer, and choose Run As -> Run on Server.
4. Right-click on the application in the Project Explorer, and choose Run As -> Run on Server.
5. Yes. Using deploy command line or deploy it with Admin Console
6. A
7. A
8. A, B, C
9. A, B, C
10. A, B

Chapter 4 – Databases

1. JDBC
2. The steps to use Java API to access database
 - a) Initialize a naming context to look up data source JNDI
 - b) Get connection from database pool and access data
 - c) Close unnecessary database connection after accessing data
3. JDBC type 4 compliant driver db2jcc4.jar
4. java:comp/env/
5. DB2, Oracle, SQL Server, MySQL, Informix
6. C, D
7. B

8. D
9. D
10. C

Chapter 5 – EJB

1. ejbd://localhost:4201
2. [EJB name] + Local and [EJB name] + Remote
3. First, refer to the EJB in the servlet; then, add the EJB to the dependency list of the servlet.
4. First, import or copy the interface file of an EJB to the application client; second, find the EJB with JNDI; finally, invoke the methods in the EJB interface
5. First, create a database pool in Community Edition; second, create the JPA project in Eclipse; then, create a database connection in Eclipse; finally, generate JPA entities
6. A, B, C
7. B, E
8. A, B
9. A, B
10. B

Chapter 6 – Messaging

1. Http is **synchronous** while message is **asynchronous**.
2. **Point-to-Point** messaging model and **Publish/Subscribe** messaging model
3. Use the following steps to develop a JMS client application that produces message:
 - Use JNDI to get Connection Factory and Destination in a created JMS resource Group.
 - Use Connection Factory to create a Connection.
 - Use Connection to create a Session.
 - Use the Session to create a MessageProducer to the Destination.
 - Use the MessageProducer to send message.
4. Create JMS resource group including Connection Factory and Destinations in Community Edition.

5. Use following Steps to use a created JMS resource group in **Web application**
 - Add the created JMS resource group component as a dependency in <environment> section of WEB-INF/geronimo-web.xml.
 - Define the JMS resource mapping in WEB-INF/web.xml.
 - Use JNDI lookup or Resource injection in Servlet to use the JMS resource.
6. C
7. A
8. B
9. A
10. D

Chapter 7 – Web services

1. No. Community Edition is a Java EE 5.0 compliant application server, so there is no difference to develop a SEI and its implement class with others.
2. Community Edition has a built-in plugin named jaxws-tools which based on wsgen and wsimport tools, we can use that to generate stubs and import them into client project.
3. Download axis2 v1.3 and import its lib into eclipse's build path.
4. Use the <servlet> and <servlet-mapping> tag in the standard Java EE deployment descriptor of web project “WEB-INF/web.xml”.
5. **Data** Web Services refer to the ability to wrap Web services around the logic provided by the database
6. C
7. A
8. D
9. E All of the above
10. B Data development project

Chapter 8 – Security

1. Security Realm, a concept of Community Edition, which is actually a user registry definition, containing users and groups which can map to Java EE roles used to protect resources.
2. In general, access to a secured Java EE resource results in a request against the user registry to validate the presented credentials (user ID/password). The user registry by itself does not enforce any security policies or constraints (authorization). It simply validates the supplied user, group, and password

information by the Java EE authentication mechanisms. Role-based security allows the developer to secure access to Java EE resources based on roles. Often the developer has no knowledge of the users and groups that will exist at runtime, so how does the developer refer to the set of users and groups that should have access to a specific artifact? The answer is through roles. Java EE allows the developer to define an arbitrary set of roles and to restrict access to artifacts based on these roles. Later, the developer (or an administrator) can map these roles to actual users and groups.

3. Java EE has two forms of role-based security: **Declarative security**, where the security constraints limiting access are defined in the deployment descriptors and the Java EE runtime is the mechanism that manages access to the Java EE artifact. Consider this as coarse-grained security, where the user will see all or none of the resource. **Programmatic security**, where the user code queries if a user is in a role, and then executes different statements depending on whether the user is in the role or not. Consider this as fine-grained security, where the user might only see a subset of the resource depending on his role.
4. In Community Edition's web console, select **Security -> Users and Groups**.
5. A Java EE (three-tiered) application using Community Edition and DB2 will improve its security and performance when using trusted contexts because the user ID's are always preserved for security, and auditing, they are not "lost" behind the database user ID used by Community Edition to connect to the database (as it would without trusted contexts). In addition, performance is improved because by using trusted contexts, each user ID does not need to create a new database connection.
6. C
7. E
8. A, D
9. B
10. E

Chapter 9 – Administering

1. Use the following steps to start the Community Edition server on Linux platform:
 - Change the current directory into <WASCE_HOME>/bin,
 - Run startup.sh.
2. To override all the default port numbers, one can change the **PortOffset** parameter in <WASCE_HOME>/var/config/config-substitutions.properties.
3. Use following Steps to add a library to the server repository:

- Log on to the administrative console
 - In the left hand navigation pane, select Services → Repository.
 - Click Browse at the right side of File, and select the directory of the JAR library.
 - Specify other fields for the library following the guide in Section 3.4.
 - Click Install.
4. The file <WASCE_HOME>/var/security/groups.properties.
 5. The var and deploy directories.
 6. C
 7. D
 8. B
 9. C
 10. A

Chapter 10 –Tuning

1. Following is the typical steps to tune a system.
 - a) Measure the performance before you apply any changes to the system by using the appropriate monitoring tools.
 - b) Identify possible performance bottlenecks in the whole system.
 - c) Make changes to remove the bottlenecks.
 - d) Return to step 1 to measure the performance.
 - e) The cycle ends when the system reaches an acceptable performance.
2. The default size of **DefaultThreadPool** is 500; the default size of **ConnectorThreadPool** is 30.
3. Update values in wasceHome/var/config/config.xml according to the steps described in section 10.2.3
4. Following is the steps to monitor the memory usage of Community Edition server.
 - a) Start up Community Edition, launch console.
 - b) In the left hand navigation pane, select Server → *Information* to open portlet to monitor the memory usage.
5. Add JAVA_OPTS environment variable, set the value to be
 - Default heap size: -Xms<mem>m
 - Maximum heap size: -Xmx<mem>m

6. A
7. D
8. E
9. D
10. A

Chapter 11 – Troubleshooting

1. Make sure “Remove all” checkbox has been selected in the un-install wizard
2. Replace the <WASCE_JAVA_HOME> with your JVM path in <WASCE_HOME>/bin/setenv script
3. To change the port number for the server instance, open the config-substitutions.properties file under <WASCE_HOME>/instances/var/config/ and change the value of portOffset.
4. Add this arguments “-Djava.rmi.server.hostname=<IP>” into <WASCE_HOME>/bin/setenv script
5. Here are the locations of each log file
 - Install and uninstall log files are in <USER_HOME> directory
 - For windows platform, default <USER_HOME> is “**systemDrive:/Documents and Settings/user/wasce_install.log**”, where **systemDrive** is the drive where you install the Windows system, and **user** is the user name you used to log on.
 - For non-windows platform, default <USER_HOME> is user ID's home directory
 - server.log is in <WASCE_HOME>/var/log
 - client.log is in <WASCE_HOME>/var/log
 - deployer.log is in <WASCE_HOME>/var/log
 - Web server log is in <WASCE_HOME>/var/catalina/logs
 - DB2 database log: Refer to section 11.5.6
6. A
7. E
8. A, B
9. A, B
10. A, B

Chapter 12 – Troubleshooting

1. Web Application Distribution Infrastructure
2. Function-centric and Application-centric
3. gsh.sh for Linux and gsh.bat for Windows
4. load="true"
5. No
6. D
7. C
8. B
9. C
10. B

B

Appendix B – Up and running with DB2

This appendix is a good foundation for learning about DB2. This appendix is streamlined to help you get up and running with DB2 quickly and easily.

In this appendix you will learn about:

- DB2 packaging
- DB2 installation
- DB2 Tools
- The DB2 environment
- DB2 configuration
- Connecting to a database
- Basic sample programs
- DB2 documentation

Note:

For more information about DB2, refer to the free e-book *Getting Started with DB2 Express-C* that is part of this book series.

B.1 DB2: The big picture

DB2 is a data server that enables you to safely store and retrieve data. DB2 commands, XQuery statements, and SQL statements are used to interact with the DB2 server allowing you to create objects, and manipulate data in a secure environment. Different tools can be used to input these commands and statements as shown in *Figure B.1*. This figure provides an overview of DB2 and has been extracted from the *Getting Started with DB2 Express-C* e-book.

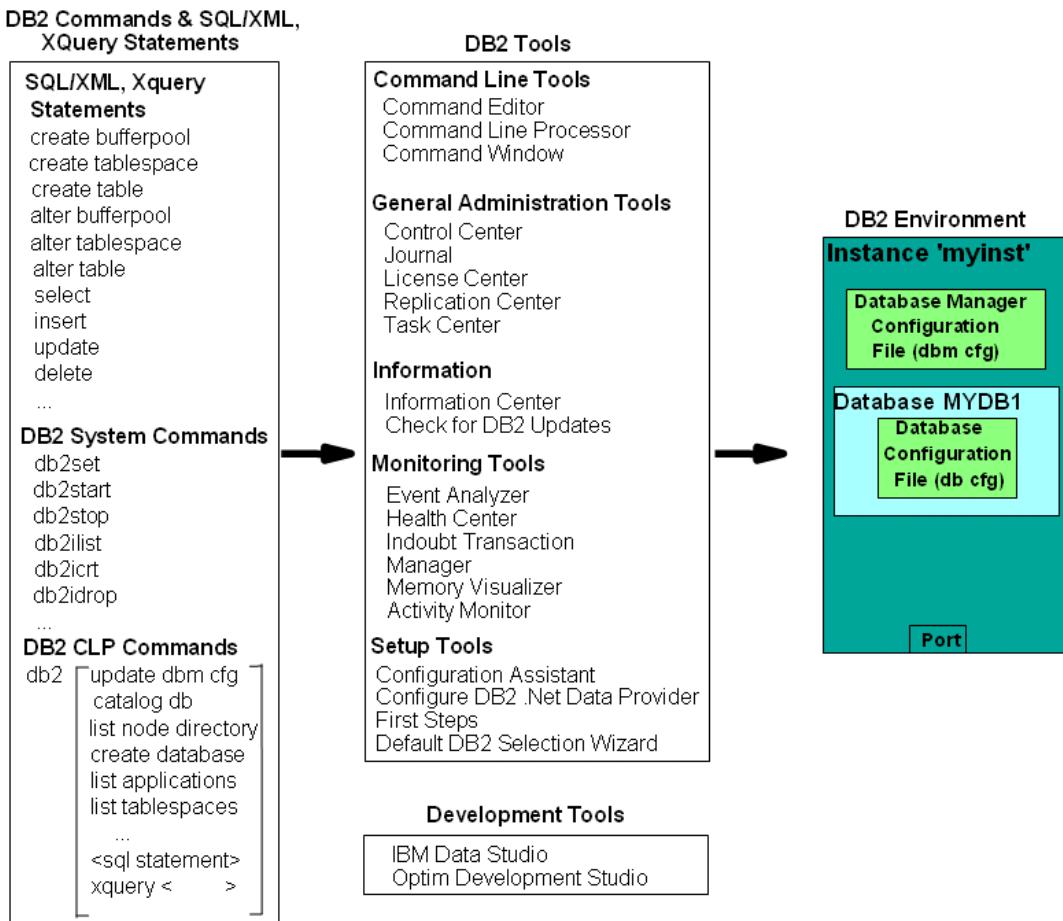


Figure B.1 - DB2 - The big picture

On the left-hand side of the figure, we provide examples of different commands and statements that users can issue. In the center of the figure, we list some of the tools where you can input these commands and statements, and on the right-hand side of the figure you can see the DB2 environment; where your databases are stored. In subsequent sections, we discuss some of the elements of this figure in more detail.

B.2 DB2 Packaging

DB2 servers, clients and drivers are created using the same core components, and then are packaged in a way that allows users to choose the functions they need for the right price. This section describes the different DB2 editions or product packages available.

B.2.1 DB2 servers

Figure B.2 provides an overview of the different DB2 data server editions that are available.

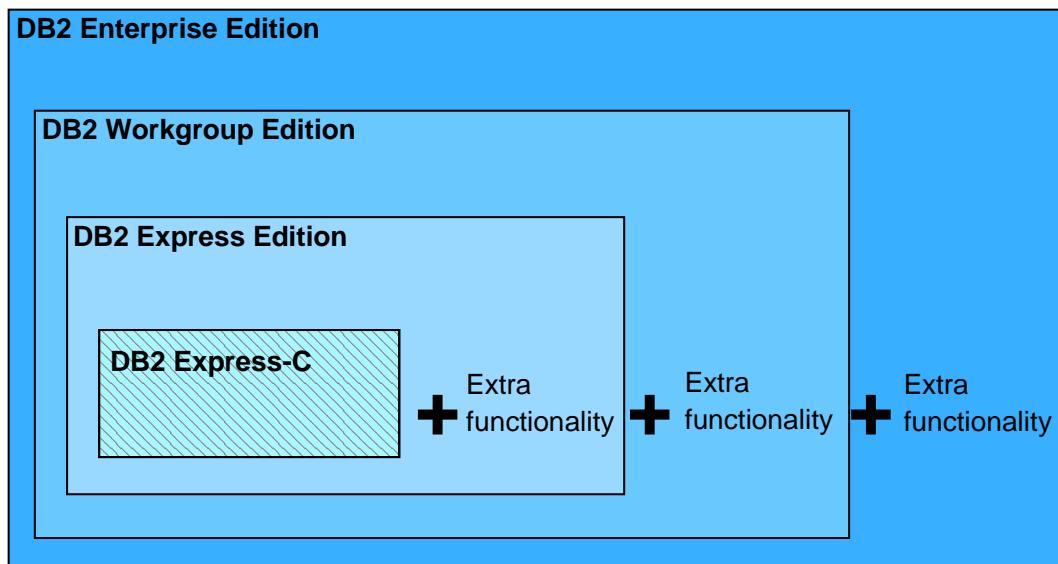


Figure B.2 - DB2 Server Packaging

As shown in *Figure B.2*, all DB2 server editions are built one on top of the other. DB2 Express-C is a free version of DB2, and it is the core component of all DB2 products. When additional functionality is added to DB2 Express-C, it becomes DB2 Express. Additional functionality added to DB2 Express, becomes DB2 Workgroup, and so on. *Figure B.2* illustrates why it is so easy to upgrade from DB2 Express-C to any other DB2 server should you need to in the future: *All DB2 servers editions are built based on DB2 Express-C*.

Also applications built for DB2 Express-C are applicable on other DB2 Editions as well. Your applications will function without any modifications required!

B.2.2 DB2 Clients and Drivers

When you install a DB2 server, a DB2 client component is also installed. If you only need to install a client, you can install either the IBM Data Server Client, or the IBM Data Server Runtime Client. *Figure B.3* illustrates these two clients.



Figure B.3 - DB2 Clients

From the above figure, you can see the IBM Data Server Runtime client has all the components you need (driver and network support) to connect and work with a DB2 Data Server. The IBM Data Server client has this same support and also includes GUI Tools and libraries for application development.

In addition to these clients, provided are these other clients and drivers:

- DB2 Runtime Client Merge Modules for Windows: mainly used to embed a DB2 runtime client as part of a Windows application installation
- IBM Data Server Driver for JDBC and SQLJ: allows Java applications to connect to DB2 servers without having to install a client
- IBM Data Server Driver for ODBC and CLI: allows ODBC and CLI applications to connect to a DB2 server without having to install a client
- IBM Data Server Driver Package: includes a Windows-specific driver with support for .NET environments in addition to ODBC, CLI and open source. This driver was previously known as the IBM Data Server Driver for ODBC, CLI and .NET.

There is no charge to use DB2 clients or drivers.

B.3 Installing DB2

In this section we explain how to install DB2 using the DB2 setup wizard.

B.3.1 Installation on Windows

DB2 installation on Windows is straight-forward and requires the following basic steps:

1. Ensure you are using a local or domain user that is part of the Administrator group on the server where you are installing DB2.
2. After downloading and unzipping DB2 Express-C for Windows from ibm.com/db2/express, look for the file `setup.exe`, and double-click on it.
3. Follow the self-explanatory instructions from the wizard. Choosing default values is normally sufficient.

4. The following is performed by default during the installation:
 - DB2 is installed in C:\Program Files\IBM\SQLLIB
 - The DB2ADMNS and DB2USERS Windows operating system groups are created.
 - The instance **DB2** is created under C:\Program Files\IBM\SQLLIB\DB2
 - The DB2 Administration Server (DAS) is created
 - Installation logs are stored in:
My Documents\DB2LOG\db2.log
My Documents\DB2LOG\db2wi.log
 - Several Windows services are created.

B.3.2 Installation on Linux

DB2 installation on Linux is straight-forward and requires the following basic steps:

1. Log on as the Root user to install DB2.
2. After downloading DB2 Express-C for Linux from ibm.com/db2/express, look for the file db2setup, and execute it: `./db2setup`
3. Follow the self-explanatory instructions from the wizard. Choosing default values is normally sufficient.
4. The following is performed by default during installation:
 - DB2 is installed in /opt/ibm/db2/v9.7
 - Three user IDs are created. The default values are listed below:
`db2inst1` (instance owner)
`db2fenc1` (Fenced user for fenced routines)
`dasusr1` (DAS user)
 - Three user groups are created corresponding to the above user IDs:
`db2iadm1`
`db2fadm1`
`dasadm1`
 - Instance `db2inst1` is created
 - The DAS `dasusr1` is created
 - Installation logs are stored in:
/tmp/db2setup.his
/tmp/db2setup.log
/tmp/db2setup.err

B.4 DB2 tools

There are several tools that are included with a DB2 data server such as the DB2 Control Center, the DB2 Command Editor, and so on. Starting with DB2 version 9.7 however; most of these tools are deprecated (that is, they are still supported but no longer enhanced) in favor of IBM Data Studio. IBM Data Studio is provided as a separate package not included with DB2. Refer to the ebook [Getting started with IBM Data Studio for DB2](#) for more details.

B.4.1 Control Center

Prior to DB2 9.7, the primary DB2 tool for database administration was the Control Center, as illustrated in *Figure B.4*. This tool is now deprecated, but still included with DB2 servers.

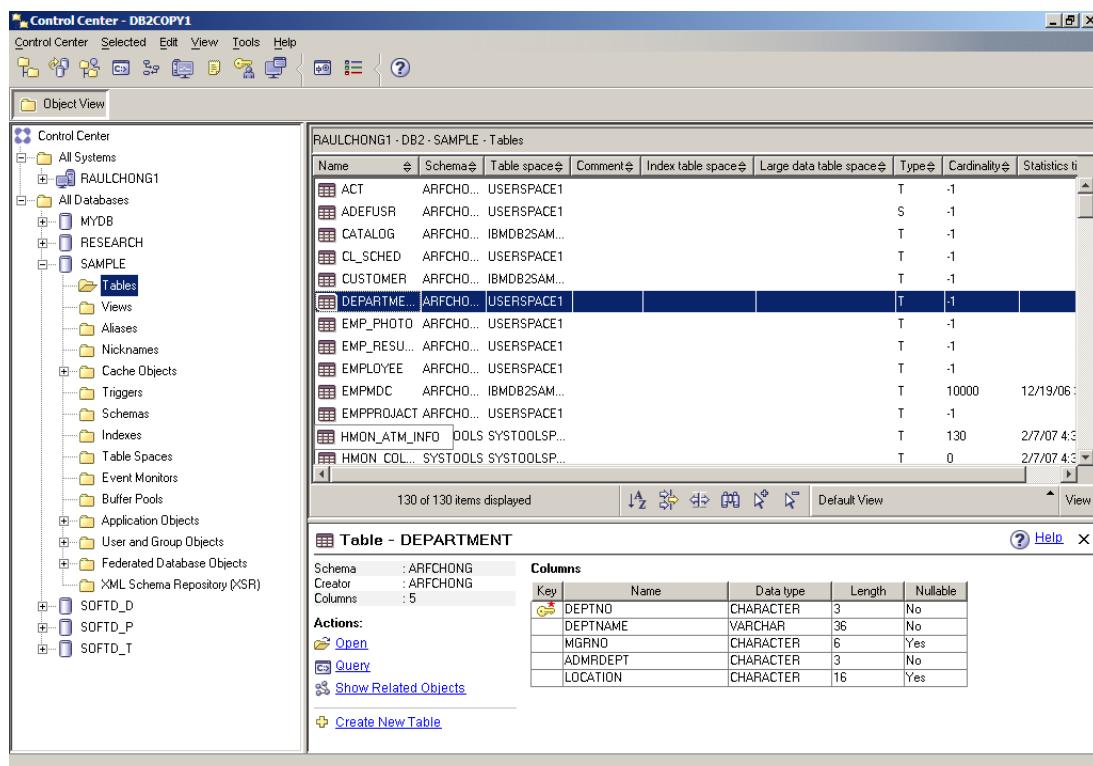


Figure B.4 - The DB2 Control Center

To start the Control Center on Windows use *Start -> Programs -> IBM DB2 -> DB2COPY1 (Default) -> General Administration Tools -> Control Center* or alternatively, type the command `db2cc` from a Windows Command Prompt or Linux shell.

The Control Center is a centralized administration tool that allows you to:

- View your systems, instances, databases and database objects;
- Create, modify and manage databases and database objects;

- Launch other DB2 graphical tools

The pane on the left-hand side provides a visual hierarchy of the database objects on your system(s), providing a folder for Tables, Views, etc. When you double-click a folder (for example, the Tables folder, as shown in *Figure B.5*), the pane on the top right will list all of the related objects, in this case, all the tables associated with the **SAMPLE** database. If you select a given table in the top right pane, the bottom right pane provides more specific information about that table.

Right-clicking on the different folders or objects in the Object tree will bring up menus applicable to the given folder or object. For example, right-clicking on an instance and choosing *Configure parameters* would allow you to view and update the parameters at the instance level. Similarly, if you right-click on a database and choose *Configure parameters*, you would be able to view and update parameters at the database level.

B.4.2 Command Line Tools

There are three types of Command Line tools:

- DB2 Command Window (only on Windows)
- DB2 Command Line Processor (DB2 CLP)
- DB2 Command Editor (GUI-based, and deprecated)

These tools are explained in more detail in the next sections.

B.4.2.1 DB2 Command Window

The DB2 Command Window is only available on Windows operating systems; it is often confused with Windows Command Prompt. Though they look the same, the DB2 Command Window, however, initializes the environment for you to work with DB2. To start this tool, use *Start -> Programs -> IBM DB2 -> DB2COPY1 (Default) -> Command Line Tools -> Command Window* or alternatively, type the command `db2cmd` from a Windows Command Prompt to launch it on another window. *Figure B.5* shows the DB2 Command Window.

The screenshot shows a Windows command-line window titled "DB2 CLP - DB2COPY1". The title bar and the first command line ("db2 connect to sample") are circled in red. The window displays the following text:

```
C:\Program Files\IBM\SQLLIB\BIN>db2 connect to sample
Database Connection Information
Database server      = DB2/NT 9.7.0
SQL authorization ID = ARFCHONG
Local database alias = SAMPLE

C:\Program Files\IBM\SQLLIB\BIN>db2 select * from staff
+----+-----+-----+-----+-----+-----+-----+
| ID | NAME  | DEPT | JOB   | YEARS | SALARY | COMM |
+----+-----+-----+-----+-----+-----+-----+
| 10 | Sanders | 20 | Mgr   | 7    | 98357.50 | -       |
| 20 | Pernal  | 20 | Sales  | 8    | 78171.25  | 612.45  |
| 30 | Marenghi | 38 | Mgr   | 5    | 77506.75  | -       |
| 40 | O'Brien | 38 | Sales  | 6    | 78006.00  | 846.55  |
| 50 | Hanes   | 15 | Mgr   | 10   | 80659.80  | -       |
| 60 | Quigley | 38 | Sales  | -    | 66808.30  | 650.25  |
| 70 | Rothman | 15 | Sales  | 7    | 76502.83  | 1152.00 |
| 80 | James   | 20 | Clerk  | -    | 43504.60  | 128.20  |
| 90 | Koonitz | 42 | Sales  | 6    | 38001.75  | 1386.70 |
| 100 | Plotz  | 42 | Mgr   | 7    | 78352.80  | -       |
| 110 | Ngan   | 15 | Clerk  | 5    | 42508.20  | 206.60  |
+----+-----+-----+-----+-----+-----+-----+
```

Figure B.5 - The DB2 Command Window

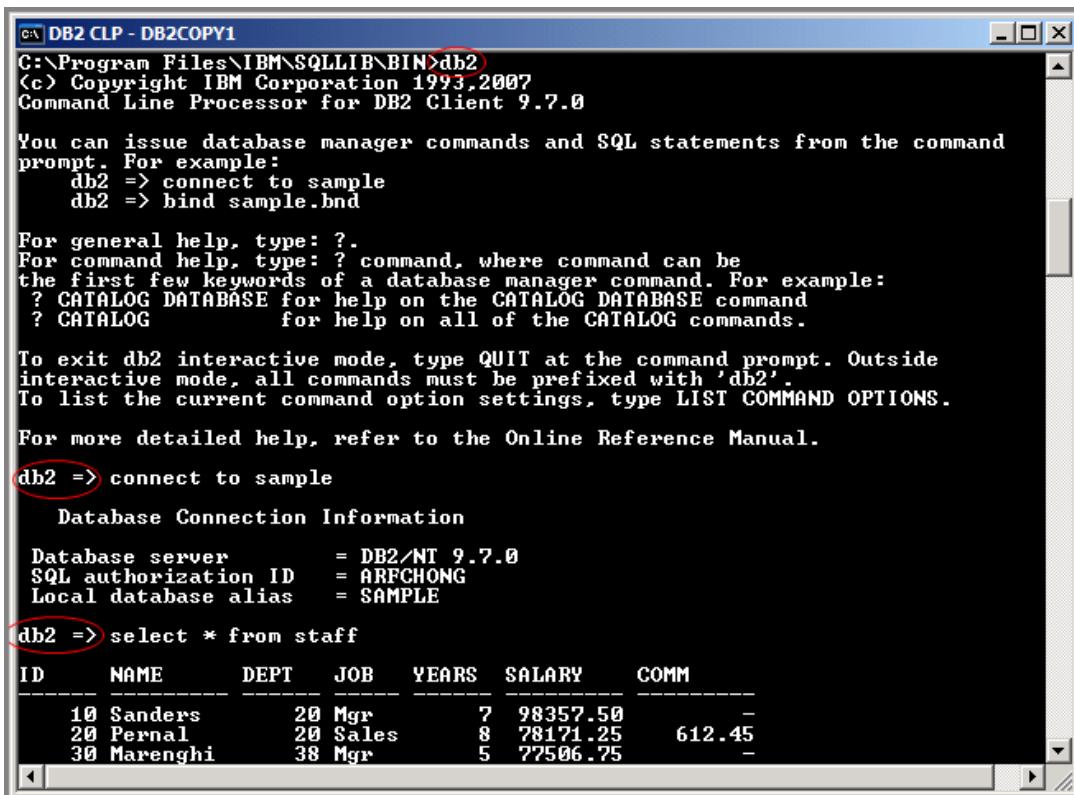
You can easily identify you are working in the DB2 Command Window by looking at the window title which always includes the words *DB2 CLP* as highlighted in the figure. From the DB2 Command Window, all commands must be prefixed with *db2*. For example, in the above figure, two statements are issued:

```
db2 connect to sample
db2 select * from staff
```

For Linux, the equivalent of the DB2 Command Window is simply the Linux shell (or terminal) where the DB2 environment has been set up by executing the *db2profile* file. This file is created by default and added to the *.login* file for the DB2 instance owner. By default the DB2 instance owner is *db2inst1*.

B.4.2.2 DB2 Command Line Processor

The DB2 Command Line Processor (CLP) is the same as the DB2 Command Window, with one exception that the prompt is *db2=>* rather than an operating system prompt. To start the DB2 Command Line Processor on Windows, use *Start -> Programs -> IBM DB2 -> DB2COPY1 (Default) -> Command Line Tools -> Command Line Processor* or alternatively from a DB2 Command Window or Linux shell type *db2* and press *Enter*. The prompt will change to *db2* as shown in *Figure B.6*.



The screenshot shows a Windows command-line interface window titled "DB2 CLP - DB2COPY1". The title bar includes standard window controls (minimize, maximize, close). The main area displays the following text:

```

C:\Program Files\IBM\SQLLIB\BIN>db2
(c) Copyright IBM Corporation 1993,2007
Command Line Processor for DB2 Client 9.7.0

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 => connect to sample

  Database Connection Information

  Database server      = DB2/NT 9.7.0
  SQL authorization ID = ARFCHONG
  Local database alias = SAMPLE

db2 => select * from staff

  ID   NAME    DEPT   JOB  YEARS  SALARY   COMM
  --  --  --  --  --  --  --
  10  Sanders  20  Mgr    7  98357.50   -
  20  Pernal   20  Sales   8  78171.25  612.45
  30  Marenghi 38  Mgr    5  77506.75   -

```

The command "db2 => connect to sample" and the resulting connection information are highlighted with red circles.

Figure B.6 - The DB2 Command Line Processor (CLP)

Note that *Figure B.6* also illustrates that when working in the CLP, you do not need to prefix commands with DB2. To exit from the CLP, type **quit**.

B.4.2.3 DB2 Command Editor

The DB2 Command Editor is the GUI version of the DB2 Command Window or DB2 Command Line Processor as shown in *Figure B.7*. This tool is deprecated for DB2 version 9.7.

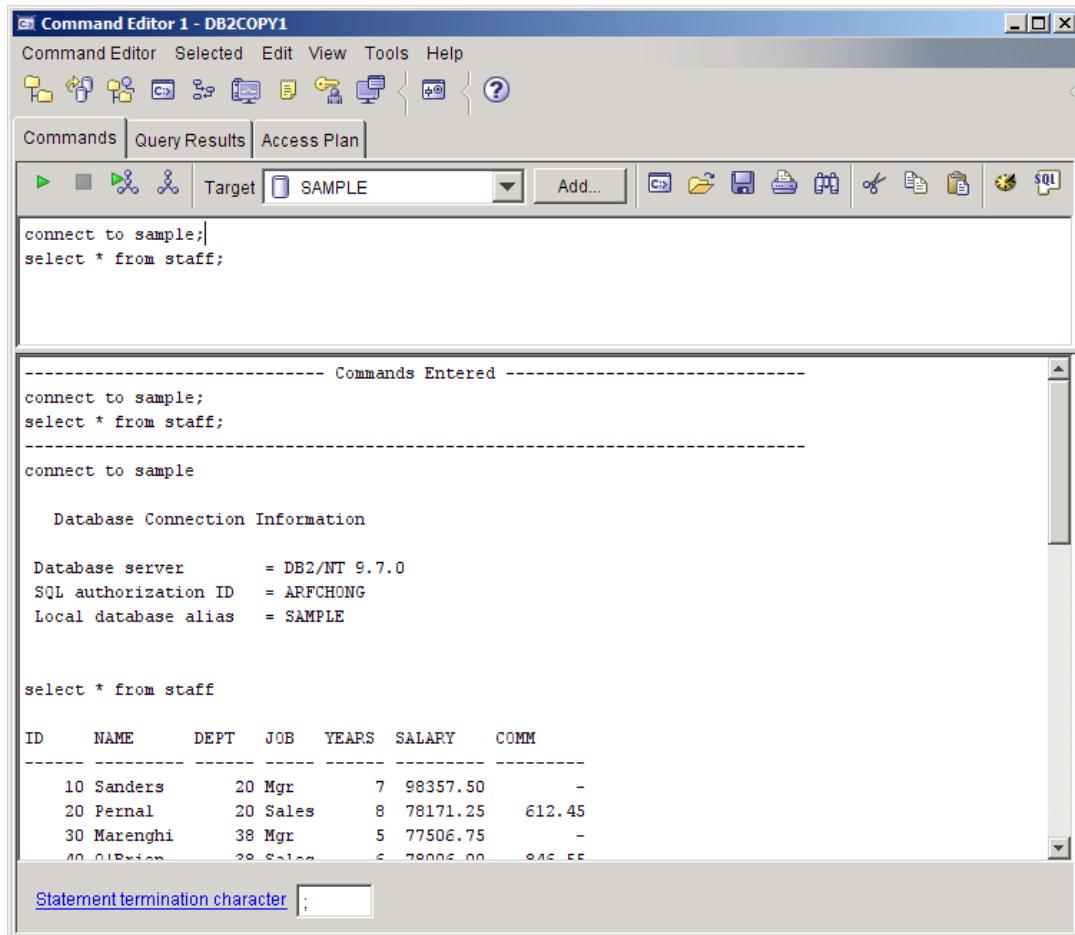


Figure B.7 - The DB2 Command Editor

B.5 The DB2 environment

Figure B.8 provides a quick overview of the DB2 environment.

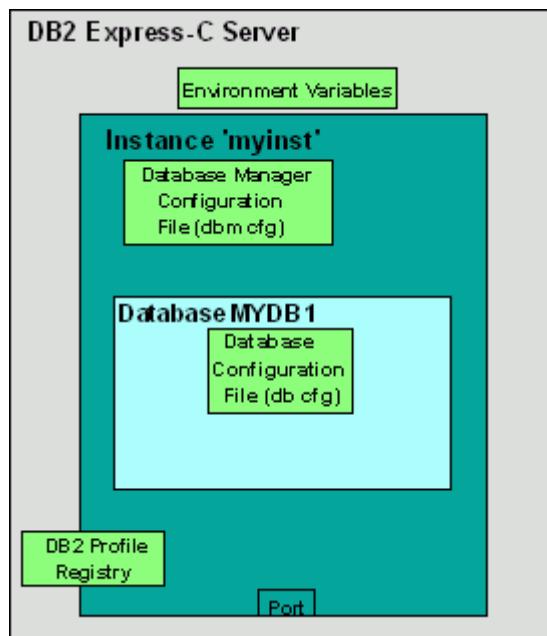


Figure B.8 - The DB2 Environment

The figure illustrates a server where DB2 Express-C has been installed. The smaller boxes in light green (Environment Variables, Database Manager Configuration File, Database Configuration File, DB2 Profile Registry) are the different areas where a DB2 server can be configured, and they will be explained in more detail in the next section. The larger dark green box represents an instance which in this example has the name *myinst*.

An **instance** is an environment where database objects can be created. On the same server, you can create several instances, each of which is treated independently. For example, you can use an instance for development, another one for test, and another one for production. *Table B.1* shows some useful commands you can use at the instance level. Note that the commands shown in this section can also be performed from DB2 GUI Tools.

Command	Description
db2start	Starts the current instance
db2stop	Stops the current instance
db2icrt <instance_name>	Creates a new instance
db2idrop <instance_name>	Drops an instance

db2ilist	Lists the instances you have on your system
db2 get instance	Lists the current active instance

Table B.1 - Useful instance-level DB2 commands

Within an instance you can create many databases. A **database** is a collection of objects such as tables, views, indexes, and so on. For example, in Figure B.8, the database **MYDB1** has been created within instance **myinst**. *Table B.2* shows some commands you can use at the database level.

Command/SQL statement	Description
create database <database_name>	Creates a new database
drop database <database_name>	Drops a database
connect to <database_name>	Connects to a database
create table/create view/create index	SQL statements to create table, views, and indexes respectively

Table B.2 - Commands and SQL Statements at the database level

B.6 DB2 configuration

DB2 parameters can be configured using the Configuration Advisor GUI tool. The Configuration Advisor can be accessed through the Control Center by right clicking on a database and choosing *Configuration Advisor*. Based on your answers to some questions about your system resources and workload, the configuration advisor will provide a list of DB2 parameters that would operate optimally using the suggested values. If you would like more detail about DB2 configuration, keep reading. Otherwise, use the Configuration Advisor and you are ready to work with DB2!

A DB2 server can be configured at four different levels as shown earlier in *Figure B.8*:

- **Environment variables** are variables set at the operating system level. The main environment variable to be concerned about is DB2INSTANCE. This variable indicates the current instance you are working on, and for which your DB2 commands will apply.
- **Database Manager Configuration File (dbm cfg)** includes parameters that affect the instance and all the databases it contains. *Table B.3* shows some useful commands to manage the dbm cfg.

Command	Description
get dbm cfg	Retrieves information about the dbm cfg
update dbm cfg using <parameter_name> <value>	Updates the value of a dbm cfg parameter

Table B.3 - Commands to manipulate the dbm cfg

- **Database Configuration File (db cfg)** includes parameters that affect the particular database in question. *Table B.4* shows some useful commands to manage the db cfg.

Command	Description
get db cfg for <database_name>	Retrieves information about the db cfg for a given database
update db cfg for <database_name> using <parameter_name> <value>	Updates the value of a db cfg parameter

Table B.4 - Commands to manipulate the db cfg

- **DB2 Profile Registry variables** includes parameters that may be platform specific and can be set globally (affecting all instances), or at the instance level (affecting one particular instance). *Table B.5* shows some useful commands to manipulate the DB2 profile registry.

Command	Description
db2set -all	Lists all the DB2 profile registry variables that are set
db2set <parameter>=<value>	Sets a given parameter with a value

Table B.5 - Commands to manipulate the DB2 profile registry

B.7 Connecting to a database

If your database is local, that is, it resides on the same system where you are performing your database operation; the connection setup is performed automatically when the database is created. You can simply issue a `connect to database_name` statement to connect to the database.

If your database is remote, the simplest method to set up database connectivity is by using the Configuration Assistant GUI tool following these steps:

1. Start the Configuration Assistant from the system where you want to connect to the database. To start this tool, use the command `db2ca` from a Windows command prompt or Linux shell. *Figure B.9* shows the Configuration Assistant.

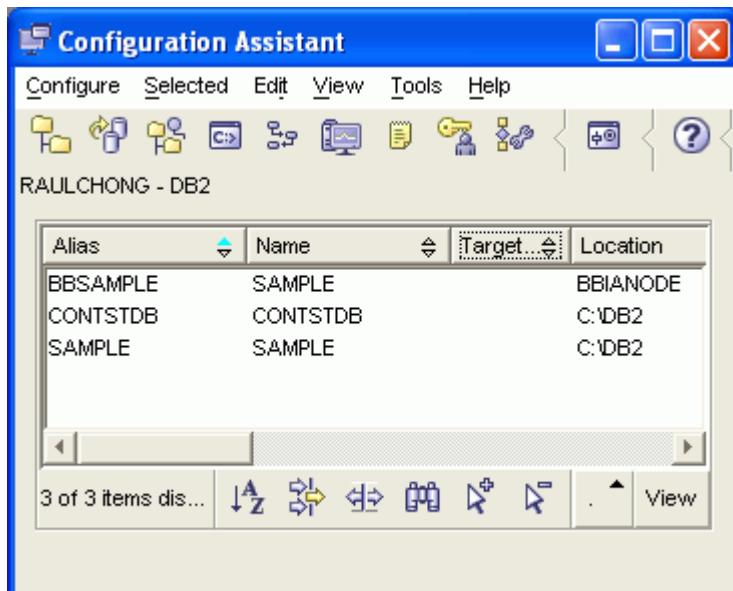


Figure B.9 - The DB2 Configuration Assistant

2. From the Configuration Assistant, click on the *Selected --> Add database using Wizard* menu
3. From the *Select how you want to set up a connection* window, you can use *Search the network* if your network is small without many hubs. If you know the name of the server where DB2 resides, choose *Known systems* and drill down all the way to the database you want to connect. Proceed with the wizard using default values. If you do not know the name of your system, choose *Other systems (Search the network)*. Note that this may take a long time if your network is large.
4. If *Search the network* does not work, go back to the *Select how you want to set up a connection* window, and choose *Manually configure a connection to a database*. Choose TCP/IP and click *next*. Input the *hostname or IP address* where your DB2 server resides. Input either the *service name or the port number*.
5. Continue with the wizard prompts and leave the default values.
6. After you finish your set up, a window will pop up asking you if you want to test your connection. You can also test the connection after the setup is finished by right-clicking on the database, and choosing *Test Connection*.

B.8 Basic sample programs

Depending on the programming language used, different syntax is required to connect to a DB2 database and perform operations. Below are links to basic sample programs which connect to a database, and retrieve one record. We suggest you first download (from <ftp://ftp.software.ibm.com/software/data/db2/udb/db2express/samples.zip>) all the sample programs in this section:

CLI program

<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0401chong/index.html#scenario1>

ODBC program

<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0401chong/index.html#scenario2>

C program with embedded SQL

<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0401chong/index.html#scenario3>

JDBC program using Type 2 Universal (JCC) driver

<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0401chong/index.html#scenario6>

JDBC program using Type 4 Universal (JCC) driver

<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0401chong/index.html#scenario8>

Visual Basic and C++ ADO program - Using the IBM OLE DB provider for DB2 (IBMDADB2)

<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0402chong2/index.html#scenario1>

Visual Basic and C++ ADO program - Using the Microsoft OLE DB Provider for ODBC (MSDASQL)

<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0402chong2/index.html#scenario2>

Visual Basic and C# ADO.Net using the IBM DB2 .NET Data Provider

<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0402chong2/index.html#scenario3>

Visual Basic and C# ADO.Net using the Microsoft OLE DB .NET Data Provider

<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0402chong2/index.html#scenario4>

Visual Basic and C# ADO.Net using the Microsoft ODBC .NET Data Provider

<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0402chong2/index.html#scenario5>

B.9 DB2 documentation

The DB2 Information Center provides the most up-to-date online DB2 documentation. The DB2 Information Center is a web application. You can access the DB2 Information Center online (<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp>), or you can download and install the DB2 Information Center to your local computer. Links to the online DB2 Information Center as well as downloadable versions are available at http://www.ibm.com/software/data/db2/9/download.html?S_TACT=download&S_CMP=exp_csite

C

Appendix C – Using the sample code

To work with the sample code provided with this book, follow these steps:

1. Install Community Edition according to the instructions in *Chapter 2*.
2. Install Eclipse and WebSphere Application Server Community Edition Eclipse Plug-in according to the instructions in *Chapter 3*.
3. Extract the `gettingStartedWithWasceEdition1st_src.zip` to a directory.
4. Launch Eclipse and go to *File > Import > Existing Projects into Workspace*. Select the directory where you extracted the src to, and then select listed projects to import. Click *Finish* to complete the process.
5. Define a new classpath variable inside Eclipse. From the Eclipse menu bar, select *Window > Preferences*. Then select the *Java -> Build Path -> Classpath Variables* page. Here create a new Variable `WASCE_HOME` and set its value to the full path of your Community Edition installation location.
6. Now you should be able to use the sample source code. Enjoy it!

Resources

Web sites

1. **WebSphere Application Server Community Edition home page**
<http://www-01.ibm.com/software/webservers/appserv/community/>
This site is the home page of Community Edition.
2. **Free Community Edition download**
<http://www.ibm.com/developerworks/downloads/ws/wasce/>
Use this web site to download Community Edition, J2EE Samples, etc.
3. **Free Eclipse based IDE for Community Edition download**
<http://download.boulder.ibm.com/ibmdl/pub/software/websphere/wasce/updates/>
Use this web site to download Eclipse based IDE for Community Edition. You can also use this link as Eclipse update manager update site.
4. **DB2 Express-C home page**
ibm.com/db2/express
This site is the home page of DB2 Express-C. You can find links to download the free DB2 Express-C from this page.
5. **IBM Data Studio home page**
<http://www-01.ibm.com/software/data/optim/data-studio/>
This site is the home page of the free IBM Data Studio, an Eclipse-based tool you can use with DB2.
6. **Free DB2 Express-C and IBM Data Studio download**
http://www.ibm.com/db2/express/download.html?S_CMP=ECDDWW01&S_TACT=D_OCBOOK08
7. **Community Edition developerWorks® space**
<http://www.ibm.com/developerworks/spaces/wasce>
Developer Community - Engage with Community Edition users and experts
8. **Community Edition Technical Support**
<http://www-01.ibm.com/software/webservers/appserv/community/support/>
When you download WebSphere Application Server Community Edition you are automatically entitled to limited online support during a 30-day trial period, at no

charge. IBM also offers three support options that enable you to purchase the support level most appropriate for your business and technical needs.

9. Community Edition information center

<http://publib.boulder.ibm.com/wasce/>

The information center provides access to the online manuals. It is the most up to date source of information.

10. Community Edition and Apache Geronimo forum:

<http://www.ibm.com/developerworks/forums/forum.jspa?forumID=541>

Use the forum to post technical questions when you cannot find the answers in the manuals yourself.

11. Developworks Community Edition zone

<http://www.ibm.com/developerworks/websphere/zones/was/wasce.html>

You can find the latest developworks content on Community Edition here.

12. DeveloperWorks Open Source Geronimo Resource

<http://www.ibm.com/developerworks/opensource/top-projects/geronimo.html>

This Web site is an excellent resource for Apache Geronimo developers, as well as for Community Edition developers.

13. Apache Geronimo home page

<http://geronimo.apache.org/>

Most documents of Geronimo also apply to Community Edition.

14. alphaWorks®

<http://www.alphaworks.ibm.com/>

This Web site provides direct access to IBM's emerging technology. It is a place where one can find the latest technologies from IBM Research.

Books

1. Free Redbook: WebSphere Application Server Community Edition 2.0 User Guide
Carla Sadtler, Mohamed Ahmed, Rafael Thomas Goz Coutinho, Gianluca Finocchiaro, Anish Pathadan, Susil Piyanandana
April 2008 - SG24-7585-00
<http://www.redbooks.ibm.com/abstracts/sg247585.html?Open>
2. Free Redbook: Experience Java EE! Using WebSphere Application Server Community Edition 2.1
Ueli Wahli, Charles P Brown, Steven Calello, Rafael Coutinho, Patrick Gan, Cedric Hurst, Maan Mehta
February 2009 - SG24-7639-00
<http://www.redbooks.ibm.com/redpieces/abstracts/sg247639.html?Open>
3. Free ebook: Getting started with DB2 Express-C (3rd Edition)
Raul F. Chong et all - June 2009
<http://www.db2university.com>
4. Free ebook: Getting started with IBM Data Studio for DB2
Debra Eaton et all - Dec 2009
<http://www.db2university.com>
5. Professional Apache Geronimo
Jeff Genender, Bruce Snyder, Sing Li
October 2006 - ISBN: 978-0-471-78543-9
Publisher: Wrox
6. DB2 9 pureXML® Guide
Whei-Jen Chen, Art Sammartino, Dobromir Goutev, Felicity Hendricks, Ippei Komi, Ming-Pang Wei, Rav Ahuja
August 2007 - SG24-7315-01
<http://www.redbooks.ibm.com/abstracts/sg247315.html>

Contact

If you encounter any problems while reading this book, please post it to the Community Edition forum, or send us an email. We will try to answer your questions in a timely manner.

- Community Edition forum:
<http://www.ibm.com/developerworks/forums/forum.jspa?forumID=541>
- Contact emails:
 - General Community Edition mailbox: cebuild@cn.ibm.com
 - General DB2 on Campus program mailbox: db2univ@ca.ibm.com

Getting started with Community Edition couldn't be easier.

Read this book to:

- Find out what Community Edition is all about
- Learn how to develop Java™ EE applications with Community Edition
- Understand how Community Edition interacts with databases such as DB2®
- Learn everyday Community Edition administration tasks
- Tune and troubleshoot Community Edition servers
- Get familiar with some advanced features of Community Edition
- Practice with hands-on exercises

WebSphere® Application Server Community Edition (Community Edition for short) is a lightweight Java EE application server built on Apache Geronimo, the open source application server project of the Apache Software Foundation. Community Edition is free to download and use and delivers the freedom to develop, deploy and distribute Java EE applications without any limitations. You can install Community Edition and get it running in a short period of time.

Community Edition is a member of the IBM® WebSphere Application Server family. It can drive business agility through a choice of innovative performance based foundations. Just like the other members in the family, Community Edition can work seamlessly with DB2 products. Moreover, in conjunction with DB2 Express-C, the free edition of DB2 database server; you can get started with your Java EE applications in no time, and at zero cost!

Get started with WebSphere Application Server Community Edition, and grow your skills from here!

To learn more or download WebSphere Application Server Community Edition, visit:
<http://www-01.ibm.com/software/webservers/appserv/community>

To learn more or download DB2 Express-C, visit ibm.com/db2/express

To socialize and watch related videos, visit channelDB2.com

This book is part of the DB2 on Campus book series, free eBooks for the community. Learn more at db2university.com



Price: 24.99USD