



C o m m u n i t y E x p e r i e n c e D i s t i l l e d

Java EE Development with Eclipse

Second Edition

Develop, debug, test, and troubleshoot Java EE 7 applications rapidly with Eclipse

Ram Kulkarni

[PACKT] open source*
PUBLISHING community experience distilled

Java EE Development with Eclipse

Second Edition

Develop, debug, test, and troubleshoot Java EE 7
applications rapidly with Eclipse

Ram Kulkarni



BIRMINGHAM - MUMBAI

Java EE Development with Eclipse

Second Edition

Copyright © 2015 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: December 2012

Second edition: September 2015

Production reference: 1240915

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78528-534-9

www.packtpub.com

Credits

Author

Ram Kulkarni

Reviewers

Aristides Villarreal Bravo

Jeff Maury

Phil Wilkins

Commissioning Editor

Neil Alexander

Acquisition Editor

Kevin Colaco

Content Development Editor

Nikhil Potdukhe

Technical Editor

Tanmayee Patil

Copy Editors

Tani Kothari

Kausambhi Majumdar

Alpha Singh

Project Coordinator

Izzat Contractor

Proofreader

Safis Editing

Indexer

Tejal Soni

Production Coordinator

Manu Joseph

Cover Work

Manu Joseph

About the Author

Ram Kulkarni has more than two decades of experience in developing software. He has architected and developed many enterprise web applications, client-server and desktop applications, application servers, IDE, and mobile applications. Also, he is the author of *Eclipse 4 RCP Development How-to* published by Packt Publishing. He blogs at ramkulkarni.com.

I would like to thank Kevin Colaco and Nikhil Potdukhe of Packt Publishing for giving me the opportunity to write this book and helping me decide the content and format.

Writing this book has been a long process, and it would not have been possible without the support and patience of my family.

I would like to thank my parents, my wife, Vandana, and son, Akash, for their continuous love and support. This book is dedicated to Vandana and Akash.

About the Reviewers

Aristides Villarreal Bravo is a Java developer and a member of the NetBeans Dream Team and Java User Groups leaders. He lives in Panamá.

He has organized and participated in various national as well as international conferences and seminars related to Java, JavaEE, NetBeans, NetBeans Platform, free software, and mobile devices. He has been a writer of tutorials and blogs on Java, NetBeans, and web developers.

He has participated in several interviews on sites such as NetBeans, NetBeans DZone, and javaHispano. Also, he has been a developer of plugins for NetBeans. He has written technical reviews of many books on PrimeFaces, that includes *Primefaces BluePrints*, *Packt Publishing*.

He is also the CEO of Javscz Software Developers.

I would like to dedicate this to Oris in the sky.

Jeff Maury is currently working as the technical lead of the Java team at SysperTec Communication, a French ISV that offers mainframe integration tools.

Prior to SysperTec Communication, in 1996, he was a cofounder of a French ISV called SCORT, a precursor to the application server concept that offered J2EE-based integration tools.

He started his career in 1988 at Marben Products, a French integration company that specialized in telecommunication protocols. At Marben Products, he started as a software developer and left as an X.400 team technical lead and Internet division strategist.

I would like to dedicate my work to Jean-Pierre ANSART, my mentor, and thank my wife, Julia, for her patience, and my three sons, Robinson, Paul, and Ugo.

Phil Wilkins has spent over 25 years in the software industry working for both multinationals and software startups. He started out as a developer and worked his way up through technical and developmental leadership roles, primarily in Java-based environments. Currently, he is working as an enterprise technical architect in the IT group of a global optical healthcare manufacturer and retailer using Oracle middleware, cloud, and Red Hat JBoss technologies.

Outside his work commitments, he has contributed his technical capabilities to supporting others in a wide range of activities that include developing community websites, providing input and support to people authoring books, developing software ideas and businesses, and reviewing a range of technical books for Packt and other publishers. Also, he is a blogger and a participant in the Oracle middleware community.

When not immersed in work and technology, he spends his downtime pursuing his passion for music and with his wife and two boys.

I'd like to take this opportunity to thank my wife, Catherine, and our two sons, Christopher and Aaron, for their tolerance for the innumerable hours that I've spent in front of a computer contributing to activities for both my employer and other IT-related activities that I've supported over the years.

www.PacktPub.com

Support files, eBooks, discount offers, and more

For support files and downloads related to your book, please visit www.PacktPub.com.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<https://www2.packtpub.com/books/subscription/packtlib>

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can search, access, and read Packt's entire library of books.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

Free access for Packt account holders

If you have an account with Packt at www.PacktPub.com, you can use this to access PacktLib today and view 9 entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface	vii
Chapter 1: Introducing JEE and Eclipse	1
Java Enterprise Edition (JEE)	1
The presentation layer	3
Java Servlet	3
Java Server Pages	3
Java Server Faces	3
The business layer	4
Enterprise Java Beans	4
The enterprise integration layer	5
Java Database Connectivity (JDBC)	5
The Java Persistent API (JPA)	5
Java Connector Architecture (JCA)	5
Web services	6
Eclipse IDE	6
Workspace	7
Plugin	8
Editors and views	8
Perspective	9
Eclipse preferences	9
Installing products	10
Installing Eclipse (Version 4.4)	10
Installing Tomcat	11
Installing the GlassFish server	13
Installing MySQL	15
Installing MySQL on Windows	15
Installing MySQL on Mac OS X	18
Installing MySQL on Linux	20
Creating MySQL users	20
Summary	21

Chapter 2: Creating a Simple JEE Web Application	23
Configuring Tomcat in Eclipse	24
Java Server Pages	29
Creating a dynamic web project	29
Creating JSP	32
Running JSP in Tomcat	39
Using JavaBeans in JSP	42
Using JSTL	47
Implementing login application using Java Servlet	51
Creating WAR	57
Java Server Faces	58
Using Maven for project management	64
Maven views and preferences in Eclipse JEE	66
Creating a Maven project	68
Maven Archetype	69
Exploring the POM	70
Adding Maven dependencies	72
The Maven project structure	75
Creating WAR using Maven	76
Summary	77
Chapter 3: Source Control Management in Eclipse	79
The Eclipse Subversion plugin	79
Installing the Eclipse Subversion plugin	80
Adding a project to an SVN repository	82
Committing changes to an SVN repository	86
Synchronizing with an SVN repository	87
Checking out a project from SVN	88
The Eclipse Git plugin	89
Adding a project to Git	90
Committing files in a Git repository	92
Viewing a file difference after modifications	93
Creating a new branch	94
Committing a project to a remote repository	97
Pulling changes from a remote repository	99
Cloning a remote repository	101
Summary	103
Chapter 4: Creating a JEE Database Application	105
Creating a database schema	106
The script for creating tables and relationships	113
Creating tables in MySQL	115

Creating a database application using JDBC	116
Creating a project and setting up Maven dependencies	116
Creating JavaBeans for data storage	119
Creating JSP to add a course	121
JDBC concepts	123
Creating a database connection	124
Executing SQL statements	125
Handling transactions	128
Using the JDBC database connection pool	129
Saving a course in a database table using JDBC	133
Getting courses from the database table using JDBC	137
Completing the add Course functionality	141
Using Eclipse Data Source Explorer	143
Creating a database application using JPA	147
Creating the user interface for adding a course using JSF	147
JPA concepts	153
Entity	153
EntityManager	154
EntityManagerFactory	154
Creating a JPA application	154
Creating a new MySQL schema	155
Setting up a Maven dependency for JPA	156
Converting a project into a JPA project	157
Creating entities	160
Configuring entity relationships	163
Configuring a many-to-one relationship	164
Configuring a many-to-many relationship	166
Creating database tables from entities	170
Using JPA APIs to manage data	173
Wiring the user interface with a JPA service class	178
Summary	181
Chapter 5: Unit Testing	183
JUnit	184
Creating and executing unit tests using Eclipse EE	185
Creating a unit test case	187
Running a unit test case	190
Running a unit test case using Maven	191
Mocking external dependencies for unit tests	192
Using Mockito	193
Calculating test coverage	198
Summary	202

Chapter 6: Debugging a JEE Application	203
Debugging a remote Java application	204
Debugging a web application using Tomcat in Eclipse EE	205
Starting Tomcat in debug mode	205
Setting breakpoints	206
Running an application in debug mode	208
Performing step operations and inspecting variables	210
Inspecting variable values	212
Debugging an application in an externally configured Tomcat	215
Using Debugger to know the status of a program execution	217
Summary	220
Chapter 7: Creating JEE Applications with EJB	221
Types of EJB	222
Session bean	222
Stateful session bean	222
Stateless session bean	222
Singleton session bean	223
Accessing session bean from the client	223
Creating a no-interface session	223
Accessing session bean using dependency injection	224
Creating session bean using the local business interface	225
Accessing session bean using the JNDI lookup	226
Creating session bean using a remote business interface	228
Accessing a remote session bean	229
Configuring the GlassFish server in Eclipse	230
Creating the CourseManagement application using EJB	233
Creating an EJB project in Eclipse	233
Configuring datasource in GlassFish 4	237
Configuring JPA	240
Creating a JPA entity	245
Creating stateless EJB	247
Creating JSF and managed bean	252
Running the example	255
Creating EAR for deployment outside Eclipse	258
Creating a JEE project using Maven	259
Summary	265
Chapter 8: Creating Web Applications with Spring MVC	267
Dependency injection	268
Dependency injection in Spring	269
Component scopes	273
Installing the Spring Tool Suite	276
Creating a Spring MVC application	277

Creating a Spring project	278
Understanding files created by the Spring MVC project template	279
Spring MVC application using JDBC	283
Configuring datasource	283
Using the Spring JDBCTemplate class	286
Creating the Spring MVC Controller	290
Calling Spring MVC Controller	290
Mapping data using @ModelAttribute	291
Using parameters in @RequestMapping	294
Using the Spring interceptor	295
Spring MVC application using JPA	299
Configuring JPA	299
Creating the Course entity	302
Creating Course DAO and Controller	305
Creating the Course list view	306
Summary	307
Chapter 9: Creating Web Services	309
JAXB	310
JAXB example	311
REST web services	317
Creating RESTful web services using Jersey	318
Implementing the REST GET request	321
Testing the REST GET request in browser	324
Creating a Java client for the REST GET web service	326
Implementing the REST POST request	329
Writing a Java client for the REST POST web service	330
Invoking the POST REST web service from JavaScript	332
Creating the REST web service with Form POST	333
Creating a Java client for the form-encoded REST web service	334
SOAP web services	335
SOAP	336
WSDL	336
UDDI	338
Developing web services in Java	338
Creating a web service implementation class	340
Using the JAX-WS reference implementation (GlassFish Metro)	342
Inspecting WSDL	343
Implementing a web service using an interface	347
Consuming a web service using JAX-WS	348
Specifying an argument name in a web service operation	351
Inspecting SOAP messages	351
Handling interfaces in an RPC-style web service	353
Handling exceptions	355
Summary	355

Chapter 10: Asynchronous Programming with JMS	357
Steps to send and receive messages using JMS	358
Creating queues and topics in GlassFish	361
Creating a JEE project for a JMS application	363
Creating a JMS application using JSP and JSP bean	365
Executing addCourse.jsp	368
Implementing a JMS queue sender class	368
Implementing a JMS queue receiver class	371
Adding multiple queue listeners	374
Implementing the JMS topic publisher	376
Implementing the JMS topic subscriber	378
Creating a JMS application using JSF and managed beans	381
Consuming JMS messages using MDB	387
Summary	390
Chapter 11: Java CPU Profiling and Memory Tracking	391
Creating a sample Java project for profiling	392
Profiling a Java application	394
Identifying resource contention	398
Memory tracking	403
Eclipse plugins for profiling memory	407
Summary	410
Index	411

Preface

Java 2 Enterprise Edition (J2EE) has been used to develop enterprise applications for many years. It provides a standard technique to implement the many aspects of an enterprise application, such as handling web requests, accessing database, connecting to other enterprise systems, and implementing web services. Over the years, it has evolved and made enterprise application development easier than before. Its name has changed as well, from J2EE to JEE, after the J2EE version 1.4. Currently, it is in version 7.

Eclipse is a popular Integrated Development Environment (IDE) for developing Java applications. It has a version specific to the JEE development too, which makes it faster to write code and easier to deploy JEE applications on a server. It provides excellent debugging and unit testing support. Eclipse has a modular architecture, and many plugins are available today to extend its functionality for performing many different tasks.

This book provides you with all the information that you will need to use Eclipse to develop, deploy, debug, and test JEE applications. The focus of this book is to provide you with practical examples of how to develop applications using JEE and Eclipse. The scope of this book is not limited to JEE technologies, but covers other technologies used in the different phases of application development as well, such as source control, unit testing, and profiling.

JEE is a collection of many technologies and specifications. Some of the technologies are so vast that separate books will have to be written on them and many have been already written. This book takes the approach of providing you with a brief introduction to each technology in JEE and provides links for detailed information. Then it moves on to develop sample applications using specific technologies under discussion and explains the finer aspects of the technologies in the context of the sample applications.

This book could be useful to you if you are new to JEE and want to get started with developing JEE applications quickly. You will also find this book useful if you are familiar with JEE but looking for hands-on approach to use some of the technologies in JEE.

What this book covers

Chapter 1, Introducing JEE and Eclipse, explains in brief the different technologies in JEE and where they fit in a typical multitier JEE application. This chapter describes installing Eclipse JEE, Tomcat, GlassFish, and MySQL, which are used to develop sample applications in the later chapters.

Chapter 2, Creating a Simple JEE Web Application, describes the development of web applications using JSP, Servlet, JSTL, and JSF. It also explains how to use Maven for project management.

Chapter 3, Source Control Management in Eclipse, explains how to use the SVN and Git plugins of Eclipse for source code management.

Chapter 4, Creating a JEE Database Application, explains the creation of database applications using JDBC and JPA. You will learn how to execute SQL statements directly using JDBC, map Java classes to database tables, and set relationships between classes using the JPA and database connection pool.

Chapter 5, Unit Testing, describes how to write and run unit tests for Java applications, mock external dependencies in unit tests, and calculate the code coverage.

Chapter 6, Debugging a JEE Application, shows the techniques used to debug JEE applications and covers the debugging support of Eclipse.

Chapter 7, Creating JEE Applications with EJB, describes the use of EJBs to code business logic in the JEE applications. Also, it explains how to connect to remote EJBs using JNDI and inject EJBs into container-managed beans.

Chapter 8, Creating Web Applications with Spring MVC, describes the creation of web applications using Spring MVC and how some of the JEE technologies can be used in a Spring MVC application.

Chapter 9, Creating Web Services, explains the creation of SOAP-based and RESTful web services in JEE applications. You will learn how to consume these web services from JEE applications as well.

Chapter 10, Asynchronous Programming with JMS, shows explains how to write applications to process messages asynchronously. It describes how to program queues and topics of messaging systems using JMS and MDBs.

Chapter 11, Java CPU Profiling and Memory Tracking, describes the techniques for profiling CPU and memory in Java applications to find performance bottlenecks.

What you need for this book

You will need JDK 1.7 or later, Eclipse JEE 4.4 or later, Tomcat 7 or later, GlassFish Server 4 or later, and MySQL Community Server 5.6 or later.

Who this book is for

If you are a Java developer who has little or no experience in JEE application development, or you have an experience in JEE technology but are looking for tips to simplify and accelerate your development process, then this book is for you.

Conventions

In this book, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "We can include other contexts through the use of the `include` directive."

A block of code is set as follows:

```
<body>
  <h2>Login:</h2>
  <form method="post">
    User Name: <input type="text" name="userName"><br>
    Password: <input type="password" name="password"><br>
    <button type="submit" name="submit">Submit</button>
    <button type="reset">Reset</button>
  </form>
</body>
```


When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:


```
try {  
    Thread.sleep(5000);  
} catch (InterruptedException e) {}
```

Any command-line input or output is written as follows:

```
>catalina.bat jpda start
```

New terms and **important words** are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "To set a breakpoint for an exception, select **Run | Java Breakpoint Exception** and select the `Exception` class from the list."

[ Warnings or important notes appear in a box like this.]

[ Tips and tricks appear like this.]

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

To send us general feedback, simply e-mail feedback@packtpub.com, and mention the book's title in the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files from your account at <http://www.packtpub.com> for all the Packt Publishing books you have purchased. If you purchased this book elsewhere, you can visit <http://www.packtpub.com/support> and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books – maybe a mistake in the text or the code – we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/submit-errata>, selecting your book, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to <https://www.packtpub.com/books/content/support> and enter the name of the book in the search field. The required information will appear under the **Errata** section.

Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors and our ability to bring you valuable content.

Questions

If you have a problem with any aspect of this book, you can contact us at questions@packtpub.com, and we will do our best to address the problem.

1

Introducing JEE and Eclipse

Java Enterprise Edition (JEE, which was earlier called J2EE) has been around for many years now. It is a very robust platform for developing enterprise applications. J2EE was first released in 1999, but underwent major changes in version 5, in 2006. Since version 5, it has been renamed **Java Enterprise Edition (JEE)**. Recent versions of JEE has made developing a multi-tier distributed application a lot easier. J2EE had focused on core services and had left the tasks that made application development easier to external frameworks, for example, MVC and persistent frameworks. But JEE has brought many of these frameworks in the core services. Along with the support for annotations, these services simplify application development to a large extent.

Any runtime technology is not good without great development tools. **Integrated Development Environment (IDE)** plays a major part in developing applications faster, and Eclipse provides just that for JEE. Not only do you get a good editing support in Eclipse, but you also get support for build, unit testing, version control, and many other tasks important in different phases of software application development.

The goal of this book is to show how you can efficiently develop JEE application using Eclipse by using many of its features during different phases of the application development. But first, the following is a brief introduction to JEE and Eclipse.

Java Enterprise Edition (JEE)

JEE is a collection of many different specifications intended to perform specific tasks. These specifications are defined by the Java Community Process (<https://www.jcp.org>) program. Currently, JEE is in version 7. However, different specifications of JEE are at their own different versions.

JEE specifications can be broadly classified in the following groups:

- Presentation layer
- Business layer
- Enterprise integration layer

Note that JEE specification does not necessarily classify APIs in such broad groups, but such classification could help in better understanding the purpose of the different standards and APIs in JEE. Before we see APIs in each of these categories, let's understand a typical JEE web application flow where each of these layers fits in.

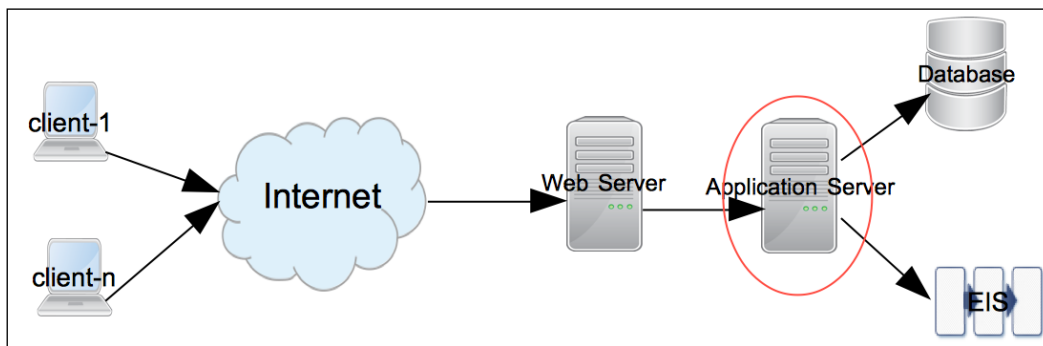


Figure 1.1 A typical JEE web application flow

Requests start from client. Client can be any application requesting services from a remote application – for example, it could be a browser or a desktop application. The request is first received by Web Server at the destination. Examples of Web Servers are Apache Web Server, IIS, Nginx, and so on. If it is a request for static content, then it is served by the web server(s). However, dynamic request typically requires an Application Server to process it. JEE servers are such Application Servers that handle the dynamic requests. Most JEE specification APIs execute in the application server. Examples of JEE application servers are WebLogic, WebSphere, GlassFish, JBoss, and so on.

Most non-trivial JEE applications access external systems such as database or **Enterprise Integration Server (EIS)** for data and process it. Response is returned from the application server to the web server and then to the clients.

The following is the brief description of each of the JEE specifications in different layers of applications that we saw previously. We will see how to use these APIs in more detail in subsequent chapters. However, note that the following is not the exhaustive list of all the specifications in JEE. We will see the most commonly used specifications here. For the exhaustive list, please visit <http://www.oracle.com/technetwork/java/javaee/tech/index.html>.

The presentation layer

JEE specifications or technologies in this group receive the request from web server and send back the response, typically, in an HTML format. However, it is also possible to return only the data from the presentation layer, for example, in **JavaScript Object Notation (JSON)** or **eXtensible Markup Language (XML)** format, which could be consumed by AJAX (Asynchronous JavaScript and XML) calls to update only part of the page, instead of rendering the entire HTML page. Classes in the presentation layer are mostly executed in a Web Container – it is a part of the application server that handles web requests. Tomcat is an example of a popular Web Container.

Now, we will take a look at some of the specifications in this group.

Java Servlet

Java servlets are server side modules, typically used to process a request and send back response in the web applications. Servlets are useful for handling requests that do not generate large HTML markup responses. They are typically used as controllers in MVC (Model View Controller) frameworks, for forwarding/redirecting requests or for generating non-HTML responses, such as PDFs. To generate an HTML response from Servlet, you need to embed the HTML code (as Java String) in the Java code. Therefore, it is not the most convenient option for generating large HTML response. JEE 7 contains Servlet API 3.1.

Java Server Pages

Like Servlets, JSPs are also server side modules used to process the web requests. JSPs (Java Server Pages) are great for handling requests that generate large HTML markup responses. In JSP pages, Java code or JSP tags can be mixed with other HTML code, such as HTML tags, JavaScript, and CSS. Since Java code is embedded in the larger HTML code, it is easier (than Servlet) to generate an HTML response from the JSP pages. JSP specification 2.3 is included in JEE 7.

Java Server Faces

Java Server Faces makes creating user interface on the server side modular by incorporating the MVC design pattern in its implementation. It also provides easy to use tags for common user interface controls that can save states across multiple request-response exchanges between the client and server. For example, if you have a page that posts form data from a browser, you can have JSF save that data in a Java Bean so that it can be used subsequently in the response to the same or different request. JSF also makes it easier to handle UI events on the server side and specify page navigation in an application.

You write the **Java Server Faces (JSF)** code in JSP, using custom JSP tags created for JSF. Java Server Faces API 2.2 is part of JEE 7.

The business layer

The business layer is where you typically write code to handle the business logic of your application. The request to this layer could come from the presentation layer, directly from the client application, or from the middle layer consisting of, but not limited to, web services. Classes in this layer are executed in the application container part of JEE Server. GlassFish and WebSphere are examples of web container plus application container.

Let us take a tour of some of the specifications in this group.

Enterprise Java Beans

Enterprise Java Beans (EJBs) are the Java classes where you can write your business logic. Though it is not a strict requirement to use EJBs to write business logic, they do provide many of the services that are essential in enterprise applications. These services are security, transaction management, component lookup, object pooling, and so on. You can have EJBs distributed across multiple servers and let the application container (also called EJB container) take care of component look up (searching component) and component pooling (useful for scalability). This can improve scalability of the application.

EJBs are of two types:

- **Session beans:** Session beans are called directly by clients or middle tier objects
- **Message driven beans:** Message driven beans are called in response to **Java Messaging Service (JMS)** events

JMS and message driven beans can be used for handling asynchronous requests. In a typical asynchronous request processing scenario, the client puts a request in a messaging queue or a topic and does not wait for immediate response. Server application gets the request message, either directly using JMS APIs or by using MDB. It processes the request and may put response in a different queue or topic to which the client would listen and get the response.

Java EE 7 contains EJB specification 3.2 and JMS specification 2.0.

The enterprise integration layer

APIs in this layer are used for interacting with external (to JEE application) systems in Enterprise. Most applications would need to access database, and APIs to access it fall in this group.

Java Database Connectivity (JDBC)

JDBC is a specification to access relational database in a common and consistent way. Using JDBC you can execute SQL statements and get results on different databases using common APIs. Database specific driver sits between the JDBC call and the database, which translates JDBC calls to database vendor specific API calls. JDBC can be used in both the Presentation and Business layers directly, but it is recommended to separate the database calls from both UI and the business code. Typically, this is done by creating **Data Access Objects (DAO)** which encapsulate logic to access the database.

JEE 7 contains JDBC specification 4.0.

The Java Persistent API (JPA)

One of the problems of using JDBC APIs directly is that you have to constantly map the data between Java Objects and the data in columns of rows in relational database. Frameworks such as Hibernate and Spring have made this process simpler by using a concept known as **Object Relationship Mapping (ORM)**. ORM is incorporated in JEE in the form of **Java Persistent API (JPA)**. JPA gives you the flexibility to map the objects to the tables in relational database and execute the queries with or without using **Structured Query Language (SQL)**. Though when used in the context of JPA, query language is called Java Persistence Query Language. JPA specification 2.1 is a part of JEE.

Java Connector Architecture (JCA)

JCA APIs can be used in JEE applications for communicating with Enterprise Integration Systems, such as SAP, Salesforce, and so on. Just like you have database drivers to broker communication between JDBC APIs and relational database, you have JCA adapters between JCA calls and EIS. Most EIS applications now provide REST APIs, which are lightweight and easy to use, so REST could replace JCA in some cases. However, if you use JCA, you get transaction and pooling support from JEE application server.

Web services

Web services are remote application components that expose self-contained APIs. Broadly, web services can be classified based on the following two standards:

- **Simple Object Access Protocol (SOAP)**
- **Representational State Transfer (REST)**

Web services can play a major role in integrating disparate applications, because they are standard based and platform independent.

JEE provides many specifications to simplify development and consumption of both types of web services, for example, JAX-WS (Java API for XML – web services) and JAX-RS (Java API for RESTful web services).

The preceding are just some of the specifications that are part of JEE. There are many other independent specifications, such as web services, and many enabling specifications, such as dependency injection and concurrency utilities, that we will see in subsequent chapters.

Eclipse IDE

As mentioned earlier, a good IDE is essential for better productivity while coding. Eclipse is one such IDE, which has great editor features and many integration points with JEE technologies. The primary purpose of this book is to show you how to develop JEE applications using Eclipse. So following is a quick introduction to Eclipse, if you are not already familiar with it.

Eclipse is an open source IDE for developing applications in many different programming languages. It is quite popular for developing many different types of Java applications. Its architecture is pluggable – there is a core IDE and many different plugins can be added to it. In fact, support for many languages is added as Eclipse plugins, including support for Java.

Along with editor support, Eclipse has plugins to interact with many of the external systems used during development. For example, source control systems such as SVN and Git, build tools such as Apache Ant and Maven, file explorer for remote systems using FTP, managing servers such as Tomcat and GlassFish, database explorer, memory and CPU profiler, and so on. We will see many of these features in the subsequent chapters.

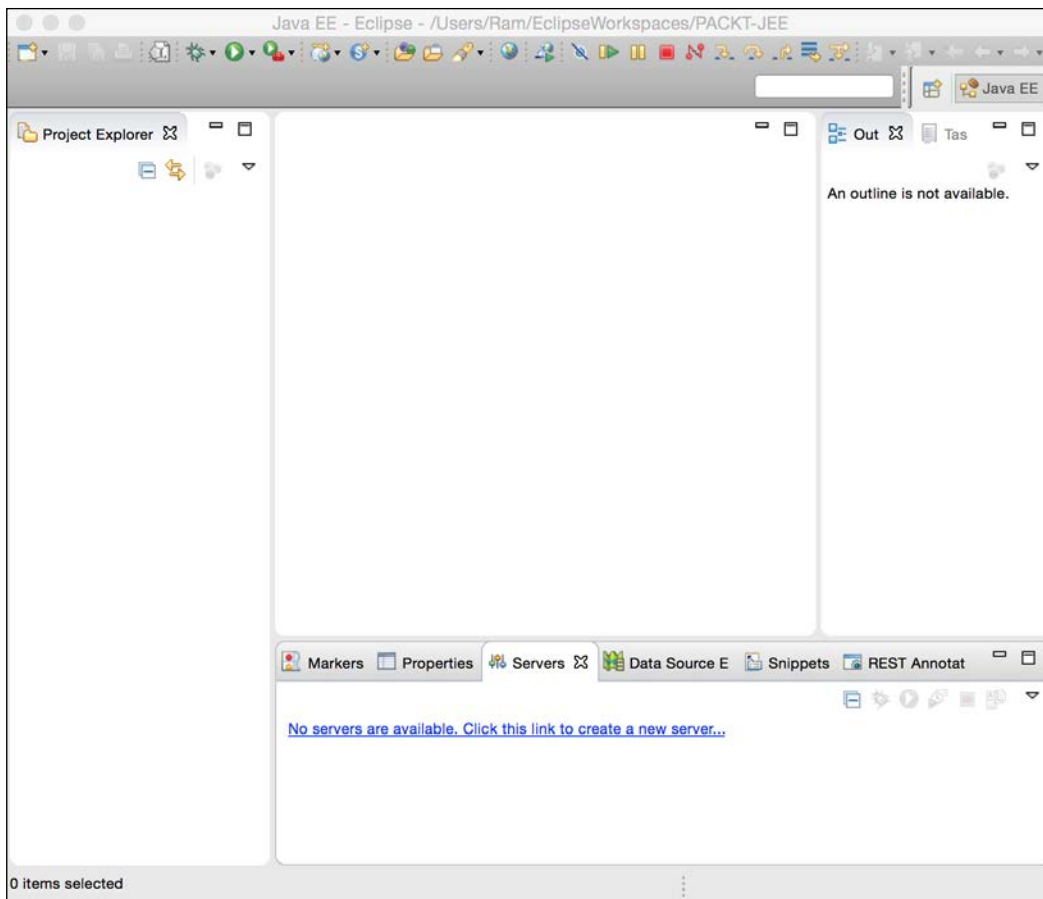


Figure 1.2 Default Eclipse View

Figure 1.2 shows the default view of Eclipse for JEE application development. When working with Eclipse, it is good to understand the following terms used in the context of Eclipse.

Workspace

The Eclipse workspace is a collection of projects, settings, and preferences. Workspace is a folder where Eclipse stores this information. You must create a workspace to use Eclipse. You can create multiple workspaces, but at a time only one can be opened by one running instance of Eclipse. However, you can launch multiple instances of Eclipse with different workspaces.

Plugin

Eclipse has pluggable architecture. Many of the features of Eclipse are implemented as plugins, for example, editor plugins for Java and many other languages, plugins for SVN and Git, and many others. Default installation of Eclipse comes with many built-in plugins and you can add more plugins for the features you want later.

Editors and views

Most windows in Eclipse can be classified either as editor or views. Editor is something where you can change the information displayed in it. View just displays the information and does not allow you to change it. An example of an editor is the Java editor where you write a code. An example of view is the outline view that displays the hierarchical structure of the code you are editing (in case of Java editor, it shows classes in a file, and methods in them).

To see all views in a given Eclipse installation, open the **Window | Show View | Other** menu.

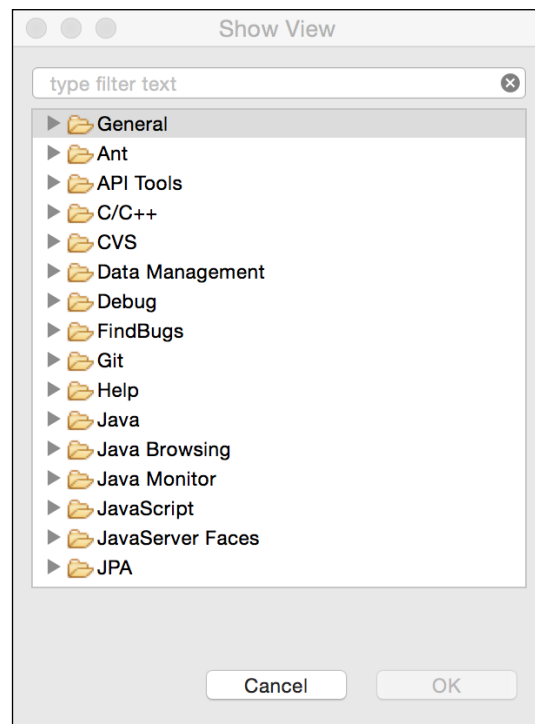


Figure 1.3 Show all Eclipse Views

Perspective

Perspective is a collection of editors and views, and how they are laid out or arranged in the main Eclipse window. At different stages of development, you need different views to be displayed. For example, when you are editing a code, you need to see the **Project Explorer** and **Task** views, but when you are debugging an application, you don't need those views, but instead want to see the variables and breakpoints view. So, the editing perspective displays, among other views and editor, the **Project Explorer** and **Task** view and the Debug perspective displays views and editors relevant to the debugging activities. You can change the default perspectives to suit your purpose, though.

Eclipse preferences

The Eclipse preferences window is where you customize many features of plugins/features. Preferences are available from the **Window** menu in Windows and Linux installation of Eclipse, and from Eclipse menu in Mac installation of Eclipse.

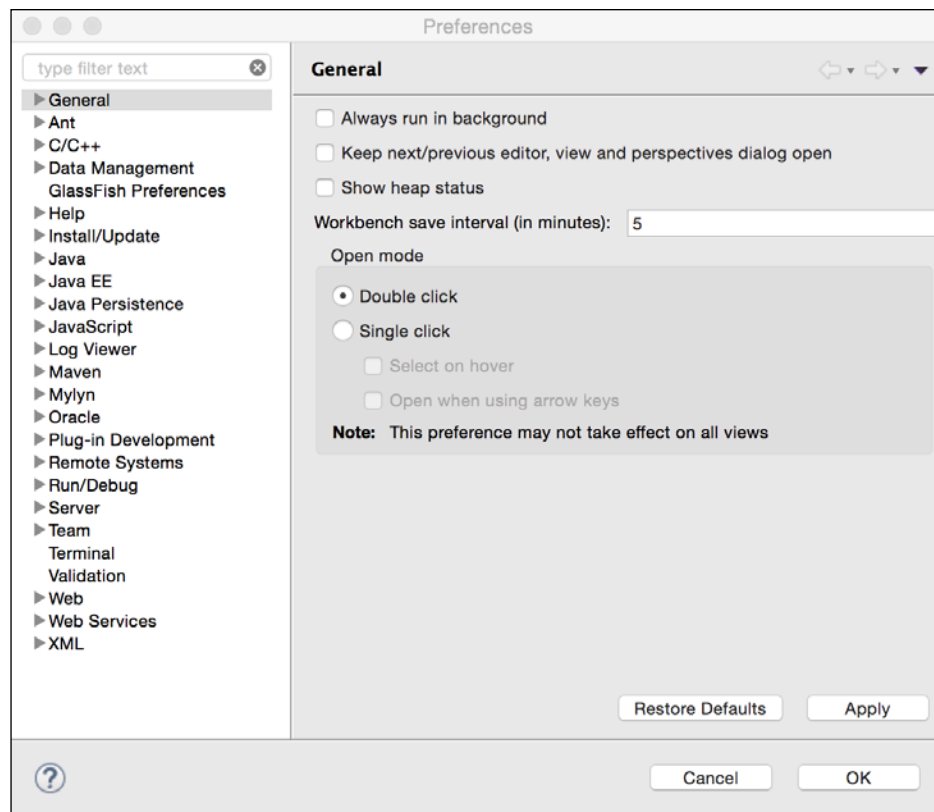


Figure 1.4 Eclipse Preferences

Installing products

In the subsequent chapters, we will see how to develop JEE applications using different APIs in Eclipse. But the applications are going to need a JEE application server and a database. We are going to use Tomcat web container for the initial few chapters and then use GlassFish JEE application server. We are going to use MySQL database. We are going to need these products for most of the applications that we are going to develop. So the following sections describe how to install and configure Eclipse, Tomcat, GlassFish, and MySQL.

Installing Eclipse (Version 4.4)

You can download Eclipse from <https://eclipse.org/downloads/>. You will see many different packages for Eclipse. Make sure you install the **Eclipse IDE for Java EE Developers** package. Select an appropriate package based on your OS and JVM architecture (32 or 64 bit). You may want to run the command `java -version` to know if the JVM is 32-bit or 64-bit.

Unzip the downloaded zip file and then run the Eclipse application (you need to install JDK before you run Eclipse). The first time you run Eclipse, you will be asked to specify a workspace. Create a new folder in your file system and select that as the initial workspace folder. If you intend to use the same folder for workspace on every launch of Eclipse, then check the **Use this as the default and do not ask again** check box.

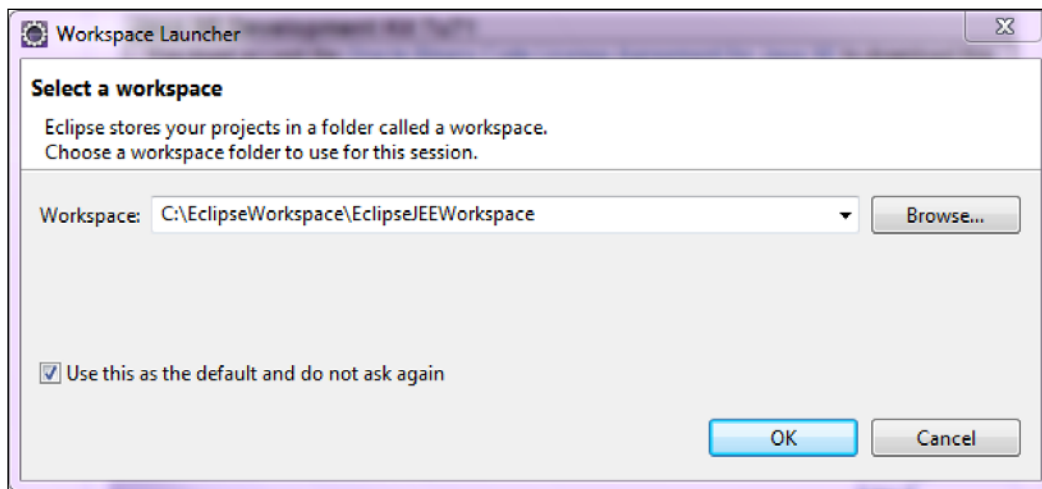


Figure 1.5 Select Eclipse Workspace

You will then see default Java EE perspective of Eclipse as shown in Figure 1.2.

Installing Tomcat

Tomcat is a Web Container. It supports APIs in the presentation layer described earlier. In addition, it supports JDBC and JPA also. It is easy to use and configure, and could be a good option if you do not want to use EJBs.

Download Tomcat from <http://tomcat.apache.org/>. At the time of writing, the latest version of Tomcat available was 8. Download the zip file and unzip in a folder. Set the `JAVA_HOME` environment variable to point to the folder where JDK is installed (the folder path should be the JDK folder, which has `bin` as one of the sub folders). Then run `startup.bat` at the Command Prompt on Windows and `startup.sh` in a Terminal window on Mac and Linux, to start the Tomcat server. If there are no errors, then you should see the message `Server startup in --ms` or `Tomcat started`.

Default Tomcat installation is configured to use port 8080. If you want to change the port, open `server.xml` under the `conf` folder and look for connector declaration like:

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
```

Change the port value to any port number you want, though in this book we will be using the default port 8080. Before we open the default page of Tomcat, we will add a user for administration of the Tomcat server. Open `tomcat-users.xml` under the `conf` folder in any text editor. At the end of the file you will see commented example of how to add users. Add the following configuration before closure of the `</tomcat-users>` tag:

```
<role rolename="manager-gui"/>
<user username="admin" password="admin" roles="manager-gui"/>
```

Here we are adding a user `admin`, with password also as `admin`, to a role called `'manager-gui'`. This role has access to web pages for managing an application in Tomcat. This and other security roles are defined in `web.xml` of the manager application. You can find it at `webapps/manager/WEB-INF/web.xml`. For more information for managing Tomcat server, see <http://tomcat.apache.org/tomcat-8.0-doc/manager-howto.html>.

After making the preceding changes, open a web browser and browse to `http://localhost:8080` (modify port number if you have changed the default port as described previously). You will see the following default Tomcat page:

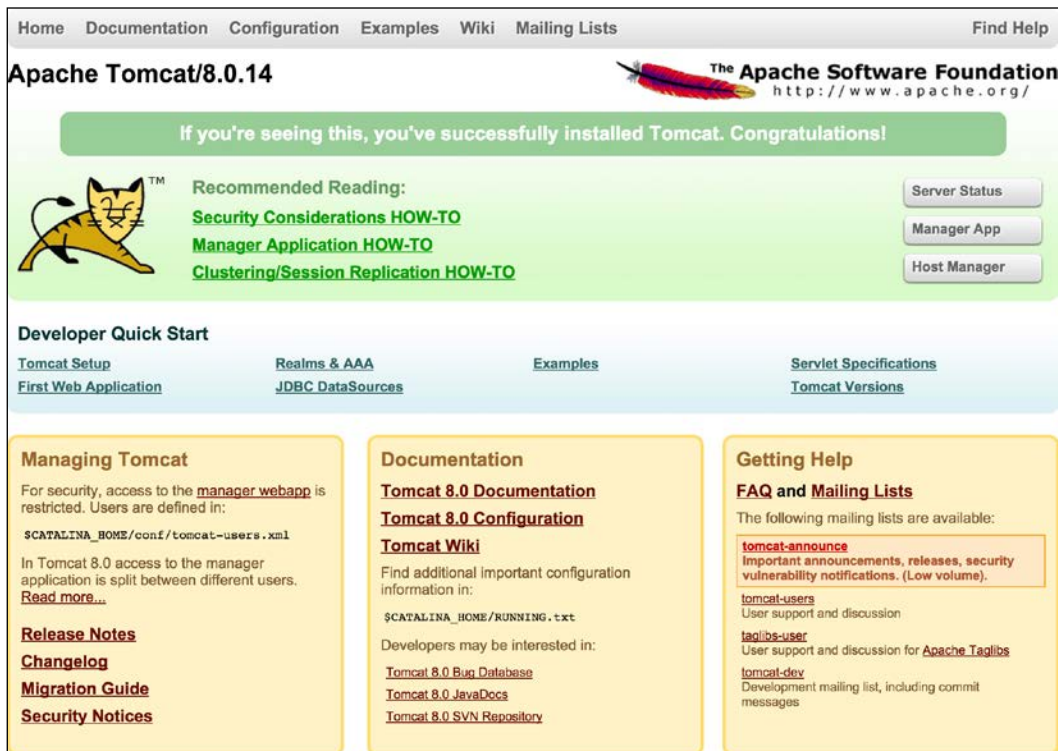


Figure 1.6 The default Tomcat web application

Click on the **Manager App** button on the right. You will be asked for the user name and password. Enter the user name and password you configured in `tomcat-users.xml` for `manager-gui`, as described earlier. After you are successfully logged in, you will see the **Tomcat Web Application Manager** page, as shown in the following image. You can see the applications deployed in Tomcat in this page. You can also deploy your applications from this page.

Tomcat Web Application Manager						
Message:		OK				
Manager						
List Applications		HTML Manager Help		Manager Help		Server Status
Applications						
Path	Version	Display Name	Running	Sessions	Commands	
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes	
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes	
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes	
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes	
/manager	None specified	Tomcat Manager Application	true	2	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes	
Deploy						
Deploy directory or WAR file located on server						
Context Path (required): <input type="text"/> XML Configuration file URL: <input type="text"/> WAR or Directory URL: <input type="text"/> <input type="button" value="Deploy"/>						
WAR file to deploy						
Select WAR file to upload <input type="button" value="Choose File"/> No file chosen <input type="button" value="Deploy"/>						

Figure 1.7 Tomcat Web Application Manager

To stop the Tomcat server, press *Ctrl*/COMMAND + C or run shutdown script in the bin folder.

Installing the GlassFish server

Download GlassFish from <https://glassfish.java.net/download.html>. GlassFish comes in two flavors: Web Profile and Full Platform. Web Profile is like Tomcat, which does not include EJB support. So download Full Platform. See <https://glassfish.java.net/webprofileORfullplatform31x.html> for comparison of Web Profile and Full Platform.

Unzip the downloaded file in a folder. Default port of GlassFish server is also 8080. If you want to change that, open `glassfish/domains/domain1/config/domain.xml` in a text editor (you could open it in Eclipse too, using the **File** | **Open File** menu option) and look for 8080. You should see it in one of the `<network-listener>`. Change the port if you want to (which may be the case if some other application is already using that port).

To start the server, run the `startserv` script (`.bat` or `.sh` depending on the OS you use). Once the server has started, open a web browser and browse to `http://localhost:8080`. You should see a page like the following screenshot:

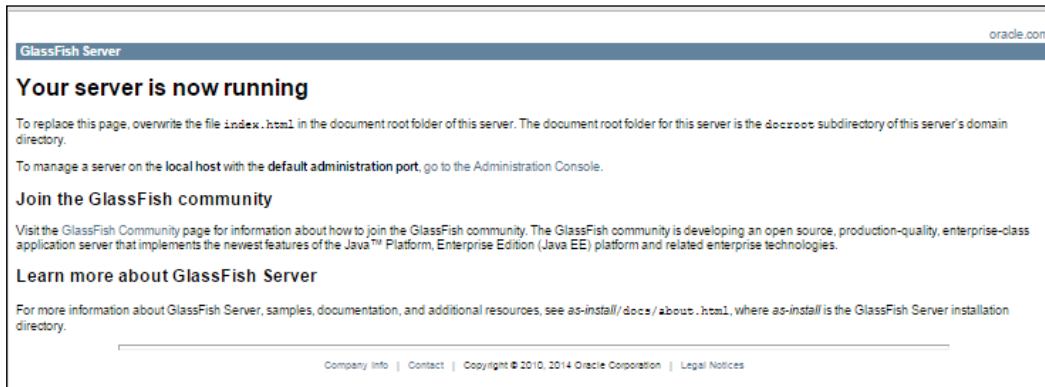


Figure 1.8 The default GlassFish web application

This page is located at `glassfish/domains/domain1/docroot/index.html`. Click on the **go to the Administration Console** link in the preceding page to open GlassFish administrator. For details on administrating GlassFish server, see <https://glassfish.java.net/docs/4.0/administration-guide.pdf>.

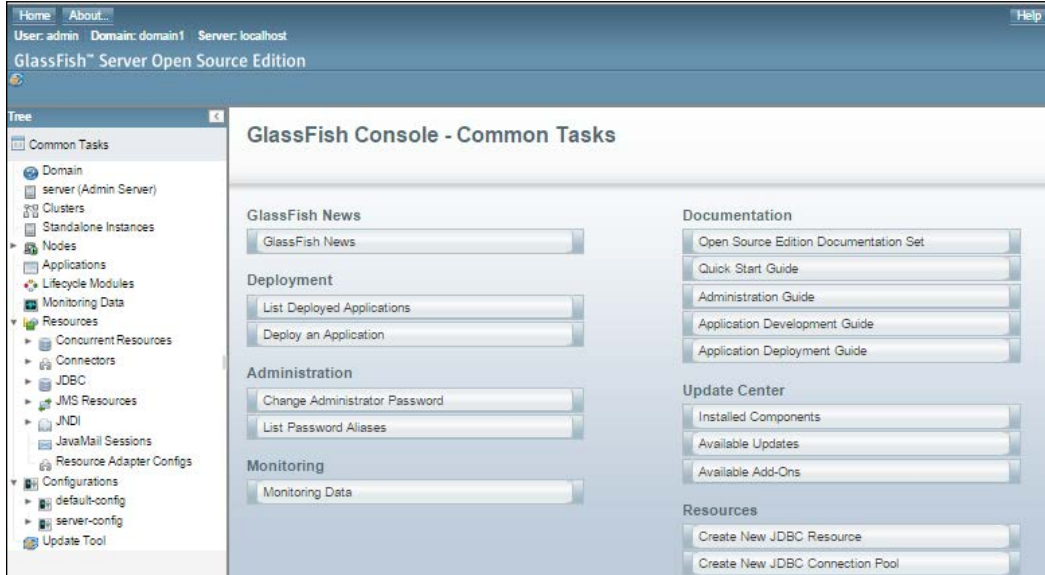


Figure 1.9 The GlassFish administrator

To stop the GlassFish server, run the `stopserv` script in the `glassfish/bin` folder.

Installing MySQL

We will be using MySQL database for many of the examples in this book. Following sections describe how to install and configure MySQL for different platforms.

Installing MySQL on Windows

Download MySQL Community Server from <http://dev.mysql.com/downloads/mysql/>. You can either download the web installer or the all in one installer. The web installer would download only those components that you have selected. Following instructions show the download options using the web installer.

Web installer first downloads a small application, and when you run that, it gives you options to select components that you want to install.

We would like to install MySQL Workbench too, which is a client application to manage MySQL Server. As of writing this chapter, MySQL Workbench required Visual C++ 2013 Runtime for Windows installation. If you don't have it already installed, you can download it from <http://www.microsoft.com/en-in/download/details.aspx?id=40784>.

1. Select the **Custom** option and click on **Next**.

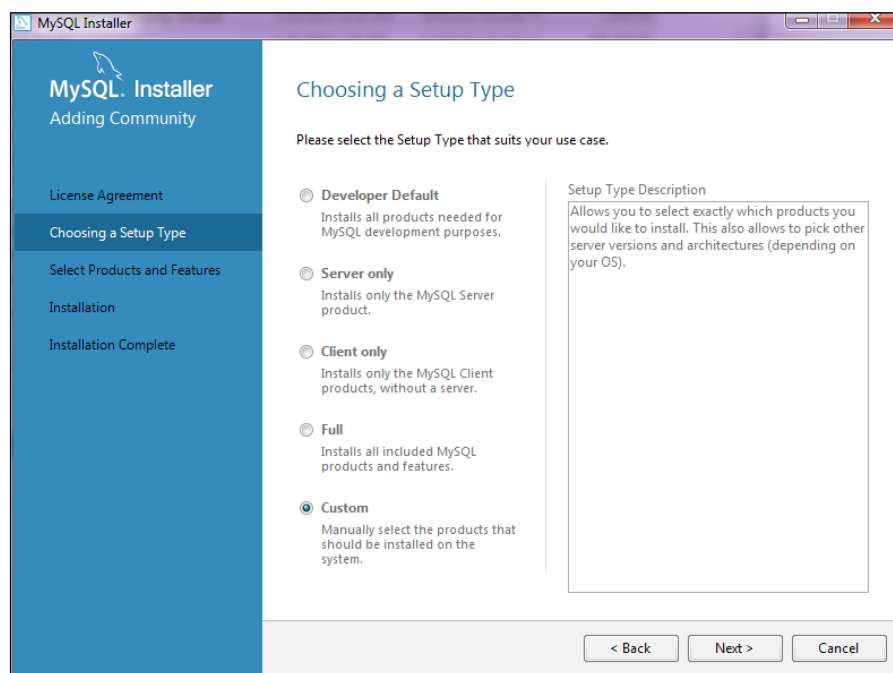


Figure 1.10 MySQL Installer for Windows

2. Select **MySQL Server** and **MySQL Workbench** products and complete the installation. During the installation of Server, you will be asked to set the root password and given the option to add more users. It is always a good idea to add user other than root for applications to use.

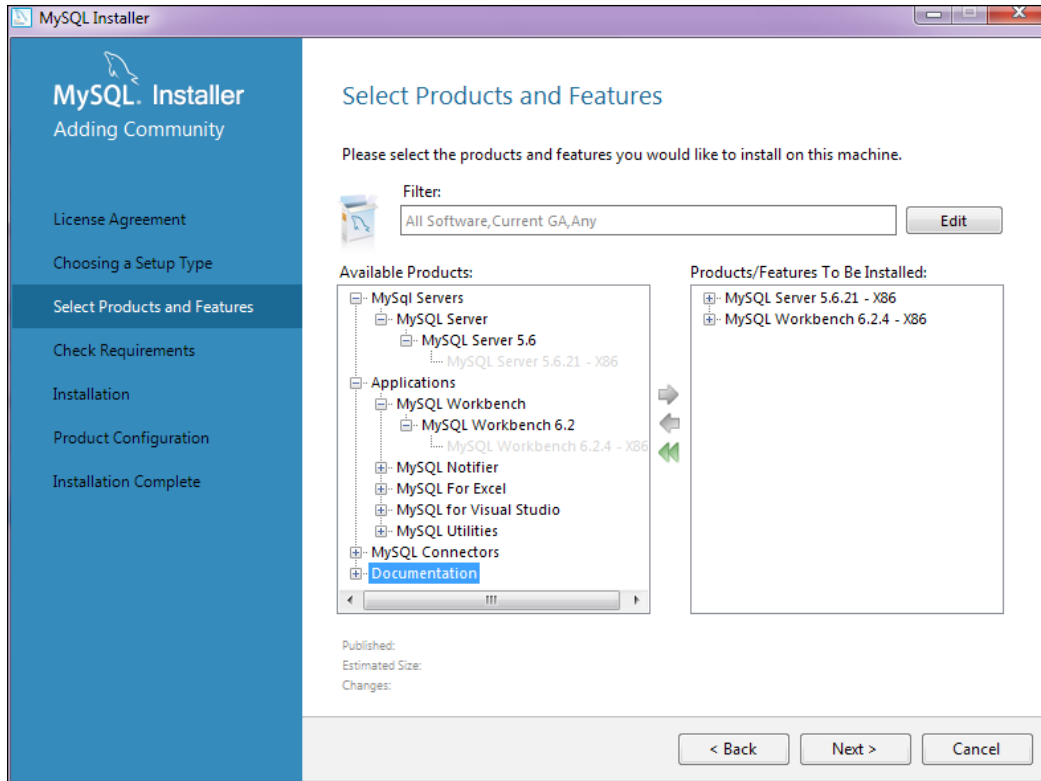


Figure 1.11 Select MySQL Products and Features to Install

3. Make sure you select All Hosts when adding a user so that you are able to access MySQL database from any remote machine that has network access to the machine where MySQL is installed.

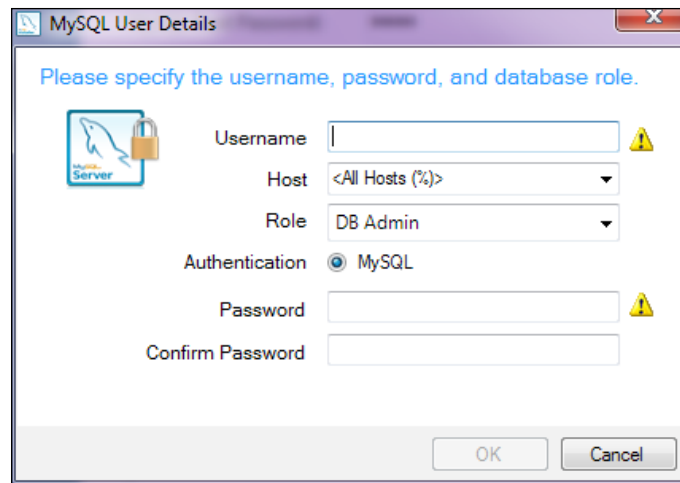


Figure 1.12 Add MySQL User

4. Run MySQL Workbench after installation. You will find that the default connection to the local MySQL instance is already created for you.

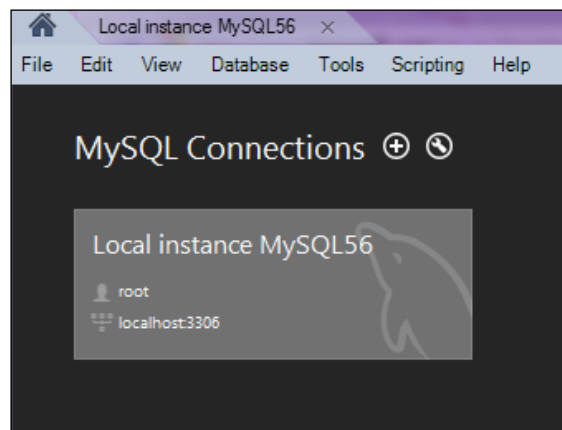


Figure 1.13 MySQL Workbench Connections

5. Click on the local connection and you will be asked to enter the root password. Enter the root password that you typed during the installation of MySQL server. MySQL Workbench opens and displays the default test schema.

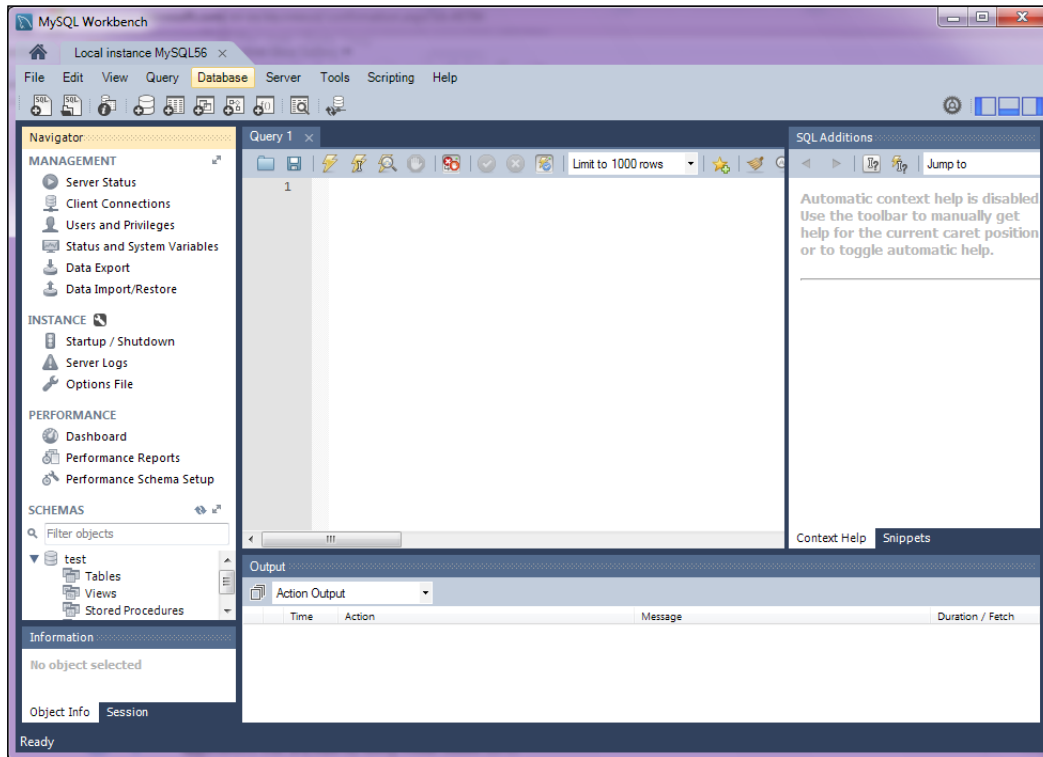


Figure 1.14 My SQL Workbench

Installing MySQL on Mac OS X

OS X versions before 10.7 had MySQL server installed by default. See <http://dev.mysql.com/doc/mysql-macosx-excerpt/5.7/en/macosx-installation-server.html> to know which version of MySQL was installed for different versions of OS X. If you are using OS X 10.7 or later, then you will have to download and install MySQL Community Server from <http://dev.mysql.com/downloads/mysql/>.

There are many different ways to install MySQL on OS X. See <http://dev.mysql.com/doc/refman/5.7/en/osx-installation.html> for installation instruction for OS X. Note that users on OS X should have administrator privileges to install the MySQL server.

Once you install the server, you can start it either from the Command Prompt or from the system preferences.

1. To start it from the Command Prompt, execute the following command in **Terminal**:

```
sudo /usr/local/mysql/support-files/mysql.server start
```
2. To start it from **System Preferences**, open the preferences and click the **MySQL** icon.

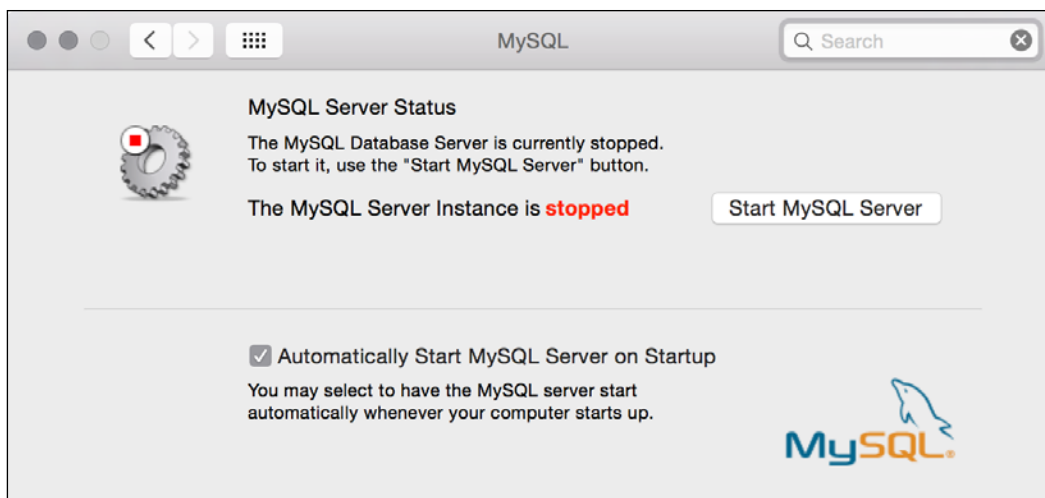


Figure 1.15 MySQL System Preferences - OSX

3. Click the **Start MySQL Server** button.
4. Next, download MySQL Workbench for OSX from <http://dev.mysql.com/downloads/workbench/>.

Installing MySQL on Linux

There are many different ways to install MySQL on Linux. Refer to <https://dev.mysql.com/doc/refman/5.7/en/linux-installation.html> for details.

Creating MySQL users

You can create MySQL user either from the Command Prompt or by using MySQL Workbench.

1. To execute SQL and other commands from the Command Prompt, open **Terminal** and type the following:

```
mysql -u root -p <root_password>
```
2. Once logged in successfully, you will see the mysql Command Prompt:

```
mysql>
```
3. To create a user, first select the mysql database.

```
mysql>use mysql;  
Database changed  
mysql>insert into user (host, user, password, select_priv, insert_priv, update_priv)  
values ('%', 'user1', password('usper1_pass'), 'Y', 'Y', 'Y');
```

The preceding command will create a user named 'user1' with password 'user1_pass' having privileges to insert, update, and select. And because we have specified host as '%', this user can access the server from any host.

If you prefer a graphical user interface to manage the users, then run MySQL Workbench, connect to the local MySQL server (see *Figure 1.13 MySQL Workbench Connections*), and click on **Users and Privileges** under the **Management** section.

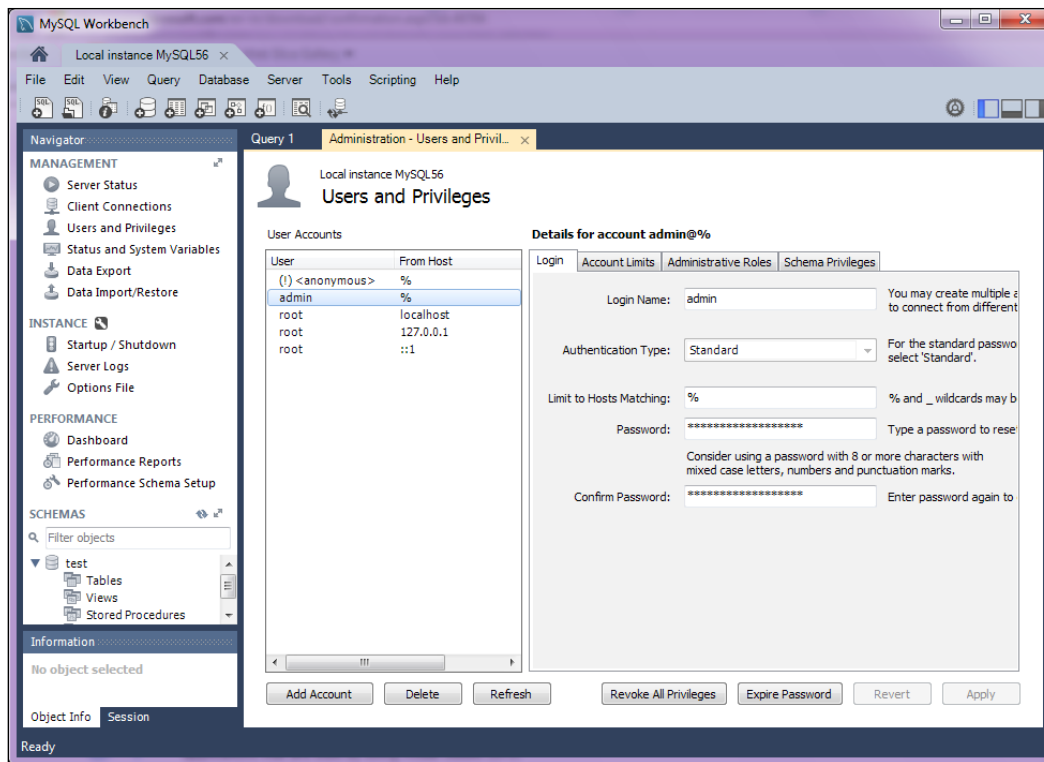


Figure 1.16 Creating a user in MySQL Workbench

Having installed all the above products, you should be in a position to start developing JEE applications. We may need a few additional software, but we will see how to install and configure them at appropriate time.

Summary

In this chapter, we had a brief introduction to different JEE APIs for the presentation layer, business layer, and Enterprise integration layer. We learnt some of the important terminologies in Eclipse IDE. We then learned how to install Eclipse, Tomcat, GlassFish, MySQL, and MySQL Workbench. We are going to use these products in this book to develop JEE applications.

In the next chapter, we will configure the JEE server and database in Eclipse, and create a simple application using Servlet, JSP, and Java Server Faces.

We will also see how to create JEE Web Applications using Servlet, JSP, and JSF. Along with that, we will learn how to use Maven to build and package the JEE applications.

2

Creating a Simple JEE Web Application

The previous chapter gave you a brief introduction to JEE and Eclipse. You also learned how to install the Eclipse JEE package and install and configure Tomcat. Tomcat is a servlet container and is easy to use and configure. Therefore, many developers use it to run JEE web applications on local machines.

In this chapter, we will see how to configure Tomcat in Eclipse and develop and deploy web applications. The advantage of configuring Tomcat in Eclipse is that you can easily start and stop the server from Eclipse, and deploy a JEE project right from within Eclipse to Tomcat.

Configuring Tomcat in Eclipse

Follow these steps for configuring the Tomcat server in Eclipse:

1. In the Java EE view of Eclipse, you will find the **Servers** tab at the bottom. Since no server is added yet, you will see a link in the **Servers** tab - **No servers are available. Click this link to create a new server...**

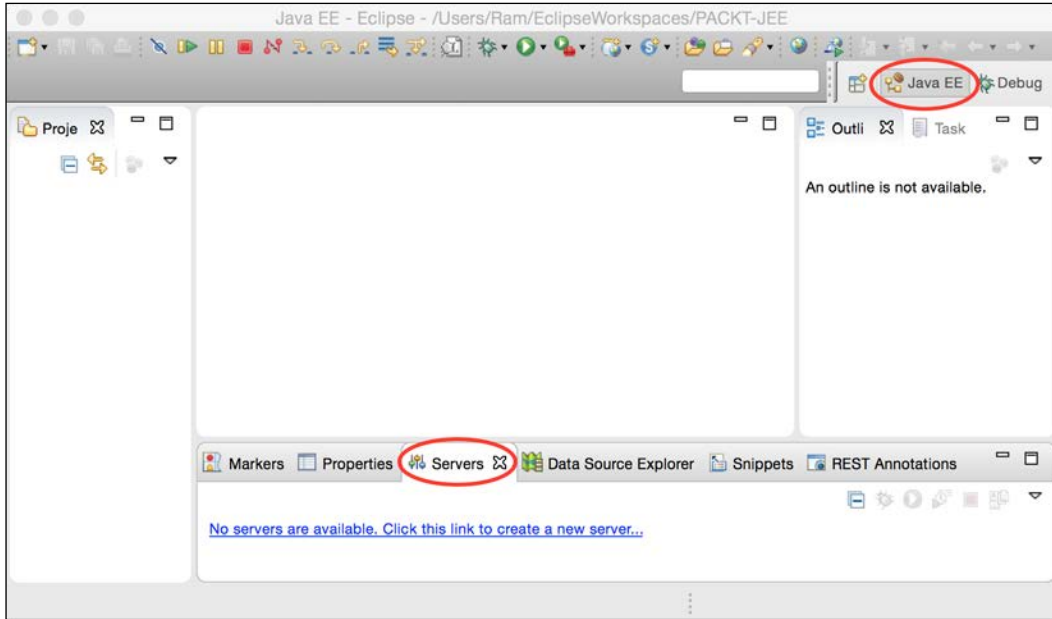


Figure 2.1 The Servers tab in Eclipse JEE