Java EE 8 Development with Eclipse

Third Edition

Develop, test, and troubleshoot Java Enterprise applications rapidly with Eclipse

Packt

www.packt.com

By Ram Kulkarni

Java EE 8 Development with Eclipse

Third Edition

Develop, test, and troubleshoot Java Enterprise applications rapidly with Eclipse

Ram Kulkarni

Packt

BIRMINGHAM - MUMBAI

Java EE 8 Development with Eclipse *Third Edition*

Commissioning Editor: Richa Tripathi
Acquisition Editor: Sandeep Mishra
Content Development Editor: Anugraha Arunagiri
Technical Editor: Madhunikita Sunil Chindarkar
Copy Editor: Safis Editing
Project Coordinator: Ulhas Kambali
Proofreader: Safis Editing
Indexer: Mariammal Chettiyar
Graphics: Tania Dutta
Production Coordinator: Nilesh Mohite

First published: December 2012
Second edition: September 2015
Third edition: June 2018

Production reference: 1260618

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

ISBN 978-1-78883-377-6

`www.packtpub.com`

To my wife Vandana and son Akash for their love and support

Contributors

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, and IDE and mobile applications. He is also the author of *Eclipse 4 RCP Development How-To*, published by Packt Publishing. You can find him at *RamKulkarni.com*.

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 continued love and support.

About the reviewer

Borja Pérez Dopazo is a software architect with around 10 years of experience. He strongly believes in reuse and standardization. He is involved in architecture, design, implementation of microservices architectures, service-oriented architectures, and development of his own frameworks.

Table of Contents

Preface

Java Enterprise Edition (JEE) has been used for developing enterprise applications for many years. It provides a standard way of implementing many aspects of enterprise applications, such as handling web requests, accessing databases, connecting to other enterprise systems, and implementing web services. Over the years, it has evolved and made enterprise application development easier than before. It has also changed its name from J2EE to JEE after J2EE version 1.4, and, more recently, from Java Enterprise Edition to Jakarta Enterprise Edition in February 2018. Currently, JEE is in version 8.

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

This book provides you with all the information you need to use Eclipse in order to develop, deploy, debug, and test JEE applications. The focus of the 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 also covers other technologies used in different phases of application development, such as source control, unit testing, and profiling. This book also covers deployment to cloud platforms, microservices, and JEE security features.

JEE is a collection of many technologies and specifications. Some of the technologies are so vast that separate books could be written on them, and some already have been. This book takes the approach of providing a brief introduction to each technology in JEE and provides links for detailed information. It then jumps to developing sample applications using specific technologies under discussion, and explains finer aspects of the technologies in the context of the sample applications.

Who this book is for

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 and are looking for a hands-on approach to using some of the technologies in JEE.

What this book covers

`Chapter 1`, *Introducing JEE and Eclipse*, explains in brief different technologies in JEE and where they fit in a typical multitiered JEE application. The chapter also describes installing Eclipse JEE, Tomcat, GlassFish, and MySQL, which are used to develop sample applications in 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*, shows how to use SVN and Git plugins of Eclipse for source code management.

`Chapter 4`, *Creating JEE Database Applications*, 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, set relationships between classes using JPA, and use database connection pools.

`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 the JEE Application*, shows techniques for debugging JEE applications and debugging support in Eclipse.

`Chapter 7`, *Creating JEE Applications with EJB*, describes using EJBs to code business logic in the JEE applications. It also explains connecting to remote EJBs using JNDI and injecting EJBs in 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 could 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 also learn how to consume these web services from JEE applications.

`Chapter 10`, *Asynchronous Programming with JMS*, shows you 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 techniques for profiling CPU and memory in the Java applications to find performance bottlenecks.

`Chapter 12`, *Microservices*, describes how to develop and deploy microservices. It also covers the deployment of microservices in Docker container.

`Chapter 13`, *Deploying JEE Applications in the Cloud*, describes how to deploy JEE applications in Amazon and Google Cloud platforms. Specifically, it describes the deployment of applications in AWS EC2, Beanstalk, Google Compute Engine, and Google App Engine. It also describes Eclipse tools that can be used for deployment to the Cloud.

`Chapter 14`, *Securing JEE Applications*, describes how to secure JEE applications using authentication and authorization features of JEE containers. It also covers some of the JEE 8 security enhancements.

To get the most out of this book

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

Download the example code files

You can download the example code files for this book from your account at `www.packtpub.com`. If you purchased this book elsewhere, you can visit `www.packtpub.com/support` and register to have the files emailed directly to you.

You can download the code files by following these steps:

1. Log in or register at `www.packtpub.com`.
2. Select the **SUPPORT** tab.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the book in the **Search** box and follow the onscreen instructions.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR/7-Zip for Windows
- Zipeg/iZip/UnRarX for Mac
- 7-Zip/PeaZip for Linux

The code bundle for the book is also hosted on GitHub at `https://github.com/PacktPublishing/Java-EE-8-Development-with-Eclipse-Third-Edition`. In case there's an update to the code, it will be updated on the existing GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: `https://www.packtpub.com/sites/default/files/downloads/JavaEE8DevelopmentwithEclipseThirdEdition_ColorImages.pdf`.

Conventions used

There are a number of text conventions used throughout this book.

`CodeInText`: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "To stop the GlassFish Server, run the `stopserv` script in the `glassfish/bin` folder."

A block of code is set as follows:

```
public class LoginBean {
  private String userName;
  private String password;
}
```

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

```
mysql>use mysql;
Database changed
mysql>create user 'user1'@'%' identified by 'user1_pass';
mysql>grant all privileges on *.* to 'user1'@'%' with grant option
```

Bold: Indicates a new term, an important word, or words that you see on screen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "Select **Dynamic Web Project** and click **Next** to open the **Dynamic Web Project** wizard."

Warnings or important notes appear like this.

Tips and tricks appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: Email `feedback@packtpub.com` and mention the book title in the subject of your message. If you have questions about any aspect of this book, please email us at `questions@packtpub.com`.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit `www.packtpub.com/submit-errata`, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the Internet, we would be grateful if you would provide us with the location address or website name. Please contact us at `copyright@packtpub.com` with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit `authors.packtpub.com`.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit `packtpub.com`.

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 with the release of version 5 in 2006. Since version 5, it has been renamed **Java Enterprise Edition** (**JEE**). Recent versions of JEE have 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 into 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 good development tools. The **Integrated Development Environment** (**IDE**) plays a major part in developing applications faster, and Eclipse provides just that for JEE. Not only do you get good code 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.

In this chapter, we are going to cover the following topics:

- Introduction to different technologies in JEE
- Introduction to the Eclipse development environment
- Installation and configuration of some of the frequently used software in this book, for example, JEE servers, Eclipse IDE, and MySQL Database Server

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

In 2017, Oracle agreed to hand over control of Java EE to Eclipse Foundation. In April 2018, Eclipse Foundation renamed Java EE as Jakarta EE. You can find more information about Jakarta EE at `https://jakarta.ee/`. At the time of writing, the latest Java EE version is 8. But all future versions of Java EE will be called Jakarta EE.

JEE

JEE is a collection of many of the Java Community Process (`https://www.jcp.org`) programs. Currently, JEE is in Version 8. However, different specifications of JEE are at their own different versions.

JEE specifications can be broadly classified into the following groups:

- Presentation layer
- Business layer
- Enterprise integration layer

Note that JEE specification does not necessarily classify APIs in the preceding broad groups, but such classification could help in better understanding the purpose of the different specifications and APIs in JEE.

Before we see APIs in each of these categories, let's understand a typical JEE web application flow, as shown in the following diagram, and where each of the preceding layers fits in:

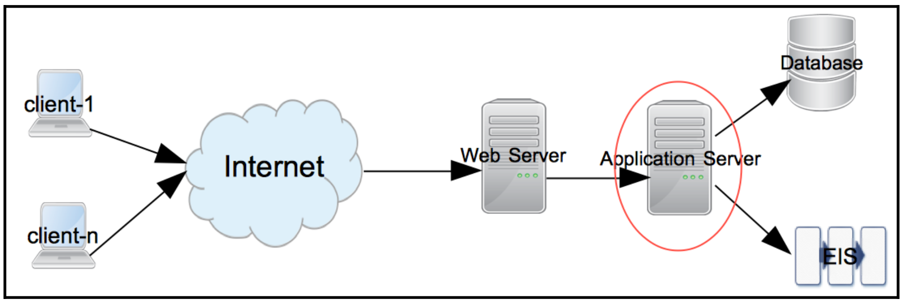


Figure 1.1: A typical JEE web application flow

Requests start from the clients. A client can be any application requesting services from a remote application—for example, it could be the browser or a desktop application. The request is first received by the web server at the destination. Examples of web servers include Apache web server, IIS, and nginx. If it is a request for static content, then it is served by the web server(s). However, a dynamic request typically requires an application server to process. JEE servers are such application servers that handle dynamic requests. Most JEE specification APIs execute in the application server. Examples of JEE application servers are WebSphere, GlassFish, and WildFly.

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

The following sections provide a brief description of each of the JEE specifications in different layers. We will see how to use these specifications and their 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 layer receive requests from the web server and send back the response, typically in HTML format. However, it is also possible to return only data from the presentation layer, for example in **JavaScript Object Notation** (**JSON**) or **eXtensible Markup Language** (**XML**) format, which could be consumed by **Asynchronous JavaScript and XML** (**AJAX**) calls to update only part of the page, instead of rendering the entire HTML page. Classes in the presentation layer are mostly executed in the web container—it is a part of the application server that handles web requests. Tomcat is an example of a popular web container.

Now let's take a look at some of the specifications in this layer.

Java Servlets

Java Servlets are server-side modules, typically used to process requests and send back responses in web applications. Servlets are useful for handling requests that do not generate large HTML markup responses. They are typically used as controllers in **Model View Controller** (**MVC**) frameworks, for forwarding/redirecting requests, or for generating non-HTML responses, such as PDFs. To generate HTML response from the servlet, you need to embed HTML code (as a Java String) in Java code. Therefore, it is not the most convenient option for generating large HTML responses. JEE 8 contains servlet API 4.0.

JavaServer Pages

Like servlets, **JavaServer Pages** (**JSPs**) are also server-side modules used for processing web requests. JSPs 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 servlets) to generate an HTML response from the JSP pages. JSP specification 2.3 is included in JEE 8.

JavaServer Faces

JavaServer Faces (**JSFs**) make creating a 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 client and server. For example, if you have a page that posts form data from a browser, you can have a JSF save that data in a Java bean so that it can be used subsequently in the response to the same or different request. JSFs also make it easier to handle UI events on the server side and specify page navigation in an application.

You write the JSF code in JSP, using custom JSP tags created for JSF. JavaServer Faces API 2.3 is part of JEE 8.

The business layer

The business layer is where you typically write code to handle the business logic of your application. Requests 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 JavaBeans

Enterprise JavaBeans (**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 the EJB container) take care of component lookup (searching component) and component pooling (useful for scalability). This can improve the 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. An application on the server side gets the request message, either directly using JMS APIs or by using MDBs. It processes the request and may put the response in a different queue or topic, to which the client would listen and get the response.

Java EE 8 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 the JEE application) systems in the enterprise. Most applications would need to access a database, and APIs to access that fall in this group.

Java Database Connectivity

Java Database Connectivity (**JDBC**) is a specification to access a relational database in a common and consistent way. Using JDBC, you can execute SQL statements and get results on different databases using common APIs. A database-specific driver sits between the JDBC call and the database, and it 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 the UI and the business code. Typically, this is done by creating **Data Access Objects** (**DAOs**) that encapsulate the logic to access the database. JDBC is actually a part of Java Standard Edition. Java SE 8 contains JDBC 4.2.

The Java Persistence API

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 or rows in the relational database. Frameworks such as Hibernate and Spring have made this process simpler by using a concept known as **Object Relational Mapping** (**ORM**). ORM is incorporated in JEE in the form of the **Java Persistence API** (**JPA**).

JPA gives you the flexibility to map objects to tables in the relational database and execute queries with or without using **Structured Query Language** (**SQL**). When used in the content of JPA, the query language is called **Java Persistence Query Language**. JPA specification 2.2 is a part of JEE8.

Java Connector Architecture

Java Connector Architecture (**JCA**) APIs can be used in JEE applications for communicating with **enterprise integration systems** (**EISes**), such as SAP, and Salesforce. Just like you have database drivers to broker communication between JDBC APIs and relational databases, you have JCA adapters between JCA calls and EISes. 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 the JEE application server.

Web services

Web services are remote application components and expose self-contained APIs. Web services can be broadly classified based on 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).

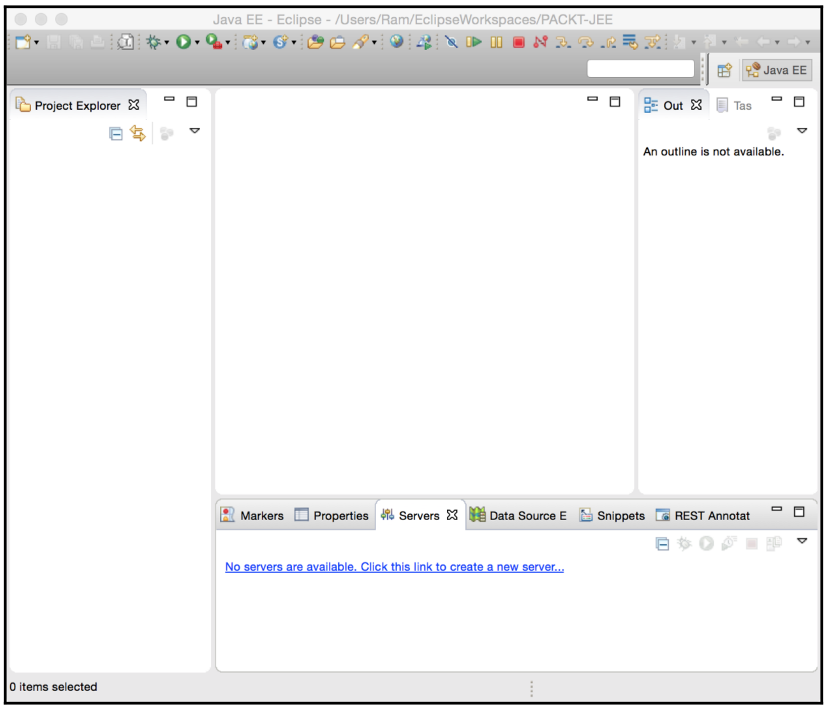
The preceding are just some of the specifications that are part of JEE. There are many other independent specifications and many enabling specifications, such as dependency injection and concurrency utilities, which we will see in subsequent chapters.

Eclipse IDE

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 the 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 component 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. Examples include source control systems such as SVN and Git, build tools such as Apache Ant and Maven, file explorers for remote systems using FTP, managing servers such as Tomcat and GlassFish, database explorers, memory and CPU profilers. We will see many of these features in the subsequent chapters. The following screenshot shows the default view of Eclipse for JEE application development:



An outline is not available.

No servers are available. Click this link to create a new server...

0 items selected

Figure 1.2: Default Eclipse view

When working with Eclipse, it is good to understand the following terms.

Workspace

The Eclipse workspace is a collection of projects, settings, and preferences. It is a folder where Eclipse stores this information. You must create a workspace to start using Eclipse. You can create multiple workspaces, but only one can be opened at a time 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 more. The 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 an editor or a view. An editor is something where you can change the information displayed in it. A 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 code. An example of a view is the outline view that displays the hierarchical structure of the code you are editing (in the case of a Java editor, it shows classes and methods in the file being edited).

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

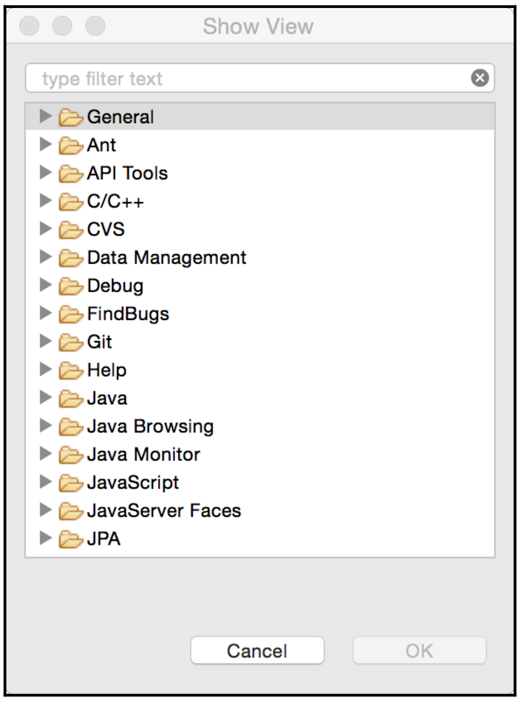


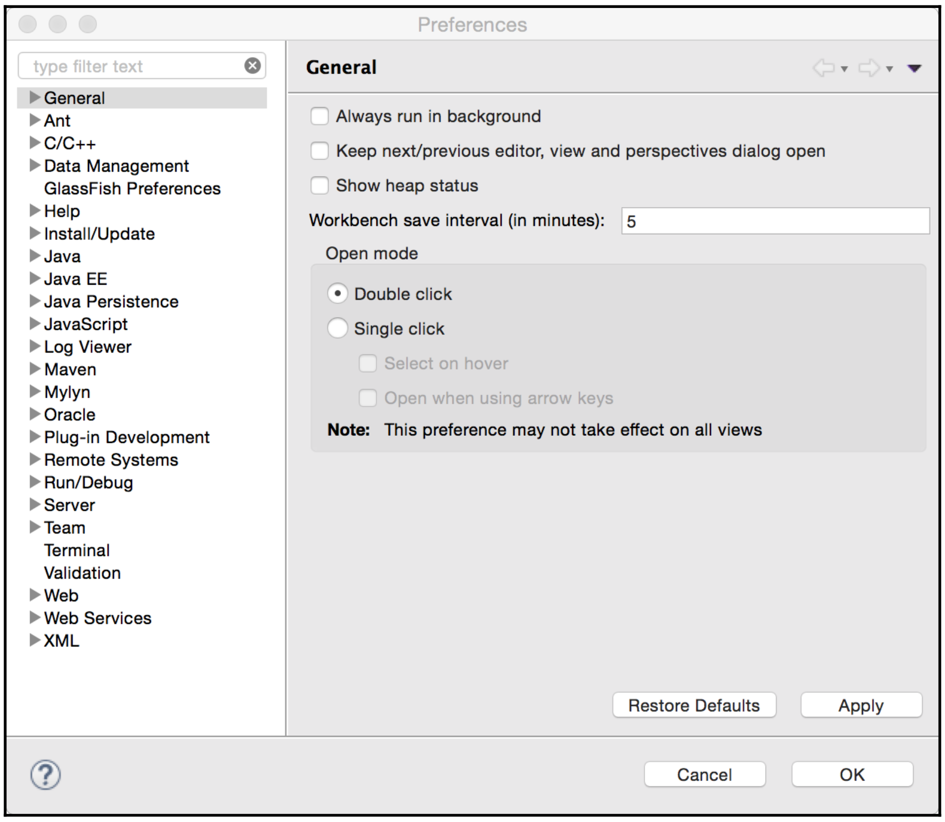
Figure 1.3: Show all Eclipse views

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 the code, you need to see **Project Explorer** and **Task** views, but when you are debugging an application, you don't need those views, but instead want to see variable and breakpoint views. So, the editing perspective displays, among other views and editors, **Project Explorer** and **Task** views, and the **Debug** perspective displays views and editors relevant to the debugging activities. You can change the default perspectives to suit your purposes.

Eclipse preferences

The Eclipse **Preferences** window (*Figure 1.4*) is where you customize many plugins/features. Preferences are available from the **Window** menu in the Windows and Linux installations of Eclipse, and from the **Eclipse** menu in Mac:



General

- [ ] Always run in background
- [ ] Keep next/previous editor, view and perspectives dialog open
- [ ] Show heap status

Workbench save interval (in minutes): 5

Open mode

- [✓ Double click]
- Single click
  - [ ] Select on hover
  - [ ] Open when using arrow keys

Note: This preference may not take effect on all views

Figure 1.4: Eclipse preferences

Installing products

In the subsequent chapters, we will learn how to develop JEE applications in Eclipse. But the applications are going to need a JEE application server and a database. We are going to use the Tomcat web container in the initial few chapters and then use the GlassFish JEE application server. We are going to use a MySQL database.

We are going to need these products for many of the applications that we are going to develop. So the following sections describe how to install and configure Eclipse, Tomcat, GlassFish, and MySQL.

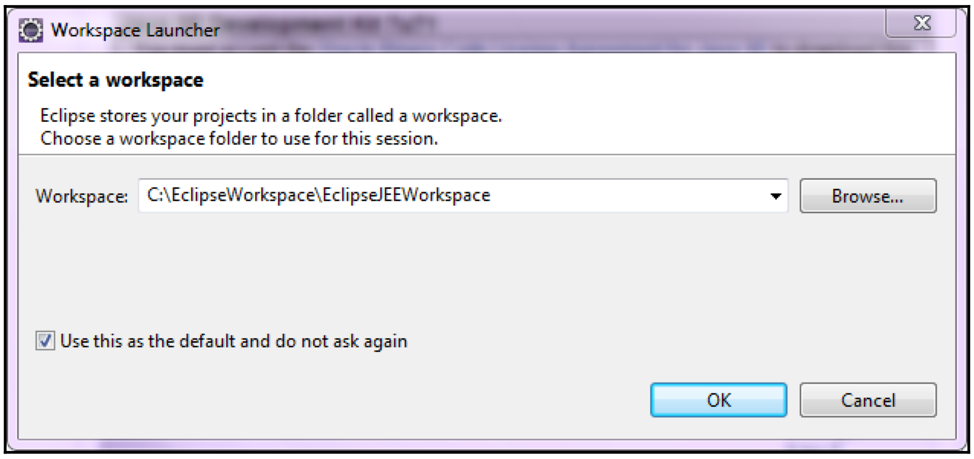
Installing Eclipse

Download the latest version of 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 whether the JVM is 32-bit or 64-bit.

TIP

If you plan to use Eclipse for AWS development, then it is recommended to download Eclipse from the Oomph installer. Refer to `https://wiki.eclipse.org/Eclipse_Installer` and `https://docs.aws.amazon.com/toolkit-for-eclipse/v1/user-guide/setup-install.html`.

Unzip the downloaded ZIP file and then run the Eclipse application (you must 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 filesystem and select that as the initial workspace folder. If you intend to use the same folder for the workspace on every launch of Eclipse, then check the **Use this as the default and do not ask again** checkbox:



Workspace Launcher

Select a workspace

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\EclipseWorkspace\EclipseJEEWorkspace

☑ Use this as the default and do not ask again

Figure 1.5: Select Eclipse workspace

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

Installing the Tomcat server

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

Download the latest version of Tomcat from `http://tomcat.apache.org/`. Unzip the downloaded file 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 subfolders). To start the server, run `startup.bat` in Command Prompt on Windows and `startup.sh` in a Terminal window on Mac and Linux. If there are no errors, then you should see the message `Server startup in --ms` or `Tomcat started`.

The 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 a connector declaration such as the following:

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

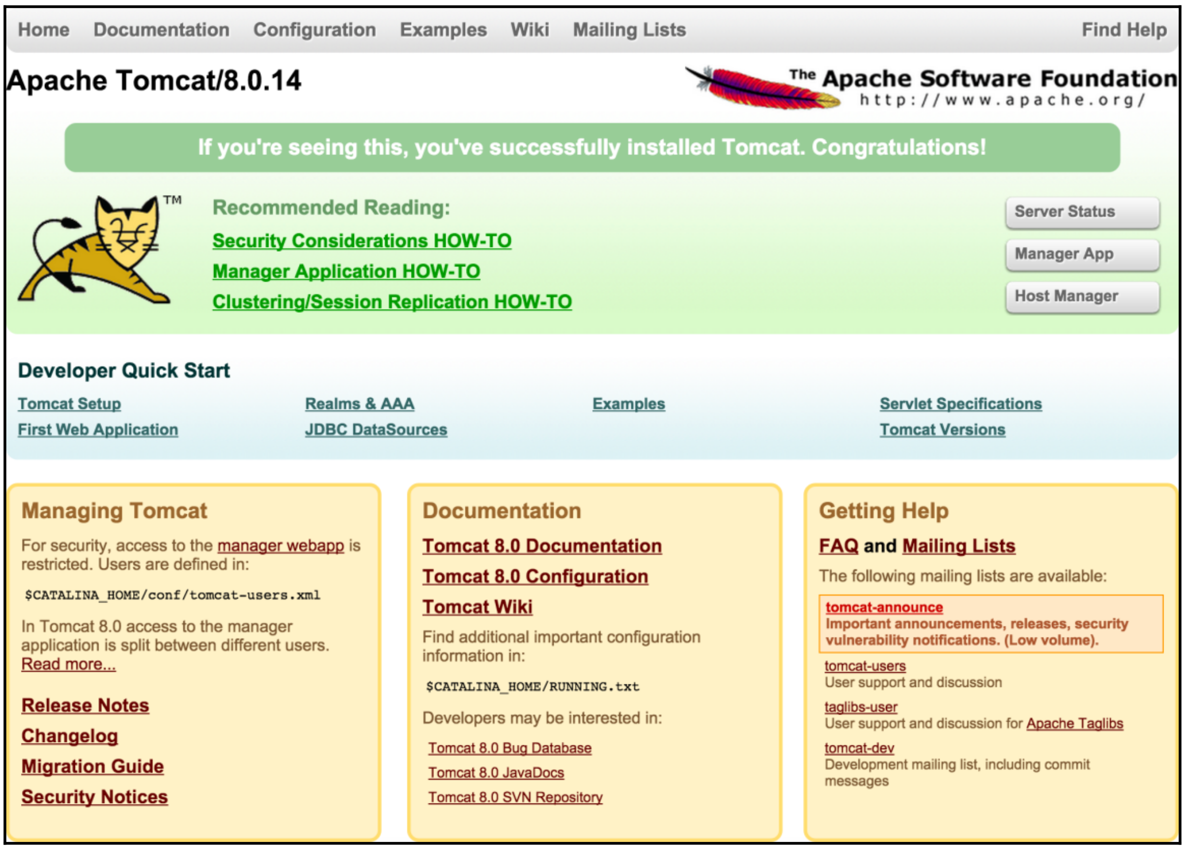
Change the port value to any port number you want, though in this book we will be using the default port `8080`. Before we open the default page of Tomcat, we will add a user for administration of the Tomcat server. Open `tomcat-users.xml` under the `conf` folder using any text editor. At the end of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


Home Documentation Configuration Examples Wiki Mailing Lists Find Help

Apache Tomcat/8.0.14

The Apache Software Foundation
http://www.apache.org/

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
Security Considerations HOW-TO
Manager Application HOW-TO
Clustering/Session Replication HOW-TO

Server Status
Manager App
Host Manager

Developer Quick Start

Tomcat Setup	Realms & AAA	Examples	Servlet Specifications
First Web Application	JDBC DataSources		Tomcat Versions

Managing Tomcat

For security, access to the manager webapp is restricted. Users are defined in:

`$CATALINA_HOME/conf/tomcat-users.xml`

In Tomcat 8.0 access to the manager application is split between different users. Read more...

Release Notes
Changelog
Migration Guide
Security Notices

Documentation

Tomcat 8.0 Documentation
Tomcat 8.0 Configuration
Tomcat Wiki

Find additional important configuration information in:

`$CATALINA_HOME/RUNNING.txt`

Developers may be interested in:

Tomcat 8.0 Bug Database
Tomcat 8.0 JavaDocs
Tomcat 8.0 SVN Repository

Getting Help

FAQ and Mailing Lists

The following mailing lists are available:

tomcat-announce
Important announcements, releases, security vulnerability notifications. (Low volume).

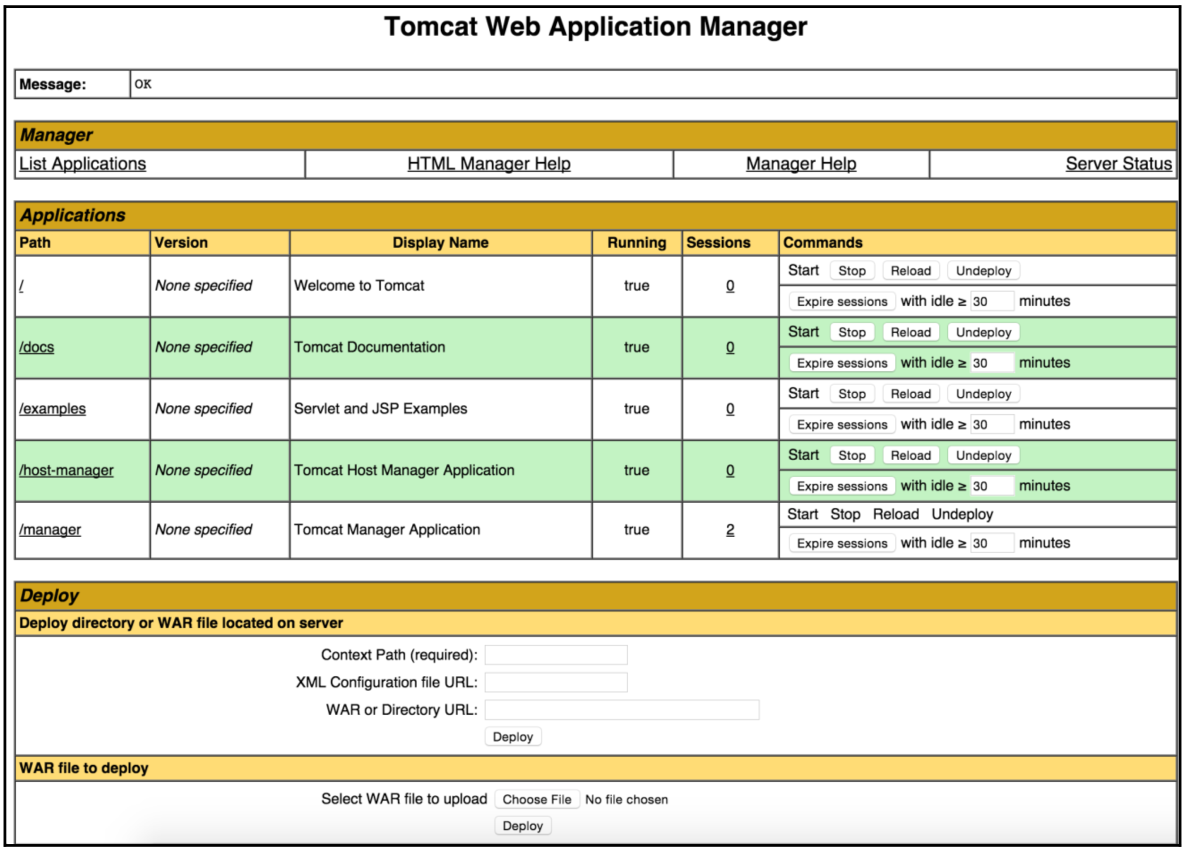
tomcat-users
User support and discussion

taglibs-user
User support and discussion for Apache Taglibs

tomcat-dev
Development mailing list, including commit messages

Figure 1.6: The default Tomcat web application

Click on the **Manager App** button on the right. You will be asked for the username and password. Enter the username 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 *Figure 1.7*. You can see all 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):
XML Configuration file URL:
WAR or Directory URL:
Deploy

WAR file to deploy

Select WAR file to upload Choose File No file chosen
Deploy

Figure 1.7: Tomcat Web Application Manager

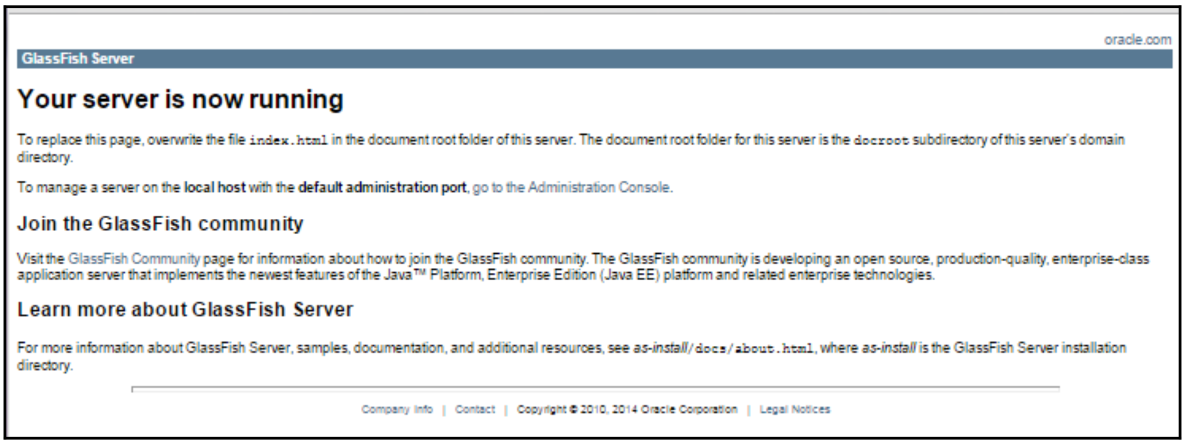
To stop the Tomcat server, press *Ctrl/cmd* + *C* or run the shutdown script in the `bin` folder.

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 the Full Platform.

Unzip the downloaded file in a folder. The default port of the GlassFish server is `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:


oracle.com

GlassFish Server

Your server is now running

To replace this page, overwrite the file `index.html` in the document root folder of this server. The document root folder for this server is the `docroot` subdirectory of this server's domain directory.

To manage a server on the **local host** with the **default administration port**, go to the Administration Console.

Join the GlassFish community

Visit the GlassFish Community page for information about how to join the GlassFish community. The GlassFish community is developing an open source, production-quality, enterprise-class application server that implements the newest features of the Java™ Platform, Enterprise Edition (Java EE) platform and related enterprise technologies.

Learn more about GlassFish Server

For more information about GlassFish Server, samples, documentation, and additional resources, see *as-install*/`docs/about.html`, where *as-install* is the GlassFish Server installation directory.

Company Info | Contact | Copyright © 2010, 2014 Oracle Corporation | Legal Notices

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 page to open the GlassFish administrator (see the following screenshot):

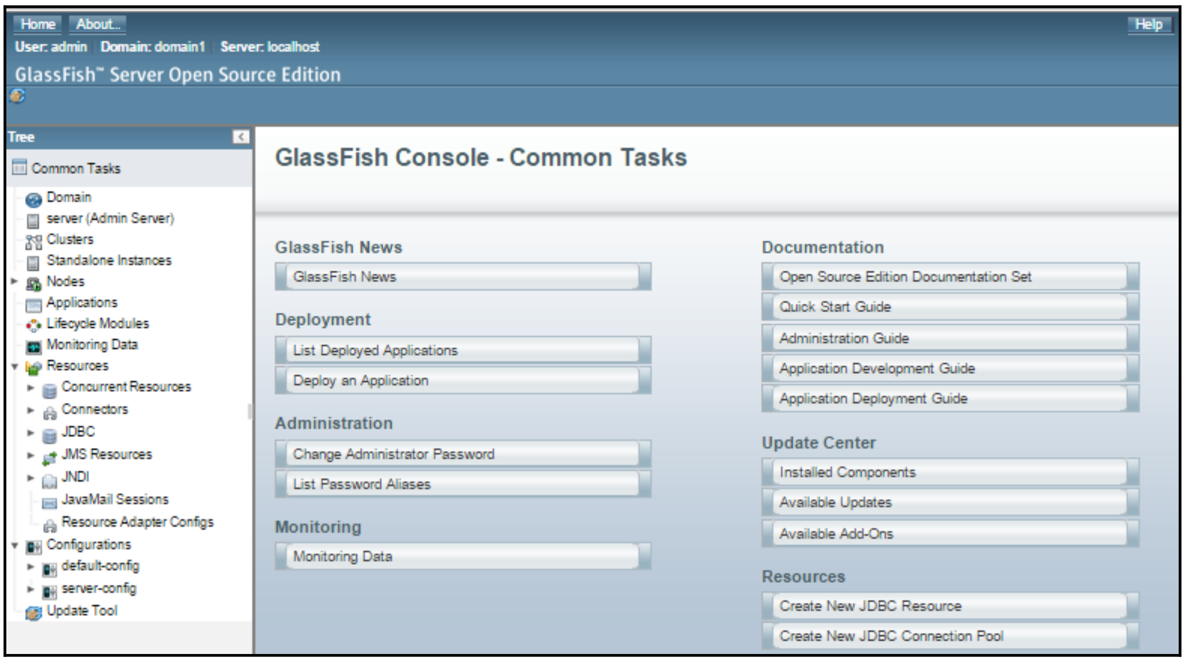


Figure 1.9: The Glassfish administrator

TIP

For details on administrating the GlassFish server, refer to `https://javaee.github.io/glassfish/doc/5.0/administration-guide.pdf`.

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

Installing MySQL

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

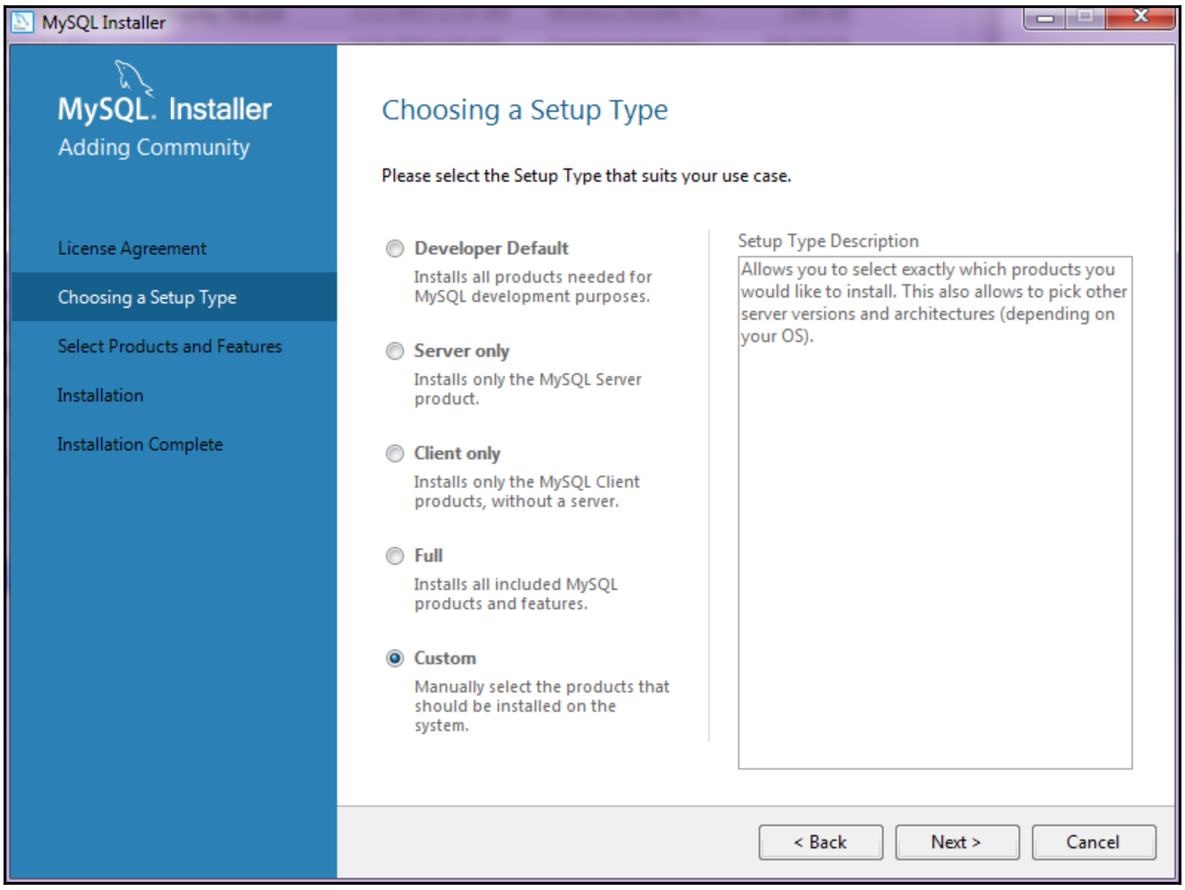
We would like to install MySQL Workbench too, which is a client application to manage MySQL Server. Download MySQL Workbench from `https://dev.mysql.com/downloads/workbench/`.

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. The following instructions show the download options using the web installer.

The web installer first downloads a small application, and it gives you options to select the components that you want to install:

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



MySQL Installer

MySQL. Installer
Adding Community

- License Agreement
- Choosing a Setup Type
- Select Products and Features
- Installation
- Installation Complete

Choosing a Setup Type

Please select the Setup Type that suits your use case.

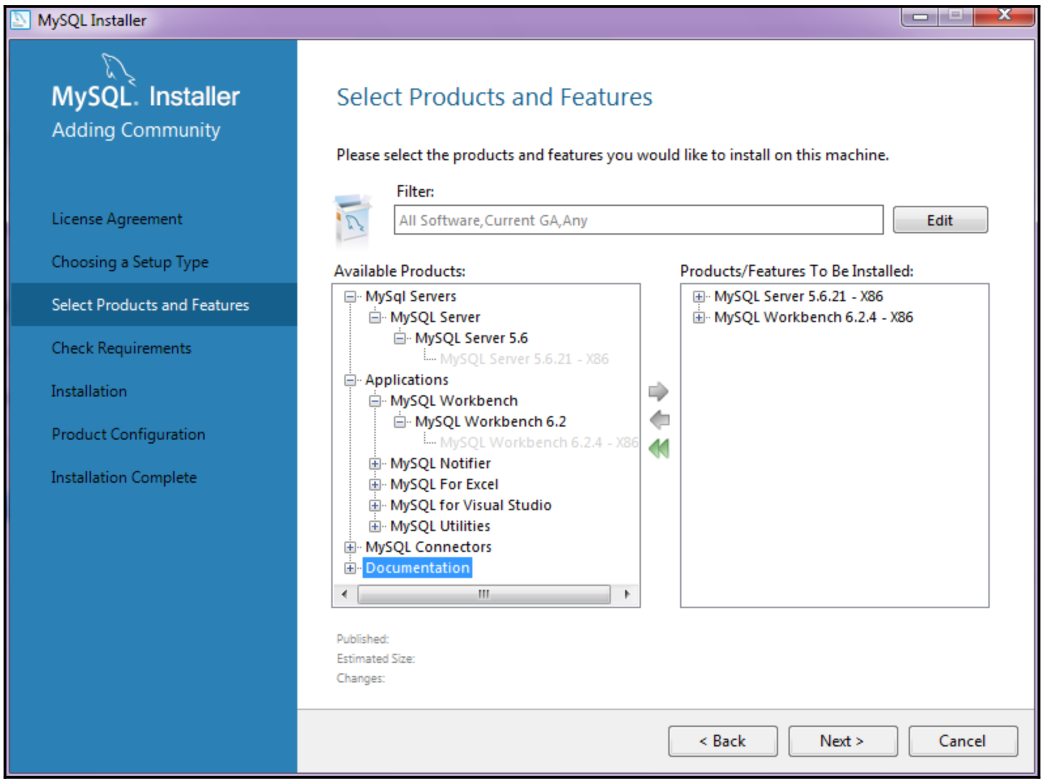
- Developer Default
  Installs all products needed for MySQL development purposes.
- Server only
  Installs only the MySQL Server product.
- Client only
  Installs only the MySQL Client products, without a server.
- Full
  Installs all included MySQL products and features.
- [✓ Custom]
  Manually select the products that should be installed on the system.

Setup Type Description

Allows you to select exactly which products you would like to install. This also allows to pick other server versions and architectures (depending on your OS).

Figure 1.10: MySQL Installer for Windows

2. Select the **MySQL Server** and **MySQL Workbench** products and complete the installation. During the installation of the 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 a user other than root for applications to use:



Select Products and Features

Please select the products and features you would like to install on this machine.

Filter: All Software,Current GA,Any

Available Products:
- MySql Servers
  - MySQL Server
    - MySQL Server 5.6
      - MySQL Server 5.6.21 - X86
- Applications
  - MySQL Workbench
    - MySQL Workbench 6.2
      - MySQL Workbench 6.2.4 - X86
  - MySQL Notifier
  - MySQL For Excel
  - MySQL for Visual Studio
  - MySQL Utilities
- MySQL Connectors
- Documentation

Products/Features To Be Installed:
- MySQL Server 5.6.21 - X86
- MySQL Workbench 6.2.4 - X86

Published:
Estimated Size:
Changes:

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:

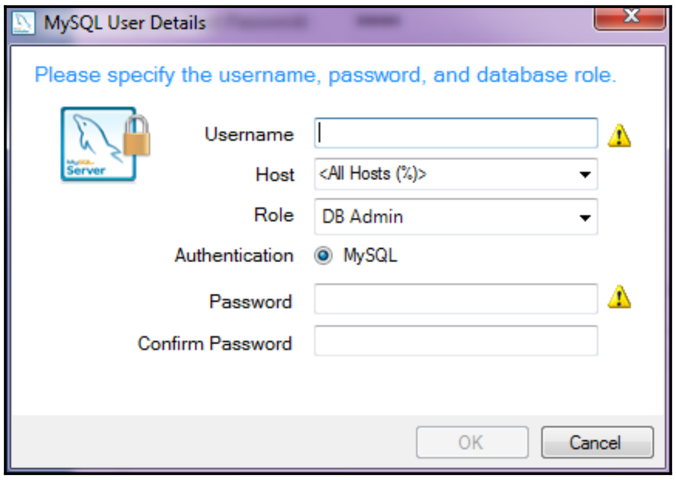


Figure 1.12: Add MySQL user

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

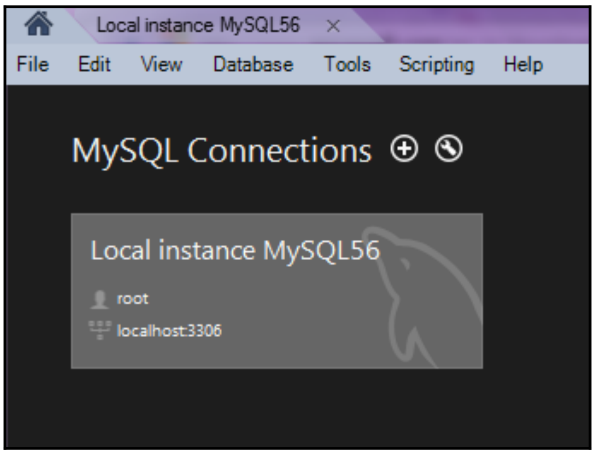
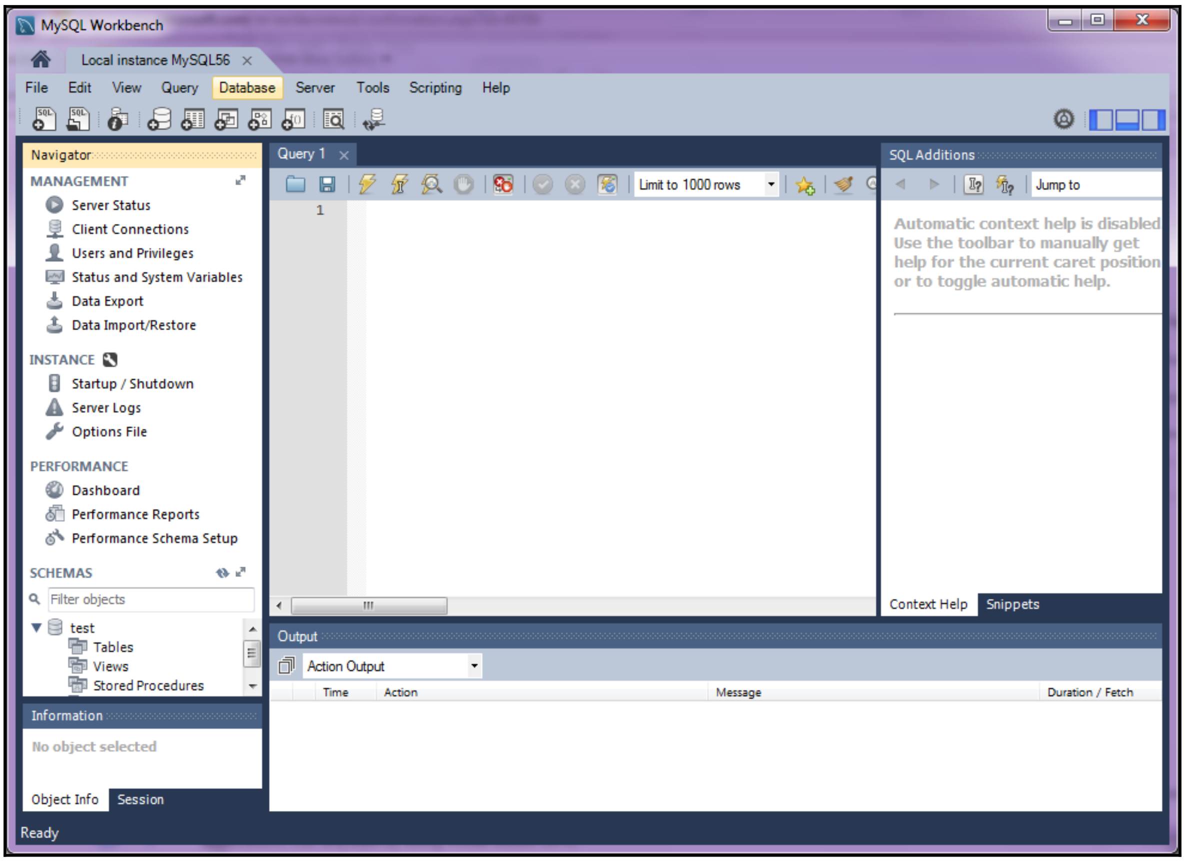


Figure 1.13: MySQL Workbench connections

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:



Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Figure 1.14: My SQL Workbench

Installing MySQL on macOS X

OS X versions before 10.7 had MySQL Server installed by default. If you are using OS X 10.7 or later, then you will need to download and install MySQL Community Server from `http://dev.mysql.com/downloads/mysql/`.

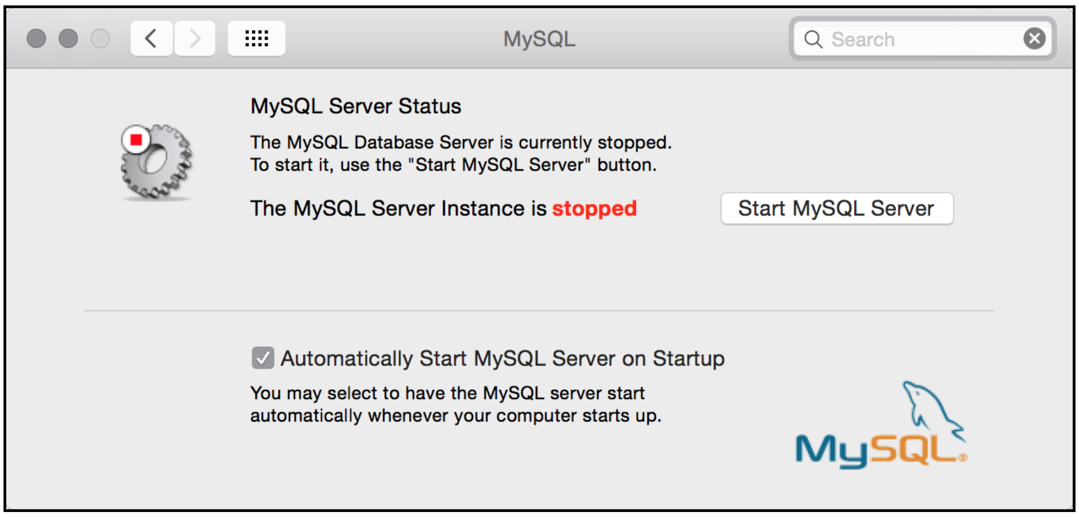
There are many different ways to install MySQL on OS X. See `http://dev.mysql.com/doc/refman/5.7/en/osx-installation.html` for installation instructions for OS X. Note that users on OS X should have administrator privileges to install MySQL Server.

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

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



MySQL Server Status

The MySQL Database Server is currently stopped.
To start it, use the "Start MySQL Server" button.

The MySQL Server Instance is stopped

☑ Automatically Start MySQL Server on Startup

You may select to have the MySQL server start automatically whenever your computer starts up.

Figure 1.15: MySQL System Preferences - OS X

3. Click the **Start MySQL Server** button.

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 users either from Command Prompt or by using MySQL Workbench:

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

```
mysql -u root -p<root_password>
```

2. Once logged in successfully, you will see the `mysql` Command Prompt:

```
mysql>
```

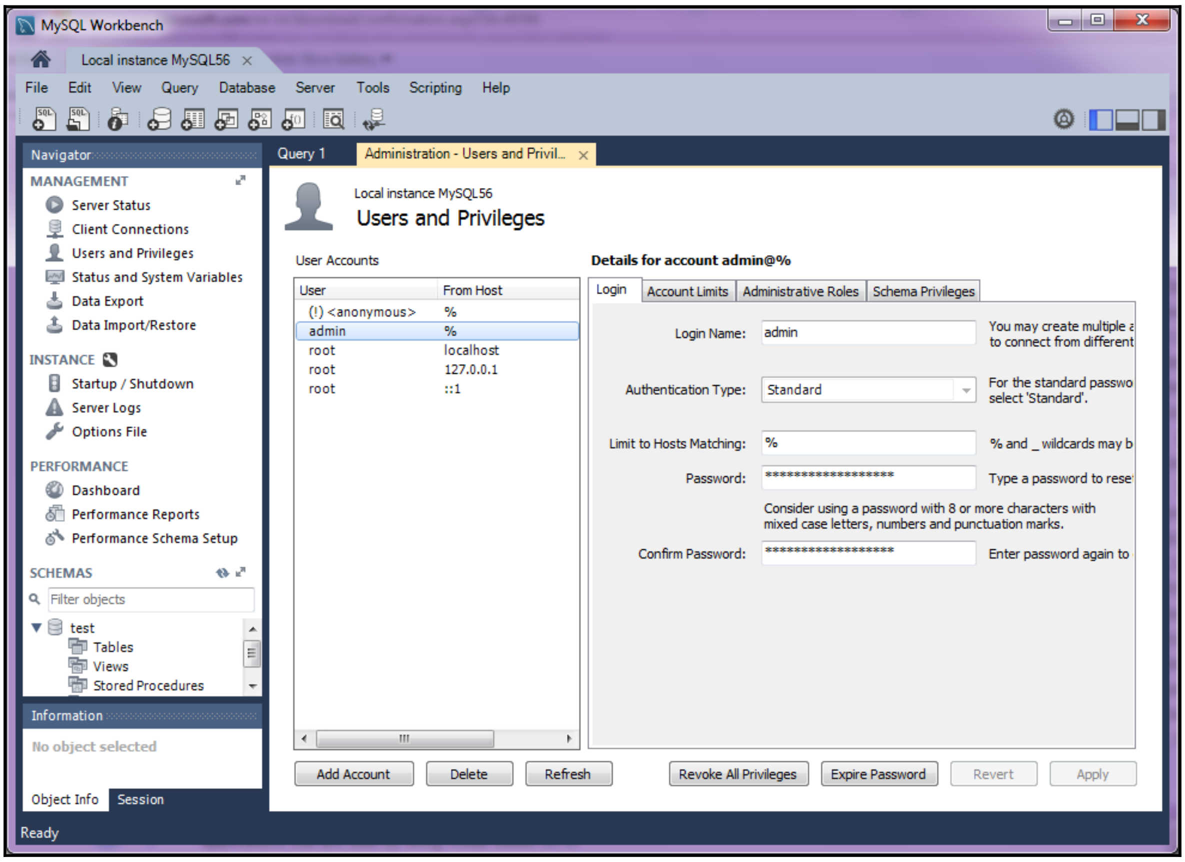
3. To create a user, first select the `mysql` database:

```
mysql>use mysql;
Database changed
mysql>create user 'user1'@'%' identified by 'user1_pass';
mysql>grant all privileges on *.* to 'user1'@'%' with grant option
```

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

See `https://dev.mysql.com/doc/refman/5.7/en/adding-users.html` for more details on adding users to MySQL database

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



Local instance MySQL56
Users and Privileges

User Accounts

User	From Host
(!) <anonymous>	%
admin	%
root	localhost
root	127.0.0.1
root	::1

Details for account admin@%

Login Name:	admin	You may create multiple a to connect from different
Authentication Type:	Standard	For the standard passwo select 'Standard'.
Limit to Hosts Matching:	%	% and _ wildcards may b
Password:	******************	Type a password to rese
	Consider using a password with 8 or more characters with mixed case letters, numbers and punctuation marks.	
Confirm Password:	******************	Enter password again to

Figure 1.16: Creating a user in MySQL Workbench

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

Summary

In this chapter, we had a brief introduction to different JEE specifications for the presentation layer, business layer, and enterprise integration layer. We learned 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 create a simple application using servlets, JSPs, and JSFs. We will also learn how to use Maven to build and package the JEE applications.