Chapter 1

Setting Up the Java Development Environment

IN THIS CHAPTER

- The Java programming language
- The Java SE Development Kit
- The JBoss application server
- Apache Ant
- The Eclipse integrated development environment

Before you start working with Java, you need to set up a Java development environment. This includes installing the Java Standard Edition (SE) Development Kit and the JBoss application server and then configuring the server to use the Java Development Kit (JDK) you just installed. After that, you must install Apache Ant and the Eclipse integrated development environment (IDE), which you also configure to use the same JDK and to control your JBoss server.

Because Java is multiplatform, and JBoss, Ant, and Eclipse are themselves written in Java, it's possible to use this Java development environment setup wherever Java runs. This chapter, though, covers only setup in Windows.

The Java Programming Language

The Java programming language is one of the most popular choices for Web application development for a number of reasons. First, Web applications written in Java are portable. That means that the same Java application you write for a Windows machine can also be run on Mac OS, Linux, or Solaris without the need for changes to your code.

In addition to portability, Java is well-supported by a number of development tools that make developing and deploying Java Web applications much easier. The Eclipse IDE, when configured to work together with other development tools, can help you write code, compile it, package and deploy the application to a server, and even run and debug the application — all from within the IDE. In fact, Eclipse, Ant, and JBoss are all written in Java themselves — a testament to the versatility of the language.

Object-oriented programming

Java is said to be an *object-oriented* programming language. In the world of object-oriented programming, *programs* are simply collections of interacting objects. *Objects* are programmatic representations of things in the real world. They are collections of *properties* (things the object has) and *behaviors* (things the object does).

The basic building block in object-oriented programming is called a *class*. A class describes the properties and behaviors of an object. For example, a `Car` class might contain properties such as `color`, `bodyStyle`, `currentSpeed`, and `mileage`. It might also have behaviors such as `accelerate`, `stop`, and `start`.

An object is a specific *instance* of a class. Whereas a class merely describes the properties and behaviors of an object, an instance contains specific values and implementations of those properties and behaviors. For example, an instance of the `Car` class might have a `color` of `"blue"`, a `bodyStyle` of `"sedan"`, and a `currentSpeed` of `"55"`. The `accelerate` behavior might increase the value of `currentSpeed` by one each time it's invoked. Each instance of a class shares the properties and behaviors of the class with all other instances of that class, but the values and implementations of those properties and behaviors can differ.

Classes may also *inherit* properties and behaviors from other classes. A `Convertible` class might inherit all the properties and behaviors of the `Car` class while adding a `topColor` property and `raiseTop` and `lowerTop` behaviors. The properties of the `Car` class, known as the *superclass* of `Convertible`, are also accessible by `Convertible`, which is considered a *subclass* of `Car`. Any code that requires a `Car` object may also use a `Convertible` (or any other subclass of `Car`) and then use it just like any other `Car`. This is known as *polymorphism* — the ability of an object to act or be treated like another object.

These concepts are the basic building blocks of all object-oriented programming languages, including Java. Understanding these concepts helps you comprehend the way object-oriented programming in Java works.

Key Java concepts

As an object-oriented language, Java makes use of all the concepts just discussed. Java also has a few key concepts that separate it from other programming languages. Among these concepts are the following:

- **Write once, run anywhere.** Applications written in Java don't run natively in the operating system. Instead, Java provides a virtual machine that runs natively in the operating system. In turn, Java programs run inside this virtual machine, which acts as a translator between the compiled Java application and the operating system, converting the Java program instructions into operating system instructions. Virtual machines are available for most major operating systems, including multiple versions of Windows, Mac OS X, and Linux.
- **Built-in libraries.** Java comes with a number of useful libraries right out of the box. These include libraries for networking, working with databases, and creating graphical user interfaces.

- **Automatic memory management.** In many other programming languages, the programmer is responsible for making sure that any memory used up by objects created by the program is freed when it's no longer needed. This is problematic for a couple of reasons. The first reason is that it requires that memory management code be mixed in with application logic, which makes the application logic harder to maintain. The second is that if the programmer forgets to add memory management code everywhere it's needed, the application could end up with what's known as a *memory leak*. An application with a memory leak continues to use more and more memory until it runs out altogether, causing the application to crash.

Java has an automatic memory manager, known as the *garbage collector*, that monitors all the objects created in the virtual machine and disposes of the ones that are no longer in use by any part of the application, thus freeing up the memory used by those objects. In this way, the Java virtual machine removes the burden of memory management from the program, leaving the programmer to concentrate on the logic of the application, not memory management.

Java syntax

This code listing shows how the previously discussed `Car` class might be written in Java:

```
package com.wiley.jfib.ch01;

public class Car
{
   public String color;
   public String bodyType;
   public boolean isStarted;
   public int currentSpeed;

   public Car()
   {
         currentSpeed = 0;
   }

   public void accelerate()
   {
         currentSpeed += 1;
   }

   public void start()
   {
         isStarted = true;
   }

   public void stop()
   {
         isStarted = false;
   }
}
```

The first line of this class is known as the *package declaration*. A package in Java is a means of grouping related classes together.

The next line of this class is known as the *class declaration*. The keyword `public` indicates that the class is able to be *instantiated* — that is, new instances of this class can be created by other code.

The next four lines of this class define the properties of the class and their data types. `Color` and `bodyType` are both *strings*, which contain character data. The `isStarted` property is a *boolean*, which is a value that can be set to `true` or `false`. The `currentSpeed` property is an *int*, which contains integer values.

The block of code starting with `public Car()` and including the code within the curly braces is known as the *constructor*. The constructor is used by other code to *instantiate* (construct an instance of) a class by using the `new` operator. For example, if some code requires an instance of the `Car` class, it can include this line to create one:

```
Car carInstance = new Car();
```

The next three blocks of code, which define the `accelerate`, `start`, and `stop` behaviors, are known as *methods*. A method in a Java class provides an interface for invoking a behavior. The `public` keyword indicates that each of these methods is available for any other code to invoke. Other method access keywords are `private`, which means that only the class itself can invoke the method, and `protected`, which means that subclasses and any classes in the same package as the class can invoke the method. The `void` keyword indicates that these methods return no value when they've finished executing. Methods can return no value; simple types of values, such as `int` or `boolean`; or any object.

Here's the code listing for the `Convertible` class:

```
package com.wiley.jfib.ch01;

public class Convertible extends Car
{
   public String topColor;
   public boolean isTopUp;

   public Convertible()
   {
         super();
         isTopUp = true;
   }

   public void raiseTop()
   {
         isTopUp = true;
   }
```

```
        public void lowerTop()
        {
              isTopUp = false;
        }
    }
```

The syntax for the `Convertible` class is the same as for the `Car` class, with a few differences. First, in the class declaration, notice that `Convertible extends Car`. This indicates that the `Convertible` class inherits the properties and behaviors of the `Car` class. Second, in the constructor, there's a call to `super()`. This call tells the constructor of the `Convertible` class to invoke the constructor of the `Car` class, the superclass of `Convertible`.

This final code listing shows an example of a method in another class that uses the `Car` class:

```
    public void driveToWork()
    {
        Car companyCar = new Car();
        companyCar.color = "black";
        companyCar.bodyType = "sedan";
        companyCar.start();
        while(companyCar.currentSpeed < 55)
        {
              companyCar.accelerate();
        }

        // some logic to determine when the
        // car arrives at work would go here

        companyCar.stop();
    }
```

This method instantiates a new `Car` object by using the `Car` class's constructor, sets its `color` and `bodyType` properties, invokes its `start()` method, and invokes `accelerate()` until the value of the `currentSpeed` property reaches 55. Finally, the `Car` object's `stop()` method is invoked.

The Java SE Development Kit

The first thing you need to get started with Java development is Java itself. The Java SE Development Kit, or JDK, contains both the Java runtime needed to run Java applications and the Java compiler needed to compile Java source code into Java applications. Much like Flex, where your MXML and ActionScript files are compiled into one or more SWF files, Java source code files are compiled into binary Java class files that can be run inside the Java virtual machine.

Installing the JDK

The most recent version of the JDK for all platforms is always available from Sun at `http://java.sun.com/javase/downloads/index.jsp`. As of this writing, JDK 6 Update 10 is the current version.

NOTE **Updates to the JDK are likely, so use the current version available.**

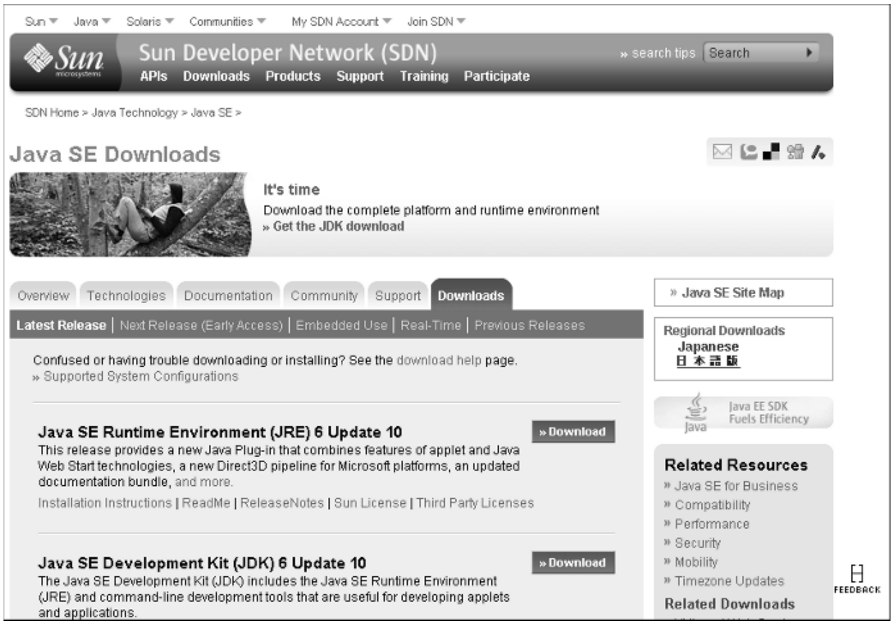
The JDK for Windows is packaged as a standard Windows installer. Once the installer has finished installing the JDK, some additional configuration steps are required.

To download the installer from the Java Web site, follow these steps:

1. **Click the Download button for the current version of the JDK on the download page, as shown in Figure 1.1.** You're automatically redirected to the next page.
2. **On the next page, as shown in Figure 1.2, choose the Windows operating system and Multi-language options from the dropdown lists, click the check box to indicate your acceptance of the license agreement, and then click Continue to go to the next page.**

FIGURE 1.1

On the Java 6 download page, click the Download button for the current version of the JDK.


Sun ▾ Java ▾ Solaris ▾ Communities ▾ My SDN Account ▾ Join SDN ▾
Sun Developer Network (SDN)
» search tips Search
APIs Downloads Products Support Training Participate
SDN Home > Java Technology > Java SE >
Overview Technologies Documentation Community Support Downloads
Latest Release | Next Release (Early Access) | Embedded Use | Real-Time | Previous Releases
» Java SE Site Map
» Download
» Download
FEEDBACK

Java SE Downloads

It's time
Download the complete platform and runtime environment
» Get the JDK download

Confused or having trouble downloading or installing? See the download help page.
» Supported System Configurations

Java SE Runtime Environment (JRE) 6 Update 10
This release provides a new Java Plug-in that combines features of applet and Java Web Start technologies, a new Direct3D pipeline for Microsoft platforms, an updated documentation bundle, and more.
Installation Instructions | ReadMe | ReleaseNotes | Sun License | Third Party Licenses

Java SE Development Kit (JDK) 6 Update 10
The Java SE Development Kit (JDK) includes the Java SE Runtime Environment (JRE) and command-line development tools that are useful for developing applets and applications.

Regional Downloads
Japanese
日本語版

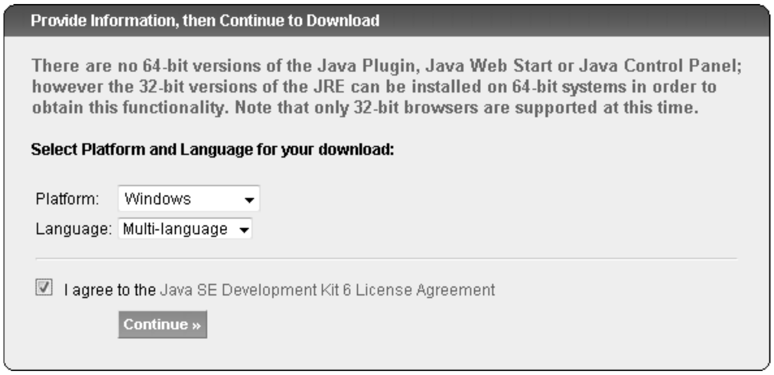
Java EE SDK Fuels Efficiency

Related Resources
» Java SE for Business
» Compatibility
» Performance
» Security
» Mobility
» Timezone Updates
Related Downloads

3. **On the final page, as shown in Figure 1.3, click the `jdk-6u10-windows-i586-p.exe` link for the Windows Offline Installation.** The offline installation option is a larger download but doesn't require a network connection to install once downloaded.
4. **Choose Save rather than Run to save the file to your computer.** Pick a location that you can remember.

FIGURE 1.2

On the next page of the Java download Web site, select the Windows operating system and Multi-language options from the dropdown lists and click the check box to accept the license agreement.



Provide Information, then Continue to Download

There are no 64-bit versions of the Java Plugin, Java Web Start or Java Control Panel; however the 32-bit versions of the JRE can be installed on 64-bit systems in order to obtain this functionality. Note that only 32-bit browsers are supported at this time.

Select Platform and Language for your download:

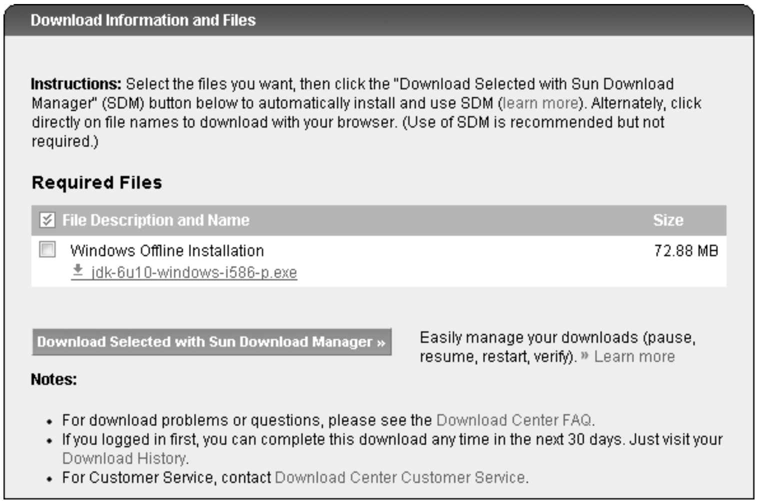
Platform: Windows

Language: Multi-language

☑ I agree to the Java SE Development Kit 6 License Agreement

FIGURE 1.3

On the final page of the Java download Web site, click the Windows Offline Installation to download the installer.



Download Information and Files

Instructions: Select the files you want, then click the "Download Selected with Sun Download Manager" (SDM) button below to automatically install and use SDM (learn more). Alternately, click directly on file names to download with your browser. (Use of SDM is recommended but not required.)

Required Files

File Description and Name	Size
Windows Offline Installation jdk-6u10-windows-i586-p.exe	72.88 MB

Easily manage your downloads (pause, resume, restart, verify). » Learn more

Notes:

- For download problems or questions, please see the Download Center FAQ.
- If you logged in first, you can complete this download any time in the next 30 days. Just visit your Download History.
- For Customer Service, contact Download Center Customer Service.

To run the installer from the saved location, follow these steps:

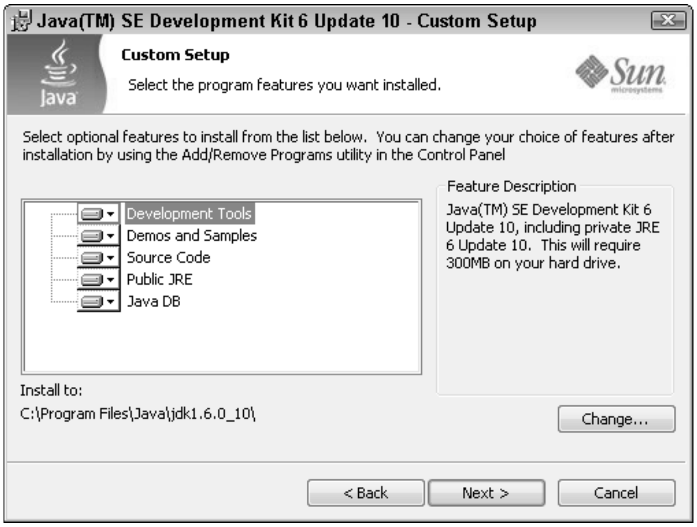
1. **Double-click the installer executable.**
2. **Read the Sun Binary Code License Agreement for the JDK and then click the Accept button to continue.**
3. **On the second screen, as shown in Figure 1.4, choose the location where you want to install the JDK and which features to include and then click Next to continue.** The default values are suitable for most users. If you need to change the install location, click Change and then choose a new location. In either case, be sure to note the location. The JDK installer displays a progress bar that shows the progress of the JDK installation before going to the next step.

NOTE **By default, the installer installs all features, including demos and sample code. The sample code is worth a look if you're new to Java development.**

4. **Once the JDK has finished installing, choose the installation location for the stand-alone Java Runtime Environment (JRE) and Java browser plug-in, as shown in Figure 1.5.** These components allow Java applications installed on your computer as well as Java applets hosted on Web sites to run. The default values are suitable for most users. If you need to change the install location, click the Change button and then choose a new location. The JDK installer displays a progress bar that shows the progress of the JRE installation before going to the next step.
5. **Click Finish to exit the installer.** Installation of the JDK and JRE is now complete.

FIGURE 1.4

The default installation location and settings. These are acceptable for most installations.



Java(TM) SE Development Kit 6 Update 10 - Custom Setup

Custom Setup

Select the program features you want installed.

Select optional features to install from the list below. You can change your choice of features after installation by using the Add/Remove Programs utility in the Control Panel

- Development Tools
- Demos and Samples
- Source Code
- Public JRE
- Java DB

Feature Description

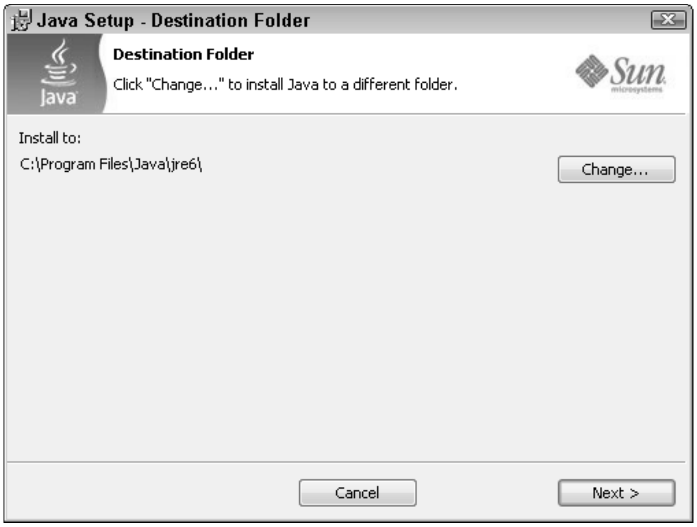
Java(TM) SE Development Kit 6 Update 10, including private JRE 6 Update 10. This will require 300MB on your hard drive.

Install to:

C:\Program Files\Java\jdk1.6.0_10\

FIGURE 1.5

The Java Runtime Environment and Java browser plug-in are installed separately from the JDK. Again, the default values are acceptable for most installations.



Java Setup - Destination Folder

Destination Folder

Click "Change..." to install Java to a different folder.

Install to:

C:\Program Files\Java\jre6\

Configuring the JDK

Although it's possible to run the tools in the JDK without any configuration, taking the time now to perform a few simple configuration steps makes it easier to configure other Java development tools later. Most Java development tools and servers expect (or at least prefer) that a couple of system environment variables have been set. These environment variables make it easier for other applications to locate your installed JDK and to execute the Java compiler and runtime without needing to configure the full path to the Java installation in each development tool you install.

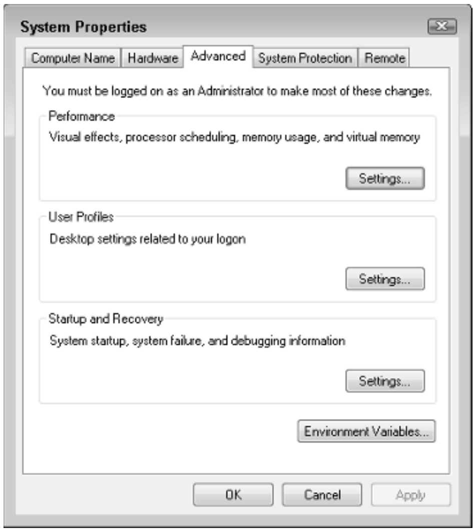
To modify the environment variables, follow these steps:

1. **Open the System Properties dialog box, as shown in Figure 1.6:**
   - **In Windows Vista:** Choose Start ➪ Control Panel ➪ System and Maintenance ➪ System ➪ Advanced System Settings.
   - **In Windows XP using the Control Panel's Classic View:** Choose Start ➪ Control Panel ➪ System and then click the Advanced tab.
   - **In Windows XP using the Control Panel's Category View:** Choose Start ➪ Control Panel ➪ Performance and Maintenance ➪ System and then click the Advanced tab.

2. **Click the Environment Variables button.** The Environment Variables dialog box opens. As shown in Figure 1.7, the dialog box is divided into two sections:
   - **User variables.** These are specific to the environment of the user currently logged into the Windows system.
   - **System variables.** These are globally available to any application run by any user on the system.

FIGURE 1.6

The Advanced tab in the System Properties dialog box is where you set necessary environment variables.



You must be logged on as an Administrator to make most of these changes.

Visual effects, processor scheduling, memory usage, and virtual memory

Desktop settings related to your logon

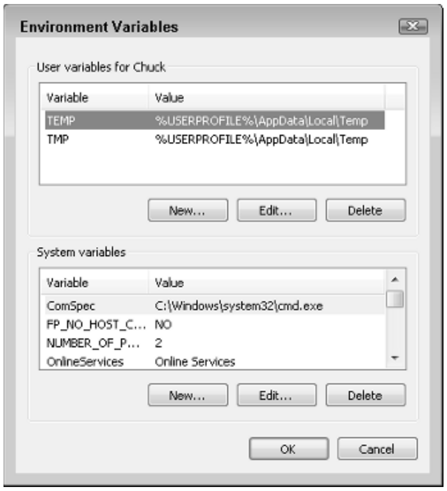
System startup, system failure, and debugging information

NOTE **Typically, you only add or modify environment variables in the User variables section. However, if your Java development machine is shared among multiple developers with different logins, it may make sense to set environment variables in the System variables section. Setting System variables may require a Windows user account with Administrator privileges.**

3. **Click the New button in the User variables section.** The New User Variable dialog box opens.
4. **Type** JAVA_HOME **in the Variable name text field, type the full path to your JDK installation in the Variable value text field, as shown in Figure 1.8, and then click OK.** The `JAVA_HOME` variable lets other applications know where your JDK is installed. Both the JBoss application server and the Eclipse IDE use this environment variable.

FIGURE 1.7

The Environment Variables dialog box. User variables are available only to the logged-in user, while System variables are globally available.



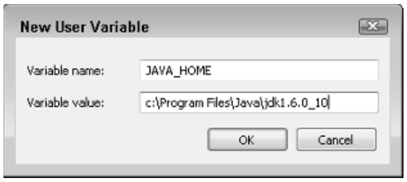
Variable	Value
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
FP_NO_HOST_C...	NO
NUMBER_OF_P...	2
OnlineServices	Online Services

5. **Click the New button in the User variables section.** The New User Variable dialog box opens.
6. **Type** PATH **in the Variable name text field, type** %JAVA_HOME%\bin **in the Variable value text field, and then click OK.** The `PATH` environment variable tells the system the specific locations to look for executable programs. By adding the `JAVA_HOME\bin` entry to the `PATH` environment variable, the system can find the Java compiler and runtime executables when they're called.
7. **Click OK to exit the Environment Variables dialog box and then click OK to save your settings.** Now that you've installed and configured the JDK, you can install other development tools that use it.

FIGURE 1.8

In the New User Variable dialog box, type JAVA_HOME in the Variable name text field and the full path to your JDK (`C:\Program Files\Java\jdk1.6.0_10`, for example) in the Variable value text field.



Variable name: JAVA_HOME
Variable value: c:\Program Files\Java\jdk1.6.0_10

NOTE **If you're adding environment variables to the System variables section, the `PATH` variable likely already exists. If so, select the `PATH` variable from the variables list and then click the Edit button rather than the New button. Type the full path to the bin directory inside the JDK installation folder (for example, `C:\Program Files\Java\jdk1.6.0_10\bin`) at the end of the path, preceded by a semicolon to separate it from the rest of the values.**

The JBoss Application Server

Java Web applications consist of a combination of compiled Java code; standard Web assets, such as HTML files and images; and Java Server Pages (JSPs), which allow dynamic content to be retrieved from the server and displayed in a browser. Java Web applications are typically packaged in a Web Application Archive (WAR) file and then deployed to an application server, which is responsible for providing the runtime resource management that the Web application needs.

The application server used to run the applications in this book is JBoss. JBoss is an open-source application server for Java Web applications. JBoss is popular for Java development because it's free to download and use, closely follows Java standards, and is easy to configure. As mentioned previously, JBoss is written in Java. You use the JDK you previously installed to run JBoss.

Installing JBoss

The JBoss application server can be downloaded from the JBoss Web site at `www.jboss.org/jbossas/downloads`. As of this writing, the latest stable version of JBoss is 4.2.3GA.

NOTE **JBoss version 5.0.0 is in its release candidate phase and may be available by the time you read this. Installation and configuration may be slightly different for this version.**

Clicking the Download link for the latest stable version takes you to the file-listing page for that release. For JDK 6 for Windows, click the download link labeled `jboss-<version number>-jdk6.zip` (for example, `jboss-4.2.3.GA-jdk6.zip`). Save the archive to your machine and then extract it to a directory of your choice. JBoss is packaged in and runs from a self-contained folder. No installation is required.

Configuring JBoss

Initial configuration of JBoss involves setting an environment variable and optionally editing JBoss's run.bat startup batch file. Once you're ready to deploy your Web application to JBoss, more detailed configuration may be necessary. The configuration steps listed here allow you to run JBoss with its default settings, which are appropriate for the Web applications in this book.

To run JBoss with its default settings, follow these steps:

1. **Open the System Properties dialog box:**
   - **In Windows Vista:** Choose Start ➪ Control Panel ➪ System and Maintenance ➪ System ➪ Advanced System Settings.

- **In Windows XP using the Control Panel's Classic View:** Choose Start ➪ Control Panel ➪ System and then click the Advanced tab.
- **In Windows XP using the Control Panel's Category View:** Choose Start ➪ Control Panel ➪ Performance and Maintenance ➪ System and then click the Advanced tab.

2. **Click the Environment Variables button.** The Environment Variables dialog box opens.
3. **Click the New button in the User variables section.** The New User Variable dialog box opens.
4. **Type** JBOSS_HOME **in the Variable name text field, type the full path to your JBoss installation in the Variable value text field, and then click OK.** The `JBOSS_HOME` variable lets other applications know where JBoss is installed on your system. This helps you use automated tools to handle deploying your compiled Java applications to the JBoss server without needing to remember or type the full installation path each time.
5. **Click OK to exit the Environment Variables dialog box and then click OK to save your settings.**

The batch file run.bat, as shown in Figure 1.9, is located in the `bin` subfolder of your JBoss install directory. This file is used to start the JBoss server for Windows and contains a number of parameters that are passed to the Java virtual machine (JVM) when starting up the JBoss server. Most of the default parameters are fine for development purposes. However, one parameter almost certainly needs to be changed.

Using Notepad, open the run.bat file by choosing File ➪ Open from the menu and then navigating to the `bin` subfolder of your JBoss install to select it. When you have opened the file, look for the following line:

```
JAVA_OPTS=%JAVA_OPTS% -Xms128m -Xmx512m
```

This line configures two parameters: `-Xms128m` and `-Xmx512m` refer to the minimum and maximum Java heap size, respectively. The *heap size* is the amount of physical memory Java uses to store its objects. These are the parameters you most likely need to adjust. When dealing with large Java applications, these default values may not be enough to meet the server's memory requirements. If you're using a development machine with a healthy amount of RAM, consider increasing the minimum and maximum heap sizes. By increasing the minimum heap size, you decrease the likelihood that the JVM needs to take the time to allocate more memory to the heap. By increasing the maximum heap size, you decrease the likelihood that the JVM runs out of memory, thus causing your application to stop responding altogether. In production environments, it's typical to set the minimum and maximum heap sizes to the same value. For development environments, the minimum heap size is less important. If you eventually find that JBoss runs out of memory with the specified maximum heap size, increase that value as needed by replacing the `-Xmx512m` value with a higher number.

NOTE **Although it's not strictly necessary to do so, using increments of 128MB is standard practice for adjusting the heap size value.**

CAUTION **Don't set the maximum heap size higher than the amount of memory your system has. The system may become unstable if the JVM tries to use all the memory your system has.**

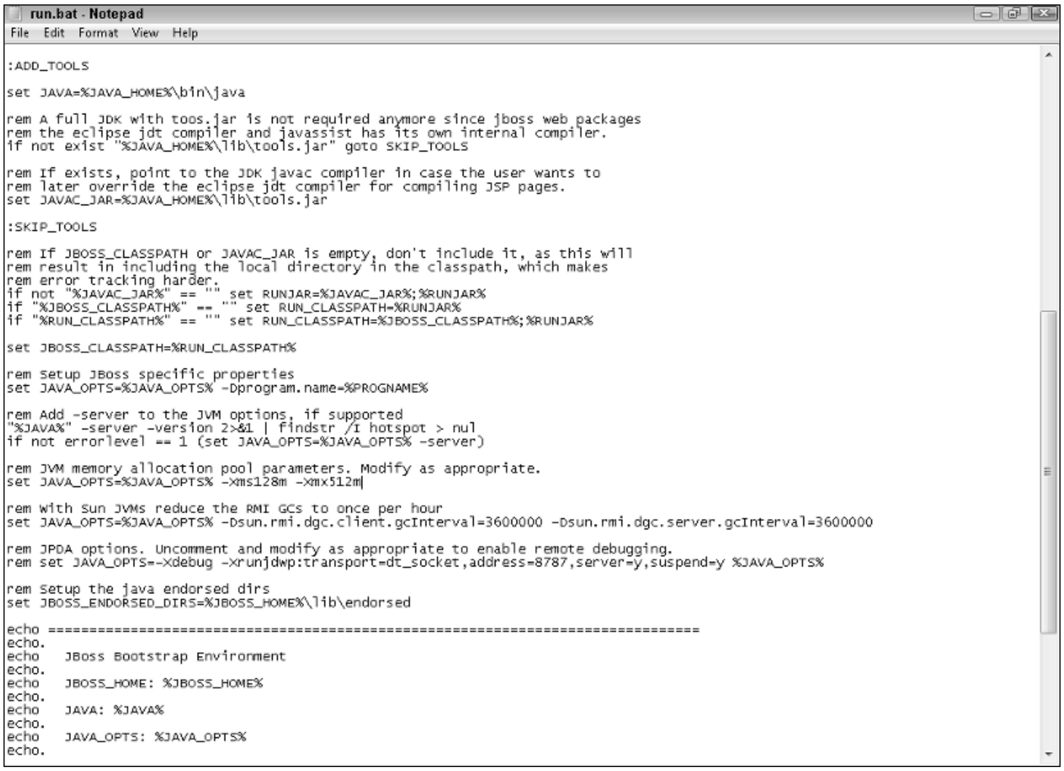
To test your JBoss installation, double-click run.bat. You see a Windows command prompt window open and some startup information scroll by. When the scrolling has stopped, the bottom line in the window indicates that the server has started. You can verify this by opening a browser and then typing the following in the address bar:

```
http://localhost:8080
```

You should see the JBoss welcome page shown in Figure 1.10. The JBoss welcome page includes links to the JBoss site, including documentation and discussion forums. These are great resources for becoming familiar with JBoss.

FIGURE 1.9

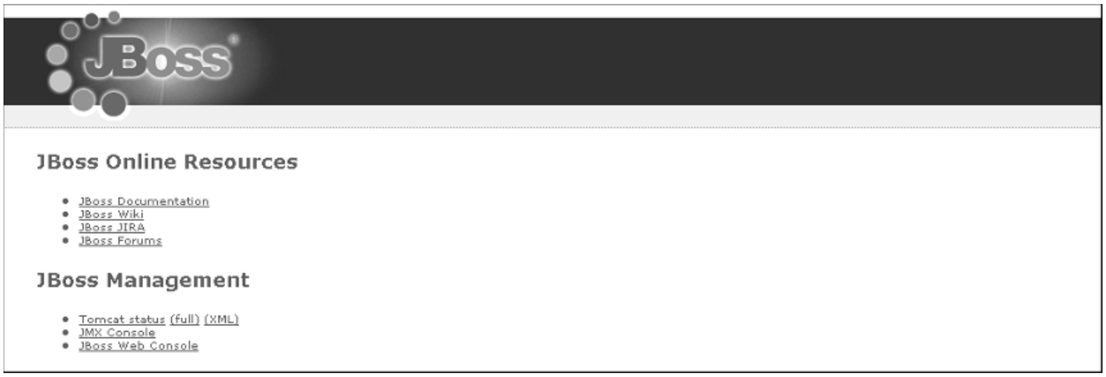
The `run.bat` file, which starts the JBoss server in Windows and contains arguments for the JVM that runs JBoss



```
:ADD_TOOLS

set JAVA=%JAVA_HOME%\bin\java

rem A full JDK with toos.jar is not required anymore since jboss web packages
rem the eclipse jdt compiler and javassist has its own internal compiler.
if not exist "%JAVA_HOME%\lib\tools.jar" goto SKIP_TOOLS

rem If exists, point to the JDK javac compiler in case the user wants to
rem later override the eclipse jdt compiler for compiling JSP pages.
set JAVAC_JAR=%JAVA_HOME%\lib\tools.jar

:SKIP_TOOLS

rem If JBOSS_CLASSPATH or JAVAC_JAR is empty, don't include it, as this will
rem result in including the local directory in the classpath, which makes
rem error tracking harder.
if not "%JAVAC_JAR%" == "" set RUNJAR=%JAVAC_JAR%;%RUNJAR%
if "%JBOSS_CLASSPATH%" == "" set RUN_CLASSPATH=%RUNJAR%
if "%RUN_CLASSPATH%" == "" set RUN_CLASSPATH=%JBOSS_CLASSPATH%;%RUNJAR%

set JBOSS_CLASSPATH=%RUN_CLASSPATH%

rem Setup JBoss specific properties
set JAVA_OPTS=%JAVA_OPTS% -Dprogram.name=%PROGNAME%

rem Add -server to the JVM options, if supported
"%JAVA%" -server -version 2>&1 | findstr /I hotspot > nul
if not errorlevel == 1 (set JAVA_OPTS=%JAVA_OPTS% -server)

rem JVM memory allocation pool parameters. Modify as appropriate.
set JAVA_OPTS=%JAVA_OPTS% -Xms128m -Xmx512m

rem With Sun JVMs reduce the RMI GCs to once per hour
set JAVA_OPTS=%JAVA_OPTS% -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000

rem JPDA options. Uncomment and modify as appropriate to enable remote debugging.
rem set JAVA_OPTS=-Xdebug -Xrunjdwp:transport=dt_socket,address=8787,server=y,suspend=y %JAVA_OPTS%

rem Setup the java endorsed dirs
set JBOSS_ENDORSED_DIRS=%JBOSS_HOME%\lib\endorsed

echo ===============================================================================
echo.
echo   JBoss Bootstrap Environment
echo.
echo   JBOSS_HOME: %JBOSS_HOME%
echo.
echo   JAVA: %JAVA%
echo.
echo   JAVA_OPTS: %JAVA_OPTS%
echo.
```

FIGURE 1.10

The JBoss welcome page provides verification of a correctly installed and configured JBoss server. It also provides links to documentation on the JBoss Web site.


JBoss Online Resources

- JBoss Documentation
- JBoss Wiki
- JBoss JIRA
- JBoss Forums

JBoss Management

- Tomcat status (full) (XML)
- JMX Console
- JBoss Web Console

Apache Ant

It's possible to do all the compilation of your Java applications simply by using the compiler included with the JDK. In reality, that process doesn't scale up well for larger projects. For example, consider a large enterprise application consisting of multiple Web applications and libraries. To build such an application by using only JDK tools would involve invoking the Java compiler with a lengthy classpath argument to compile the code, invoking separate commands to package up each Web application into its own Web application archive (WAR) file, invoking yet another command to package the WAR files and any libraries needed by the applications into an enterprise application archive (EAR) file, and finally manually copying the EAR file to the application server's deployment directory. With this many manual steps, the chance of problems arising increases.

Large software projects typically use automated build processes that not only handle the compilation of source code but also take care of other tasks, such as the packaging and deployment of the compiled application and even running unit tests to ensure that the compiled application works as expected before it's deployed to a production environment. In order to effectively automate the build process, it's necessary to use build tools that run easily and consistently every time.

The most widely used build tool in Java development is Apache Ant. Ant is a command-line tool written in Java that uses XML build files to build your Java projects. Ant's build files divide builds into discrete sets of tasks called *targets*. Ant allows you to chain together targets in such a way that a single command can compile your code, package up your application, and deploy it to the application server.

CROSS-REF **For a detailed example of writing an Ant build file and using it to deploy your application to the JBoss server, see Chapter 6.**

Installing Apache Ant

Apache Ant can be downloaded from the Apache Ant project's Web site at `http://ant.apache.org/bindownload.cgi`. As of this writing, the latest stable version of Apache Ant is 1.7.1.

Apache Ant comes packaged as a ZIP archive file. Click the download link labeled `apache-ant-<version>-bin.zip` (for example, `apache-ant-1.7.1-bin.zip`). Save the archive to your machine and then extract it to the directory of your choice. Apache Ant is self-contained and doesn't require installation.

Configuring Apache Ant

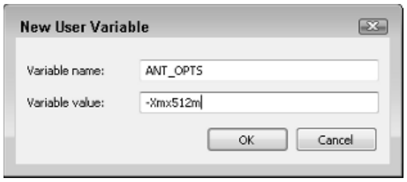
As with the JDK, you need to set a few environment variables in order to make it easy for other applications to find and work with Ant.

To modify the environment variables, follow these steps:

1. **Open the System Properties dialog box:**
   - **In Windows Vista:** Choose Start ➪ Control Panel ➪ System and Maintenance ➪ System ➪ Advanced System Settings.
   - **In Windows XP using the Control Panel's Classic View:** Choose Start ➪ Control Panel ➪ System and then click the Advanced tab.
   - **In Windows XP using the Control Panel's Category View:** Choose Start ➪ Control Panel ➪ Performance and Maintenance ➪ System and then click the Advanced tab.
2. **Click the Environment Variables button.** The Environment Variables dialog box opens.
3. **Click the New button in the User variables section.** The New User Variable dialog box opens.
4. **Type** ANT_HOME **in the Variable name text field, type the full path to your Apache Ant installation (**`c:\apache-ant-1.7.1`**, for example) in the Variable value text field, and then click OK.** The `ANT_HOME` variable lets other applications know where Ant is installed on your system.
5. **Click the New button in the User variables section again.** The New User Variable dialog box opens.
6. **Type** ANT_OPTS **in the Variable name text field, type the maximum heap size for Ant to use in the Variable value text field, and then click OK.** Because Ant is written in Java, it uses a Java heap to manage the objects it uses, much like JBoss does. Some tasks in Ant require more memory than others, and building projects with large amounts of source code or many resources to package could cause Ant to run out of memory if the heap size is too small. Setting the `ANT_OPTS` environment variable allows you to specify enough of a maximum heap size to ensure that Ant has enough memory and to easily adjust the amount of memory as your project grows. Set the value of the `ANT_OPTS` variable to the same maximum heap size as your JBoss installation by using the same `-Xmx` notation you saw in the JBoss run.bat configuration file, as shown in Figure 1.11.

FIGURE 1.11

Set the value of the ANT_OPTS environment variable to the same maximum heap size you set in JBoss's run.bat file (-Xmx512m, for example).

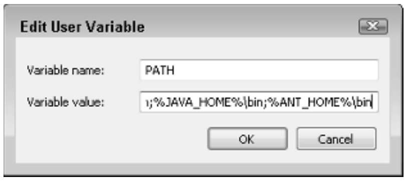


Variable name: ANT_OPTS
Variable value: -Xmx512m

7. **Click the PATH environment variable in the User variables section and then click the Edit button.** The Edit User Variable dialog box opens, and the Variable name and Variable value text fields are populated with the current values for the PATH variable.
8. **Add** %ANT_HOME%\bin **to the Variable value by using a semicolon to separate it from the** %JAVA_HOME%\bin **entry and then click OK.** See Figure 1.12. Apache Ant's executables are now available to other applications.
9. **Click OK to exit the Environment Variables dialog box and then click OK to save your settings.**

FIGURE 1.12

Add **%ANT_HOME%\bin** to the PATH environment variable. Separate it from the existing values by using a semicolon.



Variable name: PATH
Variable value: ı;%JAVA_HOME%\bin;%ANT_HOME%\bin

NOTE **If you're adding environment variables to the System variables section, the PATH variable likely already exists. If so, select the PATH variable from the variable list and then click the Edit button rather than the New button. Type the full path to the bin directory inside the Ant installation (for example, C:\apache-ant-1.7.1\bin) at the end of the path, preceded by a semicolon to separate it from the rest of the values.**

To test your Apache Ant installation, follow these steps:

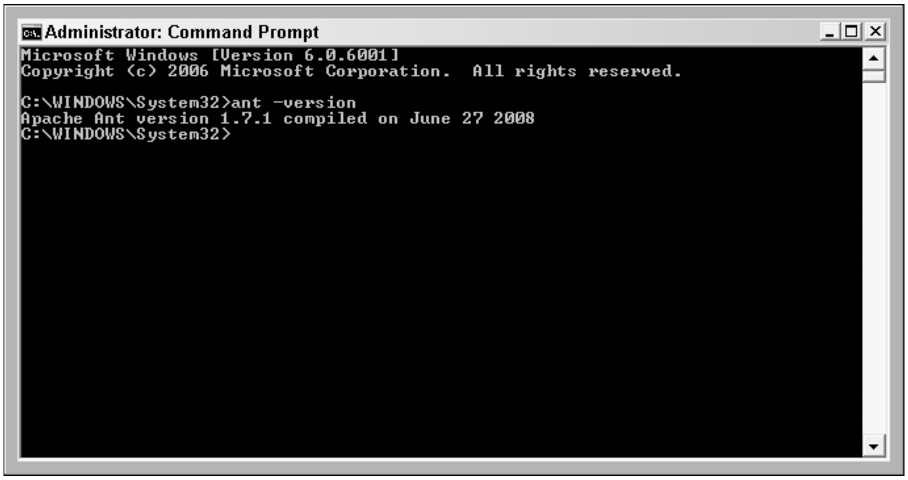
1. **Open a command prompt:**
   - **In Windows Vista:** Choose Start, type **cmd** in the Start Search box, and then choose cmd.exe from the Programs list to open the command prompt.

- **In Windows XP:** Choose Start ➪ Run, type **cmd** in the Run dialog box that opens, and then click OK to open the command prompt.

2. **Type** ant -version **in the command prompt window and then press Enter.** You should see a message, as shown in Figure 1.13, indicating the version of Apache Ant you installed. If you see a message stating that `ant` isn't recognized as a command, double-check that your `PATH` environment variable entry is accurate.

FIGURE 1.13

If Apache Ant is installed and configured correctly, running the `ant` command with the `-version` option should print a message similar to this.



```
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.

C:\WINDOWS\System32>ant -version
Apache Ant version 1.7.1 compiled on June 27 2008
C:\WINDOWS\System32>
```

The Eclipse Integrated Development Environment

Eclipse is an open-source integrated development environment (IDE) for Java and other languages. Eclipse is the most popular IDE for Java development because it's not only packed with features such as code completion and templates, but it also integrates with other standard Java tools, such as Ant and JBoss, and is extensible with a wide variety of plug-ins. Eclipse itself is written in Java, and you use the JDK you installed previously to run Eclipse as well as compile the Java code you write by using Eclipse.

Installing the Eclipse IDE

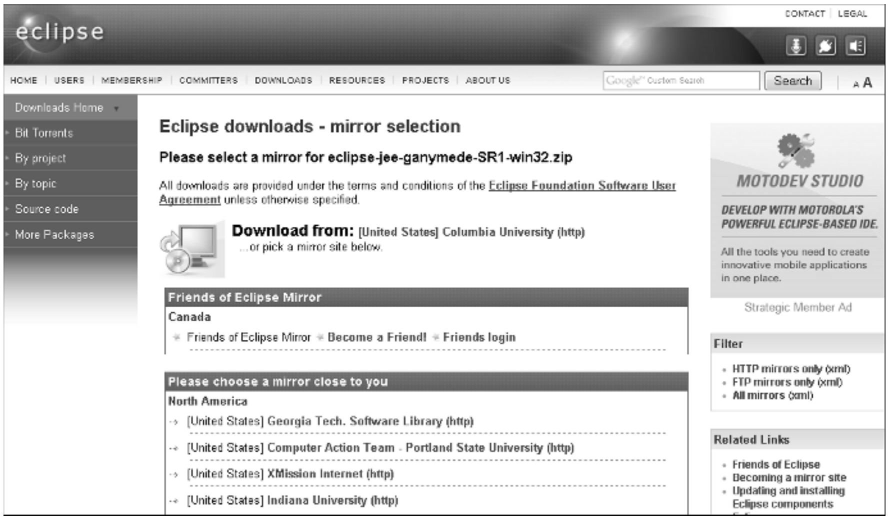
The Eclipse IDE can be downloaded from the Eclipse project's Web site at `www.eclipse.org/downloads/packages`. There are a number of packages available on this page. You should download the Eclipse IDE for Java EE Developers package. This package contains a number of tools, such as database views and XML editors, that are useful for your Java development work. As of this writing, the latest stable version of Eclipse is code-named Ganymede.

To download the Eclipse archive file, follow these steps:

1. **Click the Eclipse IDE for Java EE Developers link on the download page.** The next page opens with a list of mirror sites from which Eclipse can be downloaded, as shown in Figure 1.14. The mirror sites in this list are clones of the main Eclipse download site. Their purpose is to take some of the load off the main Eclipse download site by offering other locations from which you can download Eclipse.
2. **Click the link for the mirror site closest to you.** Choosing a mirror site that's geographically closer to you usually results in a faster download time.
3. **Click Save to download the archive to a location of your choice.**

FIGURE 1.14

The list of mirror sites where Eclipse can be downloaded. Choose the one closest to you geographically.



Eclipse downloads - mirror selection

Please select a mirror for eclipse-jee-ganymede-SR1-win32.zip

All downloads are provided under the terms and conditions of the Eclipse Foundation Software User Agreement unless otherwise specified.

Download from: [United States] Columbia University (http)
...or pick a mirror site below.

Friends of Eclipse Mirror
Canada
Friends of Eclipse Mirror · Become a Friend! · Friends login

Please choose a mirror close to you
North America
[United States] Georgia Tech. Software Library (http)
[United States] Computer Action Team - Portland State University (http)
[United States] XMission Internet (http)
[United States] Indiana University (http)

MOTODEV STUDIO
DEVELOP WITH MOTOROLA'S POWERFUL ECLIPSE-BASED IDE.
All the tools you need to create innovative mobile applications in one place.
Strategic Member Ad

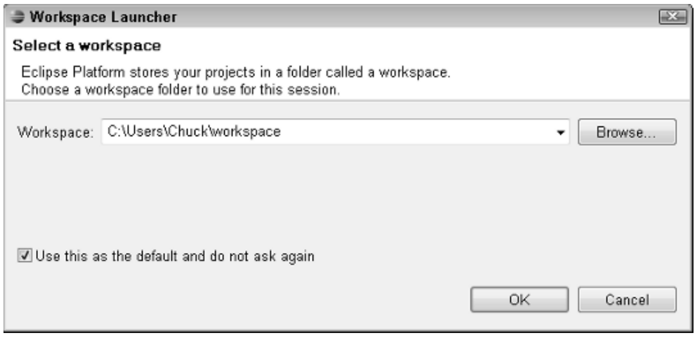
Filter
- HTTP mirrors only (xml)
- FTP mirrors only (xml)
- All mirrors (xml)

Related Links
- Friends of Eclipse
- Becoming a mirror site
- Updating and installing Eclipse components

4. **Extract the archive to a location of your choice.** Eclipse is self-contained and doesn't require installation.
5. **Start Eclipse by double-clicking** eclipse.exe **in the Eclipse directory you just extracted.** The Workspace Launcher dialog box, as shown in Figure 1.15, opens.

FIGURE 1.15

The Workspace Launcher dialog box allows you to choose a location for your Eclipse workspace and set that workspace as the default. The workspace contains Eclipse settings and properties. The default value is appropriate for most installations.



Workspace Launcher

Select a workspace

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

Workspace: C:\Users\Chuck\workspace

☑ Use this as the default and do not ask again

6. **Leave the default value for Workspace, click the Use this as the default and don't ask again check box, and then click OK.** Eclipse launches.

The workspace in Eclipse is a folder in which project configuration, global preferences, and project resources (such as source code) are stored. The default location is a folder named `workspace` under your user files directory. It's possible to have more than one workspace in Eclipse and switch between them at will. This allows you to maintain logical groupings of related projects and keep separate preferences and window layouts for each workspace. For the projects in this book, you need only one workspace.

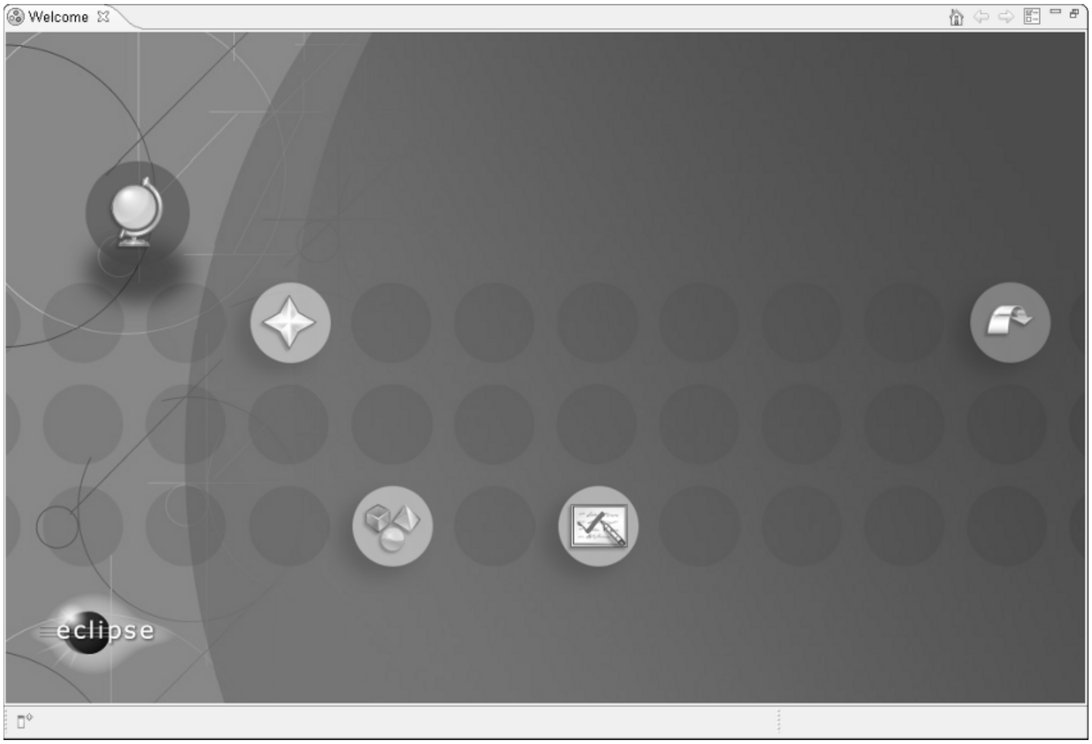
When Eclipse opens for the first time, the Welcome view is displayed. The Welcome view, as shown in Figure 1.16, provides icons that link to an overview of the Eclipse IDE, descriptions of new features in this version of Eclipse, samples, and tutorials. If you're new to Eclipse or Java development in general, it's worth spending some time with these materials.

Configuring the Eclipse IDE

Eclipse is almost infinitely configurable, and the list of configuration options can be daunting. For the Java and Flex development you do in this book, only a couple of configuration options are necessary.

FIGURE 1.16

The Welcome view in Eclipse contains links to documentation, tutorials, and other valuable information for both new and experienced developers.

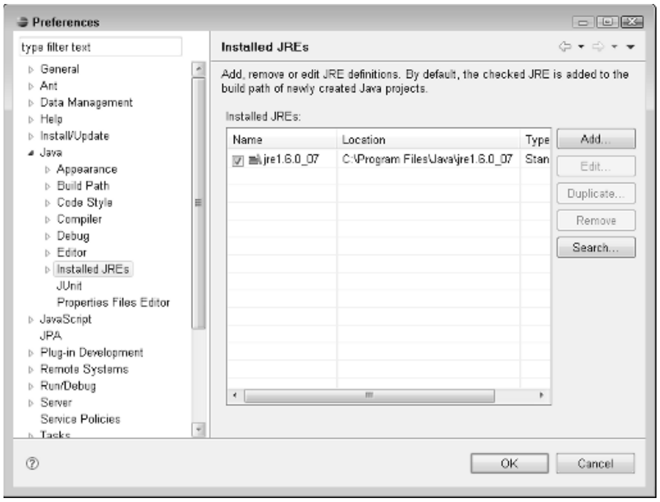


Follow these steps to configure Eclipse:

1. **Choose Window ➪ Preferences.** The Preferences dialog box opens.
2. **In the left pane, click the arrow next to Java.** The Java submenu is expanded.
3. **Choose Installed JREs from the Java submenu.** The Installed JREs list appears in the right pane, as shown in Figure 1.17. Because you set the `JAVA_HOME` environment variable when you installed the JDK, you see it listed here and selected as the default option. If the JDK you installed doesn't appear here, follow these steps to add it by using the Add JRE wizard:

FIGURE 1.17

The expanded Java submenu with the Installed JREs item selected in the Preferences dialog box



Installed JREs

Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects.

Installed JREs:

Name	Location	Type
☑ jre1.6.0_07	C:\Program Files\Java\jre1.6.0_07	Stan

1. **Click the Add button next to the Installed JREs list.** The Add JRE wizard opens.
2. **Choose Standard VM from the dropdown list and then click Next.** See Figure 1.18.
3. **Click Directory, navigate to your JDK installation directory, and then click OK.** The wizard populates the JRE name and JRE system libraries text fields for you, as shown in Figure 1.19.
4. **Click Finish.** The Add JRE wizard closes. Your JDK now appears in the Installed JREs list.

4. **In the left pane of the Preferences dialog box, click the arrow next to Ant.** The Ant submenu is expanded. Eclipse has its own built-in version of Apache Ant, but it's best for the sake of consistency to use the same Ant installation both inside and outside the Eclipse environment.
5. **Choose Runtime from the Ant submenu.** The Runtime properties appear in the right pane, with the Classpath tab selected, as shown in Figure 1.20.
6. **Click the Ant Home button on the Classpath tab.** The Browse for Folder dialog box opens.
7. **Choose your Apache Ant install directory and then click OK.** The Ant Home Entries item now shows the path to your Apache Ant installation. Expand that item to see the set of Java archive (JAR) files corresponding to your Apache Ant installation, as shown in Figure 1.21.
8. **In the left pane, click the arrow next to Server.** The Server submenu is expanded.

FIGURE 1.18

The first screen of the Add JRE wizard. Choose Standard VM from the dropdown list.

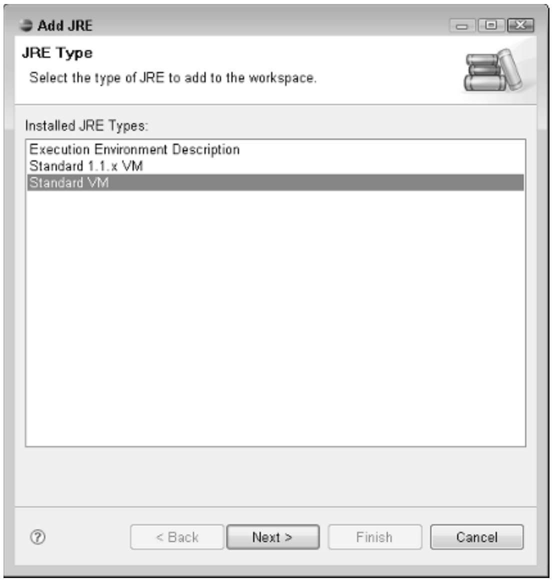
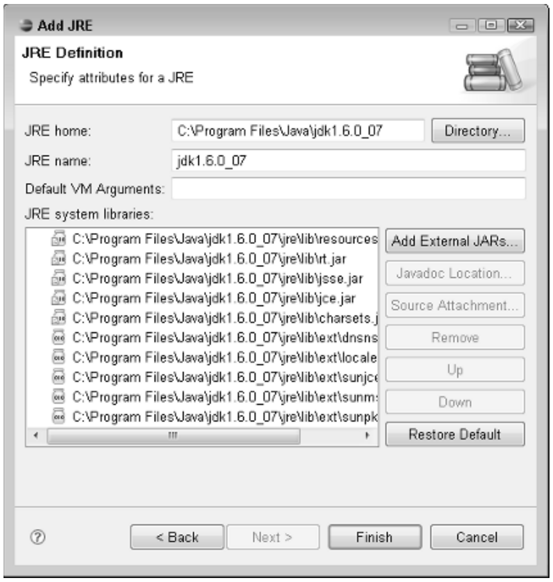


FIGURE 1.19

After you select your JDK installation directory in the second screen of the Add JRE wizard, the wizard automatically fills in the JRE name and JRE system libraries text fields for you.

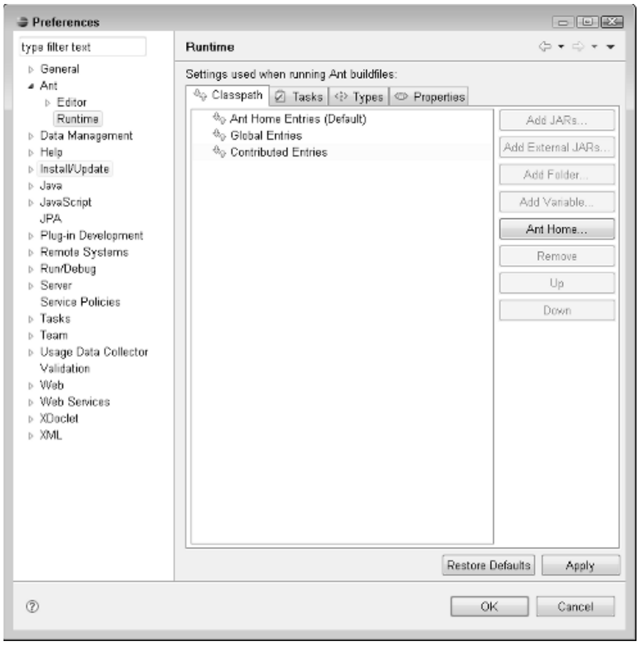


JRE home: C:\Program Files\Java\jdk1.6.0_07
JRE name: jdk1.6.0_07
Default VM Arguments:
JRE system libraries:

```
C:\Program Files\Java\jdk1.6.0_07\jre\lib\resources
C:\Program Files\Java\jdk1.6.0_07\jre\lib\rt.jar
C:\Program Files\Java\jdk1.6.0_07\jre\lib\jsse.jar
C:\Program Files\Java\jdk1.6.0_07\jre\lib\jce.jar
C:\Program Files\Java\jdk1.6.0_07\jre\lib\charsets.j
C:\Program Files\Java\jdk1.6.0_07\jre\lib\ext\dnsns
C:\Program Files\Java\jdk1.6.0_07\jre\lib\ext\locale
C:\Program Files\Java\jdk1.6.0_07\jre\lib\ext\sunjce
C:\Program Files\Java\jdk1.6.0_07\jre\lib\ext\sunm:
C:\Program Files\Java\jdk1.6.0_07\jre\lib\ext\sunpk
```

FIGURE 1.20

The expanded Ant submenu with the Runtime item selected in the Preferences dialog box. The Classpath tab is selected by default.



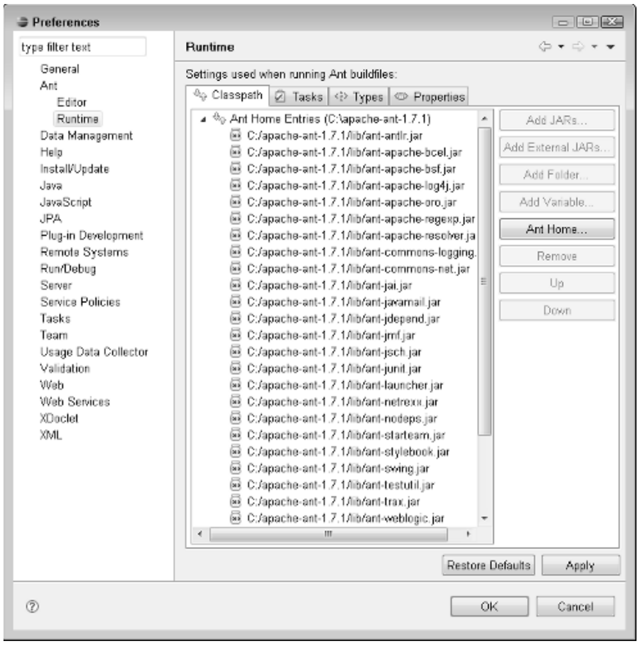
9. **Choose Runtime Environments from the Server submenu.** The Server Runtime Environments list appears in the right pane, as shown in Figure 1.22. Here, you add your JBoss installation to your Eclipse configuration. This allows you to take advantage of Eclipse's debugging features. You're able to step through your code running on your JBoss server, start and stop the server, and more — all from within Eclipse.

10. **Click the Add button next to the Server runtime environments list.** The New Server Runtime Environment wizard opens.

11. **Click the arrow next to JBoss, choose JBoss v4.2 from the expanded list, click the Create a new local server check box, and then click Next.** See Figure 1.23.

FIGURE 1.21

Once you've chosen your Ant installation directory in the Browse for Folder dialog box, Eclipse fills in the Ant Home Entries with the Ant installation's runtime JAR files.



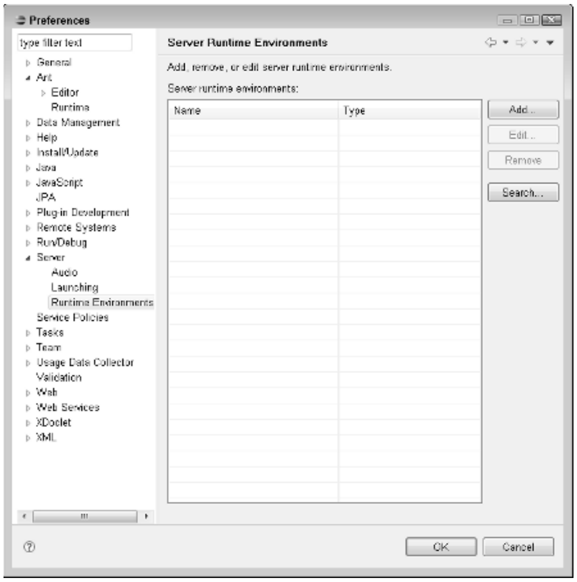
Settings used when running Ant buildfiles:

```
Ant Home Entries (C:\apache-ant-1.7.1)
    C:/apache-ant-1.7.1/lib/ant-antlr.jar
    C:/apache-ant-1.7.1/lib/ant-apache-bcel.jar
    C:/apache-ant-1.7.1/lib/ant-apache-bsf.jar
    C:/apache-ant-1.7.1/lib/ant-apache-log4j.jar
    C:/apache-ant-1.7.1/lib/ant-apache-oro.jar
    C:/apache-ant-1.7.1/lib/ant-apache-regexp.jar
    C:/apache-ant-1.7.1/lib/ant-apache-resolver.ja
    C:/apache-ant-1.7.1/lib/ant-commons-logging.
    C:/apache-ant-1.7.1/lib/ant-commons-net.jar
    C:/apache-ant-1.7.1/lib/ant-jai.jar
    C:/apache-ant-1.7.1/lib/ant-javamail.jar
    C:/apache-ant-1.7.1/lib/ant-jdepend.jar
    C:/apache-ant-1.7.1/lib/ant-jmf.jar
    C:/apache-ant-1.7.1/lib/ant-jsch.jar
    C:/apache-ant-1.7.1/lib/ant-junit.jar
    C:/apache-ant-1.7.1/lib/ant-launcher.jar
    C:/apache-ant-1.7.1/lib/ant-netrexx.jar
    C:/apache-ant-1.7.1/lib/ant-nodeps.jar
    C:/apache-ant-1.7.1/lib/ant-starteam.jar
    C:/apache-ant-1.7.1/lib/ant-stylebook.jar
    C:/apache-ant-1.7.1/lib/ant-swing.jar
    C:/apache-ant-1.7.1/lib/ant-testutil.jar
    C:/apache-ant-1.7.1/lib/ant-trax.jar
    C:/apache-ant-1.7.1/lib/ant-weblogic.jar
```

12. **Click the Browse button next to the Application Server Directory text box.** The Browse for Folder dialog box opens.
13. **Choose your JBoss installation folder, click OK, and then click Next.** See Figure 1.24.
14. **Click Finish.** The default values for Address, Port, JNDI Port, and Server Configuration, as shown in Figure 1.25, are acceptable. Your JBoss configuration now appears in the Server runtime environments list.
15. **Click OK.** The Preferences dialog box closes.

FIGURE 1.22

The expanded Server submenu in the Preferences dialog box, with the Runtime Environments item selected

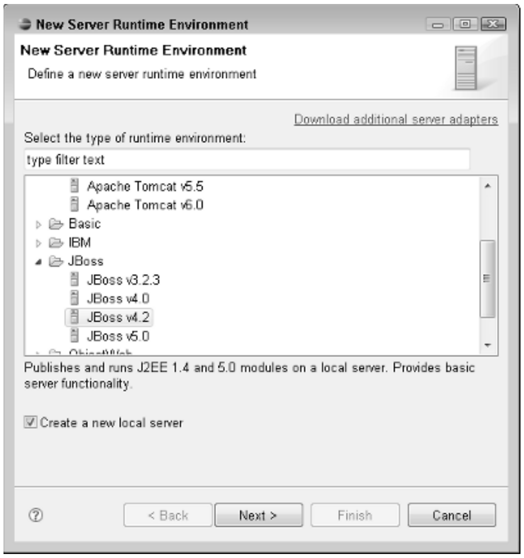


Add, remove, or edit server runtime environments.

Server runtime environments:

FIGURE 1.23

The New Server Runtime Environment wizard, with the JBoss v4.2 item selected



Define a new server runtime environment

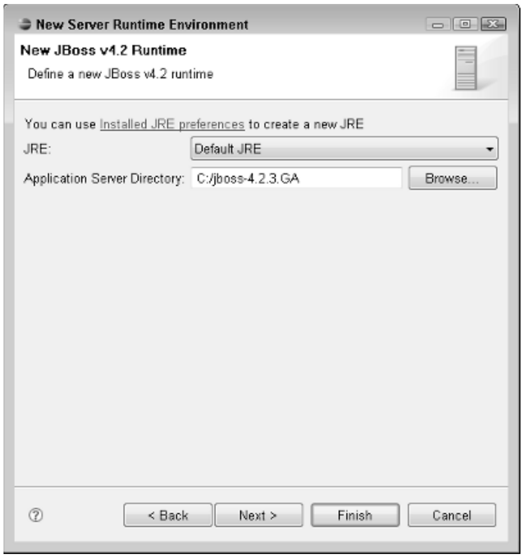
Select the type of runtime environment:

Publishes and runs J2EE 1.4 and 5.0 modules on a local server. Provides basic server functionality.

☑ Create a new local server

FIGURE 1.24

Choose your JBoss installation folder from the Browse for Folder dialog box.


New Server Runtime Environment
New JBoss v4.2 Runtime
Define a new JBoss v4.2 runtime

You can use Installed JRE preferences to create a new JRE

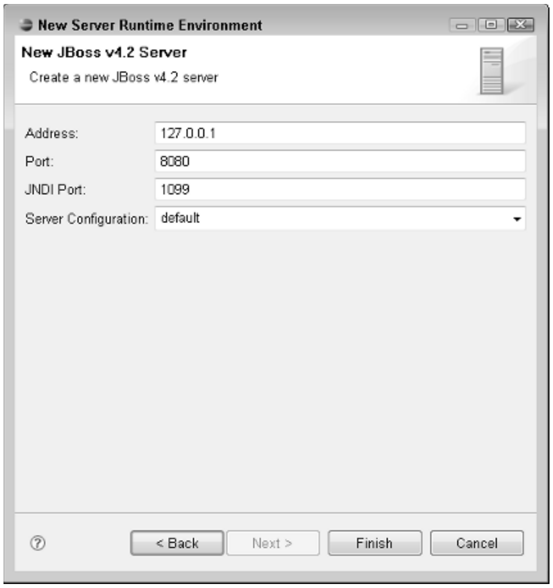
JRE: Default JRE

Application Server Directory: C:/jboss-4.2.3.GA

Browse...
< Back
Next >
Finish
Cancel

FIGURE 1.25

The default values for Address, Port, JNDI Port, and Server Configuration in the last screen of the New Server Runtime Environment wizard are fine as they are.


New Server Runtime Environment
New JBoss v4.2 Server
Create a new JBoss v4.2 server

Address: 127.0.0.1

Port: 8080

JNDI Port: 1099

Server Configuration: default

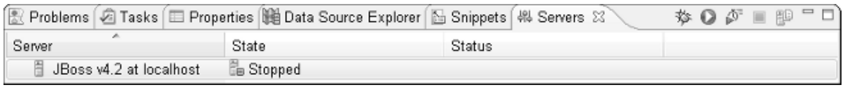
< Back
Next >
Finish
Cancel

To test your JBoss configuration, close the Welcome tab by clicking the X on the top-right corner of the tab and then follow these steps:

1. **Choose Window ➪ Show View ➪ Servers.** The Servers view, as shown in Figure 1.26, opens. You should see your JBoss configuration listed in the Servers view with the value Stopped in the State column.

FIGURE 1.26

The Servers view showing the stopped JBoss server

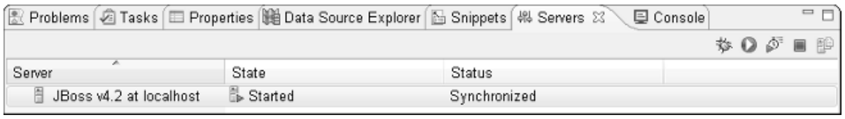


Server	State	Status
JBoss v4.2 at localhost	Stopped	

2. **Choose your JBoss server and then either click the Start the Server button (the round green button with the white arrow to the right of the Servers view tab) or press Ctrl+Alt+R.** As JBoss starts up, the Console view opens and displays JBoss startup messages. Once JBoss has started successfully, Eclipse switches back to the Servers view, and the value Started appears in the State column, as shown in Figure 1.27.
3. **Choose your JBoss server and then either click the Stop the Server button (the square red button to the right of the Servers view tab) or press Ctrl+Alt+S.** As JBoss shuts down, the Console view opens and displays JBoss shutdown messages. Once JBoss has stopped successfully, Eclipse switches back to the Servers view, and the value Stopped appears in the State column.

FIGURE 1.27

The Servers view showing the started JBoss server



Server	State	Status
JBoss v4.2 at localhost	Started	Synchronized

As mentioned previously, one of the things that makes Eclipse a popular choice among Java developers is the ability to extend its functionality through a plug-in system. The last part of configuring Eclipse for the Web applications in this book is installing one such plug-in: the Spring IDE plug-in. The Spring IDE plug-in provides a number of useful tools for developing applications by using the popular Spring Framework.

CROSS-REF **For more on the Spring Framework, see Chapter 6.**

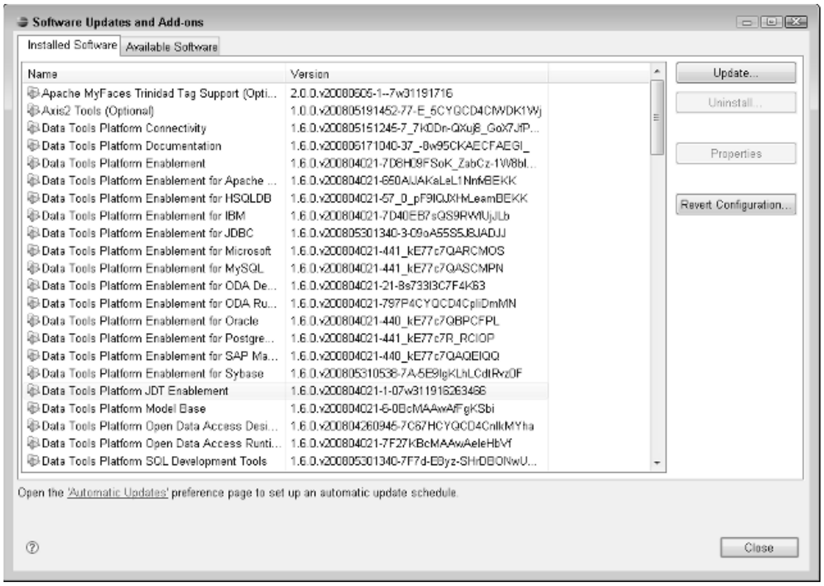
Eclipse plug-ins are typically installed by providing Eclipse's software update manager with a URL where the plug-in can be downloaded and then choosing the features you want to install. This process is used to install the Spring IDE plug-in.

To install the Spring IDE plug-in for Eclipse, follow these steps:

1. **Choose Help ➪ Software Updates.** The Software Updates and Add-ons dialog box opens with the Installed Software tab selected, as shown in Figure 1.28.

FIGURE 1.28

The Software Updates and Add-ons dialog box consists of two tabs. The Installed Software tab, selected by default, shows a list of currently installed features.



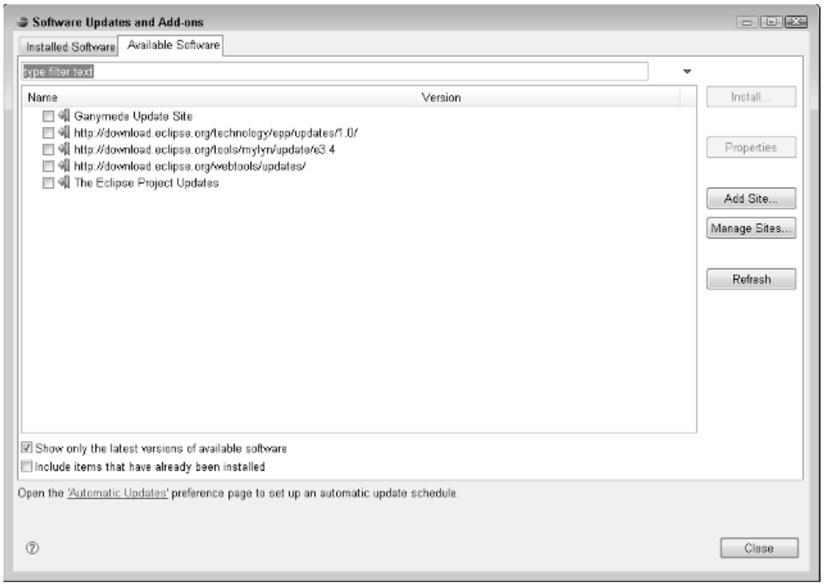
Name	Version
Apache MyFaces Trinidad Tag Support (Opti...	2.0.0.v20080605-1--7w31191716
Axis2 Tools (Optional)	1.0.0.v200805191452-77-E_5CYQCD4CMWDK1Wj
Data Tools Platform Connectivity	1.6.0.v200805151245-7_7KDDn-QXuj8_GoX7JfP...
Data Tools Platform Documentation	1.6.0.v200806171040-37_-8w95CKAECFAEGI_
Data Tools Platform Enablement	1.6.0.v200804021-7D8H09FSoK_ZabCz-1W8bl...
Data Tools Platform Enablement for Apache ...	1.6.0.v200804021-650AIJAKaLeL1NnfMBEKK
Data Tools Platform Enablement for HSQLDB	1.6.0.v200804021-57_0_pF9IQJXHMLeamBEKK
Data Tools Platform Enablement for IBM	1.6.0.v200804021-7D40EB7sQS9RWIUjJLb
Data Tools Platform Enablement for JDBC	1.6.0.v200805301340-3-09oA55S5J8JADJJ
Data Tools Platform Enablement for Microsoft	1.6.0.v200804021-441_kE77c7QARCMOS
Data Tools Platform Enablement for MySQL	1.6.0.v200804021-441_kE77c7QASCMPN
Data Tools Platform Enablement for ODA De...	1.6.0.v200804021-21-8s733I3C7F4K63
Data Tools Platform Enablement for ODA Ru...	1.6.0.v200804021-797P4CYQCD4CpliDmMN
Data Tools Platform Enablement for Oracle	1.6.0.v200804021-440_kE77c7QBPCFPL
Data Tools Platform Enablement for Postgre...	1.6.0.v200804021-441_kE77c7R_RCIOP
Data Tools Platform Enablement for SAP Ma...	1.6.0.v200804021-440_kE77c7QAQEIQQ
Data Tools Platform Enablement for Sybase	1.6.0.v200805310538-7A-5E9IgKLhLCdIRvz0F
Data Tools Platform JDT Enablement	1.6.0.v200804021-1-07w311916263466
Data Tools Platform Model Base	1.6.0.v200804021-6-0BcMAAwAfFgKSbi
Data Tools Platform Open Data Access Desi...	1.6.0.v200804260945-7C67HCYQCD4CnlkMYha
Data Tools Platform Open Data Access Runti...	1.6.0.v200804021-7F27KBcMAAwAeleHbVf
Data Tools Platform SQL Development Tools	1.6.0.v200805301340-7F7d-E8yz-SHrDBONwU...

Open the 'Automatic Updates' preference page to set up an automatic update schedule.

2. **Click the Available Software tab.** The list of currently configured software update sites is displayed, as shown in Figure 1.29. The update sites listed here are those for the features Eclipse provides upon installation. These sites can be used to add features or update existing features.

FIGURE 1.29

The Available Software tab of the Software Updates and Add-ons dialog box lists currently configured software update sites. Software update sites in Eclipse are sites where Eclipse features and plug-ins can be downloaded and installed from within Eclipse.



Name	Version
☐ Ganymede Update Site	
☐ http://download.eclipse.org/technology/epp/updates/1.0/	
☐ http://download.eclipse.org/tools/mylyn/update/e3.4	
☐ http://download.eclipse.org/webtools/updates/	
☐ The Eclipse Project Updates	

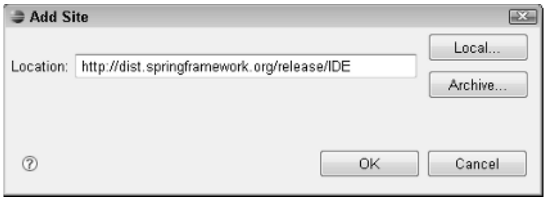
☑ Show only the latest versions of available software
☐ Include items that have already been installed

Open the 'Automatic Updates' preference page to set up an automatic update schedule.

3. **Click the Add Site button.** The Add Site dialog box opens.
4. **Type** http://dist.springframework.org/release/IDE **in the Location text field, as shown in Figure 1.30, and then click OK.** The Add Site dialog box closes, and the Spring IDE Update Site is added to the list of update sites and is expanded to display the features available for installation, as shown in Figure 1.31. There are a number of features available with Spring IDE. Aside from the Core, which provides the basic Spring IDE installation and is required, only a few of the other options are necessary here:
   - **The Spring IDE AOP Extension and Spring IDE AOP Developer Resources provide tools that help with aspect-oriented programming.** Among other things, aspect-oriented programming helps developers deal with modularizing *cross-cutting concerns* — functionality such as logging and error handling that affects multiple modules in an application.
   - **The Spring IDE Security Extension and Spring IDE Security Developer Resources provide tools that help with implementing Spring Security in Web applications.** Spring Security provides functionality for handling authentication, resource protection, and other features to make sure only users with proper credentials have access to application resources.

FIGURE 1.30

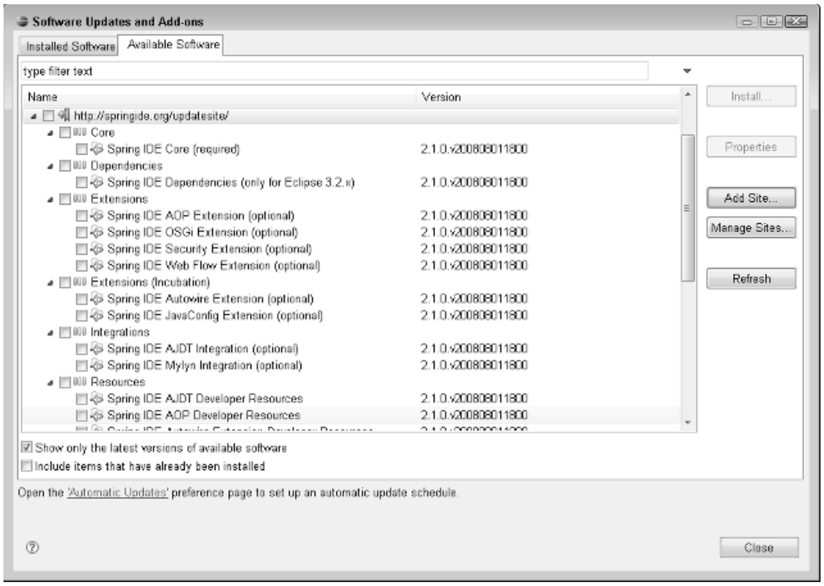
Type the URL for the Spring IDE update site in the Location text field of the Add Site dialog box.



Location: http://dist.springframework.org/release/IDE

FIGURE 1.31

Once you provide Eclipse with the URL for the Spring IDE update site, it appears in the list of configured software update sites and expands to display the features available for installation.



Name	Version
http://springide.org/updatesite/	
Core	
Spring IDE Core (required)	2.1.0.v200808011800
Dependencies	
Spring IDE Dependencies (only for Eclipse 3.2.x)	2.1.0.v200808011800
Extensions	
Spring IDE AOP Extension (optional)	2.1.0.v200808011800
Spring IDE OSGi Extension (optional)	2.1.0.v200808011800
Spring IDE Security Extension (optional)	2.1.0.v200808011800
Spring IDE Web Flow Extension (optional)	2.1.0.v200808011800
Extensions (Incubation)	
Spring IDE Autowire Extension (optional)	2.1.0.v200808011800
Spring IDE JavaConfig Extension (optional)	2.1.0.v200808011800
Integrations	
Spring IDE AJDT Integration (optional)	2.1.0.v200808011800
Spring IDE Mylyn Integration (optional)	2.1.0.v200808011800
Resources	
Spring IDE AJDT Developer Resources	2.1.0.v200808011800
Spring IDE AOP Developer Resources	2.1.0.v200808011800

☑ Show only the latest versions of available software
☐ Include items that have already been installed

Open the 'Automatic Updates' preference page to set up an automatic update schedule.

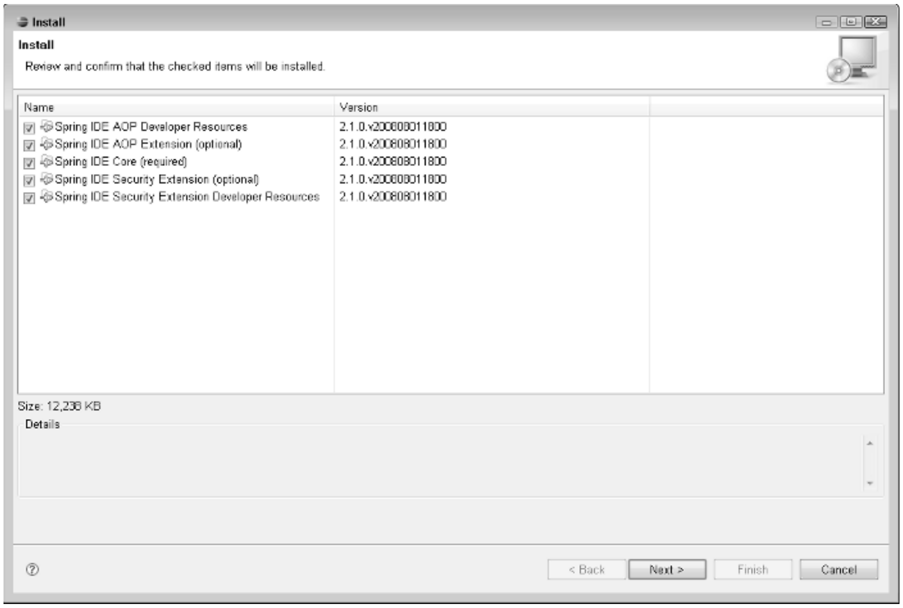
5. **Click the check boxes next to Spring IDE Core, Spring IDE AOP Extension, Spring IDE Security Extension, Spring IDE AOP Developer Resources, and Spring IDE Security Extension Developer Resources and then click the Install button.** Eclipse resolves any dependencies to make sure any selected features don't require other features that aren't present. Upon successful dependency resolution, the Install wizard, as shown in Figure 1.32, opens.

6. **Click the Next button.** The Review Licenses screen, as shown in Figure 1.33, opens. Review the license for each feature you're installing.
7. **Click the radio button to accept the license agreement for each feature and then click the Finish button.** The Install dialog box opens, showing the progress of the feature download and installation, as shown in Figure 1.34. Once all features have been installed, the Install dialog box closes, and the Software Updates dialog box opens, recommending that Eclipse be restarted.
8. **Click the Yes button to restart Eclipse.** Eclipse closes and then reopens.
9. **Once Eclipse has reopened, choose Window ➪ Show View ➪ Other.** You should see a Spring folder in the list of views.

Once you install an Eclipse plug-in and restart Eclipse, the new plug-in is available in Eclipse. Each Eclipse plug-in adds different features, views, and menu items to Eclipse. Eclipse may or may not notify you about new features available when a plug-in is installed, so it's best to read the documentation for any Eclipse plug-ins you install to understand what features the plug-ins add. The Spring IDE plug-in adds some Spring views and project types to Eclipse.

FIGURE 1.32

Once Eclipse has made sure that all required dependencies for the features you're installing are present, the Install wizard opens, displaying a list of the chosen features.



Install

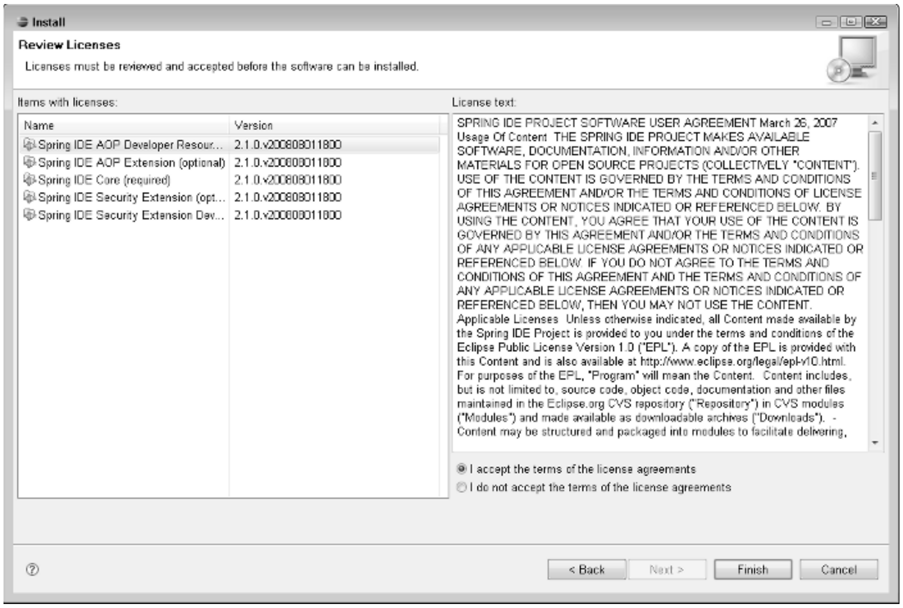
Review and confirm that the checked items will be installed.

Name	Version
✓ Spring IDE AOP Developer Resources	2.1.0.v200808011800
✓ Spring IDE AOP Extension (optional)	2.1.0.v200808011800
✓ Spring IDE Core (required)	2.1.0.v200808011800
✓ Spring IDE Security Extension (optional)	2.1.0.v200808011800
✓ Spring IDE Security Extension Developer Resources	2.1.0.v200808011800

Size: 12,238 KB

FIGURE 1.33

The Review Licenses screen displays the license agreements for the features you're installing. You can click each feature on the left side of the screen to display the license that applies to that feature.



Review Licenses

Licenses must be reviewed and accepted before the software can be installed.

Items with licenses:

Name	Version
Spring IDE AOP Developer Resour...	2.1.0.v200808011800
Spring IDE AOP Extension (optional)	2.1.0.v200808011800
Spring IDE Core (required)	2.1.0.v200808011800
Spring IDE Security Extension (opt...	2.1.0.v200808011800
Spring IDE Security Extension Dev...	2.1.0.v200808011800

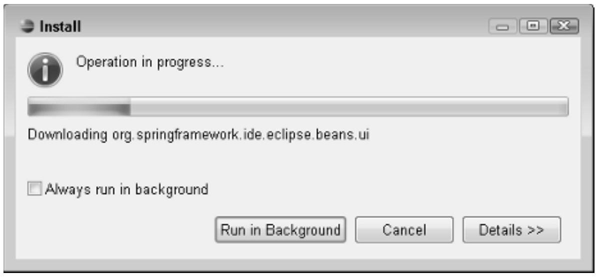
License text:

SPRING IDE PROJECT SOFTWARE USER AGREEMENT March 26, 2007 Usage Of Content THE SPRING IDE PROJECT MAKES AVAILABLE SOFTWARE, DOCUMENTATION, INFORMATION AND/OR OTHER MATERIALS FOR OPEN SOURCE PROJECTS (COLLECTIVELY "CONTENT"). USE OF THE CONTENT IS GOVERNED BY THE TERMS AND CONDITIONS OF THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW. BY USING THE CONTENT, YOU AGREE THAT YOUR USE OF THE CONTENT IS GOVERNED BY THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF ANY APPLICABLE LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW. IF YOU DO NOT AGREE TO THE TERMS AND CONDITIONS OF THIS AGREEMENT AND THE TERMS AND CONDITIONS OF ANY APPLICABLE LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW, THEN YOU MAY NOT USE THE CONTENT. Applicable Licenses Unless otherwise indicated, all Content made available by the Spring IDE Project is provided to you under the terms and conditions of the Eclipse Public License Version 1.0 ("EPL"). A copy of the EPL is provided with this Content and is also available at http://www.eclipse.org/legal/epl-v10.html. For purposes of the EPL, "Program" will mean the Content. Content includes, but is not limited to, source code, object code, documentation and other files maintained in the Eclipse.org CVS repository ("Repository") in CVS modules ("Modules") and made available as downloadable archives ("Downloads"). - Content may be structured and packaged into modules to facilitate delivering,

[✓ I accept the terms of the license agreements]
I do not accept the terms of the license agreements

FIGURE 1.34

After you accept the license agreements, Eclipse displays the progress of the feature installation. After all features have been installed, you need to restart Eclipse to ensure that all features are available.



Operation in progress...

Downloading org.springframework.ide.eclipse.beans.ui

☐ Always run in background

Summary

In this chapter, I discussed the installation and configuration of the tools you use to develop, compile, and run your Java applications. These include the Java SE Development Kit, which contains the compiler and runtime code for Java applications; the JBoss application server, the server environment in which Java Web applications are run; the Apache Ant build tools, which make the process of building, packaging, and deploying applications easier to manage and automate; and the Eclipse IDE, the development environment in which you write, compile, and debug the code for your Java Web applications.