**First Set classpath**

Set CLASSPATH=%CLASSPATH%; C:\Program Files\Apache Software Foundation\Tomcat 7.0\lib\servlet-api.jar

**Java Servlet**

**Six Steps to Running Your First Servlet**

1. Create a directory structure under Tomcat for your application.
2. Write the servlet source code. You need to import the javax.servlet package and the javax.servlet.http package in your source file.
3. Compile your source code.
4. Create a deployment descriptor.
5. Run Tomcat.
6. Call your servlet from a web browser.

**Step 1: Create a Directory Structure under Tomcat**

When you install Tomcat, several subdirectories are automatically created under the Tomcat home directory (%TOMCAT\_HOME%). One of the subdirectories is webapps. The webapps directory is where you store your web applications. A web application is a collection of servlets and other contents installed under a specific subset of the server's URL namespace. A separate directory is dedicated for each servlet application. Therefore, the first thing to do when you build a servlet application is create an application directory. This section explains how to create a directory structure for an application called myApp.

Create a directory called myApp under the webapps directory. The directory name is important because this also appears in the URL to your servlet.

Create the src and WEB-INF directories under myApp, and create a directory named classes under WEB-INF. The directory structure is shown in Figure 1.4. The src directory is for your source files, and the classes directory under WEB-INF is for your Java classes. If you have html files, you put them directly in the myApp directory. You also may want to create a directory called images under myApp for all your image files.

**Step 2: Write the Servlet Source Code**

import javax.servlet.\*;

import javax.servlet.http.\*;

import java.io.\*;

import java.util.\*;

public class TestingServlet extends HttpServlet {

public void doGet(HttpServletRequest request,

HttpServletResponse response)

throws ServletException, IOException {

PrintWriter out = response.getWriter();

out.println("<HTML>");

out.println("<HEAD>");

out.println("<TITLE>Servlet Testing</TITLE>");

out.println("</HEAD>");

out.println("<BODY>");

out.println("Welcome to the Servlet Testing Center");

out.println("</BODY>");

out.println("</HTML>");

}

}

Now, save your TestingServlet.java file to the src subdirectory under myApp. You actually can place the source files anywhere; however, it is always a good idea to be organized by storing all your source code files in the src directory.

**Step 3: Compile Your Source Code**

For your servlet source code to compile, you need to include in your CLASSPATH environment variable the path to the servlet.jar file. The servlet.jar is located in the common\lib\ subdirectory under %CATALINA\_HOME%.

NOTE

If you have forgotten how to edit the CLASSPATH environment variable, refer to Appendix A, "Tomcat Installation and Configuration."

If you are using Windows, remember that the new environment variable takes effect only for new console windows. In other words, after changing a new environment variable, open a new console window for typing your command lines.

Now, change directory to your working directory and type the following if you are using Windows:

javac -d ..\WEB-INF\classes\ TestingServlet.java

If you are using Linux/UNIX, the command is very similar, except that / is used to separate a directory from a subdirectory.

javac -d ../WEB-INF/classes/ TestingServlet.java

The -d option specifies where to place the generated class files. The command also assumes that you have placed the JDK's bin directory in the path so you can call any program in it from any directory.

**Step 4: Create the Deployment Descriptor**

A deployment descriptor is an optional component in a servlet application, taking the form of an XML document called web.xml. The descriptor must be located in the WEB-INF directory of the servlet application. When present, the deployment descriptor contains configuration settings specific to that application. Deployment descriptors are discussed in detail in Chapter 16. "Application Deployment."

For this step, you now need to create a web.xml file and place it under the WEB-INF directory under myApp.

The web.xml for this example application must have the following content.<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app

PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"

"http://java.sun.com/dtd/web-app\_2\_3.dtd">

<web-app>

<servlet>

<servlet-name>Testing</servlet-name>

<servlet-class>TestingServlet</servlet-class>

</servlet>

</web-app>

The web.xml file has one element: web-app. You should write all your servlets under <web-app>. For each servlet, you have a <servlet> element and you need the <servlet-name> and <servlet-class> elements. The <servlet-name> is the name for your servlet, by which it is known to Tomcat. The <servlet-class> is the compiled file of your servlet without the .class extension.

Having more than one servlet in an application is common. For every servlet, you need a <servlet> element in the web.xml file. For example, the following code shows how the web.xml looks if you add another servlet called Login.

<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app

PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"

"http://java.sun.com/dtd/web-app\_2\_3.dtd">

<web-app>

<servlet>

<servlet-name>Testing</servlet-name>

<servlet-class>TestingServlet</servlet-class>

</servlet>

<servlet>

<servlet-name>Login</servlet-name>

<servlet-class>LoginServlet</servlet-class>

</servlet>

</web-app>

**Step 5: Run Tomcat**

If it is not already running, you need to start Tomcat. For information on how to do that, see Appendix A, "Tomcat Installation and Configuration."

**Step 6: Call Your Servlet from a Web Browser**

You are ready to call your servlet from a web browser. By default, Tomcat runs on port 8080 in myApp virtual directory under the servlet subdirectory. The servlet that you just wrote is named Testing. The URL for that servlet has the following format:

http://domain-name/virtual-directory/servlet/servlet-name

If you run the web browser from the same computer as Tomcat, you can replace domain-name with localhost. Therefore, the URL for your servlet would be http://localhost:8080/myApp/servlet/Testing.

In the deployment descriptor you wrote in Step 4, you actually mapped the servlet class file called TestingServlet with the name "Testing" so that your servlet can be called by specifying its class file (TestingServlet) or its name (Testing). Without a deployment descriptor, your servlet must be called by specifying its class name; that is, TestingServlet. This means that if you had not written a deployment descriptor in Step 4, you would have to use the following URL to call your servlet:

http://localhost:8080/myApp/servlet/TestingServlet