

# How to use Tomcat 6 and Java SE 6 with *Murach's Java Servlets and JSP*

This document provides a chapter-by-chapter summary that identifies all of the changes that you need to make to get all of the applications, examples, and exercises in the *first edition* of *Murach's Java Servlets and JSP* to work with Tomcat 6.

Please note that Tomcat 6 only works with Java 5 or 6. As a result, you must use one of these versions of Java when working with Tomcat 6. In addition, this document assumes that you are using Java 6. To learn how to get the most from these versions of Java, we recommend our beginning Java book, *Murach's Java SE 6*.

Finally, if you want to use Tomcat 6, we recommend you get the *second edition* of *Murach's Java Servlets and JSP*, which will be available by the end of 2007. However, if you want to use the first edition of this book with Tomcat 6, you can follow the instructions below.

## Chapter 2

### Figure 2-1: Download the 6.0.X ZIP file

In figure 2-1, make sure to download the ZIP file for Tomcat 6.X. That way, the installation process will be more like the one described in chapter 2 of our book. It's also possible to download an EXE file for a setup program, but we don't recommend using this approach with this book.

### Figure 2-3: Set the JAVA\_HOME variable to the JDK 1.6

Before you set the JAVA\_HOME variable, you should download and install Java SE 6 as described in Appendix A. Then, you can use the procedure shown in figure 2-3 to set the JAVA\_HOME variable by editing the catalina.bat file that's in Tomcat's bin directory. However, the path for the JAVA\_HOME variable is likely to be longer. For example, by default, Java SE 6 is installed in a directory like this one:

```
set JAVA_HOME=C:\Program Files\Java\jdk1.6.0
```

Also, you need to set this variable immediately after the rem statements at the beginning of the bat file. For example:

```
rem          -Xdebug -Xrunjdw:transport=%JPDA_TRANSPORT%,
rem          address=%JPDA_ADDRESS%,server=y,suspend=%JPDA_SUSPEND%
rem
rem $Id: catalina.bat 500710 2007-01-28 00:24:33Z markt $
rem -----
set JAVA_HOME=C:\Program Files\Java\jdk1.6.0
```

To test this to make sure that it's working, you can start and shutdown Tomcat as shown in figure 2-6. If it starts without displaying an error message, you have successfully set the JAVA\_HOME variable.

### Figure 2-5: Class reloading does not work properly with Tomcat 6

The documentation for Tomcat 6 says that you can enable servlet reloading by opening the context.xml file in Tomcat's conf directory and adding a reloadable attribute to the Context element like this:

```
<Context reloadable="true">
```

Although this works for classes stored in the WEB-INF classes directory, it does not work for the classes that are stored in packages, which includes most of the classes presented in this book. As a result, when you use Tomcat 6 with the applications presented in this book, you need to stop and restart the server each time you want to reload a class that has been modified and recompiled. Fortunately, Tomcat 6 appears to start and stop more quickly than previous versions of Tomcat, so this isn't as much trouble as it seems at first.

### Optional: Turn off serialization

By default, Tomcat 6 attempts to save all objects that are stored in the session object every time it shuts down. This may cause Tomcat to display error messages when it shuts down and when it restarts after an unsuccessful shutdown. Since the applications in this book weren't designed to persist across Tomcat restarts, there's no need to attempt to save the objects stored in the session object when shutting down Tomcat.

To turn serialization off, open the context.xml file that's stored in Tomcat's conf directory and remove the comments from this element:

```
<Manager pathname="" />
```

### Required: Map the invoker servlet

With Tomcat 6, the invoker servlet is not mapped by default. Since all of the servlets presented in the first edition of *Murach's Java Servlets and JSP* rely on the mapping for the invoker servlet, you'll need to map the invoker servlet to use this book. To do this, you begin by opening the context.xml file in Tomcat's conf directory and adding a privileged attribute to the Context element like this:

```
<Context reloadable="true" privileged="true">
```

Then, you can continue by opening the web.xml file in Tomcat's conf directory. Within this file, you define the invoker servlet by removing the comments from this servlet element:

```
<servlet>
  <servlet-name>invoker</servlet-name>
  <servlet-class>
    org.apache.catalina.servlets.InvokerServlet
  </servlet-class>
  <init-param>
    <param-name>debug</param-name>
    <param-value>0</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>
```

In addition, you map the servlet to a URL pattern by removing the comments from this servlet-mapping element:

```
<!-- The mapping for the invoker servlet -->
<servlet-mapping>
  <servlet-name>invoker</servlet-name>
  <url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
```

It's also possible to map the invoker servlet for a single web application by adding the elements shown above to the appropriate place in the web.xml file in that application's WEB-INF directory. However, for the purposes of this book, it makes sense to map the invoker servlet for all web applications.

## Optional: Fix the admin warnings that are displayed on startup

When you install the applications that come with this book as described in appendix A, Tomcat will display a couple security warnings when it starts like this:

```
INFO: WARNING: Security role name service used in an <auth-constraint> without being defined in a <security-role>
```

Although these warnings aren't critical, you can easily fix them by adding a security-role element. To do that, open the web.xml file that's stored in the WEB-INF directory of the murach application. Then, scroll down after the login-config element and add these security-role elements:

```
<security-role>
  <description>customer service employees</description>
  <role-name>service</role-name>
</security-role>
<security-role>
  <description>system administrators</description>
  <role-name>admin</role-name>
</security-role>
```

To test this, start Tomcat as shown in figure 2-6. If you have successfully added these security role elements, Tomcat won't display any warning messages.

## Chapter 6

### Figure 6-8: The IllegalStateException

When you test the Email List application, if you don't enter data for all three text fields, the servlet shown in figure 6-8 will throw this exception:

```
java.lang.IllegalStateException: Cannot forward after response has been committed
email6.EmailServlet.doGet(EmailServlet.java:38)
```

That's because the if clause in this servlet forwards control to `get_missing_fields.jsp`, and then it tries to execute the rest of the code, which attempts to forward control to `show_email_entry.jsp`. To solve this problem, you can add an else clause around the code that writes the data to a file and forwards control to `show_email_entry.jsp` like this:

```
else
{
    // otherwise, write the data to a file and display the entry
    User user = new User(firstName, lastName, emailAddress);

    //UserIO.addRecord(user,
    //    "../webapps/murach/WEB-INF/etc/UserEmail.txt");
    ServletContext sc = getServletContext();
    String path = sc.getRealPath("WEB-INF/etc/UserEmail.txt");
    UserIO.addRecord(user, path);

    RequestDispatcher dispatcher =
        getServletContext().getRequestDispatcher(
            "/email6/show_email_entry.jsp");
    dispatcher.forward(request, response);
}
```

That way, this servlet will only forward the request to one JSP or the other.

## Chapter 9

### The missing TagExtraInfo class problem

When you test the applications for chapter 9, you may get this exception:

```
java.lang.NoClassDefFoundError: javax/servlet/jsp/tagext/TagExtraInfo
```

This message indicates that the JRE can't find the TagExtraInfo class. To make this class available to your JRE, you need to copy the jsp-api.jar and el-api.jar files from Tomcat's lib directory to Java's jre\lib\ext directory. If you followed the instructions in figure 2-1, the servlet-api.jar file should already be in this directory, which is what you want.

## Chapter 11

### The database driver not found problem

When you test either of the applications for chapter 11, you may get this exception:

```
java.lang.NullPointerException  
sql11.SQLGatewayServlet.doGet(SQLGatewayServlet.java:49)
```

This message indicates that the JRE can't create an instance of the MurachPool class. As a result, it can't get a connection to the database. This is because the JRE can't find the database driver, which is stored in the mm.mysql-2.0.8-bin.jar file. To make this class available to your JRE, you need to copy the mm.mysql-2.0.8-bin.jar from the murach\WEB-INF\lib directory to Java's jre\lib\ext directory.

## Conclusion

If you encounter any other issues when using *Murach's Java Servlets and JSP* with Tomcat 6 and Java SE 6, please send an email to [joelmurach@yahoo.com](mailto:joelmurach@yahoo.com) and I will update this document.

Thanks,

Joel Murach