

# How to use Tomcat 5.5 and J2SE 5.0 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 *Murach's Java Servlets and JSP* to work with Tomcat 5.5. Because Tomcat 5.5 was designed to work with Java 5.0, this document assumes that you are using Java 5.0.

To learn how to get the most from Java 5.0, we recommend our new edition of our beginning Java book, *Murach's Beginning Java 2, JDK 5*. We view this as an ideal companion book for *Murach's Java Servlets and JSP*, especially if you're using Tomcat 5.5.

## Chapter 2

### Figure 2-1: Download the 5.5.X ZIP file

In figure 2-1, make sure to download the ZIP file for Tomcat 5.5.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 1.5

Before you set the JAVA\_HOME variable, you should download and install J2SE 5.0 as described in Appendix A. Then, you can use the procedure shown in figure 2-3 to set the JAVA\_HOME variable. However, the path for the JAVA\_HOME variable is likely to be longer. For example, by default, J2SE 5.0 is installed in a directory like this one:

```
set JAVA_HOME=C:\Program Files\Java\jdk1.5.0
```

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

```
rem JPDA_ADDRESS (Optional) Java runtime options used when the "jpda start"
rem command is executed. The default is "jdbconn".
rem
rem $Id: catalina.bat,v 1.11 2004/09/23 20:14:48 yoavs Exp $
rem -----
set JAVA_HOME=C:\Program Files\Java\jdk1.5.0
```

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: Enable servlet reloading

With Tomcat 5.5, the procedure for enabling servlet reloading has changed. To enable servlet reloading, you must open the context.xml file in Tomcat's conf directory and add a reloadable attribute to the Context element like this:

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

### Optional: Turn off serialization

By default, Tomcat 5.5 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 5.5, the invoker servlet is no longer mapped by default. Since all of the servlets presented in *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 can modify the web.xml file in Tomcat's conf directory. First, 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>
```

Then, 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");
    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.

## Conclusion

If you encounter any other issues when using *Murach's Java Servlets and JSP* with Tomcat 5.5 and J2SE 5.0, please send an email to [joelmurach@yahoo.com](mailto:joelmurach@yahoo.com) and I will update this document.

Thanks,

Joel Murach