

# How to use Tomcat 4.1 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 4.1. The changes presented here should work for both Java 1.4 and Java 5.0.

If you decide to use Java 5.0, you might be interested in the most recent edition of our beginning Java book, *Murach's Beginning Java 2, JDK 5*. This book shows you how to get the most out of Java 5.0, and we view it as an ideal companion book for *Murach's Java Servlets and JSP*. Of course, if you decide to use Java 1.4, you can still use the original companion book, *Murach's Beginning Java 2, SDK 1.4*.

## Chapter 2

### Figure 2-1: Download the 4.1.X ZIP file

In figure 2-1, make sure to download the ZIP file for Tomcat 4.1. 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

Before you set the JAVA\_HOME variable, you should install Java. For example, appendix A describes how to install Java 1.4. Then, you can use the procedure shown in figure 2-3 to set the JAVA\_HOME variable. However, you will need to enter the entire JAVA\_HOME variable like this:

```
set JAVA_HOME=C:\j2sdk1.4.0
```

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 1.30 2003/10/06 12:23:29 remm Exp $
rem  -----
set  JAVA_HOME=C:\j2sdk1.4.0
```

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

Note that the JAVA\_HOME variable shown above is typical for Java 1.4. However, if you're using Java 5.0, the path for the JDK is likely to be longer. For example, here is a typical statement that sets the JAVA\_HOME variable for Java 5.0:

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

## Figure 2-5: Enable servlet reloading

With Tomcat 4.1, the procedure for enabling servlet reloading has changed. To enable servlet reloading, `DefaultContext` element must include a closing tag like this:

```
<DefaultContext reloadable="true"></DefaultContext>
```

## Map the invoker servlet

With Tomcat 4.1, 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, if necessary, 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 XML 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.

## 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 4.1, please send an email to [joelmurach@yahoo.com](mailto:joelmurach@yahoo.com) and I will update this document.

Thanks,

Joel Murach