Installing and Running Tomcat 5.5

The examples for the Ajax chapter of *jQuery in Action* require the services of server-side resources in order to operate. In order to address the needs of a variety of readers, the back-end code has been provided in PHP and in JSP format. If you wish to use the JSP examples, it is necessary to set up an application server that is Servlets 2.4 and JSP 2.0 capable.

But not to fear! Not only has the downloaded example code for that chapter already been set up so that it is a ready-to-go web application (there's no need for you to build anything), the **free** and easily-obtained Tomcat 5.5 web server is a snap to set up. No knowledge of Java on your part is required to do so. This document will walk you through obtaining and setting up your Tomcat server.

This, of course, assumes that you have Java itself installed upon your system. If not, please visit the Sun site for details on installing Java 1.5 for your operating system. (This is also a not-too-arduous task).

Obtaining and Unpacking the Distribution

You can download the latest version of Tomcat 5.5 from the Apache Tomcat site at:

`http://tomcat.apache.org/download-55.cgi`

At the time that this document was written, version 5.5.23 was the latest stable build. You should probably download the most recent stable version if 5.5.23 has been superceded.

For Windows you'll want to grab the Core `.zip` file (avoid the Windows Service Installer – it does a much deeper installation than is needed for simply running the examples). For OS X and other UNIX systems, download the Core `.tar.gz` file. See figure 1:

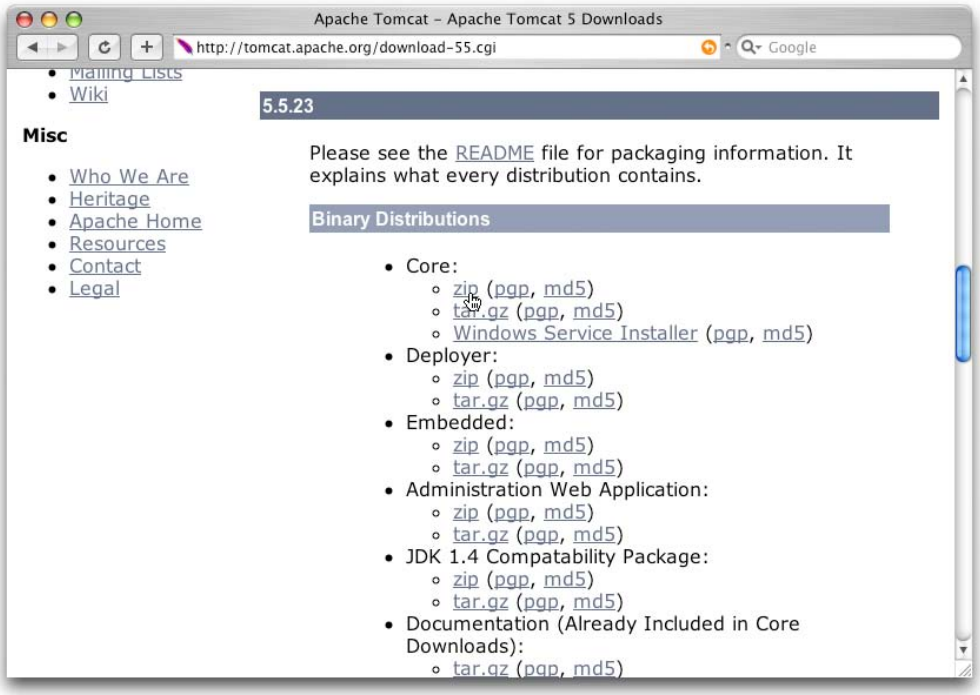

Figure 1: Dowloading the core Tomcat 5.5 distribution

Be sure to avoid the `pgp` and `md5` links – you don't need encrypted downloads.

Choose a location to unpack the `.zip` or `.tar` distribution.

On Windows, you probably want to avoid any folders with spaces in their names, so I usually unpack the distribution right to `C:\`.

On OS X, a typical location to place such installations is in the `/Library` folder. You can also choose the `/Applications` folder if you like, but many people like to reserve the `/Applications` folder for GUI applications.

On other UNIX systems, the typical location is the `/usr/local` folder. You could also use this folder on OS X, but `/Library` is more typically used on that platform as it makes the files accessible via the Finder.

Using the appropriate program (WinZip is a popular choice on Windows), unpack the folder to the desired location. On OS X, you should just be able to double-click on the downloaded `.tar` file.

This will create a folder hierarchy rooted at a folder named `apache-tomcat-5.5.23`. We will refer to this folder as `CATALINA_HOME` in the remainder of this section, and this is the environment variable name that Tomcat will use to refer to this location.

In case you were wondering, "Catalina" was the code name for Tomcat 4 and it just sort of stuck.

Setting up JAVA_HOME

In order to let Tomcat know where your Java implementation is located, you need to set up the `JAVA_HOME` environment variable. Depending upon how you installed Java previously, this might already be defined. If so, just skip along to the next section.

On Windows, if you're using Cygwin as your shell you can set up the environment variable as you would any other in your `.bash_profile` script. Otherwise, you'll need to use the Windows Control Panel to set up `JAVA_HOME` as a system-level environment variable (Control Panel -> System -> Advanced -> Environment Variables).

Set the value of the variable `JAVA_HOME` (case is important!) to point to the root folder of your Java installation. On my XP system, that folder is `C:\jdk1.5.0_06`. On my OS X box, its value is set to `/System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Home`.

On UNIX systems (to include OS X), set up the environment variable in the startup script for your particular shell (e.g. `.bash_profile` for the bash shell).

Setting up the Application Contexts

In J2EE parlance, a ***context*** is a self-contained web application. A single application server can load and service many such contexts, keeping requests to each context separate through the use of a unique prefix in the URL of each request. This prefix is known as the ***context path***.

When a context is defined to the server (we'll see how to do that in just a bit), it is assigned a unique context path and the location on the file system that serves as its "root" or ***document base.***

Let's say that one context is defined with a context path of `/abc` (context paths always start with a slash), and another with `/xyz`. If each has a file named `index.jsp` at their root folder (document base), the respective URLs would be:

http://someserver.com:8080/abc/index.jsp

```
http://someserver.com:8080/xyz/index.jsp
```

Note that the context path prefixes are used to let the server know which web application is to be accessed. These URLs also assume that the default Tomcat port of `8080` is being used.

Setting up the contexts is easy. All that you need to do is to create a small XML file that defines the context and drop it into the appropriate folder in the Tomcat installation.

Let's assume that you have downloaded and unpacked the code for this book onto your file system at `C:\jqueryinaction`. This folder is already set up to be the root (or document base) of a self-contained, working web application.

We'll assign the name `/jqia` (for JQuery in Action) to the application context for the book's web application. Note that the screen shots in the book all assume that this is the name of the context path. If you decide to use another name, you will need to adjust your URLs accordingly.

To define the context, create a file named after the context, `jqia.xml`, and within it place a single line (case counts):

```
<Context path="/jqia" docBase="c:\jqueryinaction"/>
```

The value of the `docBase` attribute is set as appropriate for wherever the expanded folders ended up on your file system. The above example assumes a Windows installation, of course.

Drop this XML file into the `$CATALINA_HOME/conf/Catalina/localhost` folder and you're good to go!

Starting Tomcat

Start the Tomcat server by executing a script that you will find in the `$CATALINA_HOME/bin` folder. For Windows use the `startup.bat` script, and for UNIX (including OS X) use `startup.sh`. When it comes time to shut down, you'll find the corresponding shutdown scripts in this same folder.

To make sure that Tomcat is up and running (after giving it a few seconds to get on its feet, of course), open a browser and enter the URL:

http://localhost:8080/

You should see a display as shown in figure 2.

If you do not see this page, go back and check your work. Unless you made a typo in the context files, or failed to set up the `JAVA_HOME` environment variable correctly, there's really no reason that Tomcat should not be up and running at this point.

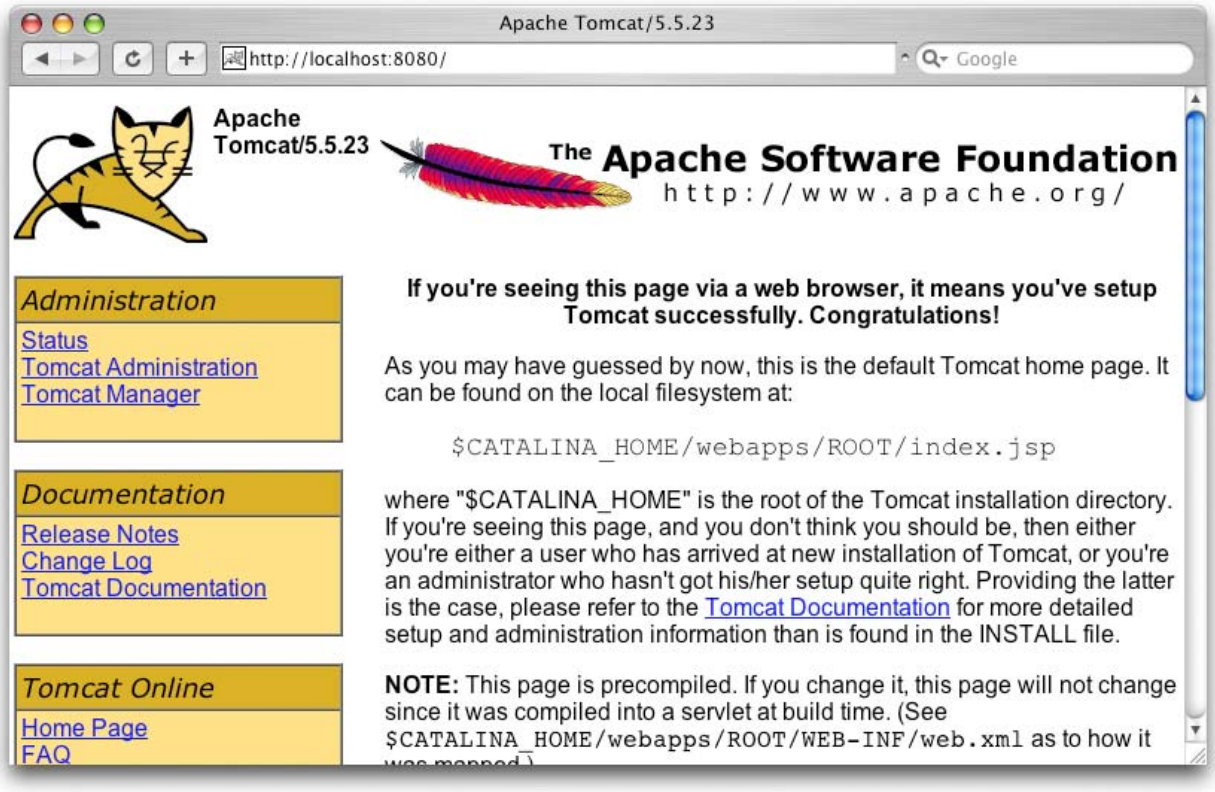

Figure 2: The Tomcat Welcome Page – success!

Once Tomcat is running, the URL to the book's web application will be:

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

The `:8080` in the URL specifies the port that Tomcat is running upon. It must be specified in the URL as shown otherwise the default of port 80 is used. You can change the port that Tomcat runs on if you'd like in the `$CATALINA_HOME/conf/server.xml` file, but it is recommended that you leave it at 8080 to avoid any conflicts with other servers (such as Apache) unless you have a really good reason to change it. Just remember to include the `:8080` in your URLs when using Tomcat to serve your files.

If all you want to do is run the example applications, that's all there is to it. If you plan on making changes to the example applications, or just want to know more about managing Tomcat, read on to the next section.

Managing Tomcat Contexts

If you are planning to make changes to the web applications supplied as examples to this book's Ajax chapter, especially if adding servlets, you may need to stop and restart contexts after making certain types of changes.

First, it is recommended that you do not change the applications as they have been provided. Rather, make a copy of the chapter code in another location on your file system and create *another* context for it. That way, you can make changes to your heart's content in the copy, but still have the original code to look back upon as a reference.

In the copy, if you make changes to the HTML files or JSP pages, you do not need to restart anything. Tomcat will detect any such changes and automatically handle serving up the new HTML files, or re-translating the JSP pages on your behalf.

However, if you make a change to the deployment descriptors (`web.xml`), or add, change and re-compile servlets, the web application context needs to be restarted to pick up those changes. The specific mechanics of compiling Java classes is beyond the scope of this document, but be sure to place the resulting class files in the proper location under the `WEB-INF/classes` folder for the context you are making changes within.

Note that you do not need to stop and restart Tomcat itself. Tomcat provides a built-in context management application that you can use to stop and start individual contexts without affecting the other contexts.

To access this "manager application", use your browser to hit the following URL:

```
http://localhost:8080/manager/html/
```

Oops! It wants you to log in as shown in figure 3:

Figure 3: Not just *anyone* can use the Manager Application

While it may seem a nuisance at the moment, this level of security is quite necessary. After all, you don't want just *anyone* to access the manager application, giving them the ability to stop and start your web applications, do you?

In order to gain access to the manager application, you will need to set up a privileged Tomcat user whose credentials you will use to log into the app.

It's actually quite simple: open the file `$CATALINA_HOME/conf/tomcat-users.xml` in any text editor. You will see the contents of the file as shown in listing 1:

Listing 1: The initial tomcat-users.xml file

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
  <user username="role1" password="tomcat" roles="role1"/>
</tomcat-users>
```

To this file, add the line:

```
  <role rolename="manager"/>
```

to the end of the role elements, and add the following line to the end of the user elements, substituting a username and password of your choosing.

```
  <user username="wallace" password="gromit" roles="manager"/>
```

Save the file.

Shutdown and restart Tomcat, and when it's back up and running, hit the manager URL again. Enter the username and password you specified in the users file when prompted. After logging in, you will see the manager application as shown in figure 4.

With this application, you can easily and quickly stop and start individual application contexts (via the `Stop` and `Start` links for each) whenever you make a change requiring a restart.

Note the entry for the application context that we set up for the `jqia` application. You will see one of these entries for any application contexts that you set up in this manner.

Now we're ready to dig into those examples!

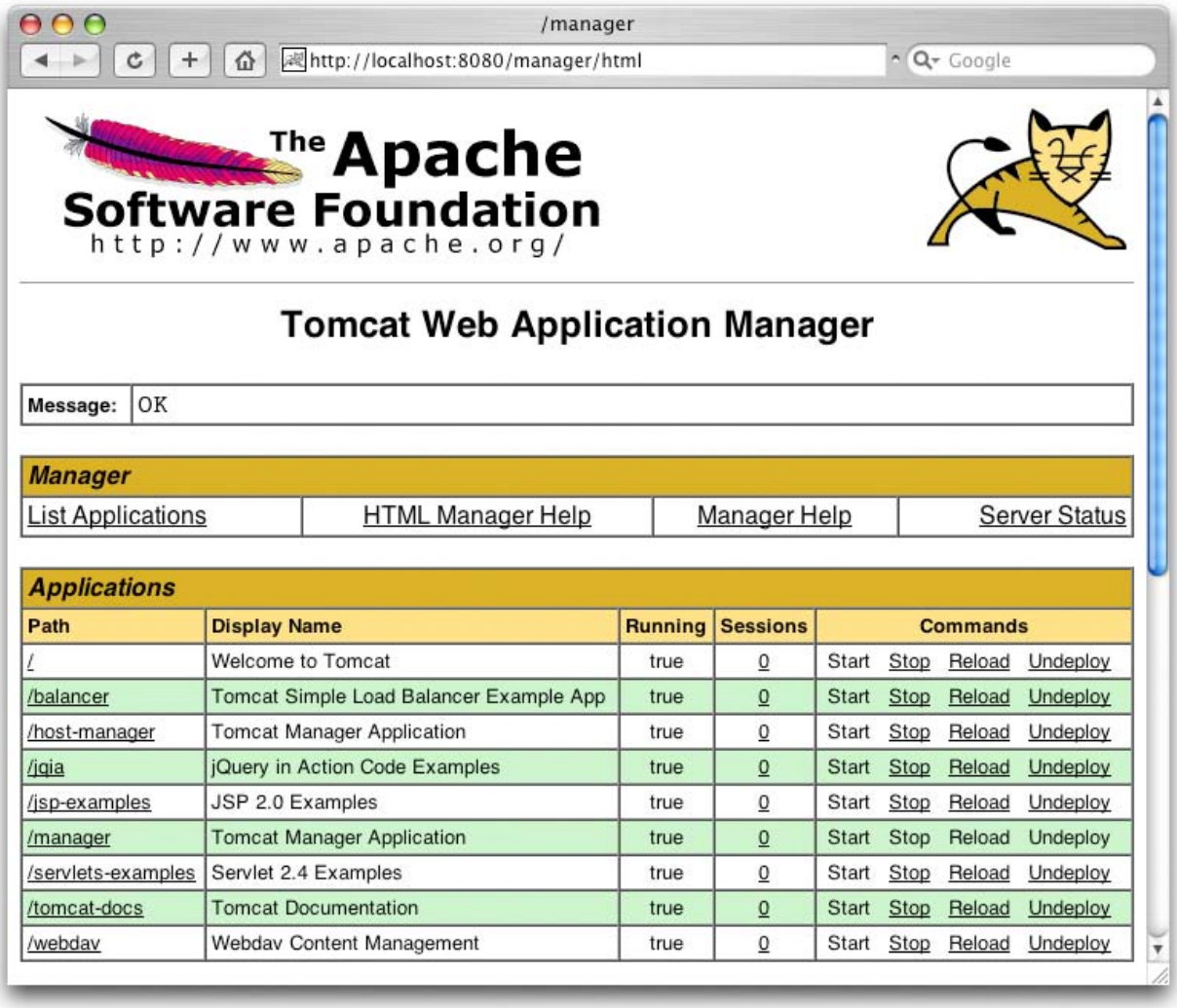

Figure 4: The Tomcat Context Manager Application