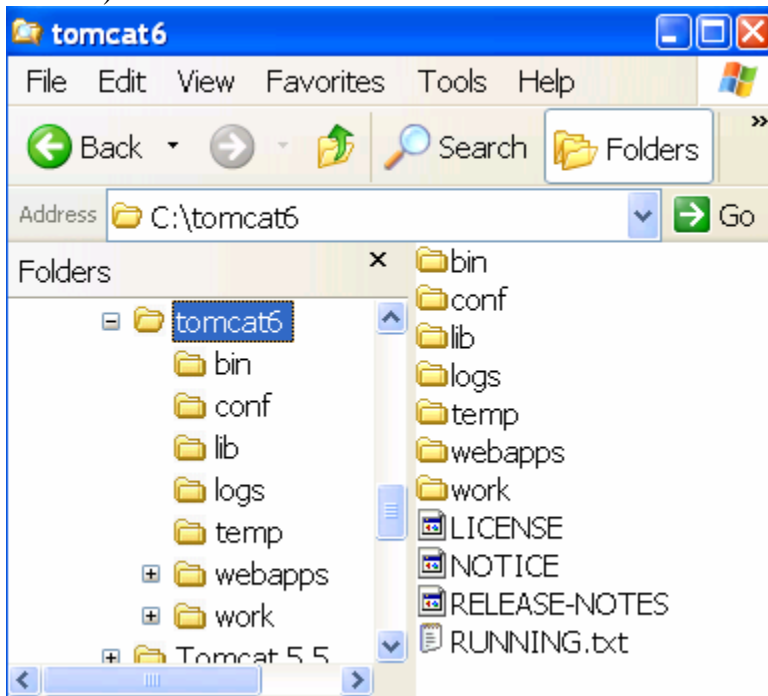


CSCI 4208 Advanced Web Applications. Spring 2008
Notes on Deploying Servlets
Last Update: Jan 27, 2008. 5:30 PM

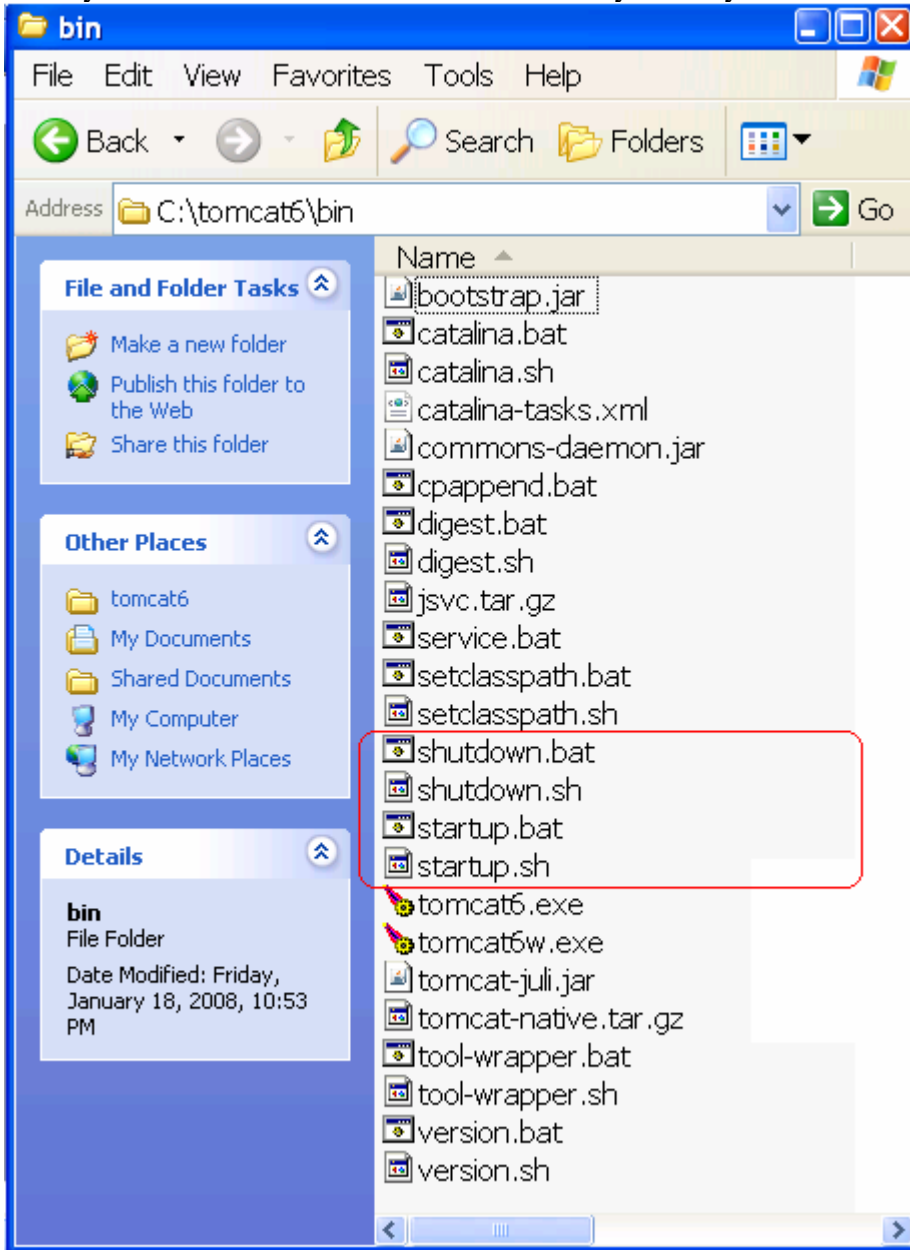
- JEE (Java Enterprise Edition) specification is a compilation of various Java APIs (such as Java Servlets, JSP, JDBC, etc.).
- An important concept in the specification is *Web Application*.
- All resources (e.g., Servlets, static HTML, image files, JavaServer Pages (JSPs), etc.) belonging to a Web Application are organized under a standard directory structure.
- Tomcat is compliant with JEE specifications.
- When you develop a Web Application and want to deploy it using Tomcat you must structure your directories to follow JEE specification. I will describe that structure in these notes.
- NOTE: In this document Install Instructions refers to HW 0.

What to do after installing Tomcat?

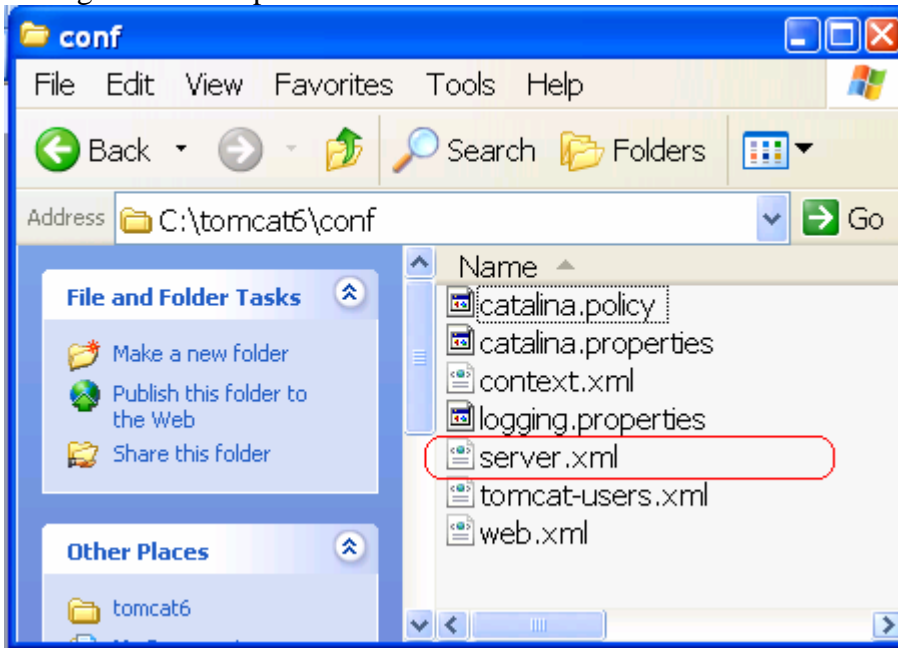
Let us see the directory structure in Tomcat. The figures below are from the install on my laptop. The structure is exactly the same on Solaris. On my laptop, Tomcat is installed in: *C:\tomcat6*. Here are the contents of this directory. For now we will only be concerned with *bin*, *conf*, *lib* and *webapps* directories (though *logs* directory will also come in handy later on).



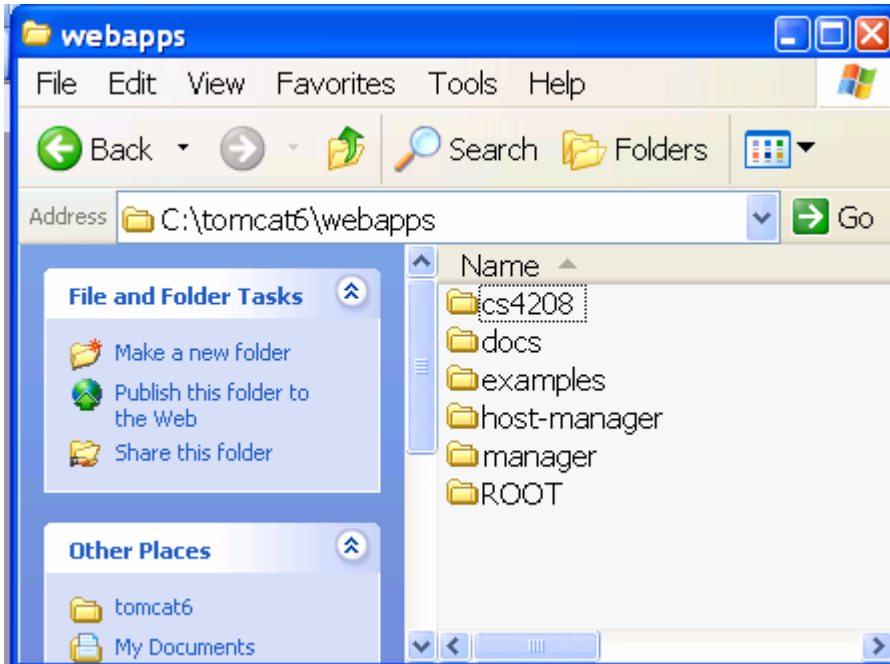
Let us look at the *bin* directory. Tomcat *startup* and *shutdown* scripts are located in this directory, with *.sh* extension for Unix and *.bat* extension for Windows. These scripts require that the environment variable **JAVA_HOME** be set. In the Install Instructions, recall you had to set this variable to the directory where you install JDK 6.



The *conf* directory has configuration files for Tomcat. Currently, the only thing there of interest to us is the file *server.xml*. The port on which Tomcat listens for HTTP Requests is specified in this file. Should you need to modify the port number you can do this by editing this file as specified in the Install Instructions.

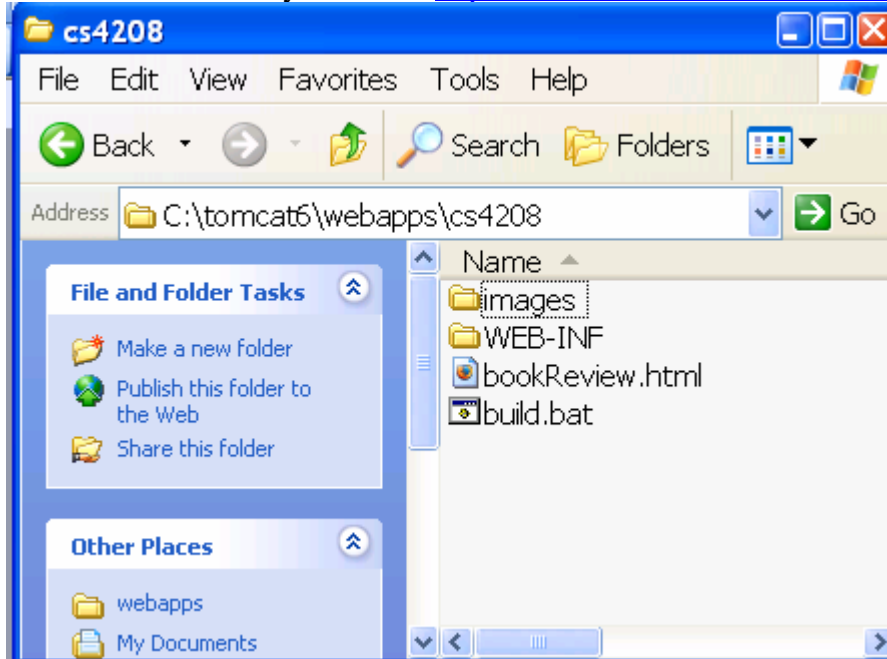


The *webapps* directory is very important to us. Any web application you implement and want to deploy can be placed in a subdirectory of *webapps*. Let us look at the contents of *webapps*. We see that a number of Web Applications appear under *webapps*. In the Install Instructions I had asked that you go to some URLs that had *examples* in the resource part of the URL. These URLs belong to the Web Application *examples* which is an example Web Application included with Tomcat.



To deploy my own servlets I created a directory *cs4208* under *webapps*. This directory will be used for the Web Application to be implemented during this course. Of course, this application must have the required directory structure.

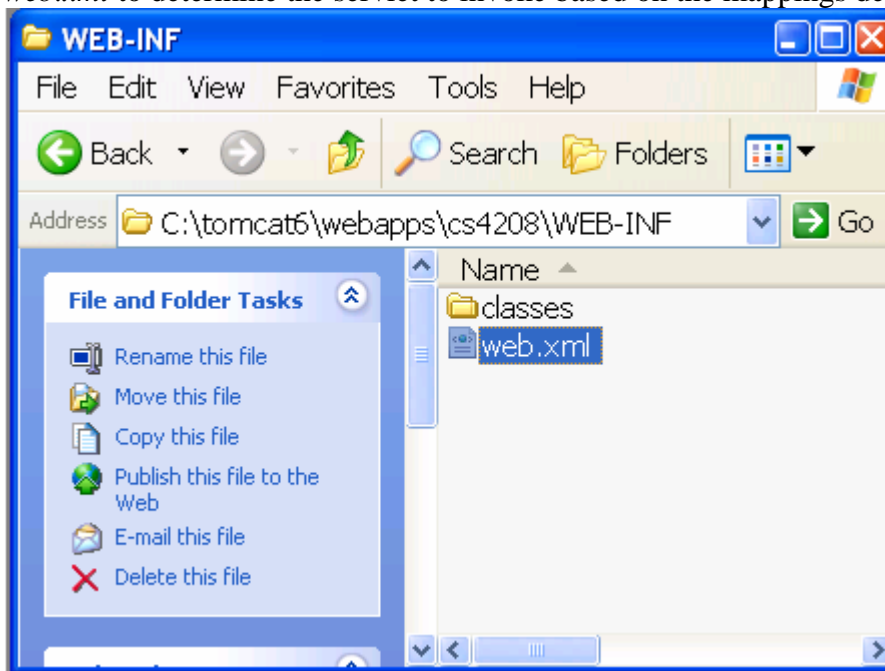
Let us examine the *cs4208* directory. In this directory we find two directories, one script (*build.bat*) and one static HTML file, *bookReview.html*. Recall that a Web Application may contain **static HTML files**. You can place such static HTML files within the top level directory of the Web Application or some sub-folder under it. For our Web Application *cs4208*, I placed *bookReview.html* right there in *cs4208* directory. Now this file can be accessed by the URL: <http://localhost:8080/cs4208/bookReview.html>.



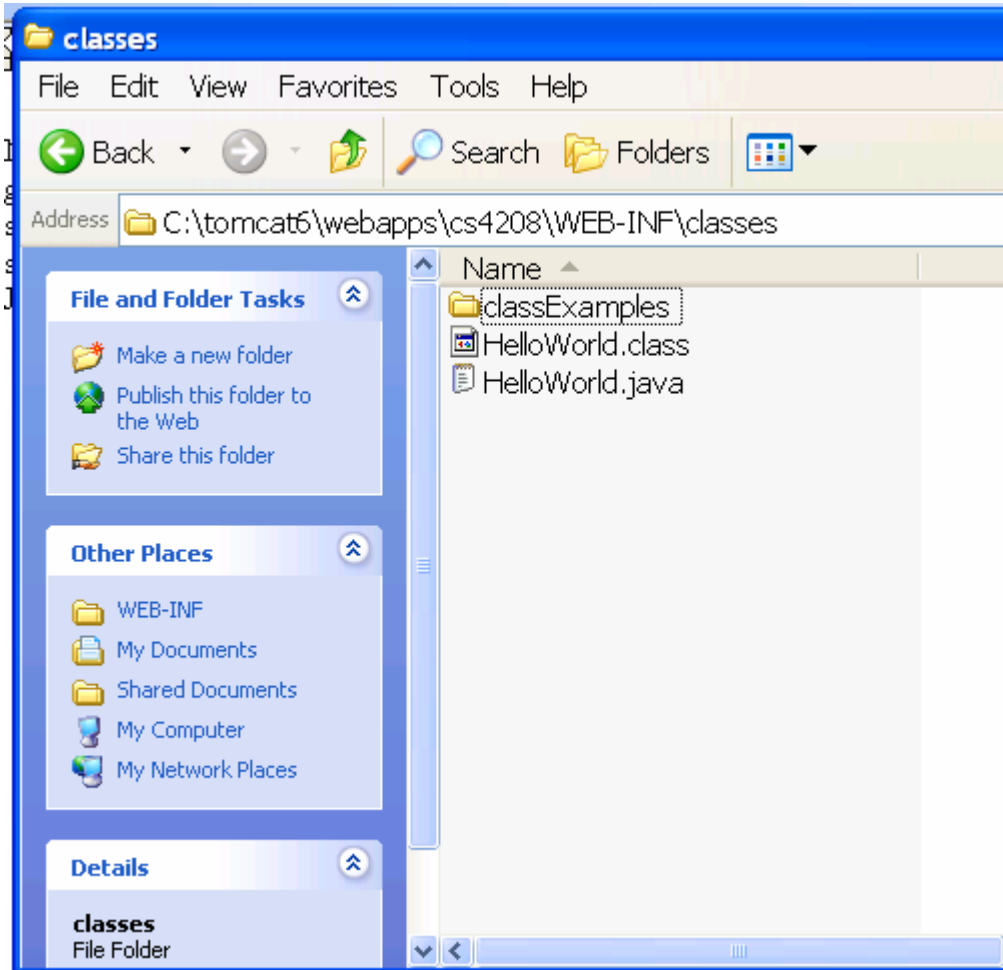
The file build.bat is NOT part of the Web Application. I created this script to compile the Java Servlets that I had added in *cs4208* Web Application. If you find this script helpful in your development, augment this script to add in your Java files and run the script to compile them. Or you can come up with your own mechanism.

The directory *images* in *cs4208* contains image files used by this Web Application. Let us now look at the contents of the directory *WEB-INF*. This directory contains a file *web.xml* and a directory *classes*. The file *web.xml* is called the Application Deployment Descriptor. Each Web Application will have its own deployment descriptor file that lists configuration parameters for that Web Application. E.g., if you look at the Web Application *examples*, under *examples /WEB-INF* you will see *web.xml* for the *examples* Web Application.

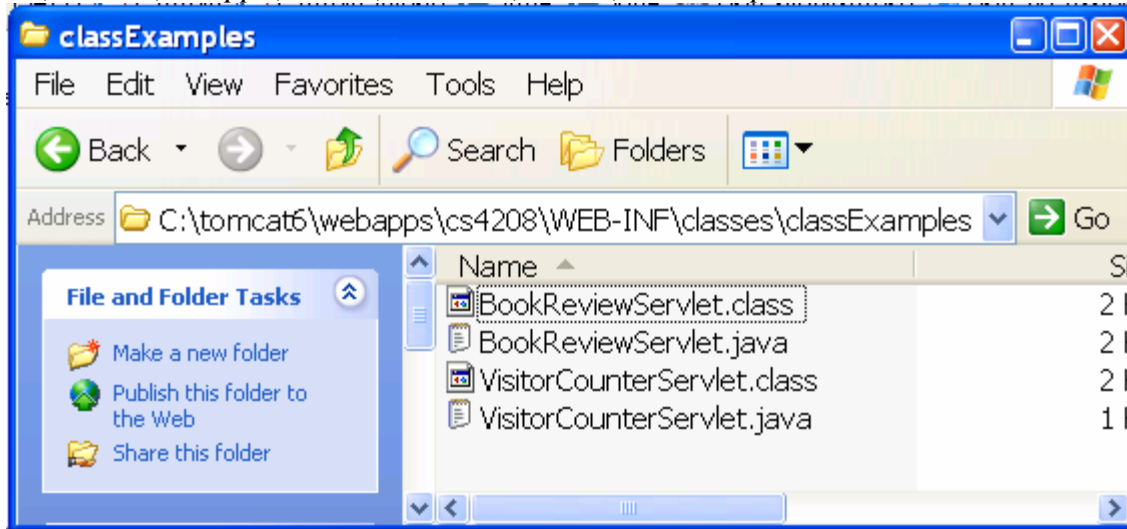
In the class slides I had given example of contents of *web.xml* for *cs4208* web application. In particular, if you create a new servlet in the *cs4208* Web Application, you will need to add servlet name and mapping in the *web.xml* file for *cs4208*. When Tomcat is processing a URL that maps to the Web Application *cs4208*, it will make use of *web.xml* to determine the servlet to invoke based on the mappings defined in this file.



Now let us look at the contents of the directory *classes*. The servlets you compile need to go into this directory. In the contents we also see the Java file for the servlet. The build script I used expects the Java files to be placed in the directories where the class files should be placed. It is **not** required that you put the Java files here. You can compile your Java files anywhere and then place the *.class* files in the correct directory.



Lastly, if your Java class is inside a package, you should replicate the directory structure of the package within the *WEB-INF/classes* directory and place your *.class* files under the correct package directory. For example, in *cs4208*, the servlet *BookReviewServlet* is in the package *classExamples*. So we see that *BookReviewServlet.class* file is placed under: *C:\tomcat6\webapps\cs4208\WEB-INF\classes\classExamples*.



Writing your own code?

I have uploaded a ZIP file called *cs4208.zip* with the contents of my *cs4208* directory. One way to proceed is for you to simply unzip this file under *\$CATALINA_HOME\webapps*.

Alternately you can first manually create *cs4208* directory under *webapps* and also create *cs4208\WEB-INF* and *cs4208\WEB-INF\classes* directories. Create *web.xml* file under the *classes* directory. Each *web.xml* file is required to contain certain XML elements. It might be better for you to start with the *web.xml* file from my ZIP file.

To add new code you will need to do the following steps:

- *Design and implement your servlet and associated objects*
- *Compile servlet and the objects*
 - *When compiling servlets classpath must include:*
\$CATALINA_HOME\lib\servlet-api.jar
- *Integrate with servlet container (Tomcat)*
 - *Place .class file in right spot*
 - *Modify web.xml*
- *Bounce Tomcat (i.e., shutdown and restart if it is already running)*
- *Test servlet by requesting the right URL*

Troubleshooting

If you cannot access any servlets in your web application, check for the following:

1. Is Tomcat running? Go to <http://localhost:8080> and see if the Tomcat top-level webpage shows up. . If it doesn't work, perhaps your **JAVA_HOME** isn't set correctly.
2. Are any Servlets running? From the Tomcat top-level webpage, click on Servlet Examples on the left side of the page. Now execute one of the servlets to see if it works.
3. Is there a problem with the *web.xml* file in your web application? This is a very common source of error. Your *web.xml* file's format may be incorrect, e.g., missing declarations or misspelled names.
4. Are the *.class* files for your servlet and associated code in the correct directories?