Assignment #8a

Setting up your own tomcat Server on mislab

mislab runs the apache web server software. apache is open-source software, available for free from www.apache.org. Over 60% of the web servers on the internet run the apache web server software.

apache does not support Java servlets by itself. To provide that support, mislab runs the tomcat server software (this is also open-source, free software). For class purposes, each student will run his or her own tomcat server. The following instructions explain how to get your tomcat server up and running.

1. First, check your grades for this class, and look for your Assigned Tomcat Port Numbers. You will have two assigned port numbers: a primary port number, and a secondary port number. Write these numbers down for later use.

2. Next, you need to edit one of your system startup files on your mislab account. "Hidden" files are not shown when you get a normal directory listing, such as with the command "ls -l". To show "all" files, use the command:

`ls -al`

Telnet to your account, and get a listing of all files in your root directory (ls -al). You should see several files whose filenames begin with a period. These are hidden files; they are not normally shown when you get a directory listing (unless you include the "a" switch, of course). You need to edit the file named .profile.

`pico .profile`

Your .profile file contains commands that you want to have executed as part of your login process. For tomcat programming, you need to set the values of four environment variables. You can do this with the export command.

You should see the following four lines in your .profile:

```
export JAVA_HOME=/usr/local/jdk

export CATALINA_HOME=/usr/local/tomcat

export CATALINA_BASE=/home/bis3523/abc123/tomcat

export CLASSPATH=.:

/usr/local/tomcat/common/lib/servlet.jar:

/usr/local/jdk/bin/mysql.jar
```

Modify the third of these lines to substitute your netid for abc123. Also make sure that your entire path specification is correct (make sure it is /home/bis3523, for instance).

The commands that you type into your .profile are exactly like commands that you would type at the command line. At the command line, you could not press <Enter> in the middle of a command. Similarly, you cannot split any of these commands across multiple lines.

Be sure to check your .profile for accuracy. Each command must be on a single line.

These environment variables specify (1) the location of mislab's Java installation, (2) mislab's tomcat (catalina) installation, (3) your tomcat directory, and (4) locations (directories or files) to be searched for Java classes.

3. You can create your own aliases for commands in your .profile file. For instance, if you want to type "dir" instead of "ls –l", you can include the following line in your .profile:

```
alias dir="ls –l"
```

For your convenience, an alias to start your tomcat server, one to shutdown your tomcat server, and one to show all running tomcat servers have been created in your .profile file.

```
alias starttom="/usr/local/tomcat/bin/startup.sh"

alias stoptom="/usr/local/tomcat/bin/shutdown.sh"

alias showtom="ps -f -C java"
```

You will be able to type starttom to start your tomcat server, stoptom to shut it down, and showtom to show all running tomcat servers.

4. Press Control-X to exit pico, saving your file.

5. Log out of your account, then log back in. The lines that you just inserted into your .profile file will be processed when you log back in (and every time you log in in the future).

Any time that you edit your .profile, you need to log out, then log back in. The commands in your .profile are processed as part of the login process.

6. Under your root directory, make a subdirectory named tomcat:

`mkdir tomcat`

7. Change to your tomcat subdirectory:

`cd tomcat`

8. Copy all of the tomcat files from mislab's tomcat directory to your own tomcat subdirectory:

`cp -R /usr/local/tomcat/* .`

(The -R switch indicates that all subdirectories should be copied, recursively. The period at the end specifies that the current directory is the destination to which files are to be copied.)

9. Get a directory listing. You should see several subdirectories, including: conf, logs, webapps, and work. Change to your conf subdirectory.

`cd conf`

10. The file server.xml is used to configure your tomcat server. You need to edit this file to specify your assigned port numbers.

`pico server.xml`

(If you do not want to use pico, you could ftp server.xml down to your local computer, edit the file, then ftp it back to your mislab account.)

On line 3, change `port=8085` to use your assigned secondary port number. (You can press Control-C to see the current line number in pico.)

On line 15, change `port=8080` to use your assigned primary port number.

11. Exit pico, saving your edited file.

12. Start your tomcat server:

`starttom`

You should get a message indicating the directories that your server is using. These should correspond to the environment variables that you set in your .profile.

13. Check to see if your tomcat server is running:

`showtom`

```
conf> showtom
UID PID PPID C STIME TTY TIME CMD
rpearson 24611 1 13 11:00 pts/2 00:00:04 /usr/local/jdk/bin/java -Djava.u
```

You should see something like:

(Note: There could be multiple tomcat servers running, since each student will run his or her own server.)

14. Get into your web browser and try to access your server. Go to the following URL:

http://mislab.business.msstate.edu:8080

(Replace the 8080 with your assigned primary port number.)

This should load the index.jsp file that is stored in your tomcat/webapps/ROOT directory. (You may have to try this more that once, since it takes a few moments for your tomcat server to actually get up and running. Its first response to a request may be slow, but subsequent responses will be fast.)

15. You should have the file counter.jsp in your tomcat/webapps/ROOT subdirectory. The .jsp filename extension indicates to the tomcat server that this is a java server page, meaning that it includes jsp code. Given your knowledge of Java, you can probably look at counter.jsp and determine to some extent what is going on in the page. When your tomcat server gets a request for this file, it (the server) translates the .jsp file into an actual java servlet. To see this in action, telnet to your account, and move to tomcat/work/Catalina/localhost/ROOT/org/apach/jsp. You will probably have two files in that directory: index_jsp.java and index_jsp.class. These two files were created when you accessed your index.jsp page.

16. In your browser, go to:

http://mislab.business.msstate.edu:8080/counter.jsp

(Replace the 8080 with your assigned primary port number.)

You should see a page which displays the current date and time, followed by an access counter.

17. Now go back to your telnet session and get a new directory listing. You will see that your tomcat server translated your .jsp file into a .java file (counter_jsp.java), then compiled that into a .class file. Your .class file generated the page that you see in the browser!

18. The first time you access counter.jsp with your browser, there will be a significant delay, because the .jsp file has to be translated into .java, then the .java file has to be compiled into byte code. On subsequent requests for the page, these steps can be skipped, so the desired page will load much more quickly.

Note that any time that you shut down your server, or any time that you edit the .jsp file, Java code generation and compilation will need to occur again. One noticeable effect of this is that the next request for the page will, once again, be slow. Another noticeable effect is that your counter will be reinitialized. This counter will count correctly, as long as you don't shut down your tomcat server and don't edit your .jsp file.

19. If you want to do some quick experimentation, you can put your own .html files in your tomcat/webapps/ROOT directory and access them via your own assigned tomcat primary port number.

20. Now that you know that you can process a .jsp correctly, test a servlet. We want to make sure that you can compile a servlet, and that you can then run it from a web page.

You should have the following two files:

`tomcat/webapps/ROOT/snooperServlet.html`

`tomcat/webapps/ROOT/WEB-INF/classes/SnooperServlet.java`

Edit SnoooperServlet.html and change the default port number of 8080 to your primary tomcat port number (this is in the href attribute of an anchor tag).

Also change the abc123 that you find in that attribute to your net id.

Move to your "classes directory". If you are currently in your home directory, you can use one, or several, cd commands to move to your classes directory, such as:

`cd tomcat/webapps/ROOT/WEB-INF/classes`

or

`cd tomcat`

`cd webapps`

`cd ROOT`

`cd WEB-INF`

`cd classes`

21. Get a directory listing, to be sure that you really have SnooperServlet.java.

`ls -l`

22. Compile SnooperServlet.java.

`javac SnooperServlet.java`

If you get an error message such as:

`-bash: javac: command not found`

then you have an error in the Path environment variable in your .profile. Since your .profile was pre-configured for you, you should not get this error message. If you have accidentally messed up your path, however, you could get this error. Your .profile should contain the following line:

`PATH=$PATH:$HOME/bin:/usr/sbin:/usr/local/jdk/bin`

Since java is in /usr/local/jdk/bin on mislab, this value should point the operating system to the correct file.

If you get an error message such as:

`SnooperServlet.java:10: package javax.servlet does not exist`

or

`SnooperServlet.java:13: cannot resolve symbol`

then you probably have an error in your classpath (also set in your .profile, but you typed this one).

23. Once you have worked the bugs out of your environment, you will successfully compile SnooperServlet.java. Get a directory listing, and make sure that you have a SnooperServlet.class.

24. In your web browser, go to:

http://mislab.business.msstate.edu:8080/snooperServlet.html

Click the anchor link. If you see a display of a lot of information, formatted in a table, you are doing great!

25. When you are finished, you can shut down your tomcat server, or you can leave it running to wait for requests.

`stoptom`

Revision #1
Created 9 January 2019 23:41:44 by Aaron Kimbrell
Updated 10 January 2019 16:40:22 by Aaron Kimbrell