Introduction

The purpose of this homework is to get you acquainted with some of the development tools that we will be using this quarter and have you write your first servlet to run on a web server.

We have provided you with two sample servlets in the HW2 download. The first is LittleCat, which displays the same cat image that we used for HW1, plus some other info related to the http request. The second servlet is Snapper. Snapper is very similar to LittleCat, and mostly illustrates how you build more than one servlet into a web application. You can use these servlets to get started and make sure you understand the steps involved in building, installing, and running a web application.

Your task is to implement a third servlet as described below and add it to the application.

Preparation:

1. If you are working on a non-lab machine, install the course software as described on the class web site software page. If you are on a lab machine, verify that the various parts and pieces are available once you have logged on. I *think* that everything we need is installed and running. However if there are any problems let us know right away.

2. One package that you need to install on non-lab machines that we did not need for HW1 is the set of jar files that include the servlet-related class libraries. These files are provided to you in tomcat4-jars.zip, available from the software page. You can unzip the archive and put the contents anywhere on your development system. On my systems they are at C:/apps/tomcat4. On the lab Unix systems they are at /cse/courses/cse403/03wi/tomcat4, and on the lab Windows systems they are at O:/cse/courses/cse403/03wi/tomcat4, assuming that \\ntdfs\cs is mapped to the O drive. Note the use of forward slashes, Ant translates them.

3. Download the zip file for this homework from the web site. Unzip it to your development directory. The structure of the directory is dictated by the fact that this is a servlet and will be running under the Tomcat server application on cubist.

4. Whatever the appropriate path to the Tomcat directory (step 2) is for your system should be entered in the build.properties file in the HW2 directory as the value of property catalina.home.

5. The Tomcat server documentation is at address

http://cubist.cs.washington.edu:8080/tomcat-docs/appdev/index.html

Various useful links are available from that page, including the Application Developer's Guide mentioned below.

6. You have an account on cubist.cs.washington.edu. Ant is installed on the system and you can add it to your path by adding `set path = ( $path /usr/local/jakarta-ant/bin)` to the file `.cshrc` in your home directory.

Description of the directories

The directory structure is described in somewhat more detail in the Application Developer's Guide and in great and complete detail in Chapter 9 of the Servlet API Specification, version 2.3. (There is a link on the software page).

HW2/src

Java source code to be compiled to the WEB-INF/classes subdirectory.

HW2/web

Static content (such as image files, HTML, JSP), and the WEB-INF subdirectory with its configuration file contents.

HW2/docs

Documentation for the application other than the generated javadoc.

HW2/build

The directory into which the "prepare" and "compile" targets will generate their output. Reconstructed completely from scratch, so no changes should be made to the contents.

HW2/dist

The directory in which distribution files are created. Reconstructed completely from scratch, so no changes should be made to the contents. The .war file in this directory is the application file that you copy to cubist for installation.

HW2/server

This directory is not part of the servlet specification. I put the server-side Ant file manager.xml and the build.properties file in this directory. You should copy these files to your account on cubist as described later.

Description of the files

build.xml and build.properties in HW2/

These are the Ant files that you use to compile and build your application. The target “dist” will clean and build the entire application archive file and documentation. The resulting application file is the .war file in the dist directory.

You should update the build.properties file so that catalina.home points to the tomcat4 directory on your system, as described in the preparation section above.

Note that the app.version is hardcoded in build.properties. If you have extra time, feel free to make this dynamic to track build numbers or cvs releases or whatever. If you do so, you’ll need to make sure that the matching installation script (server/manager.xml and server/build.properties) can figure out what to do.

web.xml in web/WEB-INF/

This is the file that defines the mapping between classes, servlets, and URLs. Note how the two examples are mapped. You will update this file to include your third servlet.

LittleCat.java and Snapper.java in HW2/src/appmain/

These are the two example servlets provided to you. They are part of package appmain. You will create a third servlet and the source code will go in this same package. My copies of these servlets are installed and running.

http://cubist.cs.washington.edu:8080/djohnson/meow
http://cubist.cs.washington.edu:8080/djohnson/snap

jkf-cat.png in web/images/

This is the cat image that is displayed by LittleCat. You can place other image files here if you extend the basic applications. Note that the script build.xml names the images directory explicitly, so if you add more directories you’ll need to modify the script to add those names or find them automatically.

manager.xml and build.properties in HW2/server/

These are components of the Ant build file that installs and removes your application on cubist. The two files should be copied to a directory HW2 on your cubist account. build.properties has the same hardcoded assumptions that the development build script has about the name and version of this application. It assumes that the web application archive file (.war) is available in the HW2/dist subdirectory when the install target is run.

The script manager.xml is run on cubist. It must be run on the same system where the Tomcat server is running. The `install` target tells Tomcat about a new copy of your application, and the `remove` target uninstalls it. Your application is installed on the path /<username>, with the path to the specific servlet defined in the web.xml file.

```
[djohnson@cubist ~/HW2]$ ant -buildfile manager.xml install
[djohnson@cubist ~/HW2]$ ant -buildfile manager.xml remove
```

The `list` target shows all the web applications that are installed. Note that for this assignment, you have shared access to the manager functions of Tomcat. Please don't mess with other people's applications!

Assignment

Implement the Razor servlet and add it to the web application. This servlet is *very* simple. It takes a URL with an index parameter and selects from a list of razor manufacturers and returns that string. There is a running copy of this servlet on cubist at

http://cubist.cs.washington.edu:8080/djohnson/device?index=3

The goal of writing this servlet is that you get some experience with the various aspects of the installation process and also that you get some exposure to the servlet lifecycle and the how to get information in and out of a servlet.

1. The main class of the application is Razor in package appmain, similar to LittleCat and Snapper.

2. An easy way to write this servlet is to duplicate Snapper.java and modify it to define the Razor class, with no change to the actual function. Modify web.xml to include the new servlet, then build and install the new application file. If all goes well, you will then have a new servlet running and you can just fiddle with the java code to get the desired functionality.

3. The actual code in Razor just selects from a list of Strings, depending on the index value it is given and returns that String as a response. The mime type of the response should be text/plain, not text/html, because we are looking ahead to our cell phone applications that will transmit as little data as possible.

4. This is the list of manufacturers that I used. You can use whatever you like.

```
private String[] devices = {
        "Gillette","Schick","Wilkinson","Remington", "Sheffield","Dubl-Duck",
        "Lifetime", "Napoleon", "Beau Brummel", "Deutsch", "Satinedge"
};
```

5. Read the servlet tutorial for information about the lifecycle of a servlet and how to write the service methods that implement a servlet. Refer to the servlet API documentation for details of the various classes involved, particularly for package javax.servlet.http. Links are on the software page.

6. Write a brief release-notes.txt file and store that in the docs directory of the application. In the release notes, include a brief description of the features of your servlet, especially if you did something fancy in addition to the basic task requested here. Describe any known bugs or limitations.

7. As with homework 1, the intent is to get you some experience writing and building a part of a web service application. As long as your servlet runs and responds correctly to inputs, it will be accepted. Again, spend the time now to figure out how the parts relate to one another, because in a few short weeks we will be making design decisions about a larger application and you will want to have a good understanding of how it all works then.

Turn in

Install your application on cubist and make sure that it is running correctly. If you have made any modifications to the two files in the server directory, make sure you copy them back from cubist into your server subdirectory before you make the zip file below.

Build a zip file that contains all the files in your directories (and maintains the directory structure) and contains your uwnetid in the file name. Using Ant, the command

```
ant -Duwnetid=youruwnetid archive
```

will build the archive in a zips directory at the same level as HW2. Verify that the zip file does indeed have all your files in it, then follow the turnin link on the web site and turn in the archive.

This homework is due before midnight, Tuesday January 21. Don’t be late, the turnin server closes automatically!