SER422 Spring 2014 Lab3 (worth a full lab grade)  
Assigned 2/10/14, due 2/20/14 at 11:55pm via submission to Moodle

In this lab you are given 2 tasks. The first asks you to code a simple Java servlet. The second asks you to add simple stateful behavior using cookies and/or hidden variables.

**Task 1: Simple servlet functionality (50%)**

Write a complete web application that does the following:

1. Provides a web form allowing a user to input a first/last name, languages the person knows how to program in, days of the week the person can meet, and at least one additional descriptive attribute you decide to add.
2. The web form must POST to a URL. The POST should be handled by your servlet, which will receive the information and persist it to a file accessible by your web context. The browser should display a message stating the transaction was successful (or not) followed by a count of the number of entries in the file, and a hyperlink back to the web form page (is there a way not to hardcode this?).
3. The servlet must accept a GET request that lists all the entries in the file. If the browser is a mobile browser then the font size should be set to 8 point font. Otherwise, use 12-point font and a pink background color.
   1. Further, the GET should take one or more query parameters that filter results by a substring of the attribute named in the query string. Example:
      1. “GET /<context>/<servlet>?firstname=ob&languages=scala+lisp&days=MW” on this file:
         1. “Bob Smith, java scala c, Monday Thursday”
         2. “Rob Roy, scala ada, Wednesday Friday”
         3. “Sue Sink, lisp C#, Monday Wednesday
   2. Would return 1 and 2 but not number 3. Note that the parameters of different attributes are AND’d together, while within a multi-valued attribute it is an OR.
4. Make certain that the results of GETs and POSTs are not cached by the browser.

*Constraints*:

1. The web application should use a single servlet.
2. The HTML generated does not have to be fancy, just readable on the browser. This lab is not about aesthetics, it is about learning servlet structure.
3. You do not have to solve the concurrency control problem of multiple accesses at the same time to your file, but if you are interested in the challenge see the extra credit below.
4. I do not care how you persist the data, except that I want it to be some kind of local file. The file format is up to you. Remember not to hardcode directory paths!

**Task 2: Add stateful behavior (50%)**

Task 2 adds 3 new features to Task 1:

*Create a landing page that remembers who you are*

Create a new landing (home) page that recognizes the user if the user has previously been to the site. If the user has not previously been to the site (or not registered) then prompt the user to complete the workflow, and then remember the user’s first and last name the next time s/he visits the site. If the user did previously register, then the landing page should display a sorted list of the top 3 matches that are closest to her/his preferences for each of the attributes above in Task 1.

*Make a multi-page web form*

Split the webform in Task 1 #1 into 4 distinct screens with a submit button at the end. Give the user “Back” and “Next” hyperlinks to navigate backward and forward through the webform, and pre-populate any HTML widgets with previously entered values. If the user submits or cancels on the last page make sure the “remembered” widget values are cleared. Upon submit or cancel the user should return to the landing page.

*Return the proper response error codes*

For Task 2 you must create multiple servlets as one web application. Furthermore, you should still support the GET list functionality from Task 1. However, Task 1 did not ask you to verify the query parameters. If an invalid set of query parameters is provided, inform the browser by returning the proper response code. Note this is NOT the same thing as writing an error message to the browser window! Also detect if a servlet receives an HTTP message type it should not receive according to these specifications, and again return the proper response code.

*Constraints:*

* + - 1. You must use a combination of browser cookies and hidden variables to complete this functionality, but it is up to you to decide where each make the most sense.
      2. This task must use multiple servlets, not a single servlet. Again, it is up to you to determine the best design.
      3. Constraints 2-4 from Task 1 still apply.

**Extra Credit (25 points):**

Can you come up with a way to solve the concurrent access to the file problem? Note that the operating system is going to block if you attempt to open the file with write permissions when another thread already has it open (for read or write). Is that a good thing? Will it ensure integrity of the file? What happens if your thread blocks for so long the Tomcat request times out?

**Hints:**

* The PhoneServlet example should be a lot of help to you
* Draw a statechart of the page flow for Task 2 so you are sure of the expected navigation, and to understand how information flows through that page flow.
* Avoid hardcoding constant values or configuration information anywhere in your web apps.
* Avoid absolute paths to resources!
* Yes, the last feature of Task 2 will require you to research in more depth the nature of response codes.

**Submission:**

You will complete this lab in pairs. I recommend one of you create a github repo and add your partner, myself (cst316), and Vasu Gupta (meetvasu15) as collaborators.

For submission, you should submit to Moodle a single zipfile named lab3\_<asurite1>\_<asurite2>.zip (or .jar) that has within it 2 WAR files (one for each application), and 2 source trees.

* The web application context for the first web application should be lab3task1 and the 2nd lab3task2.
* WAR files should not have any source (.java) files in them.
* Source trees are the directory contents of your development directory without the compiled artifacts (no “classes” or “bin” subdirectories) and no WAR files. All of your \*.java, properties, and static resources should be in this file.
* I expect a build.xml to be provided with targets “compile”, “build”, “dist”, “clean” and “deploy” (at a minimum). You should be able to customize mine quiet easily.

NOTE: A reminder that I allow you to work in pairs so you can work together to understand web development better. I do NOT let you work in pairs so you can split 2 tasks up among 2 people. If I discover you have split the tasks up instead of working on both collaboratively, you will both receive a 0 for the assignment.