SER422 Spring 2016 Lab2  
Assigned 2/3/13, due 2/18/16 at 11:59:00pm via submission to Blackboard

In this lab you are required to complete 2 tasks. The first asks you to code a simple Java servlet-based application. The second asks you to add simple stateful behavior.

**Submission:**

For submission, you should submit to Blackboard a single zipfile named lab2\_<asurite1>.zip (or .jar) that has within it 2 WAR files (one for each application), and 2 source trees.

* 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. Your ant clean target should take care of this.
* 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.

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

Write a complete web application that does the following:

1. (5 pts) 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. (20 pts) 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 an XML 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. (25 pts) The servlet must accept a GET request that lists all the entries in the file.
   1. If the browser is a Chrome browser then set a pink background color.
   2. 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=Rob&languages=scala+lisp&days=M+W&hair=blonde” on this data (this assumes the custom attribute you added was “hair”):
         1. “Bob Smith, java scala c, Monday Thursday, brunette”
         2. “Rob Roy, scala ada, Wednesday Friday, blonde”
         3. “Sue Sink, lisp C#, Monday Wednesday, black

Would return 2 but not number 1 and 3. Note that the parameters of different attributes are AND’d together, while within a multi-valued attribute it is an OR. Note that if no parameters are given to the GET then the entire set of entries should be returned. Make certain that the results of GETs are not cached by the browser.

*Constraints*:

1. The web application should use 2 separate servlets for steps 2 and 3. The context name should be /ser422/lab2task1. The POST action (step 2) should go to relative path “post\_coder”. The GET action (step 3) should go to a relative path “get\_coders”.
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. Your HTML (or your servlet-generated content) should not hardcode references to “localhost” or port “8080” anywhere in your code.
3. Persist the data in a local XML file named lab2.xml under your webapp context (not out on your local hard drive somewhere. The XML format is up to you. Remember not to hardcode directory paths, and to consider concurrent access!
4. For this task you are not required to show it works on 2 separate Tomcats running behind an httpd server. However you are required (and always will be) to demonstrate thread-safe coding practices.

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

Task 2 adds 3 new features to Task 1:

*1. Create a landing page that remembers who you are (10)*

Create a new landing (home) page that recognizes the user if s/he has previously been to the site. If the user has not been to the site then *redirect* the user to a page asking for her/his first and last name, and remember it 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.-->use session to remember through the sites, and cookie to remember whether logged in before

*2. Make a multi-page web form (25 pts)*

Split the webform in Task 1 #1 into 5 distinct screens – the first 4 asks a distinct component of Task 1 step 1, and the last page lists the complete set of answered questions for review with Submit and Cancel buttons. Give the user Prev and Next buttons to navigate backward and forward through the workflow, and pre-populate any HTML form widgets with previously entered values. Note Prev is required on the last page with your Submit and Cancel buttons. 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. All forms should use POST.

*3. Return the proper response error codes (15)*

For this lab you must create multiple servlets as one web application. However, Task 1 did not ask you to verify that each servlet returns the valid error responses depending on the request. For this step, modify your code to return the proper response code for all negative test cases. For example, what if the GET request has improper query parameters? What if the GET servlet receives a POST request? (405)Note this is NOT the same thing as writing an error message to the browser window!

*Constraints for Task 2:*

* + - 1. You must use a combination of browser cookies and http sessions to complete this functionality, but it is up to you to decide where each make the most sense.
      2. This task also must use multiple servlets. Again, it is up to you to determine the best design.
      3. Constraints 1-3 from Task 1 still apply, except change your servlet context to /ser422/lab2task2. But Task 1, constraint 4 does not. This means you have to handle a) concurrency between Tomcat instances, b) a shared resource (the XML file) between Tomcat instances, and c) ensure you enable “stickiness” so a conversation happens with the proper Tomcat instance.

*Constraints for the entire lab:*

1. Absolutely no Javascript or CSS for this lab. This includes AJAX. All functionality must be implemented via HTML and Java servlets.
2. No 3rd party frameworks, I want to see you code straight servlets.