CS 318 - Homework 8

Deadline:

Due by 11:59 pm on Wednesday, April 10, 2013

How to submit:

Submit your files for this homework using `~st10/318submit` on nrs-projects, with a homework number of `8`

Purpose:

To practice with JSP pages, Java sessions, and JDBC.

Important notes:

- There is now a "JSP rules" handout along with the Week 10 Lecture examples -- this includes how to obtain the little `cpjsp` script, and indeed those directions are directly listed along with the Week 10 Lecture examples. also. You can use the `cpjsp` script to copy and set the permissions as required for using JSP on nrs-projects.
- Note -- as with servlets, if your JSP page is using an external `.css` file, make sure you use an **ABSOLUTE URL** for each such `.css` file in a `<link>` tag (`http://nrs-projects.humboldt.edu/~whoever/wherever.css`), otherwise your file won't be "found" since the JSP is being executed from the special directory.
  - When you use any external `.css` file(s), submit a copy of it/them as part of your homework submissions.
- LOVELY `HttpServletResponse` METHOD: `sendRedirect`
  - You can use this to redirect to a specified URL!
  - for example:

```
username = request.getParameter("username");
password = request.getParameter("password");

// if either of the above are null, something is very wrong --
//     redirect to login

if ((username == null) || (password == null))
{
    session.invalidate();
    response.sendRedirect("http://nrs-projects.humboldt.edu/" +
                "~st10/whatever/start-over.html");
}
```

- from a SUN JSP TUTORIAL (previously at: http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/JSPIntro7.html ...but apparently no longer):
  - "A JSP declaration is used to declare variables and methods in a page's scripting language. The syntax for a declaration is as follows:

    ```
            <%! scripting language declaration %>
    ```

    When the scripting language is the Java programming language, variables and methods [!] in JSP declarations become declarations in the JSP page's servlet class. "
  - This seems to imply that such declarations are either local methods or DATA FIELDS;
  - SO: I take this to mean that if you'd like a variable to be, essentially, a data field for the servlet class, you should declare it using a JSP declaration tag. If it is more "local", it seems to be OK for that declaration to happen to be in a JSP scriptlet tag.
  - The other implication? You don't want to be, for example, trying to call a method of the pre-defined `request` object as part of the initialization of a variable in a JSP declaration tag... it won't work! (I accidentally tried...) The scope of the pre-defined `request` object doesn't extend to the part of a class where data field declarations are made! (...which *does* make sense -- we know it is probably a parameter of a `doGet/doPost` method. Just do such initialization within a JSP scriptlet tag instead.
- Based on my playing around, there is a built-in `out` object already related to `response`, in case you'd *like* to `out.println` something instead of using `<%` and `%>` to break in and out of Java and HTML...
- Note: All of my experiments indicate that `request.getParameter("string_of_choice")` returns `null` if there is no `string_of_choice=value` pair available to the calling servlet or JSP. This can be particularly useful to know when dealing with checkboxes (since only checked checkboxes return a name=value pair -- getting a value of `null` for such a checkbox could then indicate an unchecked checkbox...)
- JSPs can be very challenging to debug.
  - One suggestion: work very incrementally -- add a little, `cpjsp` and try it, add a little, `cpjsp` and try it; that might make it easy to know where to look if you get the dreaded 500 server error.
  - Test out bits of JDBC in a little Java command-line application first.
  - Use `try-catch` blocks WITHIN JSP scriptlet tags so `SQLExceptions`, for example, can be caught and output (a nicer alternative than a 500 server error for such a case)
- Remember to follow the style guidelines and course standards given or discussed previously for the languages used in this homework. Replace all `Xxxnn`'s below with your HSU username.
- Make sure that you have executed the scripts `create-bks.sql` and `pop-bks.sql`, and that the bookstore tables are successfully created and populated.

The Problems:

Problem 1

We'll start with JSP practice that doesn't involve JDBC or sessions just yet.

Create an HTML page of your choice meeting the following requirements:

- include `prob1` somewhere in its filename, so I'll know it is for this problem
- as always, include a comment with a URL from which I can reach this HTML page
- include a form...
  - ...including either at least three checkboxes, three radio buttons, or a drop-down/select of at least 3 options, your choice, related to some theme of your choice
  - ...whose action is a JSP whose name (of course) starts with your HSU username and whose name also includes `Prob1` somewhere within it
- The responding JSP:
  - should include the URL you would use to access that JSP from the JSP engine once it is compiled and copied there
  - should visibly display your name in the resulting response
  - should VALIDATE the form input -- we'll discuss this concept in more detail later, but for now, it should suffice that it actually explicitly check that the input received from the form is specifically one of the values expected from that form's elements. If it ISN'T, redirect back to the initial HTML form.
  - OTHERWISE, it should somehow display what the user selected from the submitting form.

Your resulting `.html` and `.jsp` files are now ready to submit. (Also submit any additional files if you use them -- for example, included `.jsp` files or `.css` files or etc. -- if any are involved.)

Problem 2

Now for some JDBC in a JSP...

- Design an attractive HTML page:
  - whose name includes `prob2` somewhere within it, so I'll know it is for this problem
  - with your name visibly somewhere within the page
  - with a form that allows the user to enter and submit his/her Oracle username and password.
  - (This initial HTML page should also include a comment giving the URL from which I can run your HTML/JSP combination.)
- The action for this page's form should be a JSP for a resulting page...
  - ...whose name (of course) starts with your HSU username and whose name also includes `Prob2` somewhere within it,
  - ...that includes the URL you would use to access that JSP from the JSP engine once it is compiled and copied there,
  - ...that queries for and attractively displays all publishers' names, cities, and states, in order of publisher name,
  - ...within an HTML table dynamically built by that JSP.

  - This resulting JSP should also visibly include within it your name.
- One more thing: what if the user either enters no username or password, or calls this JSP directly? In this case, redirect to the HTML login page.
- Now, given that this username and password are not used for display, and are not used to build a dynamic SQL statement, I *think* that additional input validation is not necessary here.
  - (These are simply being used to connect to the Oracle database -- if there is any further "junk" in them, I *think* that connection will simply fail...)

Your resulting `.html` and `.jsp` files are now ready to submit. (Also submit any additional files if you use them -- for example, included `.jsp` files or `.css` files or etc. -- if any are involved.)

Problem 3

And now for JSPs involving both JDBC and sessions.

That is, design and write a combination of HTML page and JSP page(s) that meet the following specifications:

- (make sure each HTML page and JSP visibly includes your name somewhere within its resulting HTML page when displayed in a browser,
- ...and make sure that it includes a comment giving the URL from which I can either reach it or access it via the JSP engine)
- the first page (which can be a plain HTML page or a JSP) that the user sees should include a a form simply requesting an Oracle username and password, whose action needs to be a JSP
- when you submit the username and password in this form, since a username has now been entered, this responding JSP creates a form including a drop-down select containing (**dynamically**-queried-and-created) publisher names (and the username and password should be saved via session variables for use in the next step.)
  - Ah, but what is the action for THIS form, with the drop-down of publisher names? It should be a JSP that lists all of the authors-and-titles associated with this publisher, in a dynamically-populated HTML table.
- this responding JSP, then:
  - should include the URL you would use to access that JSP from the JSP engine once it is compiled and copied there
  - dynamically-queries-and-builds a lovely HTML table of the authors-and-titles associated with the selected publisher, in alphabetical order by author
  - (or it redirects back to the original login if reached without username, password, and publisher name provided).
  - Notice that the logic responding to the publisher names drop-down DOES need to at least attempt to validate its publisher name input appropriately -- because publisher names can be a little annoyingly broad, the latest kluge I've tried is to write a little PL/SQL stored function that expects a string, and returns true if it is a publisher name, and false if it isn't. If it returns false, I would be tempted to either put some scolding message or just redirect back to the original login -- but if it returns true, I

think I can safely display the verified publisher name in the resulting page... (and I'd be delighted if anyone thinks of a less-clunky validation approach... 8-))

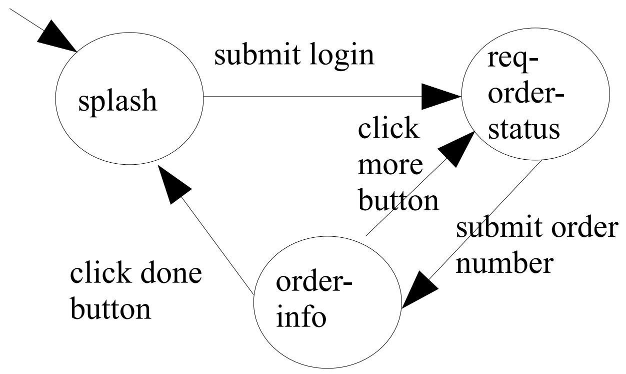
– (and use a `PreparedStatement` for your query, for some additional insurance).
– Feel free to redirect back to the login page if anything fishy is noted.
– Be sure to terminate the session here, also.

Your resulting `.jsp` files are now ready to submit. (Also submit any additional files if you use them -- for example, initial `.html` page or included `.jsp` files or `.css` files or etc. -- if any are involved.)

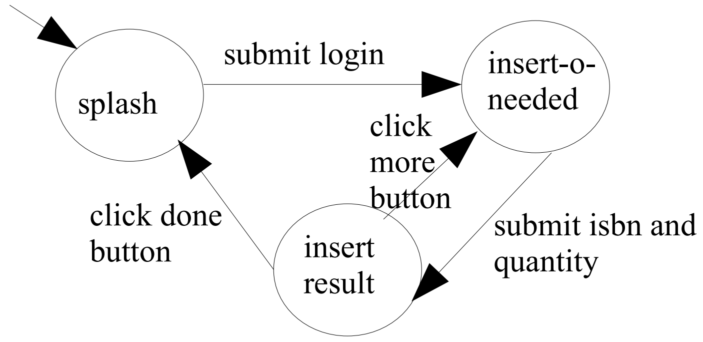
Problem 4

CONSIDER the following two little finite state machines:

4 option 1



4 option 2



You might notice that these are similar to Homework 7 Problem 3's finite statement machines, with the notable difference that these allow the user to go back and request another action OR note that he/she is done.

You should select **ONE** of these, and implement it using JSPs and sessions, noting the following additional specifications:

- You must use at least one JSP, and you must appropriately use a session.
- For either choice, you should make a new copy of Homework 7 - Problem 3's `bks-splash.html` in this homework's directory, so you don't change Homework 7's version, and make the following changes:

- Modify the HTML5 comment containing the URL I can use to view your `bks-splash.html` from your nrs-projects account to reflect this version's new directory. (Note that, for full credit, this URL must successfully display Homework 8's version of this page when I paste it into a browser.)
- Change its form's action so that it uses an appropriate JSP
- Make whatever changes are necessary (if necessary) so it uses your `bks.css` and `ck-login.js`. (Although no changes to these are required for this homework, you are expected to submit copies of them along with this homework's files.)

- The username and password must be passed along as session attributes -- and you may have as many additional session attributes as you would like.
- All displayed pages should use your `bks.css`, for a unified look-and-feel
- Any drop downs of order numbers or ISBNs or etc. must be be dynamically queried and generated -- you will not receive full credit if they are "hard-coded"
- You should invalidate your session when it is appropriate to do so.
- You should use a `PreparedStatement` for implementing any queries involving information provided by the user.
- MAKE SURE that EACH file includes a comment containing the URL where each of these can be found (for a JSP, in its source code put the URL you would use to access that JSP from the JSP engine once it is compiled and copied there).

in either case:

Your JSP(s) `.jsp` file(s), `bks-splash.html`, `bks.css` and `ck-login.js` are now ready to submit, along with any additional files if you use them -- for example, included `.jsp` files or etc. -- if any are involved.