CS 318 - Homework 7

Deadline:

Due by 11:59 pm on Wednesday, April 3, 2013

How to submit:

Submit your files for this homework using `~st10/318submit` on nrs-projects, with a homework number of `7`

Purpose:

To practice with Java servlets, Java sessions, and JDBC.

Important notes:

- Note that you are using an external `.css` file for the HTML5 pages involved in this homework. To do so for pages created by servlets in HSU's servlet setup:
  - you should 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 servlet is being executed from the servlet engine's special directory.
  - note that the `beginHtmlCss` method in the `Html318Helpers` class lets you specify the name of a CSS file to be included in a `<link>` tag in the generated "beginning page" HTML.
  - When you use an external `.css` file for the `.html` with a form that has a servlet as its action, then note that you should also use that same external `.css` file for the form generated by the servlet, for stylistic consistency within that little 2-screen application.
  - When you use any external `.css` file(s), submit a copy of it/them as part of your homework submissions.
- Servlets can be very challenging to debug.
  - One suggestion: you could implement the JDBC part as a Java command-line application whose command-line arguments are the information you would be obtaining from the request form, so that you can implement, test, and debug the JDBC portion in a "friendlier" environment. Then it should be reasonable to adapt that code to work within a servlet.
  - Another: when sessions are involved, you can implement bare-bones "skeleton" pages first, just making sure you have the navigation between the pages down, and then fill in the rest.
- Remember to follow the style guidelines and course standards given or discussed previously for the languages used in this homework.
- 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 some Java servlet practice that does not involve sessions.

Consider your HTML5 page `req-order-status.html` from Homework 4, Problem 3, and your external CSS `bks.css` that modifies it.

Make a new copy of `req-order-status.html` in this homework's directory, so you don't change Homework 4's version.

For this homework's version, the following changes should be made to `req-order-status.html`:

- Modify the HTML5 comment containing the URL I can use to view your `req-order-status.html` from your nrs-projects account to reflect this version's new directory. (Note that, for full credit, this URL must successfully display Homework 7's version of this page when I paste it into a browser.)
- Change its form's action so it uses the Java servlet described below.
- Since this incarnation really has nothing to do with `bks-splash.html`, remove the link to `bks-splash.html`.

We won't be changing your Homework 4 `bks.css` for this homework (unless you just feel like improving it). You are expected to submit a copy along with this homework's files. But, it is your choice if you want to make a copy of `bks.css` in this homework's directory, or have your modified `req-order-status.html` link to the version in your Homework 4 directory.

And, write a Java servlet `XxxnnGetOrderStatus.java` that will now serve as the action for the form in `req-order-status.html`. To make it safer (I hope) to deal right now with the order number obtained from this form from the user, use a `PreparedStatement` for implementing the needed query within this servlet. Note that it should generate a dynamic version of the page that you hard-coded as `order-info.html` in Homework 4, Problem 5.

MAKE SURE that EACH of these files includes a comment containing the URL where each of these can be found (for the servlet, in its source code put the URL you would use to access that servlet from the servlet engine once it is compiled and copied there).

Your servlet's `.java` file, `req-order-status.html`, and `bks.css` are now ready to submit.

Problem 2

We'll continue with a bit more Java servlet practice not yet involving sessions.

Now consider your HTML5 page `insert-o-needed.html` from Homework 4, Problem 4, and your external CSS `bks.css` that modifies it.

Make a new copy of `insert-o-needed.html` in this homework's directory, so you don't change Homework 4's version.

For this homework's version, the following changes should be made to `insert-o-needed.html`:

- Modify the HTML5 comment containing the URL I can use to view your `insert-o-needed.html` from your nrs-projects account to reflect this version's new directory. (Note that, for full credit, this

URL must successfully display Homework 7's version of this page when I paste it into a browser.)

- Change its form's action so it uses the Java servlet described below.
- Since this incarnation really has nothing to do with `bks-splash.html`, remove the link to `bks-splash.html`.

Again, we won't be changing your Homework 4 `bks.css` for this homework (unless you just feel like improving it). And, again, while you will be submitting a copy as part of this homework, it is your choice if you want to make a copy of `bks.css` in this homework's directory, or have your modified `insert-o-needed.html` link to the version in your Homework 4 directory.

And, write a Java servlet `XxxnnInsertOrderNeeded.java` that will now serve as the action for the form in `insert-o-needed.html`. (That is, it will make an appropriate call to the PL/SQL stored procedure `insert_order_needed` when this page's form is submitted.) In this case, for now you can have the servlet return a valid HTML5 page, also using `bks.css`, that has text summarizing what has been inserted, and including a hypertext link that links back to `insert-o-needed.html`.

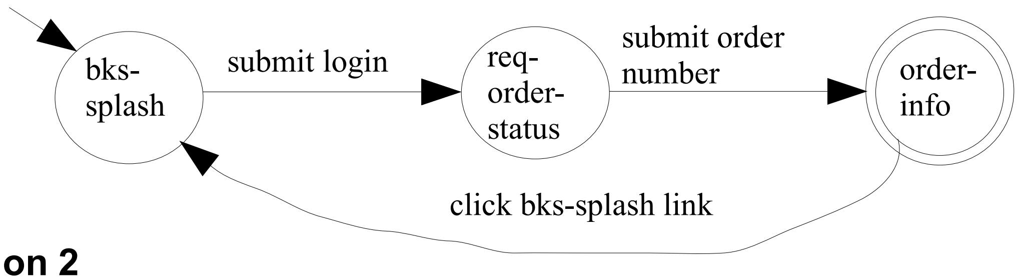
MAKE SURE that EACH of these files includes a comment containing the URL where each of these can be found (for the servlet, in its source code put the URL you would use to access that servlet from the servlet engine once it is compiled and copied there).

Your servlet's `.java` file, `insert-o-needed.html`, and `bks.css` are now ready to submit.

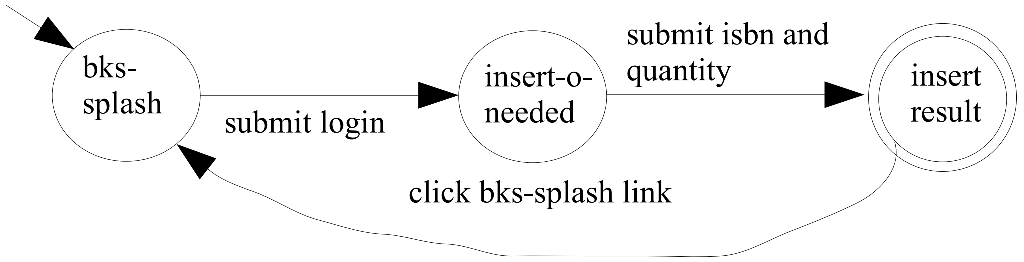
Problem 3

Consider these two little finite state machines:

3 option 1



3 option 2



These model two simple applications for which sessions would be helpful. (They are deliberately simple, to serve as a "first" session-practice.) I used a double-circle accepting state here in more of a "final state" sense, to indicate that that state is intended to be the logical end of the logical transaction/session.

Select **ONE** of these two to implement, meeting the following specifications:

- You may use one Java servlet or two in implementing your choice, but you must use at least one Java servlet, and you must appropriately use a session.
- For either choice, you should make a new copy of Homework 4's final `bks-splash.html` in this homework's directory, so you don't change Homework 4'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 7's version of this page when I paste it into a browser.)
  - Remove the links to `req-order-status.html`and `insert-o-needed.html`
  - Change its form's action so that it uses a Java servlet, described below.
  - Make whatever changes are necessary so it uses your `bks.css` and `ck-login.js`.
  - (As for `bks.css`, we won't be changing your Homework 4 `ck-login.js` for this homework (unless you just feel like improving it). You are expected to submit a copy along with this homework's files. But, it is your choice if you want to make a copy of `ck-login.js` in this homework's directory, or have your modified `bks-splash.html` link to the version in your Homework 4 directory.)

Now, what happens when you submit `bks-splash.html`'s form depends on which state machine you choose to implement.

IF you select 3 option 1:

- The Java servlet that is the action for `bks-splash.html`'s login form should:
  - obtain the login information from the requesting form
  - store that login information in session attributes
  - dynamically generate a response page containing something quite similar to `req-order-status.html`, EXCEPT:
    - ...it will use the login information to dynamically query for order numbers to populate the select/drop-down in the version of `req-order-status` that it dynamically generates.

      (For this problem, **you will lose credit if you hard-code in the order numbers** -- building the drop-down with contents from the database is the better approach, and now you can finally do it. In day-to-day operations, no one wants to have to modify and recompile the servlet every time there's a new order!)
    - ...it will NOT include username and password fields (since these will instead be saved as session attributes, so the user need not enter them again)
    - ...it will not include the direct link back to `bks-splash.html`
    - ...its form will have as its action a servlet -- either this same servlet, or a separate servlet, your choice.
- A Java servlet (either the same one that handled `bks-splash.html`'s form, or a second servlet) should handle a request from this resulting second page's form -- the action will be remarkably similar to that of Problem 1's `XxxnnGetOrderStatus.java`, dynamically generating a response page

containing something quite similar to `order-info.html`, with a few IMPORTANT differences:

- Instead of obtaining the login information from the requesting information, it NEEDS to obtain the login information from session attributes!!
- This is considered the logical "end" of this little transaction/session, so the session should then be INVALIDATED at this point.
- Change the link back to `req-order-status` to instead be a link back to `bks-splash.html`
- SO, if this is a separate servlet from the one that handled `bks-splash.html`'s form, give it a different name from Problem 1's servlet, to distinguish it!
- Be sure to still use a `PreparedStatement` for implementing the needed query within this part.

MAKE SURE that EACH of these files includes a comment containing the URL where each of these can be found (for the servlet, in its source code put the URL you would use to access that servlet from the servlet engine once it is compiled and copied there).

IF you select 3 option 2:

- The Java servlet that is the action for `bks-splash.html`'s login form should:
  - obtain the login information from the requesting form
  - store that login information in session attributes
  - dynamically generate a response page containing something quite similar to `insert-o-needed.html`, EXCEPT:
    - ...it will use the login information to dynamically query for ISBNs to populate the select/drop-down in the version of `insert-o-needed` that it dynamically generates.

      (For this problem, **you will lose credit if you hard-code in the ISBNs** -- building the drop-down with contents from the database is the better approach, and now you can finally do it. In day-to-day operations, no one wants to have to modify and recompile the servlet every time there's a new ISBN for a new title added to inventory or a new book release!)
    - ...it will NOT include username and password fields (since these will instead be saved as session attributes, so the user need not enter them again)
    - ...it will not include the direct link back to `bks-splash.html`
    - ...its form will have as its action a servlet -- either this same servlet, or a separate servlet, your choice.
- A Java servlet (either the same one that handled `bks-splash.html`'s form, or a second servlet) should handle a request from this resulting second page's form -- the action will be remarkably similar to that of Problem 2's `XxxnnInsertOrderNeeded.java`, with a few IMPORTANT differences:
  - Instead of obtaining the login information from the requesting information, it NEEDS to obtain the login information from session attributes!!
  - This is considered the logical "end" of this little transaction/session, so the session should then be INVALIDATED at this point.
  - Change the link back to `insert-o-needed` to instead be a link back to `bks-splash.html`

   - SO, if this is a separate servlet from the one that handled `bks-splash.html`'s form, give it a different name from Problem 2's servlet, to distinguish it!

- MAKE SURE that EACH of these files includes a comment containing the URL where each of these can be found (for the servlet, in its source code put the URL you would use to access that servlet from the servlet engine once it is compiled and copied there).

in either case:

Your servlet's/servlets' `.java` file(s), `bks-splash.html`, `bks.css` and `ck-login.js` are now ready to submit.