**CS 328 - Homework 9**

**Deadline**
11:59 pm on Friday, May 2, 2014

**How to submit**
Submit your files for this homework using `~st10/328submit` on nrs-projects, with a homework number of 9

**Purpose**
To get practice using PHP in combination with Oracle and sessions, and to start working on a slightly-larger application.

**Important notes**
- You can use PHP's `sprintf` function to get a string version of a number formatted as desired (for example, to a set number of fractional places).
  
  You can read more about it in the PHP manual, but, for example, this statement sets a variable `$formatted_price` to a string containing a string depiction of formatting a number `$price` to 2 fractional places:
  
  ```php
  $formatted_price = sprintf("%.2f", $price);
  ```

- There is a possibility that you might be expected to eventually demonstrate some of the applications in this homework in class before the semester is over.

- 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.

**Problem 1**
This first problem uses PHP **without** involving sessions, and with a **static** SQL query. Create a single PHP document, `auth-titles.php`, that does the following:

- When initially invoked, it creates an HTML page that attractively:
  - contains your name visibly within it
  - allows the user to enter his/her Oracle username and password

- When the above form is submitted, this same PHP document should create an HTML page that:
  - queries the database (connecting using the submitted username and password) so that it can
attractively project, in an HTML table element, all of the author names and book titles, in alphabetical order by author, and for authors with more than one title, in secondary order by title.

Submit your resulting auth-titles.php document. (If you use any server-side includes or server-side requires in this PHP document, or use any external .css files for any generated HTML pages, then submit those files as well.) ALSO include a link to your auth-titles.php from your index.html file on nrs-projects.

**Problem 2**

Fun fact: you know from in-class examples that the 2nd argument of function `oci_result` can be the name of the column in the current row whose value you want. The PHP manual page for `oci_result` confirms that this second argument can indeed also be a number, the column number (1-based?!) of the column in the current row whose value you want.

That said, this problem uses PHP with a query, but still without involving sessions. Create a single PHP document, `pub-info.php`, that does the following:

When initially invoked, it creates an HTML5 page that attractively:

- contains your name visibly within it
- allows the user to enter his/her Oracle username and password
- includes **radio buttons** allowing the user to select exactly 1 of the following:
  - publisher's order minimum
  - publisher's over-minimum discount
  - publisher's state
- includes a submit button

- When the above form is submitted, this same PHP document should create an HTML5 page that queries the database (connecting using the submitted username and password) so that it can:
  - attractively project, in an HTML table, all of the publisher names, (in alphabetical order of publisher name),
  - and also project EITHER the publishers' order minimums IF that radio button was selected,
  - OR the publishers' over-minimum discounts IF that radio button was selected,
  - OR the publisher's state IF that radio button was selected

(so, the resulting projected HTML table has 2 columns, where the 2nd column depends on which radio button was checked)

- Note that you will need to take care that no other radio button information gets "injected" by a rogue user -- exactly one of these 3 choices should be projected for all of the publishers.

Your resulting pub-choice.php is now ready to submit. (If you use any server-side includes or server-side requires in this PHP document, or use any external .css files for any generated HTML pages, then submit those files as well.) ALSO include a link to your pub-info.php from your
index.html file on nrs-projects.

Problem 3

This problem now involves PHP and sessions. Create a single title-info.php document that uses PHP sessions in doing the following:

• It needs to use session variables appropriately.

• It should use a "navigational" if statement, implementing the screens with the help of PHP functions that follow that if statement, in the style of the posted try-trio.php example.

• (screen #1) creates a form allowing the user to enter his/her Oracle username and password (and includes a submit button, and your name visibly somewhere within the HTML5 page);

• (screen #2) using the submitted username and password, the same title-info.php document dynamically-creates a form including a drop-down/select with ISBNs queried from the Oracle database (and includes a submit button);

• (screen #3) using the ISBN selected by the user, the same same title-info.php document queries for the title, author, publisher name, price, and quantity on hand for that ISBN, creating HTML5 to attractively display this information on the screen.
  
  – This query should use a bind variable for the selected ISBN. (A PHP bind variable is roughly analogous to a Java PreparedStatement's ? -- you can see an example of how to use this in the example try-insert.php posted along with the Week 13 Lecture 2 examples.)
  
  – this part should invalidate/destroy the session;
  
  – this part also includes a simple form with just a "Done" submit button whose action is this same same title-info.php document (and when this button is clicked at this point, screen #1 should again be displayed).

Submit your resulting title-info.php document. (If you use any server-side includes or server-side requires in this .php document, or use any external .css files for the generated HTML5 pages, then submit those files as well.) ALSO include a link to your title-info.php from your index.html file on nrs-projects.

Problem 4

START to design and build the 4-screen BOOKSTORE application first described in Homework 8 - Problem 4 (whose description is also included here).

By this homework's deadline, you are expected to at least complete Screens 1 and 2 (they should be implemented and functional, and should work correctly), and Screen 3 should at least display with all of the required information showing, even if its submit buttons are not yet functional and even if it doesn't yet restrict user input as specified.

One additional requirement: ALSO include a link to your application-thus-far from your index.html file on nrs-projects.

Submit all of your files for your application-thus-far, along with your latest bksales-fsm.pdf.
**General Requirements**

- you are required to use HTML5 on the client-side, and some JavaScript as well
- you are required to make appropriate use of CSS to maintain consistency between the different screens; make your screens as attractive and easy to use as possible
- you may use a combination of Java servlets (IF they are runnable... 8-( ) or PHP for the application tier. But input validation is required, regardless.
- you are required to use sessions to pass information between the four screens -- and you are required to invalidate/terminate your sessions appropriately
- you are required to make appropriate use of appropriate PL/SQL stored procedures/functions we have developed previously for this scenario (such as `sell_book`)
  - you are permitted to write and use additional PL/SQL stored procedures/stored functions as you wish.
- your elements/fields need to be laid out neatly
- your boilerplate (that's just a term for "hard-coded", non-dynamic text) should be spelled correctly.
- currency formatting should line up appropriately through all 4 currency fields and should be to two fractional places

**Screen 1:**

- include an appropriate title for your bookstore
- include a way for the user to enter an Oracle username and password (the password field should be of type "password"!)
- include a submit button with the label "Log in"
- logging in should lead to SCREEN 2.

**Screen 2:**

- If the user types in an **invalid** username/password, decide which of these options you would prefer:
  - redirect back to SCREEN 1
    OR (slightly-more-informative, but requires an extra click by the users to retry):
  - show SCREEN 1A, which simply prints a "friendly" "that username/password combination didn't work" message of your choice with a "Back" button that takes you back to SCREEN 1 when it is pressed. This screen would be designed to be readable and user-friendly.
    OR (more advanced, I think):
      - redirect back to SCREEN 1 -- but that now also includes an eye-catching "friendly" "that username/password combination didn't work" message of your choice
- include an appropriate title for this 2nd screen for your bookstore.
• populate a <select> (drop-down box) element with ordered ISBN's from the bookstore database. Have it show at least 3 ISBN's at a time (that is, make use of the <select> element size attribute, which allows you to specify this).

• include a "Quantity sold" label and textfield, with contents initially 1.

• include a "Proceed" submit button, which leads to SCREEN 3, and an "Exit" submit button, which leads back to SCREEN 1.

Screen 3:

• what must be the case, if you have reached this screen appropriately? If you determine that any of that is NOT the case, decide which of these options you would prefer:
  – redirect back to SCREEN 1
    OR (slightly-more-informative, but requires an extra click by the users to retry):
    – show SCREEN 1A, which simply prints a "friendly" message of your choice describing the problem along with a "Back" button that takes you back to SCREEN 1 when it is pressed. This screen would be designed to be readable and user-friendly.
    OR (more advanced, I think):
    – redirect back to SCREEN 1 -- but that now also includes an eye-catching "friendly" message of your choice describing the problem

• include an appropriate title for this 3rd screen for your bookstore.

• include textfields and labels populated with the ISBN selected from SCREEN 2, its Publisher name, its Quantity sold (as selected from SCREEN 2), its Title, its Author, its Price, the computed Subtotal for a sale of this quantity, the computed Tax for a sale of this quantity (use a reasonable non-zero tax rate of your choice), and the computed Total for a sale of this quantity, including tax. These must be attractively formatted.

• use JavaScript to make sure only the Quantity field can be changed by the user at this point; if the user wants to change which book is being sold, he/she needs to use the Cancel button to return to SCREEN 2 (see below).

• the Price, Subtotal, Tax, and Total should all be displayed in Currency format to 2 fractional places

• include a Complete submit button that updates the database appropriately, using the sell_book PL/SQL stored function, and then leads to SCREEN 4.

• include a Cancel submit button that leads back to SCREEN 2 without updating the database.

Screen 4:

• what must be the case, if you have reached this screen appropriately? If you determine that any of that is NOT the case, decide which of these options you would prefer:
  – redirect back to SCREEN 1
    OR (slightly-more-informative, but requires an extra click by the users to retry):
– show SCREEN 1A, which simply prints a "friendly" message of your choice describing the problem along with a "Back" button that takes you back to SCREEN 1 when it is pressed. This screen would be designed to be readable and user-friendly.

OR (more advanced, I think):

– redirect back to SCREEN 1 -- but that now also includes an eye-catching "friendly" message of your choice describing the problem

• include an appropriate title for this 4th screen for your bookstore.
• include something displaying a confirmation message of how many copies of the selected ISBN have been successfully sold.
• include an OK submit button that leads back to SCREEN 2.

Submit all of the files for your application-thus-far.

**Problem 5**

**START** to design and build the the at-least-3screen application using "your" database first described in Homework 8 - Problem 5 (whose description is also included here).

By this homework's deadline, you are expected to at least **complete** Screens 1 and 2 (they should be implemented and functional, and should work correctly), and Screen 3 should at least be reached, even if it is not yet complete.

One additional requirement: ALSO include a link to your application-thus-far from your index.html file on nrs-projects.

Submit all of your files for your application-thus-far, along with your latest your-db-fsm.pdf.

**General Requirements**

Consider "your" database. Think of a task that would be useful for someone using "your" database that would also require at least 3 screens (so that sessions will be needed). Implement this, reflecting your submitted your-db-fsm.pdf, using Java servlets (IF they are runnable) or PHP.