CS 318 - Homework 11

Deadline:

PRESENTATIONS of your work on these problems will be given **IN CLASS** on **WEDNESDAY, MAY 8, 2013.**

Homework submissions are due by 11:59 pm on **Wednesday, May 8, 2013.**

How to submit:

Submit your files for this homework using `~st10/318submit` on nrs-projects, with a homework number of `11`

Purpose

To get practice with GUI design and to consider usability in interface designs, and to complete the slightly-larger bookstore application.

Important notes:

- You are expected to be prepared to demonstrate something operational for the problems below in class on Wednesday, May 8. Note that these presentations will make up part of your homework grade for these two problems, AND will also be part of a Week 15 "lab" exercise grade (even though these presentations are being done in lecture, to give you two more days to get them ready).
- 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 - 40 points

Part of your grade for this problem (15 points) will be related to displaying something operational for this problem in-class on Wednesday, May 8; a Week 15 "lab" exercise grade will also be partially determined by this demonstration (even though you will be doing it in lecture, to give you 2 more days to prepare it.)

Now, for something a little different: a GUI design exercise:

This assignment was adapted from a user interfaces course at New York University.

The following describes a number of components for carrying out a single task.

- Lay out the components in forms within **two or more** HTML screens (implemented with PHP- or JSP- or Java-servlet-based sessions),
  - making appropriate use of an external CSS for consistency between screens,

  - and somehow appropriately using sessions between the screens;
  - (The sessions aspect could be as simple as a greeting to the user on a subsequent page, or something more intricate (perhaps summarizing one's pizza order on a final page?))
- **Your goal**: to design these following the user interface design guidelines discussed in class.

Here are a few more ground rules:

- Include within a comment the URL one could use to run your resulting design, in the "initial" source code file for this problem.
- So as to not provide any hints regarding task flow, the components are arranged in alphabetical order.
- You cannot change the provided components or boilerplate,
  - ...but you are free to add graphical elements such as lines and borders to the layout, and to play with font attributes like bold and italic.
- You are also permitted to add punctuation to the boilerplate text as desired, if you think it would be helpful;
  - you may also add additional boilerplate, if you wish.
- Finally, you may add additional components for navigation between HTML screens.

Do not worry about whether you agree with the components I've chosen; just think of this as a design exercise.

The task: Ordering a pizza

- Boilerplate "Address" and text fields (for street address, city, and zip)
- A button labelled "Cancel"
- Boilerplate "Credit Card" and a text field (with room for the text "0000 0000 0000 0000")
- Boilerplate "Crust" and at least two radio buttons; include at least two crust options of your choice (e.g., "White" or "Whole Wheat", "Thin" or "Thick", "Crispy" or "Chewy", etc.). (Do not include more than five crust options.)
- Boilerplate "Expiration" and two drop-down lists, one for month (with names of months spelled out in full, e.g., "November") and one for years (e.g. "2013").
- Boilerplate "Last Name" and one text field
- A button labelled "Order"
- Boilerplate "Phone Number" and from one to three text fields, your choice (with room for the text (XXX) XXX-XXXX)
- Boilerplate "Price" and a text field (with room for the text "$XX.XX") with the X's displaying the price of the currently selected pizza. $00.00 or $0.00 should be the initially-displayed price.
- Boilerplate "Quantity" and one text field
- Boilerplate "Size" and at least three radio buttons; include at least three pizza size options of your choice (e.g., "6 inch", "8 inch", or "12 inch", "18 inch", "21 inch", or "24 inch", etc.) (Do not include

more than five size options.)

- Boilerplate "Total" and a text field (with room for the text "$XX.XX") with the X's displaying the total cost of the order. $00.00 or $0.00 should be the initially-displayed total.
- Boilerplate "Sauce" and at least two radio buttons; include at least two pizza sauce options of your choice (e.g., "Red (Tomato Sauce)" or "White (Four Cheese)", "Marinara" or "Clam", etc.) (Do not include more than five sauce options.)
- Boilerplate "Toppings" and at least nine checkboxes; include at least nine pizza topping options of your choice (e.g., "Anchovies", "Bacon", "Broccoli", "Extra Cheese", "Mushroom", "Onion", "Pepperoni", "Peppers", "Sausage", "Tofu", "Zucchini", etc.) (Do not include more than fifteen toppings options.) You may also have subgroups of toppings within these if you wish (with appropriate subgroup headings).
- Note: you can only order **one kind** of pizza with this application, but you can order **any number** of this one kind of pie.

Submit your files implementing the above. Again, make sure that you have something operational to demonstrate in-class on Wednesday.

Problem 2 - 60 points

Part of your grade for this problem (15 points) will be related to displaying something operational for this problem in-class on Wednesday, May 8; a Week 15 "lab" exercise grade will also be partially determined by this demonstration (even though you will be doing it in lecture, to give you 2 more days to prepare it.)

Complete the 4-screen BOOKSTORE application first described in Homework 9 (whose description is also included here). By this homework's deadline, it should be implemented and functional, and should work correctly; you should modify your CSS and layout to try to apply some of the usability guidelines discussed in-class.

In a comment in your code for Screen 1, remember to include the URL I can use to run your completed application.

Submit all of your files for your application, 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, JSP, and/or PHP for the application tier-- your choice! But input validation is required, whichever combination you choose.
- 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.