CS 3340 (Spring 2017) - Homework 2 – ASP [Course Registration System]

Due date: 11:00 p.m., Thursday, March 2, 2017

Overview

This is an individual assignment. You will write a system that allows a student to register for classes. A working version of the assignment (demo) can be found at: *https://lucius.valdosta.edu/dgibson/hw/hw2_sp16/default2.aspx*

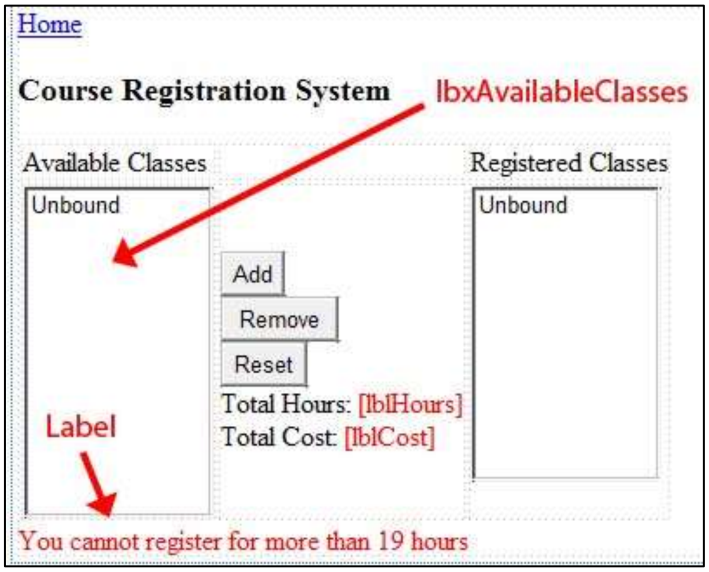
Requirements

1. Create a Web Site named *HW2*.

   a. Create *HW2* folder on a flash (or hard) drive.
   b. Open VS and choose: *File, New Website*, navigate to *HW2* folder.

2. Create a page named ***default.aspx***. The design of the page is shown on the right.

   a. Add the HTML title: "HW 2 – FirstName LastName", substituting your name.



You might choose to do the next two steps (*b* and *c*) at the very end when you have posted to Lucius.

   b. Add a link titled "Home" that links to your course homepage.

   c. Open your course homepage. Add the text, "HW 2" to the page and make it a link to your HW 2 home page.

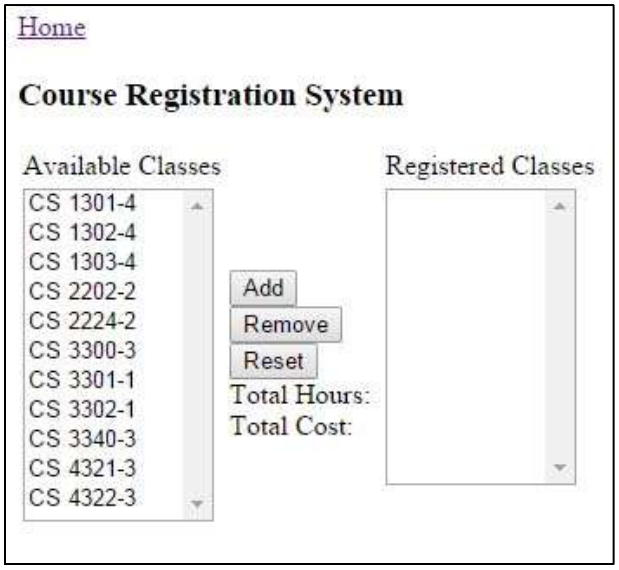
   d. Create the page as shown. It is recommended using a table to layout the part of the GUI with the ListBoxes.
   e. There are three labels. They should not show up when the page is first loaded.
   f. We will build the list of available course with code the first time the page is loaded. Begin by adding this method which builds and returns an array of ListItems:

```
private ListItem[] buildAvailableCourseList()
{
    ListItem[] tempList = { new ListItem("CS 1301-4", "CS 1301-4"),
                            new ListItem("CS 1302-4", "CS 1302-4"),
                            new ListItem("CS 1303-4", "CS 1303-4"),
                            new ListItem("CS 2202-2", "CS 2202-2"),
                            new ListItem("CS 2224-2", "CS 2224-2"),
                            new ListItem("CS 3300-3", "CS 3300-3"),
                            new ListItem("CS 3301-1", "CS 3301-1"),
                            new ListItem("CS 3302-1", "CS 3302-1"),
                            new ListItem("CS 3340-3", "CS 3340-3"),
                            new ListItem("CS 4321-3", "CS 4321-3"),
                            new ListItem("CS 4322-3", "CS 4322-3")
                          };
    return tempList;
}
```

g. Next, call the preceding method and bind the return to the ListBox. Add this code to Page_Load.

```
if (!Page.IsPostBack)        //For initial page creation
{
    ListItem[] availableCourses = buildAvailableCourseList ();
    lbxAvailableClasses.DataSource = availableCourses;
    lbxAvailableClasses.DataTextField = "Text";
    lbxAvailableClasses.DataValueField = "Value";
    lbxAvailableClasses.DataBind();
}
```

h. Run your page. It should look like as shown on the right:



CS 1301-4
CS 1302-4
CS 1303-4
CS 2202-2
CS 2224-2
CS 3300-3
CS 3301-1
CS 3302-1
CS 3340-3
CS 4321-3
CS 4322-3

3. Program the "Add" button so that all the selected courses in the Available list are moved to the Registered list. Thus, they should be removed from the Available list.

4. Program the "Remove" button so that all the selected courses in the Registered list are moved to the Available list. Thus, they should be removed from the Registered list.

5. Modify the "Add" button so that it updates the Total Hours and Total Cost. Note:

- Each credit hour costs $100
- The *Text* and *Value* both have the format: *Course Number-creditHours*. This format should not be altered.
- You will need to use *Split* method found in the String class to obtain the number of hours from the *Value* (or *Text*).
- You might want to add a multi-line textbox at the bottom of your page for debugging. I recommend using this to make sure you are obtaining the correct values for the hours for each course before programming the total hours (and cost).

Hints:

a. I recommend a helper method (*HoursRegisteredFor*) that computes the total hours that have been registered for. Just loop through the *ListItems* in the registered list, strip the hours off each, add them up, and return the total hours.

b. Then, at the end of the add method, you have code like this:

```
double hours = HoursRegisteredFor()
lblHours.Text = hours.ToString()
```

Note: Thus, you are computing "hours" and "cost" from scratch every time. It might be nice to "store" things, but we don't have a memory between postbacks (yet!).

c. You can use the two helper methods and code above in the "remove" method also!

d. Another approach is when you enter the "add" method, grab the hours from the label, then as you loop through the "available" course to find the ones selected, strip the hours off the selected ones, adding them to the value obtained from the label. Finally, put this value back into the label. This is fine (and slightly more efficient-but that is not a concern here). However, I believe the simplest approach that leads to the cleanest code is what is described above.

6. Run your page and test.

7. Modify the "Remove" button so that it updates the Total Hours and Total Cost. See hints for requirement 5 above.

8. Run your page and test.

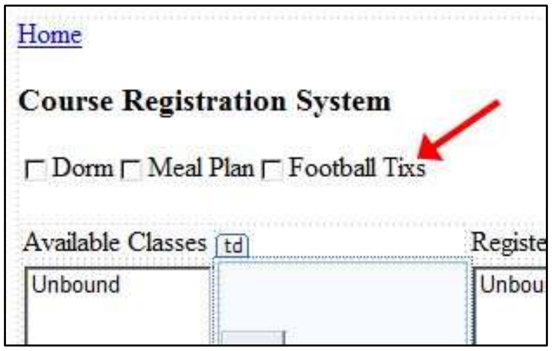
9. Program the "Reset" button so that the system is completely reset: no registered classes, total hours and cost are 0, and the original list of available courses is shown.

10. Run your page and test.

11. Modify the "Add" button so that the user cannot register for more than 19 hours. If a course puts the total hours over 19, it should not be added and an error message should be displayed as shown on the first figure in this document. The label should be hidden when it is no longer applicable. For instance, if a course is removed, or another added that does not put the total over 19 hours, or the system is reset.

12. Run your page and test.

13. Add a CheckBoxList as shown on the right.



14. Modify the "Add" (and "Remove") buttons so that they factor in the cost of any of the selected items in the CheckBoxList. The Dorm costs $1000, the meal plan is $500, and the football tickets are $50.

    Hints:

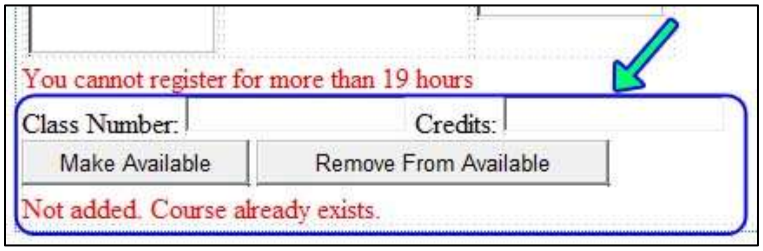
    a. I recommend a helper method (*ExtrasCost*) that computes the costs associated with the "extras" (e.g. the textboxes). Just check each *ListItem*, add cost when selected, and return total cost.

    b. Then, at the end of the add (similar for remove) method, you have code like this:

```
double hours = HoursRegisteredFor()
lblHours.Text = hours
lblCost.Text = hours * 100 + ExtrasCost()
```

15. Run your page and test.

16. Add the items shown box below to the bottom of the GUI (but above any debug TextBox you might have added). The text at the bottom ("Not added…") should be a Label.



You cannot register for more than 19 hours

Class Number: Credits:

Make Available Remove From Available

Not added. Course already exists.

17. Add a Validation control (not shown in the GUI above) to the number of credits entered which must be an integer between 1 and 10 inclusive. Note: you can get the points for this even if you do not complete requirement 15. You would simply have a Make Available click event that does nothing.

18. Add a click event for the "Make Available" button. For now, it will do nothing.

19. Run your page. Type in an invalid value for the number of credits and press the "Make Available" button. The validation error should be displayed.

20. Set the "CausesValidation" property to *false* for only these three buttons: "Add", "Remove", "Reset" (not the "Make Available" and "Remove From Available" buttons). This prevents the validation from taking place when the user is Adding a course (or Removing, etc.).

21. Program the "Make Available" button so that it takes the Class Number and Credits and adds this course to the Available list. In a moment we will modify this so that it will not add a duplicate course; for now, it will.

22. Run your page and "Make Available" a course.

23. Continuing from the previous requirement, if the **Class Number** (e.g. CS 1301) already exists (either in Available or Registered), it should not be added and the error message, "Not added. Course already exists." should be displayed. For example, "CS 1301-4" is in the list initially. If the user typed in: "CS 1301" for the class number and "4" for the credits (or "1", or "2", etc.), then this course would not be added. Hint: you need to (a) loop through the Available courses, extract the course name and compare it to what the user typed in, (b) next, loop through the registered list and do the same.

24. Run your page and test.

25. Program the "Remove From Available" button so that it takes the Class Number and removes this course from the Available list. Notes: the user is not required to enter the credits when removing a class, thus, any value there should be ignored. Recommend reading the next few requirements before coding.

26. Run your page and test.

27. Continuing from the previous requirement, if the course is in the Registered list, do not remove it, but display the error message, "Not removed. Course is registered for."

28. Run your page and test.

29. Continuing from the previous requirement, if the course is not found in either the Available or Registered lists then display the error message, "Course not found."

30. Upload to Lucius (Copy your *HW2* folder to your root folder on Lucius)

31. Test

    a. Open a browser on Lucius and test your course home page and your HW 2 home page. If something doesn't work, open your website (your course website: c:\Students\yourID), navigate to your *HW2* folder and run your page off local host.

    b. Open a browser on your local machine and test: (i) your course home page, (ii) your HW 2 home page.

    c. **Log off Lucius**

    d. You're done!

Grading Rubric

Criteria	Points	Description
1	5	Links to and from HW 2 work
2	25	Add and Remove move courses
3	20	Hours and Cost labels correct
4	10	Reset works correctly
5	20	Can't register for more than 19 hours
6	5	Checkboxes update cost on postback
7	5	Validator on number of credits
8	5	Make Available works correctly
9	5	Remove From Available works correctly
Total	**100**	

Time Estimate:

If you have done all the readings, played with the examples in the downloads, studied and thought about the material and you are a very good programmer then you might could do the assignment in 4-5 hours. I suspect for many of you it will take at least 10 hours.