SYSTEMS DESIGN / CAPSTONE PROJECT
MIS 413

Client Checkpoint #6
Complete the CRUD for the PersonTable in the DB

Objective: Complete the CRUD for the person table

Part A: Fix prior errors from your Checkpoint #4 submission

1. Check Entropy and fix any errors noted.

Part B: Toggling so only the INSERT text box objects or GRID VIEW is visible at one time:

1. Open the profile web page in the User folder
2. Replace the label that says "Add a new user" with a button, name this button _addStart, set the text to be Add a New Person, set the cssClass to be btn btn-success
3. After this button you should have a </h3>
4. After the </h3>, place a new object: a **panel** from the toolbox to the webpage.
    a. Name the panel _pnlAdd, set visible=false
    b. Move the ending </asp:panel> tag to after your submit and clear buttons as well as after the two 's
5. Coding under the _addStart button
    a. Make the panel Visible
    b. Make the grid Invisible
    c. Make the _addStart button Invisible
    d. Clear all textboxes and reset the drop down to Please Select
6. Coding under the Clear button
    a. Make the panel Invisible
    b. Make the grid Visible
    c. Make the _addStart button Visible

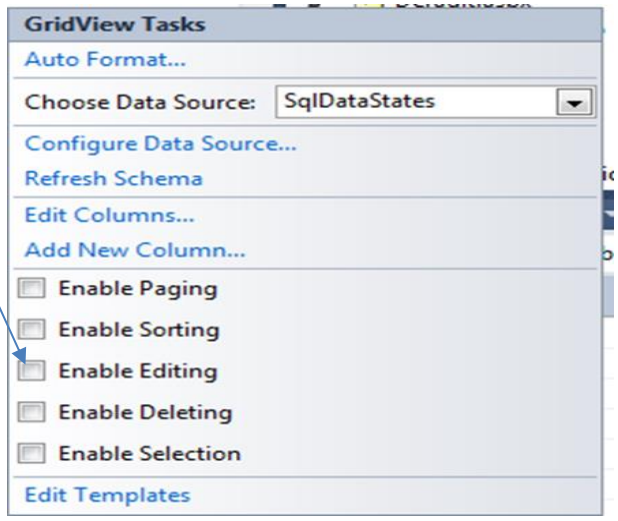
Part C: Update the Grid on the Profile page to permit "edit" and "delete"

7. Click on the sqlDataSource (silver button), and then click the little right arrow or right click and then Smart Tag
8. Select Configure
9. Continue to click to the Stored Procedure options that has 4 tabs at the top
10. Select the Update tab, and then select your PersonUpdate Stored Procedure
11. Select Delete, and then select your PersonDelete Store Procedure
12. Ignore Insert for now
13. When it asks you to refresh, answer No as you will lose your formatting for the grid

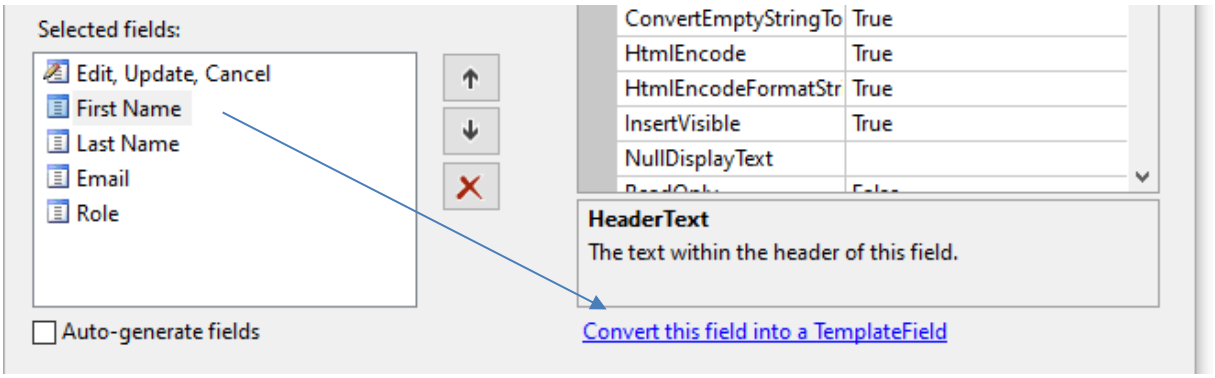
14. Now select your grid and the right smart tag (arrow)

15. To enable, EDITING, click the Enable Editing Option on the GRID View Tasks Dialog, do not click Enable Deleting,

16. Once you Enable Editing, the Grid View will manage the Update, Cancel operations for you! However, it does not enable Error Checking, so we will do that manually.



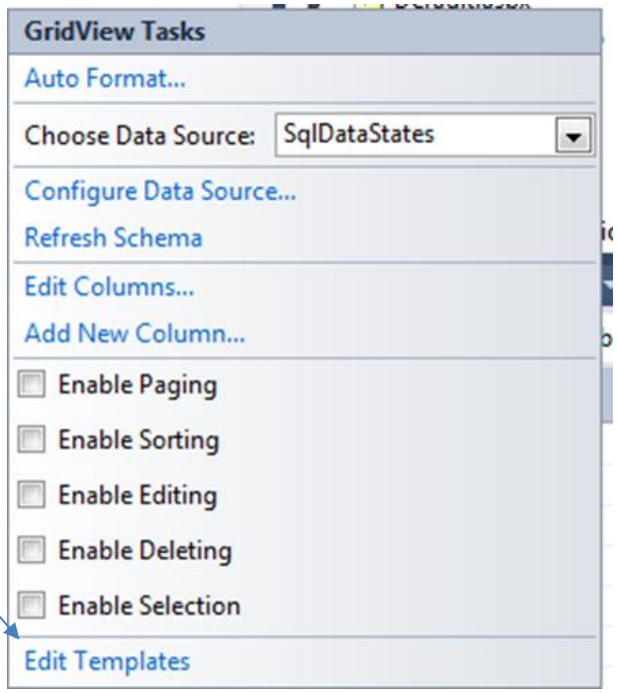
17. Select Edit Columns
18. Then select each of the column(s) you want to provide additional error checking and for each field click the "Convert this field to a Template Field". For the person table consider, firstname, lastname, email and phone. Do not convert the Role column to a text box.



19. Save and test your project to date, when you click EDIT on the grid the labels should change to Text Boxes! You should note it also made Role a text box so we need to indicate that those should not be updated

20. Back on Edit Columns for your grid, under Selected Fields select Role and then in the right properties window change the ReadOnly property to TRUE.

21. One final check before we add Error Checking, return back to grid and find the grid properties, verify that the DataKeyNames property is set to our primary key for the person table – this being personID.
22. Test your project, by running it and modifying a first name, does it update the database?



Part D: Add validation controls to provide error checking

23. Right click on the grid again to get to the GridView Tasks dialog and click the bottom Edit Templates option.

24. Select the EDIT Template option under the firstName Column

25. Add the "Required Field Validator" to the EditItemTemplate for first name. The easiest way to accomplish this is to click once in the text box, then click the right arrow on your keyboard (It should have a blinking cursor to the right of the text box). Then double click the Required Field Validator object in the Toolbox. Set the following properties:
    a. ControlToValidate- textbox1 (you do not need to change the name of this textbox)
    b. CSSClass – errored
    c. Display – Dynamic
    d. Error Message - *Required

26. Repeat for the other fields as necessary

27. Remember, some fields may require two validation controls (i.e. Email)

28. Finally, it is a good idea to let the user know the data has been updated. In the properties box for the Grid, click the EVENT option (Lightning bolt) and find the ROW UPDATED property, double click that property and a sub routine will pop up. In this subroutine you can type a message as in:

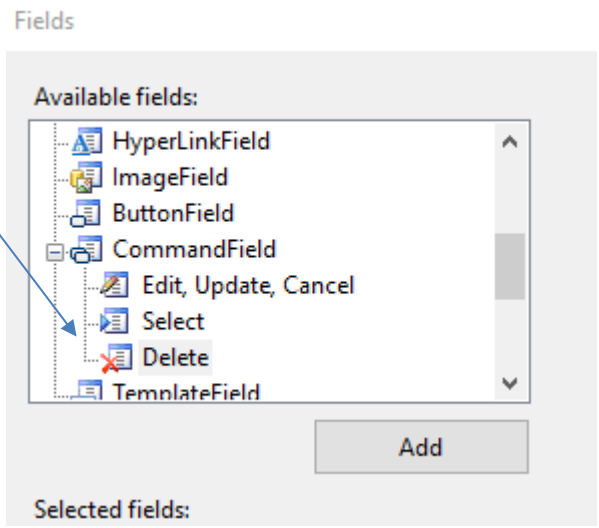
    this._message.Text = "Person was Updated";

29. Run and Test your Grid, once you update your records, open the database table to verify they have been updated

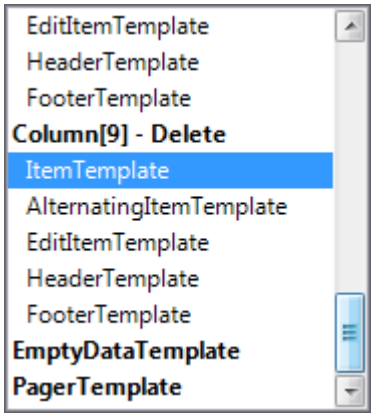
Part E: Add the delete option to your grid

30. Click on the grid, and from Grid View Tasks (right arrow), and then EDIT Columns

31. From the Available Fields List, select the **command field** then DELETE option



32. Return to Grid View Tasks and select Edit Templates, find in the drop down list, your new delete column and select ITEM Template.



33. Once you select ITEM Template, you will see a DELETE link button, click on it and change the following properties:
    a. fore color to RED
    b. modify the onclientclick property to be:
       return confirm('Are you sure you want to delete this person?')
    c. The above will add the javascript to produce an ALERT box.

34. You should give the user a message that the row has been deleted, use the ROW DELETED event to assist you.

35. Add a fake record to the database (In the database) and then test the delete option.

Part F: Adding the code to insert a new record to the database.

36. Add coding under the 'clear' button to make the grid visible and the panel invisible as well as make the _addStart button visible. Change the text on the button to be Cancel / Clear Data.

37. Coding under the SUBMIT button follows. We will use 'coding' versus a wizard here to learn how to add code to bind to a database.

    Before adding the code under the SUBMIT button:

a. **Ensure that your insert stored procedure in SQL actually inserts a new record – TEST in SQL. We will add additional SQL statements to check if the new record already exists in the table.**

b. Add the DLL to enable database binding to you page, add the following code to the TOP of the aspx.cs page (after the other using statements)

```
using System.Data;
using System.Data.SqlClient;
```

c. Following is the code to insert a record using C# code.
d. Double Click the Submit button and insert the code below (replace the code)
e. Please remember you cannot copy/paste from this pdf into Visual Studio, you may however copy the code on this page and the next page and paste into notepad and then copy from notepad into Visual Studio. This will remove any special codes the PDF files have embedded in them.

(remember you cannot do a direct copy/paste of this code into VS, and and item shown in brown will need to be changed to your specific project variables/stored procedures)

```
            // this will insert a new record into the database
           //build a link the the name/pwd/user for your particular database
           //note below the [mis413...] should be the name of your connection string,
           //open your webconfig file and find the connection string name
            string dbConn =
System.Configuration.ConfigurationManager.ConnectionStrings["mis413ConnectionString"].
ConnectionString;

            //build a connection to the database
            SqlConnection conn = new SqlConnection(dbConn);

            //use the above connection to execute a particular stored procedure
     //[substitue your validTableInsert stored procedure name below]
     using (SqlCommand cmd = new SqlCommand("[yourInsertStoredProcedureName]", conn))
            {
                    cmd.CommandType = CommandType.StoredProcedure;

                //build the parameters (input items) that the stored procedures needs
                    cmd.Parameters.AddWithValue("@roleDescription",
this._roleDescription.Text);
               // add more parameters as needed by your stored procedure

                // open the database and actually run the stored procedure, also catch
//any errors and display them in your _message label, also refreshed grid to show new
//record
                    try
                    {
                        conn.Open();
                        int intResponse = Convert.ToInt16(cmd.ExecuteScalar());
                        if (intResponse == 0)
                        {
                            this._message.Text = "Role already existed - Not Added";
                        }
                        else
                        {
                        this._message.Text = "New Role was Added";
                        }                       // following line refreshes your grid
//for the new record
```

```
                        this._nameOfYourGrid.DataBind();

                }
                    //if there are any errors with the store procedure, display
//them in the message label
                catch (SqlException ex)
                {
                    this._message.Text = "Error on inserting into the Valid Roles
Table " + ex.Message;
                }
          }
        }
```

Part G: Modify the Web.Config file to show errors on the server

38. Open the web.config file
39. Locate the </system.web> tag
40. Before that command, insert a new line with this xml tag:
    <customErrors mode= “Off” />

Part H: Test your project and email a link

1. **Please TEST your project by entering the public link as in: http://misCapstone.uncw.edu/S23FolderName, the project should load, and click the MENU in the navigation menu to see your Menu page.**

2. When you are ready for grading, and you have tested your project by using a https://miscapston address, go to Canvas and post your full http://miscapstone..... Link in the Client Checkpoint #6 homework box.