Project #4 – Convert to Web Application

DUE DATE: 3/17/20 – End of Day

Description:

Your IT department likes the direction of your application but has decided they need it to cater to a wider audience. They have now instituted a new requirement that the application be converted into a web application. In a final step, they would like you to also implement the create new entry feature, which has thus far been non-functional.

Skills Used: interfaces, multi-tier applications using class libraries & MVC architecture, unit testing, web applications using state management, data validation controls, browser cookies and databases, recognize and apply appropriate data structures

User Experience:

1. At Startup, the user is presented with a web form displaying buttons with identical meaning as the windows form version from Project #3 and immediately displaying row #1 from the access database
2. If the user clicks the Next Button, the next row in the access database is displayed and the appropriate ID value is shown in the center label situated in the area between the Next and Previous buttons. If the user attempts to click Next beyond the last row, the page simply refreshes and stays on the last item.
3. If the user clicks the Previous Button, the previous row in the access database is displayed and the appropriate ID value is shown in the center label situated in the area between the Next and Previous buttons. If the user attempts to click Previous on the first row, the page simply refreshes and stays on the first item.
4. If the user clicks the Save button, any changes made to the displayed fields will be propagated to the database.
5. If the user clicks the Reset button, the currently selected row is simply refreshed from the access database. The current selected row displayed by the web form does not change.
6. If the user clicks the New Entry button, the fields are cleared, the row label in the center should display zero ("0") and the user may type in values for each of them. Once finished entering the data, the user may click the Save button, which propagates those changes to the databases and immediately selects that row for display in the web form. The user may also click the Next button to proceed to the first record in the access database if they do not wish to save their changes.

Project Instructions

1. Clone or download the repository for this project to your local machine. If you have troubles retrieving the project from the repository URL, please contact the instructor for assistance.

   GITHUB PROJECT URL: https://github.com/gottjl01/cs203-project4-cs
   GITHUB REPOSITORY URL: https://github.com/gottjl01/cs203-project4-cs.git

2. You will need an internet connection to restore the nuget packages in this project.
3. You must make all unit tests pass with the test explorer. Do not modify any of the unit tests. Unlike previous projects, a web application cannot fully benefit from unit tests as a windows application can, so some aspects of this project will be graded on expected behavior, which is explained in the lecture the day the project is assigned and the information above under "User Experience".
4. The **PFW.CSIST203.Project3** Project in this solution must be modified as follows:
   a. Change Application to a Class Library – You must change Project3 from a Windows Form Application into a Class Library. Right-click on the project, go to properties, select the application tab on the left, change the application type to class library. You are done with this requirement!
   b. AccessPersister: CreateRow() – This method must create a data table object, add the rows manually to the datatable, call the NewRow() method on the DataTable to create the new Row object and return it.

c. AccessPersister: StoreRow() – This method has been renamed as part of the interface from UpdateRow() to StoreRow(). This method must now perform the same function it did before with updating the supplied row, but if the row has an "ID" column of zero, the row must be inserted into the database. It must then execute a 'SELECT @@IDENTITY' to retrieve the newly assigned ID of the new row and assign it to the DataRow's ID column. You will need to write the SQL statement that inserts the new row yourself.

5. The **PFW.CSIST203.Project4** Project's Default.aspx must be modified as follows (you can reach the code view of the default.aspx page by right click on it and selecting "View Code" in the dropdown):
   a. Add Project3 Reference: Right-click on Project 4, Select Add, Select Reference, Select Project > Solution, and make sure a checkmark is next to Project3. This requirement is complete!
   b. Local Persister Variable – define a local private variable that creates an instance of the AccessPersister class. Supply the following into the constructor so the persister receives the correct file path of the access database file:

      ```
      System.Web.Hosting.HostingEnvironment.MapPath("~/App_Data/sample-data.accdb")
      ```

      NOTE: A backup of the access database called 'sample-data.original.accdb' is located in the App_Data folder as a backup should you need to restore the database during web application test runs.

   c. _Default Constructor: Assign the CurrentRow to a value of 1
   d. Current Row Property: Read and store this value from the page's ViewState. The ViewState must be used to achieve proper behavior and this cannot use a standard getter/setter.
   e. Page_Load: Detect if the page is a PostBack, and if not, check the number of database rows, and if that number is greater than zero, call the UpdateDisplay() method.
   f. btnPrevious_Click: If the CurrentRow property has a value greater than 1, decrement the CurrentRow by a single value and then call the UpdateDisplay() method
   g. btnNext_Click: First, retrieve the number of rows from the persister using the CountRows() method. If the CurrentRow property is less than the max number of rows, increment the CurrentRow property by 1 and then call UpdateDisplay()
   h. btnSave_Click: First check to see if the current page is IsValid() and if so, retrieve the row pointed to by the CurrentRow property. If the DataRow returned is null, then call the CreateRow() method on the persister to create it. Then, assign all of the DataRow's columns from those values present in the textboxes on the webform and call StoreRow() with the DataRow. Also, update the lblID to show the ID column from the data row after StoreRow() is called and assign the DataRow's ID to the CurrentRow property.
   i. btnReset_Click: Call the UpdateDisplay() method
   j. btnNewEntry_Click: Set the CurrentRow property to zero and call the UpdateDisplay() method.
   k. UpdateDisplay(): Retrieve the row for the value specified by the CurrentRow property. If the row returned is null, then assign the empty string to all of the textboxes on the form and set the lblID.Text to "0". If the row is non-null, assign all of the textboxes the corresponding values from the DataRow and lblID.Text accordingly.
   l. Required Field Validators: Add required field validators to each of the textboxes that uses a reasonable message to the user informing them that the field is required.
6. If you receive 'Provider not registered' errors when connecting to the access database, please download and install the following package from Microsoft: https://www.microsoft.com/en-us/download/details.aspx?id=54920

Code Documentation

Code comments are meant to provide a brief explanation in areas where the code is less self-explanatory. I ask that you use common sense and critically think about the places a reader of your code might need some guidance. In this project, I have provided many comments in the areas that are pre-implemented that you have been instructed not to modify. However, any method, field or property that you create or modify must have comments added using the standard IDE behavior in the Visual Studio IDE by typing three slashes (///) above a method, property or field. This will cause the IDE to automatically create the commenting structure that you will need to fill in. All parameters and return information must be completed.

Submission

Zip your assignment, including all source, project and solution file(s) and submit the archive through Blackboard by the due date. Late projects will be accepted with a one-week grace period, but no later barring extenuating circumstances, which must be communicated and approved by the professor.

Grading

Your grade is determined by the following rubic:

Scoring Rubic	
Assignment Task	**Points**
Unit Tests	10
Proper User Experience Behavior with Web Application	30
Class, Method and Property Documentation	10
TOTAL	**50**

Extra Credit

You may earn additional credit on this project by implementing the same project using VB.NET. You can find the online github repos of the VB.NET version of these projects at the following URLs:

GITHUB PROJECT URL: https://github.com/gottjl01/cs203-project4-vb
GITHUB REPOSITORY URL: https://github.com/gottjl01/cs203-project4-vb.git

Please include the VB.NET solution with your normal project solution. It is best to place the two solutions in different folders and ZIP them up together for the Blackboard submission.

NOTE: The purpose of the extra credit is to encourage you to learn how to convert C# code into VB.NET code using the .NET Framework. DO NOT utilize any online or downloaded tools to automate this process. If your project is detected as having used such tools, you will receive zero extra credit, lose extra credit earned on any other projects and become ineligible for future extra credit on subsequent projects.