Project #2 – Excel file display front-end

DUE DATE: 02/11/2020 – End of Day

Description:

Your sales department would like their own application for displaying their contact information contained in an excel file. The first phase of this development is to create a front-end that displays the contents of the excel document but does not have any editing or manipulation capability. The user may also browse for the excel file that they wish to read. Your co-worker has once again defined the unit tests for you, so you will only need to implement the solution.

Skills Used: interfaces, unit testing, form-based application using data-bound controls

User Experience:

1. At Startup, the user is presented with a form displaying no field data and record number zero
2. User may click Previous or Next buttons to populate the fields with the appropriate record from the excel file
3. User cannot reduce the record number below zero
4. User cannot increase the record number beyond the maximum number of rows in the excel document
5. If the excel data contains zero rows, then the excel row number must remain at zero

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-project2-cs
   GITHUB REPOSITORY URL: https://github.com/gottjl01/cs203-project2-cs.git

2. You will need an internet connection to restore the nuget packages in this project.
3. The main form (frmMain) must be implemented in the following manner
   a. `FrmMain_Load` – Create an instance of the excel persister using the excel persister's parameterless constructor and assign it to the local variable
   b. `BtnPrevious_Click`– Decreases the row number shown in txtRow.Text by 1. Must do nothing if the user attempts to reduce the number to zero or less. If a non-null value is returned from the excel data persister then the data for that row should be displayed in the relevant boxes.
   c. `BtnNext_Click` – Increases the row number shown in txtRow by 1. Must do nothing if the user attempts to increase the number beyond the maximum number of rows in the excel document. If a non-null value is returned from the excel data persister then the data for that row should be displayed in the relevant boxes.
   d. `OnFormClosing` – This method must call its base implementation, checks to see if the persister object is null and if not, call its Dispose() method and assign null to it.
   e. `OpenToolStripMenuItem_Click` – This handler takes care of the file > open dialog. It is already implemented and should not be modified by you.
   f. txtFirstname, txtLastname, txtEmailAddress, txtBusinessPhone, txtCompany, txtTitle must show the data points from the selected row. If row zero is selected (which occurs during startup), no data should be displayed in these fields.

4. The ExcelDataPersister must be implemented in the following manner:
    a. The parameterless constructor – This should initialize the datatable with no rows. This constructor is already implemented for you.
    b. The excelFilepath constructor – If the specified file does not exist (using System.IO.File.Exists), then a System.IO.FileNotfoundException should be thrown.
    c. The excelFilepath constructor – If the specified file does exist, all data contained in the excel file should be read into a data table for the lifetime of the excel data persister using an `System.Data.OleDb.OleDbConnection`. The connection string can be retrieved from the Util class, which is located in the root of the project. The SQL command to retrieve everything from the single sheet in the excel document is the following: `SELECT * FROM [Sheet1$]`. Once everything is loaded into the data table, the command and connection objects must be properly disposed.
    d. GetRow() method – This method must return the specified row from the data table. If the row number specified is non-positive OR beyond the range of available rows, it should return null and not throw an exception.
    e. CountRows() method – This method retrieves the number of rows available in the data table.

5. If you receive 'Provider not registered' errors when connecting to the excel document, please download and install the following package from Microsoft: https://www.microsoft.com/en-us/download/details.aspx?id=54920 If you continue to receive this error message, you may need to change the bitness of your project to match your office product installation. Right Click on your project > Properties > Build > Change "Any CPU" under "Platform Target" to the appropriate bitness version (x86 or x64). The "`UtilTests+GetExcelConnectionStringMethod`" unit test will pass once you configure everything correctly. If the unit test still does not pass, then click on the settings icon (gear) under the Test Explorer window and switch "Processor Architecture for Any CPU Projects" to use the correct bitness (x86 or x64).
6. Do not modify any files except the ExcelDataPersister.cs and frmMain.cs. To receive full credit, you must get all existing unit tests to pass and documentation added to all methods, classes and properties you modified.

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**
frmMain.btnNext.Click.EventRaised Unit Test	3
frmMain.btnPrevious.Click.EventRaised Unit Test	3
frmMain.Load Unit Test	3
frmMain.OnFormClosing Unit Test	3
ExcelDataPersister.CountRows Unit Tests	6
ExcelDataPersister.Dispose Unit Tests	6
ExcelDataPersister.GetRows Unit Tests	6
ExcelDataPersister Constructor Unit Tests	6
Proper User Experience Behavior	8
Class, Method and Property Documentation	6
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-project2-vb
GITHUB REPOSITORY URL: https://github.com/gottjl01/cs203-project2-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.