# FundamentalsOfBuilding\_NET\_DesktopApplications

This document contains notes pertaining to the Pluralsight course “Fundamentals Of Building .NET Desktop Applications”. Some of the clips of the Pluralsight course are documented. The names of those clips are copied to the section headers in this document and to the table below. Using Microsoft Word, you can use the names in the table as hyperlinks to navigate to any particular clip. But using Apache Open Office, these hyperlinks do not work; instead, they merely serve as a table of contents. You can navigate to the start of any clip via bookmarks; type F5 to bring up the Navigator; then double-click Bookmark1 for 1st clip header, Bookmark 2, for 2nd clip header, etc.

|  |  |
| --- | --- |
| [Create a Windows Forms App](#_Create_a_Windows) |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

I have bypassed the earlier sections pertaining to the WinUI demo and the WPF demo. Somewhere in these sections the Pluralsight instructor has constructed the “EmployeeManager” solution. To fill in the gap

* Open Visual Studio. VisualStudio displays a welcome screen titled **Visual Studio 2019**.
* Click **Continue without code** - in the right-hand panel. Visual Studio displays an empty user interface; if the **Solution Explorer** is shown, it will be empty, and the central pane will also be empty.
* From the pull-down menu – at the top line – click **File -> New -> Project...** Visual Studio displays a dialog titled **Create a new project**.
* Type “blank” in the **search** box. Visual Studio displays the **Blank Solution** option in the right-hand pane.
* Select the **Blank Solution** option, and click the **Next** command button. Visual Studio displays a dialog titled **Configure your new project**.
* Type “EmployeeManager” in the **Solution name** text box.
* Navigate to the folder where you want this solution to reside. On my Windows 10 computer, I chose

C:\Users\sncole\source\GitRepositories\VSAndOtherPlatformProjects\

FundamentalsOfBuilding\_NET\_DesktopApplications\

* Click the **Create** command button. Visual Studio displays “EmployeeManager” as the solution name in the right-hand panel.

## Create a Windows Forms App

Open Visual Studio to the EmployeeManager solution. If you encounter any errors from Visual Studio while performing the following instructions, you might have an out-of-date version of Visual Studio. Before installing Visual studio updates, the following steps are recommended.

* Make certain that the operating system (Windows 10) is up-to-date.
* Reboot.
* Do Disk Cleanup.
* Reboot.
* Download and install Visual Studio update.

Now proceed as follows.

* Right-click the solution name in the **Solution Explorer**. Visual Studio displays a pop-up menu.
* Click **Add -> New Solution Folder**. Visual Studio displays a folder named “NewFolder1” in the **Solution Explorer**.
* Rename the new folder to “WinForms”.
* Right-click “WinForms” in the **Solution Explorer**. Visual Studio displays a pop-up menu.
* Click **Add -> New Project...** Visual Studio displays a dialog titled **Add a new project**.
* Make certain that the language choice (immediately under the Search for templates) box contains **C#**.
* Type “WinForms” in the search box. Visual Studio displays a list of templates in the right-hand panel.
* Scroll down the list of templates until you find **Windows Forms App**. (The name “Windows Forms App” should not be followed by any parenthetic text; blank indicates “Core”.) Click it; then click the **Next** command button. Visual Studio displays a dialog titled **Configure your new project**.
* Give the new project the name “EmployeeManager.WinForms”, and click the **Next** command button. Visual Studio displays a dialog titled **Additional information**.
* Select .NET 5.0 as the **Target Framework**. Click the **Create** command button. Visual Studio adds the template components of the new (EmployeeManager.WinForms) project. Visual Studio may, also, select one of the elements of this project to display (by default). On my computer it displays Form1.cs (Design).

Next we will take a tour of what the template provides, and gain familiarity with the designer.

* In the **Solution Explorer** rename Form1.cs to MainForm.cs. Visual Studio responds with a pop-up window containing . . .

You are renaming a file. Would you also like to perform a rename in this project of all references to the code element ‘Form1’?

* Click the **Yes** button. Visual Studio responds by renaming all items in the EmployeeManager.WinForms project to “MainForm”.
* If “MainForm.cs” is not showing (a blank windows form), double-click it in the **Solution Explorer** to make it visible.
* Double-click (to open) Program.cs. The instructions in the (static) method **main()** execute when the application starts. The last instruction starts MainForm().
* Right-click MainForm.cs in **Solution Explorer**. Then select **ViewCode** in the pop-up menu. This allows you to interact with the code pertaining to MainForm (e.g. events).
* **Solution Explorer** also shows MainForm.Designer.cs. This contains code that is automatically generated in response to design changes.
* To illustrate migrate to the designer image of MainForm. While MainForm is selected, open the **Properties** window. Note, for example, that the **Text** property is “Form1”.
* The icons on the 2nd line of **Properties** allow you to list the properties alphabetically or categorized. The lightning-bolt icon allows you to interact with events. For example, double-click **Load** in this list, and Visual Studio creates a skeleton for the event handler in MainForm.cs (code).
* Navigate to MainForm.cs (designer). Open the **Toolbox** (View -> Toolbox). Visual Studio displays a dialog titled **Toolbox.**
* If it is not already expanded, click **All Windows Forms** in **Toolbox**. Visual Studio responds by displaying a large number of controls that are available to be added to the form.
* Drag the **Button** control onto the MainForm designer.
* Navigate to **Properties**, and Visual Studio displays the properties of this **Button** control.
* Using this **Button**’s **Text** property, type the name “Save”, and Visual Studio displays “Save” on the button’s face.
* Change the control’s name using **(Name)** - immediately above the list of properties. Change the name to “btnSave”. Visual Studio confirms the name change by changing the name of the control in the **Properties** window – above the icons.
* Select the lightning-bolt icon to go to btnSave’s events. Double-click the **Click** event. Visual Studio responds by creating a skeleton for the event handler in MainForm.cs (code).
* Navigate to MainForm.Designer.cs. Toward the bottom of this file, you will note a line of code containing “Windows Form Designer Generated Code” followed by the declaration for btnSave -the button that we have just added to the form.
* Notice – at the top of the page – that the content of this page is the **partial class** MainForm; and the content of MainForm.cs (code) also contains **partial class** MainForm. These will be compiled into a single class. Therefore, the code that we might write in btnSave\_click() will have access to btnSave (and all of its properties).
* Navigate back to MainForm.Designer.cs. Click “+” to the left of “Windows Form Designer Generated Code”. Visual Studio displays the content of this block of code.
* Note - near the top of block of code – the code to create the button in the form. Also, after “// btnSave”, you will note the properties that were added via the designer, and the statement to install the button-click event handler.
* The comment at the top of the block of code warns the user not to use the code editor to modify the contents of the block via the code editor. (Modifications are almost always done by the designer.)
* But the Pluralsight instructor tells us that he does use the code editor to remove event handlers. The example that he used to illustrate this was deleting the btnSave\_click() event handler, which he did by removing the handler’s code from MainForm.cs (code). This caused VisualStudio to detect an error, because the deletion was incomplete; the statement in the hidden block in MainForm.Designer.cs has (as we observed) a statement to install the event handler. He corrected the error by removing that statement, as well. ***But I tried deleting the event handler in the button’s properties by erasing the name “btnSave\_Click”. This action caused both the event handler and its install statement to be erased.***
* The block of code is sandwiched between .SuspendLayout() and .ResumeLayout(). Without these statements, the run-time would be responding to the added controls and their properties before all of the statements in the block have been executed. This could cause the page to flicker, and it could result in additional time required to load the form and its controls.
* Delete the button from MainForm. Save the MainForm.cs (designer).
* In **SolutionExplorer** right-click the project, EmployeeManager.WinForms. Visual Studio responds with a pop-up menu.
* Click **Set as Startup Project** in the pop-up menu.
* Click from the pulldown menu **Debug -> Start Without Debugging**. Visual Studio compiles the project and runs the project – displaying the MainForm.