# Introduction to Windows Forms

This document contains notes pertaining to the Pluralsight course “Introduction to Windows Forms”. 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.

|  |  |
| --- | --- |
| [Forms](#_Forms) | [Demo: Forms](#_Demo:_Forms) |
| [Demo: Controls](#_Demo:_Controls) |  |
|  |  |
|  |  |
|  |  |

## Forms

The form displayed when the application starts initially is normally invoked automatically. With regard to secondary forms that are displayed later in the process of application execution, the statements to display such forms are

Show()

or

ShowDialog()

Show() displays the form modelessly, whereas ShowDialog() displays the form modally.

Application start-up also involves the use of the **Application** object. The statement

Application.Run(form)

performs initialization in addition to displaying the start-up form. To end the application one could use

Application.Exit, but this is not really necessary, because closing the start-up form implicitly invokes Application.Exit.

## Demo: Forms

We start by building a code-only project; we will not use template code, and we will not use the **Toolbox**.

* Open Visual Studio. In the right panel click **Create a new project**. Visual Studio displays the **Create a new project** dialog.
* Open the languages drop-down list near the top of the right panel. Choose **Visual Basic**. Type “empty” in the search box at the top of the right panel. Visual Studio reduces the number of candidates in the right-panel list. Select **Empty Project (.NET Framework)** with **Visual Basic** as the language. Click **Next**. Visual Studio displays the **Configure your new project** dialog.

***It is not necessary to have allocated a folder for the solution before creating the solution and 1st project. Visual Studio automatically allocates a new subfolder under the folder selected in the following step, and it copies the solution name as the name of this new subfolder.***

* Type “Simple” into the **Project name** text box. Use the **Location** browser to select a folder into which the solution will reside. Type “CodeOnly” into the **Solution name** text box. Uncheck the check box below the **Solution name** text box. Click **Create**. Visual Studio displays the solution with empty left and center panels. If the **Solution Explorer** is not visible in the right panel, navigate to make it visible.
* Now create the class that contains the windows form. Right click “Simple”; then select **Add -> Class...** from the context menu. Visual Studio displays the dialog **Add New Item**.
* Change the name in the **Name:** text box (bottom left) to “MyForm”. Click **Add**. Visual Studio displays the class MyForm in the center panel.

***By default, classes created in C# are not “public”. It is necessary to type “public” as a prefix to the class name, unless you deliberately want it to be private. The same applies to properties and methods in the class.***

* We want class MyForm to inherit from Systems.Windows.Forms.Form. In order to do that we will have to reference the Systems.Windows.Forms assembly. Right-click “Simple”; then select **Add -> Reference...** from the context menu. Visual Studio displays the dialog **Reference Manager**.
* Expand **Assemblies** in the left panel. Click **Framework** to make it highlighted. Scroll down the list in the center panel to **System.Windows.Forms**, and hover the mouse over this item. Click - to check - the checkbox to the left of this item. Click **OK**. Visual Studio displays **System.Windows.Forms** in the **Solution Explorer** under **References**.
* Navigate back to the body of Class MyForm. Type the line

Inherits Form

Hover the mouse over “Form”, and type <ctrl>.; then accept the invitation “Imports System.Windows.Forms”.

* Add the following code to Class MyForm – a skeleton of the constructor.

Public Sub New() (the VB constructor evidently uses “New” instead of the class’s name)

End Sub

* The constructor can be used to set values for various desired properties. At this point the only property will be the “Text” property, which shows up as the window’s title. In the body of the constructor type

Me.Text = “My Form”

Now we need to supply the “Main” method, the entry point for this application.

* The “Main” method goes inside a module. Right click “Simple”; then select **Add -> Module...** from the context menu. Visual Studio displays the dialog **Add New Item** with **Module** selected (by default) in the center pane. (***In C# we add a new class instead of “module”.***)
* Change the name in the **Name:** text box (bottom left) to “Startup”. Click **Add**. Visual Studio displays the module “Startup” in the center panel.
* Type the following code in the body of Startup.

Public Sub Main()

Dim form as New MyForm

Application

End Sub

***In C# we write “public static void Main()” instead of “Public Sub Main()”.***

* Hover the mouse over “Application”, and type <ctrl>.; then accept the invitation “Imports System.Windows.Forms”.
* Append “.Run(form)” after “Application”. The statement “Application.Run(form)” does 3 things. (1) It starts listening in on the Windows message loop. (2) It sets “form” as the main form for the application. (3) It shows the form.

This completes the construction of the “Simple” application. Build and run the application. The application shows a form whose title is “My Form”. When the C# version of this application runs, the form is displayed, but the console-window is also displayed. To prevent the console window from displaying

* Right-click the project name. Then select **Properties** from the context menu. Visual Studio displays the properties with tab selectors in the left panel.
* With the **Application** tab selected, refer to the **Output type:** dropdown list near the top of the right panel. Use the down-arrow to the right of the dropdown list to change the type to **Windows Application**.
* Close the **Properties** dialog. Rerun the application, and observe that console window is no longer displaying.

## Demo: Controls