



Programming With  
Microsoft Visual  
Basic 2015

# *Creating Simple Programs with VB.NET 2015*

Assistant Lecturer

*Nawfal Turki Obais*

2<sup>Lec</sup>

# Let's Make a Simple Program

We start by making a Program Plan:

A simple description of the desired characteristics and functionality.

Often includes an efficient method of solution (algorithm)

Example: a plan for adding two decimal numbers.

Simple 'Welcome' program (plan):

**Program purpose:** Display a simple message; exit.

We will use 2 Buttons (each called a 'Control')

We will use Visual Studio's Design Window to create these.

**Desired functionality** (program behavior):

TextBox:

**Allows our user to input his/her name**

Clicking Button 1 ("Welcome" Button):

Display 'Hello <User Name>! Welcome to Visual Basic.'

Clicking Button 2 ('Exit' Button):

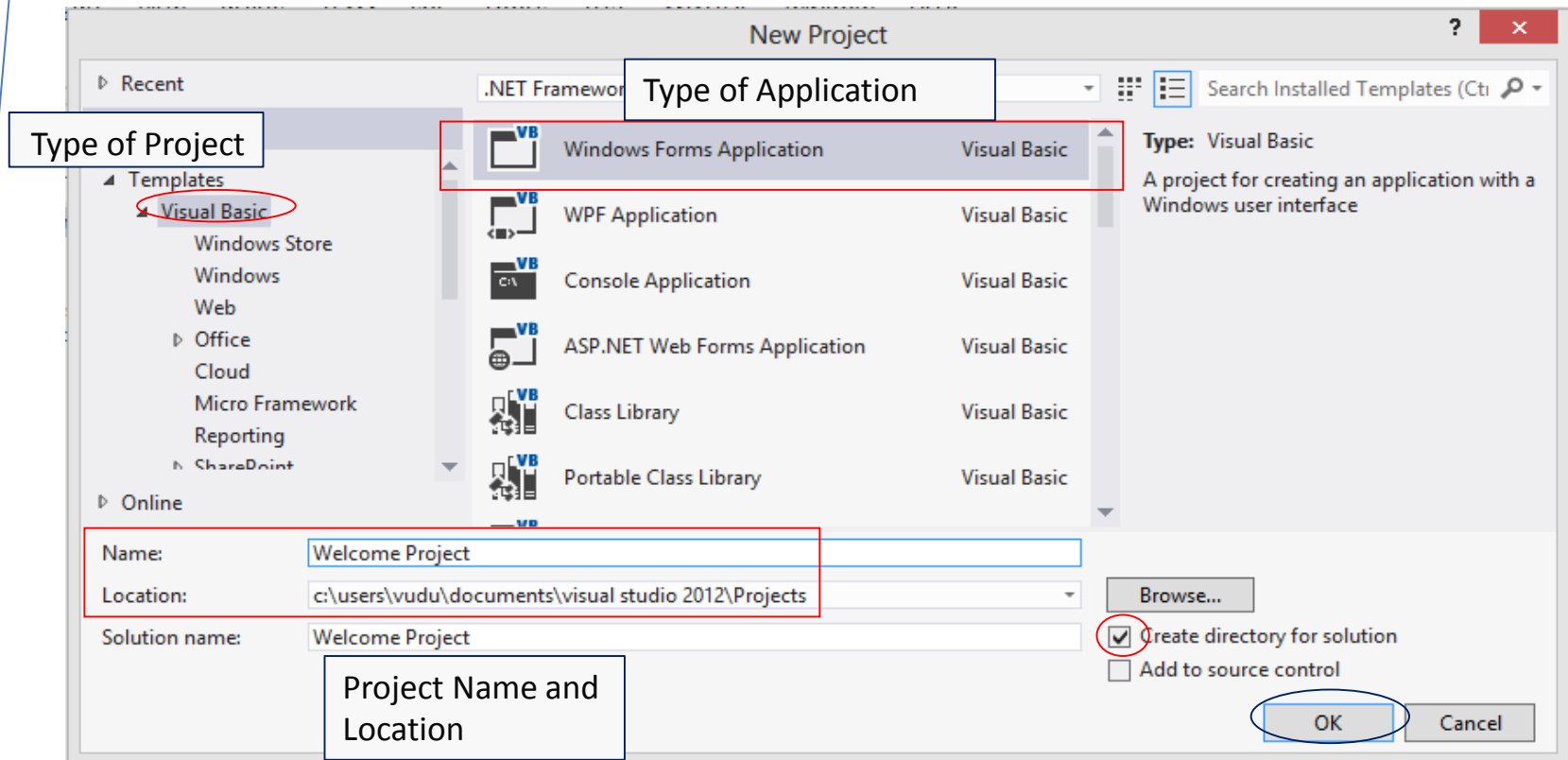
Exit (close the program)

We will add each **Control** to our **Form** using the **Design Window**...

...and then add some simple VB .NET code.

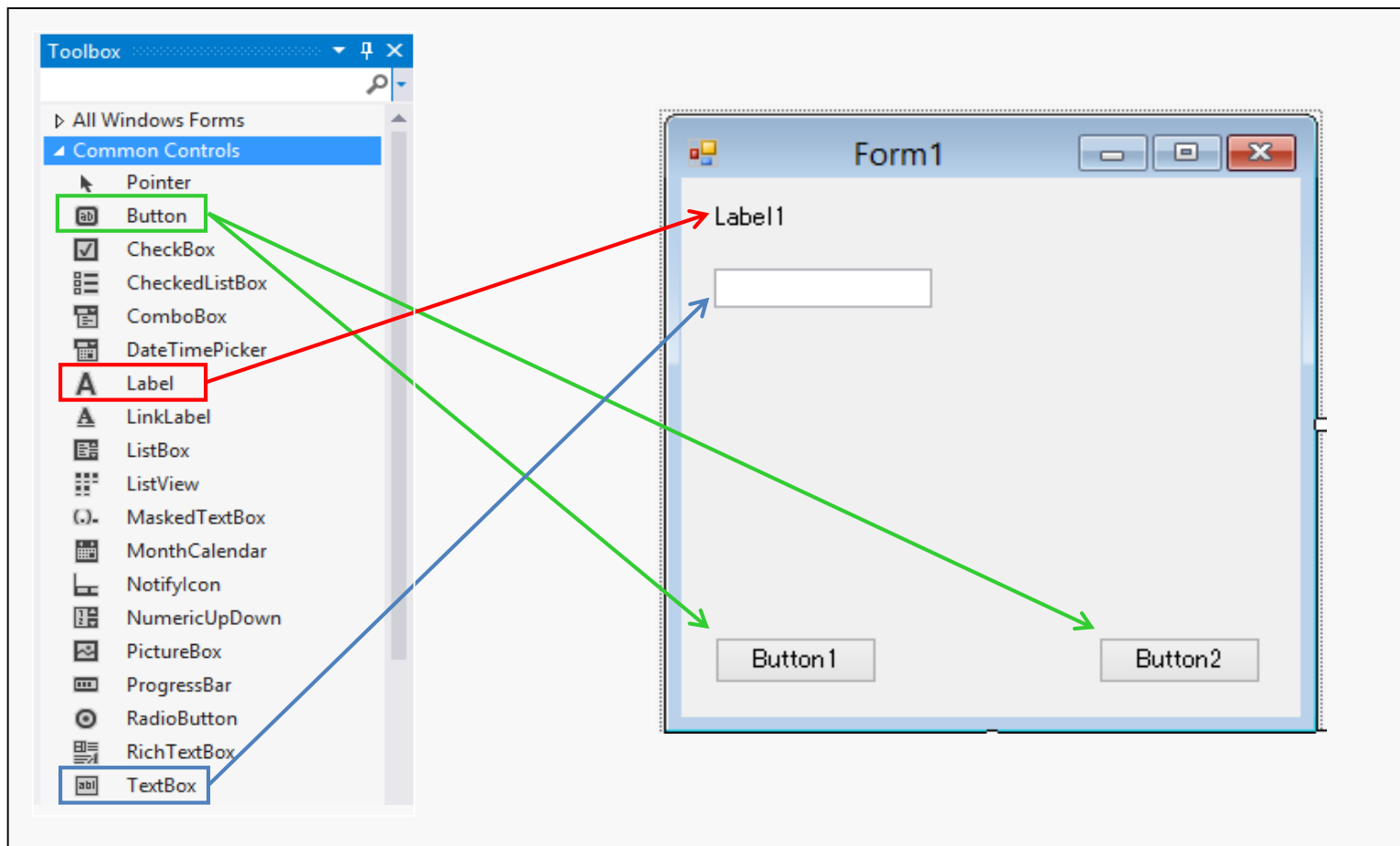
# Step 1: Making the Project

Click 'New Project' in the start screen to display the New Project Dialog:  
Choose settings to make a VB Windows Form (WinForm) Project, as below:  
Name your project 'Welcome Project'; also, keep your default location.



# Step 2: Form and Controls Arrangement

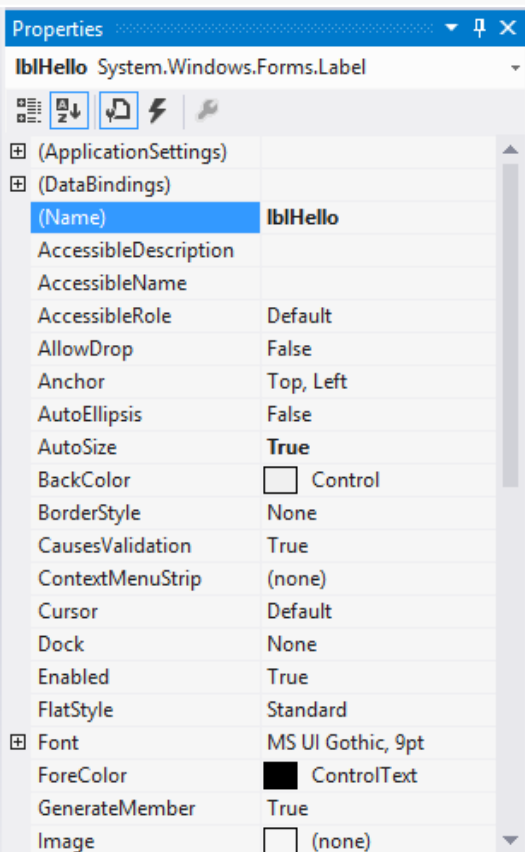
We now add 1 Label , 1 TextBox, and 2 Buttons to our form...



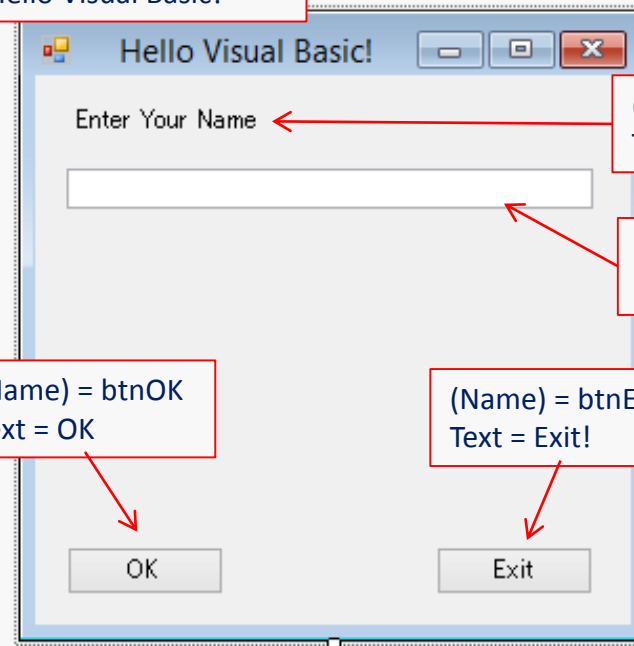
# Step 3: Setting Control Properties

## Adjust the Properties of each Control...

1. Select the desired control (by single-clicking)  
→ Its Properties will be shown in the Properties Window
2. Click each desired Property, one by one.



(Name) = Form1  
Text = Hello Visual Basic!



(Name) = Label1  
Text = Enter Your Name

(Name) = txtWelcome  
Text =

(Name) = btnOK  
Text = OK

(Name) = btnExit  
Text = Exit!

Notice the distinctive way we name our Controls...

### (Name)

Indicates the name used in code to identify the object.

# Step 4: Adding Program Code

Now, let's add the VB code to make the program work...

We do this by coding a **Subroutine** (mini-program) for each active Control.

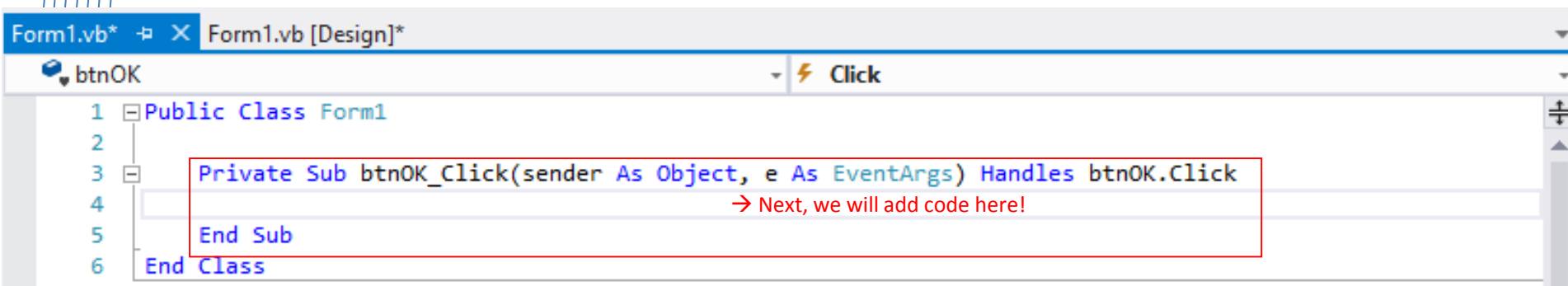
By "active" we mean a Control that will be coded by us to respond to user input.

This type of Subroutine is called an **Event Handler**.

Let's make our Program respond when a user clicks btnOK

To get started, just **double-click** the Control, **btnOK** in the Design Window...

This takes us to **Code View** and gives us an empty **Event Handler** (red box)



```
Form1.vb*  Form1.vb [Design]*  
btnOK Click  
1 Public Class Form1  
2  
3 Private Sub btnOK_Click(sender As Object, e As EventArgs) Handles btnOK.Click  
4  
5 End Sub  
6 End Class
```

→ Next, we will add code here!

Next, we will add VB code to Handle the Click Event of btnOK...

Events are identified as ControlName + . + EventName → **btnOK.Click**

Event Handlers are named similarly, but using an underbar → **btnOK\_Click**



# Step 4 (cont.): Adding Program Code

Now, add the VB code below to **btnOK\_Click**:

Note: Any code we put it in **btnOK\_Click** will execute whenever **btnOK** is pressed, one time from top to bottom.

```
Private Sub btnOK_Click(sender As Object, e As EventArgs) Handles btnOK.Click
    'Display a MessageBox greeting to our user
    MessageBox.Show("Hello " & txtWelcome.Text & ". Welcome to Visual Basic!", _
        "Hello User Message")
End Sub
```

Here, we will display a small **MessageBox** to welcome our user.

The 1<sup>st</sup> line (green text) is a **comment statement**, which does nothing.

The 2<sup>nd</sup> line, **MessageBox.Show()** accepts 2 arguments separated by a comma and a statement extender ( \_ ):

1. Here, the 1<sup>st</sup> argument makes a String to hold our message, in 4 steps:
  - a. First, we make a short **String** ("Hello ")
  - b. The user's name is then fetched from the **Text Property** of **txtWelcome**
  - c. We then make a third String: " Welcome to Visual Basic!"
  - d. Our message is then made from these 3 Strings using the & operator.
2. The 2<sup>nd</sup> argument specifies the text for the **Title Bar** of the **MessageBox**

# Step 4 (cont.): Adding Program Code

Next, let's add the VB code for btnExit\_Click:

First, return to the **Design Window** (left tab), and double-click **btnExit**.

Next, add the code shown below to your new, empty Event Handler:

```
Private Sub btnExit_Click(sender As Object, e As EventArgs) Handles btnExit.Click
    End
End Sub
```

Here, we are coding to let the user exit our program by pressing btnExit.

Using a single VB Keyword → **End**

This style of programming is known as “**Event Driven Programming**”

In this style, our program behaves like a simple automaton (robot)...

It waits for one of our defined user Actions to happen...

1. User Clicks the OK button (btnOK)
2. User Clicks the Exit button (btnExit)

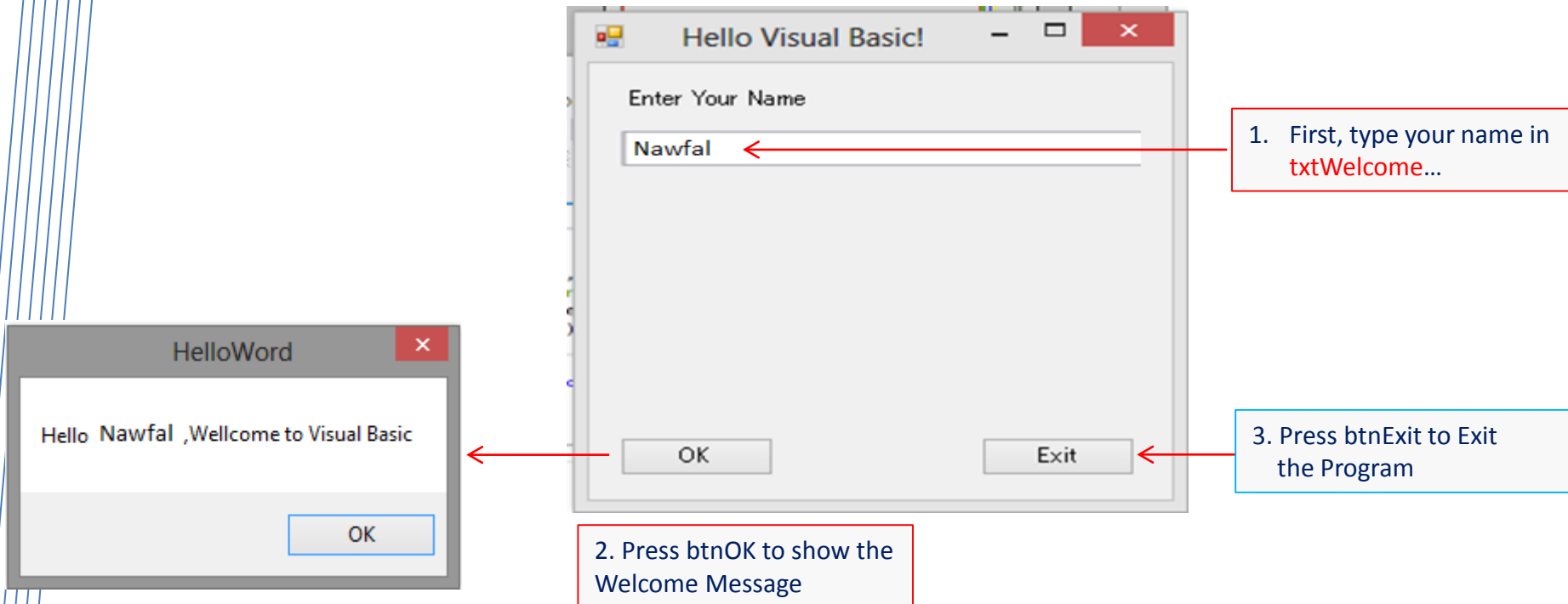
Then, it responds to each action by executing the corresponding Handler.



# Step 5: Program Testing

Click the green triangle (Start) to Compile and Run your Program:

- Here, **Compiling** means taking your VB source code and converting it into a machine-usable form.
- Then, test your program (as the User):



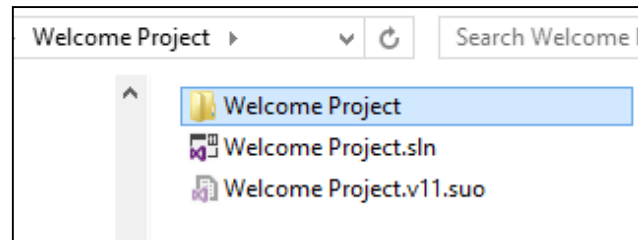
# Step 6<sub>a</sub>: Saving the Program

☀ To save your Program (Visual Studio Solution):

1. Select 'File' from the Visual Studio 2015 Main Menu...
2. Select 'Save All' to save all files (note: there is no general-use 'Save As').

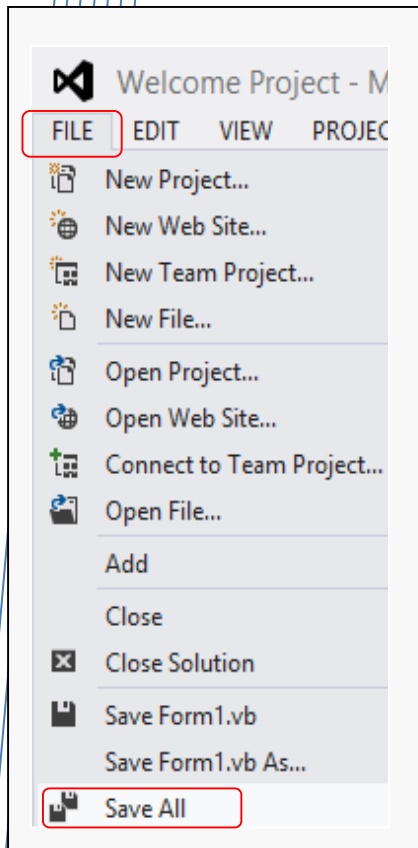
☀ To confirm, first check your **Visual Studio Projects Folder**:

- MyDocuments > Visual Studio 2015 > MyProjects > Welcome Project



- Here, you are in your 'Welcome Project' **Solution Folder**, and you will see :

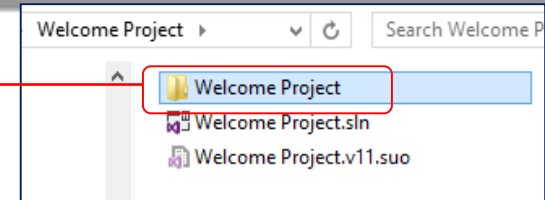
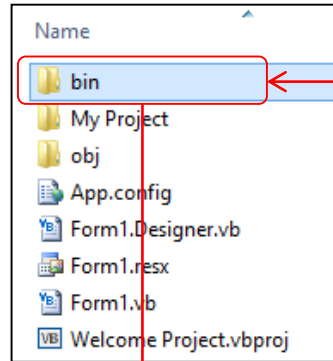
- The 'Welcome Project' folder is your **Project Folder**
  - Note: You have only 1 Project in this Solution.
- 'Welcome Project.sln' is your **Solution File**
  - (This is the icon you click to open your solution in VS 2015)



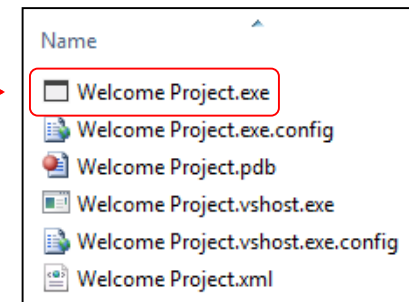
# Step 6<sub>6</sub>: Saving the Program

☀ Next, let's find your executable file ...

- ( = Welcome Project.exe )
- First, enter your **Project Folder**...

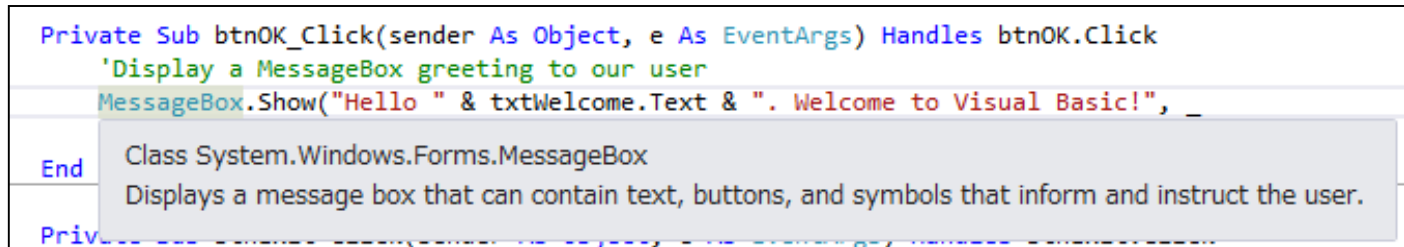


- Then, enter your Project's **bin folder** to view your exe file.
  - You may run your program directly by clicking this icon...
    - Note: your Project will NOT open.
  - Or, more conveniently, from within Visual Studio (as usual).



# Using Visual Studio Help

- **Visual Studio 2015** features an extensive set of **Help Tools**, including:
  - A Window-based Help System allowing you to view documentation;
  - An intelligent, programmable tool-tip based system called **Intellisense**



```
Private Sub btnOK_Click(sender As Object, e As EventArgs) Handles btnOK.Click
    'Display a MessageBox greeting to our user
    MessageBox.Show("Hello " & txtWelcome.Text & ". Welcome to Visual Basic!", _
End
Class System.Windows.Forms.MessageBox
    Displays a message box that can contain text, buttons, and symbols that inform and instruct the user.
Private Sub btnOK_Click(sender As Object, e As EventArgs) Handles btnOK.Click
```

- You will become familiar with Intellisense as you gain experience; however, be aware that you may access the **VS Help Window** in several ways:
  1. Through the **Visual Studio Main Menu (> Help > View Help)**:





***Thank You***