

Forum:[Hack Forums](#)

Thread: [\[Tutorial\]Beginners Guide to Vb.Net part 2\[Tutorial\]](#)

SpencerXZX - October 23, 2011, 5:17 pm

You can find part 1 [here](#):

Part 2:

Add a Button to your form.

Instead of double clicking the Button tool in the toolbox to add the control to the form, we'll explore another way to do it.

With your Form displayed in the Visual Basic Design environment, do the following:

- Click on the Button tool in the toolbox with the left hand mouse button, but click only once
- Move your mouse to a blank area of your form - the mouse pointer will turn into a cross
- Press and hold down the left mouse button
- Drag across the form with the button held down
- Let go of the mouse button when you're happy with the size
- A Button is drawn

You can use the above method to draw most of the controls onto the form - labels, buttons, textboxes, etc.

The Button control, just like all the other controls we've seen so far, has a list of properties. One of these properties is the Text property. At the moment, your button will say "Button 1". You can change that to anything you like.

- Click on the Button to highlight it
- Click on Text in the Property Box
- Click in the box next to the word "Text"
- Delete the word "Button 1"
- Type "Add two numbers"
- Click back on the Form

Now add a Textbox to your form using one of the methods outlined (either double-click, or draw).

The Font property of the Button has also been changed, here, in exactly the same way as we changed the Font property of the Label and Textbox previously.

To get our first look at the code window, double click your Button control. The code window will appear.

The part to concentrate on for the moment is where your cursor is flashing on and off. Because you double-clicked the Button control, the cursor will be flashing between the lines Private Sub and End Sub.

Here's the parts we're concentrating on, the parts in bold:

```
Code Private Sub Button1_Click(ByVal sender As System.Object, _  
ByVal e As System.EventArgs) _
```

Handles Button1.Click

End Sub

And here's what each part means:

Private:

Private means that no other part of the programme can see this code except for our button.

Sub

Short for Subroutine. The "Sub" word tells VB that some code follows, and that it needs to be executed.

Button1

This is the name of our button. You might think that we've just erased the word "Button1" when we changed the Text property, so why does VB insist that it's still called Button1? We'll, the Name property of the control is the important one. If you change the Name property, VB will change this button name for you.

_Click ():

This is something called an Event. In other words, when the button is clicked, the Click Event will fire, and the code we're going to write will be executed.

End Sub

The subroutine ends right here. This signifies the end of our code.

Don't worry if you don't understand all of that. It will become clearer later. Let's add our code. Click your mouse on the blank line after Private Sub Button1_Click, etc, but before End Sub. Type the following code:

```
Code Dim number1 As Integer
```

```
Dim number 2 As Integer
```

```
Dim answer As Integer
```

```
number1 = 3
```

```
number2 = 5
```

```
answer = number1 + number2
```

```
MsgBox (answer)
```

Before we explore what's happening here, save your work and then click Debug > Start Debugging from the Visual Basic Menu, or press F5 on your keyboard. This will launch your programme. Click the Button once. You should now get a messagebox with the number 8 in it.

Stop your programme, and return to the Design Environment. If you can't see your code, you can click the Tabs at the top of the Window.

Click the "Form1.vb [Design]" tab to see your Form.

OK, what happened there? Well, what happened is we've just written a programme to add two numbers together, and we displayed the result by using a Message Box - your very first real programme! But let's break that code down a little more.

```
Code Dim number1 As Integer
```

```
Dim number 2 As Integer
```

```
Dim answer As Integer
```

```
number1 = 3
```

```
number2 = 5
```

```
answer = number1 + number2
```

MsgBox (answer)

- First, we started with the Dim word, indicating to Visual Basic that we wanted to set up a variable
- Then we gave the variable a name (number1)
- Next, we "told" VB that what is going inside the variable is a number (As Integer)
- Two more variables were set up in the same way, number2 and answer.

After setting up the three variables, here's what we did:

- Told Visual Basic that what is going into the first variable was the number 3, and what is going into the second variable was the number 5. To put something into a variable, you use the equals (=) sign. But it's not really an equals sign – it's an assignment operator. You are assigning the value of 3 to the variable called number1.

Code number1 = 3

number2 = 5

The next part is a little bit more complicated, but not too complicated. What we wanted to do was to add two numbers together. So we said:

Code number1 + number2

Visual Basic already knows how to add up: all we need to do is "tell" it to add up. We do the "telling" in the traditional, mathematical way - with the plus sign (+). What Visual Basic will do is to look at what we've stored inside number1, and look at what's inside number2. It sees the 3, sees the five, and also sees the plus sign. Then Visual Basic adds them up for you. Except we also did something else. We said to Visual Basic "When you've finished adding up the two variables number1 and number2, store the result in that other variable we set up, which is called answer."

So, the whole line:

Code answer = number1 + number2

means:

"Add the variable called number1 to the variable called number2. Then store the result in the variable called answer."

Think of it as working from the right-hand side of the equals sign first. Then when you have the answer, assign it the variable on the left of the equals sign. The final part of the programme used Visual Basic's in-built Message Box. We'll learn more about the Message Box later. For now, think of it as a handy way to display results. Message boxes are quite handy when you want to display the result of some code. But we have a textbox on the form, and we might as well use that.

So delete the line: MsgBox answer. Type the word Textbox1, then type a full stop(period). You should see a drop-down box appear. This is a list of the Properties and Methods that the Textbox can use.

Scroll down until you see the word "Text". Double click the Text property and the drop-down box will disappear. (This drop-down box is known as IntelliSense, and is very handy. It means you can just select a property or method from the list without having to type anything.) The Text property you have chosen is the same Text property that you set from the Properties Window earlier. Here, we're setting the property with our code; before, we set it at design time. But the result is the same – the Text property of the textbox will be set to a value of our choosing. To set a value, type an equals sign, then type a value for the Text property. We want the contents of the variable called answer to appear in the textbox. So the rest of the code is just this:

Code Textbox1.Text = answer

Run your code again, and press the Button on the form. You should see the number 8 appear in

the textbox.

OK, time for your first exercises. They're not too painful, and hopefully they'll give you a better idea of what variables are. And besides, programming is about doing, not talking. So off we go!

Exercise

Delete the values "3" and "5" and replace them with numbers of your own Exercise

Delete the plus sign in between number1 and number2, and replace them with each of the following in turn

- (the minus sign)

* (the multiplication sign in VB is the asterisk sign)

/ (the divide sign in VB is the forward slash)

Exercise

Set up another Integer variable. Give it the name number3. Assign a value of 10 to this new variable. Multiply the value of your new variable by the variable called answer. Display the result in your textbox.

Another way to assign values to variables is when you first set them up. You can do this:

```
Code Dim number3 As Integer = 10
```

This is exactly the same as saying:

```
Dim number3 As Integer
```

```
number3 = 10
```

It's up to you which method you use. But the objective is the same – to assign a value to a variable.

String Variables

So we've just learnt something about variables, what they are and how to set one up. We learnt about the word "integer", and that integer variables held numbers. But what if we don't want numbers? After all, our first Form asked users to type in their First Name and Last Name. Names are not numbers, so what do we do then? Well that's where Strings come in.

What is a String?

Actually a string is nothing more than text. And if we want Visual Basic to store text we need to use the word "String". To set up a variable to hold text we need to use As String and not As Integer. If the information we want to store in our variables is a First Name and a Last Name, we can set up two variables like this.

```
Code Dim FirstName As String
```

```
Dim LastName As String
```

Again we've started with the Dim word. Then we've called the first variable FirstName.

Finally, we've ended the line by telling Visual Basic that we want to store text in the variable - As String.

So we've set up the variables. But there is nothing stored in them yet. We store something in a variable with the equals sign (=). Let's store a first name and a last name in them:

```
Code FirstName = "Bill"
```

```
LastName = "Gates"
```

Here, we said to Visual Basic "Store the word 'Bill' into the variable FirstName and store the word

'Gates' into the variable called LastName. But pay attention to the quotation marks surrounding the two words. We didn't say Bill, we said "Bill". Visual Basic needs the two double quotation marks before it can identify your text, your String.

So remember: if you're storing text in a variable, don't forget the quotation marks! To test all this out, add a new Button to your Form. Set the Text property of the Button to "String Test".

Double click your new button, and add the following code:

```
Code Dim FirstName As String
Dim LastName As String
Dim FullName As String
FirstName = "Bill"
LastName = "Gates"
FullName = FirstName & LastName
TextBox1.Text = FullName
```

There's a line there that needs explaining:

```
Code FullName = FirstName & LastName
```

In the two lines of code above that one, we stored the string "Bill" and the string "Gates" into two variables. What we're doing now is joining those two variables together. We do this with the ampersand symbol (&). The ampersand is used to join strings together. It's called Concatenation.

Once Visual Basic has joined the two strings together (or concatenated them), we're saying "store the result in the variable called FullName". After that, we tell VB to display the result in our Textbox. So, once you've typed the code, start your programme and test it out. Once the programme is running, click the Button and see what happens.

The textbox has displayed the text stored in our variables, "Bill" and "Gates". We joined them together with the ampersand (&). But as you can see, the two words are actually joined as one. We can add a bit of space between the two words by using another ampersand. Change this line:

```
Code FullName = FirstName & LastName
to this:
```

```
Code FullName = FirstName & " " & LastName
```

What we're saying here is join this lot together: the variable called FirstName and a single blank space and the variable called LastName. When you've finished concatenating it all, store the result in the variable FullName. Notice that we don't surround FirstName and LastName with quotation marks. This is because these two are already string variables; we stored "Bill" into FirstName and "Gates" into LastName. So VB already knows that they are text.

Exercise

Remove one of the ampersand symbols (&) from this line in your code:

```
Code FullName = FirstName & " " & LastName
```

Move your cursor down a line or two. You should see that part of your code has a wiggly blue line under it.

VB is telling you that it has problems with this line of code. If you hold your mouse over the wiggly blue line, VB tries to provide an explanation:

The explanations VB provides are sometimes enigmatic. But you will know that there is a problem. If you run the code, you'll get an error message. Click the NO button. Put the ampersand back in,

and all will be well.

Exercise

Amend your code so that the textbox reads Gates Bill when the button is clicked.

Exercise

Add another string variable to your code. The variable should hold a middle name. Display the first name, the middle name and the last name in the textbox.

Points to remember:

- Your variable names cannot include spaces. So MiddleName would be all right, but Middle Name will get you an error message
- When you're putting text into your new variable, don't forget the two double quotes
- Remember to put in enough ampersands in your FullName = line of code

Assigning values from Textboxes to your Variables Instead of putting direct text into your variables, such as "Bill" or "Gates", you can get text

from a textbox and put that straight into your variables. We'll see how that's done now. First, do this:

- Add a new textbox to your form
- With the textbox selected, locate the Name property in the Properties area:

The current value of the Name property is Textbox2. This is not terribly descriptive. Delete this name and enter txtLastName. Scroll down and locate the Text property. Delete the default text, and just leave it blank.

Click on your first textbox to select it. Change the Name property from Textbox1 to txtFirstName.

What we've done is to give the two textboxes more descriptive names. This will help us to remember what is meant to go in them. Unfortunately, if you view your code (click the Form1.vb tab at the top, or press F7 on your keyboard), you'll see that the blue wiggly lines have returned (though the 2010 edition of the software may actually change the name in the code as well, so you won't see the wiggly blue lines.)

If you hold your cursor over the Textbox1, you'll see a message. It's displaying this message because you changed the name of your Textbox1. You now no longer have a textbox with this name. Change Textbox1 into txtFirstName and the wiggly lines will go away. (Change it in your Button1 code as well.) Your code should now read:

```
Code txtFirstName.Text = FullName
```

Run your programme again. If you see any error messages, stop the programme and look for the wiggly lines in your code.

We'll now change our code slightly, and make use of the second textbox. You'll see how to get at the text that a user enters.

Locate these two lines of code

```
Code FirstName = "Bill"
```

```
LastName = "Gates"
```

Change them to this

```
Code FirstName = txtFirstName.Text
```

```
LastName = txtLastName.Text
```

Remember: the equals (=) sign assigns things: Whatever is on the right of the equals sign gets assigned to whatever is on the left. What we're doing now is assigning the text from the textboxes directly into the two variables.

Amend your code slightly so that the WholeName is now displayed in a message box. Your code should now be this:

```
Code Dim FirstName As String
Dim LastName As String
Dim WholeName As String
FirstName = txtFirstName.Text
LastName = txtLastName.Text
WholeName = FirstName & " " & LastName
MsgBox(WholeName)
```

Run your programme. Enter "Bill" in the first textbox, and "Gates" in the second textbox. Then click your "String Test" button.

Before we changed the code, we were putting a person's name straight in to the variable FirstName

```
Code FirstName = "Bill"
```

But what we really want to do is get a person's name directly from the textbox. This will make life a whole lot easier for us. After all, not everybody is called Bill Gates! In the line `FirstName = txtFirstName.Text` that is what we're doing - getting the name directly from the textbox. What we're saying to Visual Basic is this:

- Look for a Textbox that has the Name `txtFirstName`
- Locate the Text property of the Textbox that has the Name `txtFirstName`
- Read whatever this Text property is
- Put this Text property into the variable `FirstName`

And that's all there is too reading values from a textbox - just access its Text property, and then pop it into a variable.

Exercise:

- Add a third textbox to your form
- Change its Name property to `txtWholeName`
- Add labels to your form identifying each textbox (A quick way to add more labels is to use the toolbox to add one label. Then right click on that label. Choose Copy from the menu. Right click on the form, and select Paste.)
- Write code so that when the "String Test" button is clicked, the whole of the person's name is displayed in your new textbox.

This tutorial is unfinished, I will be updating this constantly with more and more information. Enjoy.

YoloFaggens - December 18, 2013, 7:18 am

Why isn't this pumped yet? this is helpful.
just look at this sweetness.

thanks for the tut xd