

What is a Sub Procedure?

DocumentID : What_is_a_Sub.PDF
Author : Michele Harris
Date : 07-02-2009
Version : 1.2

Contents

What is a Sub Procedure?	2
Syntax of a Sub Procedure.....	2
Basic Sub Procedure Example	3
Creating Your Own Sub Procedure	3
Calling Your Own Sub Procedure.....	6

What is a Sub Procedure?

In Visual Studio, Procedures are employed to organize a group of related statements to perform a specific task.

Sub procedures typically receive or process data, change properties of an object, or display an output. Unlike functions, Sub procedures do not return a new value associated with their names. If a value is already present when entering the procedure, however, sub procedures are capable of returning a modified value.

Sub routines can be private—meaning they're associated with a particular object or event, and are not explicitly used by other objects in your code—or they can be public. Variables and statements in public sub procedures, conversely, can be utilized by other objects and events in the code.

Syntax of a Sub Procedure

A Sub Procedure's syntax resembles the following:

```
Sub ProcedureName([arguments])
```

```
    procedure statements
```

```
End Sub
```

Above is a Sub Procedure in its simplest, elemental form. You will likely encounter more complex Sub Procedures, but here are its fundamental building blocks. The ProcedureName, quite simply, is the name of the Procedure you're calling. Arguments are list of optional comments used by the Sub Procedure. Multiple arguments are separated by commas. The Procedure statements are simply one or more lines of code that perform all the procedure's work.

Basic Sub Procedure Example

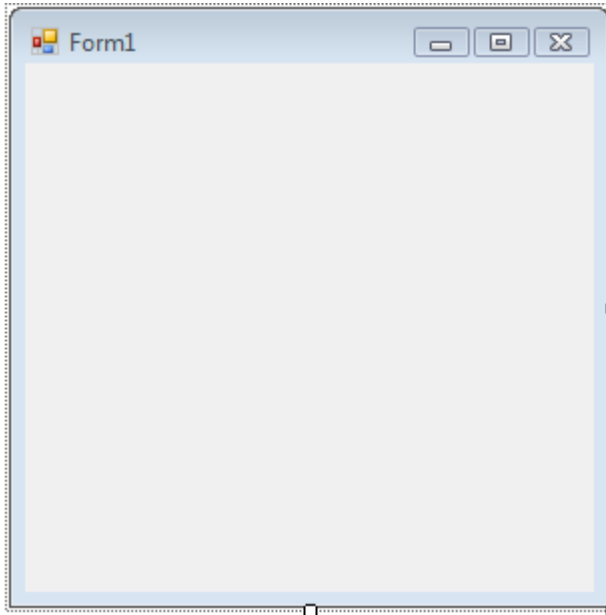
Here's an example of one of the simplest possible Sub Procedures following the same syntax I showed above.

```
Sub Goodbye ()  
    End  
End Sub
```

Here "Goodbye" serves as the Procedure's name, the parentheses take the place of any optional arguments (which we will forgo in this example), and "End" functions as the statement that powers this procedure. Any time this procedure is run, the program will end. Of course, to tell Visual Basic we're finished with this procedure, it's necessary to add "End Sub" after our statement.

Creating Your Own Sub Procedure

Creating a basic Sub Procedure is extremely simple. Open a new Windows Form Project in Visual Basic called "My First Sub". You should see a familiar form appear, much like the one below.



Place your mouse cursor over the form, and double click. This should take you to the Code Editor, where you should find the following procedure:

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load  
End Sub
```

Believe it or not, this alone, is a Sub Procedure! Since it lacks any sort of statements, it can't really do anything at this point. If you don't believe me, try debugging it by pressing F5—nothing happens. Yet already it has a name (Form1_Load) tied to a particular event (the Form's loading), following with the necessary arguments.

However, let's see if we can get this procedure to do something.

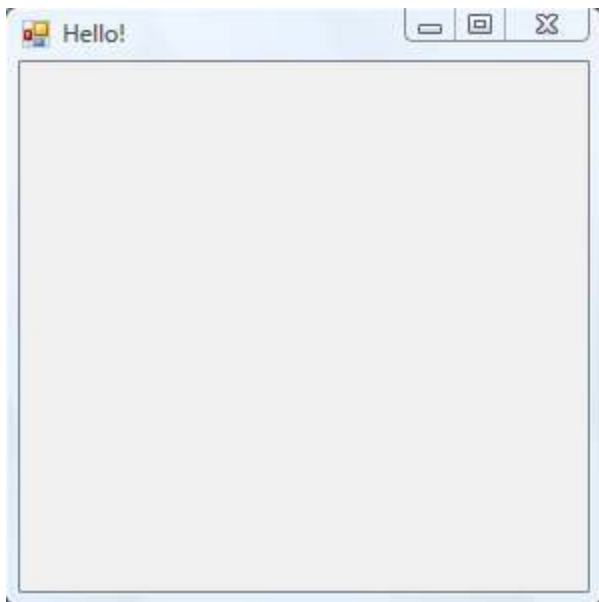
Place your cursor right after "Handles MyBase.Load", and hit return. Enter the following statement on that new line:

```
Me.Text = "Hello!"
```

Your screen should now look like this.

```
Public Class Form1  
  
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load  
        Me.Text = "Hello!"  
    End Sub  
End Class
```

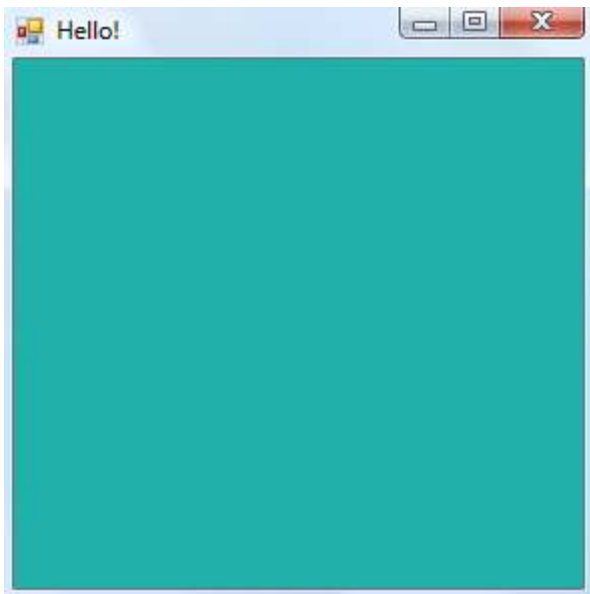
Now press F5 to debug your form. You'll notice that "Hello!" now appears in the form's text!



Let's add another statement! Click the stop button to exit the debug mode, and double click on the form to view the Code Editor. Place your cursor right after "Hello!", and press return. Enter the following statement on a new line.

```
Me.BackColor = Color.LightSeaGreen
```

Now press F5 to start debugging again. Notice anything different?

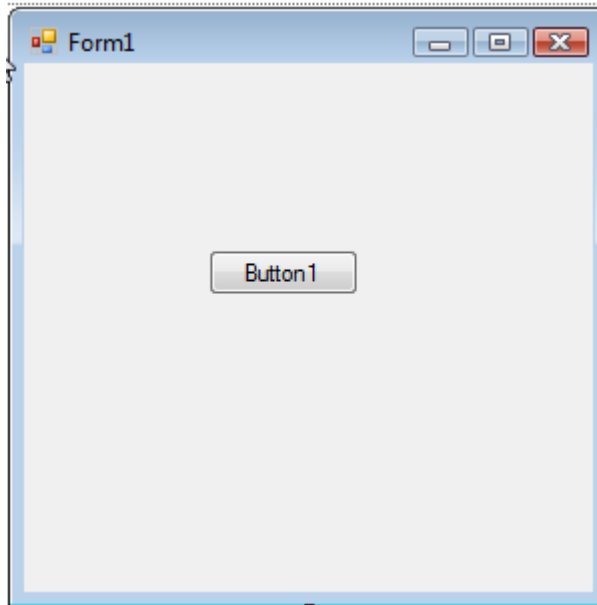


With one short line of code in a Sub Procedure, we were able to transform the entire background of this form to Light Sea Green.

Calling Your Own Sub Procedure

One of the benefits of Visual Studio is that you can use Sub Procedures elsewhere in your code, merely by referencing their names. Such a process is referred to as *calling* a Sub Procedure. Calling a Sub Procedure allows you to utilize that procedure's code without retyping it elsewhere. You can even (and will often) call a Sub Procedure within another procedure!

Go back to the design view of the "My First Sub" form you created. Hover your mouse over the ToolBox Bar on the left side of the screen until the menu appears. Double click the button option to add a new button onto your form.



Double anywhere on the form (except on the button), which will take you to the Code Editor. On the line below the “End Sub” statement, type the following procedure.

```
Sub Goodbye ()  
    End  
End Sub
```

Your screen should display the following.

```
Public Class Form1  
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load  
        Me.Text = "Hello!"  
        Me.BackColor = Color.LightSeaGreen  
    End Sub  
    Sub Goodbye()  
        End  
    End Sub  
End Class
```

Now, go back to the Design View of your form. Double click your newly created button, which will take you to the following screen.

```
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Me.Text = "Hello!"
        Me.BackColor = Color.LightSeaGreen
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    End Sub

    Sub Goodbye ()
        End
    End Sub
End Class
```

Place your cursor right after “Handles Button1.Click” in the Button1_Click Sub

Procedure. Hit return, and enter the following code in the new line:

```
Goodbye ()
```

Your screen should appear as below.

```
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Me.Text = "Hello!"
        Me.BackColor = Color.LightSeaGreen
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Goodbye ()
    End Sub

    Sub Goodbye ()
        End
    End Sub
End Class
```

Now press F5 to start Debugging. This time, you’ll notice that your new button appears on the form. Click the button, and the program closes. Congratulations! You’ve successfully called a Sub Procedure!