Stop Watch Code

Stopwatch Application - Writing Code

All that's left to do is write code for the application. We write code for every event a response is needed for. In this application, there are three such events: clicking on each of the buttons.

1. Double-click anywhere on the form to open the code window. Or, select 'View Code' from the project window. Or, press <F7> while the form is active.

2. At the very top of the code window, type this line (to enforce explicit variable declaration):

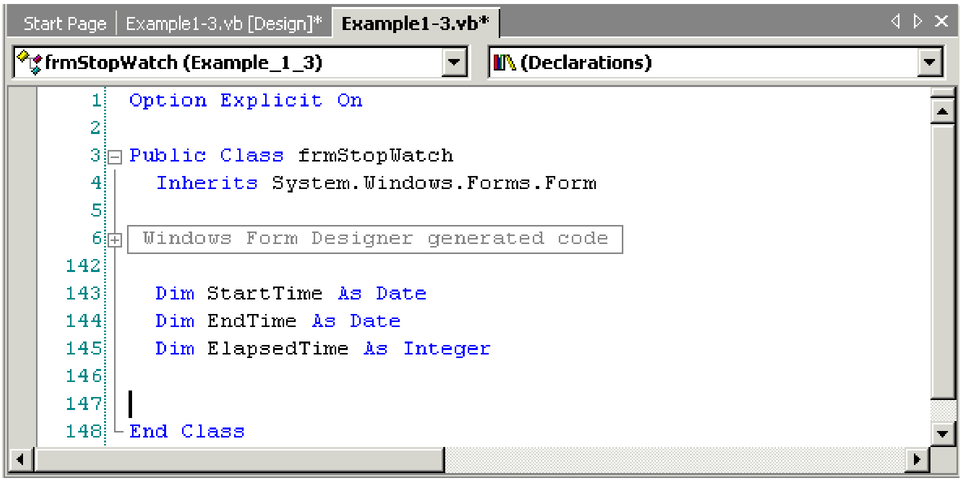
```
Option Explicit On
```

3. Under the line marked **Windows Form Designer generated code** declare three form level variables:

```
Dim StartTime As Date
Dim EndTime As Date

Dim ElapsedTime As Integer
```

This establishes **StartTime**, **EndTime**, and **ElapsedTime** as variables with form level scope. At this point, the Code window should look like this:



```
Option Explicit On

Public Class frmStopWatch
  Inherits System.Windows.Forms.Form

Windows Form Designer generated code

  Dim StartTime As Date
  Dim EndTime As Date
  Dim ElapsedTime As Integer

End Class
```

4. Select the **btnStart** object in the Object box of the code window. Choose **Click** from the procedure box. Or, try double-clicking the button control. Type the following code which begins the timing procedure. Note the **Sub** and **End Sub** statements are provided for you:

```
  Private Sub btnStart_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnStart.Click

    'Establish and print starting time

    StartTime = Now

    lblStart.Text = Format(StartTime, "hh:mm:ss")

    lblEnd.Text = ""

    lblElapsed.Text = ""

  End Sub
```

In this procedure, once the **Start Timing** button is clicked, we read the current time and print it in a label box. We also blank out the other label boxes. In the code above (and in all code in these notes), any line beginning with a single quote (') is a comment. You decide whether you want to type these lines or not. They are not needed for proper application operation.

5. Now, select the **btnEnd** object in the Object box. Choose **Click** from the procedure box. Add this code:

```
  Private Sub btnEnd_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEnd.Click

    'Find the ending time, compute the elapsed time
    'Put both values in label boxes

    EndTime = Now

    ElapsedTime = DateDiff(DateInterval.Second, StartTime, EndTime)

    lblEnd.Text = Format(EndTime, "hh:mm:ss")

    lblElapsed.Text = Format(ElapsedTime, "0")

  End Sub
```

Here, when the **End Timing** button is clicked, we read the current time (**End Time**), compute the elapsed time, and put both values in their corresponding label boxes.

6. Finally, select the **btnExit** object in the Object box. Choose **Click** from the procedure box. That button's Click event code:

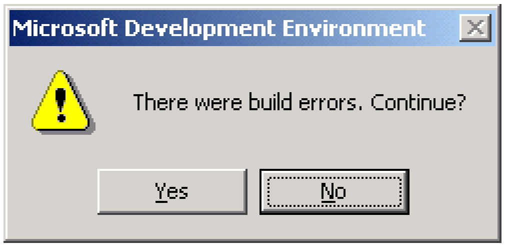
```
  Private Sub btnExit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnExit.Click
```

```
    Me.Close()
End Sub
```

This routine simply closes the form (identified by the keyword Me) once the **Exit** button is clicked. Did you notice as you typed in the code, how the Intellisense feature of Visual Basic .NET worked? Before trying to run the application, we need to take a brief, but important, interlude.

Running An Application (Startup Object)

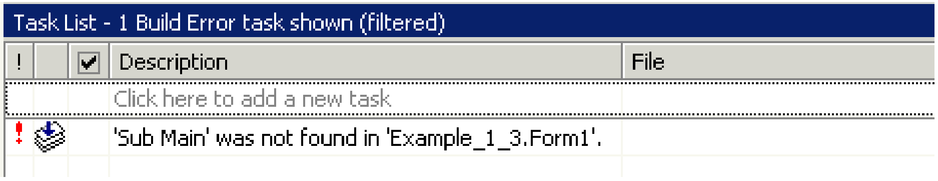
- Try to run your application by clicking the **Start** button on the toolbar, or by pressing **<F5>.** You will (most likely) see this error message:



Microsoft Development Environment

There were build errors. Continue?

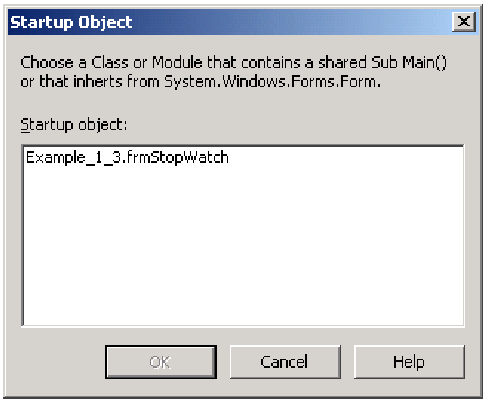
Select **No** and look in the **Task List** window for the build errors (errors that occur while trying to make your application work):



Task List - 1 Build Error task shown (filtered)

!		☑	Description	File
			Click here to add a new task	
!			'Sub Main' was not found in 'Example_1_3.Form1'.	

- What's going on here? This message essentially means that Visual Basic .NET cannot find the **startup object**, meaning the form it is supposed to display. When we rename a form (by setting its **Name** property) in a Visual Basic .NET application, we need to let the environment know we've done this. To solve the current problem, double-click the line with the **Sub Main** error message in the Task List window. You will see a dialog similar to this:

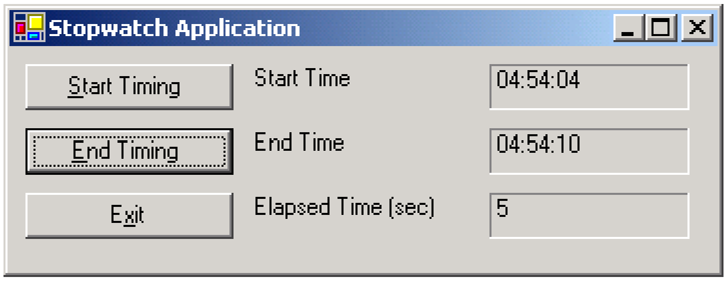


Choose a Class or Module that contains a shared Sub Main() or that inherts from System.Windows.Forms.Form.

Startup object:

Example_1_3.frmStopWatch

Select the **frmStopWatch** object and click **OK**. Now, try running the application again (click the **Start** button in the toolbar or press **<F5>**. The application should run just fine now. Try it out. Here's a short run I made:



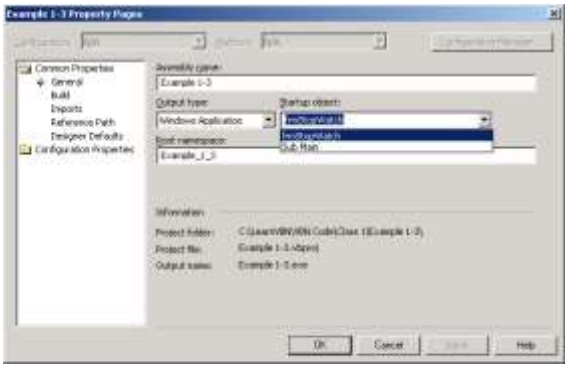
Start Time: 04:54:04

End Time: 04:54:10

Elapsed Time (sec): 5

If your application doesn't run, recheck to make sure the code is typed properly. Save your application.

- In every application you build with Visual Basic .NET, you will have to define the startup object.
- You can either do what we just did – define the startup object the first time you try to run an application or you can be proactive and define it before you run an application.
- To do this, go to the Solution Explorer window and right-click the project name. Select Properties from the drop down menu. You will see this dialog box:



Assembly name: Example 1-3

Output type: Windows Application

Startup object: frmStopWatch

Root namespace: Example_1_3

Information

Project folder: C:\LearnVBN\VBN Code\Class 1\Example 1-3\

Project file: Example 1-3.vbproj

Output name: Example 1-3.exe

The **Startup object** dropdown box allows you to select the desired form.

- If you have the time, some other things you may try with the **Stopwatch Application**:

A. Try changing the form color and the fonts used in the label boxes and buttons.

B. Notice you can press the 'End Timing' button before the 'Start Timing' button. This shouldn't be so. Change the application so you can't do this. And make it such that you can't press the 'Start Timing' until 'End Timing' has been pressed. Hint: Look at the button **Enabled** property.

C. Can you think of how you can continuously display the 'End Time' and 'Elapsed Time'? This is a little tricky because of the event-driven nature of Visual Basic .NET. Look at the **Timer** control. By setting the **Interval** property of this control to **1000** and the **Enabled** property to **True**, it will generate its own events (the **Tick** event) every one second. Put code similar to that in the **btnEnd_Click** event in the Timer control's Tick event and see what happens. Also, see the exercise at the end of the class for help on this one. The Timer control will not appear on the form, but in a 'tray' below the form. This happens because the Timer control has no user interface.