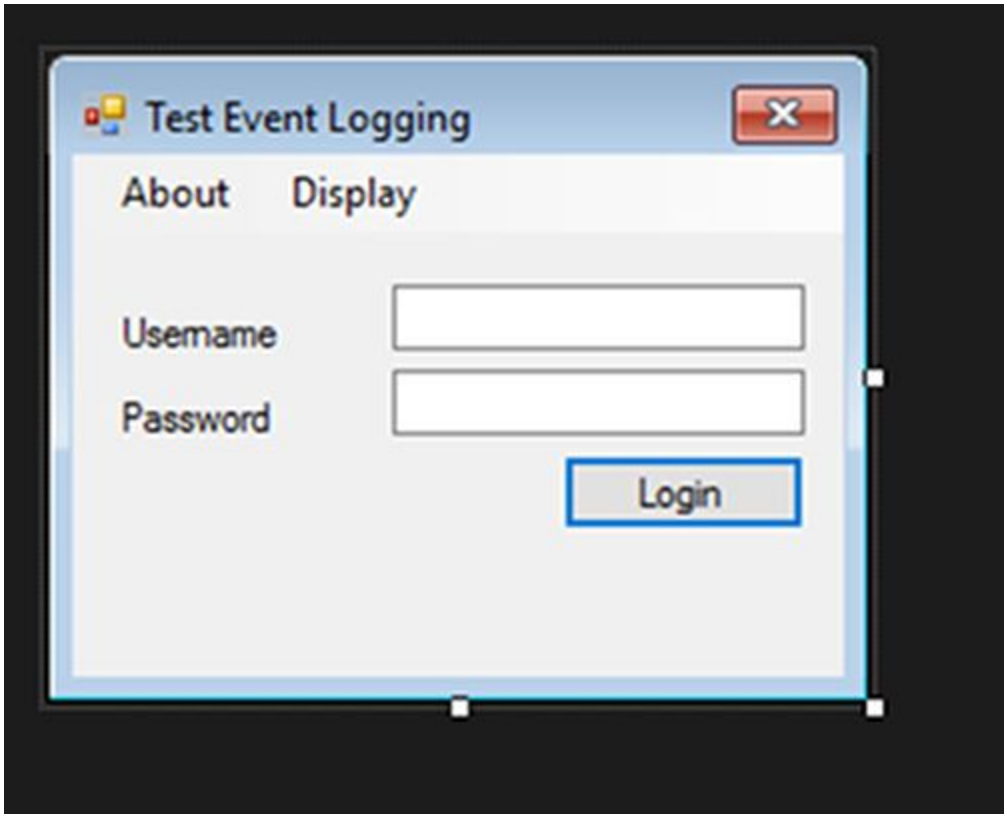
Event logs in VB.NET

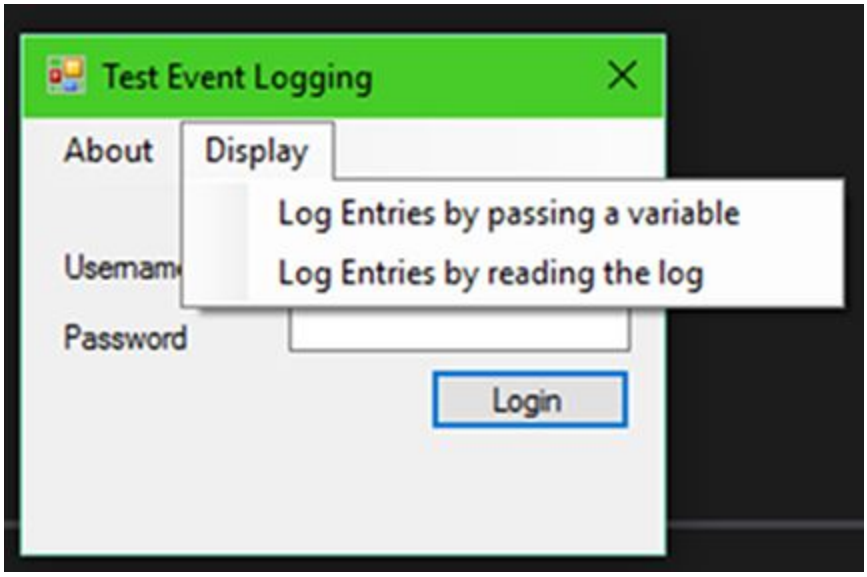
Reading and writing to a Windows Events Log of your own
http://www.thescarms.com/dotnet/EventLog.aspx
https://support.microsoft.com/en-us/kb/814564

This project consists of 3 forms and 1 module.

- The main login form(form1) containing a menu strip, error label, login button and username/password labels and textboxes.

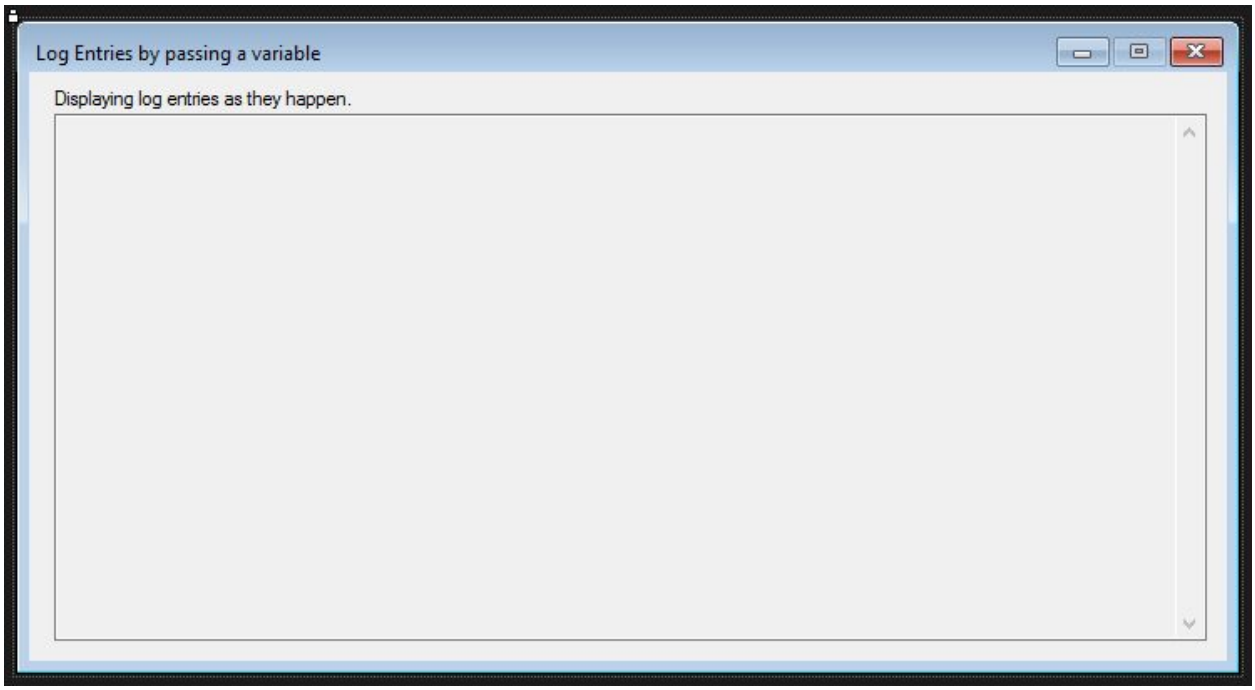


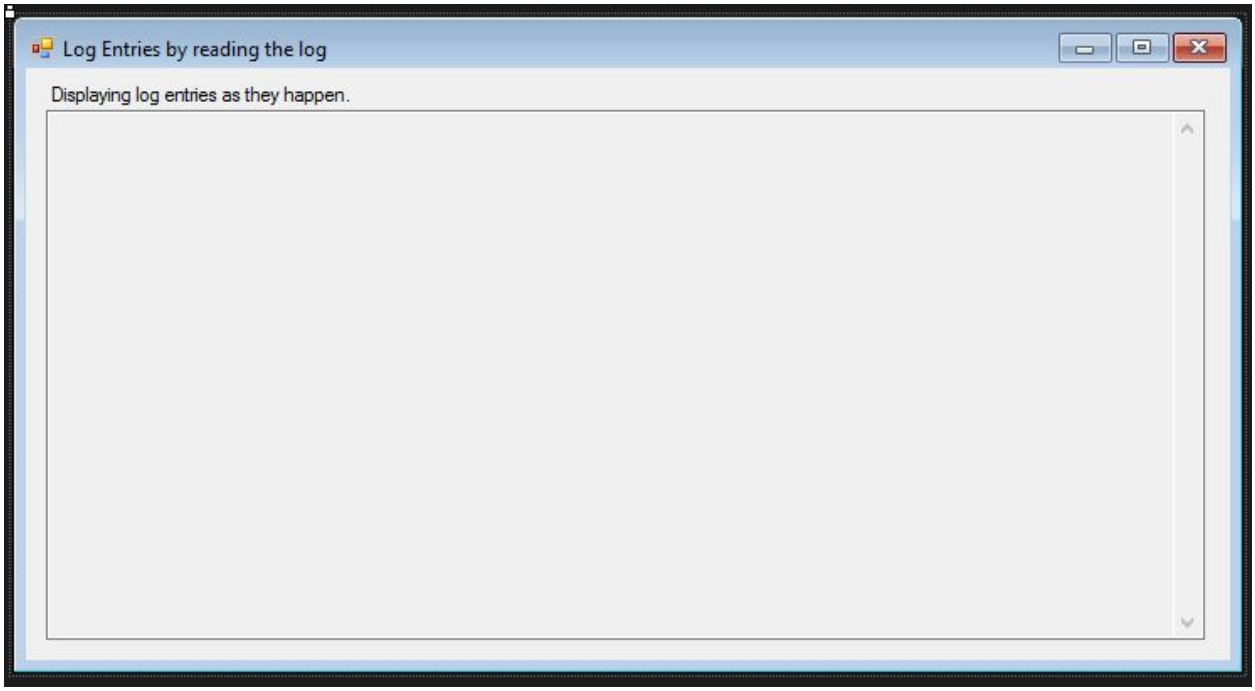
The menu strip contains:



You can ignore the About or use it as you wish. I had it linking to this document in my program.

- Two additional forms(form2, form3) that display a large textbox that will contain log data as submitted into the Windows Events Log.





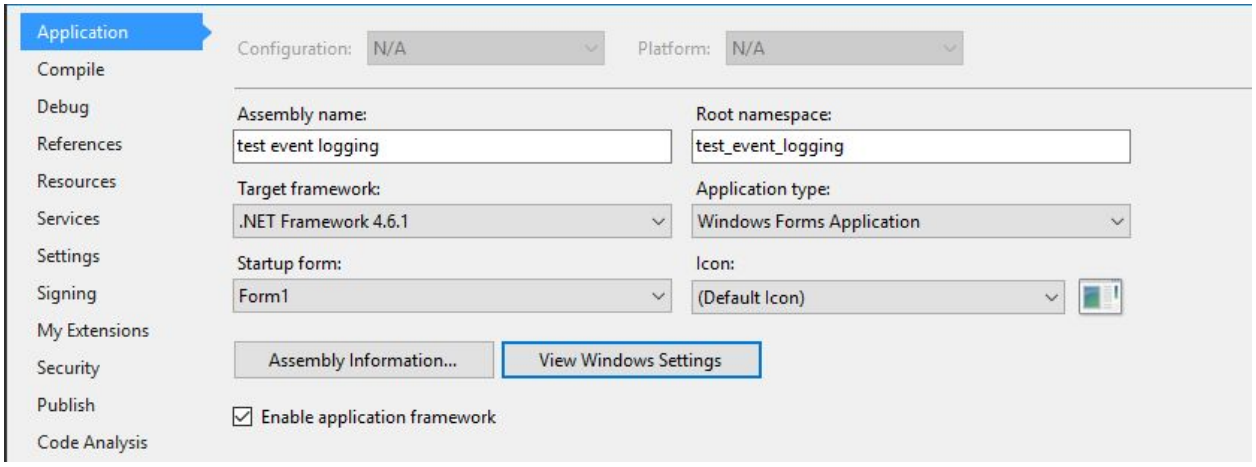
Once the forms contents are arranged, lock the forms and then modify some properties.

- Form1:
    - StartPosition: CenterScreen
    - FormBorderStyle: FixedSingle
    - Text: Test Event Logging
    - AcceptButton: Button1
- Form2, Form3:

        - MaximizeBox: False
        - TextBox1:
            - Text: “”
            - ReadOnly: True
            - ScrollBars: Vertical
            - TabStop: False
- Form2:
    - Text: Log Entries by passing a variable
- Form3:
    - Text: Log Entries by reading the log

Finally, for the startUp sub in module1, we have to have administrator permission to check some things in the EventLog object. So the code properties of the program are set to request that permission upon running.

Right click the top item in the Solution Explorer box and click Properties. This brings up the following image.



In here, click the View Windows Settings button. This brings up a window of text.

Change the “<requestedExecutionLevel” tag to this:

```
<requestedExecutionLevel  level="requireAdministrator"
uiAccess="false" />
```

Save and save often.

Now onto the code !

Within module1 we will place the following code above the “Module Module1” block. This interfaces with the EventLog Object.

```
Option Explicit On
'Option Strict On ' readFromLog sub is using late binding

Imports System
Imports System.Diagnostics ' Allows access to the EventLog
Object
```

```
Imports System.Threading
```

Option Strict does not allow late binding and that is needed within the sub readFromLog( ).

Within the Module Module1 block we create 5 subs.

- A startUp sub called on program startup within Form1.
  - It checks the log on the computer to see if it exists
  - Sets the options in the EventLog object
  - Links an event handler to fire onto our custom sub OnEntryWritten, after a write event from the EventLog object. It fires on its own in another processing thread.
- A writeToLog sub to write into the Windows Events Log.
- A readFromLog sub to read all the existing entries.
- OnEntryWritten for processing data after the EventLog wrote something. Runs in a separate thread from the UI.
- A deleteLog sub to clear and delete the log we created on program shutdown

```
Public Sub startUp
```

```
Public Sub startUp()

        ' Create a new log.
        '
        ' A Security Exception will fire if I'm not running with administrator privileges when checking EventLog.SourceExists("MyNewSource")
        ' This is why the program is set to elevate when executed, http://stackoverflow.com/questions/3080725/debug-a-program-that-needs-administrator-rights-under-windows-7
        '
https://msdn.microsoft.com/en-us/library/6s7642se(v=vs.110).aspx
        If Not EventLog.SourceExists("MyNewSource") Then
            EventLog.CreateEventSource("MyNewSource", "MyNewLog")
        End If

        ' Set a few options
        With myLog
            .Source = "MyNewSource" ' Look for this name within the Windows Event Logs. In Windows 10 it was located under Applications and Services Logs
            .Log = "MyNewLog"
            .EnableRaisingEvents = True
        End With

        ' Add an event to fire when an entry is written. Shows up in Form2 textbox1
```

```
        AddHandler myLog.EntryWritten, AddressOf OnEntryWritten


End Sub
```

```
Public Sub writeToLog
```

```
' Purpose of this module, to write something to the event logs
    Public Sub writeToLog(msg As String, Optional ByVal errorType
As String = "information") '
https://msdn.microsoft.com/en-us/library/system.diagnostics.eventlo
gentrytype(v=vs.110).aspx

        ' Second parameter options:
        '   error, warning, information, failureaudit, successaudit
        ' default: information

        If msg <> "" Then
            Select Case errorType
                Case "information"
                    myLog.WriteEntry(msg,
EventLogEntryType.Information)
                Case "error"
                    myLog.WriteEntry(msg, EventLogEntryType.Error)
                Case "warning"
                    myLog.WriteEntry(msg,
EventLogEntryType.Warning)
                Case "failureaudit"
                    myLog.WriteEntry(msg,
EventLogEntryType.FailureAudit)
                Case "successaudit"
                    myLog.WriteEntry(msg,
EventLogEntryType.SuccessAudit)
            End Select
        End If
End Sub
```

The writeToLog sub takes 2 parameters; a text message and an event alert level. The sub has the options for the latter.

```
' Second parameter options:
        '   error, warning, information, failureaudit, successaudit
        ' default: information
```

If unspecified, the default is just plain Information.

Something I want to relate before going further.

The program has two different methods of relaying the data from the writeToLog() sub.

1. One branch is the Handler being fired. It fires and passes the data into OnEntryWritten and a module1 variable within, upon

completion. Then the Form1 timer tick will read that variable and display it on Form2.TextBox.Text. This round about method is because the Handler firing is on a different processing thread that the main thread, which also handles the UI. So I cannot directly update a textbox from the OnEntryWritten sub. Only set variables for the main thread to use.

2. The second branch is also fired from the Form1 timer tick. It makes use of the EventLog object to read the previously written log entries and updates Form3.TextBox.Text, within the sub readFromLog().

```
Public Sub readFromLog
```

```
Public Sub readFromLog()

        ' Dim List to hold the data and then we can reverse the
list so the newest events are at the top
        Dim dataList As New List(Of String)
        Dim tmpStr As String

        For Each entry In myLog.Entries
            With entry
                tmpStr = entry.TimeWritten + ": " + entry.Message +
" - " + entry.Source + vbCr
                dataList.Add(tmpStr)
            End With
        Next

        Form3.TextBox1.Text = String.Join(vbCrLf,
dataList.ToArray.Reverse)

End Sub
```

Rather than reading one event at a time like the write sub, this will pull all the information at once. From there it is sorted in reverse within the list. This way the most recent entry is at the top of the textbox(no scrolling!). Then we apply it to Form3.TextBox1.Text.

```
Public Sub OnEntryWritten
```

```
' Handy function to display the log entry
    ' Fired from the startUp sub with the "AddHandler
myLog.EntryWritten, AddressOf OnEntryWritten"
    Public Sub OnEntryWritten(ByVal [source] As Object, ByVal e As
EntryWrittenEventArgs)

        ' This collects the data into a variable for pickup by the
Form1 timer function
        ' This is needed because the AddHandler makes another
thread which cannot change the UI thread directly.
        ' This allows an update to the UI after the background
```

```
thread is finished.

        ' Add a date/time stamp to make it wasy to read on the
textbox tack on the error message that was passed to Windows Event
Logging
        newData = DateTime.Now.ToString("F") + ": " +
e.Entry.Message

End Sub
```

As stated above, this is called upon after a write to a log. The variable newData contains the information to pass into Form2.

```
Public Sub deleteLog
```

```
' This will clear and then delete the log we created.
    Public Sub deleteLog()

        ' Delete your new log.

        ' First check if it still exists
        If EventLog.SourceExists("MyNewSource") Then
            myLog.Clear()
            EventLog.Delete("MyNewLog") ' This is the
recommendation of the VB.Net IDE. Different from referring article
- myLog.Delete("MyNewLog")
        End If
End Sub
```

This will clear all log entries and then delete the log from the Windows Events Log system(I clean up after myself for this demo).

```
Form1
```

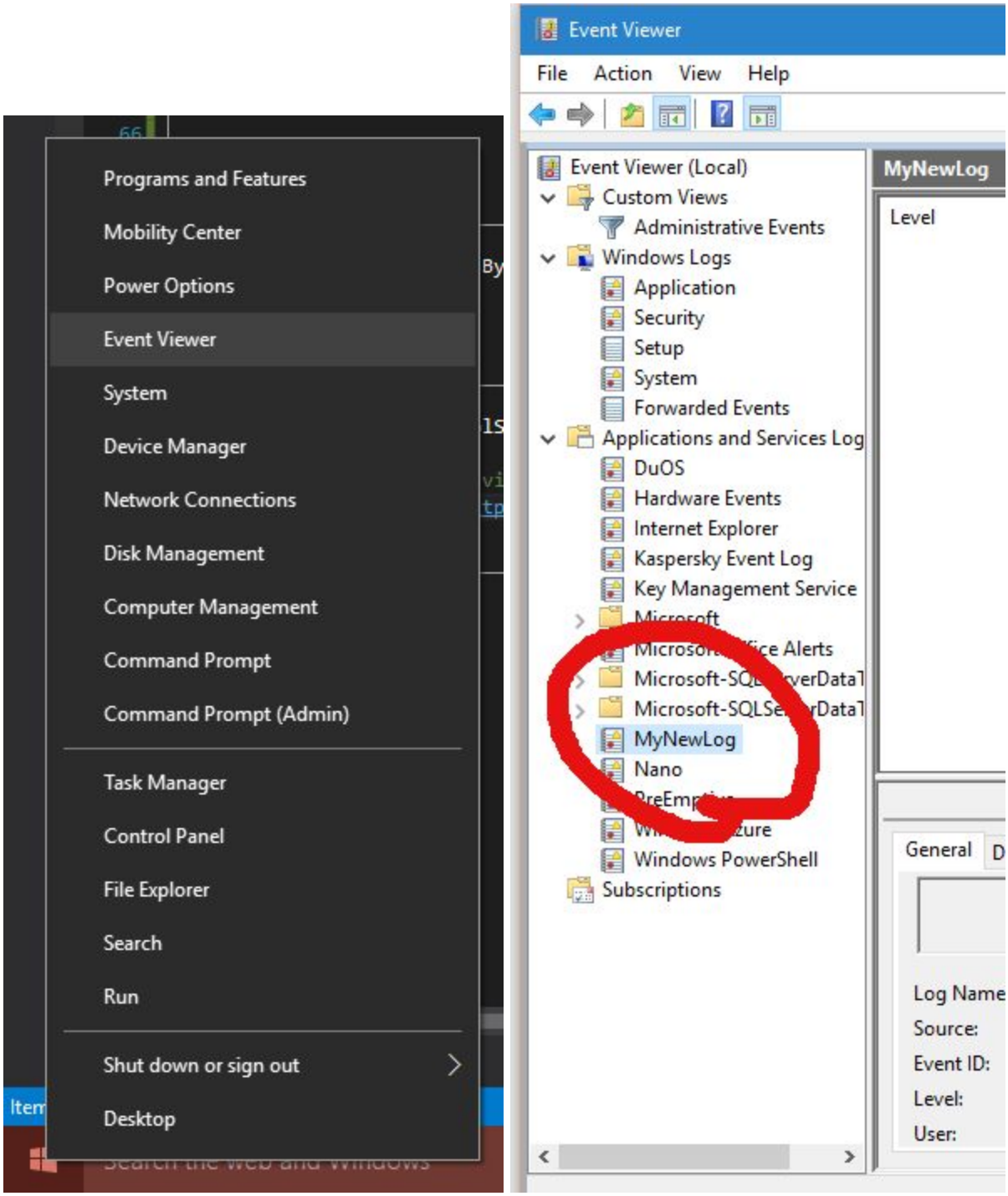
```
Public Class Form1

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

        ' Make sure the text boxes have something to compare
        If TextBox1.Text <> "" And TextBox2.Text <> "" Then
            ' Hard set username and password - Does it match ?
            If TextBox1.Text = "username" And TextBox2.Text =
"password" Then
                ' clear the error label if it was used.
                Label1.Text = ""

                ' Login to whatever ....

                ' Check if login was good and
                ' Note Success to log
```

```
                writeToLog("User logged in", "information")

            Else
                ' Tell the user that the login was incorrect
                Label1.Text = "Error. Incorrect username/password."

                ' Write to log the infraction
                writeToLog("Incorrect login", "error")

            End If
        End If

    End Sub

    Private Sub LogEntriesToolStripMenuItem_Click(sender As Object,
e As EventArgs) Handles LogEntriesToolStripMenuItem.Click

        Form2.Visible = True

    End Sub

    Private Sub Form1_FormClosing(sender As Object, e As
FormClosingEventArgs) Handles Me.FormClosing

        Module1.deleteLog()

    End Sub

    Private Sub Form1_Load(sender As Object, e As EventArgs)
Handles MyBase.Load

        ' Setup the module1 code for use on first run of our code
        Module1.startUp()

        Timer1.Start()

    End Sub

    Private Sub Timer1_Tick(sender As Object, e As EventArgs)
Handles Timer1.Tick
        ' Check for new data to display

        ' We can use the data brought over from module1 OR retrieve
it from the Windows Events Log(to prove it was written).

        ' From Module1, display on Form2
        If Module1.newData <> "" Then
            Form2.TextBox1.Text += Module1.newData + vbCrLf ' Add
to the textbox
            Module1.newData = "" ' empty the variable for the next
timer tick

            ' The passed Module1.newData variable allows us to know
```

```
when the data is different thus to update even Form3. Or we can
compare against a Static variable
            ' From Events Log, display on Form3
' or we can just blindly update Form3.TextBox.Text on every tick,
outside of this IF block, even when we do not need to.
            readFromLog()
        End If



    End Sub

    Private Sub
LogEntriesByReadingLogToolStripMenuItem_Click(sender As Object, e
As EventArgs) Handles LogEntriesByReadingLogToolStripMenuItem.Click

        Form3.Show()

    End Sub
End Class
```

This should all be explanatory. We are just interfacing everything to the module1 subs. Button1 and Timer1 are the subs to understand.

The Events Logs are found in the following location:


Programs and Features
Mobility Center
Power Options
Event Viewer
System
Device Manager
Network Connections
Disk Management
Computer Management
Command Prompt
Command Prompt (Admin)
Task Manager
Control Panel
File Explorer
Search
Run
Shut down or sign out
Desktop
Event Viewer
File Action View Help
Event Viewer (Local)
Custom Views
Administrative Events
Windows Logs
Application
Security
Setup
System
Forwarded Events
Applications and Services Log
DuOS
Hardware Events
Internet Explorer
Kaspersky Event Log
Key Management Service
MyNewLog
Nano
Windows PowerShell
Subscriptions
MyNewLog
Level
General
Log Name
Source:
Event ID:
Level:
User: