.NET

LAB MANUAL

PART II

WEB FORMS

ASP.Net provides two types of programming models:

- **Web Forms** - this enables you to create the user interface and the application logic that would be applied to various components of the user interface.
- **WCF Services** - this enables you to remote access some server-side functionalities.

Web Forms

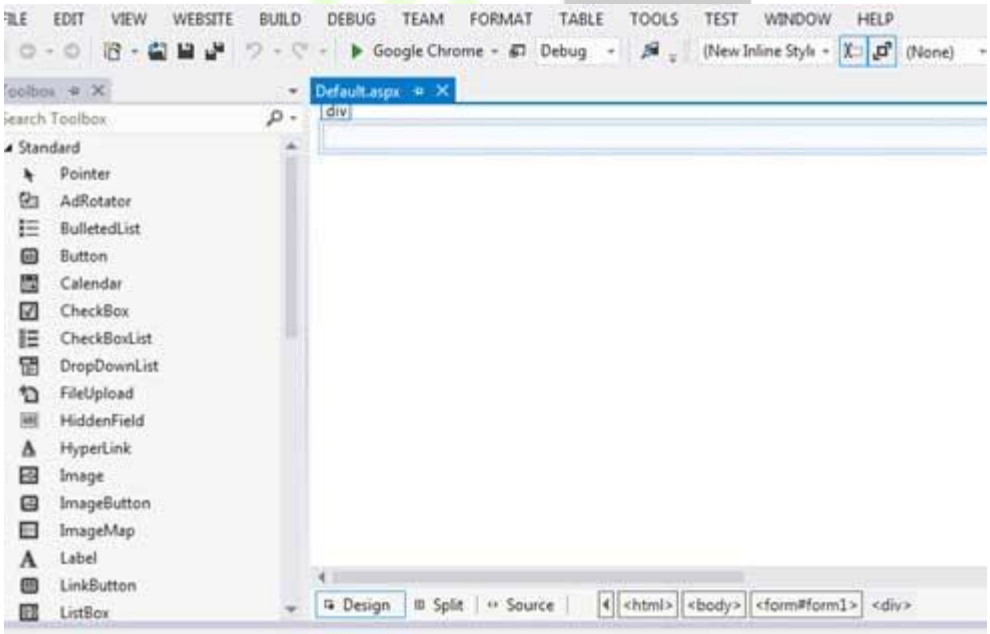
Web forms consists of:

- User interface
- Application logic

User interface consists of static HTML or XML elements and ASP.Net server controls. When you create a web application, HTML or XML elements and server controls are stored in a file with **.aspx** extension. This file is also called the page file.

The application logic consists of code applied to the user interface elements in the page. You write this code in any of .Net language like, VB.Net, or C#.

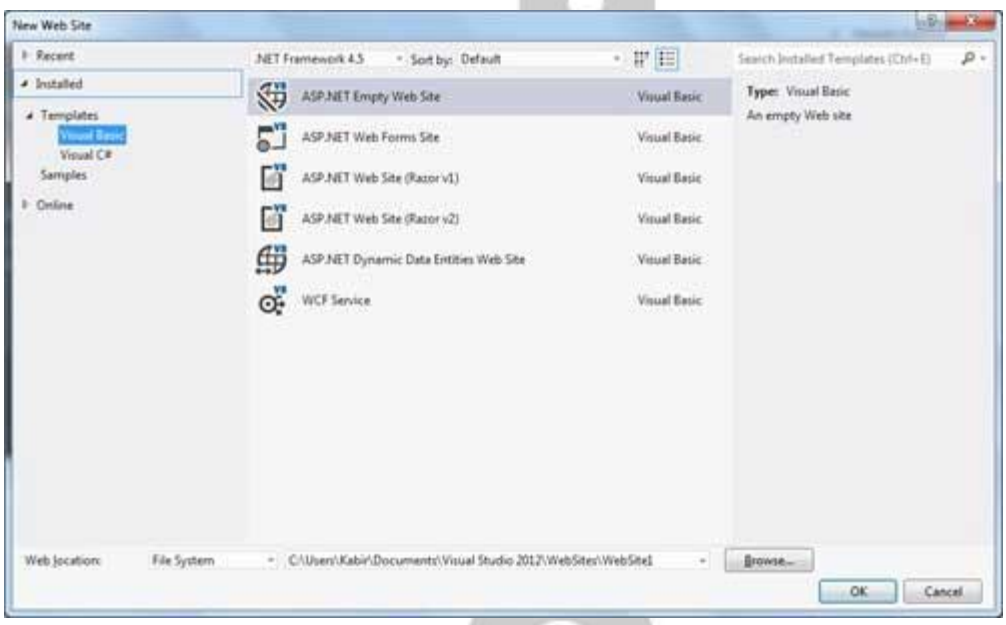
The following figure shows a Web Form in Design view:



Example

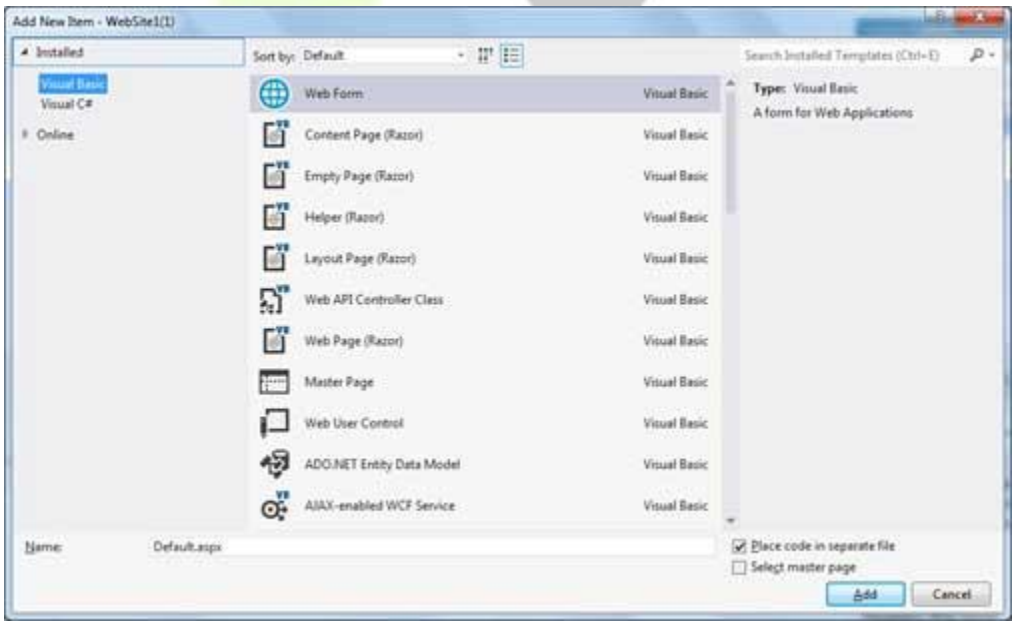
Let us create a new web site with a web form, which will show the current date and time, when a user clicks a button. Take the following steps:

- Select File -> New -> Web Site. The New Web Site Dialog Box appears.



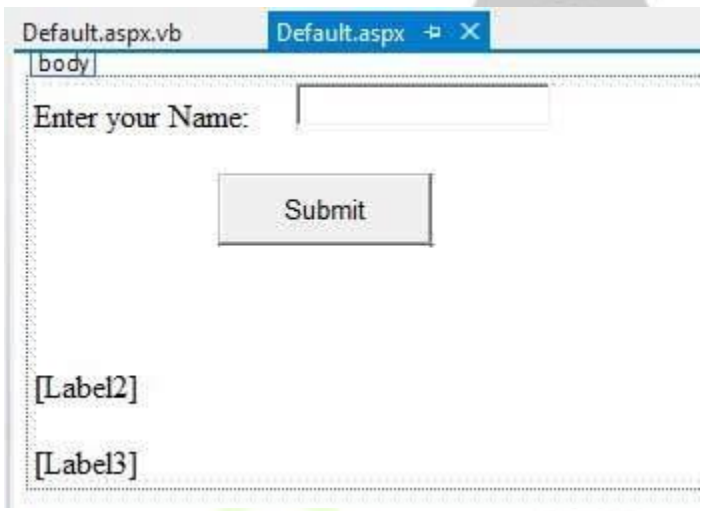
- Select the ASP.Net Empty Web Site templates. Type a name for the web site and select a location for saving the files.

- You need to add a Default page to the site. Right click the web site name in the Solution Explorer and select Add New Item option from the context menu. The Add New Item dialog box is displayed:



- Select Web Form option and provide a name for the default page. We have kept it as Default.aspx. Click the Add button.
- The Default page is shown in Source view

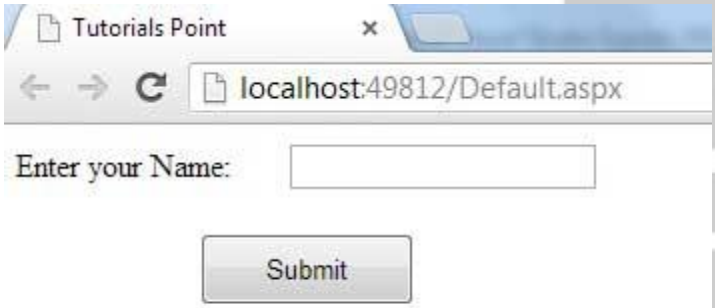
- Set the title for the Default web page by adding a value to the
- To add controls on the web page, go to the design view. Add three labels, a text box and a button on the form.



- Double-click the button and add the following code to the Click event of the button:

```
Protected Sub Button1_Click(sender As Object, e As EventArgs) _
Handles Button1.Click
   Label2.Visible = True
   Label2.Text = "Welcome to .NET lab" + TextBox1.Text "hello" + TextBox1.Text
   Label3.Text = "You visited us at: " + DateTime.Now.ToString()
End Sub
```

When the above code is executed and run using Start button available at the Microsoft Visual Studio tool bar, the following page opens in the browser:



Enter your name and click on the Submit button:

VB.NET SIMPLE TEXTREADER

Textreader and TextWriter are the another way to read and write file respectively, even though these are not stream classes.

The StreamReader and StreamWriter classes are derived fromTextReader and TextWriter classes respectively. The following program using TextReader , Read the entire content of the file into a String

PROCEDURE:

Create a form with a button added to it and enter the code by double clicking the button.

```
Imports System.IO
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, _
                ByVal e As System.EventArgs) Handles Button1.Click
        Try
            Dim line As String
            Dim readFile As System.IO.TextReader = New _
                                        StreamReader("C:\Test1.txt")
            line = readFile.ReadToEnd()
            MsgBox(line)
            readFile.Close()
            readFile = Nothing
        Catch ex As IOException
            MsgBox(ex.ToString)
        End Try
    End Sub
End Class
```

CREATE AN XML FILE IN VB.NET

XML is a platform independent language, so the information formatted in XML can be used in any other platforms (Operating Systems). If we create an XML file in one platform it can be used in other platforms also.

For creating a new XML file in VB.NET, we are usingXmlTextWriter class . The class takes FileName and Encoding as argument. Also we are here passing formating details . The following source code creating an XML file product.xml and add four rows in the file.

PROCEDURE:

Create a form with a button added to it and enter the code by double clicking the button.

```
Imports System.Xml
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim writer As New XmlTextWriter("product.xml",
System.Text.Encoding.UTF8)
        writer.WriteStartDocument(True)
        writer.Formatting = Formatting.Indented
        writer.Indentation = 2
        writer.WriteStartElement("Table")
        createNode(1, "Product 1", "1000", writer)
        createNode(2, "Product 2", "2000", writer)
        createNode(3, "Product 3", "3000", writer)
        createNode(4, "Product 4", "4000", writer)
        writer.WriteEndElement()
        writer.WriteEndDocument()
        writer.Close()
    End Sub
    Private Sub createNode(ByVal pID As String, ByVal pName As String, ByVal
pPrice As String, ByVal writer As XmlTextWriter)
        writer.WriteStartElement("Product")
        writer.WriteStartElement("Product_id")
        writer.WriteString(pID)
        writer.WriteEndElement()
        writer.WriteStartElement("Product_name")
```

```
        writer.WriteString(pName)
        writer.WriteEndElement()
        writer.WriteStartElement("Product_price")
        writer.WriteString(pPrice)
        writer.WriteEndElement()
        writer.WriteEndElement()
    End Sub
End Class
```

Output of the above source code :

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<Table>
  <Product>
    <Product_id>1</Product_id>
    <Product_name>Product 1</Product_name>
    <Product_price>1000</Product_price>
  </Product>
  <Product>
    <Product_id>2</Product_id>
    <Product_name>Product 2</Product_name>
    <Product_price>2000</Product_price>
  </Product>
  <Product>
    <Product_id>3</Product_id>
    <Product_name>Product 3</Product_name>
    <Product_price>3000</Product_price>
  </Product>
  <Product>
    <Product_id>4</Product_id>
    <Product_name>Product 4</Product_name>
    <Product_price>4000</Product_price>
  </Product>
</Table>
```

CREATE A TREE VIEW FROM XML

XML is a self describing language and it gives the data as well as the rules to extract what the data it contains. Reading an XML file means that we are reading the data embedded in tags in an XML file.

PROCEDURE:

Create a form with a button added to it and enter the code by double clicking the button.

```
Imports System.Xml
Imports System.io
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim xmldoc As New XmlDataDocument()
        Dim xmlnode As XmlNode
        Dim fs As New FileStream("tree.xml", FileMode.Open, FileAccess.Read)
        xmldoc.Load(fs)
        xmlnode = xmldoc.ChildNodes(1)
        TreeView1.Nodes.Clear()
        TreeView1.Nodes.Add(New TreeNode(xmldoc.DocumentElement.Name))
        Dim tNode As TreeNode
        tNode = TreeView1.Nodes(0)
        AddNode(xmlnode, tNode)
    End Sub

    Private Sub AddNode(ByVal inXmlNode As XmlNode, ByVal inTreeNode As
TreeNode)
        Dim xNode As XmlNode
        Dim tNode As TreeNode
```

```
        Dim nodeList As XmlNodeList
        Dim i As Integer
        If inXmlNode.HasChildNodes Then
            nodeList = inXmlNode.ChildNodes
            For i = 0 To nodeList.Count - 1
                xNode = inXmlNode.ChildNodes(i)
                inTreeNode.Nodes.Add(New TreeNode(xNode.Name))
                tNode = inTreeNode.Nodes(i)
                AddNode(xNode, tNode)
            Next
        Else
            inTreeNode.Text = inXmlNode.InnerText.ToString
        End If
    End Sub
End Class
```

CODE FOR XML FILE

```
<Tree>
<Product1>
<Product_id>1</Product_id>
<Product_name>PRODUCT1</Product_name>
<Product_price>1000</Product_price>
</Product1>
<Product2>
<Product_id>2</Product_id>
<Product_name>PRODUCT2</Product_name>
<Product_price>2000</Product_price>
</Product2>
<Product3>
<Product_id>3</Product_id>
<Product_name>PRODUCT3</Product_name>
<Product_price>3000</Product_price>
</Product3>
<Product4>
<Product_id>4</Product_id>
<Product_name>PRODUCT4</Product_name>
<Product_price>4000</Product_price>
</Product4>
</Tree>
```