Unit 6

5.0 Visual Studio and ASP.NET Web Controls

(Text Book Chapter 04)

➤5.0 Visual Studio and ASP.NET Controls

5.0 Visual Studio

The latest version of Visual Studio is Visual Studio 2013. Pages 83-95 of your text book introduces the basic components of the development environment. To understand properly, you MUST have hands on while you read these pages of the chapter. Student is therefore advised to read the said pages and use the do-it-yourself approach.

5.1 Web Controls

Web server controls are one of the coolest things about ASP.NET. Think of them as HTML controls on steroids :-). Web server controls are just like HTML controls. Web controls are easily identified by the fact that they have a

```
runat="server“
```

attribute. They are also handled at the server by the ASP.NET runtime when a page containing them is requested. They can expose and raise server events which you can use to interact with them. They are identified by giving them an id attribute which you can then use to reference the control in your code. In all these ways web controls are just like HTML controls.

5.1.1 How Are They Different to HTML?

The most obvious difference between the two is that the tags for web controls don't look like HTML tags. This is because they're not as closely tied to the basic HTML tags as HTML controls are. While HTML controls tend to emit their corresponding HTML tag (with the attributes you've set), they almost always just emit that HTML tag. Web controls can emit multiple HTML tags in all sorts of combinations (or whatever else is needed) in order to accomplish their task. They perform higher level functions and do not necessarily map directly to any one HTML tag. Their object model is generally more complex and is usually more abstract than that of an HTML control.

➢5.0 Visual Studio and ASP.NET Controls

5.1.2 Basic Web Control Declarations

They look like HTML script. Here is a basic example of a 'Label' control.

```
<asp:label id="lblSample" runat="server" />
```

The example 5.1 shows a simple case where an **asp:Label** and a **asp:TextBox** is created.

```
@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="Chapter5WebApplication1._Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
   <asp:Label Text="Type Something Here" runat="server"
                    ReadOnly="False" ></asp:Label>
   <asp:TextBox ID="textColor" runat="server"
                    ReadOnly="false"></asp:TextBox>
   </div>
   </form>
</body>
</html>
```

Example 5.1

When executed, the following (figure 5.1) will be displayed on the browser.

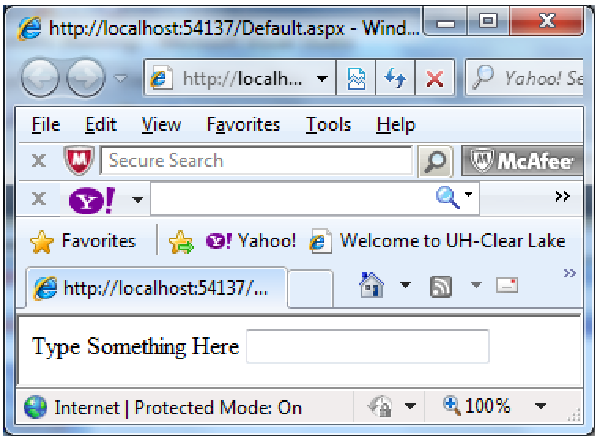

http://localhost:54137/Default.aspx - Wind...
http://localh...
Yahoo! Se
File Edit View Favorites Tools Help
Secure Search
McAfee
Favorites
Yahoo!
Welcome to UH-Clear Lake
http://localhost:54137/...
Type Something Here
Internet | Protected Mode: On
100%

Figure 5.1

➤5.0 Visual Studio and ASP.NET Controls

The example 5.2 show a another simple case where first you create a button, and then at the click of the button, a method is executed, and produces a output string to be displayed back onto the browser.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="Chapter5WebApplication2._Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
</head>
<script runat="server">
  void Button1_Click(object sender, EventArgs e){
    Response.Write("The Button Was Clicked");
  }
</script>
<body>
  <form id="form1" runat="server">
  <p>
  <asp:Button ID="Button1" Text="Click This Button" OnClick="Button1_Click"
      runat="server" Font-Bold="true" Font-Names="Verdana"   Font-Size="Larger"></asp:Button>
  </p>
  </form>
</body>
</html>
```

Example 5.2

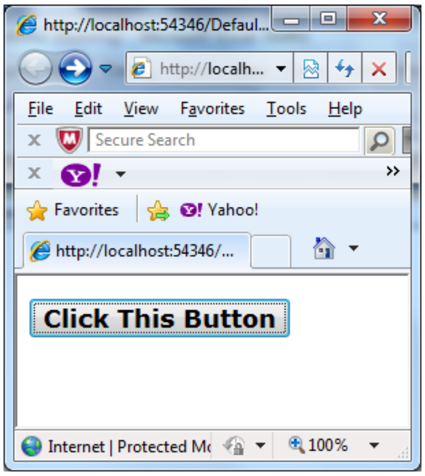

http://localhost:54346/Defaul...
File Edit View Favorites Tools Help
Secure Search
Favorites
Yahoo!
http://localhost:54346/...
Click This Button
Internet | Protected Mo
100%

Figure 5.2-1

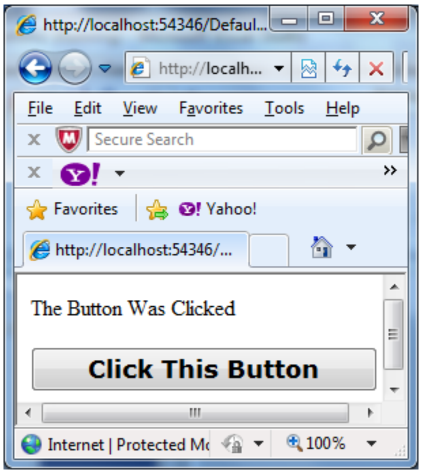


Figure 5.2-2

➢5.0 Visual Studio and ASP.NET Controls

5.2 Default Page

In many instances the first ‘page’ in HTML is named as ‘index.html’. Similarly in ASP.NET the ‘start up’ page is named (by default) as **Default.aspx**.
The name ‘Default’ can be changed if you wish, so that the start up page of your application is not necessarily be the Default.aspx. The Default.aspx like any other ASP.NET web form consists of three sections. Following paragraphs discuses briefly about these sections.

- **@Page Directive**: The @Page directive enables you to specify attributes and values for an ASP.Net Page to be used when the page is parsed and compiled. Every .aspx files should include this @Page directive to execute. There are many attributes belong to this directive. We shall discuss some of the important attributes here.

- **Language**: This attribute tells the compiler about the language being used in the code-behind. Values can represent any .NET-supported language, including Visual Basic, C#, or JScript .NET.

- **AutoEventWireup:** For every page there is an automatic way to bind the events to methods in the same master file or in code behind. The default value is True.

- **CodeBehind:** Uses the Code-Behind model. This is the indication where the ‘behind the scene’ code is.

- **Inherits:** Specifies a code-behind class for the page to inherit. This can be any class derived from the Page class.

There are many other attributes you could use on the page directive. See the MSDN documentation.

➤5.0 Visual Studio and ASP.NET Controls

5.2.1 The DOCTYPE

Valid XHTML Web pages must contain a DOCTYPE declaration that identifies the page as an XHTML page and references the XHTML schema to which it conforms. The page must also include attributes on the HTML tag that reference the XHTML namespace. ASP.NET automatically creates a DOCTYPE declaration when the page is rendered. If the DOCTYPE declaration is removed, XHTML compliance will not be met. The page will not be considered an XHTML page and will not reference the XHTML schema.

The Visual Studio inserts a DOCTYPE declaration like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-Transitional.dtd">
```

This declaration contains the name of the **Document Type Definition**, DTD, which happens to coincide with the element type for the document's root element: html. It also contains a public identifier and a system identifier for the external subset. The public identifier specifies who publishes the DTD (W3C). This DTD contains all HTML elements and attributes, including presentational and deprecated elements (like font). Framesets are not allowed. The markup must also be written as well-formed in which language that is written (EN, for English). The system identifier is the address of the actual document type definition at W3C's web site.

Transitional DOCTYPEs are meant for those making the transition from older markup to modern ways. Strict DOCTYPEs are actually the default – the way HTML 4.01 and XHTML 1.0 were constructed to be used.

5.2.3 The Code-Behind Class

The logic for the Web Forms page consists of code that we create to interact with the form. The programming logic is in a separate file from the user interface file. This file is the "code-behind“ file and has an ".aspx.vb" (VB) or ".aspx.cs" (C-Sharp) extension. The logic we write in the code-behind file can be written in Visual Basic or Visual C#. The code-behind class files for all Web Forms pages in a project are compiled into the project dynamic-link library (.dll) file. The .aspx page file is also compiled, but differently. The first time a user loads the aspx page, ASP.NET automatically generates a .NET class file that represents the page, and compiles it to a second .dll file. The generated class for the aspx page inherits from the code-behind class that was compiled into the project .dll file. When the user requests the Web page URL, the .dll files run on the server and dynamically produces the HTML output for your page.

The default code-behind file name is Defalut.aspx.cs for a C# code behind. Example 5.3 shows how code behind for presentation code shown in Figure 5.2 will look like. You can write new methods and modify this code-behind file as you like.

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

namespace WebApplication1
{
  public partial class _Default : System.Web.UI.Page
  {
    protected void Page_Load(object sender, EventArgs e)
    {
    }
  }
}
```

Example 5.3

➢5.0 Visual Studio and ASP.NET Controls

5.3 Adding Event Handlers

An important feature of ASP.NET is that it allows you to program Web pages using an event-based model that is similar to that in client applications. As a simple example, you can add a button to an ASP.NET Web page and then write an event handler for the button's click event. Although this is common in Web pages that work exclusively with client script (by handling the button's OnClick event in dynamic HTML), ASP.NET brings this model to server-based processing.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="WebApplication1._Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
</head>
<body>
  <form id="form1" runat="server">
  <div>
    <asp:Button id="Button1" Text="Click This Button"  OnClick="Button1_Click"
   runat="server" Font-Bold="True" Font-Names="Verdana"
                              Font-Size="Larger"></asp:Button>
   <asp:TextBox ID="Textbox1" runat="server"></asp:TextBox>
  </div>
  </form>
</body>
</html>
```

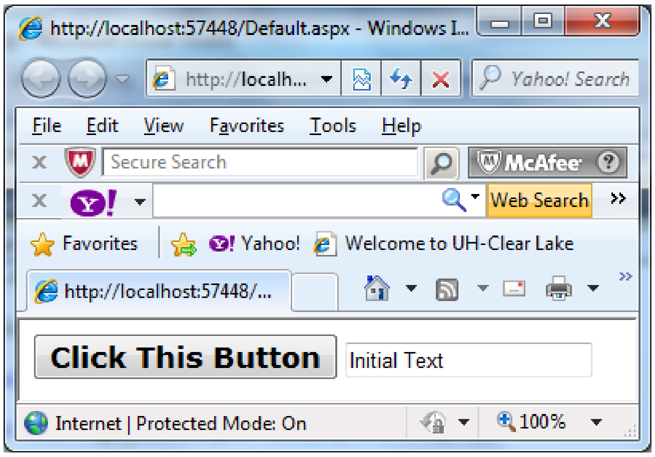
Example 5.4-1 – A Default.aspx file

When you click the "click This Button", the action is taken by the Event handler residing in the corresponding **.aspx.cs** file. See example 5.4-2

➢5.0 Visual Studio and ASP.NET Controls

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace WebApplication1
{
   public partial class _Default : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {
         Textbox1.Text = "Initial Text";
      }
      protected void Button1_Click(object sender, EventArgs e)
      {
         Textbox1.Text = "The Button Was Clicked!";
      }
   }
}
```

Example 5.4-2 – A Default.aspx.cs file


http://localhost:57448/Default.aspx - Windows I...
File Edit View Favorites Tools Help
Click This Button
Initial Text
Internet | Protected Mode: On
100%

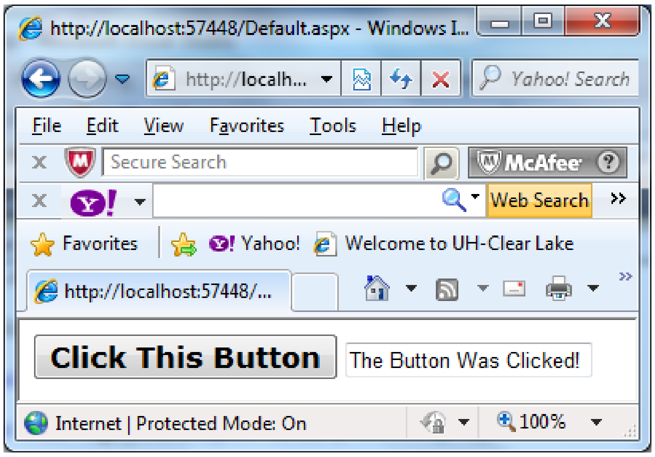

http://localhost:57448/Default.aspx - Windows I...
File Edit View Favorites Tools Help
Click This Button
The Button Was Clicked!
Internet | Protected Mode: On
100%

Figure 5.3-1 -At the start of the page   Figure 5.3-2 –After clicking the button

➢5.0 Visual Studio and ASP.NET Controls

5.4 Visual Studio Web Server

A Web Server is an application. It is rather a server application. Your Visual Studio 2013 has it's own embedded Web Server. Actually it is the famous windows IIS 8 (Internet Information Services) server that is running at the back by Visual Studio 2013. When a Web Application is launched, the application is "Given" (deployed) to the server. Server picks it up and then "Given" it to the ASP.NET framework. ASP.NET compiles and runs the code and prepares the output. The output is packed into a Response object. Output contains HTML tags. HTML is what browser understands. When the HTML is sent back to the browser, browser displays it.

The Visual Studio 2013's server is a scaled down server. It accepts requests from your application only in your machine. We call it the local host. In fact when you start your web application on Visual Studio 2013 you can see something like:

```
    http://localhost:5638/YourApplicationname/Default.aspx
```

This means your local machine's web server uses the **port 5638** for transactions with your client (browser). Other users (outside users) cannot see your web application. Your application does not run to be visible over the internet.