

# **Beginning Web Applications with ASP.NET**

**Instructor: Frank Stepanski**

## **Overview**

From this class, you will learn the basics of developing web applications using the Microsoft web technology ASP.NET (version 3.5) and using the free web development tool Visual Web Developer 2008 Express. You will also learn the basics of programming using C#.

**Visual Web Developer 2008 Express** will allow you to create web pages (.aspx) that use the .NET framework and the specific objects and controls of ASP.NET.

Free download: <http://www.microsoft.com/express/download/>

It is fully compatible with any version of Windows XP or Vista.

For specific questions: <http://www.microsoft.com/express/support/>

The auto-install executable downloads everything you need to get started, but if you need additional components, like the .NET 3.5 Framework, you can get it here:

<http://www.microsoft.com/downloads/details.aspx?FamilyID=333325fd-ae52-4e35-b531-508d977d32a6&DisplayLang=en>

## **Audience for this Class**

Although I try to explain these topics from the beginner viewpoint with no experience in web development, (having some experience with HTML and using a scripting or programming language is helpful.) it would be helpful if you have some experience with HTML and using a scripting or programming language.

If you have taken either one of my ASP classes: [Basic Web Applications using ASP](#) or [Database E-Commerce Basics with ASP](#) then you should have a solid foundation for the topics in learning ASP.NET. Any experience with other scripting languages such as JavaScript or another programming language (PHP) will be helpful as well.

## Evolution from ASP to ASP.NET

ASP had many limitations which were the result of having a scripting language (VBScript) as its core language. When developing larger web applications, scripting languages are notoriously slow, resource intensive, and use late-bound object type variables (variant data types).

Since VBScript only had late-bound objects, it made it impossible to create a true [IDE](#) (integrated development environment) because these objects could not be recognized at design time. This prevented Microsoft from providing ASP programmers any of the powerful debugging, IntelliSense, and error checking tools found in programming languages like Visual Basic or Visual C++.

Without debugging tools, ASP programmers were hard pressed to troubleshoot the problems in their scripts.

ASP.NET circumvents these problems. For starters, ASP.NET web pages are executed within the [CLR](#) (common language runtime). The CLR is a virtual machine that simply allows you to write your ASP.NET pages in almost any programming language that supports the .NET framework. This gives the developer (you and me) the flexibility of using a real programming language instead of a limited scripting language.

### ASP.NET File Extensions

Since ASP pages have a file extension of **.asp**, what would be the file extension of ASP.NET?

#### **.aspx**

When you go to a web browser and the web page has this extension, you know it is an ASP.NET page.

An additional extension would be **.aspx.cs**. This is the file extension where all your C# code will be stored (“code-behind”). You never see this extension on the web because this file gets compiled on the web server that contains the .NET Framework.

**Note:** The **.cs** stands for C# (# is “sharp”).

If you use a different programming language for ASP.NET pages like VB.NET, it would have an **.aspx.vb** extension instead.

If you're a little confused, don't worry, it will become clearer soon enough. ☺

**Note:** You can have the code-behind file (**.aspx.cs**) appear in the **.aspx** itself, but I will be using separate files in my examples.

### **Six Important Facts about ASP.NET** (warning: some technical jargon will be used)

#### 1. ASP.NET is integrated within the .NET Framework:

The [.NET Framework](#) is divided into an almost painstaking collection of functional parts, with a staggering total of more than 7000 types (functions, classes, interfaces, etc.). The .NET framework allows for the development of Windows and Web applications. This means ASP.NET is only a part of the entire framework of .NET. But don't worry, you don't need to know that much of .NET to get started; I just wanted to give you a quick idea of how vast it is.

#### 2. ASP.NET is Compiled, Not Interpreted (like ASP):

ASP.NET applications actually go through two stages of compilation. In the first stage, the C# code you write (or whatever .NET language it's written in) is compiled into an intermediate language called Microsoft Intermediate Language (MSIL). The second level compilation happens just before the page is actually executed. At this point, the IL code is compiled into low-level native machine code. This stage is known as just-in-time (JIT) compilation and it takes place in the same way for all .NET applications (including Windows applications).

#### 3. ASP.NET is Multi-language:

Though you will probably choose one language over another, you can use any .NET supported programming language to create your ASP.NET web pages. You can even use a different programming language for each new ASP.NET web page, although you probably don't want to. The default supported and most popular languages for ASP.NET are C# and VB.NET.

#### 4. ASP.NET is Object-Oriented:

ASP provides a small set of objects. These objects are really just a thin layer over the raw details of HTTP and HTML. On the other hand, ASP.NET is truly object-oriented. Not only does your code have full access to all objects in the .NET Framework, but you can

also exploit all the conventions of OOP (object-oriented programming) environment, such as encapsulation and inheritance. For example, you can create reusable classes, standardize code with interfaces and bundle useful functionality in a distributable, compiled component.

**Note:** One of the best examples of object-oriented thinking in ASP.NET is found in server-based controls. Server-based controls are the epitome of encapsulation. Developers can manipulate server controls programmatically using code to customize their appearance, provide data to display, and even react to events. The low-level HTML details are hidden away behind the scenes. Instead of forcing the developer to write raw HTML manually, the control objects render themselves to HTML when the page is finished rendering. In this way, ASP.NET offers server controls as a way to abstract the low-level details of HTML and HTTP programming.

#### 5. ASP.NET is Easy to Deploy and Configure:

Every installation of the .NET Framework provides the same core classes. As a result, deploying an ASP.NET application is relatively simple. In most cases, you simply need to copy all the files to a directory on a web server (web host). There are configuration settings that can be made in a **web.config** file.

**Note:** Configuration settings using the **web.config** file will not be covered in this class, but there are many resources on the web and books to help you. Ask me and I will point you in the right direction. ☺

#### 6. ASP.NET is Multidevice and Multibrowser:

One of the greatest challenges web developers face is the wide variety of browsers they need to support. Different browser brands, versions, and configurations differ in their support of HTML. Web developers need to decide whether they should render their content according to HTML 3.2, HTML 4.0, XHTML 1.0 or others. ASP.NET server controls render their HTML adaptively by taking the client's capabilities into account.

**Note:** One example is the ASP.Net validation controls, which use JavaScript and DHTML to enhance their behavior if the client supports it. This allows the validation controls to show dynamic error messages without the user sending back the page to the server for more processing. The features are optional since the controls are intelligent enough to identify if their browser supports this functionality or not.

## Web Server for ASP.NET

To view your ASP.NET web pages, it needs to be requested through a web browser running on a web server. The web server then dispatches the request to the ASP.NET engine installed on the web server. The ASP.NET engine processes the page and returns the resulting HTML markup to the browser.

ASP.NET web pages are normally developed and tested locally on your PC. You could develop them locally and test them on your webhost (FTP-ing the files over each time). Though it makes more sense to test them on your own PC first, right?

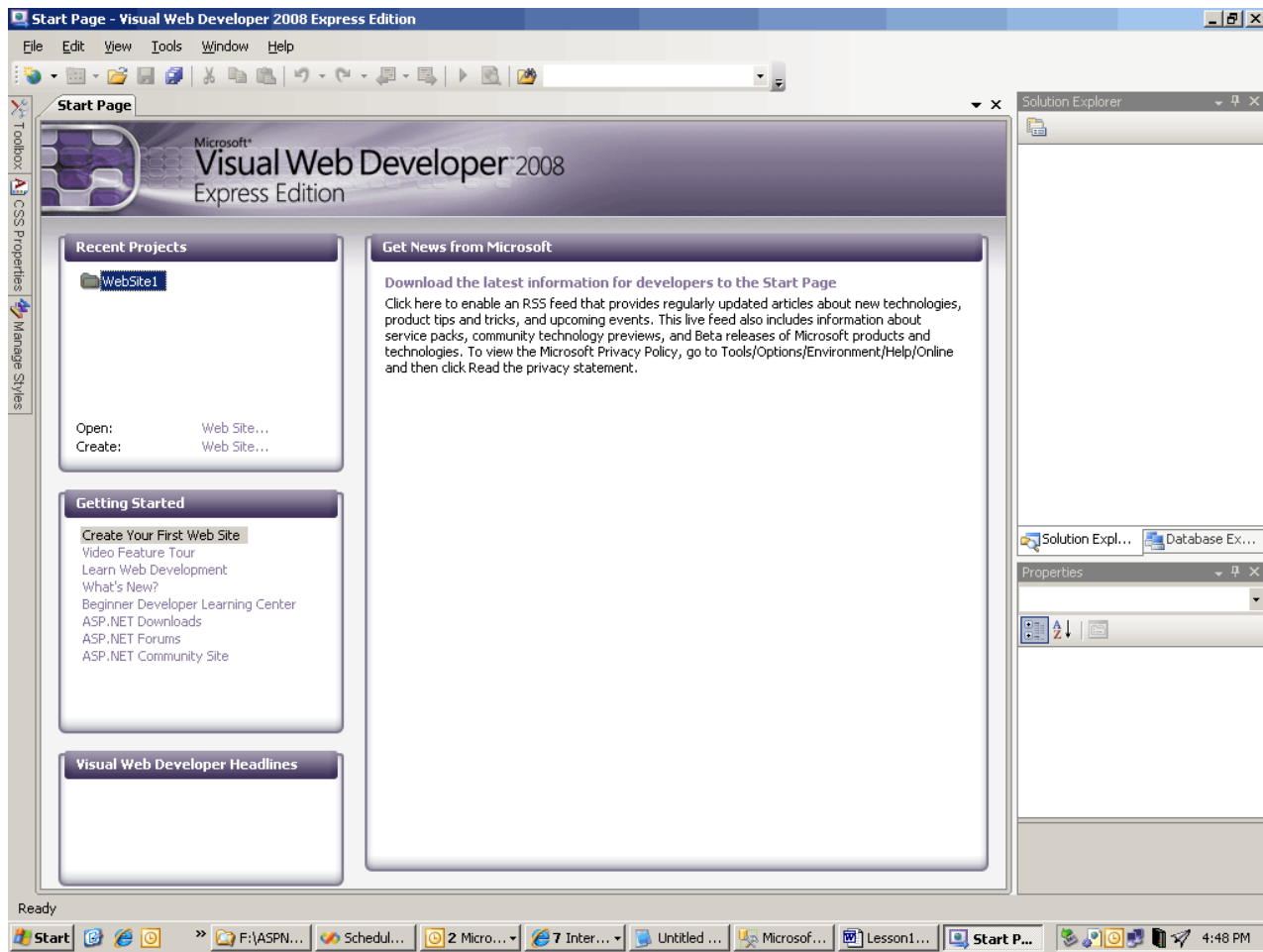
So how do you get a local web server setup to test them?

You don't.

Visual Web Developer includes a built-in web server. So anytime you want to test your **.aspx** web pages that you've developed, all you have to do is launch a web browser from Visual Web Developer and it will create a running instance of a temporary web server. No fuss, no muss. 😊

**Note:** If you have XP Professional or Vista Home Premium then you have a Microsoft web server ([IIS](#) – Internet Information Services) already. You can use this web server instead if you want. I will show you where you can FTP up your website to pull files from a directory created in IIS. I will not cover how to set or use IIS in this class. That is beyond this class but, if you are having some IIS issues post a question and I will help guide you in the right direction. 😊

## Brief Tour of Visual Web Developer



**Figure 1 – Visual Web Developer 2008 Express**

When you open Visual Web Developer, the Start Page is initially shown (Figure 1). This Start Page includes a list of Recent Projects in the upper-left corner, a Getting Started section with some links for accomplishing common tasks in the bottom left-corner, and a list of recent articles on Microsoft’s MSDN site in the right column.

On the left you’ll find the Toolbox. On the Start Page, the Toolbox is empty, but when you’re working with an ASP.NET page, the Toolbox contains the plethora of ASP.NET Web controls that can be added to the page. We’ll be discussing what Web controls are and their use in later lessons.

To the right of the screen, you'll find the Solution Explorer. Again, on the Start Page this is empty, but when you load or create an ASP.NET website, the Solution Explorer will list the website's files. These files include database files, HTML pages, ASP.NET pages, image files, CSS files, configuration files, and so on. In addition to the Solution Explorer, the right portion of the screen is also home to the Database Explorer (not covered in this class).

**Note:** If you accidentally close the Solution Explorer, you can re-open it by going to View->Solution Explorer from the menu.

## **Creating a New ASP.NET Website**

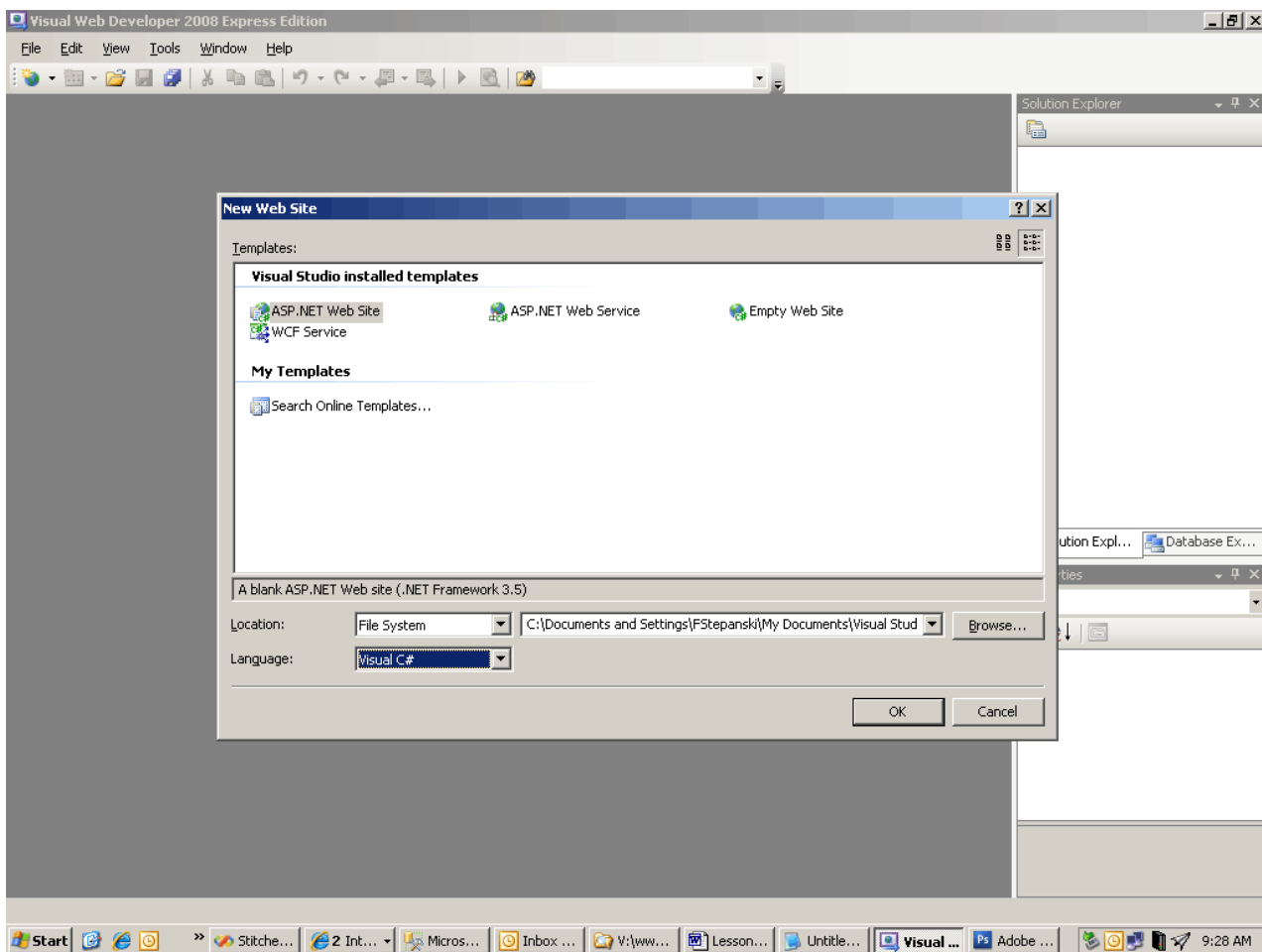
To create and work with an ASP.NET page, we must first create an ASP.NET website.

You can go to the File menu and choose the New Web Site option, click on the New Website icon in the Toolbar, or you can click the Create Web Site link in the Recent Projects pane of the Start Page.

All of these approaches bring up the New Web Site dialog box (Figure 2). Select the ASP.NET Web Site template and change the Language option to C#. Before you click OK, change the Location dropdown to a place where you want your files stored.

Normally, you will create a new folder location for each new website.

**Note:** For this class you can just add new files (**.aspx**) for each new lesson instead of creating a new website each time, but it is up to you.

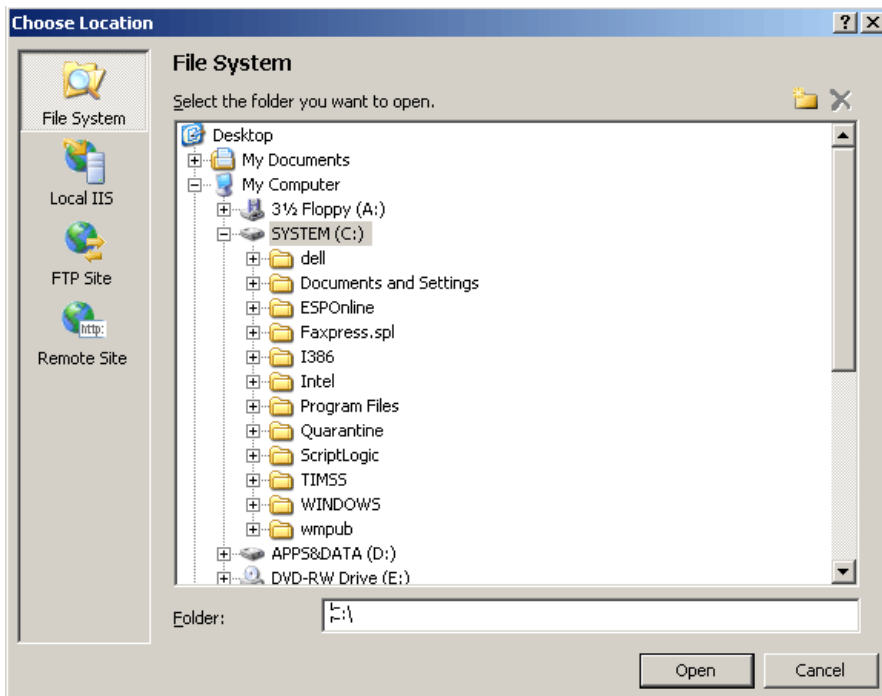


**Figure 2 – Creating a New ASP.NET Website**

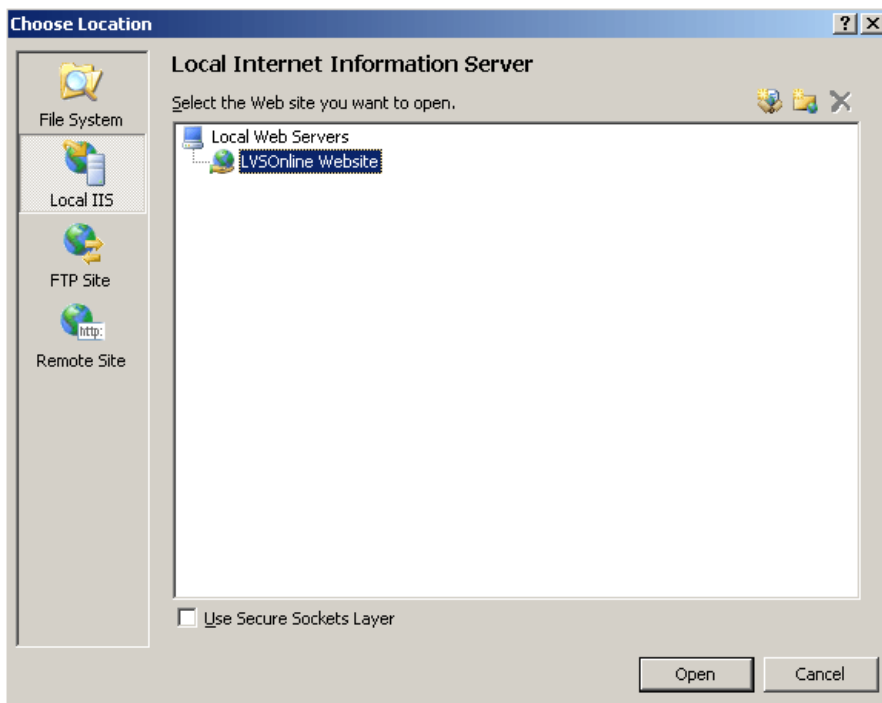
You can click the “Browse” button to find the folder (Figure 2a) you want your ASP.NET website to be stored. This will store all your files that will be used throughout your website such as the .aspx, .aspx.cs, .html, .css, .js and so forth.

If you want to use IIS as your default web server, then after you click “Browse”, select the IIS icon (Figure 2b) and it will display the available websites that are setup in IIS that you can use.

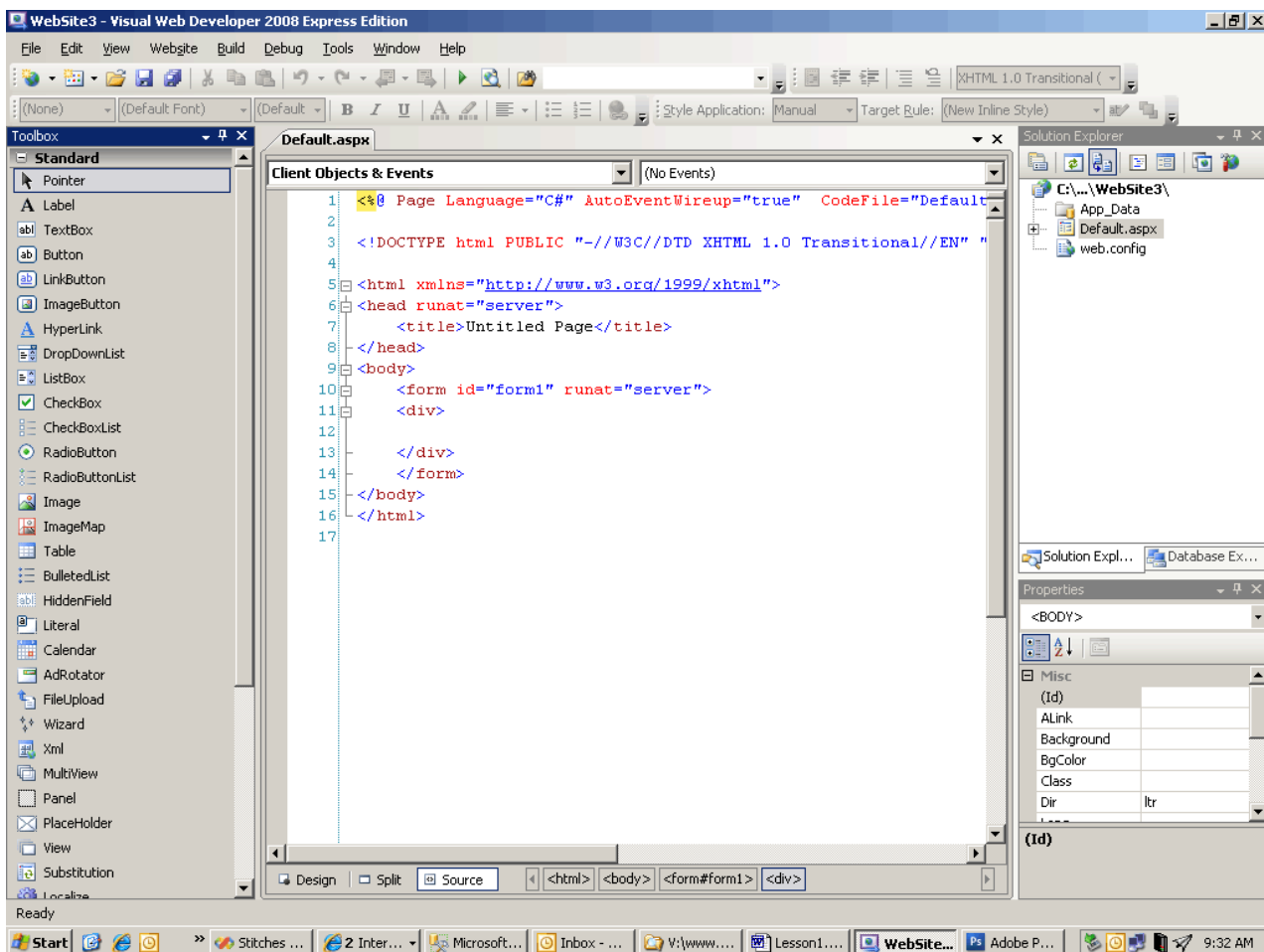
**Reminder:** I will not be covering IIS or its use in this class. But feel free to ask about it and I will try to help (since I can’t see what you’re doing on your PC) or point you in the right direction. 😊



**Figure 2a – Choosing a folder location for your new ASP.NET Website**



**Figure 2b – Choosing a website in IIS to be used as your default web server**



**Figure 3 – The Workspace of Your First ASP.NET Website – Default.aspx**

## ASP.NET Website: Overview

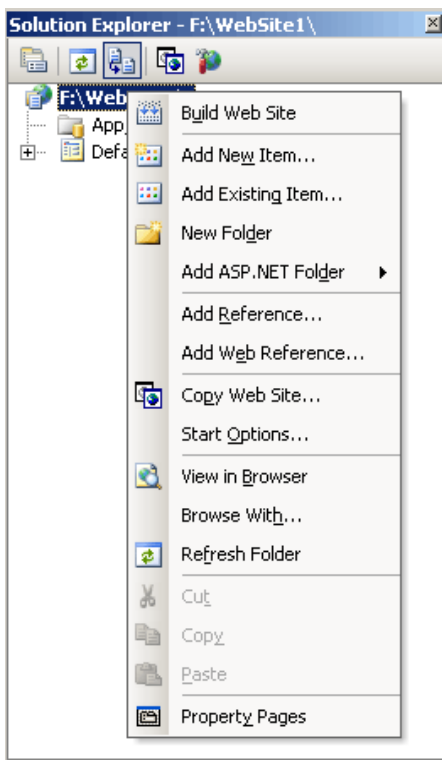
A website (created in Visual Web Developer) is a collection of resources: static and dynamic web pages, graphic files, style sheets, configuration files, and so on. In addition to various files, a website may contain subdirectories, each of which may contain its own set of files and further subdirectories. A website is akin to a folder on your personal computer: It's a repository for files and subfolders.

When you create a new website, it will already have created a **web.config** file along with a single ASP.NET page, **default.aspx** (composed of two files: **default.aspx** and **default.aspx.cs**).

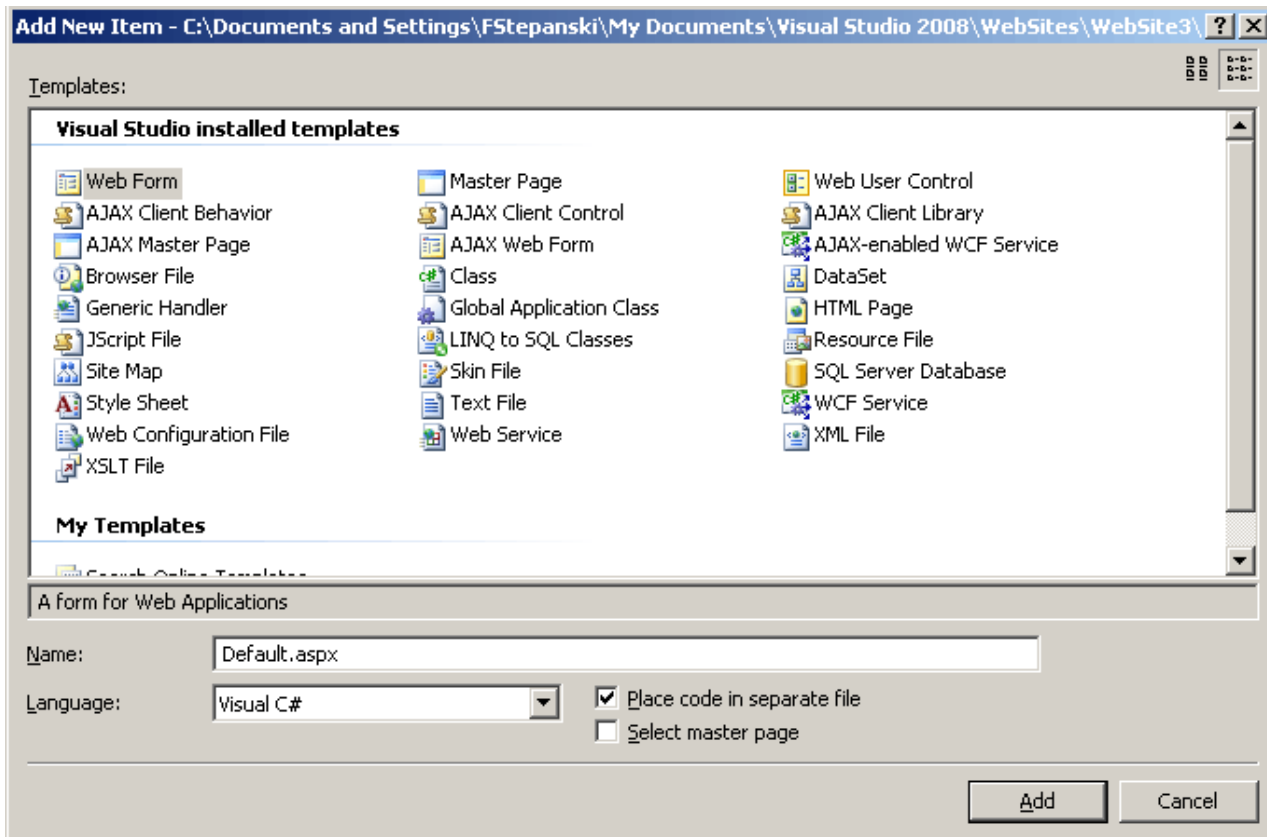
**Note:** Even though I have been talking about websites, files and stuff, a website can have as few as one file or as many as thousands of files. Other web tools (i.e. Dreamweaver) forces (“very strongly suggested”) you to create a “website”, but simply it is just a way of storing and managing your files.

## ASP.NET Website: Adding Content

You can easily add additional files (folders too) through the Solution Explorer. From the Solution Explorer, start by right-clicking on the website name; this will bring up the context menu (Figure 4). You then select “Add New Item” (Figure 5) to add the specific file to your website.



**Figure 4 – Solution Explorer – Adding new items to your website**



**Figure 5 – Adding a new item to your website**

To add a new ASP.NET page, select Web Form and make sure the Language is C# and the Place code in separate file is checked as well.

It's called a Web Form because it is essentially a regular HTML form that is processed on the web server. This form contains web controls (discussed in a later lesson) that are compiled and processed on the web server as well.

## ASP.NET Website: Reviewing default.aspx



```
1 <%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
2
3 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www
4
5 <html xmlns="http://www.w3.org/1999/xhtml" >
6 <head runat="server">
7     <title>Untitled Page</title>
8 </head>
9 <body>
10     <form id="form1" runat="server">
11     <div>
12
13     </div>
14     </form>
15 </body>
16 </html>
17
```

Figure 6– Reviewing the source HTML of default.aspx

This is a screenshot (Figure 6) of what you will see when you open and view the “Source” of **default.aspx** from the Solution Explorer (by double-clicking on it). When you start adding web controls; the HTML will look a little different of course. (The HTML will look a little different when you start adding web controls. )

When reviewing the HTML, it basically looks like any regular HTML page with a few differences.

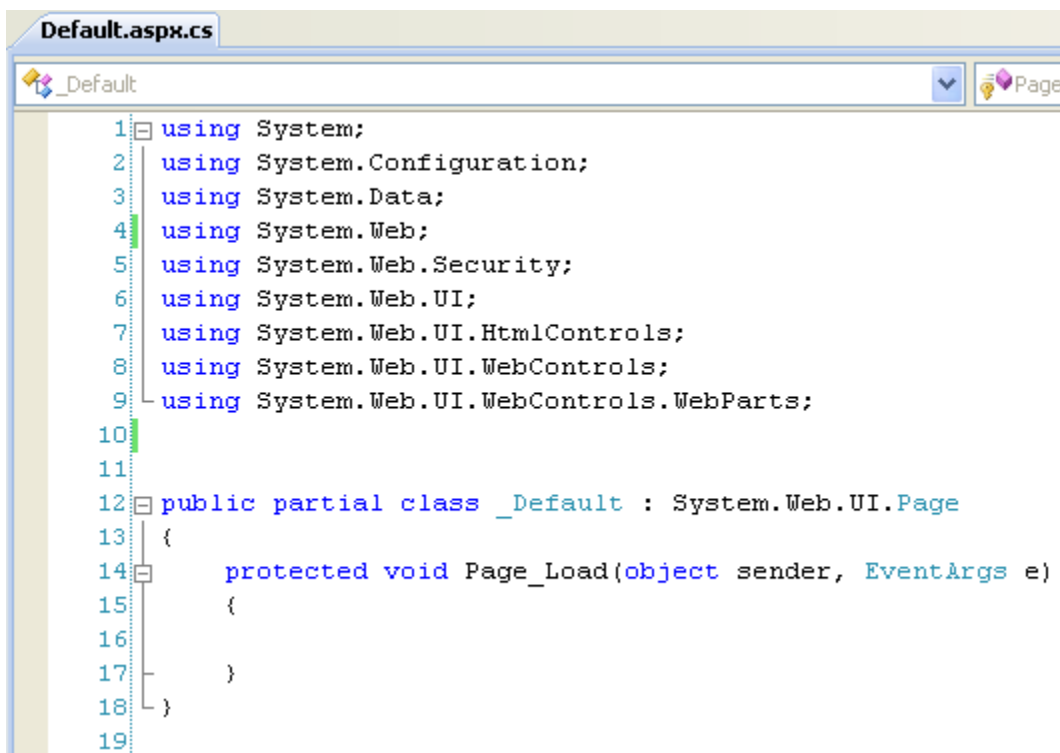
The top line is a declaration to the ASP.NET engine installed on the web server that it’s an ASP.NET page. Then the rest is normal HTML apart from that *runat=“server”* tag and attribute.

All server controls must be placed within this tag and it tells the ASP.NET engine to execute the code behind it on web server.

**Note:** When you add server controls to your page, it automatically gets placed with the `<form runat="server">` tags. But if for some reason it does not, just copy and paste the code.

## ASP.NET Website: Reviewing default.aspx.cs

As I mentioned earlier, all your C# code you write (discussed next lesson) will be stored in an **.aspx.cs** file.



```
1 using System;
2 using System.Configuration;
3 using System.Data;
4 using System.Web;
5 using System.Web.Security;
6 using System.Web.UI;
7 using System.Web.UI.HtmlControls;
8 using System.Web.UI.WebControls;
9 using System.Web.UI.WebControls.WebParts;
10
11
12 public partial class _Default : System.Web.UI.Page
13 {
14     protected void Page_Load(object sender, EventArgs e)
15     {
16
17     }
18 }
19
```

**Figure 7– Reviewing the C# for default.aspx.cs**

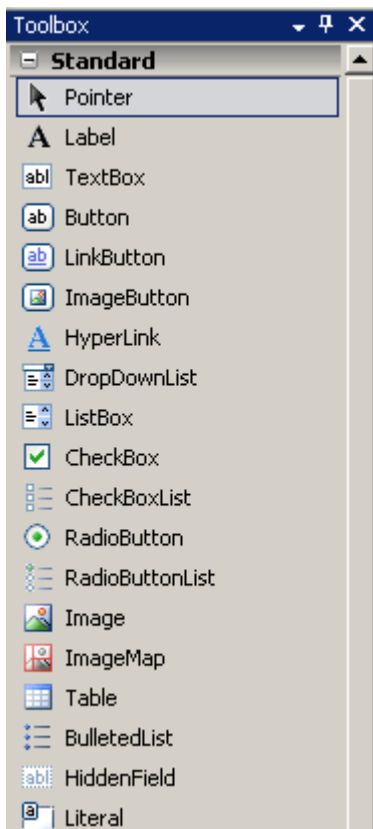
This page will contain all our C# code that will respond to the different page events (discussed in a later lesson) as well as any custom methods you create.

The top declarations allow the page to use many of the libraries that are part of the .NET Framework. The code in lines 15 – 18 declares the event for the Page loading (*Page\_Load*).

The cool thing about ASP.NET pages is that the .NET Framework is used for all types of application development including Windows applications. This means you can create web pages that can behave like a desktop application on the web. ☺

Before finishing this lesson lets go through adding a few controls and little C#.

### ASP.NET Website: Adding Some Controls to default.aspx



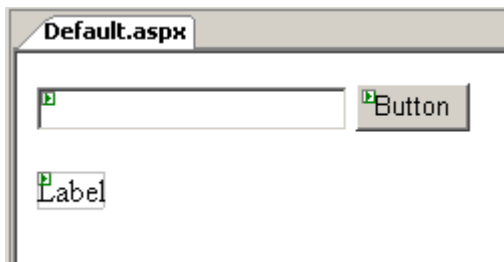
**Figure 8 – Toolbox**

This screenshot (Figure 8) shows the many web controls you can add to an ASP.NET page. The ones we will be using for this example are Label, TextBox and Button.

**Note:** If you accidentally close the Properties Window, you can re-open it by going to View->Properties Window from the menu.

Drag each one of these controls on the page and try to make it look like the example (Figure 9). The source of the example is in the next screenshot (Figure 10).

**Note:** The **default.aspx** page needs to be opened in “Design” mode to see how the controls will look on your page. You can be in “Source” mode as well, but it’s harder to arrange the controls unless you’re very familiar with HTML.



**Figure 9– Drag Controls from Toolbox to default.aspx**

```
10 <form id="form1" runat="server">
11 <div>
12 <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
13  
14 <asp:Button ID="Button1" runat="server" Text="Button" />
15 <br />
16 <br />
17 <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
18 </div>
19 </form>
```

**Figure 10 – Source of default.aspx after dragging controls**

Every time you add a control to your ASP.NET page, once it’s compiled on the server, it will create the code for it to render correctly on a web browser.

What that means in English is that the code that you see in your **default.aspx** in Visual Web Developer, will not be the same code that is sent to the web browser after the

ASP.NET engine on the web server renders it. The ASP.NET engine creates the necessary HTML for the web browser that is requesting the page and sends back what it should look like based upon what controls are on your ASP.NET page (including properties and code used).

So basically a lot of stuff is going on behind the scenes on the web server before you see any results in the web browser. Honest.

### ASP.NET Website: Adding Some C#

```
11 public partial class _Default : System.Web.UI.Page
12 {
13     protected void Page_Load(object sender, EventArgs e)
14     {
15
16     }
17     protected void Button1_Click(object sender, EventArgs e)
18     {
19         Label1.Text = TextBox1.Text;
20     }
21 }
22
```

Figure 11 – C# in default.aspx.cs

OK, here are some quick steps to replicate this with the least amount of fuss:

1. In the “Design” view of **default.aspx**, double-click on the Button control
2. This will put you in the **default.aspx.cs** page and within the *Button1\_Click* code (magically created for you).
3. Type in *Label1.Text = TextBox1.Text*

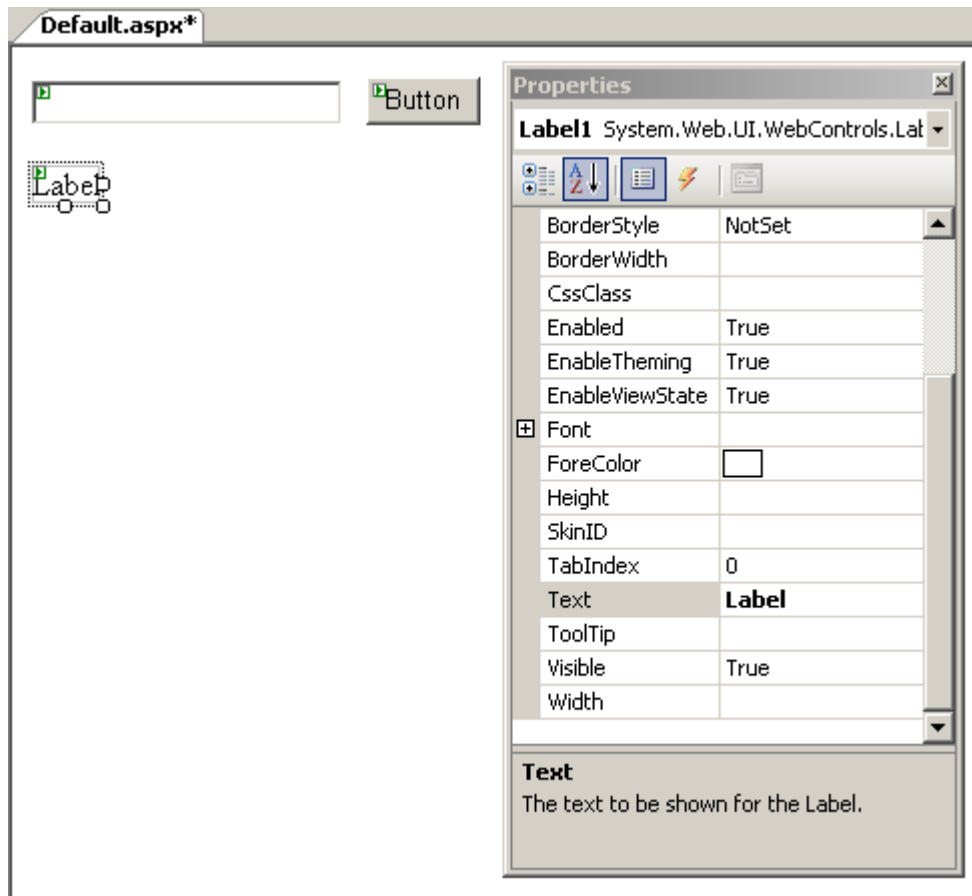
That’s it.

What you did was....

You told Visual Web Developer that you want code in the Click event of the Button. The line of code you typed in says, “When the Button is clicked, whatever is typed in TextBox1 put it in Label1”. The *Text* property of the TextBox and Label are what is visually displayed on the page.

**Note:** C# is case-sensitive.

## ASP.NET Website: Changing Control’s Properties



**Figure 12 – Properties Window of Label1 control**

Every control has properties. A property is like an adjective of something and is usually visual: color, size, text, width, height and style.

In this example, I want to remove the default text of Label1.

I selected my label control on **default.aspx** and the Properties Window displayed the specific properties for the web control. Every time you select a control on your page, the Properties Window automatically changes to display that controls specific properties.

The *Text* property for a label control is what gets displayed on the page when it is visible (you can toggle the *Visible* property too). I want the label to display nothing until the Button is clicked (Button1) and then that one line of code will get executed.

**Note:** By default, a Label control's *Text* property is the name itself, so I just removed it.

What happens when you remove it???

Now the Label control on the page has its name in brackets [Label1]. That means its *Text* property is empty.

Although it does makes sense if you think about it....

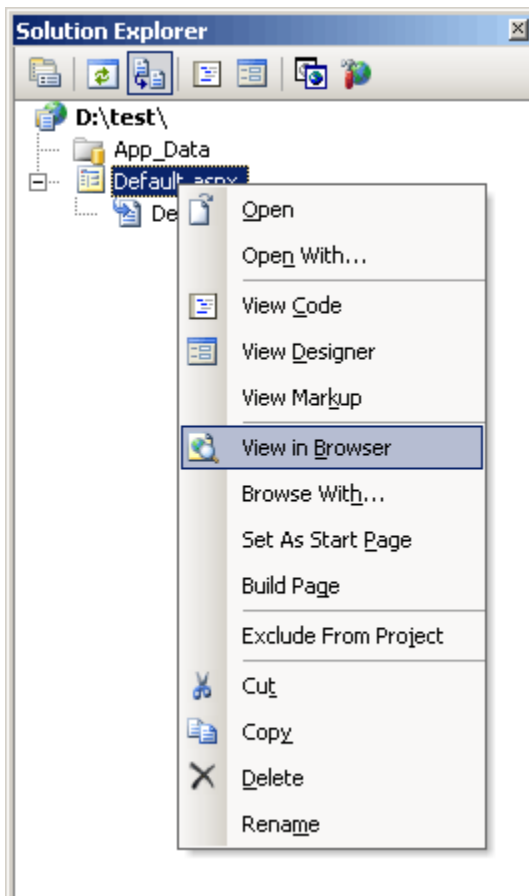
The only way to visually see a label control on an ASP.NET page (in "Design" mode) is by its *Text* property, so how else would you see it?

Visual Web Developer puts it in brackets so you know its empty, but you can still see it to re-select it and change its properties if needed. Wow, those Microsoft people are pretty smart, huh?

Anyways, let's test out our finished example....

## ASP.NET Website: Testing our Website

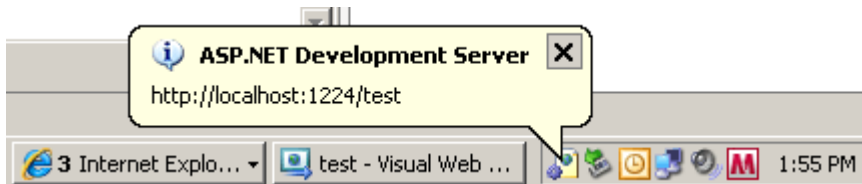
To test our website, you can just right-click the **default.aspx** page in the Solution Explorer and select “View in Browser” (Figure 13).



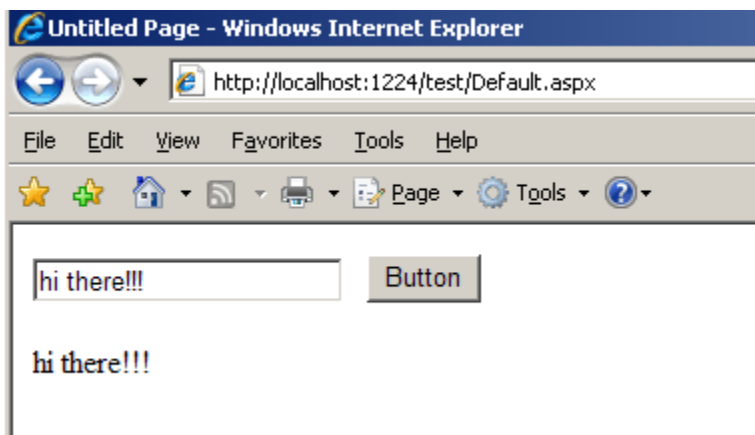
**Figure 13 – Testing default.aspx in a web browser**

This will open your default web browser and allow you to view your web page through a web server the Visual Web Developer provides. This temporary web server will show an icon (Figure 14) in the taskbar.

Once you close the web browser, after you're done testing, you can let the web server continue to run or close it each time you test a new ASP.NET page.



**Figure 14 – Visual Web Developer server running**



**Figure 15 – Viewing default.aspx in a web browser**

You can test your first ASP.NET locally and see how it works.

**My version (with a few little additions):**

[http://fsjay7123.brinkster.net/ASPNET\\_Basics/Lesson1/Default.aspx](http://fsjay7123.brinkster.net/ASPNET_Basics/Lesson1/Default.aspx)

## Note: Free Brinkster Host

<http://www.brinkster.com/hosting/FreeDeveloper.aspx>

The free web host account Brinkster offers does not support the 3.5 version of the .NET framework (only 2.0). We will not be covering anything that is specific to the 3.5 version. To use the free host you will have to change the 'Target Framework' of the .NET build of your website. After you create your website, right-click the Property Pages of the site and change the framework (build) to version 2.0. You will also have to remove the existing 'Reference' of System.XML.Linq as well.

You also may have to remove the Linq references in the .aspx.cs as well. They would be *using System.Linq* and *using System.Xml.Linq*.

Let me know if you have any problems with the Brinkster free account.

## Assignment for Lesson 1

1. Install Visual Web Developer Express 2008 (register it too).
2. Setup your Brinkster account (if you need one).
3. Upload your default.aspx (with any additions you want to add).
4. Post on the class discussion board that your Lesson 1 is ready with a direct link to your page, such as:

<http://fsjay7123.brinkster.net/frank-lesson1.aspx>

I will respond with a review on the class discussion board.

**Disclaimer:** There will be many topics that I will not be able to cover in detail so I will try to include links to resource sites (i.e. Microsoft) throughout the lessons to give you additional information.

## **Additional Resources for ASP.NET:**

Great Introduction to the basics:

<http://www.w3schools.com/aspnet/default.asp>

Microsoft ASP.NET Development Center:

<http://msdn.microsoft.com/en-us/asp.net/aa336567.aspx>

Microsoft ASP.NET Learning Video Series:

<http://msdn.microsoft.com/en-us/asp.net/bb498194.aspx>

Copyrighted 2008 © Frank Stepanski

Used with Permission :: LVS Online Classes / LVS Associates

Lessons, files and content of these classes cannot be reproduced and/or published without the express written consent of the author.

Use of this site implies agreement with the [Terms of Use](#)