# Certificate in C# Programming

## Creating Web Applications in C#

By Randal Root

# Module 01

In this Module, we will see how and why ASP.NET applications are created. This covers the typical languages and technologies used in these applications.

This Module corresponds to the first part of the course Text book.

Text: Beginning ASP.NET 4.5 in C#, by Matthew MacDonald, ISBN-13: 978-1-4302-4251-2

“Part 1: Introducing .NET

You could start coding an ASP.NET application right away by following the examples in the second part of this book. But to really master ASP.NET, you need to understand a few fundamental concepts about the.NET Framework.

Chapter 1 sorts through the Microsoft jargon and explains what the .NET Framework really does and why you need it. Chapter 2 introduces you to C# with a comprehensive language tour. Finally, Chapter 3 explains the basics of modern object-oriented programming.” (MacDonald, 2012, p. xxxv)

**Why are we not using a newer book?**

So far there is not a newer book for this subject. This is due to ASP.NET being redone and renamed to ASP.NET Core (a lightweight version of ASP.Net that runs on multiple operation systems.) We will look at ASP.NET Core later in the course.

## Basic HTML

*“It would be difficult to describe* ***early websites*** *as web applications. Instead, the first generation of websites often* ***looked more like brochures****, consisting mostly of fixed HTML pages that needed to be updated by hand*.” (MacDonald 2012, p. 3)

**Demo01.html**: A standard HTML page

<!DOCTYPE html>

<head>

<meta charset="utf-8" />

<title>Demo01</title>

</head>

<body>

<h1>Test Header</h1>

<p>Test Paragraph</p>

</body>

</html>

## Web Application Projects versus Web Site Projects in Visual Studio

Visual Studio is the most common tool used to develop ASP.NET web sites. It can also be used to create HTML and CSS files.

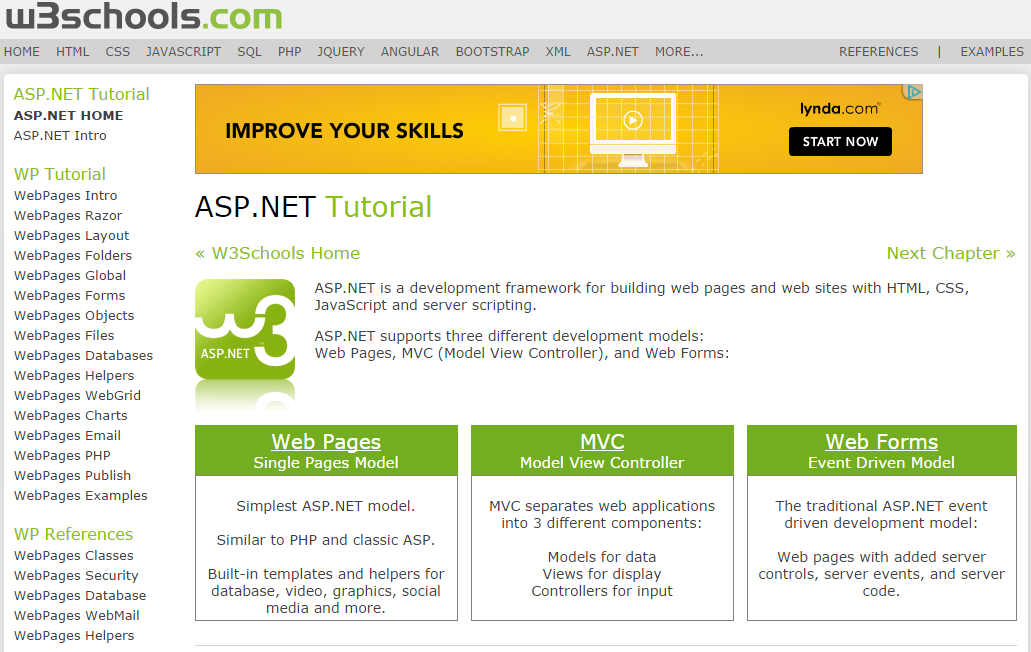
When you want to create a web site in Visual Studio, you must first decide between **two different development models**, either the Web Site Project model or Web Application Project model.

“In Visual Studio you can create web application projects or web site projects. You create or open a web application project by choosing New Project or Open Project in the Visual Studio File menu. You create or open a web site project by choosing New Web Site or Open Web Site in the File menu. It’s best to choose the right type before you create a web project, because it can be **time-consuming, difficult, and error-prone to convert from one type to the other**.

Note: For new development, we **recommend** that you **choose web application projects**. This topic explains that web site projects have some advantages, but many developers who choose web site projects eventually find that the disadvantages outweigh any perceived advantages. In addition, as new ASP.NET features are developed, they won’t always be made available for web site projects. For example, the next Visual Studio release after Visual Studio 2012 will have new tooling for creating web projects, and this new tooling will work only with web application projects. For more information, see Creating an ASP.NET Web Project in Visual Studio 2013.” (MSDN 2014 <https://docs.microsoft.com/en-us/aspnet/visual-studio/overview/2013/creating-web-projects-in-visual-studio> \_

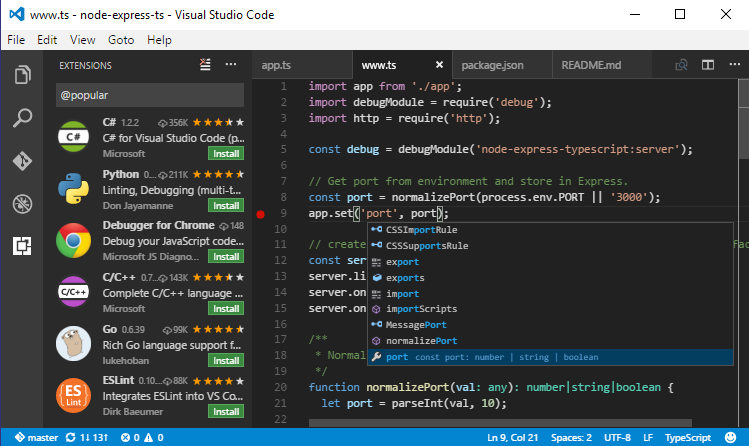
## Online Tutorials

In addition to your book you will want to get familiar with the various online tutorials such as the one on the W3Schools website: <https://www.w3schools.com/asp/default.asp>.

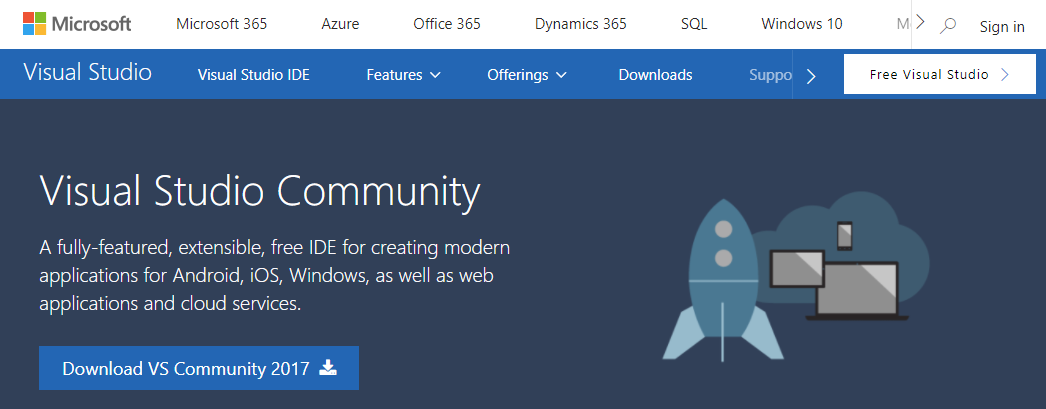


## Additional Editing Tools

Several companies, including Microsoft, have created additional editing tools for creating ASP.NET applications. Some tutorials that you find on the internet will assume that you know of them or have them installed. For example, the tutorial on the W3Schools site uses Microsoft’s Web Matrix (a dead product) or Visual Studio Code (The replacement to WebMatrix).



Of course, you can just use Visual Studio Community Edition and that is what we will do in this class.

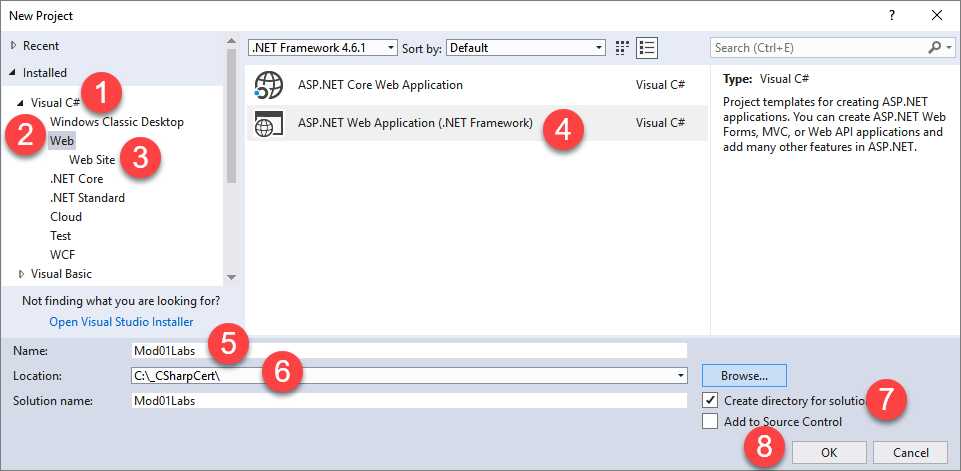


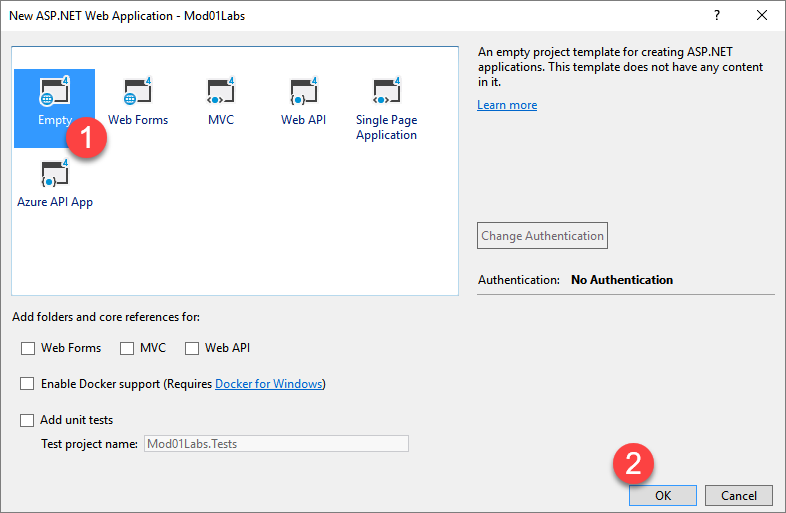
**LAB 01: Creating an ASP.NET project**

In this lab you will create an HTML web page in a new Visual Studio solution with an ASP.NET Empty Web **Application** project in it.

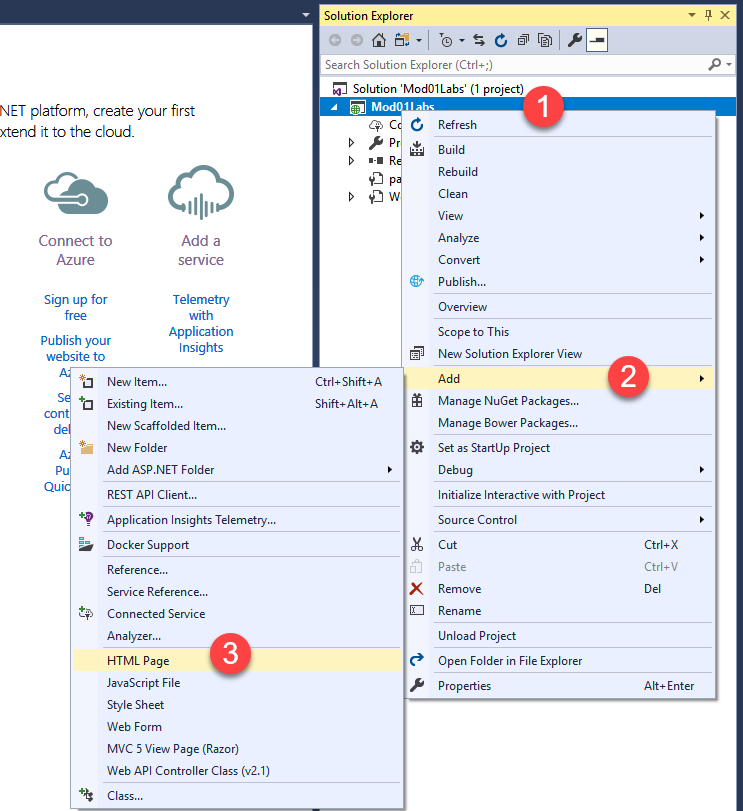
1) Create a folder and subfolder on your C: Drive called “C:\\_CSharpCert”

2) Create a Visual Studio solution and project as shown in the following pictures.

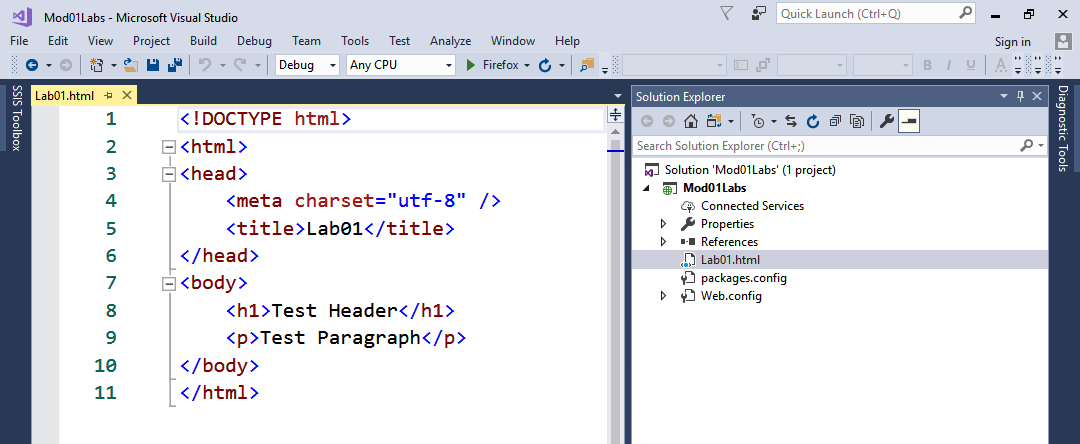




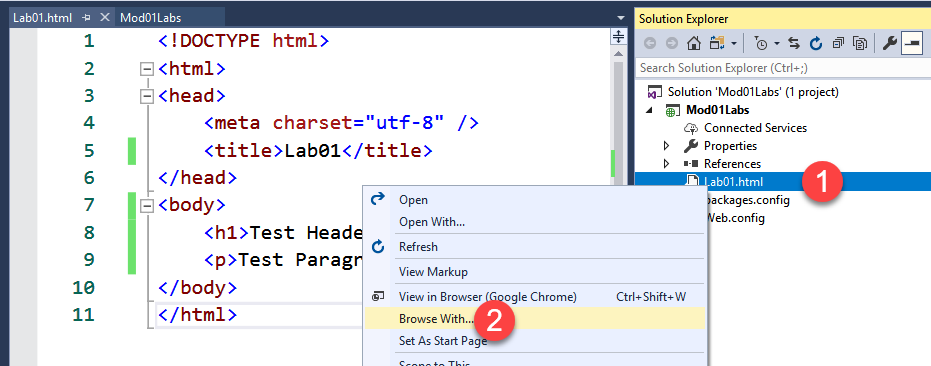
3) Add a new HTML page called Lab01 in the project using the code in Demo01.

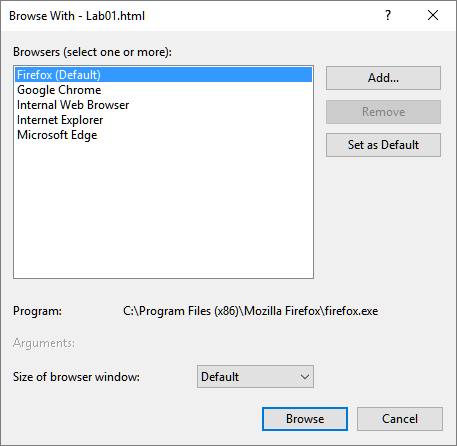


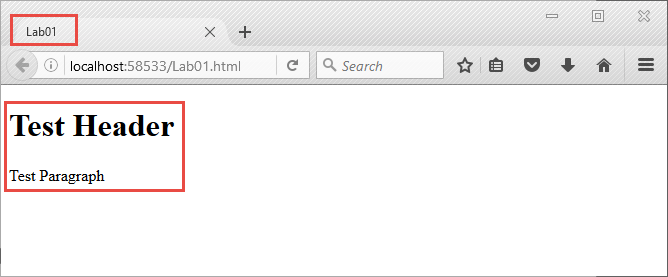
Add text and code as shown here:



4) Browse the page and verify that it displays the text as show here:







This lab should take about 5 minutes

## HTML Forms

HTML 2.0 introduced HTML forms. These forms have input tags that allow for interactive web pages.

**Demo02.html**: A simple HTML Form

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Demo02</title>

</head>

<body>

<h1>This is a Test Form</h1>

<p>The input tag provides several controls allowing a page to be interactive</p>

<form>

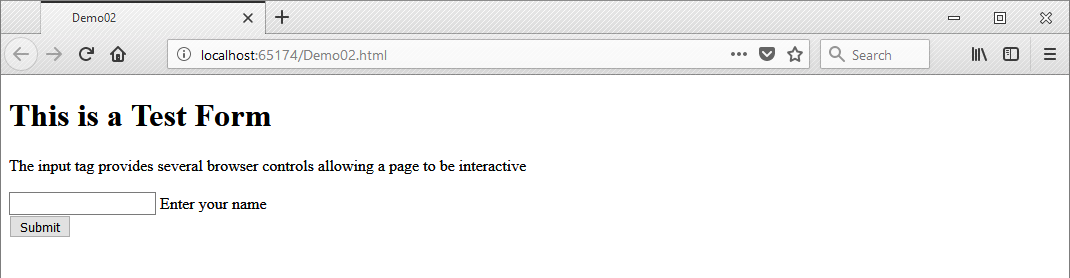
<input type="text" /> Enter your name<br />

<input type="submit" value="Submit" />

</form>

</body>

</html>



## JavaScript

JavaScript was also added to allow for the manipulation of user input.

**Demo03.html**: Processing data with JavaScript

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Demo03</title>

**<script type="text/javascript">**

**function readText(objForm) {**

**strTestVar = objForm.txtUserName.value;**

**alert("You typed: " + strTestVar);**

**}**

**</script>**

</head>

<body>

<h1>This is a Test Form</h1>

<p>The input tag provides several controls allowing a page to be interactive</p>

<form>

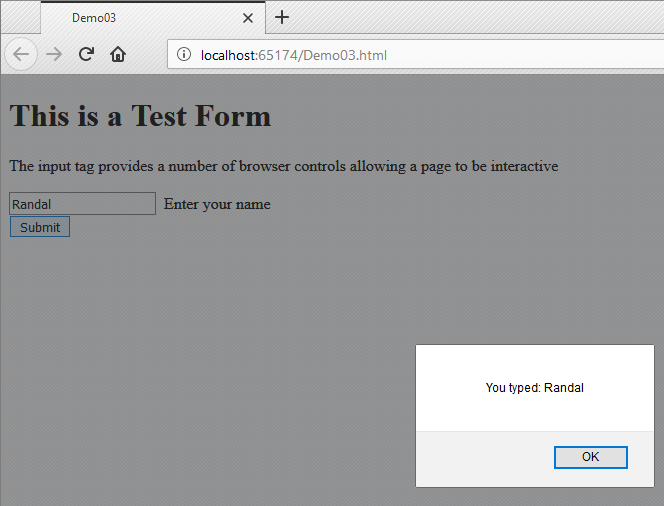
<input type="**text**" **name**="txtUserName" />&nbsp; Enter your name<br />

<input type="**button**" value="Submit" onclick="**readText**(this.form)" />

</form>

</body>

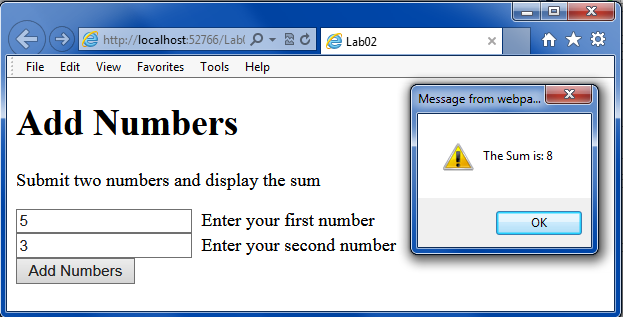
</html>



“The latest version of the HTML language, HTML5, introduced a few new form controls for the first time in the history of the language. For the most part, ASP.NET doesn’t use these, because they aren’t supported in all browsers (and even the browsers that support them aren’t always consistent). However, ASP.NET will use optional HTML5 frills, such as validation attributes (see Chapter 9), when they’re appropriate. That’s because browsers that don't support these features can ignore them, and the page will still work.” (MacDonald 2012, p. 6)

**LAB 02**

In this lab you will create an HTML web page that allows a user to add two numbers together using JavaScript.



Hints:

* Look up the JavaScript functions parseInt() and toString()

This lab should take about 10 minutes

## ASP.NET

“Early web development platforms had two key problems. First, they didn’t always scale well… Second, they provided little more than a bare-bones programming environment…

### Server-Side and Client-Side Programming

ASP.NET is designed first and foremost as a server-side programming platform. That means that all ASP.NET code runs on the web server. When the ASP.NET code finishes running, the web server sends the user the final result—an ordinary HTML page that can be viewed in any browser…

ASP.NET uses server-side programming to avoid several problems:

Isolation: Client-side code can’t access server-side resources.

Security: End users can view client-side code.

Thin clients: In today’s world, web-enabled devices such as tablets and smartphones are everywhere…

In recent years, there’s been a renaissance in client programming, particularly with JavaScript...

Fortunately, ASP.NET takes advantage of this change in two ways:

JavaScript frills: In some cases, ASP.NET allows you to combine the best of client-side programming with server-side programming. For example, the best ASP.NET controls can “intelligently” detect the features of the client browser. If the browser supports JavaScript, these controls will return a web page that incorporates JavaScript for a richer, more responsive user interface. You’ll see a good example of this technique with validation in Chapter 9.

ASP.NET’s Ajax features: Ajax is a set of JavaScript techniques used to create fast, responsive pages with dynamic content. In Chapter 25, you’ll learn how ASP.NET…

TIP: it’s worth noting that ASP.NET is not the best platform for writing complex, app-like client-side programs—at least not on its own. For example, ASP.NET isn’t much help to developers who want to build a real-time browser-based game or the next Google Maps. If this is what you want, it’s largely up to you to add the huge amounts of complex JavaScript that you need to your ASP.NET web forms. However, if you’d prefer to create an e-commerce hub or a business site, or a site that displays and manages large amounts of data, ASP.NET is the perfect fit.” (MacDonald 2012, p.6-8)

### Creating ASP.NET Scripts

ASP.NET code uses a Script tag just like JavaScript, but it must include:

* The runat=”server” attribute
* A directive stating the page is using C#
* An asp: namespace for an ASP.NET tag; this is not standard HTML

**Demo04.aspx**: Using ASP.NET scripts with the “Single Page” option

**<%@ Page Language="C#" %>**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Demo04</title>

**<script runat="server">**

**void Page\_Load()**

**{**

**lblServerTime.Text = DateTime.Now.ToString();**

**}**

**</script>**

</head>

<body>

<form id="form1">

<div>

The current time is:

**<asp:Label ID="lblServerTime" runat="server" />**

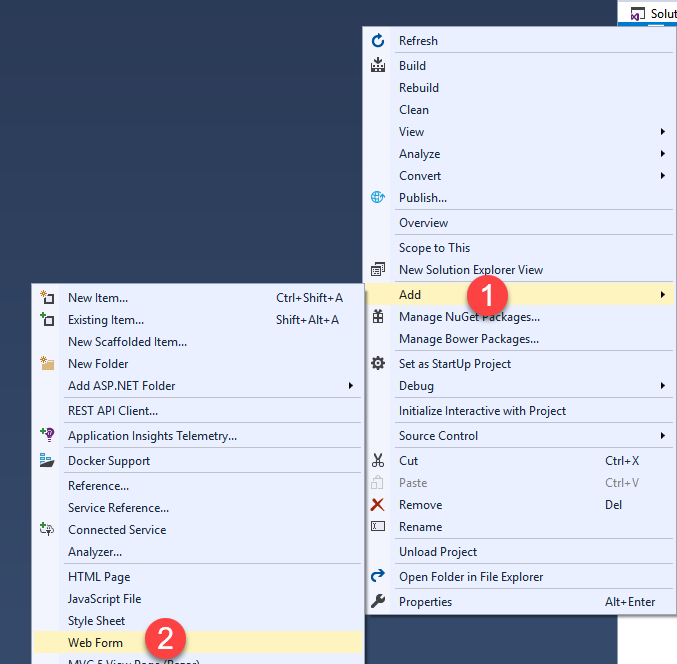
</div>

</form>

</body>

</html>

ASP.Net also has an option to use a separate code page to help you divide the Server-side processing code and your Client-side processing code, by creating a "**Web Form**."



**Demo05.aspx**: Using ASP.NET scripts with the “CodeBehind page” option

**<%@ Page**

**Language="C#"**

**AutoEventWireup="true"**

**CodeBehind="Demo05.aspx.cs"**

**Inherits**="**Module01.Demo05**" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>Demo05</title>

</head>

<body>

<form id="form1" runat="server">

<div>

The current time is: <asp:Label ID="lblServerTime" runat="server" />

</div>

</form>

</body>

</html>

**Demo05.aspx.cs:** The CodeBehind page

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Module01

{

public partial **class Demo05 : System.Web.UI.Page**

{

protected void **Page\_Load**(object sender, EventArgs e)

{

lblServerTime.Text = DateTime.Now.ToString();

}

}

}

**Demo05.aspx.designer.cs**

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace **Module01**{

public partial class Demo05 {

/// <summary>

/// form1 control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.HtmlControls.HtmlForm **form1**;

/// <summary>

/// lblServerTime control.

/// </summary>

/// <remarks>

/// Auto-generated field.

/// To modify move field declaration from designer file to code-behind file.

/// </remarks>

protected global::System.Web.UI.WebControls.Label **lblServerTime**;

}

}

**Note**: Web Forms are considered "Old Tech" now and are out of fashion. While you may still see them in real-life, they should not be your "go-to" option for new pages!

### Mixing Client and Server side code

Of course you can have both client and server side code in your web pages

**Demo06.aspx**: Single page with both client and server side code

<%@ Page Language="C#" %>

<!DOCTYPE html>

<html>

<head runat="server">

<meta charset="utf-8" />

<title>Demo06</title>

<script type="text/javascript">

function startTime() {

document.getElementById('spanNowClt').innerHTML = new Date();

}

</script>

<script runat="server">

void Page\_Load()

{

spanNowSvr.InnerText = DateTime.Now.ToLocalTime().ToLongTimeString();

}

</script>

</head>

<body onload="startTime()">

<form id="form1" runat="server">

<div>

<p>

Client Code: <span id="spanNowClt" />

<br />

</p>

<p>

Server Code: <span id="spanNowSvr" runat="server" />

<br />

</p>

</div>

</form>

</body>

</html>

### Create a Website with Separate HTML and ASP.NET pages

Traditionally, developers would create sites that would include both HTML and Server side processing. Pages using this technology include PHP, CGI, and of course ASP/ASP.NET.

Demo07a.html: A Typical HTML Client Side page

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Demo07a</title>

<script type="text/javascript">

function SendForm1Data()

{ document.forms["Form1"].submit(); }

function SendForm2Data()

{ document.forms["Form2"].submit(); }

</script>

</head>

<body>

<div>

<p>Send Data with Get:</p>

<!—NOTE: When using multiple forms, each must have a unique name! -->

<form name="**Form1**" method="**get**" action="Demo07**b**.aspx">

<input id="Text1" type="text" value="Get data" name="DataFromText1" />

<input id="Submit1" type="**submit**" value="submit" />

<input id="Button1" type="**button**" value="button" **onclick**="SendForm1Data()" />

</form>

</div>

<hr />

<div>

<p>Send Data with Post:</p>

<form name="**Form2**" method="**post**" action="Demo07**c**.aspx">

<input id="Text2" type="text" value="Post data" name="DataFromText2" />

<input id="Submit2" type="**submit**" value="submit" />

<input id="Button2" type="**button**" value="button" **onclick**="SendForm2Data()" />

</form>

</div>

</body>

</html>

You can process data sent from a web page in two different ways. The first is using the “Get” action. The second is using the “Post Action.”

**Demo07b.aspx**: Using GET

<%@ Page Language="C#" %>

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Demo07b</title>

<script runat="server">

void Page\_Load(object sender, System.EventArgs e)

{

string strData = Request.QueryString["DataFromText1"];

Response.Write("<h1><span style=\"color:Blue\">" + strData + "</span></h1>");

}

</script>

</head>

<body>

This page will not display anything until another page sends <i>this page</i> some data using a Query string!

</body>

</html>

Now here is a Post example…

**Demo07c.aspx**: Using POST

<%@ Page Language="C#" %>

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Demo07c</title>

<script runat="server">

void Page\_Load(object sender, System.EventArgs e)

{

string strData = Request.Form["DataFromText2"];

Response.Write("<h1><span style=\"color:Red\">" + strData + "</span></h1>");

}

</script>

</head>

<body>

This page will not display anything until another page sends <i>this page</i> some data using an HTTP Post!

</body>

</html>

### The ASP.NET Web Form Model

Cross-Page Posting (posting data across separate pages) is the currently the favored (and older) development model. However, when ASP.NET was released, it favored a Single-Page Posting model, show here:

**Demo08.aspx**: Using the Single-Page Posting model

<%@ Page Language="C#" %>

<!DOCTYPE html>

<html>

<head id="Head1" runat="server">

<meta charset="utf-8" />

<title>Demo08</title>

<script runat="server">

protected void ButtonLogin\_Click(object sender, EventArgs e)

{

//Olde, ASP, way of getting form data...

string strName = Request.Form["TextBoxName"];

string strPassword = Request.Form["TextBoxPassword"];

Response.Write("<span style=\"color:Red\">Name: " + strName + "</span><hr/>");

Response.Write("<span style=\"color:Red\">Password: " + strPassword + "</span>");

//The modern ASP.NET way of getting form data ...

strName = TextBoxName.Text;

//Response.Write(strName);

LabelInfo.Text = "Name: " + strName + "<hr/> Password: " + TextBoxPassword.Text;

LabelInfo.ForeColor = System.Drawing.Color.Green;

}

</script>

<style type="text/css">

.style1 {

width: 27%;

}

.style2 {

width: 76px;

}

</style>

</head>

<body>

<form id="form1" runat="server">

<div>

<br />

Please Login in to accessed your information.

<br />

<br />

<table class="style1">

<tr>

<td class="style2">

<asp:Label ID="LabelName" runat="server" Text="Name"></asp:Label>

</td>

<td>&nbsp;

<asp:TextBox ID="TextBoxName" runat="server" Width="150px"></asp:TextBox>

</td>

</tr>

<tr>

<td class="style2">

<asp:Label ID="LabelPassword" runat="server" Text="Password"></asp:Label>

</td>

<td>&nbsp;

<asp:TextBox ID="TextBoxPassword" runat="server" Width="150px"></asp:TextBox>

</td>

</tr>

</table>

<br />

<asp:Button ID="ButtonLogin" runat="server" OnClick="ButtonLogin\_Click"

Text="Login" Width="253px" />

<br />

<br />

&nbsp;<asp:Label ID="LabelInfo" runat="server" Text="(Information)"></asp:Label>

</div>

</form>

</body>

</html>

### ASP.NET and Object Oriented Programming (OOP)

ASP.NET is designed to work with OOP methodologies. This includes the creation of customer classes for your web site. Here is an example of one such application from your book.

**“Building a Basic Class**

In the next example, you’ll see how to construct a .NET class piece by piece. This class will represent a product from the catalog of an e-commerce company. The Product class will store product data, and it will include the built-in functionality needed to generate a block of HTML that displays the product on a web page. When this class is complete, you’ll be able to put it to work with a sample ASP.NET test page.” (MacDonald 2012, p.49)

**Default.aspx**

<%@ Page Language="C#" %>

<!DOCTYPE html>

***<!-- Note: The book's author places server script tags outside of the HTML header -->***

<script runat="server">

private void Page\_Load(object sender, EventArgs e)

{

Product saleProduct = new Product("Kitchen Garbage", 49.99M, "garbage.jpg");

Response.Write(saleProduct.GetHtml());

}

</script>

<html>

<head>

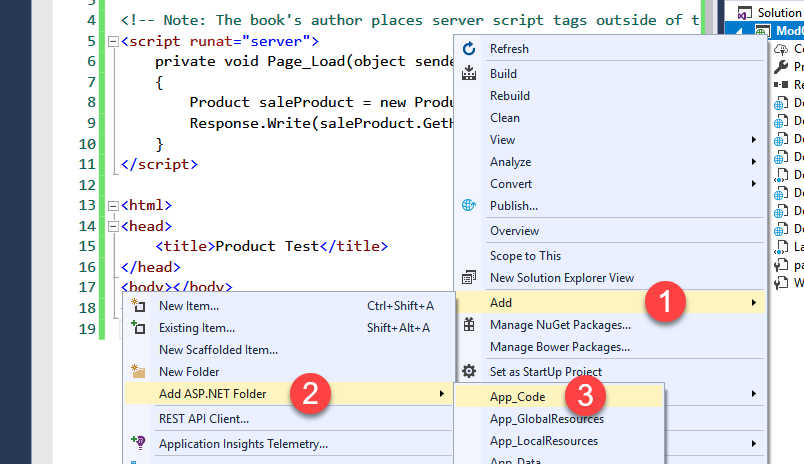
<title>Product Test</title>

</head>

<body></body>

</html>

**Important**: The following Products class file should be placed in a new **App\_Code folder** and depending on if you use a Code-Behind or Single Page option you must set the file’s Build Action property. For the Single file web pages set the Build Action to “Content”, but for web pages using the Code-Behind option you use the “Compile.”  Also, after you change this you may have to close and reopen Visual Studio before intellisense will work!



**Product.cs**

using System;

// Define the delegate that represents the event.

public delegate void PriceChangedEventHandler();

public class Product

{

public string Name {get; set;}

// Define the event.

public event PriceChangedEventHandler PriceChanged;

private decimal price;

public decimal Price

{

get

{

return price;

}

set

{

price = value;

// Fire the event, provided there is at least one listener.

if (PriceChanged != null) { PriceChanged(); }

}

}

public string ImageUrl {get; set;}

public string GetHtml()

{

string htmlString;

htmlString = "<h1>" + Name + "</h1><br>";

htmlString += "<h3>Costs: " + Price.ToString() + "</h3><br>";

htmlString += "<img src='" + ImageUrl + "' />";

return htmlString;

}

public Product(string name, decimal price)

{

Name = name;

Price = price;

}

public Product(string name, decimal price, string imageUrl)

{

Name = name;

Price = price;

ImageUrl = imageUrl;

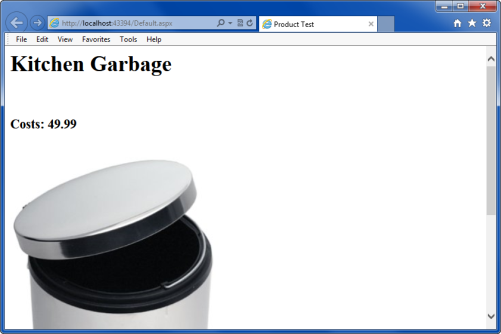
}

}

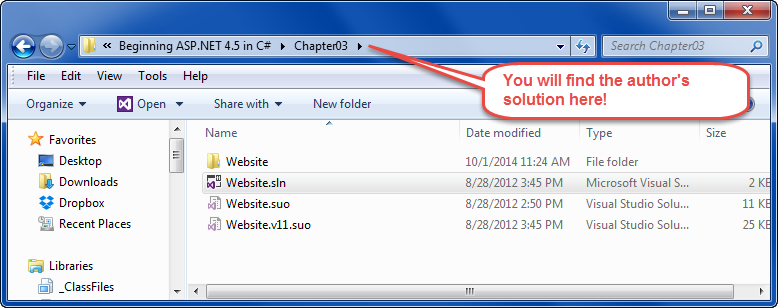
**LAB 03**

Create an ASP.NET web page that allows a user to view data from a C# server-side object.

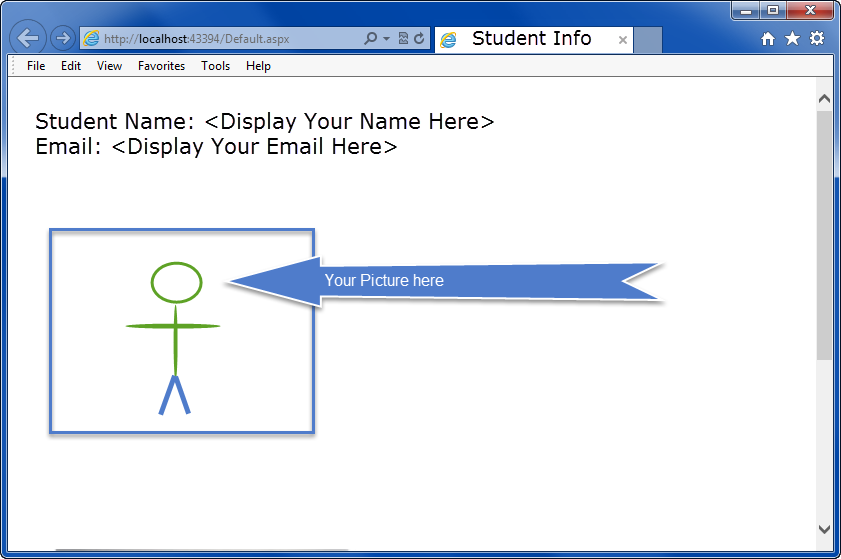
For this lab we will use the similar to the author’s example on Page 49 of your book. This example code web page can be found in a completed form in the author’s downloadable book files (<http://www.apress.com/9781430242512>)



1) Review the Author’s example



2) Create your own example by creating a Student Class and a “Student Info” web page.



This lab should take about 30 minutes